"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

PROYECTO INTEGRADOR DE CARRERA

ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

DISEÑO Y PROTOTIPO DE SISTEMA INTELIGENTE PARA SEMÁFOROS DEL DISTRITO METROPOLITANO DE QUITO PARA DAR PRIORIDAD A PERSONAS CON DISCAPACIDAD.

FREDDY BOLÍVAR ALMACHI TOAQUIZA

ING. DAVID CANDO, MG.

AÑO 2017

"Responsabilidad con pensamiento positivo"

Datos generales:

TEMA:	Diseño y prototipo de sistema inteligente para semáforos del Distrito Metropolitano de Quito para dar prioridad a personas con Discapacidad.
ESTUDIANTE:	Freddy Bolívar Almachi Toaquiza
CARRERA:	Ingeniería en Electrónica Digital y Telecomunicaciones
TUTOR:	Ing. David Cando
ASESOR TÉCNICO:	Ing. David Cando
FECHA:	06 de Febrero del 2017

RESUMEN

En la actualidad el sistema de semaforización del Distrito Metropolitano de Quito busca agilitar los procesos de movilidad en la urbe para peatones y medios de transporte, dentro de este proceso no se ha considerado el tema de discapacidad, ya que excluyen a un grupo vulnerable de la sociedad que busca su interacción con el medio que lo rodea.

El sistema de semaforización actual cuenta con pulsadores dedicados a dar prioridad a los peatones en ciertos lugares de la urbe, el inconveniente con esta técnica es que no cumple con el objetivo para el cual fueron implementados.

En la Provincia de Pichincha – Cantón Quito existen 55.932 personas con algún tipo de discapacidad que no cuentan con las facilidades para movilizarse en las intersecciones con un elevado flujo vehicular, para solventar este inconveniente es necesario proponer un diseño de un prototipo de sistema inteligente que interactúe de forma eficiente y confiable con el sistema de semaforización actual.

El sistema inteligente estará constituido por una plataforma microcontroladora enlazada a sensores RFID (Radio Frequency Identification), los que identificarán mediante ondas de radio la señal de un dispositivo vinculado al sistema de control. La información obtenida se procesará de forma sistemática lo que garantizará el desempeño y la convivencia de los dos sistemas al momento de interactuar en circunstancias reales.

PALABRAS CLAVES

Semaforización
Discapacidad
Movilidad
Plataforma microcontroladora
RFID

ABSTRACT

At present the system of traffic of the Metropolitan District of Quito seeks to accelerate the processes of mobility in the city for pedestrians and means of transport, within this process the issue of disability has not been considered, excluding a vulnerable group of the society that Look for your interaction with the environment that surrounds it.

The real traffic signal system has to push-buttons dedicated to give priority to pedestrians in certain places of the city, the drawback with this technique is that it does not meet the objective for which it was implemented.

In the Province of Pichincha - Canton Quito there are 55,932 people with some type of disability that do not have the facilities to mobilize in the intersections with a high vehicular flow, to solve this problem it is necessary to propose a prototype of intelligent system that interact efficiently and reliably with the real traffic signal system.

The intelligent system consists of a microcontroller platform linked to RFID (radio frequency identification) sensors, which identify by radio waves the signal of a device linked to the control system. The information obtained is processed in a systematic way to guarantee the performance and coexistence of the systems in the moment of interacting in real circumstances.

KEYWORDS

Traffic lights
Disability
Mobility
Microcontroller platform
RFID

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTO

Quiero agradecer a Dios por brindarme la sabiduría y la fuerza de hacer las cosas correctamente cada día y no decaer con el transcurso del tiempo.

A mi amada esposa e hija por ser la luz en el arduo camino, a mi familia por su incondicional apoyo.

A mis profesores que con su exigencia y sabiduría formaron valores que van más allá de las aulas. Gracias

Freddy

SECCIÓN I

1.1. Problema de Investigación

En el Distrito Metropolitano de Quito se busca a diario agilitar los procesos de movilidad para el sistema de transporte público y privado, con la implementación de semáforos en las intersecciones de mayor flujo vehicular que garantizan la integridad de los conductores y peatones, sin embargo estas mejoras implementadas no facilitan la interacción de las personas con discapacidad.

Actualmente en la provincia de Pichincha – Cantón Quito existen 2.239.191 personas de las cuales 55.932 (2,5%) poseen algún tipo de discapacidad, esta minoría busca desenvolverse de forma independiente e interactuar con el entorno que los rodea sin ocasionar o sufrir algún tipo de accidente.

El desenvolvimiento de este grupo vulnerable de la sociedad se ve afectado por el sistema de semaforización existen en el Distrito Metropolitano de Quito, debido a que no posee algún tipo de sistema que permita dar prioridad a las persona con discapacidad para su integridad de una forma segura y confiable.

1.2. Objetivo General.

Diseñar e implementar un prototipo de sistema inteligente de cruce peatonal en intersecciones críticas de la Urbe que interactúe con la infraestructura actual de semáforos del Distrito Metropolitano de Quito, para dar prioridad a personas con algún tipo de discapacidad.

1.3. Objetivos Específicos.

- Analizar la factibilidad de diseñar un sistema inteligente de cruce peatonal en intersecciones críticas, enfocado a personas con algún tipo de discapacidad del Distrito Metropolitano de Quito.

- Fundamentar teóricamente el funcionamiento de un sistema inteligente autónomo a través de una plataforma microcontroladora basada en ARDUINO con interacción de sensores inalámbricos.
- Diseñar un prototipo de sistema inteligente de cruce peatonal, basado en un microcontrolador ARDUINO MEGA con interacción de sensores RFID.
- Implementar un prototipo de sistema inteligente en conjunto con un sistema de semáforos en una intersección de dos vías principales en una maqueta demostrativa.
- Pruebas de funcionamiento del prototipo de sistema inteligente en conjunto con un sistema de semáforos de una intersección de dos vías en doble sentido.

1.4. Introducción

Debido al crecimiento de las urbes y sus problemas de movilidad tanto para transporte público y privado, los semáforos han evolucionado de forma constante con el fin de garantizar su seguridad y resguardar su integridad en todo momento.

El primer semáforo fue creado en 1868 por el Ingeniero británico J.P. Knight, este fue implementado en la calle 10 de Diciembre en el exterior del Parlamento de la Ciudad de Londres, se caracterizó por funcionar de forma manual y por poseer dos luces, una roja y otra verde.

Después de 46 años se instaló el primer semáforo eléctrico con luces incandescentes en Estados Unidos, en 1933 Dinamarca implementa la luz peatonal, 20 años después Nueva York incorpora mensajes de voz para advertir peligro. Para 1961, las palabras fueron remplazadas por imágenes denominadas muñecos peatonales. En la actualidad el funcionamiento de los semáforos tiende a ser implementado a través de microcontroladores para dar mayor fiabilidad a este sistema.

Los microcontroladores son plataformas electrónicas en la cual convergen varios circuitos integrados que se interconectan a través de un programa que se desarrolla en un lenguaje de programación que varía de acuerdo al fabricante.

La interacción del microcontrolador con su entorno exterior está regida por sus entradas las cuales pueden ser de forma digital o analógica, estas tienen la capacidad de interactuar con distintos tipos de sensores y tarjetas estandarizadas (shields) con el fin de desarrollar aplicaciones de mayor complejidad.

Los sensores que trabajan en conjunto con los microcontroladores están relacionados con la frecuencia, niveles de ruido, temperatura, proximidad, movimiento etc. Dentro de los sensores de frecuencia se encuentran los RFID (Radio Frequency Identification), estos sistemas permiten almacenar y recuperar información de forma remota con la utilización de lectores y etiquetas. Las etiquetas o tags pueden ser pasivas o activas, esto depende de la aplicación que se desarrolla.

1.5. Hipótesis

El sistema de semaforización del Distrito Metropolitano de Quito, se ha innovado con la implementación de semáforos peatonales, con el fin de resguardar la integridad del peatón sin interrumpir el flujo vehicular. Dentro de la innovación del sistema de semaforización no fue considerado un grupo vulnerable de la sociedad, que son las personas con discapacidad; ya que el tiempo de cruce peatonal es insuficiente.

Para facilitar la interacción de las personas con discapacidad se ha visto la necesidad de proponer un sistema inteligente que interactúe con los semáforos vehiculares y peatonales. El siguiente sistema inteligente contará con una plataforma microcontroladora basada en ARDUINO MEGA en conjunto con sensores RFID (Radio Frequency Identification), integrados con un lenguaje de programación que permitirá la correcta interacción con el circuito de control de los semáforos.

El microcontrolador procesará la información proveniente de los sensores, los cuales interactuarán a través de un protocolo de comunicación PSI (Serial Peripheral Interface), con el fin de incrementar el tiempo de cruce peatonal y así resguardar la integridad de la persona con discapacidad sin alterar el funcionamiento del sistema de semaforización.

SECCIÓN II

2.1. Marco Teórico

2.1.1. Semáforo Generalidades

La palabra semáforo es de origen griego (sema), que significa señal, y (foros), que significa portar; en la antigüedad el semáforo era una torre de seis metros que se utilizó en comunicaciones de emergencia a través de fogatas o banderas.

En 1868 en el mes de diciembre fue instalado el primer semáforo en Londres diseñado por John Peake Knight de profesión Ingeniero Ferroviario, el diseño de este semáforo se caracterizó por tener dos lámparas a gas de color rojo y verde, dos brazos mecánicos para indicar el sentido y el momento de proseguir de los vehículos, este era operado de forma manual. Las luces verdes y rojas se heredaron de la marina las cuales ya eran utilizadas para evitar colisiones entre los barcos.

Earnest Sirrine adapto el diseño de Knight a un sistema eléctrico en 1910, este modelo se instaló en Cleveland en 1914, para evitar accidentes de tránsito; en 1920 William Potts (Oficial de Policía de Detroit) agrego la luz amarilla con el fin de advertir el cambio de luces. La producción de los semáforos de tres luces se desarrolló a finales de los años 40.

En la actualidad es un sistema electrónico que sirve para controlar el flujo vehicular y peatonal en intersecciones mediante la visualización de tres tonos de luces según como se muestra en la Figura 1, los cuales son:

- **VERDE:** Permite la libre circulación vehicular en el sentido del flujo de tránsito y también admite girar a la derecha o izquierda según sea el caso.
- **AMARILLO:** Alerta a los vehículos y peatones que el intervalo de tiempo del color rojo está por comenzar. Por lo general esta interrupción es de 3 a 6 segundos.
- **ROJO:** El flujo vehicular debe detenerse de forma obligatoria antes del cruce peatonal hasta que transcurra el intervalo de tiempo predeterminado.

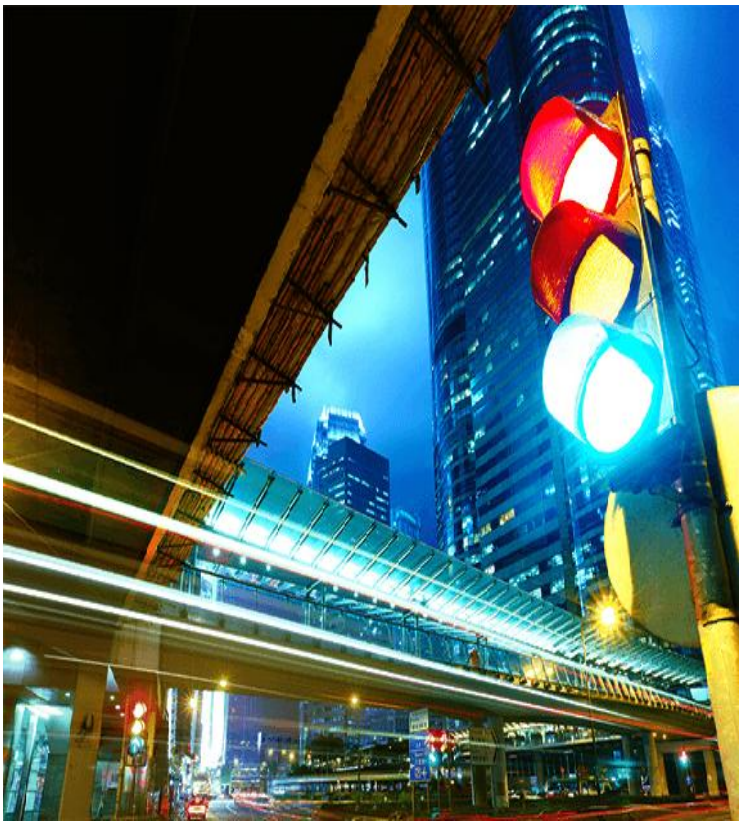


Figura 1. Semáforo convencional

Fuente: AO Kasperky Lab.
https://blog.kaspersky.es/los-semaforos-son-faciles-hackear/4170/

2.1.1.1. Clasificación de los Semáforos

Los semáforos se clasifican de la siguiente manera:

- Vehicular: Controla el flujo vehicular en intersecciones de Urbes.
- Semáforos direccionales: Advierte el sentido de giro del objeto a direccionar.
- Peatonal: Enfoca su funcionamiento a intervenir el flujo peatonal en los cruces de vía.
- Bicicletas: Controla el flujo de bicicletas en las urbes que cuentan con vías apropiadas para este medio de transporte.
- Semáforos intermitentes o de destello: Su diseño permite advertir la presencia de peligro en el sector.

2.1.1.2. Sincronización de luces de los semáforos

La sincronización de tiempos para el intercambio de la activación de cada una de las luces, está regida por la siguiente formula en función de la Ec. 1.

Intervalo de cambio de fase = Amarillo + Rojo

$$y = \left(t + \frac{v}{2a}\right) + \left(\frac{W+L}{v}\right) \qquad \text{Ec. 1}$$

Fuente: Calculo de los tiempos de semaforización
Fernández Y. y Rodríguez G., (2013)

Donde:

y = intervalo de cambio de fase, ámbar mas todo rojo (s)

t = tiempo de percepción-reacción del conductor (usualmente 1.00s)

v = velocidad de aproximación de los vehículos (m/s)

a = tasa de deceleración (valor usual 3.05 m/s2)

W = ancho de la intersección (m)

L = longitud del vehículo (valor sugerido 6.10 m)

A demás es necesario considerar los siguientes factores:

1. Número de carriles, condiciones físicas y geométricas.
2. Fluctuación vehicular en cada dirección.
3. Flujo de transporte público y privado.
4. Flujo peatonal.

2.1.2. Microcontrolador

La Figura 2 muestra que es un sistema electrónico, inmerso en un circuito integrado en el cual convergen los elementos de un procesador programable, que se caracteriza por poseer un sistema físico que se ajusta a las necesidades de su entorno. Su tamaño reducido y su capacidad de configuración hicieron que su utilización tenga un crecimiento vertiginoso, lo que amplía su campo de aplicación.

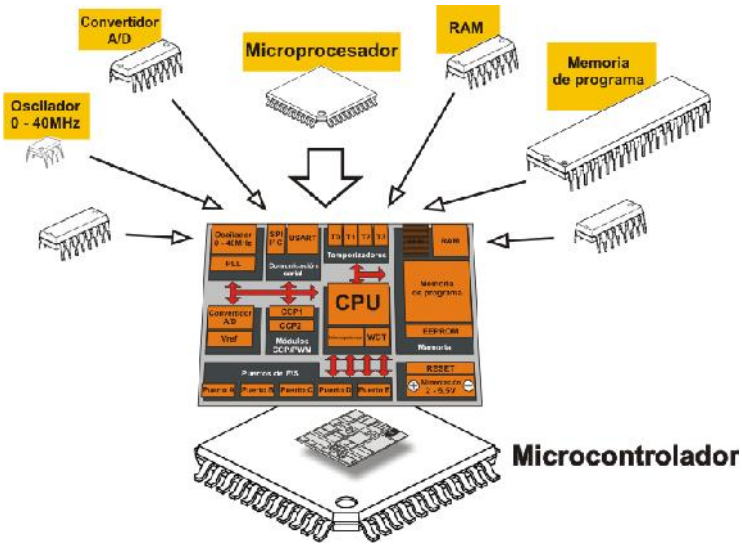


Figura 2. Estructura interna de un microcontrolador en un chip.

Fuente: Microcontroladores PIC.
Verle M., (2009).

Estos sistemas utilizan una tecnología compleja en la que interaccionan distintos conceptos correlacionados entre sí, que dificultan establecer normas que garanticen su fiabilidad, mantenibilidad y modificación de los sistemas constituidos por ellos para mejorar sus prestaciones en función de su diseño.

2.1.2.1. Componentes del microcontrolador

De acuerdo a Valdés Fernando. y Pallas Ramón (2007); un microcontrolador combina todos sus subsistemas disponibles para que interactúen entre sí, con el fin de sincronizar y procesar los datos recibidos en cada uno de los módulos internos según como se indica la Figura 3.

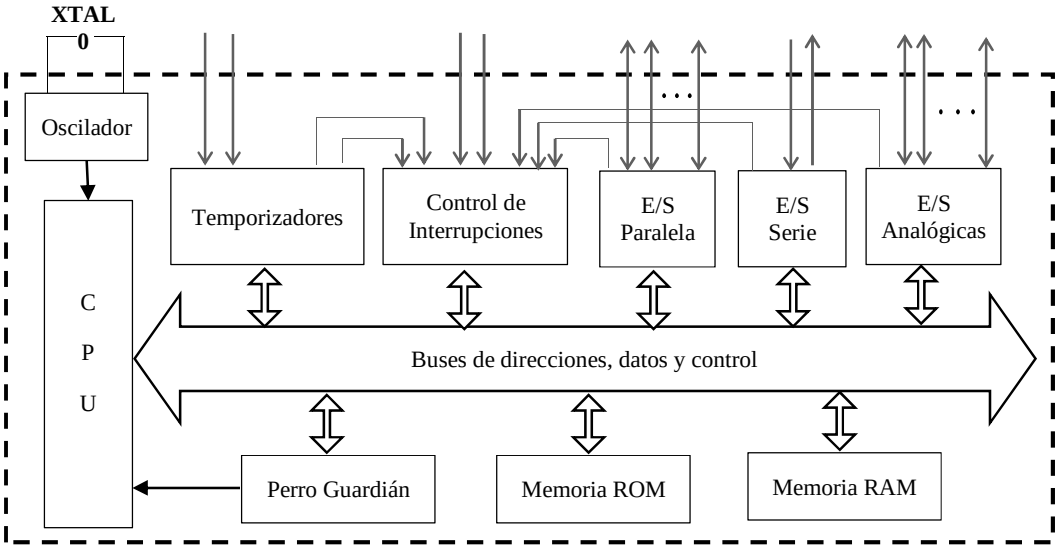


Figura 3. Diagrama de bloques general de un microcontrolador.

Fuente: Microcontroladores fundamentos y aplicaciones con PIC
Valdés, F., y Pallàs, R., (2007)

Oscilador: Genera pulsos a una frecuencia fija que sincroniza todas las operaciones internas, esta determina la velocidad de ejecución de las instrucciones y depende de los circuitos semiconductores según como se indica en la figura 4. Para estabilizar la frecuencia se utilice un cristal de cuarzo o resonador cerámico.

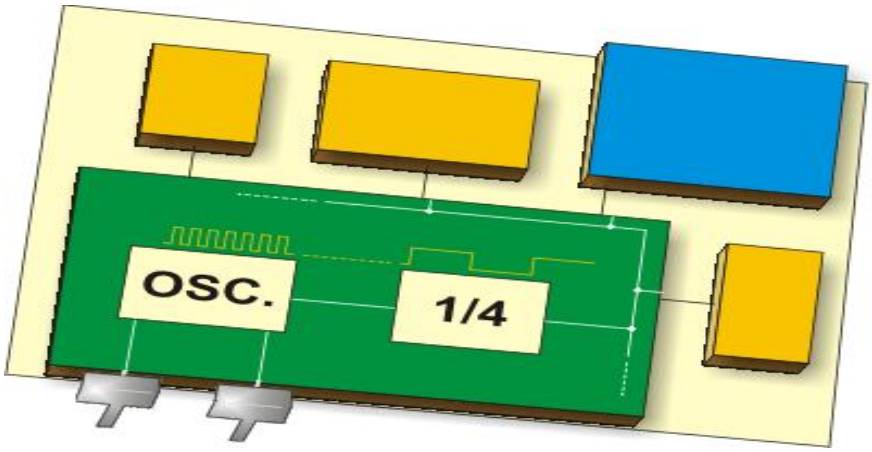


Figura 4. El oscilador permite que el funcionamiento sea armónico y síncrono.

Fuente: Microcontroladores PIC.
Verle M., (2009).

CPU (Unidad Central de Proceso), es el componente primordial del microcontrolador, esta extrae las instrucciones de forma ordenada desde la memoria donde reposan, las decodifica y ejecuta. Además en la CPU está inmersa la ALU (Unidad Aritmético Lógico), esta unidad es la encargada de las operaciones sobre los datos de los registros. Está constituida por entradas para dos operandos, salidas de resultados y líneas de control que permiten ejecutar una de las operaciones disponibles.

Los registros son espacios de memoria existentes dentro de la CPU así como se evidencia en la figura 5; sirven de almacenamiento temporal para los datos necesarios que se ejecutan en distintas instrucciones. Son de alta velocidad pero su capacidad de almacenamiento es baja.

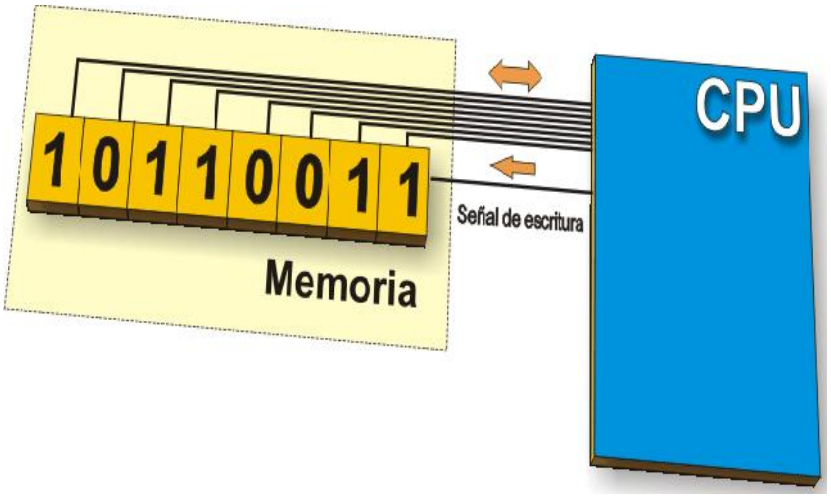


Figura 5. Estructura básica de un registro.

Fuente: Microcontroladores PIC.
Verle M., (2009).

Memoria ROM (Read Only Memory), es solo de lectura y no es volátil, es utilizada para almacenar permanentemente los programas.

Memoria RAM (Random Acces Memory) es de lectura y escritura, es volátil, es decir, la información almacenada es borrada cuando existe la ausencia de energía. En esta se almacenan los datos creados y utilizados por el programa.

Las entradas y salidas son importantes, pues a través de ellas el microcontrolador interacciona con su entorno exterior (periféricos). Están conformadas por puertos en paralelo o serie tal como se muestra en la figura 6. Los puertos paralelos pueden agruparse hasta 8 líneas de entradas y salidas digitales. Los puertos seriales pueden presentarse de varios tipos, según la norma de comunicación implementada (RS-232C, I2C, SPI, USB, etc.). Se consideran como entradas y salidas, los temporizadores y la gestión de las interrupciones.

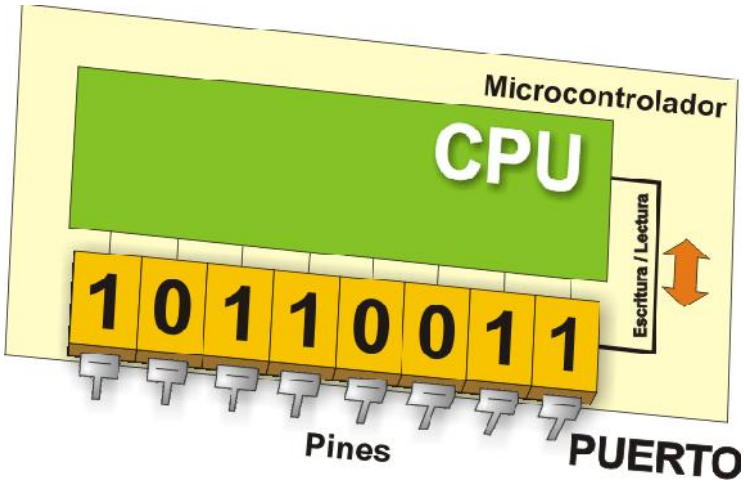


Figura 6. Los puertos de entrada y salida permiten la conexión hacia el exterior.

Fuente: Microcontroladores PIC.
Verle M., (2009).

Perro Guardián (WDT: Watchdog Timer), de acuerdo a la Figura 7, consta de un oscilador y un contador binario de N bits. El oscilador envía pulsos de forma periódica y permanente al contador, cuando alcanza el valor máximo se desbordará el registro lo que originará una señal de reseteo.

Un temporizador de vigilancia (WDT) obliga a un microcontrolador incorporado a restablecer sus funciones (restablecimiento de hardware) en respuesta a un estado de software no válido. Tales estados pueden ser tan simples como un bit de registro con conmutación causada por un error de software o un evento como EMI (interferencia electromagnética).

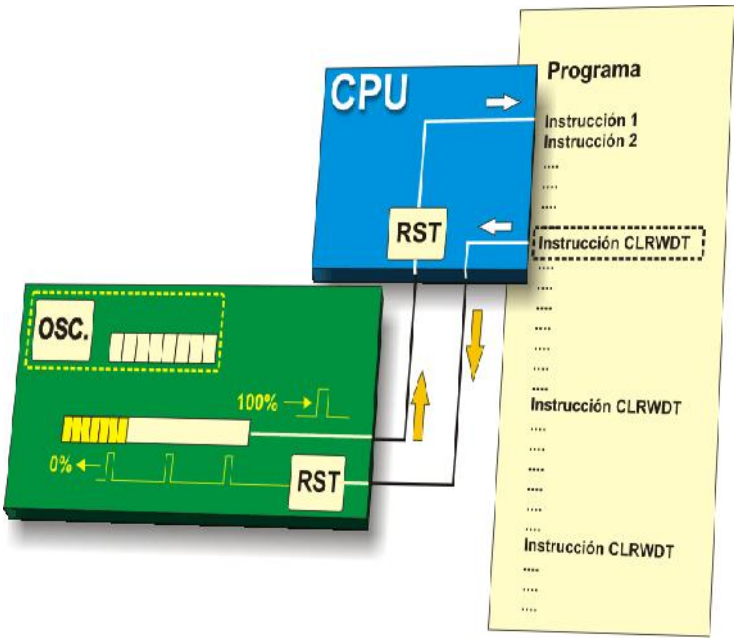


Figura 7. Diagrama básico de un WDT.

Fuente: Microcontroladores PIC.
Verle M., (2009).

Buses de direcciones, datos y control realizan la transferencia de información entre unidades. Según el procesador estos pueden ser de 8, 16, 32, 64 bits.

2.1.2.2. Protocolos de comunicación

De acuerdo a Verle Milán (2009), para transmitir información entre componentes electrónicos, se lo puede realizar de distintas formas. De forma serial, en este tipo de comunicación se transmite la información bit a bit por un único canal. También se lo puede realizar de forma paralela, en esta se envían varios bits simultáneamente en canales separados y de forma sincronizada.

Existe una gran variedad de protocolos y estándares de comunicación, reconocidos por la inmensa variedad de dispositivos electrónicos, debido a esto se detallarán los protocolos más importantes que ATmega utiliza para comunicarse con los distintos dispositivos electrónicos.

2.1.2.2.1. Protocolo I2C/TWI

I2C (Inter – Integrated Circuit) o TWI (Two – Wire), Utilizado en la transmisión de datos entre circuitos integrados, se caracteriza por utilizar dos vías de transmisión de información, la una es la línea SDA que sirve para transferir los datos y la línea SCL que está relacionada al envío de la señal de reloj (sincronización) como se evidencia en la figura 8.

Cada uno de los dispositivos conectados al bus I2C, posee una dirección exclusiva de 7 bits, los dispositivos pueden ser configurados como maestro o esclavo. Un dispositivo maestro es el que inicia la transmisión de datos y genera la señal de reloj. La propagación serial por lo general está en 100 kbps (Estándar) o 10 kbps (Baja). En la actualidad existen sistemas con velocidad de transmisión de 3.4 Mbps.

La transmisión es half dúplex (la comunicación solo se puede establecer en un sentido al mismo tiempo), es decir que al momento que un dispositivo empiece a recibir información, este tendrá que esperar a que el emisor termine de transmitir para responderle.

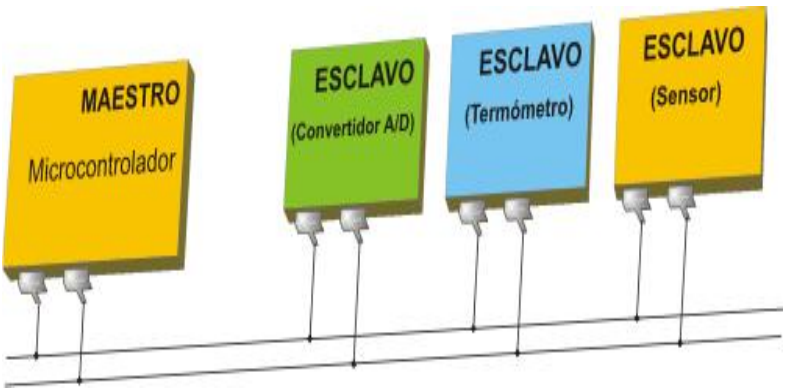


Figura 8. I2C comunicación serial entre microcontrolador y circuitos integrados.

Fuente: Microcontroladores PIC.
Verle M., (2009).

2.1.2.2.2. Protocolo SPI

SPI (Serial Peripherical Interface), este sistema de comunicación permite controlar una gran cantidad de dispositivos electrónicos digitales que permitan un flujo de bits serial de

forma sincronizada, al igual que I2C un dispositivo debe funcionar como maestro y los otros como esclavos y viceversa según como se muestra en la figura 9.

Este protocolo posee cuatro líneas; línea uno SCK, envía la señal de reloj a todos los dispositivos, que se genera en el dispositivo maestro, la segunda denominada SS, el dispositivo maestro la utiliza cuando desea comunicarse con un dispositivo esclavo; la tercera MOSI se utiliza para el envío de datos a un esclavo elegido desde el dispositivo principal y la cuarta MISO permite enviar la información en sentido contrario (respuesta). La transmisión es full dúplex, es decir la información puede enviarse en ambos sentidos. La transferencia de datos es superior a la de I2C.

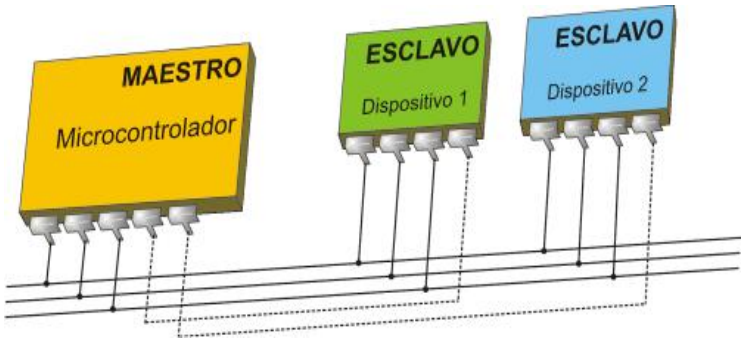


Figura 9. SPI comunicación serial a través de cuatro líneas.

Fuente: Microcontroladores PIC.
Verle M., (2009).

2.1.3. Arduino

De acuerdo a Torrente Oscar (2013); Arduino se cataloga como una serie de circuitos electrónicos de código abierto que se fundamenta en los microcontroladores ATMEL, estos implementan componentes que permiten su desarrollo en un entorno amigable para desarrollar objetos que interactúen con elementos del exterior.

Existen tarjetas estandarizadas (shields) que permiten expandir el hardware con el fin de desarrollar arquitecturas más robustas. El entorno de desarrollo se encuentra disponible para los sistemas operativos de Windows, MAC y LINUX.

La Tabla 1, muestra los modelos más reconocidos dentro de la amplia gama de Arduinos.

Tabla 1. Características de Arduinos más utilizados

ARDUINO	MICRO CONTROLADOR	MEMORIA FLASH	VELOCIDAD DE RELOJ	PINES I/O DITALES
UNO	ATmega 328	32 kB	16 MHz	14
LEONARDO	ATmega 32u4	32 kB	16 MHz	20
DUE	AT91SAM3X8E	512 kB	84 MHz	54
YUN	ATmega32U4	512 kB	84 MHz	54
ETHERNET	ATmega 328	32 kB	16 MHz	14
MEGA	ATmega 2560	256 kB	16 MHz	54
NANO	ATmega 168	16 kB	16 MHz	14
MICRO	ATmega 32u4	32 kB	16 MHz	20

Fuente: Elaborado por el autor

2.1.3.1.Arduino Mega

Arduino mega cuenta con las siguientes propiedades:

- **Programación.-** Permite su programación con el software de Arduino IDE, cuenta con una pre programación del cargador de arranque que accede a cargar un nuevo código sin utilizar un programador de hardware externo, su comunicación es a través de protocolo STK500, también se puede utilizar la programación serial ICSP de encabezado a través del Arduino ISP.
- **Alimentación.-** Se alimenta mediante una conexión USB o una fuente externa. La tarjeta admite un rango de alimentación de 6 a 20 V, si el voltaje es menor a 7 V la placa puede ser inestable, si el voltaje es superior a 12 V puede ocasionar daños a la placa, el rango optimo se enmarca entre los 7 y 12 V.
- **Memoria.-** El ATmega 2560 posee 256 kB de memoria flash donde se recopilar el código de programación, de los cuales 8 kB para el cargador de arranque, 8 kB destinados a la memoria SRAM y 4 kB de EEPROM.

- **Pines de Entrada y salida.-** Cuenta con 54 pines digitales con un voltaje de 5V, cada pin puede recibir o enviar 20 mA, tiene una resistencia pull – up (desconectada por defecto) de 20-50 kΩ. Un máximo de 40 mA es el valor que no debe superarse para evitar daños permanentes en el microcontrolador.
- Cuenta con 16 entradas analógicas, cada uno de los cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes).
- **Características físicas.-** Su longitud es de 101.52 mm, ancho 53.3 mm y su peso 37 g.

2.1.4. Sistemas RFID

De acuerdo a Finkenzeller Klaus (2010), RFID (Radio Frequency Identification) se trata de sistemas de identificación, la cual permite el almacenar y recuperar datos de forma remota, utiliza un dispositivo lector (Reader) y una antena con un transponder (Etiquetas / Tags), la comunicación se la realiza a través de ondas de radio.

2.1.4.1.Características

Este sistema está compuesto por un lector, un software (middleware) y las etiquetas o tag's. La Figura 10 muestra la interacción de sus componentes.

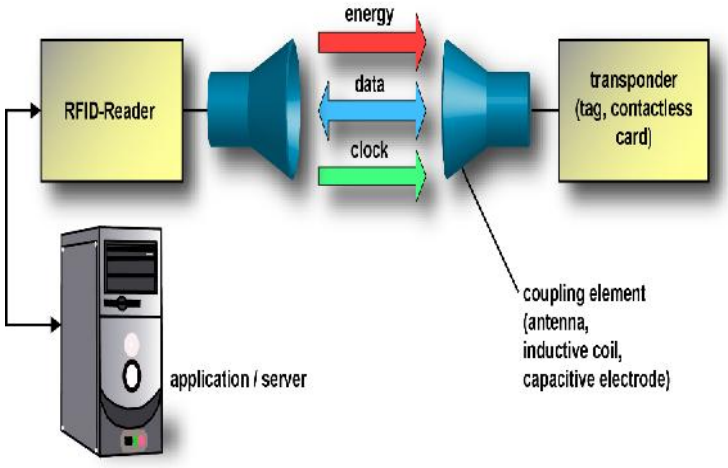


Figura 10. Componentes principales de cada sistema RFID.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

Lector o Transreceptor, se encarga de enviar señales cuando detecta la señal de algún tag; lee los datos y envía al subsistema para procesarla. Las señales pueden contener energía, según la estructura del tag.

Middleware o tratamiento de datos, a través del cual se procesa y almacena información, permite seleccionar datos útiles para desarrollar las aplicaciones estructuradas (software).

La etiqueta o tag, conformada por una antena que permite su comunicación y un chip que contiene una memoria interna con su identificación.

La tecnología RFID es la interacción a través de ondas de radio frecuencia, estas ondas pueden modificarse en frecuencia y amplitud. Según la figura 11 los espectros de frecuencias son de acuerdo al rango de operación.

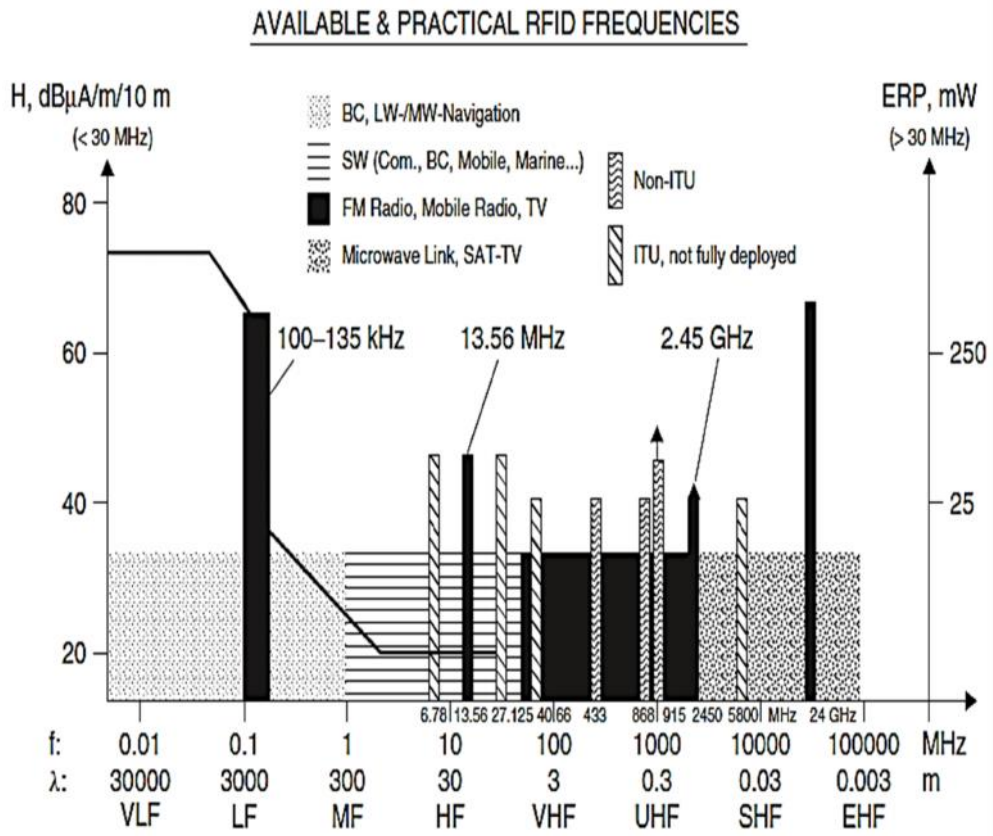


Figura 11. Espectro de frecuencias de acuerdo al rango de operación.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

En la Tabla 2 se detalla los rangos de frecuencias de operación de RFID.

Tabla 2. Frecuencias Tecnología RFID.

	Rango de operación	**Rango de lectura**	**Consumo de energía**	**Rango de transferencia de datos**	**Tamaño y longitud de onda**
Baja LF	30 y 300 kHz	10 cm	Bajo	Alto	10 y 1 km
Media MF	300 kHz y 3 MHz	20 cm	Bajo	Alto	1000 y 100 m.
Alta HF	3 y 30 MHz	1 m	Moderado	Alto	100 y 10 m.
Ultra Alta UHF	300 MHz y 3 GHz	Hasta 3 m	Moderado	Rápido	1 m y 1 dm
Súper Alta SHF	3 y 30 GHz	Hasta 10 m	Moderado	Súper rápido	1 dm y 1 cm

Fuente: Elaborado por el autor

Las frecuencias van desde los 30 kHz hasta 30 GHz, con un rango de 10 cm hasta 10 m y con una longitud de 1 cm hasta 1km.

Su estructura permite una interacción bidireccional. El proceso de comunicación se divide en dos, estos dependerán de forma directa de la frecuencia y la distancia, estos son:

- **Far-Field (Campo Lejano),** Utiliza acoplamiento capacitivo (o acoplamiento de propagación) que energiza la etiqueta RFID. El acoplamiento capacitivo se produce cuando la antena del lector RFID propaga la energía de RF hacia el exterior y esa energía se utiliza para activar la etiqueta. La etiqueta entonces devuelve una parte de esa energía RF a la antena del lector como una respuesta que se conoce como retrodispersión.

- **Near-Field (Campo Próximo),** Utiliza acoplamiento inductivo, lo que significa la aplicación de un campo magnético para activar la etiqueta RFID. Se crea un campo magnético en la región de campo cercano que permite que la antena del lector RFID active la etiqueta. La etiqueta entonces responde con una perturbación en el campo magnético que el lector recoge y decodifica.

Cada etiqueta contiene un registro exclusivo, el cual se denomina EPC (Código Electrónico de Producto), es un número de identificación propio, único e intransferible. Los EPC's tienen varios tamaños, entre ellos los tag's de 64 y 96 bits, aunque en la actualidad se utiliza de 128 y 256 bits.

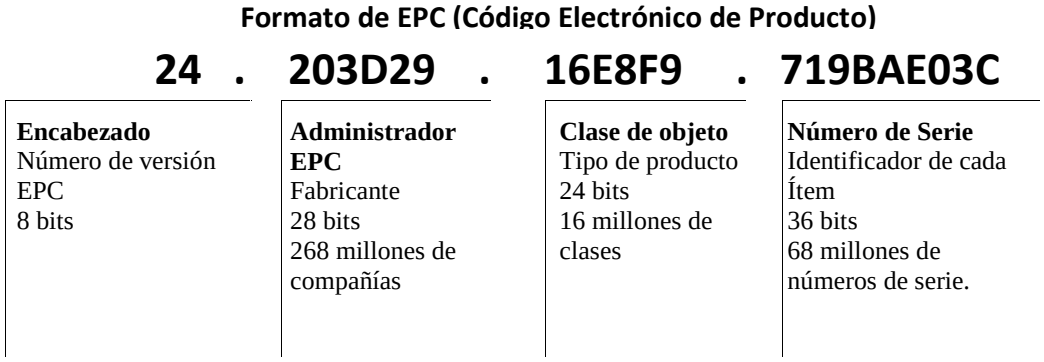


Figura 12. Estructura de un EPC

Fuente: RFID POINT
http://www.rfidpoint.com/que-es-rfid/el-estandar-epc/

La Figura 12 muestra las 4 partes fundamentales de los EPC: Cabecera indica la versión del código, utiliza 8 bits. Numero Manager indica a quien pertenece, utiliza 28 bits. Clase de objeto representa la clase de objeto a identificar, utiliza 24 bits y el número de serie único que usa como máximo 36 bits.

Los sistemas RFID cuentan con sistemas de seguridad, basados en procesos de encriptación para garantizar la transmisión y recepción de datos. Entre los cuales se pueden mencionar:

- **Criptografía simétrica o clave secreta:** Utiliza solo una clave para cifrar y descifrar (Cifrado simétrico), esta puede ser conocida por el transmisor y receptor.

- **Algoritmo DES (Estándar de Cifrado de Datos):** Es el más común en aplicaciones del área financiera, este tipo de cifrado adquiere una clave que sirve para codificar y decodificar información, está compuesta por 64 bits, 56 conforman la clave y los 8 restantes son de paridad.
- **IDEA (Algoritmo Internacional de Cifrado de Datos):** Es un cifrado de clave privada, en bloques de texto de 64 bits, utiliza una combinación de 128 bits que generan 52 subcombinaciones de 16 bits.
- **Criptografía asimétrica o clave pública:** Utiliza diferentes claves en el proceso de encriptar y desencriptar, en este sistema la clave para encriptar puede darse a conocer, utiliza números muy extensos, lo que retarda el proceso.

2.1.4.2.Componentes del sistema RFID

Según Finkenzeller Klaus (2010), los componentes de los sistemas RFID están basados de la siguiente manera.

2.1.4.2.1. Transponder, etiqueta o Tag.

Las etiquetas de RFID se constituyen de la siguiente manera:

- Memoria no volátil que contiene toda información sobre su identificación.
- Memoria ROM, almacena la codificación y funciones del tag.
- Memoria RAM, almacena la información durante la interacción con el lector.
- Antena, se emplea para energizar la etiqueta, la recepción y transmisión de datos.

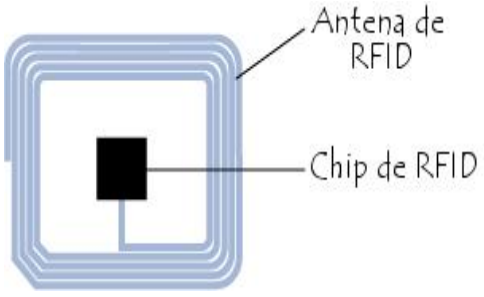


Figura 13. Estructura de una etiqueta.

Fuente: Identificación por radiofrecuencia (RFID)
https://ccsos2014291260.wordpress.com/clases-2/page/6/

La figura 13 muestra la forma física de la etiqueta, en la cual se muestra un solo chip en el cual están inmersas las memorias. Para emitir y recibir datos es necesario pequeñas cantidades de energía (micro watts), por lo cual las etiquetas se clasifican en activas y pasivas.

2.1.4.2.2. Etiquetas activas

Poseen su propia fuente de voltaje, esta puede ser a través de una batería o una célula solar; según la Figura 14, la fuente de alimentación se utiliza para proporcionar tensión al chip. Por lo tanto, el campo magnético o electromagnético recibido por el lector ya no es necesario alimentar el chip. Esta condición puede aumentar de forma sustancial el intervalo de comunicación (distancia). El tiempo estimado de vida útil esta entre 5 y 7 años.

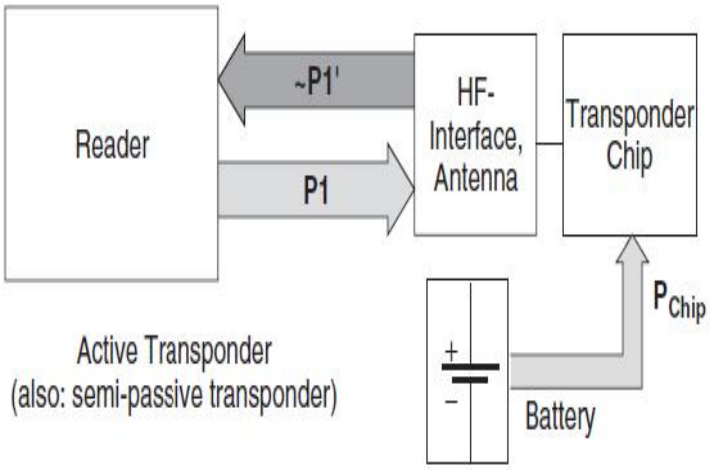


Figura 14. Esquema general de etiquetas activas.
Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

2.1.4.2.3. Etiquetas pasivas

Este tipo de etiquetas no poseen ninguna fuente de alimentación. A través de su antena, el campo magnético o electromagnético del lector proporciona toda la energía necesaria para operar. De acuerdo a la Figura 15, la etiqueta toma la energía que proviene del lector para el proceso de comunicación. Si la etiqueta está situada fuera del alcance del lector, esta no tiene ninguna fuente de voltaje, por lo tanto no podrá funcionar.

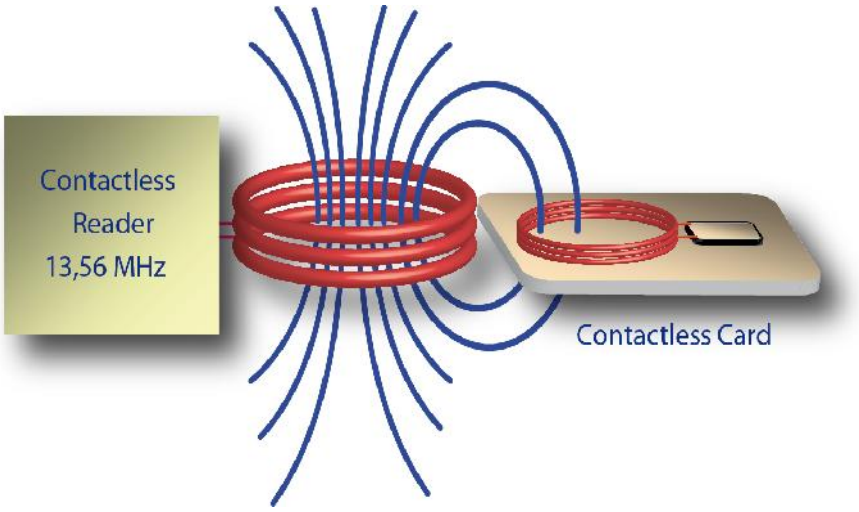


Figura 15. Funcionamiento de etiquetas pasivas.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

En la Tabla 3 se resume el rango de frecuencias en las cuales operan las etiquetas pasivas de RFID. Las frecuencias en las que operan las etiquetas pasivas tienen un alcance de 0 cm a 10 m, a mayor distancia la compatibilidad con líquidos y metales aumenta pero a mayor frecuencia mayor velocidad.

Tabla 3. Frecuencias de Etiquetas Pasivas

Frecuencia	Alcance	Compatibilidad con Líquidos y Metales	Velocidad	Precio
LF Baja Frecuencia (125 – 134 kHz)	0 – 10 cm	Fácil	Baja	Alta
HF Alta Frecuencia (13.56 MHz)	0 – 1 m	Difícil	Alta	Medio
UHF Ultra Alta Frecuencia (860 – 960 MHz)	10 cm – 10 m	Muy difícil	Muy alta	Bajo

Fuente: Tutorial sobre circuitos RFID.

2.1.4.2.4. Lectores

Los lectores son los encargados de enviar la señal de radio frecuencia, adquirir los datos de las etiquetas por radio frecuencia y transmitir la información obtenida hacia el sistema de procesamiento (midware), en un rango determinado, así como se muestra en la Figura 16. Estos pueden dividirse en:

- Lectores que poseen sistemas de bobina simple, permiten transmitir energía e información, su estructura es sencilla, son económicos pero de poco alcance.
- Lectores que se encuentran constituidos por sistemas interrogadores, dependen de la etiqueta, son sofisticados, permiten acondicionar, detectar y corregir errores, la frecuencia de operación es mayor.

Los lectores están conformados por un trascriptor de señales, que envía y recibe ondas de radio, poseen gran exactitud, eficacia y flexibilidad, producen bajo ruido de radiación. Para que un lector funcione correctamente y de acuerdo a un lugar determinado es necesario considerar lo siguiente:

- Sensibilidad: Capacidad de divisar señales procedentes de los tag's de hasta – 60 dBm de potencia.
- Selectividad: Permite seleccionar señales de etiquetas RFID, dentro de un rango de frecuencias recibidas, algunas señales poseen más potencia que otras, debido a que estas señales se encuentran cerca de las telefonías, lo que puede interferir.
- Alcance dinámico: Capacidad de detectar señales que provienen de varias etiquetas a ubicadas a diferentes distancias.
- Inter operatividad: Permite trabajar con diferentes marcas de etiquetas y lectores RFID.

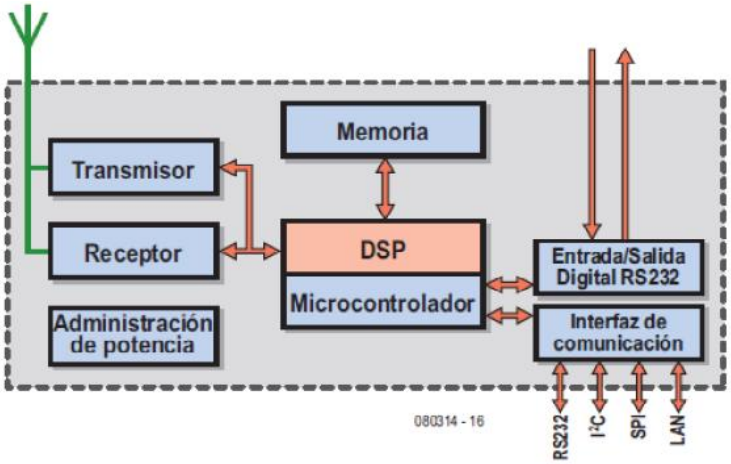


Figura 16. Diagrama de bloques de un lector.

Fuente*:* Principios de la tecnología RFID
http://docplayer.es/2773682-Capitulo-3-principios-de-la-tecnologia-rfid.html

2.1.4.2.5. Antenas

Las leyes de la física dicen que la radiación de las ondas electromagnéticas se puede observar en todos los conductores que transportan voltaje y / o corriente. En contraste con estos efectos, que tienden a ser parásitos, una antena es un componente en el que la radiación o recepción de ondas electromagnéticas se ha optimizado en gran medida para ciertos rangos de frecuencias mediante el ajuste fino de las propiedades de diseño según como se muestra en la figura 17.

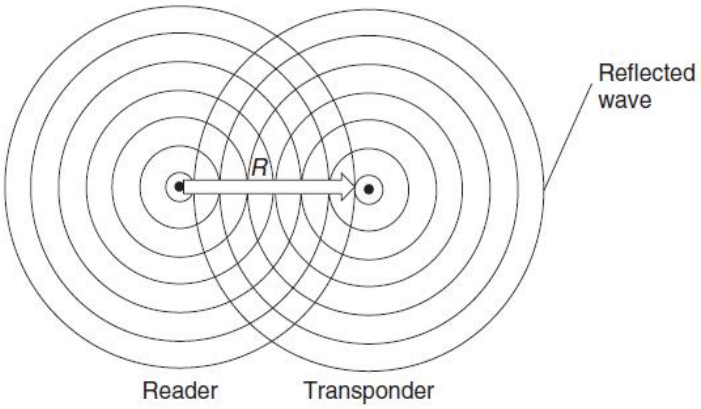


Figura 17. Ondas emitidas y reflejadas desde una etiqueta.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

2.1.4.3. Frecuencias RFID

Los sistemas RFID manejan diferentes frecuencias de operación entre las principales se pueden mencionar las siguientes:

- Baja frecuencia está en el rango de 9 kHz a 135 kHz, posee una gran demanda a nivel mundial, se utilizan a cortas distancias de materiales metálicos. La distancia de reconocimiento está por debajo de 1.5 m, sus aplicaciones se enfocan a identificar objetos y animales.
- Alta frecuencia está en el rango de los 13,56 MHz, es muy conocida, no actúan a distancias cortas de los metales. Por lo general se utilizan en rastreo de productos, traslado de equipajes en terminales áreas o controles de acceso.
- Ultra-alta frecuencia se encuentra en el rango de 433 MHz y 860-960 MHz: Los equipos con esta clase de frecuencias no pueden ser utilizados a nivel mundial ya que no poseen una normativa global para su uso y su aplicación depende de la legalidad del país. Su principal aplicación es la trazabilidad con etiquetas activas.
- Microondas operan a 2,45 GHz y 5,8 GHz, se emplean con mayor frecuencia en las etiquetas activas, para cubrir grandes distancias con velocidades de transmisión elevadas. Su principal aplicación es el monitoreo y trazabilidad de personas y objetos.

2.1.4.4. Middleware

La Figura 18 muestra que el middleware es un sistema que permite enlazar el sistema RFID con el sistema de información. La información obtenida desde el sistema RFID, se procesará con la ayuda del programa de gestión, el cual está enlazado a una base de datos en tiempo real que garantizará rapidez y efectividad.

Figura 18. Interacción del MIDDLEWARE

Fuente: Middleware RFID
https://ccsos2014291260.wordpress.com/clases-2/page/6/

2.1.4.5. Funcionamiento

La figura 19 muestra el proceso básico de comunicación entre el lector, el medio de transmisión y las etiquetas de un sistema RFID los cuales se comunican sin una línea de vista.

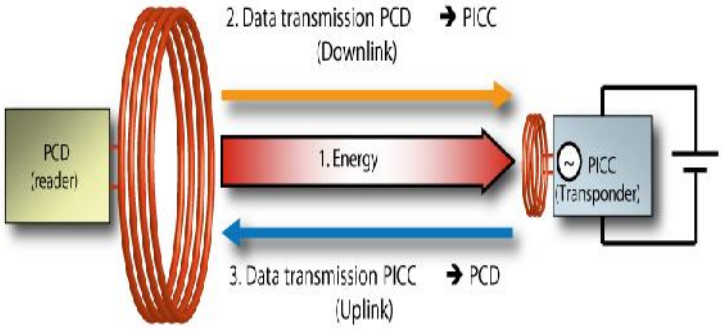


Figura 19. Proceso de comunicación sistemas RFID.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

La transferencia de datos entre el lector y la etiqueta en sistema RFID se muestra en la figura 20, en esta requiere de tres bloque principales:

- Codificación de señal (procesamiento de señal) y el modulador (circuito portador) en el lector (transmisor),
- Medio de transmisión (canal) y el demodulador (circuito portador)
- Decodificación de señales (procesamiento de señales) en la etiqueta (receptor).

Un sistema de codificación de señales toma el mensaje a transmitir y su representación de señal, lo ajusta a las características del canal de transmisión. Este proceso implica proporcionar al mensaje cierto grado de protección contra la interferencia o la colisión y contra la modificación intencional de ciertas características de la señal. La codificación de la señal no debe confundirse con la modulación, por lo que se la denomina codificación en la banda base.

La modulación es el proceso de alterar los parámetros de señal de un portador de alta frecuencia, es decir, su amplitud, frecuencia o fase, en relación con una señal modulada, la señal de banda base.

El medio de transmisión, transmite el mensaje a una distancia predeterminada. Los únicos medios de transmisión utilizados en los sistemas RFID son los campos magnéticos (acoplamiento inductivo) y las ondas electromagnéticas (microondas).

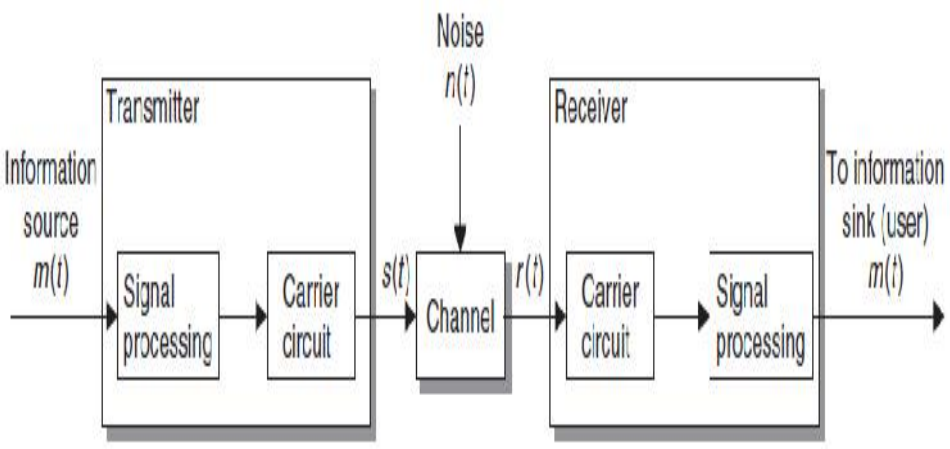


Figura 20. Transmisión de datos en un sistema RFID.

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

La tarea de decodificación de señal es reconstruir el mensaje original desde la señal recibida codificada en banda base y reconocer cualquier error de transmisión y marcarlo como tal.

2.1.4.6. Tipos de modulación en RFID

De acuerdo a Finkenzeller Klaus (2010), los procedimientos de modulación digital utilizados en los sistemas RFID son los siguientes:

- ASK: Modulación por desplazamiento de amplitud.
- FSK: Modulación por desplazamiento de frecuencia.
- PSK: Modulación por desplazamiento de fase.

2.1.4.6.1. ASK (Modulación por desplazamiento de amplitud)

Según la Figura 21, en la modulación por desplazamiento de amplitud, la amplitud de una oscilación portadora se conmuta entre dos estados 0 y 1 (codificación) por una señal de código binario.

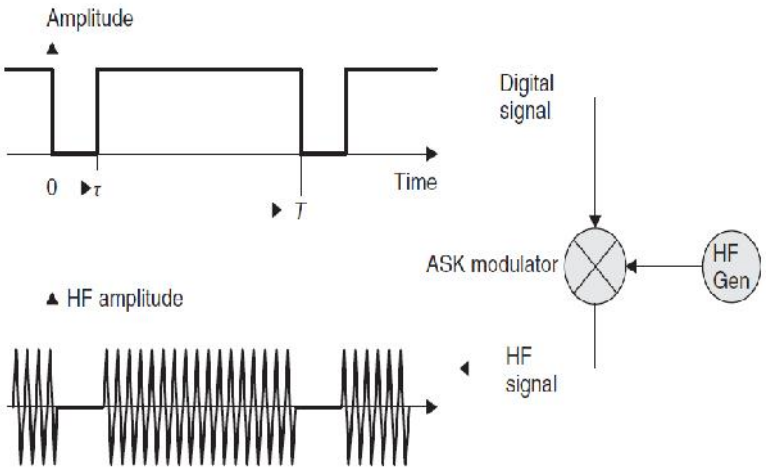


Figura 21. Modulación ASK

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

2.1.4.6.2. FSK (Modulación por desplazamiento de frecuencia)

De acuerdo a la Figura 22, en Modulación por desplazamiento de frecuencia, la frecuencia de una oscilación portadora se conmuta entre dos frecuencias f1 y f2 por una señal de código binario.

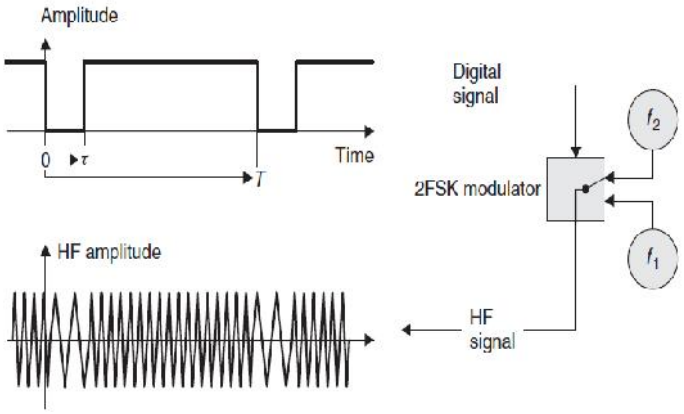


Figura 22. Modulación FSK

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

2.1.4.6.3. PSK (Modulación por desplazamiento de fase)

La Figura 23 muestra que en la modulación por desplazamiento de fase, los estados binarios 0 y 1 de una señal de código se convierten en estados de fase correspondientes de la oscilación portadora, en relación con una fase de referencia. En PSK la señal se conmuta entre los estados de fase 0◦ y 180◦.

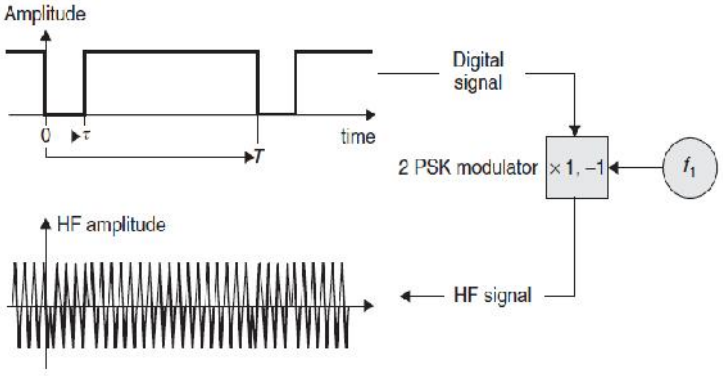


Figura 23. Modulación PSK

Fuente: RFID – HANDBOOK
Finkenzeller, K (2010)

2.2. Marco Conceptual

- **Algoritmo.-** Es la unión de varias operaciones que se ejecutan de forma ordenada y sistemática para dar solución a un determinado problema.
- **Arduino.-** Es una placa electrónica que está constituida por un microcontrolador de la marca ATMEGA, que permite desarrollar soluciones de acuerdo a su entorno exterior mediante un lenguaje de programación de código abierto.
- **Banda base.-** Es la frecuencia generada por un dispositivo en su estado natural.
- **Campo magnético.-** Es la interacción de corriente eléctrica con un resultado de un campo de fuerza que es directamente proporcional al flujo eléctrico.
- **Codificación.-** Es el proceso de conversión de datos a través de normas o estándares establecidos.
- **Código binario.-** Es el conjunto de instrucción de un microcontrolador que pueden ser representadas solo por 0 y 1.
- **Conmutación.-** Es el proceso ordenado de pasar de un estado lógico a través de una instrucción.
- **Decodificación.-** Es el proceso ordenado y sistemático de convertir los datos codificados en información con la aplicación de normas o estándares que permite obtener la señal en su estado original antes de ser codificado.
- **Demodulación.-** Es un proceso sistemático para recuperar información transferida a través de una señal portadora.
- **Discapacidad.-** Deficiencia o limitaciones de actividades están pueden afectar de manera corporal o intelectual.
- **EMI (Interferencia Electromagnética).-** Es una señal que se propaga en el espacio y degrada las señales que se encuentran en su entorno.
- **Encriptación.-** Es el proceso de convertir la información en caracteres especiales, la cual solo podrá ser legible mediante una contraseña.
- **Longitud de onda.-** Es la distancia transcurrida entre dos puntos determinados en función del tiempo.
- **Microcontrolador.-** Sistema electrónico que permite desarrollar aplicaciones de

acuerdo a su entorno a través de un lenguaje de programación que se estructura según los requerimientos del usuario final.

- **Modulación.-** Es el proceso de agregar información a una señal electrónica para ser transmitida.
- **Ondas electromagnéticas.-** Son ondas que viajan en el espacio a muy alta velocidad (300.000 km/s), sin necesidad de utilizar un medio físico.
- **Oscilación.-** Es un movimiento continuo que fluctúa en el tiempo.
- **Portadora.-** Son las ondas que sirve de medio para la transmisión de datos desde un transmisor a un receptor.
- **Protocolo.-** Es el conjunto de normas, estándares que permiten establecer normativas para la elaboración de un determinado fin.
- **Radiación.-** Es el proceso en el cual partículas de energía son irradiadas y se propagan en el espacio.
- **RFID.-** Proviene de las siglas en inglés (Radio Frequency Identification), son dispositivos que permiten almacenar y recuperar información de forma remota.
- **Semáforo.-** Dispositivo electrónico conformado por tres luces de color (rojo, amarillo y verde) que regula el flujo vehicular y peatonal a través de la interacción de dichas luces en función de un determinado tiempo asignado a cada luz.
- **Sincronización.-** Es el proceso por el cual dos señales pueden coincidir en función de la fase y tiempo.
- **Shields.-** Son módulos electrónicos que permiten extender las posibilidades de funcionamiento de un circuito principal.

SECCIÓN III

3.1. Metodología.

El diseño y prototipo de sistema inteligente para semáforos del Distrito Metropolitano de Quito para dar prioridad a personas con discapacidad fue realizado en función de datos estadísticos con un tipo de muestra cuantitativa.

De acuerdo a la figura 24 en la Provincia de Pichincha Cantón Quito existen 55.932 personas con algún tipo de discapacidad distribuidos en 25.613 correspondiente al sexo femenino con el 46% y 30.319 al sexo masculino con el 54%.

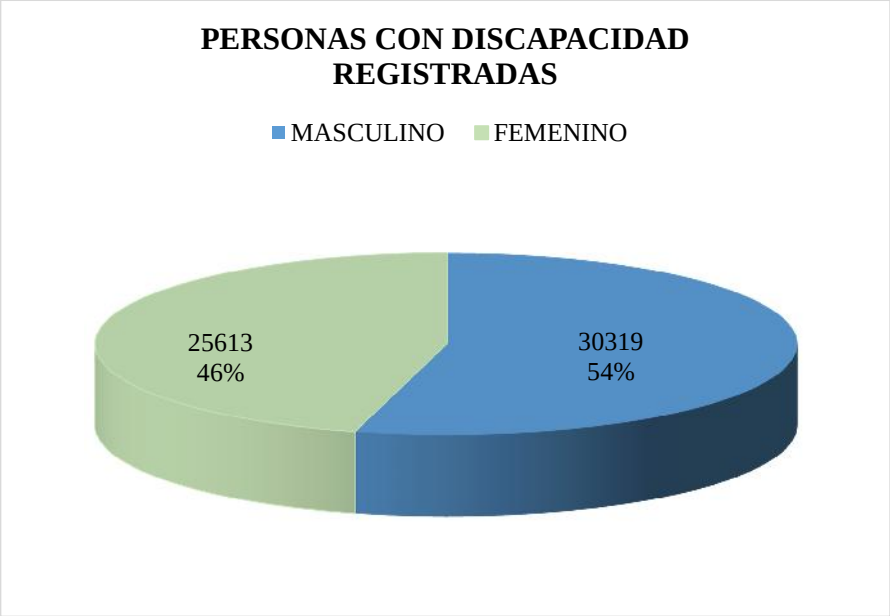


Figura 24. Grafica estadística de personas con discapacidad

Fuente: Consejo Nacional para la igualdad de discapacidades (CONADIS) 2016

El mayor índice de personas con algún tipo de discapacidad oscila entre las edades de 30 a 65 años con 29.846 personas y mayores de 65 años con 10.250 según se muestra la figura 25.

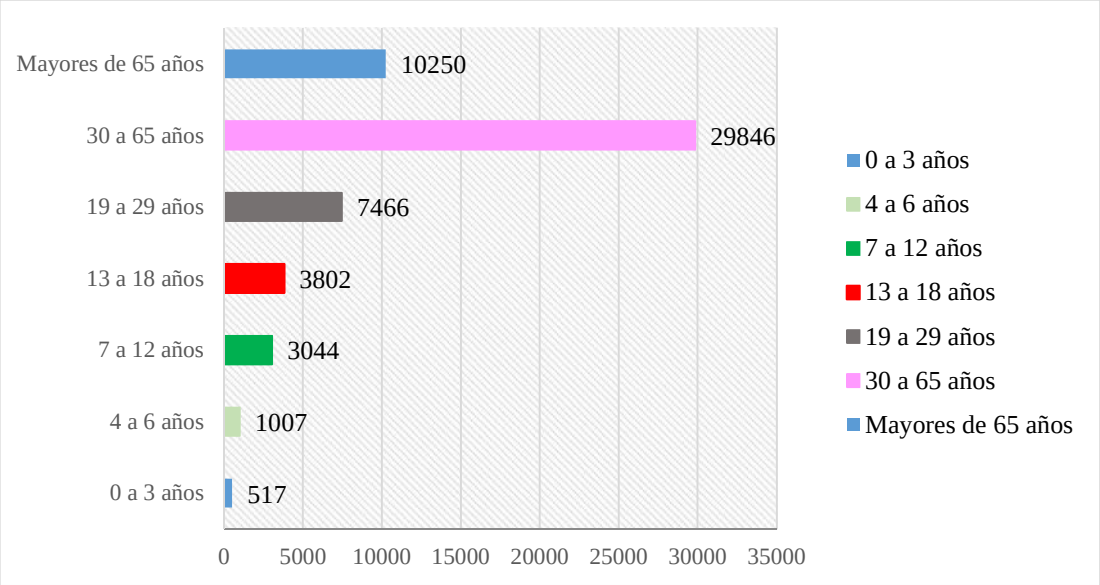


Figura 25. Gráfica estadística de discapacidad de acuerdo a la edad

Fuente: Consejo Nacional para la igualdad de discapacidades (CONADIS) 2016.

De acuerdo a los datos proporcionados por el Consejo Nacional para la igualdad de Discapacidades (CONADIS), el grupo con mayor vulnerabilidad a sufrir algún tipo de incidente o accidente son personas que poseen discapacidad física y visual entre el rango de edad de 18 a más 65 años.

De acuerdo al Art. 200 de la Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial; y su Reglamento establece que una persona con discapacidad, movilidad reducida y grupos vulnerables tendrán los siguientes derechos y preferencias.

De acuerdo al inciso: a) en las intersecciones, pasos peatonales, cruces cebra y donde no existan semáforos tendrán derecho de paso los peatones; es obligatorio que todo usuario vial ceder el paso y mantenerse detenido hasta que concluya el cruce peatonal.

Conforme a la ley vigente y los datos estadísticos obtenidos, movilizarse en la urbe quiteña para las personas con discapacidad física y visual es un gran desafío, debido a que los vehículos motorizados no cumple inciso: a) y el sistema de semaforización actual no cuenta con algún tipo de subsistema autónomo que les permita dar prioridad a este grupo vulnerable de la sociedad.

En función a lo antes mencionado es necesario proponer un prototipo de sistema inteligente enmarcado en una tecnología que permita desarrollar un modelo incluyente que interactúe de forma autónoma, eficiente y confiable para los semáforos del Distrito Metropolitano de Quito en los cruces de vías, con el fin de facilitar la interacción de este grupo vulnerable con la sociedad.

3.2. Propuesta

3.2.1. Diseño del sistema inteligente

3.2.1.1. Diagrama de Bloques

En la Figura 26 se muestra los bloques por los cuales estará constituido el sistema inteligente de cruce peatonal para intersecciones críticas que se enfocará a dar prioridad a las personas con algún tipo de discapacidad.

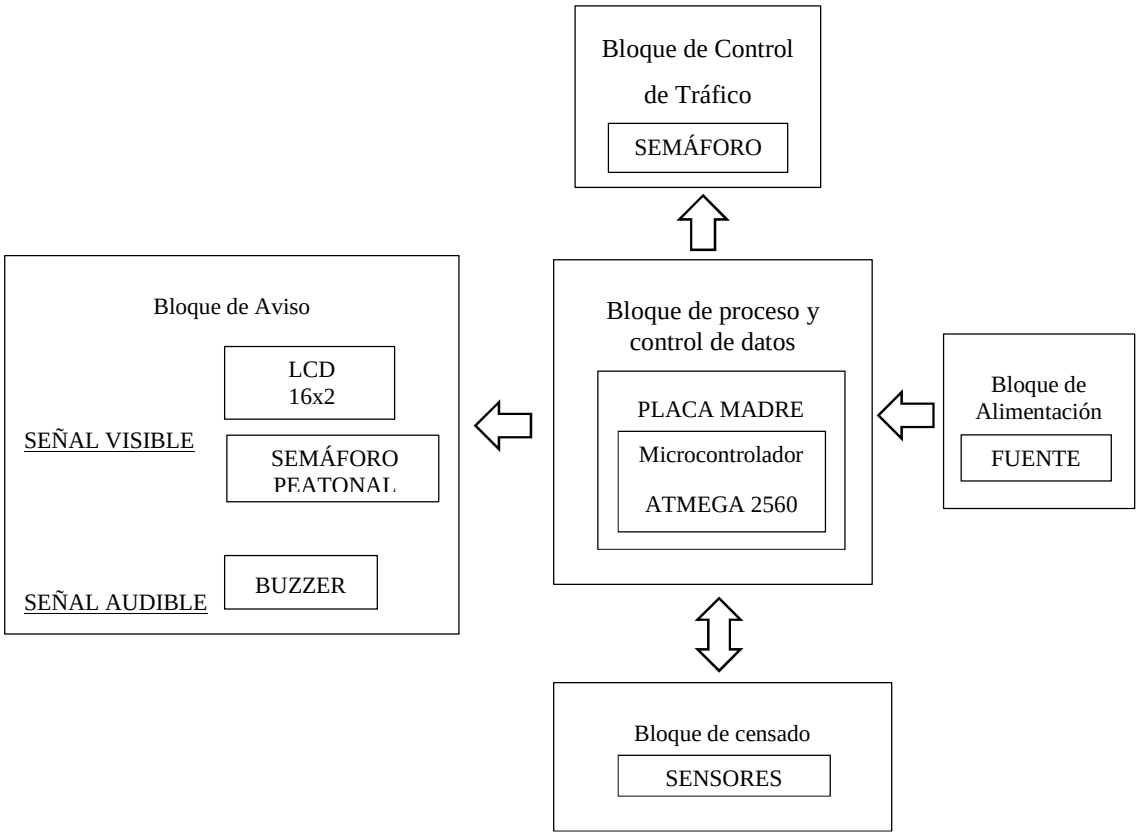


Figura 26. Diagrama de Bloques Sistema Inteligente

Fuente: Elaborado por el autor

Cada uno de los bloques estará constituidos de la siguiente manera:

- **Bloque de alimentación:** Este sistema se lo hará a través de una fuente externa de poder mediante un convertidor de AC a DC que reducirá el voltaje de 110 V a 6 V.
- **Bloque de control y proceso de datos:** Este circuito está conformado por la placa madre y un Arduino MEGA con un microcontrolador ATmega 2560, en el cual convergerán las señales de bloques del censado que controlará los bloques de aviso y control de tráfico el cual interactuará en conjunto con los procesos complementarios de conmutación y regulación de voltaje por lo cual se utilizarán transistores 2N2222 y resistencias de 200 y 220 Ω al 2% con una potencia de ½ W.
- **Bloque de censado:** Constituido por sensores RFID – RC522 como se indica en la Figura 27, que a su vez se dividirán en lectores y etiquetas. Su capacidad de detección máxima es de 4 cm, alimentación de 3.3 V con un consumo de 13 mA, frecuencia de operación de 13,56 MHz, velocidad de procesamiento hasta 10 Mbps y con una dimensión física de 40 x 60 mm.

Figura 27. RFID – RC522

Fuente: Punto Flotante S.A.
http://www.puntoflotante.net/

El proceso de comunicación de datos con el bloque de control y proceso de datos se realizará a través de un protocolo SPI. Como se aprecia en la figura 28, la comunicación es bidireccional.

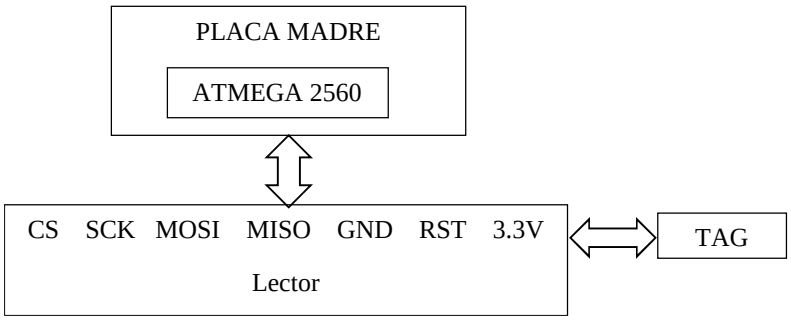


Figura 28. Comunicación sensores RFID

Fuente: Elaborado por autor

- **Bloque de aviso:** Como se observa en la figura 29, este bloque está conformado por señal auditiva y visual.

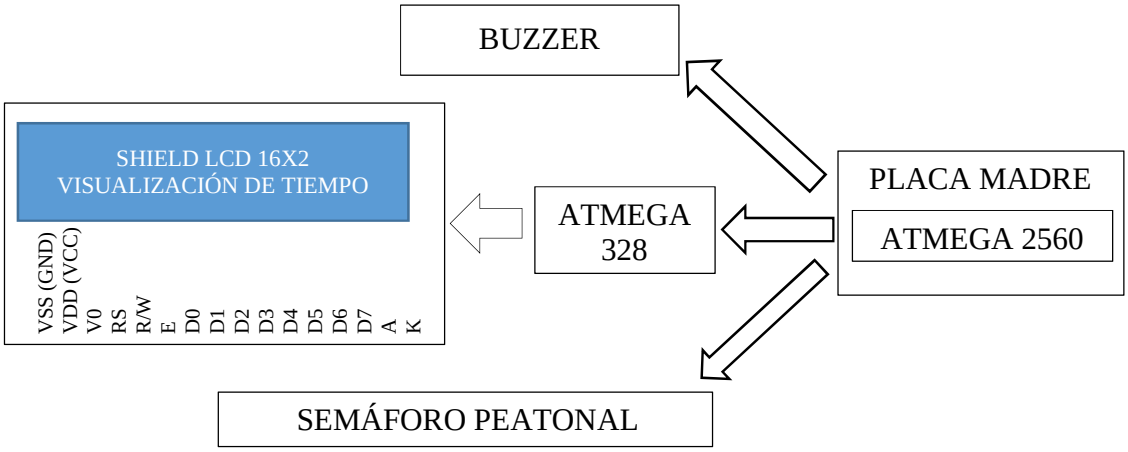


Figura 29. Comunicación Bloque de aviso

Fuente: Elaborado por el autor

- ✓ Las señales auditivas se producirán en buzzers.
- ✓ Las señales visuales se reflejaran de dos maneras

  - Semáforo Peatonal.- Estará conformado por uno led rojo y otro verde, su control será mediante el microcontrolador ATMEGA 2560.

- Pantalla LCD 16x2.- Permitirá visua hlizar el tiempo que está transcurriendo en el semáforo peatonal, estará conectado a un microcontrolador Arduino Uno ATMEGA 328.

Figura 30. Shield LCD 16X2

Fuente: Create Arduino
http://www.prometec.net/lcd-keypad-shield/

- **Bloque de control de tráfico:** Este empleará semáforos de flujo vehicular con tres luces LED's de alto brillo (rojo, verde y amarillo); su funcionamiento será gobernado por el bloque de proceso y control de datos que interactuará con el bloque de censado.

3.2.2. Diseño de Circuito

De acuerdo a la figura 31 el circuito estará compuesto por ocho lectores RFID-RC522; veinte leds emisores de luz que se subdividirán en doce a los semáforos vehiculares y ocho para cruces peatonales, a los ánodos se conectarán resistencias de 220Ω y los cátodos se unirán en serie que descenderán a tierra; cuatro transistores 2N2222, a la base de cada transistor se le proveerá una resistencia de 220Ω, cuatro buzzers (piezo eléctrico) los cuales serán enlazados a los emisores de cada transistor.

Todos estos componentes se integrarán a un Arduino Mega2560 a través de la placa madre, pines de 5V, 3.3V y GND así los elementos que requieran ser energizados.

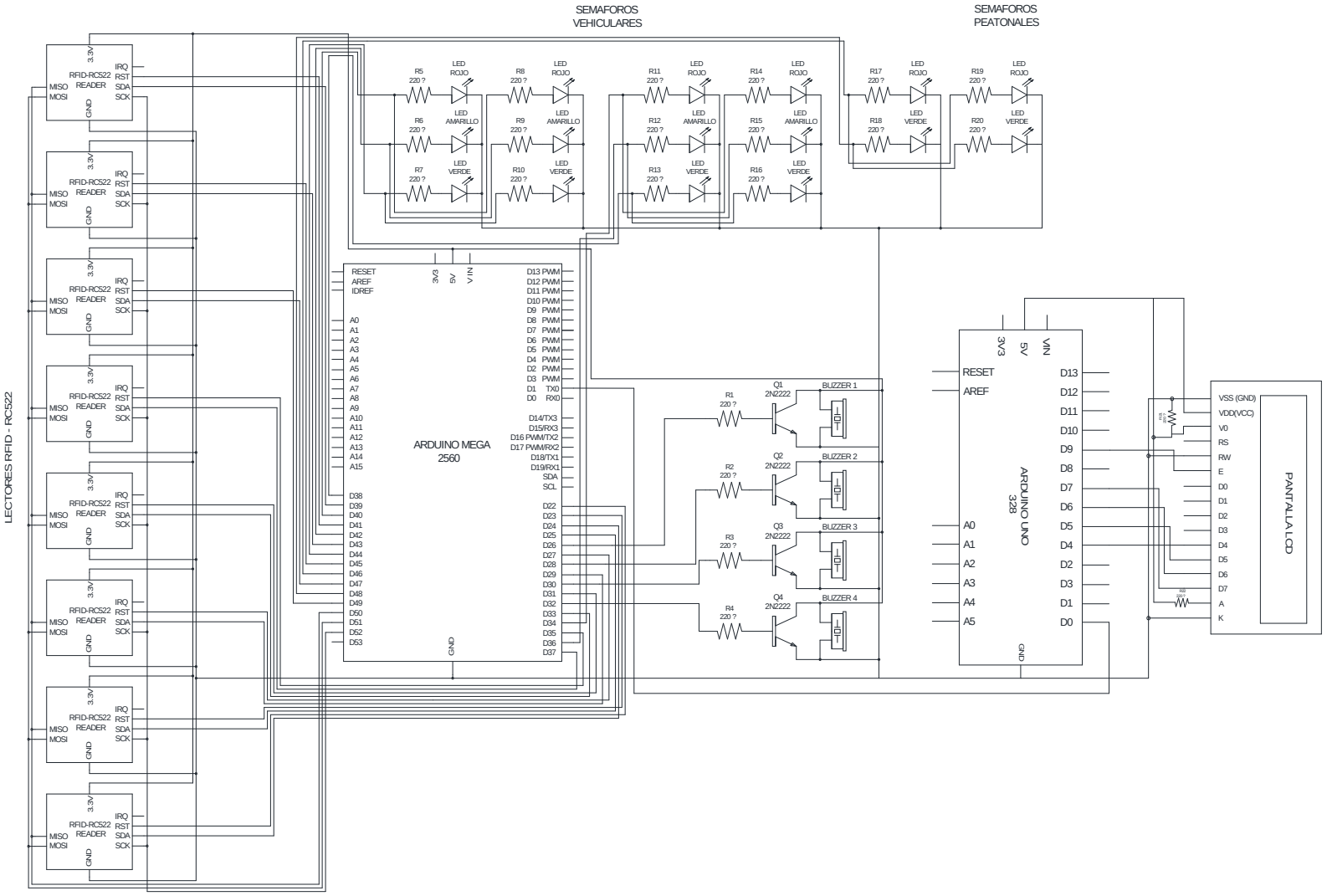


Figura 31. Diagrama de circuito
Fuente: Elaborado por el autor

3.2.3. Programación

Para el siguiente sistema de semaforización de las la intersección de la Av. 10 de Agosto y Av. Colon se partirá desde la determinación de tiempos de los semáforos vehiculares para luego ser implementados de acuerdo al flujograma correspondiente.

3.2.3.1. Determinación de tiempos para semáforos vehiculares.

Para determinar el tiempo de duración de cada una de las luces (rojo, amarillo y verde) se procedió a medir tiempo de cambio de luces de los semáforos de tres intersecciones en consideración con el flujo vehicular en hora normal y hora pico, los tres cruces viales que se evaluaron fueron.

Figura 32. Intersección Av. América y Av. Cuero y Caicedo
Fuente: Google Maps

Figura 33. Intersección Av. Mariana de Jesús y Av. 10 de Agosto
Fuente: Google Maps

Figura 34. Intersección Av. 10 de Agosto y Av. Cristóbal Colón
Fuente: Google Maps

En la Tabla 4 se puede apreciar la disminución de tiempo en el cambio de luces entre la hora normal y hora pico, de acuerdo a la Figura 32, 33, 34.

Tabla 4. Cronometraje de tiempo de luces

	SENTIDO NORTE – SUR					
	INTERSECCIÓN 1 Av. Mariana de Jesús y Av. 10 de Agosto		**INTERSECCIÓN 2 Av. América y Av. Cuero y Caicedo**		**INTERSECCIÓN 3 Av. Cristóbal Colón y Av. 10 de Agosto**	
	LUZ VERDE	**LUZ ROJO**	**LUZ VERDE**	**LUZ ROJO**	**LUZ VERDE**	**LUZ ROJO**
MIERCOLES Y VIERNES HORA NORMAL 10:00 am - 15:00 pm	1 m 18 s	1 m 01 s	1 m 9 s	42 s	1 m 01 s	43 s
MIERCOLES Y VIERNES HORA PICO 7:00 – 9:00 am 16:00 – 19:00 pm	1 m 18 s	1 m 01 s	1 m 21 s	37 s	1 m 14 s	30 s

Fuente: Elaborado por el autor

De acuerdo a los tiempos cronometrados en la intersección 2, en hora pico la luz verde de se incrementa en 12 segundos y la luz roja tiene una disminución de 5 segundos.

Mientras en la intersección 3 en hora pico la luz verde de se incrementa en 13 segundos y la luz roja tiene una disminución de 13 segundos.

De acuerdo a los datos obtenidos se determinó que existen dos tiempos de sincronismos para los semáforos en los ejes longitudinales con mayor flujo de tráfico en horas pico.

3.2.3.2. Determinación de tiempo de los semáforos peatonal para personas con discapacidad.

Según los estudios realizados y obtenidos por el Consejo Nacional para la Igualdad de Discapacidad (CONADIS) se ha determinado que en la Provincia de Pichincha - Cantón Quito existe 55.932 personas con algún tipo de discapacidad según como se observa en la figura 35.

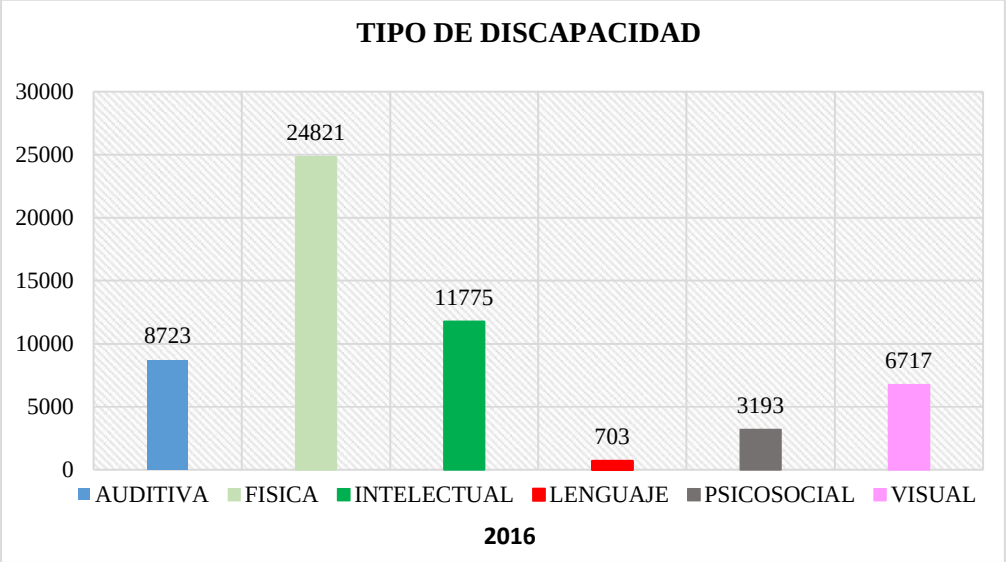


Figura 35. Tabla estadística de personas con discapacidad

Fuente: Consejo Nacional para la Igualdad de Discapacidad (CONADIS) 2016

De acuerdo al estudio realizado en los cruces peatonales, en la hora normal (10:00 am – 15:00 pm) existió un promedio de 6 personas con discapacidad que hacen uso de los semáforos peatonales y en la hora pico (7:00 am – 9:00 am / 16:00 pm – 19:00 pm) hubo un promedio de 8 personas con discapacidad; el incremento de personas en la hora pico está relacionado con la jornada laboral, como se evidencia en la figura 36.

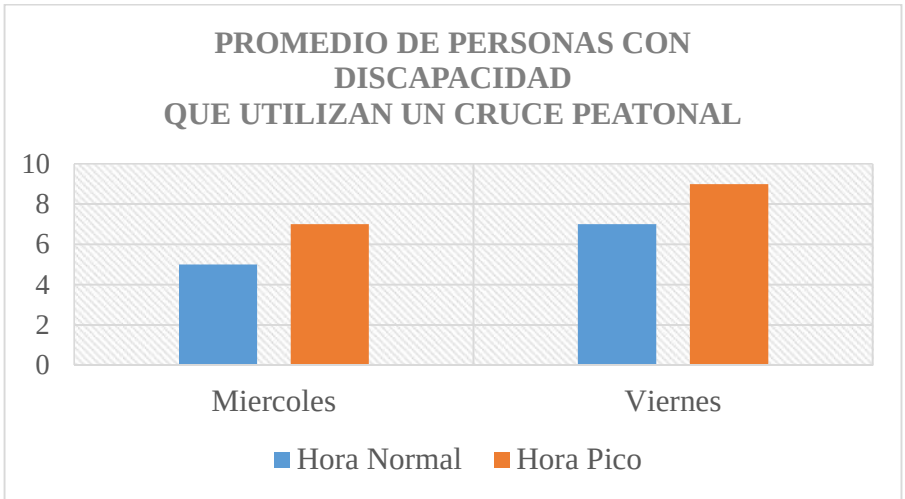


Figura 36. Cuadro estadístico de N° personas vs Días

Fuente: Elaborado por el autor

Según las investigaciones realizadas a personas con discapacidad dentro del Distrito Metropolitano de Quito, se ha obtenido que el tiempo de cruce en las vías para las personas discapacitadas es muy corto en especial en personas no videntes y personas de la tercera edad que sufran algún tipo de discapacidad según como se muestra en la figura 37, por lo que se ha visto en la necesidad de aumentar el tiempo de los semáforos peatonales en 15 segundos más al tiempo establecido.

a) b)

Figura 37. Personas con discapacidad física a) Persona tercera edad con discapacidad, b) Persona no vidente

Fuente: Elaborado por el autor

3.2.3.3. Flujograma de funcionamiento de semáforos con integración del sistema inteligente.

De acuerdo a la Figura 38 el funcionamiento del sistema inteligente se desarrollará de la siguiente manera:

- Se inicia el programa con una variable la cual se denominará Bandera este podrá tomar valores de 0 o 1 según el caso.
- Determinará dos condiciones; condición 1 hora pico, condición 2 hora normal.

CONDICIÓN 1

- Si la respuesta es negativa, comenzará la condición de flujo hora normal donde dará paso a la vía 1 (Av. 10 de Agosto) y se detendrá la vía 2 (Av. Colón) y se iniciará un temporizador de 1 minuto.
- Pregunto, temporizador llego a su fin (si o no).
- Si la respuesta es negativa, iniciará una subrutina donde pregunta detecto un Tag (bandera es = 0) cuando es negativo inicia un bucle de lazo cerrado de 1 minuto; si la respuesta es afirmativa (Bandera = 1), indica que se ha detectado un tag por lo cual se sumará un tiempo de 15 segundos más al proceso, es decir el bucle de lazo cerrado ahora vale 1 minuto 15 segundos.
- Si la respuesta en el temporizador es afirmativa se detendrá la vía 1 (Av. 10 de Agosto) y permitirá el paso en la vía 2 (Av. Colón) y se iniciará un temporizador de 43 segundos.
- Pregunto, temporizador llego a su fin (si o no).
- Si la respuesta es negativa, iniciará una subrutina donde pregunta detecto un Tag (bandera es = 0) si la respuesta es negativa inicia un bucle de lazo cerrado de 43 segundos; si la respuesta es afirmativa (Bandera = 1), indica que se ha detectado un tag por lo cual se sumará un tiempo de 15 segundos más al proceso, es decir el bucle de lazo cerrado ahora vale 58 segundos.

CONDICIÓN 2

- Si la respuesta es afirmativa, comenzará la condición de flujo hora pico donde dará paso a la vía 2 (Av. Colón) y se detendrá la vía 1 (Av. 10 de Agosto) y se iniciará un temporizador de 30 segundos.
- Pregunto, temporizador llego a su fin (si o no).
- Si la respuesta es negativa, iniciará una subrutina donde pregunta detectó un Tag (bandera es = 0) si la respuesta es negativo inicia un bucle de lazo cerrado de 30 segundos; si la respuesta es afirmativa (Bandera = 1), indica que se ha detectado un tag por lo cual se sumará un tiempo de 15 segundos más al proceso, es decir el bucle de lazo cerrado ahora vale 45 segundos.

- Si la respuesta en el temporizador es afirmativa se detendrá la vía 2 (Av. Colón) y permitirá el paso en la vía 1 (Av. 10 de Agosto) y se iniciará un temporizador de 1 minuto 14 segundos.
- Pregunto, temporizador llego a su fin (si o no).
- Si la respuesta es negativo, iniciará una subrutina donde pregunta detecto un Tag (bandera es = 0) si la respuesta es negativo inicia un bucle de lazo cerrado de 1 minuto 14 segundos; si la respuesta es afirmativa (Bandera = 1), indica que se ha detectado un tac por lo cual se sumará un tiempo de 15 segundos más al proceso, es decir el bucle de lazo cerrado ahora vale 1 minuto 29 segundos.

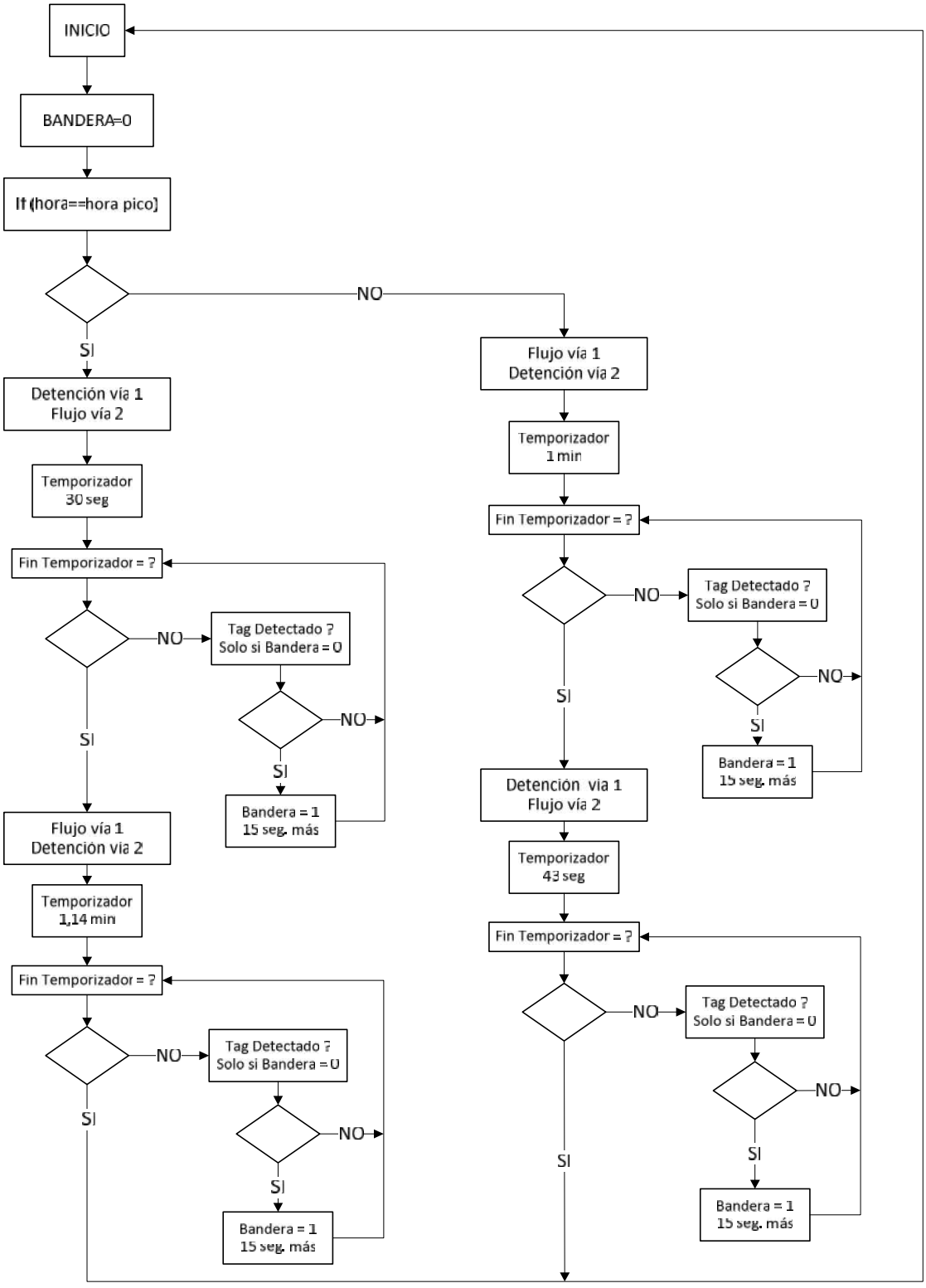


Figura 38. Flujograma de sistema inteligente
Fuente: Elaborado por el autor

3.2.4. Diseño de maqueta demostrativa

Para el presente proyecto se determinó realizar una maqueta demostrativa con las intersecciones de la Av. 10 de Agosto y Av. Cristóbal Colon, la cual se muestra según el plano de la Figura 39.

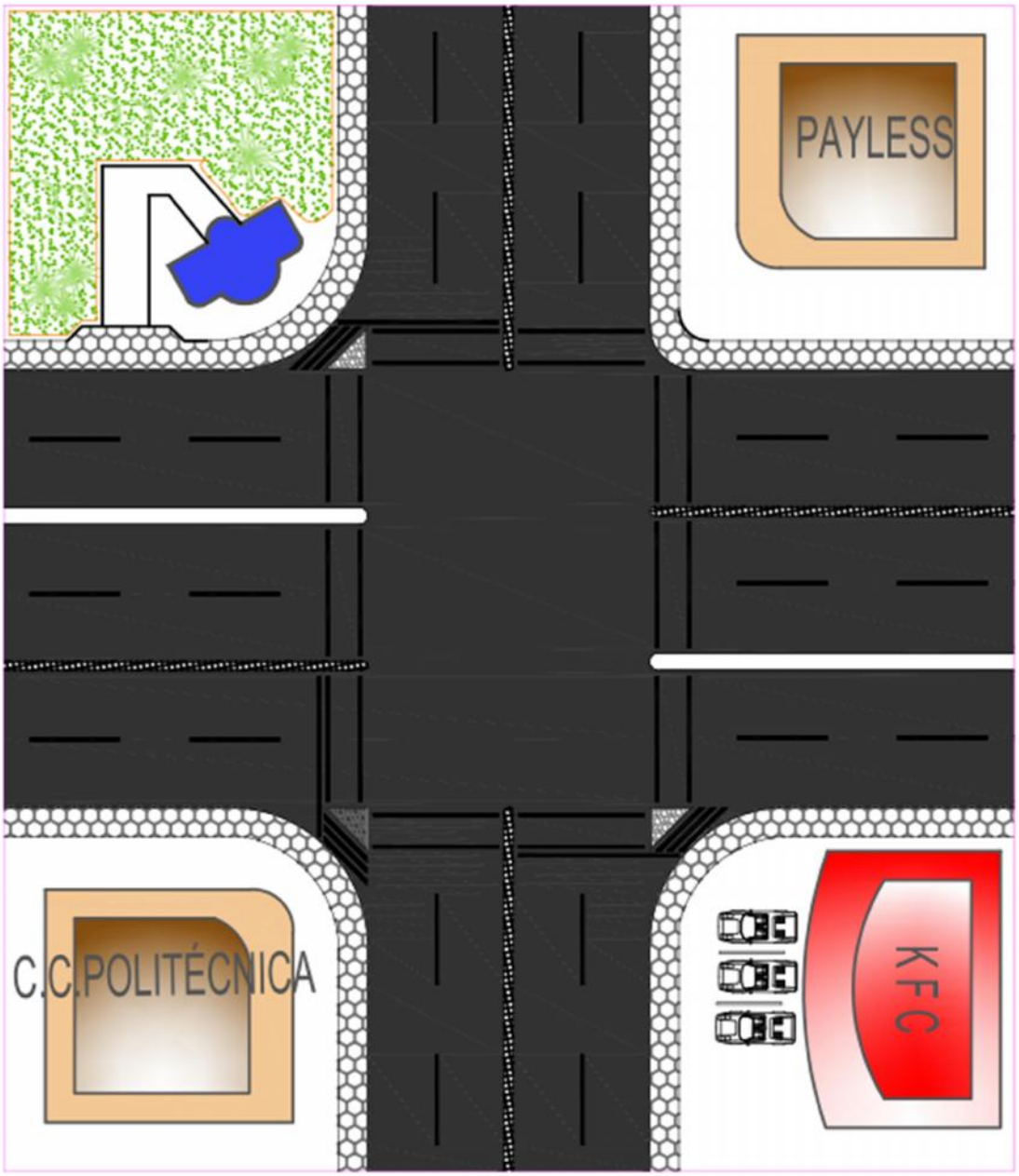


Figura 39. Planos de intersección de dos vías.
Fuente: Elaborado por el autor

3.2.5. Implementación del sistema inteligente

El sistema inteligente se implementará de acuerdo al siguiente orden.

3.2.5.1. Placa Madre

La placa madre se elaboró de acuerdo al circuito en PCB, según como se aprecia en la figura 40, en este se incorporarán borneras para los RFID, LEDS, transistores, resistencias y buzzers, el cual facilitará las conexiones del cableado.

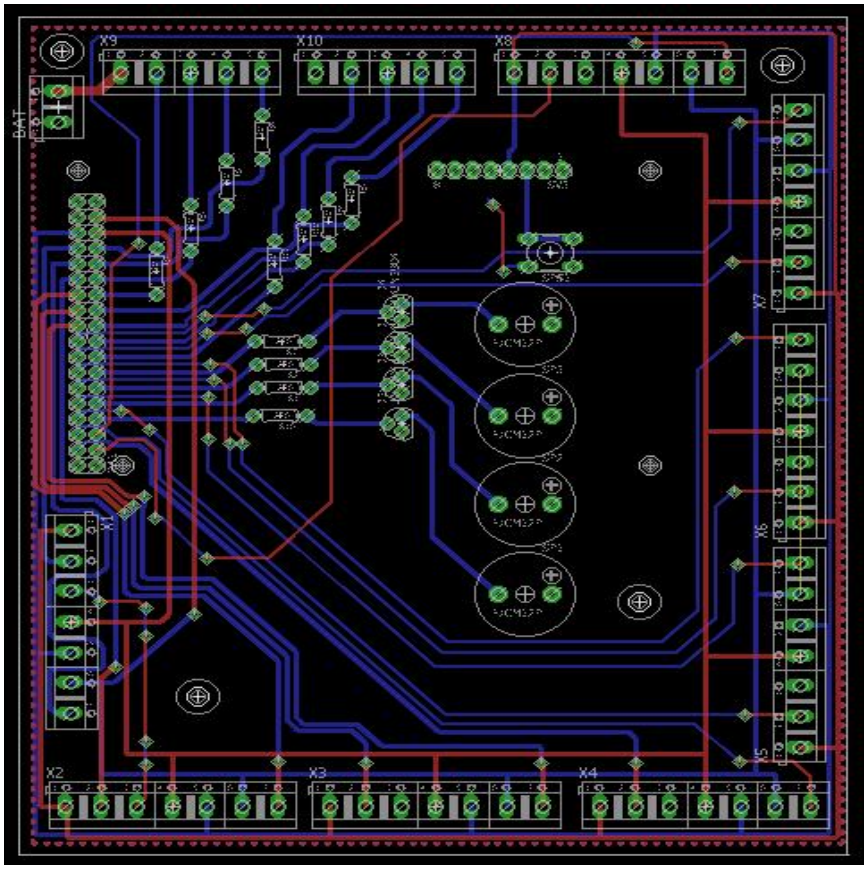
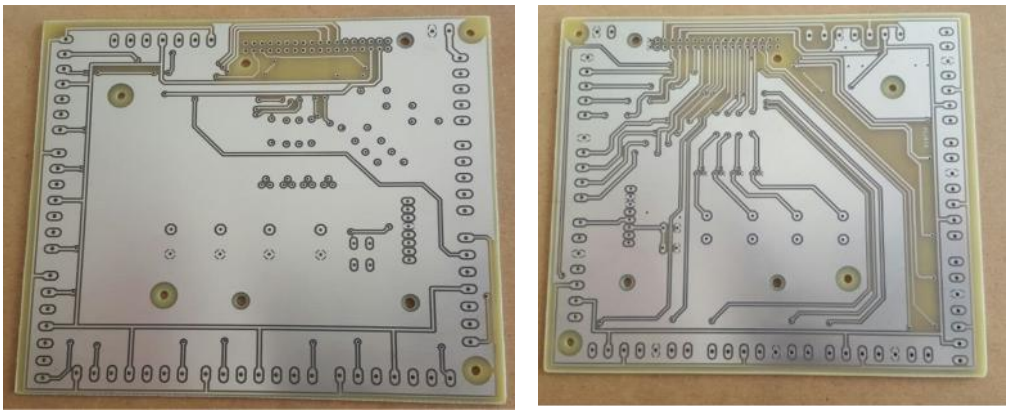


Figura 40. Esquema de placa madre en PCB
Fuente: Elaborado por el autor

La respectiva elaboración de la placa se lo realizo en una baquelita impresa a dos caras de 14 x 12 cm, así como se muestra en la Figura 41.



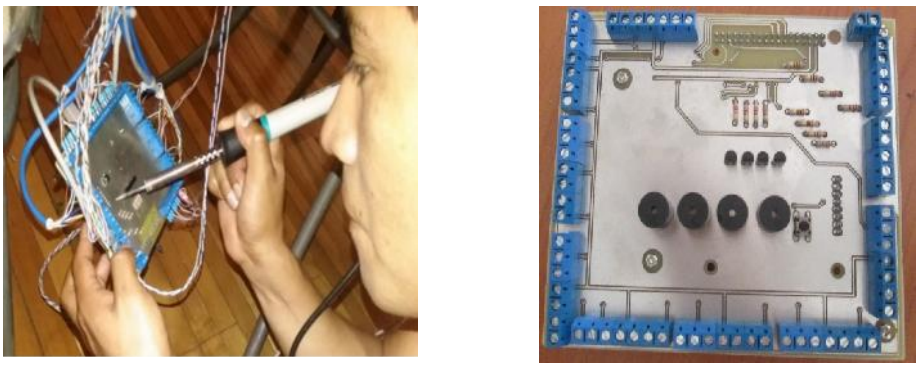
a) b)

Figura 41. Circuito impreso **a)** Placa baquelita cara 1 **b)** Placa baquelita cara 2
Fuente: Elaborado por el autor

3.2.5.1.1. Montaje de compontes electrónicos

La distribución de los elementos en la placa madre se montaron según la figura 42 y de la siguiente manera:

- Las borneras para los lectores RFID y las luces de los semáforos vehiculares y peatonales hacia los costados para tener una mejor manipulación en las conexiones.
- Transistores 2N2222
- Resistencia de 200 y 220 Ω
- Zumbadores



a) b)

Figura 42. Montaje de elementos **a)** Soldadura de elementos **b)** Placa con elementos
Fuente: Elaborado por el autor

3.2.5.2. Maqueta demostrativa

Para la elaboración de la maqueta se consideró una escala apropiada para optimizar las conexiones y que no existan conflictos con los elementos; por lo tanto, la maqueta que se utilizará para la exposición del funcionamiento del sistema inteligente es de tipo demostrativo en una escala de 1:100, donde se ajustarán las edificaciones y demás mobiliario.

De acuerdo a la figura 43 la base de la maqueta, se realizó con cartón WPC C.16 Reverso Café 85 x 120 cm.

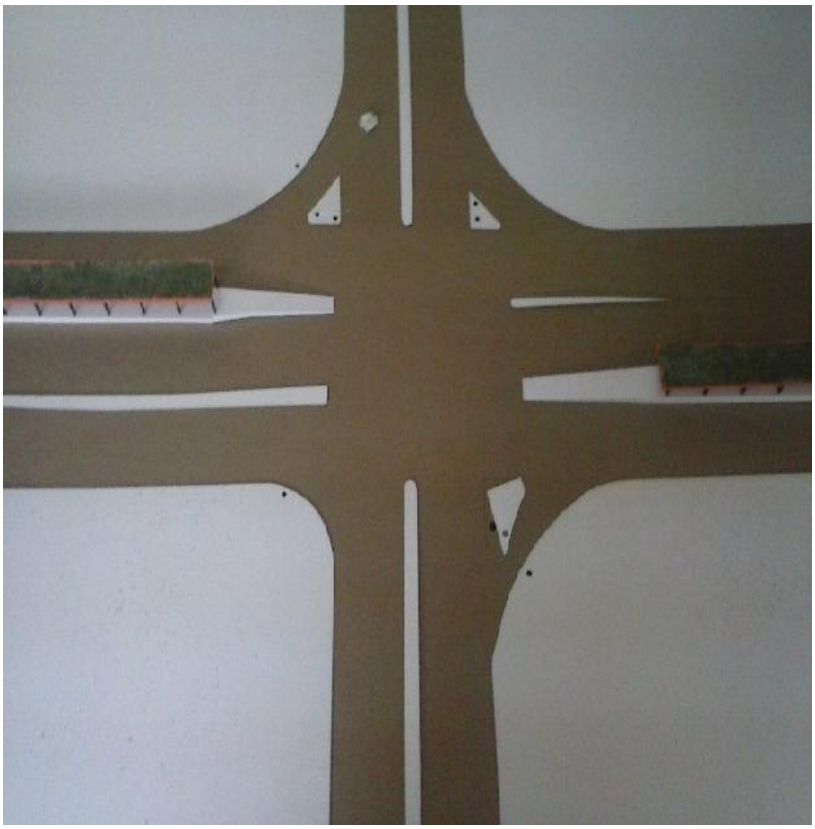
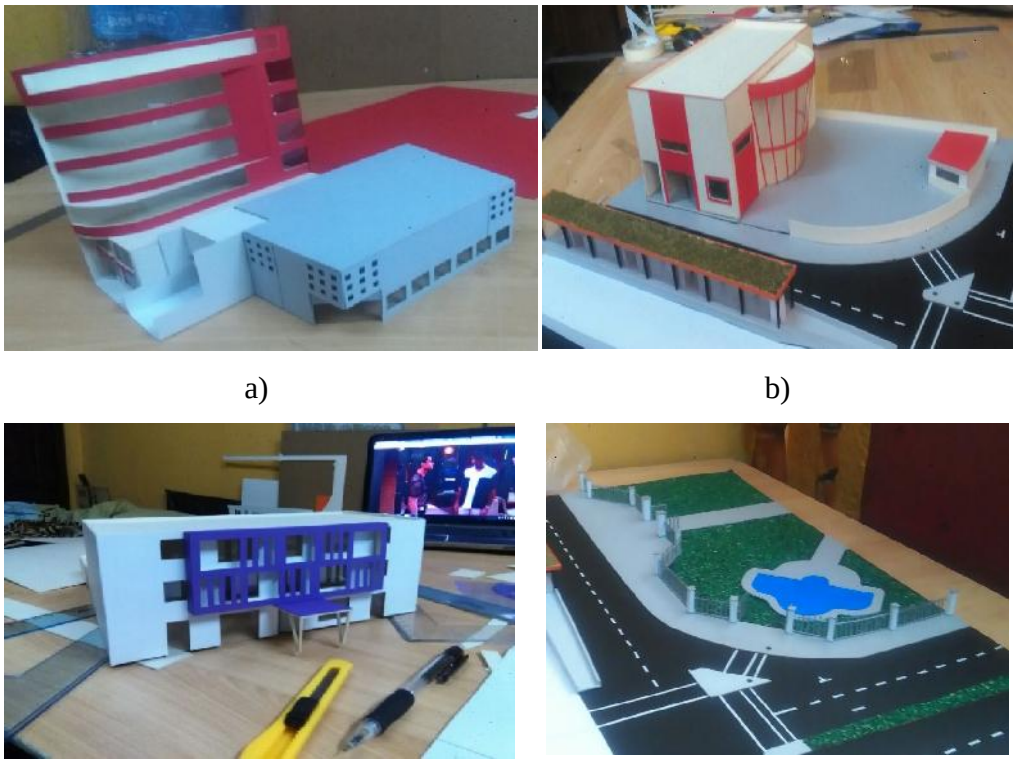


Figura 43. Elaboración de maqueta demostrativa
Fuente: Elaborado por el autor

Para las edificaciones se realizó en cartón paja como se muestra en la figura 44, donde se dio las formas correspondientes a los edificios y paradas de trolebús, para su decoración se utiliza papel contact de distintos colores, acrílicos para las ventas, papel tipo césped para el parque, parterre y estación, para la vía se utilizó cartulina negra y blanca.



a) b)

c) d)

Figura 44. Edificaciones de las intersecciones **a)** Edificio Payless, **b)** Edificio KFC, **c)** Edificio de Centro de Capacitación Politécnica, **d)** Jardín de la Circasiana

Fuente: Elaborado por el autor

Todos los componentes antes mencionados fueron incorporados entre sí a través de pegamento liquito UHU según como se muestra la Figura 45.

a) b)

Figura 45. Maqueta demostrativa **a)** Vista Superior **b)** Vista Diagonal

Fuente: Elaborado por el autor

3.2.5.3. Montaje de lectores RFID-RC522

Para el diseño se implementó ocho lectores RFID-RC522 que fueron ubicados en cada intersección de la vía según como se muestra en la figura 46, su instalación y fijación se lo realizo a través de pegamento líquido UHU.

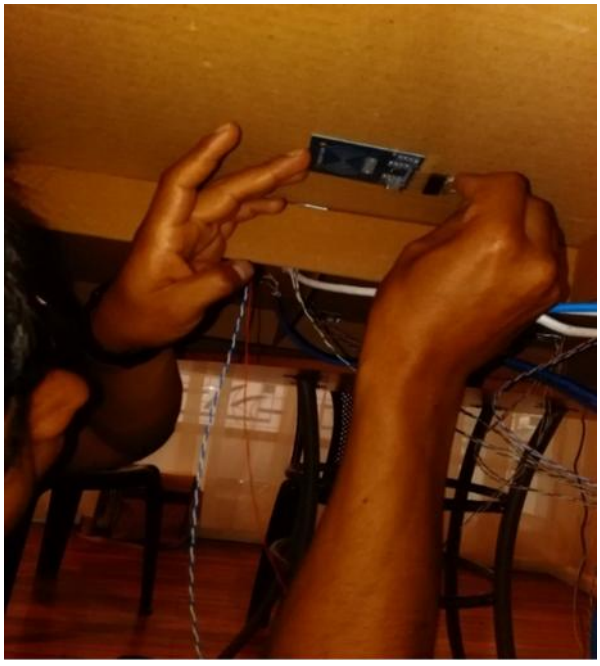
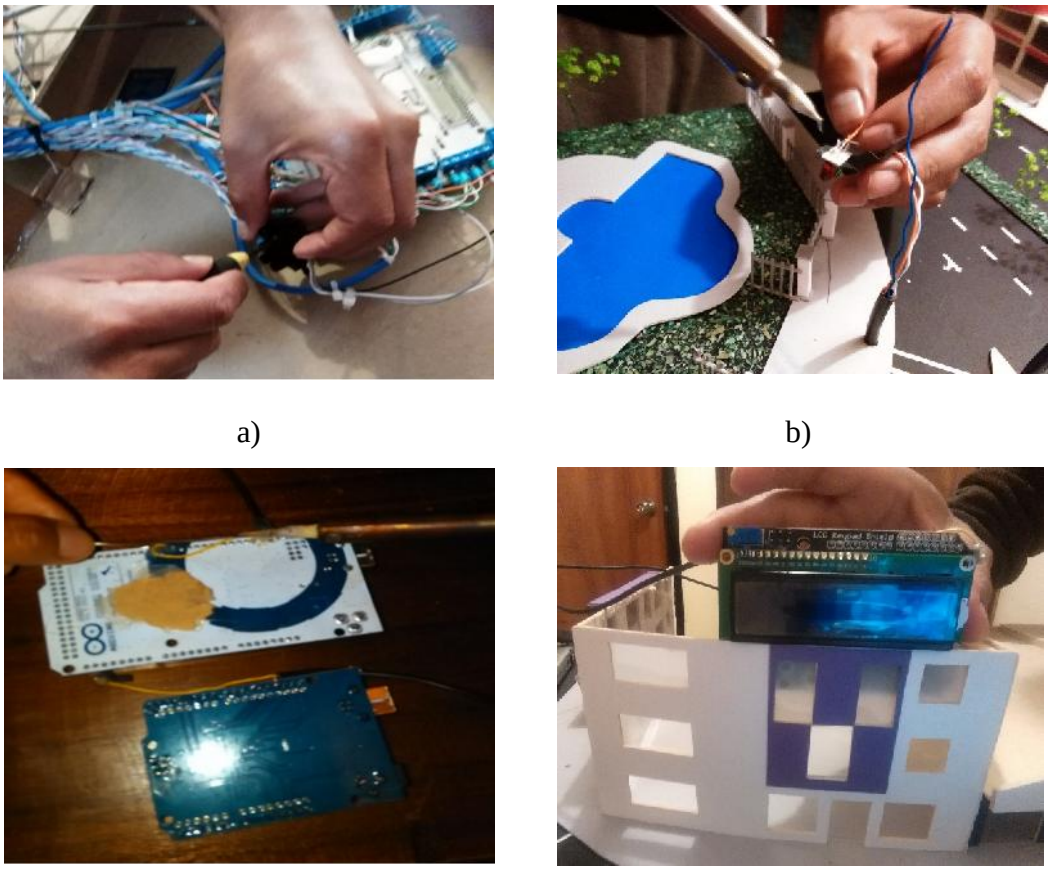


Figura 46. Montaje de Lectores
Fuente: Elaborado por el autor

3.2.5.4. Cableado de semáforos, lectores RFID-RC522 y Shield LCD

Para las conexiones respectivas de los sensores RFID, semáforos, buzzers y Shield LCD se utilizó 10 m. de cable convencional de red UTP (Unshielded Twisted Pair), ya que es cable multipar y lo cual facilitó llevar en una sola chaqueta las instalaciones de los RFID y semáforos según como se los muestra en la figura 47.



a) b)

c) d)

Figura 47. Conexión de cableado maqueta, a) Conexión cableado a placa, b) Soldado de semáforo, c) Soldado conexión Arduino, d) Ubicación de Shield LCD

Fuente: Elaborado por el autor

3.2.5.5. Programación

La respectiva programación se realizará en base al flujograma de la Figura 38; por lo tanto, su elaboración es de la siguiente manera:

- Declaro las librerías a utilizar que son: SPI, MFRC522 (Lectores de TAG), EEPROM.
- Definir los pines RST y SS para cada uno de los RFID.
- Declaro los pines del buzzers.
- Declaro los pines de los LED Verde, LED Amarillo, LED Rojo de los dos sentidos de los semáforos vehiculares.

- Declaro los pines de los LED Verde, LED Rojo de los semáforos peatonales.
- Se crea los objetos a usar el cual permitirá hacer un llamado a la librería y decirle con que pines van a trabajar los MFRC522.
- Se declaró las variables globales

  - ✓ Proceso 1 = 0 (Dará acceso al sentido norte – sur Horario Normal)
  - ✓ Contador = 0 (Resetea el contador)
  - ✓ Horario = 0 (Sistema Hora Normal)
  - ✓ Acceso = 0 (Presencia de Tag)
  - ✓ Acceso1 = 0 (Varia los valores cuando detecta una presencia de Tag)
  - ✓ a = 0 (Reestablece la variable para que el contador no quede En un ciclo infinito)

  - ✓ Leíble = 0
  - ✓ b = 0
  - ✓ c = 0

- Se define las constante de alcance global, donde se registrará el ID de los tag que serán los usuarios del sistema.
- Se insertó variables para definir los tiempos fijos en cada semáforo, para el sistema de hora normal se indicó las variables int tiempo 1ax hasta el 1dx será la intersección SUR – NORTE y del 1ex hasta el 1hx será la intersección ESTE – OESTE y para el sistema de hora pico se indicó la variable int tiempo 2ax hasta el 2dx que será la intercesión SUR – NORTE y del 2ex hasta el 2hx será la intersección ESTE – OESTE.
- Cuando el sistema detecte un tag, el programa variará los tiempos de los semáforos mediante las siguientes variables: int tiempo 1a hasta el 1h para la hora normal y del 2a hasta el 2h para la hora pico. Con estas variables el tiempo fijo no cambiará lo que hará es aumentar 15 segundos más sin afectar al tiempo ya establecido y así dará prioridad a las personas con discapacidad.
- Parametrización del menú de configuración con el void setup () donde se iniciará el arranque de los puertos seriales, SPI y ejecución cada uno de los lectores.

- Definición de puertos de salida para Buzzers, LED de semáforos y LED de peatones.
- Realizo un serial println para imprimir los datos a través del puerto serial
- Leo la memoria EEPROM, donde me indicará si es hora normal u hora pico, si HORARIO vale 1, la memoria EEPROM escribe (0,1) donde ejecutará hora pico y dará una alerta de 10 pitidos de 50 ms; pero si HORARIO es diferente de 1, la memoria EEPROM escribirá (0,0) y ejecutará hora normal y dará una alerta de 5 pitidos de 500 ms.
- Retardo de 2 s para que el programa se ejecute.
- La variable void loop() se realiza de forma repetitiva
- Si HORARIO == 0 me permitirá ejecutar únicamente mi horario normal
- El CONTADOR ++, la cual se empleará para ejecutar el conteo del tiempo el cual permitirá incrementar 1s a la vez, luego permite imprimir CONTEO para realizar una pregunta si el PROCESO1==0 solo se permitirá el paso de vehículos en sentido NORTE – SUR y viceversa.
- Imprimo y ejecuto el proceso de Horario Normal.
- Si CONTADOR es < a mi variable tiempo1a, este encenderá la luz verde por 10 segundos.
- Si CONTADOR es >= tiempo1b y CONTADOR < tiempo1c la luz amarilla se encenderá por 3 s.
- Si CONTADOR >= tiempo 1d, ha cumplido su tiempo este cambiará a rojo.
- Si ACCESO==1, si se ha presenciado un tag la variable de acceso no dejará el ingreso a ninguna otra sentencia.
- Si el PROCESO1 = 1 permitirá el paso vehicular en sentido ESTE – OESTE y viceversa.
- Cuando ACCESO1=1, se ha detectado un tag, y ACCESO1=2, en esta condición el tiempo se incrementará en el contador con el fin de alargar el periodo del proceso normal del cruce.
- Para la visualizar el contador en el LCD se incluyó la librería LiquidCrystal.

3.2.5.6. Pruebas de funcionamiento

Para probar el funcionamiento de todo el sistema, se lo dividió en secciones y de esta manera obtener un resultado de acuerdo a los parámetros establecidos para dichas secciones.

3.2.5.6.1. Funcionamiento del sistema de semaforización.

De acuerdo a la tabla 5 se comprueba las conexiones de las luces de los semáforos.

Tabla 5. Comprobación de conexión de semáforos

INTERSECCIÓN	**LED ENCENDIDOS**					
	VERDE		**AMARILLO**		**ROJO**	
	SI	**NO**	**SI**	**NO**	**SI**	**NO**
Sur – Norte	✓		✓		✓	
Norte – Sur		✓	✓		✓	
Oeste – Este	✓		✓		✓	
Este – Oeste	✓			✓		✓

Fuente: Elaborado por el autor

Las luces que no se encendieron fueron por un problema de conexión en el cableado. Según la tabla 6 los inconvenientes con las luces fueron solventados.

Tabla 6. Comprobación de conexión de semáforos

INTERSECCIÓN	LED ENCENDIDOS					
	VERDE		**AMARILLO**		**ROJO**	
	SI	**NO**	**SI**	**NO**	**SI**	**NO**
Sur – Norte	✓		✓		✓	
Norte – Sur	✓		✓		✓	
Oeste – Este	✓		✓		✓	
Este – Oeste	✓		✓		✓	

Fuente: Elaborado por el autor

3.2.5.6.2. Funcionamiento pantalla LCD

De acuerdo a la tabla 7 se comprueba las conexiones del Shield LCD.

Tabla 7. Comprobación de conexión Shield LCD

Shield LCD	SI	NO
Encendió Pantalla		✓

Fuente: Elaborado por el autor

El Shield LCD no se encendió debido a un problema de conexión de cables así el Arduino Mega2560. Según la tabla 8 el inconveniente con el Shield LCD fue solventado.

Tabla 8. Comprobación de conexión Shield LCD

Shield LCD	SI	NO
Encendió Pantalla	✓	

Fuente: Elaborado por el autor

3.2.5.6.3. Funcionamiento de sensores RFID

Como se muestra en la Tabla 9, el 62,5% de los sensores RFID presentaron fallas al momento de la prueba de operación.

Tabla 9. Comprobación sensores RFID

SENSORES	DETECTO	
	SI	**NO**
Tag 1	✓	
Tag 2		✓
Tag 3		✓
Tag 4		✓
Tag 5	✓	
Tag 6		✓
Tag 7		✓
Tag 8	✓	

Fuente: Elaborado por el autor

Los inconvenientes fueron por la ubicación de los sensores (espesor del material de la base de la maqueta) y falsos contactos entre los pines macho y hembra de los conectores, lo cual se solvento con la reubicación de los módulos y ajuste de los terminales. Con estas acciones correctivas el funcionamiento de los sensores fue satisfactorio como se lo evidencia en la tabla 10.

Tabla 10. Comprobación sensores RFID

SENSORES	DETECTO	
	SI	**NO**
Tag 1	✓	
Tag 2	✓	
Tag 3	✓	
Tag 4	✓	
Tag 5	✓	
Tag 6	✓	
Tag 7	✓	
Tag 8	✓	

Fuente: Elaborado por el autor

3.2.5.6.4. Interacción de los sensores RFID con el sistema de semaforización.

De acuerdo a la Tabla11, todos los sensores interactúan con los semáforos pero intervienen en su operación normal, finalizan el contador y dan paso a la siguiente intersección. Ninguno de los Tags incrementa tiempo en el cruce peatonal.

Tabla 11. Interacción de sensores RFID con los semáforos

SENSORES	**SENTIDO ESTE – OESTE**				**SENTIDO NORTE – SUR**			
	Interactúa con los semáforos		Incrementa tiempo		Interactúa con los semáforos		Incrementa tiempo	
	SI	NO	SI	NO	SI	NO	SI	NO
TAG 1,2,5,6	✓			✓	✓			✓
TAG 3,4,7,8	✓			✓	✓			✓

Fuente: Elaborado por el autor

La interrupción en el proceso de funcionamiento de los semáforos y el no incremento en el tiempo del cruce peatonal, se debieron a un conflicto en la declaración de variables del programa; lo cual se resolvió con la declaración de variables fijas y variables que fluctúen el tiempo del contador. Como muestra en la tabla 12 la interacción y el incremento de tiempo se realizan de acuerdo a los requerimientos del cruce peatonal para personas con discapacidad sin alterar el proceso de funcionamiento de los semáforos.

Tabla 12. Interacción de sensores RFID con los semáforos

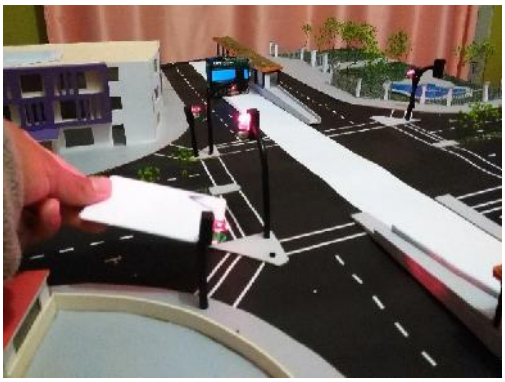
SENSORES	**SENTIDO ESTE – OESTE**				**SENTIDO NORTE – SUR**			
	Interactúa con los semáforos		Incrementa tiempo		Interactúa con los semáforos		Incrementa tiempo	
	SI	NO	SI	NO	SI	NO	SI	NO
TAG 1,2,5,6	✓		✓		✓		✓	
TAG 3,4,7,8	✓		✓		✓		✓	

Fuente: Elaborado por el autor

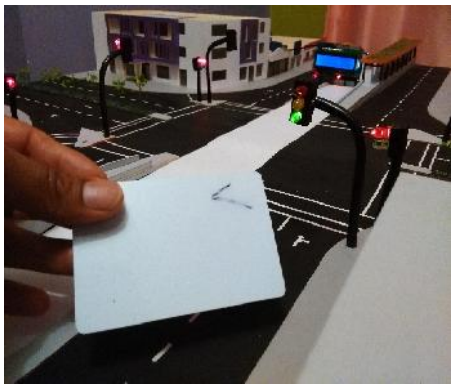
El sistema de semaforización de la maqueta demostrativa se desarrolló en escenarios, los cuales son contemplados en hora pico y hora normal.

Cada vez que el dispositivo se apague y encienda existirá una variación de velocidad en el sonido de los buzzers; si el pitido es rápido indicará que el programa se ejecutará en la opción de hora pico, y si el pitido es lento se asumirá que es hora normal.

Según como se muestra en la figura 48, el sistema reconocerá solo tags registrados en el software para realizar el incremento del tiempo en el contador, pero si un tags no está registrado este emitirá un sonido de negación y no adicionará tiempo por lo tanto seguirá su operación normal.



a)



b)

c)

Figura 48. Pruebas de Funcionamiento a) Prueba en semáforo incorrecto b) Prueba en semáforo en verde c) Prueba de funcionamiento Pantalla LCD

Fuente: Elaborado por el autor

3.2.5.7. Análisis Económico

En función de la factibilidad técnica y operativa se determinó que los recursos financieros que se utilizaron para la implementación el sistema inteligente serán de acuerdo a la Tabla 13, en la cual se detalla cantidades, costo unitario y costo total de cada uno de los elementos que intervendrán en el proceso de elaboración del prototipo.

Tabla 13. Presupuesto de prototipo

ITEM	CANT.	DESCRIPCIÓN	COSTO UNITARIO	COSTO TOTAL
1	1	Arduino Mega	$ 30,00	$ 30,00
1	1	Arduino Uno	$ 13,00	$ 13,00
1	1	Shield LCD	$ 16,00	$ 16,00
2	8	Lectores RFID – RC522	$ 10,25	$ 82,00
3	4	Buzzers	$ 1,00	$ 4,00
4	4	Transistor 2N2222	$ 0,10	$ 0,40
5	20	Resistencia 200 y 220 ohm	$ 0,14	$ 2,64
6	4	Header Macho	$ 0,55	$ 2,20
7	1	Transformador	$ 15,00	$ 15,00
8	4	Postes	$ 1,00	$ 4,00
9	9	Bornera Triple	$ 0,30	$ 2,70
10	24	Bornera Doble	$ 0,20	$ 4,80
11	3	Cable UTP	$ 0,60	$ 1,80
12	1	Placa Madre Doble cara	$ 48,00	$ 48,00
13	1	Maqueta demostrativa	$ 150	$ 150
		TOTAL COSTOS		**$ 376,54**

Fuente: Elaborado por el autor

La propuesta se enmarca dentro de un proyecto económico factible, ya que el prototipo satisface las necesidades requeridas y cumple la función para la cual fue diseñado.

En la práctica el presupuesto se alterará notablemente, ya que los componentes que se utilizarán son de tipo industrial para operar en un régimen de 24 / 7 y en condiciones ambientales extremas.

Los costos de implementación en una intersección del Distrito Metropolitano de Quito tendrán un valor aproximado según como se detalla en la tabla 14.

Tabla 14. Presupuesto implementación real del sistema inteligente

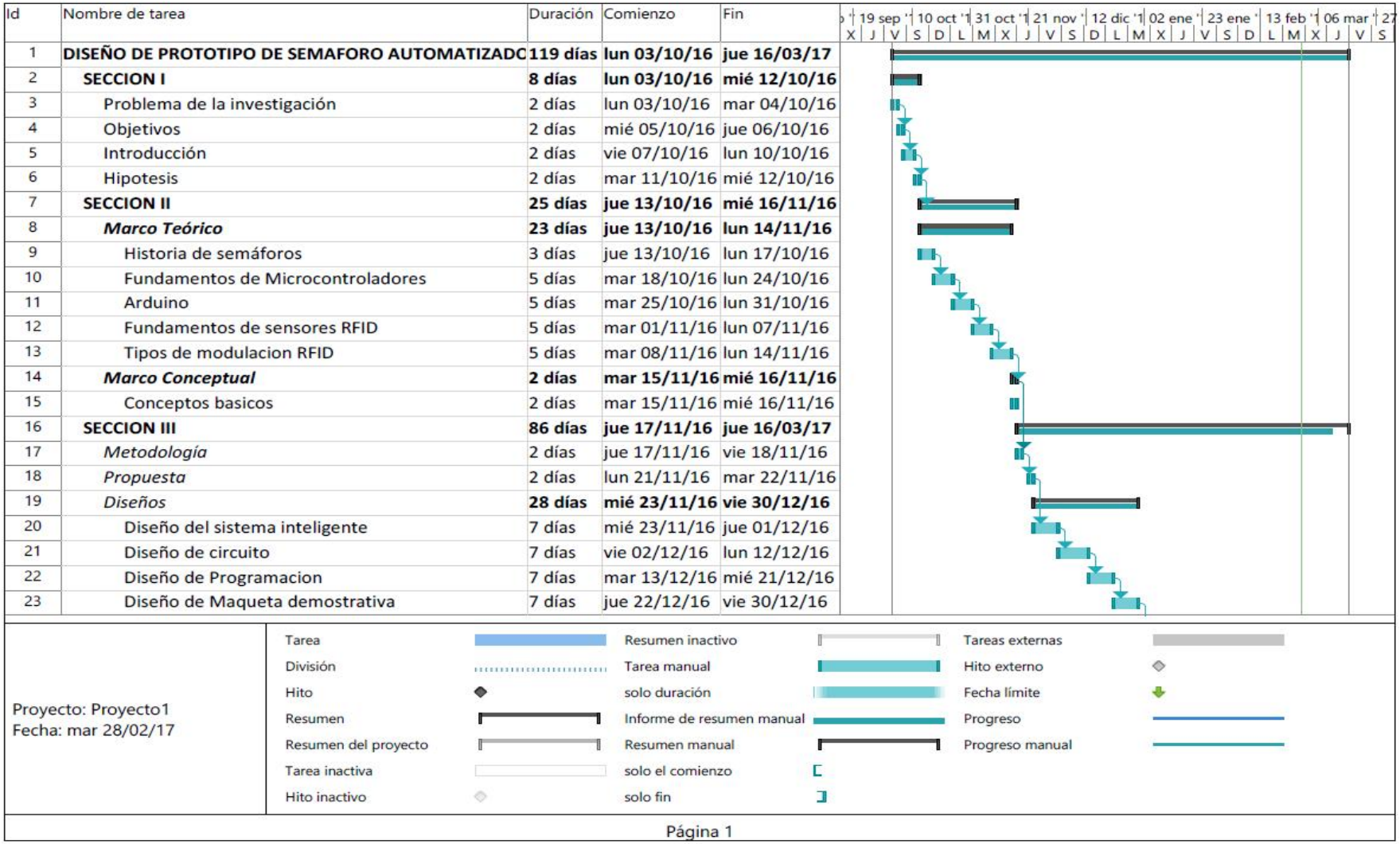
ITEM	CANT.	DESCRIPCIÓN	COSTO UNITARIO	COSTO TOTAL
1	1	Simatic S7-1500 Digital Imput/output	$ 2.500	$ 2.500,00
2	8	SIMATIC RF1000	$ 1.200	$ 9.600,00
3	1	Fuente de voltaje	$ 250,00	$ 250,00
4	1	Caja de conexiones 60x60x30	$ 450,00	$ 450,00
5	4	Cable Profibus Siemens Simatic Net Precio Por 100 m	$ 550,41	$2201,63
6	3	Cable Flexible 100 m #12	$ 40,00	$ 120,00
7	4	Rieles DIM	$ 8,00	$ 32,00
8	33	Bornera tipo DIM	$ 1,50	$ 49,50
9	1	Breaker 10 A	$ 15,00	$ 15,00
10	240 H/H	Programación	$ 12,50	$ 3.000,00
		TOTAL COSTOS		**$ 18.218,13**

Fuente: Elaborado por el autor

Los valores unitarios de los elementos electrónicos y accesorios son referenciales ya que podrán variar de acuerdo al fabricante, marca, desempeño y protección IP.

3.3. Cronograma

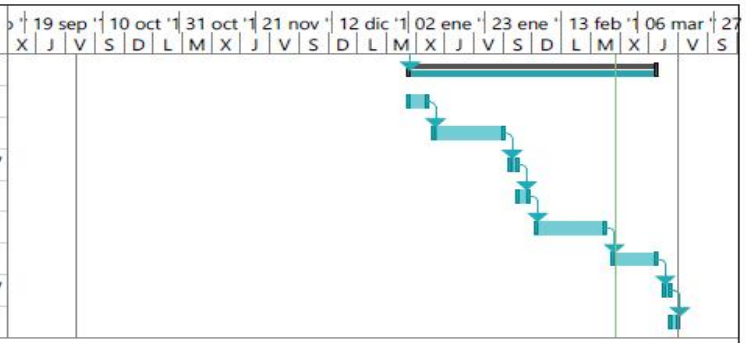
De acuerdo a la figura 49 y 50 se muestra el cronograma de actividades desde su inicio hasta su culminación.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**DISEÑO DE PROTOTIPO DE SEMAFORO AUTOMATIZADC**	**119 días**	**lun 03/10/16**	**jue 16/03/17**
2	**SECCION I**	**8 días**	**lun 03/10/16**	**mié 12/10/16**
3	Problema de la investigación	2 días	lun 03/10/16	mar 04/10/16
4	Objetivos	2 días	mié 05/10/16	jue 06/10/16
5	Introducción	2 días	vie 07/10/16	lun 10/10/16
6	Hipotesis	2 días	mar 11/10/16	mié 12/10/16
7	**SECCION II**	**25 días**	**jue 13/10/16**	**mié 16/11/16**
8	***Marco Teórico***	**23 días**	**jue 13/10/16**	**lun 14/11/16**
9	Historia de semáforos	3 días	jue 13/10/16	lun 17/10/16
10	Fundamentos de Microcontroladores	5 días	mar 18/10/16	lun 24/10/16
11	Arduino	5 días	mar 25/10/16	lun 31/10/16
12	Fundamentos de sensores RFID	5 días	mar 01/11/16	lun 07/11/16
13	Tipos de modulacion RFID	5 días	mar 08/11/16	lun 14/11/16
14	***Marco Conceptual***	**2 días**	**mar 15/11/16**	**mié 16/11/16**
15	Conceptos basicos	2 días	mar 15/11/16	mié 16/11/16
16	**SECCION III**	**86 días**	**jue 17/11/16**	**jue 16/03/17**
17	*Metodología*	2 días	jue 17/11/16	vie 18/11/16
18	*Propuesta*	2 días	lun 21/11/16	mar 22/11/16
19	*Diseños*	**28 días**	**mié 23/11/16**	**vie 30/12/16**
20	Diseño del sistema inteligente	7 días	mié 23/11/16	jue 01/12/16
21	Diseño de circuito	7 días	vie 02/12/16	lun 12/12/16
22	Diseño de Programacion	7 días	mar 13/12/16	mié 21/12/16
23	Diseño de Maqueta demostrativa	7 días	jue 22/12/16	vie 30/12/16

Figura 49. Cronograma de actividades Página 1
Fuente: Elaborado por el Autor

Id	Nombre de tarea	Duración	Comienzo	Fin
24	*Implementacion*	**50 días**	**lun 02/01/17**	**vie 10/03/17**
25	Montaje de componentes placa madre	5 días	lun 02/01/17	vie 06/01/17
26	Maqueta demostrativa	15 días	lun 09/01/17	vie 27/01/17
27	Montaje de sensores RFID	2 días	lun 30/01/17	mar 31/01/17
28	Cableado semaforos y sensores	3 días	mié 01/02/17	vie 03/02/17
29	Programacion Arduino	15 días	lun 06/02/17	vie 24/02/17
30	Pruebas de funcionamiento	10 días	lun 27/02/17	vie 10/03/17
31	*Conclusiones*	**2 días**	lun 13/03/17	mar 14/03/17
32	*Recomendaciones*	**2 días**	mié 15/03/17	jue 16/03/17



Proyecto: Proyecto1
Fecha: mar 28/02/17

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual

Página 2

Figura 50. Cronograma de actividades página 2
Fuente: Elaborada por el autor

SECCION IV

4.1. Conclusiones

- De acuerdo al levantamiento de información se realizó el análisis de factibilidad de las distintas intersecciones del Distrito Metropolitano de Quito, el sistema de semaforización peatonal no está adaptado para la interacción de personas con discapacidad; por lo cual se concluye, la necesidad de instaurar un sistema inteligente para los cruces peatonales, con el fin de incluir a este grupo vulnerable de la sociedad.
- Según los fundamentos teóricos, el sistema inteligente de cruce peatonal, puede tener varias alternativas de tecnologías como son: PIC, AVR, PLC y ARDUINO; no obstante, la plataforma Arduino se ajusta al proyecto por su versatilidad en programación, configuración y conexión, para interactuar con sensores inalámbricos como Bluetooth, infrarrojo y RFID; dentro de estos la identificación por radio frecuencia cumple con los requerimientos para interactuar con la plataforma microcontroladora a través de sus tags.
- El diseño del sistema inteligente de cruce peatonal, está basado en el funcionamiento de la estructura del microcontrolador Arduino Mega 2560 con la comunicación de los sensores RFID de corto alcance (HF), que permite una interacción puntual entre el usuario a través de un protocolo de comunicación SPI.
- La información generada del sistema inteligente de cruce peatonal no afecta el funcionamiento de la plataforma principal de semaforización, ya que el lenguaje de programación permitió desarrollar una estructura jerárquica, de lo contrario; el sistema adaptará sus tiempos de cruce peatonal, cuando haya detectado a una persona con discapacidad que posea un tag registrado, y una vez finalizado regresará a su programación principal.
- La implementación del prototipo de sistema inteligente a través de una maqueta demostrativa, generó una perspectiva clara y concisa de la importancia de dotar

de sistemas auxiliares que permitan interactuar con mayor seguridad y confianza a personas con algún tipo de discapacidad.

- Las pruebas realizadas sobre el prototipo fueron satisfactorias, sin embargo dentro del proceso se tuvieron que realizar modificaciones como; la ubicación de los lectores RFID, ajustes de terminales de los sensores y definición de constantes y cambiables dentro de la programación.

4.2. Recomendaciones

- Para mejorar la autonomía del sistema inteligente se recomienda desarrollar un Gestor de base de datos, que será alimentado con la información que diariamente se generará en el sistema con información como la hora, fecha, ubicación y datos relevantes de los usuarios.
- Además se podrá implementar protocolos de comunicación TCP/IP, para monitorear el sistema en tiempo real, con el fin de detectar anomalías en el proceso de funcionamiento y solventarlas en menor tiempo, con lo cual se garantizará la disponibilidad de la plataforma con un alto grado de confiabilidad.
- Con la implementación de sensores adicionales como: velocidad, identificación de vehículos de emergencia y de proximidad, se podrá mejorar la funcionalidad del sistema a través de su programación en la estructura existente, esto es un aspecto muy importante, ya que la plataforma se volverá más robusta al brindar mayores beneficios a la sociedad.
- En lo que respecta a los tags se podrán reemplazar por sensores de medio alcance (tags UHF), para generar mayor cobertura y sustituir la activación puntual, de esta manera se generará una interacción más dinámica entre el sistema y el usuario final.

4.3. Bibliografía

Armau, V., y Orduña, J., (1996). ARQUITECTURA Y PROGRAMACION DE MICROCONTROLADORES.

Avendaño, J., y Clavijo, G., (2012) DISEÑO DE LA RED SEMAFÓRICA DE LA CALLE MARISCAL LAMAR DESDE LA CALLE MANUEL VEGA HASTA LA CALLE TARQUI. Tesis Ingeniería. Universidad de Cuenca, Ecuador. Recuperado de http://dspace.ucuenca.edu.ec/bitstream/123456789/778/1/ti904.pdf

Geddes M., (2016). Arduino Project Handbook

Goilav, N., y Loi, G., (2016). Arduino Aprender a desarrollar para crear objetos inteligentes. Recuperado de https://books.google.com.ec/books?id=R6RCxQl_H6YC&pg=PA37&dq=arduino&hl=es-419&sa=X&ved=0ahUKEwiyiemij8_QAhUHziYKHeXzAbEQ6AEIMzAC#v=onepage&q=arduino&f=false

Hughes, M., (2016), Arduino a Technical Reference

Mandado, E., Menéndez, L.M., Fernández, L., y López, E. (2007). Microcontroladores PIC. Sistema integrado para el autoaprendizaje.

Muro, L., (2007). Arquitectura de microprocesadores RISC y CISC.

Urbina, R., (2011) Tutorial sobre circuitos RFID. Tesis Licenciatura. Universidad de las Américas, Puebla. Recuperado de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/urbina_r_rd/indice.html

4.3.1. Referencias Bibliográficas

Fernández Y. y Rodríguez G., (2013). “Cálculo de los tiempos de semaforización”.

Finkenzeller, K (2010): “RFID Handbook”.

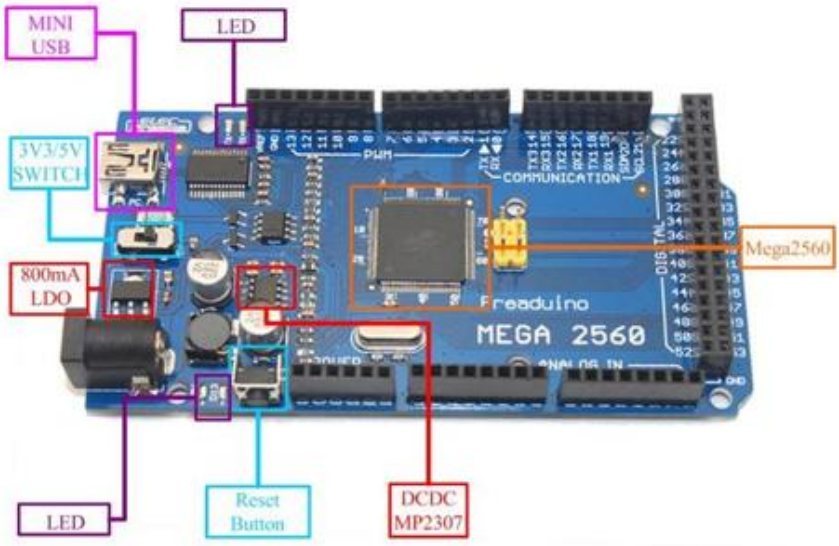
Torrente, O., (2013). ARDUINO Curso práctico de formación. Recuperado de https://books.google.com.ec/books?id=6cZhDmf7suQC&pg=PA78&dq=arquitectura+de+un+microcontrolador&hl=es&sa=X&ved=0ahUKEwjFkPWE5KTQAhWGbiYKHWOOCEgQ6AEIUzAJ#v=onepage&q=arquitectura%20de%20un%20microcontrolador&f=false

Valdés, F., y Pallàs, R., (2007). MICROCONTROLADORES: FUNDAMENTOS Y APLICACIONES CON PIC.

Verle M., (2009). “Microcontroladores PIC”.

4.4. Anexos

ARDUINO MEGA 2560



VISIÓN DE CONJUNTO

El Arduino Mega 2560 es una placa microcontroladora basada en el ATmega2560 (hoja de datos). Cuenta con 54 pines de entrada / salida digital (de los cuales 14 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, Y un botón de reinicio. Contiene todo lo necesario para soportar el microcontrolador; Simplemente conéctelo a un ordenador con un cable USB o conéctelo con un adaptador ACto-DC o una batería para empezar. El Mega es compatible con la mayoría de los escudos diseñados para el Arduino Duemilanove o Diecimila.

ESQUEMA Y DISEÑO DE REFERENCIA

Archivos EAGLE: arduino-mega2560-reference-design.zip

Esquema: arduino-mega2560-schematic.pdf

RESUMEN

Microcontrolador	ATmega2560
Tensión de funcionamiento	5V
Tensión de entrada (recomendado	7-12V
Tensión de entrada (límites)	6-20V
Pines de E / S digitales	54 (de los cuales 14 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente CC por pin de E / S	40 mA
Corriente CC para 3.3V Pin	50 mA

Poder

El Arduino Mega puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.

La alimentación externa (no USB) puede venir desde un adaptador AC-DC (verrugas de pared) o una batería. El adaptador puede conectarse enchufando un conector positivo de centro de 2,1 mm en el conector de alimentación de la tarjeta. Las derivaciones de una batería se pueden insertar en los conectores de clavijas Gnd y Vin del conector POWER.

La placa puede funcionar con un suministro externo de 6 a 20 voltios. Si se suministra con menos de 7V, sin embargo, el pin 5V puede suministrar menos de cinco voltios y la placa puede ser inestable.

Si utiliza más de 12V, el regulador de tensión puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios.

El Mega2560 se diferencia de todos los tableros anteriores en que no utiliza el chip driver FTDI USB-toserial. En su lugar, cuenta con el Atmega8U2 programado como un convertidor USB a serie.

Los pines de alimentación son los siguientes:

● VIN. El voltaje de entrada a la tarjeta Arduino cuando se utiliza una fuente de alimentación externa (a diferencia de 5 voltios de la conexión USB u otra fuente de alimentación regulada). Usted puede suministrar voltaje a través de este pin, o, si el suministro de voltaje a través de la toma de alimentación, el acceso a través de este pin.

● 5V. La fuente de alimentación regulada utiliza para alimentar el microcontrolador y otros componentes de la placa. Esto puede venir ya sea de VIN a través de un regulador de a bordo, o ser suministrado por USB u otro suministro regulado de 5V.

● 3V3. Una fuente de 3.3 voltios generada por el regulador de a bordo. El consumo máximo de corriente es de 50 mA.

● GND. Pasadores de tierra.

MEMORIA

El ATmega2560 tiene 256 KB de memoria flash para almacenar código (de los cuales 8 KB se utiliza para el gestor de arranque), 8 KB de SRAM y 4 KB de EEPROM (que se pueden leer y escribir con la librería EEPROM).

ENTRADA Y SALIDA

Cada uno de los 54 pines digitales de Mega puede ser utilizado como entrada o salida, usando las funciones pinMode (), digitalWrite () y digitalRead (). Trabajan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia pull-up interna (desconectada por defecto) de 20-50 kOhms. Además, algunos pins tienen funciones especializadas:

● Serial: 0 (RX) y 1 (TX); Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Se utiliza para recibir (RX) y transmitir (TX) datos en serie TTL. Los pins 0 y 1 también están conectados a los pines correspondientes en el chip ATmega8U2 USB-to-TTL Serial.

● Interrupciones externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3) y 21 (interrupción 2). Estos pines pueden configurarse para activar una interrupción a un valor bajo, un flanco ascendente o descendente o un cambio de valor. Vea la función attachInterrupt () para más detalles.

● PWM: 0 a 13. Proporcione salida PWM de 8 bits con la función analogWrite ().

● SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines admiten comunicación SPI utilizando la biblioteca SPI. Los pines SPI también se dividen en el ICSP

, Que es físicamente compatible con el Uno, Duemilanove y Diecimila.

● LED: 13. Hay un LED incorporado conectado a la clavija digital 13. Cuando el pin es valor ALTO, el LED está encendido, cuando el pasador está en BAJO, está apagado.

● I2C: 20 (SDA) y 21 (SCL). Soporte para comunicación I2C (TWI) vía cable

Biblioteca (documentación en el sitio web de Wired). Tenga en cuenta que estos pines no están en el mismo lugar que los tornillos I2C en el Duemilanove o Diecimila.

El Mega2560 tiene 16 entradas analógicas, cada una de las cuales proporciona 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto miden desde tierra hasta 5 voltios, aunque es posible cambiar el extremo superior de su rango usando la función de alfiler AREF y analogReference ().

Hay un par de otros alfileres en el tablero:

aunque es posible cambiar el extremo superior de su rango usando la función de alfiler AREF y analogReference ().

Hay un par de otros alfileres en el tablero:

● AREF. Tensión de referencia para las entradas analógicas. Se utiliza con analogReference ().

● Restablecer. Lleve esta línea LOW para reiniciar el microcontrolador. Se utiliza normalmente para agregar un botón de reinicio a los escudos que bloquean el de la placa.

COMUNICACIÓN

El Arduino Mega2560 tiene una serie de instalaciones para comunicarse con una computadora, otro Arduino, u otros microcontroladores. El ATmega2560 proporciona cuatro UARTs de hardware para la comunicación en serie TTL (5V). Un ATmega8U2 en la placa canaliza una de ellas a través de USB y proporciona un puerto virtual a software en la computadora (las máquinas con Windows necesitarán un archivo .inf, pero las máquinas OSX y Linux reconocerán la placa como un puerto COM automáticamente. Incluye un monitor en serie que permite que se envíen datos textuales sencillos hacia y desde la placa Los LED RX y TX de la placa parpadearán cuando se transmitan datos a través del chip

ATmega8U2 y la conexión USB al ordenador (pero no para la comunicación serie en Pines 0 y 1).

Una biblioteca de SoftwareSerial permite la comunicación en serie en cualquiera de los pines digitales del Mega2560.

El ATmega2560 también soporta comunicación I2C (TWI) y SPI. El software Arduino incluye una biblioteca de cables para simplificar el uso del bus I2C; Consulte la documentación en el sitio web de Cableado para obtener más detalles. Para la comunicación SPI, utilice la biblioteca SPI.

PROGRAMACIÓN

El Arduino Mega se puede programar con el software Arduino (descarga). Para obtener más información, consulte la referencia y los tutoriales.

El ATmega2560 en el Arduino Mega viene pre-quemado con un cargador de arranque que le permite cargar un nuevo código sin el uso de un programador de hardware externo. Se comunica utilizando el protocolo STK500 original (referencia, archivos de cabecera C).

También puede omitir el cargador de arranque y programar el microcontrolador a través de la cabecera ICSP (InCircuit Serial Programming); Vea estas instrucciones para más detalles.

RESTABLECIMIENTO AUTOMÁTICO (SOFTWARE)

En lugar de requerir una pulsación física del botón de reinicio antes de una carga, el Arduino Mega2560 está diseñado de tal forma que permite su reinicio mediante el software que se ejecuta en una computadora conectada. Una de las líneas de control de flujo de hardware (DTR) del ATmega8U2 está conectada a la línea de reposición del ATmega2560 a través de un condensador de 100 nanofarad. Cuando esta línea se afirma (tomada baja), la línea de reinicio se cae el tiempo suficiente para restablecer el chip. El software Arduino utiliza esta función para poder cargar código simplemente pulsando el botón de carga en el entorno Arduino. Esto significa que el gestor de arranque puede tener un tiempo de espera más corto, ya que la reducción de DTR puede ser bien coordinada con el inicio de la carga.

Esta configuración tiene otras implicaciones. Cuando el Mega2560 está conectado a un ordenador que ejecuta Mac OS X o Linux, se restablece cada vez que se realiza una conexión desde el software (a través de USB). Durante el siguiente medio segundo aproximadamente, el cargador de arranque se está ejecutando en el Mega2560. Mientras está programado para ignorar datos malformados (es decir, cualquier cosa además de una carga de nuevo código), interceptará los primeros pocos bytes de datos enviados a la placa después de que se abra una conexión.

Si un boceto que se ejecuta en la placa recibe una configuración única u otros datos cuando se inicia por primera vez, asegúrese de que el software con el que se comunica espera un segundo después de abrir la conexión y antes de enviar estos datos.

El Mega2560 contiene una traza que se puede cortar para deshabilitar el auto-reset. Las almohadillas a ambos lados de la traza se pueden soldar juntas para volver a habilitarlo. Tiene la etiqueta "RESET-EN". También puede desactivar el restablecimiento automático al conectar una resistencia de 110 ohmios de 5V a la línea de reinicio; Vea este hilo del foro para más detalles.

Protección de sobreintensidad USB

El Arduino Mega2560 tiene un polyfuse reiniciable que protege el USB de su computadora, puertos de cortocircuitos y sobrecorriente. Aunque la mayoría de las computadoras proporcionan su propia protección interna, el fusible proporciona una capa adicional de protección. Si se aplica más de 500 mA al puerto USB, el fusible romperá automáticamente la conexión hasta que se quite el cortocircuito o la sobrecarga.

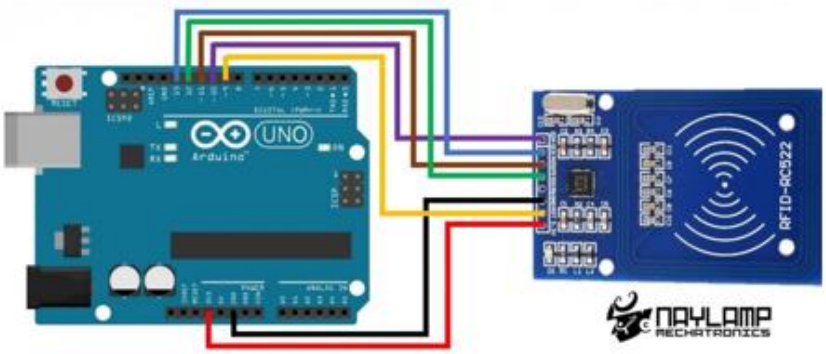
CARACTERÍSTICAS FÍSICAS Y ESCUDO COMPATIBILIDAD

La longitud máxima y el ancho de la PCB Mega2560 son 4 y 2,1 pulgadas respectivamente, con el conector USB y la toma de corriente que se extiende más allá de la dimensión anterior. Tres orificios de tornillo permiten que la placa se fije a una superficie o caja. Obsérvese que la distancia entre los pines digitales 7 y 8 es 160 mil (0,16 "), no un múltiplo par de la separación de 100 mil de los otros pines.

El Mega2560 está diseñado para ser compatible con la mayoría de los escudos diseñados para el Uno, Diecimila o Duemilanove. Los pines digitales 0 a 13 (y los pines AREF y GND adyacentes), las entradas analógicas 0 a 5, el encabezado de alimentación y la cabecera ICSP se encuentran en ubicaciones equivalentes.

Además, el UART principal (puerto serie) se encuentra en los mismos pines (0 y 1), al igual que las interrupciones externas 0 y 1 (pines 2 y 3, respectivamente). SPI está disponible a través de la cabecera ICSP tanto en el Mega2560 como en Duemilanove / Diecimila. Tenga en cuenta que I2C no se encuentra en los mismos pines en el Mega (20 y 21) que el Duemilanove / Diecimila (entradas analógicas 4 y 5).

MÓDULO LECTOR RFID-RC522 RF CON ARDUINO
INTRODUCCIÓN



El Módulo Lector RFID-RC522 RF utiliza 3.3V como voltaje de alimentación y se controla a través del protocolo SPI, así como el protocolo UART, por lo que es compatible con casi cualquier micro controlador, Arduino o tarjeta de desarrollo. El RC522 utiliza un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos de 13.56Mhz. Como se hará una lectura y escritura de la tarjeta, es necesario conocer las características de los bloques de memoria una tarjeta: La tarjeta que viene con el módulo RFID cuenta con 64 bloques de memoria (0-63) donde se hace lectura y/o escritura. Cada bloque de memoria tiene la capacidad de almacenar hasta 16 Bytes. El número de serie consiste de 5 valores hexadecimales, se podría utilizar esto para hacer una operación dependiendo del número de serie.

Características del Módulo Lector RFID-RC522 RF

Modelo: MF522-ED
Corriente de operación: 13-26mA a 3.3V
Corriente de stand by: 10-13mA a 3.3V
Corriente de sleep-mode: <80uA
Corriente máxima: 30mA
Frecuencia de operación: 13.56Mhz
Distancia de lectura: 0 a 60mm
Protocolo de comunicación: SPI
Velocidad de datos máxima: 10Mbit/s
Dimensiones: 40 x 60 mm
Temperatura de operación: -20 a 80°C
Humedad de operación: 5%-95%
Máxima velocidad de SPI: 10Mbit/s

A continuación se muestra una tabla con los pines del Módulo Lector RFID-RC522 RF, así como la conexión que tendrá con el Arduino UNO, NANO y MEGA.
Conexión del entre el módulo RFID y Arduino

Módulo RC522	Arduino Uno, Nano	Arduino Mega
SDA (SS)	10	53
SCK	13	52
MOSI	11	51
MISO	12	50
IRQ	No conectado	No conectado
GND	GND	GND
RST	9	9
3.3V	3.3V	3.3V

SIEMENS

S7-1500

Potencia + eficiencia: SIMATIC S7-1500 plus TIA Portal

El ultimate plus de automatización

La serie de controladores SIMATIC S7-1500 constituyen la nueva generación de controladores de TIA Portal y de automatización. SIMATIC S7-1500 le asegura el más alto nivel de eficiencia y de usabilidad para aplicaciones de rango medio y alto en máquinas y sistemas de automatización.

>

Diseño

Los controladores SIMATIC S7-1500 integran:

Display para puesta en marcha y diagnóstico, desde el que poder diagnosticar tanto el funcionamiento de la CPU como de sus módulos. El Display puede acoplarse y desacoplarse de la CPU durante su funcionamiento. Protección posible con password vía TIA Portal. Ciclo de vida mayor de 50.000 horas de operación.

Interfaz PROFINET integrada en cada CPU. PN IRT (V2.2.), lo que le asegura tiempos de respuesta y alta precisión en el comportamiento de la máquina. Web Server integrado para la visualización de información de servicio y diagnosis.

Concepto de memoria innovada. Suficiente memoria para cada aplicación. Capacidad hasta 2 GB para datos de proyecto, archivos, recetas y documentos.

Concepto de diagnosis optimizado. Eficiente análisis de fallo desde Display, Web Sever, STEP 7 o HMI. Imposible pérdida de mensajes de error, aun estando la CPU apagada.

Tecnología integrada. Motion Control con conexión rápida a los accionamientos PROFIdrive. TRACE, grabación de hasta 16 variables para una optimización precisa de los programas de control y accionamientos. PID integrado para tareas de lazo cerrado, ahorros de tiempo

>

CPUs estándares

Las CPUs estándares se caracterizan por su total modularidad.

Por este motivo, en su estructura podremos encontrarnos lo siguiente:

Una Unidad Central de Proceso (CPU) para ejecutar el programa de usuario

Una o varias Fuentes de alimentación

Módulos de señal de entradas / salidas digitales / analógicas

Módulos tecnológicos y de comunicación si se precisan

¿ Safety? Por supuesto, en los módulos F de seguridad.

>

CPUs compactas

SIMATIC S7-1511C-1PN/CPU 1512C-1PN

Los dos nuevos controladores SIMATIC integran entradas / salidas en el mismo bloque que la CPU. Esto les hace recomendable para aquellas aplicaciones donde el ahorro de espacio sea importante, tales como máquinas de fabricación en serie.

Debido a su diseño compacto, la CPU 1511C integra 32 entradas/salidas digitales, 5 entradas analógicas y 2 salidas analógicas en un bloque de ancho 85 mm. La CPU 1512C integra 64 entradas / salidas digitales, 5 entradas y 2 salidas analógicas en un ancho de 110 mm. Ambas CPUs pueden ampliarse mocularmente con más entradas / salidas. Adiciona a esto, es de destacar las funciones tecnológicas integradas tales como contaje, medida y posicionamiento, Tales ventajas suponen un ahorro sustancial en comparación con los controladores modulares, lo que supone un ahorro considerable en el stock de repuestos.

Al igual que el resto de la gama, las CPUs compactas integran interfaz PROFINET IO Controller / IO device con dos puertos para configuración en linea o anillo.

>

CPUs de Seguridad

Las CPUs de Seguridad SIMATIC S7-1500F, son idóneas para aquellas apicaciones de seguridad y estándar en las que se requiere gran memoria de datos y programa.

Gracias al interfaz PROFINET IO IRT integrado, permitirá el establecimiento en planta de una configuración lineal (Switch integrado) o en estrella, admitiendo estructuras distribuidas de automatizacion adicional a la periferia centralizada.

El interfaz adicional integrado PROFINET con dirección IP específica es útil para el establecimiento de una red adicional independiente de la primarea. El interfaz PROFINET, permite la conexión de periferia de entradas / salidas a la CPU via PROFIBUS (o PROFIsafe)

Las CPUs SIMATIC S7-151xF de seguridad están certificadas para funcionalidad de seguridad según norma EN61508 (version 2010), estando certificadas para aplicaciones hasta SIL 3 (IEC62061) y PL (ISC 13849).

Tanto el programa como la configuración de seguridad podrá ser protegida mediante password. Quiénes hoy en día estén trabajando con las CPUs SIMATIC S7-300F/400F, disponen de herramientas de migración a los nuevos contraoldores SIMATIC S7-151xF.

CPUs de Seguridad: **CPU 1511F-1 PN; CPU 1513F-1 PN; CPU 1515F-2 PN; CPU 1516F-3 PN/DP; CPU 1517F-3 PN/DP; CPU 1518F-4 PN/DP**

>

CPUs SIPLUS

SIPLUS SIMATIC S7-1500 constituye la última generación de controladores en incorporarse a TIA Portal. Constituye el ultimate plus para sus tareas de automatización en extremas condiciones, tales como:

- Rangos de temperatura: -40°C a 70°C
- Condiciones de condensación, humedad, incrementando os grados de protección (suciedad, agua
- Extraordinarias condiciones de ambiente (atmósferas de polución de gas)
- Varios rangos de voltaje

> Más información

Sistemas RFID SIMATIC RF

para optimizar el flujo de materiales y la logística

Estos sistemas RFID electrónicos e inteligentes identifican de forma rápida, económica y segura, son insensibles a la suciedad y almacenan datos directamente en el producto u objeto. Así dirigen y optimizan la producción, el flujo de materiales y garantizan una logística eficiente.

Todo el que trabaja con sistemas de identificación tiene unas expectativas muy concretas. Así, por ejemplo, unos buscan etiquetas inteligentes a bajo precio para tareas de logística, mientras que otros lo que necesitan son portadores de datos robustos para líneas de montaje. En la logística de transportes, sin embargo, un componente esencial son los transpondedores de gran alcance.

Sistemas aptos para cualquier aplicación:

- Líneas de montaje
- Sistemas de manutención y transporte
- Fabricación industrial
- Almacenes
- Logística
- Distribución
- Preparación de pedidos (picking)
- Logística de transportes

Los sistemas RFID de Siemens llevan años demostrando su eficacia en las más variadas aplicaciones y garantizan seguridad y fiabilidad las 24 horas del día. Destacadas empresas del sector industrial apuestan en todo el mundo por los sistemas RFID de Siemens, desde hace más de 25 años.

A destacar

- Identificación automática y segura con un 100% de seguridad en la transferencia
- Posibilidad de almacenar directamente en el producto datos de producción y calidad
- Insensibles a fluctuaciones de temperatura y suciedad
- Amplia gama de portadores de datos reutilizables, desde la etiqueta inteligente hasta el transpondedor de 64 kbytes
- Comunicación flexible con el sistema de automatización: serie, vía PROFIBUS, PROFINET o Ethernet
- La integración inmediata en SIMATIC reduce los costes de ingeniería
- Conformidad con las normas ISO 14443, ISO 15693, ISO 18000-4 así como EPCglobal e ISO/IEC 18000-6.

Sistema RFID	MOBY E	SIMATIC RF300	MOBY D	SIMATIC RF600	MOBY U
Frecuencia	13,56 MHz	13,56 MHz	13,56 MHz	865 - 868 MHz (Europa) 902 - 928 MHz (Norteamérica)	2,4 GHz
Distancia de escritura/lectura	hasta 0,1 m	hasta 0,2 m	hasta 0,9 m	hasta 10 m (2 x 2 antenas, montadas en oposición)	hasta 3,0 m
Normas	ISO 14443	ISO 15693	ISO 15693 ISO 18000-3	EPC Gen 1, EPC Gen 2, ISO 18000-6B, ISO 18000-6C	ISO 18000-4

Sistemas RFID SIMATIC RF

Principio de funcionamiento de los sistemas RFID

Frente a los sistemas de identificación convencionales, los sistemas RFID de Siemens ofrecen grandes ventajas:

Gran fiabilidad gracias a la transferencia de datos sin contacto, portadores (tag/transpondedor) para una gestión de datos centralizada o descentralizada e integración homogénea en el sistema: todo ello permite incorporarlos de forma rápida y sencilla en la aplicación y así ahorrar tiempo y dinero.

Gracias a nuestros sistemas RFID, los productos u objetos están acompañados de datos expresivos desde el primer momento. El portador de datos simplemente se fija al producto, portaproducto, objeto o su unidad de transporte o embalaje y en él se escriben datos sin establecer contacto. El portador de datos almacena de esta forma todos los datos específicos de la aplicación. Es indiferente si se trata de una pieza de carrocería en una fábrica de automóviles o de recipientes para la composición de pedidos en un centro de distribución.

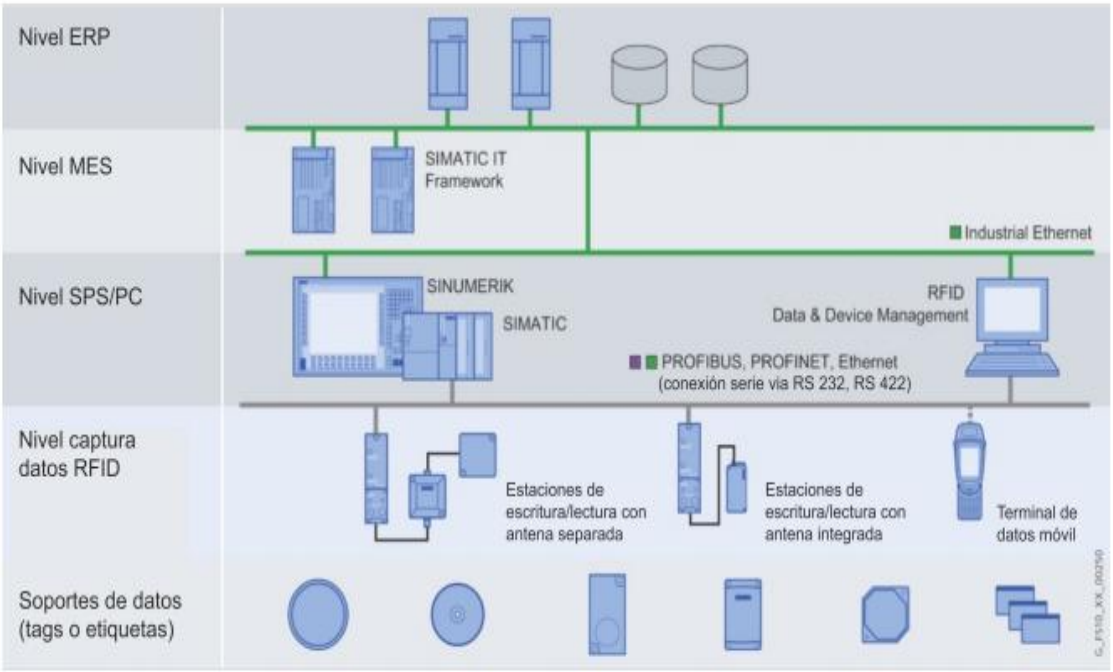
Es posible almacenar hasta 64 kbytes de datos, legibles en cada puesto de trabajo o estaciones de fabricación, sitios en los que también pueden completarse. Esto permite sincronizar óptimamente el flujo de materiales y datos.

Componentes plenamente compatibles

- Portadores de datos
- Estaciones de escritura/lectura así como terminales de mano
- Antenas
- Módulos de comunicación para conectar al sistema de automatización (PROFIBUS, PROFINET, Ethernet)
- Software para la integración en sistema

Integración flexible en sistema

Sean cuales sean los requisitos: Los sistemas RFID SIMATIC RF ofrecen una integración sencilla en sistemas SIMATIC o SINUMERIK y en redes PROFIBUS, Ethernet o de PC y se pueden conectar a cualquier PLC. Una amplia gama de módulos de comunicación, bloques de función así como potentes drivers y librerías de funciones simplifica y acelera la integración en cualquier aplicación.



Sistemas RFID para la banda de HF

Resumen de datos técnicos

Sistema RFID	MOBY E				SIMATIC RF300	
						
Distancia de escritura/lectura	hasta 100 mm				hasta 200 mm	
Velocidad de transferencia de datos						
■ Lectura	típ. 400 bytes/s				típ. 3 kbytes/s (modo ISO típ. 0,6 kbytes/s)	
■ Escritura	típ. 350 bytes/s				típ. 3 kbytes/s (modo ISO típ. 0,4 kbytes/s)	
Memoria	EEPROM				FRAM/EEPROM	
Normas	ISO 14443-A				ISO 15693	
Homologaciones	CE, UL, FCC, CSA				CE, UL, FCC, CSA	
Apto para grupo de portadores	■ (sólo con SIM)				no	
Apto para multitag	■ (sólo con SIM)				no	
Frecuencia	13,56 MHz				13,56 MHz	
Portadores de datos (tags)	**Designación**	**Cap. de memoria**	**Temp. de empleo**	**Grado de prot.**	**Designación**	**Capacidad de memoria**
	MDS E600	752 bytes	-25 ... +60 °C	IP68	RF320T	20 bytes
	MDS E624	752 bytes	-25 ... +125 °C	IP67/IPX9K	RF340T	8 kbytes
	MDS E611	752 bytes	-25 ... +75 °C	IP67	RF350T	32 kbytes
	MDS E623	752 bytes	-25 ... +85 °C	IP67/IPX9K	RF360T	8 kbytes
					RF370T	32 kbytes/64 kbytes
					RF380T	32 kbytes
					MDS D100	112 bytes
					MDS D124	112 bytes
					MDS D139	112 bytes
					MDS D160	112 bytes
					MDS D324	992 bytes
Estaciones de escritura/lectura	**Designación**	**Temperatura de empleo**		**Grado de prot.**	**Designación**	**Temperatura de empleo**
■ Estacionarias, con antena separada	SIM 70 con ANT 0	-25 ... +70 °C		IP65/67	RF350R	-25 ... +70 °C
	SIM 70 con ANT 1	-25 ... +70 °C		IP65/67		
	SLG 75	-25 ... +70 °C		IP65		
■ Estacionarias, con antena integrada	SLG 72	-25 ... +70 °C		IP65	RF310R	-25 ... +70 °C
	SIM 72	-25 ... +70 °C		IP65	RF340R	-25 ... +70 °C
					RF380R	-10 ... +70 °C
■ Terminal de mano, con antena integrada	STG E	-20 ... +60 °C		IP54	RF310M	-10 ... +50 °C
Antenas	**Designación**	**Temperatura de empleo**		**Grado de prot.**	**Designación**	**Temperatura ambiente**
	SLA 71	-25 ... +70 °C		IP67	ANT 1	-25 ... +70 °C
	ANT 1	-25 ... +70 °C		IP67	ANT 18	-25 ... +70 °C
	ANT 4	-25 ... +70 °C		IP67	ANT 30	-25 ... +70 °C
	ANT 12	-25 ... +70 °C		IP67		
	ANT 18	-25 ... +70 °C		IP67		
	ANT 30	-25 ... +70 °C		IP67		
Conexión al sistema de automatización						
■ Directa	Conexión serie a otros PLCs, PCs, otros sistemas				Conexión serie a otros PLCs, PCs, otros	
■ Vía módulo de comunicación (ASM)	SIMATIC S7-300, S7-400; PROFIBUS DP; PROFINET; Conexión serie a otros PLCs, PCs, otros sistemas				Sistemas SIMATIC S7-300, S7-400; PROFIBUS DP;	
Referencia genérica	**6GT23**				**6GT280**	

PROGRAMACIÓN ARDUINO MEGA 2560

```
//***** CARGO LAS DIRECTIVAS DEL PREPROCESADOR *****
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>
//*****

//***** DEFINICION DE PINES A USAR *****
#define RST1_PIN 49          //*** PIN RESET RFID
#define SS1_PIN 47           //*** PIN SS RFID

#define RST2_PIN 45          //*** PIN RESET RFID
#define SS2_PIN 43           //*** PIN SS RFID

#define RST3_PIN 41          //*** PIN RESET RFID
#define SS3_PIN 39           //*** PIN SS RFID

#define RST4_PIN 23          //*** PIN RESET RFID
#define SS4_PIN 25           //*** PIN SS RFID

#define RST5_PIN 27          //*** PIN RESET RFID
#define SS5_PIN 29           //*** PIN SS RFID

#define RST6_PIN 31          //*** PIN RESET RFID
#define SS6_PIN 33           //*** PIN SS RFID

#define RST7_PIN 35          //*** PIN RESET RFID
#define SS7_PIN 37           //*** PIN SS RFID

#define RST8_PIN 22          //*** PIN RESET RFID
#define SS8_PIN 24           //*** PIN SS RFID

#define BUZZER1 32           //*** Pin Buzzer 1
#define BUZZER2 30           //*** Pin Buzzer 2
#define BUZZER3 28           //*** Pin Buzzer 3
#define BUZZER4 26           //*** Pin Buzzer 4

#define LV1 40
#define LA1 38
#define LR1 36

#define LV2 48
#define LA2 46
#define LR2 44
```

```
#define LPV 42
#define LPR 34
//*****

//***** CREACIÓN DE OBJETOS A USAR *****
MFRC522 mfrc522_1 (SS1_PIN, RST1_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_2 (SS2_PIN, RST2_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_3 (SS3_PIN, RST3_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_4 (SS4_PIN, RST4_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_5 (SS5_PIN, RST5_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_6 (SS6_PIN, RST6_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_7 (SS7_PIN, RST7_PIN);      //Creamos el objeto para el RC522
MFRC522 mfrc522_8 (SS8_PIN, RST8_PIN);      //Creamos el objeto para el RC522
//*****

//***** DECLARACION DE VARIABLES GLOBALES *****
int PROCESO1=0;
int CONTADOR=0;
int HORARIO=0;
int ACCESO=0, ACCESO1=0, a=0, LEIBLE=0, b=0, c=0;

byte ActualUID[4];

//***** CONSTANTES DE ALCANCE GLOBAL
byte Usuario1[4]= {0x13, 0xA7, 0x1D, 0x2B} ;
byte Usuario2[4]= {0x09, 0x9B, 0xF9, 0x65} ;

int tiempo1ax=8;//58;
int tiempo1bx=10;//60;
int tiempo1cx=11;//61;
int tiempo1dx=11;//61;

int tiempo1ex=8;//58;
int tiempo1fx=10;//60;
int tiempo1gx=11;//61;
int tiempo1hx=11;//61;

int tiempo2ax=5;//58;
int tiempo2bx=7;//60;
int tiempo2cx=8;//61;
int tiempo2dx=8;//61;

int tiempo2ex=5;//58;
int tiempo2fx=7;//60;
int tiempo2gx=8;//61;
int tiempo2hx=8;//61;
```

```
int tiempo1a;
int tiempo1b;
int tiempo1c;
int tiempo1d;

int tiempo1e;
int tiempo1f;
int tiempo1g;
int tiempo1h;

int tiempo2a;
int tiempo2b;
int tiempo2c;
int tiempo2d;

int tiempo2e;
int tiempo2f;
int tiempo2g;
int tiempo2h;

int g=0;
//***** MENU DE CONFIGURACIONES *****
void setup()
{
  Serial.begin(9600);                 //*** INICIAMOS LA COMUNICACION SERIAL
  SPI.begin();                        //*** INICIAMOS LA COMUNICACION SPI

  mfrc522_1.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 1
  mfrc522_2.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 2
  mfrc522_3.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 3
  mfrc522_4.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 4
  mfrc522_5.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 5
  mfrc522_6.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 6
  mfrc522_7.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 7
  mfrc522_8.PCD_Init();               //***** INICIAMOS EL LECTOR RFID 8

  //***** DECLARACIONES DE SALIDAS EN PUERTOS *****
  pinMode(BUZZER1, OUTPUT);
  pinMode(BUZZER2, OUTPUT);
  pinMode(BUZZER3, OUTPUT);
  pinMode(BUZZER4, OUTPUT);

  pinMode(LV1, OUTPUT);
  pinMode(LA1, OUTPUT);
  pinMode(LR1, OUTPUT);

  pinMode(LV2, OUTPUT);
```

```
  pinMode(LA2, OUTPUT);
  pinMode(LR2, OUTPUT);

  pinMode(LPV, OUTPUT);
  pinMode(LPR, OUTPUT);
  //*****

  Serial.println("CONTROL DE SEMAFOROS PARA PERSONAS CON
DISCAPACIDAD");

  HORARIO = EEPROM.read(0);
  if(HORARIO!=0 && HORARIO!=1)
  HORARIO=0;

  if(HORARIO==0)
  {
    HORARIO=1;
    EEPROM.write(0,1);
    Serial.println("EJECUCION DE HORA PICO");
    BEEP2(10,50);
  }
  else if(HORARIO==1)
  {
    HORARIO=0;
    EEPROM.write(0,0);
    Serial.println("EJECUCION DE HORA NORMAL");
    BEEP2(5,500);
  }

  delay(2000);

  igualar();
}

//***** EJECUTO CONSTANTEMENTE *****
void loop()
{
  //***** EJECUCIONES EN HORA NORMAL *****
  if(HORARIO==0)
  {
    CONTADOR++;
    Serial.println(CONTADOR);
    Serial.println("a");
    if(PROCESO1==0)
    {
      Serial.println("EJECUTANDO PROCESO1=0 EN HORARIO NORMAL");
```

```
if(CONTADOR < tiempo1a)       //*** se enciende la luz verde por 57 segundos
{
  //Serial.println("OK1");
  digitalWrite(LV1,HIGH);
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,LOW);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,LOW);
  digitalWrite(LR2,HIGH);

  digitalWrite(LPV,LOW);
  digitalWrite(LPR,HIGH);

  //***** SONIDO DE CRUCE DE NO VIDENTES *****
  /*if(b==0)
  {
    digitalWrite(BUZZER1,HIGH);
    digitalWrite(BUZZER2,HIGH);
    b=1;
  }*/

}
if((CONTADOR >= tiempo1b && CONTADOR <tiempo1c) )

{
  if(a==1)
  {
    CONTADOR = tiempo1a;
    a=0;
  }
  //Serial.println("OK2");
  Serial.println("b");
  delay(10);

  digitalWrite(LV1,LOW);
  digitalWrite(LA1,HIGH);
  digitalWrite(LR1,LOW);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,LOW);
  digitalWrite(LR2,HIGH);
}

if(CONTADOR >= tiempo1d)
{
  if(ACCESO==1)
```

```
    {
      ACCESO=2;
    }
    PROCESO1=1;
    CONTADOR=0;
    if (g==1){
    igualar();
    }

    Serial.println("b");
    digitalWrite(LV1,LOW);
    digitalWrite(LA1,LOW);
    digitalWrite(LR1,HIGH);

    digitalWrite(LV2,HIGH);
    digitalWrite(LA2,LOW);
    digitalWrite(LR2,LOW);

    digitalWrite(LPV,HIGH);
    digitalWrite(LPR,LOW);

    //Serial.println("OK3");
   }
}

if(PROCESO1==1)
{
  Serial.println("EJECUTANDO PROCESO1=1 EN HORARIO NORMAL");

  if(CONTADOR < tiempo1e)
  {
    //Serial.println("ok1");
    digitalWrite(LV1,LOW);
    digitalWrite(LA1,LOW);
    digitalWrite(LR1,HIGH);

    digitalWrite(LV2,HIGH);
    digitalWrite(LA2,LOW);
    digitalWrite(LR2,LOW);

    digitalWrite(LPV,HIGH);
    digitalWrite(LPR,LOW);

      //Serial.println("OK4");

    //***** SONIDO DE CRUCE DE NO VIDENTES *****
    /*if(b==0)
```

```
  {
    digitalWrite(BUZZER3,HIGH);
    digitalWrite(BUZZER4,HIGH);
    b=1;
  }*/
}

if(CONTADOR >= tiempo1f && CONTADOR <tiempo1g)
{
  //Serial.println("ok4");
  Serial.println("b");
  digitalWrite(LV1,LOW);
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,HIGH);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,HIGH);
  digitalWrite(LR2,LOW);

  //Serial.println("OK5");

  if(c==1)
  {
    CONTADOR = tiempo1e;
    c=0;
  }
}

if(CONTADOR >= tiempo1h)
{
  //Serial.println("ok2");
  if(ACCESO1==1)
  {
     ACCESO1=2;
  }

  PROCESO1=0;
  CONTADOR=0;
   if (g==2){
  igualar();
  }
    //Serial.println("OK6");
  Serial.println("b");
  digitalWrite(LV1,HIGH);
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,LOW);
```

```
      digitalWrite(LV2,LOW);
      digitalWrite(LA2,LOW);
      digitalWrite(LR2,HIGH);

      digitalWrite(LPV,LOW);
      digitalWrite(LPR,HIGH);
      //LEIBLE=0;
     }
    }

    //delay(10);

    //***** LECTURAS DE RFID CON 5 DELAYS DE 200ms PARA QUE SE CUMPLA
EL CONTEO DE 1 SEG
    LECTURASRF();
    //delay(200);

    digitalWrite(BUZZER1,LOW);
    digitalWrite(BUZZER2,LOW);
    digitalWrite(BUZZER3,LOW);
    digitalWrite(BUZZER4,LOW);
    b=0;

    LECTURASRF();
    delay(200);
    LECTURASRF();
    delay(200);
    LECTURASRF();
    delay(200);
    LECTURASRF();
    delay(200);
   }

   //***** EJECUCIONES EN HORA PICO *****
   if(HORARIO==1)
   {
    CONTADOR++;
    Serial.println(CONTADOR);
    Serial.println("a");
    if(PROCESO1==0)
    {
     Serial.println("EJECUTANDO PROCESO1=0 EN HORARIO NORMAL");

     if(CONTADOR < tiempo2a)
     {
      //Serial.println("OK1");
      digitalWrite(LV1,HIGH);
```

```
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,LOW);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,LOW);
  digitalWrite(LR2,HIGH);

  digitalWrite(LPV,LOW);
  digitalWrite(LPR,HIGH);

  //***** SONIDO DE CRUCE DE NO VIDENTES *****
  /*if(b==0)
  {
    digitalWrite(BUZZER1,HIGH);
    digitalWrite(BUZZER2,HIGH);
    b=1;
  }*/
}
if((CONTADOR >= tiempo2b && CONTADOR <tiempo2c) )
{
  if(a==1)
  {
    CONTADOR = tiempo2a;
    a=0;
  }
  //Serial.println("OK2");
  Serial.println("b");
  delay(10);

  digitalWrite(LV1,LOW);
  digitalWrite(LA1,HIGH);
  digitalWrite(LR1,LOW);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,LOW);
  digitalWrite(LR2,HIGH);
}
     if(CONTADOR >= tiempo2d)
{
  if(ACCESO==1)
  {
    ACCESO=2;
  }
  PROCESO1=1;
  CONTADOR=0;
  if (g==1){
  igualar();
```

```
    }

    Serial.println("b");
    digitalWrite(LV1,LOW);
    digitalWrite(LA1,LOW);
    digitalWrite(LR1,HIGH);

    digitalWrite(LV2,HIGH);
    digitalWrite(LA2,LOW);
    digitalWrite(LR2,LOW);

    digitalWrite(LPV,HIGH);
    digitalWrite(LPR,LOW);

    //Serial.println("OK3");
   }
}

if(PROCESO1==1)
{
  Serial.println("EJECUTANDO PROCESO1=1 EN HORARIO NORMAL");

  if(CONTADOR < tiempo2e)
  {
    //Serial.println("ok1");
    digitalWrite(LV1,LOW);
    digitalWrite(LA1,LOW);
    digitalWrite(LR1,HIGH);

    digitalWrite(LV2,HIGH);
    digitalWrite(LA2,LOW);
    digitalWrite(LR2,LOW);

    digitalWrite(LPV,HIGH);
    digitalWrite(LPR,LOW);



    //Serial.println("OK4");

    //***** SONIDO DE CRUCE DE NO VIDENTES *****
    /*if(b==0)
    {
      digitalWrite(BUZZER3,HIGH);
      digitalWrite(BUZZER4,HIGH);
      b=1;
    }*/
```

```
}

if(CONTADOR >= tiempo2f && CONTADOR <tiempo2g)
{
  //Serial.println("ok4");
  Serial.println("b");
  digitalWrite(LV1,LOW);
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,HIGH);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,HIGH);
  digitalWrite(LR2,LOW);

  //Serial.println("OK5");

  if(c==1)
  {
    CONTADOR = tiempo2e;
    C=0;
  }
}

if(CONTADOR >= tiempo2h)
{
  //Serial.println("ok2");
  if(ACCESO1==1)
  {
     ACCESO1=2;
  }

  PROCESO1=0;
  CONTADOR=0;
   if (g==2){
  igualar();
  }
    //Serial.println("OK6");
  Serial.println("b");
  digitalWrite(LV1,HIGH);
  digitalWrite(LA1,LOW);
  digitalWrite(LR1,LOW);

  digitalWrite(LV2,LOW);
  digitalWrite(LA2,LOW);
  digitalWrite(LR2,HIGH);

  digitalWrite(LPV,LOW);
```

```
      digitalWrite(LPR,HIGH);
      //LEIBLE=0;
     }
    }

   //delay(10);


   //***** LECTURAS DE RFID CON 5 DELAYS DE 200ms PARA QUE SE CUMPLA
EL CONTEO DE 1 SEG
   LECTURASRF();
   //delay(200);

   digitalWrite(BUZZER1,LOW);
   digitalWrite(BUZZER2,LOW);
   digitalWrite(BUZZER3,LOW);
   digitalWrite(BUZZER4,LOW);
   b=0;

   LECTURASRF();
   delay(200);
   LECTURASRF();
   delay(200);
   LECTURASRF();
   delay(200);
   LECTURASRF();
   delay(200);
  }
}

//***** LECTURAS RF *****
void LECTURASRF()
{
 if(PROCESO1==1)
 {
  //Serial.println("LEYE RF1 RF2 RF5 RF6");
  RF1();
  RF2();
  RF5();
  RF6();
 }
 if(PROCESO1==0)
 {
  //Serial.println("LEYE RF3 RF4 RF7 RF8");
  RF3();
  RF4();
  RF7();
```

```
   RF8();
  }
}
//*****

void RF1 ()
{
  //Serial.println("BUSCANDO DATO EN RF1");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_1.PICC_IsNewCardPresent())
        {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
          if ( mfrc522_1.PICC_ReadCardSerial())
          {
              //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
              //Serial.print(F("Card UID:"));
              for (byte i = 0; i < mfrc522_1.uid.size; i++) {
                    //Serial.print(mfrc522_1.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    //Serial.print(mfrc522_1.uid.uidByte[i], HEX);
                    ActualUID[i]=mfrc522_1.uid.uidByte[i];
              }
              //Serial.print("     ");
              //***** COMPARO SI ES IGUAL A ALGUUN SUARIO REGISTRADO
              if(compareArray(ActualUID,Usuario1))
                {
                 Serial.println("Acceso CONCEDIDO...RF1");
                 ACCESO1=1;
                 sumar1();
                 //LEIBLE=1;
                 BEEP1(2,100);
                 c=1;
                }
              else if(compareArray(ActualUID,Usuario2))
                {
                 Serial.println("Acceso CONCEDIDO...RF1");
                 ACCESO1=1;
                 sumar1();
                 //LEIBLE=1;
                 BEEP1(2,100);
                 c=1;
                }
              else
              {
                //Serial.println("Acceso denegado...RF1");
                BEEP1(10,50);
              }
```

```
            //*****SE TERMINAL LA LECTURA Y COMUNICACION
             mfrc522_1.PICC_HaltA();

         }
  }
}

void RF7 ()
{
  //Serial.println("BUSCANDO DATO EN RF7");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_2.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
         if ( mfrc522_2.PICC_ReadCardSerial())
         {
             //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
             //Serial.print(F("Card UID:"));
             for (byte i = 0; i < mfrc522_2.uid.size; i++)
             {
                   //Serial.print(mfrc522_2.uid.uidByte[i] < 0x10 ? " 0" : " ");
                   //Serial.print(mfrc522_2.uid.uidByte[i], HEX);
                   ActualUID[i]=mfrc522_2.uid.uidByte[i];
             }
             //Serial.print("     ");
             //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
             if(compareArray(ActualUID,Usuario1))
               {
                Serial.println("Acceso CONCEDIDO...RF7");
                ACCESO=1;
                sumar2();
                //LEIBLE=1;
                BEEP1(2,100);
                a=1;
               }
             else if(compareArray(ActualUID,Usuario2))
               {
                Serial.println("Acceso CONCEDIDO...RF7");
                ACCESO=1;
                sumar2();
                //LEIBLE=1;
                BEEP1(2,100);
                a=1;
               }
             else
               {
               //Serial.println("Acceso denegado...RF7");
```

```
              BEEP1(10,50);
            }

            //*****SE TERMINAL LA LECTURA Y COMUNICACION
            mfrc522_2.PICC_HaltA();

        }
  }
}

void RF2 ()
{
  //Serial.println("BUSCANDO DATO EN RF2");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_3.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
        if ( mfrc522_3.PICC_ReadCardSerial())
        {
            //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
            //Serial.print(F("Card UID:"));
            for (byte i = 0; i < mfrc522_3.uid.size; i++) {
                  //Serial.print(mfrc522_3.uid.uidByte[i] < 0x10 ? " 0" : " ");
                  //Serial.print(mfrc522_3.uid.uidByte[i], HEX);
                  ActualUID[i]=mfrc522_3.uid.uidByte[i];
            }
            //Serial.print("     ");
            //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
            if(compareArray(ActualUID,Usuario1))
              {
               Serial.println("Acceso CONCEDIDO...RF2");
               ACCESO1=1;
               sumar1();
               //LEIBLE=1;
               BEEP1(2,100);
               c=1;
              }
            else if(compareArray(ActualUID,Usuario2))
              {
               Serial.println("Acceso CONCEDIDO...RF2");
               ACCESO1=1;
               sumar1();
               //LEIBLE=1;
               BEEP1(2,100);
               c=1;
              }
            else
```

```
            {
              //Serial.println("Acceso denegado...RF2");
              BEEP1(10,50);
            }

            //*****SE TERMINAL LA LECTURA Y COMUNICACION
            mfrc522_3.PICC_HaltA();

        }
  }
}

void RF8 ()
{
  //Serial.println("BUSCANDO DATO EN RF8");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_4.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
          if ( mfrc522_4.PICC_ReadCardSerial())
          {
            //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
            //Serial.print(F("Card UID:"));
            for (byte i = 0; i < mfrc522_4.uid.size; i++) {
                  //Serial.print(mfrc522_4.uid.uidByte[i] < 0x10 ? " 0" : " ");
                  //Serial.print(mfrc522_4.uid.uidByte[i], HEX);
                  ActualUID[i]=mfrc522_4.uid.uidByte[i];
            }
            //Serial.print("     ");
            //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
            if(compareArray(ActualUID,Usuario1))
              {
               Serial.println("Acceso CONCEDIDO...RF8");
               ACCESO=1;
               sumar2();
               //LEIBLE=1;
               BEEP1(2,100);
               a=1;
              }
            else if(compareArray(ActualUID,Usuario2))
              {
               Serial.println("Acceso CONCEDIDO...RF8");
               ACCESO=1;
               sumar2();
               //LEIBLE=1;
               BEEP1(2,100);
               a=1;
```

```
              }
             else
             {
              //Serial.println("Acceso denegado...RF8");
              BEEP1(10,50);
             }/*ACCESO=1;
                a=1;
                //LEIBLE=1;
                BEEP1(5,500);
              }*/

             //*****SE TERMINAL LA LECTURA Y COMUNICACION
             mfrc522_4.PICC_HaltA();

         }
  }
}

void RF3 ()
{
  //Serial.println("BUSCANDO DATO EN RF3");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_5.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
         if ( mfrc522_5.PICC_ReadCardSerial())
         {
             //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
             //Serial.print(F("Card UID:"));
             for (byte i = 0; i < mfrc522_5.uid.size; i++) {
                    //Serial.print(mfrc522_5.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    //Serial.print(mfrc522_5.uid.uidByte[i], HEX);
                    ActualUID[i]=mfrc522_5.uid.uidByte[i];
             }
             //Serial.print("     ");
             //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
             if(compareArray(ActualUID,Usuario1))
              {
               Serial.println("Acceso CONCEDIDO...RF3");
               ACCESO=1;
               sumar2();
               //LEIBLE=1;
               BEEP1(2,100);
               a=1;
              }
             else if(compareArray(ActualUID,Usuario2))
              {
```

```
                Serial.println("Acceso CONCEDIDO...RF3");
                ACCESO=1;
                sumar2();
                //LEIBLE=1;
                BEEP1(2,100);
                a=1;
               }
              else
               {
               //Serial.println("Acceso denegado...RF3");
               BEEP1(10,50);
              }

              //*****SE TERMINAL LA LECTURA Y COMUNICACION
              mfrc522_5.PICC_HaltA();

        }
  }
}

void RF6 ()
{
  //Serial.println("BUSCANDO DATO EN RF6");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_6.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
          if ( mfrc522_6.PICC_ReadCardSerial())
          {
              //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
              //Serial.print(F("Card UID:"));
              for (byte i = 0; i < mfrc522_6.uid.size; i++) {
                    //Serial.print(mfrc522_6.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    //Serial.print(mfrc522_6.uid.uidByte[i], HEX);
                    ActualUID[i]=mfrc522_6.uid.uidByte[i];
              }
              //Serial.print("     ");
              //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
              if(compareArray(ActualUID,Usuario1))
               {
                Serial.println("Acceso CONCEDIDO...RF6");
                ACCESO1=1;
                sumar1();
                //LEIBLE=1;
                BEEP1(2,100);
                c=1;
               }
```

```
            else if(compareArray(ActualUID,Usuario2))
              {
               Serial.println("Acceso CONCEDIDO...RF6");
               ACCESO1=1;
               sumar1();
               //LEIBLE=1;
               BEEP1(2,100);
               c=1;
              }
            else
              {
              //Serial.println("Acceso denegado...RF6");
              BEEP1(10,50);
            }

            //*****SE TERMINAL LA LECTURA Y COMUNICACION
            mfrc522_6.PICC_HaltA();

        }
  }
}

void RF5 ()
{
  //Serial.println("BUSCANDO DATO EN RF5");
  //***** REVISO SI HAY TARJETA PRESENTE
  if ( mfrc522_7.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
          if ( mfrc522_7.PICC_ReadCardSerial())
          {
            //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
            //Serial.print(F("Card UID:"));
            for (byte i = 0; i < mfrc522_7.uid.size; i++) {
                  //Serial.print(mfrc522_7.uid.uidByte[i] < 0x10 ? " 0" : " ");
                  //Serial.print(mfrc522_7.uid.uidByte[i], HEX);
                  ActualUID[i]=mfrc522_7.uid.uidByte[i];
            }
            //Serial.print("     ");
            //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
            if(compareArray(ActualUID,Usuario1))
              {
               Serial.println("Acceso CONCEDIDO...RF5");
               ACCESO1=1;
               sumar1();
               //LEIBLE=1;
               BEEP1(2,100);
```

```
            c=1;
           }
          else if(compareArray(ActualUID,Usuario2))
           {
            Serial.println("Acceso CONCEDIDO...RF5");
            ACCESO1=1;
            sumar1();
            //LEIBLE=1;
            BEEP1(2,100);
            c=1;
           }
          else
           {
           //Serial.println("Acceso denegado...RF5");
           BEEP1(10,50);
          }

          //*****SE TERMINAL LA LECTURA Y COMUNICACION
          mfrc522_7.PICC_HaltA();
        }
  }
}

void RF4 ()
{
  //Serial.println("BUSCANDO DATO EN RF4");
  //***** REVISO SI HAY TARJETA PRESENTE
  if (mfrc522_8.PICC_IsNewCardPresent())
       {
     //***** SELECCIONO LA TARJETA PARA LECTURA SERIAL
          if ( mfrc522_8.PICC_ReadCardSerial())
          {
              //***** IMPRIMO EN EL TERMINAL EL ID DE LA TARJETA
              //Serial.print(F("Card UID:"));
              for (byte i = 0; i < mfrc522_8.uid.size; i++) {
                    //Serial.print(mfrc522_8.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    //Serial.print(mfrc522_8.uid.uidByte[i], HEX);
                    ActualUID[i]=mfrc522_8.uid.uidByte[i];
              }
              //Serial.print("     ");
              //***** COMPARO SI ES IGUAL A ALGUUN USUARIO REGISTRADO
              if(compareArray(ActualUID,Usuario1))
               {
                Serial.println("Acceso CONCEDIDO...RF4");
                ACCESO=1;
                sumar2();
                //LEIBLE=1;
```

```
              BEEP1(2,100);
              a=1;
             }
            else if(compareArray(ActualUID,Usuario2))
             {
              Serial.println("Acceso CONCEDIDO...RF4");
              ACCESO=1;
              sumar2();
              //LEIBLE=1;
              BEEP1(2,100);
              a=1;
             }
            else
             {
             //Serial.println("Acceso denegado...RF4");
             BEEP1(10,50);
            }

            //*****SE TERMINAL LA LECTURA Y COMUNICACION
            mfrc522_8.PICC_HaltA();

        }

  }
}

//***** FUNCION PARA REALIZAR LA COMPARACION DE LOS VECTORES
boolean compareArray(byte array1[],byte array2[])
{
 if(array1[0] != array2[0])return(false);
 if(array1[1] != array2[1])return(false);
 if(array1[2] != array2[2])return(false);
 if(array1[3] != array2[3])return(false);
 return(true);
}
//*****

//***** FUNCIONES PARA LA REPRODUCCION DE LOS BEEP *****
void BEEP1 (int CONTEO, int TIEMPO)
{
 for(int CONT=0; CONT<CONTEO; CONT++)
 {
   digitalWrite(BUZZER1,HIGH);
   delay(TIEMPO);
   digitalWrite(BUZZER1,LOW);
   delay(TIEMPO);
 }
```

```
}

void BEEP2 (int CONTEO, int TIEMPO)
{
  for(int CONT=0; CONT<CONTEO; CONT++)
  {
    digitalWrite(BUZZER2,HIGH);
    delay(TIEMPO);
    digitalWrite(BUZZER2,LOW);
    delay(TIEMPO);
  }
}

void BEEP3 (int CONTEO, int TIEMPO)
{
  for(int CONT=0; CONT<CONTEO; CONT++)
  {
    digitalWrite(BUZZER3,HIGH);
    delay(TIEMPO);
    digitalWrite(BUZZER3,LOW);
    delay(TIEMPO);
  }
}

void BEEP4 (int CONTEO, int TIEMPO)
{
  for(int CONT=0; CONT<CONTEO; CONT++)
  {
    digitalWrite(BUZZER4,HIGH);
    delay(TIEMPO);
    digitalWrite(BUZZER4,LOW);
    delay(TIEMPO);
  }
}
//*****

void sumar1(){
tiempo1a=tiempo1a+15;
tiempo1b=tiempo1b+15;
tiempo1c=tiempo1c+15;
tiempo1d=tiempo1d+15;

tiempo2a=tiempo2a+15;
tiempo2b=tiempo2b+15;
tiempo2c=tiempo2c+15;
tiempo2d=tiempo2d+15;
```

```
g=1;
}

void sumar2(){
 tiempo1e=tiempo1e+15;
 tiempo1f=tiempo1f+15;
 tiempo1g=tiempo1g+15;
 tiempo1h=tiempo1h+15;

 tiempo2e=tiempo2e+15;
 tiempo2f=tiempo2f+15;
 tiempo2g=tiempo2g+15;
 tiempo2h=tiempo2h+15;

 g=2;

}

void igualar(){
tiempo1a=tiempo1ax;
tiempo1b=tiempo1bx;
tiempo1c=tiempo1cx;
tiempo1d=tiempo1dx;

tiempo1e=tiempo1ex;
tiempo1f=tiempo1fx;
tiempo1g=tiempo1gx;
tiempo1h=tiempo1hx;

tiempo2a=tiempo2ax;
tiempo2b=tiempo2bx;
tiempo2c=tiempo2cx;
tiempo2d=tiempo2dx;

tiempo2e=tiempo2ex;
tiempo2f=tiempo2fx;
tiempo2g=tiempo2gx;
tiempo2h=tiempo2hx;
}
```

PROGRAMACION ARDUINO UNO

```
//**** Uso de la biblioteca LiquidCrystal

#include <LiquidCrystal.h>
```

```
//***** SELECCIÓN DE PINS A UTILIZAR POR EL PANEL LCD
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
// **** DEFINIMOS ALGUNOS VALORES UTILIZADOS POR EL PANEL Y LOS BOTONES
int lcd_key     = 0;
int adc_key_in  = 0;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5
char recvChar;
int cont=0;

// **** LEER LOS BOTONES
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);
 // mis botones cuando se leen están centrados en estos valores: 0, 144, 329, 504, 741
 // Se añade aproximadamente 50 valores y comprobamos si estamos cerca
 if (adc_key_in > 1000) return btnNONE;
 // For V1.1 us this threshold
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 250)  return btnUP;
 if (adc_key_in < 450)  return btnDOWN;
 if (adc_key_in < 650)  return btnLEFT;
 if (adc_key_in < 850)  return btnSELECT;

 // For V1.0 comment the other threshold and use the one below:
/*
```

```
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 195)  return btnUP;
 if (adc_key_in < 380)  return btnDOWN;
 if (adc_key_in < 555)  return btnLEFT;
 if (adc_key_in < 790)  return btnSELECT;
*/
 return btnNONE;  // when all others fail, return this...
}

void setup()
{
 lcd.begin(16, 2);
 lcd.setCursor(0,0);
 lcd.print("Contador: ");
  Serial.begin(9600);
}

void loop()
{
 lcd.setCursor(0,1);
 //lcd.print(millis()/1000);
 lcd.print(cont);

if (Serial.available()){
recvChar = Serial.read();
if (recvChar=='a'){
cont++;
}
if (recvChar=='b'){
cont=0;
lcd.setCursor(0,1);
```

```
 //lcd.print(millis()/1000);
 lcd.print("         ");
}
}
}
```

"Responsabilidad con pensamiento positivo"

ACTA DE APROBACIÓN DEL PLAN DEL PROYECTO INTEGRADOR DE CARRERAS.

1. INFORMACIÓN GENERAL

Tema:	Diseño y prototipo de sistema inteligente para semáforos del Distrito Metropolitano de Quito para dar prioridad a personas con discapacidad.
Estudiante:	Freddy Bolívar Almachi Toaquiza
Carrera:	Electrónica Digital y Telecomunicación
Tutor:	Ing. David Cando, Mg
Asesor Técnico:	Ing. David Cando, Mg
Fecha:	03/10/2016

2. EVALUACIÓN

No	Componentes	Norma	Clave
1.	Tema	1	C (1) CP (0.5) NC(0) NA(-)
2.	Problema (justificación)	1	
3.	Objetivo General	1	
4.	Objetivo Especifico	1	
5.	Hipótesis	1	
6.	Marco Teórico	1	
7.	Metodología	1	
8.	Cronograma	1	
9	Bibliografía	1	
10	Presentación y Exposición	1	
TOTAL		10	

(C) Cumple: Con lo establecido en el componente de manera íntegra.

(CP) Cumple Parcialmente: Cumple lo establecido en el componente de manera parcial o con observaciones.

(NC) No Cumple: Ausencia del componente, incoherencia entre el componente y su contenido u omisión de contenido.

(NA) No Aplica: En el caso que la temática del trabajo no amerite la inclusión de uno o más componentes.

3. OBSERVACIONES FINALES

Hasta cinco líneas para establecer criterios finales acerca del trabajo.

4. DECLARATORIA DE APROBACIÓN / MODIFICACIÓN O RECHAZO:

( ) Aprobado ( ) Modificación ( ) Rechazado

Firma Tutor Metodológico

Firma Tutor Técnico

Firma Director de Carrera