UNIVERSIDAD NACIONAL DE MAR DEL PLATA

FACULTAD DE INGENIERIA
Departamento Ingeniería Elécrica

Proyecto Final de Grado

Noviembre, 2019

Diseño y construcción de un prototipo de controlador de semáforos leds

CÓDIGO

Autor
Alumno: BIDEGAIN, Octavio
Ingeniería Electromecánica

Tutor: Ing. Guillermo J. Murcia
Co-Tutor: Dr. Ing. Jorge L. Strack

Evaluadores:
Ing. Gustavo Belliski
Ing. Máximo Menna
Ing. Oscar Noguera

Contenido

AGRADECIMIENTOS

Quiero agradecer a todos aquellos que me acompañaron a lo largo de la carrera y que con su apoyo permitieron que alcance mis objetivos.

A mis padres y familia por brindarme las bases de mi desarrollo, por acompañarme en cada momento difícil. Agradecerles por tanto esfuerzo, trabajo y constancia para que no me falte nada y por confiar en cada una de mis decisiones.

A ella, mi complemento perfecto, que a lo largo de este camino ha sabido esperarme, entenderme y motivarme. Te agradezco porque no hubiese podido sin que estés a mi lado.

A mis amigos que saben cómo sacarme una sonrisa y hacer que todo sea más sencillo. Agradecerles por todos los momentos compartidos.

A mi Director y Co-Director por la confianza y apoyo para realizar este proyecto. Por respetar mis tiempos y por su inmensa predisposición.

OBJETIVOS DEL TRABAJO

El objetivo de este trabajo es diseñar e implementar un prototipo de controlador de semáforos leds que cumpla con los requerimientos de las Normas vigentes, que cuente con prestaciones similares a las que poseen los controladores comerciales y que presente un bajo costo de construcción en comparación a las alternativas comerciales existentes.

El punto de partida de este proyecto surgió en la Práctica Profesional Supervisada que realicé en la empresa Grupo Núcleo S. A. donde, entre otros proyectos, programé controladores de semáforos comerciales importados para ser implementados en la ciudad de Mar del Plata. Durante esta experiencia observé la poca flexibilidad del software de configuración utilizado como interfaz para el usuario y su elevado costo. De ambas consideraciones surgió la propuesta de realizar un prototipo con similares prestaciones, pero de menor costo y con una interfaz gráfica más intuitiva, aprovechando los conocimientos de Electrónica Aplicada, Instrumentación Virtual y Automatismos Industriales entre otras, adquiridos durante la cursada de las distintas asignaturas.

A lo largo del presente trabajo se detallará el diseño y la construcción de un prototipo de un controlador de semáforos sumado a una interfaz gráfica.

El objetivo general es lograr un controlador que sea sencillo, que satisfaga los requisitos de funcionamiento mínimos que expresen las normativas vigentes y que sea económico. Para ello, el controlador y la interfaz fueron implementadas mediante Arduino y LabVIEW respectivamente.

1. INTRODUCCION

Cuando dos o más vías se intersectan, según el tamaño y nivel de tráfico de estas, puede resultar peligroso y necesitar de una regulación. Es allí donde los semáforos toman importancia.

Los semáforos son dispositivos de señalización mediante los cuales se regula la circulación de vehículos, bicicletas y peatones en vías, asignando el derecho de paso o prelación de vehículos y peatones secuencialmente, por las indicaciones, operadas por una unidad de control de ahora en más llamada controlador. [1]

Los semáforos cuentan con tres luces cuyos colores estandarizados son:

- Rojo: Señal que determina la detención vehicular.
- Amarillo: Señal que determina prevención por cambio en el flujo del tránsito.
- Verde: Señal que permite el movimiento vehicular.

Las principales funciones de los semáforos son:

- Interrumpir periódicamente el tránsito de una corriente vehicular o peatonal para permitir el paso de otra corriente vehicular.
- Eliminar o reducir el número y gravedad de algunos tipos de accidentes, principalmente los que implican colisiones perpendiculares.
- Proporcionar un ordenamiento del tránsito.
- Controlar la circulación por carriles.
- Regular la velocidad de los vehículos para mantener la circulación continua a una velocidad constante.

El semáforo busca lograr un tránsito ordenado y seguro. Para ello es necesario un previo estudio del lugar a instalar el semáforo y el tráfico de este. La ingeniería de tránsito determina teóricamente la manera en que los semáforos deben funcionar. Cuando la instalación de semáforos se justifica, pero está mal proyectada o, ni siquiera se justifica e igualmente se instalan, ocasionan desventajas como:

- Demoras excesivas en el tránsito.
- Fomentan la desobediencia a las indicaciones semafóricas.
- Inducen a usar rutas menos convenientes, para evitar dichos dispositivos.
- Incrementan significativamente la frecuencia de accidentes. [2]

El control de los semáforos en cada intersección lo realiza el controlador, quien está conectado a los semáforos y da las órdenes de encendido y apagado de las lámparas. Los controladores pueden ser de diferentes tecnologías, electromecánicos, electrónicos o micropocesadores. Este encendido y apagado de las lámparas se realiza de acuerdo con el plan de tráfico, el cual determina los movimientos autorizados que se realizan en la intersección, su orden y el tiempo de duración.

A continuación, se definen algunos conceptos que se utilizan en la regulación semafórica:

- El *ciclo* es el tiempo que transcurre entre el encendido de una lámpara hasta que se vuelve a encender luego de haber completado toda la secuencia.

- Las *fases* se definen como los estados de luces que permanecen invariables por un determinado tiempo.

- El *desfasaje* es el tiempo que tarda en comenzar el ciclo un semáforo en referencia a otro.

- El *ancho de banda* es el tiempo desde que se inicia la onda verde hasta el último vehículo que puede ingresar en la onda verde ya con la velocidad constante.

- El *programa* es el conjunto de instrucciones que definen la forma en que operará el semáforo durante un tiempo.

- El *plan* está definido como el conjunto de programas que se aplican a un día típico de semana, fin de semana o especial.

En su concepto básico el semáforo no contempla lo que sucede en otras intersecciones. Con la evolución del tráfico el problema dejó de ser local y, en ciertos lugares, es necesario estudiar lo que sucede aguas arriba y aguas debajo para poder garantizar un tránsito fluido. La coordinación o sincronización es la encargada de determinar cómo se relacionan semáforos contiguos para asegurar que un vehículo encuentre la menor cantidad de semáforos en rojo y evitar así la mayor cantidad de paradas posibles.

A partir de esto surge la Onda Verde que es una sucesión de semáforos sincronizados para lograr que un vehículo los encuentre a todos en verde si circula a una velocidad determinada. Para generar una Onda Verde existen diversos tipos de métodos, desde gráficos espacio-tiempo hasta soluciones evolutivas basadas en algoritmos genéticos.

La sincronización se puede dar de distintas maneras, según la precisión que se desee será su complejidad. A continuación, se detallan algunos métodos de lograr coordinación.

- Maestro – Escalvo: Mediante la utilización de una placa y un protocolo de comunicación, el controlador maestro transmite datos a todos los controladores esclavos indicándoles el momento de su encendido.

- Coordinado por Reloj de Tiempo Real: Cada controlador tiene programado un tiempo de desfasaje respecto del anterior inmediato o de un controlador de referencia y todos obtienen la hora, minutos y segundos de un mismo reloj satelital.

- Coordinado por Frecuencia de Red (50/60 Hz): Se utiliza para formar una onda verde tomando como referencia de coordinación la hora, minutos y segundos de un reloj de tiempo basado en la frecuencia de la red eléctrica.

- Coordinado por Señal de 220 Vca: En este caso, el controlador espera por una señal de tensión del controlador que lo precede y mediante un accionamiento electromecánico comienza el funcionamiento.

1.1. Tecnologías

En el mercado se pueden encontrar diversos tipos de controladores de semáforos. A continuación, se mencionan las principales clases.

Semáforos de tiempos fijos: Son aquellos en el cual el ciclo, la duración y secuencia de intervalos son invariables y están definidos por un programa establecido con anticipación. Un semáforo puede tener varios programas, con el objeto de activarlos a diferentes horas del día para satisfacer mejor la demanda del tránsito.

Una de las principales ventajas de los dispositivos de tiempos fijos es la versatilidad a la hora de coordinarlos para generar una Onda Verde. Por lo general, los dispositivos que se instalan en esquinas aisladas cuentan con esta funcionalidad anticipando un futuro crecimiento de tránsito en la zona. Además, el costo de este tipo de equipos suele ser menor al de los demás modelos.

Semáforos totalmente accionados por el tránsito: En este caso el ciclo es variable, dependiendo de la duración que tenga la fase según el tránsito real. Para lograr este funcionamiento se utilizan diferentes periféricos (neumáticos, bobinas de inducción, rayos infrarrojos, cámaras, etc.).

- Inductivos: La detección de vehículos se produce por medio de un cambio de inductancia, provocado por el paso del vehículo sobre un alambre enterrado bajo la calzada.
  Si se coloca dos espiras a una distancia conocida se puede medir la velocidad de los vehículos. Tienen como desventaja su poca practicidad a la hora de realizar mantenimiento correctivo.

- Radar de Microondas: Emiten energía a altas frecuencias en la dirección en la que se desplazan los vehículos. Detectan el flujo de tránsito y su velocidad por el cambio en la frecuencia de la señal emitida debido al efecto Doppler, que es proporcional a la velocidad del vehículo.
  Este tipo de detectores son transportables y miden con gran precisión la velocidad, no generan un impacto visual en la calzada y no influye el clima en su correcto funcionamiento. Como desventaja, no detecta a menos que cuenten

con una tecnología más compleja, vehículos parados o con baja velocidad de circulación (≤ 10 km/h).

- Visión artificial: Su funcionamiento se basa en analizar y procesar imágenes capturadas por cámaras. Se detecta cambios de parámetros en una sucesión de imágenes para lograr obtener la velocidad y flujo del tráfico.
  Permiten tener sistemas de detección de infracciones, no generan impacto visual y los datos obtenidos son muy fiables. Como desventaja tienen un alto costo inicial, y son vulnerables al clima o condiciones que afecten la visibilidad de la vía.

- Ultrasonido: Emiten ondas de sonido perpendicularmente sobre la calzada. Se detecta el tránsito gracias a la diferencia de tiempos en llegar la onda reflejada en el caso que lo haga sobre el pavimento o sobre un vehículo. La frecuencia de las ondas es superior a las audibles. Son muy sensibles al clima. [3]

Semáforos semiaccionados por el tránsito: Disponen de medios para ser accionados en uno o más accesos. Estos semáforos son aplicables a las intersecciones de vías con alto volumen y altas velocidades, con calles secundarias de tránsito relativamente liviano. La indicación es normalmente verde en la calle principal, cambiando de estado cuando se detecta vehículos o peatones en la calle secundaria.

Semáforos controlados por computadoras: En este caso los cambios de fases los envía una central por medios de comunicación como por ejemplo ethernet o GSM. [4] Cuentan con software de tele supervisión y telecontrol, logrando detecciones de fallas y su corrección en tiempo real.

1.2. Fallas

Durante la operación del semáforo puede suceder anomalías inesperadas separadas en dos grandes grupos: fallas mayores y fallas menores.

Las fallas mayores son aquellas que atentan contra la seguridad de las personas por no garantizar el correcto funcionamiento del sistema de señalización. Las fallas mayores requieren ser detectadas, que el controlador cambie a modo de falla (amarillo intermitente) y ser registradas. A continuación, se describen las fallas mayores.

Verde conflictivo: presentación simultánea de señales de verde, permitiendo la circulación de tránsito conflictivo en una intersección.

Ausencia de rojo: se da cuando una señal debe estar encendida en rojo y permanece apagada.

También se pueden considerar fallas mayores errores en comprobaciones del programa o planes de tránsito.

Las fallas menores son aquellas que no constituyen una falla mayor y que pueden ser identificadas y registradas. Este tipo de fallas no debe forzar al controlador a cambiar

a modo de falla. Son aquellas que influyen en la duración de las señales o alteran el orden de las fases, pero no atentan contra la seguridad.

1.3. Onda verde – Método gráfico

Cómo ya se mencionó, existen varias formas de cálculo de una onda verde. Cuando el diseño es sencillo, vías de uno y doble sentido, se puede realizar manualmente mediante el método gráfico.

Los parámetros que están en juego en una coordinación en un sentido son:

- Distancia entre intersecciones.
- Velocidad a la que se desea la onda verde.
- Ciclo común para todas las intersecciones.
- Ancho de la onda verde
- Desfase
- Tiempo de verde efectivo de cada intersección.

Donde se realiza un tanteo entre los primeros 4 parámetros, siempre y cuando estos no estén fijos, lográndose encontrar el desfase y el tiempo verde efectivo de cada intersección mediante el diagrama espacio-tiempo. El diagrama es un sistema de coordenadas cartesianas donde el eje de las ordenadas corresponde a las distancias en metros de la vía a coordinar y el eje de las abscisas al tiempo en segundos. A continuación, se puede observar un ejemplo a modo de orientación.

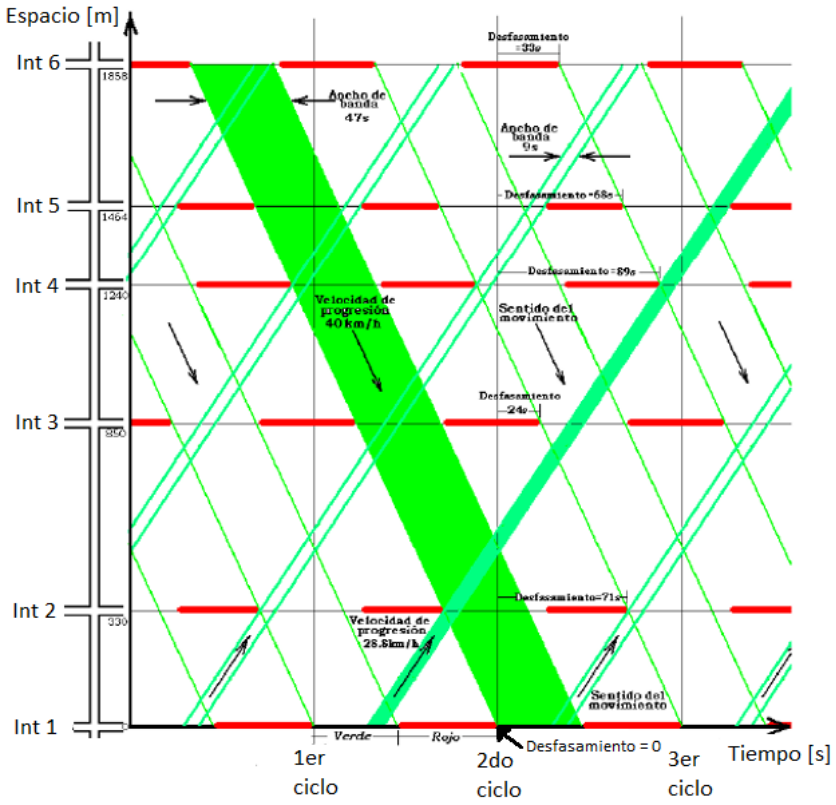

Figura 1.2.1 – Gráfico espacio-tiempo

En el eje de abscisas se trazan líneas de referencia que representan la duración del ciclo común. En el eje de ordenadas se trazan líneas paralelas que representan la ubicación de cada intersección. Del cruce de estas se permite dibujar el reparto del ciclo de cada intersección, representando con franjas en negro el verde efectivo y con franjas en rojo el rojo efectivo, lo que resulta ser el diagrama de fase correspondiente a cada intersección. Además, se permite observar el desfase que tiene cada una de las intersecciones respecto a la de referencia.

Por líneas paralelas, comprendidas entre los espacios de verde efectivo, se forma la onda verde. El ancho de banda va a ser la separación medida en segundos entre las líneas que limitan la onda verde y representa el tiempo disponible que tiene un vehículo para circular sin detenerse a la velocidad fijada.

De la Figura 1.2.1 se nota que realizar una coordinación en un solo sentido puede resultar sencillo, mientras que realizar una coordinación en ambos sentidos no tanto. Cabe destacar que cuando se coordina en dos sentidos estos pueden tener velocidades y anchos de bandas distintos. En este caso, la solución no es única y será a criterio del proyectista establecer el diseño más favorable.

El diagrama espacio-tiempo es un método sencillo de prueba y error, en el mercado se encuentran diferentes softwares que permiten su cálculo.

1.4. Ofertas comerciales

Los tipos y funcionalidades de los semáforos son abundantes. Existe un controlador para cada esquina en particular, controladores de los más generales y controladores modulares que permiten la adaptación a medida que pasa el tiempo y se incrementan las necesidades. A continuación, se presentarán distintos controladores que pueden encontrarse en el mercado.

1.4.1 Ofertas nacionales

1.4.1.1. MICRO ELECTRONICA S.R.L.

Es una empresa argentina, fundada en 1986, que se encuentra situada en la ciudad de Olavarría, Pcia. de Buenos Aires. Desarrolla y fabrica íntegramente diferentes dispositivos electrónicos que se utilizan en la industria. [5]

Entre sus productos cuenta con *controladores semafóricos programables con GPS.*

Figura 0.1 – Controlador semafórico programable con GPS de Micro electronica S.R.L

Características técnicas y funcionales:

- Tensión de funcionamiento 110/220 Vca.
- Frecuencia de funcionamiento 60/50 Hz.
- Equipo totalmente electrónico.
- Basado en microcontroladores de 8 bits.
- Salidas de potencia de estado sólido
- Salidas de hasta 8 movimientos.
- Programación mediante PC a través de RS485.
- Programación mediante teclado y display LCD.
- Posee calendario.
- Tiene hasta 24 salidas según el modelo.
- Alarma de falla en lámparas.
- Sincronismo de onda verde mediante una central, un reloj externo o por señal de 220 Vca.
- Opcional módulo de GPS.
- Módulos desmontables. Placa de potencia y lógica separadas.

1.4.1.2. NORTELEC SERVICIOS S.R.L.

Es una empresa argentina radicada en San Fernando, Pcia. de Buenos Aires. Desde 1972 se dedica al alumbrado público, semaforización, embellecimiento urbano y señalización vial. Desde sus inicios comenzó a especializarse en Electromecánica, tableros eléctricos y montajes industriales, llegando a atender a las sedes argentinas de reconocidas marcas mundiales. [6]

Entre sus productos cuentan con el siguiente controlador semafórico:

Figura 0.2 – Controlador de semáforos de la empresa Nortelec Servicios S.R.L

Características técnicas y funcionales:

- Tensión de funcionamiento: 150 a 250 V.
- Señales de entrada y salida optoacopladas.
- Salidas de potencia mediante triacs.
- Programas de señales almacenados en memoria no volátil.
- Cuenta con Led´s para identificación de estados.
- Detección y registro de fallas de lámparas rojas y verdes.
- Detección y registro de fallas de tensión.
- Recepción de pulso de sincronismo.
- Transmisión de datos.
- Pulso de reloj.
- Modo titilante nocturno.
- 12 salidas a lámparas.
- Sistema que permite el conexionado de cámaras de seguridad.
- Sincronismo satelital (GPS).

1.4.1.3. SYNTICO S.A. y TACUAR S.R.L

SYNTICO S.A es una empresa argentina, radicada en La Plata, Pcia. Buenos Aires. Se dedica a la prestación de servicios profesionales en el área de la ingeniería. Se especializa fundamentalmente en desarrollos digitales para empresas, consultoría y cálculos de ingeniería en el ámbito del transporte público de pasajeros, control de tránsito vehicular y medio ambiente. Desarrollos e implementación de tecnologías ITS. [7]

Tacuar S.R.L fue fundad en 1976, en la ciudad capital de la provicia de Santa Fe. Se dedica al desarrollo de tecnologías para semaforización y control de tránsito. Vende sus productos dentro del país y exporta a Paraguay, Bolivia, Ecuador y Chile, entre otros países de Latinoamérica.

Entre sus productos, ambas empresas, cuentan con el controlador de semáforos inteligente modelo CET-234 N. El cual es utilizado en la ciudad de Necochea, Pcia. Buenos Aires, Argentina.

Figura 0.3 – Controlador CET-234N de Syntico S.A.

Características técnicas y funcionales

- Tensión de funcionamiento: 220V.
- Cuenta con placas modulares. Una placa central CPU, varias placas de potencia y una placa de comunicaciones.
- Se basa en microprocesadores.
- Comunicación RS 232 para programar desde PC.
- Módulo teclado-visor para programar.
- Cuenta con 8 salidas o 6 salidas según su versión.
- Cuenta con calendario.
- Cuenta con borneras auxiliares para sensores de tráfico.
- Puede funcionar aislado, como maestro, coordinado por reloj en tiempo real, coordinado por frecuencia de red, coordinado por señal de 220 Vca.

1.4.2. Ofertas internacionales

1.4.2.1. TRAFICTEC S.A.

Es una empresa mexicana dedicada al desarrollo y venta de productos especializados en el control de tráfico y seguridad.

Pioneros en México al introducir, en 1998, el controlador de semáforos basado en tecnología GPS. Recientemente vuelve a innovar con el establecimiento de un práctico método de programación semafórico, por medio de un software para PC y una memoria portable. [8]

Entre sus productos se encuentre el controlador de semáforos modelo GTC 8.

Figura 0.1 – Controlador GTC 8 de Trafictec S.A.

Características técnicas y funcionales

- Tensión de funcionamiento 120/220 VCA.
- Receptor GPS para sincronización.
- Llave de datos “token”, para mover programas desde la PC al controlador.
- Capacidad de 2 programas diferentes de manejo de tráfico.
- 40 intervalos por ciclo.
- 7 planes diurnos más un plan nocturno.
- Calendario para programar 12 días festivos.
- Conector para monitor de Conflictos (opcional)
- Puerto serial RS232 para comunicaciones con sistemas centralizados.
- Software basado en plataforma Windows, multi-lenguaje y con pantallas gráficas.
- Indicadores LED por salida.

1.4.2.2. SIEMENS AG

Es una empresa alemana, fundada en 1847. Se desempeña a nivel mundial sobre casi todos los tópicos de la ingeniería, desde electricidad, pasando por gas, petróleo, obras civiles, telefonía, movilidad, energía renovable, etc.

Además, se dedica a la construcción de controladores semafóricos y sistemas de control de tráfico. Entre sus productos se encuentra el modelo CL-S214 PLUS que se encuentra instalado en algunas intersecciones de la ciudad de Mar del Plata, Pcia. Buenos Aires, Argentina.

Características técnicas y funcionales

- Tensión de funcionamiento 110/220 VCA.
- Frecuencia de funcionamiento 60/50 Hz.
- Etapa de potencia optoacoplada.
- Alarma de falla de lámpara roja y verde.
- 2 movimientos.
- Comunicación a red mediante protocolos internos mediante RS232 y RS485.
- Sistema modular.
- Sincronización con reloj interno generado a través de la frecuencia de la red.
- Módulo GPS para sincronizar el reloj a la hora satelital.
- Cuenta con un kit opcional RF para sincronización mediante señales de Radio Frecuencia.
- Cuenta con un kit opcional de GSM.
- Posee un módulo opcional de Comunicaciones Ethernet para vincular el controlador a un centro de control. Posee una página web embebida para la configuración interna. [9]

1.4.2.3. SHENZHEN NOBLE OPTO CO.

Es una empresa china, fundada en 2008. Se dedica al desarrollo y fabricación de luces de semáforos, controladores de tránsito y sistemas de tráfico basados en energía solar.

Shenzhen comercializa sus productos al resto del mundo entre los cuales se puede encontrar el modelo “2nd generation controller system”.

Figura 0.1 – Controlador 2nd generation controller system de Shenzhen Noble Opto Co.

Características técnicas y funcionales

- Tensión de funcionamiento 110/220 VCA.
- Frecuencia de funcionamiento 60/50 Hz.
- Configuración a través de control remoto por WIFI.
- Configuración a través de comunicación Ethernet.
- Modulo GPRS para sincronización.
- Reloj basado en hora satelital GPS.
- Control en tiempo real mediante Ethernet.
- Detección de fallas en lámparas verdes y roja.
- Función para detección de flujo de tránsito a través de sensores.
- Función de botón peatonal. [10]

1.5. Implementación

A lo largo del presente trabajo se detallará el diseño y la construcción de un controlador de semáforos de tiempos fijos sumado a una interfaz gráfica.

El objetivo general es lograr un controlador que sea sencillo, que cumpla con las normativas vigentes y económico. Para ello, el controlador y la interfaz fueron implementadas mediante Arduino y LabVIEW respectivamente.

1.5.1. Arduino

Alrededor del año 2005 como un proyecto de estudiantes universitarios en Italia. Buscaban contar con placas desarrolladoras de hardware económicas, que sean accesibles al presupuesto de un estudiante y que, por ese entonces, no existían. Así nació Arduino, una placa de desarrollo de hardware basada en un microcontrolador ATmega8 con su circuito impreso y un Entorno de Desarrollo Integrado (IDE) que consiste en un editor de código, un compilador y un depurador que se utiliza para programar las placas mencionadas.

Arduino utiliza la filosofía de hadware y software libre. Es por ello por lo que además de la compañía existe una comunidad mundial qué, mediante un foro, está predispuesta a ayudar, apoyando tanto a los recién iniciados como a los más expertos.

Las principales ventajas de las placas Arduino son:

- Simplifica el trabajo con microprocesadores.
- Amplia documentación y fácil de encontrar.
- Bajo costo.
- Amplia variedad de periféricos diseñados especialmente para Arduino como, por ejemplo, Wifi Shield, Ethernet Shield, entre otros.
- Es multiplataforma.

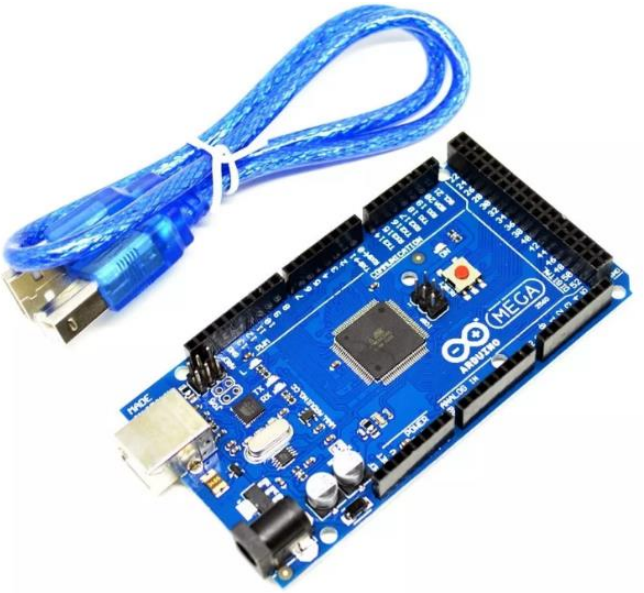

Figura 1.5.1.1 – Placa Arduino Mega 2560

El lenguaje de programación de Arduino está basado en C++. Arduino posee una programación estructurada. Esto facilita su diseño, escritura e interpretación.

Una de las grandes ventajas de trabajar con Arduino, es que se crean módulos de diversos temas listos para usarse, sin necesidad de intercalar un protoboard. En muchos casos son placas que vienen diseñadas para encajar encima de pines hembra que disponen los Arduinos.

A estos tipos de placas se les llama Shields y para que funcionen correctamente hay que incorporarles las librerías correspondientes a los sketchs. Las librerías son trozos de código hechas por terceros que facilita la implementación y la comprensión de los códigos de programación.

1.5.2. LabVIEW

LabVIEW, por sus siglas en inglés Laboratory Virtual Engineering Workbench, es un entorno de desarrollo para programación gráfica de National Instruments. Es comúnmente utilizado para adquisición de datos, instrumentos de control, automatización industrial, etc.

El lenguaje de programación usado en LabVIEW, llamado “G”, está basado en la estructura gráfica de diagramas en bloques en la cual el programador conecta distintos bloques a través de sus nodos mediante cables o “wires”. Estos cables transmiten variables y cualquier nodo puede ejecutarse tan pronto como todos sus datos de entradas estén disponibles. Dado que este podría ser el caso de múltiples nodos simultáneamente, G es capaz de ejecutar de manera paralela.

El entorno gráfico permite que personas con poca experiencia en programación desarrollen soluciones en un marco de tiempo reducido en comparación con otros sistemas.

2. NORMATIVA

Las normas técnicas son documentos aprobados por organismos reconocidos que establecen especificaciones técnicas basadas en los resultados de la experiencia y del desarrollo tecnológico, que hay que cumplir en determinados productos, procesos o servicios.

Las normas pueden clasificarse en función de su carácter como voluntaria u obligatoria y de su alcance como nacional o internacional, este último busca facilitar el comercio a nivel mundial.

El cumplimiento de normas y su certificación, para los usuarios, asegura que los productos y servicios sean seguros, confiables y de buena calidad. Para las empresas, son herramientas estratégicas que reducen los costos al minimizar el desperdicio y los errores, aumentando la productividad y dándoles la posibilidad de acceder a nuevos mercados.

A lo largo de este capítulo se presentan normas que rigen la semaforización a nivel nacional e internacional.

2.1. Normas nacionales - IRAM

El Instituto Argentino de Normalización y Certificación (IRAM) es una asociación civil sin fines de lucro cuyas finalidades específicas, en su carácter de Organismo Argentino de normalización, son establecer normas técnicas, sin limitaciones en los ámbitos que abarquen, además de propender al conocimiento y a la aplicación de la normalización como base de la calidad, promoviendo las actividades de certificación de productos y de sistemas de la calidad en las empresas para brindar seguridad al consumidor.

IRAM es el representante de Argentina en la International Organization for Standardization (ISO), en la Comisión Panamericana de Normas Técnicas (COPANT) y en la Asociación MERCOSUR de Normalización (AMN) [12].

2.1.1. IRAM 62968 – Semáforos LED para el control de tránsito vehicular

Las cabezas de semáforo se utilizan para transmitir mensajes de seguridad al usuario de la vía pública a fin de administrar los derechos de paso y regular el tránsito.

La correcta percepción de la señal luminosa depende de su color, intensidad luminosa, la distribución de su intensidad luminosa, luminancia, uniformidad de la luminancia, la luminancia del entorno y el efecto fantasma, entre otros.

Esta norma sólo se aplica a señales luminosas que utilizan LED como fuentes luminosas, de colores rojo, amarillo y verde, con diámetros de 200 mm y 300 mm. Define los requisitos para el funcionamiento visual, estructural, ambiental y de los ensayos de los semáforos para tránsito vehicular [13].

Los soportes, columnas, señales luminosas portátiles y para tránsito peatonal, se excluyen específicamente del objeto y campo de aplicación de esta norma.

2.1.2. IRAM 62969 – Sistemas de señalización del tránsito vehicular. Compatibilidad electromagnética (CEM)

La gama de productos incluidos en el campo de aplicación de esta norma son los sistemas y dispositivos de señalización del tránsito vial incluyendo por ejemplo semáforos, dispositivos de señalización y señales, reguladores y gabinetes, soportes, interconexiones, enlaces, detectores, equipos de control y alimentación eléctrica. Los sistemas de señalización del tránsito vial que operan juntamente con otros sistemas por ejemplo alumbrado público ó sistemas ferroviarios, deben cumplir también con las normas respectivas y no se debe reducir la seguridad de funcionamiento del conjunto de los equipos. Los equipos de los centros de control están excluidos de esta norma. Los elementos que tengan una función de telecomunicación también tienen que cumplir los Reglamentos Nacionales vigentes [14].

Esta norma define las condiciones de los ensayos CEM a realizar, criterios de funcionamiento, límites de emisión y ensayos de inmunidad.

2.1.3. IRAM 62970 – Semáforos LED para el control de tránsito peatonal

Las cabezas de semáforo con señales luminosas para el tránsito peatonal que utilizan LED como fuentes luminosas son el campo de aplicación de esta norma.

Las señales luminosas para el tránsito peatonal deben ser de color blanco, anaranjado y de sección cuadrada.

Esta norma incluye las señales peatonales que incorporan un contador digital regresivo integrado en la sección inferior del semáforo peatonal, correspondiente a la figura del *hombre caminando* (de color blanco) que muestra el tiempo restante para el cruce seguro de la intersección.

Define los requisitos para el funcionamiento visual, estructural, ambiental y de los ensayos de los semáforos para el tránsito peatonal.

Se excluyen específicamente del objeto y campo de aplicación los soportes, columnas, señales luminosas portátiles y para tránsito automotor [15].

2.1.4. IRAM 62020 – Equipamiento para la gestión de tránsito. Controladores de tránsito

El controlador de tránsito es el equipamiento programable para el accionamiento de semáforos con capacidad de procesamiento, dispositivos de seguridad funcional, señales de entradas y salidas e interfaces de comunicación utilizado para el control de tránsito.

Esta norma define los requisitos de control, funcionales y de seguridad de los controladores de tránsito instalados para controlar los semáforos que regulan el tránsito de vehículos y peatones en vías semaforizadas. Incluye también requisitos mecánicos, ambientales y de seguridad eléctrica. [16]

Dentro de los requisitos de control se definen los modos de operación, los métodos de operación, las prioridades de los métodos de operación. Además, se especifica qué el controlador debe tener la capacidad de funcionar de manera aislada o integrado a un grupo de controladores en coordinación con estos.

Se definen cuatro modos de operación. En primer lugar, el modo inicio dónde se debe ejecutar una secuencia de amarillo intermitente siempre que se energice el controlador. En segundo lugar, el modo reposo dónde el controlador debe permanecer en amarillo intermitente. En tercer lugar, el modo control dónde el controlador debe ejecutar la secuencia según los parámetros previamente establecidos. Por último, el modo falla dónde el controlador debe ejecutar una secuencia de amarillo intermitente. A continuación, se presenta la relación entre los diferentes modos de operación.

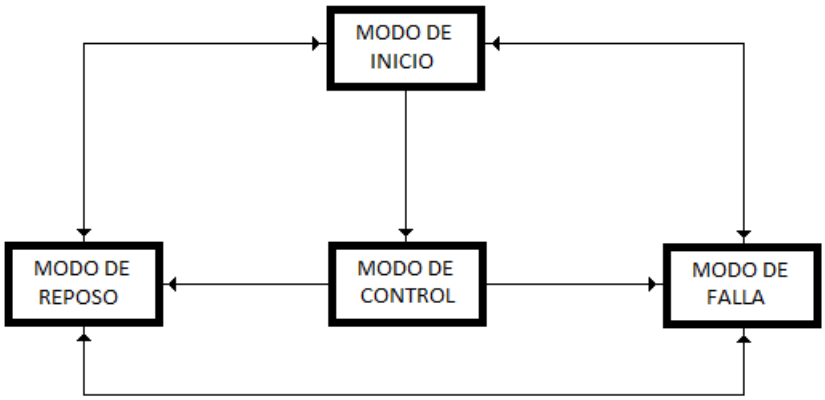

Figura 2.1.4.1 – Modos de operación

Dentro del modo control se establece que el controlador debe operar según los métodos de operación siguientes o la combinación de algunos de ellos.

- Método de operación de tiempos fijos.
- Método de operación por agenda horaria.
- Método de operación actuado (en respuesta a demandas externas que se produzcan).
- Método de operación sincronizado.
- Método de operación manual.
- Método de emergencia.
- Método de operación remota (opcional).

Los requisitos funcionales describen las prestaciones del controlador de tránsito. Dentro de grupo se definen los siguientes puntos.

- Entradas y su procesamiento: son señales digitales, pueden ser memorizadas o no y deben proceder de una fuente definida como por ejemplo un detector de vehículos, un control manual externo, un mensaje generado por computadora, una señal de sincronismo, etc.

- La exactitud de los parámetros de tiempo de ser de +/- 100 ms.
- Debe disponer de salidas de potencia y controlar grupos de señales como semáforos vehiculares o semáforos peatonales.
- Debe disponer de una interfaz de operador dónde se pueda hacer consultas de estados, fallas y de tiempo.
- Debe contar con un dispositivo de programación para acceder y modificar los datos programados. Este dispositivo puede ser conectable o parte integral del controlador.
- El controlador de tránsito puede disponer de uno o más puertos de comunicación para mantener una conectividad con una interfaz de programación externa, conexión a un centro de control para operación centralizada o conexión local de distintos equipamientos (por ejemplo: detectores virtuales de tránsito, cámaras de video, etc.)

En cuanto a requisitos de seguridad referidos al control de tránsito, se especifica que el controlador debe disponer de medios para verificar las fallas que se puedan producir. Como resultado de esta verificación se deben realizar actuaciones apropiadas para mantener la seguridad del sistema de control de tránsito. Estas actuaciones dependen de que la falla sea mayor o menor.

Una falla puede proporcionar una o más salidas de indicación de falla a un equipo remoto, o indicar mediante algún otro medio la existencia de la falla.

Las fallas mayores (ausencia de rojos o verdes simultáneos) se deben identificar y registrar y su detección debe generar que el controlador de tránsito cambie a modo de falla. En cuanto a las fallas menores, aquellas que no atentan contra la seguridad del tránsito, pueden identificarse y registrarse, pero no deben generar que el controlador cambie a modo de falla.

El controlador de tránsito debe funcionar a una tensión nominal de 220 Vca (+10% -20%).

2.2. Normas internacionales

El Comité Europeo de Normalización (CEN) es una asociación que reúne a los organismos nacionales de normalización de 34 países europeos.

CEN es uno de los tres organismos europeos de normalización, junto con CENELEC y ETSI, que han sido reconocidos oficialmente por la Unión Europea y por la Asociación Europea de Libre Comercio (AELC) como responsables del desarrollo y la definición de estándares voluntarios a nivel europeo [17].

2.2.1. EN 12675

Esta norma europea especifica los requisitos de seguridad funcional para los controladores de señales de tráfico. Es aplicable a los equipos de control de señales de tráfico instalados de forma permanente y temporal, pero excluye los equipos portátiles de control de tráfico.

Los peligros que se tratan en esta norma incluyen, entre otros, los siguientes tipos de posibles fallos de señal:

- Que una señal roja no se muestre cuando debe estar encendida.
- Que una señal verde se muestre cuando debe estar apagada.
- La falta de visualización de la secuencia de señal correcta para el tráfico.

En esta norma se definen los modos de funcionamiento (modo de espera, modo de falla), la detección de fallas mayores y menores y almacenamiento de fallas [18].

2.3. Criterios adoptados

En función de las normas expuestas, de las tecnologías disponibles, de las alternativas comerciales y de las necesidades surge la realización de un controlador de semáforos que cuente con las siguientes características y funcionalidades.

Criterios técnicos por normativa:

- Tensión de funcionamiento: 160 - 250 Vca.
- Tensión de no funcionamiento: 70 Vca.
- Frecuencia de funcionamiento 50 Hz.
- Frecuencias menores a 100 Hz no deben presentar flicker.
- Potencia mínima: 4 W.
- Tiempo de conmutación de luces menor a 100 ms.
- Factor de potencia: 0,9.
- THD menor a 20 %.
- Funcionamiento en modo inicio, modo control, modo de espera y modo de falla.
- Detección de fallas de lámparas roja y verdes.
- Detección de fallas por tensión.

Criterios funcionales adoptados por normativa:

- Modos de operación: Inicio, control, espera, falla.
- Métodos de operación: Tiempos fijos, Agenda horaria, Actuado, Sincronizado, Manual, Emergencia, Remota.
- Detección y registro de fallas.
- Dispositivos para configuración y mantenimiento.

Criterios adoptados por diseño:

- Funcionamiento por tiempos fijos.
- Configurable a través de display LCD.
- Configurable a través de software en PC.
- Coordinación a través de reloj satelital.
- Salidas para 1 intersección de 4 vías vehiculares independientes.
- Salidas de potencia optoacopladas.
- Software para control de estados en tiempo real.
- Software para diseño de onda verde a través de gráfico espacio-tiempo.
- Calendario para configuración según día del año o semana.

- Configuración para distintos horarios.

3. DISEÑO DEL CONTROLADOR PROPUESTO

A lo largo de este capítulo se describirá como está compuesto el controlador y su funcionamiento. El controlador interactúa con las cabezas de semáforos dándoles órdenes para establecer el estado de encendido o apagado. A su vez, tiene comunicación con una central y con el Sistema de Posicionamiento Global (GPS). (Ver Figura 3.1).

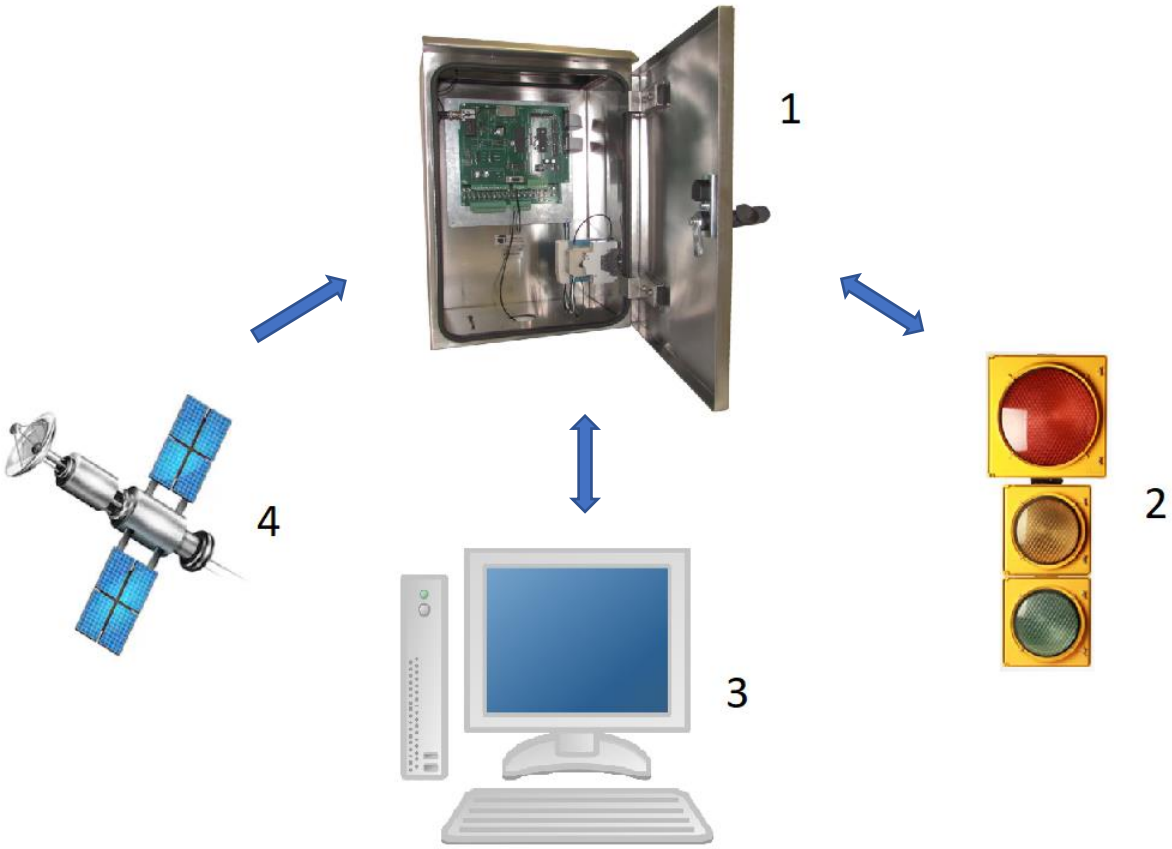

Figura 3.1 – Diagrama global

Se busca que el usuario pueda programar el controlador in situ o remotamente, desde un teclado y una pantalla LCD o desde el software central respectivamente. Dentro de los parámetros a establecer podrá optar controlar una calle de una sola vía, una intersección de dos calles simples, una intersección de una calle simple y una avenida de doble sentido y una intersección de dos avenidas.

Como se mencionó en capítulos previos, el principio de funcionamiento es por tiempos fijos donde el usuario tendrá la posibilidad de establecer la duración de tiempos verde y amarillo y del ciclo para satisfacer sus necesidades.

Cuenta con tres modos de funcionamiento, además del modo normal, para ejecutar tiempos verdes distintos o amarillo intermitente que se activaran en días y horarios fijados a criterio del usuario. También dispone de un detector de fallas de luces verdes y rojas que envía mensajes de manera remota para informar rápidamente al usuario el evento producido.

El usuario tendrá la posibilidad de utilizar el controlador de manera aislada o en sincronismo para generar una onda verde mediante un desfasaje previamente establecido.

3.1. Controlador

Es quién vincula a cada uno de los elementos aislados y garantiza el correcto funcionamiento del semáforo. Está compuesto de una fuente de alimentación, una etapa lógica y una etapa de potencia. La etapa lógica comprende tres placas Arduino Mega 2560 para lograr una estructura modular, de ahora en más Arduino Tiempo, Arduino Comunicación y Arduino Controlador. La etapa de potencia es la que permite encender las lámparas en la intersección a partir de las señales emitidas por el controlador. Esto se logra mediante una etapa de acondicionamiento que se realiza por medio de optoacopladores y relés electromecánicos.

3.1.1. Etapa lógica

Es aquella que se encarga de realizar las comparaciones y toma de decisiones para que el semáforo funcione acorde a lo establecido por el usuario. Básicamente uno de los módulos realiza las comparaciones contra los datos del GPS, otro mantiene la comunicación con el usuario, cualquiera sea esta, y el último otorga la orden de encendido de luces reales mediante la etapa de potencia.

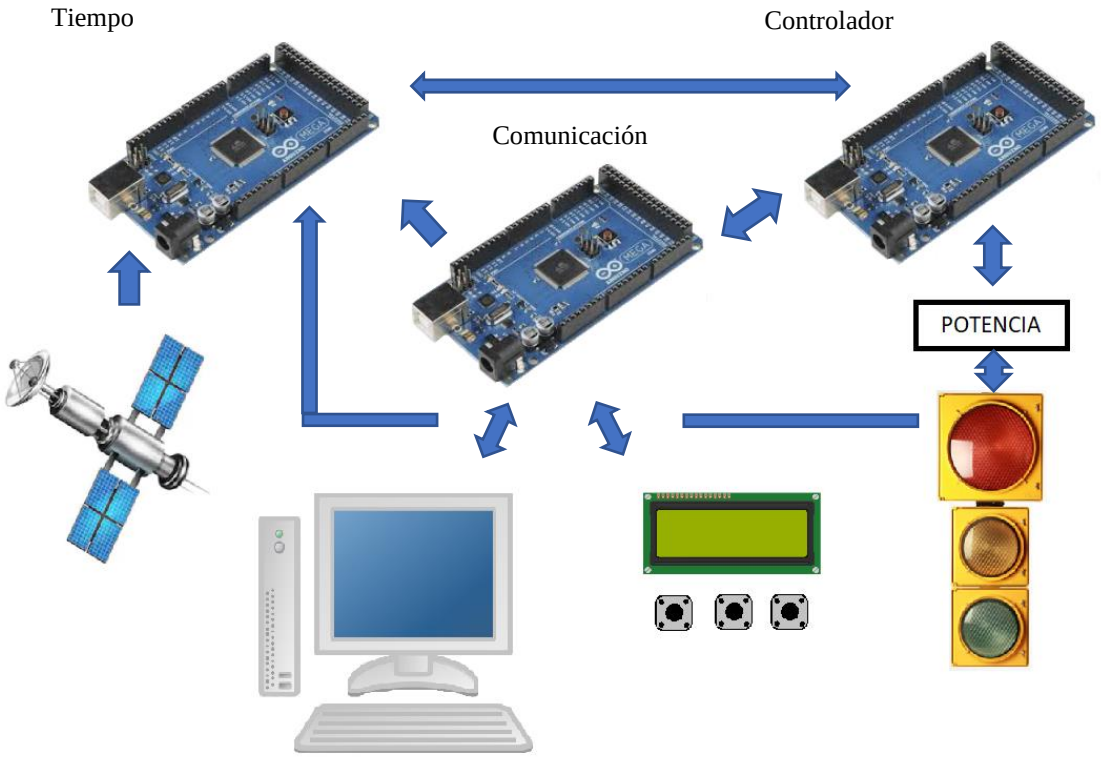

Figura 3.1.1.1 – Esquema de funcionamiento etapa lógica

Tantos las placas Arduino como el software y el menú manejan parámetros de configuración seteables por el usuario. Estos parámetros son:

Modo de funcionamiento	Variable	Especificación
------	*CantSemaforos*	Cantidad de semáforos a controlar. (1-4)
Normal	*RV1*	Tiempo verde del semáforo 1.
	RV2	Tiempo verde del semáforo 2 – Si corresponde.
	RV3	Tiempo verde del semáforo 3 – Si corresponde.
	RV4	Tiempo verde del semáforo 4 – Si corresponde.
	RA	Tiempo amarillo.
	Desfasaje	Tiempo de desfasaje respecto del controlador de referencia – En caso de ser parte de una onda verde.
	Ciclo	Ciclo según los parámetros seteados, no configurable.
Día Distinto	*RV1D*	Tiempo verde del semáforo 1.
	RV2D	Tiempo verde del semáforo 2 – Si corresponde.
	RV3D	Tiempo verde del semáforo 3 – Si corresponde.
	RV4D	Tiempo verde del semáforo 4 – Si corresponde.
	DesfasajeD	Tiempo de desfasaje respecto del controlador de referencia – En caso de ser parte de una onda verde.
	DiaD	Día de la semana que se debe activar el modo (De lunes a domingo).
	HoraDOn	Hora en la que se debe activar el modo.
	MinutoDOn	Minuto en el que se debe activar el modo.
	HoraDOff	Hora en la que se debe desactivar el modo.
	MinutoDOff	Minuto en la que se debe desactivar el modo.
	CicloD	Según parámetros seteados, no configurable.
Día Feriado	*RV1F*	Tiempo verde del semáforo 1.
	RV2F	Tiempo verde del semáforo 2 – Si corresponde.
	RV3F	Tiempo verde del semáforo 3 – Si corresponde.
	RV4F	Tiempo verde del semáforo 4 – Si corresponde.
	DesfasajeF	Tiempo de desfasaje respecto del controlador de referencia – En caso de ser parte de una onda verde.
	DiaF	Día del mes que se debe activar el modo (1-31).
	MesF	Mes del año que se debe activar el modo (1-12).
	CicloF	Ciclo según los parámetros seteados, no configurable.

Tabla 3.1.1.2 – Parámetros utilizados para la programación

El modo de funcionamiento Normal es aquel que debe ejecutarse la mayor parte del tiempo. Conforme se explicó anteriormente, se puede elegir entre 1 y 4 semáforos con tiempos de verdes distintos cada uno. Luego, se selecciona el tiempo de amarillo, común para todos los semáforos quedando así establecido el tiempo rojo de cada vía y por ende el ciclo completo. Por último, se permite elegir un tiempo de desfasaje, esta variable tomará importancia cuando el controlador este dentro de una onda verde.

Como variantes se presentan los modos Día Distinto y Día Feriado, ambos buscan brindar flexibilidad al usuario para obtener distintos comportamientos del equipo en

períodos de tiempo que sean necesarios. De esta manera se pueden obtener resultados cómo priorizar una onda verde en un sentido por la mañana y en el sentido opuesto por la tarde o incluso dejar en intermitente un semáforo en un día no laboral.

El modo Día Distinto permite elegir un día de la semana y un intervalo de tiempo en el que el controlador se accione con tiempos distintos a los normales, según las necesidades del usuario. En caso de que se elija un intervalo de tiempo para todos los días de la semana y no para un día solo, el accionar durante el intervalo será intermitente amarillo.

El modo Día Feriado permite elegir un día del año para que el comportamiento del controlador sea a tiempos distintos del normal. Cabe destacar que si el Día Feriado coincide con el día de la semana que debe ejecutarse el modo Día Distinto se impondrá el modo Día Feriado.

Cabe destacar que se puede extender la cantidad de Día Distintos hasta todos los días de la semana y la cantidad de Día Feriado hasta la totalidad de los días feriados nacionales.

Además de los modos de funcionamiento mencionados, existe el modo de funcionamiento Start que se ejecuta una única vez al energizarse el controlador. Este modo permite un encendido en amarillo intermitente durante un período de tiempo no seteable por el usuario, pero si modificable mediante software.

3.1.1.1 Arduino Controlador

El Arduino Controlador se encarga de dar las órdenes a la etapa de potencia para lograr el correcto encendido de las lámparas de los semáforos según el modo de funcionamiento que debe ejecutarse, entrega los valores de los estados de las lámparas al Arduino Comunicación y las fechas y horarios seteadas para cada modo al Arduino Tiempo. Recibe del Arduino Tiempo el modo en el que debe funcionar y del Arduino Comunicación los parámetros de configuración si estos fueron modificados.

Como puede observarse en el siguiente diagrama, el Arduino Controlador logra sus interacciones mediante comunicación Serial y señales digitales.

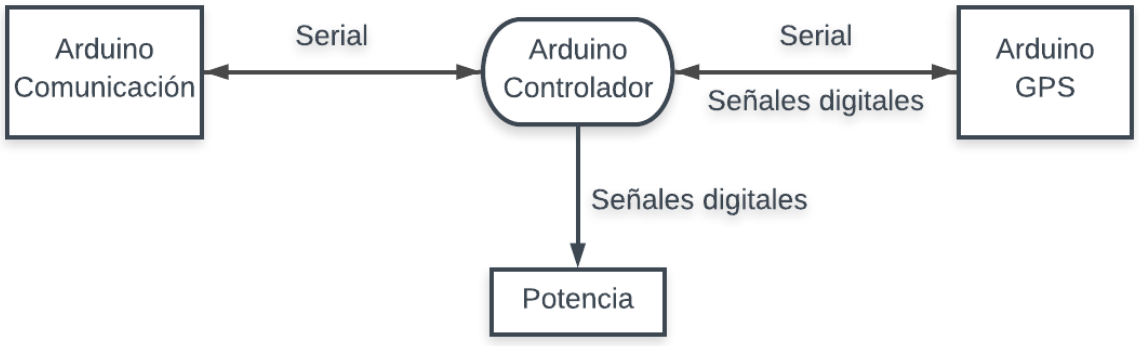

Figura 3.1.1.1.1 – Diagrama de comunicaciones del Arduino Controlador

El código de programación principal se ejecuta cíclicamente mientras se encuentre en funcionamiento y para ejecutar funciones aisladas, inesperadas, pero de alta importancia se utilizan interrupciones. Por lo tanto, el programa se divide en un programa principal, una interrupción que se ejecuta cada 1 segundo, de ahora en más Interrupción Software, y varias interrupciones que se ejecutan cuando determinados pines de entrada del controlador reciben algún tipo de evento, de ahora en más Interrupción Hardware.

A continuación, se pueden observar los diagramas de flujo de las distintas partes del programa. En la parte principal se esperan los datos recibidos mediante comunicación Serial desde el Arduino Comunicación. Estos datos son los parámetros de configuración que serán guardados en la EEPROM para que estén disponibles en cualquier momento.

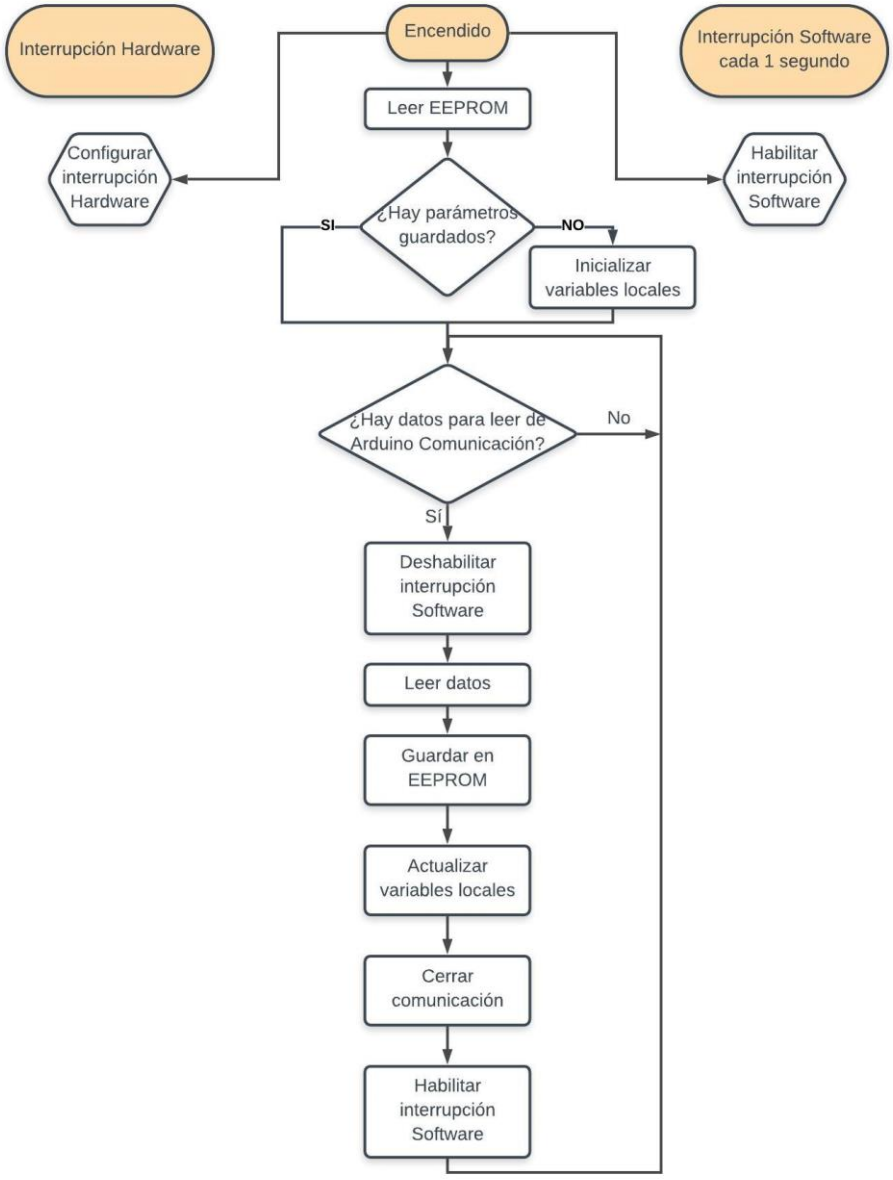

Figura 3.1.1.1.2 – Diagrama de flujo del Arduino Controlador

La interrupción mediante software, que se ejecuta cada 1 segundo, permite el correcto andar del reloj interno y la actualización de las salidas según el modo que corresponda. Es destacable qué todos los controladores que se encuentren dentro de una misma onda verde se resetean a las 00 hs para evitar así desfasajes no esperados que atenten contra la fluidez del tránsito debido a alguna deriva provocada por el reloj interno.

En la siguiente figura se menciona la consulta con una variable llamada “tiempo”, esta variable no es más que un contador interno que se incrementa cada 1 segundo gracias a la interrupción software.

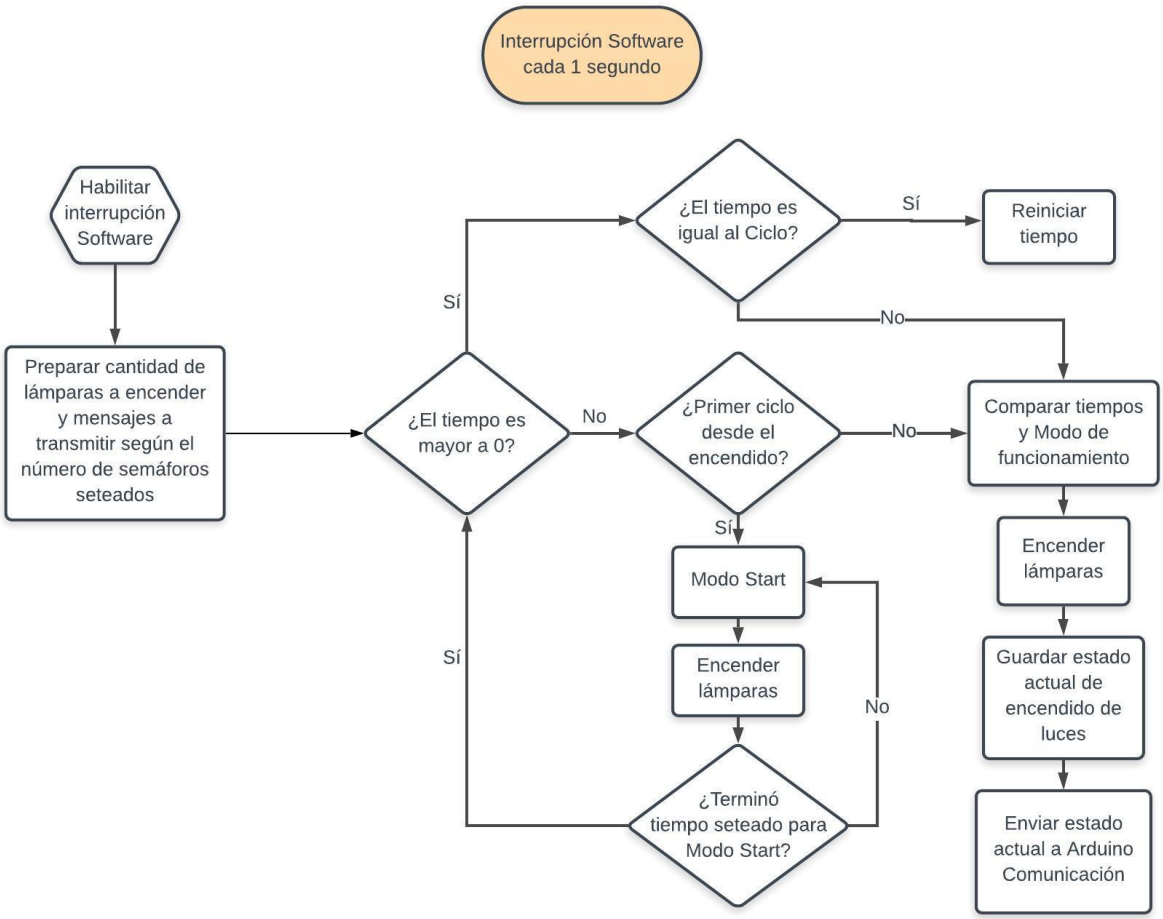

Figura 3.1.1.1.3 – Diagrama de flujo de la interrupción por Software del Arduino Controlador

Además, cuenta con interrupciones mediante Hardware. Los eventos fueron configurados para que se detecten cuando existe un flanco de la señal de LOW a HIGH. A través de este tipo de interrupciones se establecerá el modo de funcionamiento a ejecutarse como puede verse en el siguiente diagrama de flujo.

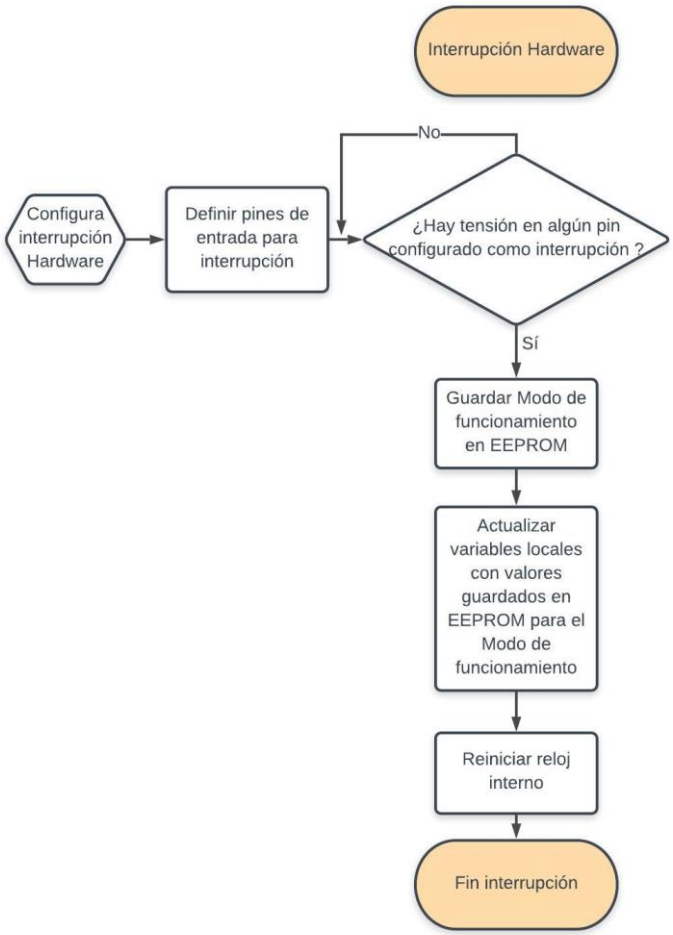

Figura 3.1.1.1.4 – Diagrama de flujo de la interrupción por Hardware del Arduino Controlador

3.1.1.2 Arduino Comunicación

El Arduino Comunicación recibe los parámetros de configuración tanto del Software Central como del panel frontal del controlador y los estados de las luces del Arduino Controlador. Además, envía el estado de las luces al Software Central, los parámetros guardados al panel frontal y los cambios de los parámetros de configuración al Arduino Controlador.

Como puede observarse el siguiente diagrama, el Arduino Comunicación logra sus interacciones mediante comunicación Serial, señales digitales y los protocolos de comunicación I2C y Ethernet.

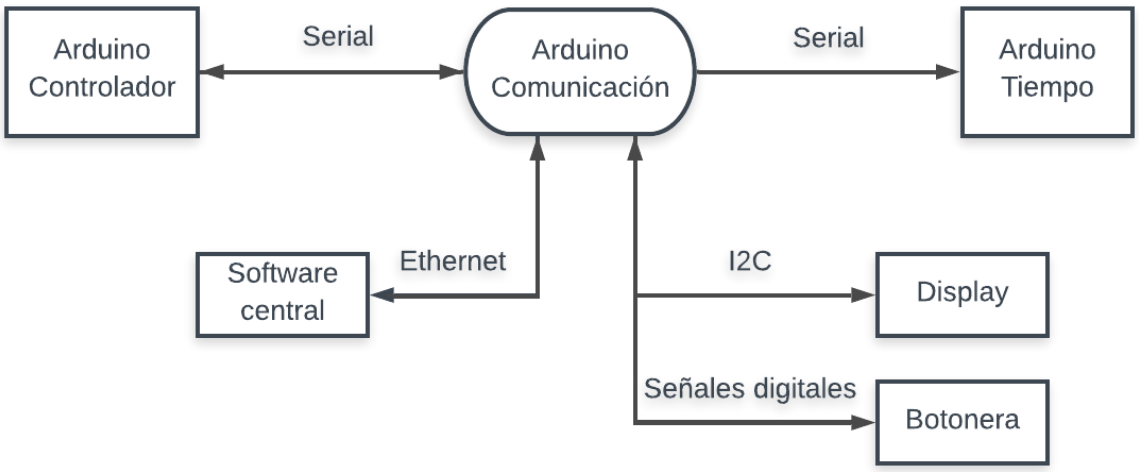

Figura 3.1.1.2.1 – Diagrama de comunicaciones del Arduino Comunicación

En el caso del Arduino Comunicación el programa se divide en un programa principal y una interrupción mediante software que se ejecuta cada 1 segundo. El programa principal está pendiente de que se establezca la conexión vía Ethernet con el software central. De esta manera, y según las indicaciones del soft, enviarle el estado de las luces en tiempo real o recibir las modificaciones de los parámetros de configuración (Ver figura 3.1.1.2.2). Además, está pendiente de la interacción del usuario con el menú principal vía botonera y display. En caso de que el botón OK del teclado fuese presionado el programa se encargará únicamente de recibir las modificaciones mediante esta vía quitándole prioridad a la comunicación Ethernet.

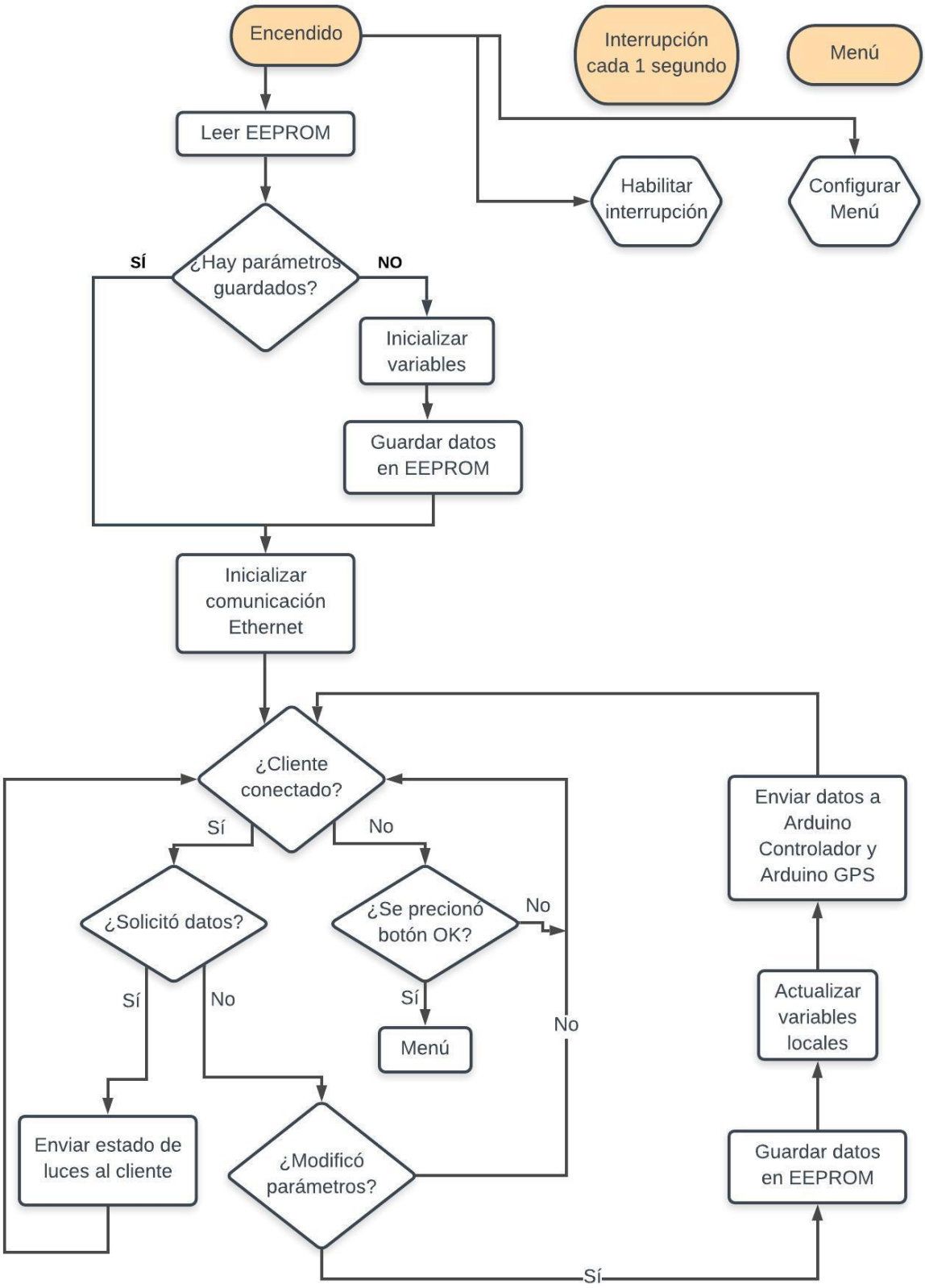

Figura 3.1.1.2.2 - Diagrama de flujo del Arduino Comunicación

La interrupción se encarga de recibir cada 1 segundo el nuevo estado de las luces que envía el Arduino Controlador. Este estado es guardado y enviado al cliente cuando se lo solicita en el programa principal.

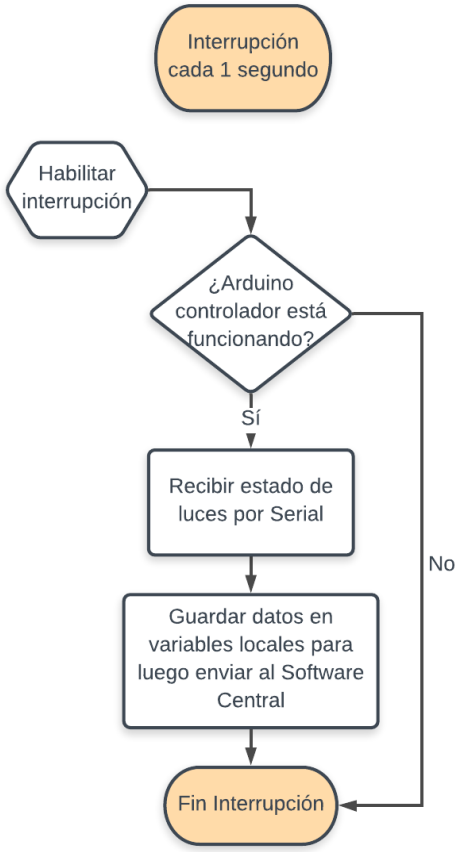

Figura 3.1.1.2.3 – Diagrama de flujo de la interrupción por Software del Arduino Comunicación

El menú principal busca una sencilla y rápida configuración del dispositivo. Para el correcto funcionamiento, mientras el menú está ejecutándose, se deshabilita la interrupción. Una vez finalizada la interacción con el usuario, ya sea que este haya guardado o no los cambios, nuevamente se habilita la interrupción como puede observarse en el siguiente diagrama de flujo.

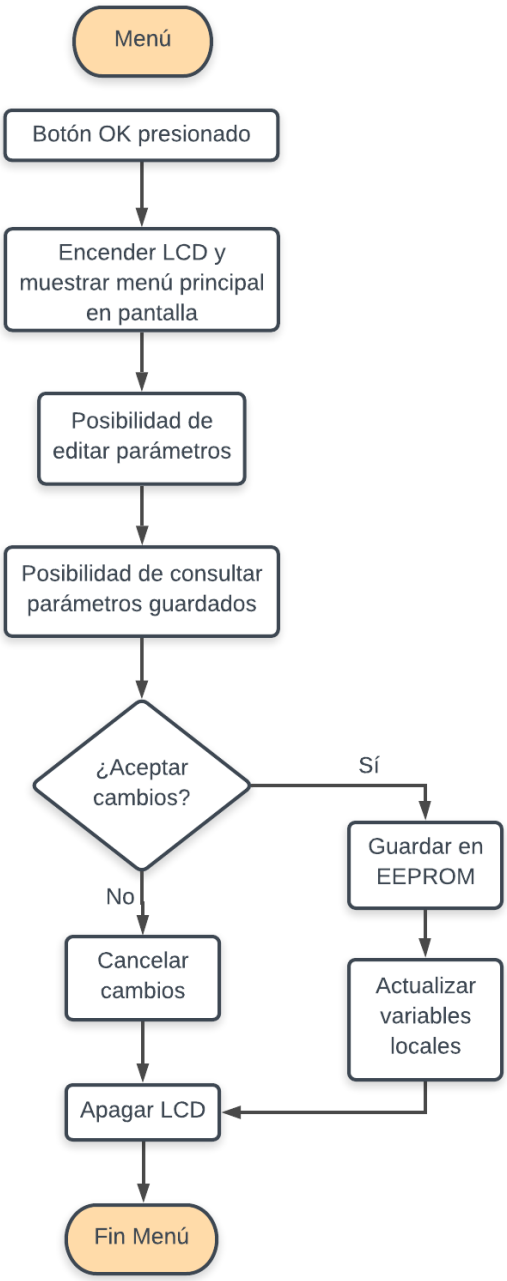

Figura 3.1.1.2.4 – Diagrama de flujo del menú

3.1.1.3 Arduino Tiempo

El Arduino Tiempo recibe del GPS los datos temporales, de los semáforos si hubo cualquier tipo de falla y del Arduino Controlador las fechas y horarios seteadas para el accionamiento de los distintos modos de funcionamiento. Envía al Arduino Controlador el modo que se debe ejecutar.

Como puede observarse el siguiente diagrama, el Arduino Comunicación logra sus interacciones mediante comunicación Serial y señales digitales.

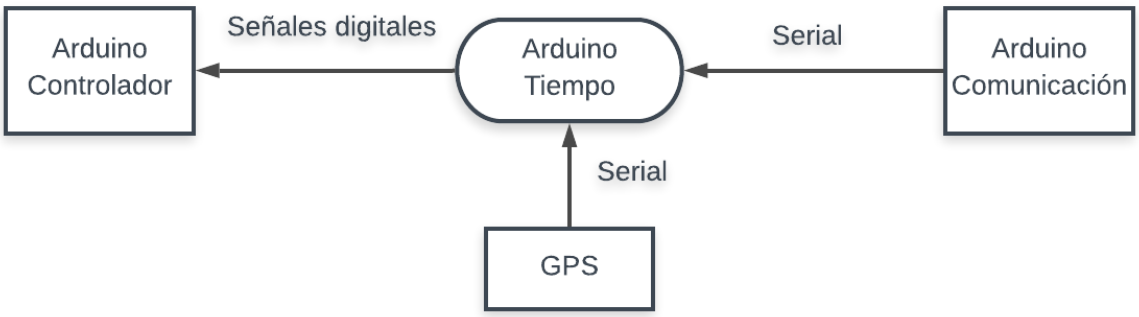

Figura 3.1.1.3.1 – Diagrama de comunicaciones del Arduino Tiempo

En este caso el programa está compuesto por un programa principal y una interrupción que se ejecuta cada 1 segundo. Durante el programa principal se consulta sobre los datos temporales como fecha de Día Distinto, horario del intervalo y fecha de Feriado, en caso de que no haya datos se inicializan las variables en 0. Además, se configura y se esperan los datos que llegan del GPS para ser guardados.

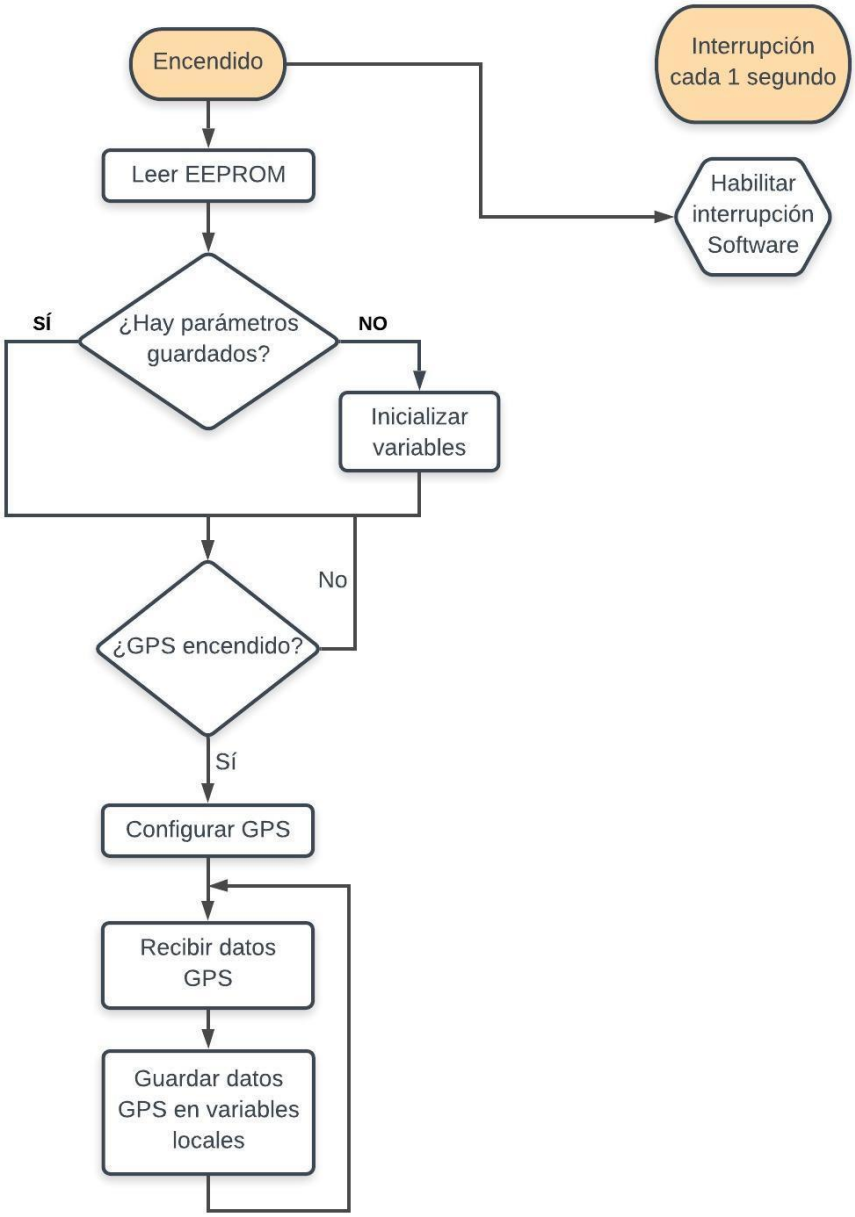

Figura 3.1.1.3.2 - Diagrama de flujo del Arduino Tiempo

Durante la interrupción se comparan los valores recibidos desde el GPS con los valores guardados y se determina en qué modo de funcionamiento debe encontrarse el dispositivo. Luego, envía una señal digital a Arduino Controlador para que este ejecute el modo correcto. Además, corrobora la presencia de datos en el Serial. En caso de que se hayan modificado los parámetros de configuración serán guardados en la EEPROM.

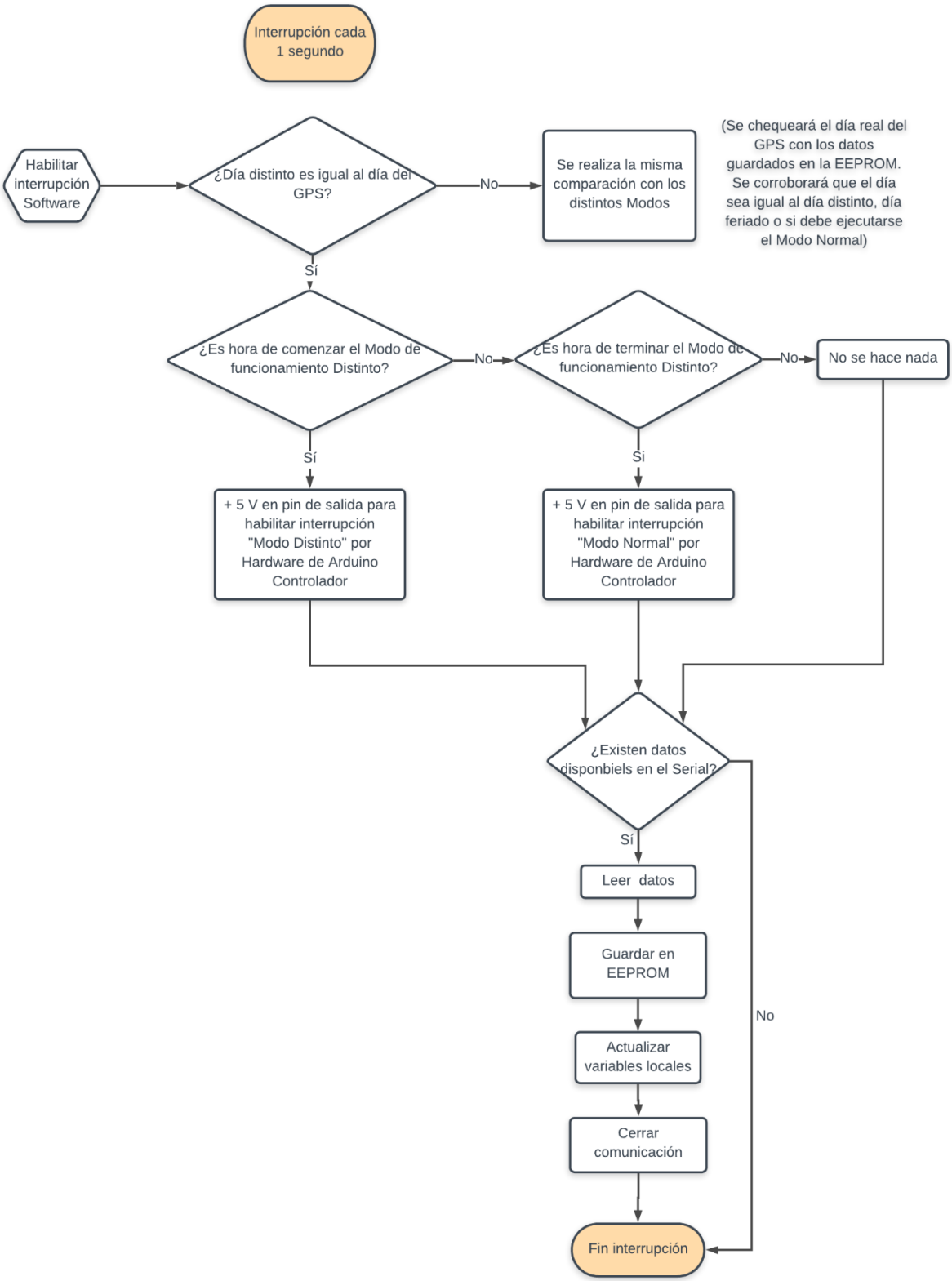

Figura 3.1.1.3.3 – Diagrama de flujo de la interrupción por Software del Arduino Tiempo

3.1.2. Electrónica de potencia

En esta parte se busca interconectar el circuito de la lógica compuesto por los arduinos, gps, display, etc y las lámparas de los semáforos. La etapa de potencia es esencial para lograr controlar voltajes y corrientes de niveles significativos y de otra naturaleza. Es decir, mediante un circuito de 5 V de corriente continua se controla la alimentación de circuitos de 220 V de corriente alterna y para ello se utilizan relés.

Los relés son dispositivos de altos tiempos de conmutación pero que, gracias a su robustez, garantizan una alta cantidad de maniobras. En pos de mantener la robustez y disminuir los tiempos de conmutación se seleccionaron relés de estado sólido que presentan tiempos de conmutación de 10 ms como máximo.

Por lo tanto, la etapa de potencia está compuesta por 12 relés, uno por cada lámpara que se desea controlar.

3.1.3. Selección de componentes principales

A medida que se desarrolló el proyecto se seleccionaron los distintos componentes comerciales teniendo en cuenta sus ventajas y desventajas.

3.1.3.1. Placa Arduino

En cada uno de los módulos se utilizó una placa Arduino Mega 2560 que, cómo se mencionó en la introducción, cuenta con 55 pines configurables para ser configurados como entradas y salidas que brindan la oportunidad de implementar un programa con aún más prestaciones. Además, dispone de una alta capacidad de procesamiento.

Una de las grandes solicitudes del proyecto hacia las placas Arduino son las múltiples comunicaciones Serial. La Arduino Mega 2560 dispone de 3 juegos de pines de naturaleza Tx Rx que permiten implementar todas las comunicaciones evitando futuros problemas al definir por software pines que no tienen ese fin especifico.

3.1.3.2. Shield Ethernet

Para lograr la comunicación con la PC se optó por el protocolo de comunicación Ethernet debido a que está basado en el protocolo TCP/IP que garantiza un grado muy elevado de fiabilidad, es de utilización a nivel mundial, es fácilmente configurable por cualquier persona que está inmersa en el mundo de las redes de datos y es aceptado por todos los dispositivos.

En vistas de realizar la comunicación a través del protocolo Ethernet se seleccionó la Shield Ethernet. Esta placa está compuesta de un chip Ethernet Wiznet 5100, un conector Ethernet estándar RJ45 y un lector de tarjeta Micro SD. Es de conexión encastrable y su implementación al sketch es por intermedio de la librería <Ethernet.h>.

Figura 3.1.3.2.1 – Shield Ethernet W5100

3.1.3.3. Display LCD 20x4 + Módulo I2C

Para generar la interacción con el usuario se adoptó que la presentación del menú de configuración de parámetros sea a través de un display LCD 20x4 (20 caracteres y 4 líneas). Para su correcto funcionamiento se implementó la librería <LiquidCrystal.h>.

Cómo puede observarse en la siguiente imagen, el display LCD 20x4 posee 16 pines de conexión y para la transmisión de mensajes debe darse una indicación distinta a cada uno de los 8 pines de datos. Para simplificar el cableado, conexionado y código de programación se implementó el módulo I2C. Este módulo necesita referencia de de tensión y la conexión al pin Serial Data (SDA) y al pin Serial Clock (SCL). Por lo tanto, la transmisión de datos, utilizando el protocolo de comunicación Inter-Integrated Circuit (I2C), paso de 8 pines digitales a simplemente dos. El pin SDA es por el cual se transmiten los datos en ambas direcciones y el pin SCL permite que viaje la señal de reloj. La librería para manejar el bus I2C que se utilizó fue <Wire.h>.

Figura 3.1.3.3.1 – Display LCD + Módulo I2C

3.1.3.4. Módulo GPS

Como se ha mencionado a lo largo de este capítulo, para la correcta sincronización de los controladores es necesario que todos tengan una misma referencia del tiempo. En virtud de lo ante dicho se optó por la recepción de datos de tiempo del sistema de posicionamiento global a través del módulo comercial A7 Ai Thinker GPS+GPRS. Este dispositivo está integrado por un módulo GPS, un módulo GPRS y un módulo GSM. En este caso será de utilidad únicamente el módulo GPS.

Figura 3.1.3.4.1 – Módulo A7 Ai Thinker GPS+GPRS

Para el correcto funcionamiento se debe abrir la comunicación y solicitar por medio de comandos AT la recepción de datos y la frecuencia de envío de estos, luego una vez que no se desea recibir más datos se debe cerrar la comunicación cómo se puede observar en el código de programación del Arduino Tiempo a través de los comandos llamados A, B, C, D y E. Cabe destacar que el módulo envía los datos crudos, es decir, envía los datos en el formato estándar de comunicaciones marítimas llamado NMEA (National Marine Electronics Association). Este protocolo envía los datos en varias líneas separadas por un caracter de fin de línea y separados entre comas. Por lo tanto, para facilitar la consulta se implementó la librería <Tiny.GPS++.h> que se encarga de desglosar los datos y entregarlos en un formato simple.

```
$GPGGA,022617.00,3329.73891,S,06003.84668,W,1,08,1.43,9.1,M,16.7,M,,*5B
$GPGSA,A,3,11,32,13,01,20,31,17,23,,,,,2.62,1.43,2.19*0D
$GPGSV,3,1,09,01,84,300,22,11,57,347,33,13,18,310,09,17,16,235,26*7F
$GPGSV,3,2,09,19,08,353,08,20,55,216,16,23,50,297,21,31,39,107,32*78
$GPGSV,3,3,09,32,58,140,20*4B
$GPGLL,3329.73891,S,06003.84668,W,022617.00,A,A*68
$GPRMC,022618.00,A,3329.73910,S,06003.84673,W,0.243,,230214,,,A*7F
$GPVTG,,T,,M,0.243,N,0.450,K,A*27
```

Figura 3.1.3.4.2 – Ejemplo de datos crudos del protocolo NMEA

3.1.3.5. Level Shifter

A la hora de la implementación de módulo GPS hay que considerar que, si bien su tensión de funcionamiento es 5 V, la transmisión de datos se realiza en 3,3 V. Por lo tanto, para lograr una correcta comunicación con la placa Arduino es necesario

adaptar los niveles de tensión. Para este fin se utilizó el módulo comercial integrado que se observa en la siguiente figura.

Figura 3.1.3.5.1 – Módulo Level Shifter

Este adaptador puede convertir líneas de entrada y salida de 5 V a 3,3 V y viceversa. En total cuenta con 4 líneas, dos del tipo Rx input de 5 V asociadas a las Rx output de 3,3 V y dos del tipo Tx input de 5 V asociadas a las Tx output, todas las líneas son bidireccionales.

3.1.4. Librerías

Además de las librerías mencionadas para controlar los módulos comerciales se utilizaron otras. A continuación, se presenta a modo de resumen una tabla con todas las librerías utilizadas y sus funciones principales.

Libería	**Descripción**	**Funciones utilizadas**	**Descripción**
Ethernet.h	Permite utilizar el Arduino Ethernet Shield.	Begin()	Inicializa la librería Ethernet
		IPAdress()	Define una dirección IP
		EthernetServer()	Crea un servidor que escucha por las conexiones entrantes del puerto definido
		Server.write()	Escribe datos a todos los clientes conectados al servidor
		EthernetClient()	Crea un cliente que se conecta a una determinada IP y puerto
		Connect()	Conecta a una IP y puerto Especificado
		Client.read()	Lee el siguiente byte recibido desde el servidor.
		Client.flush()	Borra todos los bytes que han sido escritos en el cliente, pero no leídos.
LiquidCrystal_i2c.h	Adicionalmente a la librería Wire, permite la comunicación con el módulo LCD a través del módulo I2C	LiquidCrystal()	Crea una variable del tipo LiquidCristal
		setCursor()	Posiciona el cursor en un byte de la pantalla
		Backlight()	Permite encender la luz del LCD
		Print()	Permite escribir una cadena de caracteres.
		Blink() / NoBlink()	Habilita o deshabilita cursor en forma de bloque
TinyGPS++.h	Permite parsear los datos crudos que llegan desde el GPS.	GPSrespuesta	Datos crudos que envía el GPS.
		GPS.time.isValid()	Devuelve true si tiene un valor de horario correcto. False si no es así.
		GPS.time.hour() GPS.time.minute() GPS.time.second()	Devuelve los valores de hora minuto y segundos para lograr el formato HH:MM:SS
		GPS.date.day() GPS.date.month() GPS.date.year()	Devuelve los valores de día mes y año para lograr el formato DD/MM/AAAA.
Eeprom.h	Permite leer y escribir los bytes reservados como memoria eeprom.	Read()	Devuelve el valor guardado en el byte de referencia.
		Update()	Permite consultar si el valor guardado en la dirección de la Eeprom es igual al valor a escribir para no sobreescribir el mismo valor debido a que la memoria tiene una vida útil finita.
SPI.h	Es necesaria para utilizar el bus de comunicación SPI en el Ethernet Shield		
Wire.h	En necesaria para toda comunicación con dispositivos I2C.		

Tabla 3.1.4.1 – Librerías utilizadas en la programación y sus funciones principales

3.1.5. Funciones

En programación, una función es un grupo de instrucciones con un objetivo particular y que se ejecuta al ser llamada desde otra función o procedimiento. Una función puede llamarse múltiples veces e incluso llamarse a sí misma.

Las funciones pueden recibir datos desde afuera al ser llamadas a través de los parámetros y pueden entregar un resultado.

Una función tiene un nombre y un conjunto de instrucciones que son ejecutadas cuando se llama a la función. Son funciones setup() y loop() de las que ya se ha hablado anteriormente.

Las funciones de usuario pueden ser escritas para realizar tareas repetitivas y para reducir el tamaño de un programa. Segmentar el código en funciones permite crear piezas de código que hacen una determinada tarea y volver al área del código desde la que han sido llamadas.

Para lograr un código más simple, ordenado y práctico se crearon distintas funciones que son mencionadas a continuación.

3.1.5.1 Calendario()

El usuario puede elegir que un día de la semana, ejemplo martes, el dispositivo funcione con otros valores de tiempos. Para ello es necesario saber el día de la semana en función de los datos que nos brinda el GPS. Por lo tanto, se crea la función calendario del tipo integer (entero) que tiene como parámetros de entrada el día, el mes y el año y devuelve el día de la semana de lunes a domingo.

Esta función esta implementada en el Arduino GPS y permite mediante un algoritmo saber qué día de la semana es cualquier fecha. El algoritmo consiste en una cuenta compuesta de 4 coeficientes A, B, C y D que devuelve un número del 0 al 6 (domingo a sábado). [19]

Para entender el funcionamiento adoptemos como ejemplo la fecha 29/08/2019

Coeficiente A tiene en cuenta el siglo de la fecha en cuestión. El algoritmo que se implementó es únicamente para los años 2000-2099 por lo tanto A=0.

Coeficiente B tiene en cuenta el año de la fecha en cuestión. Este coeficiente adopta el valor de la suma entre los últimos dos dígitos del año y el entero menor que resulta de dividir los últimos dos dígitos del año en 4. Ejemplo: Al año 2019 se divide 19 por 4, resultando ser 4,75 por lo tanto B toma el valor de 19+4=23.

El coeficiente C tiene en cuenta los años bisiestos. El valor de C depende de si el año es bisiesto, y si el mes es enero o febrero, dadas estas dos condiciones C=-1, cualquier otro caso el valor de C es 0. En nuestro ejemplo C=0.

El coeficiente D tiene en cuenta el mes siguiendo los valores de la siguiente tabla:

Mes	Valor D
Enero	6
Febrero	2
Marzo	2
Abril	5
Mayo	0
Junio	3
Julio	5
Agosto	1
Septiembre	4
Octubre	6
Noviembre	2
Diciembre	4

Tabla 3.1.5.1.1 – Valores del coeficiente D para cada mes del año

En nuestro ejemplo D=1.

Por último, hay que sumar los 4 coeficientes y el valor del día. Ese valor puede estar desfasado y ser mayor a 6, por lo tanto, hay que restarle 7 hasta que su valor este comprendido entre 0 y 6.

$$SUMA = A + B + C + D + Día = 0 + 23 + 0 + 1 + 29 = 53$$

$$Día\ de\ la\ semana = 53 - 7 - 7 - 7 - 7 - 7 - 7 - 7 = 4$$

Dónde los valores de los días de la semana son los siguientes y el 29/08/2019 que según el algoritmo es Jueves lo cual es correcto.

Día	Posición de la semana
Domingo	0
Lunes	1
Martes	2
Miércoles	3
Jueves	4
Viernes	5
Sábado	6

Tabla 3.1.5.1.2 – Valor de cada uno de los días de la semana

3.1.5.2 byteToInt()

Es una función del tipo uint8_t que recibe un parámetro del tipo byte y devuelve el mismo parámetro traducido al tipo uint8_t. Esta función fue necesaria para realizar la transferencia de datos a través de la comunicación Serial entre Arduinos. Es utilizada en todos los Arduinos.

3.1.5.3 Imprimir Display()

Son múltiples funciones que son del tipo void, no tienen parámetros de entrada ni de salida, se encargan de limpiar la pantalla del Display y presentar los datos que correspondan. Son implementadas en el Arduino Comunicación.

3.1.5.4 SendBytes()

Son funciones del tipo void que se encarga de enviar el parámetro de entrada a través del Serial que corresponda según el destino. Existen dos funciones, una para el Serial 1 y otra para el Serial 3 y son utilizadas en los tres Arduinos.

3.1.5.5 Boton()

Es una función del tipo int que tiene como parámetro de entrada el pin donde está configurado el botón del menú y como parámetro de salida un 1 o un 0. Esta función se creó para evitar, mediante software, los efectos de rebote que puede tener la señal de un pulsador. Por lo tanto, se ingresa a la función y se consulta el estado del pulsador, si el estado es 1 se espera 500 ms y se vuelve a consultar, si sigue siendo 1 se acepta la pulsación y la función devuelve un 1, caso contrario devuelve un 0. Se utiliza en el Arduino Comunicación.

3.1.5.6 EncenderLuces()

Es una función del tipo void que tiene como parámetros de entrada un array booleano llamado “estado” y un String llamado “datosenviar”. Estado es un conjunto de 1 y 0 que definen cómo deben ser encendidas las lámparas del semáforo y datosenviar tiene el mismo sentido, pero es simplemente un carácter que luego de enviado al software de la PC será traducido en el mismo array que “estado”. Esta función se implementó en el Arduino Controlador.

Por último, según se encendió cada lámpara se sensa los valores entregados dispositivo de detección fallas.

3.1.5.7 Modo()

Son múltiples funciones del tipo void, sin parámetros de entrada ni de devolución que se encargan de definir el estado de las variables locales según el modo de funcionamiento que corresponde. Esta función se ejecuta por medio de una interrupción por Hardware. Consulta el estado de las variables para el modo en cuestión guardadas dentro de la EEPROM para sobreescribir las variables locales. Estas funciones están implementadas en el Arduino Controlador.

3.1.6. Sensor de lámpara

Para el control de los estados de las lámparas y la detección de anomalías se diseñó un sensor que entrega +5 Vcc cuando la lámpara se encuentra encendida y 0 Vcc cuando no. De esta manera se logra detectar lámparas rojas quemadas y lámparas verdes en simultaneo.

En la siguiente figura se puede observar el conexionado del sensor. Se utilizó un puente de diodos para rectificar la onda, el capacitor C1 para eliminar el ripple de la señal, un optoacoplador modelo 4N26 destinado a aislar eléctricamente los circuitos de potencia y control y por último un capacitor C2 y una resistencia R1.

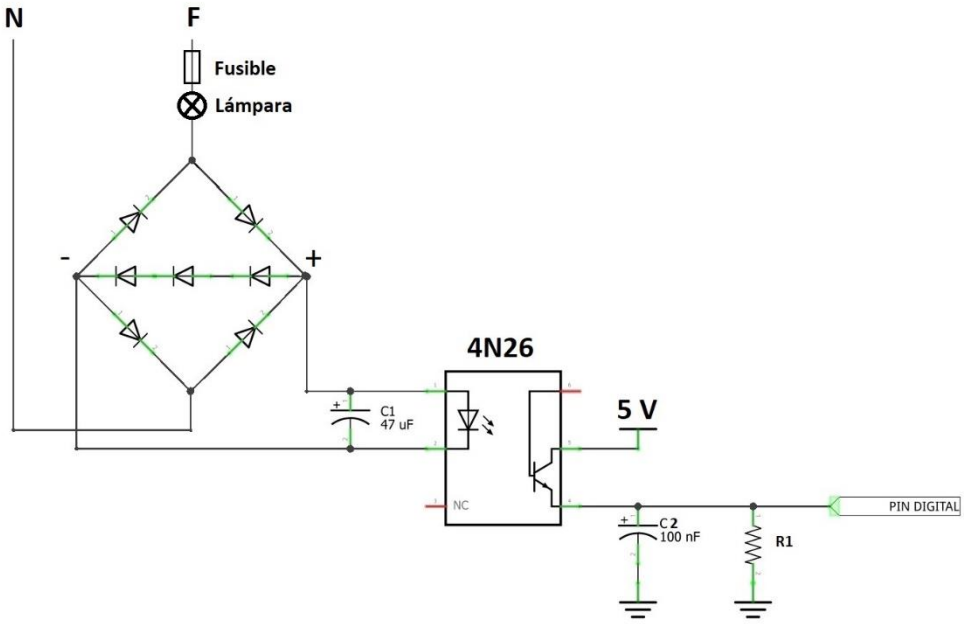

Figura 3.1.6.1 – Esquema de conexión del sensor de lámparas

El puente de diodos cuenta con una rama compuesta de 3 diodos en serie destinados a establecer un valor de tensión entre los terminales 1 y 2 del optoacoplador que cómo máximo será 3,3 V (el fabricante garantiza una caída de tensión de 1,1 V al circularle 1 A, corriente máxima que soportan los diodos).

De esta manera se logra detectar a través del pin digital al que se conecte el sensor si la lámpara se encuentra encendida o no.

3.1.7. Conexionado

En función de lo antepuesto se presenta el conexionado entre los distintos elementos que componen el controlador.

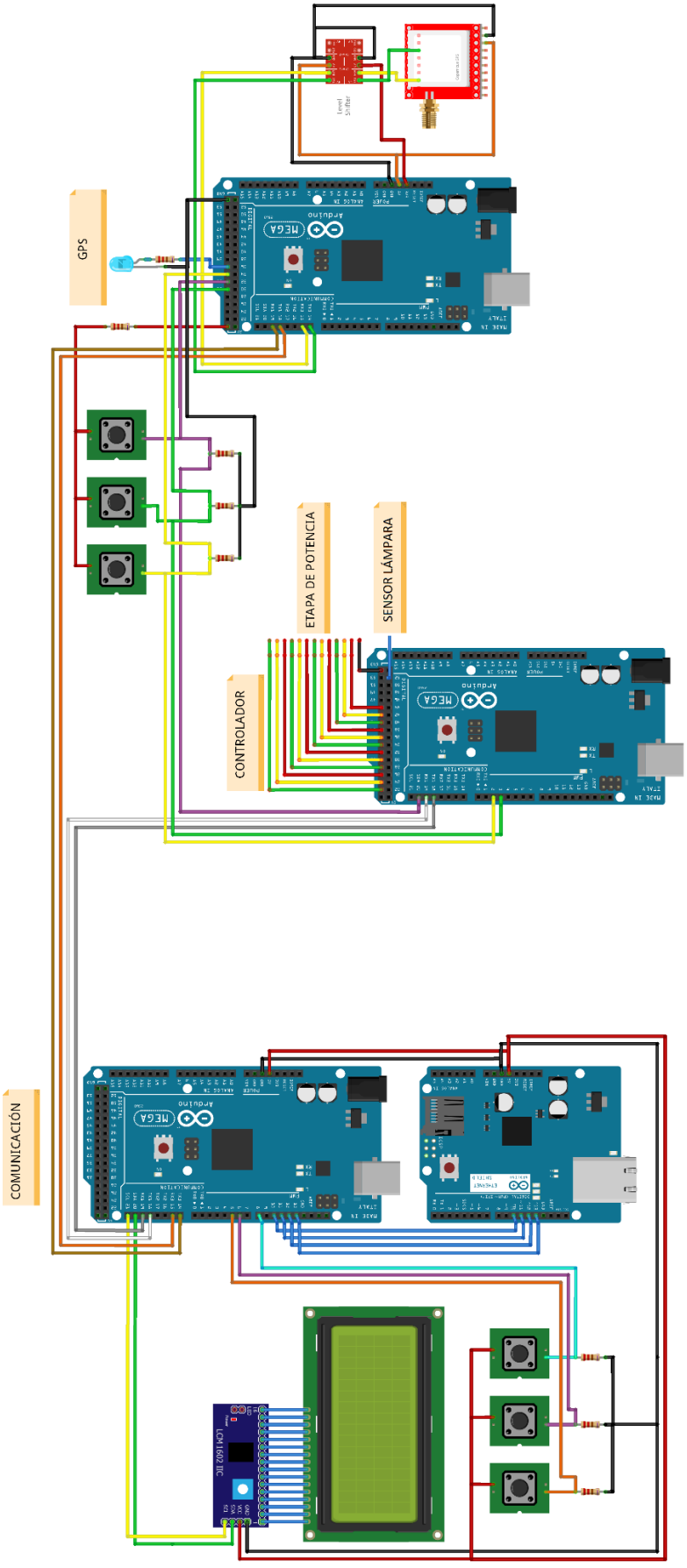

Figura 3.1.7.1 – Esquema de conexión de la etapa lógica

3.2. Fuente de alimentación

En vistas de las necesidades del circuito y de los requerimientos impuestos por las normas, se diseñó la fuente de alimentación. A continuación, se detallan las especificaciones técnicas.

- Tensión de entrada: 120 – 220 Vac
- Tensión de salida: 5 Vcc
- Corriente máxima de salida: 1 A
- Tiempo de suministro sin tensión de entrada: 1 hr mínimo.

Como puede observarse en la Figura 3.1.3.1, la fuente se basa en un transformador de tensión, dos circuitos integrados reguladores de tensión de la familia 78XX y distintos componentes electrónicos como resistencias, capacitores, relés y diodos.

Los objetivos principales de la fuente de alimentación son obtener 5 Vcc estables a la salida y garantizar siempre el suministro, aunque la fuente de energía haya desaparecido (sin presentar huecos de tensión). A continuación, se especifican los pasos tenidos en cuenta para lograr lo anteriormente dicho.

El ingreso de tensión desde la red de distribución eléctrica es 220 V y 50 Hz. Para reducir su valor y lograr una tensión continua se utilizó un transformador de relación de transformación 220/24 V seguido de un puente de diodos (ver figura 3.2.1). Por lo tanto, para el rango de valores de tensión de entrada especificados anteriormente se obtienen los siguientes valores:

Vin 220 V -> Vp in 311 V

$$\frac{311\ V}{9{,}167} = 33{,}95\ V$$

Vin 120 V -> Vp in 169,7 V

$$\frac{169{,}7\ V}{9{,}167} = 18{,}51\ V$$

Asumiendo una caída de 0,7 V por cada diodo y sabiendo que en cada semiciclo conducen únicamente dos diodos se obtienen 32,55 V y 17,11 V como valores máximos y mínimos de tensión continua respectivamente. A continuación, la onda rectificada pasa por un primer capacitor (C1) para disminuir el ripple y lograr mejor calidad de tensión continua.

Hasta el momento la tensión es continua pero su valor es variable siguiendo las oscilaciones que presenta la energía suministrada por la distribuidora eléctrica. Para ello, se utiliza el circuito integrado regulador de tensión 7815. Este integrado garantiza a la salida un valor de 15 Vcc a pesar de que la tensión de entrada fluctué. Sin embargo, la tensión de entrada debe superar, al menos, en 2 volts la tensión de salida. Condición que, hasta un valor de 120 Vac está asegurado. El fabricante de esta familia de integrados recomienda colocar un capacitor a la entrada, para filtrar transitorios o picos no deseados, y uno a la salida para filtrar el ripple nuevamente. Por lo tanto, en A se obtiene un valor de tensión de 15 Vcc.

Luego del punto A, continuando el circuito, aparecen 2 relés; cabe destacar que el relé R1 es de 12 V mientras que el relé R2 es de 6 V.

La primera situación se da cuando en A existe 15 V, por lo tanto, la bobina de R1 se encuentra excitada interconectado 2 con 3 y llegando a D2 la tensión. D2 conduce permitiendo tener en B, aproximadamente, 14,4 V contemplando la caída de tensión del diodo. Los capacitores C5 y C7 cumplen la misma función que C1 y C2. En este caso, la tensión es superior a la tensión de ruptura del diodo Zener Dz, excitando la base del transistor BC548 que permite la referencia a masa de la bobina de R2 interconectándose 2 con 3 de dicho relé permitiendo la constante carga de la batería.

En el momento que se produce un corte de suministro de energía eléctrica el punto A se queda sin referencia de potencial. Es momento de alimentar el controlador a través de la batería de reserva. En el instante en qué la tensión baja de 12 V el relé R1 conmuta a su posición des energizada permitiendo que su salida esté conectada a su terminal 4 y por ende al terminar 3 del relé 2. Si la caída de tensión fuera abrupta, el relé R2 que es de 6 V también conmutaría no permitiendo que el circuito sea alimentado por la batería. Para que esto no suceda se debe garantizar una tensión mayor a 10.5 V en B y así siempre superar la tensión de ruptura del Zenner. Para ello está el capacitor C4, un capacitor de 5000 uF cargado a una tensión de 14.4 V aproximadamente y de gran capacidad para poder soportar este transitorio.

Como se mencionó anteriormente, el capacitor debe garantizar 10.5 V en B durante el tiempo de conmutación del relé R1, que según fabricante es un tiempo máximo de 10 ms. Suponiendo que el capacitor se encuentra cargado a 14,4 V, en un circuito RC, y que en 10 ms debe llegar a 10,5 V se obtiene que:

$$V\ (t) = \ V_o * e^{\ -\frac{t}{RC}} \tag{3.2.1}$$

$$10{,}5\ V = 14{,}4\ V\ e^{\ -\frac{10\ ms}{R*5000\ uF}} \rightarrow R = 6{,}33\ \Omega$$

Dentro de un circuito RC:

$$I_o = \frac{14{,}4\ V}{6.33\ \Omega} = \ 2{,}27\ A$$

$$I\ (10\ ms) = \ I_o * e^{\ -\frac{t}{RC}} = 1{,}65\ A$$

Se puede observar que, llegando a los 10,5 V en los 10 ms (peor condición) siempre se va a suministrar una corriente mayor a la necesaria.

La fuente le brinda al controlador la autonomía suficiente para que ante un corte de suministro se reporte la falla a un centro de control.

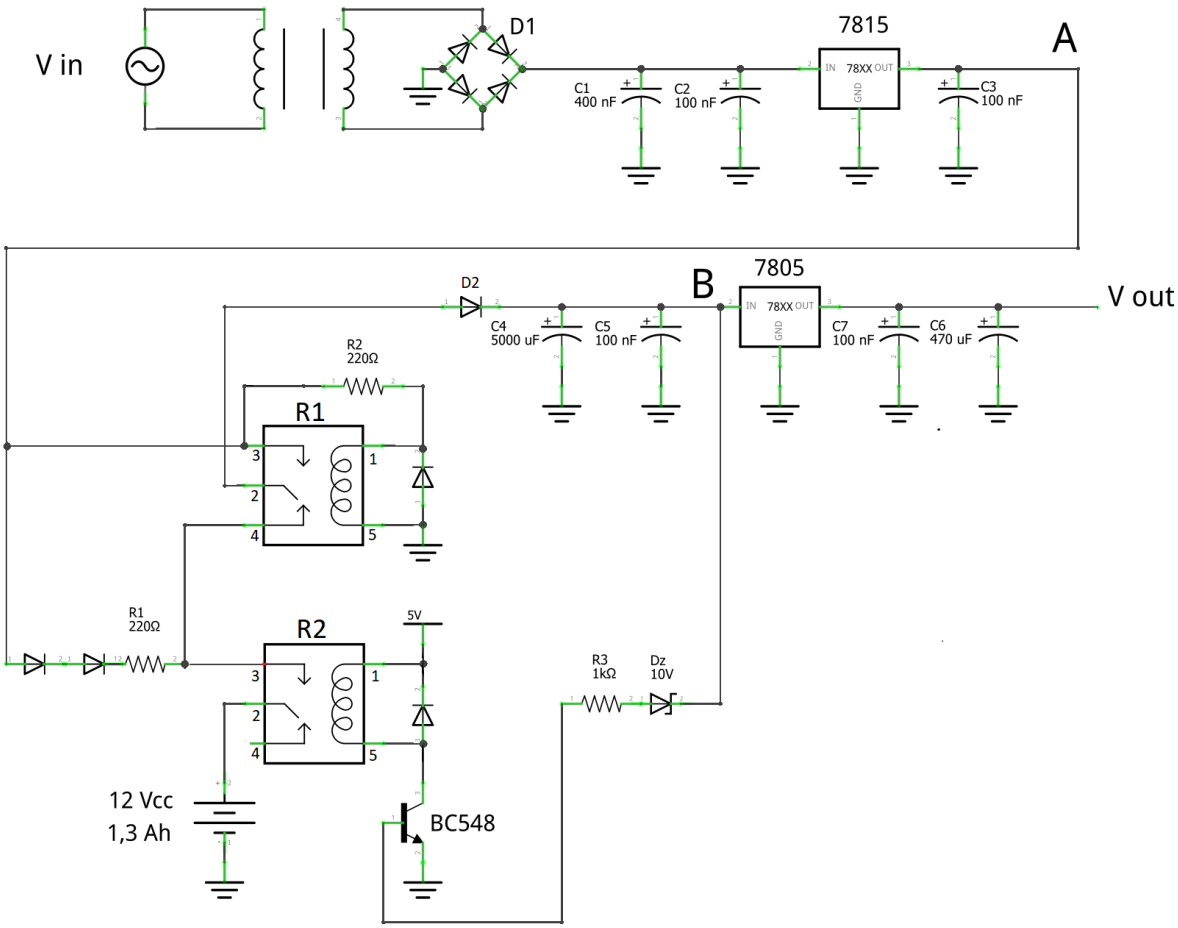

Figura 3.2.1 – Esquema de conexionado del circuito de la fuente de alimentación

4. SOFTWARE DE MONITOREO Y CALCULADOR DE ONDA VERDE

A lo largo de los capítulos se ha nombrado la vinculación del controlador con el Software Central por medio de comunicación Ethernet. Este programa está dividido en dos grandes secciones. La primera permite, en tiempo real, el monitoreo cualquiera de los semáforos instalados en la red, el control de los parámetros relevantes y la detección de anomalías de funcionamiento (de acá en adelante “software de monitoreo”). La segunda, otorga la posibilidad de diseñar ondas verdes adaptadas a las necesidades dando como resultado los parámetros que se deben setear en cada uno de los semáforos para lograr su correcto funcionamiento (de acá en adelante “software onda verde”).

A lo largo del capítulo se describirá el código de cada uno de los programas y su implementación. La interfaz con el usuario y su correcto uso será descripta en el Capítulo 5 - Manual de Usuario.

4.1. Software de monitoreo

Se desarrolló un programa que sea capaz de monitorear el estado de las señales luminosas a distancia. Esto tiene como objetivo tener un control visual rápido sobre la cantidad de semáforos que maneja el controlador, el modo en el que se encuentra funcionando ya sea en modo control o en intermitente amarillo, y si este último fuese el caso, si el controlador ha sufrido alguna falla y así lograr una pronta normalización del servicio. Además, se agregó una solapa que permita leves modificaciones que fuesen de utilidad generarlas a la distancia en sus parámetros básicos, estos son: cantidad de semáforos, el tiempo de verde y de amarillo de cada uno de ellos.

En la figura 4.1.1 se puede observar la fracción del código dedicada a la obtención, análisis y muestra de datos enviados por el Arduino Comunicación. Se establece la comunicación con la IP establecida al controlador que se quiera supervisar y comienza la comunicación. Los datos recibidos varían desde la “a” hasta la “p” en orden alfabético, incluyendo la “x”. Cada uno de estos datos es procesado y determina el estado de luces que se encuentra encendido y si hubo una falla en el controlador.

En la figura 4.1.2 se puede observar la fracción del código dedicada al ingreso de datos por parte del usuario, su procesamiento y posterior envío al Arduino Comunicación.

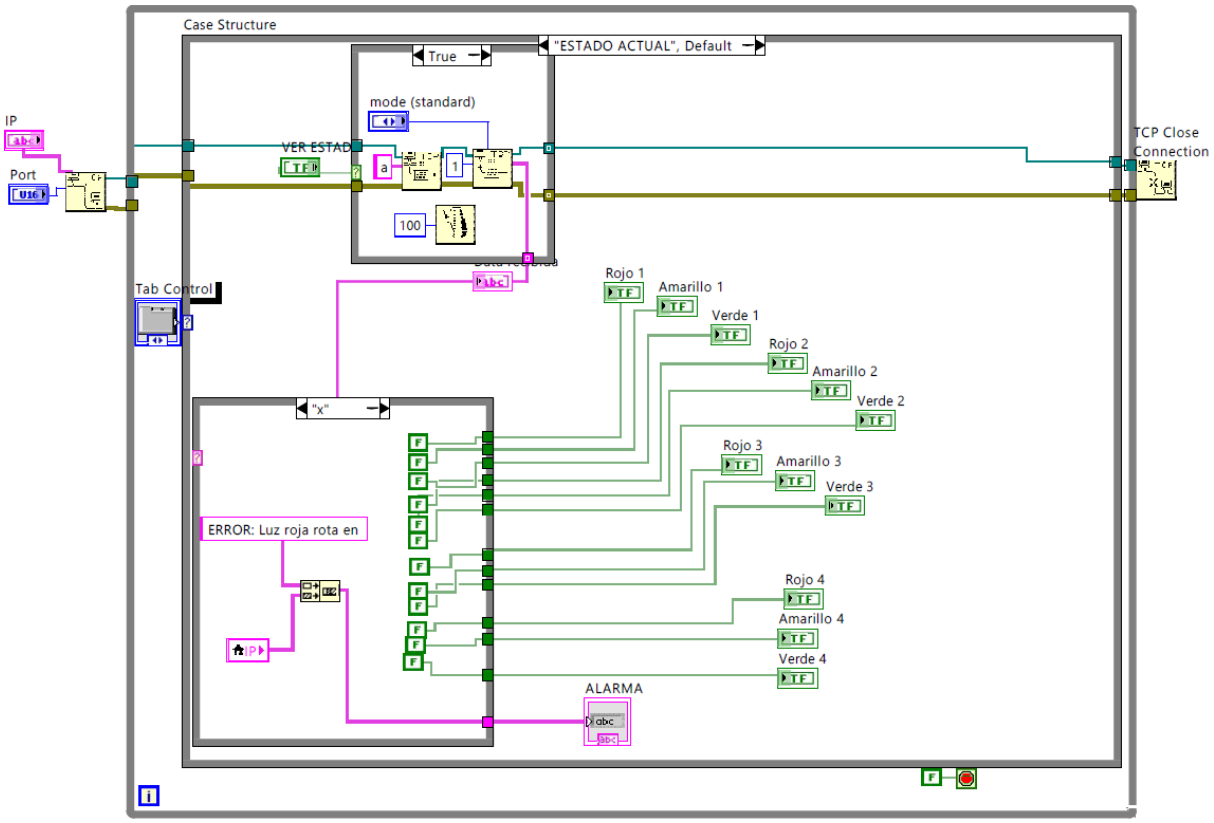

Figura 4.1.1 – Software de monitoreo – Panel visual

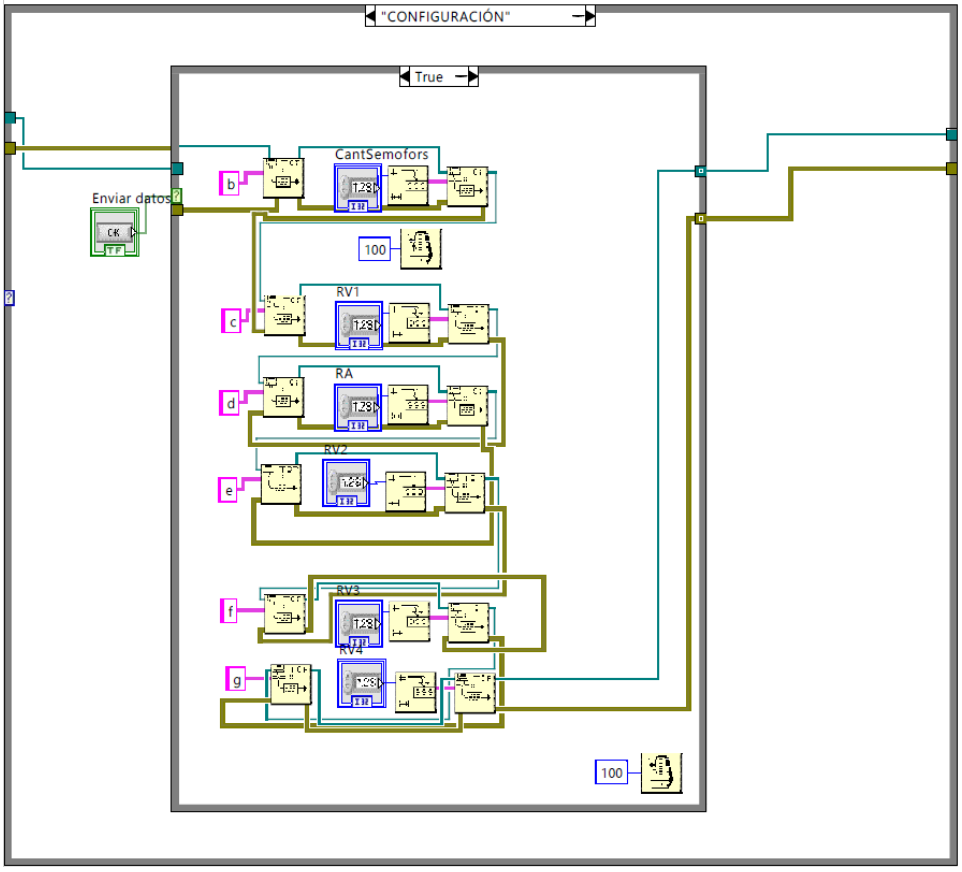

Figura 4.1.2 – Software de monitoreo – Parte de control

4.2. Software onda verde

Ante la problemática planteada en la introducción, dónde calcular los tiempos de desfasajes de los semáforos coordinados es engorroso y requiere mucho tiempo dedicado a prueba y error o softwares muy sofisticados, surgió la necesidad de adicionarle a la interfaz remota un calculador de onda verde.

Se busca que el usuario cuente con una herramienta que le facilite la programación de los controladores con el fin de lograr los resultados de la onda verde de manera más eficaz.

Brinda la gran posibilidad de extraer los desfasajes entre semáforos que se encuentren a cualquier distancia entre sí, dónde la velocidad de la onda puede ser de hasta 60 km/h, dónde la cantidad de autos que ingresen en la onda sea elección del usuario y, lo más importante, dónde pueda optar por tener tránsito fluido en ambas direcciones de circulación y con velocidades diferentes. El usuario puede modificar a gusto cada uno de los parámetros seleccionados a medida que estos se reflejan en un gráfico de rápida comprensión.

De la combinación del estudio de las arterias, del ingenio del usuario y de la practicidad del software se logra obtener el mejor resultado de coordinación de semáforos.

Para lograr la implementación desea, se debió trabajar con varios programas auxiliares (llamados subVI). Estos programas, gracias a la particularidad de LabVIEW, pueden invocarse en rutinas más grandes funcionando como nodos, dónde ingresan ciertos parámetros y devuelven otros.

El código puede dividirse en tres grandes partes. En primer lugar, se encuentra dónde se tratan todos los datos de ingreso y se establecen las propiedades para los controles numéricos mostrados en el panel frontal (ver Figura 4.2.1). En segundo lugar, se realizan las cuentas de cada uno de los puntos a graficarse, se concatenan los datos y se establecen las propiedades estéticas del gráfico (ver Figura 4.2.2). En tercer lugar, se procesan los datos para mostrar los desfasajes correspondientes a los datos ingresados y procesados en la primera parte del código (ver Figura 4.2.3). Además, se establecen las propiedades de los controles numéricos mostrados en el panel frontal y se evalúa si el usuario seleccionó la opción de que la onda verde sea en dos sentidos, en caso de ser afirmativo, se habilita el ingreso de datos de la segunda onda verde y se procesan los datos para mostrar los respectivos desfasajes y tiempos de verde mínimos (ver Figura 4.2.4).

En la primera parte del código fue necesario implementar un subVI llamado Rectas. Como parámetros de ingreso tiene cada una de las distancias de las intersecciones en metros, el ciclo en segundos, la velocidad en m/s, el tiempo verde del semáforo de referencia y el ancho de la onda verde en segundos. No devuelve ningún dato, sino que genera variables globales (pueden ser utilizadas en cualquier VI simplemente invocándolas) con cada uno de los puntos en los que se cruzan todas las rectas representadas en el gráfico. Estos puntos son aquellos que permitirán trazar los gráficos más fácilmente y calcular los offset. Su código se puede observar en la Figura 4.2.5.

En la última parte del código se implementó un subVI llamado Offset Tverde. Como parámetros de ingreso tiene las variables globales generadas por el subVI Rectas, el ciclo en segundos y el número de elemento que permite la posición dentro de un array que selecciona la intersección en cuestión. Como datos de salida tiene el desfasaje (Offset) y el tiempo verde mínimo para dicha intersección. Su código se puede observar en la Figura 4.2.6.

A continuación, se muestra el código general y cada uno de los subVI utilizados.

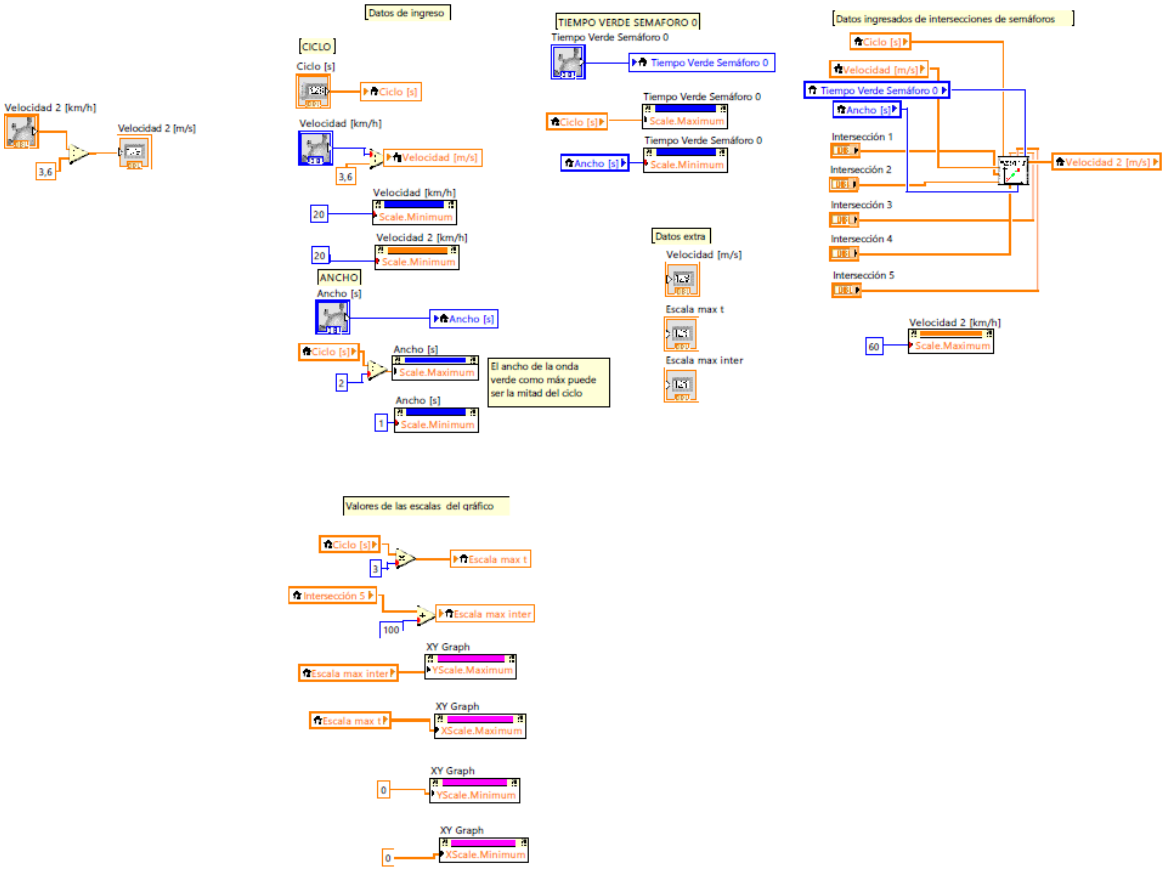

Figura 4.2.1 – Código general Software Onda Verde – Tratamiento de datos

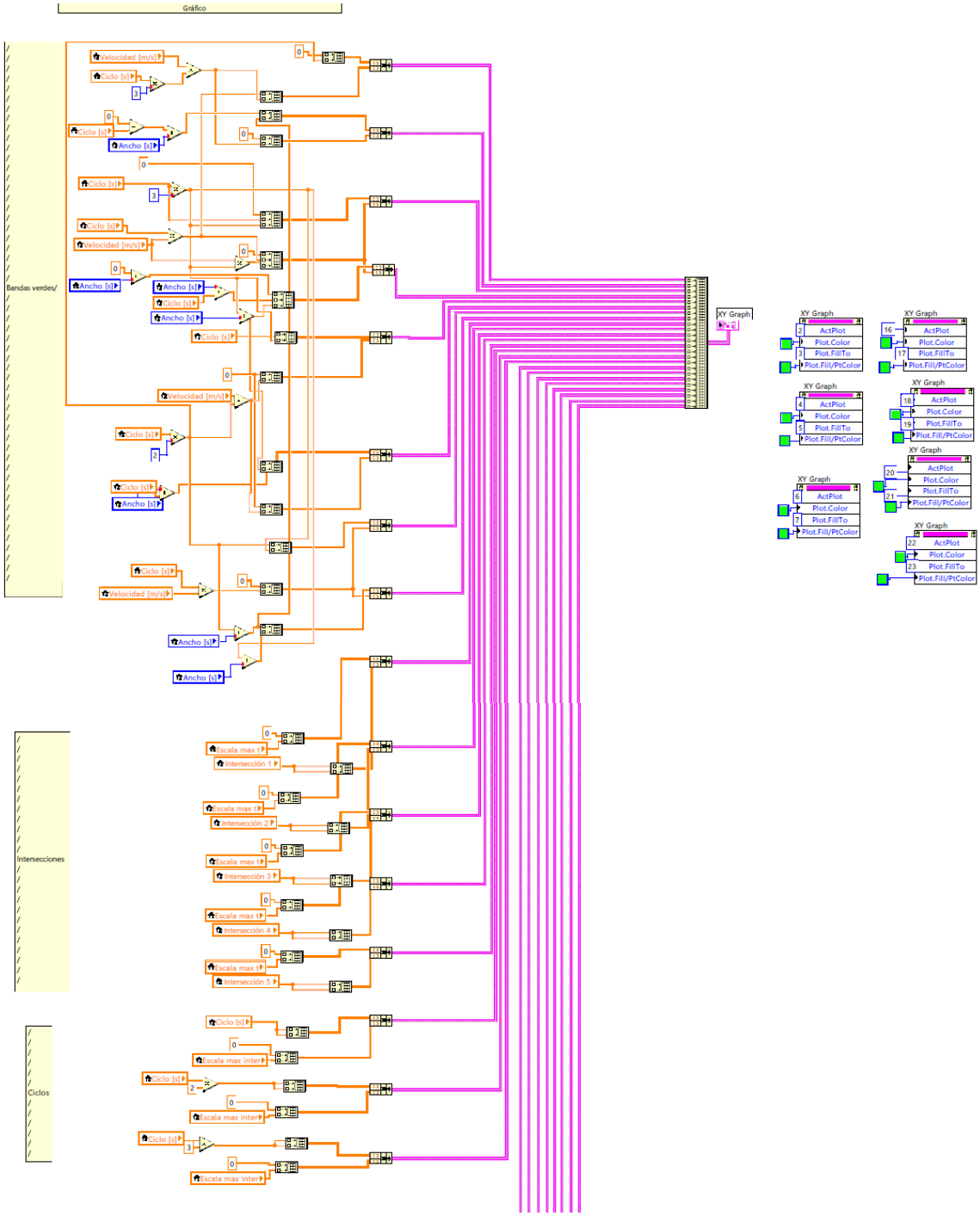

Figura 4.2.2 – Código general Software Onda Verde – Generación de gráfico onda verde en una sola dirección

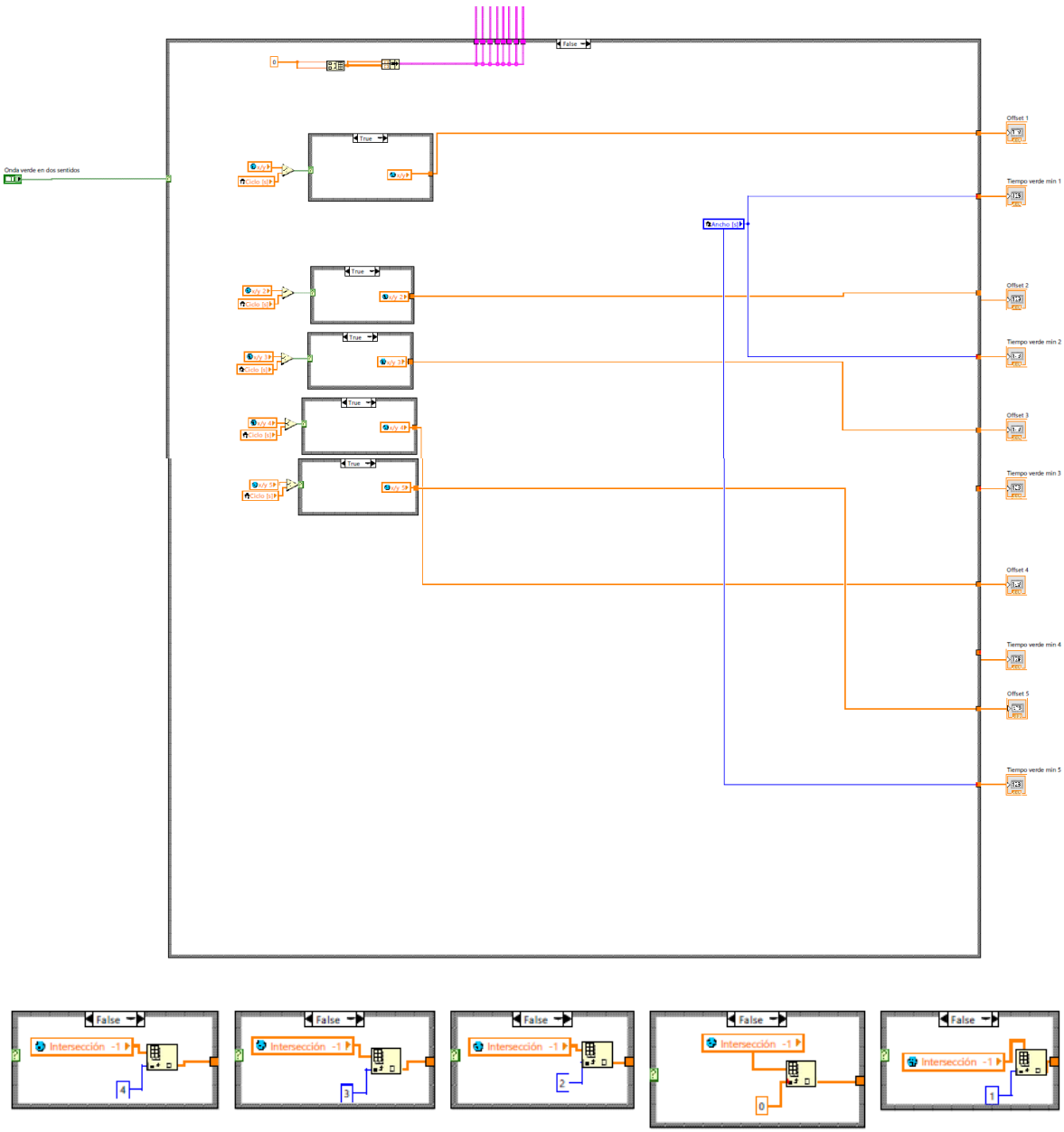

Figura 4.2.3 – Código general Software Onda Verde – Generación de gráfico onda verde en una sola dirección

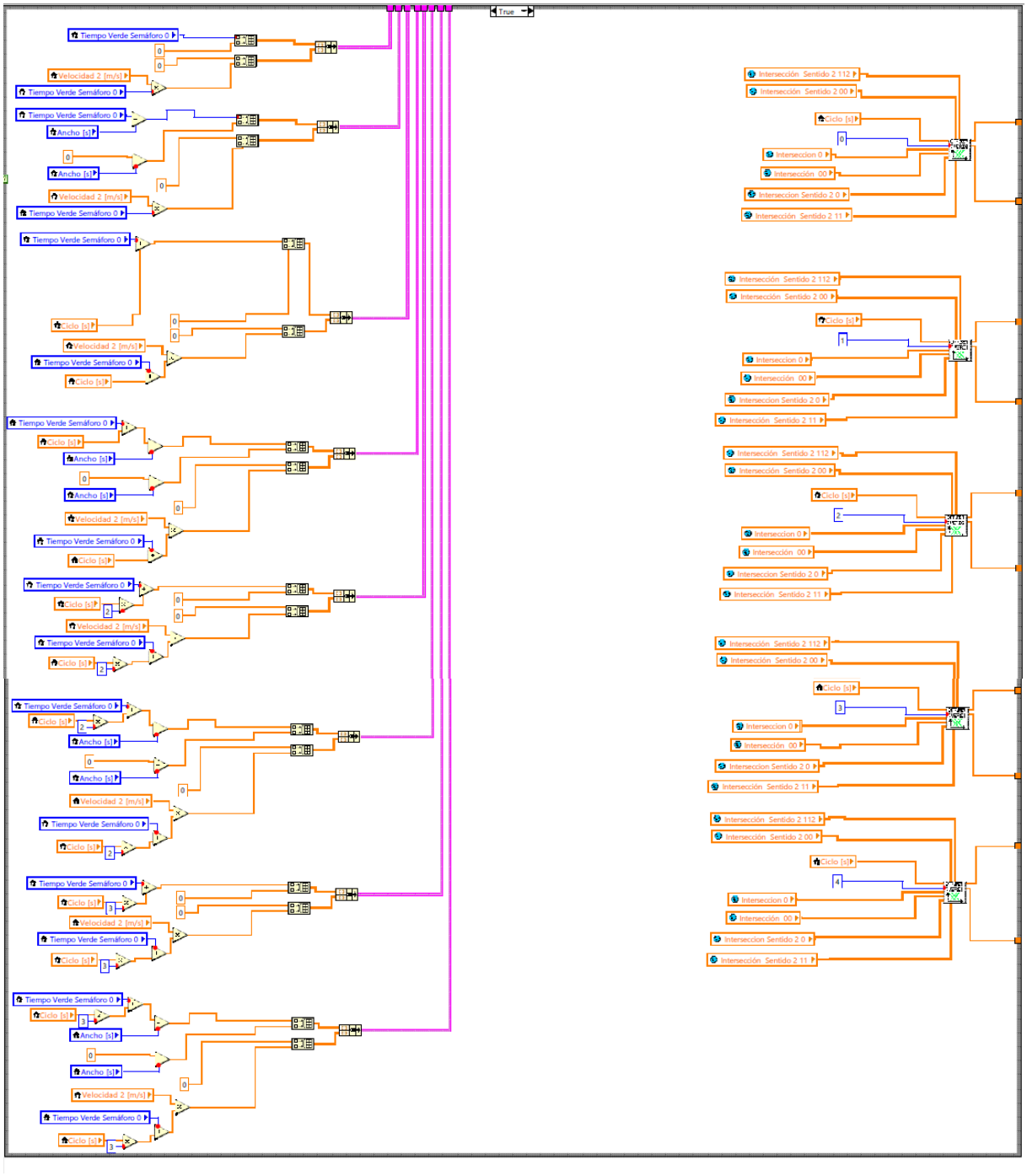

Figura 4.2.4 – Código general Software Onda Verde – Generación de gráfico onda verde en dos direcciones

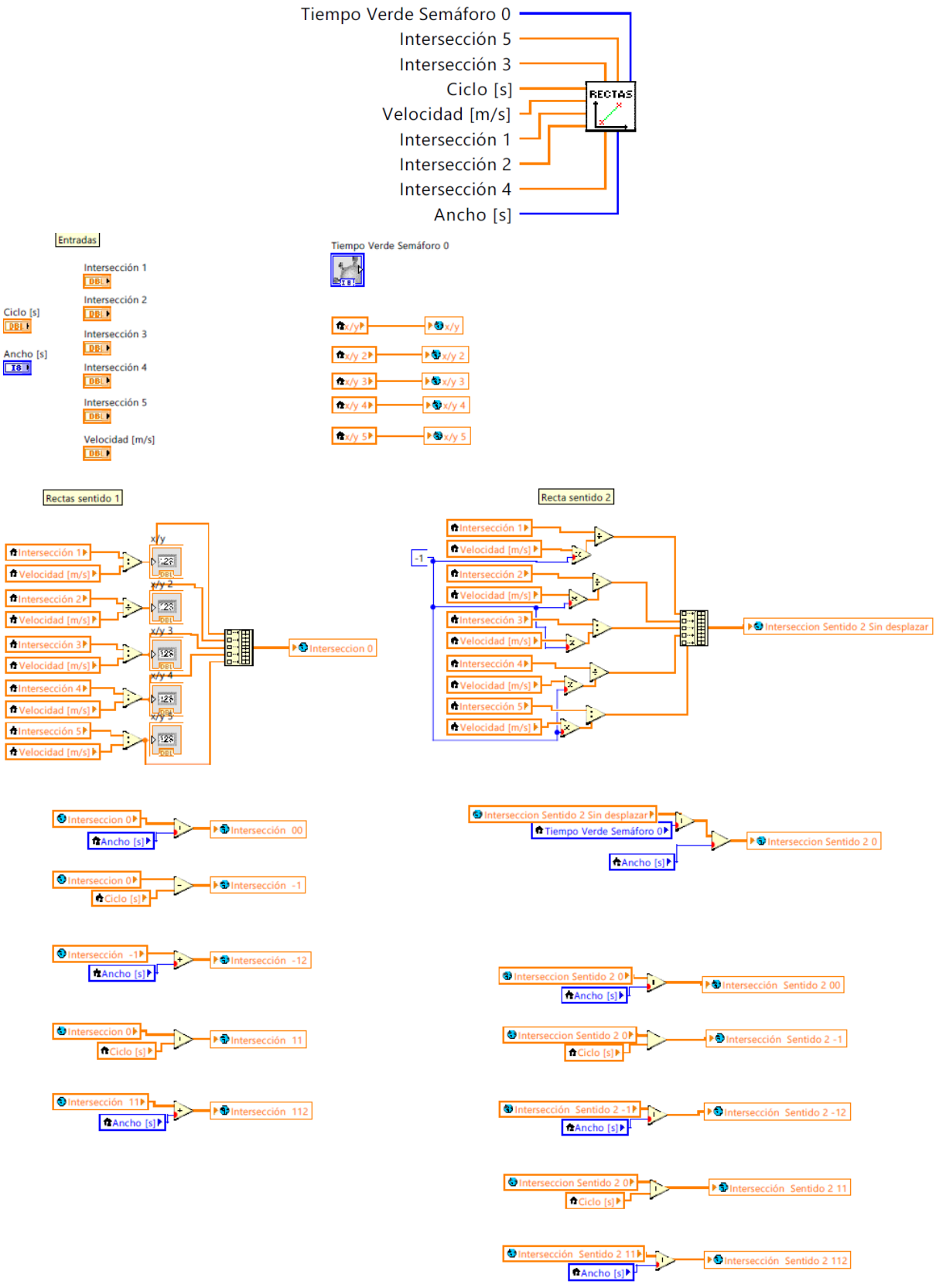

Figura 4.2.5 – Código subVI Rectas

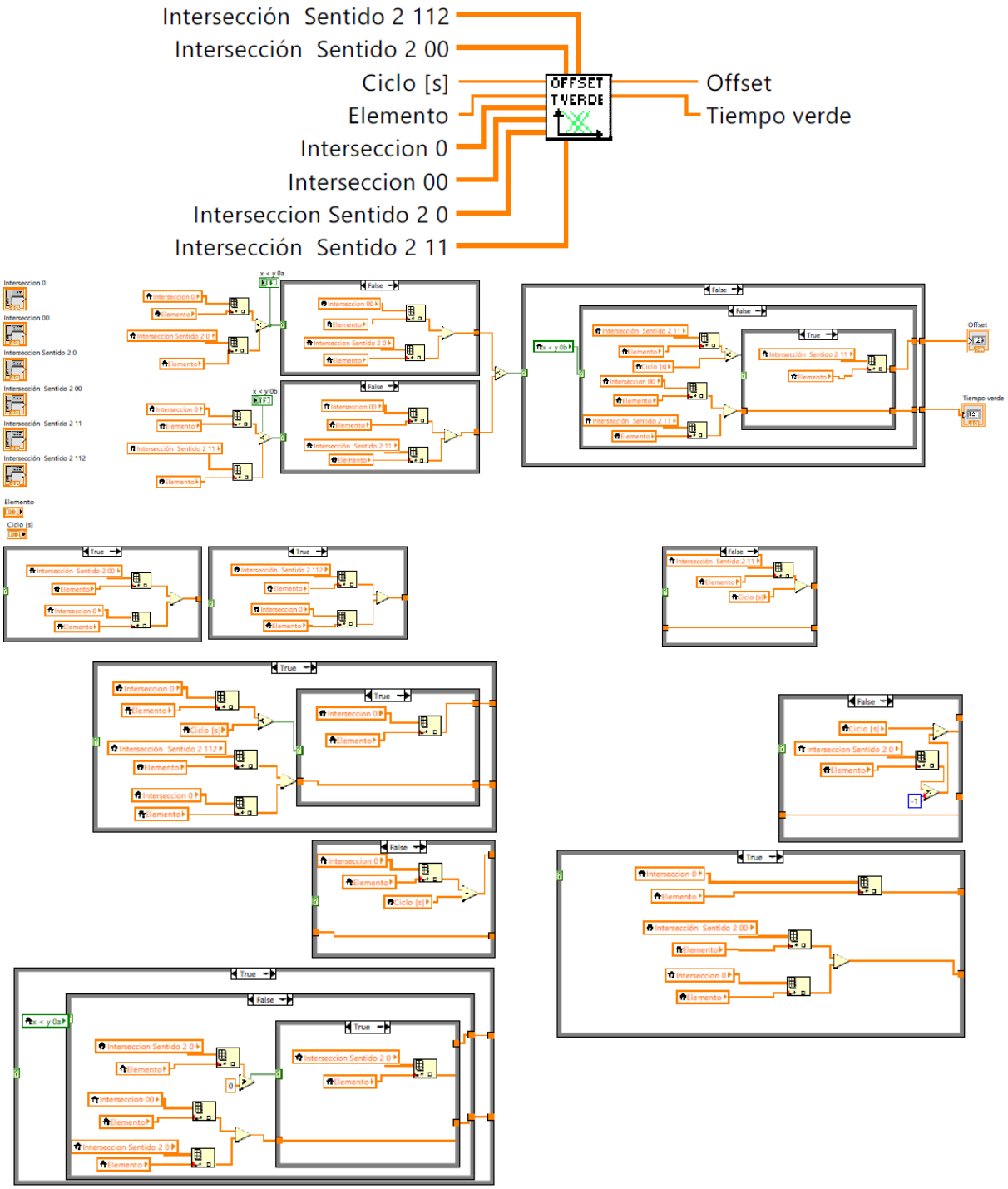

Figura 4.2.6 – Código subVI Offset Tverde

5. MANUAL DE USUARIO

El controlador cuenta con salidas de potencia, entradas digitales, señales luminosas y puertos de comunicación, que deben ser utilizados correctamente para lograr un funcionamiento óptimo. A continuación, se detallará la configuración del controlador y su conexionado para lograr cumplir con las necesidades del usuario.

5.1. Conexionado

El dispositivo debe alimentarse de la red eléctrica a través de su entrada de 220 Vca, como puede observarse en las figuras 5.1.1 y 5.1.2. La entrada de energía pasa por una fuente de alimentación, descripta en el capítulo anterior, que es la encargada de energizar su etapa de control y de potencia. La etapa de control dispone de periféricos como pulsadores, display, señales luminosas y entradas de comunicación, mientras que la etapa de potencia cuenta con relés y 13 borneras.

El controlador posee 6 pulsadores nombrados, en el esquema de conexionado, a los pulsadores OK, ARRIBA, ABAJO, EMERGENCIA, NORMAL y DIA DISTINTO. Los primeros tres pulsadores permiten la navegación por el Menú Principal y la consulta de los valores de los parámetros. Los últimos tres pulsadores, ubicados a la izquierda en el esquema de conexionado, permiten la imposición del modo seleccionado por sobre el que se encuentra funcionando.

Además, posee 14 señales luminosas. Desde L1 hasta L12, se encargan de reflejar las señales de tránsito que el controlador indica en cada uno de los semáforos. El led declarado en el esquema de conexionado como FALLLA busca la fácil detección visual de una anomalía. En el caso del led SEÑAL GPS se encarga, cuando está encendido, de mostrar que la recepción de datos del GPS es correcta.

Display LCD 20x4: permite la navegación por el Menú Principal y la consulta de los valores de cada uno de los parámetros.

Como puede observarse en los siguientes esquemas, el controlador presenta 13 borneras de potencia. Estas borneras son de 220 Vca y, como se indica en el esquema funcional, en las borneras 1-4-7-10 deben conectarse las fases de las lámparas verdes de los semáforos, en las borneras 2-5-8-11 las fases de las lámparas amarillas de los semáforos, en las borneras 3-6-9-12 las fases de las lámparas rojas de los semáforos. Por último, el neutro común a todas las lámparas de los semáforos debe conectarse en la bornera BN. Cada una de las lámparas de los semáforos se encuentra nombrada desde C1 hasta C12 con sus respectivos fusibles de protección nombrados desde F1 hasta F12.

El prototipo cuenta con dos entradas utilizadas para comunicación. Por un lado, la entrada de la antena GPS, destinada a recibir los datos utilizados para la coordinación y para el calendario. Por el otro, la entrada de la comunicación ethernet a través del conector RJ45, destinado a la interacción con el Software Central.

Desde la etapa de control y en paralelo a cada uno de los leds L1-12, se encuentran los pines X1-12 utilizados para comandar los relés. Cabe destacar que, aunque no se encuentra esquematizado, la etapa optoacoplada de cada uno de los relés precisa de

referencia de + 5V y GND y que en paralelo a cada lámpara C1-12 hay colocado un capacitor de 2200 pF para disminuir el ruido electromagnético entre la etapa de potencia y lógica.

Cabe destacar que todas las referencias del tablero y del esquema funcional se encuentran puestas a tierra.

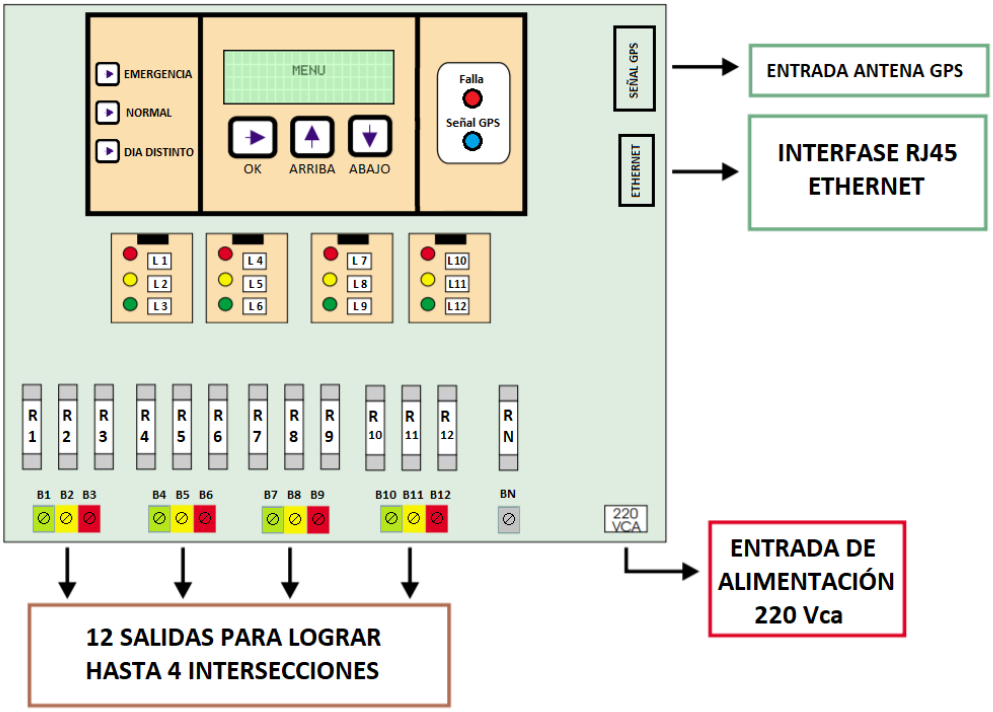

Figura 5.1.1 – Esquema de conexionado exterior

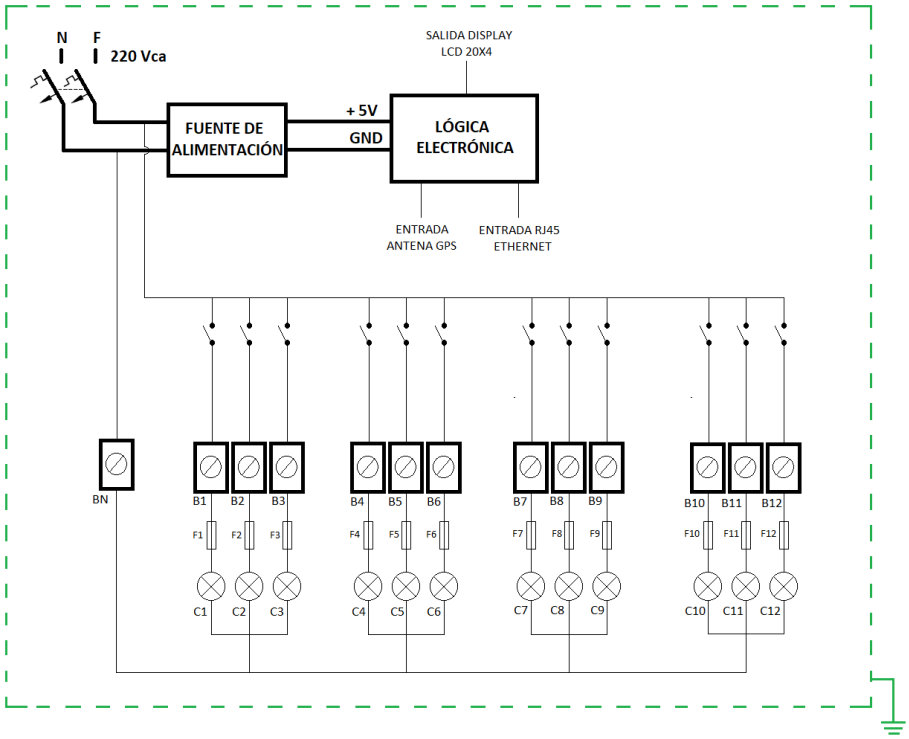

Figura 5.1.2 – Esquema funcional

5.2. Configuración in situ

Como se mencionó anteriormente, la manera de configurar el dispositivo en el lugar de instalación es a través de tres pulsadores nombrados como OK, ARRIBA y ABAJO y un display LCD 20x4 que se encarga de la mostrar las leyendas correspondientes.

En primer lugar, cuando el equipo se encuentra funcionando, el display está apagado hasta que se presiona OK. Gracias a ARRIBA y ABAJO se puede navegar por un menú compuesto por 6 pantallas, ver figura 4.2.1. Las primeras tres pantallas contienen cada uno de los parámetros de configuración de los modos de funcionamiento, en la cuarta pantalla se permite consultar los valores que tiene guardado el dispositivo y por último se da la opción de aceptar los cambios realizados o cancelar si no se desea que permanezcan estos cambios.

La primera pantalla aparece con la leyenda Modo Normal haciendo referencia al modo de funcionamiento explicado en capítulos anteriores. En este submenú es el único lugar donde se puede configurar la cantidad de semáforos que tiene la intersección, luego este parámetro es adoptado por los demás modos de funcionamiento. Seguido permite variar los valores de cada uno de los tiempos verdes y amarillos, sin embargo, si el controlador se seteó para trabajar con 2 semáforos, los valores de los tiempos verdes 3 y 4 no son necesarios cargarlos (pueden tener cualquier valor). Una vez establecidos los tiempos con los que se va a trabajar, se permite modificar el Desfasaje para estar coordinado en una onda verde. Por último, se permite observar el valor del ciclo que queda prestablecido. Todos los valores de tiempos, desfasaje y ciclos deben estar expresado en segundos. Cabe destacar que para modificar cada parámetro hay que posicionarse sobre este, presionar OK, modificar el valor con ARRIBA/ABAJO y nuevamente presionar OK.

El segundo submenú contiene los parámetros necesarios para configurar el Modo Feriado. En primer lugar, se setean los tiempos verdes y desfasaje (si corresponde), de la misma manera que en el Modo Normal. A continuación, se configura el día y el mes que se busca que el controlador funcione de la manera que lo estamos configurando. Por último, se puede observar el ciclo del Modo Feriado.

El tercer submenú contiene los parámetros necesarios para configurar el Modo Día Distinto. Este modo, como se mencionó anteriormente, permite un determinado día de la semana y durante un determinado intervalo de tiempo, el controlador funcione con parámetros distintos al Modo Normal o Feriado. Se configuran los tiempos verdes y desfasaje (si corresponde) de la misma manera que los anteriores modos. Luego se debe determinar el día de la semana que se busca que funcione con los tiempos verdes que se setearon anteriormente. Los valores que adopta este parámetro van de lunes a domingo y “todos los días”. Cabe destacar que, si se establece “todos los días” como día de la semana, el funcionamiento será intermitente amarillo. Luego se establece el intervalo de tiempo en el formato HH:MM de 0 a 24 hs y de 0 a 59 minutos respectivamente. La hora y minutos ON son los establecidos para que comience a operar en este modo y la hora y minutos OFF para que vuelva al Modo Normal o Feriado según corresponda. Por último, se puede observar el ciclo del Modo Día Distinto.

La cuarta pantalla sirve de consulta de los valores con los que se encuentra funcionando el dispositivo antes de cualquier cambio. Se presentan todos los valores mencionados en los tres submenús referenciados con una sigla que representa el modo de funcionamiento dónde N es Modo Normal, F es Modo Feriado y D es Modo Día Distinto.

Por último, se presentan dos pantallas para aceptar o cancelar los cambios presionando OK según la elección.

Es importante destacar que a medida que se navega por el menú el controlador continúa sus tareas. Sí los cambios fueron cancelados, es decir, se ingresó al menú únicamente a consultar valor, el controlador no modifica su accionar. En cambio, si se realizan cambios en los parámetros de funcionamiento, según indica la Norma IRAM 62020, el dispositivo inicia un proceso de destello amarillo (similar al que realiza al iniciar por primera vez) para no generar un cambio brusco en los parámetros que pueda ocasionar accidentes.

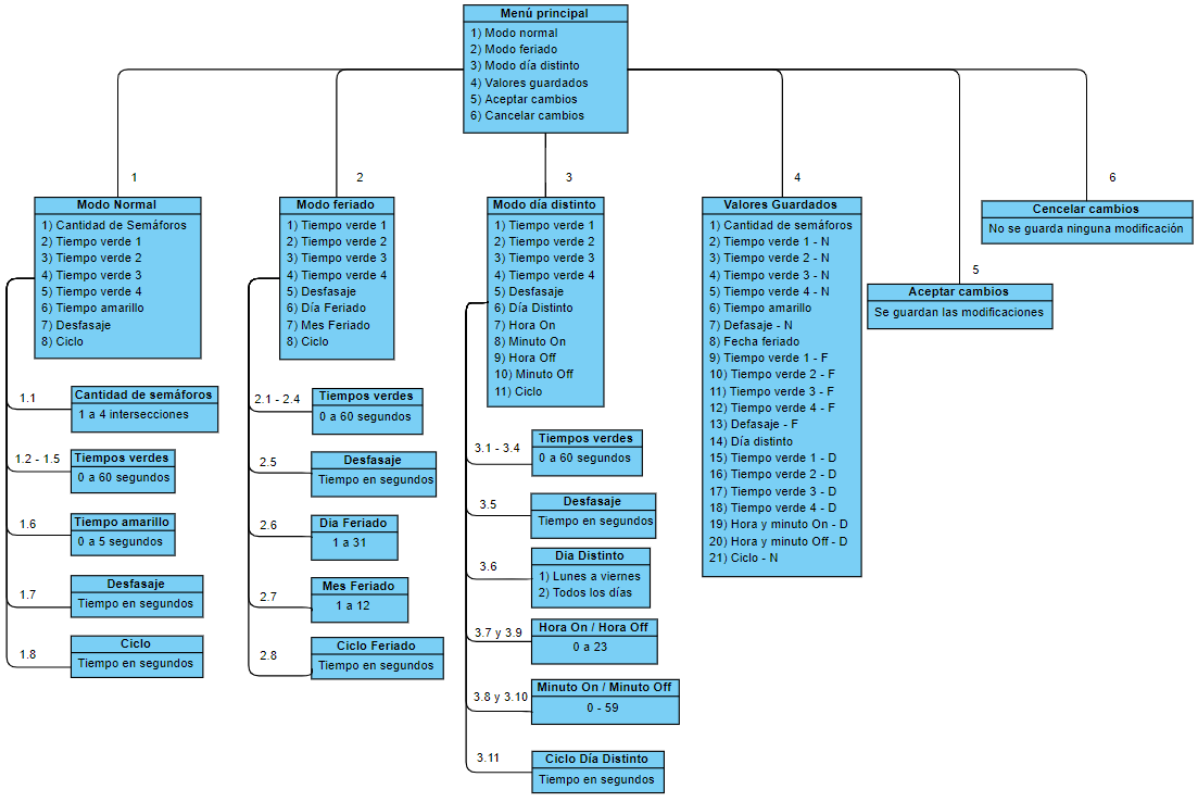

Figura 5.2.1 – Menú principal

5.3. Configuración remota

Además de la configuración descripta anteriormente que implica estar presencialmente en el lugar de instalación, el controlador puede configurarse y monitorearse en forma remota. La idea principal de este software es tener una supervisión rápida del estado de los controladores y pequeñas modificaciones de parámetros. Es importante hacer referencia a que los parámetros que se pueden modificar a distancia no son los mismos que los que se pueden modificar in situ por el hecho de no estar monitoreando lo que ocurre con el tránsito cuando se modifican los

parámetros. A continuación, se presenta el software central y su interfaz con el usuario.

5.3.1 Software Central

Como se mencionó en el capítulo 4, el software central se encuentra dividido en un software de monitoreo y un calculador de onda verde. A continuación, se exponen cada una de sus interfases gráficas.

5.3.1.1 Software de monitoreo

La pantalla principal está compuesta en dos partes. La sección 1 permite elegir entre dos subpantallas llamadas Estado Actual y Configuración, la sección 2 permite setear los parámetros necesarios para la comunicación con un semáforo.

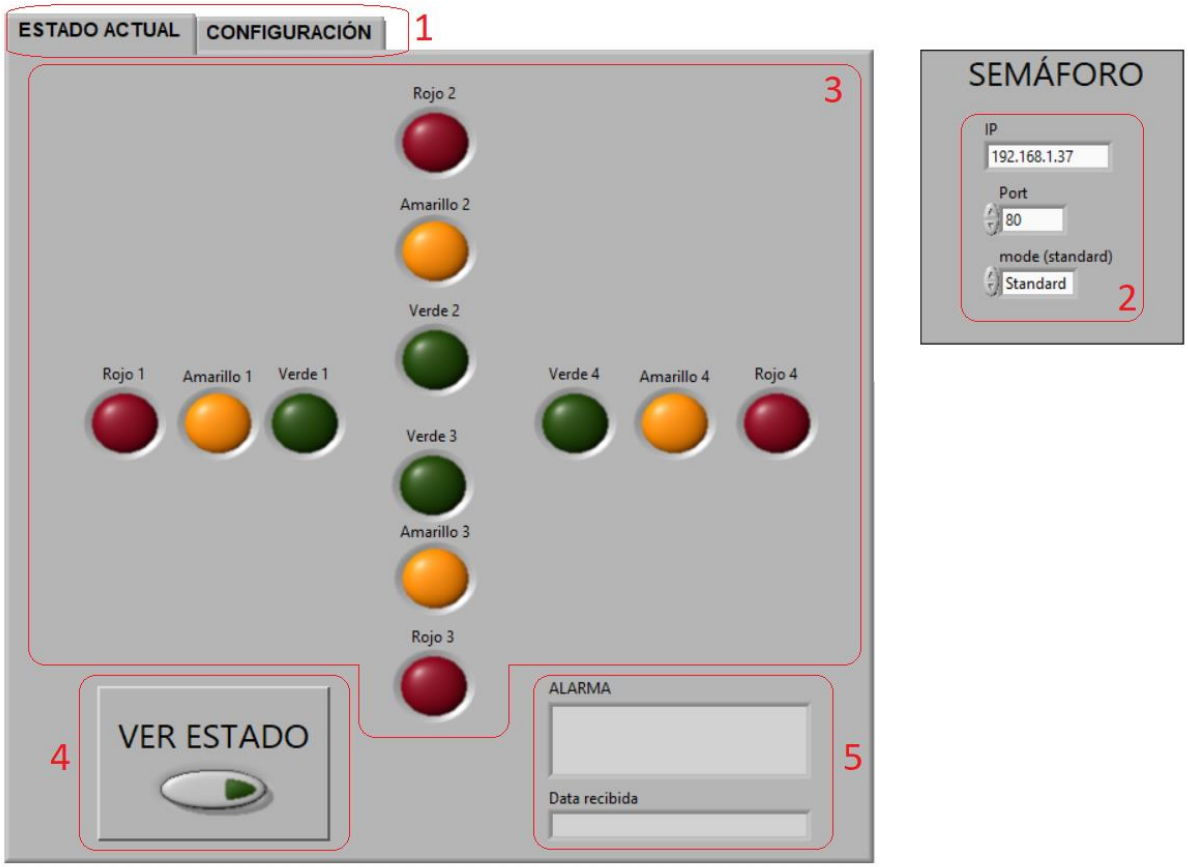

Figura 5.3.1.1.1 – Pantalla principal Software de monitoreo – Subpantalla Estado Actual

En la figura anterior se observa la subpantalla Estado Actual formada de 3 partes. El recuadro número 4 permite activar o desactivar la comunicación con el semáforo que se estableció en el recuadro 2 bajo su número de IP. Una vez activada la comunicación, en el recuadro número 3 comenzarán a reflejarse las señales luminosas en tiempo real. Además, cuenta con un recuadro número 5 dónde se pueden observar dos tipos de mensajes, en el superior se presenta cualquier tipo de alarma que haya sucedido en un semáforo, el texto indica: “ERROR: Luz roja rota en

[número de IP del controlador]". Luego en la parte inferior se muestra los datos recibidos desde el controlador, este campo no es de utilidad para el usuario.

A continuación, puede observarse la subpantalla Configuración. Esta subpantalla permite modificar los parámetros de funcionamiento del semáforo de manera remota que son únicamente los tiempos de funcionamiento del Modo Normal y la cantidad de semáforos. En el recuadro 1 se ingresan los valores y el recuadro cuenta con el botón que debe ser presionado para que se confirmen las modificaciones y se envíen los datos.

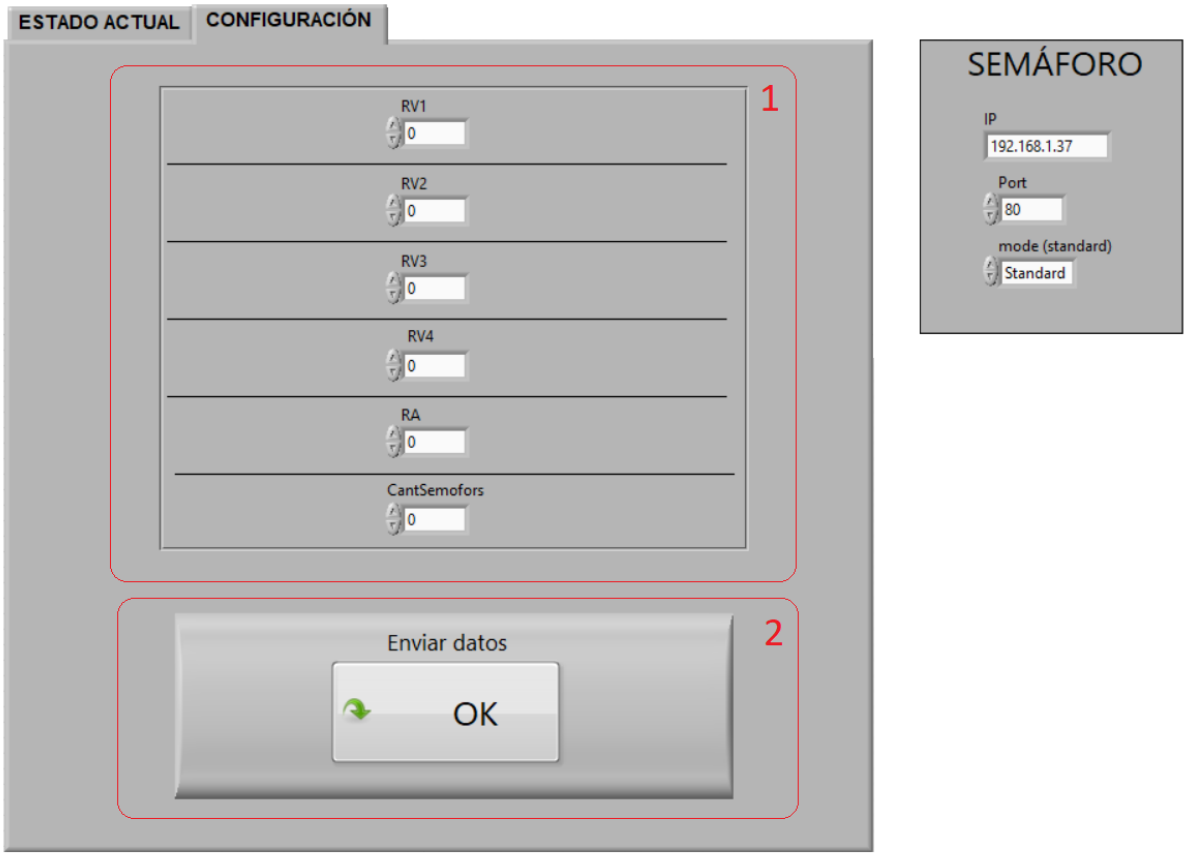

Figura 5.3.1.1.2 – Pantalla principal Software de monitoreo – Subpantalla Configuración

5.3.1.2 Software onda verde

La pantalla principal está compuesta por tres partes. En el recuadro 1 se ingresan los datos que definen la onda verde tal como se vio en el capítulo Introducción. Se puede ingresar hasta 5 semáforos y cada uno debe estar referido a un semáforo colocado en el 0, debe ingresarse el valor del Ciclo que es igual para todos los semáforos, la velocidad a la que se desea que se genere la onda verde y el ancho de la onda verde la cual es quién permitirá mayor o menor flujo dentro de la onda verde. En el recuadro 2 se podrá observar gráficamente y de manera dinámica la variación de cada uno de los parámetros definidos anteriormente. En el eje de ordenadas se grafican la distancia entre semáforos, en el eje de abscisas la duración del ciclo y en verde la pendiente y el ancho de la onda verde.

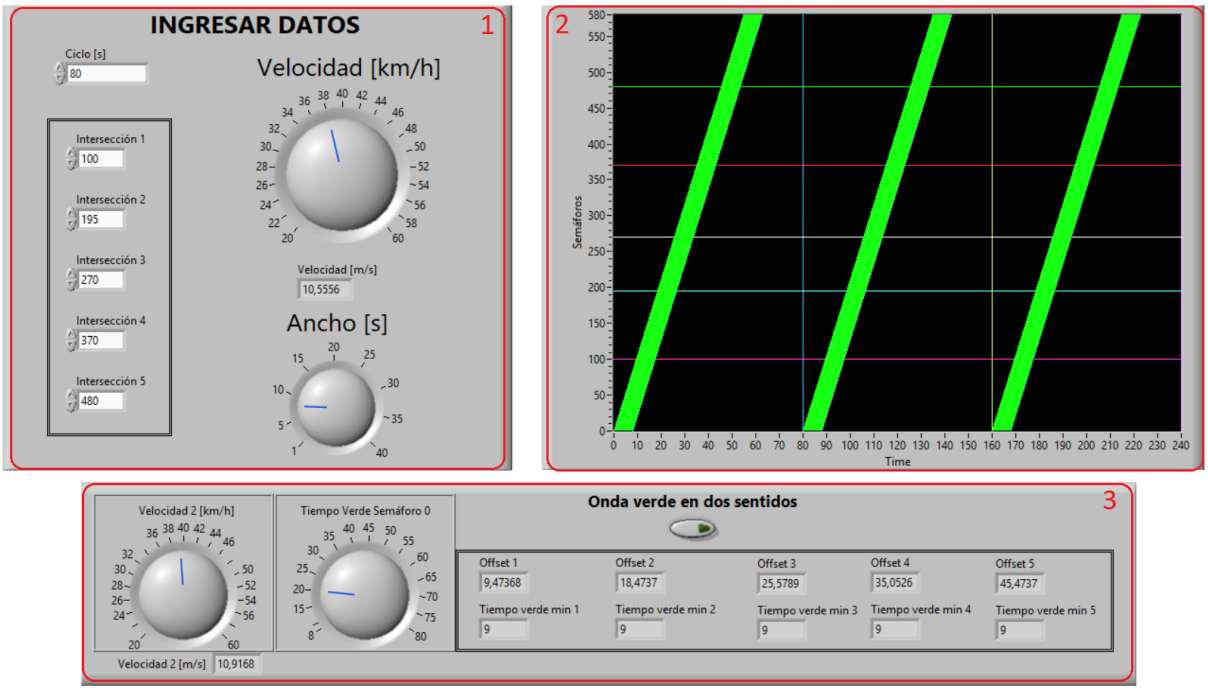

Figura 5.3.1.2.1 – Pantalla principal Software onda verde – Onda en un sentido

Por último, el recuadro 3 da la opción de agregar onda verde en ambos sentidos de la vía, cuando esta opción se encuentra seleccionada se permite ingresar la velocidad de la segunda onda verde, permitiendo que pueda ser distinta en ambos sentidos. Esto da la posibilidad de agilizar el tránsito en determinados intervalos del día hacia la dirección que tenga mayor tráfico.

Como segundo valor a ingresar se encuentra el Tiempo Verde Semáforo 0, esto se debe a que, aunque haya un ancho de onda verde, el tiempo de verde del primer semáforo puede ser mayor que este ancho (no es recomendable) lo que generaría un desplazamiento de la segunda onda verde a lo largo del eje del tiempo como puede observarse en la Figura 5.3.1.2.3. Al final se presenta un recuadro con los valores para configurar cada uno de los semáforos. El primer dato es Offset que es el desfasaje que debe tener el controlador en comparación con el controlador de referencia (ubicado en la posición 0 de la onda verde) y el segundo dato es el Tiempo Verde Mínimo que debe tener el controlador, en este caso es para garantizar la onda verde en ambos sentidos, en el caso que sea en un solo sentido el tiempo verde mínimo es el ancho de la onda.

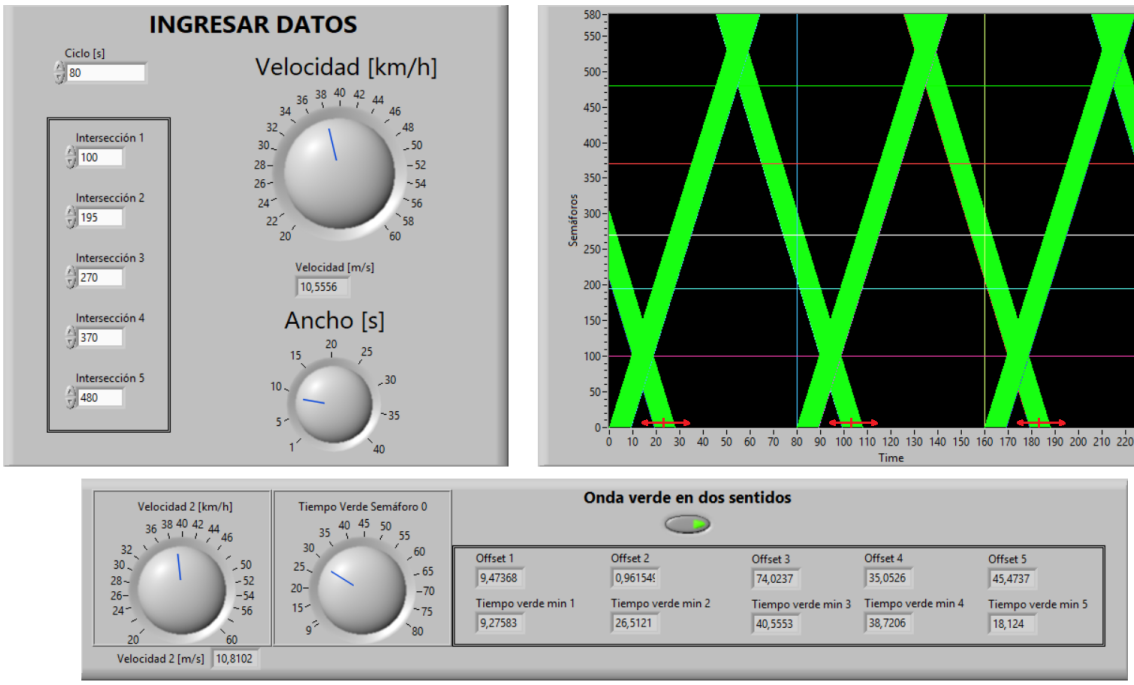

Figura 5.3.1.2.2 – Pantalla principal Software onda verde – Onda en dos sentidos

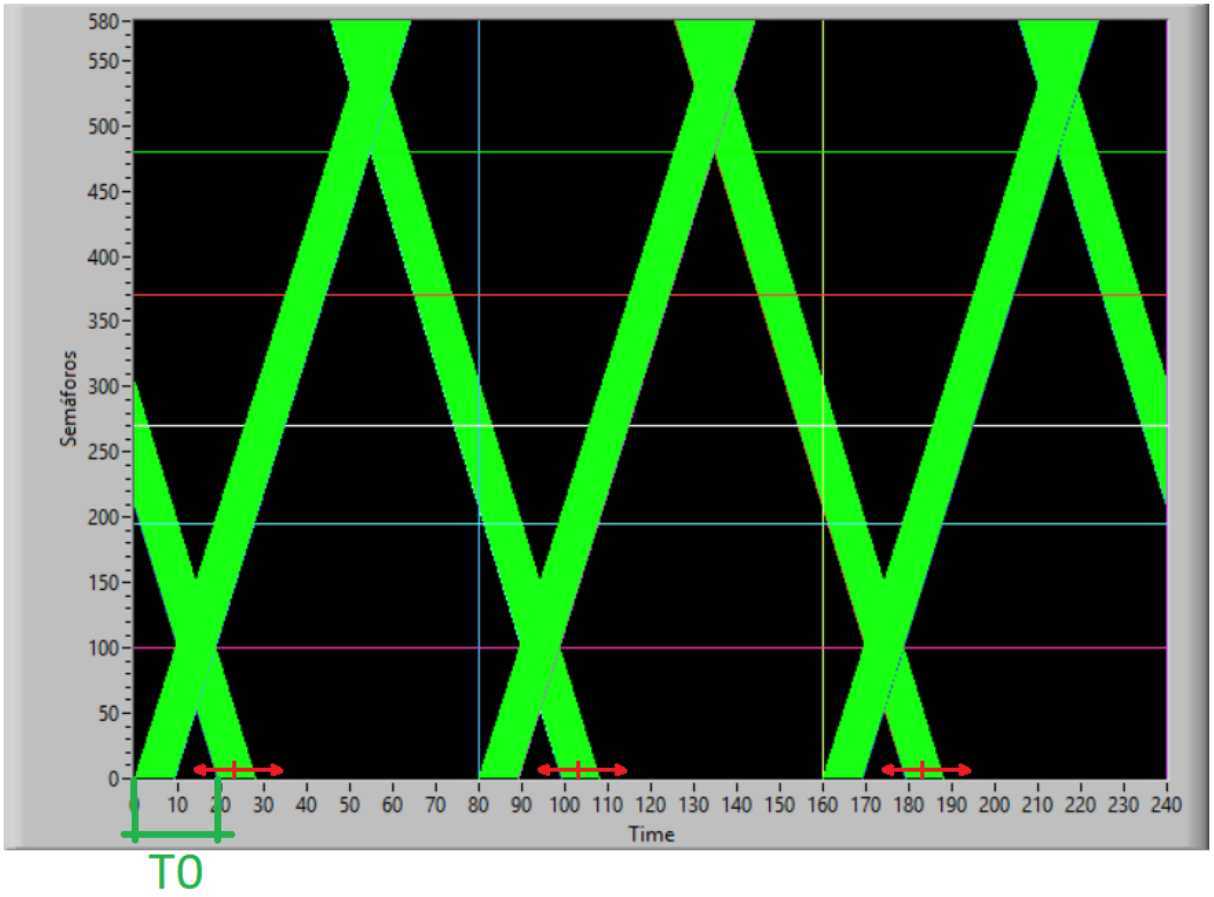

Figura 5.3.1.2.3 – Tiempo Verde Semáforo 0

6. ANÁLISIS ECONÓMICO

6.1. Costo del prototipo

Para la construcción del prototipo fue necesario la obtención de los componentes descriptos en el capítulo 3. A continuación, se presenta detalladamente cada uno con su precio unitario, en dólares.

Controlador				
	Item	Cantidad	Precio unit. [u$D]	Precio total [u$D]
1	Arduino Mega 2560	3	15,6	46,8
2	Módulo Ethernet W5100	1	12	12
3	Módulo I2C	1	2	2
4	Módulo A7 Ai Thinker GPS	1	35,6	35,6
5	Módulo Level Shifter	1	0,75	0,75
6	Módulo LCD 20x4	1	7,35	7,35
7	Relés estado sólido 5 V	12	3	36
8	Pulsador	3	0,1	0,3
9	LEDS	30	0,1	3
10	Resistencias	10	0,03	0,3
11	Estaño	1	1,66	1,66
12	Placa PCB 10x15 cm	1	4,5	4,5
SUBTOTAL CONTROLADOR				$ 150,25
Fuente de Alimentación				
13	Transformador 220 / 24	1	4,82	4,82
14	Diodo	8	0,083	0,7
15	Diodo Zenner	1	0,1	0,1
16	Capacitores varios	7	0,33	2,35
17	Relé 12 V	1	0,85	0,85
18	Relé 6 V	1	0,85	0,85
19	Transistor BC548	1	1,25	1,25
20	Integrado 7805	1	0,6	0,6
21	Integrado 7815	1	0,65	0,65
22	Batería 12 V 1,3 Ah	1	8,3	8,3
23	Estaño	1	1,66	1,66
SUBTOTAL FUENTE ALIMENTACIÓN				$ 22,15
TOTAL				u$D 173

Taba 6.1 – Detalle de componentes utilizados y su valor

6.2. Criterio de selección de componentes

Se puede observar que los pilares fundamentales del costo del prototipo son los Arduino Mega 2560, el Módulo A7 Ai Thinker GPS y los Relés de estado sólido 5V representando el 68,5% del costo.

El Módulo A7 Ai Thinker GPS agrega un gran valor a las prestaciones del controlador brindándole al usuario la posibilidad de generar ondas verdes sincronizadas por gps, modificar el modo de funcionamiento a lo largo del día, en ciertos días de la semana

o en días feriados. Como se puede observar en el análisis de presupuestos de controladores ya inmersos en el mercado, el incremento de un controlador con GPS frente al mismo sin este es del 28,5% por lo tanto se justifica el 20,5% que representa en este proyecto.

El tercer pilar son los relés de estado sólidos (SRR) utilizados para el control de las señales luminosas. El costo de relés electromecánicos para el mismo fin era de 1,15 dólares cada uno, representando un valor final de $14 resultando el costo final del controlador de $128 frente a los $150 con relés de estado sólido. En este caso se optó por tener un incremento del 14% del costo de controlador contemplando las grandes ventajas de los SRR cómo fiabilidad por no tener piezas móviles, mayor número de conmutaciones, menores tiempos de respuesta, aislamiento eléctrico de los circuitos.

Por último, se presupuestó un gabinete estanco con un grado de IP 65 y medidas 400x300x160 mm por el valor de 33 dólares lo que elevaría el costo total del producto a 200 dólares.

6.3. Presupuestos de controladores comerciales

A través de indagaciones a distintas marcas nacionales e internacionales se consiguió el presupuesto de dos controladores fabricados en Argentina. Cómo se mencionó en la introducción, ambos equipos cuentan con características técnicas y prestaciones similares a las del prototipo desarrollado.

En primer lugar, se obtuvo el presupuesto del controlador de la marca MICRO ELECTRONICA. El controlador no tiene una estructura modular, por lo tanto, su valor varía dependiendo si cuenta o no con sincronización a través de GPS. El dispositivo con 4 movimientos y 12 salidas tiene un valor de u$D 700 y su valor aumenta a u$D 900 si se selecciona el modelo con GPS incluido.

En segundo lugar, se obtuvo el presupuesto del controlador de la marca TACUAR SRL, modelo CET 234N. En este caso el modelo es único y cuenta con una estructura modular para la sincronización de GPS. El valor, sin contar el GPS, es de u$D 1662.

6.4. Otras tecnologías disponibles

Como se comentó al comienzo del presente escrito, se seleccionó Arduino y LabVIEW para el desarrollo del prototipo por sus grandes ventajas ya descriptas y por los conocimientos adquiridos a lo largo de la carrera de Ingeniería Electromecánica. Sin embargo, se podría haber logrado los mismos resultados con la implementación de otras tecnologías. A continuación, se evaluarán los costos del desarrollo del controlador de tránsito basado en controlador lógico programable (PLC por sus siglas en inglés).

Un PLC es una pequeña computadora ampliamente utilizada en la automatización industrial. Los PLC están diseñados para controlar varias señales de entrada y salidas y se caracterizan por su gran robustez antes vibraciones, temperaturas y ruido eléctrico.

6.4.1. Alternativa Siemens

A continuación, se presentan las características técnicas del PLC Logo 8 de la marca Siemens que se encuentre en el mercado con un valor de 250 dólares.

- Display 20x6 y botonera para navegación.
- 8 entradas digitales.
- 4 salidas digitales expandibles hasta 24 (salidas a relé 10 A).
- Tensión de alimentación 12/24 V DC. (Existen modelos con alimentación en AC).
- Comunicación Ethernet y Web Server integrado.
- Slot para tarjeta SD.
- Configuración a través de PC o desde botonera.

Figura 6.4.1 – PLC Logo 8 de Siemens

Cuenta con las características para lograr las prestaciones básicas del prototipo desarrollado.

Agregar sincronización de tiempo implica agregar otro módulo modelo CMR2020 que tiene el valor de 225 dólares. A continuación, se enuncian sus características técnicas.

- Envío / Recepción de SMS.
- Rastreo de posición vía GPS.
- Sincronización del reloj vía GPS.
- 2 entradas digitales.
- 2 salidas digitales.
- Comunicación Ethernet.

Figura 6.4.2 – Módulo CMR2020 de Siemens

Si además se pretende tener una terminal de dialogo como una pantalla HMI, dónde se podrían reflejar las señales luminosas y tener un menú más flexible, el costo se elevaría en 210 dólares.

Por lo tanto, el prototipo desarrollado con las mismas prestaciones, pero desarrollado con PLC de la marca siemens, tendría un valor de entre 475 dólares y 685 dólares dependiendo si tiene o no la terminal de dialogo.

6.4.2. Alternativa industria argentina

La segunda alternativa busca reducir costos con marcas no tan reconocidas en el mercado como lo es Siemens. En este caso se evaluó un PLC de la marca Slicetex modelo STX 8140 con un valor de 215 dólares.

A continuación, se presentan las características técnicas.

- Alimentación 12/24 V DC.
- 5 entradas digitales.
- 4 salidas digitales (opciones relé electromecánico, transistor NPN o relé de estado sólido).
- Reloj interno a través de RTC.
- Comunicación Ethernet, servidor web integrado.
- Soporte ModBus TCP.
- Conexión a HMI.

Figura 6.4.2.1 – PLC marca Slicetex modelo STX 8140

A este modelo es necesario agregarle un panel con LCD y teclado de la misma marca modelo SH-300 que es la más básica que presenta la marca con un valor de 250 dólares.

A continuación, se presentan las características técnicas.

- Alimentación 12/24 V DC.
- Display LCD 192x64 píxeles.
- 14 teclas de propósito general.
- Puerto de comunicación RS232 / RS485 / RS422.

Figura 6.4.2.2 – Pantalla HMI marca Slicetex modelo SH 300

Por lo tanto, una alternativa de la industria argentina tendría un costo de 465 dólares.

6.4. Comparación

Resumiendo lo expuesto anteriormente, se cuenta con el prototipo desarrollado con un costo de aproximadamente 200 dólares, una alternativa con PLC de la marca Siemens con un valor que varía de 475 a 685 dólares y una alternativa con PLC de industria argentina con un valor de 465 dólares.

Cabe destacar que, la alternativa de Siemens cuenta con un reconocido prestigio por su alta calidad de sus productos y su implementación en el mercado. La desventaja de esta alternativa la falta de carácter modular lo que implica tener un equipo con altísimas prestaciones que no son aprovechadas al máximo, esto se puede observar en el módulo GPS y en la pantalla HMI, ambas agregan dos funciones muy importantes al controlador como es la sincronización del tiempo y la navegación por menús, pero agregan muchas otras funciones que no son necesarias en este prototipo.

En cuanto a la alternativa de Slicetex, se cuenta con un producto menos reconocido en el mercado, pero con un valor similar a la alternativa Siemens. Además, no se logró encontrar módulo que cuente con coordinación de tiempo a través de GPS, una característica clave en el prototipo desarrollado. Por estas razones se continua el análisis simplemente con la alternativa de la marca Siemens.

Por lo tanto, se logró construir un prototipo acorde a las necesidades con menos de la mitad del costo que la alternativa con PLC de la marca Siemens. Es importante marcar que, a todas las alternativas analizadas no se les agregó las horas hombre de programación, de todas formas, se considera que es similar tanto para la programación de Arduino como la para programación en PLC.

6. CONCLUSIONES

A través de los conocimientos adquiridos a lo largo de la carrera de Ingeniería Electromecánica, se logró diseñar y construir un prototipo de controlador de semáforos sencillo, modular, económico.

La implementación de cada uno de los conceptos obtenidos sumada a la dedicación dio paso a que el diseño del prototipo sea lo más sencillo posible. Las múltiples consultas e investigaciones del mercado, con la detección de ventajas y desventajas, lograron obtener el diseño modular. La consulta y cumplimiento de las normativas vigentes permitieron que el prototipo sea confiable y seguro. El desarrollo, búsqueda de nuevas tecnologías, su elección e implementación redujeron el costo.

A lo largo del proyecto quedo demostrada la flexibilidad, sencillez y potencia de Arduino y LabVIEW, confirmando su correcta elección para la construcción del prototipo. Además de obtener un costo inferior a sus alternativas tecnológicas.

Se logró que el prototipo este a la altura de características técnicas de los controladores de semáforos vigentes en el mercado y acorde a su costo.

De lo expuesto anteriormente, se concluye que se han cumplido los objetivos propuestos al inicio del proyecto.

7. PROPUESTAS A FUTURO

Durante el desarrollo del prototipo han surgido diferentes alternativas para su diseño y construcción. A continuación, se presentan una serie de tópicos a tratarse en futuros proyectos.

- Implementación de tecnologías futuras.
- Implementación de PLC para un controlador más exigente.
- Agregado de demandas externas a través de periféricos para obtener un control totalmente accionado por el tránsito. Estos periféricos pueden ser neumáticos, bobinas de inducción, rayos infrarrojos, etc.
- Cámaras de seguridad limitantes de velocidad para disciplina del tránsito.
- Aumento de memoria para tener mayor flexibilidad.
- Implementación de otros métodos de sincronización como puede ser maestro-esclavo, mediante frecuencia de red o por pulsos.

8. REFERENCIAS

[1] M. d. T. d. Colombia, «www.medellin.gov.co,» [En línea]. Available: https://www.medellin.gov.co/movilidad/documents/seccion_senalizacion/cap7_semaforos.pdf. [Último acceso: 02 2019].

[2] V. G. Valencia Alaix, PRINCIPIOS SOBRE SEMÁFOROS, Medellín, 2000.

[3] J. M. Huidobro, «ACTA,» 2016. [En línea]. Available: https://www.acta.es/medios/articulos/ciencias_y_tecnologia/039001.pdf. [Último acceso: 25 02 2019].

[4] J. Carrasco Avendaño y G. Wazhima Clavijo, DISEÑO DE LA RED SEMAFÓRICA DE LA CALLE MARISCAL LAMAR DESDE LA CALLE MANUEL VEGA HASTA LA CALLE TARQUI, Cuenca, 2012.

[5] H. O. Etchevarne, «Micro electrónica,» [En línea]. Available: http://electronica-micro.com.ar/Manuales/ControladorGPS/CONTROLA-GPS.pdf. [Último acceso: Febrero 2019].

[6] N. S. SRL, «Nortelec Servicios SRL,» [En línea]. Available: http://www.nortelecservicios.com.ar/controlador-semaforico.html. [Último acceso: Febrero 2019].

[7] S. SRL, «Syntico,» [En línea]. Available: http://www.syntico.com.ar/espanol/Docs/synticocet234n.pdf. [Último acceso: Febrero 2019].

[8] T. S.A., «TRAFICTEC S.A.,» [En línea]. Available: http://www.trafictec.com/fichas/GTC8.pdf. [Último acceso: Febrero 2019].

[9] S. AG, «Controlador de Tránsito CL-S214 PLUS,» 2006.

[10] S. N. O. CO, «2nd generation trafffic management system».

[11] R. educativos, «SlidShare,» 04 02 2018. [En línea]. Available: https://es.slideshare.net/recursoseducativos/estructura-programa-arduino. [Último acceso: 24 08 2019].

[12] I. A. d. N. y. Certificación, «IRAM,» [En línea]. Available: www.iram.org.ar. [Último acceso: Marzo Marzo].

[13] Instituto de Normalización y Certificación , IRAM 62968 - Semáforos LED para el control de tránsito vehicular, 2016.

[14] Instituto de Normalización y Certificación, IRAM 62696 - Sistemas de señalización del tránsito vehicular - Compatibilidad electromagnética (CEM), 2014.

[15] Instituto de Normalización y Certificación, IRAM 62970 - Semáforos LED para el control de tránsito peatonal, 2018.

[16] I. d. N. y. Certificación, IRAM 62020 - Equipamiento para la gestión de tránsito. Controladores de tránsito. Requisitos y ensayos., 2019.

[17] European Committee for Standardization, «CEN,» [En línea]. Available: https://www.cen.eu/about/Pages/default.aspx. [Último acceso: Marzo 2019].

[18] British Standard - European Standard, EN 12675 - Traffic signal controllers - Fuctional safesty, 2001.

[19] F. Blanco-Silva, «http://eseprimo.blogspot.com,» 12 04 2005. [En línea]. Available: http://eseprimo.blogspot.com/2005/04/de-la-semana-fue.html. [Último acceso: 2019].

ANEXO A: Código de programación Arduino Controlador

```
                        ////////////////////////////
                        //    BIDEGAIN OCTAVIO    //
                        //    VERSIÓN 21/11/2019  //
                        // CONTROLADOR DE SEMAFOROS //
                        ////////////////////////////

#include <SPI.h>
#include <EEPROM.h>
#include <Wire.h>


long T0 = 0;
volatile int menu;
volatile int intermitente;
int MostrarMenu;
int salir;
int Entrar;
int CantSemaforos;
int Posicion;
int BotonOk;
int BotonArriba;
int BotonAbajo;

int sumatiemposD;
int sumatiemposF;

int cuentainicio;
int tstart;

volatile int RV1;
volatile int RV2;
volatile int RV3;
volatile int RV4;

int RV1N;
int RV2N;
int RV3N;
int RV4N;

int RV1D;
int RV2D;
int RV3D;
int RV4D;

int RV1F;
int RV2F;
int RV3F;
int RV4F;

int RR;
int RR1;
int RA;
volatile int t;
int D2, D3, D6, D8;
int corrient;
int j;
int CantSemaforos_eeprom=1;

//POSICION EN LA MEMORIA EEPROM ------------------------------------
int RV1N_eeprom=2;
int RA_eeprom=3;
```

```
int RV2N_eeprom=5;
int RV3N_eeprom=6;
int RV4N_eeprom=7;

int modoactual;

byte CantSemaforos_eeprom_valor;
byte RV1N_eeprom_valor;
byte RV2N_eeprom_valor;
byte RV3N_eeprom_valor;
byte RV4N_eeprom_valor;
byte RA_eeprom_valor;
byte MS_eeprom_valor;

int RV1D_eeprom=9;
int RV2D_eeprom=10;
int RV3D_eeprom=11;
int RV4D_eeprom=12;

byte RV1D_eeprom_valor;
byte RV2D_eeprom_valor;
byte RV3D_eeprom_valor;
byte RV4D_eeprom_valor;


// VARIABLES EEPROM MODO FERIADO ----------------------------------

int RV1F_eeprom=15;
int RV2F_eeprom=16;
int RV3F_eeprom=17;
int RV4F_eeprom=18;

byte RV1F_eeprom_valor;
byte RV2F_eeprom_valor;
byte RV3F_eeprom_valor;
byte RV4F_eeprom_valor;

//DECLARACIONES ---------------------------------------------------

 char caracterEntrada;
 int entradaDigital, b0, b1, b2, b3, dato;
 String canal;
 boolean estado;
 int contador;
 int h;

//SETUP ----------------------------------------------------------------

void setup() {
  Serial1.begin(9600);
  Serial.begin(115200);
   pinMode (23,OUTPUT);
      pinMode (25,OUTPUT);
        pinMode (27,OUTPUT);
    pinMode (29,OUTPUT);
      pinMode (31,OUTPUT);
        pinMode (33,OUTPUT);
    pinMode (35,OUTPUT);
      pinMode (37,OUTPUT);
        pinMode (39,OUTPUT);
    pinMode (41,OUTPUT);
```

```
      pinMode (43,OUTPUT);
        pinMode (45,OUTPUT);

    pinMode (22,OUTPUT);
      pinMode (24,OUTPUT);
        pinMode (26,OUTPUT);
    pinMode (28,OUTPUT);
      pinMode (30,OUTPUT);
        pinMode (32,OUTPUT);
    pinMode (34,OUTPUT);
      pinMode (36,OUTPUT);
        pinMode (38,OUTPUT);
    pinMode (40,OUTPUT);
      pinMode (42,OUTPUT);
        pinMode (44,OUTPUT);

 // INICIALIZAR SALIDAS ---------------------------------------------
 sumatiemposD=0;
 sumatiemposF=0;
 h=0;
 intermitente=0;
 BotonOk = 0;
 Posicion = 0;
 CantSemaforos = 2;
 t=0;
 Entrar = 0;
 menu=0;
 RR=1;


 // INICIALIZO CON LAS VARIABLES DE LA MEMORIA EEPROM --------------
CantSemaforos_eeprom_valor=EEPROM.read(CantSemaforos_eeprom);
RV1N_eeprom_valor=EEPROM.read(RV1N_eeprom);
RV2N_eeprom_valor=EEPROM.read(RV2N_eeprom);
RV3N_eeprom_valor=EEPROM.read(RV3N_eeprom);
RV4N_eeprom_valor=EEPROM.read(RV4N_eeprom);
RA_eeprom_valor=EEPROM.read(RA_eeprom);

RV1D_eeprom_valor=EEPROM.read(RV1D_eeprom);
RV2D_eeprom_valor=EEPROM.read(RV2D_eeprom);
RV3D_eeprom_valor=EEPROM.read(RV3D_eeprom);
RV4D_eeprom_valor=EEPROM.read(RV4D_eeprom);

RV1F_eeprom_valor=EEPROM.read(RV1F_eeprom);
RV2F_eeprom_valor=EEPROM.read(RV2F_eeprom);
RV3F_eeprom_valor=EEPROM.read(RV3F_eeprom);
RV4F_eeprom_valor=EEPROM.read(RV4F_eeprom);


if (RV1N_eeprom_valor==255){RV1N_eeprom_valor=0;}
if (RV2N_eeprom_valor==255){RV2N_eeprom_valor=0;}
if (RV3N_eeprom_valor==255){RV3N_eeprom_valor=0;}
if (RV4N_eeprom_valor==255){RV4N_eeprom_valor=0;}

if (RV1D_eeprom_valor==255){RV1D_eeprom_valor=0;}
if (RV2D_eeprom_valor==255){RV2D_eeprom_valor=0;}
if (RV3D_eeprom_valor==255){RV3D_eeprom_valor=0;}
if (RV4D_eeprom_valor==255){RV4D_eeprom_valor=0;}

if (RV1F_eeprom_valor==255){RV1F_eeprom_valor=0;}
if (RV2F_eeprom_valor==255){RV2F_eeprom_valor=0;}
```

```
if (RV3F_eeprom_valor==255){RV3F_eeprom_valor=0;}
if (RV4F_eeprom_valor==255){RV4F_eeprom_valor=0;}

CantSemaforos=CantSemaforos_eeprom_valor;
RV1=RV1N_eeprom_valor;
RV2=RV2N_eeprom_valor;
RV3=RV3N_eeprom_valor;
RV4=RV4N_eeprom_valor;
RA=RA_eeprom_valor;

RV1D=RV1D_eeprom_valor;
RV2D=RV2D_eeprom_valor;
RV3D=RV3D_eeprom_valor;
RV4D=RV4D_eeprom_valor;

RV1F=RV1F_eeprom_valor;
RV2F=RV2F_eeprom_valor;
RV3F=RV3F_eeprom_valor;
RV4F=RV4F_eeprom_valor;

 // INICIALIZACIÓN DE CONTADOR -------------------------------------
 cuentainicio = 0;
 tstart=10;
 contador = 0;
 estado = false;
 configuraInterrupcion1(1,0,0,62499);

pinMode(2,INPUT);
pinMode(3,INPUT);
pinMode(21,INPUT);
pinMode(20,INPUT);

attachInterrupt(digitalPinToInterrupt(2), ModoNormal, RISING);
attachInterrupt(digitalPinToInterrupt(3), ModoDiaDistinto, RISING);
attachInterrupt(digitalPinToInterrupt(20), Intermitente, RISING);

}// FIN DEL SETUP ------------------------------------------------

void loop(){
  if (Serial1.available())
    {
     deshabilitaInterrupcion1();
     h=h+1;
     if (h==1){ CantSemaforos = byteToInt(Serial1.read());}
     if (h==2){ RV1N = byteToInt(Serial1.read());}
     if (h==3){ RA = byteToInt(Serial1.read());}
     if (h==4){ RV2N = byteToInt(Serial1.read());}
     if (h==5){ RV3N = byteToInt(Serial1.read());}
     if (h==6){ RV4N = byteToInt(Serial1.read());}
     if (h==7){ RV1D = byteToInt(Serial1.read());}
     if (h==8){ RV2D = byteToInt(Serial1.read());}
     if (h==9){ RV3D = byteToInt(Serial1.read());}
     if (h==10){ RV4D = byteToInt(Serial1.read());}
     if (h==11){ RV1F = byteToInt(Serial1.read());}
     if (h==12){ RV2F = byteToInt(Serial1.read());}
     if (h==13){ RV3F = byteToInt(Serial1.read());}
     if (h==14){ RV4F = byteToInt(Serial1.read());
     h=0;
     EEPROM.update(CantSemaforos_eeprom,CantSemaforos);
     EEPROM.update(RV1N_eeprom,RV1N);
     EEPROM.update(RA_eeprom,RA);
```

```
    EEPROM.update(RV2N_eeprom,RV2N);
    EEPROM.update(RV3N_eeprom,RV3N);
    EEPROM.update(RV4N_eeprom,RV4N);

    EEPROM.update(RV1D_eeprom,RV1D);
    EEPROM.update(RV2D_eeprom,RV2D);
    EEPROM.update(RV3D_eeprom,RV3D);
    EEPROM.update(RV4D_eeprom,RV4D);

    EEPROM.update(RV1F_eeprom,RV1F);
    EEPROM.update(RV2F_eeprom,RV2F);
    EEPROM.update(RV3F_eeprom,RV3F);
    EEPROM.update(RV4F_eeprom,RV4F);

       if (modoactual==1){
       t=0;
          RV1=EEPROM.read(RV1N_eeprom);
          RV2=EEPROM.read(RV2N_eeprom);
          RV3=EEPROM.read(RV3N_eeprom);
          RV4=EEPROM.read(RV3N_eeprom);
       }
       if (modoactual==2){
       t=0;
          RV1=EEPROM.read(RV1D_eeprom);
          RV2=EEPROM.read(RV2D_eeprom);
          RV3=EEPROM.read(RV3D_eeprom);
          RV4=EEPROM.read(RV3D_eeprom);}
       if (modoactual==3){
          t=0;
          RV1=EEPROM.read(RV1F_eeprom);
          RV2=EEPROM.read(RV2F_eeprom);
          RV3=EEPROM.read(RV3F_eeprom);
          RV4=EEPROM.read(RV3F_eeprom);}
       if (modoactual==4){Intermitente();}

    RV1N_eeprom_valor=EEPROM.read(RV1N_eeprom);
    RV2N_eeprom_valor=EEPROM.read(RV2N_eeprom);
    RV3N_eeprom_valor=EEPROM.read(RV3N_eeprom);
    RV4N_eeprom_valor=EEPROM.read(RV4N_eeprom);
    RA_eeprom_valor=EEPROM.read(RA_eeprom);

    RV1D_eeprom_valor=EEPROM.read(RV1D_eeprom);
    RV2D_eeprom_valor=EEPROM.read(RV2D_eeprom);
    RV3D_eeprom_valor=EEPROM.read(RV3D_eeprom);
    RV4D_eeprom_valor=EEPROM.read(RV4D_eeprom);

    RV1F_eeprom_valor=EEPROM.read(RV1F_eeprom);
    RV2F_eeprom_valor=EEPROM.read(RV2F_eeprom);
    RV3F_eeprom_valor=EEPROM.read(RV3F_eeprom);
    RV4F_eeprom_valor=EEPROM.read(RV4F_eeprom);

        cuentainicio=0;
        configuraInterrupcion1(1,0,0,62499);
    } // último if
  }   // if serial1
}     // loop

uint8_t byteToInt(byte byte1)
{
   return (uint8_t)byte1;
}
```

```
//FUNCIÓN configuraInterrupcion1() ----------------------
void configuraInterrupcion1(boolean b0, boolean b1, boolean b2, int dN) {
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 OCR1A = dN;
 TCCR1B |= (1 << WGM12);
 TCCR1B |= (b2 << CS10);
 TCCR1B |= (b1 << CS11);
 TCCR1B |= (b0 << CS12);
 TIMSK1 = (1 << OCIE1A);
 sei();}

//FUNCIÓN deshabilitaInterrupcion1() -----------------------------
void deshabilitaInterrupcion1(){
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 TCCR1B |= (0 << CS10);
 TCCR1B |= (0 << CS11);
 TCCR1B |= (0 << CS12);
 sei(); }

// INTERRUPCIÓN 1 ------------------------------------------------
ISR(TIMER1_COMPA_vect) {

  if ((cuentainicio>0) && (cuentainicio<=tstart))
        {cuentainicio=cuentainicio+1;}
  if (cuentainicio==tstart+2)
        {cuentainicio=cuentainicio+1;
        ModoNormal();}
  if (cuentainicio==tstart+1)
        {boolean estado[12]={0,0,0,0,0,0,0,0,0,0,0,0};
        cuentainicio=cuentainicio+1;}
  if (cuentainicio==0)
        {ModoStart();
        cuentainicio=cuentainicio+1;}
  if (intermitente==0){
       switch(CantSemaforos){
       case 1:
            if ((t >= 0 && t < RR)||(t>=RV1+RA+RR+RA && t<2*(RV1+RA+RR+RA)))
                  {boolean estado[12]={0,0,1,0,0,0,0,0,0,0,0,0};
                   String datosenviar= "c";
                   EncenderLuces(estado, datosenviar);
                   t = t + 1;}
else if ((t >= RR && t < RR + RA)|| (t >= RV1+RA+RR  && t < RV1+RA+RR+RA ))
                 {boolean estado[12]={0,1,0,0,0,0,0,0,0,0,0,0};
                  String datosenviar= "b";
                  EncenderLuces(estado, datosenviar);
                  t = t + 1;}
            else if (t >= RR + RA && t < RV1 + RA + RR)
                   {boolean estado[12]={1,0,0,0,0,0,0,0,0,0,0,0};
                   String datosenviar= "a";
                    EncenderLuces(estado, datosenviar);
                    t = t + 1;}
            else if (2*(RV1+RA+RR+RA))
                     {t=0;
                     deshabilitaInterrupcion1();
                     configuraInterrupcion1(1,0,0,62499);}
                if (t ==0 )
                     {boolean estado[12]={0,0,1,0,0,0,0,0,0,0,0,0};
```

```
                    String datosenviar= "c";
                    EncenderLuces(estado, datosenviar);
                    t = t + 1;}
                break;// fin cantidad de semaforos 1

            case 2:
if ((t >= 0 && t < RR) || (t >= RV1+RR+(2*RA) && t < RV1+(2*RA)+(2*RR)))
                    {boolean estado[12]={0,0,1,0,0,1,0,0,0,0,0,0};
                     String datosenviar= "f";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}
else if ((t >= RR && t < RR + RA) || (t >= RV1+RA+RR && t < RV1+RA+RR+RA))
                    {boolean estado[12]={0,1,0,0,0,1,0,0,0,0,0,0};
                     String datosenviar= "e";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >= RR+RA && t < RV1+RA+RR))
                    {boolean estado[12]={1,0,0,0,0,1,0,0,0,0,0,0};
                     String datosenviar= "d";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >= RV1+(2*RA)+(2*RR) && t < RV1+(3*RA)+(2*RR)) ||
(t >= RV1+(3*RA)+(2*RR)+RV2 && t < RV1+(4*RA)+(2*RR)+RV2))
                    {boolean estado[12]={0,0,1,0,1,0,0,0,0,0,0,0};
                     String datosenviar= "g";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (t >= RV1+(3*RA)+(2*RR) && t < RV1+(3*RA)+(2*RR)+RV2)
                    {boolean estado[12]={0,0,1,1,0,0,0,0,0,0,0,0};
                     String datosenviar= "h";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (t==RV1+(4*RA)+(2*RR)+RV2)
                    {t=0;
                     deshabilitaInterrupcion1();
                     configuraInterrupcion1(1,0,0,62499);}

if (t == 0)
                    {boolean estado[12]={0,0,1,0,0,1,0,0,0,0,0,0};
                     String datosenviar= "f";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1; }
              break; //case 2

              case 4:
if ((t >= 0 && t < RR)|| (t >= RV1+2*RA+RR && t < RV1+2*RA+2*RR) ||
(t>= RV1+4*RA+2*RR+RV2 && t< RV1+4*RA+3*RR+RV2)||
(t>= RV1+6*RA+3*RR+RV2+RV3 && t< RV1+6*RA+4*RR+RV2+RV3))
                    {boolean estado[12]={0,0,1,0,0,1,0,0,1,0,0,1};
                     String datosenviar= "k";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >= RR && t < RR+RA) || ((t >= RV1+RA+RR) && (t < RV1+2*RA+RR)))
                    {boolean estado[12]={0,1,0,0,0,1,0,0,1,0,0,1};
                     String datosenviar= "j";
                     EncenderLuces(estado, datosenviar);
```

```
                    t = t + 1;}

else if (((t >= RR+RA) && (t < RV1+RA+RR)))
                    {boolean estado[12]={1,0,0,0,0,1,0,0,1,0,0,1};
                     String datosenviar= "i";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (((t >= RV1+2*RA+2*RR) && (t < RV1+3*RA+2*RR)) ||
((t >= RV1+3*RA+2*RR+RV2) && (t < RV1+4*RA+2*RR+RV2)))
                    {boolean estado[12]={0,0,1,0,1,0,0,0,1,0,0,1};
                     String datosenviar= "l";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >= RV1+3*RA+2*RR) && (t < RV1+3*RA+2*RR+RV2))
                    {boolean estado[12]={0,0,1,1,0,0,0,0,1,0,0,1};
                     String datosenviar= "m";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (((t >= RV1+4*RA+3*RR+RV2) && (t < RV1+5*RA+3*RR+RV2)) ||
((t >=  RV1+5*RA+3*RR+RV2+RV3)&&(t < RV1+6*RA+3*RR+RV2+RV3)))
                    {boolean estado[12]={0,0,1,0,0,1,0,1,0,0,0,1};
                     String datosenviar= "n";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >=  RV1+5*RA+3*RR+RV2) && (t < RV1+5*RA+3*RR+RV2+RV3))
                    {boolean estado[12]={0,0,1,0,0,1,1,0,0,0,0,1};
                     String datosenviar= "w";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (((t >=  RV1+6*RA+4*RR+RV2+RV3) && (t < RV1+7*RA+4*RR+RV2+RV3)) ||
((t >=  RV1+7*RA+4*RR+RV2+RV3+RV4) && (t < RV1+8*RA+4*RR+RV2+RV3+RV4)))
                    {boolean estado[12]={0,0,1,0,0,1,0,0,1,0,1,0};
                     String datosenviar= "o";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if ((t >=  RV1+7*RA+4*RR+RV2+RV3) && (t < RV1+7*RA+4*RR+RV2+RV3+RV4))
                    {boolean estado[12]={0,0,1,0,0,1,0,0,1,1,0,0};
                     String datosenviar= "p";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}

else if (t=RV1+8*RA+4*RR+RV2+RV3+RV4)
                    {t=0;
                     deshabilitaInterrupcion1();
                     configuraInterrupcion1(1,0,0,62499);}

if (t == 0)
                    {boolean estado[12]={0,0,1,0,0,1,0,0,1,0,0,1};
                     String datosenviar= "k";
                     EncenderLuces(estado, datosenviar);
                     t = t + 1;}
                break;
            }//switch
  }
```

```
  if (intermitente==1){
    Serial1.flush();
    switch(CantSemaforos){
      case 1:
          if (t >= 0 && t<2)
            {boolean estado[12]={0,1,0,0,0,0,0,0,0,0,0,0};
             String datosenviar= "x";
             EncenderLuces(estado, datosenviar);
             t = t+1;}

          if ( t>=2 && t<4)
            {boolean estado[12]={0,0,0,0,0,0,0,0,0,0,0,0};
             String datosenviar= "x";
             EncenderLuces(estado, datosenviar);
             t = t + 1;}

          if (t>=4)
             {t=0;}
       break;

        case 2;
         if (t >= 0 && t<2)
             {boolean estado[12]={0,1,0,0,1,0,0,0,0,0,0,0};
              String datosenviar= "x";
              EncenderLuces(estado, datosenviar);
              t = t+1;}

         if ( t>=2 && t<4)
             {boolean estado[12]={0,0,0,0,0,0,0,0,0,0,0,0};
              String datosenviar= "x";
              EncenderLuces(estado, datosenviar);
              t = t + 1;}

         if (t>=4)
             {t=0;}
      break;

         case 4:
         if (t >= 0 && t<2)
             {boolean estado[12]={0,1,0,0,1,0,0,1,0,0,1,0};
              String datosenviar= "x";
              EncenderLuces(estado, datosenviar);
              t = t+1;}

         if ( t>=2 && t<4)
             {boolean estado[12]={0,0,0,0,0,0,0,0,0,0,0,0};
              String datosenviar= "x";
              EncenderLuces(estado, datosenviar);
              t = t + 1;}

         if (t>=4)
             {t=0;}
         break;} }
}// FIN DE LA INTERRUPCIÓN
```

```
// FUNCIONES ---------------------------------------------------
void EncenderLuces(bool estado[12], String datosenviar){
            bool a;
            bool b;
            bool c;
            bool d;
            bool e;
            bool f;
            bool g;
            bool h;
            bool i;
            bool j;
            bool k;
            bool l;
            if(estado[0]==0){a=1;}
            else {a=0;}
            if(estado[1]==0){b=1;}
            else {b=0;}
            if(estado[2]==0){c=1;}
            else {c=0;}
            if(estado[3]==0){d=1;}
            else {d=0;}
            if(estado[4]==0){e=1;}
            else {e=0;}
            if(estado[5]==0){f=1;}
            else {f=0;}
            if(estado[6]==0){g=1;}
            else {g=0;}
            if(estado[7]==0){h=1;}
            else {h=0;}
            if(estado[8]==0){i=1;}
            else {i=0;}
            if(estado[9]==0){j=1;}
            else {j=0;}
            if(estado[10]==0){k=1;}
            else {k=0;}
            if(estado[11]==0){l=1;}
            else {l=0;}

            digitalWrite(23,a);
            digitalWrite(25,b);
            digitalWrite(27,c);
            digitalWrite(29,d);
            digitalWrite(31,e);
            digitalWrite(33,f);
            digitalWrite(35,g);
            digitalWrite(37,h);
            digitalWrite(39,i);
             digitalWrite(41,j);
             digitalWrite(43,k);
             digitalWrite(45,l);

             digitalWrite(22,estado[0]);
             digitalWrite(24,estado[1]);
             digitalWrite(26,estado[2]);
             digitalWrite(28,estado[3]);
             digitalWrite(30,estado[4]);
             digitalWrite(32,estado[5]);
             digitalWrite(34,estado[6]);
             digitalWrite(36,estado[7]);
             digitalWrite(38,estado[8]);
```

```
            digitalWrite(40,estado[9]);
            digitalWrite(42,estado[10]);
            digitalWrite(44,estado[11]);

          Serial1.print(datosenviar);}

void ModoStart(){
  intermitente=1;
  t=0;
  RV1=RV1N_eeprom_valor;
  RV2=RV2N_eeprom_valor;
  RV3=RV3N_eeprom_valor;
  RV4=RV4N_eeprom_valor;
  modoactual=1;}

void ModoNormal(){
  modoactual=1;
  intermitente=0;
  t=0;
  RV1=EEPROM.read(RV1N_eeprom);
  RV2=EEPROM.read(RV2N_eeprom);
  RV3=EEPROM.read(RV3N_eeprom);
  RV4=EEPROM.read(RV3N_eeprom);}

void ModoDiaDistinto(){
  modoactual=2;
  intermitente=0;
  t=0;
  RV1=EEPROM.read(RV1D_eeprom);
  RV2=EEPROM.read(RV2D_eeprom);
  RV3=EEPROM.read(RV3D_eeprom);
  RV4=EEPROM.read(RV3D_eeprom);}

void ModoFeriado(){
  modoactual=3;
  intermitente=0;
  t=0;
  RV1=EEPROM.read(RV1F_eeprom);
  RV2=EEPROM.read(RV2F_eeprom);
  RV3=EEPROM.read(RV3F_eeprom);
  RV4=EEPROM.read(RV3F_eeprom);
  sumatiemposF=RV1+RV2+RV3+RV4;
  if (sumatiemposF==0){intermitente=1;}}

void Intermitente(){
  modoactual=4;
  t=0;
  intermitente=1;
  cuentainicio=tstart+3;}
```

ANEXO B: Código de programación Arduino Comunicación

```
                        ////////////////////////////
                        //    BIDEGAIN OCTAVIO    //
                        //    VERSIÓN 21/11/2019  //
                        // CONTROLADOR DE SEMAFOROS //
                        ////////////////////////////

// LIBRERIAS
#include <Ethernet.h>
#include <Wire.h>
#include <EEPROM.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>

// ETHERNET
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192,168,1,37);
EthernetServer server(80);

// VARIABLES
int T=0;
int estado;
int estado1;
long T0 = 0 ;
volatile int menu;
int errorsemaforo;
int error=0;

// VARIABLES MENU ------------------------------------------
int entrar;
int salir;
int Entrar;
int Entrarr;
int Posicion;
int posicionmenunormal;
int posicionferiado;
int posiciondia;
int BotonOk;
int BotonArriba;
int BotonAbajo;

// VARIABLES MODO NORMAL -----------------------------------
int CantSemaforos;
int RV1;
int RV2;
int RV3;
int RV4;
int RR;
int RA;
int Desfasaje;

// VARIABLES DIA DE LA SEMANA DISTINTO ---------------------
String diasemana[] =
{"Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Todos
los dias","Ningun dia"};
int RV1D;
int RV2D;
int RV3D;
int RV4D;
int DesfasajeD;
int diaD;
int horaDOn;
```

```
int minutoDOn;
int horaDOff;
int minutoDOff;

// VARIABLES FERIADO ---------------------------------------------
int RV1F;
int RV2F;
int RV3F;
int RV4F;
int DesfasajeF;
int diaF;
int mesF;

int Ciclo;
int CicloD;
int CicloF;
int t;

int OnInt=0; // 0 solo para prueba
int OffInt=0; // 0 solo para prueba
int D2, D3, D6, D8;
int corrient;
int j;

// VARIABLES EEPROM MODO NORMAL ----------------------------------
int CantSemaforos_eeprom=1
int RV1_eeprom=2
int RA_eeprom=3;
int RV2_eeprom=5;
int RV3_eeprom=6;
int RV4_eeprom=7;
int Desfasaje_eeprom=8;
int Ciclo_eeprom=26;

byte CantSemaforos_eeprom_valor;

byte RV1_eeprom_valor;
byte RV2_eeprom_valor;
byte RV3_eeprom_valor;
byte RV4_eeprom_valor;
byte RA_eeprom_valor;
byte Desfasaje_eeprom_valor;
int Ciclo_eeprom_valor;

// VARIABLES EEPROM MODO DIA DE LA SEMANA DISTINTO ---------------

int RV1D_eeprom=9;
int RV2D_eeprom=10;
int RV3D_eeprom=11;
int RV4D_eeprom=12;
int DesfasajeD_eeprom=13;
int diaD_eeprom=14;
int horaDOn_eeprom=22;
int minutoDOn_eeprom=23;
int horaDOff_eeprom=24;
int minutoDOff_eeprom=25;

byte RV1D_eeprom_valor;
byte RV2D_eeprom_valor;
byte RV3D_eeprom_valor;
byte RV4D_eeprom_valor;
```

```
byte DesfasajeD_eeprom_valor;
byte diaD_eeprom_valor;
byte horaDOn_eeprom_valor;
byte minutoDOn_eeprom_valor;
byte horaDOff_eeprom_valor;
byte minutoDOff_eeprom_valor;

// VARIABLES EEPROM MODO FERIADO --------------------------------

int RV1F_eeprom=15;
int RV2F_eeprom=16;
int RV3F_eeprom=17;
int RV4F_eeprom=18;
int DesfasajeF_eeprom=19;
int diaF_eeprom=20;
int mesF_eeprom=21;

byte RV1F_eeprom_valor;
byte RV2F_eeprom_valor;
byte RV3F_eeprom_valor;
byte RV4F_eeprom_valor;
byte DesfasajeF_eeprom_valor;
byte diaF_eeprom_valor;
byte mesF_eeprom_valor;


byte MS_eeprom_valor;
LiquidCrystal_I2C lcd(0x27,39,4);

 // VARIABLES
 char caracterEntrada;
 int entradaDigital, b0, b1, b2, b3, dato;
 byte vector1[2], vector2[10];
 String canal;
 int contador;
 int i;
 int dato1, dato2, dato3, dato4;

void setup() {

//INICIALIZAR LCD

 lcd.init();
 errorsemaforo=0;
 BotonOk = 0;
 Posicion = 0;
 posicionmenunormal=0;
 posicionferiado=0;
 CantSemaforos = 2;
 t=0;
 lcd.noBacklight();
 lcd.clear();
 Entrar = 0;
 Entrarr=0;
 j=0;
 menu=0;
 RR=1;

//INICIALIZAR COMUNICACIÓN CON ARDUINO CONTROLADOR
  Serial1.begin(9600);
  Serial.begin(9600);
```

```
  Serial3.begin(115200);

//BOTONES
    pinMode (8,INPUT);                          //BOTONOK
      pinMode (6,INPUT);                        //BOTONARRIBA
        pinMode (3,INPUT);                      //BOTONABAJO

// INICIALIZAR CON LAS VARIABLES DE LA MEMORIA EEPROM
    CantSemaforos_eeprom_valor=EEPROM.read(CantSemaforos_eeprom);

    RV1_eeprom_valor=EEPROM.read(RV1_eeprom);
    RV2_eeprom_valor=EEPROM.read(RV2_eeprom);
    RV3_eeprom_valor=EEPROM.read(RV3_eeprom);
    RV4_eeprom_valor=EEPROM.read(RV4_eeprom);
    RA_eeprom_valor=EEPROM.read(RA_eeprom);
    Desfasaje_eeprom_valor=EEPROM.read(Desfasaje_eeprom);
    Ciclo_eeprom_valor=EEPROM.read(Ciclo_eeprom);

    RV1D_eeprom_valor=EEPROM.read(RV1D_eeprom);
    RV2D_eeprom_valor=EEPROM.read(RV2D_eeprom);
    RV3D_eeprom_valor=EEPROM.read(RV3D_eeprom);
    RV4D_eeprom_valor=EEPROM.read(RV4D_eeprom);
    DesfasajeD_eeprom_valor=EEPROM.read(DesfasajeD_eeprom);
    diaD_eeprom_valor=EEPROM.read(diaD_eeprom);
    horaDOn_eeprom_valor=EEPROM.read(horaDOn_eeprom);
    minutoDOn_eeprom_valor=EEPROM.read(minutoDOn_eeprom);
    horaDOff_eeprom_valor=EEPROM.read(horaDOff_eeprom);
    minutoDOff_eeprom_valor=EEPROM.read(minutoDOff_eeprom);

    RV1F_eeprom_valor=EEPROM.read(RV1F_eeprom);
    RV2F_eeprom_valor=EEPROM.read(RV2F_eeprom);
    RV3F_eeprom_valor=EEPROM.read(RV3F_eeprom);
    RV4F_eeprom_valor=EEPROM.read(RV4F_eeprom);
    DesfasajeF_eeprom_valor=EEPROM.read(DesfasajeF_eeprom);
    diaF_eeprom_valor=EEPROM.read(diaF_eeprom);
    mesF_eeprom_valor=EEPROM.read(mesF_eeprom);

    if (RV1_eeprom_valor==255){RV1_eeprom_valor=0;}
    if (RV2_eeprom_valor==255){RV2_eeprom_valor=0;}
    if (RV3_eeprom_valor==255){RV3_eeprom_valor=0;}
    if (RV4_eeprom_valor==255){RV4_eeprom_valor=0;}
    if (Desfasaje_eeprom_valor==255){Desfasaje_eeprom_valor=0;}
    if (Ciclo_eeprom_valor==255){Ciclo_eeprom_valor=0;}

    if (RV1D_eeprom_valor==255){RV1D_eeprom_valor=0;}
    if (RV2D_eeprom_valor==255){RV2D_eeprom_valor=0;}
    if (RV3D_eeprom_valor==255){RV3D_eeprom_valor=0;}
    if (RV4D_eeprom_valor==255){RV4D_eeprom_valor=0;}
    if (DesfasajeD_eeprom_valor==255){DesfasajeD_eeprom_valor=0;}
    if (diaD_eeprom_valor==255){diaD_eeprom_valor=8;}
    if (horaDOn_eeprom_valor==255){horaDOn_eeprom_valor=0;}
    if (minutoDOn_eeprom_valor==255){minutoDOn_eeprom_valor=0;}
    if (horaDOff_eeprom_valor==255){horaDOff_eeprom_valor=0;}
    if (minutoDOff_eeprom_valor==255){minutoDOff_eeprom_valor=0;}

    if (RV1F_eeprom_valor==255){RV1F_eeprom_valor=0;}
    if (RV2F_eeprom_valor==255){RV2F_eeprom_valor=0;}
    if (RV3F_eeprom_valor==255){RV3F_eeprom_valor=0;}
    if (RV4F_eeprom_valor==255){RV4F_eeprom_valor=0;}
    if (DesfasajeF_eeprom_valor==255){DesfasajeF_eeprom_valor=0;}
    if (diaF_eeprom_valor==255){diaF_eeprom_valor=0;}
```

```
    if (mesF_eeprom_valor==255){mesF_eeprom_valor=0;}

    CantSemaforos=CantSemaforos_eeprom_valor;
    RV1=RV1_eeprom_valor;
    RV2=RV2_eeprom_valor;
    RV3=RV3_eeprom_valor;
    RV4=RV4_eeprom_valor;
    RA=RA_eeprom_valor;
    Desfasaje=Desfasaje_eeprom_valor;
    Ciclo=Ciclo_eeprom_valor;

    RV1D=RV1D_eeprom_valor;
    RV2D=RV2D_eeprom_valor;
    RV3D=RV3D_eeprom_valor;
    RV4D=RV4D_eeprom_valor;
    DesfasajeD=DesfasajeD_eeprom_valor;
    diaD=diaD_eeprom_valor;
    horaDOn=horaDOn_eeprom_valor;
    minutoDOn=minutoDOn_eeprom_valor;
    horaDOff=horaDOff_eeprom_valor;
    minutoDOff=minutoDOff_eeprom_valor;

    RV1F=RV1F_eeprom_valor;
    RV2F=RV2F_eeprom_valor;
    RV3F=RV3F_eeprom_valor;
    RV4F=RV4F_eeprom_valor;
    DesfasajeF=DesfasajeF_eeprom_valor;
    diaF=diaF_eeprom_valor;
    mesF=mesF_eeprom_valor;


// INICIALIZAR ETHERNET
  Ethernet.begin(mac, ip);
  server.begin();
  configuraInterrupcion1(1,0,0,62499);
}  // Setup

// LOOP ------------------------------------------------------------
void loop() {
  EthernetClient client = server.available();
  if (client) {
    byte empty = 0;


    if (client.connected()) {
      while (client.connected()) {

        switch(client.read()){
          case 'a':
          if (error==1){client.flush();}
          server.write(estado);
          if (boton(8)==1){MostrarMenu(1);}
          break;

          case 'b':
          CantSemaforos=client.read()-48;
          break;

          case 'c':
          RV1=client.read()-48;  //RV1
          break;
```

```
case 'd':
RA=client.read()-48;    //RA
break;

case 'e':
RV2=client.read()-48;    //RV2
break;

case 'f':
RV3=client.read()-48;
break;

case 'g':
RV4=client.read()-48;    //RV4

if (CantSemaforos==2){
            Ciclo=((2*RR) + (4*RA) + RV1 + RV2);
            CicloD=((2*RR) + (4*RA) + RV1D + RV2D);
            CicloF=((2*RR) + (4*RA) + RV1F + RV2F);
                  }
if (CantSemaforos==3){
            Ciclo=((3*RR) + (6*RA) + RV1 + RV2 + RV3);
            CicloD=((3*RR) + (6*RA) + RV1D + RV2D + RV3D);
            CicloF=((3*RR) + (6*RA) + RV1F + RV2F + RV3F);
                  }
if (CantSemaforos==4){
            Ciclo=((4*RR) + (8*RA) + RV1 + RV2 + RV3 + RV4);
            Ciclo=((4*RR) + (8*RA) + RV1D + RV2D + RV3D + RV4D);
            Ciclo=((4*RR) + (8*RA) + RV1F + RV2F + RV3F + RV4F);
                  }
sendBytes(CantSemaforos);
sendBytes(RV1);
sendBytes(RA);
sendBytes(RV2);
sendBytes(RV3);
sendBytes(RV4);

sendBytes(RV1D);
sendBytes(RV2D);
sendBytes(RV3D);
sendBytes(RV4D);

sendBytes(RV1F);
sendBytes(RV2F);
sendBytes(RV3F);
sendBytes(RV4F);

sendBytesGps(Desfasaje);

sendBytesGps(CicloD);
sendBytesGps(DesfasajeD);
sendBytesGps(diaD);
sendBytesGps(horaDOn);
sendBytesGps(minutoDOn);
sendBytesGps(horaDOff);
sendBytesGps(minutoDOff);

sendBytesGps(CicloF);
sendBytesGps(DesfasajeF);
sendBytesGps(diaF);
```

```
          sendBytesGps(mesF);
          break;

      }  // switch
    }    //while
    }    //if client.connected()

    if (boton(8)==1){MostrarMenu(1);}
    Serial.println("You lost your client");
    client.stop();
    }//if (client)
    else {
            if (boton(8)==1){MostrarMenu(1);}

      } //else
}       // fin del loop


//FUNCIÓN configuraInterrupcion1() ---------------------------------
void configuraInterrupcion1(boolean b0, boolean b1, boolean b2, int dN) {
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 OCR1A = dN;
 TCCR1B |= (1 << WGM12);
 TCCR1B |= (b2 << CS10);
 TCCR1B |= (b1 << CS11);
 TCCR1B |= (b0 << CS12);
 TIMSK1 = (1 << OCIE1A);
 sei();
}

//FUNCIÓN deshabilitaInterrupcion1() -------------------------------
void deshabilitaInterrupcion1(){
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 TCCR1B |= (0 << CS10);
 TCCR1B |= (0 << CS11);
 TCCR1B |= (0 << CS12);
 sei();
}

// INTERRUPCIÓN 1 ------------------------------------------------
ISR(TIMER1_COMPA_vect) {
          if (Serial1.available() > 0)
          {
          estado = Serial1.read();
              if (estado=='x'){error=1;}
              else {error==0;}
          }
}

// MENU ----------------------------------------------------------
void MostrarMenu (int entrar){
  while (entrar==1){
  noInterrupts();
  deshabilitaInterrupcion1();

         if (Posicion==0){
           lcd.clear();
```

```
      lcd.noBlink();
      lcd.backlight();
      lcd.setCursor(0,1);
      lcd.print("        MENU        ");
      delay(500); }
      if (boton(6)==1 and Posicion<12){
         Posicion=Posicion+1;}
      if (boton(5)==1 and Posicion>1){
         Posicion=Posicion-1;}

// ------ CADA UNA DE LAS OPCIONES DE MENU
switch(Posicion){
  case 1:
        imprimir_1();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                        Entrar=1;
                        posicionmenunormal=1;
                        delay(500);}
        while (Entrar==1){
            if (boton(6)==1 and posicionmenunormal<8){
                 posicionmenunormal=posicionmenunormal+1;}
            if (boton(5)==1 and posicionmenunormal>1){
                 posicionmenunormal=posicionmenunormal-1; }
            switch(posicionmenunormal){
               case 1:
                   imprimir_CantSemaforos();
                   lcd.noBlink();
                   editar();
                   if (boton(8)==1){
                           Entrarr=1;
                           delay(500);}
                   while (Entrarr==1){
                                   aceptar();
                                   if (boton(6)==1 and CantSemaforos<=4){
                                     CantSemaforos=CantSemaforos+1;
                                     imprimir_CantSemaforos();
                                     aceptar();}
                                   if (boton(5)==1 and CantSemaforos>0){
                                     CantSemaforos=CantSemaforos-1;
                                     imprimir_CantSemaforos();
                                     aceptar();}
                   if (boton(8)==1){Entrarr=0;}}
     break;

     case 2:
           imprimir_RV1();
           lcd.noBlink();
           editar();
           if (boton(8)==1){
                           Entrarr=1;
                           delay(500);}
           while (Entrarr==1){
                              aceptar();
                              if (boton(6)==1 and RV1<30){
                                   RV1=RV1+1;
                                   imprimir_RV1();
                                   aceptar();}
                              if (boton(5)==1 and RV1>0){
                                   RV1=RV1-1;
```

```
                                    imprimir_RV1();
                                    aceptar();}
                                if (boton(8)==1) {Entrarr=0;} }
break;

case 3:
       imprimir_RV2();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
       while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and RV2<30){
                                RV2=RV2+1;
                                imprimir_RV2();
                                aceptar();}
                          if (boton(5)==1 and RV2>0){
                                RV2=RV2-1;
                                imprimir_RV2();
                                aceptar();}
                          if (boton(8)==1) {Entrarr=0;} }
break;

case 4:
       imprimir_RV3();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
       while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and RV3<30){
                                RV3=RV3+1;
                                imprimir_RV3();
                                aceptar();}
                          if (boton(5)==1 and RV3>0){
                                RV3=RV3-1;
                                imprimir_RV3();
                                aceptar();}
                          if (boton(8)==1) {Entrarr=0;}}
break;

case 5:
       imprimir_RV4();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and RV4<30){
                                RV4=RV4+1;
                                imprimir_RV4();
                                aceptar();}
                        if (boton(5)==1 and RV4>0){
                                RV4=RV4-1;
                                imprimir_RV4();
```

```
                                aceptar();}
                        if (boton(8)==1) {Entrarr=0;}}
  break;

  case 6:
         imprimir_RA();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
         while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and RA<30){
                                  RA=RA+1;
                                  imprimir_RA();
                                  aceptar();}
                          if (boton(5)==1 and RA>0){
                                  RA=RA-1;
                                  imprimir_RA();
                                  aceptar();}
                          if (boton(8)==1) {Entrarr=0;}}
  break;

  case 7:
         imprimir_Desfasaje();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
         while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and Desfasaje<30){
                                  Desfasaje=Desfasaje+1;
                                  imprimir_Desfasaje();
                                  aceptar();}
                          if (boton(5)==1 and Desfasaje>0){
                                  Desfasaje=Desfasaje-1;
                                  imprimir_Desfasaje();
                                  aceptar();}
                          if (boton(8)==1) {Entrarr=0;}}
  break;

  case 8:
         imprimir_volver();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                        Entrar=0;
                        delay(500);
                        posicionmenunormal=1;}
break;
    }// SWITCH POSICIONMENUNORMAL
    } // WHILE
break;
//-------------------------------------------------------------
case 2:
       imprimir_2();
       lcd.noBlink();
       editar();
```

```
          if (boton(8)==1){
                           Entrar=1;
                           posicionferiado=1;
                           delay(500);}
          while (Entrar==1){
             if (boton(6)==1 and posicionferiado<8){
               posicionferiado=posicionferiado+1;}
             if (boton(5)==1 and posicionferiado>1){
               posicionferiado=posicionferiado-1;}
          switch(posicionferiado){

case 1:
        imprimir_RV1F();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                          Entrarr=1;
                          delay(500);}
        while (Entrarr==1){
                         aceptar();
                         if (boton(6)==1 and RV1F<30){
                                     RV1F=RV1F+1;
                                     imprimir_RV1F();
                                     aceptar();}
                                     }
                         if (boton(5)==1 and RV1F>0){
                                     RV1F=RV1F-1;
                                     imprimir_RV1F();
                                     aceptar();}
                         if (boton(8)==1) {Entrarr=0;} }
  break;

  case 2:
          imprimir_RV2F();
          lcd.noBlink();
          editar();
          if (boton(8)==1){
                           Entrarr=1;
                           delay(500);}
          while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and RV2F<30){
                                      RV2F=RV2F+1;
                                      imprimir_RV2F();
                                      aceptar();}
                          if (boton(5)==1 and RV2F>0){
                                      RV2F=RV2F-1;
                                      imprimir_RV2F();
                                      aceptar();}
                          if (boton(8)==1) {Entrarr=0;} }
  break;

  case 3:
          imprimir_RV3F();
          lcd.noBlink();
          editar();
          if (boton(8)==1){
                           Entrarr=1;
                           delay(500);}
          while (Entrarr==1){
                           aceptar();
```

```
                        if (boton(6)==1 and RV3F<30){
                                RV3F=RV3F+1;
                                imprimir_RV3F();
                                aceptar();}
                        if (boton(5)==1 and RV3F>0){
                                RV3F=RV3F-1;
                                imprimir_RV3F();
                                aceptar();}
                        if (boton(8)==1) {Entrarr=0;} }
break;

case 4:
       imprimir_RV4F();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and RV4F<30){
                                RV4F=RV4F+1;
                                imprimir_RV4F();
                                aceptar();}
                        if (boton(5)==1 and RV4F>0){
                                RV4F=RV4F-1;
                                imprimir_RV4F();
                                aceptar();}
                        if (boton(8)==1) {Entrarr=0;}}
 break;

case 5:
       imprimir_DesfasajeF();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                         aceptar();
                         if (boton(6)==1 and DesfasajeF<30){
                                 DesfasajeF=DesfasajeF+1;
                                 imprimir_DesfasajeF();
                                 aceptar();}
                         if (boton(5)==1 and DesfasajeF>0){
                                 DesfasajeF=DesfasajeF-1;
                                 imprimir_DesfasajeF();
                                 aceptar();}
                         if (boton(8)==1){Entrarr=0;} }
break;

case 6:
       imprimir_diaF();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and diaF<31){
```

```
                                    diaF=diaF+1;
                                    imprimir_diaF();
                                    aceptar();}
                             if (boton(5)==1 and diaF>0){
                                    diaF=diaF-1;
                                    imprimir_diaF();
                                    aceptar();}
                             if (boton(8)==1) {Entrarr=0;} }
  break;

  case 7:
         imprimir_mesF();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                           Entrarr=1;
                           delay(500);}
         while (Entrarr==1){
                            aceptar();
                            if (boton(6)==1 and mesF<12){
                                    mesF=mesF+1;
                                    imprimir_mesF();
                                    aceptar();}
                            if (boton(5)==1 and mesF>0){
                                    mesF=mesF-1;
                                    imprimir_mesF();
                                    aceptar();}
                            if (boton(8)==1) {Entrarr=0;}}
  break;

  case 8:
         imprimir_volver();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                           Entrar=0;
                           delay(500);
                           posicionferiado=1;}
  break;
      } // SWITCH POSICIONFERIADO
     } // WHILE
break;
-------------------------------------------------------------------------
case 3:
       imprimir_DiaDistinto();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                         Entrar=1;
                         posiciondia=1;
                         delay(500);}
       while (Entrar==1){
         if (boton(6)==1 and posiciondia<11){
            posiciondia=posiciondia+1;}
         if (boton(5)==1 and posiciondia>1){
            posiciondia=posiciondia-1;}
       switch(posiciondia){
case 1:
       imprimir_RV1D();
       lcd.noBlink();
       editar();
```

```
        if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
        while (Entrarr==1){
                         aceptar();
                         if (boton(6)==1 and RV1D<30){
                                 RV1D=RV1D+1;
                                 imprimir_RV1D();
                                 aceptar();}
                         if (boton(5)==1 and RV1D>0){
                                 RV1D=RV1D-1;
                                 imprimir_RV1D();
                                 aceptar();}
                         if (boton(8)==1) {Entrarr=0;} }
break;

case 2:
       imprimir_RV2D();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and RV2D<30){
                                RV2D=RV2D+1;
                                imprimir_RV2D();
                                aceptar();}
                        if (boton(5)==1 and RV2D>0){
                                RV2D=RV2D-1;
                                imprimir_RV2D();
                                aceptar();}
                        if (boton(8)==1) {Entrarr=0;} }
break;

case 3:
       imprimir_RV3D();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and RV3D<30){
                               RV3D=RV3D+1;
                               imprimir_RV3D();
                               aceptar();}
                        if (boton(5)==1 and RV3D>0){
                               RV3D=RV3D-1;
                               imprimir_RV3D();
                               aceptar();}
                        if (boton(8)==1) {Entrarr=0;}}
break;

case 4:
       imprimir_RV4D();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
```

```
                        Entrarr=1;
                        delay(500);}
         while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and RV4D<30){
                                  RV4D=RV4D+1;
                                  imprimir_RV4D();
                                  aceptar();}
                          if (boton(5)==1 and RV4D>0){
                                  RV4D=RV4D-1;
                                  imprimir_RV4D();
                                  aceptar();}
                          if (boton(8)==1) {Entrarr=0;} }
break;

case 5:
        imprimir_DesfasajeD();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
        while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and DesfasajeD<30){
                                  DesfasajeD=DesfasajeD+1;
                                  imprimir_DesfasajeD();
                                  aceptar();}
                          if (boton(5)==1 and DesfasajeD>0){
                                  DesfasajeD=DesfasajeD-1;
                                  imprimir_DesfasajeD();
                                  aceptar();}
                          if (boton(8)==1) {Entrarr=0;}}
break;

case 6:
        imprimir_diaD();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
        while (Entrarr==1){
                          aceptar();
                          if (boton(6)==1 and diaD<8){
                                  diaD=diaD+1;
                                  imprimir_diaD();
                                  aceptar();}
                          if (boton(5)==1 and diaD>0){
                                  diaD=diaD-1;
                                  imprimir_diaD();
                                  aceptar();}
                          if (boton(8)==1) {Entrarr=0;}}
break;

 case 7:
         imprimir_horaDOn();
         lcd.noBlink();
         editar();
         if (boton(8)==1){
                         Entrarr=1;
```

```
                        delay(500);}
          while (Entrarr==1){
                              aceptar();
                              if (boton(6)==1 and horaDOn<24){
                                    horaDOn=horaDOn+1;
                                    imprimir_horaDOn();
                                    aceptar();}
                              if (boton(5)==1 and horaDOn>0){
                                    horaDOn=horaDOn-1;
                                    imprimir_horaDOn();
                                    aceptar();}
                              if (boton(8)==1) {Entrarr=0;}}
break;

case 8:
       imprimir_minutoDOn();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                        Entrarr=1;
                        delay(500);}
       while (Entrarr==1){
                        aceptar();
                        if (boton(6)==1 and minutoDOn<60){
                              minutoDOn=minutoDOn+1;
                              imprimir_minutoDOn();
                              aceptar();}
                        if (boton(5)==1 and minutoDOn>0){
                              minutoDOn=minutoDOn-1;
                              imprimir_minutoDOn();
                              aceptar();}
                        if (boton(8)==1) {Entrarr=0;} }
break;

case 9:
       imprimir_horaDOff();
       lcd.noBlink();
       editar();
       if (boton(8)==1){
                       Entrarr=1;
                       delay(500);}
       while (Entrarr==1){
                         aceptar();
                         if (boton(6)==1 and horaDOff<24){
                               horaDOff=horaDOff+1;
                               imprimir_horaDOff();
                               aceptar();}
                         if (boton(5)==1 and horaDOff>0){
                               horaDOff=horaDOff-1;
                               imprimir_horaDOff();
                               aceptar();}
                         if (boton(8)==1) {Entrarr=0;}}
break;

case 10:
        imprimir_minutoDOff();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                         Entrarr=1;
                         delay(500);}
```

```
        while (Entrarr==1){
                              aceptar();
                              if (boton(6)==1 and minutoDOff<60){
                                    minutoDOff=minutoDOff+1;
                                    imprimir_minutoDOff();
                                    aceptar();}
                              if (boton(5)==1 and minutoDOff>0){
                                    minutoDOff=minutoDOff-1;
                                    imprimir_minutoDOff();
                                    aceptar();}
                              if (boton(8)==1) {Entrarr=0;}}
break;

case 11:
        imprimir_volver();
        lcd.noBlink();
        editar();
        if (boton(8)==1){
                           Entrar=0;
                           delay(500);
                           posiciondia=1;}
  break;
      } // SWITCH POSICIONFERIADO
     } // WHILE
break;
-------------------------------------------------------------------------
case 4:
              CantSemaforos_eeprom_valor=EEPROM.read(CantSemaforos_eeprom);
                      RV1_eeprom_valor=EEPROM.read(RV1_eeprom);
                      RV2_eeprom_valor=EEPROM.read(RV2_eeprom);
                      RV3_eeprom_valor=EEPROM.read(RV3_eeprom);
                      RV4_eeprom_valor=EEPROM.read(RV4_eeprom);
                      RA_eeprom_valor=EEPROM.read(RA_eeprom);
                      Desfasaje_eeprom_valor=EEPROM.read(Desfasaje_eeprom);
                      Ciclo_eeprom_valor=EEPROM.read(Ciclo_eeprom);

                      RV1D_eeprom_valor=EEPROM.read(RV1D_eeprom);
                      RV2D_eeprom_valor=EEPROM.read(RV2D_eeprom);
                      RV3D_eeprom_valor=EEPROM.read(RV3D_eeprom);
                      RV4D_eeprom_valor=EEPROM.read(RV4F_eeprom);
                    DesfasajeD_eeprom_valor=EEPROM.read(DesfasajeD_eeprom);
                      diaD_eeprom_valor=EEPROM.read(diaD_eeprom);
                      horaDOn_eeprom_valor=EEPROM.read(horaDOn_eeprom);
                      minutoDOn_eeprom_valor=EEPROM.read(minutoDOn_eeprom);
                      horaDOff_eeprom_valor=EEPROM.read(horaDOff_eeprom);
                    minutoDOff_eeprom_valor=EEPROM.read(minutoDOff_eeprom);
                      RV1F_eeprom_valor=EEPROM.read(RV1F_eeprom);
                      RV2F_eeprom_valor=EEPROM.read(RV2F_eeprom);
                      RV3F_eeprom_valor=EEPROM.read(RV3F_eeprom);
                      RV4F_eeprom_valor=EEPROM.read(RV4F_eeprom);
                    DesfasajeF_eeprom_valor=EEPROM.read(DesfasajeF_eeprom);
                      diaF_eeprom_valor=EEPROM.read(diaF_eeprom);
                      mesF_eeprom_valor=EEPROM.read(mesF_eeprom);


                     // borrar();
                      lcd.setCursor(0,0);
                      lcd.backlight();
                      lcd.print("  VALORES DEFAULT   ");
                      lcd.setCursor(0,1);
                      lcd.print("CANT SEMAF:  ");
```

```
  lcd.print(CantSemaforos_eeprom_valor);
  lcd.print("      ");
  lcd.setCursor(0,2);
  lcd.print("TIEMPO VERDE1: ");
  lcd.print(RV1_eeprom_valor);
  lcd.print("   ");
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("TIEMPO VERDE2: ");
  lcd.print(RV2_eeprom_valor);
  lcd.print("   ");
break;

case 5:
  lcd.setCursor(0,0);
  lcd.backlight();
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(0,1);
  lcd.print("TIEMPO VERDE3: ");
  lcd.print(RV3_eeprom_valor);
  lcd.print("   ");
  lcd.setCursor(0,2);
  lcd.print("TIEMPO VERDE4: ");
  lcd.print(RV4_eeprom_valor);
  lcd.print("   ");
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("TIEMPO AMARILLO: ");
  lcd.print(RA_eeprom_valor);
  lcd.print(" ");
break;

 case 6:
  lcd.setCursor(0,0);
  lcd.backlight();
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(0,1);
  lcd.print("DESFASAJE: ");
  lcd.print(Desfasaje_eeprom_valor);
  lcd.print("   ");
  lcd.setCursor(0,2);
  lcd.print("FECHA FERIADO ");
  lcd.print(diaF_eeprom_valor);
  lcd.print("/");
  lcd.print(mesF_eeprom_valor);
  lcd.print(" ");
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("F-TIEMPO VERDE1: ");
  lcd.print(RV1F_eeprom_valor);
  lcd.print(" ");


break;

case 7:
  lcd.setCursor(0,0);
  lcd.backlight();
```

```
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(0,1);
  lcd.print("F-TIEMPO VERDE2: ");
  lcd.print(RV2F_eeprom_valor);
  lcd.print(" ");
  lcd.setCursor(0,2);
  lcd.print("F-TIEMPO VERDE3: ");
  lcd.print(RV3F_eeprom_valor);
  lcd.print(" ");
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("F-TIEMPO VERDE4: ");
  lcd.print(RV4F_eeprom_valor);
  lcd.print(" ");
break;

case 8:
  lcd.setCursor(0,0);
  lcd.backlight();
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(0,1);
  lcd.print("F-DESFASAJE: ");
  lcd.print(DesfasajeF_eeprom_valor);
  lcd.print("    ");
  lcd.setCursor(0,2);
  lcd.print("                    ");
  lcd.setCursor(0,2);
  lcd.print("DIA:");
  lcd.print(diasemana[diaD_eeprom_valor]);
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("D-TIEMPO VERDE1: ");
  lcd.print(RV1D_eeprom_valor);
  lcd.print(" ");

break;

case 9:
  lcd.setCursor(0,0);
  lcd.backlight();
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(0,1);
  lcd.print("D-TIEMPO VERDE2: ");
  lcd.print(RV2D_eeprom_valor);
  lcd.print(" ");
  lcd.setCursor(0,2);
  lcd.print("D-TIEMPO VERDE3: ");
  lcd.print(RV3D_eeprom_valor);
  lcd.print(" ");
  lcd.setCursor(19,3);
  lcd.print(" ");
  lcd.setCursor(0,3);
  lcd.print("D-TIEMPO VERDE4: ");
  lcd.print(RV4D_eeprom_valor);
  lcd.print(" ");
break;

case 10:
  lcd.setCursor(0,0);
```

```
  lcd.backlight();
  lcd.print("  VALORES DEFAULT   ");
  lcd.setCursor(14,1);
  lcd.print("      ");
  lcd.setCursor(0,1);
  lcd.print("HORA ON:       ");
  lcd.print(horaDOn_eeprom_valor);
  lcd.print(":");
  lcd.print(minutoDOn_eeprom_valor);
  lcd.setCursor(15,2);
  lcd.print("     ");
  lcd.setCursor(0,2);
  lcd.print("HORA OFF:      ");
  lcd.print(horaDOff_eeprom_valor);
  lcd.print(":");
  lcd.print(minutoDOff_eeprom_valor);
  lcd.setCursor(0,3);
  lcd.print("Ciclo:            ");
  lcd.print(Ciclo);

break;

case 11:
  borrar();
  lcd.noBlink();
  lcd.setCursor(0,1);
  lcd.backlight();
  lcd.print("  ACEPTAR CAMBIOS   ");
  lcd.setCursor(0,3);
  lcd.backlight();
  lcd.print("                    ");

  if (boton(8)==1){
    lcd.setCursor(0,1);
    lcd.print("                      ");
    lcd.noBacklight();
    Posicion=0;
    entrar = 0;


    sendBytes(CantSemaforos);
    sendBytes(RV1);
    sendBytes(RA);
    sendBytes(RV2);
    sendBytes(RV3);
    sendBytes(RV4);

    sendBytes(RV1D);
    sendBytes(RV2D);
    sendBytes(RV3D);
    sendBytes(RV4D);

    sendBytes(RV1F);
    sendBytes(RV2F);
    sendBytes(RV3F);
    sendBytes(RV4F);


    sendBytesGps(Desfasaje);
    sendBytesGps(DesfasajeD);
    sendBytesGps(diaD);
```

```
                    sendBytesGps(horaDOn);
                    sendBytesGps(minutoDOn);
                    sendBytesGps(horaDOff);
                    sendBytesGps(minutoDOff);
                    sendBytesGps(DesfasajeF);
                    sendBytesGps(diaF);
                    sendBytesGps(mesF);

                    configuraInterrupcion1(1,0,0,62499);


                    EEPROM.update(CantSemaforos_eeprom,CantSemaforos);
                    EEPROM.update(RV1_eeprom,RV1);
                    EEPROM.update(RV2_eeprom,RV2);
                    EEPROM.update(RV3_eeprom,RV3);
                    EEPROM.update(RV4_eeprom,RV4);
                    EEPROM.update(RA_eeprom,RA);
                    EEPROM.update(Desfasaje_eeprom,Desfasaje);

                    EEPROM.update(RV1D_eeprom,RV1D);
                    EEPROM.update(RV2D_eeprom,RV2D);
                    EEPROM.update(RV3D_eeprom,RV3D);
                    EEPROM.update(RV4D_eeprom,RV4D);
                    EEPROM.update(DesfasajeD_eeprom,DesfasajeD);
                    EEPROM.update(diaD_eeprom,diaD);
                    EEPROM.update(horaDOn_eeprom,horaDOn);
                    EEPROM.update(minutoDOn_eeprom,minutoDOn);
                    EEPROM.update(horaDOff_eeprom,horaDOff);
                    EEPROM.update(minutoDOff_eeprom,minutoDOff);

                    EEPROM.update(RV1F_eeprom,RV1F);
                    EEPROM.update(RV2F_eeprom,RV2F);
                    EEPROM.update(RV3F_eeprom,RV3F);
                    EEPROM.update(RV4F_eeprom,RV4F);
                    EEPROM.update(DesfasajeF_eeprom,DesfasajeF);
                    EEPROM.update(diaF_eeprom,diaF);
                    EEPROM.update(mesF_eeprom,mesF);

                  if (CantSemaforos==2){
                    Ciclo=((2*RR) + (4*RA) + RV1 + RV2);
                    CicloD=((2*RR) + (4*RA) + RV1D + RV2D);
                    CicloF=((2*RR) + (4*RA) + RV1F + RV2F);
                             }
                  if (CantSemaforos==3){
                    Ciclo=((3*RR) + (6*RA) + RV1 + RV2 + RV3);
                    CicloD=((3*RR) + (6*RA) + RV1D + RV2D + RV3D);
                    CicloF=((3*RR) + (6*RA) + RV1F + RV2F + RV3F);
                             }
                  if (CantSemaforos==4){
                    Ciclo=((4*RR) + (8*RA) + RV1+ RV2 + RV3 + RV4);
                    Ciclo=((4*RR) + (8*RA) + RV1D + RV2D + RV3D + RV4D);
                    Ciclo=((4*RR) + (8*RA) + RV1F + RV2F + RV3F + RV4F);



                     delay(500);
               // Para que no se me junte con el primer if y se vuelva a
prender por mantener pulsado
                     }
                   }
                 break;
```

```
                    case 12:
                      borrar();
                      lcd.noBlink();
                      lcd.setCursor(0,1);
                      lcd.backlight();
                      lcd.print("  CANCELAR CAMBIOS  ");
                      lcd.setCursor(0,3);
                      lcd.backlight();
                      lcd.print("                    ");

                      if (boton(8)==1){
                        lcd.setCursor(0,1);
                        lcd.print("                      ");
                        lcd.noBacklight();
                        Posicion=0;
                        entrar = 0;
                      }

                    break;

                      }//switch
      }//while
}//void


// ENVIAR PARAMETROS A CONTROLADOR POR SERIAL1 -------------------------
void sendBytes(uint8_t value)
{Serial1.write(value);}

// ENVIAR PARAMETROS A ARDUINO TIEMPO POR SERIAL3 ----------------------

void sendBytesGps(uint8_t value)
{Serial3.write(value);}

// RECIBIR PARAMETROS A ARDUINO TIEMPO POR SERIAL3 --------------------
uint8_t byteToInt(byte byte1)
{return (uint8_t)byte1;}

// EFECTO REBOTE DEL PULSADOR -----------------------------------------
int boton(int pin){
  if (digitalRead(pin)==1){
    delay(500);
    if (digitalRead(pin)==1){
      return 1;}}
       else {return 0;}}


//IMPRIMIR DISPLAY ---------------------------------------------------
void borrar(){
  lcd.setCursor(0,0);
   lcd.print("                    ");

   lcd.setCursor(0,2);
   lcd.print("                    ");}
```

```
void imprimir_1(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("     MENU NORMAL    ");
        lcd.setCursor(8,2);
        lcd.noBlink();
        lcd.setCursor(9,2);
        lcd.print("    ");}

void imprimir_CantSemaforos(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print(" CANTIDAD SEMAFOROS ");
        lcd.setCursor(8,2);
        lcd.noBlink();
        lcd.setCursor(9,2);
        lcd.print(CantSemaforos);}

void imprimir_RV1(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("    TIEMPO VERDE 1  ");
        lcd.setCursor(9,2);
        lcd.print(RV1);}

void imprimir_RV2(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("    TIEMPO VERDE 2  ");
        lcd.setCursor(9,2);
        lcd.print(RV2);
  }

void imprimir_RV3(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("    TIEMPO VERDE 3  ");
        lcd.setCursor(9,2);
        lcd.print(RV3);}

void imprimir_RV4(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("    TIEMPO VERDE 4  ");
        lcd.setCursor(9,2);
        lcd.print(RV4);}

void imprimir_RA(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("  TIEMPO AMARILLO   ");
        lcd.setCursor(9,2);
        lcd.print(RA);}
```

```
void  imprimir_Desfasaje(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("     DESFASAJE      ");
        lcd.setCursor(9,2);
        lcd.print(Desfasaje);}


// DIA DISTINTO DE LA SEMANA ---------------------------------------

void imprimir_RV1D(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-  TIEMPO VERDE 1  ");
        lcd.setCursor(9,2);
        lcd.print(RV1D);}

void imprimir_RV2D(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-  TIEMPO VERDE 2  ");
        lcd.setCursor(9,2);
        lcd.print(RV2D);}

void imprimir_RV3D(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-  TIEMPO VERDE 3  ");
        lcd.setCursor(9,2);
        lcd.print(RV3D);}

void imprimir_RV4D(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-  TIEMPO VERDE 4  ");
        lcd.setCursor(9,2);
        lcd.print(RV4D);}

void  imprimir_DesfasajeD(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-   DESFASAJE      ");
        lcd.setCursor(9,2);
        lcd.print(DesfasajeD);}

void  imprimir_diaD(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-      DIA         ");
        lcd.setCursor(6,2);
        lcd.print(diasemana[diaD]);}

void  imprimir_horaDOn(){
        borrar();
```

```
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-    HORA ON       ");
        lcd.setCursor(9,2);
        lcd.print(horaDOn);}

void  imprimir_minutoDOn(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-     MINUTO ON    ");
        lcd.setCursor(9,2);
        lcd.print(minutoDOn);}

void  imprimir_horaDOff(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-    HORA OFF      ");
        lcd.setCursor(9,2);
        lcd.print(horaDOff);}

void  imprimir_minutoDOff(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("D-    MINUTO OFF    ");
        lcd.setCursor(9,2);
        lcd.print(minutoDOff);}

// FERIADO ------------------------------------------------------------

void imprimir_RV1F(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-  TIEMPO VERDE 1  ");
        lcd.setCursor(9,2);
        lcd.print(RV1F);}

void imprimir_RV2F(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-  TIEMPO VERDE 2  ");
        lcd.setCursor(9,2);
        lcd.print(RV2F);}

void imprimir_RV3F(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-  TIEMPO VERDE 3  ");
        lcd.setCursor(9,2);
        lcd.print(RV3F);}

void imprimir_RV4F(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-  TIEMPO VERDE 4  ");
```

```
        lcd.setCursor(9,2);
        lcd.print(RV4F);}

void  imprimir_DesfasajeF(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-   DESFASAJE      ");
        lcd.setCursor(9,2);
        lcd.print(DesfasajeF);}

void  imprimir_diaF(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-      DIA         ");
        lcd.setCursor(9,2);
        lcd.print(diaF);}

void  imprimir_mesF(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("F-      MES         ");
        lcd.setCursor(9,2);
        lcd.print(mesF);}

void imprimir_volver(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("       VOLVER       ");
        lcd.setCursor(0,2);
        lcd.print("       <-----       ");}

// MODO INTERMITENTE-------------------------------------------------

void imprimir_DiaDistinto(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("  MODO DIA DISTINTO ");
        lcd.setCursor(8,2);
        lcd.noBlink();
        lcd.setCursor(9,2);
        lcd.print("    ");}

void imprimir_2(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("    MODO FERIADO    ");
        lcd.setCursor(8,2);
        lcd.noBlink();
        lcd.setCursor(9,2);
        lcd.print("    ");}

void imprimir_5(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
```

```
        lcd.print("   VALORES DEFAULT  ");
        lcd.setCursor(8,2);
        lcd.noBlink();
        lcd.setCursor(9,2);
        lcd.print("    ");}

void imprimir_error(){
        borrar();
        lcd.setCursor(0,1);
        lcd.backlight();
        lcd.print("        ERROR       ");}

void editar(){
  lcd.setCursor(0,3);
  lcd.print("               MODIF");}

void aceptar(){
  lcd.setCursor(0,3);
  lcd.print("                  OK");
  lcd.setCursor(9,2);
  lcd.blink();}
```

ANEXO C: Código de programación Arduino Tiempo

```
                        ////////////////////////////
                        //    BIDEGAIN OCTAVIO    //
                        //    VERSIÓN 21/11/2019  //
                        // CONTROLADOR DE SEMAFOROS //
                        ////////////////////////////


#include <TinyGPS++.h>
#include <EEPROM.h>
TinyGPSPlus gps;

// Parámetros medición tensión
float Ract=0; //
float Rant; //
float Urms=0;    // En valores analógicos
float volt=0;    // Multiplicado por los factores
int n=0;
int N=80;
int hola; // borrar
// Parámetros Calendario ----------------------------------
int si=0;
int sisi=0;
int B;
int bb;
int C;
int D;
int E;
int cuenta;
int meses[12] = {6,2,2,5,0,3,5,1,4,6,2,4}; //agregue el primer 0 para que
el 6 que corresponde a enero quede en el 1
String diasemana[] =
{"Domingo","Lunes","Martes","Miercoles","Jueves","Viernes","Sabado"};
String modos[]= {"Normal", "Intermitente", "Dia Distinto","Feriado"};
int ModoActual;
int cuentanueva;

// Parámetros GPS ----------------------------------------

char caracterEntrada;
boolean GPSapagado;
boolean GPSstandby;
char GPSchar;
String GPSrespuesta;
int hora;
int minuto;
int segundo;
int centisegundo;
float hms;
int dia;
int diaa;
int mes;
int ano;

// PARAMETROS AUXILIARES ---------------------------------
int h;
int inicio;
int tRealSeg;
int tRealSinDesf;
int nciclos;
int sumatiemposD;
int sumatiemposF;
```

```
int diaReal;

int prueba=0;
int otro=0;

// MODOS ---------------------------------------------------

int ModoNormal=31;
int ModoDiaDistinto=33;
int Intermitente=35;
int ModoFeriado=37;


// Parámetros MODOS ----------------------------------------
int Ciclo;
int Desfasaje;

int CicloD;
int DesfasajeD;
int diaD;
int horaDOn;
int minutoDOn;
int horaDOff;
int minutoDOff;

int CicloF;
int DesfasajeF;
int diaF;
int mesF;

// PARAMETROS EEPROM NORMAL---------------------------------
int Ciclo_eeprom=7;
int Desfasaje_eeprom=8;

byte Ciclo_eeprom_valor;
byte Desfasaje_eeprom_valor;

// PARAMETROS EEPROM DIA DISTINTO---------------------------
int CicloD_eeprom=12;
int
DesfasajeD_eeprom=13;
         //POSICION EN LA MEMORIA EEPROM
int
diaD_eeprom=14;
   //POSICION EN LA MEMORIA EEPROM
int horaDOn_eeprom=22;
int minutoDOn_eeprom=23;
int horaDOff_eeprom=24;
int minutoDOff_eeprom=25;

byte CicloD_eeprom_valor;
byte DesfasajeD_eeprom_valor;
byte diaD_eeprom_valor;
byte horaDOn_eeprom_valor;
byte minutoDOn_eeprom_valor;
byte horaDOff_eeprom_valor;
byte minutoDOff_eeprom_valor;

// PARAMETROS EEPROM FERIADO--------------------------------
int CicloF_eeprom=18;
```

```
int
DesfasajeF_eeprom=19;
         //POSICION EN LA MEMORIA EEPROM
int
diaF_eeprom=20;
         //POSICION EN LA MEMORIA EEPROM
int mesF_eeprom=21;

byte CicloF_eeprom_valor;
byte DesfasajeF_eeprom_valor;
byte diaF_eeprom_valor;
byte mesF_eeprom_valor;

// SETUP ----------------------------------------------------
void setup() {
 inicio=0;
 h=0;
//Inicializo comunicación con el Arduino Comunicación
  Serial.begin (115200);     // inicio comunicacion serial
  Serial3.begin(9600);       // Inicializa el puerto serie 3 (comunicación
con A7 GPS+GPRS)
  Serial1.begin(115200);     // Comuniciacón Arduino Comunicación

//Configuro Interrupciones
  configuraInterrupcion1(1,0,0,62499);
  GPSapagado = true;
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(ModoNormal, OUTPUT);
  pinMode(ModoDiaDistinto, OUTPUT);
  pinMode(ModoFeriado, OUTPUT);
  pinMode(Intermitente, OUTPUT);
  pinMode(31, OUTPUT);
  pinMode(33, OUTPUT);
  pinMode(35, OUTPUT);
  pinMode(45, OUTPUT);
  digitalWrite(31,0);      // NORMAL
  digitalWrite(33,0);      // INTERMINTENTE
  digitalWrite(35,0);      // DISTINTO
  digitalWrite(45,0);


   pinMode(53, OUTPUT);

// INICIALIZO CON LAS VARIABLES DE LA MEMORIA EEPROM

    Ciclo_eeprom_valor=EEPROM.read(Ciclo_eeprom);
                  // LEE LA EEPROM Y SIGUE FUNCIONANDO EN EL ULTIMO ESTADO
    Desfasaje_eeprom_valor=EEPROM.read(Desfasaje_eeprom);
                           // LEE LA EEPROM Y SIGUE FUNCIONANDO EN EL
ULTIMO ESTADO

    CicloD_eeprom_valor=EEPROM.read(CicloD_eeprom);
                    // es la variable para almacenarlo
    DesfasajeD_eeprom_valor=EEPROM.read(DesfasajeD_eeprom);
    diaD_eeprom_valor=EEPROM.read(diaD_eeprom);
    horaDOn_eeprom_valor=EEPROM.read(horaDOn_eeprom);
    minutoDOn_eeprom_valor=EEPROM.read(minutoDOn_eeprom);
    horaDOff_eeprom_valor=EEPROM.read(horaDOff_eeprom);
    minutoDOff_eeprom_valor=EEPROM.read(minutoDOff_eeprom);
```

```
    CicloF_eeprom_valor=EEPROM.read(CicloF_eeprom);
                    // es la variable para almacenarlo
    DesfasajeF_eeprom_valor=EEPROM.read(DesfasajeF_eeprom);
    diaF_eeprom_valor=EEPROM.read(diaF_eeprom);
    mesF_eeprom_valor=EEPROM.read(mesF_eeprom);

    if (Ciclo_eeprom_valor==255){Ciclo_eeprom_valor=0;}
    if (Desfasaje_eeprom_valor==255){Desfasaje_eeprom_valor=0;}

    if (CicloD_eeprom_valor==255){CicloD_eeprom_valor=0;}
    if (DesfasajeD_eeprom_valor==255){DesfasajeD_eeprom_valor=0;}
    if
(diaD_eeprom_valor==255){diaD_eeprom_valor=8;}
 // 8 es Ningun día en el vector diasemana[]
    if (horaDOn_eeprom_valor==255){horaDOn_eeprom_valor=0;}
    if (minutoDOn_eeprom_valor==255){minutoDOn_eeprom_valor=0;}
    if (horaDOff_eeprom_valor==255){horaDOff_eeprom_valor=0;}
    if (minutoDOff_eeprom_valor==255){minutoDOff_eeprom_valor=0;}

    if (CicloF_eeprom_valor==255){CicloF_eeprom_valor=0;}
    if (DesfasajeF_eeprom_valor==255){DesfasajeF_eeprom_valor=0;}
    if (diaF_eeprom_valor==255){diaF_eeprom_valor=0;}
    if (mesF_eeprom_valor==255){mesF_eeprom_valor=0;}

    Ciclo=Ciclo_eeprom_valor;
    Desfasaje=Desfasaje_eeprom_valor;

    CicloD=CicloD_eeprom_valor;
    DesfasajeD=DesfasajeD_eeprom_valor;
    diaD=diaD_eeprom_valor;
    horaDOn=horaDOn_eeprom_valor;
    minutoDOn=minutoDOn_eeprom_valor;
    horaDOff=horaDOff_eeprom_valor;
    minutoDOff=minutoDOff_eeprom_valor;

    CicloF=CicloF_eeprom_valor;
    DesfasajeF=DesfasajeF_eeprom_valor;
    diaF=diaF_eeprom_valor;
    mesF=mesF_eeprom_valor;


    hola=0;
}// FIN SETUP


// LOOP ----------------------------------------------------

void loop() {

// GPS -----------------------------------------------------

// AT+GPS=1 ------------------------------------------------
      while (GPSapagado){
        Serial3.println("AT+GPS=1");
        Serial.println("AT+GPS=1");
        delay(1000);
        while (Serial3.available()) {
          GPSchar = (char)Serial3.read();
          GPSrespuesta += GPSchar;
        }
        Serial.print(GPSrespuesta);
```

```
        if (GPSrespuesta.indexOf("OK")!= -1){
          GPSapagado = false;
          GPSrespuesta="";
        }
      }


// AT+GPSRD=N ----------------------------------------------
    if (GPSapagado == false) {
      GPSstandby = true;
      while (GPSstandby){
        Serial3.println("AT+GPSRD=1");
        Serial.println("AT+GPSRD=1");
        delay(1000);
        while (Serial3.available()) {
          GPSchar = (char)Serial3.read();
          GPSrespuesta += GPSchar;
        }
        Serial.print(GPSrespuesta);
        if (GPSrespuesta.indexOf("OK")!= -1){
          GPSstandby = false;
          GPSrespuesta="";
        }
      }
    }

// LECTURA GPS----------------------------------------------
    if (GPSapagado == false && GPSstandby == false) {
      while (caracterEntrada != 'D'){
        if (Serial3.available()) {
          GPSchar = (char)Serial3.read();
          GPSrespuesta += GPSchar;
          gps.encode(GPSchar++);
        }
        if (GPSrespuesta.endsWith("\r\n\r\n")) {
        //Serial.print(GPSrespuesta);
        GPSrespuesta="";
          if (gps.time.isValid()){

            // HORA ------------------------------------------
            if ((gps.time.hour() <= 12) && (gps.time.hour() > 2))
            {Serial.print(F("0"));
            hora=gps.time.hour()-3;}
            if ((gps.time.hour() >= 13))
            {hora=gps.time.hour()-3;}
            if (gps.time.hour() == 0) hora=21;
            if (gps.time.hour() == 1) hora=22;
            if (gps.time.hour() == 2) hora=23;

            Serial.print(hora);

            // MINUTOS ---------------------------------------
            Serial.print(F(":"));
            if (gps.time.minute() < 10) Serial.print(F("0"));
            minuto=gps.time.minute();
            Serial.print(minuto);
            Serial.print(F(":"));

            // SEGUNDOS --------------------------------------
            if (gps.time.second() < 10) Serial.print(F("0"));
```

```
            segundo=gps.time.second();
            Serial.println(segundo);

            tRealSeg=(hora*60*60)+(minuto*60)+segundo;

            // DIA ------------------------------------------------
            if ((gps.time.hour() == 0) || (gps.time.hour() == 1)
||(gps.time.hour() == 2))
            {Serial.print(gps.date.day()-1);
            dia=gps.date.day()-1;} // Dia (1-31) (u8)
            else {Serial.print(gps.date.day());
            dia=gps.date.day();}
            Serial.print(F("/"));


            // MES ------------------------------------------------
            Serial.print(gps.date.month()); // Month (1-12) (u8)
            Serial.print(F("/"));
            mes=gps.date.month();

            // AÑO ------------------------------------------------
            ano=gps.date.year();
            Serial.println(gps.date.year()); // Year (2000+) (u16)
            if (ano==2019){digitalWrite(37,1);}

// CALENDARIO PARA SABER DIA DE LA SEMANA ----------------------------
            if (ano>=2019)
            {Serial.println(diasemana[calendario(dia,mes,ano)]);}
            diaReal=calendario(dia,mes,ano);


             Serial.print("Ciclo: ");
             Serial.println(Ciclo);
             Serial.print("Desfasaje: ");
             Serial.println(Desfasaje);
             Serial.print("CicloD: ");
             Serial.println(CicloD);
             Serial.print("DesfasajeD: ");
             Serial.println(DesfasajeD);
             Serial.print("CicloF: ");
             Serial.println(Ciclo);
             Serial.print("DesfasajeF: ");
             Serial.println(DesfasajeF);
             Serial.print("Feriado: ");
             Serial.print(diaF);
             Serial.print("/");
             Serial.println(mesF);
             Serial.print("Dia distinto: ");
             if (diaD==7){Serial.println("Todos los dias");}
             else if (diaD==8) {Serial.println("Ningun dia");}
             else {Serial.println(diasemana[diaD]);}
             Serial.print("Hora inicio intervalo: ");
             Serial.print(horaDOn);
             Serial.print(":");
             Serial.println(minutoDOn);
             Serial.print("Hora fin intervalo: ");
             Serial.print(horaDOff);
             Serial.print(":");
             Serial.println(minutoDOff);
             Serial.print("MODO: ");
             Serial.println(modos[ModoActual]);
```

```
        if (inicio==0){
          if ((diaD==8) || (diaD!=8 && diaD!=calendario(dia,mes,ano))){
               tRealSinDesf=tRealSeg-Desfasaje;
               nciclos=tRealSinDesf/Ciclo;
               if (tRealSeg = (((nciclos+2)*Ciclo)+Desfasaje)){
                 ModoActual=0;
                 pinMode(ModoNormal,1);
                 pinMode(ModoDiaDistinto,0);
                 pinMode(ModoFeriado,0);
                 pinMode(Intermitente,0);}
               }
               if ((diaD==calendario(dia,mes,ano))){
              tRealSinDesf=tRealSeg-DesfasajeD;
               nciclos=tRealSinDesf/CicloD;
               if (tRealSeg = (((nciclos+2)*CicloD)+DesfasajeD)){
                 ModoActual=2;
                 pinMode(ModoNormal,0);
                 pinMode(ModoDiaDistinto,1);
                 pinMode(ModoFeriado,0);
                 pinMode(Intermitente,0);}
               }
               if (diaF==dia && mesF==mes){
                   tRealSinDesf=tRealSeg-DesfasajeF;
                   nciclos=tRealSinDesf/CicloF;
                   if (tRealSeg = (((nciclos+2)*CicloF)+DesfasajeF)){
                     ModoActual=3;
                   pinMode(ModoNormal,0);
                   pinMode(ModoDiaDistinto,0);
                   pinMode(ModoFeriado,1);
                   pinMode(Intermitente,0);}
             }
               }

// SINCRONIZAR A LAS 00 HS ---------------------------------------

             if (diaF!=dia && mesF!=mes && diaD!=calendario(dia,mes,ano)){
              if (hora==0 && minuto==0 && segundo==Desfasaje){
                ModoActual=0;
                  pinMode(ModoNormal,1);
                  pinMode(ModoDiaDistinto,0);
                  pinMode(ModoFeriado,0);
                  pinMode(Intermitente,0);}
              }
              if (diaF==dia && mesF==mes){
              if (hora==0 && minuto==0 && segundo==DesfasajeF){
                ModoActual=1;
                   pinMode(ModoNormal,0);
                  pinMode(ModoDiaDistinto,0);
                  pinMode(ModoFeriado,1);
                  pinMode(Intermitente,0);}
             }
              if (diaD==calendario(dia,mes,ano)){
              if (hora==0 && minuto==0 && segundo==DesfasajeD){
                ModoActual=2;
                   pinMode(ModoNormal,0);

                  pinMode(ModoDiaDistinto,1);
                  pinMode(ModoFeriado,0);
                  pinMode(Intermitente,0);}
              }
```

```
            gps.encode(32);
          }
          else {Serial.println("Hora no valida");}
        }
        if (Serial.available()) {
          caracterEntrada = (char)Serial.read();}
      }
    }

    if (caracterEntrada == 'D') {
      GPSstandby = false;
      while (!GPSstandby){
        Serial3.println("AT+GPSRD=0");
        Serial.println("AT+GPSRD=0");
        delay(100);
        while (Serial3.available()) {
          GPSchar = (char)Serial3.read();
          GPSrespuesta += GPSchar;
        }
        Serial.print(GPSrespuesta);
        if (GPSrespuesta.indexOf("OK")!= -1){
          GPSstandby = true;
          GPSrespuesta="";
        }
      }
    }

    if (caracterEntrada == 'E') {
      GPSapagado = false;
      while (!GPSapagado){
        Serial3.println("AT+GPS=0");
        Serial.println("AT+GPS=0");
        delay(100);
        while (Serial3.available()) {
          GPSchar = (char)Serial3.read();
          GPSrespuesta += GPSchar;
        }
        Serial.print(GPSrespuesta);
        if (GPSrespuesta.indexOf("OK")!= -1){
          GPSapagado = true;
          GPSrespuesta="";
        }
      }
    }
} // Loop


//FUNCIÓN configuraInterrupcion1() --------------------------------
void configuraInterrupcion1(boolean b0, boolean b1, boolean b2, int dN) {
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 OCR1A = dN;
 TCCR1B |= (1 << WGM12);
 TCCR1B |= (b2 << CS10); // Se pone a b0 el bit CS10 (Clock Select bit 10)
 TCCR1B |= (b1 << CS11); // Se pone a b1 el bit CS11 (Clock Select bit 11)
 TCCR1B |= (b0 << CS12); // Se pone a b2 el bit CS12 (Clock Select bit 12)


 TIMSK1 = (1 << OCIE1A);
```

```
 sei();
}

//FUNCIÓN deshabilitaInterrupcion1() ------------------------------
void deshabilitaInterrupcion1(){
 cli();
 TCCR1A = 0;
 TCCR1B = 0;
 TCCR1B |= (0 << CS10);
 TCCR1B |= (0 << CS11);
 TCCR1B |= (0 << CS12);
 sei();
}

ISR(TIMER1_COMPA_vect) {
           if (diaD==diaReal){
            if ((horaDOn==hora) && (minutoDOn==minuto)){
                  if (ModoActual==2){
                    digitalWrite(35,0);
                   }
                  if (ModoActual==0){
                    ModoActual=2;
                    digitalWrite(35,1);
                    }
                  }
            if ((horaDOff==hora) && (minutoDOff==minuto)){
                if (ModoActual==0){
                    digitalWrite(31,0);
                   }
                  if (ModoActual==2){
                    ModoActual=0;
                    digitalWrite(31,1);
                    }
                        }
           }

           if (diaD==7){
            if ((horaDOn==hora) && (minutoDOn==minuto)){
                  if (ModoActual==2){
                    digitalWrite(33,0);
                   }
                  if (ModoActual==0){
                    ModoActual=2;
                    digitalWrite(33,1);
                    }
                  }
            if ((horaDOff==hora) && (minutoDOff==minuto)){
                if (ModoActual==0){
                    digitalWrite(31,0);
                   }
                  if (ModoActual==2){
                    ModoActual=0;
                    digitalWrite(31,1);
                    }
                        }
           }

          if (diaD==diaReal){
            if ((horaDOn==hora) && (minutoDOn==minuto)){
              ModoActual=2;
                pinMode(ModoNormal,0);
```

```
                pinMode(ModoDiaDistinto,1);
                pinMode(ModoFeriado,0);
                pinMode(Intermitente,0);
              }
            if ((horaDOff==hora) && (minutoDOff==minuto)){
               ModoActual=0;
               pinMode(ModoDiaDistinto,0);
                 pinMode(ModoNormal,1);
                 pinMode(Intermitente,0);
                pinMode(ModoFeriado,0);
                 }
            }

// CONDICIONES
if (Serial1.available())
   {

    h=h+1;
    if (h==1){ Ciclo = byteToInt(Serial1.read());}
    if (h==1){ Ciclo = byteToInt(Serial1.read());//}
    if (h==1){ Desfasaje = byteToInt(Serial1.read());}

    if (h==3){ CicloD = byteToInt(Serial1.read());}
    if (h==2){ DesfasajeD = byteToInt(Serial1.read());}
    if (h==3){ diaD = byteToInt(Serial1.read());}
    if (h==4){ horaDOn = byteToInt(Serial1.read());}
    if (h==5){ minutoDOn = byteToInt(Serial1.read());}
    if (h==6){ horaDOff = byteToInt(Serial1.read());}
    if (h==7){ minutoDOff = byteToInt(Serial1.read());}

    if (h==10){ CicloF = byteToInt(Serial1.read());}
    if (h==8){ DesfasajeF = byteToInt(Serial1.read());}
    if (h==9){ diaF = byteToInt(Serial1.read());}
    if (h==10){ mesF = byteToInt(Serial1.read());


    EEPROM.update(Ciclo_eeprom,Ciclo);
    EEPROM.update(Desfasaje_eeprom,Desfasaje);


    EEPROM.update(CicloD_eeprom,CicloD);
    EEPROM.update(DesfasajeD_eeprom,DesfasajeD);
    EEPROM.update(diaD_eeprom,diaD);
    EEPROM.update(horaDOn_eeprom,horaDOn);
    EEPROM.update(minutoDOn_eeprom,minutoDOn);
    EEPROM.update(horaDOff_eeprom,horaDOff);

    EEPROM.update(minutoDOff_eeprom,minutoDOff);

    EEPROM.update(CicloF_eeprom,CicloF);
    EEPROM.update(DesfasajeF_eeprom,DesfasajeF);
    EEPROM.update(diaF_eeprom,diaF);
    EEPROM.update(mesF_eeprom,mesF);

    h=0;
    }
   }
}


// Calendario (día de la semana) ----------------------------
```

```
int calendario (int dia, int mes, int ano)
{
  //COEFICIENTE B (AÑO) ------------------------------------
  bb = ano-2000;
  B = bb+(bb/4);

  //COEFICIENTE C (AÑO bisiesto) ---------------------------
  if (ano==2020 || ano==2024 || ano==2028 || ano==2032 || ano==2036 ||
ano==2040){
    if (mes==1 || mes==2){
      C = -1;
    }
    else {C=0;}}

  //COEFICIENTE D (mes) ------------------------------------
   D=meses[mes-1];

  //COEFICIENTE E (AÑO bisiesto) ---------------------------
  diaa=dia;
  while(diaa>=7){diaa=diaa-7;}
  E=diaa;

  //CUENTA -------------------------------------------------
   cuenta = B+C+D+E;
  cuentanueva=cuenta;
  while(cuentanueva>=7){cuentanueva=cuentanueva-7;}
return cuentanueva;
}

// recibir PARAMETROS
uint8_t byteToInt(byte byte1)
{return (uint8_t)byte1;}
```