ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

SIMULACIÓN DE UN PARQUEADERO INTELIGENTE PARA LA UNIVERSIDAD DE LAS AMÉRICAS EN LA SEDE QUERI

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

GABRIELA ALEXANDRA CHARRO SANGOLUISA
gabriela.charro@epn.edu.ec

DIRECTOR: ING. MÓNICA DE LOURDES VINUEZA RHOR
monica.vinueza@epn.edu.ec

Quito, Enero 2020

DECLARACIÓN

Yo, Gabriela Alexandra Charro Sangoluisa, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Sin perjuicio de los derechos reconocidos en el primer párrafo del artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación-COESC-, soy titular de la obra en mención y otorgo una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a La Escuela Politécnica Nacional. Entregaré toda la información técnica pertinente. En el caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Gabriela Alexandra Charro Sangoluisa

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Gabriela Alexandra Charro Sangoluisa, bajo mi supervisión.

Ing. Mónica Vinueza Rhor

DIRECTOR DEL PROYECTO

DEDICATORIA

A mi esposo Víctor y a mis hijos Alex Gabriel y Juan David porque han sido mi inspiración para nunca decaer y cada vez ser mejor, con esto quiero demostrarles que nada en esta vida es imposible, que todo se puede conseguir con esfuerzo y dedicación.

Los amo infinitamente.

Gabriela Alexandra Charro Sangoluisa

AGRADECIMIENTO

A Dios por darme la vida y la salud, por permitirme cumplir el objetivo planteado.

A la Escuela Politécnica Nacional y a todos los profesores de la Escuela de Formación de Tecnólogos por sus conocimientos impartidos y en especial a mi directora de tesis Ing. Mónica Vinueza por siempre alentarme a creer en mí misma y a trabajar duro por conseguir mis sueños.

A mis padres Oswaldo Charro y Delia Sangoluisa y a mi hermana Karla Charro, por el apoyo durante toda la etapa de mis estudios, gracias por cuidar de mis hijos, por su motivación y consejos. Gracias por alentarme a continuar a pesar de las adversidades.

A mi esposo Víctor Herrera por apoyarme e impulsarme a dar este paso, cuando lo vi todo perdido tú me demostraste que nunca es tarde para volver a intentarlo, gracias mi amor por apoyarme en todo aspecto a lograr este objetivo.

Gabriela Alexandra Charro Sangoluisa

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto consiste en la simulación de un parqueadero inteligente para la sede Queri de la Universidad de las Américas.

Este proyecto ha sido realizado pensando en mejorar el flujo vehicular en los alrededores de la sede Queri y hacer un uso más efectivo del tiempo de los conductores al brindar información anticipada sobre la disponibilidad de parqueaderos.

El proyecto inicia con la metodología usada para la simulación del parqueadero inteligente, a continuación, se describen los requerimientos necesarios para el desarrollo del proyecto, la simulación en *Proteus* en donde se detalla los elementos electrónicos a ser utilizados y su respectiva configuración.

A continuación, se describe el proceso de implementación del parqueadero, del cableado siguiendo las normas requeridas para el mismo, de los sensores, placas electrónicas y tablero electrónico aplicando manuales y recomendaciones referente a sistemas de parqueos inteligentes.

Finalmente se presenta la simulación del parqueadero inteligente, donde se realizan pruebas para verificar el correcto funcionamiento, así como también un presupuesto referencial para la implementación del mismo.

Se incluyen también las conclusiones y recomendaciones recogidas de la realización de este trabajo.

Palabras clave: Proteus, parqueadero inteligente, sensores, tablero electrónico.

ABSTRACT

This project consists in the simulation of an intelligent parking in the University of the Americas, Queri campus.

This project has been made to improve the vehicular flow around the Queri campus and make more effective use of driver's time because gives advance information on the availability of parking spaces.

The project starts with the methodology used for the simulation of intelligent parking, then describes the requirements necessary for the development of the project, the simulation in Proteus where it details the electronic elements to be used and their respective configuration.

Next, the process of implementation of the parking lot, of the wiring following the norms required for it, of the sensors, electronic boards and electronic board, applying manuals and recommendations referring to intelligent parking systems is described.

Finally, the simulation of the intelligent parking is presented, where tests are carried out to verify the correct functioning, as well as a referential budget for the implementation of the same.

It also includes the conclusions and recommendations collected from the completion of this work.

Keywords: Proteus, intelligent parking, sensors, electronic board.

1 INTRODUCCIÓN

Actualmente en la ciudad de Quito se evidencia un problema de congestionamiento vehicular, el cual se ve reflejado también en los alrededores de la Universidad de las Américas en especial en la sede Queri.

La actual infraestructura cuenta con una sola vía de acceso la cual fue diseñada en un principio para soportar un poco volumen de tránsito, además no cuenta con información anticipada hacia los conductores sobre la disponibilidad de parqueaderos libres, esto genera congestionamiento y pérdida de tiempo en horas pico.

Al no tener un aviso de parqueaderos disponibles los vehículos se ven obligados a ingresar a la sede hasta el redondel que existe para poder salir nuevamente hasta la calle José Queri.

En este trabajo se simula un parqueadero inteligente a través de la instalación de sensores infrarrojos en cada estación que detectan la presencia o ausencia de un vehículo y de esta manera se brinda al usuario información en tiempo real sobre el estado de los parqueaderos para que este pueda tomar una decisión antes de ingresar al lugar, de tal forma que se ayude a mejorar el flujo vehicular en los alrededores de la sede Queri y hacer un uso más efectivo del tiempo de los conductores.

Los sensores infrarrojos que detectan vehículos medianos, envían señales a dos tarjetas ARDUINO UNO receptoras de datos, estas a su vez transmiten los datos de manera digital a un tablero electrónico, que indica la cantidad de espacios libres y ocupados; además la recepción de esta información permite llevar un control de la palanca de acceso al parqueadero la misma que permanece activa mientras existe disponibilidad de parqueos, caso contrario se bloquea impidiendo el paso de más vehículos.

Se realiza la simulación del sistema de control de parqueo utilizando Proteus 8.7 Profesional, se determinan los dispositivos electrónicos más adecuados para el diseño y se presenta un presupuesto referencial de todo el sistema de control.

1.1 Marco Teórico

Módulo Arduino UNO [1]

Arduino es una plataforma electrónica de código abierto basada en hardware y software relativamente de fácil uso.

Estas tarjetas de desarrollo son capaces de leer entradas, sensores, interruptores, etc. Así mismo son capaces de activar un LED, activar motores, etc.

Se puede programar cualquier tipo de tarjeta Arduino enviando instrucciones al microcontrolador embebido por medio del *software* libre "Arduino IDE".

La popularidad de las tarjetas de desarrollo en cuestión está dada debido a su versatilidad, y sobre todo el hecho de ser de código abierto. Es así que, a través de los años se ha utilizado Arduino en infinidad de proyectos, desde objetos cotidianos a complejos proyectos científicos. Existe una gran comunidad que ha desarrollado y contribuido con conocimiento a esta plataforma, lo que es de gran ayuda tanto para novatos como para expertos en el campo.

Con este tipo de tarjeta de desarrollo el conocimiento es libre y de fácil acceso, es fácil de usar para principiantes, sin embargo, es suficientemente flexible para los usuarios avanzados. Su costo es relativamente bajo y es compatible con gran cantidad de hardware disponible en el mercado.

Funcionamiento

En primera instancia, se tiene una interfaz de entrada, que puede estar directamente unida a los periféricos, o conectarse a ellos por puertos.

El objetivo de esa interfaz de entrada es llevar la información al microcontrolador, el cual se encarga de procesar esos datos.

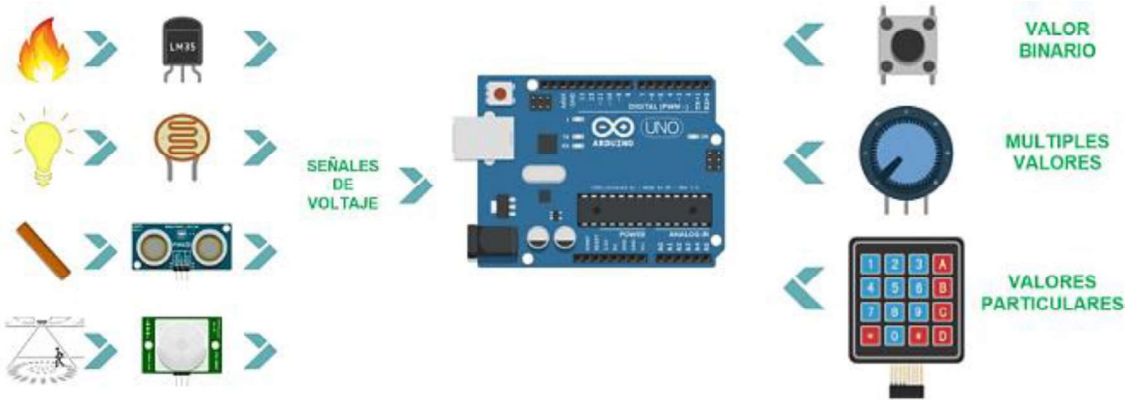

Figura 1.1 Interfaz de entrada de ARDUINO UNO [1]

La segunda fase hace referencia al microcontrolador que en la tarjeta Arduino es el Atmega328p, el mismo varía dependiendo de las necesidades del proyecto en el que se desea usar la placa.

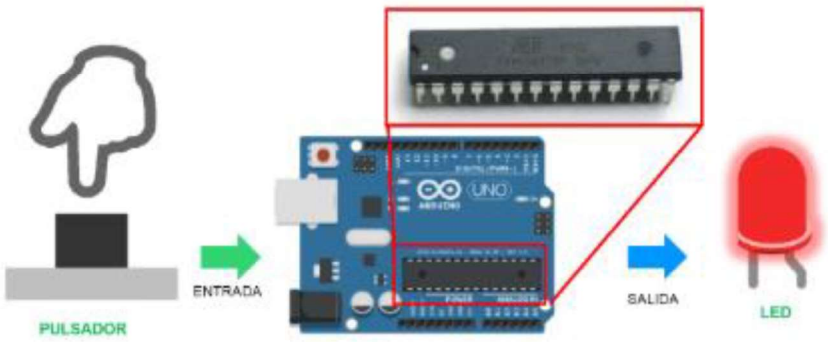

Figura 1.2 Microcontrolador de ARDUINO UNO [1]

Por último, está la interfaz de salida, que lleva la información procesada a los periféricos encargados de hacer el uso final de esos datos, que en algunos casos puede bien tratarse de otra placa en la que se centralizará y procesará nuevamente la información, o sencillamente, una pantalla encargada de mostrar la versión final de los datos.

La Figura 1.3 muestra algunos de los dispositivos que se pueden conectar a la tarjeta:

Figura 1.3 Interfaz de salida de ARDUINO UNO [1]

Otro aspecto importante es la memoria, Arduino tiene tres tipos de memoria; SRAM**,** donde Arduino crea y manipula las variables cuando se ejecuta. Es un recurso limitado y se debe supervisar su uso para evitar agotarlo.

EEPROM es una memoria no volátil para mantener datos después de un reset o apagado, las EEPROMs que tienen un número limitado de lecturas/escrituras por lo que se debe tener en cuenta a la hora de usarla y la memoria de programa o FLASH que usualmente es de 1 (Kb) a 4 (Mb) (controladores de familias grandes). Donde se guarda el sketch.

Estructura

La tarjeta Arduino UNO está estructurada de la siguiente manera:

- El botón de reset, reinicia la tarjeta para empezar a correr el programa nuevamente.
- El puerto USB, comunica la tarjeta con el computador para compilar el programa en el microcontrolador.
- La fuente de alimentación, alimenta con energía a la tarjeta cuando esté desconectada del computador.
- Los pines digitales funcionan tanto como salidas y entradas digitales, con voltajes de 5 o 0 voltios únicamente.
- El controlador, es el núcleo de la tarjeta, aquí se procesa toda la información.
- Las entradas analógicas, sirven únicamente como entradas de una variable analógica.
- Los pines de energía que tienen salidas de 5 (V) y 3.3 (V), así como pines de tierras o GND.

Figura 1.4 Estructura de una placa de ARDUINO UNO [1]

Las características de *hardware* que posee el modelo de Arduino UNO se encuentran detalladas en la siguiente tabla:

Tabla 1.1 Especificaciones técnicas de ARDUINO UNO

Microcontrolador	ATmega328
Voltaje de funcionamiento	5(V)
Alimentación (Recomendada)	7-12(V)
Voltaje máximo de entrada (no recomendado)	20(V)
Pines digitales I/O	14 (de los cuales 6 dan salida PWM)
Pines de entrada analógica	6
Corriente DC por I/O Pin	40(mA)
Corriente DC para el pin 3.3 (V)	50(mA)
Memoria Flash	32 Kb (ATmega328) 0.5Kb usados por *bootloader*
SRAM	2Kb (ATmega328)
EEPROM	1Kb (ATmega328)
Velocidad de reloj	16(MHz)

Software

Para programar cualquier tarjeta Arduino se puede seleccionar entre un editor de código en línea o se puede instalar la aplicación en un ordenador con sistema operativo: *Windows, Mac OS o Linux.*

El editor de código oficial se llama "*Arduino Create*" y se puede acceder desde cualquier navegador o instalarlo en un ordenador, la aplicación es gratuita al igual que las librerías oficiales.

En ciertas ocasiones puede ser necesaria la instalación de un driver adicional, en este caso se debe tener en cuenta el código del circuito utilizado junto a la tarjeta de desarrollo.

Los *drivers* también se pueden descargar de forma gratuita desde la página de los desarrolladores y existen para diversos sistemas operativos.

Librerías

Las librerías son secciones de código, con funciones simples de usar, que facilitan el uso de cualquier componente que se conecte con la tarjeta, tales como Servomotores, *Ethernet*, GPS, etc.

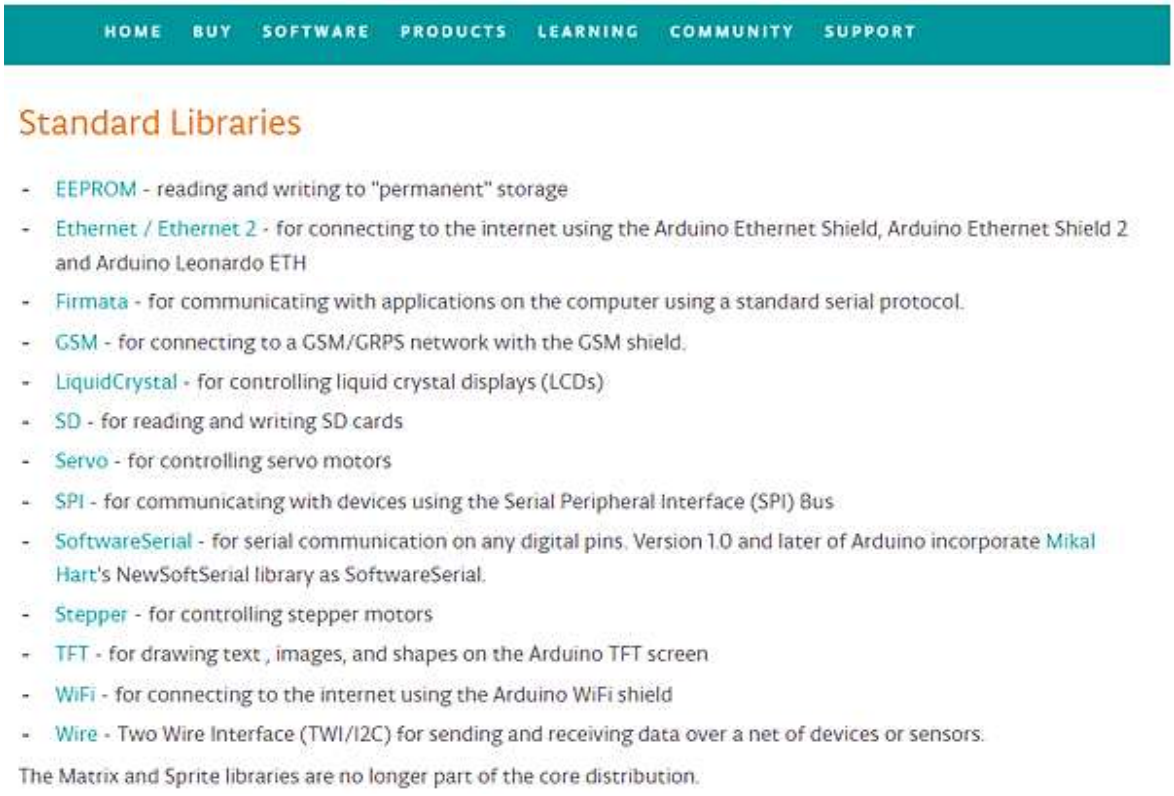

Figura 1.5 Librerías estándar de ARDUINO UNO [1]

Una de las grandes ventajas que presenta el lenguaje C y Arduino es la capacidad de incluir librerías para facilitar la programación, para incluir una librería estándar basta con dar clic en el menú Programa > Incluir Librería y seleccionar la librería requerida.

También se puede incluir librerías obtenidas de otras fuentes o creadas por nosotros mismo, para esto se debe descargar la librería correspondiente e incluirla como un archivo .zip.

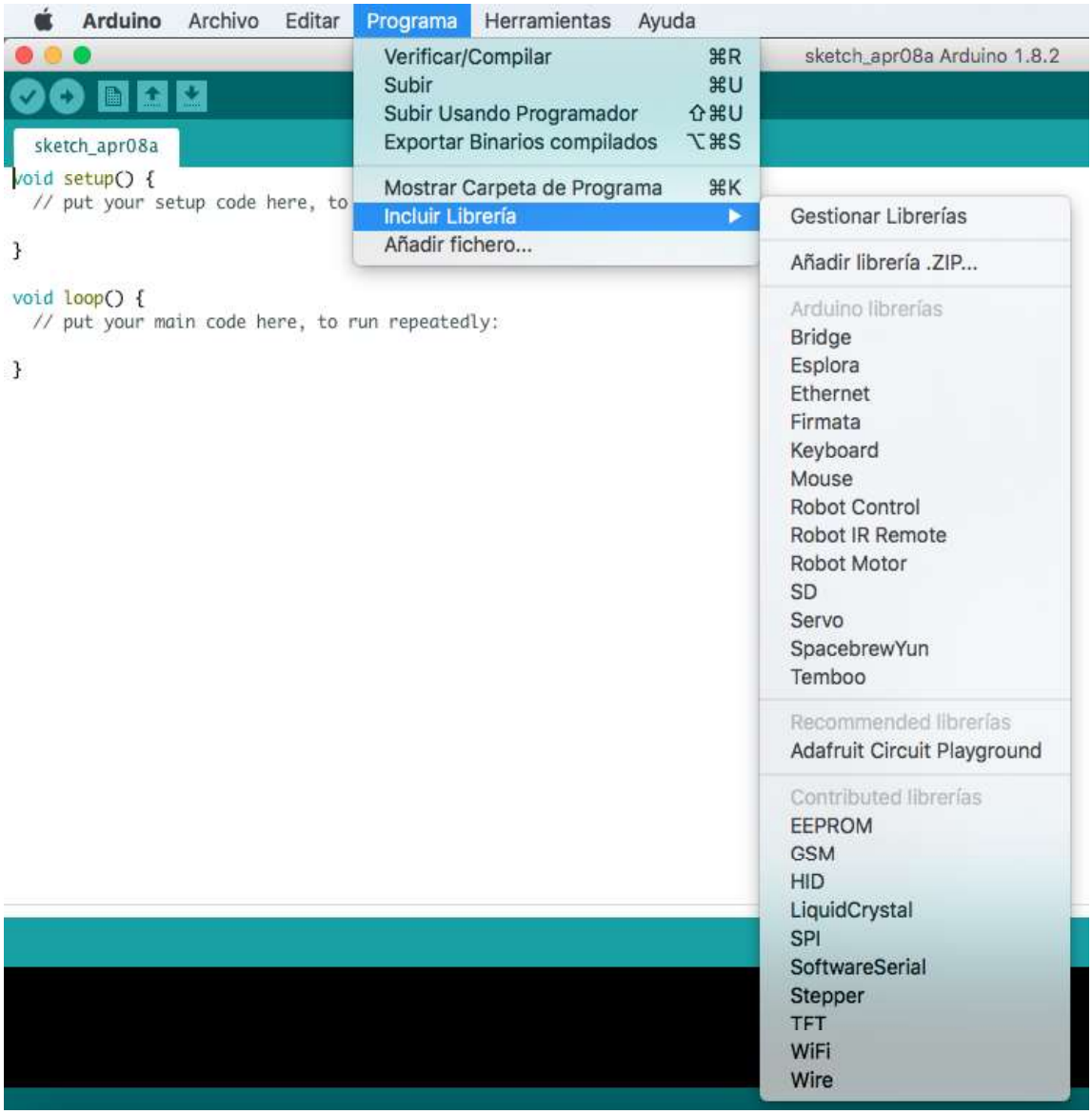

Figura 1.6 Entorno de Arduino [1]

Existen disponibles una gran cantidad de librerías provenientes de Arduino que se pueden obtener desde la opción Gestionar librería.

Sensores Infrarrojos

Características [2]

El sensor infrarrojo es un dispositivo opto-electrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión. Todos los cuerpos emiten una cierta cantidad de radiación, esta resulta invisible para los ojos, pero no para estos aparatos electrónicos, ya que se encuentran en el rango del espectro justo por debajo de la luz visible.

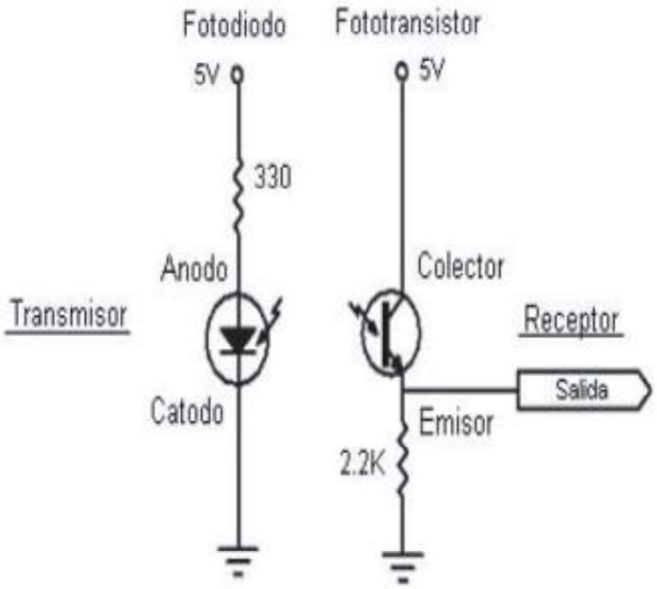

Figura 1.7 Configuración primaria de un sensor [2]

Infrarrojo *SHARP* GP2Y0A02YK [3]

El sensor de distancia *Sharp* GP2Y0A02 permite obtener la distancia entre el sensor y algún objeto dentro del rango de 20 a 150 (cm). Integra tres dispositivos: Un detector sensitivo de posición (PSD), un diodo emisor de infrarrojos (IRED) y un circuito procesador de señales.

La diferencia en la reflectividad de los materiales, así como la temperatura de funcionamiento no afectan en gran medida la operación de este sensor debido al método de detección usado basado en triangulación.

El dispositivo entrega una salida en voltaje correspondiente a la distancia de detección. También se puede utilizar como un sensor de proximidad con la ayuda de un comparador o mediante *software*.

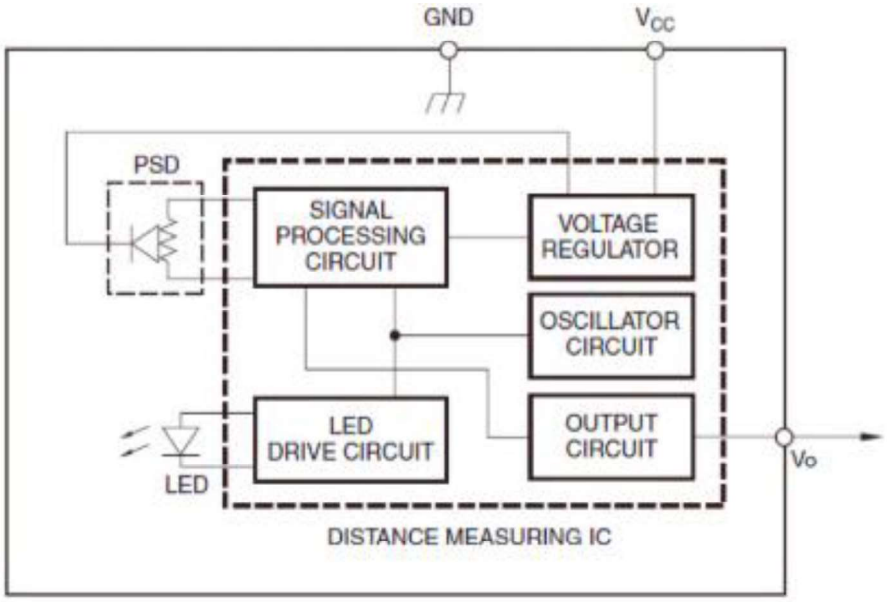

Figura 1.8 Diagrama de bloques del sensor *SHARP* GP2Y0A02YK [3]

Una ventaja adicional es que no son sensibles a la luz ambiental o el sol, enemigo de los sensores infrarrojos, un *SHARP* usa una luz infrarroja intermitente con una frecuencia determinada, que en el receptor es filtrada y elimina cualquier otra fuente de luz diferente a la frecuencia emitida.

Para evitar resultados erróneos, el fabricante recomienda colocarlos en posición perpendicular a la dirección del movimiento.

Por tanto, si el movimiento se va a realizar en un plano horizontal, la colocación vertical del sensor sería la más adecuada.

El GP2Y0A02 utiliza un conector JST de 3 pines que hacen referencia a Vo Voltaje de salida, GND conexión a tierra y Vcc voltaje de operación.

En la siguiente figura se muestra la distribución de los pines de conexión del sensor:

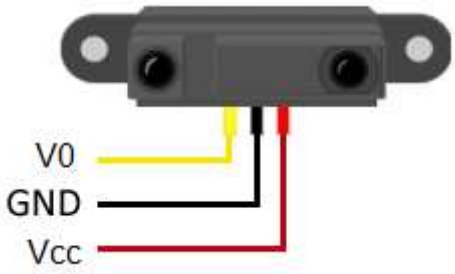

Figura 1.9 Descripción de pines del sensor *Sharp* GP2Y0A02YK [3]

Especificaciones

Las especificaciones eléctricas del sensor *Sharp* GP2Y0A02YK se encuentran en el ANEXO A

Observaciones

Para esta tesis se van a utilizar 186 sensores: *Sharp* GP2Y0A02YK, los cuales van a cada uno de los parqueaderos para la detección de cada automóvil al instante que este ocupe o deje un lugar en el estacionamiento.

Bus I2C

Características [4]

El bus I2C, facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de inteligencia, sólo requiere de dos líneas de señal y un común o tierra.

Fue diseñado por *Philips* y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 (MHz).

La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos.

Descripción de las señales

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar conectando a la alimentación por medio de resistores *pull-up* lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

- SCL (*System Clock)* es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (*System Data*) es la línea por la que se mueven los datos entre los dispositivos.
- GND (Tierra) común de la interconexión entre todos los dispositivos enganchados al bus.

Las dos líneas del bus están en un nivel lógico alto cuando están inactivas. En principio, el número de dispositivos que se puede conectar al bus no tiene límites, aunque hay que observar que la capacidad máxima sumada de todos los dispositivos no supere los 400 picofaradios (pF).

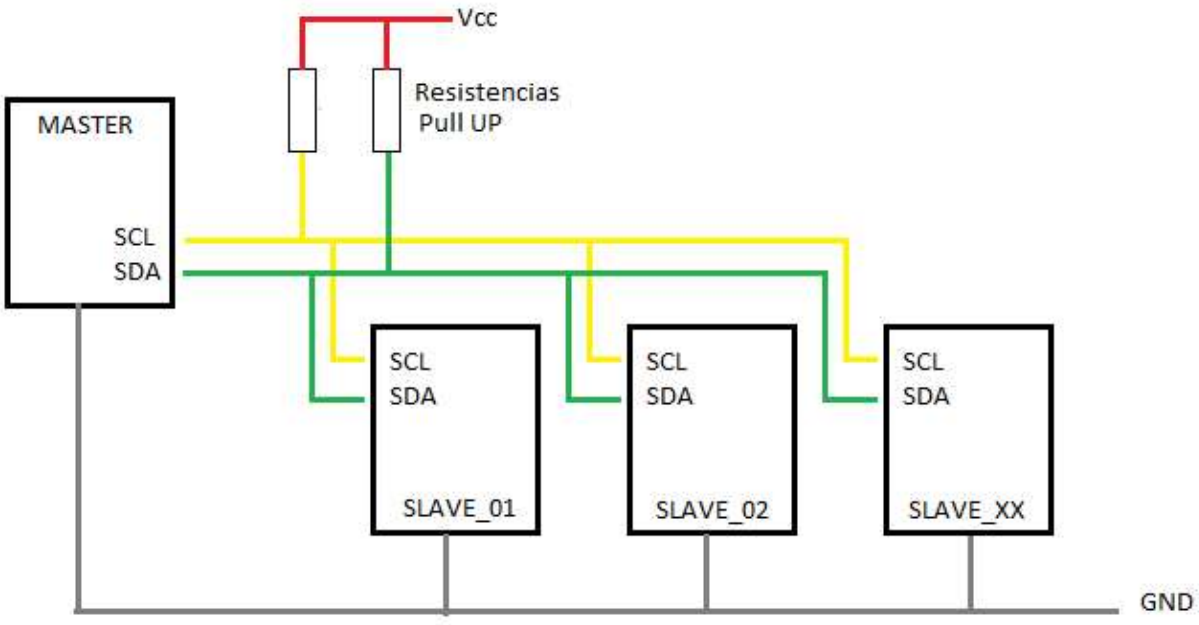

Figura 1.10 Funcionamiento Bus I2C [4]

Expansor PCF8575

Características [5]

El PCF8575 proporciona expansión de E/S (Entrada/Salida) remotas de propósito general para la mayoría de las familias de microcontroladores a través del reloj serial de interfaz I2C (SCL) y datos seriales (SDA).

El dispositivo consta de un puerto casi bidireccional de 16 bits y una interfaz I2C-bus. El PCF8575C tiene un bajo consumo de corriente e incluye salidas enclavadas con una alta capacidad de accionamiento de corriente para impulsar directamente los LED. También posee una línea de interrupción (INT) que se puede conectar a la lógica de interrupción del microcontrolador.

Al enviar una señal de interrupción en esta línea, la E/S remota puede informar al microcontrolador si hay datos entrantes en sus puertos sin tener que comunicarse a través del bus I2C. Esto significa que el dispositivo es un transmisor/receptor esclavo I2C-bus.

Cada transmisión de datos desde el PCF8575C debe constar de un número par de bytes, el primer byte se denominará P07 a P00 y el segundo byte como P17 a P10. El tercero será referido como P07 a P00 y así sucesivamente.

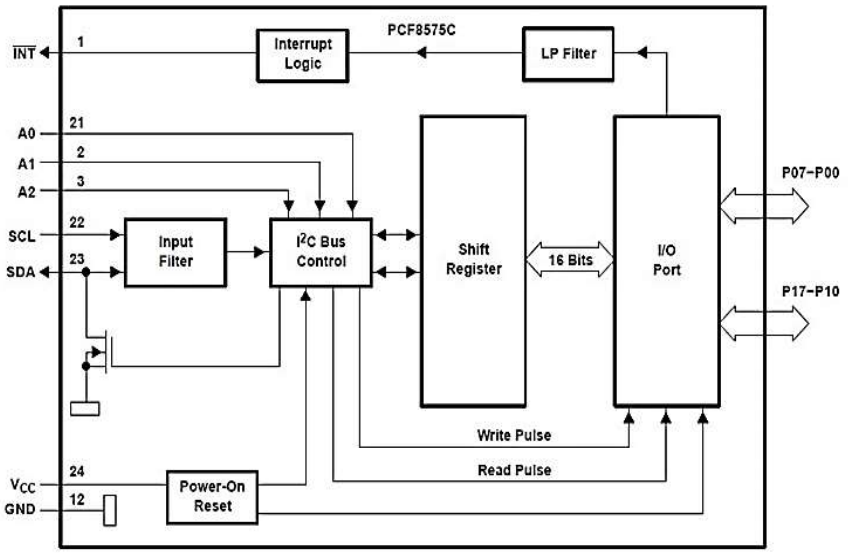

Figura 1.11 Diagrama lógico PCF8575C [5]

Configuración de pines

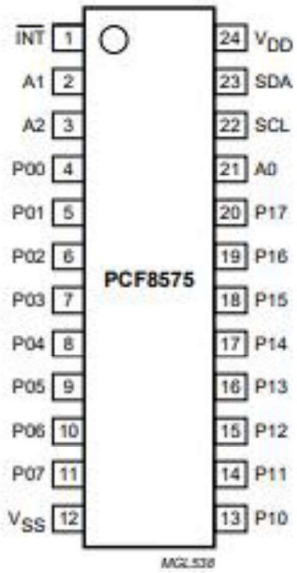

Figura 1.12 Distribución de pines del expansor PCF8575C [5]

Este expansor de E/S de 16 bits para el bus bidireccional de dos líneas (I2C) está diseñado para operaciones de 2.5(V) a 5.5(V_{CC}).

La expansión para conectar otros microcontroladores se realiza a través de la interfaz I2C SCL (reloj serie) y SDA (datos en serie).

El dispositivo cuenta con un puerto de entrada / salida (E/S) cuasi bidireccional de 16 bits (P07 – P00, P17 – P10), incluidas salidas enclavadas con capacidad de accionamiento de alta corriente para controlar directamente los LED.

Cada E/S cuasi bidireccional se puede usar como entrada o salida sin el uso de una señal de control de dirección de datos. En el encendido, las E/S son altas. En este modo, solo una fuente de corriente a V_{CC} está activa.

Características principales del expansor PCF8575C

Las principales características del expansor utilizado en el circuito son:

- Tensión de alimentación de funcionamiento de 4,5 a 5,5 (V).
- Bajo consumo de corriente en espera de 10(μA) máximo

- I2C-bus a puerto paralelo expansor
- 400 (Kbits/s) FAST I2C-bus
- Salida de interrupción de drenaje abierto
- Puerto de E/S remotas de 16 bits para el bus I2C
- Compatible con la mayoría de los microcontroladores
- Salidas bloqueadas con alta capacidad de accionamiento de corriente para controlar directamente los LED
- Dirección por 3 pines de dirección de *hardware* para uso de hasta 8 dispositivos
- Paquete SSOP24.

Observaciones

El expansor utilizado en el proyecto es PCF8575C, debido a la cantidad de sensores que se manejan, por su capacidad de expansión y porque se tiene conocimiento de sus conexiones y uso.

2 METODOLOGÍA

En este proyecto se realizarán las siguientes actividades:

- Se analizó el área del parqueadero de la sede Queri para conocer la disponibilidad ya que se estima tiene una capacidad de 186 parqueaderos aproximadamente y abarca un área aproximada de 3800 metros cuadrados.

- Se diseñó el control del parqueo inteligente a base de sensores conectados en cada estación vehicular, los cuales transmitirán señales hacia una tarjeta electrónica que contiene un microcontrolador que emite la información para ser mostrada en el tablero electrónico mímico.
  Con la información emitida también se llevó el control de la palanca de acceso al parqueadero para que cuando el sistema detecte que el mismo está lleno, la barra no se abra. También se diseñará la conectividad requerida para el número de parqueos especificados y el área señalada.

- Se realizó la simulación del sistema de control para observar el funcionamiento inteligente del parqueo utilizando Proteus 8 Profesional determinando los dispositivos electrónicos más adecuados para el diseño.

- Se presentó un presupuesto referencial de todo el sistema de control.

En el proyecto se utilizó métodos del tipo analítico, exploratorio, experimental y deductivo; empezando por el análisis del área del parqueadero de la sede Queri para conocer la disponibilidad y el área que el mismo abarca.

2.1 Tipo de Investigación

Método Analítico

El método analítico permitió conocer el funcionamiento de todos los elementos utilizados en el proyecto, así como los diferentes programas utilizados tanto para la simulación como

para la programación para lograr que el mismo funcione y permita obtener los resultados propuestos.

Método Exploratorio

Este método se aplicó para aumentar el grado de familiaridad con el problema planteado y así obtener información más completa sobre el mismo; con esto se podrá identificar variables promisorias, establecer prioridades para investigaciones posteriores o sugerir afirmaciones verificables para poder encontrar una solución al problema que no haya sido tomada en cuenta en el pasado.

Los resultados de este tipo de tipo de investigación nos dan un panorama superficial del tema, pero es el primer paso para cualquier tipo de investigación posterior que se quiera llevar a cabo.

Método Experimental

El método experimental se utilizó para realizar la simulación del parqueadero, mediante el desarrollo del programa para la tarjeta Arduino y así validar el correcto funcionamiento de cada elemento utilizado, también se realizaron pruebas y correcciones con el fin de obtener los resultados esperados.

Método Deductivo

Con el uso del método deductivo se pudo obtener conclusiones y recomendaciones a tomar en cuenta, las mismas fueron consecuencia de observaciones realizadas y aportaron para poder simular un prototipo eficiente que cumple con los parámetros establecidos al principio.

El proyecto inició con el análisis del área del parqueadero de la sede Queri para poder conocer la disponibilidad, capacidad y área aproximada del mismo. Con los datos obtenidos en el mismo se diseñó el control de parqueo inteligente, en cual se utilizaron sensores infrarrojos en cada estación vehicular, que transmiten las señales hacia las tarjetas electrónicas Arduino UNO responsables de emitir la información para el tablero electrónico.

Con los datos emitidos por las tarjetas Arduino UNO también se controla la palanca de acceso al parqueadero para que cuando el sistema detecte que el mismo está lleno, la barra no se active.

También se diseñó la conectividad de todos los dispositivos requerida para el número de parqueos especificados y el área señalada.

Se realizó la simulación del sistema de control para observar el funcionamiento inteligente del parqueo utilizando *Proteus 8 Profesional* con los dispositivos electrónicos más adecuados para el diseño.

Con estas premisas se presentó también un presupuesto referencial de todo el sistema de control.

3 RESULTADOS Y DISCUSIÓN

3.1 Análisis del área para el parqueo inteligente

Para realizar el análisis del área para el parqueadero, se observó la disponibilidad del estacionamiento, el mismo se encuentra ubicado en la calle José Queri entre las Avenidas De los Granados y Eloy Alfaro, abarca un área de 4874 (m^2) y tiene 186 estaciones de parqueo distribuidas según la siguiente figura:

Figura 3.1 Distribución de estaciones de parqueo

Según la norma INEN 2248 – Estacionamientos [7], cada estación de parqueo cumple con las siguientes medidas:

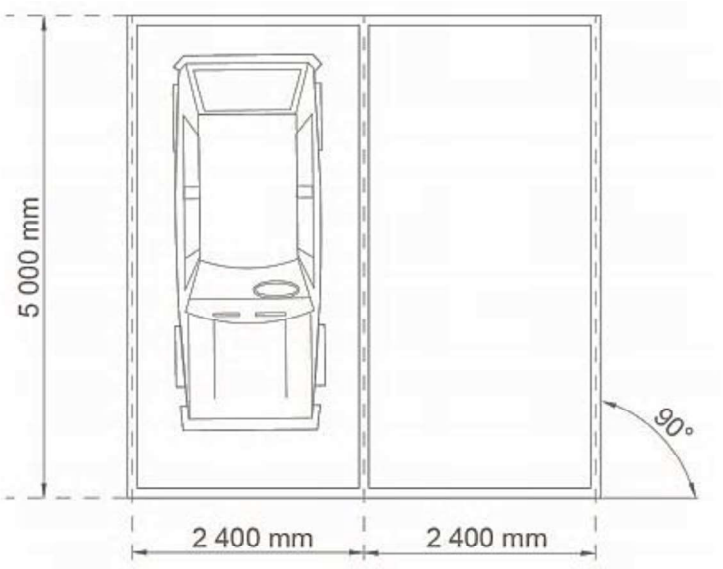

Figura 3.2 Dimensiones de la estación de parqueo [7]

En el ingreso del parqueadero se encuentran las palancas de ingreso y salida, las cuales se activan mediante un equipo dispensador de *tickets* donde el usuario presiona el botón del dispensador y este emite uno que contiene la información del acceso y a la vez activa la palanca para que el vehículo ingrese al parqueadero. De la misma forma al momento de salir del estacionamiento el *ticket* es introducido en el dispensador y este valida la información recibida y activa la palanca.

Figura 3.3 Ingreso al parqueadero de la Sede Queri

Con estos antecedentes y tomando en cuenta las características estructurales del parqueadero para ver si no existirán obstáculos al momento de realizar el cableado y la colocación de los sensores, se estableció los siguientes requerimientos:

- Selección del cable
- Número y tipo de dispositivos
- Fuentes de alimentación
- Recorrido del cableado y lugar dónde se ubicarán los dispositivos
- Distancia del recorrido del cable

En el siguiente esquema del sistema explica el funcionamiento del mismo en forma general:

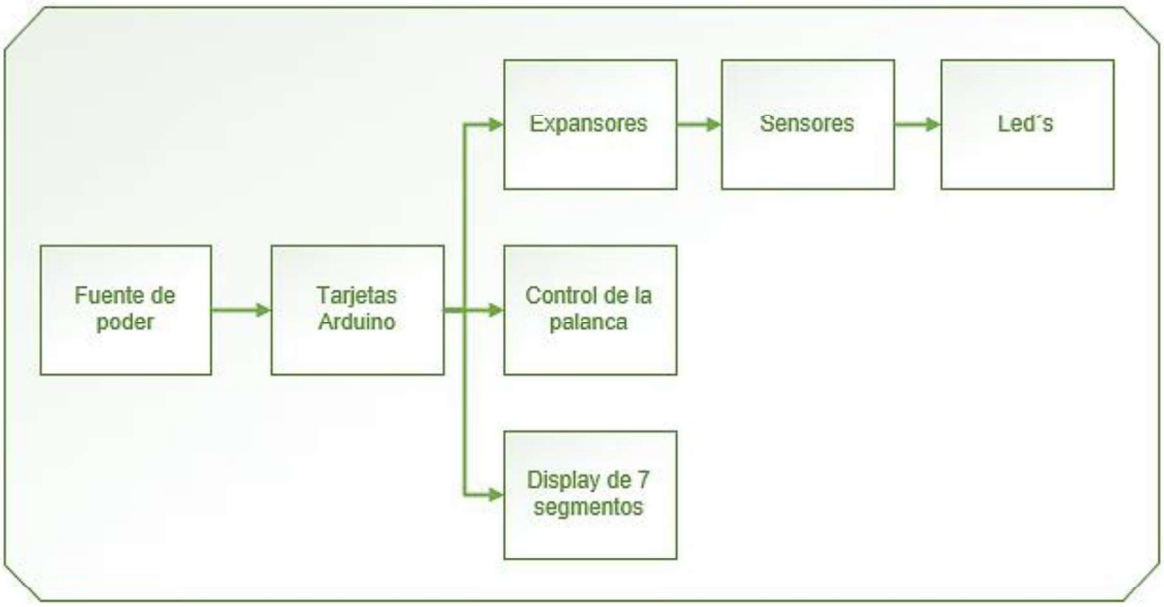

Figura 3.4 Esquema del funcionamiento general del sistema

La alimentación del sistema se realizará a través de una fuente de poder, las tarjetas Arduino UNO son las que llevarán a cabo el control de todo el sistema de la siguiente manera. Los sensores y los diodos led se localizarán en cada estación de parqueo y a la vez estarán conectados a los expansores detectando la ausencia o presencia de un vehículo y enviando una señal a las tarjetas según sea el caso, esta información será receptada por las mismas para controlar la habilitación de la palanca de acceso al parqueadero; la misma que se activa siempre y cuando existan puestos libres, como la visualización de la disponibilidad o no de estaciones de parqueo en el *Display* de 7 segmentos.

Selección del cable [16]

Para el sistema de control de parqueos se utilizará cable UTP por los beneficios que este tiene, como bajo costo, flexibilidad y capacidad de adaptación en cuanto a los estándares de cableado estructurado. Como no se va a manejar distancias superiores a los 110 (m), el cable UTP Cat 6 es una buena opción, para precautelar la transmisión de datos sin pérdida se puede colocar un seguidor de voltaje en el punto más alejado del parqueadero.

Para la implementación se debe realizar un cableado horizontal desde el lugar de ubicación de los sensores hasta el centro de control donde se ubican las tarjetas Arduino UNO que controlan todo el sistema.

El cable seleccionado es el UTP Cat 6, par trenzado de cobre, transmisión a frecuencias de hasta 250 (MHz), 100 (Ohm).

Número de dispositivos y su tipo

De acuerdo al número de estaciones que tiene el parqueadero se han determinado la siguiente cantidad de dispositivos.

Tabla 3.1 Dispositivos para la simulación

Estaciones de parqueo	Sistema de control
186 Sensores tipo *Sharp* GP2Y0A02YK	2 Tarjetas Arduino UNO
186 LED´s rojos de 24 V	12 Expansores PCF8575
186 LED´s verdes de 24 V	6 *Display* De 7 Segmentos de 9x12 Cm color rojo
93 fuentes de poder de 24 V	1 Relé
93 fuentes de poder de 5 V	Resistencias
Circuito LM555	Capacitores
Circuito LM324	Potenciómetros
Resistencias	Optoacoplador
Capacitores	
Potenciómetros	

Fuentes de alimentación

Para poder realizar la alimentación del sistema de control de parqueo se debe utilizar dos tipos de fuentes; de 5 (V_{DC}) para alimentar los sensores y el sistema de control, de 24 (V_{DC}) para alimentar a los leds que mostrarán la disponibilidad de las estaciones, el led verde señala una estación libre y el led rojo indica una estación ocupada. Se ocuparán leds de 9 (Watts) 24 (V) por tratarse de un espacio abierto.

En el parqueadero existe una alimentación de 120 (V_{AC}) por lo que se utilizará reguladores de voltaje a 5 (V_{DC}) y a 24 (V_{DC}).

Recorrido del cableado

Para determinar el recorrido del cableado primero se debe establecer el lugar donde se deben ubicar los sensores, se ha tomado en cuenta las dimensiones de la estación de parqueo que son 2.4 (m) de ancho y 5 (m) de largo, también se tomó en cuenta la distancia promedio de los vehículos que ingresan al parqueadero, esto con el fin de determinar un punto donde el sensor pueda detectar la presencia de un objeto.

La distancia desde el borde del vehículo hasta el punto central de la llanta delantera es en promedio 95 (cm) y la distancia desde el piso hasta el punto medio de la llanta delantera es de 50 (cm).

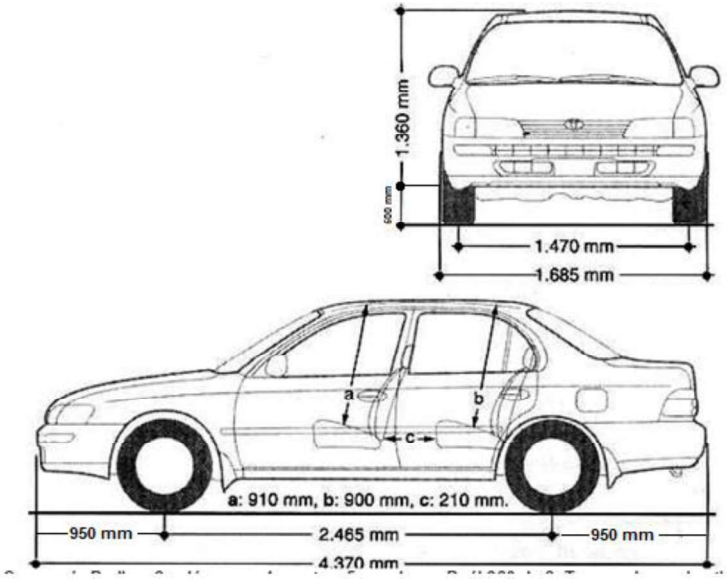

Figura 3.5 Distancia promedio de un vehículo [16]

En cada estación se colocará un parante que contendrá: dos sensores, dos leds rojos, dos leds verdes, un regulador de voltaje a 5 (V_{DC}) y un regulador de voltaje 24 (V_{DC}).

Este parante se ubicará entre dos estaciones de parqueo contiguas con el fin de optimizar recursos.

En la siguiente figura se puede visualizar el esquema de conexión eléctrico.

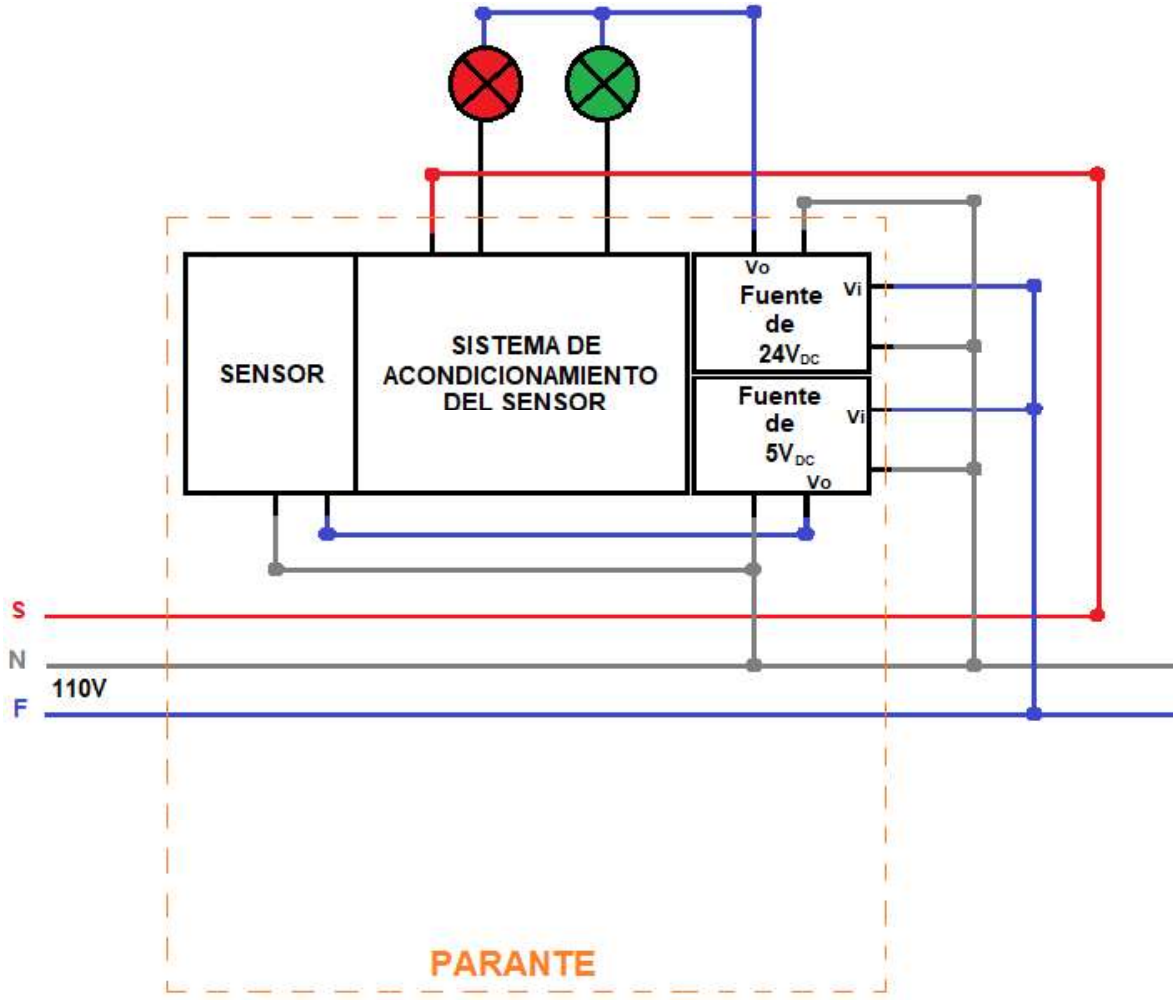

Figura 3.6 Esquema eléctrico del sensor [16]

Debido a que el parqueadero está en un terreno de espacio abierto, se debe realizar obra civil para la interconexión de todos los dispositivos, siguiendo las recomendaciones de la norma ANSI/TIA 569 se debe realizar una canalización subterránea utilizando tubería para ductos de 110 (mm), 4''.

También se debe cavar una zanja de 0,45 (cm) de ancho por 0,60 (cm) de profundidad desde las estaciones de parqueo hasta el centro de control que se ubicará cerca de las palancas de accesos al parqueadero, esto con el fin de llevar el cable dentro de la tubería por todos los lugares de parqueo donde estarán instalados los dispositivos.

Con estos antecedentes y siguiendo el diseño del parqueadero se pudo realizar un esquema del recorrido del cableado y la ubicación de los dispositivos.

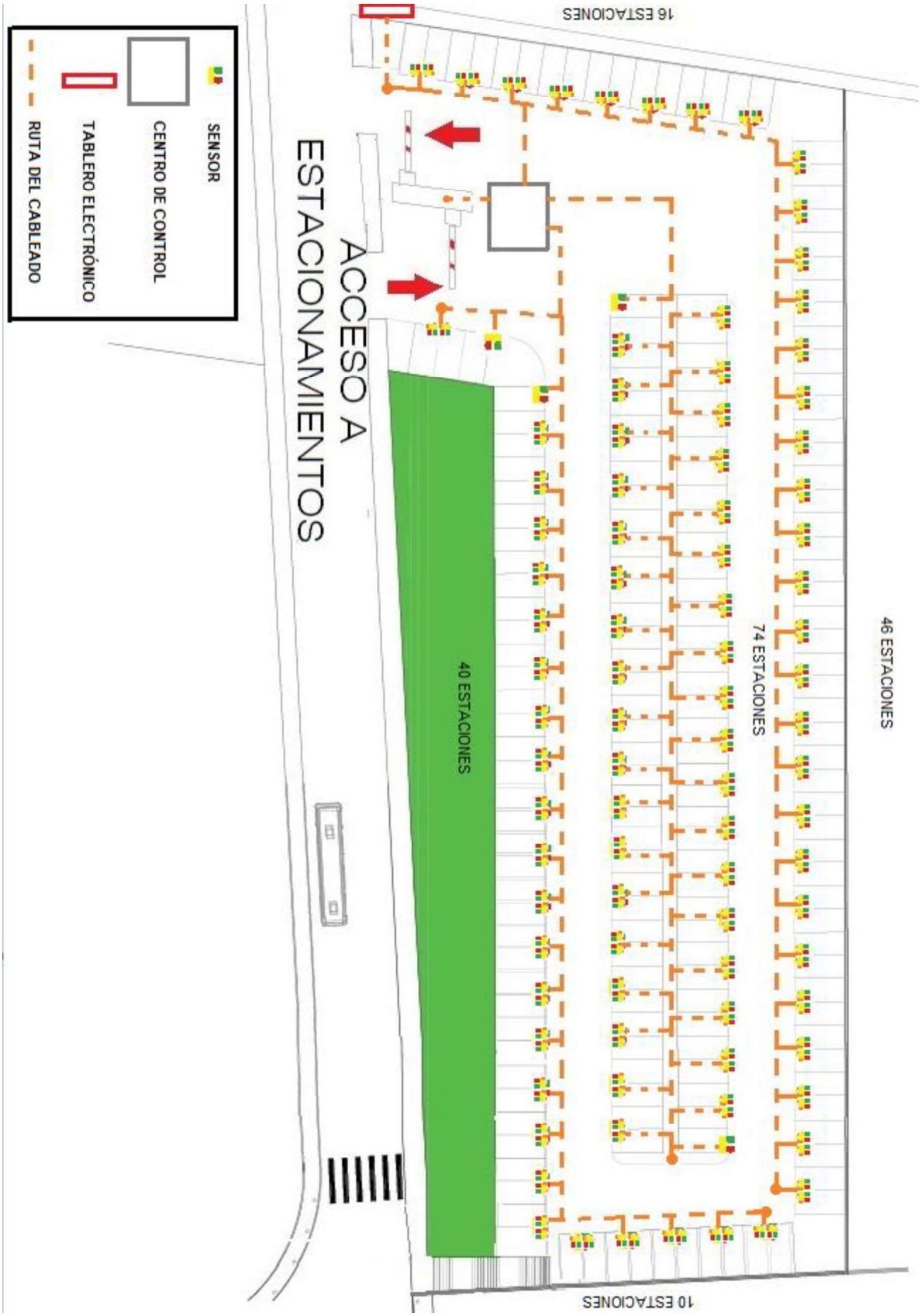

Figura 3.7 Recorrido del cableado en el parqueadero [16]

Distancia del recorrido del cable

En el siguiente esquema se detalla la distancia del recorrido del cable tomando en cuenta las dimensiones obtenidas en la visita realizada al parqueadero.

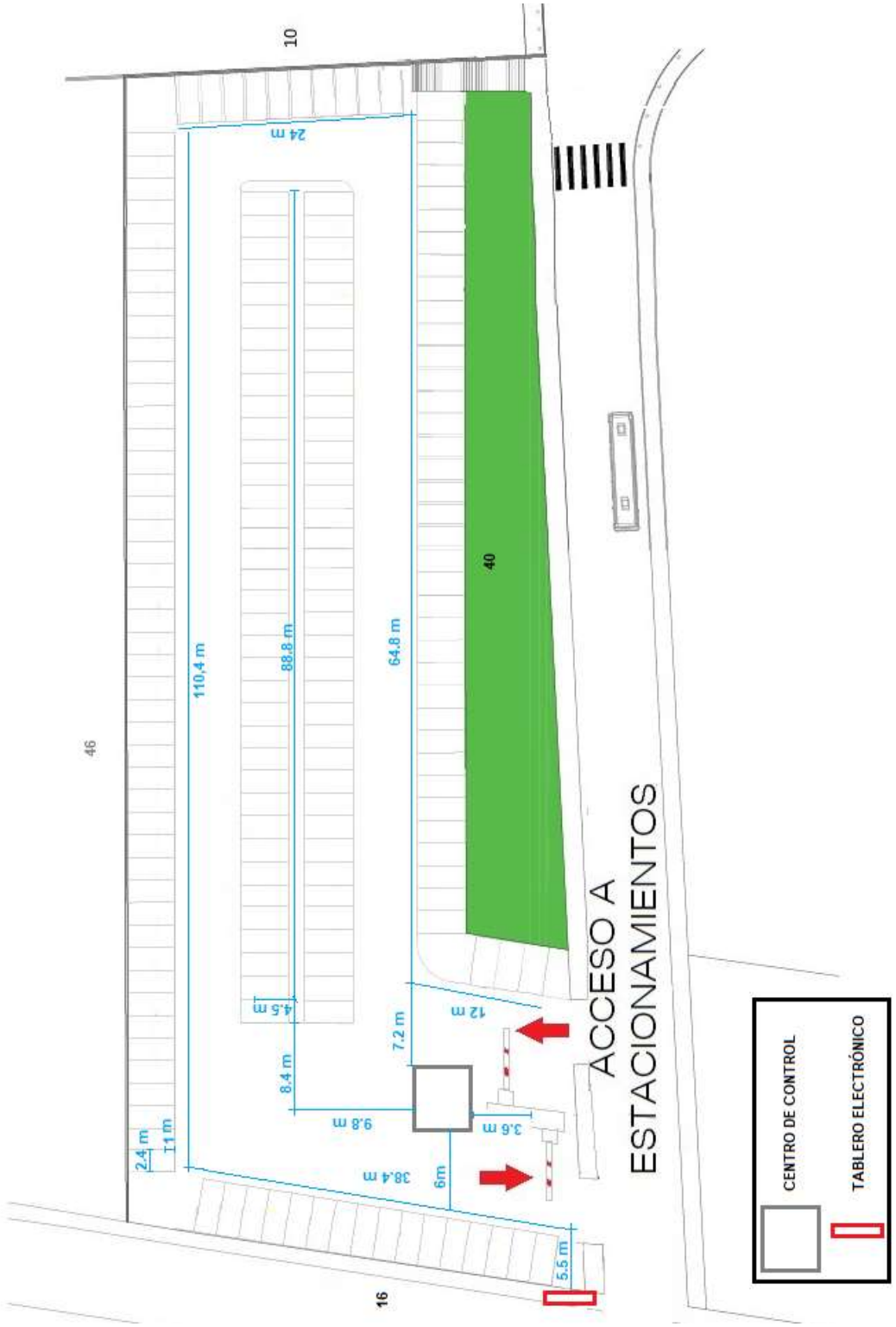

Figura 3.8 Distancia del recorrido del cable [16]

3.2 Diseño del sistema de control del parqueo inteligente

Con el programa AutoCAD se realiza el diseño del sistema de control del parqueo inteligente, en el mismo se pueden observar las medidas del parqueadero, de las estaciones y la ubicación final de todos los dispositivos utilizados.

El sistema de control de parqueo inteligente se encuentra alimentado por una fuente de 120 (V_{AC}); como la distancia promedio es de 200 (m) y la corriente promedio es de poco menos de 1 (A) el grosor del cable a utilizar es el #8 AWG sólido. Para emitir y recibir las señales de los dispositivos conectados se utiliza el Cable UTP Cat 6 multifilar que tiene características que evitan la diafonía y el ruido. Cada estación de parqueo posee un Sensor analógico infrarrojo de distancia y dos focos led de (9(W) de 24V (rojo y verde) los mismos que están alimentados con fuentes de poder de 5 y 24 V_{DC} respectivamente, como la alimentación es de 120 (V_{AC}) se utilizan transformadores de voltaje.

En el cuarto de control se colocan los dispositivos que realizan la operación del sistema, aquí se encuentran las Tarjetas Arduino UNO, los expansores I2C, el sistema de control de la palanca de acceso y el Codificador a 7 segmentos para los displays que forman el Tablero electrónico como todos estos dispositivos trabajan a 5(V) se utiliza de igual manera el respectivo transformador de voltaje.

El tablero electrónico se coloca en el ingreso a la Sede Queri en lugar visible para los conductores, este está formado por 6 displays de 7 segmentos divididos de la siguiente manera: 3 displays se utilizan para indicar los espacios disponibles y los otros 3 para indicar los espacios ocupados. El tablero cuenta con adecuada rotulación para que el usuario pueda distinguir a simple vista lo antes indicado.

La palanca de acceso es controlada por un relé que se activa cuando no existen espacios disponibles para parquear, el relé tiene su propio sistema de aislamiento conformado por un Diodo rectificador 1N4001, 1 Transistor 2N3055, 1 Optoacoplador 4N25 y resistencias de 220 (Ω), 27 (KΩ) y 100 (Ω), este sistema es utilizado como estabilizador de voltaje y sus elementos permiten una conexión eléctricamente aislada entre dos circuitos que operan a distintos voltajes, pues al tratarse de una bobina puede introducir interferencia y ocasionar una lectura errónea de datos. En la siguiente figura se muestra el plano arquitectónico.

Figura 3.9 Diagrama de conectividad

3.3 Simulación del sistema de control

Para realizar la simulación del sistema de control del parqueadero se utiliza *Proteus 8.7 Profesional*, la misma se ha dividido en 4 partes, las cuales se acoplan en la simulación final para presentar el funcionamiento del parqueadero.

1.- Fuente
2.- Sensor
3.- Control del Motor de la palanca de acceso
4.- Simulación final

Fuente

Se utiliza varias fuentes para alimentar las diferentes etapas de la simulación.

Para los sensores se utiliza un transformador a 5(V_{DC}), que va a alimentar a dos sensores; de la misma forma también hago uso de un transformador a 24(V_{DC}) el cual va a alimentar a los focos led rojo y verde que irán en cada estación.
La simulación de las fuentes se muestra en la siguiente figura:

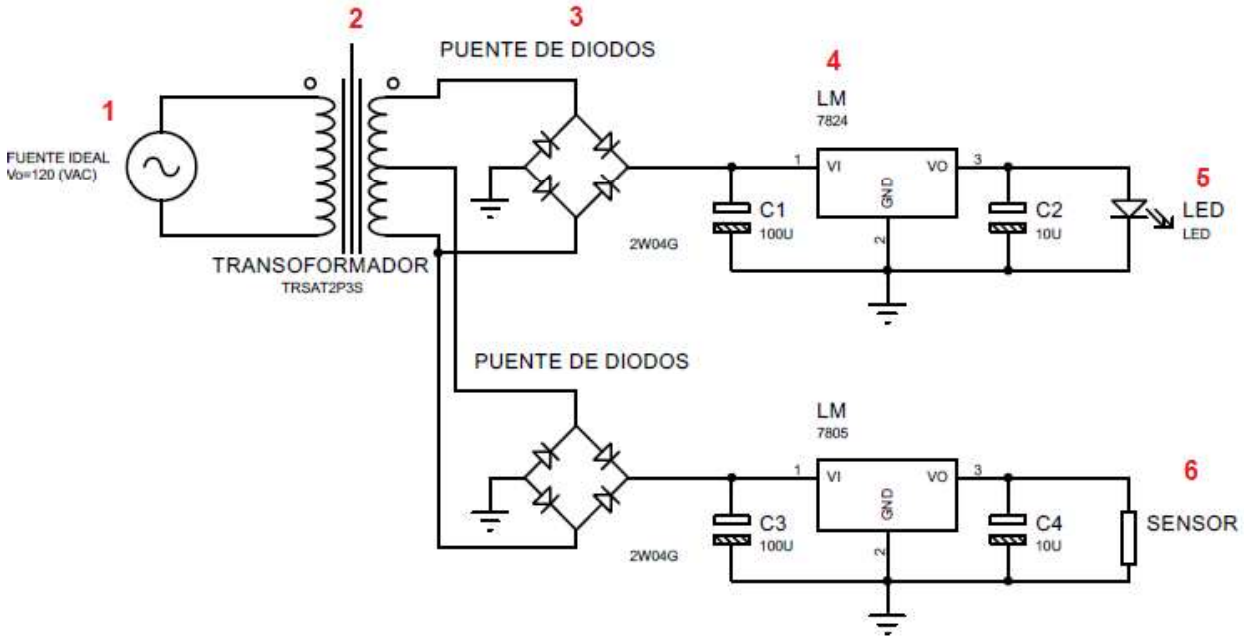

Figura 3.10 Simulación de las fuentes de alimentación

La fuente consta de los siguientes elementos:

1. Fuente de 120 V_{AC}
2. Transformador con tap central que pasa de 24 a 12 (V)
3. Puente de diodos
4. Regulador de voltaje para obtener 24 y 5 (V) respectivamente
5. El diodo led representa a los focos rojo y verde que se colocan en cada estación de parqueo los mismos que trabajan a 24 Voltios, no requiere dimensionar ninguna resistencia ya que se utilizarán focos de 9(W) con V_{DC} de 24 (V)
6. La resistencia R1 representa al sensor que está ubicado en cada estación, los mismos que trabajan a 5 (V).

Para determinar los elementos que se deben colocar en la simulación y que se usarán en el diseño del parqueo se realizaron los siguientes cálculos:

- Relación de transformación

  Bobinado { n1=5 (24 V_{RMS})
  n2=10 (12 V_{RMS})

  V_{pico}=24 x √2=33.84 (V)
  V_{pico}=12 x √2=16.92 (V)

- Cálculo de la corriente para los focos

  Se escoge un foco de 9(W) con V_{DC} de 24 (V)
  $I_{foco\ verde}$=375 (mA)

  $I_{foco\ rojo}$=375 (mA)

  $I_{total\ focos}$ = 750 (mA)

  $V_{nominal\ focos}$ = 24 (V_{DC})

P_{Tcarga} = 24(V) x 750 (mA) = 18(W) ≫ Disipa el regulador

$P_{Regulador} = (V_{in} – V_{out})(I_{TOTAL}) = (33.84 - 24)(0.75) = 7.38$ (W)

≫ Se requiere un disipador de 10(W) ≈

- Cálculo de los capacitores y potencia del puente de diodos

$P_{TOTAL} = P_{Tcarga} + P_{Regulador} = 18 + 7.38 = 25.38$ (W)

$V_{in\ máx} = V_{pico}$ x 10% holgura

$V_{in\ máx}$ = 33.84 x 1.1 = 37.22 (V)

$V_{in\ mín}$ = Vpico x 90% holgura

$V_{in\ mín}$ = 33.84 x 0.9 = 30.45 (V)

$$I_{min} = \frac{P_{TOTAL}}{V_{in\ máx}} = \frac{25.38}{37.22} = 0.68\ (A)$$

$$I_{máx} = \frac{P_{TOTAL}}{V_{in\ min}} = \frac{25.38}{30.45} = 0.85\ (A)$$

$$\gg \frac{I_{máx} \text{ x Factor de proporcionalidad}}{V_{máx} \text{ x Frecuencia del puente}} = \frac{0.85 \text{ x } 5}{37.22 \text{ x } 120} \approx 951\ (\mu F) \approx 1000\ \mu F$$

- Diodos

$V_{máx}$ = 37.22 (V)

$I_{máx}$ = 0.85 (A)

- Para calcular la caída de tensión

$I_{in} = I_{out} / N = 0.85 / 5 = 0.17$ (A)

Máx 5% -- 6(V)

$$R = \rho \frac{L}{A}$$

Donde:

$$R = \frac{V}{I}$$

L ≈ 200 (m) (La distancia máxima al punto mas lejano)

V = 6 (V)

I = 0.17

Reemplazando en la fórmula inicial:

$$\frac{6}{0.17 \times 74} = 0.0171 \times \frac{200}{A}$$

A = 7.086 mm^2, validando en la tabla de cables eléctricos podemos utilizar un cable # 8 = 8.366 mm^2

$$R = \rho \frac{L}{A}$$

$$R = 0.0171 \times \frac{200}{8.366} = 0.41\ (\Omega)$$

V = I x R = (0.17*74)(0.41) = 5.08 (V) caída de voltaje máximo

$I_{TOTAL} = 0.17 \times 186 + I_{Brazo}^{1} = 32.25$ (A) ; se puede utilizar un breaker ≈ 40 (A)

Sensor

El sensor utilizado para la simulación es el *Sharp* GP2Y0A02YK, el cual es un sensor medidor de distancia, pero también se lo puede ocupar como detector de objetos, funciona perfectamente en la intemperie ya que la luz del sol tiene poca influencia sobre él y no interfiere en las detecciones que realiza.

El sensor emite un rayo infrarrojo de 1,50 (m) en sentido horizontal, la distancia a la que se puede detectar un objeto se debe determinar en la simulación utilizando el potenciómetro que está conectado al comparador LM324, en este caso se ha declarado una distancia entre 30 y 50 (cm) para que el sensor indique la presencia de un objeto con un 1 lógico.

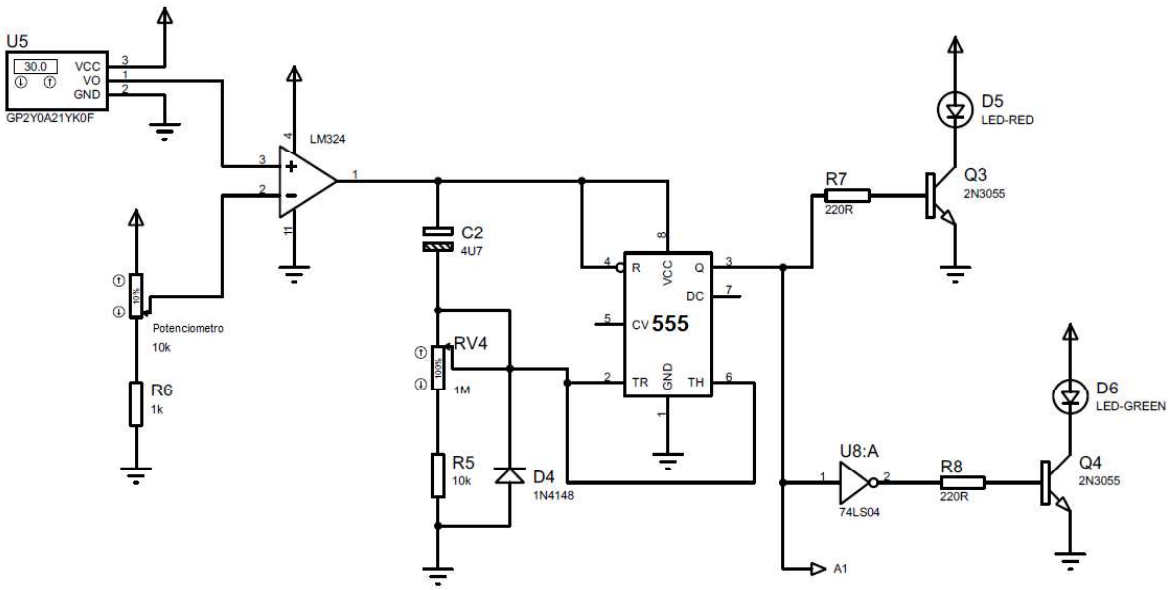

Figura 3.11 Simulación del sensor detector de objetos

En la simulación se puede observar los siguientes dispositivos:

1. Sensor Sharp GP2Y0A02YK que emite una luz infrarroja de hasta 1,50 (m).

2. Comparador LM324 que recibe la señal tanto del sensor como del potenciómetro para emitir la señal deseada.

Si el voltaje del sensor es mayor, el comparador entrega un 1 lógico, en cambio, si el voltaje del potenciómetro es mayor el comparador entrega un 0 lógico.

3. Potenciómetro mediante el cual se regula la distancia a la que se va definir como “Ocupado”, es decir un 1 lógico.

4. Capacitor el mismo que se carga de acuerdo al tiempo de estancia del objeto y da un 1 lógico si se trata de un vehículo estacionado, porque puede darse el caso que no se trate de un auto sino de una persona o cualquier otro objeto que haya cortado la señal infrarroja.

5. Temporizador LM555 determina el tiempo de espera antes de la activación de los led rojo y verde, si el tiempo de estadía del objeto supera los 10 a 15 segundos, el temporizador entrega un 1 lógico que hará encender el LED rojo lo que significa que la estación de parqueo se encuentra ocupada.

   Caso contrario, si se obtiene un 0 lógico este pasa a un inversor lógico para activar el led verde que significa que la estación de parqueo se encuentra libre.

6. Inversor lógico para poder colocar en 1 el 0 lógico que se obtiene del temporizador para poder activar el led rojo de disponible.

7. Led rojo que indica que la estación se encuentra ocupada.

8. Led verde que indica que la estación se encuentra disponible.

El funcionamiento del Sensor se lo muestra en las siguientes figuras:

1. En la figura 3.12 se encuentra encendido el led verde que indica disponibilidad ya que la luz infrarroja no ha sido cortada por ningún objeto.

2. En la figura 3.13 se simula que existe la presencia de un objeto al disminuir la distancia en el sensor, en este caso se observa que se activa el led rojo que indica que la estación de parqueo se encuentra ocupada.

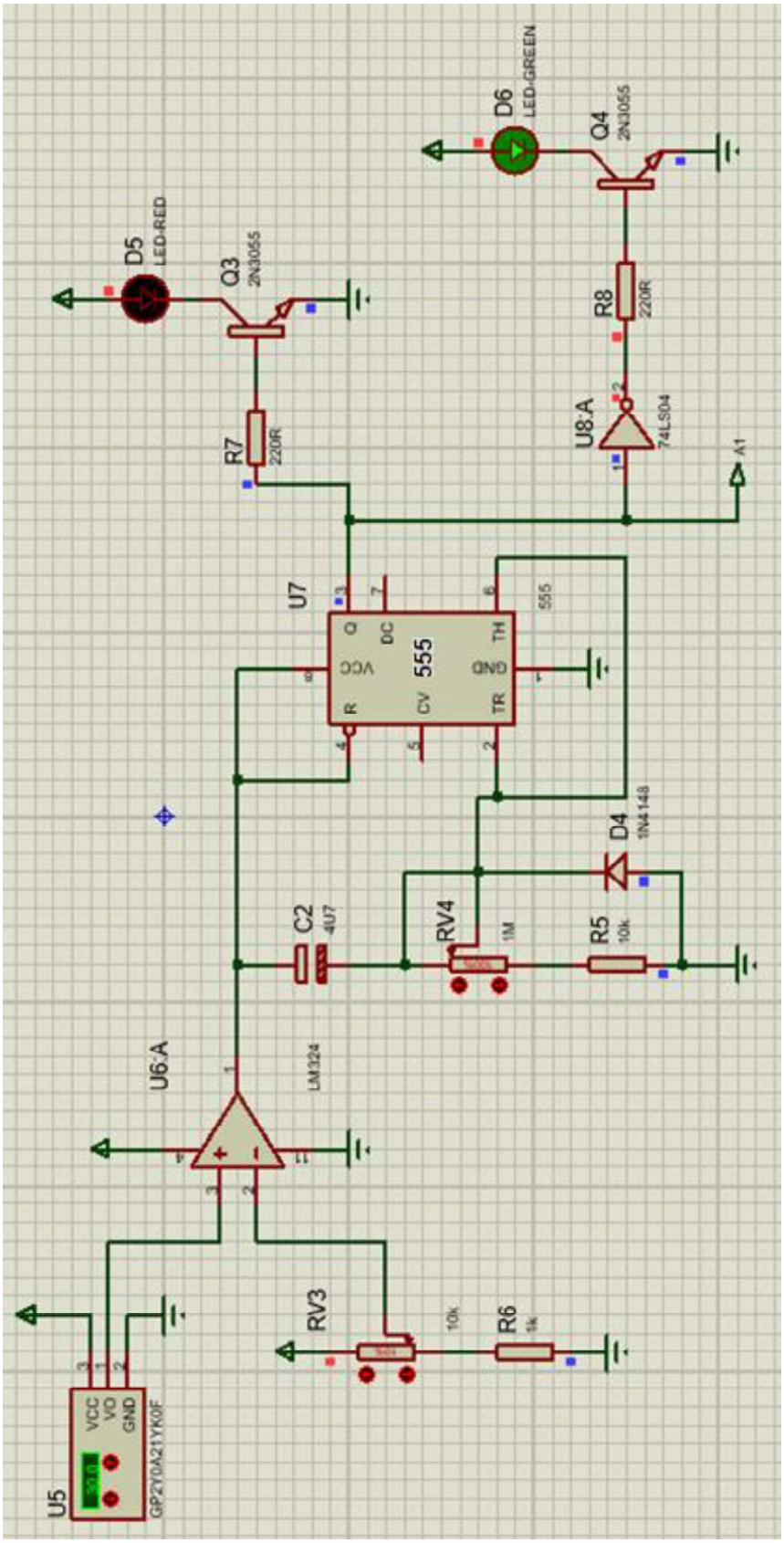

Figura 3.12 Simulación del sensor en estado disponible

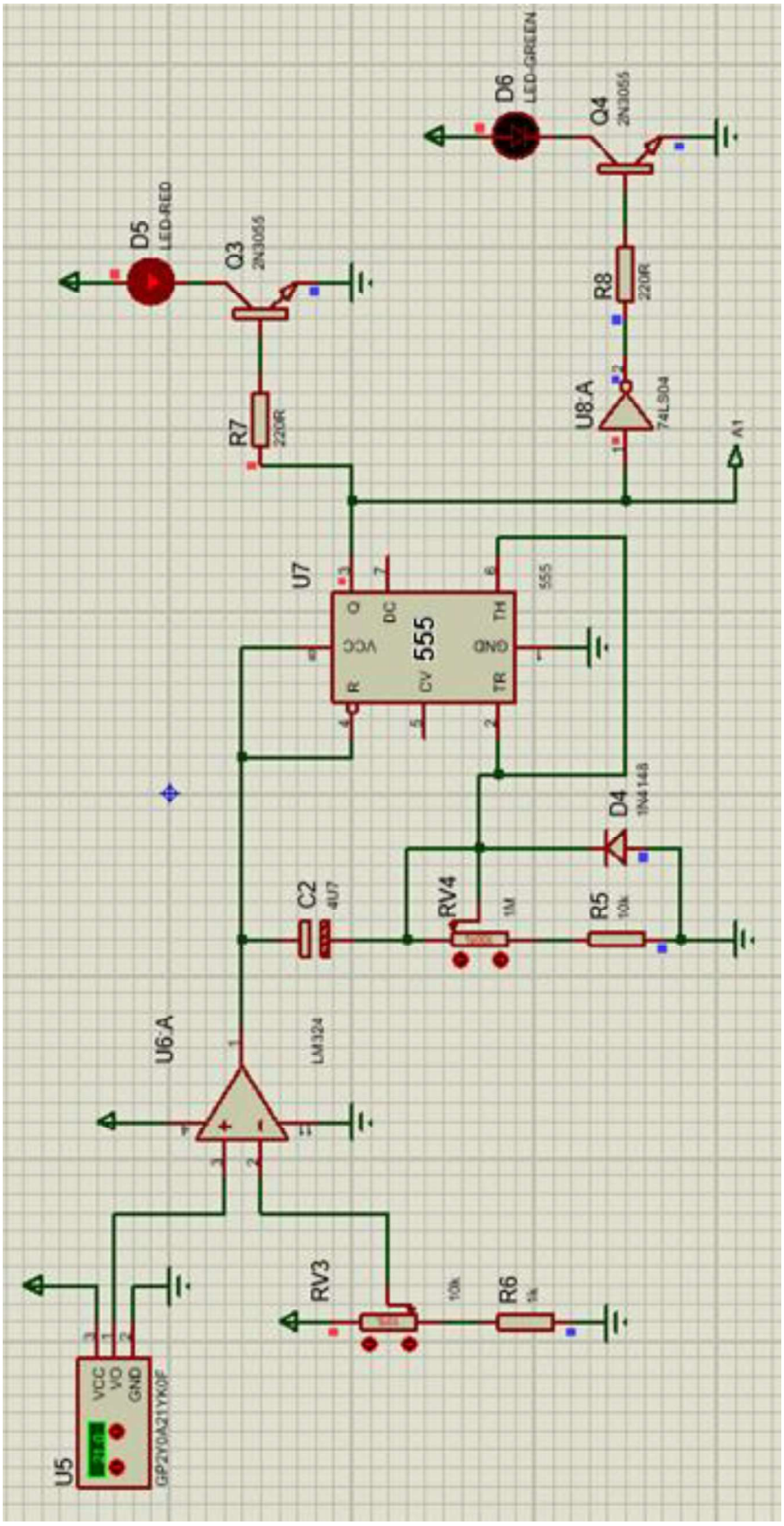

Figura 3.13 Simulación del sensor en estado ocupado

Para poder realizar la simulación en *Proteus 8.7* se establece que el tiempo de estadía del objeto sea de 10 a 15 segundos para considerar que no se trata de un auto parqueado en la estación, en el caso práctico se podría estipular un tiempo máximo de 2 minutos.

Control de la palanca de acceso

Para poder realizar el control de la palanca de acceso al parqueadero se colocó un Relé, el mismo que va conectado en serie con la alimentación del sistema de la palanca. El relé se activa en el caso que todas las estaciones de parqueo se encuentren ocupadas haciendo que el resto del sistema se desactive.

En la Figura 3.14 y Figura 3.15 se simula el funcionamiento del sistema de la palanca y del relé, cuando llega un 1 lógico quiere decir que todas las estaciones se encuentran ocupadas y que no se permitirá el paso a más vehículos al parqueadero y la palanca de acceso se desactiva, mientras lo anterior no ocurra lo autos pueden ingresar a parquear en las estaciones.

El sistema que se ha diseñado antecesor al relé se coloca ya que se debe aislar por la corriente que consume el este dispositivo y como está formado por una bobina puede introducir interferencia produciendo errores, en este caso podría no desactivar la palanca cuando todas las estaciones de parqueo se encuentren ocupadas.

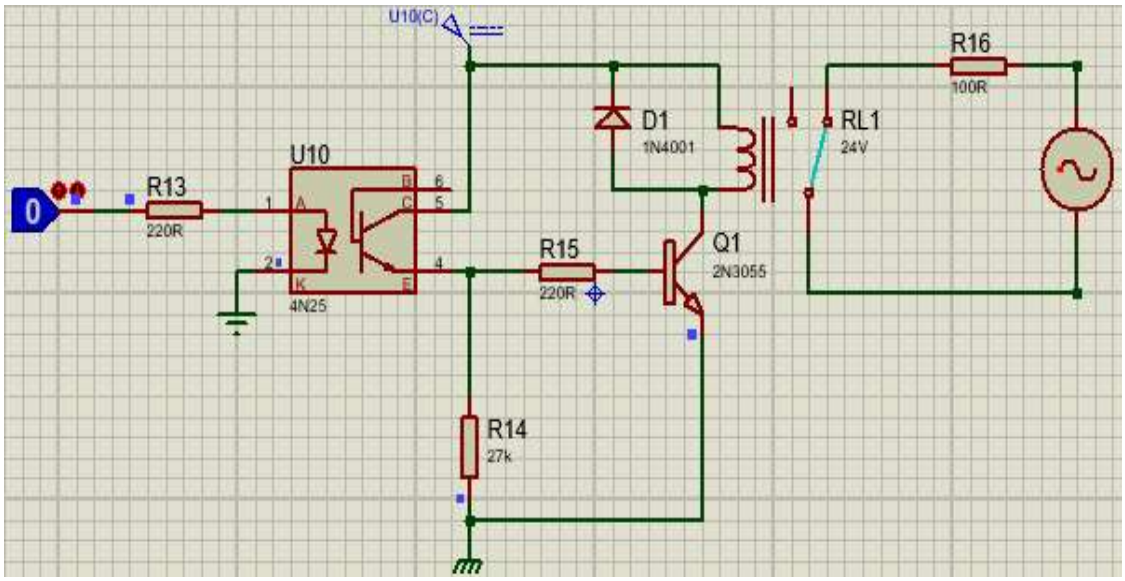

Figura 3.14 Simulación cuando hay estaciones de parqueo disponibles

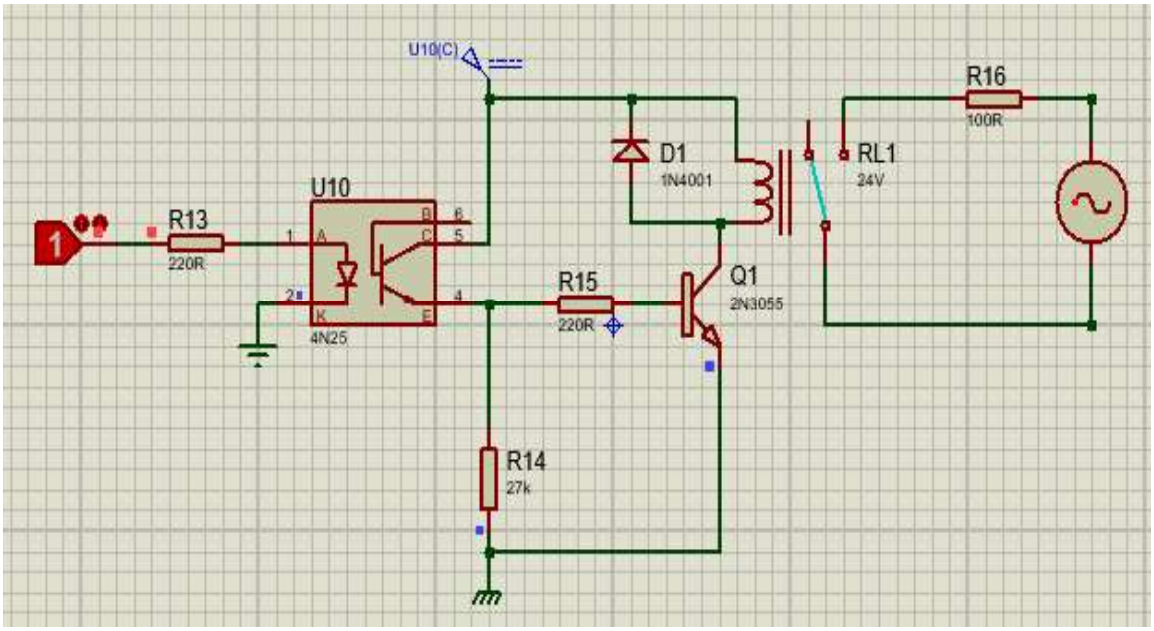

Figura 3.15 Simulación cuando no hay estaciones de parqueo disponibles

Desarrollo del programa para la tarjeta Arduino UNO

La programación se realizó para dos módulos de Arduino UNO, se utilizó dos por temas de costo, estas tarjetas son estables, tienen máxima capacitancia y soportan hasta 8 dispositivos conectados a ellas.

El lenguaje de programación de Arduino está basado en C++ y aunque la referencia para el lenguaje de programación de Arduino está en su página principal, también es posible usar comandos estándar de C++.

El lenguaje de programación Arduino se puede dividir en tres partes principales: funciones, valores (variables y constantes) y estructura.

En la Figura 3.16 y Figura 3.17 se puede observar la distribución de los dispositivos conectados a las tarjetas Arduino UNO y su funcionamiento.

También se detallan los diagramas de flujo de la programación de cada tarjeta Arduino.

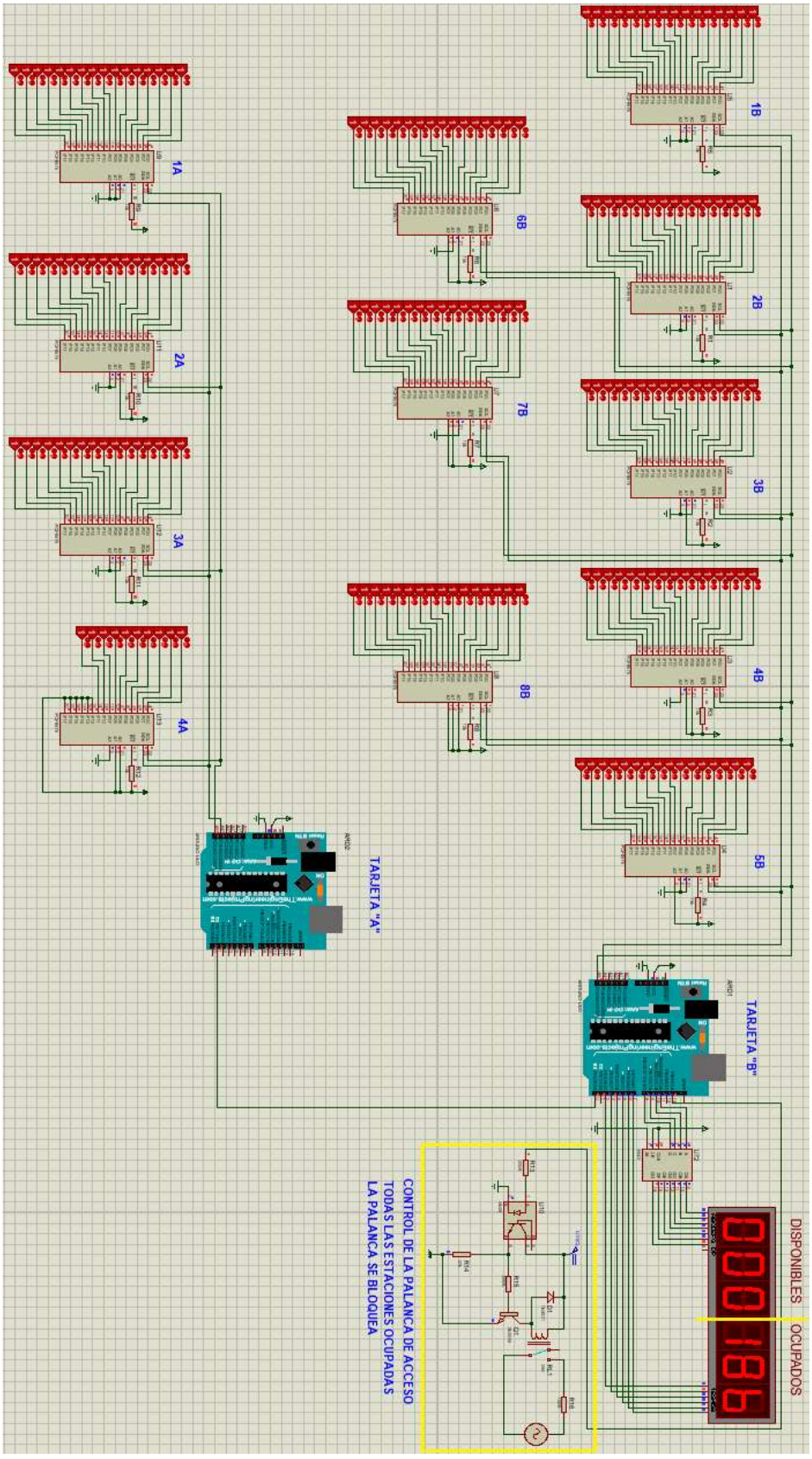

Figura 3.16 Simulación del sistema de parqueo sin las estaciones disponibles

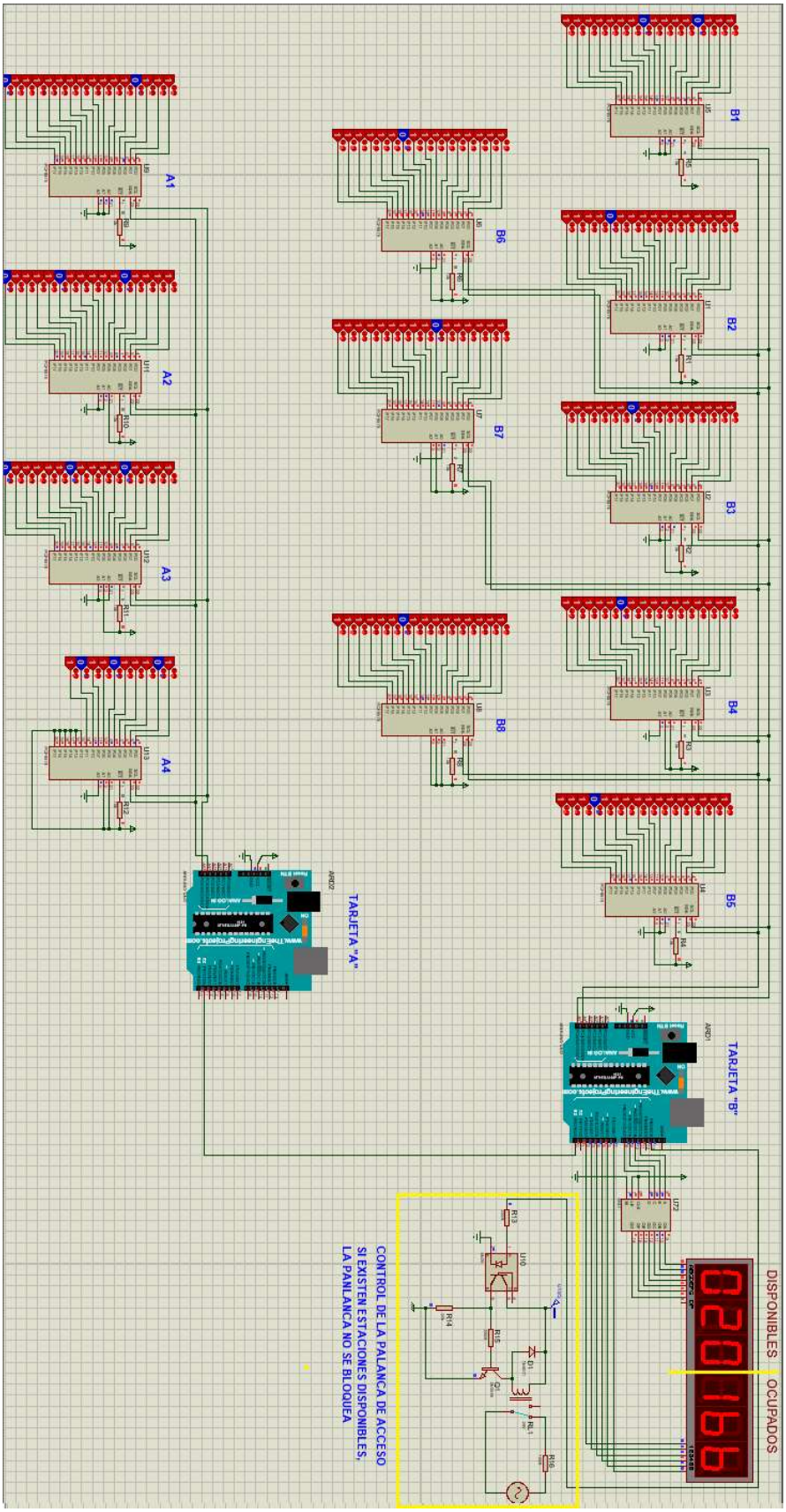

Figura 3.17 Simulación del sistema de parqueo con las estaciones disponibles

Programación módulo A

La programación en el módulo A se realizó de acuerdo al siguiente diagrama de flujo:

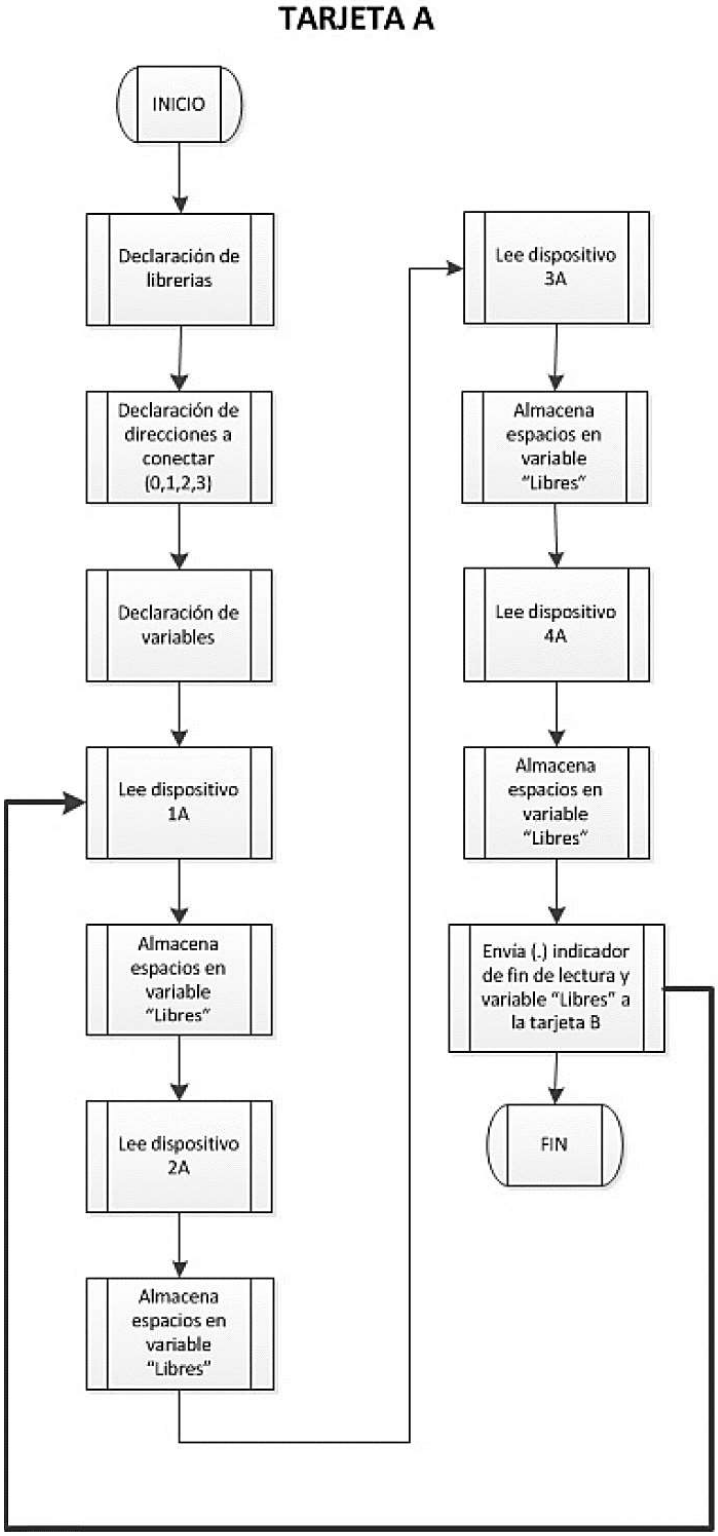

Figura 3.18 Diagrama de flujo Módulo A

La programación se puede explicar de la siguiente manera:

1. Se declara las librerías, en este caso utilizo la librería “Wire.h” para trabajar con I2C y la librería “I2cDiscreteIoExpander.h” por los expansores que estoy utilizando.

2. Se declara las direcciones a las que se van a conectar el bus de datos, en el caso del módulo A, voy a ocupar las direcciones 0, 1, 2, 3 que hacen referencia a los expansores 1A, 2A, 3A, 4A.

3. Se declaran las variables y los tipos de datos

4. Se inicializa el bucle

5. Empieza la lectura de cada uno de los expansores, cada uno tiene 16 entradas de datos, es decir 16 bits. Se recopila la información del expansor a modo de barrido y se obtienen los puestos disponibles y ocupados los cuales están señalados con un 0 y 1 lógico. Para recopilar los datos se va tomando cada bit y se divide para 2 en binario. El procedimiento se repite con los tres dispositivos restantes.

6. Obtiene el valor total de espacios libres y los almacena en la variable “libres”

7. Cuando el módulo ha terminado la lectura de los dispositivos conectados, imprime un “.” (punto), esto es para indicar al módulo B que la comunicación terminó.

Lo antes mencionado traducido en el lenguaje de programación C++ para el módulo A se desarrolló de la siguiente manera:

```
1. #include <Wire.h>
2.
3. #include <I2cDiscreteIoExpander.h>
4.
5. // instantiate I2cDiscreteIoExpander object
6. I2cDiscreteIoExpander device[8] = { 0, 1, 2, 3};
7.
8. long data;
9. int n;
10.
11.     void setup()
```

```
12.     {
13.       // initialize i2c interface
14.       Wire.begin();
15.
16.       Serial.begin(9600);
17.
18.     }
19.
20.
21.     void loop()
22.     {
23.
24.       uint8_t status, i;
25.       static uint16_t j;
26.       int libres = 0;
27.        for (i = 0; i < 4; i++)
28.       {
29.       //Intento leer palabra de 16 bits
30.
31.       status = device[i].digitalRead();
32.       if (TWI_SUCCESS == status)
33.       {
34.         data = device[i].getPorts();
35.         for(j=0 ; j<16 ; j++){
36.             n = data % 2;
37.             libres = libres + n;
38.             data = data / 2;
39.         }
40.       }
41.     }
42.     Serial.print(libres);
43.     Serial.print(".");
44.     delay(100);
45.     }
```

Programación módulo B

La programación en el módulo B se realizó de acuerdo al siguiente diagrama de flujo:

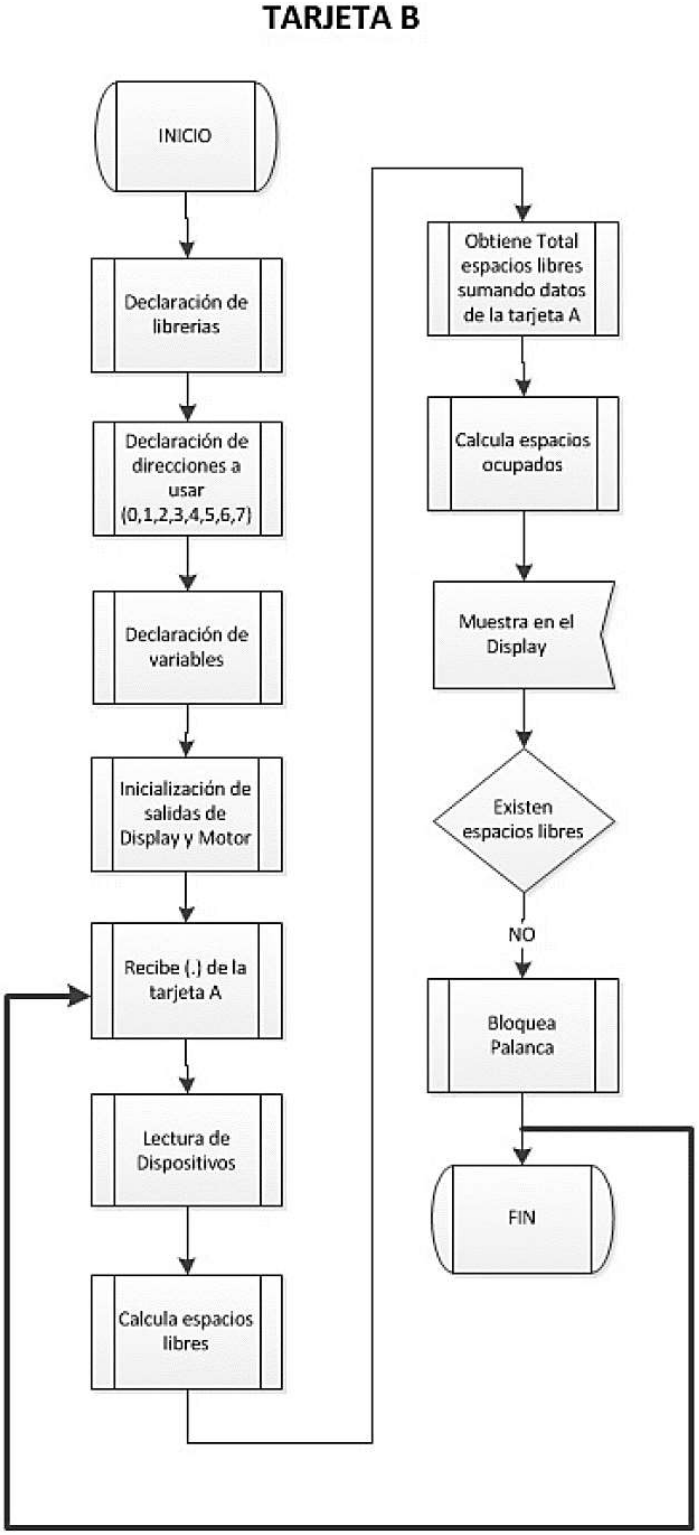

Figura 3.19 Diagrama de flujo Módulo B

La programación se puede explicar de la siguiente manera:

1. Se declaran las librerías, en este caso utilizo la librería “Wire.h” para trabajar con I2C y la librería “I2cDiscreteIoExpander.h” por los expansores que estoy utilizando.

2. Se declaran las direcciones a las que voy a conectar el bus de datos, en el caso del módulo A, voy a ocupar las direcciones 0, 1, 2, 3 que hacen referencia a los expansores 1B, 2B, 3B, 4B, 5B, 6B, 7B, 8B.

3. Se declaran las variables y los tipos de datos

4. Se inicializan las salidas para los Displays y el motor

5. Se inicializa el bucle

6. La variable libres0= son los datos del Módulo A y la variable libres0 se utiliza para acumular los datos del otro módulo.

7. Empieza la lectura de cada uno de los expansores, de la misma forma que en el módulo A.

8. Se obtiene el total de espacios libres adicionando los datos que se obtuvo del Módulo A.

9. Para obtener el valor total de espacios libres como número entero se aplica la fórmula:

   Libres0= libres0 x 10 + (lectura - 48)

   Esto se aplica debido a que la lectura llega por el puerto en código ASCII y se debe transformar a entero, a este puerto van a llegar valores de 0 a 9.

10. Se muestra el mensaje en los displays.

11. Se realiza el cálculo para mostrar las centenas, decenas y unidades.

12. Se aplica la fórmula para calcular los espacios ocupados con la fórmula

$$Ocupados = 186 - libres$$

13. Se repite el cálculo para mostrar las centenas, decenas y unidades.

14. Se revisan los espacios disponibles, si todas las estaciones están ocupadas se envía un 1 lógico para que empiece a funcionar el sistema que desactiva la palanca de acceso.

15. Se realiza el proceso de acuerdo a los segmentos que se colocan en On para mostrar cada número en el display según la tabla de verdad del decodificador a 7 segmentos.

Tabla 3.2 Tabla de verdad de un decodificador a 7 segmentos

ENTRADAS				**SALIDAS**							**DECIMAL**
A	**B**	**C**	**D**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9

En la siguiente figura se aprecia la definición que se utliza para mostrar números en el *Display* de 7 segmentos.

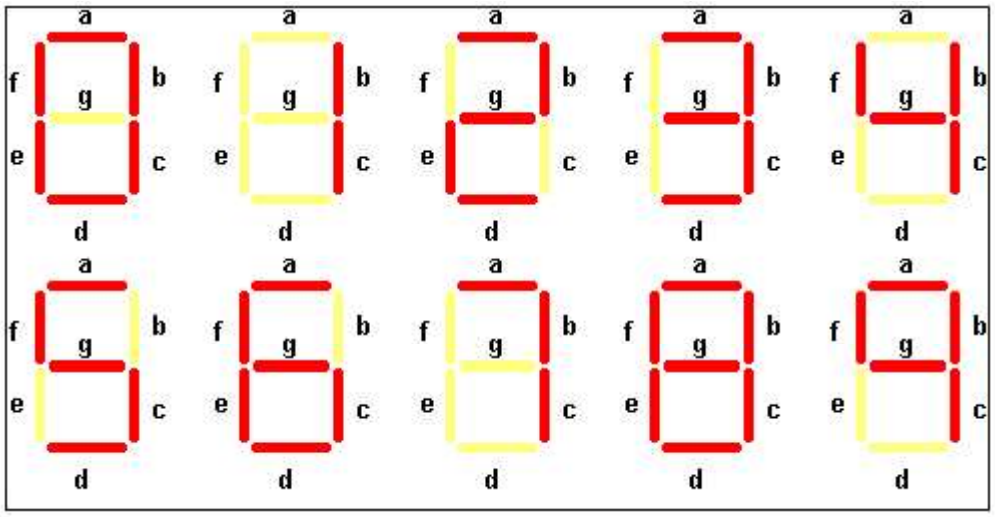

Figura 3.20 Descripción para mostrar números en el *Display*

Lo antes mencionado traducido en el lenguaje de programación C++ para el módulo B se desarrolló de la siguiente manera:

```
#include <Wire.h>

#include <I2cDiscreteIoExpander.h>

// instantiate I2cDiscreteIoExpander object
I2cDiscreteIoExpander device[8] = { 0, 1, 2, 3, 4, 5, 6, 7};

long data;
int n;
     // Declaro variables enteras para mostrar unidades, decenas y centenas en el display

      int centena = 0;
      int decena = 0;
      int unidad = 0;
     // Variable entera para para calcular espacios ocupados
      int ocupados = 0;
     // Variable entera que es utilizada para el retraso
      int t = 3;
     // Variable que es utilizada para obtener el valor entero de espacios libres
      int libres0 = 0;
     // Variable equivalente a byte un tipo de dato que puede almacenar entre 0 y 255
      uint8_t status, i;
     // Variable estática para leer una palabra de 16 bits
      static uint16_t j;
     // Variable entera para calcular espacios libres totales
```

```
26.       int libres = 0;
27.      // Variable de character para leer el dato recibido
28.       char lectura;
29.
30.
31.      void setup()
32.      {
33.        // initialize i2c interface
34.        Wire.begin();
35.
36.        Serial.begin(9600);
37.      // Declaro los pines donde irán conectados los
   expansores I2C
38.        pinMode(13,OUTPUT);
39.        pinMode(2,OUTPUT);
40.        pinMode(3,OUTPUT);
41.        pinMode(4,OUTPUT);
42.        pinMode(5,OUTPUT);
43.        pinMode(6,OUTPUT);
44.        pinMode(7,OUTPUT);
45.        pinMode(8,OUTPUT);
46.        pinMode(9,OUTPUT);
47.        pinMode(10,OUTPUT);
48.        pinMode(11,OUTPUT);
49.        pinMode(12,OUTPUT);
50.
51.      }
52.
53.      void loop()
54.      {
55.
56.      // Leo dato recibido
57.       if (Serial.available() > 0) {
58.          lectura = Serial.read();
59.
60.          if (lectura == '.') {
61.            libres = 0;
62.
63.            //Leo todos los dispositivos conectados al bus
   I2C
64.            for (i = 0; i < 8; i++)
65.              {
66.              //Intento leer palabra de 16 bits
67.              status = device[i].digitalRead();
68.
69.              if (TWI_SUCCESS == status)
70.                {
71.                data = device[i].getPorts();
72.
73.                for(j=0 ; j<16 ; j++){
74.                  n = data % 2;
75.                  libres = libres + n;
```

```
76.                 data = data / 2;
77.                 }
78.               }
79.             }
80.           libres = libres + libres0;
81.           libres0 = 0;
82.         }
83.
84.         else{
85.           libres0 = libres0 * 10 + (lectura - 48);
86.         }
87.       }
88.
89.     // Muestro mensaje en display
90.     //Muestro centena de espacios libres, se realiza el
   cálculo para obtener sólo la centena de un número de 3
   dígitos.
91.       centena = libres / 100;
92.       digitalWrite(7,LOW);
93.       binario(centena);
94.       digitalWrite(2,HIGH);
95.       delay(t);
96.
97.     //Muestro decena de espacios libres, se realiza el
   cálculo para obtener sólo la decena de un número de 3
   dígitos.
98.       decena = (libres - centena * 100) / 10;
99.       digitalWrite(2,LOW);
100.      binario(decena);
101.      digitalWrite(3,HIGH);
102.      delay(t);
103.
104.    //Muestro unidad de espacios libres, se realiza el
   cálculo para obtener sólo la unidad de un número de 3
   dígitos.
105.      unidad = (libres - centena * 100 - decena * 10);
106.      digitalWrite(3,LOW);
107.      binario(unidad);
108.      digitalWrite(4,HIGH);
109.      delay(t);
110.
111.      ocupados = 186 - libres;
112.    //Muestro centena de espacios ocupados, se realiza el
   cálculo para obtener sólo la centena de un número de 3
   dígitos.
113.      centena = ocupados / 100;
114.      digitalWrite(4,LOW);
115.      binario(centena);
116.      digitalWrite(5,HIGH);
117.      delay(t);
118.
```

```
119.    //Muestro decena de espacios ocupados, se realiza el
   cálculo para obtener sólo la decena de un número de 3
   dígitos.
120.     decena = (ocupados - centena * 100) / 10;
121.     digitalWrite(5,LOW);
122.     binario(decena);
123.     digitalWrite(6,HIGH);
124.     delay(t);
125.
126.    //Muestro unidad de espacios ocupados, se realiza el
   cálculo para obtener sólo la unidad de un número de 3
   dígitos.
127.     unidad = (ocupados - centena * 100 - decena * 10);
128.     digitalWrite(6,LOW);
129.     binario(unidad);
130.     digitalWrite(7,HIGH);
131.     delay(t);
132.
133.    //Reviso si quedan espacios disponibles
134.    if (libres == 0) {
135.      digitalWrite(13, HIGH);
136.    }
137.    else {
138.      digitalWrite(13,LOW);
139.    }
140.    }
141.     // Se compara el valor de cada variable con el valor
   especificado en cada case, si los valores coinciden se
   ejecuta el código dentro del case correspondiente y se
   habilitan y deshabilitan los pines para ir formando los
   números en el display de 0 a 9.
142.    int binario(int valor){
143.      switch(valor){
144.        case 0:
145.          digitalWrite(8,LOW);
146.          digitalWrite(9,LOW);
147.          digitalWrite(10,LOW);
148.          digitalWrite(11,LOW);
149.          break;
150.        case 1:
151.          digitalWrite(8,LOW);
152.          digitalWrite(9,LOW);
153.          digitalWrite(10,LOW);
154.          digitalWrite(11,HIGH);
155.          break;
156.        case 2:
157.          digitalWrite(8,LOW);
158.          digitalWrite(9,LOW);
159.          digitalWrite(10,HIGH);
160.          digitalWrite(11,LOW);
161.          break;
162.        case 3:
```

```
163.           digitalWrite(8,LOW);
164.           digitalWrite(9,LOW);
165.           digitalWrite(10,HIGH);
166.           digitalWrite(11,HIGH);
167.           break;
168.         case 4:
169.           digitalWrite(8,LOW);
170.           digitalWrite(9,HIGH);
171.           digitalWrite(10,LOW);
172.           digitalWrite(11,LOW);
173.           break;
174.         case 5:
175.           digitalWrite(8,LOW);
176.           digitalWrite(9,HIGH);
177.           digitalWrite(10,LOW);
178.           digitalWrite(11,HIGH);
179.           break;
180.         case 6:
181.           digitalWrite(8,LOW);
182.           digitalWrite(9,HIGH);
183.           digitalWrite(10,HIGH);
184.           digitalWrite(11,LOW);
185.           break;
186.         case 7:
187.           digitalWrite(8,LOW);
188.           digitalWrite(9,HIGH);
189.           digitalWrite(10,HIGH);
190.           digitalWrite(11,HIGH);
191.           break;
192.         case 8:
193.           digitalWrite(8,HIGH);
194.           digitalWrite(9,LOW);
195.           digitalWrite(10,LOW);
196.           digitalWrite(11,LOW);
197.           break;
198.         case 9:
199.           digitalWrite(8,HIGH);
200.           digitalWrite(9,LOW);
201.           digitalWrite(10,LOW);
202.           digitalWrite(11,HIGH);
203.           break; //La sentencia switch termina con un
   break.
204.       }
205.     }
```

3.4 Presupuesto referencial del proyecto

El presupuesto referencial del proyecto se obtuvo mediante la proforma de una tienda electrónica y el mismo se encuentra detallado a continuación y en el Anexo D se encuentra la proforma solicitada al proveedor.

Tabla 3.3 Presupuesto referencial del proyecto

ITEM	DESCRIPCIÓN	CANTIDAD	UNITARIO	MONTO
1	Adaptador de voltaje de 5VDC - 2Amp	1	5.00	5.00
2	Adaptador de voltaje de 24VDC - 2Amp	1	16.5	16.50
3	Foco LED impermeable, 24 V, para exteriores, 9W - rojo	186	27.18	5055.48
4	Foco LED impermeable, 24 V, para exteriores, 9W - verde	186	27.18	5055.48
5	Sensor de rango IR análogo 150cm distancia gp2y0a02yk0f	186	16.2	3013.20
6	LM324	186	0.31	57.66
7	LM555	186	0.15	27.90
8	Potenciómetros de 10K logarítmico	186	0.26	48.36
9	Potenciómetros de 1M logarítmico	186	0.26	48.36
10	Resistencias de 10K - 1K - 2-220Ohm - 1-27K	376	0.01	3.76
11	Diodos 1N4148	186	0.08	14.88
12	Compuerta NOT 74LS04	186	0.48	89.28
13	Capacitor electrolítico de 4,7uf - 50V	186	0.04	7.44
14	Transistores 2n3055	373	1.25	466.25
15	Optoacoplador 4N25	1	0.4	0.40
16	Diodo 1N4001	1	0.08	0.08
17	Relé de 5 pines bobina de 24VDC 1 contacto	1	0.45	0.45
18	PCF8575 Remote 16-Bit	12	6.6	79.20
19	Resistencias de 10k 1/4W	12	0.01	0.12
20	Arduino UNO	2	15.5	31.00
21	Codificador 4543	1	3.2	3.20
22	Display de 7 segmentos Anodo común de 18"	6	4	24.00
23	Cable UTP Cat 6 multifilar 2 Bobinas	823	0.43	353.89
24	Cable de luz #12 sólido	823	0.32	263.36
25	Breakers 40 Amp 1 Polo	1	6.25	6.25
26	Tubo Desague EC 110 mm x 3 m - Plastigama	200	12	2400
27	Tubo estructural cuadrado de acero 6.00 m GALVANIZADO	15	3	45.00
28	Mano de obra civil (5 trajadores) por día	7 días	25	875.00
			SUMAN	$ 17,991.50
			Subtotal	$ 17,991.50
			12 % IVA	$ 2,158.98
			TOTAL	$ 20,150.48

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El análisis del área del parqueadero en la sede Queri permitió conocer la disponibilidad del mismo para poder llevar a cabo la simulación del sistema de control, establecer los lugares donde se deben ubicar los sensores, las fuentes, las tarjetas Arduino, el tablero electrónico y el resto del sistema de control.

- Con el diseño del sistema de control del parqueo inteligente se logró mitigar el problema de la congestión vehicular en la zona y brindar información en tiempo real a los usuarios sobre la disponibilidad de las estaciones.

- Para la simulación del sistema de control se realizó el cálculo de caídas de voltaje, protecciones del sistema, tipos de dispositivos electrónicos, entre otros. Esto con el fin de obtener los resultados esperados y poder presentar un presupuesto referencial del sistema.

- La norma ANSI TIA 569-A indica cómo se debe enrutar el cableado en espacios y recorridos de edificios comerciales de telecomunicaciones y la norma ANSI TIA EIA 569 hace referencia a la interconexión de edificios o espacios, en este caso entre las estaciones de parqueo con el centro de control.

- El protocolo I2C requiere resistencias de Pull-UP de las líneas a Vcc. En Arduino no se instalan estas resistencias, ya que la librería Wire activa las resistencias internas de Pull-UP que tienen un valor de entre 20-30 (KΩ).

- Una ventaja de los sensores *SHARP* GP2Y0A02YK es que no son sensibles a la luz ambiental o el sol que es un gran enemigo de los sensores infrarrojos, este sensor utiliza una luz intermitente con una frecuencia determinada, que en el receptor es filtrada y elimina cualquier otra fuente de luz diferente a la frecuencia emitida.

4.2 Recomendaciones

- Para poder realizar la visualizar la simulación del sistema de control de parqueo se debe utilizar el software *Proteus Desig Suite 8.7* pues en versiones anteriores puede existir inconvenientes para ejecutar la simulación.

- Se debe tomar en cuenta la distancia a la que se desea que el sensor detecte un objeto con el fin de evitar errores en la recepción de datos, pues al tratarse de un sensor analógico éste puede detectar objetos en una distancia de hasta 1,50 metros.

- No es pertinente utilizar sensores de ultrasonidos HC-SR04 detectores de objetos, pues en estos no se puede discriminar el tipo de objeto que detecta y los factores externos como el ruido afectan el funcionamiento del mismo.

- Los displays que se deben implementar en el tablero deben ser de una medida superior a los comunes, ya que para la simulación se utilizó los displays de 7 segmentos de ánodo común de tamaño normal.

- Para la protección de los reguladores de voltaje utilizados en la simulación de la fuente se puede utilizar disipadores de potencia de 10 (W).

- Al realizar el cálculo de la corriente total que circula por el sistema se obtuvo un valor de 32,25 (A) por lo que se debe utilizar un breaker de 40 (A) de 1 polo para proteger al sistema contra cortos y sobre cargas.

5 BIBLIOGRAFÍA

[1] «hipertextual,» 03 2014. [En línea]. Available: https://hipertextual.com/archivo/2014/03/hardware-novatos-arduino/. [Último acceso: 01 07 2019].

[2] «Wikipedia,» 27 05 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Sensor_infrarrojo. [Último acceso: 01 07 2019].

[3] «Naylampmechatronics,» 07 01 2014. [En línea]. Available: https://naylampmechatronics.com/sensores-proximidad/204-sensor-infrarrojo-de-distancia-sharp-gp2y0a02.html. [Último acceso: 01 07 2019].

[4] «Robots Argentin,» [En línea]. Available: http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/. [Último acceso: 03 07 2017].

[5] «NXP,» www.nxp.com, 25 02 2019. [En línea]. Available: https://www.nxp.com/products/analog/interfaces/ic-bus/ic-general-purpose-i-o/remote-16-bit-i-o-expander-for-i2c-bus:PCF8575. [Último acceso: 03 07 2019].

[6] «Electrontools,» www.electrontools.com, 09 03 2016. [En línea]. Available: https://www.electrontools.com/Home/WP/2016/03/09/display-7-segmentos/. [Último acceso: 10 07 2019].

[7] «www.areatecnologia.com,» [En línea]. Available: https://www.areatecnologia.com/electronica/como-es-un-led.html. [Último acceso: 10 07 2019].

[8] «es.wikipedia.org,» 13 07 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Resistencia_el%C3%A9ctrica. [Último acceso: 18 07 2019].

[9] Teoman, «www.comofunciona.co.com,» 14 09 2018. [En línea]. Available: http://comofunciona.co.com/una-resistencia-electrica/. [Último acceso: 18 07 2019].

[10] I. Mecafenix, «www.ingmecafenix.com,» 2019. [En línea]. Available: https://www.ingmecafenix.com/electronica/el-capacitor/. [Último acceso: 18 07 2019].

[11] «www.artefactos.leame.com,» [En línea]. Available: http://artefactos.leame.com/reguladores-de-tension-78xx-y-79xx/. [Último acceso: 18 07 2019].

[12] «www.electrontools.com,» 09 03 206. [En línea]. Available: https://www.electrontools.com/Home/WP/2016/03/09/regulador-de-voltaje-7805/. [Último acceso: 18 07 2019].

[13] «www.microjpm.com,» [En línea]. Available: https://www.microjpm.com/products/l7824-regulador-de-voltaje/. [Último acceso: 18 07 2019].

[14] I. Mecafenix, «www.ingmecafenix.com,» 2019. [En línea]. Available: https://www.ingmecafenix.com/electronica/temporizador-555/. [Último acceso: 18 07 2019].

[15] «www.habitatyvivienda.gob.ec,» 06 2018. [En línea]. Available: https://www.habitatyvivienda.gob.ec/wp-content/uploads/downloads/2018/06/NTE-INEN-2248-ESTACIONAMIENTOS.pdf. [Último acceso: 15 07 2019].

[16] N. Plasencia, «Github,» 05 11 2017. [En línea]. Available: https://github.com/NestorPlasencia/hackspace-electronica/wiki/Semana-1-Estructura-y-Programacion-de-Arduino. [Último acceso: 01 07 2019].

6 ANEXOS

Anexo A Datasheets de los elementos utilizados

Anexo B Proforma para la implementación del parqueadero