UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de una plataforma capaz de administrar el control de usuarios y automatizar el proceso de generación de credenciales en un punto de acceso MikroTik

Trabajo de graduación presentado por Edwin Fernando Coronado Roche
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

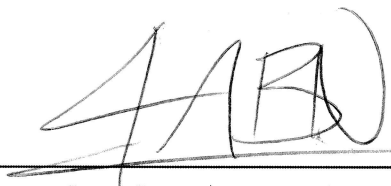
Diseño e implementación de una plataforma capaz de administrar el control de usuarios y automatizar el proceso de generación de credenciales en un punto de acceso MikroTik

Trabajo de graduación presentado por Edwin Fernando Coronado Roche
para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2019

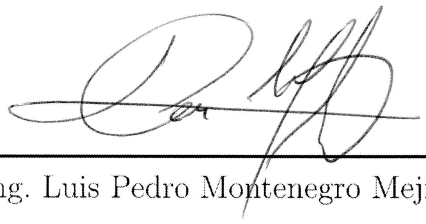
Vo.Bo.:



(f) _____

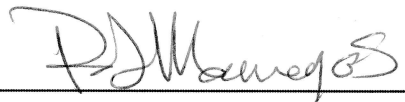
Ing. José Antonio Bagur Nájera

Tribunal Examinador:



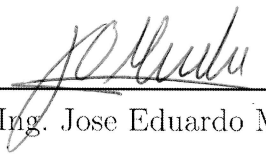
(f) _____

Ing. Luis Pedro Montenegro Mejicanos



(f) _____

Ing. Pablo Daniel Mazariegos de la Cerda



(f) _____

Ing. Jose Eduardo Morales Espinoza

Fecha de aprobación: Guatemala, 2 de diciembre de 2019.

Prefacio

Este trabajo es el fruto del aprovechamiento de varias habilidades y varios conocimientos adquiridos -unos cuantos de las muchas ramas que existen en la electrónica-, durante estos años de estudio de la Ingeniería Electrónica. El proyecto es un acercamiento hacia el desarrollo de una plataforma desde su fase de investigación y desarrollo, hasta pruebas del prototipo final.

Agradezco, en primer lugar y sobre todas la cosas, a Dios por todas la oportunidades que me ha entregado en la vida.
A mis padres, Edwin Coronado y Surama Roche, por su apoyo y por la oportunidad de permitirme estudiar en una casa de estudios tan prestigiosa.

A mis hermanos, Bryan Coronado y Katerine Coronado, por su apoyo incondicional en todo momento y toda ocasión, por su guianza y nuevos alientos cada día que pasaba.

A mi amigo y asesor, José Bagur, por su guianza, sus correcciones y su atención que permitieron la realización de este proyecto.

A mis amigos incondicionales de la carrera (Dan Álvarez, Jose Ángel Ochoa, Guillermo De León y en especial a Jose Pablo Ortega) por todos esos momentos inolvidables, ese apoyo mutuo, esa paciencia y por su ayuda en este proyecto.

A mis amigos incondicionales de la vida (Abraham Rodríguez, Santiago Solórzano, Sebastían Morales y los demás de la promoción del colegio) por su atención y apoyo en todo momento.

A mis nuevos amigos de la iglesia que últimamente han tenido un gran impacto positivo en mi vida, por sus consejos, su comprensión, guianza y sabiduría.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto es un acercamiento hacia el desarrollo de una plataforma -capaz de automatizar el proceso de generar y controlar usuarios en un *hotspot*- desde su fase de investigación -estudio sobre productos similares, red alámbricas e inalámbricas, microcontroladores y sistemas en un chip- y desarrollo -diseño y fabricación de la circuitería necesaria para el funcionamiento del producto- hasta pruebas del prototipo final.

En la sección inicial de investigación se menciona como ciertas plataformas similares son soluciones implementables, sin embargo, carecen de ciertas cualidades que no les permite ser una opción accesible en ciertos casos. En el caso de este proyecto se tiene un dirección final del tipo *"Plug and Play"*, de esta manera esta solución es más viable -en términos de simplicidad en la implementación- que otras disponibles en el mercado. En el diseño y desarrollo de la plataforma se consideraron ciertos criterios para cumplir con el requerimiento de funcionamiento de una forma eficaz. Por último, se observó como la plataforma diseñada e implementada fue capaz de lograr su objetivo y poder funcionar de una manera autónoma, buscando la mínima cantidad de interacción humana, salvo por la interacción con el cliente.

Abstract

This project is an aproach to the development of a platform from its investigation phase -studies about similar solutions, wireless and wired networks, microcontrollers and sytems on a chip- and develop phase -designing and fabrication of the involved circuitry neccessary for the right operation of the solution- to the test phase of the final prototype.

On the initial section of the investigation it is mentioned how similar platforms are implementable, nevertheless, those lack some attributes that doesnt allow these to be a viable option in some cases. This project has a "Plug and Play" intention, so this becomes a more viable option -in terms of implementation simpleness- compared to others that are available in the market. On the design and development phase some desgin criteria was accounted in order to satisfy funcionality requeriments in a effective way. Lastly, it was observed how the designed and produced plataform was capable of accomplish its objective and function autonomously, in search of the minimum human interaction, except for the user interacting with the platform.

CAPÍTULO 1

Introducción

En ocasiones es necesario la disponibilidad de un servicio de ancho de banda en lugares públicos, para lograr esta tarea se pueden colocar diferentes routers que funcionen en modo *hotspot*. Un *hostpot* o un punto de acceso, es una localidad que provee acceso a internet por medio de una red local inalámbrica (WLAN, por sus siglas en inglés). Sin embargo, si la red se encuentra de forma libre es muy probable que se encuentre saturada la mayor parte del tiempo, inhabilitando su utilidad. Esta saturación usualmente surge en dos escenarios, el primero sucede al momento de que la cantidad de clientes conectados a la red inalámbrica es mayor a la que el equipo está capacitado. El segundo escenario surge al momento de que la cantidad de usuarios conectados requieran de forma constante un ancho de banda mayor al que posee asignado el *hotspot*, obteniendo una navegación pobre e inconsistente.

De lo anterior, una solución ante el problema es generar credenciales únicas a los usuarios, de esta manera se podría limitar y administrar la cantidad de usuarios que hagan uso de la red al mismo tiempo. No obstante, este es un proceso manual que requiere de un operador que sea capaz de generar tales credenciales a demanda de los usuarios y administrar la caja, si es que se desea lucrar con el servicio. Asimismo, este debe poseer el conocimiento necesario para sincronizar las credenciales generadas con el *hotspot*, en caso de que no posea dichos conocimientos es necesaria una plataforma o *software* de terceros que sea capaz de manejar la información entre el operador y el *hotspot*.

Por lo tanto, este proyecto tiene como objetivo remover el operador y automatizar el proceso de administración de los usuarios y venta de servicios de internet de forma temporal. El proyecto consta de dos componentes principales, el primer componente es el router Mikrotik, este tendrá la tarea de proveer el servicio de ancho de banda a los usuarios generados. El segundo componente sería el controlador principal o administrador que genere las credenciales en base al servicio seleccionado, además, este componente debería ser capaz de recibir monedas como forma de pago. Cabe mencionar que se consideró otra forma de pago para proveer el servicio, siendo por medio de tarjetas RFID que ciertos usuarios poseen. Una vez adquirido el servicio temporal el usuario podrá recibir los datos de acceso por medio de una pantalla LCD y de forma física por medio de una impresora.

CAPÍTULO 2

Antecedentes

En la actualidad existen dos enfoques en la implementación de *hotspots*, el primero de ellos siendo un enfoque social donde se busca proveer un servicio a la comunidad con el objetivo de ser una fuente de desarrollo. Por esta rama de las plataformas existe una amplia cantidad de información sobre plataformas implementadas y desarrolladas utilizando herramientas de *software* libre o propietario. El segundo enfoque está más destinado a los negocios, buscando aprovechar los recursos disponibles para una redistribución de navegación por internet, al por menor. En este enfoque existe una menor cantidad de información ya que varias de las plataformas disponibles cuentan con *software* propietario y en ocasiones la implementación de estas soluciones incrementan el desembolso en la inversión inicial.

En el documento [1] se encuentra una investigación sobre la implementación de un *hotspot* como un servicio social para una comunidad, esta implementación no requiere de interacción humana para su administración ya que todos los usuarios están limitados a un mismo ancho de banda, lo cual tampoco es deseable en un enfoque de negocios.

En el documento [2] se encuentra una implementación de un *hotspot* con herramientas de *software* libre que puede generar *tickets* para la venta; sin embargo, requiere de lo que es la interacción humana para la administración de los usuarios.

De igual manera, empresas como "InternetPorFichas.com" [3] y "MikroTickets" [4] ya proveen plataformas para administrar la venta de internet por *tickets*; no obstante, es hasta este segundo semestre del 2019 que MikroTickets ofrece una solución de tipo "*Plug and Play*", la cual viene con un precio elevado comparado con una solución de origen local.

CAPÍTULO 3

Justificación

Una de las principales motivaciones de este proyecto es el poder aplicar teórica y experimentalmente los conocimientos adquiridos en el transcurso de la carrera. Asimismo, el campo al cual corresponde el proyecto, las telecomunicaciones, es uno de los campos en los cuales se desea crecer profesionalmente.

Por otra parte, un dispositivo como el que se propone podrá ofrecer nuevos nichos de mercado a empresas que se dedican a proveer servicio de internet o ISPs (por sus siglas en inglés, *Internet Service Provider*), dado que puede ser atractivo debido a su simplicidad de implementación por su diseño "*Plug and Play*" capaz de requerir mínima interacción humana, a un costo reducido en comparación con plataformas similares, así como un tamaño compacto que permita implementaciones en áreas limitadas o comercios limitados. De igual forma, este dispositivo podrá ser un valor agregado a los servicios que proveen dichas empresas, de manera que, los usuarios pueden adquirir u obtener tarjetas RFID que les permita hacer uso del dispositivo en forma de cortesía por un tiempo definido.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar e implementar una plataforma capaz de administrar y proveer servicio de internet a los usuarios de una forma eficaz y con mínima interacción humana por medio de un *hotspot*.

4.2. Objetivos específicos

- Configurar el equipo Mikrotik para que funcione como un *hotspot*.
- Conectar el controlador y el equipo Mikrotik para la transferencia de credenciales de los usuarios generados.
- Generar los usuarios y contraseñas de una forma pseudoaleatoria.
- Conectar el dispositivo tragamonedas con el controlador para poder llevar el control de créditos.
- Conectar el dispositivo receptor RFID con el controlador de usuarios.
- Conectar la impresora térmica para poder entrega al usuario una copia física del *ticket* generado.

CAPÍTULO 5

Alcance

La plataforma descrita en este documento tiene como alcance un sistema capaz de poder administrar usuarios en un *hotspot* MikroTik. Este sistema debe mantener cierto grado de profesionalismo que le permita ser considerado como una plataforma viable comparado con las que existen en el mercado actualmente. Asimismo, debe tener un enfoque "*Plug and Play*" donde no requiera de configuraciones mayores en el lugar de implementación, de esta manera minimizando asistencia técnica. Además, debe ser una plataforma donde la interacción con el usuario sea directa o de tipo "*straigthforward*" para evitar la mayor cantidad de interacción humana externa.

CAPÍTULO 6

Marco teórico

6.1. Redes de transmisión de datos

En la actualidad el requerimiento de poder transmitir y recibir datos es cada vez mayor, algo que hace tiempo parecía ser un lujo se ha convertido en una necesidad. Este crecimiento se ha dado debido al incremento de información que entidades y corporaciones manejan en las actividades de día a día; sin embargo, nada de esto sería posible si no existe una red capaz de proveer una conexión estable, confiable y segura por donde pueda fluir la información.

6.1.1. Topología de una red

La topología de una red es la configuración física y lógica de esta, está compuesta por nodos, elementos de una topología que reciben y transmiten información, y las líneas, medio por donde esta fluye. La configuración física es la distribución geométrica de los nodos y las líneas, mientras que la configuración lógica es la forma en que fluye la información.

Topología punto a punto

La topología punto a punto es una de las más simples ya que cuenta con un solo objetivo, ofrecer una conexión lógica entre dos nodos o terminales. Esta topología es utilizada usualmente para realizar enlaces corporativos entre dos localidades, con la topología punto a punto se pueden expandir redes de área local a pesar que se encuentren distantes, geográficamente hablando.

Topología de bus

Es una topología donde existe un único canal de transmisión de datos, todos los nodos comparten este canal. La información viaja de una forma secuencial lo que limita los anchos de banda que se pueden transmitir. Asimismo, debido a la dependencia total de la única

línea de transmisión todos los nodos pierden conexión al momento que esta sufra de algún daño.

Topología de estrella

Esta topología conecta todos los nodos o terminales entre sí, pero no de una forma directa ya que todos están conectados a un nodo central. El nodo central es el encargado de realizar direccionamiento o enrutamiento para que la información sea entregada a su destino. La topología de estrella posee la ventaja que su un nodo o una línea llegase a fallar los nodos restantes no perderán comunicación entre ellos. La topología de estrella es comúnmente utilizada para las redes de área local.

Topología de malla

La topología de malla es una configuración en donde los nodos o terminales poseen conexión directa entre ellos mismos sin depender de un nodo central. Esta configuración permite no depender de un nodo o línea en específico ya que es posible que la información fluya por diversas rutas. Usualmente esta topología es utilizada para las redes de área amplia ya que permite obtener redundancia en la red y segmentar la información para que esta tome diversas rutas al mismo tiempo, haciendo que el flujo total o *throughput* se incremente.

6.1.2. Tipos de redes

Redes locales

Una red área local o LAN (por sus siglas en inglés, *Local Area Network*) es una red que permite la interconexión de dispositivos e intercambio de información entre ellos en un área geográfica limitada [5]. En la Figura 1 se observa una topología de bus de una red LAN.

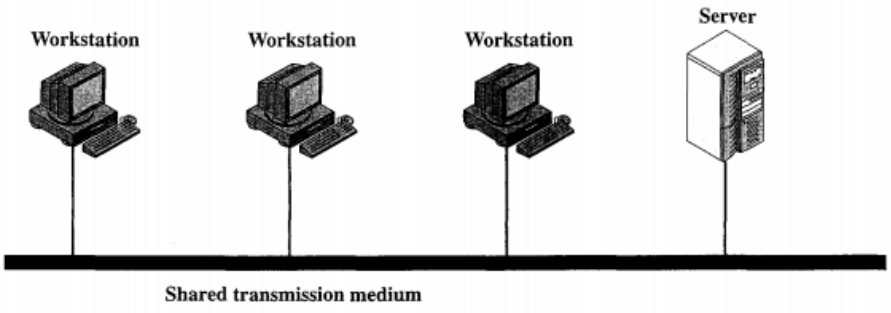


Figura 1: Topología convencional de una red LAN alámbrica [5]

Redes personales

Una red de área personal o PAN (por sus siglas en inglés, *Personal Area Network*) es una red que permite la interconexión de dispositivos en un área pequeña (menor a los 10 m), esta área se considera como el área de trabajo de un individuo. Usualmente, la información que se transmite en este tipo de redes es mínima y no requiere de mucho ancho de banda para

su funcionamiento, sin embargo, para este tipo de redes se acostumbra a utilizar protocolos que prioricen las bajas potencias sobre *throughput*.

Redes amplias

Una red de área local por si sola no es capaz de proveer lo que usualmente conocemos como "navegar en internet". Esta actividad requiere de una red que sea capaz de conectarse con diversos servicios y aplicaciones que se encuentran esparcidas por el mundo lo cual no es posible en una red LAN. Este tipo de redes que cubren un gran área geográfica son conocidas como redes de área amplia o WAN (por sus siglas en inglés, *Wide Area Network*), asimismo, en muchos casos se puede considerar una red WAN como un conjunto de diversas redes LAN.

Redes inalámbricas

En la última década ha habido una explosión en el mercado de dispositivos que son capaces de conectarse a redes inalámbricas, siendo estas redes que utilizan ondas de radio para conectar los dispositivos, sin la necesidad de utilizar cables de ningún tipo [6]. Estas redes también pueden ser clasificadas de la misma manera como lo son las redes alámbricas, por lo tanto puede existir redes inalámbricas de área local o WLAN (*Wireless Local Area Network*), o redes inalámbricas de área amplia o WWAN (*Wireless Wide Area Network*).

Las redes inalámbricas poseen la versatilidad de no requerir una infraestructura física para su funcionamiento, esto puede reducir costos al momento de su implementación y permite el levantamiento de redes de una forma práctica y eficiente. Asimismo, las redes inalámbricas han evolucionado de forma que pueden cubrir grandes extensiones de área y grandes distancias con la implementación de transmisores y receptores de alta potencia y alta sensibilidad, respectivamente. Dado esto, diversas organizaciones y entidades optan por la implementación de este tipo de redes ya que son capaces de cumplir con sus requerimientos.

No obstante, las redes inalámbricas poseen ciertas desventajas comparado con su complemento alámbrico, estas desventajas no les ha permitido ser adoptadas por completo en la industria y redes de telecomunicación. Entre ellas se encuentran las limitaciones de ancho de banda que se puede transportar, este depende de factores como la modulación utilizada para la transmisión de datos, mayores órdenes de modulación permiten mayores anchos de banda; sin embargo, estas transmisiones son más susceptibles a errores, por lo tanto se utilizan diversas técnicas de modulación para llegar a un equilibrio entre un ancho de banda aceptable y una alta fiabilidad en las transmisiones.

Además, este tipo de redes son más susceptibles al ruido electromagnético, en ambientes demasiados ruidosos la implementación no es viable. Por último, otra desventaja de estas redes es que para la transmisión de datos utiliza un recurso limitado, siendo este el espectro electromagnético. Dado esto existen regulaciones legales del espectro con el objetivo de poder hacer uso del espectro electromagnético sin crear interferencias entre dispositivos que utilicen frecuencias similares.

Es de importancia mencionar que existen diversas tecnologías sobre las cuales se puede montar una red inalámbrica, cada una de ellas se utiliza dependiendo de la aplicación que se le dará. Existen protocolos como el Bluetooth (protocolo IEEE 802.15.1) que es una tecnología que permite la implementación de redes WPAN de baja potencia y baja transmi-

sión de datos. La tecnología WiFi (protocolo IEEE 802.11) es una tecnología equilibrada en términos de potencia electromagnética y anchos de banda notables, actualmente esta tecnología puede manejar anchos de banda superiores a los 1Gpbs. Este protocolo es utilizado para la implementación de redes WLAN, sin embargo, dependiendo de la distancia podría considerarse que también puede ser utilizada para las redes WWAN. Además, existen otros protocolos que se diseñaron con el objetivo de cubrir grandes extensiones de tierra (distancias superiores a los 30 km) y tratando de proveer anchos de banda considerables, uno de ellos es el *Long Term Evolution* o LTE, este protocolo es utilizado para la implementación de redes WWAN.

6.1.3. Componentes de una red inalámbrica

Al momento de implementar una red inalámbrica es necesario contar de ciertos componentes que permitan la interconexión entre los dispositivos, entre los cuales son:

Antenas direccionales

Las antenas direccionales son las antenas donde enfoca su patrón de radiación en una dirección. Usualmente estas antenas son utilizadas en topologías de punto a punto ya que son capaces de proveer enlaces confiables y con mayor tolerancia ante el ruido electromagnético debido a su estrecho patrón de radiación.

Antenas omnidireccionales

Las antenas omnidireccionales son las antenas donde el patrón de radiación es en todas las direcciones, es decir se posee una cobertura de unos 360°. Es común encontrar este tipo de antenas en topologías de estrella y de malla, en las topología de estrella permite que los nodos o terminales conectarse con el nodo principal. En la topología de malla permite que dispositivos que se encuentren dispersados en un área puedan comunicarse entre ellos sin necesidad de poseer un nodo central.

Antenas sectoriales

Este tipo de antenas posee un patrón similar de radiación al de la omnidireccionales, sin embargo, este no es tan amplio ya que no cubre los 360° si no solamente un segmento o sector. Usualmente este tipo de antenas se utilizan en topologías de estrella y de malla pero con la ventaja que se puede realizar una selección de área por donde se desea irradiar la potencia.

Punto de acceso

Un punto de acceso o AP (por sus siglas en inglés, *access point*) es un dispositivo que interconecta dispositivos terminales para realizar una red inalámbrica de área local (WLAN). Este usualmente posee antenas omnidireccionales pero dependiendo de la aplicación se le pueden acoplar otros tipos de antenas para cumplir con los requerimientos de red. Asimismo, un *access point* puede realizar enrutamiento y estos poseen una puerta de enlace o *gateway* para poder conectarse con una red WAN (Internet); sin embargo, esto no siempre fue así ya que los AP requerían estar conectados a un enrutador o *router* para poder realizar el

enrutamiento hacia internet. Un *access point* que capaz de redireccionar tráfico desde los clientes hacia internet y viceversa es conocido como un *hotspot*.

Es de importancia mencionar que los *access points* tienen diversos modos de funcionamiento, el primero es modo raíz o *root*, este modo es el que permite que los dispositivos se conecten a esta estación y se crea la red WLAN. El segundo modo es el modo de repetidora o *repeater*, en este modo el AP se encuentra dentro del rango de una señal de WiFi para poder tomar la señal y reconstruirla, de esta forma se extiende el área donde la señal puede ser utilizada. El último modo, modo puente o *bridge*, es un modo donde los equipos se conectan a través de AP de una forma transparente como si no existiese ningún componente entre ellos.

Clientes inalámbricos

En una red inalámbrica uno de los componentes principales son los clientes que se contectarán a la red, anteriormente estos eran solamente equipos de cómputo; sin embargo, con el paso del tiempo y con la incursión de dispositivos móviles capaces de conectarse, recibir y transmitir información en una red WLAN y WWAN esto dejó de ser cierto. Un dispositivo es capaz de recibir y transmitir información debido a que poseen tarjetas de red o NICs (*Network Interface Controller* término utilizado en inglés), este componente permite transcribir la información lógica que se maneja dentro de un dispositivo a señales eléctricas y electromagnéticas.

6.2. Plataforma MikroTik

MikroTik es una compañía letona que fue fundada en 1996 con el objetivo de desarrollar *routers* y equipos inalámbricos para el uso en sistemas dedicados para proveedores de servicios de internet o ISP (por sus siglas en inglés, *Internet Service Provider*). En 1997 la empresa creó el sistema operativo RouterOS como una respuesta ante el requerimiento de sistemas capaces de manejar redes de alto rendimiento. En el año 2002 la empresa MikroTik tomó la decisión de crear sus propios equipos físicos o *hardware* bajo la marca RouterBOARD [7].

6.2.1. RouterOS

El sistema operativo RouterOS es un sistema basado en Linux, este fue diseñado por MikroTik. RouterOS viene como sistema operativo para los equipos RouterBOARD pero puede ser instalado en computadoras personales. El instalar el sistema operativo en una computadora personal permite dotarla con funcionalidades que son propias de los *routers*, entre estas funcionalidades podemos encontrar enrutamiento o *routing*, cortafuegos o *firewall*, manejo de anchos de banda, funcionamiento en modo punto de acceso y *hotspot*, red privada virtual o VPN (por sus siglas en inglés, *Virtual Private Network*), calidad de servicio o QoS (por sus siglas en inglés, *Quality of Service*), etc [8]. Entre las cualidades del sistema operativo poseemos las siguientes:

Configuración

RouterOS posee la versatilidad que posee varios métodos de configuración y acceso. Posee una interfaz que funciona por medio de linea de comando o CLI (por sus siglas en inglés, *Command Line Interface*), esta puede ser accesada físicamente con la computadora personal o remotamente por medio de protocolos como Telnet, MacTelnet (propietario de MikroTik) y SSH. Asimismo, se puede configurar los equipos por medio de una interfaz web y la aplicación Winbox para la plataforma Windows.

Firewall

El sistema operativo de MikroTik provee componentes de seguridad con el objetivo de proteger a los equipos que se conecten al sistema de amenazas externas e internas. El firewall implementado provee como función básica el filtrado de paquetes, esta función bloquea o permite el flujo de ciertos paquetes dentro de la red o externos a ella. Además, el firewall es capaz de funcionar en capa de aplicación del modelo OSI, es decir que el filtrado de paquetes se puede dar en base al contenido que estos poseen. Otra funcionalidad que posee el firewall implementado en el RouterOS es la traducción de dirección de red o NAT (por sus siglas en inglés, *Network Address Translation*), esta funcionalidad permite intercambiar paquetes entre diferentes redes cuando estas son ajenas entre ellas.

Enrutamiento

Una cualidad de este sistema operativo es la capacidad de trabajar con enrutamiento (o *routing*) estático y dinámico. Este soporta protocolos de enrutamiento para IPv4 (RIP v1 y v2, OSPF v2 y BGP v4) y para IPv6 (RIPng, OSPFv3 y BGP).

Conmutación de etiquetas multiprotocolo

La conmutación de etiquetas multiprotocolo o MPLS (por sus siglas en inglés, *Multiprotocol Label Switching*) es un estándar de conmutación de paquetes que se diseño con el objetivo unificar las soluciones propietarias de nivel en el modelo OSI. Asimismo, este protocolo es capaz de mejorar el rendimiento ya que la conmutación de paquetes o *packet switching* ocurre por medio de etiquetas propias del paquete en vez de utilizar las direcciones IP de destino de estos, dado esto al utilizar etiquetas la conmutación de paquetes toma un menor tiempo incrementando el rendimiento de una red.

Capacidades en base a licencia

El sistema operativo RouterOS funciona en base a licencias, actualmente la empresa MikroTik maneja cinco niveles de licencias. Los diferentes niveles de licenciamiento indican las limitaciones del sistema operativo, de esta manera el usuario puede seleccionar el nivel de licencia que se acomode a sus necesidades al momento de realizar una instalación en una computadora personal. En el caso de las placas RouterBOARD estos vienen con un nivel de licencia definido; sin embargo, es posible comprar un nivel más alto de licencia, aunque esto no es del todo recomendado ya que es posible que el *hardware* no posea el rendimiento para trabajar con más equipos de los que permite la licencia original. En la Figura 2 tenemos un cuadro comparativo entre los diferentes niveles de licencias.

Level number	0 (Trial mode)	1 (Free Demo)	3 (WISP CPE)	4 (WISP)	5 (WISP)	6 (Controller)
Price	no key	registration required	volume only	$45	$95	$250
Initial Config Support	-	-	-	15 days	30 days	30 days
Wireless AP	24h trial	-	-	yes	yes	yes
Wireless Client and Bridge	24h trial	-	yes	yes	yes	yes
RIP, OSPF, BGP protocols	24h trial	-	yes(*)	yes	yes	yes
EoIP tunnels	24h trial	1	unlimited	unlimited	unlimited	unlimited
PPPoE tunnels	24h trial	1	200	200	500	unlimited
PPTP tunnels	24h trial	1	200	200	500	unlimited
L2TP tunnels	24h trial	1	200	200	500	unlimited
OVPN tunnels	24h trial	1	200	200	unlimited	unlimited
VLAN interfaces	24h trial	1	unlimited	unlimited	unlimited	unlimited
HotSpot active users	24h trial	1	1	200	500	unlimited
RADIUS client	24h trial	-	yes	yes	yes	yes
Queues	24h trial	1	unlimited	unlimited	unlimited	unlimited
Web proxy	24h trial	-	yes	yes	yes	yes
User manager active sessions	24h trial	1	10	20	50	Unlimited
Number of KVM guests	none	1	Unlimited	Unlimited	Unlimited	Unlimited

Figura 2: Niveles de licenciamiento para RouterOS [7]

6.2.2. RouterBOARD

RouterBOARD es el nombre que MikroTik le otorgó al *hardware* que ellos han diseñado para sus routers y equipos de redes inalámbricas. Estos equipos son muy similares a una computadora ya que comparten diversos componentes a nivel de su arquitectura. El diseñar y fabricar sus propios equipos permitieron a MikroTik proveer soluciones competitivas en el mercado y capaces de competir con corporaciones que poseen años en el mercado, asimismo, MikroTik puede ofrecer soluciones configurables por el usuario que permiten adquirir solo los componentes necesarios.

El primer componente que comparten ambos equipos es la unidad de procesamiento central o CPU (por sus siglas en inglés, *Central Processing Unit*), este componente ejecuta instrucciones entregadas por el sistema operativo, cabe mencionar que RouterOS es un sistema operativo que está optimizado y es capaz de utilizar múltiples núcleos de procesadores. Esta característica de los equipos RouterBOARD se debe considerar al momento de realizar implementaciones de redes ya que la capacidad máxima de los usuarios concurrentes que se encuentran conectados al router tiene una gran dependencia de las características el CPU.

Otro componente que comparten el *hardware* RouterBOARD y una computadora personal es la memoria de acceso aleatorio o RAM (por sus siglas en inglés, *Random Access Memory*) y memoria de lectura o ROM (por sus siglas en inglés, *Read Only Memory*), ambas memorias son utilizadas durante el funcionamiento de los equipos MikroTik, sin embargo, la memoria RAM tienen gran importancia respecto a la cantidad de clientes ya que al momento que estos generen tráfico todas las reglas y configuraciones del cliente se cargan en esta memoria.

Por último, los equipos RouterBOARD poseen diversas ranuras o *slots* de expansión miniPCI o incluso miniPCI-E. Estas ranuras permiten que la placa madre del router pueda

utilizar componentes como tarjetas de red inalámbricas, puertos USB, puertos ethernet, puertos SFP o incluso expandir puertos ya existentes por medio de la intalación de placas hijas RouterBOARD.

6.3. Microcontroladores

En el año 1971 Intel lanzó al mercado el primer microprocesador en un chip, el Intel 4004. Sin embargo, para este entonces ya existía una demanda de microcontroladores en la industria, la empresa Texas Instruments ya había lanzado su microcontrolador TMS1802 "Calculadora en un chip" que fue utilizado en sus calculadoras de bolsillo. A pesar que ya existían otros chips destinados a esta tarea, el TMS1802 tenía la particularidad de ser multipropósito debido a su arquitectura. [9].

Un microcontrolador es un circuito integrado diseñado con el objetivo de controlar una o múltiples tareas. Este circuito integrado contiene todos los componentes que conforman una computadora e incluso más dentro de un mismo chip. En la Figura 3 se puede observar la arquitectura de un microcontrolador. En las siguientes subsecciones encontramos ciertos componentes de una arquitectura de un microcontrolador [10].

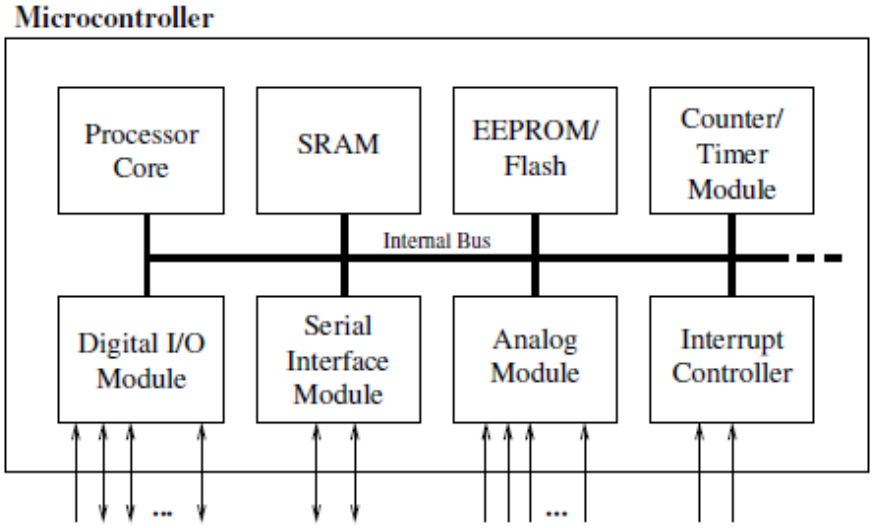


Figura 3: Arquitectura de un microcontrolador [10]

6.4. Arquitectura de un microcontrolador

6.4.1. Unidad central de procesamiento (CPU)

La unidad central de procesamiento o CPU (por sus siglas en inglés, *Central Proccessing Unit*) es un componente que comparten las computadoras y los microcontroladores. El CPU es el componente que ejecuta las instrucciones que están descritas en el programa, el ren-

dimiento de un microcontrolador se verá afectado por cuántas instrucciones puede realizar por ciclo, o si es el caso, cuántos ciclos son requeridos por una instrucción. Asimismo, la frecuencia máxima de operación de un microcontrolador tiene un impacto directo sobre el rendimiento de este. Internamente el CPU posee componentes que son necesarios para su funcionamiento, entre ellos poseemos la ALU, los registros y el *Stack Pointer*.

Unidad de lógica aritmética (ALU)

La unidad de lógica aritmética o ALU (por sus siglas en inglés, *Arithmetic Logic Unit*) es el componente encargado de realizar las operaciones aritméticas del procesador, esta unidad utiliza registros como fuente de información y realiza operaciones entre ellos, luego es almacenado en otro registro y en memoria. Además, esta unidad es diseñada dependiendo de la complejidad de las operaciones que realizará, es decir una ALU diseñada para realizar operaciones con registros de 32 bits y punto flotante será más compleja que una para 8 bits.

Registros

Los registros son una unidad de memoria volátil que funcionan en conjunto con la ALU. Usualmente un CPU posee un número definido de registros de propósito general, estos pueden ser utilizados como fuente y destino para extraer y almacenar datos de una operación. Asimismo, existen registros dedicados, estos dependen de cada arquitectura del CPU, estos registros tienen funciones específicas que solo son utilizados al momento de realizar ciertas operaciones, un ejemplo de este tipo de registros es el registro acumulador. El acumulador es un registro donde se guarda el resultado de una operación aritmética luego de que se finaliza dicha instrucción.

Puntero de pila

La pila es un segmento de memoria que posee instrucciones e información utilizada para ejecutar dichas instrucciones. El puntero de pila es un registro que se encarga de apuntar a la siguiente instrucción que se debe ejecutar, además, es capaz de regresar (o saltar) a una dirección en específico de la pila.

6.4.2. Memoria

La memoria es un componente esencial en las arquitecturas de computadoras y microcontroladores ya que es el componente que almacena las instrucciones y los resultados de estas. Esta se puede dividir en tres tipos de memoria según su funcionalidad y no operación, como lo es la memoria de acceso aleatorio y la memoria de solo lectura.

Registros

La memoria de registros es un bloque de memoria que se encuentra dentro de la unidad de procesamiento central, esto sucede ya que es la memoria que requiere del menor tiempo de acceso para no interferir con el rendimiento del CPU durante la ejecución de instrucciones. Sin embargo, debido a esto esta tiende a ser pequeña (órdenes de magnitud menores que las memorias externas al CPU). Esta memoria es de tipo volátil, es decir una vez el microcontrolador pierde energía todos los datos almacenados en los registros regresan a su

estado natural o por defecto.

Memoria de datos

La memoria de datos es un bloque de memoria que se encuentra externa al CPU, sin embargo, siempre se encuentra dentro de un microcontrolador. Esta memoria es utilizada usualmente para almacenamiento de información o datos utilizados durante la ejecución del programa.

Memoria de programa

La memoria de programa es un bloque de memoria no volátil que solo es modificada cuando se "programa" las instrucciones que se desean ejecutar dentro del microcontrolador. Este tipo de memoria es la de mayor tamaño dentro de un microcontrolador, esta memoria puede indicar qué tan extenso puede ser el programa o la cantidad de instrucciones que el microcontrolador puede ejecutar.

6.4.3. Entradas y salidas digitales

Las entradas y salidas o I/O (por sus siglas en inglés, *Input/Output*) de un microcontrolador son los pines digitales que este posee. Un pin digital es una interfaz del microcontrolador que utiliza funciones discretas en magnitud y continuas en el tiempo, es decir solo puede poseer valores definidos en voltaje. Usualmente, un grupo de ocho pines es agrupado como un puerto y estos pueden ser bidireccionales.

Los microcontroladores poseen registros que funcionan en conjunto con los puertos, el primer registro es el registro de dirección de información o DDR (por sus siglas en inglés, *Data Direction Register*), este registro tiene como funcionalidad establecer el modo de funcionamiento del cada pin de cada puerto, ya que este puede funcionar como entrada o salida. El segundo registro es el registro del puerto o *Port Register*, este registro controla el estado del pin por puerto en modo salida. El último registro, el registro de entrada de puerto o PIN (*Port Input Register*) es el registro que posee el estado del pin por puerto en modo de entrada.

6.4.4. Interfaces de comunicación

Un elemento clave en los microcontroladores son las interfaces de comunicaciones, estas permiten al microcontrolador enviar y recibir información a otros dispositivos como microntroladores, módulos que se utilizan e incluso computadoras. Estas interfaces se pueden categorizar dependiendo de la forma en que se transmiten los datos (de forma serial o paralela), sincronía (síncrono o asíncrono), tipo de conexión (conexión de bus o de punto a punto). Asimismo, existen diversos tipos de interfaces que poseen diferente modo de funcionamiento y funcionalidad.

Una interfaz serial es aquella en donde la información es enviada de forma secuencial, es decir bit por bit. Sin embargo, esta forma puede limitar el *throughput* de información ya que es limitada por la frecuencia de la interfaz serial. Una interfaz paralela es aquella que

utiliza diversas líneas para enviar y recibir información, de esta manera el *throughput* se incrementa pero el microcontrolador requiere de una mayor cantidad de pines para poder enviar y recibir la información.

La sincronía de una interfaz se refiere entre la relación que existe en el reloj del equipo transmisor y receptor, una interfaz síncrona es aquella que envía y recibe información solo cuando este reloj lo indica. Una interfaz asíncrona no sigue esta regla y puede enviar y recibir información en cualquier momento o al momento de que ocurra algún evento.

Una topología de bus es aquella donde múltiples módulos o microcontroladores se encuentran conectados utilizando un mismo medio de transmisión, sin embargo, es requerida una jerarquía para poder establecer al controlador maestro y a los dispositivos esclavos, de esta forma el maestro podrá seleccionar a que dispositivo esclavo enviar su información, al igual de esta forma el esclavo podrá mandar información solo al maestro.

SCI (UART)

La interfaz de comunicación universal asíncrona transmisor-recepetor o UART (por su siglas en inglés, *Universal Asynchronous Receiver-Transmitter*) es una interfaz que envía los datos de una forma secuencial asíncrona. Esto es posible ya que transmisor del dispositivo A se encuentra conectado al receptor del dispositivo B y viceversa, este no requiere de un reloj principal para indicar en qué momento se envía y se recibe la información. Esta interfaz es de tipo *full-duplex*, por lo tanto puede ser utilizada para enviar y recibir información al mismo tiempo.

SPI

La interfaz de periféricos serial o SPI (por sus siglas en inglés, *Serial Peripheral Interface*) es una interfaz síncrona de punto a punto que funciona con la jerarquía de maestro y esclavo (o esclavos si es necesario). Esta interfaz también funciona en modo *full-duplex* lo cual la hace atractiva en los proyectos. Es de importancia mencionar que la interfaz utiliza de cuatro lineas para su funcionamiento, las cuales son las siguientes:

- **MOSI** (Master Out, Slave In): Por medio de esta línea el esclavo o los esclavos reciben la información.
- **MISO** (Master In, Slave Out): Por medio de esta línea los esclavos transmiten información al maestro.
- **SCK** (System Clock): Esta línea es por donde el maestro transmite la señal de reloj.
- **SS** (Slave Select): Por medio de esta línea el dispositivo maestro es capaz de seleccionar el esclavo al cual la información debe ser recibida.

IIC (I^2C)

La interfaz inter-circuitos integrados o I^2C (por su siglas en inglés, *Inter-Integrated Circuits*) es una interfaz de comunicación síncrona. Al igual que la interfaz SPI esta funciona con la jerarquía de maestro-esclavo pero esta interfaz permite el funcionamiento de más de

un maestro dentro de un mismo bus de información. Además, esta interfaz utiliza solamente dos líneas de transmisión de datos. Esto es posible ya que en esta interfaz existe el direccionamiento de esclavos, es decir cada esclavo posee una dirección única y de esta forma es posible comunicarse utilizando solamente 2 líneas, sin embargo, solo puede funcionar en modo *half-duplex* ya que una de ellas es el reloj principal (*Serial Clock Line* o SCL) y la otra es la línea por donde la información es transmitida (*Serial Data Line* o SDA).

6.5. Microprocesadores

Un microprocesador es un circuito integrado de alto rendimiento y multipropósito, sin embargo, este carece de ciertos componentes en su arquitectura que lo diferencian respecto a los microcontroladores. En la Figura 4 es la arquitectura de un microprocesaador y se logra observar que este cuenta con solo dos componentes dentro del mismo empaquetado, el primero de ellos siendo la unidad de procesamiento central o CPU (por sus siglas en inglés, *Central Processing Unit*) y el segundo es la unidad de memoria.

El CPU en un microprocesador se considera de alto rendimiento debido a que comúnmente sus arquitectura utilizan instrucciones y registros de 32 y 64 bits, permitiendo a los microprocesadores manejar y operar datos de punto flotante con una mayor precisión y manejar instrucciones más complejas que aquellas utilizadas en los microcontroladores

La unidad de memoria en un microprocesador está limitada a memoria de registros y de datos, la memoria de programa se encuentra externa al microprocesador debido a que esta tiende a ser órdenes de magnitud más grande que la memoria interna del microprocesador y sus tiempos de acceso son mayores lo cual puede ralentizar el rendimiento del microprocesador.

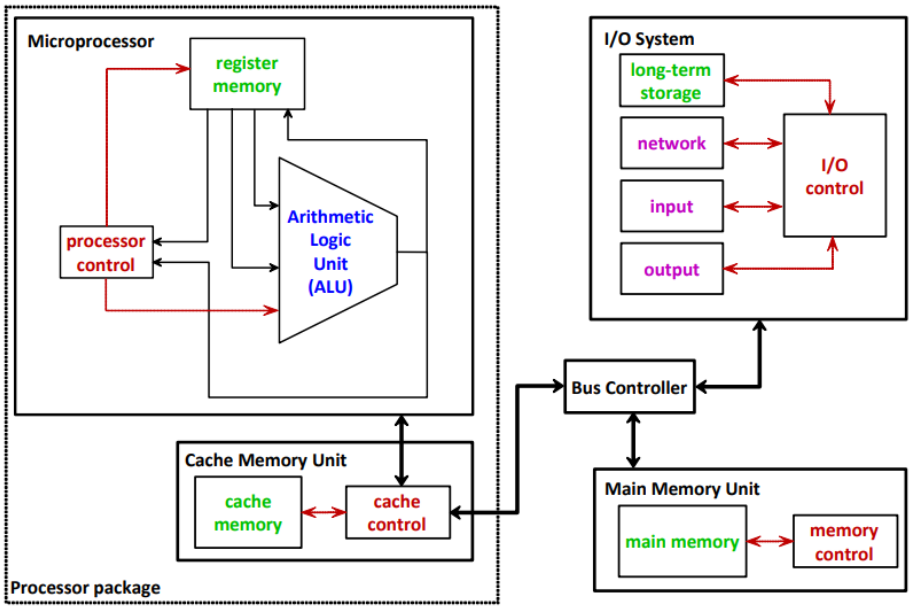


Figura 4: Arquitectura de un microprocesador [11]

6.6. Sistema en un chip

Sistema en un chip o SoC (por sus siglas en inglés, *System on a Chip*) es un circuito integrado que posee un procesador, un bus de datos y otros elementos en un sustracto monolítico. Elementos como memoria volátil, memoria no-volátil, entradas y salidas o I/O (por sus siglas en inglés, *Input/Output*) sistemas de procesamiento de señales, circuitos de señales mixtas y circuitos lógicos, son elementos que se pueden encontrar integrados en un solo chip [12].

CAPÍTULO 7

Estructura de la plataforma de automatización

La implementación final de la plataforma de automatización utiliza una Raspberry Pi como el controlador encargado de procesar las señales de las entradas, de las salidas y comunicación con el *router* MikroTik. La computadora Raspberry Pi cuenta con un SoC (*System on a Chip*) BCM2835, este posee un procesador ARM1176JZF-S que corre a una frecuencia de 700Mhz (para el modelo Raspberry Pi 1 B+) y es capaz de manejar diferentes interfaces de comunicación serial UART, SPI e I2C, al igual que entradas y salidas digitales con interrupciones.

La Figura 5 es un diagrama de la estructura final del controlador de usuarios, asimismo, se puede observar cómo diferentes componentes electrónicos utilizan diferentes interfaces de comunicación para la transmisión y recepción de información; siendo esta una de las razones por la cual se utilizó la plataforma Raspberry Pi como controlador principal, ya que posee el suficiente poder de procesamiento para controlar estos dispositivos de forma simultánea mientras se ejecuta el programa principal.

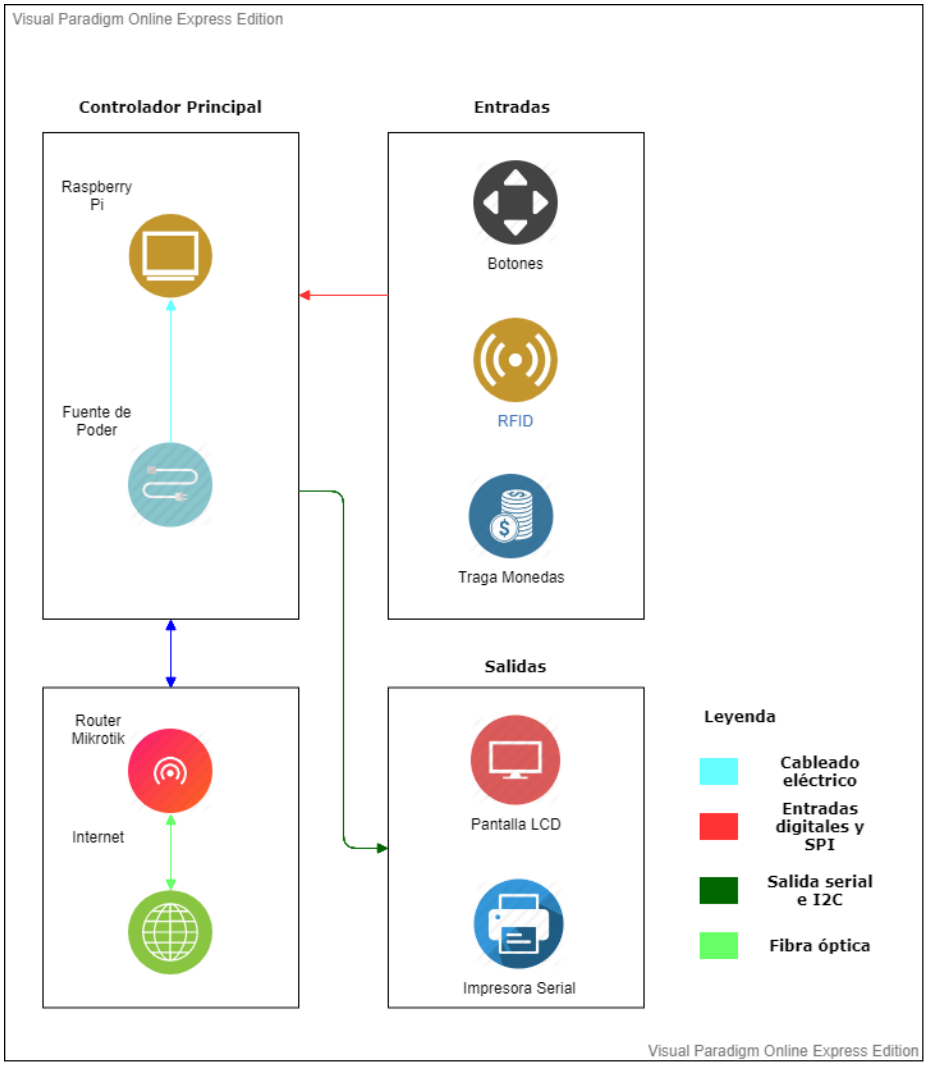


Figura 5: Estructura de la plataforma de automatización implementada.

CAPÍTULO 8

Componente de red - MikroTik RouterBOARD 951G 2HnD

El enrutador o *router* MIkroTik RouterBOARD 951G 2HnD es una equipo destinado a ambientes *Small-Office/Home-Office* o SOHO, debido a su forma compacta, pero con un alto rendimiento. Este dispositivo posee un procesador que trabaja a una frecuencia de unos 600 MHz y cuenta con 100 MiB de memoria RAM. Además, cuenta con 5 puertos Gigabit y con un punto de acceso 2.4Ghz 802.11b/g/n. En la Figura 6 se observa el router utilizado en la implementación.

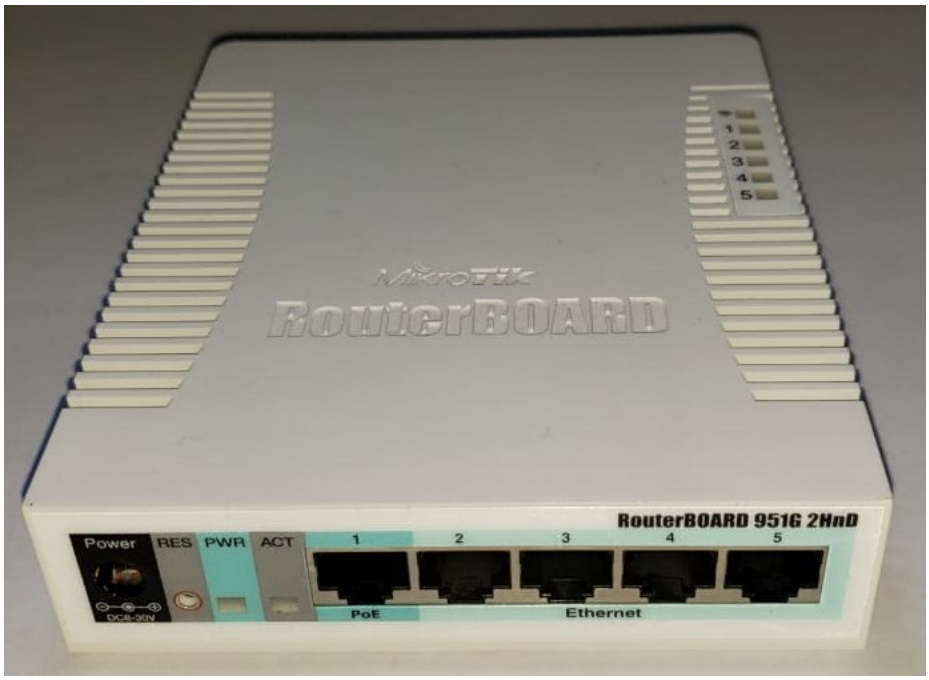


Figura 6: Router MikroTik RB951G 2HnD

8.1. Configuración router MikroTik RB951G

Al momento de adquirir el equipo este posee una configuración inicial que le permite funcionar como un *router*. No obstante, es posible que estos no cuenten con alguna configuración inicial que permita la navegación a internet, en ambos casos se utiliza la aplicación WinBox para la configuración, esta puede ser descargada por medio del sitio oficial de MikroTik [7]. WinBox es utilizada para configurar, administrar y monitorear todos los equipos MikroTik.

Una vez descargada la aplicación se debe conectar la computadora a uno de los puertos del router MikroTik, usualmente se omite el primer puerto ya que este estará conectado la red WAN. Luego, en WinBox se observa una pestaña con el nombre de *Neighbors*, esta pestaña muestra todo equipo MikroTik que se encuentre conectado a la computadora o dentro de la red LAN. En la Figura 7 se observa que el equipo no posee ninguna dirección IP, esto se debe a que no se encuentra configurado algún servidor DHCP (por sus siglas en inglés, *Dinamic Host Configuration Protocol*) ni posee ninguna dirección IP estática en los puertos. Es de importancia mencionar que al momento de configurar el equipo se está realizando por medio de una conexión en la capa 2 del modelo ISO / OSI, al momento de conectarse utilizando la dirección IP del equipo se utiliza la capa 3 de este modelo, se recalca esto debido a que es posible que al realizar configuraciones se pierda conexión con el equipo que apliquen a este nivel. Asimismo, al trabajar con la capa 2 se requiere una conexión directa para poder configurar el equipo, mientras que utilizando la capa 3 es posible configurar equipos dentro de la misma red o de una forma remota.

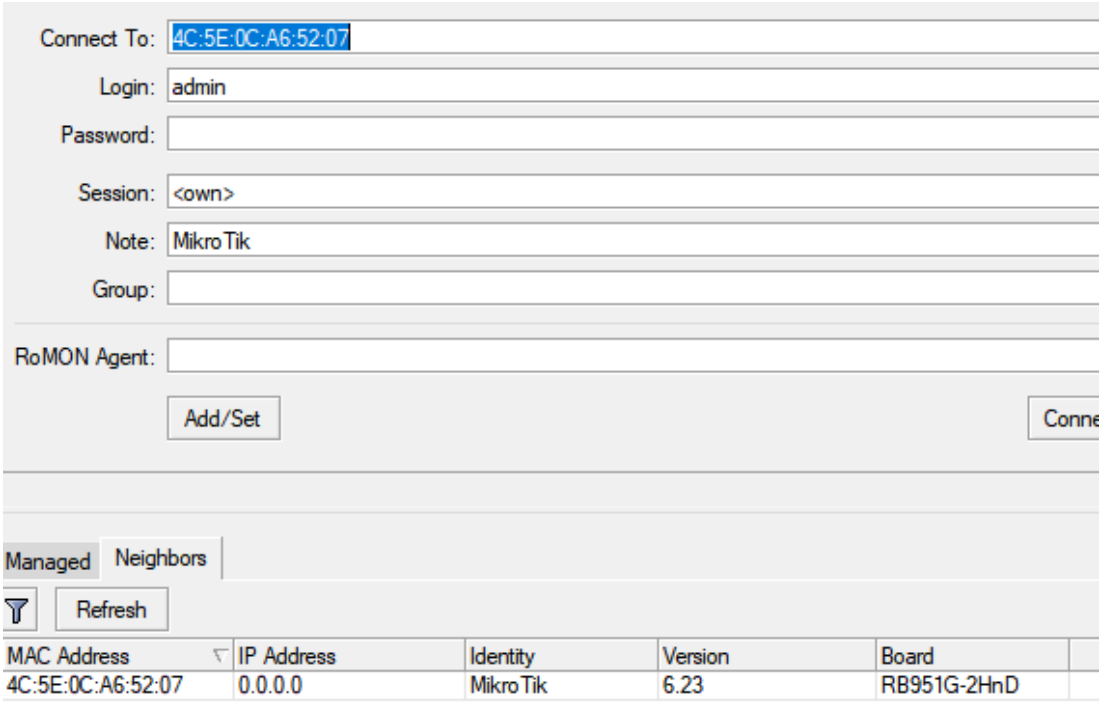


Figura 7: Ventana inicial de la aplicación WinBox

Una vez dentro de la aplicación se observa a un costado de están todas las pestañas que corresponden a un menú de configuración, al presionar alguna de ellas se abrirá una ventana dentro del mismo ambiente. Para la configuración se hizo uso del asistente *Quick Set* que provee la aplicación WinBox, este asistente permite una configuración sencilla y eficiente. En la Figura 8 se muestra la ventana del asistente.

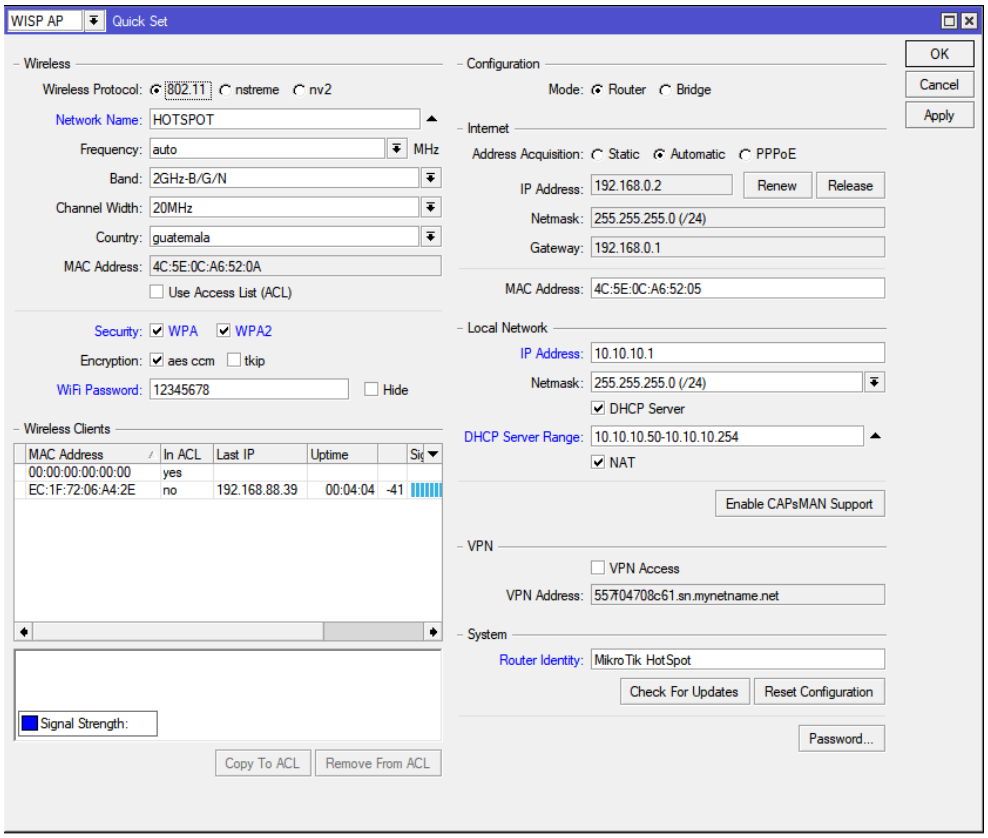


Figura 8: Asistente de configuración para el router MikroTik

El proceso de configuración es un proceso sencillo que no presenta mayor reto, sin embargo, cabe mencionar que en el apartado de "Internet" que se encuentra dentro de la ventana es la configuración de cómo estará conectado el router a Internet -o la red WAN a la cual se estará conectando-. La opción "Automatic" crea un cliente DHCP en el puerto ether1 del router MikroTik, de esta forma el proveedor de servicios o ISP (por sus siglas en inglés, *Internet Service Provider*) otorga una dirección IP a dicho puerto.

8.2. Configuración router MikroTik como hotspot

El router MikroTik tiene la capacidad de funcionar como un *hotspot* que permite la autenticación de usuarios que concede la navegación a internet. Para configurar este modo se debe ir al menu *Ip* > *Hotspot*, una vez en esta ventana se presiona el botón que dice "Hotspot Setup", este es un asistente que permite una configuración sencilla de un *hotspot*. En la Figura 9 se puede observar el botón que abre el asiste de configuración.

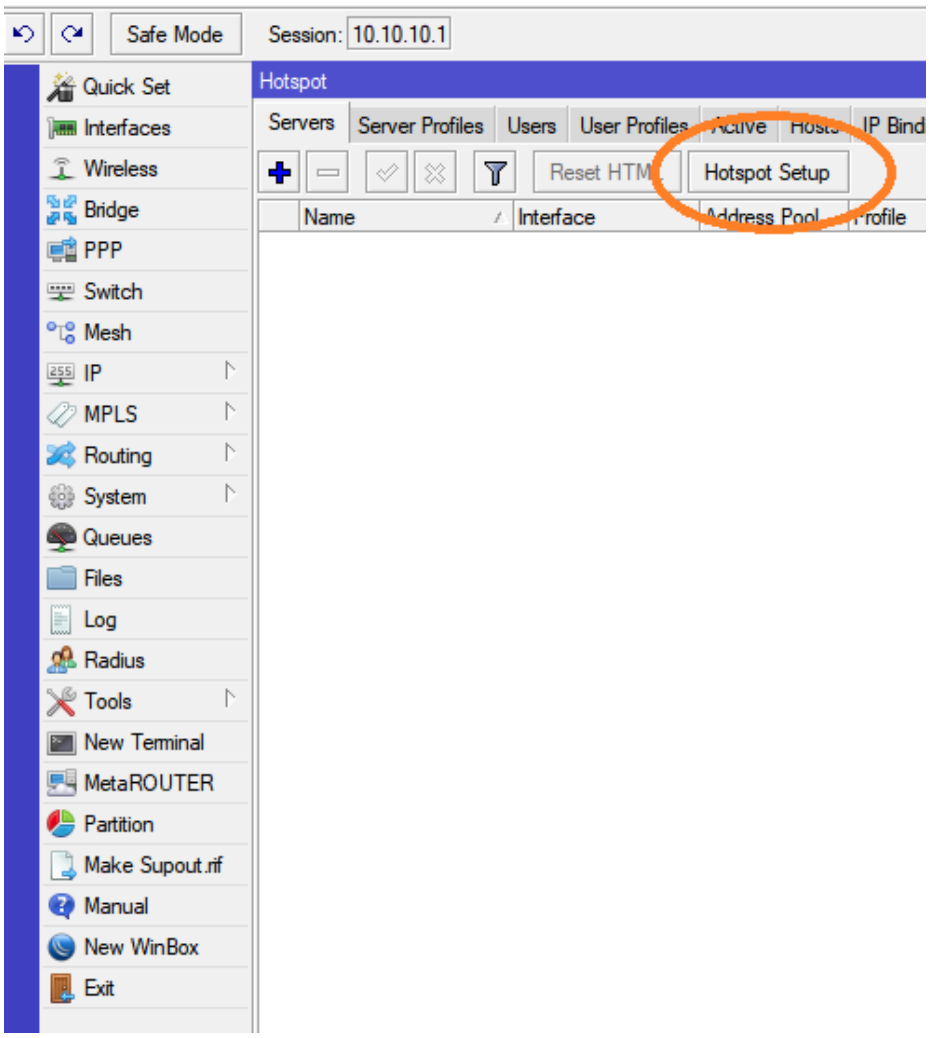


Figura 9: Asistente de configuración para el *hotspot*

La configuración inicial para el *hotspot* es la siguiente:

1. **Selección de interfaz para montar *hotspot***
   - Se seleccionó la interfaz inalámbrica *wlan1*
2. **Dirección IP del servidor *hotspot***
   - Se utilizó la dirección IP 10.5.50.1 con máscara de red 255.255.255.0 (/24)
3. **Definición de dirección IP para asignar**
   - El rango de dirección IP que se asignará a los usuarios es de la 10.5.50.25 a la 10.5.50.254
4. **Certificado SSL**
   - Ninguno

5. **Servidor SMTP**

   - Predeterminado (0.0.0.0)

6. **Definición de servidor DNS**

   - Se configuraron dos servidores, siendo estos el 8.8.8.8 y 8.8.4.4

7. **Definición del dominio de acceso, este será el nombre del dominio que contendrá al portal cautivo donde se ingresan las credenciales**

   - tesishotspot.com

8. **Creación de la primer cuenta de usuario para el *hotspot*, siendo la del administrador.**

Una versión visual y detallada de la configuración del *hotspot* se encuentra en [13]. Es de importancia mencionar que debido a la configuración generada por el asistente *Quick Set* existe una incompatibilidad que no permite el funcionamiento del *hotspot*, esto se puede observar en el servidor generado se encuentra de un color rojo indicando un error. Esto se debe a que se ha montado un servidor en la interfaz *wlan1*, no obstante, esta se encuentra en modo puente o *bridge* con las interfaces *ether2-5*, para corregir esto se debe ir a la ventana *Bridge* y en la pestaña *Ports* se debe deshabilitar o remover el puente con la interfaz. En la Figura 10 se puede observar dicha ventana.

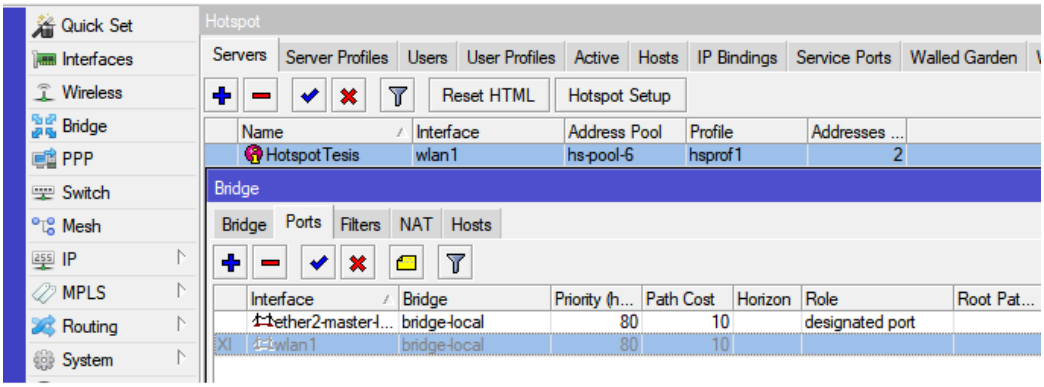


Figura 10: Corrección de incompatibilidad entre servidor *hotspot* y puente con la red inalámbrica

Una vez configurado el router MikroTik como *hotspot* es necesario configurar los perfiles que utilizaran los usuarios generados. Los perfiles son reglas de comportamiento de los usuarios, para configurar los perfiles se debe ir a la siguiente pestaña *Hotspot* > *User Profiles* > *Símbolo "+"*, luego se abre la ventana que se muestra en la Figura 11, la cual es la configuración utilizada para el perfil de navegación de 3Mbps.

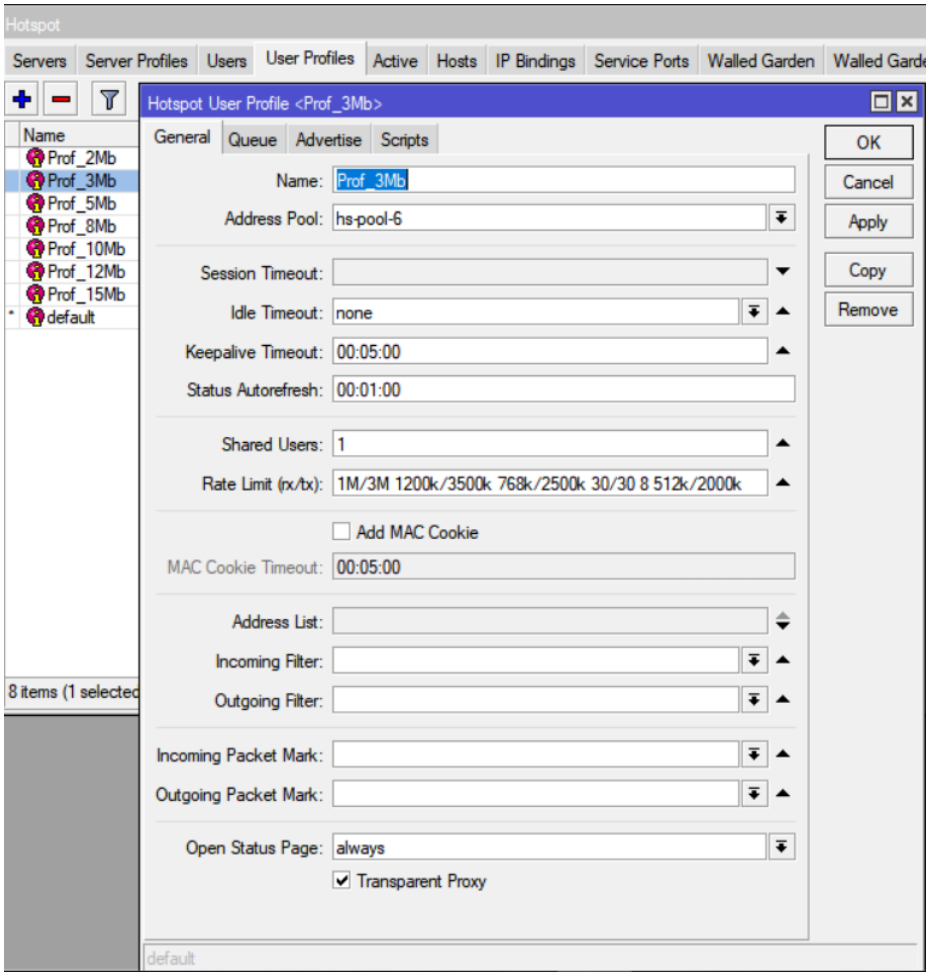


Figura 11: Ventana de creación de perfiles para los usuarios.

Donde:

- *Name*: Nombre con el que se identificará el perfil, este se utilizará al momento de crear usuarios desde la Raspberry Pi.
- *Address Pool*: Lote de direcciones IP que utilizará el perfil, se utilizó el lote generado en la configuración del *hotspot*.
- *Session Timeout*: Este parámetro es para definir el límite de tiempo del usuario, sin embargo, este se deja en blanco ya que el límite de tiempo varía dependiendo del usuario y se coloca al momento de generar el usuario desde la Raspberry Pi.
- *Keepalive Timeout*: Este es el tiempo que mantiene la sesión del usuario activa al momento que este se desconecte de la red, una vez consumido el usuario deberá iniciar sesión nuevamente.
- *Shared Users:* Este parámetro define la cantidad de dispositivos que se pueden conectar por usuario y contraseña generado.

- *Rate Limit (rx/tx)*: Es la tasa límite de transferencia de datos para el *upstream* y *downstream* del usuario. Los parámetros posteriores al primer grupo son la configuración de la ráfaga de la tasa de transferencia. La ráfaga en una conexión es un período de tiempo definido en el cual las tasas máximas de transferencia son aumentadas con el objetivo de proveer una mejor experiencia en la navegación web. Asimismo, el penúltimo parámetro de los datos establece la prioridad que poseerá el perfil, mejorando lo que es la calidad de servicio.

8.3. Eliminación de usuarios del hotspot

Una vez completada la configuración del *hotspot* se procedió a configurar la eliminación de los usuarios, esto se realizó utilizando dos funciones que provee la plataforma RouterOS. La primera función, es la habilidad de correr *scripts*, programas escritos con el objetivo de automatizar tareas. La segunda función, es el *scheduler*, esta función permite programar la fecha y hora de ejecución de los *scripts* de acuerdo a las necesidades del administrador de red.

8.3.1. Eliminación de usuarios utilizando un script

El *script* es el código encargado de eliminar todos los usuarios que existen dentro del servidor *hotspot*. Al momento de ejecutar tal código este recorre la lista de usuarios y lo elimina uno por uno, hasta llegar al final de la lista. Para configurar los *scripts* se debe ir a la pestaña *System* > *Script* > *Símbolo "+"* , luego se abre la ventana que se muestra en la Figura 12. Una vez eliminados los usuarios los clientes ya no podrán autenticarse e iniciar sesión y la única forma de obtener navegación nuevamente es adquiriendo el servicio otra vez.

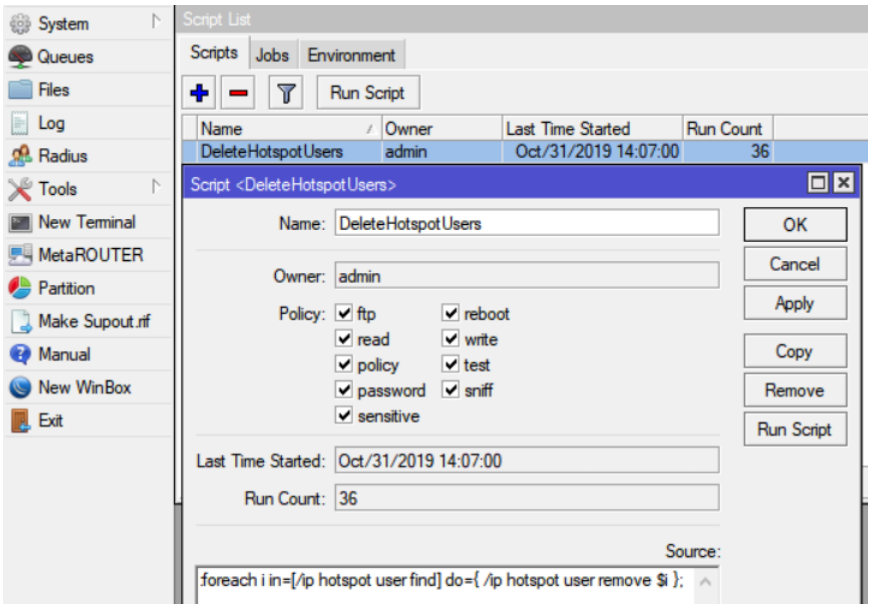


Figura 12: Ventana de creación de *scripts*.

Asimismo, es posible añadir el *script* utilizando la interfaz de línea de comandos o CLI (por sus siglas en inglés, *Command Line Interface*) con el siguiente comando:

```
/system script add name=DeleteHotspotUsers policy=ftp,read,write,policy,
    ↪ test,winbox,api source=":foreach␣i␣in=[/ip␣hotspot␣user␣find]␣do={␣/
    ↪ ip␣hotspot␣user␣remove␣$i␣};"
```

8.3.2. Programación de la ejecución de los scripts utilizando scheduler

El *scheduler* es una función del RouterOS que permite ejecutar los *scripts* de forma automática y con la capacidad de definir el día, hora e intervalo de ejecución. Para configurarlo se debe ir a la pestaña *System* > *Scheduler* > *Símbolo "+"* , luego se abre la ventana que se muestra en la Figura 13. La plataforma se configuró para que todos los días a las 3 horas se ejecutara el *script* de eliminación de usuarios.

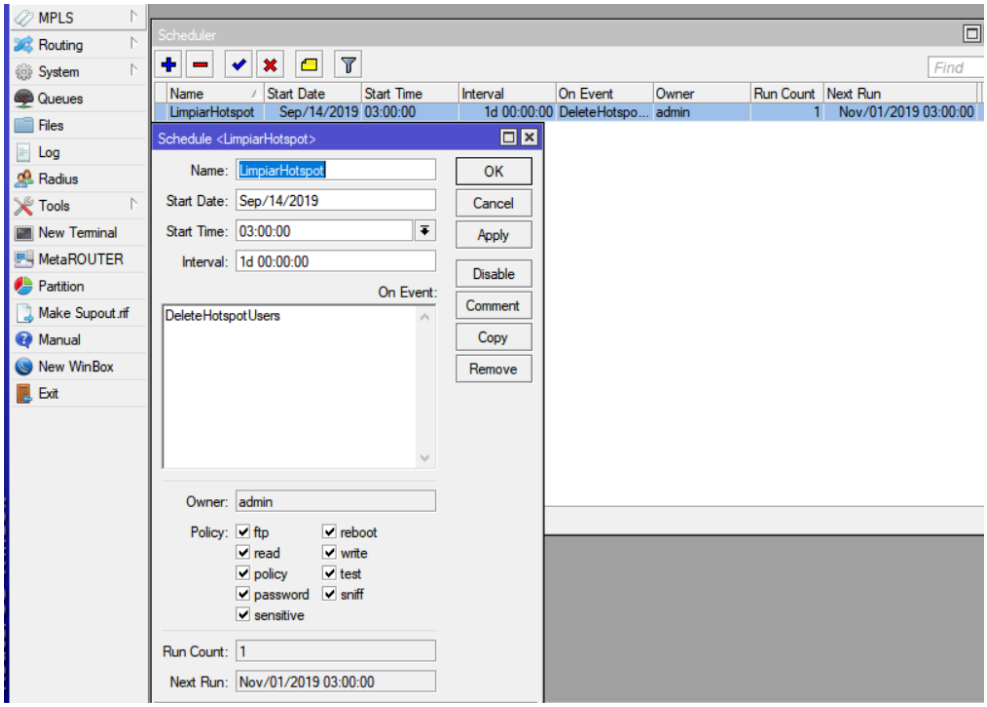


Figura 13: Ventana de programación del *scheduler*

Asimismo, es posible configurar el *scheduler* utilizando la interfaz de línea de comandos o CLI (por sus siglas en inglés, *Command Line Interface*) con el siguiente comando:

```
/system scheduler add interval=1d name=LimpiarHotspot on-event=
    ↪ DeleteHotspotUsers policy=ftp,reboot,read,write,policy,test,password
    ↪ ,sniff,sensitive start-date=sep/14/2019 start-time=03:00:00
```

CAPÍTULO 9

Componentes del controlador principal

9.1. Sistema operativo Raspbian Buster

En la implementación se utilizó el sistema operativo Raspbian Buster, este sistema operativo está basado en Debian y fue optimizado especialmente para el *hardware* de la plataforma Raspberry Pi. Cabe mencionar que existen otros sistemas operativos disponibles para esta plataforma, sin embargo, no todos proveen soporte para las librerías de los componentes utilizados o fueron optimizados para ciertas aplicaciones en específico. Existen dos maneras de instalar el sistema operativo en la tarjeta de memoria microSD, la primera es montar la imagen del sistema operativo (se debe descargar el archivo con extensión ".iso") y luego se o monta la imagen utilizando el programa Win32DiskImager. La segunda manera es utilizando la versión NOOBs que se encuentra en el sitio de descarga, en este caso solo se deben extraer el contenido del archivo comprimido (archivo con extensión ".zip") dentro de la tarjeta de memoria microSD y colocarla dentro de la RaspberryPi, una vez encendida se deben seguir los pasos en pantalla para la instalación del sistema operativo.

Una vez instalado el sistema operativo es recomendable buscar actualizaciones del sistema operativo y de los paquetes instalados, esto se realiza escribiendo los siguientes comandos dentro de la consola con permisos de administrador.

Commandos para actualizar el sistema operativo y paquetes y/o librerías instaladas por el usuario :

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
```

9.2. Lector RFID RC522

En la actualidad los lectores de identificación por radiofrecuencias o RFID (por sus siglas en inglés, *Radio Frequency Identification*) han ganado popularidad debido a la simplicidad con la que se puede autenticar un usuario o dispositivo. Por lo tanto en la implementación final se optó por colocar un lector RFID en el sistema para que ciertos usuarios puedan gozar del servicio de *hotspot* sin necesidad de monedas, en calidad de servicio agregado.

El lector RFID RC522 está basado en el circuito integrado MFRC52202HN1 de NXP, este está diseñado para comunicación sin contacto a una frecuencia de 13.56Mhz y trabaja bajo el estándar ISO/IEC 14443 A/MIFARE y NTAG. Asimismo, el IC MFRC522 tiene capacidad de escribir y leer las tarjetas RFID pasivas. A pesar de que el IC es capaz de utilizar las interfaces SPI, Serial UART e I2C la placa que posee la antena fue diseñada para funcionar con la interfaz SPI dada la configuración de los pines del circuito. El módulo RFID se encuentra en la Figura 14. La conexión eléctrica utilizada es la que se encuentra en el tutorial de la página web *PiMyLifeUp*, que a su vez se encuentra en [14]. Además, mayor información del circuito integrado se puede encontrar en [15].

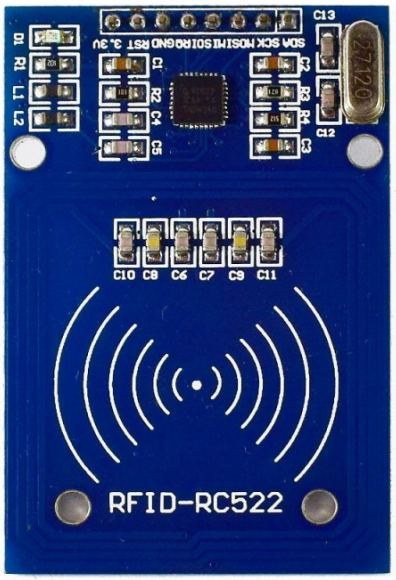


Figura 14: Módulo RFID RC522 utilizado en la implementación de la plataforma de automatización

Antes de iniciar la lectura y escritura de tarjetas RFID es necesario preparar el sistema operativo Raspbian instalando librerías necesarias de Python y del módulo RFID RC522. Primero se instalan (o si es el caso se actualizan con el mismo comando) las librerías de Python que son indispensables para la instalación de otros módulos que habilitan las interfaces de comunicación I2C, SPI y los puertos de propósito general o GPIO (por sus siglas en inglés, *General Purpose Input/Output*). Una vez instaladas estas librerías es necesario habilitar la interfaz de comunicación SPI dentro de la configuración de la Raspberry Pi, luego se reinicia y se procede a instalar las librerías del módulo RFID.

Comandos utilizados para instalar y actualizar librerías necesarias de Python:

```
pi@raspberrypi ~ $ sudo apt-get install build-essential git python3-dev
    ↪ python3-pip python3-smbus
pi@raspberrypi ~ $ sudo pip3 install RPi.GPIO
```

Comandos utilizados para instalar las librerías del módulo RFID RC522:

```
pi@raspberrypi ~ $ sudo pip3 install spidev
pi@raspberrypi ~ $ sudo pip3 install mfrc522
```

9.3. Pantalla LCD 16x2 I2C

La pantalla LCD es el componente que permite establecer una comunicación visual entre la plataforma de automatización y el usuario. En el proyecto se utilizó una pantalla de cristal líquido o LCD (por sus siglas en inglés, *Liquid Crystal Display*) de 16 columnas por dos renglones, con la capacidad de mostrar hasta 32 caracteres de forma simultánea, sin embargo, solamente se puede imprimir un carácter cada cierto período de tiempo lo cual se debe considerar para la sincronía del programa final. Asimismo, esta pantalla cuenta con un controlador que funciona por medio de comunicación I2C, este controlador reduce la cantidad de pines que se deben utilizar ya que reduce de una conexión en paralelo a una conexión serial utilizando el bus de datos I2C. En la Figura 15 se encuentra la pantalla utilizada en la implementación final. Para la conexión eléctrica se siguieron los pasos del tutorial de *The Raspberry Pi Guy* que se encuentra en [16]. Mayor información acerca de los tiempos requeridos e instrucciones de la pantalla LCD se pueden encontrar en [17].

Figura 15: Pantalla LCD I2C utilizada en la implementación de la plataforma de automatización

Una vez realizada la conexión eléctrica se procedió a habilitar la interfaz de comunicación I2C de la Raspberry Pi y se reinició. Luego se instaló la librería encargada de manejar la comunicación entre la Raspberry Pi y la pantalla LCD utilizando los siguientes comandos.

Comandos utilizados para descargar e instalar el módulo LCD 16X2 I2C:

```
pi@raspberrypi ~ $ git clone https://github.com/the-raspberry-pi-guy/lcd
pi@raspberrypi ~ $ cd lcd
pi@raspberrypi ~ $ sudo sh install.sh
```

Es de importancia mencionar que los controladores I2C poseen una dirección hexadecimal que son sus identificadores y en ocasiones el driver de la pantalla LCD puede no estar configurado con la dirección correcta, para solucionar este problema se debe escribir el siguiente comando en consola y una vez mostrada la dirección I2C se debe corregir la dirección en el archivo "lcddriver.py".

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
```

9.4. Impresora térmica

La impresora térmica es un tipo especial de impresora que no requiere de tinta o tóner para imprimir, si no, utiliza un cabezal térmico en conjunto con papel térmico -que es reactivo al calor-. Entonces, una zona papel que sea expuesta a una cantidad de suficiente calor se tornará de color negro dando el efecto de impresión. La impresora térmica es la encargada de proveer un *ticket* con los datos generados en caso de que el usuario no los haya podido observar e ingresar en su dispositivo en el tiempo establecido. La impresora utilizada es la que se encuentra en el enlace [18] y se observa en la Figura 16.

Figura 16: Impresora térmica que utiliza interfaz de comunicación Serial UART

En la actualidad se considera como "sencillo" el instalar impresoras en sistemas operativos no basados en Linux, ya que es cuestión de utilizar el instalador correspondiente. Sin embargo, en sistemas operativos de tipo Unix el proceso es más complejo, ya que por

defecto no viene instalado algún servidor de impresiones. El servidor de impresiones es el encargado de manejar los datos que se desean imprimir, llevar la cola de impresiones, utilizar los controladores o *drivers* correspondientes que realizan la transmisión y rasterización de los objetos a imprimir. La rasterización es el proceso de tomar un objeto gráfico vectorial y convertirlo a píxeles o puntos que la impresora puede imprimir. En este proyecto se utilizó el sistema de impresión común de Unix o CUPS (por sus siglas en inglés, *Common Unix Printing System*).

Inicialmente se configuró la impresora térmica utilizando la guía elaborada por Phillip Burgess para la empresa Adafruit [19], sin embargo, al momento de instalar los drivers el sistema operativo mostró diversas alarmas y advertencias sobre la obsolecencia que presentaban los *drivers*, ya que estos fueron modificados por última vez en el año 2015. Dado esto, se utilizó la guía elaborada por el usuario *Scruss* [20] siendo una versión más actualizada con la cual se obtuvieron mejores resultados de impresión. Es de importancia mencionar que la impresora utilizada está basada en la impresora térmica ZJ-58, por lo tanto, los drivers utilizados para la impresora fueron los que se encuentran en el repositorio de GitHub del usuario Aleksey N. Vinogradov, los cuales se encuentran en [21]. Una vez instalados los drivers se debe realizar la configuración de la impresora en CUPS tal como lo dice en la guía [20].

En el siguiente bloque de código se encuentran los comandos utilizados para descargar e instalar los *drivers* de la impresora térmica basada en el modelo ZJ-58:

```
pi@raspberrypi ~ $ sudo apt install build-essential cmake libcups2-dev
    ↪ libcupsimage2-dev
pi@raspberrypi ~ $ mkdir build && cd build && cmake /pi/home/build
pi@raspberrypi ~ $ cmake --build .
pi@raspberrypi ~ $ sudo make install
```

Cabe recalcar que por las constantes actualizaciones que existen en el sistema operativo Raspbian, dichos comandos pueden ser obsoletos en dado momento, debido a esto se recomienda verificar guías similares para la instalación de la impresora térmica.

9.5. Aceptador de monedas

El aceptador de monedas es el elemento que permite al usuario ingresar los créditos necesarios para adquirir el servicio de *hotspot*. Es módulo es encargado de recibir, validar y aceptar las monedas previamente configuradas. Inicialmente, el módulo utilizado era un aceptador de monedas de tres tipos de monedas, sin embargo, se observó que el modelo puede validar hasta seis diferentes tipos de monedas. Entonces, se configuraron cuatro tipos de monedas de dos denominaciones diferentes, siendo las primeras dos monedas las de Q0.50 en sus dos presentaciones disponibles, según el Banco de Guatemala. Las otras dos monedas configuradas fueron las de Q1.00, nuevamente en sus dos presentaciones que se encuentran circulando actualmente. Mayor información de las monedas que se encuentran en circulación en Guatemala se encuentra en [22]. En la Figura 17 se encuentra el aceptador de monedas utilizado y este se encuentra en [23].

Figura 17: Aceptador de monedas utilizado en la implementación de la plataforma de automatización

Se utilizaron las guías y vídeos encontrados en [24] y [25] para la configuración del aceptador de monedas. Los tutoriales dan una explicación detallada del proceso correcto ya que la hoja de datos incluida con el producto provee instrucciones pobres. Para la configuración del aceptador de monedas se utilizaron los parámetros que se encuentran el Cuadro 1.

Parámetro	Significado	Valor utilizado
E	Cantidad de monedas a configurar. Se pueden configurar hasta 6 tipos de monedas en este modelo.	4 Tipos de monedas, luego en los siguientes parámetros aparecerá un número indicando el tipo de moneda configurar
Hx	Cantidad de muestra de la moneda x, a mayor sea el valor mejor será la precisión del aceptador de monedas ante monedas con desgaste y/o sucias.	20 Muestras para cada tipo de moneda
Px	Cantidad de pulsos que genera el dispositivo por la moneda x.	10 Pulsos para las monedas de Q0.50 20 Pulsos para las monedas de Q1.00
Fx	Precisión del sistema para reconocer la moneda x.	7 en nivel de precisión, según la hoja de datos con este valor tiene una precisión del 95 %

Cuadro 1: Parámetros configurados en el aceptador de monedas.

El aceptador de monedas utilizado es el CH926, el cual está basado en la familia CH92x y JY92x, siendo x la cantidad de diferentes monedas que puede validar. Estos aceptadores de monedas cuentan con cuatro pines, dos de ellos están destinados para la alimentación del dispositivo, uno utilizado para realizar pruebas de funcionamiento, y el último, está destinado como la salida del tren de pulsos generada al momento de aceptar una moneda.

Asimismo, estos cuentan con dos *switches* que configuran el modo de funcionamiento de la salida, siendo una opción normalmente abierto o NO (por sus siglas en inglés, *Normally Open*) o normalmente cerrado o NC (por sus siglas en inglés, *Normally Closed)*. Se utilizó en la posición "*NO*" dado que se utilizará en lógica *Pull-Up*, entonces, en la programación se desea detectar los flancos negativos de la salida del módulo. El segundo *switch* controla el ancho del pulso que envía la salida, posee tres opciones siendo "Rápido, Medio y Lento" o "*Fast, Medium y Slow*" en los cuales cada pulso generado es de 20 ms, 50 ms y 70 ms, respectivamente. Adicionalmente, cada pulso viene seguido de una pausa de 100 ms. El *switch* se colocó en la posición "*Fast*" ya que la Raspberry Pi es capaz de detectar pulsos de ese ancho debido a su alta frecuencia de muestreo - de unos 10 MHz-.

Una vez configurado correctamente el aceptador de monedas de acuerdo a las necesidades del proyecto se procedió a probar y observar su funcionamiento. En la Figura 18 se encuentra el primer pulso generado para las monedas de Q0.50, en esta medición se observa cómo el pulso generado es de 19.8 ms. En la Figura 19 se encuentra el primer pulso generado para las monedas de Q1.00, en esta medición se observa cómo el pulso generado es de 20.2 ms.

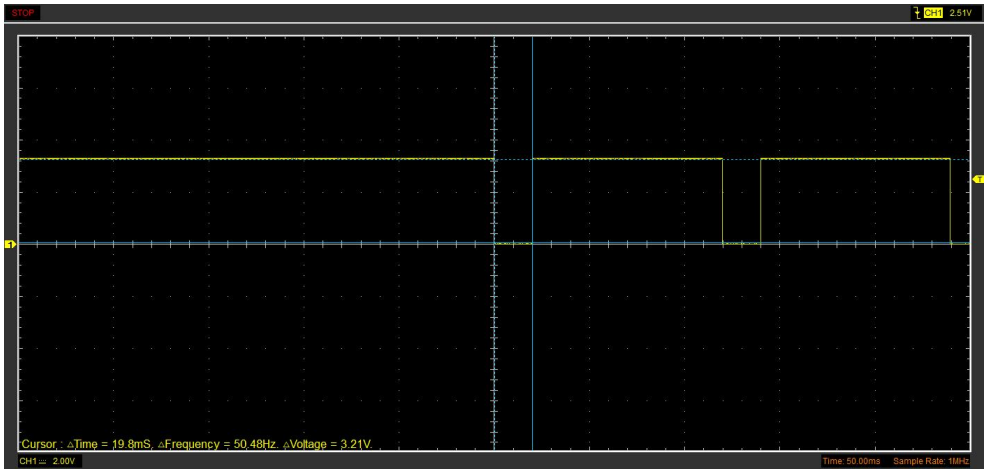


Figura 18: Pulso generado para la moneda de Q0.50

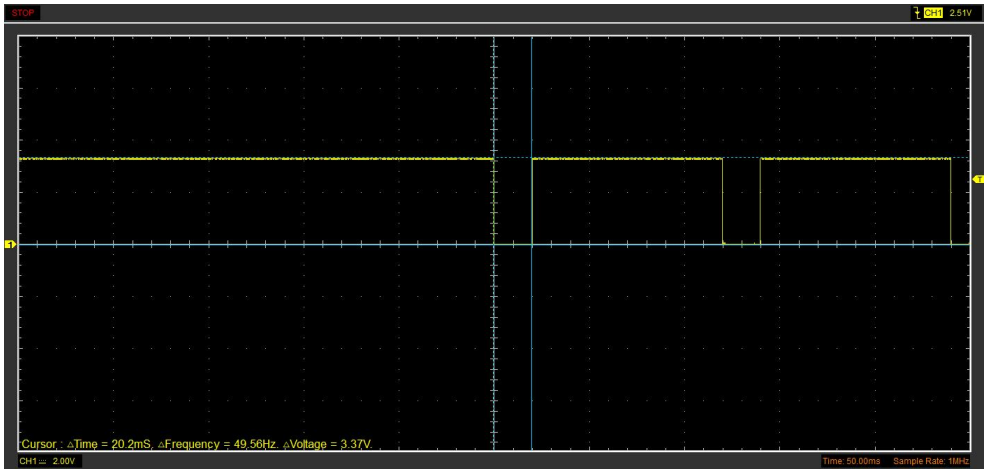


Figura 19: Pulso generado para la moneda de Q1.00

En las Figuras 20 y 21 se encuentran los trenes de pulsos generados para las monedas de Q0.50 y Q1.00. Se puede observar que tal como se describió en la configuración del aceptador de monedas la cantidad de pulsos generados corresponde a los parámetros utilizados, sin embargo, es de importancia recalcar que a mayor cantidad de pulsos mayor será el tiempo que le tome al controlador en reconocer el tipo de moneda ingresada debido a tiempo de pausa entre los pulso, siendo de 100 ms por pulso generado. En el caso de la moneda de Q0.50 el tren de pulsos tiene un período de 1.10 s y 2.30 s para la moneda de Q1.00.

Asimismo, este aceptador de monedas tiene la desventaja que al ingresar dos monedas de forma sucesiva no hay manera de diferenciar el tipo de monedas ya que no existe un tiempo que permita distinguir los trenes de pulsos, siendo un mayor problema al momento de configurar más tipos de monedas. Una técnica para suprimir este comportamiento es asignar un valor a cada pulso utilizando un divisor común entre las monedas, de esta manera los créditos se acumulan en base a los pulsos recibidos. En el caso del proyecto se asumió como si cada pulso tuviese un valor de Q0.05.

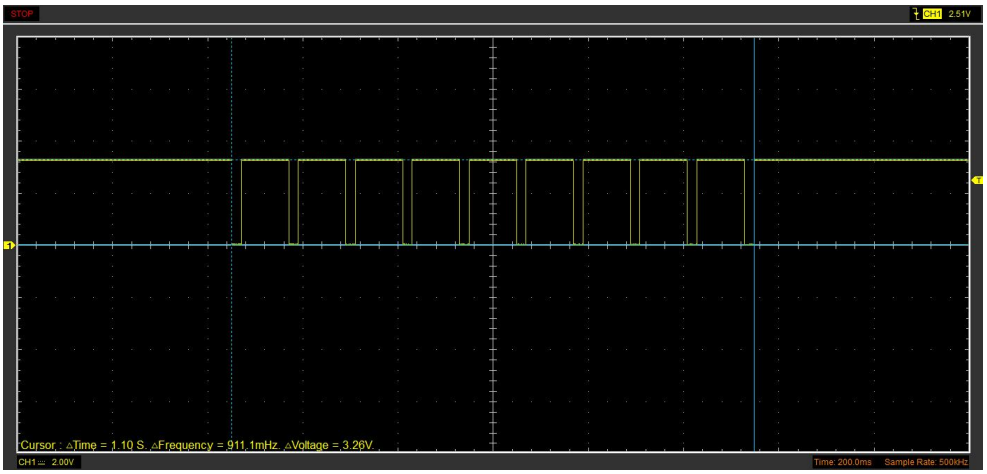


Figura 20: Tren de pulsos generados para la moneda de Q0.50

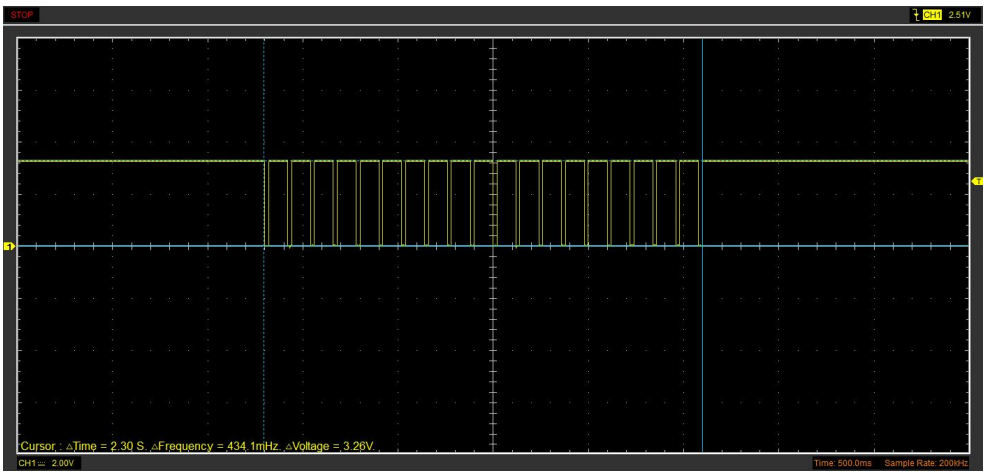


Figura 21: Tren de pulsos para la moneda de Q1.00

CAPÍTULO 10

Diseño e implementación de la plataforma de automatización

Uno de los principales retos del proyecto era la integración de los diferentes módulos de manera que estos puedan trabajar correctamente en conjunto y de forma simultánea. Por lo tanto, se diseñó un circuito que cumpliese con los requerimientos necesarios para el funcionamiento.

10.1. Diseño del circuito implementado

El circuito y la placa de circuito impreso o PCB (por sus siglas en inglés, *Printed Circuit Board*) fueron diseñados con el programa Altium Designer. El diseño del circuito fue basado en los componentes que se encontraban en el mercado local de componentes electrónicos, siendo esto una limitación para el diseño final. Además, se optó por diseñar dos circuitos, el primero de ellos siendo la placa principal encargada de interconectar la Raspberry Pi con todos los módulos y con las diferentes fuentes de alimentación. El segundo circuito está encargado de interconectar todos los botones y poseer solo los pines de salida que estos van conectados a la Raspberry Pi, esto se realizó de esta manera ya que este circuito puede ser montado directamente en la superficie principal donde yacen las otras entradas y salidas que interactúan con el usuario.

En la Figura 22 se encuentra el esquemático del circuito principal, una versión ampliada se encuentra en el área de anexos 37. En la Figura 23 se encuentra el esquemático del circuito secundario, una versión ampliada se encuentra en el área de anexos 36.

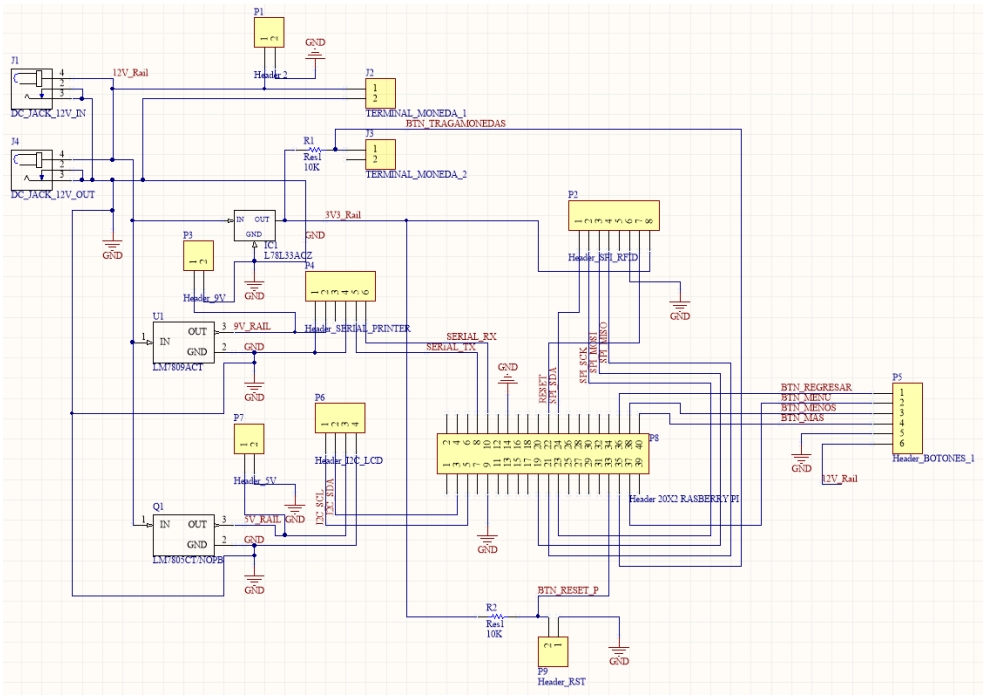


Figura 22: Esquemático del circuito principal

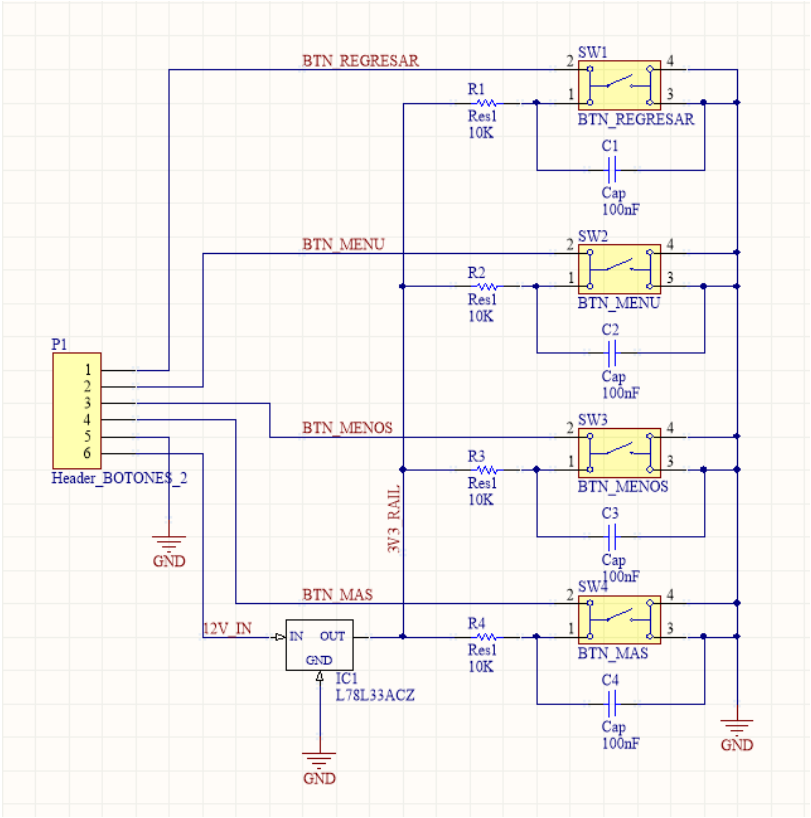


Figura 23: Esquemático del circuito secundario

Al momento de diseñar el circuito principal se tomaron ciertos criterios de diseño, siendo el primero de ellos buscar la manera de poseer la menor cantidad de fuentes de alimentación dentro de la plataforma final. Sin embargo, para lograr esto se requiere de una fuente de alimentación de 12 V capaz de entregar hasta 2 A dado que debe alimentar ambos componentes y la corriente pico del consumo combinado llega muy cercano a 1.3 A. Se menciona que es corriente pico dado solo sucede al momento de imprimir los *tickets* generados. Entonces, se colocaron dos conectores DC de forma que sea un tipo "*passthrough*" donde pueda ser una entrada para la alimentación y una salida para alimentar el componente de red.

Otro criterio considerado fue que el cableado entre los componentes no complique su implementación o instalación, debido a esto hubo una incursión de un *header* de 20x2 pines en el circuito diseñado. Siendo el *header* de las mismas dimensiones que el puerto GPIO, de esta manera se podría utilizar un cable cinta o similar para conectar el circuito principal con la Raspberry Pi, ya que se mantuvo el orden y conexión de todos los pines GPIO. Además, se colocaron pines extra como posibles puntos de prueba o de alimentación, ya que se encuentra un par de pines para cada riel de voltaje utilizado en el circuito (12 V, 9 V, 5 V y 3.3 V).

Una vez diseñado el circuito se procedió a diseñar la placa de circuito impreso o PCB, nuevamente este se diseñó en base a ciertos criterios que se debían considerar antes de su producción. El primero de ellos era la disponibilidad de los componentes que se encontraban en el mercado local, el segundo, el *footprint* de los componentes disponibles. El *footprint* es la huella de un componente y depende del empaquetado del mismo, este afecta el tipo posicionamiento que va a tener dentro de la placa, las dimensiones que posee físicamente y si este interfiere con otros componentes en el circuito, el tipo de conectores a utilizar, y por último, el tipo de fabricación que debe poseer la placa para garantizar el funcionamiento del circuito.

En el proyecto todos los componentes son *through hole*, lo cual significa que todos los componentes se colocan en el lado superior del PCB pero son soldados en el lado inferior del mismo. Es de importancia recalcar que al momento de realizar le diseño del PCB es necesario corroborar los *footprints* de los componentes virtuales con los de su contra partes reales, ya que en ocasiones estos pueden ser incorrectos o pueden tener una enumeración errónea en los pines, resultando en PCBs defectuosos en el mejor de los casos, y en el peor de los casos, puede resultar en equipos dañados y heridas personales debido a corto circuitos. En las Figuras 24 y 25 se encuentran los PCBs diseñados. Es de importancia mencionar que en la Figura 24 se notan dos coloraciones distintas para los rastros o *traces*, porque es un PCB de doble lado, es decir posee material conductor en ambos lados.

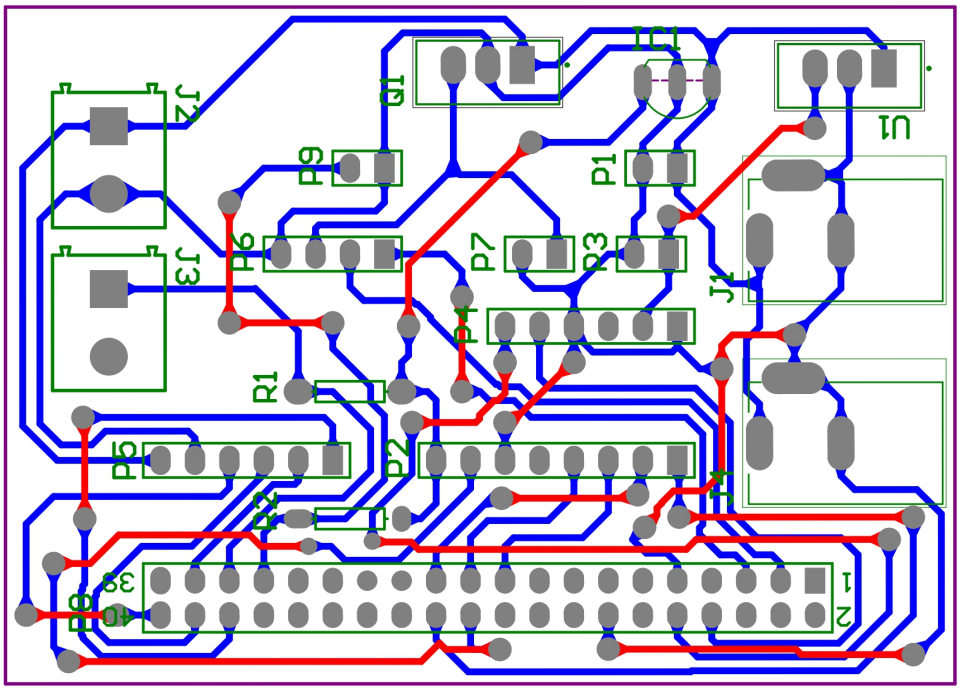


Figura 24: PCB diseñado para el circuito primario con dimensiones de 70 mmx 50 mm

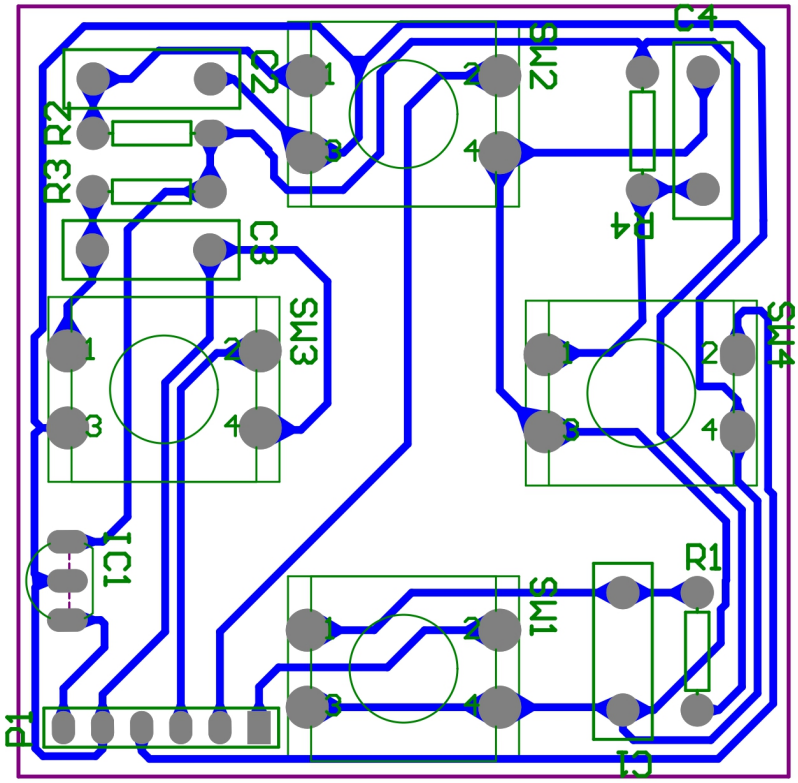


Figura 25: PCB diseñado para el circuito secundario con dimensiones de 50 mmx 50 mm

Otro criterio de diseño a considerar es la potencia que deben disipar los rastros del PCB o *traces* del PCB. Para esto se utilizó la herramienta que provee Advanced Circuits en su página web [26], esta se basa en el Estándar Genérico para Diseño de Placas de Circuitos Impresos IPC-2221A, el cual se encuentra en [27]. La ecuación del requerimiento mínimo del conductor en base a la corriente de flujo y el máximo cambio de temperatura del conductor es la siguiente [28]:

$$I = k\Delta T^{(0.44)} A^{(0.725)} \tag{1}$$

donde
I es la corriente en amperios
A es el área transversal en milis cuadrados
ΔT es el cambio de temperatura permitido en °C
k es la constante de disipación de calor donde:
$k = 0.048$ para rastros del PCB externos
$k = 0.024$ para rastros del PCB internos

Luego se despeja para el área quedando siguiente manera:

$$A = \left(\frac{I}{(k * \Delta T)^{0.44}} \right)^{\frac{1}{0.725}} \tag{2}$$

Luego el área transversal se divide por el grueso de cobre que posee la placa, dando como resultado el ancho del rastro del PCB.

$$T_w = \frac{A}{Cu_{Thickness}} \tag{3}$$

donde
T_w es el ancho del rastro del PCB o *Trace Width*
$Cu_{Thickness}$ es el ancho del cobre en oz/ft^2

Una vez calculado el ancho mínimo del rastro del PCB se verificó que estuviese dentro de las capacidades de fabricación de la máquina de prototipos LPKF que se encuentra en el departamento de electrónica, luego se procedió a soldar los componentes y los resultados se encuentran en las Figuras 26 y 27.

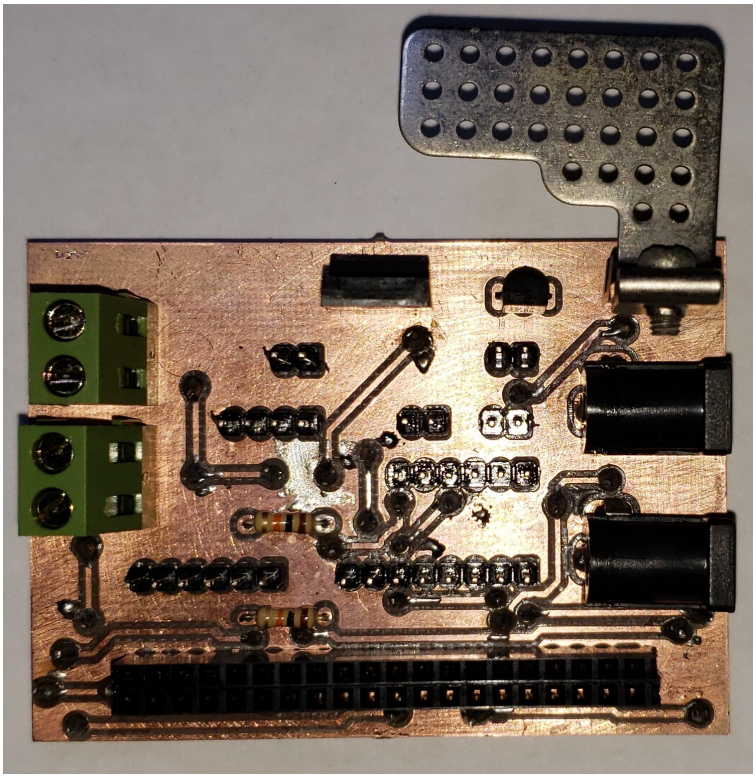


Figura 26: PCB del circuito principal en físico

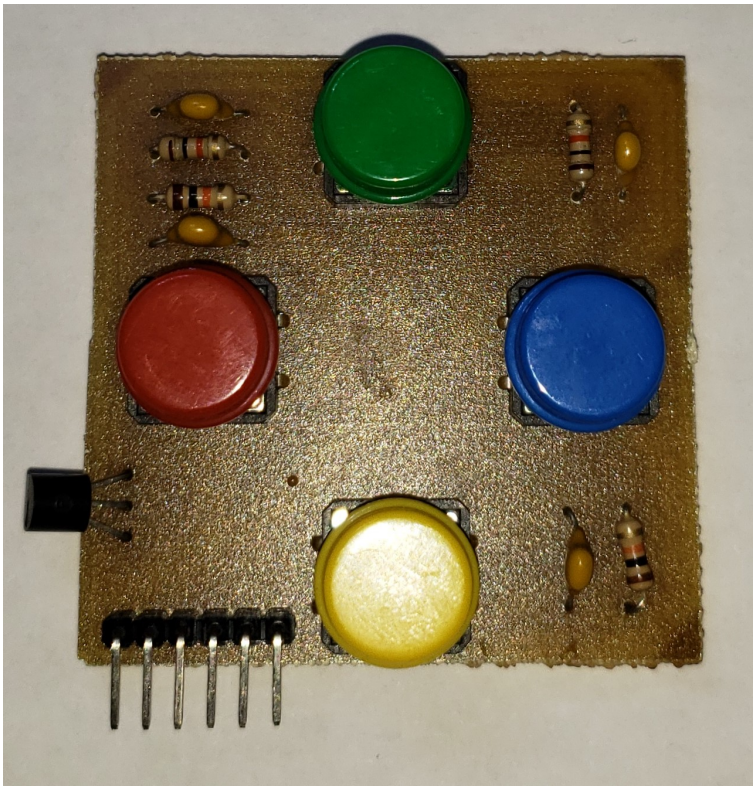


Figura 27: PCB del circuito secundario en físico

10.2. Programación del controlador principal

El lenguaje de programación utilizado para la implementación final de la plataforma fue Python. Este es un lenguaje de programación que busca la sencillez -en términos de sintaxis- sin comprometer la robustez y el funcionamiento. La plataforma de Raspberry Pi es una plataforma que cuenta con un amplio contenido y soporte por parte de su comunidad, lo cual le ha permitido a la plataforma ser reconocida y utilizada en diversidad de proyectos.

Una vez configuradas las librerías de los componentes a utilizar se procedió a desarrollar el código principal del programa. Sin embargo, durante el transcurso de la programación se observaron que habían ciertas funciones en las librerías que no se acoplaban por completo a los requerimientos del programa. Por lo tanto, se desarrollaron nuevas funciones que muestran un comportamiento apto para el programa, a continuación se muestran las soluciones a tales comportamientos. Asimismo, se consideraron las interrupciones en el programa para que la plataforma pueda funcionar correctamente.

10.2.1. Impresión limitada en pantalla

Uno de los mayores problemas que se encontró en las librerías provistas para la pantalla LCD I2C era que la impresión estaba limitada a cadenas de 16 caracteres, siendo una desventaja para la transmisión de mensajes hacia el usuario. La librería posee una función que permite desplegar más de 32 caracteres de forma que el texto se va corriendo de la segunda línea hacia la primera hasta culminar con el texto. Sin embargo, esta función se encuentra encapsulada dentro de ciclos que inhiben la capacidad de realizar otras tareas mientras está siendo ejecutada dicha función. Además, existe alta recursividad dentro de la función y el funcionamiento de esta está ligado a retrasos, al momento de disminuirlos la función presentaba *bugs*.

Por lo tanto, se procedió a implementar una función que permita ingresar cadenas de texto más largas que los 32 caracteres disponibles, luego la función divide la cadena en trozos que son capaces de desplegarse en ambas líneas de la pantalla, toma una pequeña pausa y luego realiza lo mismo hasta el terminar el mensaje, todo esto lo realiza con la capacidad de funcionar ante las interrupciones de los botones. En caso de alguna interrupción sea activada se cambia el valor de una bandera y luego al regresar a la función desarrollada se verifica si la bandera fue activada, en caso de que sí termina el proceso y limpia la pantalla para prepararse para la siguiente impresión en pantalla. El código desarrollado se encuentra en el capítulo de Anexos 15.2.1.

10.2.2. Código utilizado en el aceptador de monedas

El aceptador de monedas es un componente que no cuenta con una librería como tal dado, dado que su funcionamiento es más simple y directo comparado con los otros componentes del sistema. Lo único que se requiere es poder hacer lectura de los pulsos generados y para esto se utilizaron interrupciones el pin de entrada de la Raspberry Pi. En el archivo "pruebaMoneda.py" se encuentra el código base que se desarrolló en Python. Este permite el funcionamiento correcto del aceptador de monedas y está basado en el código de Arduino del

tutorial que se encuentra en [25]. Además, se agregaron ciertas funciones como "currentMilliTime" y "screenSaver". La primera función devuelve el tiempo de ejecución del programa en ms, ya que está función no se encuentra implementada como tal en Python. La segunda función, "screenSaver", se encarga de desplegar en pantalla un animación para indicar que está leyendo los pulsos que recibe del tragamonedas, de esta manera el usuario percibe que su moneda fue aceptada, todo esto sin interrumpir con el ciclo general del programa principal. El archivo "pruebaMoneda.py" se encuentra en la sección de anexos.

10.2.3. Interrupciones para las entradas digitales

En el transcurso del documento se han mencionado las interrupciones y cómo estas fueron necesarias en la programación final del controlador principal. Sin embargo, es hasta esta sección donde se indica la forma en que se emplearon las interrupciones. Una interrupción es evento que sucede al momento de recibir una señal en los puertos de entrada o de salida, estos eventos tienen prioridad sobre el programa en ejecución de tal manera que puede pausar algún proceso en ejecución, comenzar un nuevo proceso y una vez terminado este segundo proceso sigue la ejecución del programa principal en la posición donde se originó la interrupción, continuando la linealidad del programa. En Python las interrupciones se sujetan a cada pin que se desea activar la interrupción utilizando el siguiente código:

```
#Se adjunta la interrupcion al pin deseado
GPIO.add_event_detect(4, GPIO.RISING, callback=interrupt, bouncetime=200)
```

Donde:
El primer parámetro es el pin al cual se le desea sujetar la interrupción, la numeración depende si se encuentra en modo BOARD o modo BCM la numeración de los pines.
El segundo parámetro indica el flaco de la señal que se desea detectar.
El tercer parámetro "callback" es la función que se desea ejecutar una vez se activó.
El último parámetro "bouncetime" es un temporizador que permite prevenir dos o más interrupciones del mismo tipo durante ese período de tiempo.

En el programa principal se utilizaron interrupciones para las entradas de los botones y del aceptador de monedas. Estas permitieron detectar los cambios de estados de estas señales mientras el programa principal se ejecuta. En el código del programa principal se deseaba detectar los flancos descendientes debido a que se utilizó lógica *Pull-Up* para estos componentes. Además, las funciones que se llaman una vez es activada la interrupción deben ser cortas y tratando de minimizar instrucciones para evitar que se llame una interrupción dentro de otra interrupción.

10.2.4. Bitácora del programa

La bitácora o *log* de un programa es un registro de todos los eventos que ocurren durante el tiempo de ejecución del mismo, sin embargo, se debe especificar en el código principal qué eventos se deben registrar. En el caso del proyecto es la encargada de llevar el registro de las credenciales generadas -usuarios y contraseñas generadas-, el ancho de banda, el tiempo contratado, fecha y hora exacta de generación del usuario y el control de créditos acumulados

que posee la plataforma. En Python se utilizó la librería "logging" que está diseñada para la implementación de bitácoras, de igual forma, se utilizaron los "handlers" que posee la librería. Los "handlers" son rutinas que manejan los eventos que surjan en la ejecución del programa. Estos se utilizaron para poder generar una nueva bitácora a media noche, mientras la bitácora anterior es almacenada en un archivo nombrado por su fecha. La versión básica del código utilizado para la bitácora se encuentra en el archivo "pruebaLogGeneracio.py" que se encuentra en la sección de anexos y está basado en el código de [29].

10.2.5. Generación de las credenciales del usuario

Uno de los requerimientos del programa principal es que sea capaz de poder generar las credenciales de los usuarios de forma aleatoria. Si bien esto presenta un reto ya que usualmente los algoritmos generan números pseudo-aleatorios, es decir, el conjunto de los números generados depende de un conjunto de números previos conocidos como "semillas". Con el objetivo de mitigar la posible repetición de credenciales generadas se utilizó la librería de Python "secrets" para generar los usuarios y contraseñas. Dicha librería es capaz de generar números aleatorios seguros -según la documentación de Python- por lo cual puede ser utilizada para generación de contraseñas y *tokens* de autenticación [30]. Entonces, la librería fue utilizada para generar números aleatorios hexadecimales de 3 bytes de largo. Luego, antes de enviar el usuario generado al router MikroTik se verifica en la bitácora si dichas credenciales no hayan sido generadas previamente. La versión básica del código utilizado para la generación de credenciales se encuentra en el archivo "pruebaLogGeneracio.py" que se encuentra en la sección de anexos. En la función " buscarUser(usertofind)" que se encuentra en 15.2.3 es la encargada de leer el archivo actual de la bitácora para verificar que las credenciales no ya hayan sido generadas en el transcurso del día.

10.2.6. Comunicación Raspberry Pi - MikroTik

Una vez verificado que las credenciales generadas se encuentran disponibles se procede a enviarlas al router MikroTik en conjunto con los otros parámetros que definen el tipo de servicio -tiempo y ancho de banda máximo contratado-. Para la comunicación entre la Raspberry Pi y el router MikroTik se utilizó el protocolo de comunicación Telnet, este se implementa importando la librería de Python "telnetlib" e instanciando un objeto de clase telnetlib. Luego se inicia la comunicación y se espera hasta que el router MikroTik responda con ciertos comandos.

Una versión básica del código utilizado para la comunicación entre la Raspberry Pi se encuentra en la sección de anexos, con el nombre de "PruebMikrotik.py". El código en 15.2.4 tiene como objetivo el agregar un nuevo usuario al servidor *hotspot* que se encuentra corriendo en el MikroTik, en la versión final del programa principal se utilizó la siguiente expresión para agregar los nuevos usuarios generados:

```
command_1='/ip␣hotspot␣user␣add␣server=HotspotTesis␣name='+str(
    ↪ usuarioGenerado)+'␣password='+str(contrasenaGenerada)+'␣profile=
    ↪ Prof_'+str(megasCompra)+'Mb␣limit-uptime=0h'+str(tiempoCompra)+'
    ↪ m0s'
```

CAPÍTULO 11

Resultados del funcionamiento de la plataforma

La plataforma implementada es funcional y automatiza el proceso de generación de credenciales de los usuarios. Las credenciales generadas son únicas y permiten al usuario autenticarse exitosamente en el portal cautivo del *hotspot*. Una vez autenticado el usuario es capaz de gozar el servicio contratado por el tiempo definido, luego este pierde conexión a internet.

En caso de que el cliente utilizase las credenciales nuevamente este no podrá gozar del servicio, dado a que el router MikroTik lleva el control de la cuota de tiempo por el cual se utilizó el servicio y este bloquea cualquier tipo de conexión a internet una vez consumida su cuota, la única opción restante al cliente es realizar de nuevo el proceso de generación de credenciales. Es decir, adquirir el servicio de internet nuevamente. En las Figuras 28 y 29 se encuentra la estructura física de la primer versión de la plataforma, en ambas se encuentran todos los componentes mencionados en este documento.

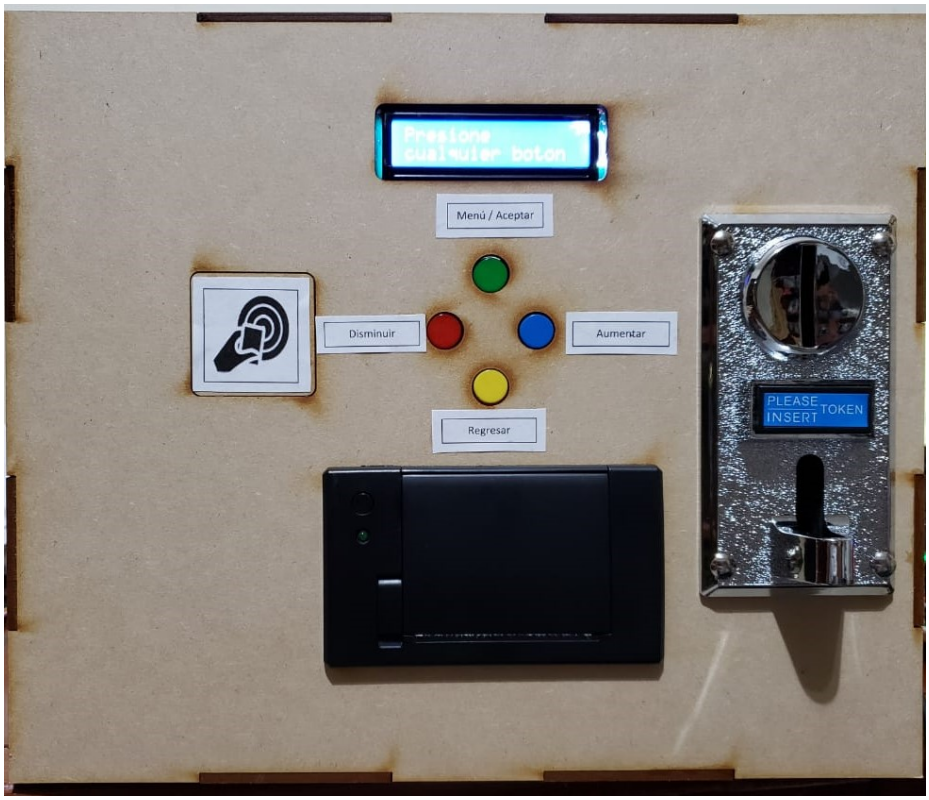


Figura 28: Estructura final de la primera versión de la plataforma de automatización

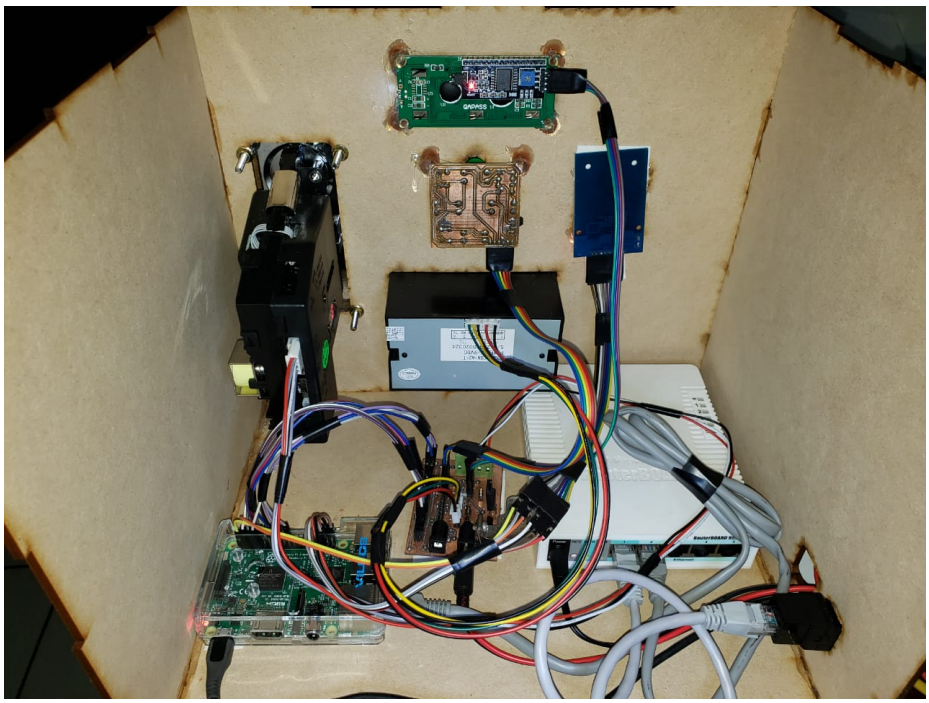


Figura 29: Vista interna de la estructura final de la primera versión de la plataforma de automatización

En una de las pruebas realizadas se obtuvo el *ticket* impreso que se observa en la Figura 30, en este se encuentra los parámetros configurados durante el proceso de generación de credenciales e información adicional respecto al proceso de uso.

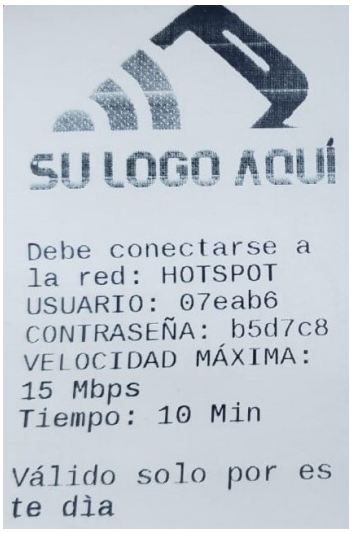


Figura 30: *Ticket* impreso luego de finalizar el proceso de generación de credenciales

En la Figura 31 se observa el consumo en tiempo real de uno de los usuarios generados, en este caso era un usuario con un ancho de banda máximo de 8Mbps.

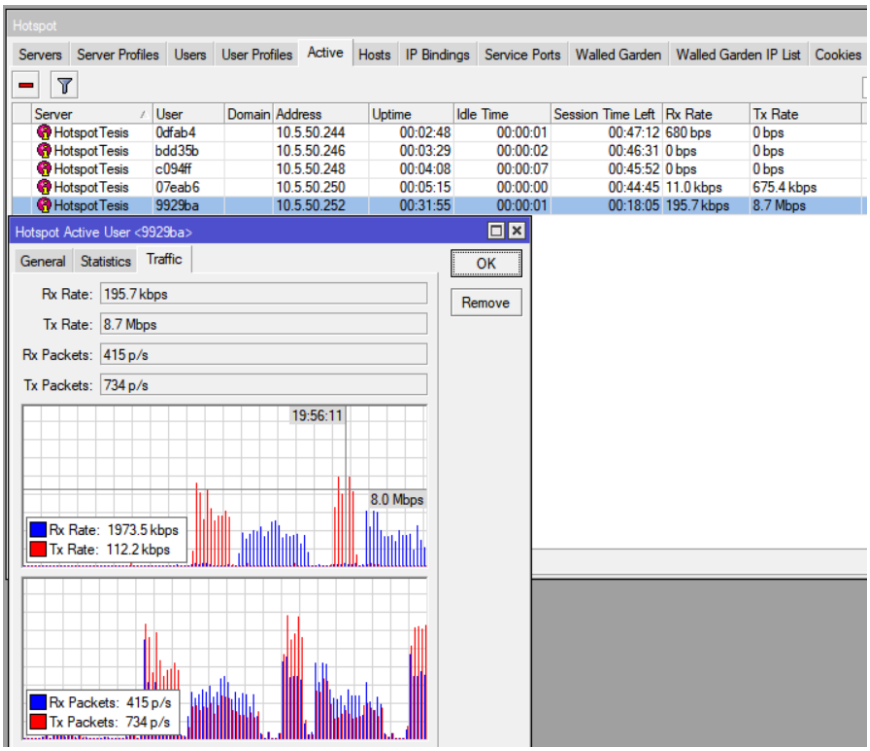


Figura 31: Gráfica de tráfico de un usuario en el *hotspot*

11.0.1. Segunda versión de la plataforma

La segunda versión de la plataforma de automatización es un conjunto de mejoras que se realizaron según la retroalimentación obtenida. Esta segunda versión tuvo tres cambios mayores, siendo los siguientes: se cambió la pantalla por una LCD 20x4, se utilizó un gabinete metálico para acomodar todos los componentes de la plataforma y se cambiaron los botones con los que interactúa el usuario.

El cambio de la pantalla LCD 20x4 permite una mejor comunicación con el usuario, ya que se pueden desplegar mensajes de una forma más clara y legible. Para realizar este cambio se tuvo que modificar varias cadenas de caracteres o *strings* que eran utilizadas para desplegar los mensajes en pantalla. Asimismo, las impresiones en pantalla que eran intermitentes fueron eliminadas o modificadas, ya que fue posible mostrar los mensajes por completo sin necesidad de limpiar la pantalla. Para la pantalla LCD20x4 se utilizó un modelo que contaba con un controlador que le permite comunicarse con la Raspberry Pi por medio de comunicación I2C. De esta manera los cambios en los *drivers* fueron mínimos. En la Figura 32 se encuentra la pantalla utilizada.

Figura 32: Pantalla utilizada en la segunda versión de la plataforma.

En la primera versión del prototipo se utilizó madera MDF de 3.2 mm, sin embargo, esta no es apta para el campo por su débil construcción. Por lo tanto, se procedió a utilizar un gabinete de acero inoxidable de la marca Hoffman de 300 mm × 300 mm × 400 mm. Dentro de este se colocaron casi todos los componentes de la plataforma exceptuando al *router* MikroTik. El *router* no se colocó dentro del gabinete dado que es de metal y se crea una jaula de Faraday, anulando el campo electromagnético que irradian las antenas del WiFi. Debido a esto el *router* MikroTik se colocó en la parte externa del gabinete. En las Figuras 33 y 34 se encuentran las vistas externas e internas de la segunda versión de la plataforma.

Figura 33: Vista externa de la segunda versión de la plataforma en el gabinete metálico


Figura 34: Vista interna de la segunda versión de la plataforma en el gabinete metálico

Por último, el PCB del circuito secundario fue modificado de tal manera que los botones ya no se encuentran soldados a este, sino, estos están conectados por medio de terminales de tornillos. De esta manera, se pueden sujetar los botones al gabinete y dan una mayor sensación de firmeza. En la Figura 35 se encuentra el PCB modificado.

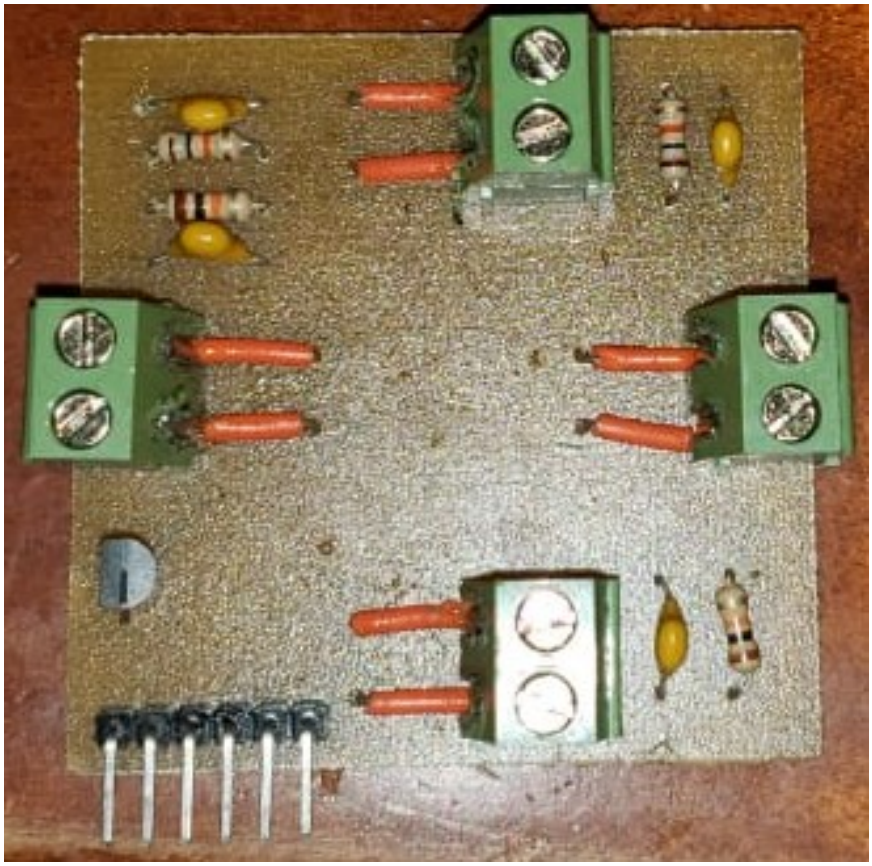


Figura 35: PCB modificado para los nuevos botones.

CAPÍTULO 12

Conclusiones

- El resultado final de la plataforma diseñada es completamente funcional, ya que es capaz de automatizar el proceso de generación de credenciales y administración de usuarios en un *hotspot* MikroTik.
- La Raspberry Pi está configurada para ejecutar el programa principal de forma automática al momento de iniciar el sistema operativo en caso de que esta sufra de algún corte en la energía eléctrica.
- La bitácora del programa principal permite llevar un control sobre el uso de la plataforma.
- La Raspberry Pi está configurada para almacenar la bitácora diariamente y de esta manera se obtiene un registro histórico del uso de la plataforma.
- Los módulos descritos en el documento fueron implementados correctamente, diseñando códigos que se acoplaron a los requerimientos de funcionamiento.
- La plataforma diseñada es sencilla en términos de implementación y armado.
- La plataforma diseñada posee la suficiente robustez para permanecer funcionando por días sin presentar anomalía alguna.

CAPÍTULO 13

Recomendaciones

- Durante la realización del proyecto se recomienda utilizar plataformas de versionamiento, de esta manera se puede trabajar en los códigos del programa con la existencia de un respaldo del original.
- Al momento de diseñar la placa del circuito impresa se recomienda realizar un diseño capaz de funcionar como un *shield*, de esta manera la plataforma mantendrá una forma más compacta.
- Se recomienda buscar diferentes alternativas para la pantalla LCD 16x2, de esta manera se podrán mostrar los mensajes de una forma más clara hacia el usuario. (Esto se realizó en la segunda versión de la plataforma.)
- Se recomienda utilizar un aceptador de billetes para futuras implementaciones, sin embargo, es necesario considerar el precio ya que eleva notablemente el costo de la plataforma.
- Durante las pruebas de funcionamiento de la plataforma se observó que el rollo de papel térmico puede ser no suficiente para varias impresiones de los *tickets*, por lo tanto, se recomienda utilizar rollos más largos a los 0.85 m que permitan una mayor cantidad de impresiones.

CAPÍTULO 14

Bibliografía

[1] *Diseño de un Hostpot para los pasrques de la ciudad de San Gabriel – Cantón Montúfa*, [Visitado el 10. Ene. 2019], Escuela politécnia Nacional, 2015. dirección: `http://bibdigital.epn.edu.ec/handle/15000/10670`.

[2] *Diseño e Implementación de un Portal Cautivo que Permite a la Venta de Tickets de Internet para un Hotspot, Empleando Heramientas de Software Libre*, [Visitado el 10. Ene. 2019], Escuela politécnia Nacional, 2011. dirección: `http://bibdigital.epn.edu.ec/handle/15000/3953`.

[3] I. por fichas. (2017). "¿Quién es Internet por fichas?" [Visitado el 10. Dic. 2018], dirección: `https://www.internetporfichas.com/quienes-somos.html`.

[4] MikroTickets. (2018). "¿Quiénes somos?" [Visitado el 10. Dic. 2018], dirección: `https://www.mikrotickets.com.mx`.

[5] W. Stallings y W. Stallings, *Wireless communications and networks*, es, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005, OCLC: ocm56617248, ISBN: 978-0-13-191835-1 978-0-13-196790-8.

[6] J. Salazar, "REDES INALÁMBRICAS," en *Redes Inalámbricas*, České vysoké učení technické v Praze Fakulta elektrotechnická, 2010, págs. 6-20.

[7] MikroTik. (1996). "About Us." [Visitado el 15. Feb. 2019], dirección: `https://mikrotik.com/aboutus`.

[8] Pondi. (2008). "RouterOS." [Visitado el 10. Feb. 2019], dirección: `http://www.mikrotik-routeros.net/routeros.aspx`.

[9] *Old Calculator Web Museum - News Archive - Texas Instruments TMS 1802 Single Chip Calculator IC Introduction*, [Visitado el 15. Feb. 2019], dic. de 2016. dirección: `https://www.oldcalculatormuseum.com/n-ti-tms1802.html`.

[10] B. W. Gunther Gridling, *Introduction to Microcontrollers*, [Visitado el 15. Feb. 2019], feb. de 2007. dirección: `https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf`.

[11] M. Land, *Microprocessor and Microcontroller Architecture*, [Visitado el 15. Jun. 2019]. dirección: `http://cs.hadassah.ac.il/staff/martin/embedded/slide03-1.pdf`.

[12] P. Zhang, "Advanced Industrial Control Technology," *Advanced Industrial Control Technology*, 2010. DOI: 10.1016/C2009-0-20337-0.

[13] O. Robles, *MikroTik - Hotspot Server con Portal Cautivo*, [Visitado el 15. Feb. 2019]. dirección: http://soporte.syscom.mx/networking/mikrotik/mikrotik-hotspot-server-con-portal-cautivo.

[14] PiMyLifeUp, *Build your own Raspberry Pi RFID Attendance System*, [Visitado el 15. Jun. 2019], 2019. dirección: https://pimylifeup.com/raspberry-pi-rfid-attendance-system/.

[15] Mifare®, *MFRC522 Datasheet*, [Visitado el 15. Jun. 2019], 2016. dirección: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[16] T. R. P. Guy, *Raspberry Pi - Mini LCD Display Tutorial*, [Visitado el 15. Jun. 2019], 2016. dirección: https://www.youtube.com/watch?v=fR5XhHYzUK0.

[17] L. XIAMEN AMOTEC DISPLAY CO., *SPECIFICATIONS OF LCD MODULE*, [Visitado el 15. Jun. 2019], 2008. dirección: https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf.

[18] Sparkfun, *Thermal Printer*, [Visitado el 18. Mar. 2019], 2019. dirección: https://www.sparkfun.com/products/14970.

[19] P. Burgess, *Networked Thermal Printer using Raspberry Pi and CUPS*, [Visitado el 10. Abr. 2019], 2019. dirección: https://learn.adafruit.com/networked-thermal-printer-using-cups-and-raspberry-pi/overview.

[20] Scruss, *Thermal Printer driver for CUPS, Linux, and Raspberry Pi: zj-58*, [Visitado el 10. Abr. 2019], 2019. dirección: http://scruss.com/blog/2015/07/12/thermal-printer-driver-for-cups-linux-and-raspberry-pi-zj-58/.

[21] A. N. Vinogradov, *zj-58*, [Visitado el 10. Abr. 2019], 2019. dirección: https://github.com/klirichek/zj-58.

[22] Skipped, *Aleación y peso unitario de las monedas de actual circulación, de conformidad con resolución JM-18-2008 y Acuerdo No. 32-2008 del Congreso de la República de Guatemala*, [Visitado el 22. May. 2019], 2016. dirección: http://www.banguat.gob.gt/inc/ver.asp?id=/Publica/monedasybilletes/monedasybilletes2019.htm&e=144618.

[23] S. Electronics, *Coin Acceptor - Programmable (3 coin types)*, [Visitado el 18. Mar. 2019], 2019. dirección: https://www.sparkfun.com/products/11719.

[24] V. Par, *CH923/CH925/CH926/CH928/JY923/JY925/JY926/JY928 coin acceptor: Features and caveats*, [Visitado el 22. May. 2019], 2017. dirección: http://blog.deconinck.info/post/2017/02/25/CH923/CH926/CH928-coin-acceptor-features-and-caveats.

[25] Skipped, *Make Money With Arduino*, [Visitado el 22. May. 2019], 2016. dirección: https://www.instructables.com/id/Make-Money-with-Arduino/.

[26] A. Circuits, *Trace Width Calculator*, [Visitado el 18. Ago. 2019], 2018. dirección: https://www.4pcb.com/trace-width-calculator.html.

[27] IPC, *IPC-2221A Generic Standard on Printed Board Design*, [Visitado el 1. Ago. 2019], 2003. dirección: https://www.sphere.bc.ca/class/downloads/ipc_2221a-pcb%20standards.pdf.

[28] ——, "IPC-2221A Generic Standard on Printed Board Design," en *IPC-2221A Generic Standard on Printed Board Design*, Association Connecting Electronics Industries, 2003, págs. 40-42.

[29] S. Løvås. (2014). "TimedRotatingFileHandler with format and console prints." [Visitado el 20. Ago. 2019], dirección: `https://stackoverflow.com/questions/32018544/timedrotatingfilehandler-with-format-and-console-prints`.

[30] R. Hettinger. (2016). "secrets — Generate secure random numbers for managing secrets." [Visitado el 25. Ago. 2019], dirección: `https://docs.python.org/3/library/secrets.html`.

CAPÍTULO 15

Anexos

15.1. Esquemáticos de los circuitos diseñados

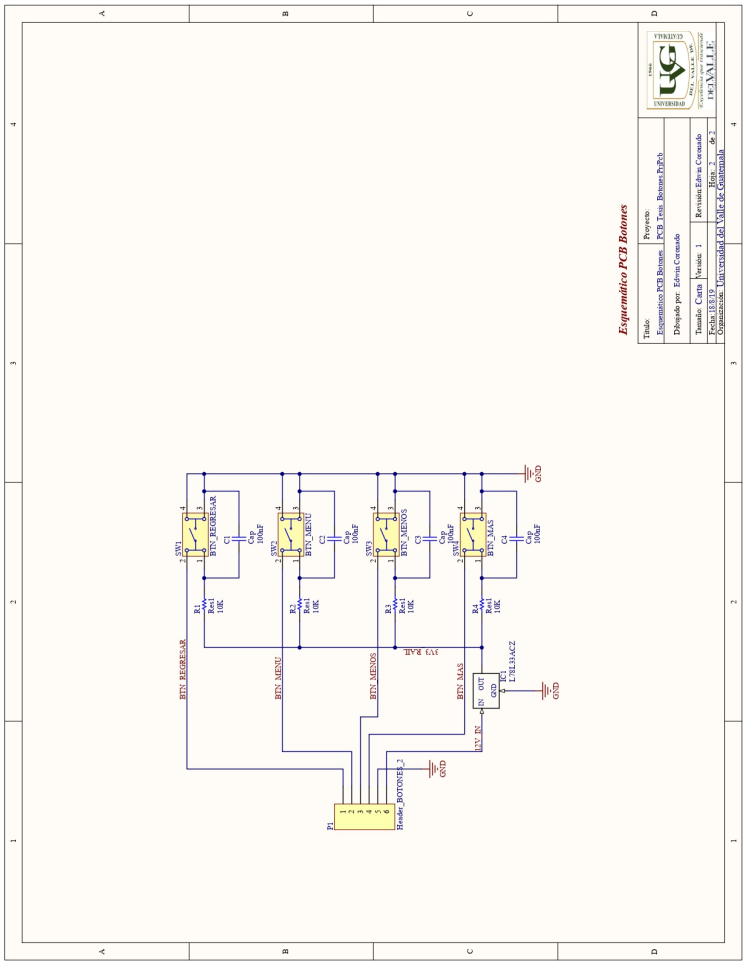


Figura 36: Esquemático de la placa secundaria

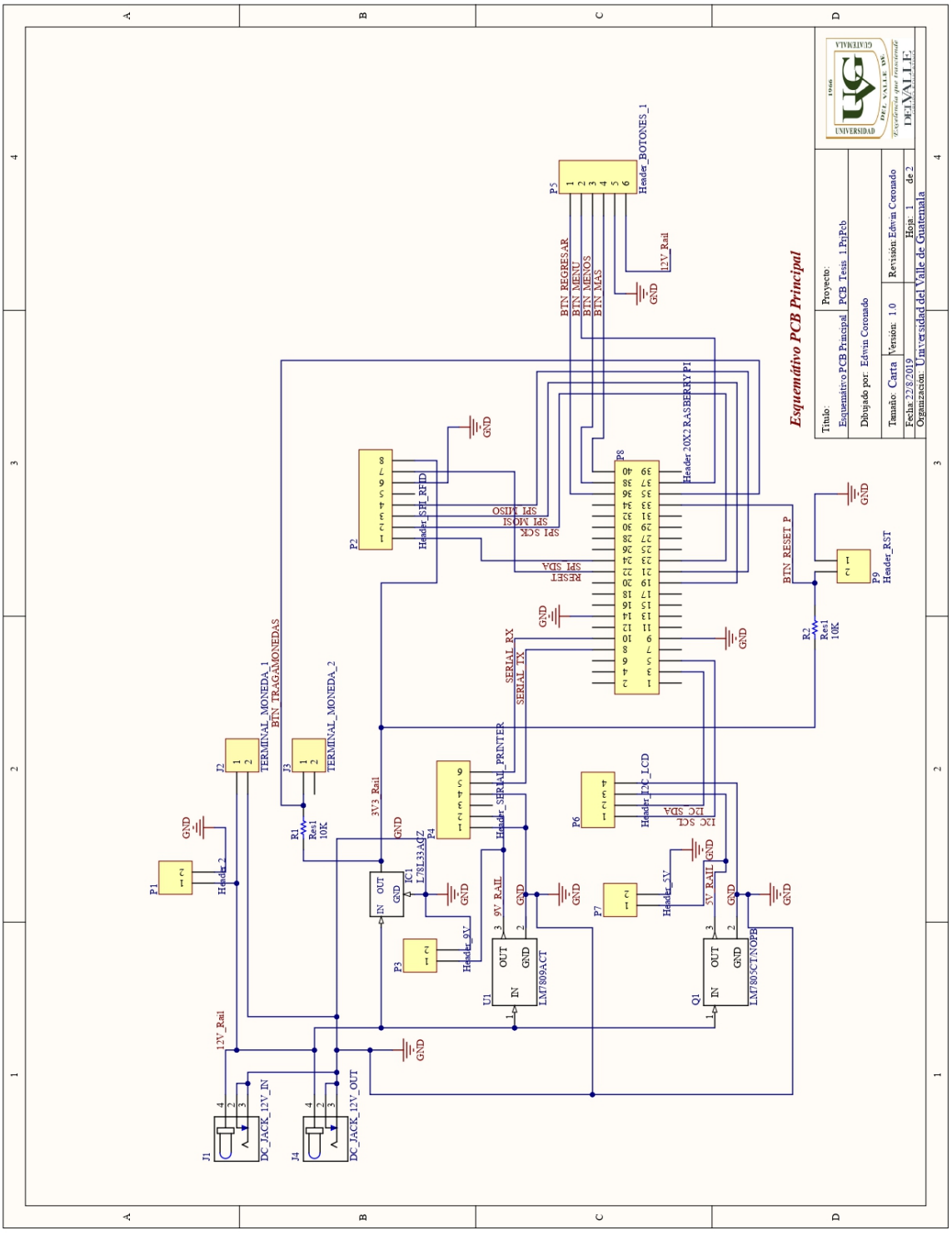


Figura 37: Esquemático de la placa principal

15.2. Código fuente funciones desarrolladas

15.2.1. Impresión limitada en pantalla

```
#Funcion creada para sustituir rolling text debido a que agrega bugs al
    ↪ programa
def imprimir_Texto(display, texto = '', num_columnas = 16):
    #Definicion de variables Globales
    global long_stringInterrupt, long_stringisRunning
    if (len(texto) > num_columnas):
        long_stringisRunning = True
        textoArray = texto.split()
        textoArray.append('')
        textoArray.append('')
        arrayImprimir = []
        i = 0
        if (long_stringInterrupt == False):
            while i < len(textoArray):
                #Anadimos este if para poder interrumpir el display
                if(long_stringInterrupt == True):
                    display.lcd_clear()
                    long_stringisRunning = False
                    break

                if (textoArray[i]==''):
                    break
                elif (( len(textoArray[i]) + len(textoArray[i+1]) )<=15 and
                    ↪ textoArray[i+1] != ''):
                    arrayImprimir.append(textoArray[i] + "␣" + textoArray[i
                        ↪ +1])
                    i += 1
                elif (( len(textoArray[i]) + len(textoArray[i+1]) )>15 and
                    ↪ textoArray[i+1] != ''):
                    arrayImprimir.append(textoArray[i])
                elif (( len(textoArray[i]) + len(textoArray[i+1]) )<15 and
                    ↪ textoArray[i+1] == ''):
                    arrayImprimir.append(textoArray[i])

                i += 1
            if (len(arrayImprimir)%2 != 0):
                arrayImprimir.append('')
        else:
            display.lcd_clear()
            time.sleep(0.1)
            long_stringisRunning = False

        if (long_stringInterrupt == False and long_stringisRunning == True):
```

```
            for i in range(int(len(arrayImprimir)//2)):
                if (long_stringInterrupt == True ):
                    display.lcd_clear()
                    break
                else:
                    display.lcd_clear()
                     #print("1 - " + arrayImprimir[2*i])
                    display.lcd_display_string(arrayImprimir[2*i],1)
                    #print("2 - " + arrayImprimir[2*i + 1])
                    display.lcd_display_string(arrayImprimir[2*i + 1],2)
                    time.sleep(1)
        else:
            long_stringInterrupt = False
            display.lcd_clear()
        long_stringisRunning = False
        display.lcd_clear()
```

15.2.2. pruebaMoneda.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Referencia Codigo para tragamonedas
# instructables.com/id/Make-Money-with-Arduino


import sys
sys.path.insert(1,'/home/pi/lcd')
import lcddriver
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD) #Enumeracion de pines en orden ie PIN #

display = lcddriver.lcd()

#Numeracion de Pines en modo BOARD
pinLectorMonedas = 35

#Variable del estado de la moneda
estadoMoneda = 0

#Acumulador que lleva control de los creditos actuales
creditosActuales = 0

#Variable utilizadas para la animacion de aceptar moneda
#currentMillisMoneda = 0
```

```
#previousMillisMoneda = 0

#Variable que guarda en ms el inicio del pulso
inicioPulso = 0

#Contador que nos permite determinar el caracter a animar
contadorAnimacion = 0

#Variable que indica si la animacion debe estar corriendo o no
animationRun = False

GPIO.setup(pinLectorMonedas,GPIO.IN,pull_up_down = GPIO.PUD_UP)

currentMilliTime = lambda:int(round(time.time() * 1000))

msjAnimacion1 = "Leyendo␣Moneda"
msjAnimacion2 = ["****************","*␣**************","**␣************","
    ↪ ***␣***********","****␣**********","*****␣**********"
,"******␣*********","*******␣********","********␣*******","*********␣******"
    ↪ ,"**********␣*****","***********␣****",
"************␣***","*************␣**","**************␣*","***************␣",
    ↪ "␣***************"]


def monedaInsertada(channel):
    global estadoMoneda, creditosActuales, inicioPulso
    inicioPulso = currentMilliTime()
    creditosActuales = creditosActuales + 0.05
    estadoMoneda = 1

def screenSaver():
    global animationRun, contadorAnimacion

    if(animationRun == True):
        display.lcd_display_string(msjAnimacion1, 1)
        display.lcd_display_string(msjAnimacion2[contadorAnimacion], 2)
    else:
        display.lcd_clear()

    if(contadorAnimacion >= 16):
        contadorAnimacion = 0
    else:
        contadorAnimacion = contadorAnimacion + 1

GPIO.add_event_detect(pinLectorMonedas, GPIO.FALLING, callback=
    ↪ monedaInsertada, bouncetime=50)

display.lcd_display_string("Prueba␣Display", 1)
```

```
time.sleep(2)
display.lcd_clear()

try:
    while True:
        if(estadoMoneda == 1):
            estadoMoneda = 0
            print("Creditos␣actuales:␣" + str(creditosActuales))
            print("Tiempo␣pulso␣" + str(currentMilliTime()))

        if(currentMilliTime() - inicioPulso > 150):
            animationRun = False
            #display.lcd_clear()
            #print("Animation " + str(animationRun))
        else:
            animationRun = True
            #print("Animation " + str(animationRun))
        screenSaver()
except KeyboardInterrupt: #Para salirnos del programa con ctrl+c
    print ("Limpiando␣programa!")
    GPIO.cleanup()
    time.sleep(5)
```

15.2.3. pruebaLogGeneracion.py

```
# -*- coding: cp1252 -*-
#Programa que genera usuarios y contrasenas Aleatorios
#Esta es una version mejorada que utiliza la liberia LOG para un mejor
    ↪ funcionamiento

import time
import random
import secrets
from datetime import datetime
#Liberia utilizada para la bitacora
import logging, logging.handlers



#Linea necesaria para guardar en un path especifico
import os
os.path.join

import sys
sys.path
```

```
#sys.path.append(r'C:\Users\Edwin Coronado\OneDrive\Documentos\2019\
    ↪ Documentos Tesis\Raspberry Pi\genUsers')
#Version Windows
#sys.path.append(r'C:\Users\Edwin Coronado\OneDrive\Documentos\2019\
    ↪ Documentos Tesis\Raspberry Pi\logs')
#Version Linux
sys.path.append('/home/pi/proyectoTickets/logs')

#-------------Version4 Version funcional, guarda los Logs cada intervalo de
    ↪ tiempo
#Basado en el codigo de:
#https://stackoverflow.com/questions/32018544/timedrotatingfilehandler-with-
    ↪ format-and-console-prints

#Directory Log Version Windows
#directorylog = r'C:\Users\Edwin Coronado\OneDrive\Documentos\2019\
    ↪ Documentos Tesis\Raspberry Pi\logs'

#Directory Log Version Linux
directorylog = '/home/pi/proyectoTickets/logs'
#Nombre log secundario
backuplog = os.path.join(directorylog,'backuplog.log')


#get the root logger
rootlogger = logging.getLogger()
#set overall level to debug, default is warning for root logger
rootlogger.setLevel(logging.DEBUG)

#En caso de que se quiera hacer a media noche
#filelog = logging.handlers.TimedRotatingFileHandler('./mylogfile.log',
                #when='midnight', interval=1)
#En caso que se quiera hacer cada 2 Segundos
#filelog = logging.handlers.TimedRotatingFileHandler(backuplog,
                #when='S', interval=2)
#En caso que se quiera hacer cada 10 Minuto
filelog = logging.handlers.TimedRotatingFileHandler(backuplog,
                when='M', interval=5)
filelog.setLevel(logging.DEBUG)
fileformatter = logging.Formatter('%(asctime)s␣-␣␣%(levelname)␣-8s-␣%(
    ↪ message)s')
filelog.setFormatter(fileformatter)
rootlogger.addHandler(filelog)

#setup logging to console
console = logging.StreamHandler()
console.setLevel(logging.INFO)
```

```
formatter = logging.Formatter('%(asctime)s - %(levelname) -8s- %(message)s
    ↪ ',
                        datefmt='%Y-%m-%d %H:%M:%S')
console.setFormatter(formatter)
rootlogger.addHandler(console)

#get a logger for my script
logger = logging.getLogger(__name__)

logger.info('This is logged to console and file')
logger.debug('This is only logged to file')


#-------------FIN VERSION 4 Logger

#-----Codigo Generador de Usuarios
#Funcion que busca en el documento especificado si existe el usuario
    ↪ generado
def buscarUser(usertofind):
    if os.path.exists(backuplog) == True:
        with open(backuplog) as f:
            datafile = f.readlines()
        found = False

        for line in datafile:
            if usertofind in line:
                #print(line)
                return True
        return False
    else:
        return False


#Funcion que genera el usuario y verifica que no haya sido generado durante
    ↪ el tiempo
def generarUsuario():
    #Se queda en un ciclo mientras genera un numero aleatorio nuevo
    while True:
        random.seed(datetime.now())
        print(datetime.now())
        #randNum = str(random.random())
        #randNum = str(random.SystemRandom())
        randNumF = secrets.token_hex(3)
        randContF = secrets.token_hex(3)
        while randNumF == randContF:
            randContF = secrets.token_hex(3)
            print( "Usuario y contrasena son iguales: "+randContF)
        #randNumF = randNum[-7:-1]
```

```
        #print(randNumF)
        #Revisa si el numero generado ya se genero previamente
        if buscarUser(randNumF) == True:
            print('Usuario duplicado '+randNumF)
            pass
        elif buscarUser(randNumF) == False:
            print('Usuario NO encontrado '+randNumF)
            logger.info('Se ha generado el USUARIO: ' + randNumF + " con
                ↪ CONTRASENA: " + randContF)
            return randNumF, randContF
            break

contador = 30
while True:
    generarUsuario()
    contador -= 1
    time.sleep(20)
    print (contador)
    if contador <= 0:
        break

    pass
```

15.2.4. PruebMikrotik.py

```
# -*- coding: utf-8 -*-

#Referencias https://forum.mikrotik.com/viewtopic.php?t=61786
import telnetlib,time
#config_user_password_port_etc.
#Direccion IP del MikroTik a conectar
HOST='DireccionIPMikroTik'
#Puerto sobre el cual correria la comunicacion telnet, por defecto es el 23
PORT='23'
#Se colocan los comandos "+c" y "+t" luego del usuario administrador para
    ↪ deshabilitar la consola interactiva de MikroTik
#De esta manera es mas compatible con la libreria telnetlib y omite
    ↪ caracteres especiales que no reconozca dicha libreria
user= 'UsuarioAdministradorMikroTik+c+t'
password= 'ContrasenaDelMikroTik'

#Commando utilizado para agregar un nuevo usuario al MikroTik
command_1='/ip hotspot user add server=HotspotTesis name=user9 password=
    ↪ user9 profile=Prof_2Mb limit-uptime=0h90m0s'
command_3='quit'
```

```
#Commando en MikroTik que busca si el usuario definido existe, en caso de
    ↪ que si exista devuelve un True
command_2=":if␣([:len␣[:ip␣hotspot␣user␣find␣name=user7]]␣>␣0)␣do={:put␣true
    ↪ ;}␣else={:put␣false;}"

#tn=telnetlib.Telnet(HOST,PORT)
tn = telnetlib.Telnet()
tn.open(HOST,PORT)
#input user
tn.read_until(b"Login:␣")
tn.write(user.encode('UTF-8') + b"\n")
#input password
tn.read_until(b"Password:␣")
tn.write(password.encode('UTF-8') + b"\n")
time.sleep(2)
print("Se␣ha␣iniciado␣Sesion")

print(tn.read_until(b']␣>'))
tn.write(command_2.encode('UTF-8')+b"\r\n")

if(tn.read_until(b'true')):
    print("Usuario␣Repetido")
elif(tn.read_until(b'false')):"""
print("Usuario Valido")
print(tn.read_until(b'] >'))
#print(tn.read_eager())
tn.write(command_1.encode('UTF-8')+b"\r\n")
time.sleep(2)


print(tn.read_until(b'] >'))
#print(tn.read_eager())
tn.write(command_3.encode('UTF-8')+b"\r\n")

print(tn.read_all())
tn.close()
```