Institución Universitaria

GESTIÓN DE SENSORES Y MONITOREO DE RED BAJO HERRAMIENTAS RASPBERRY PI - ARDUINO

Favián Osvaldo Apráez Cuatindioy

Carlos Andrés Aguirre Giraldo

Instituto Tecnológico Metropolitano

Ingeniería en Telecomunicaciones

Medellín, Colombia

2016

GESTIÓN DE SENSORES Y MONITOREO DE RED BAJO HERRAMIENTAS RASPBERRY PI - ARDUINO

Favián Osvaldo Apráez Cuatindioy

Carlos Andrés Aguirre Giraldo

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:

Ingeniero en Telecomunicaciones

Director (a):

Alexander Arias Londoño

IEo, MSc.

Línea de Investigación:

Monitoreo de redes

Instituto Tecnológico Metropolitano

Ingeniería en Telecomunicaciones

Medellín, Colombia

2016

Agradecimientos

A nuestro Director, el Profesor Alexander Arias, y su incondicional apoyo durante el desarrollo de esta investigación.

Al programa de Ingeniería en Telecomunicaciones de la Universidad Instituto Tecnológico Metropolitano (ITM) por permitirnos realizar las pruebas de laboratorio de este trabajo.

De igual modo agradecemos todo el apoyo que nos brindaron.

Resumen

En el desarrollo de este trabajo se plantea el diseño e implementación de un sistema de gestión de variables (sistema domótico) y monitoreo de redes con aplicaciones OpenSource y OpenHardware basado en herramientas de bajo coste como Raspberry pi y Arduino.

Raspberry Pi como servidor principal tiene la capacidad de controlar el monitoreo de red donde se evalúa los servicios públicos y privados de un equipo, bajo la aplicación Nagios y con conexión remota; como también puede controlar el sistema domótico (Arduino) el cual fue integrado por intermedio de una comunicación serial, ayudando a tomar datos con el sensor de temperatura y la activación de un actuador, todo siendo supervisado por una interfaz gráfica que se desarrolló bajo el lenguaje de programación Python, logrando verificar cada servicio: DIRECCIÓN IP, CPU, RAM, Disco Duro, ICMP, SSH de cada equipo, aparte de ello se crea mediante código de programación la visualización de los recursos locales de la maquina por intermedio del Display LCD.

Aspectos importantes del prototipo, es la integración de los dispositivos Arduino, Bluetooth, Dht-11, LCD, que son necesarios para el desarrollo de la comunicación y control. Pensando en la portabilidad y poder visualizar los datos por el celular se creó una App para verificar los resultados de humedad, temperatura y la activación de un actuador por medio del bluetooth que está conectado directamente al Arduino, permitiendo un trabajo proactivo y de fácil administración, este proyecto se basó en encontrar una solución administrativa de red, con dispositivos de bajo costo, teniendo en cuenta comunicación y supervisión que intervienen varios protocolos de comunicación y un amplio manejo de lenguaje de programación como python Bash y C.

Tabla de contenido

Lista de figuras

Tabla de Tablas

Lista de Símbolos y abreviaturas

RPI: *Raspberry Pi*

CPU: *Unidad de procesamiento Lógico*

GPU: *Unidad procesamiento grafico*

DHT11*: Sensor de Temperatura Humedad*

VPN: *Virtual Private Network*

HC-05*: Dispositivo Bluetooth*

TIC: *Tecnología Información Comunicación*

SLA: *acuerdo nivel de servicio*

SSH*: Security Shell*

GPIO*: General Purpose Input/Output*

PWM*: Pulse Width Modulation*

ARM*: Arquitectura Risc (Reduced Instruction Set Computer) ordenador con Conjunto Reducido de Instrucciones*

UC: *Unidad de Control*

ALU: *Unidad Aritmético lógica*

IDE*: Integrated Development Environment*

NRPE: *Nagios Remote Plugin Executor*

RDP *(Remote Desktop Protocol)*

1. Introducción

Cada vez las organizaciones en el desarrollo de sus redes se enfrentan a procesos complejos siendo indispensable conocer el estado de sus redes y tener el control de estas, para ello el monitoreo constante es indispensable a: equipos, servicios y dispositivos, para identificar cualquier tipo de falla sobre la red y lograr su atención en el menor tiempo posible. Es por esto que el proyecto incorpora dispositivos de bajo coste que ayuden a controlar y verificar los diferentes estados que se presentan en una red, para ello la implementación de un dispositivo como la Raspberry Pi que ahora en adelante la llamaremos RPI, pequeña placa de bajo coste con una arquitectura de procesador ARM. Ya que este dispositivo tiene la facilidad de instalar herramientas OpenSource. Ante el crecimiento de los sistemas de gestión y monitoreo de redes se piensa en facilitar la implementación de un sistema de monitoreo de red con aplicaciones OpenSource, convirtiéndose en un actividad cada vez más indispensable en el mundo de las redes o transmisiones de información que hoy en día tienen las organizaciones, e identificando las causas del problema que se están presentando en cualquier acceso de red, otorgándole al administrador un dispositivo de bajo costo que unifique e integre y sea una herramienta portable, de bajo consumo de potencia, con acceso remoto para verificar la red que se vaya a censar o a monitorear, además, a eso se tiene en cuenta mantener el correcto funcionamiento de cada uno de los componentes del sistema que intervienen en la conformación de la red, siendo este el objetivo que le da un valor agregado, para un mejor desempeño, adaptando seguridad y calidad del servicio de la red.

Ahora se indican cada uno de los objetivos en los que se centra el desarrollo del proyecto contando con diseño, desarrollo de compatibilidad e integración a sistemas de monitoreo-domótica, que se detallan a continuación;

OBJETIVOS

General

Diseñar un sistema de Gestión de variables y monitoreo remoto de redes con aplicaciones OpenSource y Open Hardware.

Específicos

- Evaluar diferentes aplicaciones OpenSource de gestión y monitoreo de red que tengan las condiciones específicas de adaptación a herramientas de bajo coste.

- Diseñar el esquema para la gestión y monitoreo de red con aplicaciones OpenSource con la herramienta Raspberry pi.

- Implementar el sistema de gestión y monitoreo de red bajo la herramienta de bajo coste Raspberry pi con la aplicación OpenSource adecuada a este tipo de herramienta.

- Diseñar un sistema de administración domótico que se integre al sistema de gestión y monitoreo.

2. Marco Teórico

2.1. Comunicación inalámbrica

Recientes tendencias de comunicación inalámbricas en manejo de bajo costo han sugerido el uso de la Raspberry Pi como la semilla de toda una revolución, y aunque originalmente este dispositivo mini PC, fue creado como una solución orientada a entornos educativos, con posibilidades y presentaciones que se han logrado convertir en base de todo tipo de proyectos de hardware y software.

2.2. Estructura y arquitectura de Raspberry PI

Es una placa reducida que trabaja como un ordenador con su propio software e interfaces con periféricos para conexiones de red, el cual se integra con otros dispositivos mencionados en la Figura 1.1 que ayudan a facilitar la implementación, de sistemas con diferentes enfoques a través de los puertos GPIO, soportando diferentes lenguajes de programación que son indispensables para la configuración de plugins y scripts en los archivos del sistema básicamente es un ordenador practico y de bajo coste, en la Tabla 1.1 se especifica las características principales.

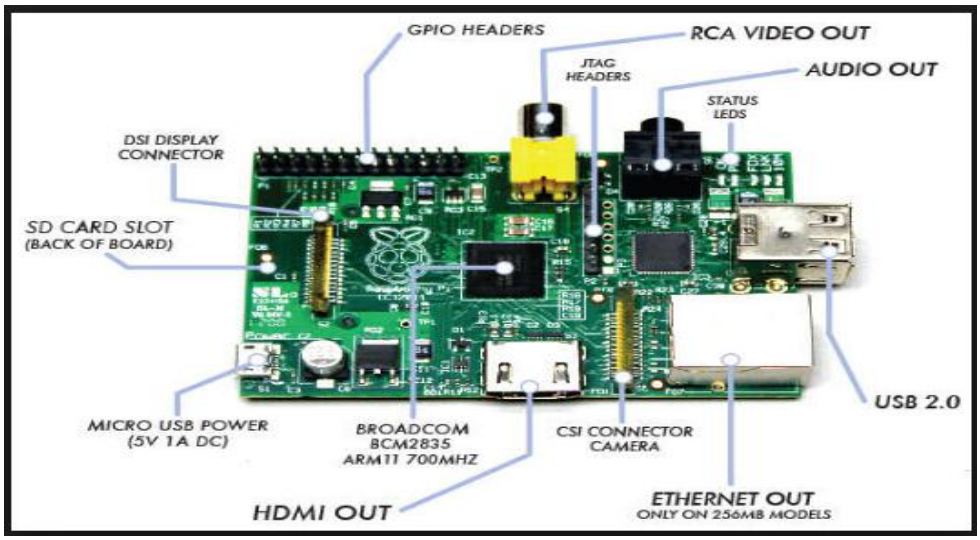


Figura 1.1 frontal del Rasberry Pi.

.

Tabla 1.1 Características principales de la placa Raspberry PI

SISTEM (Soc) Chip	**Fabricante –Broadcom BCM2836**
Cpu	**ARM Cortex-A7 4 nucleos 900MHz**
Almacenamiento	**Micro SD**
Gpu	**Broadcom Video Core 250 Mhz Open GL ES 2.0**
Ram	**1GB LPDDR2 SDRAM 450MHz**
Usb 2.0	**4**
Salidas de Video	**HDMI 1.4 1920x1200 pixeles**
Ethernet	**Si 10/100 Mbs**
Consumo	**5v 700mA depende del trabajo de los 4 cores**

2.2.1. Puertos Entrada / Salida

La RPI interactúa con puertos de entrada y salida, para el funcionamiento de componentes externos que se muestra en la Figura 1.2 para un funcionamiento independiente, estableciendo una interfaz que decodifica el bus de direcciones, el bus de datos se utiliza para el paso de datos entre el periférico y la memoria. Las líneas especiales de control sirven para coordinar y sincronizar la transferencia.

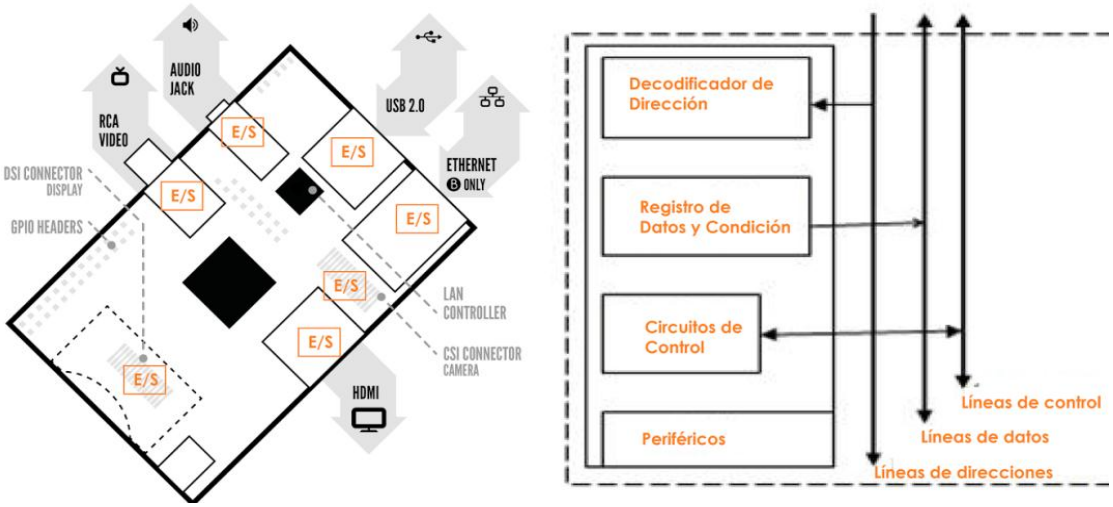


Figura 1.2 Entrada y salida.

2.2.2. Componentes externos

En el análisis de nuestra placa RPI es muy importante observar y reconocer los componentes externos teclado, mouse, microSD como se muestra en la Figura 1.3 que son generales en nuestro dispositivo, siendo un modelo genérico de un ordenador. (Richmond, 2014)

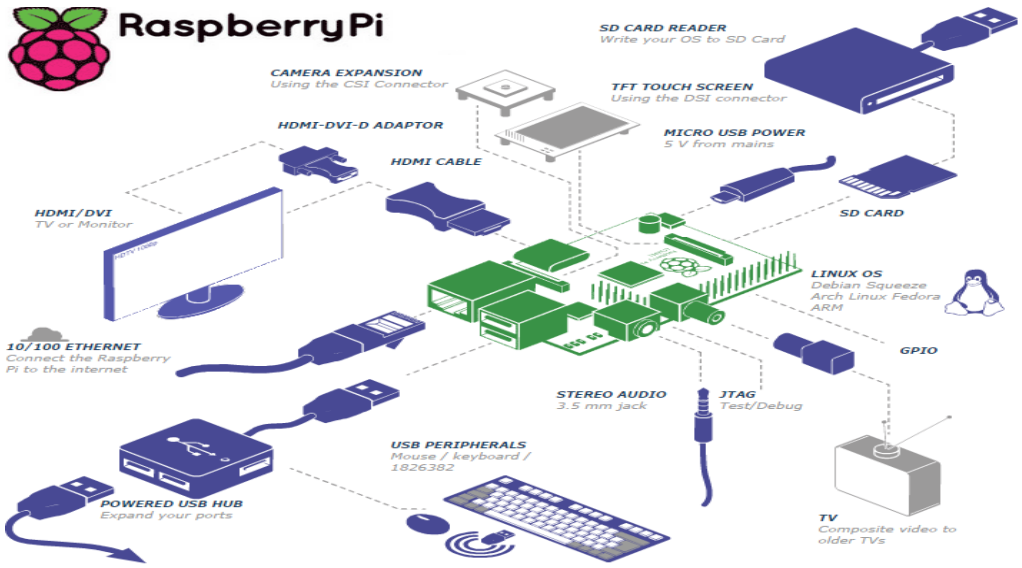


Figura 1.3 Componentes externos.

2.3. Sistema operativo Raspbian

El sistema Raspbian es una distribución de sistema operativo GNU/Linux, que es libre basado en Debian para la placa Raspberry PI, la fundación que trabaja con esta herramienta se ha hecho cargo de proporcionar el hardware del modelo B, trabajando junto a la comunidad que ha creado distintos sistemas operativos especialmente basados en Linux para el manejo de aplicaciones Opensoftware.

Estos sistemas operativos son cargados desde una simple tarjeta SD, creando hasta el momento diferentes sistemas operativos dentro de los cuales RASPBIAN Debían Jessie, RASPBIAN Lite Debian, Jessie, PIDORA Fedora Remix, OSMC, OPENELEC (Herrmann & Salgado, 2014).

2.4. Nagios

Nagios es una herramienta GPL (General Public License) de monitoreo que permite el control de los servicios, procesos y recursos de equipos de red, está escrito en lenguaje C y su licencia que lo determina como Software libre, asegura que siempre se tendrán actualizaciones disponibles junto a una gran comunidad de desarrolladores que lo soportan.

Esta herramienta OpenSource corre bajo SO Linux, funcionando correctamente también en SO UNIX, siendo una alternativa para la gestión de infraestructuras TIC empresariales, ya que consiste en un funcionamiento con arquitectura cliente-servidor, verificando los recursos (con agente) y servicios (sin agente), informando proactivamente a administradores de red o clientes finales de posibles problemas de red y servicios.

Cuando es detectado un error Nagios es capaz de alertar a contactos administrativos sobre diferentes modos de comunicación para informar del estado del servicio que ha provocado el error, incluyendo informes de estado.

2.4.1. Características Nagios

- Monitoreo de Servicios de Red
- Monitoreo de Host y sus recursos como CPU, Memoria, Discos, etc.
- Desarrollo de Plugins para el chequeo de una infinidad de plataformas y servicios (muchos de ellos por aporte de la comunidad de Nagios)
- Capacidad de Services Checks en paralelo
- Capacidad de Definir Host/Servicios padres o hijos, lo que permite detectar el origen del problema en caso de no ser de la propia máquina (Ejemplo: la caída de un server por la falla de un Switch)
- Definición de Contactos para el envío de notificaciones.
- Maneja los eventos de manera proactiva
- Log de eventos
- Interface Web para la visualización de estados de servicio, históricos, Archivo de Log, etc.
- Integración con herramientas que la comunidad ha desarrollado
- Multiplataforma, fue desarrollado originalmente para correr sobre Linux
- Supervisión Continua de la plataforma de TIC esto permite mejorar los SLA y mejorar los tiempos de disponibilidad (Gonz, 2015).

2.5. Arduino UNO

Es una Placa que maneja plataforma de hardware libre, basada en un micro controlador en entorno de desarrollo, fue diseñada para facilitar el uso en los proyectos electrónicos en lenguaje C y C++ con interfaz USB que emula la comunicación serial para más practicidad, la placa de Arduino que se muestra en la Figura 1.4, cuenta con una memoria, conexiones de entrada/salida, pines digitales y analógicos, realiza instrucciones almacenadas en un programa de forma cíclica.

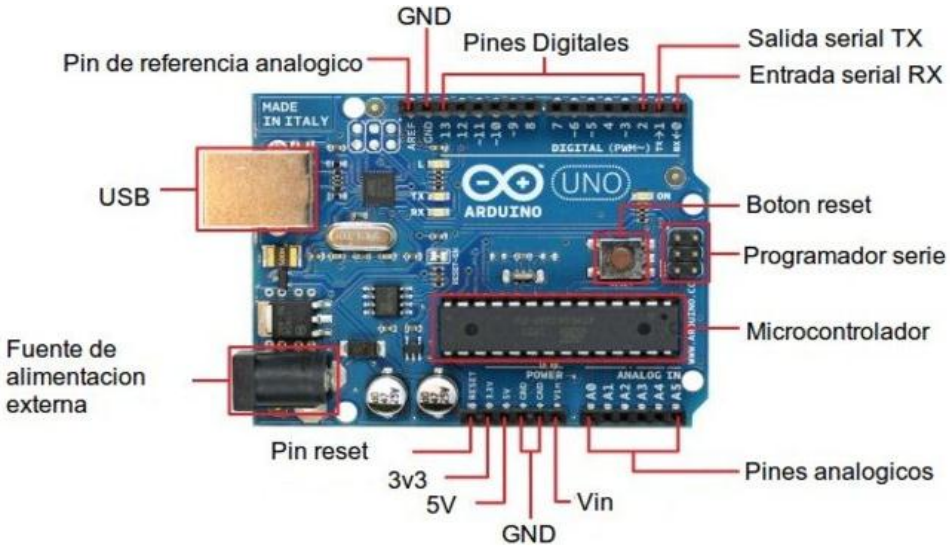


Figura 1.4 Arduino UNO.

2.5.1. Componentes de la placa Arduino uno

Entradas y salida: El micro controlador recibe información de las entradas (read), la procesa y escribe un 1 o un 0 (5v o 0v) en las salidas (Write), actuando sobre el dispositivo que tenemos conectado

Botón de reset: Permite reiniciar el programa y cargar uno nuevo

Pines de entrada y salida: pines de entrada y salida: Permiten conectar elemento que dan información y crean actuaciones.

Puertos USB: Se cargan las instrucciones a ejecutar, el programa que es realizado en el entorno de programación de Arduino (Rojas & Puentes, 2013).

2.6. Modulo Bluetooth HC-05

Este dispositivo cuenta con una configuración de comunicación bidireccional para una conexión punto a punto, la cual se puede acceder a la configuración por

medio del pin KEY que da ingreso a los comandos AT para realizar las respectivas configuraciones para la comunicación Bluetooth y Arduino.

Este módulo HC-05 como se muestra en la Figura 1.5 tienen dos pines extra (además, de TX, RX, VCC, GND) esta etiquetado como “Key” y “State”.

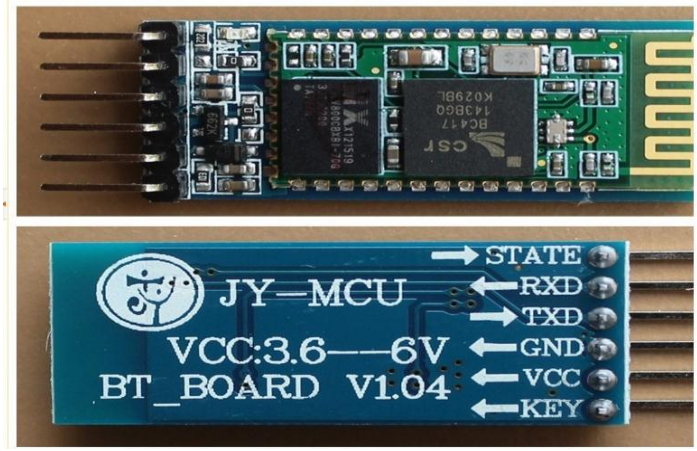


Figura 1.5 Bluetooth HC-05.

- Especificación: Bluetooth v2.0 + EDR (Enhanced Data Rate)
- Puede configurarse como maestro, esclavo, y esclavo con auto conexión (Loopback) mediante comandos AT
- Chip de radio: CSR BC417143
- Frecuencia: 2.4 GHz, banda ISM
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Antena de PCB incorporada
- Potencia de emisión: ≤ 4 dBm, Clase 2
- Alcance 5 m a 10 m
- Sensibilidad: ≤ -84 dBm a 0.1% BER
- Velocidad: Asincrónica: 2.1 Mbps (Max.)/160 Kbps, sincrónica: 1 Mbps/1 Mbps
- Seguridad: Autenticación y encriptación (Password por defecto: 1234)

- Módulo montado en tarjeta con regulador de voltaje y 6 pines suministrando acceso a VCC, GND, TXD, RXD, KEY y status LED (STATE) (Wavesen, 2014)
- Consumo de corriente: 50 mA
- El pin RX del módulo requiere resistencia de pull-up a 3.3 V (4.7 k a 10 k). Si el micro controlador no tiene resistencia de pull-up interna en el pin Tx se debe poner externamente.
- Niveles lógicos: 3.3 V. Conectarlos a señales con voltajes mayores, como por ej. 5 V, puede dañar el módulo
- Voltaje de alimentación: 3.6 V a 6 V (Wavesen, 2014).

2.7. Sensor temperatura y Humedad DHT-11

Es un dispositivo que nos permite medir la temperatura y la humedad relativa, como se muestra en la Figura 1.6 conectado a pines digitales, el DHT-11 lee números enteros, sus características se muestra en la Tabla 1.2 especificando cada parámetro del sensor.

Cuenta con un pequeño micro controlador interno para hacer el tratamiento de la señal para pasar a capturar y luego mostrar los datos obtenidos por el dispositivo (Uk, 2010).

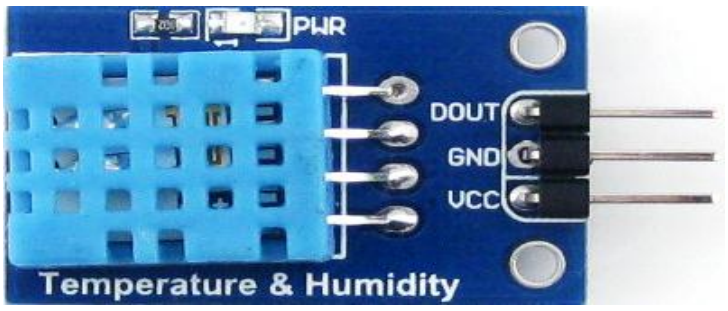


Figura 1.6 Sensor DHT-11.

Tabla 1.2 Parámetros sensor DHT 11.

Parámetro	DHT11
Alimentación	3Vdc ≤ Vcc ≤ 5Vdc
Señal de Salida	Digital
Rango de medida Temperatura	De 0 a 50 °C
Precisión Temperatura	±2 °C
Resolución Temperatura	0.1°C
Rango de medida Humedad	De 20% a 90% RH
Precisión Humedad	4% RH
Resolución Humedad	1%RH
Tiempo de sensado	1s
Tamaño	12 x 15.5 x 5.5mm

2.8. Lenguajes de programación

2.8.1. Lenguaje Python

Este lenguaje de programación hace referencia a la organización y sintaxis, que favorece el código legible, como se muestra en la Figura 1.7 es de código abierto, administrado por la organización sin fines de lucro Python Software Foundation el intérprete de Python puede extenderse a nuevas funcionalidades de tipos de datos implementados en C o C++ u otros lenguajes accesibles desde C. Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizas.

Además, soporta múltiples lenguajes, incluyendo programación orientada a objetos *(class),* tiene un sistema de tipiado dinámico y manejo automatizado (Summerfield, 2013).

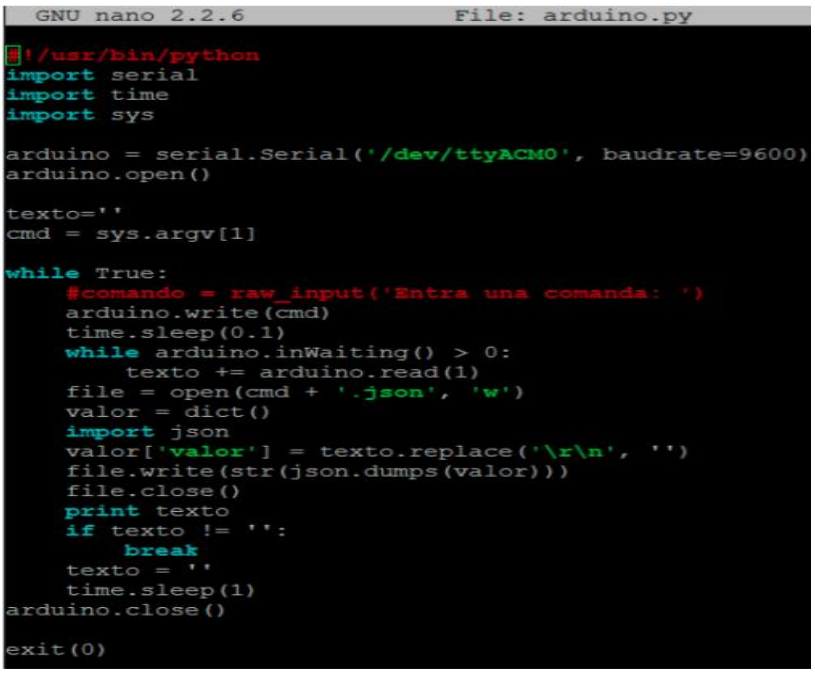


Figura 1.7 Lenguaje programación Python.

2.8.2. Lenguaje Bash

Es un programa informático, que está basado en el Shell de Unix. Cuya función consiste en interpretar órdenes, en un lenguaje de programación en consola. (Campanelli, 2009).

2.8.3. Lenguaje C

Este lenguaje ofrece control de flujo con estructuras sencillas con un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo que no está especializado en ningún tipo de aplicación, este lenguaje ha sido estrechamente ligado al sistema operativo UNIX (Esteban, 2014).

2.9. Sistema Android

Es un sistema operativo para dispositivos móviles como tablets, celulares, PDAs, entre otros, desarrollado principalmente por Google con la filosofía de código abierto por lo tanto cualquier persona puede descargar el código fuente, modificarlo dependiendo las necesidades para compartir los cambios con la comunidad.

La arquitectura de Android tiene estructura en aplicaciones como correo electrónico, programas SMS, calendario, mapas, navegador y contactos entre otros; las librerías están conformadas por un conjunto de bibliotecas de C/C++, administrador de superficies, medios de framework, SQLite, Open GLjES, FreeType, Webkit, SGL, SSL, LBC; Runtime conformado por el núcleo de librerías basadas en Java y la máquina virtual, Dalvik permitiendo que cada aplicación Android corra su propio proceso con su propia instancia (Gavilema y Mullo, 2014).

2.9.1. Dispositivo móvil

Este equipo Moto G cuenta con características que se muestra en la Figura 1.8 que es usado para lograr el balance ideal para el manejo de la aplicación app inventor y Arduino, mostrando los datos deseados, con especificaciones de procesador Quad Core Snapdragon 400, 1GB de RAM, 8GB de almacenamiento y actualización a Android kit kat (Motorola, 2014).

Figura 1.8 Características equipo Móvil.

2.10. App Inventor

Este Software de programación está orientado a los eventos de forma visual, como se muestra en la Figura 1.9, permitiendo al usuario elaborar aplicaciones para Android mediante el entrelazado de bloques hecho en Java diseñada por Google, disponible en la Web de entorno online de forma gratuita.

Requiere para su funcionamiento la instalación del JDK versión 7 para el entorno App inventor (Raúl C., 2015).

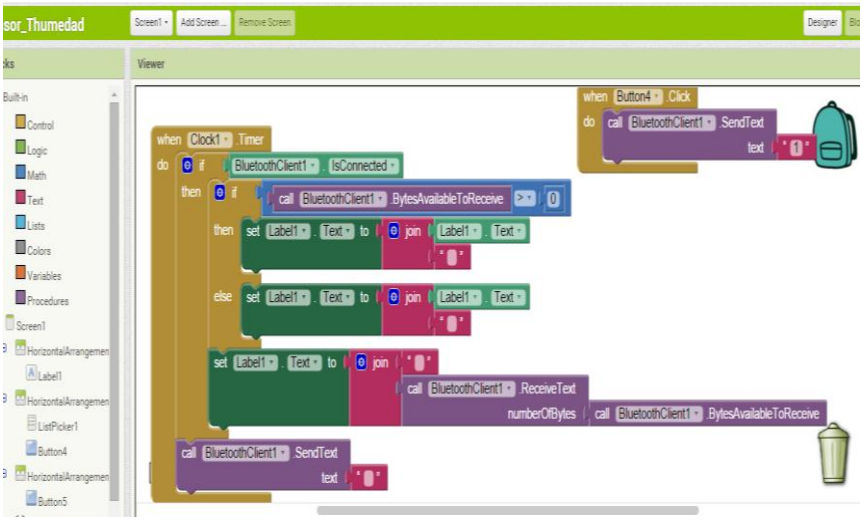


Figura 1.9 Bloques de programación.

2.11. Acceso Remoto de Pc a RPI

Es acceder desde una computadora ubicada físicamente en otro lugar a la RPi, a través de una red local o de forma externa.

En el acceso remoto interviene protocolos, siendo estos un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red, por medio de intercambio de mensajes, y programas en ambas computadoras que permitan recibir y enviar los datos necesarios.

2.11.1. LogMein Hamachi

Es un software que al instalarse se habilita para tener acceso remoto desde cualquier sitio, añadiendo un red para comunicar dos dispositivos en diferentes lugares como se muestra en la Figura 1.10. Las VPN proporcionan un método para conectarse directamente a una red remota como si estuviera en el lugar para así proteger el tráfico de conexión entre la red y cliente, utilizando encriptación para evitar ataques permitiendo a los administradores tener más seguridad en el envío de información (Hamachi, 2016).

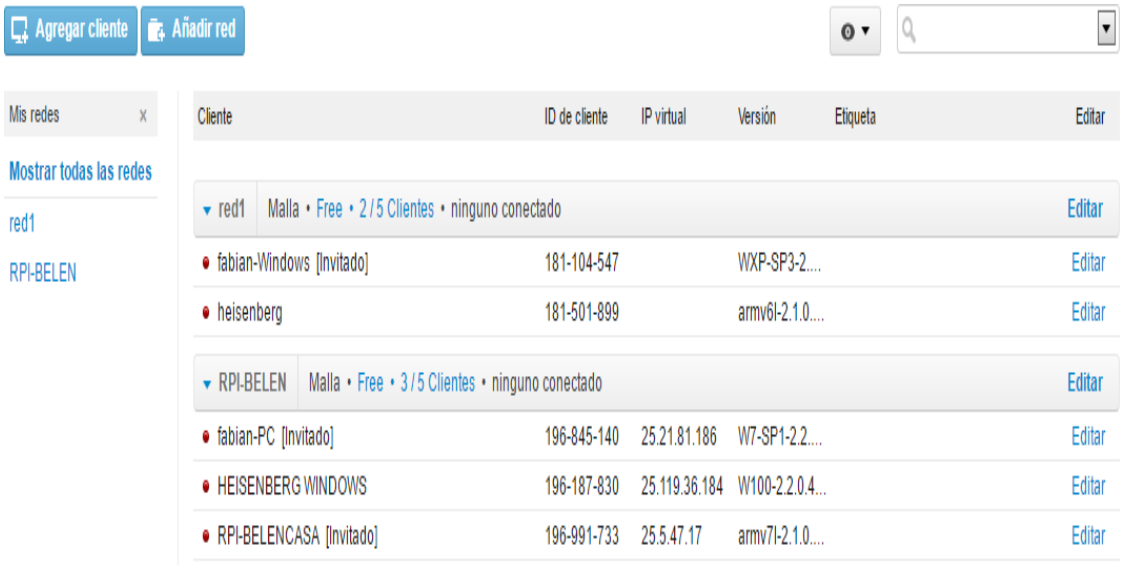


Figura 1.10 Interfaz VPN

2.11.2. Acceso Remoto del Programa XRDP

Este control es una conexión para la Raspberry Pi y poder a través del protocolo RDP establecer una comunicación con el dispositivo desde un equipo bajo Windows, que se muestra en la Figura 1.11 teniendo acceso desde nuestro equipo a la interfaz gráfica de la RPI (Windows, Ken, & Nils, 2013).

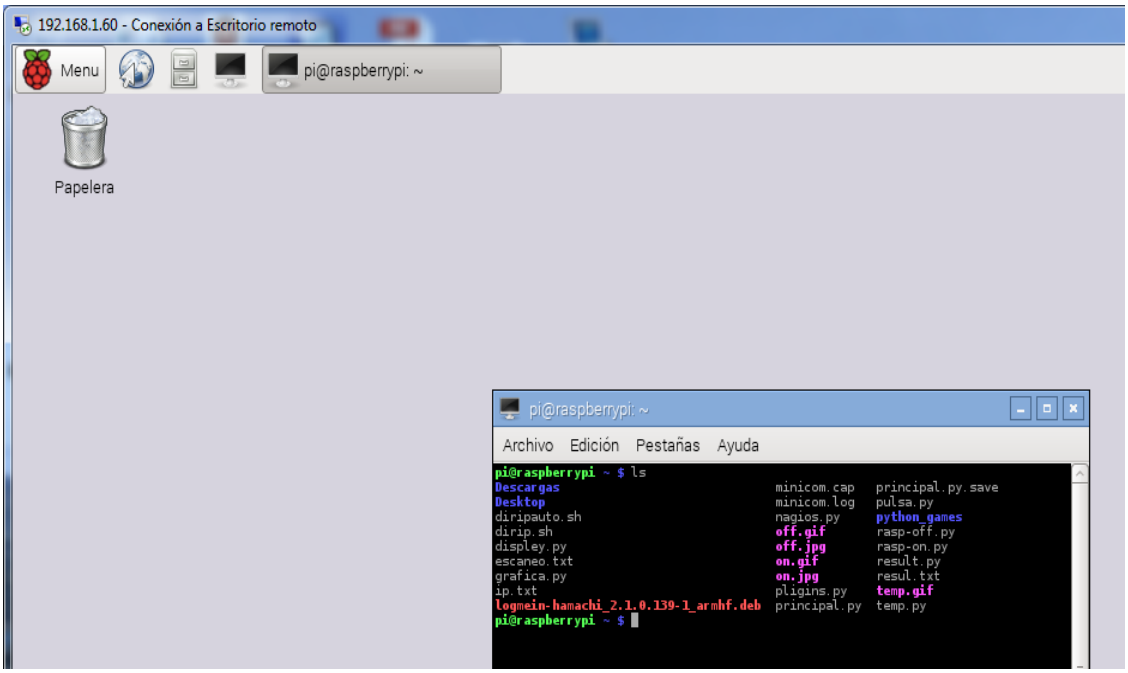


Figura 1.11 Interfaz Rpi desde el equipo Windows.

2.11.3. NRPE (Nagios Remote Plugin Executor)

Es un daemon que permite ejecutar remotamente plugins de Nagios en equipos Linux, estableciendo con el plugin **Check_nrpe** una comunicación protegida con **SSL** instalado en el equipo remoto, y así comprobar con el plugin de Nagios el servicio y recursos de la maquina remota (Xmartic, 2013).

3. Metodología

3.1. Sistema de Monitoreo de Red

3.1.1. Evaluación de la estructura general de Gestión y Monitoreo

Se planteó una estructura general estableciendo posibles herramientas de bajo coste que ayudaran a cumplir con el objetivo de integrar los dos sistemas gestión domótico y monitoreo de red; como se muestra en la

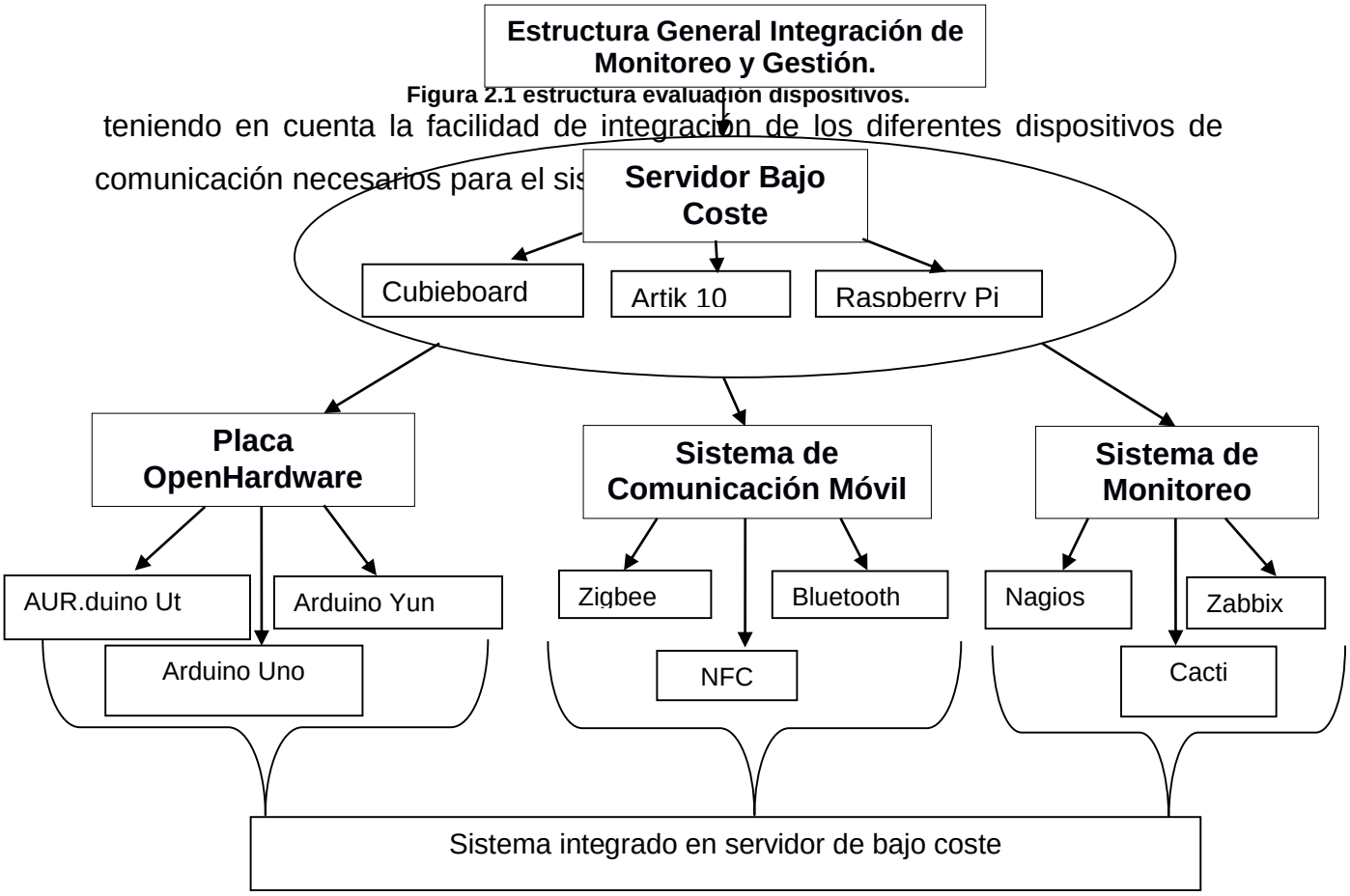


Figura 2.1 estructura evaluación dispositivos.

teniendo en cuenta la facilidad de integración de los diferentes dispositivos de comunicación necesarios para el sis

Figura 2.1 estructura evaluación dispositivos.

3.1.2. Gestión de red con integración de sistema Domótico

Se plantearon diferentes conceptos que ayudaran a tener un entendimiento en la integración de un sistema de monitoreo de redes con un sistema de gestión de sensores y actuadores (sistema domótico) sobre una placa de bajo coste que pueda ser administrada de forma remota.

La placa RPI se adecúo como servidor, cumpliendo con la función principal de integrar y gestionar los dos sistemas, estableciendo una comunicación en bloques, en el que Arduino tiene el control del sensor Dht-11 y la App, los cuales por intermedio del Bluetooth HC-05 reciben y envían la información, además el dispositivo RPI tiene el control para administrar los servicios y recursos locales del equipo bajo la herramienta Nagios como se muestra en la Figura 2.2 especificando los requerimientos necesarios de los dos sistemas que trabajan en un dispositivo reducido y de características limitadas (Arriola Navarrete, 2011).

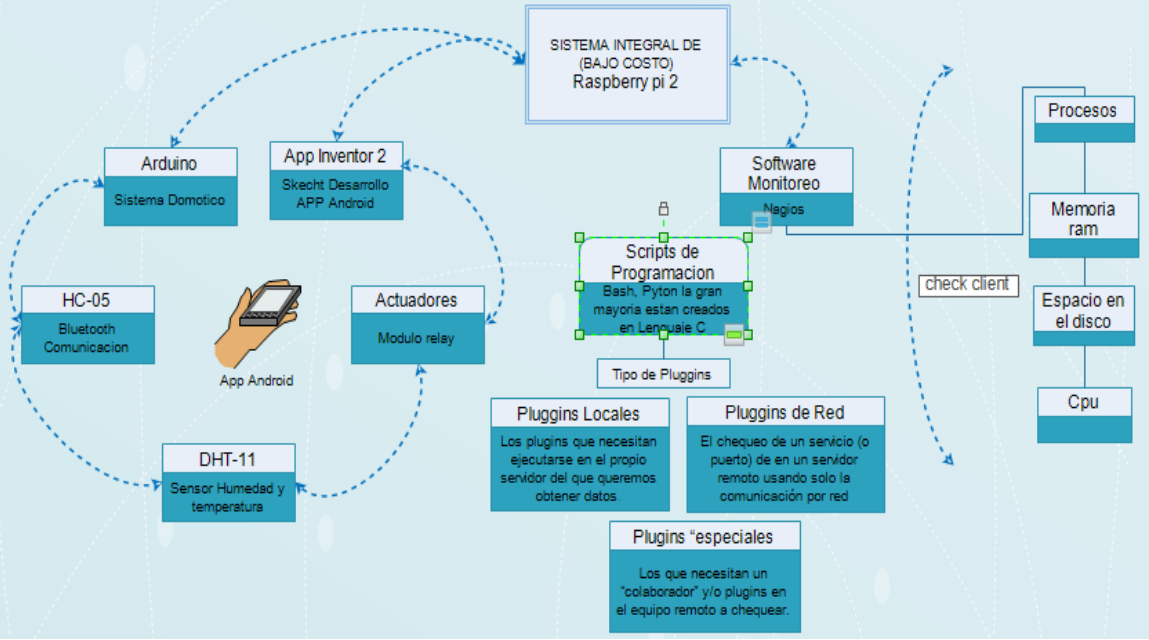


Figura 2.2 bloques de gestión y monitoreo.

3.1.3. Herramientas de monitoreo OpenSource soportadas por la placa de bajo coste

Se estudiaron las diferentes herramientas de monitoreo OpenSource detallando cada una de las características que sea acorde para la arquitectura como se muestra en la Tabla 1.3, que se especifican ser soportadas por la RPI como: Nagios, Zabbix y Cacti. Analizando cada una de las herramientas de monitoreo se establece que Nagios es una herramienta que soporta varios lenguajes de programación (Shell, script, c++,perl, ruby, Python, php, bash, java,) adaptando los plugins para la realización de monitoreo con una interfaz gráfica de fácil gestión, interpretación y acceso remoto.

Tabla 1.3 Evaluación de herramientas OpenSource.

Características	Nagios Core	Cacti	zabbix
Estabilidad de	3.2.0	0.8.gi	1.8.1

versiones			
Sistema operativo	Linux only	Multiplataforma	multiplataforma
Entorno grafico	Si	Si	Si
Snmp	Atreves de plugin	Si	Si
Estadísticas	Si	Si	Si
Auto descubrimiento	Si	Atreves de plugin	Si
Plataforma	Shell, script, c++,perl, ruby, Python, php, bash, java, c, etc..	Bash, perl, php	Php, c
Agentes	Si	No	desconocido
Licencia	Gpl	Gpl	Gpl
Respuesta	Open source los servicios responde de una manera fácil de uso, sacando estadisiticas de los servicios PUBLICOS Y PRIVADOS.	Multi ususario suporta monitoreo de tráfico, tiene un acceso de fácil uso y configuración.	Maneja un agente dedicado buen GUI, Open source
Monitoreo de servicios	Smtp ,pop 3,http,nntp, Icmp,snmp,ftp,tcp,(todo en vía pluging) imap etc.	Snmp	Smtp ,pop 3,http,nntp, icmp,snmp,ftp,tcp imap
Acceso remoto	Ssh, telnet, snmp, ssl	telnet ssh	Snmp ssh
Web aplicación	Total control	Total control	Total control
Genera alertas	Si	Si	Si

3.1.4. Arquitectura y Funcionamiento

En el seguimiento del trabajo se plantea un diseño básico de los dispositivos que intervienen en el montaje de la arquitectura de red mostrado en la Figura 2.3, para la realización del funcionamiento de la integración de monitoreo de red y el

acceso domótico, se adaptó el dispositivo RPI como un sistema central para gestionar, evaluar los servicios por medio de la activación de un terminal con acceso remoto, y así luego controlar y obtener la información más detallada de los agentes instalados en el equipo, junto a un sensor de temperatura y humedad, que captura los datos para luego procesarlos por medio de la placa Arduino e interpretarlos por medio del sistema central y la App que estará instalada en el equipo móvil.

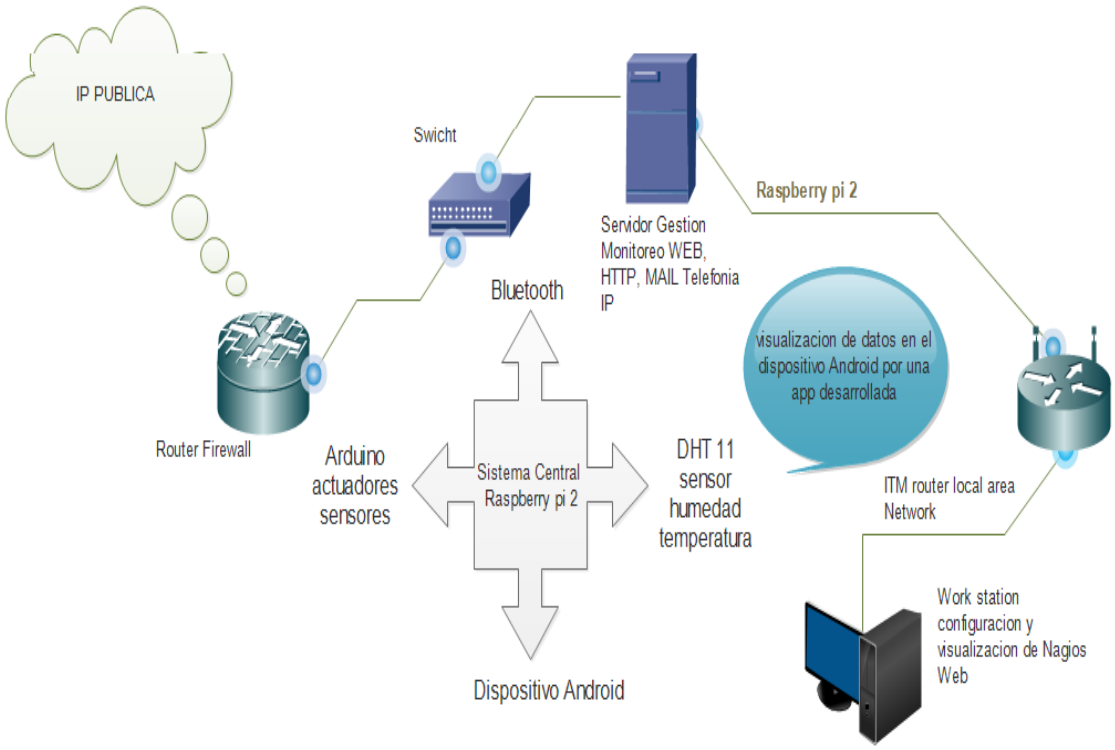


Figura 2.3 Esquema de funcionamiento en la red.

3.1.5. Instalación del S.O Raspbian

En la instalación del Sistema Operativo de la Placa Raspberry PI, se adquirió una microSD de 8 GB, teniendo la imagen ISO del S.O para proceder a extraer y copiar la imagen, una vez terminado el proceso simplemente se debe insertar la imagen iso a la placa para a hacer la configuración.

3.1.5.1. Configuración Raspbian en la placa RPI.

Para el asistente de configuración inicial se debe ejecutar la siguiente orden: sudo raspi-config, se despliega un menú en consola con las opciones de configuraciones extendiendo la imagen ISO sobre la micro SD, mostrando las opciones que se presentan en el menú como se muestra en la Figura 2.4 del sistema operativo el cual es útil para configurar las opciones principales:

Activar inicio en modo escritorio: Esta opción permitió configurar la Raspberry Pi iniciando en modo escritorio directamente.
Opciones de internacionalización: Se configuro el idioma de la Raspberry Pi periféricos y el teclado.

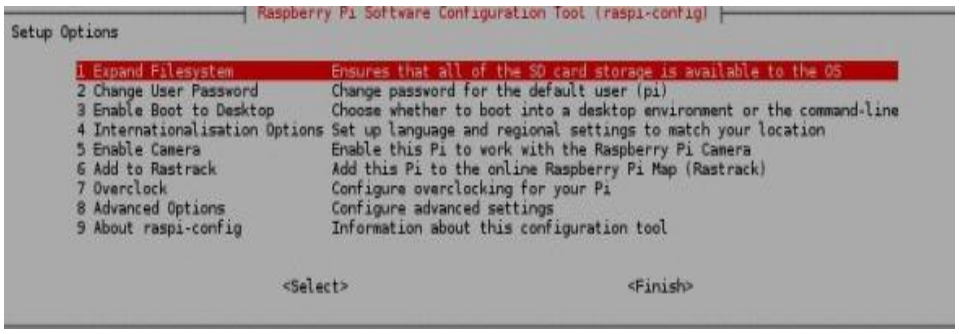


Figura 2.4 Interface de configuración.

3.1.5.2. Contraseña en RPI

El S.O Raspbian viene por defecto con un usuario “pi” y una contraseña “Raspberry”, que permite acceder a modo privilegiado (súper usuario) para modificar los archivos, configuraciones y carpetas del sistema

3.1.5.3. Habilitación y configuración SSH

Habilitando el servicio SSH en la RPI nos permitió acceder y ejecutar los comandos desde la consola-terminal desde nuestro ordenador local, accediendo a los ficheros de configuración mediante la comunicación desde el programa putty para Windows, como se muestra en la Figura 2.5, permitiendo una conexión remota sea desde otro equipo o de una red diferente, accediendo de forma segura al servidor de monitoreo y guardar la configuración sin ningún problema.

Se utilizó el siguiente comando para la instalación:

sudo apt-get install ssh

Se inicia el servicio con los siguientes comandos:

sudo etc/ini.d/ssh start

sudo service ssh start

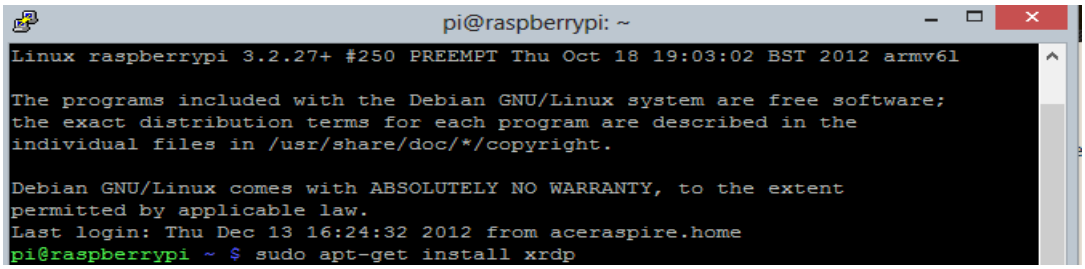


Figura 2.5 RPI por Aplicación PUTTY.

3.1.6. Configuración de la aplicación para conexión con acceso a escritorio Remoto

El escritorio remoto en Windows es una aplicación instalada por defecto, ingresando la IP de la RPI, usuario y contraseña, con usuario “pi” y la contraseña por defecto, "Raspberry", como se muestra en la Figura 2.6, todo con el objetivo de poder visualizar la interfaz gráfica del servidor de gestión y monitoreo desde cualquier equipo, realizando la instalación en la placa RPI desde un terminal por

el servicio SSH, utilizando el siguiente comando para la configuración (Chacon M., 2014).

sudo apt-get install xrdp

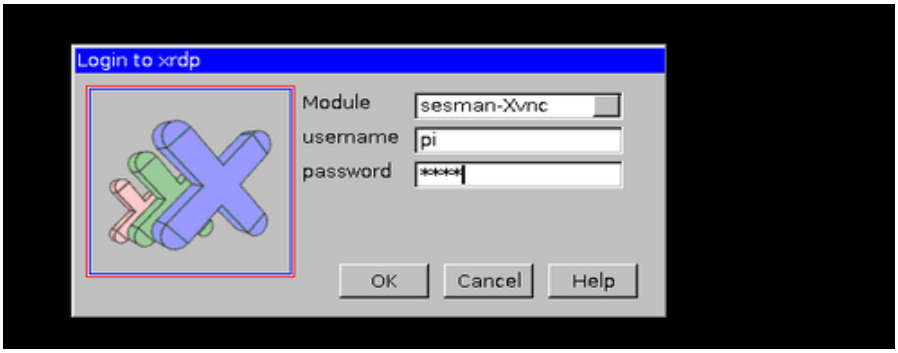


Figura 2.6 Conexión Remota.

3.1.7. Instalación de la aplicación de acceso remoto Hamachi

Se hace la configuración e instalación para la comunicación VPN de la RPI para acceder desde una red diferente, de forma remota y poder verificar el estado de conexión de la red y servicios.

Los comandos de configuración que se deben instalar en la consola para nuestro servidor de Gestión y monitoreo se detallan a continuación:

"instalación comandos iniciales para conexión Vpn"

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install lsb

Descargando el Hamachi para la configuración

sudo wget https://secure.logmein.com/labs/logmein-hamachi_2.1.0.139-1_armhf.deb

"instalación Hamachi"

sudo dpkg -i logmein-hamachi_2.1.0.139-1_armhf.deb

sudo /etc/init.d/logmein-hamachi start

"instalación en el archivo chkconfig"

sudo apt-get -y install chkconfig

sudo chkconfig -s logmein-hamachi 2

"iniciando y parando el servicio en Hamachi"

sudo service logmein-hamachi stop
sudo service logmein-hamachi start

iniciando Login Hamachi

sudo hamachi login

"ingresar la cuenta de logmain para la cuenta de correo electrónico"

Verificando email

sudo hamachi attach $email

"ingresar el identificador nickname de RPI "

sudo hamachi set-nick $nick

" **La RPI debe estar ahora en la red hamachi "**

3.1.8. Instalación y configuración de Nagios

Con esta herramienta de monitoreo nos permite observar los equipos conectados que están en una red, por medio de la interfaz gráfica web del sistema Nagios sobre la RPI, visualizando los equipos configurados al escribir la dirección Ip del

servidor así como se muestra en la Figura 2.7, para verificar la red que se está monitoreando junto a los servicios locales y públicos del servidor.

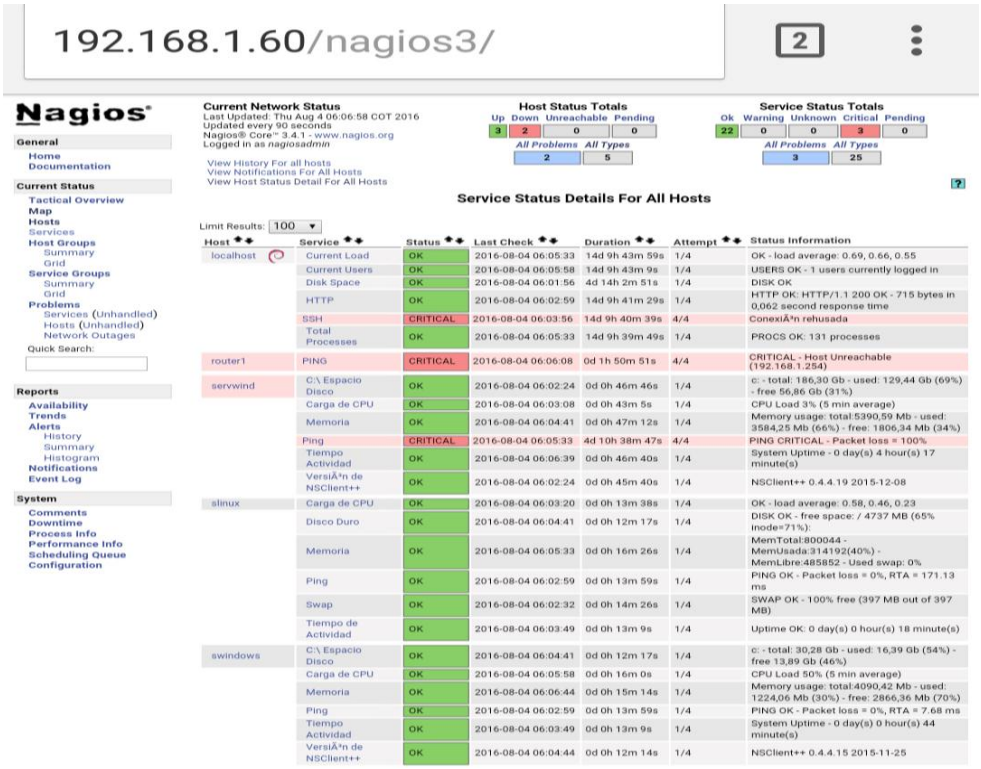


Figura 2.7 Interfaz Nagios.

3.1.8.1. Propiedades de los servicios en Nagios

Se especificó la función que cumple cada una de estas configuraciones, que se tendrán en cuenta para más adelante en el fichero de instalación de Nagios.

Use: Indica el tipo de servicio, usaremos “generic-service”.
Host_name: Establece el host o hosts (separados por comas) a los que afectará este servicio.
Service_description: Texto descriptivo de la funcionalidad del servicio.

Check_command: Es el comando que se lanzará la medición del servicio del host, los comandos predefinidos están en la **Ruta /usr/lib/nagios/plugins**: Rutas de acceso a los ficheros de programación.
Check interval: El tiempo transcurrido entre cada ejecución del comando.
Retry check interval: El tiempo que transcurre desde que hay una ejecución del comando fallida y el siguiente intento.

Los archivos de configuración que fueron necesarios y se utilizaron para la instalación dentro de los ficheros, está en la raíz que se encuentra en la ruta: **/usr/local/nagios/etc./objects**

Rutas y ficheros de configuración:
commands.cfg
contacts.cfg
localhost.cfg
printer.cfg
switch.cfg
templates.cfg
windows.cfg

La instalación usará comandos básicos **apt-get**, comenzando con una actualización:

sudo apt-get upgrade
sudo apt get update

Luego pasamos instalar nagios3 con los siguientes comandos:

sudo apt-get install nagios3 nagios-plugins nagios-nrpe-plugin
sudo apt-get install nagios-nrpe-server nagios-plugins

En la instalación se configuro los scripts dentro de los archivos que tiene la información para los equipos y servicios que se van a monitorear, ahora se accede de forma web con el usuario “nagiosadmin” y con contraseña 123456, para ver la interfaz de la red en el dispositivo, descubriendo el propio servidor que se integra automáticamente en el panel de Nagios.

En la configuración se añadieron equipos de forma organizada en ficheros según su tipo switches, routers y equipos de trabajo, en un fichero llamado; **switch.cfg** y para los equipos Windows en otro **windows.cfg.**

Como estamos conectados a nuestro propio router, para el enrutamiento, creamos el fichero **switch.cfg** en la ruta**: /etc/nagios3/objects/switch.cfg** con las siguientes líneas de configuración de los procesos, que se detallaran en la interfaz web de NAGIOS para ver el estado de los servicios.

```
# HOSTS DEFINITIONS
# ROUTER 1
define host {
        use generic-host
        host_name router1
        display_name Router1
        address 192.168.1.1
        check_period 24x7
        check_interval 5
        retry_interval 1
        max_check_attempts 10
}

# SERVICE DEFINITIONS
define service {
        use generic-service
        host_name router1
        service_description PING
```

check_command check_ping!200.0,20%!600.0,60%
normal_check_interval 5
retry_check_interval 1
}

Ahora que hemos añadido un nuevo host con un nuevo servicio en el archivo **switches.cfg**, le decimos a Nagios que verifique ese fichero al iniciar el servicio, que se editó en **nagios.cfg** situado en la ruta. **/etc/nagios3/nagios.cfg**, quitando el signo **(#)** de la línea que es **sudo nano /etc/nagios3/objects/switch.cfg**

En el primer segmento de código ya añadimos el host (router) y en el segundo que se creó, es un servicio que hace ping cada 5 minutos, si falla, lo intenta cada minuto, devolviendo una alerta que nos dice si esta crítico o no, se añade esta ruta para colocar el servicio de monitoreo activo y ver el servicio ICMP.

cfg_file=/etc/nagios3/objects/switch.cfg

Ahora para que la configuración tenga efecto y muestre el servicio que se escribió, se reinició el servicio Nagios3 para que actualice la herramienta de monitoreo (Gonz, 2015).

sudo service nagios3 restart

3.1.8.2. Configuración del agente Nagios en Linux

Los servicios se catalogan públicos y privados, definiendo los públicos como los que adquieren la información del servidor, y los privados como lo que obtienen las características o recursos de un ordenador de trabajo, como son el uso de la memoria, la carga de la CPU, y el espacio de disco duro, así tal como se muestra en la Figura 2.8 monitoreando los servicios específicos, entre el servidor Nagios y el equipo que trabaja para enviar la información de cómo se comportan los recursos de nuestros ordenadores.

En el monitoreo de equipos Linux como los de Windows, cambia en algunas configuraciones como la definición de los servicios, pero la estructura de configuración es la misma.

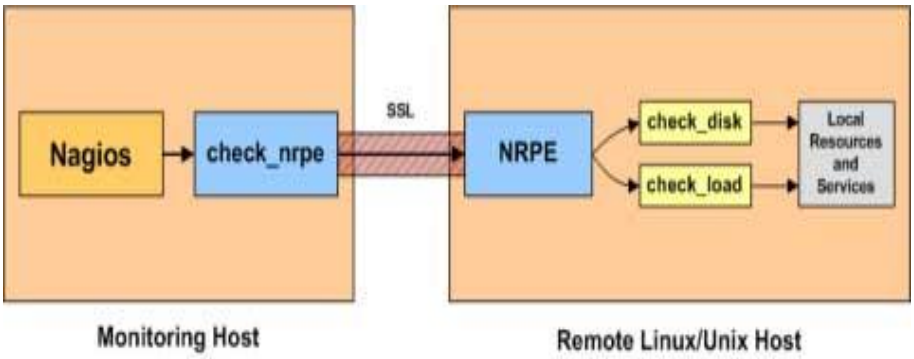


Figura 2.8 Esquema de plugin NRPE.

En el servidor Nagios solo es necesario instalar "**check_nrpe**", que provee el paquete para la instalación y la verificación de recursos del equipo que se monitoreo, para ejecutar y desplegar el proceso de deamon NRPE y así ver qué servicios están instalados y listos para mostrar el estado de consumo.

Los pasos de configuración son:

Servidor Nagios en RPI: Instalar check_nrpe.

Servidor Monitoreo: En Nagios la Raspberry pi después de haber verificado la correcta instalación de cada una de los plugins, se procede a modificar el fichero de configuración, que se describe a continuación:

Modificando el archivo **Linux.cfg** que es la ruta de instalación

3.1.8.3. Monitoreo remoto NAGIOS desde un equipo Linux

En el equipo remoto con sistema Linux se realizó una modificación en el archivo de configuración, especificando la ruta **nrpe.cfg allow_host=ip** que muestra en la Figura 2.9 tomando la dirección IP del servidor de monitoreo

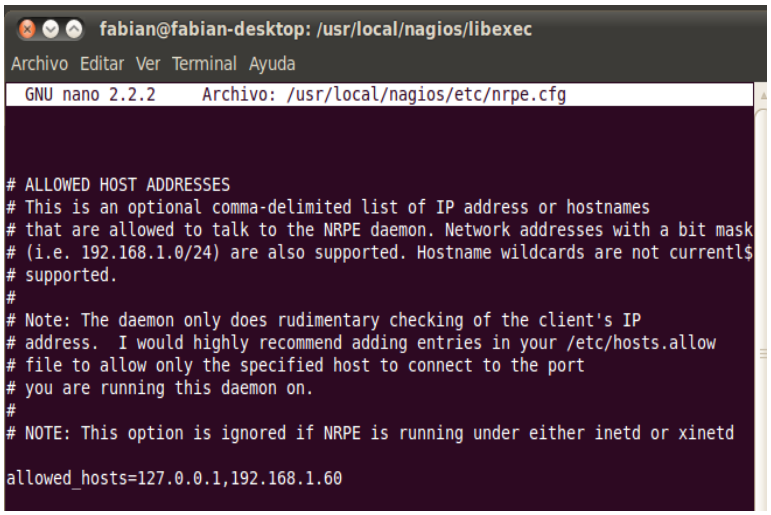


Figura 2.9 Ruta de configuración.

Ahora para activar los servicios se escribió los plugins que se definen con las líneas de código para editar los **comandos check**, y poder monitorear los recursos del equipo remoto que se muestra en la Figura 2.10, publicando los recursos y resultados que nos arrojó el consumo del sistema Linux, que se observaron desde el servidor, y desde del equipo remoto en la ruta que se escribe a continuación (Xmartic, 2013).

/usr/local/nagios/libexec/

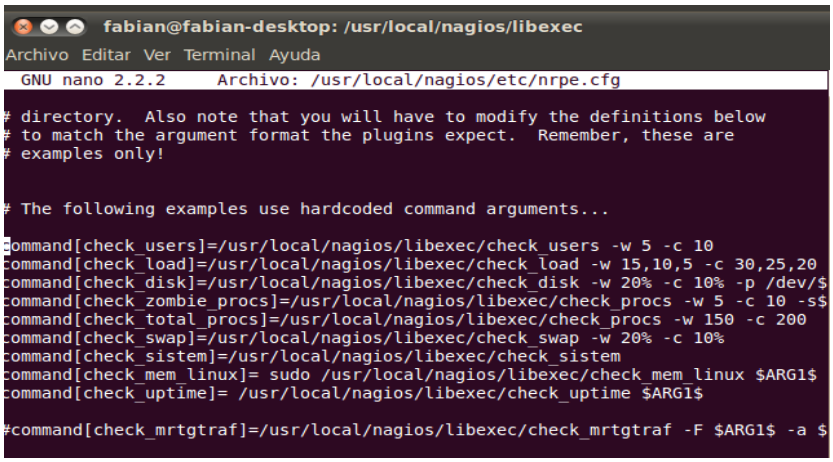


Figura 2.10 Fichero de configuración.

3.1.9. Monitoreo equipos con el agente NSclient++ con Windows

En el trabajo se utilizó un agente NSClient++ para equipos Windows, compatible con la herramienta Nagios, para la verificación de los servicios locales del equipo, como se muestra en la Figura 2.11 indicando la estructura de monitoreo para este sistema.

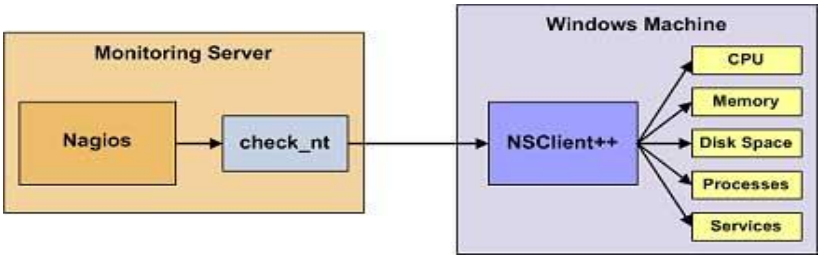


Figura 2.11 Estructura Monitoreo de servicios Windows.

Para la configuración se coloco la dirección 192.168.1.55 que trabaja con la ip del servidor Nagios, y el password que es Colombia 2015 instalando NSClient++, en el ordenador de Windows que se muestra en la Figura 2.12.

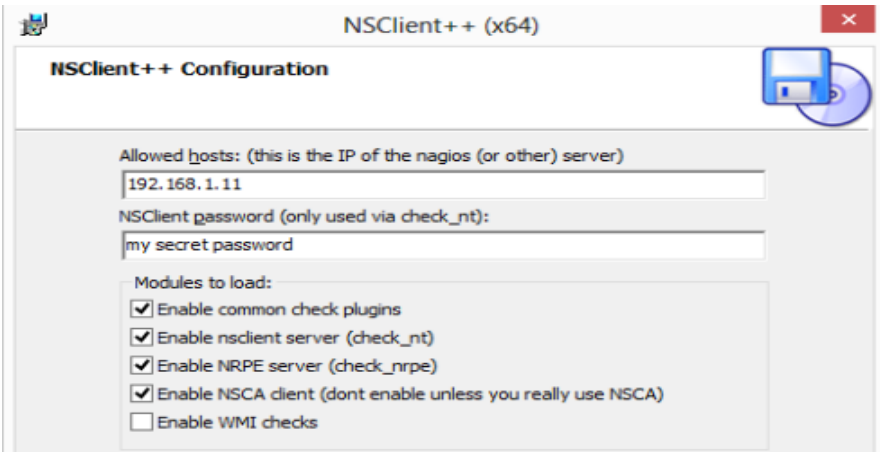


Figura 2.12 Interfaz configuración Nsclient++.

Para el monitoreo se ubica un computador con Windows para que lo detecte Nagios y se verifiquen los servicios, para luego separar los diferentes tipos de host o equipos por ficheros de configuración, para cada uno creando la ruta que se especifica;

/etc/nagios3/objects/windows.cfg en donde especificaremos la definición de un equipo Windows y sus servicios, para crear el fichero desde el terminal usamos: **sudo nano /etc/nagios3/objects/windows.cfg**

Añadimos el siguiente contenido, dándole nuestra dirección ip del servidor

```
# host definitions
define host {
        use windows-server
        host_name miservidor
        alias Mi Servidor Windows
        address 192.168.1.55
}
# service definitions
# Obtenemos la versión instalada de NSClient++
define service {
        use generic-service
        host_name miservidor
```

```
        service_description Versión de NSClient++
        check_command check_nt!CLIENTVERSION
}
# Carga de CPU
define service {
        use generic-service
        host_name miservidor
        service_description Carga de CPU
        check_command check_nt!CPULOAD!-l 5,80,90
}
#Carga Memory
define service{

use                             generic-service

host_name                       Windows

service_description             Memory Usage

check_command                   check_nt!MEMUSE!-w 80 -c 90 }

#Disco Duro
define service{

        use                     generic-service

        host_name                       winserver

        service_description     C:\ Drive Space

        check_command                   check_nt!USEDDISKSPACE!-l c -w 80 -c 90

        }
```

Este servicio se define en el Host name (miservidor) y junto agregamos dos servicios, y obtendremos la carga de CPU y la Versión del agente que tienen instalado, los dos utilizando el comando **check_nt**.

El valor del campo "**host_name**" de la definición de servicio indica a que equipos afecta el servicio, y colocamos como nombre "miservidor" como también tenemos que modificar el fichero **/etc/nagios3/nagios.cfg** para quitar el numeral (#) de las siguientes líneas de los archivos de configuración:

cfg_file=/etc/nagios3/objects/windows.cfg

cfg_file=/etc/nagios3/objects/templates.cfg

3.1.9.1. Contraseña en el NSClient++

En la instalación del agente NSClient++ en windows es necesario especificar los comandos, para el fichero “check_nt” e incluya las credenciales editando dentro de la ruta **/etc/nagios3/commands.cfg** para después abrir la ruta de nuestra configuración.

sudo nano/etc/nagios3/commands.cfg

Con la contraseña “123456” que se establece en NSClient++, como se muestra en la Figura 2.13 para monitorear los servicios de Windows.

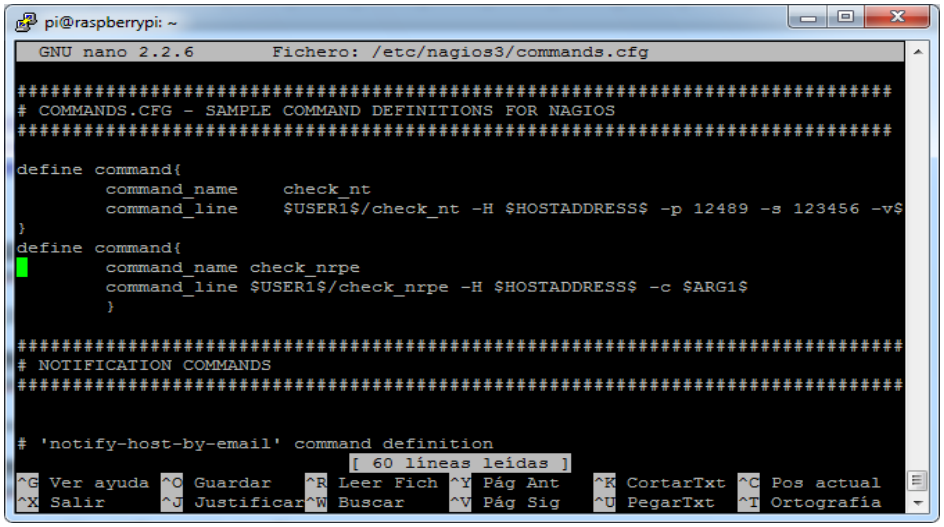


Figura 2.13 Fichero de Configuración.

define command{

command_name check_nt

command_line$USER1$/check_nt -H $HOSTADDRESS$ -p 123456 -s PASSWORD -v $ARG1$ $ARG2$

}

Reiniciamos el servicio nagios3 para una actualización y tenga efecto el cambio usando la línea de comandos; **sudo service nagios3 restart**

3.2. Sistema Domótico.

3.2.1. Introducción Servidor RPI y Arduino

Este dispositivo se encargara de integrar el sistema domótico con Arduino, el cual se encarga de hacer un despliegue de control de sensores y actuadores, dado que nuestro servidor de gestión y monitoreo RPI es limitado en cuanto a especificaciones de rendimiento, pero cabe resaltar que tiene que repartir los procesos para realizar este tipo de tareas, para ello se coloca un cable USB para conectar y hacer la comunicación serial para la programación en la herramienta Arduino, el cual se encarga de controlar los sensores de entrada y actuadores de salida, para que en función del programa y lectura de los sensores se produzca una serie de estados. Ver anexo A.

3.2.1.1. Instalación y configuración el IDE de Arduino

Se descargó el programa de instalación con la última versión en la página https://www.arduino.cc/ para después descomprimir el fichero de instalación, y la placa entre en comunicación con el programa instalado en el ordenador.

Se conectó la placa y se inicia con una verificación de "USB Serial Converter" para la comunicación bidireccional, que tendrá la alimentación y la compilación del programa que se subirá a la placa.

Después se hizo la selección del puerto serie en el dispositivo de la placa Arduino en el menú Tools Serial Port (Herramientas | Puertos Serie COM1) para la comunicación de los puertos serie del hardware entre la placa Arduino y el ordenador.

3.2.2. Configuración Del Bluetooth Hc-05 al Arduino

Se estableció con una conexión entre los pines de transmisión y recepción cruzada entre Arduino y Bluetooth como se muestra en la Figura 2.14, además se requiere del cable USB para acceder al módulo HC-05 en modo comandos AT por intermedio del Arduino, para realizar cambio de contraseña, comunicación inalámbrica, velocidad de transmisión y configuracion del estado (maestro/esclavo).

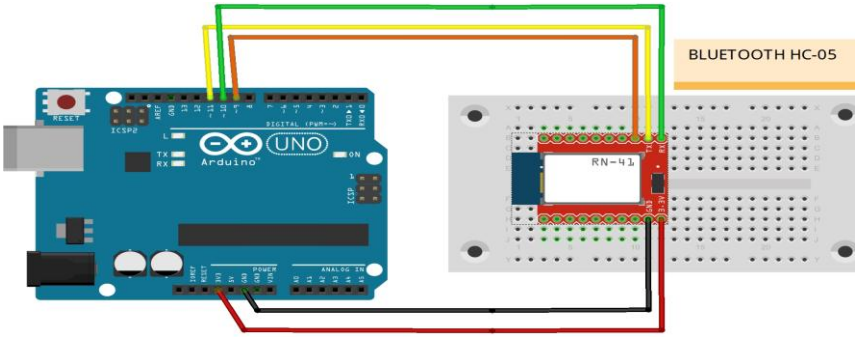


Figura 2.14 Diseño conexión Arduino Bluetooth.

3.2.2.1. El programa de control en el sketch

Se conectó el modulo Bluetooth HC-05 desde el pin digital 9 a 5VDc, y el pin digital 8 para forzarle a entrar en el modo comandos AT, para programar en el sketch y compilar en el Arduino UNO, y entrar a modo configuración.

```
#include <SoftwareSerial.h>
SoftwareSerial BT1 // RX | TX
void setup()
{ pinMode(8, OUTPUT);       // Al poner en HIGH forzaremos el modo AT
    pinMode(9, OUTPUT);       // cuando se alimente de aqui
    digitalWrite(9, HIGH);
    delay (500) ;             // retardo antes de encender el modulo
    Serial.begin(9600);
```

```
    Serial.println("Levantando el modulo HC-05");
    digitalWrite (8, HIGH);    //Enciende el modulo
    Serial.println("Esperando comandos AT:");
    BT1.begin(34800);
  }
void loop()
  {  if (BT1.available())
          Serial.write(BT1.read());
     if (Serial.available())
          BT1.write(Serial.read());   }
```

Con este procedimiento se garantizó que el modulo HC-05 entra en modo comandos AT, y en la consola es donde se programó. En el dispositivo por medio de consola se enviara un mensaje de Status en el momento que inicia en el modo AT, y para verificar que está listo y ser configurado.

En la figura 2.15 muestra el ingreso de comandos para la configuración del Bluetooth HC-05 en el puerto serial COM4, y verificar los parámetros Dirección MAC, password, Tipo de comunicación, paridad par/impar, o si esta Maestro/ esclavo.

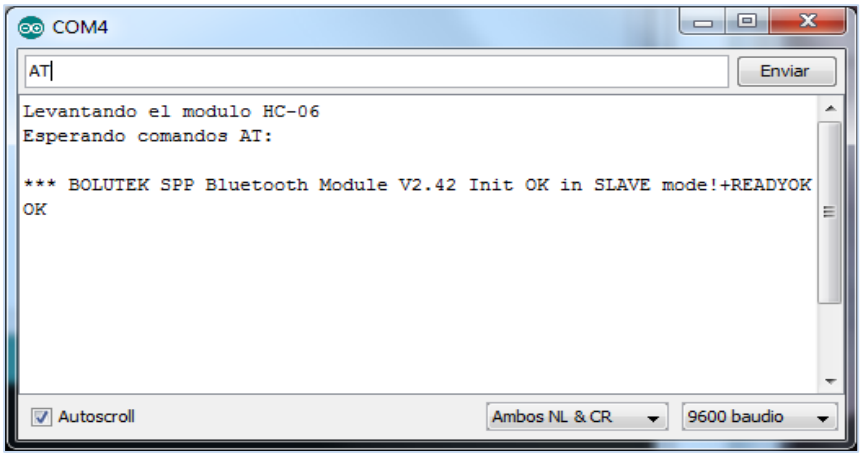


Figura 2.15 Comandos AT Configuración.

3.2.2.2. Comandos AT Utilizados

Comando AT es usado para testear el modulo y ver si está funcionando.

- AT+RESET Comando AT+RESET se encarga de resetear el modulo al estado de inicio.
- AT+ROLE? Comando AT+ROLE? averigua el modo de el modulo, 0 = Esclavo, 1 = Maestro , 2 = Esclavo Bucle.
- AT+ROLE= Comando AT+ROLE= es usado para configurar el modulo como maestro o esclavo, 0 = Esclavo, 1 = Maestro, 2 = Esclavo-Bucle. Tú tienes que configurar un módulo como maestro y el otro como esclavo.
- AT+CMODE? Comando AT+CMODE? averigua el modo de enlace. 0 = Dirección de enlace bluetooth específica, 1 = Cualquier dirección de enlace bluetooth, 2 = Esclavo – Bucle.
- AT+CMODE= Comando AT+CMODE= es usado para configurar el modo de enlace del módulo. . 0 = Dirección de enlace bluetooth específica, 1 = Cualquier dirección de enlace bluetooth, 2 = Esclavo – Bucle. Es mejor configurar el modo a “1” si sólo tienes un dispositivo par, porque es más fácil hacer esto en par.
- AT+PSWD? Comando AT+PSWD? averigua el password de el modulo.
- AT+PSWD= Comando AT+PSWD es usado para configurar el password de aparejamiento.
- AT+UART? Comando AT+UART? averigua la configuración de baudios UART (TX y RX).
- AT+UART= Comando AT+UART= es usado para asignar la velocidad de TX y RX del módulo.
- AT+NAME? Comando AT+NAME? averigua el nombre asignado al módulo.
- AT+NAME= Comando AT+NAME= asigna un valor al módulo bluetooth.

- AT+ADDR? Comando AT+ADDR? averigua la dirección MAC del dispositivo.

3.2.2.3. Interfaz de programación Arduino Uno

En el entorno de la línea de código de Arduino, el void setup maneja el ingreso de las variables en su registro principal para la comunicación, alimentación, tiempo de respuesta, y los pines que ejecutaran el funcionamiento del actuador, seguido del void loop para hacer la rutina de las tareas que contiene la programación; condicionales y envíos de datos, mostrando información del sensor y activación del actuador.

Terminado de escribir cada una de las líneas de código se compila y se verifica si el volcado del código es exitoso, en el que aparecerá un mensaje "compilado", que se muestra en la Figura 2.16 en la barra de mensajes (Lee, Levanti, & Kim, 2014).

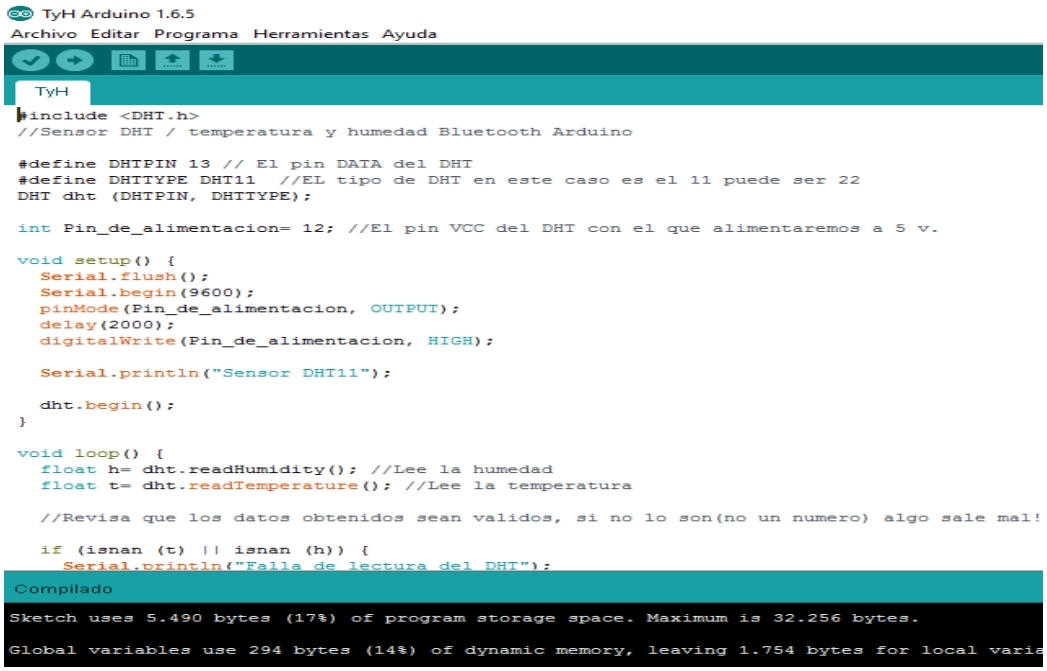


Figura 2.16 sketch Arduino Uno.

Seguido de la programacion se realizo un montaje a modo de prueba, conectado en la protoboar arduino, bluetooh Hc-05 y sensor de temperatura Dht-11 como se muestra en la Figura 2.17 para verificar la captura de los datos de temperatura y humedad ademas el encendido y apagado del diodo led, por medio de la App del equipo movil.

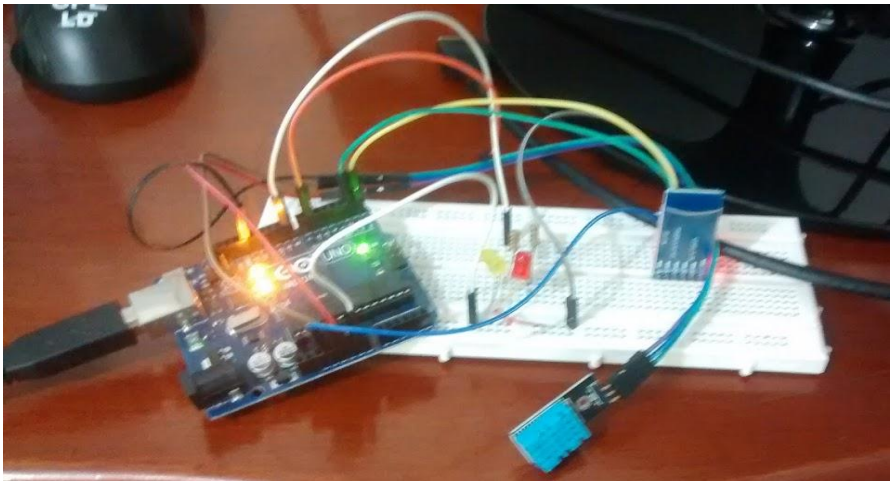


Figura 2.17 Protoboard Arduino y Bluetooth.

3.2.3. Funcionamiento App Inventor Arduino Bluetooth

Esta configuración con la aplicación **App inventor** se integró con Arduino teniendo en cuenta primero en instalar nuestra app en el Móvil, para luego cargar la librería y programar el sensor de temperatura y humedad.

Se definió el pin y modelo de sensor de temperatura que se utilizó, para iniciar el puerto serie que comunica con Bluetooth y el sensor de temperatura. Además se energizo ambos módulos con el pin de 3.3V. Seguidamente en el loop llamamos al procedimiento de leer un dato por el puerto serie, si la lectura es correcta se comprueba que la temperatura y la humedad, es mostrada en el dispositivo móvil.

3.2.3.1. Lectura y Recepción de datos enviado por Arduino

En esta parte de nuestra programación por bloques se ejecuta la lectura de los datos utilizando la variable text para mostrar el orden de lectura de la temperatura y humedad, recibiendo la lectura de los datos enviados por la placa Arduino por medio del dispositivo Bluetooth, como se muestra en la Figura 2.18 variable que se utilizó para alternar la lectura de la temperatura y la humedad (Raúl C., 2015).

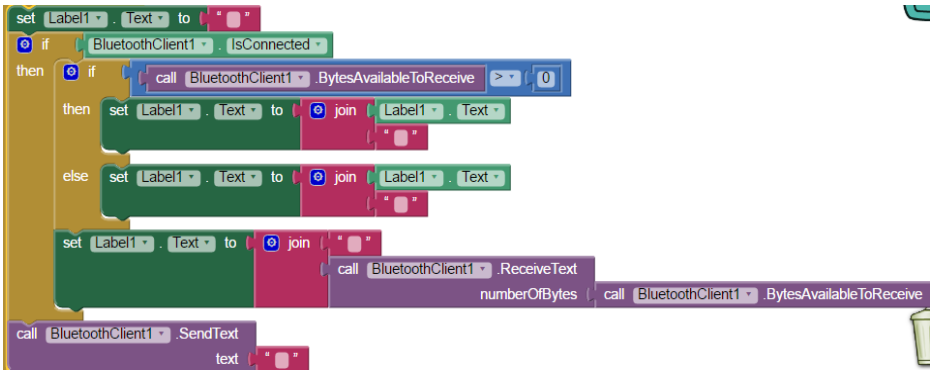


Figura 2.18 Bloque de Programación.

3.2.3.2. Conexión y Recepción de Texto enviado por Arduino Uno

Se realizó la configuración como se muestra en Figura 2.19 para la comunicación del bluetooth, conexión, sesion y presentación de la aplicación.

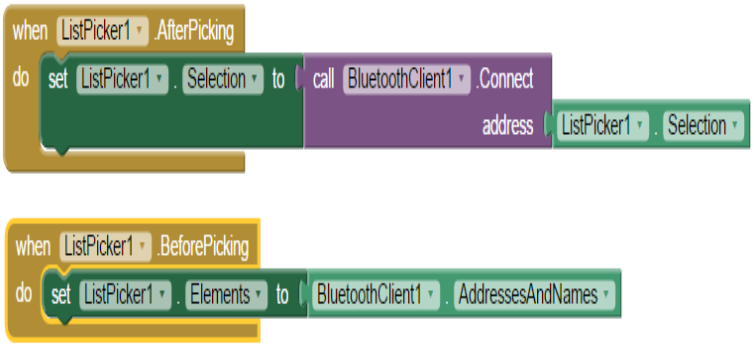


Figura 2.19 Envió y Recepción de texto.

3.2.3.3. Botones para activar la orden de Off/On del actuador

Se realizó un bloque de programación para los botones de envío y respuesta de un dato, para que lo interprete el sistema Arduino, encendiendo o apagando un actuador, en un puerto digital como se muestra en la Figura 2.20.

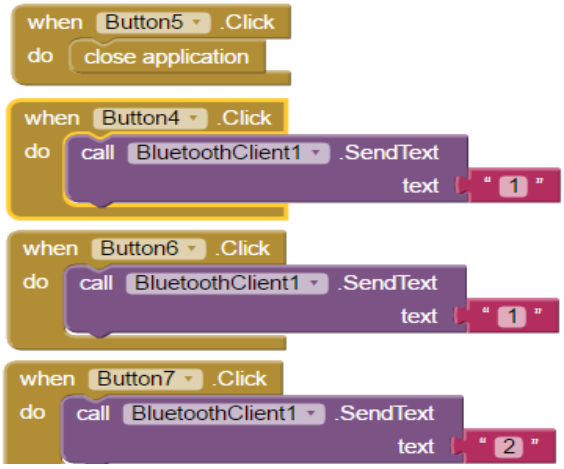


Figura 2.20 Control de botones.

3.2.3.4. Datos enviados de la Humedad y Temperatura

Figura 2.21 Interfaz de usuario App inventor.

3.2.4. Comunicación Raspberry pi Arduino Uno

La configuración que se hizo para la comunicación serial entre Arduino y Raspberry PI, fue conectando por medio de cable USB, y haciendo uso del programa Python, que permitió hacer el control e integración sobre nuestras dos placas para interactuar desde la RPI los sensores y actuadores que están conectados en el Arduino. Ver anexo C

Pasos:

Arduino Uno: Conexiones, Compilar el programa

Raspberry Pi: Comandos, ejecución del sistema de ficheros

La instalación de la comunicación serial, es ingresando a el terminal para escribir en la línea de comandos el siguiente código.

```
sudo apt-get install python serial
```

La manera en que se establece la comunicación, para hacer la integración entre estos dos dispositivos es colocando este script que se muestra a continuación en el fichero de configuración de RPI.

```
Import serial
arduino =serial.serial('/dev/ttyaACM0',9600)
Print("starting”!)
Comando=raw_input('introduce un comando')
#input
arduino.write(comando)#mandar un comando hacia el arduino
If comando== '1':
Print('led encendido')
Print('led encendido')
Elif comando=='2';
Print('led apagado')
 Arduino.close()#Finalizamos la comunicación
```

3.2.5. Configuración LCD de la RPI

Los **pines GPIO de la RPI** se utilizaron para realizar la conexión, los cuales muestran el sistema de E/S (Entrada/Salida) como se muestra en la Figura 2.22, para la configuración que se realizó para el display LCD, visualizando las direcciones ip y los recursos de los equipos en pantalla. Ver anexo B.

Todos los pines son de tipo "unbuffered", es decir, no disponen de buffers de protección, teniendo cuidado con las magnitudes de (voltajes e intensidad) cuando se conectó los componentes a los pines de alimentación.

Los Pines de alimentación son de 5v, a 3.3v (limitados a 50mA) y tierra (GND) que alimentan estos voltajes a la placa RPI. A continuación se anexo un esquema donde observa cómo debe conectarse el display a la placa Raspberry pi.

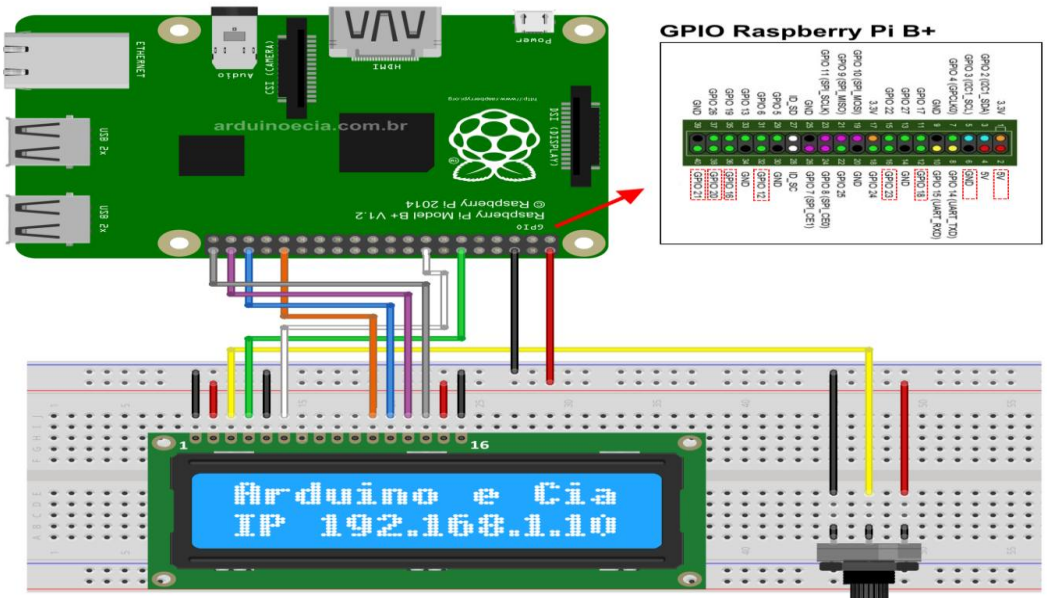
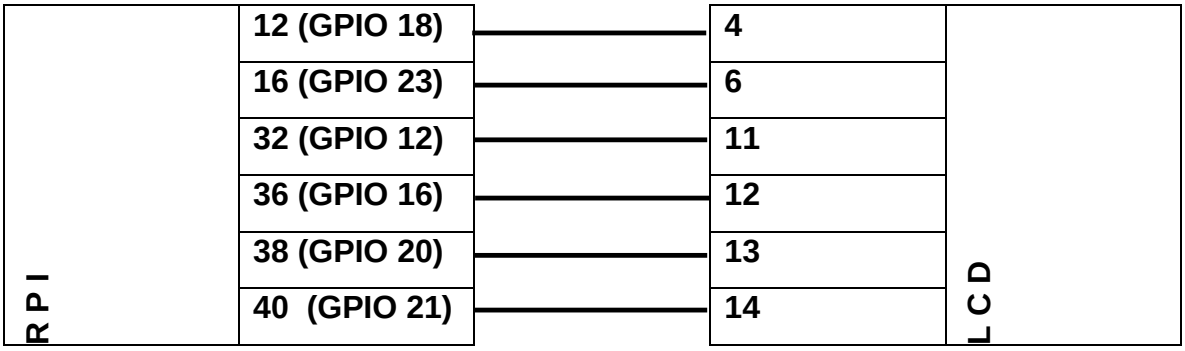


Figura 2.22 Montaje puertos RPI Gpio.

3.2.5.1. Conexiones Display LCD 16x2.

Estos son cada uno de los puertos GPIO de la RPI que se utilizaron para controlar un display LCD de 16x2, el cual es configurado y controlado desde la placa.



3.3. Diagramas de bloques

3.3.1. Esquema del diagrama en Hardware

El dispositivo Raspberry Pi maneja un gran desempeño en la integración de todos los dispositivos como se muestra en la Figura 3.1, como son, conexiones, protocolos de comunicación, configuraciones de servicios, captura y envió de datos que se va a mostrar.

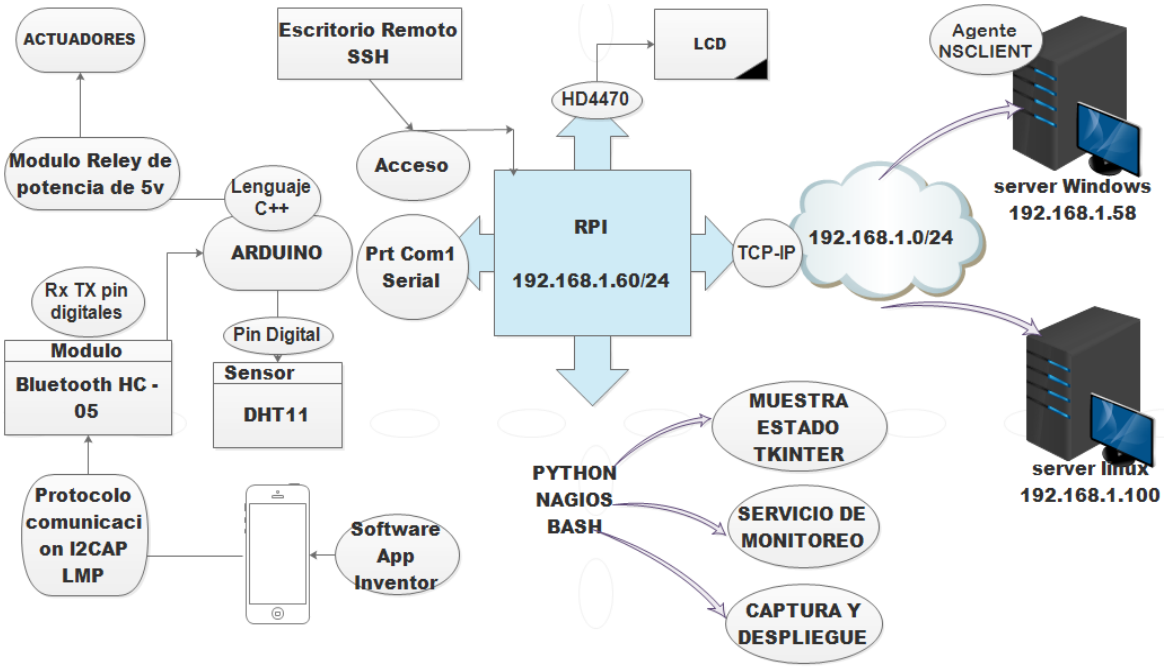


Figura 3.1 Integración dispositivos RPI Arduino Uno.

La RPI se estableció como un servidor central de gestión y monitoreo que interactúa, con la parte domótica, y otros dispositivos que por su amplio desarrollo a nivel de hardware son necesarios para la implementación y que se detallan a continuación:

Arduino: Se implementó el dispositivo para la configuración de la parte domótica que logra la comunicación con los demás dispositivos que toman medidas de

temperatura y de humedad, también ejecuta la parte de potencia con un módulo relay,

Sensor Humedad Dht11: Este dispositivo se comunica con el Arduino para mostrar la información en el dispositivo móvil o en el servidor de gestión.

Bluetooth HC-05: Se interconecta con los pines digitales del Arduino para la transmisión y recepción de información de datos, que se interpreta en el dispositivo móvil en la app que se desarrolló mostrando la temperatura y la humedad.

Modulo Relevo: Este dispositivo se interconecta en la Raspberry pi para dar la alimentación del relevo, que maneja un voltaje de 5v el cual le hace llegar, para encender la parte de potencia y activar el actuador, junto a ello en el dispositivo Arduino se conecta en un pin digital para mandar la orden de apagar o encender el actuador, desde el móvil o bien sea desde el servidor de gestión de monitoreo.

LCD: En este display se identifica el poll de direcciones activas que detecta NAGIOS en alguna de las redes que se conecta al servidor, para enviarlas a los puertos GPIO, que permita obsérvalas y verificar que ip están asociadas a la red.

Server Linux Windows: En este equipo se montaron servicios locales y públicos para el monitoreo observando cómo es el comportamiento de los procesos que trabaja en un segundo plano, y para Windows se instala un agente que se llama Nsclient que se expone todos los servicios que se configuraron en el servidor de gestión y monitoreo para saber los estados de la red local.

Acceso al servidor: se instaló por comandos VPN, SSH y un escritorio remoto que permita ingresar desde cualquier equipo de trabajo.

Se hizo la instalación de todos los plugins para monitorear los servicios, de los dispositivos que se integran, dentro de los archivos de configuración en Python,

Bash y los ficheros de Nagios que se programó, para la captura de los servicios de monitoreo de red, domótico, estados, parámetros y métricas de la red.

3.3.2. Arquitectura de software

El manejo de la parte lógica del software que se implementó para el desarrollo de cada una de las funciones, se centró en la gestión y monitoreo en el que se hizo un balance en el código de programación en Python, bash y C, la cual se especificó por módulos de cada fichero que se crearon en el sistema, con un diseño de interfaz para el orden de los botones, sensores, actuadores, display LCD, sesión de IP activas en la red, como se muestra en la Figura 3.2, e información que se despliega con captura y envió de información en el sistema RPI con cada una de sus configuraciones.

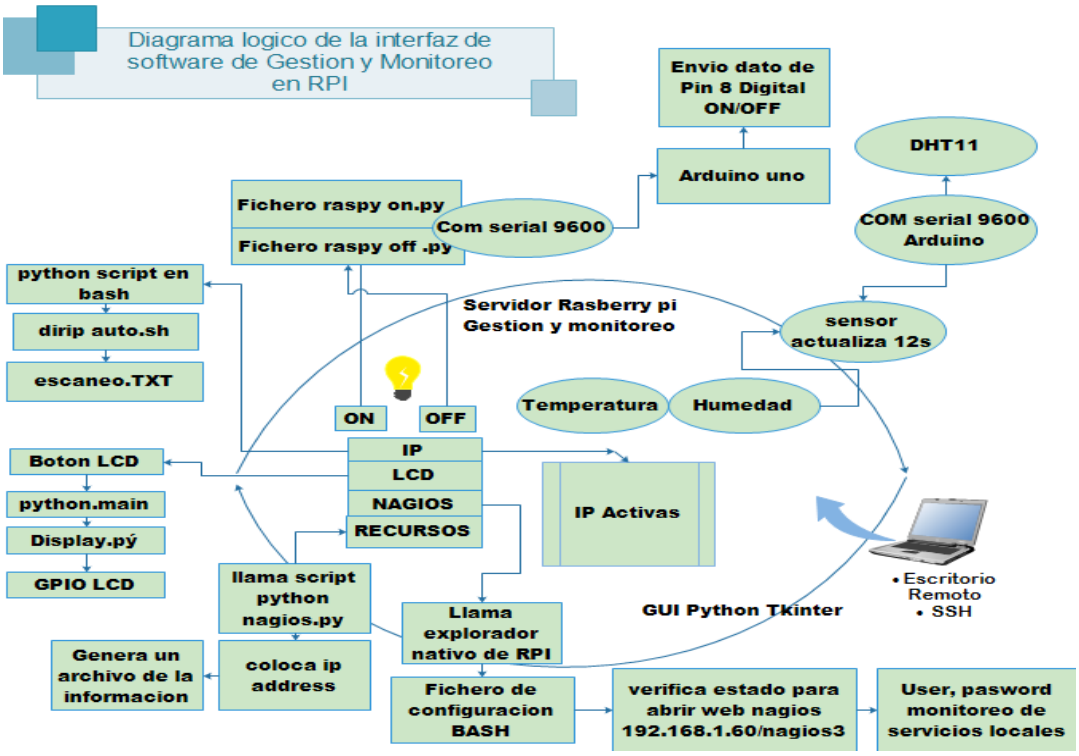


Figura 3.2 Diagrama Gestión y Monitoreo.

Script de temperatura y humedad; se programa en Python una interfaz gráfica con Tkinter donde este captura los datos enviados por el Arduino, donde se configura el pin digital, para simular un puerto COM 1 serie, con transmisión a 9600 baudios, con los que se actualizan cada 12s aproximadamente y llegan a nuestro servidor en el fichero principal.py donde se guardan las líneas de programación en que se mostraran los datos en la interfaz Tkinter.

Con la siguiente sentencia se abre la comunicación con el arduino, quien es el que tiene la función de interpretar las mediciones que toma el sensor:

arduino=serial.Serial('/dev/ttyACM0',9600)

Los datos son enviados en un Json y capturados e interpretado por Python, que se indica a continuación:

data=json.loads(caracter) H=data['Humedad'] T=data['Temperatura']

Script de IPActivas; La función crea un archivo de resultados llamado escaneo.txt, por intermedio de un script en Bash llamado diripauto.sh, el cual verifica las ip´s activas que se encuentran en la red local, y posteriormente mostrados en la interfaz gráfica de Tkinter(Python) por intermedio de un scroll. os.system("bash /home/pi/diripauto.sh"), con esta sentencia se logra llamar un script de Bash en el script de programación Python.

Botones ON/OFF; Se desarrolló en Tkinter (Python) el cual tiene como finalidad interactuar con el Arduino mediante la comunicación serial y este a su vez dar la orden respectiva al actuador final. Cuando el botón está en ON u OFF, lo que este realiza es abrir la comunicación y enviar un 1 o 2 respectivamente al arduino, y que es interpretado y ejecutado por este dispositivo encendiendo o apagando el actuador. Con estas líneas de programación en Python se logra escribir en el arduino un 1 o 2 según sea el caso comando = '1' o '2' - Arduino.write (comando).

LCD display: El botón LCD llama un script en Python display.py con el siguiente código os.system("sudo python /home/pi/displey.py"), el cual llama el archivo generado txt, donde fueron almacenadas cada una de las ip's activas que se encuentran en el segmento de red. Para después mostrarlas por LCD 16x2, el cual se encuentra configurado en los puertos GPIO de la Raspberry Pi.

Script Nagios: La función que se le indico a este botón es básicamente abrir el explorador web donde se encuentran configurados los servidores monitoreados para redirigir la aplicación Opens source Nagios sobre la Raspberry Pi.

Script Recursos: Abre una ventana Tkinter (python), donde se puede digitar la ip del servidor monitorizado con la aplicación OpenSource Nagios, donde nos indica en ventana los recursos que se encuentran configurados y monitorizados respectivamente. Además, podemos mostrar esta información por el terminal de la Herramienta de Bajo coste Raspberry Pi.

Con el siguiente código se puede capturar y mostrar en el terminal de la Rpi, nos indica la información de disco duro del servidor monitorizado por Nagios, ***commands. getoutput ('/usr/local/nagios/libexec/check_nt -H' +str(vara)+' -p 12489 -s 123456 -v USEDDISKSPACE -l c -w 80').***

Para indicar la memoria del servidor utilizamos el siguiente código ***commands.getoutput('/usr/local/nagios/libexec/check_nt -H'+str(vara)+' -p 12489 -s 123456 -v MEMUSE -w 80 -c 90'),***

4. Resultados y Discusión

El propósito fundamental del desarrollo del trabajo fue integrar dos sistemas domótico y monitoreo de red, bajo un dispositivo de características limitadas que cumpliera con el objetivo de un servidor de bajo coste.

4.1. Desarrollo e implementación del prototipo Hardware

Se logró desarrollar un prototipo que mantuviera comunicación con otros dispositivos necesarios Arduino, Bluetooth, LCD, Sensor Dht-11 y Relay como se indica en la Figura 4.1 que fueron indispensables para cumplir con el objetivo de integrar dos sistemas en uzna placa reducida. Uno de las limitaciones que se presento fue la capacidad de procesamiento del servidor RPI, para lo cual fue necesario integrar otro dispositivo que permita repartir las diferentes funciones utilizando los puertos GPIO a puertos digitales en Arduino, como lo muestra la Figura 4.2 para que no haya una saturación en los procesos, ejecutando cada tarea sin ningún tipo de problema y a su vez obteniendo una disminución de consumo de energía.

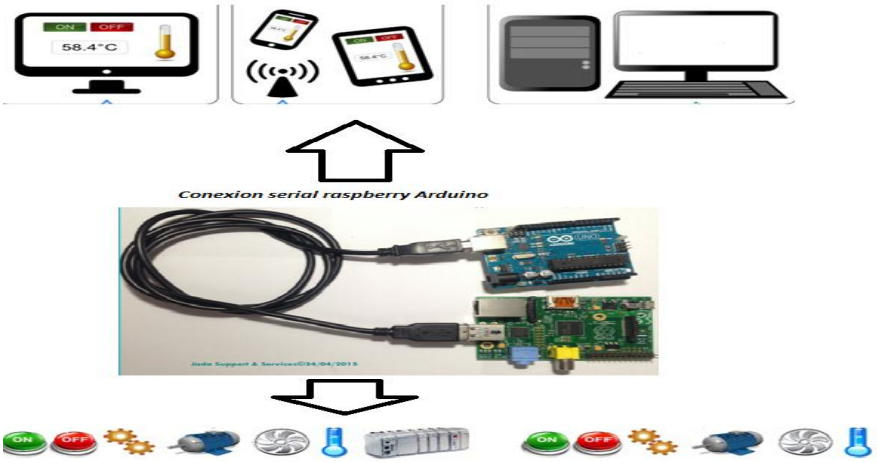


Figura 4.1 integración Raspberry Arduino.

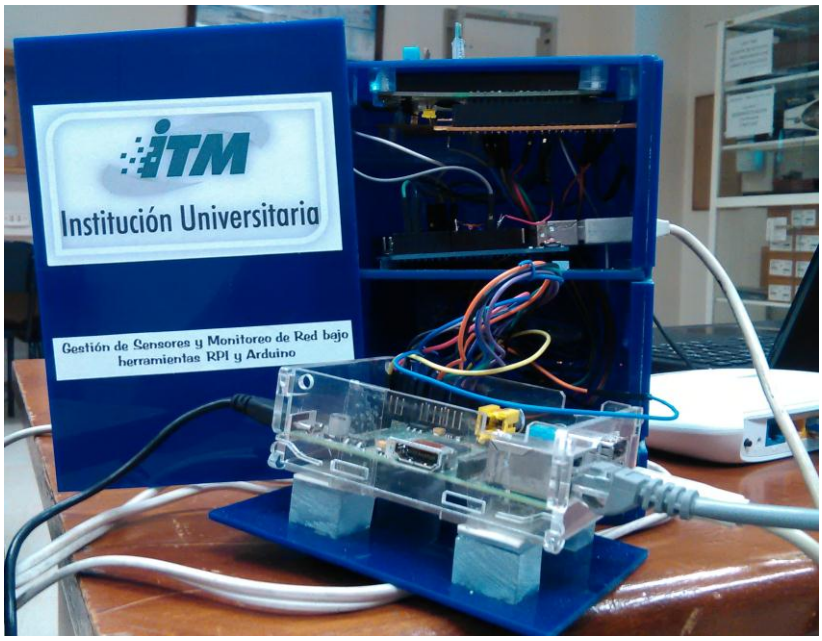


Figura 4.2 Prototipo Hardware.

4.2. Programación en Raspberry pi

El dispositivo Raspberry Pi se programó como un servidor central de fácil instalación, configuración y puesta en marcha, además el sistema instalado en la placa no es tan diferente de otras distribuciones Linux que se instalan en un PC, contando con total integración a diferentes lenguajes de programación en nuestro caso Python, Bash y C. Por intermedio de la programación de los puertos GPIO se pudo establecer la captura, procesamiento y presentación visual de los datos como se muestra en la Figura 4.3 por intermedio de una interfaz gráfica bajo python.

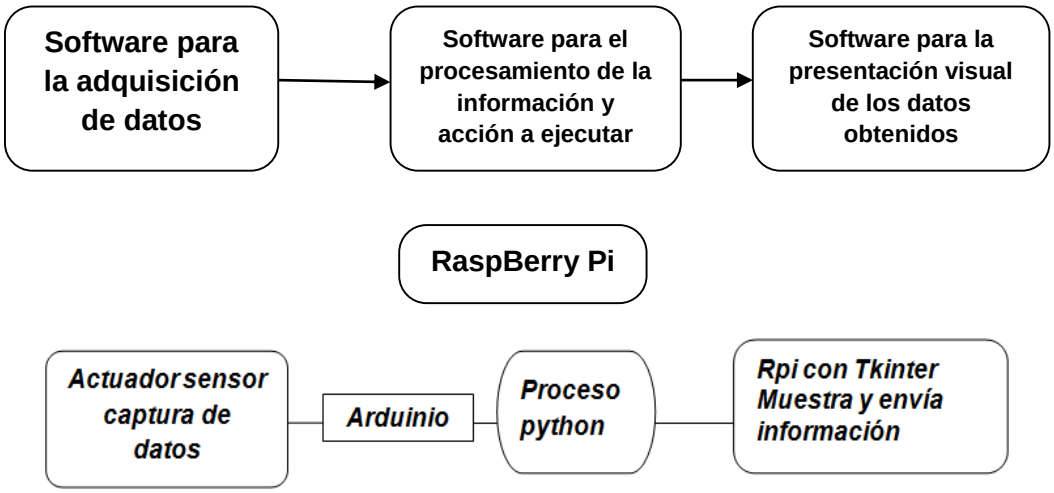


Figura 4.3 procesos de programación RPI.

4.3. Programación aplicación móvil

En la realización de la aplicación para el equipo móvil se pensó como una solución agregada al trabajo, para una alternativa de innovación y portabilidad para el usuario, encontrando la forma de agilizar procesos de la RPI para una mayor eficiencia, gracias a la integración del Arduino el sensor Dht-11 y el Bluetooth, se obtienen los datos adquiridos por captura de la humedad y temperatura, Figura 4.4 como se muestra en la junto a la activación de un actuador para luego enviarlo y por conexión Bluetooth, se pueda presentar la información de una forma organizada en una interfaz gráfica intuitiva y de fácil manejo para el equipo móvil, teniendo como limitación el tipo de conexión, y la distancia que tiene un rango de 5 a 10 metros.

Figura 4.4 Programación App Resultado.

4.4. Programación de la interfaz gráfica con Tkinter

Se programó una interfaz gráfica que presentara los datos capturados y procesados por Arduino, y Sensor Dht-11, que mostrara cada una de las IPs activas que se encuentran en un segmento de red, además teniendo el control de sistema domótico como se muestra en la Figura 4.5, teniendo la capacidad de presentar los datos de monitoreo de recursos con Nagios como se indica en la Figura 4.6. Una de las limitaciones es el tiempo de respuesta que toma para mostrar los datos en la interfaz gráfica.

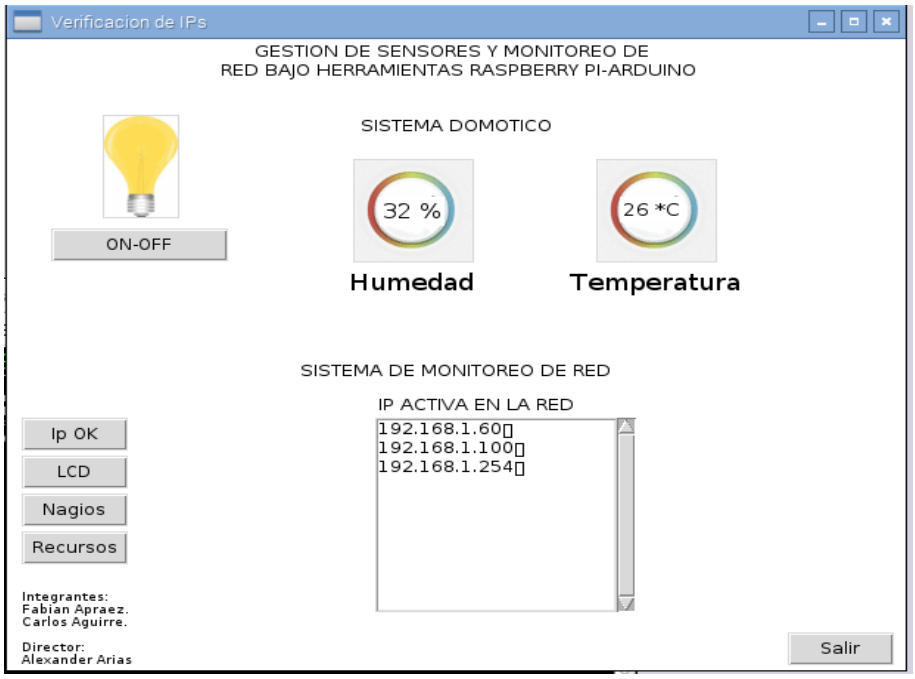


Figura 4.5 Interfaz gráfica Tkinter.

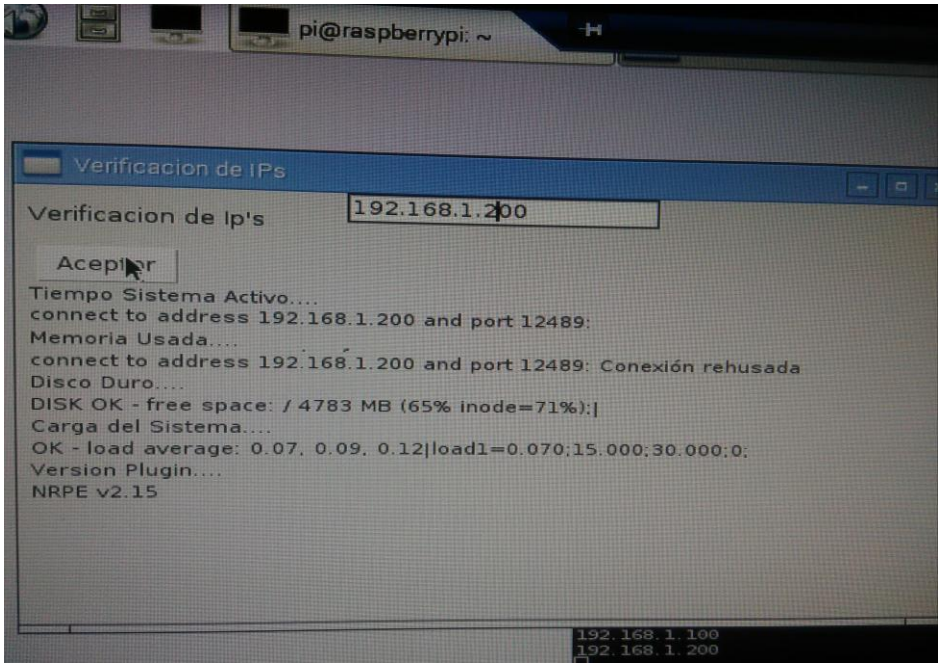


Figura 4.6 Presentación en Tkinter de recursos.

4.5. Comunicación serial RPI y Arduino

Se realiza la integración entre RPI y Arduino para la programación en el fichero en Python, y la identificación del puerto conectado por el cable USB al dispositivo Arduino, como se muestra en la Figura 4.7 importando el código en la interfaz de configuración con el inicio de transmisión, y creando la ruta de comunicación entre las dos placas. Una de las limitaciones es la velocidad de transferencia, teniendo un retardo al envió de un dato.

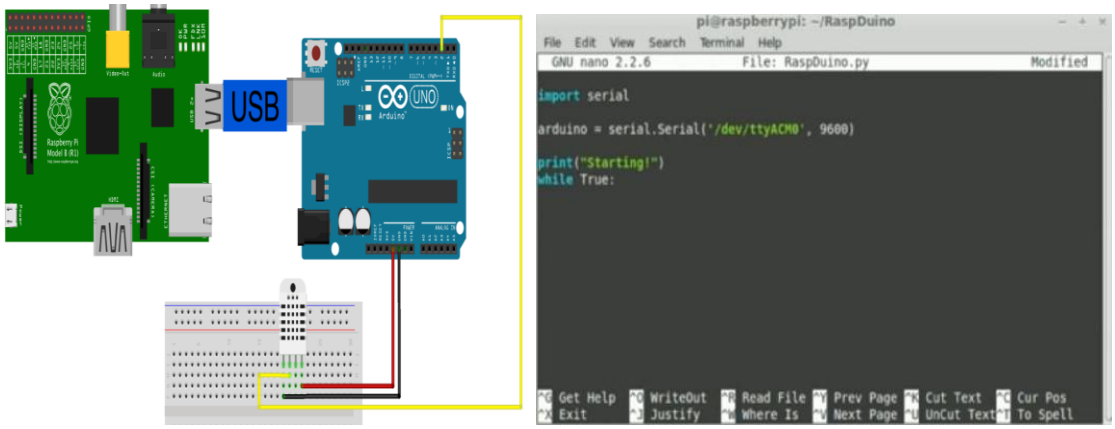


Figura 4.7 Integración Rpi Arduino.

4.6. Monitoreo Nagios – RPI

La Raspberry Pi tiene un procesamiento limitado cuenta con una arquitectura reducida para instalar una herramienta de gestión y monitoreo Open Source que se puede configurar y desplegar en la placa (Open Hardware) como se muestra en la Figura 4.8. Además fue posible obtener resultados de monitoreo mediante líneas de comando desde el terminal (Raspbian) de la RPI, de los diferentes recursos que se están monitoreando en cada uno de los equipos intervenidos, a lo que también es posible realizar mediante lenguaje de programación Python.

Ver anexo A. Se realizaron unas pruebas en el laboratorio de telecomunicaciones N 101 del Instituto Tecnológico Metropolitano como se muestra en la Figura 4.9 para observar los servicios monitoreados, obteniendo resultados de los recursos de los equipos de la red como son la carga de la CPU en cada uno de los equipos que se tenía en el momento, indicando la capacidad total, espacio libre del DISCO DURO y MEMORIA, como se muestra en la Figura 4.10 desde la interfaz gráfica de Nagios. Una de las restricciones que se presenta en este tipo de placas es la falta de procesamiento para desarrollar proyectos a gran escala.

Módulo de administración y gestión de Monitoreo.

• Módulo de control y adquisición de datos.

• Módulo I/O.

• Periféricos.

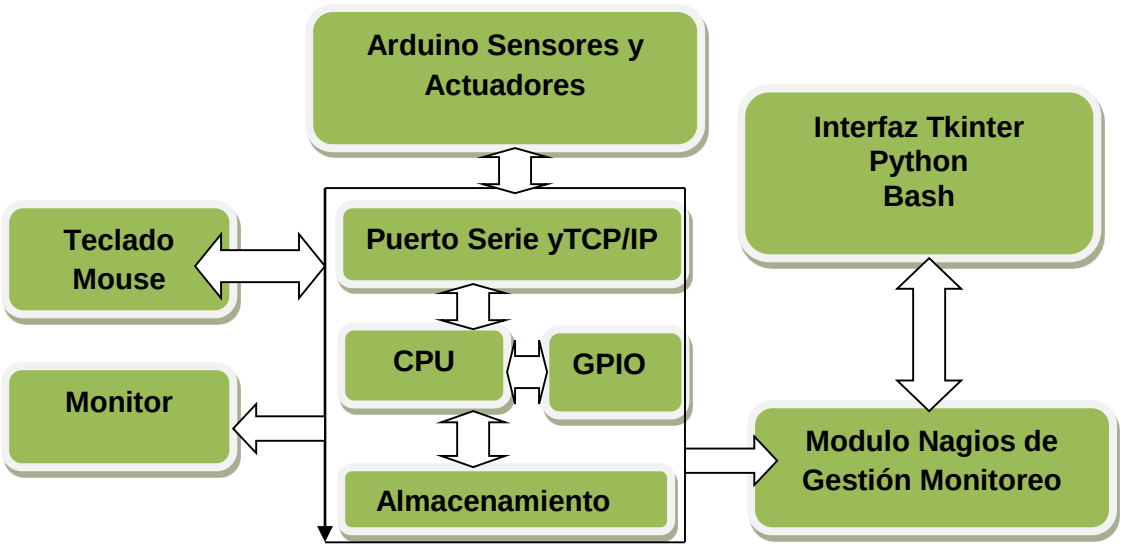


Figura 4.8 Bloques de Gestión Rpi.

Figura 4.9 Laboratorio de telecomunicaciones ITM.

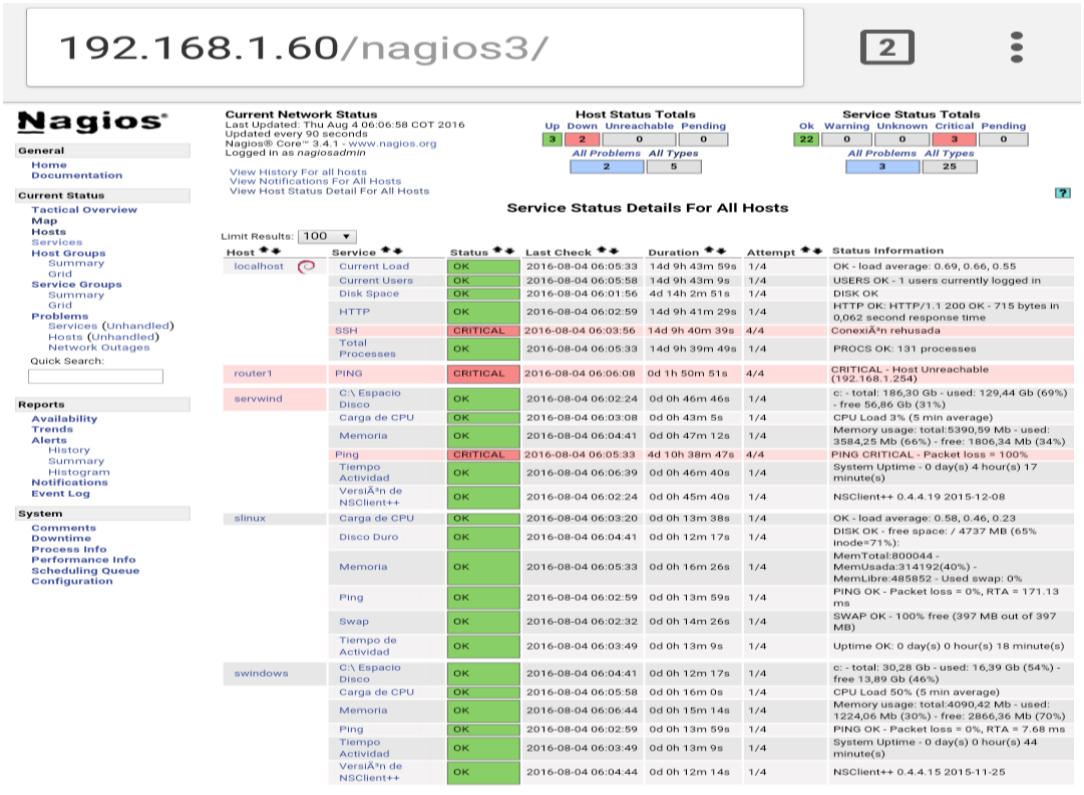


Figura 4.10 Interfaz de servicios Nagios.

4.7. Acceso Remoto

Se puede establecer un directorio remoto para la implementación del proyecto en la placa RPI, que permite una comunicación en la ejecución de una aplicación por medio de un acceso de un terminal, mostrando la información que se recibe por medio del servidor RPI, como se muestra en la Figura 4.11 permitiendo la seguridad a nivel de capa de transporte, para ver la configuración grafica con acceso y habilitación de protocolos de comunicación SSH y la aplicación XRDP, la limitación que se encontró es el procesamiento que puede tener un retardo en el tiempo respuesta, mostrándose algo desfavorable cada vez que se accede de forma remota para ejecutar la aplicación.

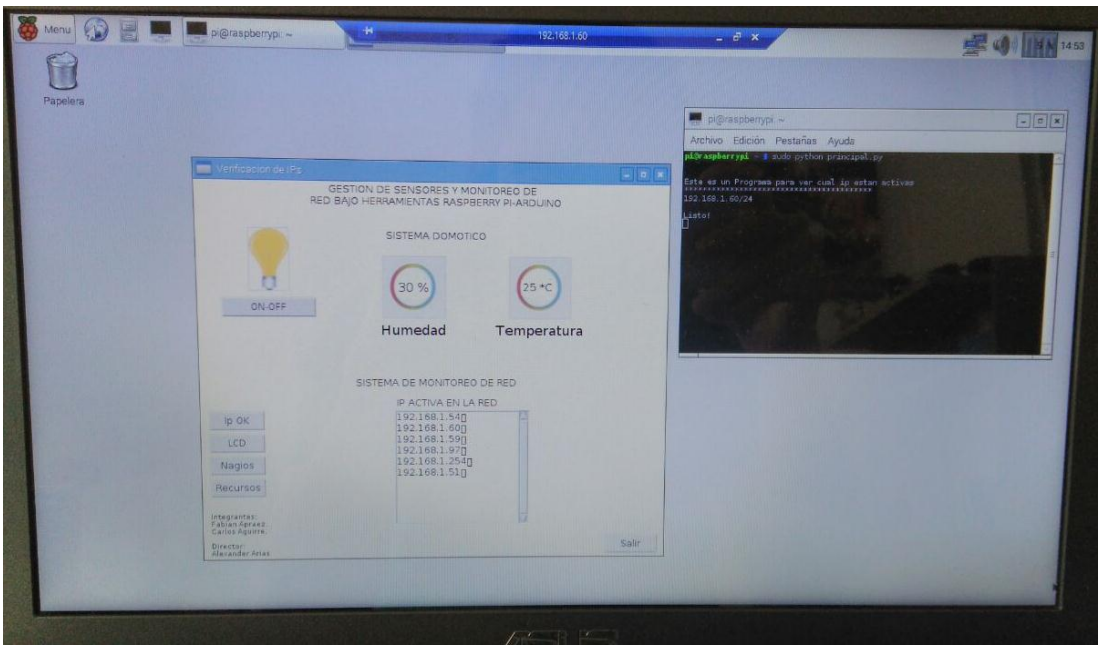


Figura 4.11 Acceso remoto Xrdp.

4.8. Análisis del procesamiento de cpu Rpi

Se analizó la capacidad de procesamiento y recuperacion que tiene la rpi como servidor principal e integrador en el manejo de los dos sistemas tanto domotico como monitoreo de red, el cual se evidencio que el procesamiento es un factor que presenta limitaciones al ejecutar varios procesos al tiempo, haciendo que la rpi llegue a un nivel muy alto de procesamiento; cuando se ejecuta el archivo principal.py podemos observar el porcentaje de ocupacion y de actividad como se indica en la Figura 4.12, pero tambien es de gran importancia recalcar que los tiempos utilizados para recuperar los porcentajes de ocupacion son bajos como se indica en la Figura 4.13.

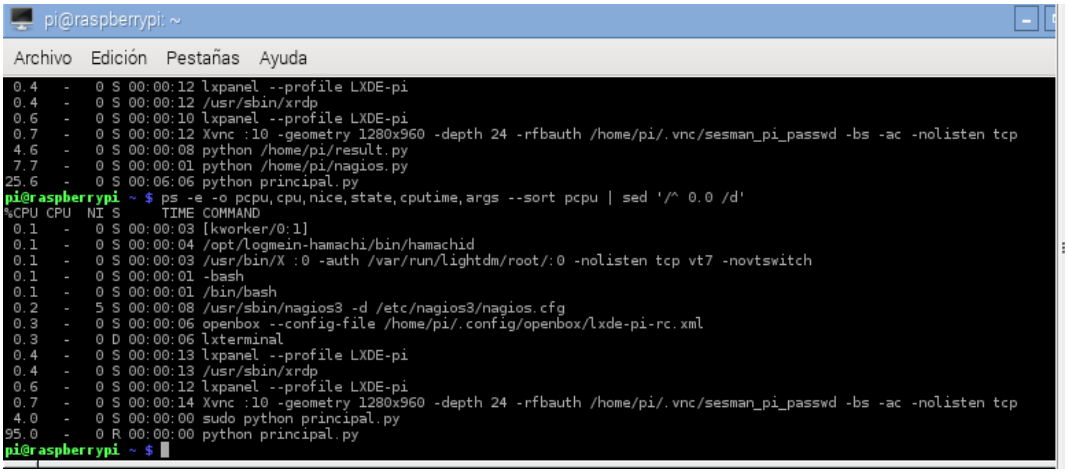


Figura 4.12 Administrador de tareas.

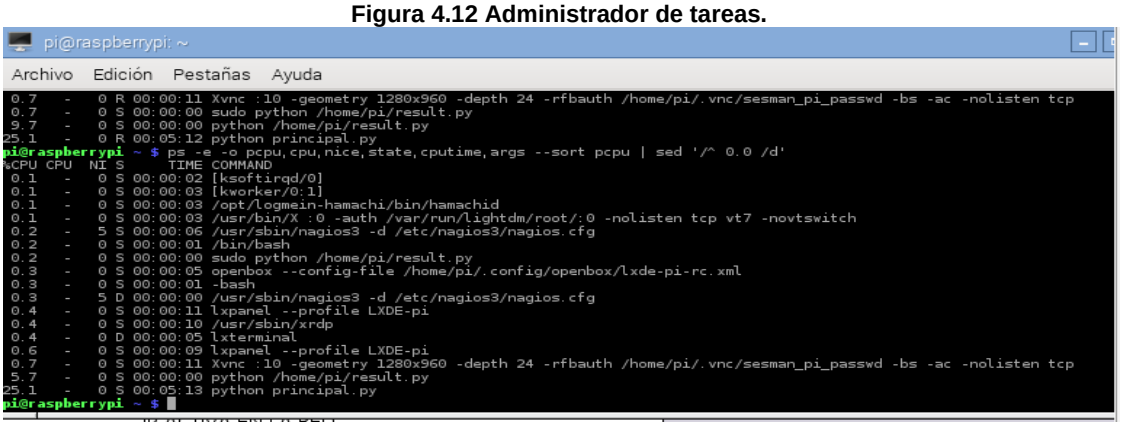


Figura 4.13 Porcentajes activación de tareas.

4.9. Análisis puerto Gpio en LCD

Realizando la configuración de los puertos Gpio en la RPI se implementó un display LCD para ver la programación en el anexo B, presentando la información de los recursos, servicios locales de los equipos, y las ips activas de la red local como se muestra en la Figura 4.14, el funcionamiento que tuvo como resultado con las actividades programadas que se ejecutaron a la vez, ocasiono una gran ocupación en el procesamiento de la placa, afectando la ejecución de actividades y aplicaciones del servidor RPI, buscando una solución que se realizó mediante código python, sirve para ejecutar esta tarea de lectura e interpretación de los Gpio en segundo plano, y ahorrar consumo y repartir equitativamente en la ocupación de los procesos para tener capacidad de los recursos del servidor para realizar estas tareas.

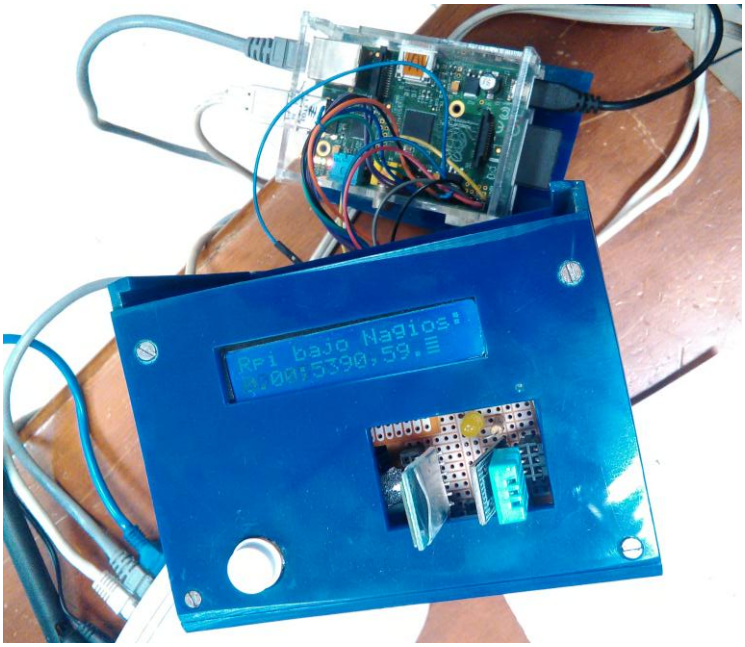


Figura 4.14 Prototipo LCD de la RPI

4.10. Análisis del consumo de energia de la Rpi

En los resultados obtenidos se conecto la placa para medir con un multimetro las variables de voltaje y corriente alrededor de una hora, observando si tenia un cambio significativo en estos parametros, rectificando la eficiencia del consumo energetico que tiene la placa como se muestra en la Figura 4.15, y especificando en la Tabla 1.4 los parametros del dispositivo, mostrando una gran ventaja en este tipo de implementaciones para reducir costo y generar un beneficio en el ahorrro de energia.

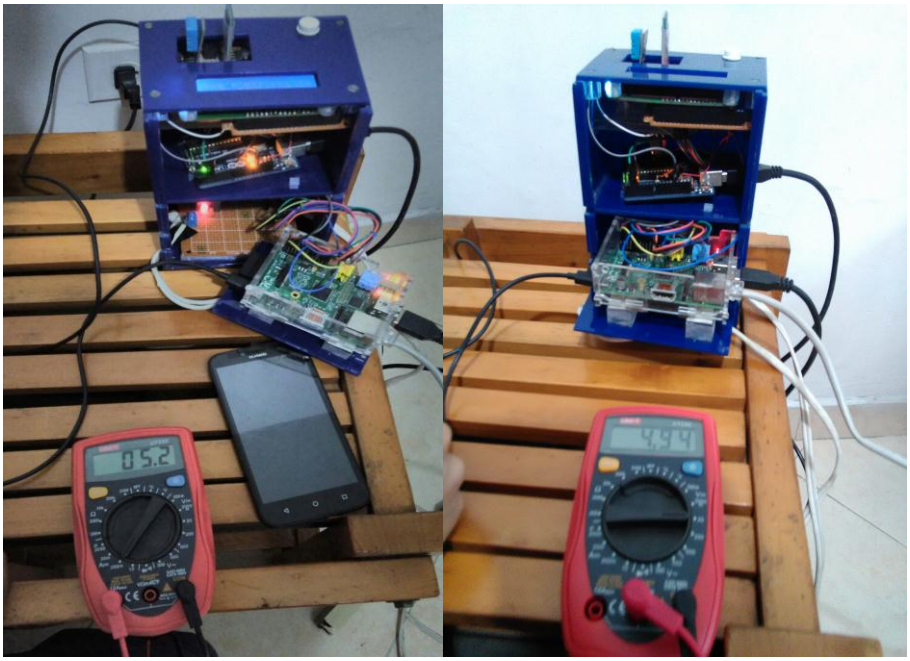


Figura 4.15 Resultados de Medición voltaje y corriente.

Tabla 1.4 Características Rpi y Arduino.

Nombre	***Arduino Uno***	***Raspberry pi +B***
Modelo	***R3***	***Modelo b***
Procesador	***Atmega 328***	***ARM11***

Velocidad de reloj	***16Mhz***	***700Mhz***
Ram	***2kb***	***256Mb***
Flash	***32kb***	***Sd 8GB***
EEPROM	***1Kb***	***n/a***
Entrada de voltaje	***7-12v***	***5v***
Entrada de Corriente	***42mA(3w)***	***700mA(3.5w)***
Digital GPIO	***14***	***8***
Ethernet	***n/a***	***10/100***
USB	***N/A***	***2xUsb 2.0***
Video de salida	***N/A***	***HDMI***
Salida de sonido	***N/A***	***HDMI***

Calculo de la potencia

Este calculo fue el que se obtuvo tomando los datos que nos arrojo el multimetro, voltimetro, aplicando la formula de potencia disipada/consumida, como se muestra en la ecuacion (1) en el que se tiene conectado a la energia la RPI, Arduino, rele, bluetooth, y todo el prototipo.

Ecuacion (1) $P = V * I$

$$4.94\,V * 520mA = 2.5W$$

Analisis y medicion de variables de potencia

Se realizan todas las conexiones correspondientes en el multimetro y voltimetro para la obtencion de datos, luego se enciende la placa RPI, y se procede a medir los datos del consumo de energia dentro de los 60 minutos, ejecutando en el dispositivo el ingreso de tareas al sistema de monitoreo-domotico, teniendo en cuenta los tiempos determinados y comportamientos, que se fueron observando en cada dato que nos arrojo la medicion en tiempo real, las cuales son variables de tiempo voltaje y corriente, en las cuales sirvio para hallar el resultado de la potencia, como se muestra en la Tabla 1.4 finalizando se hace un promedio de cuanto consume en total el dispositivo.

La desviacion estandar del consumo de potencia es 0.0128, el cual muestra que los valores de potencia se mantienen en unos niveles estables alrededor de 2.57W.

Tabla 1.5 Resultados de Potencia

ITEM	TIEMPO (Min)	VOLTAJE (V)	CORRIENTE (mA)	POTENCIA (W)	
1	5	4,94	520	2,57	
2	10	4,93	525	2,59	
3	15	4,95	522	2,58	
4	20	4,94	524	2,59	
5	25	4,92	519	2,55	
6	30	4,93	522	2,57	
7	45	4,95	523	2,59	
8	60	4,94	520	2,57	**Desviación Estándar**
Promedio		4,938	521,88	2,58	0,012806133

4.11. Diseño del prototipo final

Se pensó Inicialmente en un material acrílico para realizar el prototipo, con un espacio que se tuvo en cuenta para cada uno de los dispositivos, tomando medidas específicas de los entrepaños como se muestra en la Figura 4.16, con una posición horizontal para las conexiones, y un orden que se dio a la distribución, que fue de abajo hacia arriba, colocando por módulos el Reley, RPI, Arduino, módulo de comunicación, LCD y el botón, con un diseño sencillo portable, practico, y fácil de organizar, uno de las limitaciones es la fragilidad del material que está hecha la caja.

Figura 4.16 Distribución y diseño del Prototipo

5. Conclusiones y recomendaciones

- Se evaluaron 3 aplicaciones Opensource para la integración y compatibilidad con la RPI y Arduino, y la herramienta Nagios cumplió con el soporte y gestión de monitoreo con fácil adaptación del hardware y software.

- En el diseño que se planteó para el proyecto la Raspberry pi, conto con varias características de protocolos de comunicación, que intervienen los diferentes tipos de conexiones físicas y puertos, con un amplio manejo de lenguaje de programación como python Bash y C.

- El proyecto se basó en encontrar una solución administrativa de red, con dispositivos de bajo costo, teniendo en cuenta aplicación, comunicación, supervisando los servicios locales y públicos de cualquier red de trabajo.

- Con el sistema demótico Arduino se obtiene resultados que ayudan a mejorar la RPI en el rendimiento de repartir los procesos que se integran, con el fin de designar varias tareas y no haya una saturación en los procesos sensores y actuadores, e interfaz gráfica que hace parte de la gestión y monitoreo.

5.1. Recomendaciones Futuras

Este proyecto se tiene varios aspectos de proyección y de investigación para tener en cuenta:

- La facilidad de integración de hardware RPI y Arduino.
- La programación representa un gran alcance para optimizar la comunicación entre estos dispositivos, mejorando el despliegue de recursos y servicios.
- La herramienta RPI se adapta a la herramienta Nagios, facilitando la gestión de monitoreo, y el manejo de sensores y actuadores en Arduino pero con una limitante de procesamiento y respuesta, teniendo en cuenta que esto es por ahora un principio, y con la ayuda del desarrollo de las nuevas tecnologías e investigación, esto tendrá un gran impacto futuro en el mundo de las telecomunicaciones.
- Adaptar los códigos a las nuevas versiones de la RPI.
- Implementar una red de dispositivos RPI en diferentes puntos de una misma red y diferentes VLans para el monitoreo completo de la red.

Anexos

Se explica detalladamente los códigos de programación que se utilizaron, para la realización y ejecución de los ficheros que se mencionaron para la configuración del sistema de la RPI, y el dispositivo Arduino.

A. Programación Arduino

A-1. Componentes de la placa Arduino UNO

- **Elementos analógicos:** Potenciómetro, ldr, ntc, Zumbador, motor eléctrico, Led. Señales analógica: puede tomar infinitos valores entre su valor mínimo y máximo.
- **Elementos digitales:** Pulsador, detector de presencia, led, timbre. Señales digitales solo puede tomar dos valores, el máximo asociado a 1 o acierto “on” y el mínimo asociado a cero, falso o “off”.
- **Entradas y salida:** El micro controlador recibe información de las entradas (read), la procesa y escribe un 1 o un 0 (5v ó 0v) en las salidas (Write), actuando sobre el dispositivo que tenemos conectado
- **Botón de reset:** Permite reiniciar el programa y cargar uno nuevo
- **Pines de entrada y salida:** pines de entrada y salida: Permiten conectar elemento que dan información y crean actuaciones.
- **Puertos USB:** Se cargan las instrucciones a ejecutar, el programa que es realizado en el entorno de programación de Arduino.

A-2. Variables de programación

Estructura de control

Se explicará brevemente las estructuras de programación para un completo desarrollo de la aplicación en ARDUINO para llegar al objetivo del proyecto.

En esta parte declaramos todas las variables que vamos a usar en el programa, que permite asociar nombres a números lo que nos será de mucha utilidad a la hora de modificar los programas.

Bucle para (condición inicial; condición final; incremento) es un comando para repetir la misma operación un cierto número de veces.

Void setup{}: es la función de configuración de los pines de Arduino y sólo se ejecuta una vez, mientras que *loop*() se ejecuta una y otra vez hasta que se apague el sistema, o se gasten las baterías Bucle infinito.

Void loop{}: es la parte del programa que se ejecuta de forma cíclica.

If (condición) {}: Es un comando que sirve para discriminar si se dio una determinada condición. Las comparaciones son: igualdad,= = desigualdad, > mayor que menor que, >= mayor o igual que, y <= menor o igual que. Todo lo que figure entre las llaves será ejecutado sólo si se da la condición entre paréntesis.

If else: Verifica si se cumple una condición y ejecuta lo que está entre llaves, si no se cumple ejecuta lo que está debajo del else.

While: Ejecuta el conjunto de instrucciones entre llaves mientras se cumpla la condición.(Chacon M., 2014)

A-3. Desarrollo de la programación sketch Arduino Uno

En esta parte del trabajo se incluirá la programación en el sketch de Arduino para compilarla y después pasarla a la placa, con el fin de iniciar con la comunicación que se tiene para los actuadores y sensores para este proyecto.

```
#include <DHT.h>
//Sensor DHT / temperatura y humedad Bluetooth Arduino
#include <SoftwareSerial.h>
#define txPin 3 // definimos txpin en que pin de arduino se encuentra
#define rxPin 2 // definimos rxpin en que pin de arduino se encuentra
#define PIN 6// El pin encendio al leer dato de temperatura
#define DHTPIN 13 // El pin DATA del DHT
#define DHTTYPE DHT11  //EL tipo de DHT es el 11
```

```
DHT dht (DHTPIN, DHTTYPE);

int led8 = 8;                  // pin que se utilizara para encender el led

int Pin_de_alimentacion= 12 //El pin VCC del DHT alimentamos a 5 v.
Software Serial bluetooth(rxPin, txPin);      // pin comunicación bluettoth

void setup () {
 Serial.flush();
  pinMode(Pin_de_alimentacion, OUTPUT);
  pinMode(led8,OUTPUT);
 //delay(2000);
 // digitalWrite(Pin_de_alimentacion, HIGH);
  bluetooth.println("Sensor DHT11");
  dht.begin();
  bluetooth.flush();   // Borramos el buffer del serial para evitar errores
  pinMode(rxPin, INPUT);  // Configuramos los pines del bluetooth
  pinMode(txPin, OUTPUT);
  bluetooth.begin(9600); // marcamos la velocidad del puerto bluetooth
 Serial.begin(9600);        // marcamos la velocidad del Puerto serial
}

void loop () {
   h=0;
   t=0;
   delay(1500);
   h= dht.readHumidity();              //Lee la humedad
   t= dht.readTemperature();         //Lee la temperatura
```

En el siguiente código revisa que los datos obtenidos sean válidos, si no lo son algo salió mal e imprime informándonos que hay un fallo de lectura en el DHT-11

```
    if (isnan (t) || isnan (h)) {
      bluetooth.println("Falla de lectura del DHT");
}else {
  bluetooth.println("Humedad: ");
  bluetooth.print(h,0);
  bluetooth.print(t,0);
  bluetooth.print("°C"); //Escribe el valor de la temperatura en grados celcius
  delay(500);
  Serial.print ("{\"Humedad\": ");
  Serial.print(h,0);
  Serial.print(",");
  Serial.print("\"Temperatura\": ");
  Serial.print(t,0);

}
```

El Arduino imprime la temperatura y la humedad que son medidos por el DHT 11 de tal manera que a la hora de capturarlos en python estos sean entendidos como un JSON. Facilitando el trabajo en el lenguaje de programación.

```
if (Serial.available()) {              //Si está disponible
   c = Serial.read();                  //Guardamos la lectura en una variable
   Serial.flush();
}
if (bluetooth.available()) {          //Si está disponible
    d = bluetooth.read();            //Guardamos la lectura en una variable
    bluetooth.flush();
}
```

El código anterior se encarga de leer los datos suministrados (1 o 2) ya sea por Raspberry pi o Bluetooth.

```
if(t>=30.00){
   digitalWrite(led8,HIGH);
} else if (t<=20.00){
    digitalWrite(led8,LOW);
  }else if (c == '1') {                  //Si es una '1', enciendo el LED
     digitalWrite(led8, HIGH);
     c=0;
   }else if (c == '2') {                 //Si es una '2', apago el LED
     digitalWrite(led8, LOW);
     c=0;
   }else if (d =='1'){
       digitalWrite(led8,HIGH);
      d=0;
   }else if(d =='2'){
       digitalWrite(led8,LOW);
       d=0;
  }
}
```

Después de leer los datos enviados por parte de Raspberry pi o Bluetooth (1 o 2) estas líneas de código se encargan de interpretar y ejecutar según sea el caso. Además, existe una condición en la cual puede automáticamente tomar la decisión de encender o apagar según el dato leído por la (temperatura) en el DHT11.

B. Programación LCD display en la RPI

B-1. Líneas de código y funcionamiento para puertos GPIO Rpi

El siguiente **Mediante estas líneas de código** se verifica cada una de las ip´s activas que se encuentren en dicho segmento de red, las guarda en un archivo txt de forma automática, para luego realizar un proceso de impresión por LCD, el cual se encarga de imprimir en pantalla dichas ip´s activas.

Este código se encarga de establecer la configuración necesaria de los puertos GPIO de la Raspberry pi para mostrar la información requerida a través de la pantalla LCD de dos líneas.

```
#!/usr/bin/python
# coding: utf-8
import RPi.GPIO as GPIO
import time

# Define GPIO to LCD mapping
LCD_RS = 7   #GPIO 7  - PIN 12
LCD_E  = 8   #GPIO 8  - PIN 16
LCD_D4 = 25  #GPIO 25  - PIN 32
LCD_D5 = 24  #GPIO 24  - PIN 36
LCD_D6 = 23  #GPIO 23  - PIN 38
LCD_D7 = 18  #GPIO 18  - PIN 40

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
```

Se configura cada uno de los puertos GPIO y la pantalla LCD de dos líneas, los cuales intervendrán para mostrar la información requerida.

```
def main():
```

```
  # Main program block
  GPIO.setwarnings(False)
  GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers
  GPIO.setup(LCD_E, GPIO.OUT)  # E
  GPIO.setup(LCD_RS, GPIO.OUT) # RS
  GPIO.setup(LCD_D4, GPIO.OUT) # DB4
  GPIO.setup(LCD_D5, GPIO.OUT) # DB5
  GPIO.setup(LCD_D6, GPIO.OUT) # DB6
  GPIO.setup(LCD_D7, GPIO.OUT) # DB7

  # Initialise display
  lcd_init()

  while True:

    # Send some test
    lcd_string("Rasbperry Pi",LCD_LINE_2)
    lcd_string("Direcciones OK ",LCD_LINE_1)

    time.sleep(3) # 3 second delay

    arch = open("/home/pi/escaneo.txt")
    for linea in arch.readlines():
      #lcd_clear()
      lcd_string("lp " + linea,LCD_LINE_2)
      time.sleep(5.0)
    quit()
def lcd_init():
  # Initialise display
  time.sleep(E_DELAY)

def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command
  GPIO.output(LCD_RS, mode) # RS
  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)
```

```
# Send string to display

message = message.ljust(LCD_WIDTH," ")

lcd_byte(line, LCD_CMD)

for i in range(LCD_WIDTH):
  lcd_byte(ord(message[i]),LCD_CHR)
```

Con el código anterior lo que primero se realiza en programar los puertos GPIO de la Raspberry como salida para luego empezar a imprimir un texto ya sea en la línea 1 y 2, dado que el LCD es de (16x2). Seguido a esto se procede a imprimir las ip´s que en el código anterior habían respondido y se almacenaban como ip´s activas.

En el código anterior se encarga de leer línea a línea el contenido del archivo creado con la programación anteriormente mencionada para ser mostrada por la pantalla LCD de dos líneas.

```
if __name__ == '__main__':
 try:
   main()
 except KeyboardInterrupt:
   pass
 finally:
   lcd_byte(0x01, LCD_CMD) #limpia display
   lcd_string("     Goodbye!",LCD_LINE_1)
   time.sleep(8.0)
   lcd_byte(0x01, LCD_CMD)
   GPIO.cleanup()
```

Para finalizar se ha establecido un tiempo antes de limpiar la pantalla LCD y así termina la ejecución de esta programación.
(Marcelino García, 2014)

C. Comunicación serial RPI /Arduino

C-1. Programación RPI Arduino en Python

Este código final mostrara el contenido de todo el trabajo que se realizó para la integración entre placas sobre la Gestión de los sensores, monitoreo de red en las herramientas Raspberry Pi-Arduino, en el cual se detallara su funcionamiento.

```
#!/usr/bin/python
import itertools
from Tkinter import *
from subprocess import Popen
import os
import Tkinter
import sys
import time,serial, json
#arduino=serial.Serial('/dev/ttyACM0',9600)
```

En la primera parte es importante llamar cada uno de las librerías necesarias, que son fundamentales para el desarrollo de la programación en python.

C-2. Funcionamiento de líneas de código

Para la configuración del fichero que se encuentra en la librería **python.serial** se importó las funciones que se utilizaron para el puerto serie estableciendo la comunicación con el Arduino uno, configurando el puerto y los baudios para la transmisión de comunicación, cuando se introduzca un comando en un while-true se hace una petición que se enviara Arduino donde hay dos variables que son H y T para mostrarnos en nuestra interfaz la temperatura y la humedad y al mismo tiempo mandar un pulso de 5v para encender y apagar un actuador

```
ventana = Tkinter.Tk()
ventana.geometry("610x500+300+100")
ventana.title("Verificacion de IPs")
ventana.config(bg="white") #Color al fondo

var=StringVar()
```

```
label=Label(ventana,textvariable=var, font=("Agency FB", 12), bg="white",
justify=CENTER)#.place(y=10)
var.set("GESTION DE SENSORES Y MONITOREO DE \n RED BAJO HERRAMIENTAS
RASPBERRY PI-ARDUINO")
label.pack()

var1=StringVar()
label1=Label(ventana,textvariable=var1, font=("Agency FB", 12),
bg="white")#.place(x=240,y=40)
var1.set("SISTEMA DOMOTICO")
label1.pack(pady=25)
```

En las anteriores líneas de código se carga una ventana con **Tkinter,** al cual se le da todas las especificaciones y configuraciones necesarias (Python Foundation, 2016).

```
def update():
   ventana.after(15000,update)
   contador=1
   arduino=serial.Serial('/dev/ttyACM0',9600)
   arduino.readline()
   arduino.readline()
   while (contador<=1):
     caracter=arduino.readline()
     if caracter !='n':
         try:
           data=json.loads(caracter)
           H=data['Humedad']
           T=data['Temperatura']
           hum=Label(ventana, text=(H,"%"), font=("Agency FB", 15),
bg="white").place(x=253,y=125)
           temp=Label(ventana, text=(T,'*C'), font=("Agency FB",13), bg="White").place(x=416,
y=125)
           contador=contador+1
           arduino.close()
         except ValueError:
           print ""
```

En el anterior segmento de código lo que básicamente hace es abrir la comunicación serial entre Raspberry pi-Arduino donde se procede a capturar los datos que son transmitidos por el Arduino y el DHT 11 y son mostrados en la ventada creada en el apartado anterior para que los datos se envíen y se refresque cada 20 segundos aproximadamente (Sphinx, 2016).

```
def toggle(icycle=itertools.cycle([False, True])):
    state = next(icycle)
    led['text'] = str(state)
    if led['text']=="True":
     led.config(text="ON")
     label2=Label(ventana, image=on).place(x=65, y=60)
     os.system("sudo python /home/pi/rasp-on.py")
    else:
     led.config(text="OFF")
     label2=Label(ventana, image=of).place(x=65, y=60)
     os.system("sudo python /home/pi/rasp-off.py")

led = Button(ventana,text="ON-OFF", width=12, command=toggle)#.place(x=30,y=150)
led.place(x=30,y=150)
#led.pack(pady=80)

var3=StringVar()
label4=Label(ventana,textvariable=var3, font=("Agency FB", 12), bg="white").place(x=250,
y=280)
var3.set("IP ACTIVA EN LA RED")
#label4.pack(pady=160)
```

Finalizamos creando los botones de IP-OK es el que se encarga de hacer un descubrimiento de ip´s activas en nuestra red. El botón LCD es el encargado de mostrar cada una de las ip´s activas en nuestra red por intermedio de la pantalla LCD de dos líneas. El botón ON-OFF que sería el interruptor del actuador que tiene a cargo el Arduino, cabe resaltar que este actuador está controlado por una aplicación de celular o por la Raspberry PI. El ultimo botón es el que nos lleva directamente a la página de monitoreo de red implementado bajo NAGIOS (“Python Tkinter Label,” 2016).

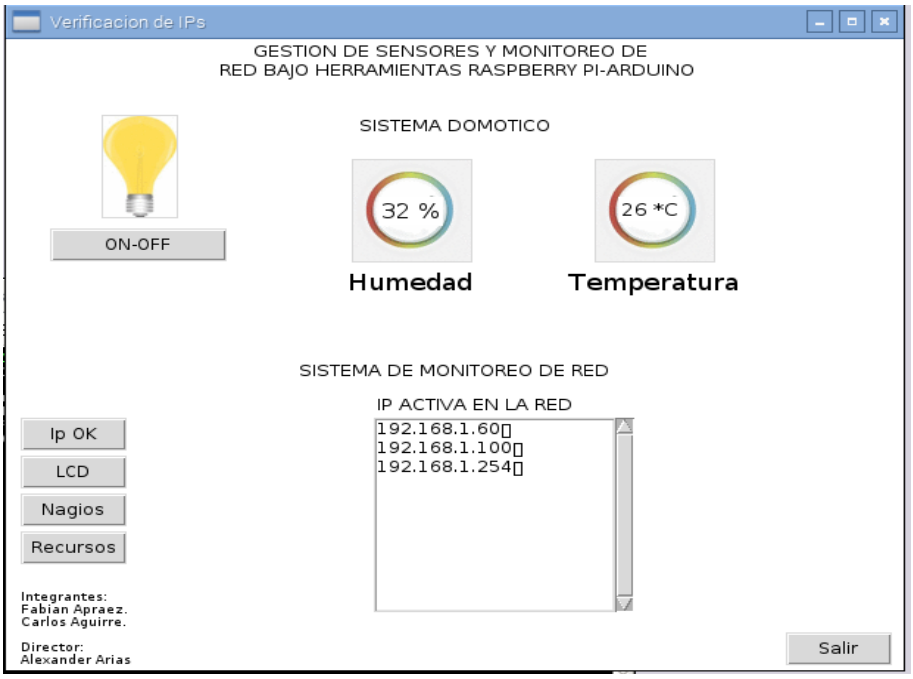


Figura 5.1 Interfaz de Usuario Tkinter.

C-3. Programación RPI en código Bash

Se configura a continuación con detalle el código utilizado para realizar el descubrimiento de las direcciones IP´s activas en nuestra red local, con el código que fue desarrollado bajo el lenguaje de programación BASH.

```
#!/bin/bash
rm /home/pi/ip.txt
echo 'Este es un Programa para ver cuál ip están activas'
ip addr show eth0 | grep -v inet6 | grep 'inet' | sed 's/^[ \t]*//' | cut -f2 -d" " 1> /home/pi/ip.txt
RED="$(cat ip.txt | sed -n "1 p" )"
echo '****************************************'
echo $RED
```

Mediante estas líneas de código lo que se logra básicamente es obtener la IP con la cual está configurada nuestra Raspberry pi, luego se procede a obtener la red y mascara a la que pertenece. Seguido a esto hace un escaneo de direcciones IP´s

que se encuentran activas en la red, cabe resaltar que estas IP´s activas son almacenadas en un archivo txt (Lenschinski, 2014).

El código anterior está desarrollado en BASH y es ejecutado desde Python. (Campanelli, 2009).

C-4. Recurso verificación IP y procesos activos

```
#!/usr/bin/python
from Tkinter import *
import os
import Tkinter
import commands
```

Llama todas las librerías necesarias para el desarrollo del trabajo.

```
ventan = Tkinter.Tk()
ventan.geometry("500x300+100+100")
ventan.title("Verificacion de IPs..")
ventan.config(bg="white")
var = StringVar()
label = Label(ventan,bg="white", textvariable=var, font=("Agency FB", 12)).place(x=5, y=10)
var.set("Verificacion de Ips..")

ip=StringVar()
vari=StringVar()
varis=StringVar()
```

Con el código anterior se carga una ventana con **Tkinter,** donde se establecen todos los parámetros y configuraciones necesarias.

```
entrada=Entry(ventan,textvar=ip).place(x=135, y=10)
```

Mediante el anterior código lo que realiza es capturar la ip que se digita en el cuadro de texto.

```
label2=Label(ventan,bg="white", relief=RIDGE,text="Tiempo Sistema Activo....",font=("Agency
FB",11)).place(x=5,y=75)
label3=Label(ventan,bg="white", textvar=vari,wraplength=300,justify=LEFT,font=("Agency
FB",11)).place(x=5,y=89)
```

```
label4=Label(ventan,bg="white", relief=RIDGE,text="Memoria Usada....",font=("Agency
FB",11)).place(x=5,y=105)
label5=Label(ventan,bg="white", textvar=varis,wraplength=490,justify=LEFT, font=("Ageny
FB",11)).place(x=5,y=120)
label6=Label(ventan,bg="white", relief=RIDGE,text="Disco Duro.... “,font=("Agency
FB",11)).place(x=5,y=135)
label7=Label(ventan,bg="white", textvar=varix,wraplength=370,justify=LEFT, font=("Ageny
FB",11)).place(x=5,y=150)
label8=Label(ventan,bg="white", relief=RIDGE,text="Carga del Sistema....",font=("Agency
FB",11)). place(x=5,y=165)
```

Imprime por pantalla toda la información recogida por el sistema de monitoreo “Nagios”.

```
def ejecutar(vara):
  va=commands.getoutput('/usr/lib/nagios/plugins/check_nrpe -H' +str(vara)+' -c
check_uptime')
  if va=="CHECK_NRPE: Error - Could not complete SSL handshake.":
    va= commands.getoutput('/usr/local/nagios/libexec/check_nt -H' +str(vara)+' -p 12489 -s
123456 -v UPTIME')
    vari.set(va)
  else:
    vari.set(va)

  va=commands.getoutput('/usr/lib/nagios/plugins/check_nrpe -H' +str(vara)+' -c check_load')
  if va=="CHECK_NRPE: Error - Could not complete SSL handshake.":
    va=commands.getoutput('/usr/local/nagios/libexec/check_nt -H' +str(vara)+' -p 12489 -s 123456 -v
CPULOAD -l 5,80,90')
    varif.set(va)
  else:
    varif.set(va)
```

Lo que realmente se encarga de hacer el anterior código es comunicarse con el sistema de monitoreo Nagios y traer cada uno de los valores que se le preguntan. Como Memoria, Disco duro, Tiempo de actividad del servidor entre otros. Para luego ser mostrados en pantalla.

```
def Salir():
  ventan.quit() # Muestra una ventana
```

```
boton1=Button(ventan,bg="white",text="Aceptar", command=lambda: ejecutar(ip.get()),
    width=6).place(x=10,y=40)

boton2=Button(ventan,bg="white",text="Cerrar",
command=Salir,width=6).place(x=420,y=250)

ventan.mainloop()
```

Son dos botones que se crean en la ventana con Tkinter y que cumplen diferentes actividades, por ejemplo, el botón 1 ejecuta todo el código anteriormente descrito. Mientras el botón 2 se encarga de salir de la ventana creada con Tkinter en Python (Python Foundation, 2016).

D. Diagramas general de flujo

Este diagrama explica el funcionamiento de cada uno de los botones que se colocaron en la interfaz que se desarrolló en la herramienta Raspberry pi, de manera lógica de como captura envía y muestra la información

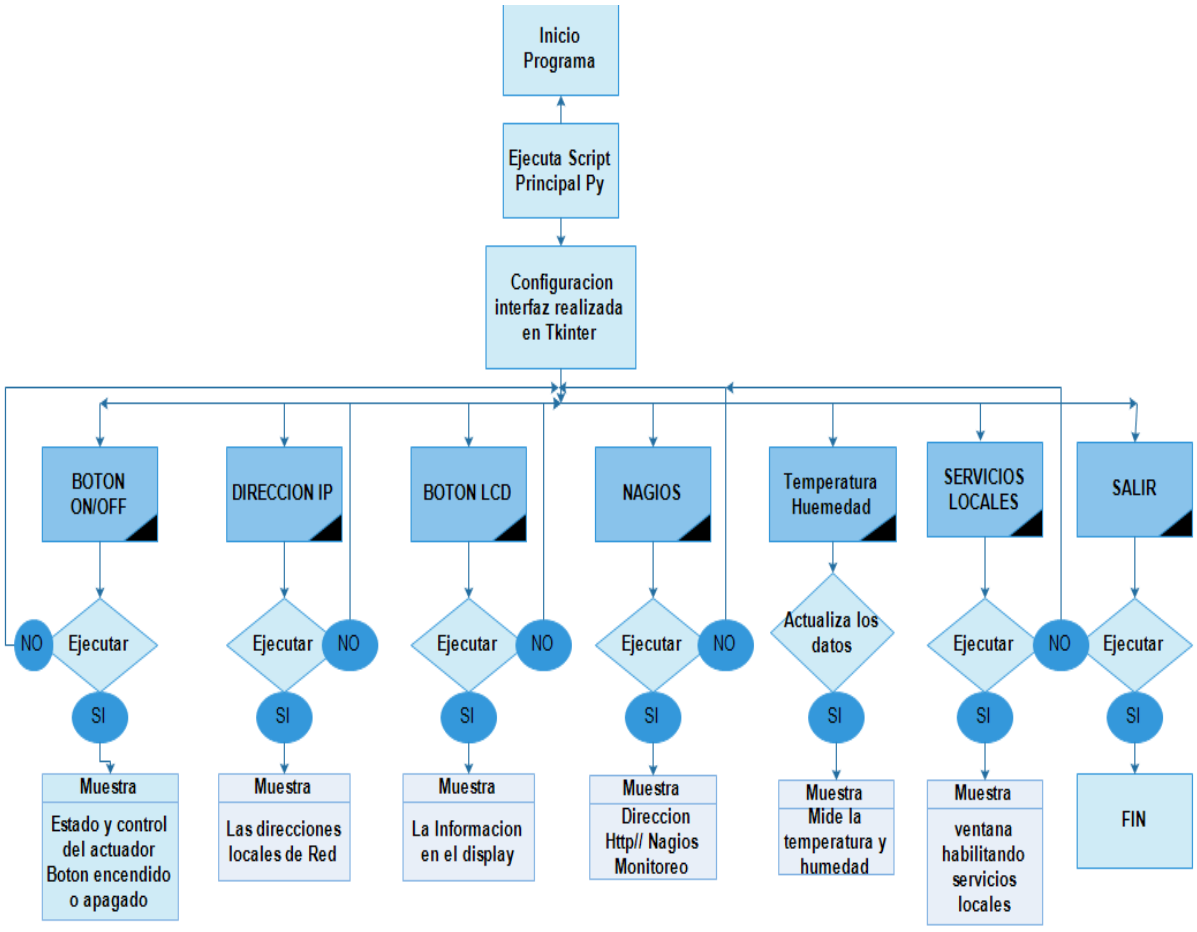
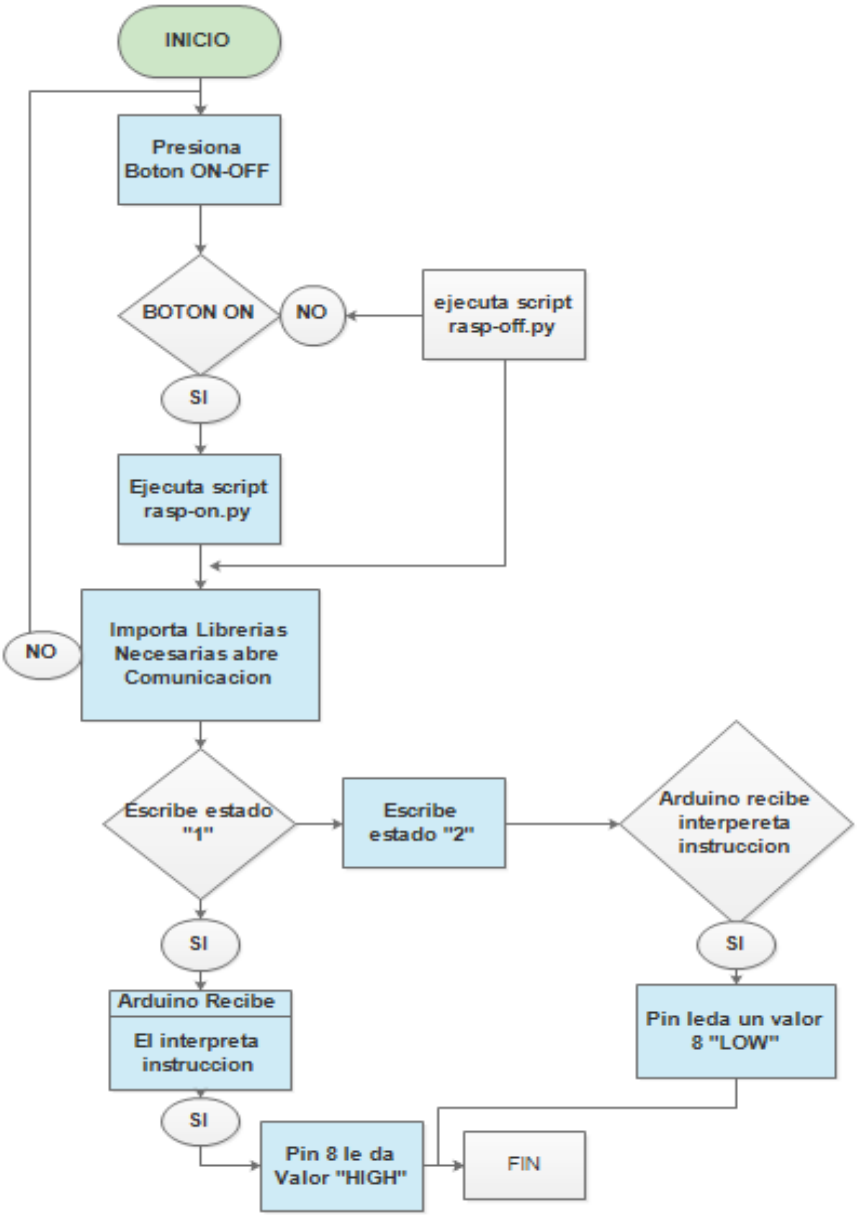
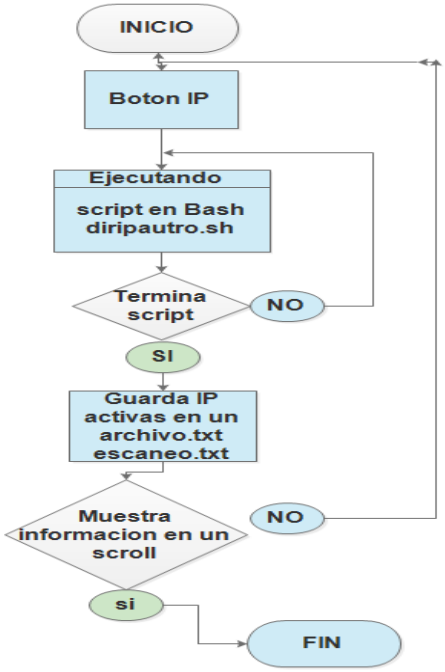


Diagrama Botón OFF/ON

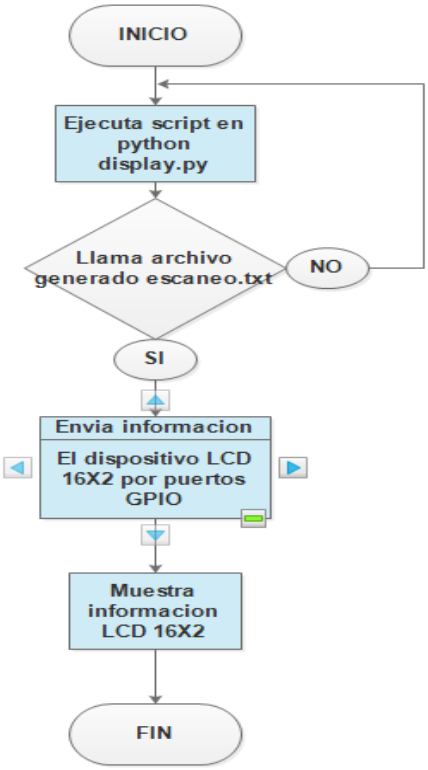
Enciende y apaga el actuador que se configuro en los ficheros de configuración de la RPI.





Boton de direcciones IP

El Scroll o ventana muestra las direcciones activas que arroja nuestra red local, para detectar que direcciones ip estan en nuestra Red.



Botón Python Display

Muestra la información de direcciones ip y los servicios locales en nuestra LCD.

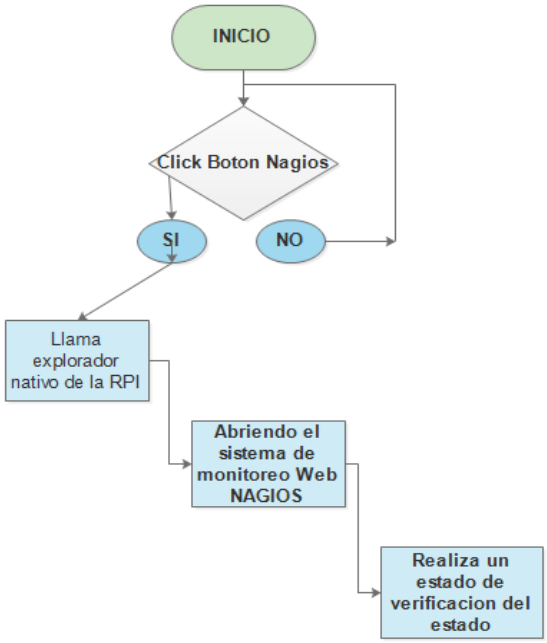


Diagrama de botón Nagios web

Manda la dirección del servidor NAGIOS para autenticar y verificar los servicios activos y pasivos que se muestra en la interfaz.

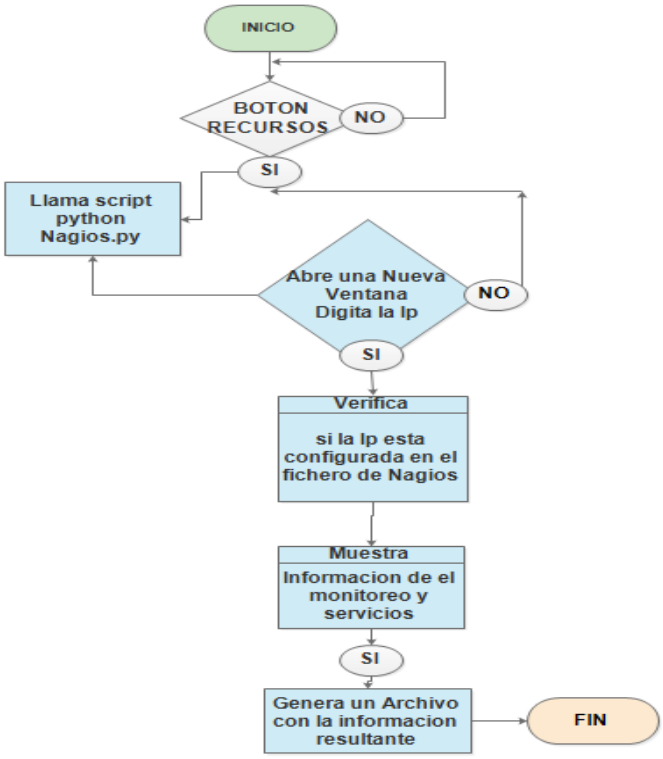


Diagrama Botón Recursos

Abre una nueva ventana emergente para diagnosticar la direccion IP de cada equipo que esta en la red, y saber que servicios estan activos.

Bibliografía

Arriola Navarrete, O. (2011). Open access y software libre: un área de oportunidad para las bibliotecas. *Biblioteca Universitaria*, *14*(1), 35–38. Retrieved from http://www.revistas.unam.mx/index.php/rbu/article/view/27169/25272

Campanelli, G. (2009). Demasiado Personal: Bash desde Python. *Demasiado Personal*. Retrieved from http://gedece.blogspot.com.co/2009/10/bash-desde-python.html

Chacon M., O. P. (2014). *Análisis para un sistema domótico con la arquitectura Arduino y Raspberry Pi, sobre TCP/IP.* (Tesis de Pregrado). Universidad del Azuay, Ecuador.

Esteban, E. V. B. (2014). *Lenguaje C*. Retrieved from http://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf

Gonz, R. R. (2015). Instalación y configuración de nagios core 4.0.4. Retrieved from https://exchange.nagios.org/components/com_mtree/attachment.php?link_id=6527&cf_id=24

Hamachi, D. L. (2016). LogMeIn Hamachi Guía de primeros pasos. Retrieved from https://secure.logmein.com/ES/welcome/documentation/ES/pdf/Hamachi/LogMeIn_Hamachi_GettingStarted.pdf

Herrmann, J. A., & Salgado, F. (2014). *Implementación de Sistema Domótico con Servidor Raspberry* (Tesis de Pregrado). Universidad Politécnica de Madrid, Madrid.

Lee, S., Levanti, K., & Kim, H. S. (2014). Network monitoring: Present and future. *Computer Networks*, *65*, 84–98. https://doi.org/10.1016/j.comnet.2014.03.007

Lenschinski, E. (2014). networking - Checking host availability by using ping in bash scripts - Stack Overflow.

Marcelino García. (2014, April). Python: Pantalla LCD 16x2 Raspberry Pi - HeTPro. Retrieved from http://hetpro-store.com/TUTORIALES/python-lcd-raspberry-pi/

Motorola. (2014). Moto G™. Retrieved from http://d2y1qqq73yfji4.cloudfront.net/media/manual/files/Moto G LTE.pdf

Python Foundation. (2016). Graphical User Interfaces with Tk — Python 3.5.1 documentation. Retrieved from https://docs.python.org/3/library/tk.html

Python Tkinter Label. (2016). Retrieved from https://www.tutorialspoint.com/python/tk_label.htm

Raúl C., (@raulconm). (2015). Guia de Iniciacion App Inventor Raúl C. (@raulconm). Retrieved from http://codeweek.eu/resources/spain/guia-iniciacion-app-inventor.pdf.

Richmond. (2014). Raspberry Pi for beginners. In H. Kelly (Ed.), *booazine series* (pp. 125–131). Aaron Asadi. Retrieved from http://www.imagine-publishing.co.uk/

Rojas, F. M., & Puentes, Y. J. R. (2013). Arduino y Android una Pareja para Aplicaciones de Ubicuidad. Retrieved from http://www.laccei.org/LACCEI2013-Cancun/RefereedPapers/RP060.pdf.

Sphinx. (2016). JSON encoder and decoder — Python documentation. Retrieved from https://docs.python.org/3/library/json.html

Summerfield, M. (2013). *Python in Practice: Create Better Programs Using Concurrency, Libraries, and Patterns*. Retrieved from papers3://publication/uuid/79DB00E2-5F8C-405C-85B7-E3BBEE8BB989

Uk, D. (2010). Temperature Sensor DHT 11 Humidity & Temperature Sensor. *DHT11 Datasheet*, 9. Retrieved from http://www.micropik.com/PDF/dht11.pdf

Wavesen. (2014). HC Serial Bluetooth Products User Instructional Manual. *Www.Wavesen.Com*. Retrieved from www.wavesen.com

Windows, D., Ken, P. O. R., & Nils, H. (2013). Jefe terminal. *Xrdp Ayuda a Los Clientes de Terminal Windows a Conectarse a Linux*, 29–31. Retrieved from WWW.LINUX-MAGAZINE.ES

Xmartic. (2013, July). Nagios NRPE – Supervisando máquinas Linux. Retrieved from http://www.nagios-cl.org/nagios-nrpe-supervisando-maquinas-linux

Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

FIRMA ESTUDIANTES Carlos Andres Aguirre

Javier O. Aparicio C.

FIRMA ASESOR

FECHA ENTREGA: 9/Nov/2016

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD

_____

RECHAZADO_____ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____