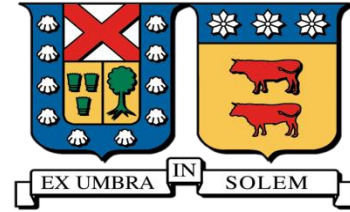


Universidad Técnica Federico Santa María.

Departamento de Electricidad.

T.U. en Automatización y Control.



IoT & Raspberri Pi

Integrante: Hans Pinochet

Profesor guía: Helmut Contreras

Año 2018

Dedicatoria

En este espacio, quisiera nombrar a esas personas que me impulsaron día a día para llegar hasta aquí, aquellas que en momentos de oscuridad me brindaron de esa energía necesaria para seguir adelante y cumplir cada una de mis metas y sueños, sin importar el tamaño de ellos.

Primero que todo agradecer a mis padres, Flavio y Sandra; que sin su guía a lo largo de mi corta vida no estaría escribiendo estas palabras.

A mi amados, Carolina y nuestro hijo Luciano Valentín . Su sonrisa es mi motor para levantarme todas las mañanas y superarme como alumno, como padre, como humano. Gracias por todo su apoyo y ánimo en esta aventura.

A mis amigos, familia y profesores, este nuevo logro es en gran parte gracias a todos ustedes; quisiera dedicarles mi trabajo de título a ustedes, personas de bien, seres que ofrecen amor, bienestar y alegría a todo aquel que lo necesita.

Índice

INDICE DE MATERIAS

<u>Introducción</u>	2
----------------------------------	---

Capítulo 1: Marco Teórico, IoT

<u>1. HISTORIA</u>	3
<u>1.1. ORIGEN Y EVOLUCIÓN</u>	4
1.1.1. <u>Siglo XIX</u>	4
1.1.2. <u>20's – 50's</u>	5
1.1.3. <u>60's-80's</u>	6
1.1.4. <u>90's</u>	8
<u>2. DEFINICION</u>	9
<u>2.1. FUNCIONALIDAD</u>	11
<u>2.2. CARACTERÍSTICAS</u>	11
2.2.1. <u>Ventajas</u>	12
2.2.2. <u>Desventajas</u>	12
<u>2.3. APLICACIONES</u>	13

Capítulo 2: Raspberry & Python

<u>1. RASPBERRY</u>	16
<u>1.1. HISTORIA</u>	17
<u>1.2. CARACTERÍSTICAS</u>	18
<u>2. PYTHON</u>	20
<u>2.1. HISTORIA</u>	20
<u>2.2. CARACTERÍSTICAS</u>	23
<u>3. CODIGO DE PYTHON</u>	24

Capítulo 3: Software de Estación climática

1. <u>SEÑALES DE ENTRADA Y SALIDA</u>	34
1.1. <u>GPIO</u>	34
1.2. <u>PUERTO SERIE</u>	37
1.3. <u>PLATAFORMAS IOT</u>	40
1.4. <u>PYTHON PIP</u>	41
2. <u>WEEWX</u>	42
2.1. <u>CODIGO</u>	43
2.2. <u>INSTALACIÓN</u>	48
3. <u>WEATHER-UTIL</u>	52
3.1. <u>SCRIPT</u>	54
3.1.1. <u>Código</u>	55
3.2. <u>MAILUTILS Y SSMTP</u>	57
3.3. <u>CRON</u>	60
<u>Conclusión</u>	62
<u>Bibliografía</u>	64

INDICE DE FIGURAS

<u>Figura 1-1. Nodos Iniciales</u>	7
<u>Figura 1-2. Definiciones del IoT</u>	9
<u>Figura 1-3. Interacciones que permite el IoT.</u>	10
<u>Figura 2-1. Tabla de operaciones booleanas.</u>	30
<u>Figura 3-1. Esquema de GPIO.</u>	35

INDICE DE TABLAS

<u>Tabla 1-1. Características Raspberry Pi 3 Modelo B+</u>	18
--	----

Introducción

IoT. Hace aproximadamente 20 años esta palabra era apenas conocida por un selecto grupo de personas cercanas al área de la electrónica, la programación y la investigación.

Kevin Ashton, profesor del MIT, en aquel tiempo, utilizó esta expresión por primera vez de forma pública, volviéndola en uno de los tópicos con mayor crecimiento y expectación de la última década.

Hoy en día el “Internet de las cosas” (acuñado IoT por sus siglas en inglés) es un tema ya extendido y comprendido por la mayoría del público, hecho que permite pensar en las vastas posibilidades de este mismo.

Por ello, a través de este documento, exploraremos el campo de la intercomunicación máquina a máquina y su implementación a través de un micro controlador.

Para orientarnos a través de nuestra investigación, los objetivos establecidos son los siguientes:

Objetivo General:

- Aplicación tarjeta Raspberry PI en Internet de las cosas.

Objetivos específicos:

- Comprender el funcionamiento del internet de las cosas a través del estudio de los servicios disponibles.
- Determinar las capacidades de la tarjeta Raspberry a través del estudio de esta.

- Lograr comprender la programación del dispositivo a través del estudio del lenguaje de programación que promueve, Python.
- Demostrar las capacidades del internet de las cosas a través de una implementación con el microcontrolador Raspberry.

Capítulo 1: Marco Teórico, IoT

1. **HISTORIA**

Como hablamos en la introducción, Kevin Ashton fue el precursor de este término, utilizándolo por primera vez en 1999, en el título de una presentación que realizó a la empresa Procter & Gamble.

Ahora bien, no se utilizó de manera pública hasta 2009, cuando el profesor tuvo una presentación en el RFID journal, donde explicó el potencial que tiene el IoT:

“Si tuviésemos ordenadores que fuesen capaces de saber todo lo que pudiese saberse de cualquier cosa –usando datos recolectados sin intervención humana- seríamos capaces de hacer seguimiento detallado de todo, y poder reducir de forma importante los costes y malos usos. Sabríamos cuando las cosas necesitan ser reparadas, cambiadas o recuperadas, incluso si están frescas o pasadas de fecha. El Internet de las Cosas tiene el potencial de cambiar el mundo como ya lo hizo internet. Incluso más.”

Con esta frase, se abrió paso a un mundo de posibilidades, solo limitadas por la imaginación de uno mismo. Pero, a pesar de la creencia popular, su origen se remonta a muchísimos años atrás y para ello, debemos analizar la evolución tecnológica que nos permitió llegar hasta acá.

1.1 ORIGEN Y EVOLUCIÓN

A través del presente capítulo profundizaremos la historia que atraviesan las distintas disciplinas, tales como informática, telemetría, electricidad, electrónica, entre muchas otras, para culminar en lo que hoy en día conocemos como IoT. Para ello, debemos analizar un amplio rango de fechas y acontecimientos, siendo cada una de ellas un pilar fundamental para el desarrollo tanto de la tecnología que conocemos y utilizamos actualmente.

1.1.1 Siglo XIX

Si queremos conocer sus principios, debemos remontarnos a muchísimos años atrás, exactamente a 1874, donde se produjeron los primeros experimentos registrados de telemetría en la historia de la humanidad, llevados a cabo por científicos franceses.

La investigación consistía en instalar dispositivos de información meteorológica y de profundidad de nieve en la cima del Monte Blanco, montaña ubicada entre Francia e Italia y que tiene una altura de 4800 metros. Mediante un enlace de radio de onda corta, los datos eran enviados a París.

1.1.2 20's – 50's

El concepto de interconectar los objetos y que estos sean inteligentes estuvo muy presente en los pensamientos e ideas de científicos de gran calibre y renombre, como Nikola Tesla, “el padre de la electricidad” o Alan Turing “el padre de la inteligencia artificial”.

Ejemplo de ello, es la entrevista otorgada a la revista Collier's en 1926, donde anticipó el crecimiento lo inalámbrico y de comunicación:

“Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico... y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo.”

De igual manera, Alan Turing en 1950 escribió un artículo en el Computing Machinery and Intelligence in the Oxford Mind Journal, en el cual anticipaba la futura necesidad de darle inteligencia y comunación a los dispositivos:

“... también se puede sostener que es mejor proporcionar la máquina con los mejores órganos sensores que el dinero puede comprar, y después enseñarle a entender y hablar inglés. Este proceso seguirá el proceso normal de aprendizaje de un niño.”

Ahora bien, para la desdicha de muchísimos, aunque estos y muchos más grandes pensadores de la época tenían grandes ideas relacionadas a este tópico, estas nunca lograron materializarse, por la simple razón de que la tecnología en ese periodo era demasiado inmadura, restringiendo así la posibilidad de realizar experimentos, logrando un truncamiento por un par de años.

1.1.3 60's-80's

Se tuvo que esperar hasta finales de la década de los 60 y mayoritariamente de la década de los 70 para empezar a ver los primeros protocolos de comunicaciones, que se volvieron en la base de lo que hoy conocemos como Internet. Este crecimiento se logró gracias a la creación de la **Red ARPANET**.

La Red ARPANET, que debe su nombre a sus siglas en inglés Advanced Research Projects Agency Network, fue creada por el Departamento de Defensa de los Estados Unidos, para poder establecer una comunicación entre los distintos organismos gubernamentales que existían en la nación. Para lograr esta conexión, fue necesario el desarrollo de dos temas independientes; por un lado la generación de redes de datos, la arquitectura y su implementación, y por el otro lado la creación de ARPA (Agencia de Investigación de Proyectos Avanzados)

Joseph Carl Robnett Licklider, informático teórico, psicólogo e investigador de la inteligencia artificial que en 1962 comenzó a experimentar con la conexión de dos o más equipos. Este mismo año hace público su concepto de “Red Galáctica”, planteando la idea de poder tener computadores interconectados globalmente.

Para volver realidad esta idea, se tuvo que esperar a 1964, donde Leonard Kleinrock, científico en computación y profesor de la Universidad de California (UCLA), publica su primer libro sobre la teoría de la conmutación de paquetes, en el cual trataba la comunicación de paquetes en vez de circuitos, hecho que permitió un enorme avance en el área de las redes informáticas.

Ya en 1965, se logra la comunicación entre dos computadores; un TX-2 localizado en Massachusetts con un Q-32 ubicado en California a través de la línea telefónica conmutada de baja velocidad, consumando así la primera red de área amplia del mundo.

Ya llegando al final de la década, en 1969 la empresa de alta tecnología BBN instala el primer IMP (Procesador de mensajes de interconexión) en UCLA y se conectó en Stanford Research Institute (SRI) con lo cual se obtuvo el segundo nodo. Un mes después se añadieron dos nodos más, en la Universidad de California de Santa Bárbara y en la Universidad de Utah, logrando 4 nodos para la etapa inicial de ARPANET, dándole punto de partida a lo que hoy conocemos como internet.

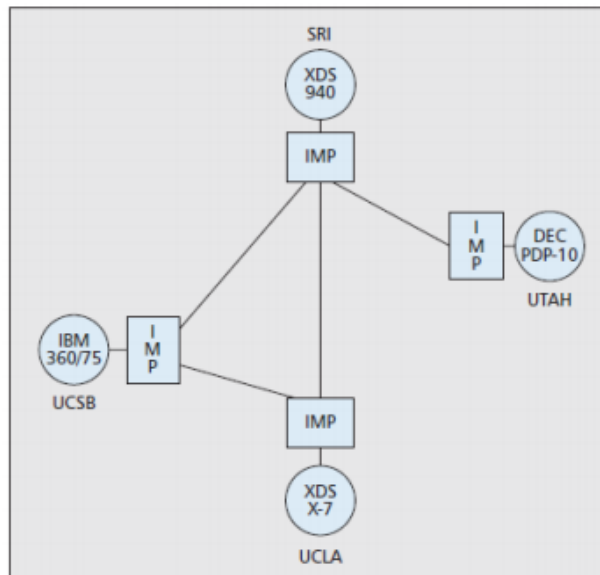


Figura 1-1. Nodos iniciales

Ya en la década de los 70 y 80 el avance de estas redes se vio deteriorada por varios motivos, los principales fueron:

- ARPANET fue creado exclusivamente para uso militar y académico.
- Hubo una carencia de comunicaciones rápidas y de bajo costo para largas y medias distancias.

- La falta de normas, que permitió la creación de distintos protocolos de comunicación que eran incompatibles entre ellas.

1.1.4 90's

En este periodo se logró dar a luz lo que hoy conocemos como Internet. Se creó un protocolo de comunicaciones estándar, el TCP/IP, que permitió un modelo universal, logrando sacar del mercado a las redes no estandarizadas y dando origen a la que hoy nos permite conectarnos globalmente.

Fue tanta la fascinación que creó el Internet, que en 1990, se logró volver realidad la conexión a objetos. John Romkey, desarrollador del protocolo TCP/IP, en el evento INTEROP, en Estados Unidos, presentó una tostadora que se podía apagar y prender en remoto, convirtiéndola en el primer objeto conectado a internet.

Después de este acontecimiento la idea de la conexión de objetos sufrió una baja por el principal enemigo en ese momento, el hecho de que todo era cableado. Sumado a la eventualidad de que en esos años el hardware tenía un costo demasiado elevado, mermo los avances en esta área por algunos años.

La última revolución se logró gracias a la “popularización de conectividad inalámbrica”, ya fuera en forma bluetooth o Wi-Fi, ya que vienen incluidos en casi todos los smarthpones que se fueron lanzando al mercado desde comienzos del siglo XXI.

Esto igualmente permitió la creación de nuevos conceptos como el WSN (Wireless Sensor Networks) o M2M (Machine to Machine).

2. DEFINICION

Para definir este tema, ocupamos una imagen que nos permita entenderlo con mayor facilidad.



Figura 1-2. Definiciones del IoT

Gracias a esto, podemos definir el concepto de IoT. Esta misma es una gran red de información, basada en elementos con inteligencia propia que pueden reconocer el entorno a su alrededor para facilitar tareas al usuario, trabajos ya sea cotidianos como complejos procesos industriales, generando comodidad y bienestar.

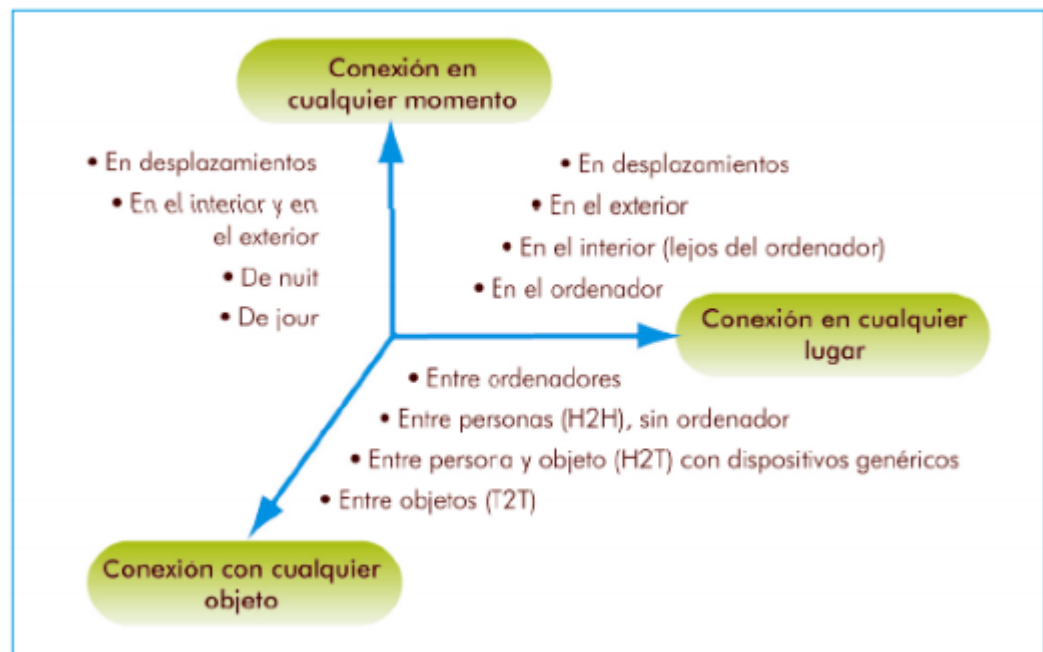


Figura 1-3. Interacciones que permite el IoT.

Esta nos permite la interconexión de objetos con el medio y la transmisión continua de información en base al censo realizado por ejemplo, sensores. Nos da una amplia gama de contactos, ya sea entre ordenadores, personas, persona y objeto, entre otras. Nos concede información en tiempo real en cualquier momento y lugar.

2.1 **FUNCIONALIDAD**

La forma en que funciona es relativamente simple, se basa en el concepto de la facultad de los objetos de conectarse de forma remota. Todos los objetos que se conectan al ciberespacio tienen una MAC. La dirección MAC (Media Access Control) es la dirección física dada por el fabricante y es única para cada dispositivo. Esta misma permite identificarlas y controlarla, ya sea para que realice una acción como para enviar datos.

2.2 **CARACTERÍSTICAS**

Analizando el IOT, podemos ver que tiene una gran gama de características, pero nos avocaremos a las 3 principales:

- **Conexión:** Básicamente la razón de existir de nuestro concepto. Lograr el flujo de información a través de la red entre 2 objetos, de forma rápida, accesible y simple.
- **Sensibilidad:** A los objetos se les otorgará la facultad de “sentir” el mundo real, creando la ilusión de que tienen sentidos. Esto les permite un dominio de la

información obtenida y sabiendo en muchos casos cuando los datos obtenidos son valiosos o no.

- **Interrelación:** El hecho de que el objeto tenga la capacidad de comunicarse permite una interacción más cercana con el usuario, creando un cierto sentimiento de familiaridad, concepto muy difícil de encontrar en la tecnología de hoy en día.

2.2.1 Ventajas

- **Información en tiempo real:** la información que obtenemos es prácticamente instantánea y puede ser obtenida de cualquier lugar con conexión a internet. Este hecho es muy importante cuando se tiene que controlar variables críticas por ejemplo.
- **Versatilidad:** uno de sus mayores puntos a favor y una de las principales razones por la cual la gente quiere implementar el IoT en sus vidas. La capacidad casi camaleónica de poder mezclarse y aplicarse en prácticamente todo, desde pequeños proyectos domésticos y cotidianos hasta grandes proyectos con grandes envergaduras con una extrema facilidad.
- **Bajo costo:** es una realidad innegable el hecho de que hoy en día se busca siempre un acorde entre calidad/precio, permitiendo la introducción del IoT en el mercado gracias a los económicos micro controladores y sensores que se utilizan

2.2.2 Desventajas

A pesar de todos los puntos fuertes del IoT, no podemos negar que como todo en este mundo, tiene cosas buenas como malas, por ello no podemos dejar de lado el lado desfavorable de este mismo.

- **Seguridad y privacidad:** Uno de los problemas que afecta al mundo del internet es el hecho de exponer al usuario a un posible hackeo o robo de información, algo muy recurrente y que por desgracia no tiene un remedio absoluto.
- **Complejidad:** El uso de esta tecnología puede producir un cierto desaliento para personas no tan acometidas con el mundo de la electrónica o la programación, aunque hay que recalcar que el paso de los años ha ayudado a mejorar este punto y volverlo más amigable con el usuario.

2.3 APLICACIONES

Como lo hemos repetido a través de este documento, el IoT puede ser aplicada hasta donde nuestra imaginación nos permita, por los que nos llevaría muchísimo tiempo enumerar o nombrar todas sus aplicaciones.

Lo que si podemos hacer es sub dividirlo en 3 categorías principales:

- IoT en el uso doméstico: cuando pensamos en su uso particular, se nos viene rápidamente a la mente lo que es la domótica y la inmotica, que es la automatización de casas y edificios enteros respectivamente.
- IoT en el uso industrial: es bien sabido que las industrias en su mundo tan competitivo y demandante prácticamente se les obliga a mejorar constantemente sus herramientas, ejemplo de ello es poder ver en zonas mineras autos controlados de forma remota y a través del IoT, controlar almacenes de logística al llevar un conteo en tiempo real del inventario,

acumular datos de interés para la industria y hasta manipular estos factores cuando son críticos.

- IoT en las ciudades: un ejemplo verdaderamente simple y real de este caso es la masivamente famosa aplicación para conductores, Waze, que gracias a la navegación asistida y la transmisión en tiempo real, nos permite saber el estado de las calles a todos sus usuarios por igual, permitiendo a ellos mismos dar aviso de algo importante a su alrededor y alertar al resto de los navegantes.

Al comienzo de este capítulo se empezó profundizando el origen y evolución de los elementos que permitieron la creación del IoT. Fue un proceso largo con altos y bajos pero que nos permitió al mismo tiempo entender el por qué el termino es tan poco conocido siendo algo incluso tan rutinario en la vida del común pensante. De forma paralela, se logró comprender las diferentes formas en que puede ser aplicado, demostrando la amplia gama de usos que tiene y dejando la puerta abierta a más posibilidades según el usuario lo desee.

Capítulo 2: Raspberry & Python

1. **RASPBERRY**

En términos genéricos, la Raspberry Pi es una placa de computadora de tamaño y precio reducido tipo SBC (Single Board Circuit) que se traduce como una computadora completa en un solo circuito. Funciona con un Sistema en Chip o SoC por sus siglas en inglés.

El concepto es un computador pero eliminando todos los accesorios que no alteren su funcionamiento básico. La placa soporta varios componentes utilizados en un computador normal permitiéndole trabajar como tal.

“Puede usarlo para aprender a programar y crear proyectos de electrónica, y para muchas de las cosas que hace su PC de escritorio, como hojas de cálculo, procesamiento de textos, navegación en Internet y juegos. También reproduce video de alta definición. La Raspberry Pi está siendo utilizada por adultos y niños de todo el mundo para aprender programación y creación digital.”

Página oficial de Raspberry.

1.1 HISTORIA

La idea de este proyecto nació en 2006, por un grupo de desarrolladores de la Universidad de Cambridge. El proyecto surgió del ideal de promover la enseñanza de la computación en los niños.

Los primeros diseños de la Raspberry Pi se hicieron a base del micro controlador marca Atmel ATmega 644, que funcionaba con 8 bits y una memoria flash de 64 kB.

En mayo de 2009 se crea la **Fundación Raspberry Pi**, en Caldecote, South Cambridgeshire, Reino Unido.

El responsable de la arquitectura de software y hardware de la Raspberry pi fue el co-fundador Eben Upton, antiguo trabajador de la empresa Broadcom, quien se contactó con profesores, estudiantes y simpatizantes de la informática para promover y animar a los niños a aprender de la informática a través de este microcontrolador.

En agosto de 2011, se fabricaron 50 placas Alpha del modelo inicial, el Modelo A; y en Diciembre de 2011, se fabricaron 25 placas Beta del modelo B.

El primer lote creado fue de 10.000 placas y se fabricaron en Taiwán y China en vez del Reino Unido como todo el mundo esperaba, esto debido a que estos países ofrecían unas tarifas más baratas de producción y plazos de entrega del producto más rápidas.

Se empezó a vender oficialmente el 29 de febrero de 2012 (Modelo B). Las dos tiendas donde se vendían eran Premier Farnell y RS Components y en 6 meses llegarían a vender 500.000 unidades.

El 6 de Septiembre se anunció que la producción de las placas se haría en Reino Unido, en una fábrica de Sony y que proveería 30.000 unidades mensualmente, creando así 30 nuevos puestos de trabajo.

El 4 de Febrero de 2013, se lanzó el modelo A, que venía con solo 256Mb de RAM y sin puerto Ethernet, pero con un precio más módico que el modelo B.

1.2 CARACTERÍSTICAS

Como elección de modelo a analizar se optó por la **Raspberry Pi 3 B+**, por las mejoras que ha trae consigo este modelo, las cuales se expondrán a continuación.

Especificaciones	Raspberry Pi 3 B+
CPU	Broadcom BCM2837B0 Cortex-A53 (ARM V8) 64-bit, 1.4GHz
GPU	VideoCore IV
RAM	1 GB
Almacenamiento	Micro SD
Puertos	HDMI 4 USB 2.0 CSI camara DSI display
Conectividad	Ethernet de hasta 300Mbps Bluetooth 4.2

	Wi-Fi 802.11 b/g/n/ac
GPIO	40 pines
Alimentación	2.5 A – 5 V vía micro USB

Tabla 1-1. Características Raspberry Pi 3 Modelo B+

Este modelo ha destacado por traer pequeñas, pero muy importantes mejoras tales como:

- Aumento en la frecuencia de la CPU, siendo el primer modelo en llegar a 1,4 GHz.
- Mejor disipación del calor, evitando que lleguen a quemarse componentes y dándole una vida útil aún más longeva.
- Aumento significativo de la conexión Ethernet, pasando de 100 a 300 Mbps, resolviendo el problema de cuello de botella a la hora de traspasar información al PC.
- La conexión Wi-Fi se ha mejorado teniendo ahora una banda dual que nos permite llegar a los 100 Mbps en redes de 5GHz.
- La posibilidad de alimentación por PoE con un HAT.

2. **PYTHON**

Python es un lenguaje de programación interpretado, lo que significa que no es necesario compilar el código fuente para ejecutarlo. Tiene una estructura de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos.

Su sintaxis elegante permite un código legible, junto a su tipado dinámico y el hecho de ser multiplataforma le permite trabajar con sistemas operativos como Unix, Windows, OS/2, Mac, Amiga, entre otros, convierten a Python en uno de los lenguajes de programación más populares en el mundo ideal para el desarrollo de aplicaciones y páginas web.

2.1 **HISTORIA**

Cuando hablamos de la historia de Python, debemos remontarnos a comienzos de los noventa al Centro para las matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica) lugar perteneciente a los Países Bajos , donde el europeo Guido van Rossum se planteó el objetivo de crear un nuevo lenguaje de programación que fuera el sucesor del lenguaje de programación ABC, un lenguaje creado por este mismo centro y el cual el mismo Guido ayudó a crear. A pesar de que ABC se había creado como una

alternativa para el lenguaje BASIC y este mismo destacaba por su facilidad de aprendizaje y uso para principiantes no tuvo un gran auge por el elevado precio del hardware necesario para su uso.

Guido Van Rossum nombro al lenguaje en honor al grupo de humoristas británicos Monty Python del cual él era fanático.

En 1991, Van Rossum lanzó al público el código de la versión 0.9.0 en alt.sources. A pesar de ser una versión primitiva a comparación con la que se conoce actualmente, esta ya tenía disponible clases con herencias, funciones y otros elementos que le permitieron partir con el pie derecho.

En 1994 se concibieron 2 hitos importantes para Python, el primero fue comp.lang.python, un foro para los usuarios de Python donde podían compartir ideas y crear debates relacionados a este mismo, hecho que aumento su popularidad; y el segundo fue el lanzamiento de la versión 1.0.

1995 fue un año de cambios, siendo este el último año donde la compañía CWI estuvo trabajando en Python, liberando la versión 1.2 antes de que van Rossum continuara su trabajo bajo el ala de la Corporation for National Research Initiatives (CNRI) en Reston, Virginia, corporación con la cual lanzó muchas versiones del software.

Durante los años que estuvo en CNRI van Rossum creó la iniciativa Computer Programming for Everyone (CP4E), cuyo objetivo era hacer más accesible la programación a la gente pero gracias a la simplicidad del mismo Python en sí, esto no se volvió una prioridad, dando como resultado que el proyecto fuera dado de baja en 2007.

El comienzo del siglo XX vino de la mano con cambios como el hecho de que el equipo principal de desarrolladores de Python se cambió a BeOpen.com para en conjunto dar a luz al equipo de BeOpen Python Labs. CNRI pidió que la versión fuera lanzada al momento que el equipo se cambiara.

Aquí se lanzó el famoso Python versión 2.0, el cual tuvo una gran adición del lenguaje de programación funcional Haskell. Haskell es un lenguaje de programación con el lema

“una función es un ciudadano de primera clase”, el cual no obliga al programador a declarar que tipo es el dato ingresado y si lo hace, le da al programador una amplia gama de tipos, incluyendo los que más se ocupan en la actualidad, siendo int, float, bool, double, char, entre otros.

En 2001 se crea la Python Software Foundation, fundación que desde la versión 2.1 es la dueña de todo el código, documentación y especificaciones del lenguaje.

El último gran cambio en el lenguaje fue la actualización a Python 3.0 en el año 2008, el cual se diseñó para rectificar las fallas fundamentales que se fueron encontrando en el diseño del lenguaje.

Una de estas fallas fue el hecho de que al ir acumulando actualizaciones, igualmente acumuló muchas formas distintas e innecesarias de realizar la misma tarea, por lo que Python 3.0 se centró en cambiar esto al eliminar constructores duplicados y módulos con el objetivo de tener un solo modo obvio de realizar las cosas.

Al comienzo Python 2.x y Python 3.x fueron pensadas para convivir en paz mientras se liberaban actualizaciones de ambas versiones al mismo tiempo, ejemplo de ello fue que la versión 2.6 y 3.0 pero todo esto cambio en 2010.

En esta fecha se liberaban la versión 2.7 y 3.1 sin embargo se informaba a los usuarios que esta sería la última actualización de la versión 2.X ya que Guido van Rossum quería solo concentrarse en Python 3.

La versión 2.7 siguió recibiendo actualizaciones de seguridad y se esperaba que en 5 años todo el mundo ya estuviera adecuado a su versión hermana, aunque para sorpresa de todos incluido el mismo van Rossum esto no ocurrió, por lo que la vida de esta misma se extendió hasta 2020. Llegado ese momento la versión 2.7 dejará de tener soporte y pasará a ser totalmente obsoleta.

A causa de la continua actualización de Python 3.0, esta dejó de ser compatible con su versión hermana, obligando a los usuarios de 2.X a editar el código o realizarlo de cero si quieren que funcione en la versión 3.X.

Uno de los proyectos más grandes y ambiciosos realizados en Python es el servidor de Dropbox, un servicio de alojamiento de archivos multiplataforma operado por la compañía del mismo nombre. Hoy en día este servicio cuenta con más de 500 millones de usuarios registrados.

Actualmente van Rossum está trabajando en la plantilla de Dropbox y aunque fue él con su equipo el encargado de todas las actualizaciones de Python hasta la fecha, el 12 de julio de 2018 anunció su retiro de los procesos de decisión.

2.2 CARACTERÍSTICAS

1. **Multipropósito:** Un punto clave de Python es que su lenguaje puede emplearse en la creación de todo tipo de programas siendo limitado meramente por la imaginación del programador.
2. **Interpretado:** Muchos lenguajes de programación deben ser compilados antes de su ejecución, mientras que Python no necesita de ello. Si hablamos técnicamente, si se realiza una compilación, pero esta se realiza de forma automática y transparente para el programador.
3. **Multiplataforma:** Python se creó originalmente para utilizarse en Unix, pero nunca se limitó solo a ello, creando versiones para múltiples sistemas operativos.
4. **Orientado a objetos:** Esta forma de programación es la más cercana a como haríamos las cosas en el mundo real, volviendo a Python en una de las opciones preferidas de los usuarios.
5. **Interactivo:** Dentro de Python existe un intérprete por línea de comandos, este ejecuta cada sentencia y produce un resultado visible ayudando a comprender mejor el lenguaje y probar rápidamente el código.

6. **Sintaxis clara:** Python tiene una sintaxis muy visual gracias a una notación con márgenes obligatoria. Esto ayuda a que los programadores adopten un estilo parecido al programar y que se pueda analizar y entender rápidamente.

3. CODIGO DE PYTHON

- **String**

Un bloque de texto es lo que llamamos string o cadena. Python lo toma como un objeto y nos permite trabajar con ello.

Ejemplo:

```
Print ("Buenos días")
```

En este ejemplo, la cadena es la frase “Buenos días” y le pedimos al programa que lo imprima para nosotros. Python no se complica con cual comilla se ocupe, si una simple (‘) o una doble (“), solo pide que se ocupe el mismo tipo de comilla al comienzo y al final de cada cadena.

- **Errores**

Todo programador comete errores, lo importante es poder identificarlos y para ellos Python nos ayuda a través de mensajes. Los más comunes son:

SyntaxError: EOL while scanning a string literal.

Esto significa que la cadena no fue cerrada o bien que no se ocupó la misma comilla.

NameError.

Esto ocurre cuando no se ocupan comillas. Python trata las palabras sin comillas como comandos, como print o if.

- **Variables**

Al programar en general, debemos crear sistemas para lidiar con datos que cambian todo el tiempo. Es importante tener presente que pueden significar distintas cosas en distintos momentos. Por eso se usan las variables para definir cosas que están sujetas a cambios.

Ej:

```
fecha_hoy = 27-10-2018
```

```
temperatura_hoy = 27 ° C
```

- **Aritmética**

Python puede hacer directamente sumas, resta, división, multiplicación y combinaciones entre ellos. Toda operación debe estar asignada a una variable.

Python también ofrece el módulo operador (%) que devuelve el sobrante de la división.

- **Comentarios**

Para comenzar un comentario se ocupa el símbolo (#). El texto que sigue al signo no es tomado en cuenta por la máquina.

- **Booleanos**

Este tipo de dato se ocupa para definir si una variable es verdadera o falsa.

Ejemplo:

```
tengo_22 = True
```

```
im_Russian = False
```

- **Valor**

Python automáticamente asigna un tipo de dato a una variable

Ejemplo:

7 es int. 7. es float. "7" es cadena.

- **Acceso por índice**

A cada carácter de una cadena se le asigna un número. Este número se le conoce como índice.

Ejemplo:

```
"cats" [ 0 ] = letra C.
```

Siempre la primera letra de la cadena empieza representada por el número 0.

- **String Methods (Métodos de cadena)**

Los String Methods permiten al usuario realizar tareas específicas para las cadenas.

Nos enfocaremos en 4 métodos de cadena:

1. `len()`
2. `lower()`
3. `upper()`
4. `str()`

1. `len()` nos permite saber la longitud (el número de caracteres) de la cadena.
Profiene de la palabra `length` (longitud).

Ejemplo:

```
parrot = "Norwegian Blue"
```

```
print len(parrot)
```

Esto nos arrojará como resultado el número 14, número de los caracteres que hay en la variable `parrot`.

2. `lower()` es un método para deshacerse de todas las mayúsculas de la cadena.

Ejemplo:

```
parrot = "Norwegian Blue".lower()
```

```
print parrot
```

Esto nos dará como resultado la palabra `norwegian blue`.

3. `upper()` es un método similar al anterior que nos permite volver a toda la cadena a mayúsculas.

Ejemplo:

```
parrot = "norwegian blue".upper()
```

```
print parrot
```

Esto nos permite obtener como respuesta NORWEGIAN BLUE.

4. `str ( )` es un método que nos permite transformar palabras que no son cadena a cadena.

Ejemplo:

```
pi = 3.14
```

```
print str(pi)
```

Ya con esto el programa nos permite ver 3.14 como respuesta.

- **Notación de puntos**

Son los métodos `.lower( )` y `.upper( )`, estos solo funcionan con cadenas, mientras que los métodos `len( )` y `str( )` pueden trabajar con otros tipos de datos

- **Concatenación de cadenas**

En ocasiones tendrás que combinar una cadena con cosas que no son una cadena

Ejemplo:

```
print "The value of pi is around " + str(3.14)
```

- **Formación de cadenas con %**

Cuando quieres imprimir una variable con una cadena, existe un método mucho mejor que concatenación de cadenas.

% es utilizado después de la cadena para reemplazar el símbolo %s que está dentro de la cadena, como veremos en el ejemplo a continuación

Ejemplo 1:

```
string_1 = "Camelot"
```

```
string_2 = "place"
```

```
print "Let's not go to %s. 'Tis a silly %s." % (string_1, string_2)
```

También puede ocuparse para imprimir una variable que es un número entero, incluso puede anteponerse un 0 si se precisa usando %02 como se verá a continuación

Ejemplo 2:

```
day = 6
```

```
print "03 - %s - 2019" % (day)
```

```
# 03 - 6 - 2019
```

```
print "03 - %02d - 2019" % (day)
```

```
#03 - 06 - 2019
```

- **Control de flujo**

El control de flujo nos da la habilidad de elegir qué ocurrirá dependiendo de la opción que elijamos. Para empezar con ello primero debemos conocer las funciones de comparación:

Igual a (==)

Distinto a (!=)

Menor a (<)

Menor o igual a (<=)

Mayor a (>)

Mayor o igual a (>=)

También se usa:

Boolean Operators								
AND			OR			NOT		
A	B	A AND B	A	B	A OR B	A	NOT A	
True	True	True	True	True	True	True	False	
True	False	False	True	False	True	False	True	
False	True	False	False	True	True			
False	False	False	False	False	False			

Figura 2-1. Tabla de operaciones booleanas.

Los operadores booleanos no se leen de izquierda a derecha, estos tienen un orden de prioridades, a menos claro que usemos paréntesis obtener la estructura deseada.

Primero se le da prioridad al operador “not” , siguiendo el operador “and” y por último el operador “or”.

- **Sintaxis Condicional**

“if” es una declaración condicional que ejecuta un código específico después de verificar si su expresión es verdadera.

Ejemplo:

```
if 8 < 9 :
```

```
    print “Ocho es menor que nueve.
```

Es importante denotar el espacio antes del print. 4 espacios, para darle a entender a Python que estamos entrando a un nuevo bloque de código.

“else” es una declaración que complementa a “if” permitiendo al programa correr una línea de código en caso de que no ocurra el “if”.

Al utilizar más de un “if”, todos los “if” intermedios son nombrados “elif”.

- **Entrada**

Utilizar una entrada, también conocida como input, nos permite introducir una cadena al código desde la consola.

Ejemplo_

```
name = raw_input("¿Cuál es tu nombre?")
```

```
print name
```

- **Cadena.isalpha()**

El método .isalpha() nos permite consultar si la cadena introducida contiene caracteres que no sean letras. En caso de contener números, nos devolverá un "False".

Como sabemos, hay elementos con mejores y peores capacidades para poder explotar el IoT. En este caso, el micro controlador Raspberry Pi es muy eficiente en esta área, hecho que se comprobó a través del estudio de sus características que la han vuelto tan famosa.

Es innegable que el hecho de abordar un lenguaje de programación nuevo es intimidante tanto para entendidos en el tema y aún más para principiantes, por lo que en la segunda parte de este capítulo abordamos Python, un lenguaje muy enriquecedor para el usuario y sobre todo muy cómodo y amigable especialmente para principiantes, por su sintaxis tan clara, una de las razones por la cual hoy en día Python es uno de los lenguajes más famosos y mayormente enseñados a nivel mundial.

Capítulo 3: Software de Estación climática

1. **SEÑALES DE ENTRADA Y SALIDA**

La Raspberry Pi, micro controlador constantemente comparado con Arduino, nos permite diversas formas de obtener señales de entrada, a diferencia de su competencia en el mercado. Hasta el presente existen diversas formas de obtener señales de entrada y generar señales de salidas las cuales veremos a continuación:

1.1 **GPIO**

General Purpose Input/ Output son una serie de conexiones que como su nombre indica, pueden funcionar como entrada o salida. Estos están incluidos en todos los modelos de Raspberry y representan la interfaz entre el micro controlador y el mundo exterior.



Figura 3-1. Esquema de GPIO.

Como se puede apreciar en la imagen, no todos los pines tienen la misma función:

- **Pines de alimentación:** se pueden distinguir pines de 5v, 3v3 (limitados a 50 mA) y tierra (GND o Ground), que aportan alimentación a estos voltajes para tus circuitos. Te pueden servir como fuente de alimentación aunque también se pueden ocupar otras fuentes como pilas por ejemplo. Cabe recalcar que los pines son unbuffered, esto significa que no tienen buffers de protección por lo que hay que tener cuidado con las magnitudes (voltaje e intensidad) para no dañar la placa.
- **GPIO normales:** son conexiones configurables que se pueden programar para los proyectos.
- **GPIO especiales:** se encuentran algunos pines destinados a una interfaz UART, con conexiones TXD y RXD que sirven para comunicaciones en serie, por ejemplo, conectar una placa Arduino.

Ejemplo de programación de GPIO:

A través de este ejemplo controlaremos una luz LED a través del lenguaje Python en el sistema operativo Raspbian.

Primero que todo, se deben descargar los drives controladores de GPIO

```
1 sudo apt-get install python-dev python-rpi.gpio
```

Ya con esto, podemos proceder a crear un código fuente que nos permita controlar el encendido y apagado de la luz LED.

```
3 #!/usr/bin/env python
4 #LED.py
5
6 import time
7 import pigpio
8 pi = pigpio.pi()
9
10 pi.set_mode (17, pigpio.OUTPUT) #Ponemos el pin 17 como salida
11
12 pi.set_servo_pulsewidth (17, 1300) #Iniciamos pulsos cada 1.3 segundos para encender el LED
13
14 pi.stop() #Terminamos el programa.
```

1.2 PUERTO SERIE

La Raspberry nos permite obtener al controlar otros dispositivos o micro controladores a través del puerto serie. Un ejemplo de ello es la gran cantidad de proyectos existentes donde se emplea una placa Arduino junto a la Raspberry, siendo la primera la utilizada para obtener información, ya sea a través de sensores o componentes conectadas a la misma y finalmente enviar esos datos a la Raspberry. Es posible hacer la operación inversa y enviar información desde la Raspberry al Arduino, volviendo así la Raspberry la generadora de salida y al Arduino la adquisidora de señales de entrada.

Ejemplo de conexión Raspberry – Arduino:

En la placa Arduino se escribe el siguiente programa

```
void setup(){  
  Serial.begin(9600);  
}  
void loop(){  
  Serial.println("Hello Pi");  
  delay(2000);  
}
```

Este programa le dice al Arduino que a través de la conexión serial, envíe el mensaje “Hello Pi”.

En la Raspberry se escribe el siguiente programa

```
#este comando nos permite recibir información a través del puerto serial
import serial

#/dev/ttyACM0 es el nombre del puerto.
#9600 son los baudios en el puerto serie y deben ser iguales en ambos aparatos.
ser = serial.Serial('/dev/ttyACM0', 9600)

#nos permite leer lo procedente del Arduino
while 1 : ser.readline()

#en la pantalla aparecerá este mensaje
"Hello Pi"
```

Ejemplo de conexión Arduino – Raspberry

En este programa, Arduino encenderá y apagará una luz LED del pin 12 tantas veces como ordene la Raspberry.

```

[sourcecode language="cpp"]

const int ledPin = 12;

void setup(){
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  if (Serial.available()) {
    light(Serial.read() - '0');
  }
  delay(500);
}

void light(int n){
  for (int i = 0; i < n; i++) {
    digitalWrite(ledPin, HIGH);
    delay(100);
    digitalWrite(ledPin, LOW);
    delay(100);
  }
}

[/sourcecode]

```

En la Raspberry se escribe el siguiente programa, enviando la señal de que prenda y apague 3 veces.

```
ser.write('3')
```

1.3 **PLATAFORMAS IOT**

Estas son páginas web creadas especialmente para permitir una interconexión entre los dispositivos conectados. El mercado de las plataformas IoT está en su auge y continua expansión por lo que se pueden encontrar plataformas dedicadas tanto a la recolección, análisis y grafica de datos, como otras orientadas a mantener un control del microcontrolador de forma totalmente remota. Algunas páginas IoT recomendadas son:

- <https://thingspeak.com/> plataforma asociada a MATLAB, permite recaudación y análisis de los datos obtenidos a través del micro controlador.
- <https://arest.io/> plataforma que nos permite controlar de forma sencilla y remota nuestra Raspberry Pi.
- <http://thethingbox.io/> es un software listo para usar que se instala en la tarjeta SD, permitiendo control de IoT.
- <https://www.dataplicity.com/> permite el control, manejo y edición del micro controlador de forma remota con solo el correo electrónico.

1.4 PYTHON PIP

Python pip es un sistema de gestión de datos usado para administrar paquetes de datos escritos en Python. La mayor ventaja es la facilidad que le da al usuario de instalar o desinstalar paquetes de datos de software de Python de forma rápida y sencilla con una orden a través de solo una línea de comando. Es muy utilizado para enviar la información almacenada a la Raspberry a algún fichero que el usuario especifique en Google Drive en tiempo real.

Ejemplo de implementación

```
#se instala Python pip
sudo apt-get install python-pip

# se instala gspread
#permite conexión con google drive
sudo pip install gspread
```

Ya después de instalar estos paquetes, solo queda la configuración de conexión de Google Drive, en el cual se debe especificar el correo y fichero donde se almacenaran los datos.

2. WEEWX

De entre todas las plataformas y aplicaciones que existen, se decidió por el software WeeWX, un software de código libre que nos permite interactuar con estaciones del clima, esta nos permite producir reportes, gráficos y hasta páginas HTML. Las razones principales de porque se decidió por este software es el hecho de que está escrito en Python, sumado al hecho de que es de código libre, nos permite ver y analizar su código fuente. Otra razón muy importante que nos atrajo es que puede utilizarse en diversos micro controladores y, por supuesto, uno de ellos es la RaspberryPi. A través de este capítulo analizaremos y explicaremos las partes principales del código, que son el fichero de configuración general de WeeWX, el que nos permitirá configurar y entender cómo funciona este software.

2.1 CODIGO

```
1
2  debug = 0
3
4  WEEWX_ROOT = /
5
6  socket_timeout = 20
7
```

Debug es la parte del programa que prueba y depura (elimina) los errores que se puedan ir encontrando en el log. Log es el registro secuencial en un archivo donde ocurren todos los acontecimientos del software. Existen 2 opciones. 0 que es el nivel estándar de depuración y 1, que permite al programa mantener una constante y más intensiva depuración, pero los ficheros de log adquirirán un tamaño y peso enorme dentro del sistema.

WEEWX_ROOT es la dirección donde se guardan nuestros archivos de datos, también llamado directorio (root).

socket_timeout es el tiempo de espera para desconectar una conexión FTP o HTTP. Estos últimos 2 datos no se modifican a menos que haya un completo conocimiento del software.

```

 8  [Station]
 9
10      location = Concepcion, Chile
11
12      latitude = -36.779071
13
14      longitude = -73.108583
15
16      altitude = 2, meter
17
18      station_type = WeatherLink
19
20      station_url =
21
22      rain_year_start = 5
23
24      week_start = 0

```

location es donde describimos la locación de la estación. En este caso, se introducirá, Concepción, Chile.

latitude y **longitude** es la ubicación en grados decimales.

altitude nos pide la altura a la que está la estación, la cual debemos especificar junto con un sistema de medida (feet o meter).

station_type el programa nos pide indicar la clase de estación que estamos usando. Debe ser una de las admitidas por WeeWX.

station_url se utiliza en el caso de tener una página web. Para ello se le introduce el URL de la página.

rain_year_start el programa pregunta en que mes del año empieza a llover siendo 1 enero, 2 febrero y así sucesivamente. Cabe denotar que hay estaciones que hacen esto automáticamente.

week_start es para decidir con que día el programa empieza la semana. 0 equivale a lunes y 6 a domingo.

```
26 [Simulator]
27
28     loop_interval = 2.5
29
30     mode = generator
31
32     start = 2011-01-01 00:00
33
34     driver = weewx.drivers.simulator
```

También existe la opción de implementar el modo simulador, que nos permite testear el programa cuando no hay alguna consola de control de clima accesible.

loop_interval es el tiempo en segundos que hay entre los intervalos de loop.

mode existe este concepto para poder elegir entre 2 opciones. **simulator** que nos permite una simulación en tiempo real ofreciendo al programa el “dormir” entre los los intervalos de loop y **generator** que emite paquetes de loop lo más rápido posible. Esta última opción se recomienda a la hora de hacer pruebas.

start es para indicar la fecha de partida al programa. Si no se elige una fecha en concreto, por defecto se elegirá la fecha y hora actual en ese momento.

driver es para agregar algún driver que sea necesario descargar para el correcto funcionamiento del programa.

```

39     [[AWEKAS]]
40
41     #username =
42
43     #password =
44
45     [[CWOP]]
46
47     #station =
48
49     #passcode =
50
51     [[PWSweather]]
52
53     #station =
54
55     #password =
56
57     [[WOW]]
58
59     #station =
60
61     #password =
62
63     [[Wunderground]]
64
65     #station = |
66
67     #password =
68

```

register_this_station es para indicarle a WeeWX si queremos aparecer en el mapa de estaciones de la página oficial. En esta sección es solo posible contestar con TRUE o FALSE

A continuación nos aparecen todos los servicios webs disponibles donde enviar los datos. La estructura es la misma en todas por lo que no profundizaremos en ellas, solo aclararemos que si no se desea enviar datos, se debe anteponer un # en cada campo.

Ejemplo:

#station =

#password =

```

69 [StdReport]
70
71     SKIN_ROOT = /etc/weewx/skins
72
73     HTML_ROOT = /var/www/weewx
74
75     skin = Standard
76
77 #     #[[[Units]]]
78 #     [[[Groups]]]
79 #         group_altitude = meter
80
81 #         group_speed2 = meter_per_second2
82
83 #         group_pressure = mbar
84
85 #         group_rain = mm
86
87 #         group_rainrate = mm_per_hour
88
89 #         group_temperature = degree_C
90
91 #         group_degree_day = degree_C_day
92 #
93 #         group_speed = meter_per_second
94

```

SKIN_ROOT es para indicar donde se encuentran todos los “skin”. Skin es lo que nos permite cambiar la apariencia del software para darle una personalización.

HTML_ROOT nos permite establecer donde se guardaran los informes y páginas web que generemos.

data_binding es lo que permite la unión entre los dos términos anteriores y el software, por lo que este nunca se altera.

Skin es para decidir cuál usar. Para ello, podemos descargar una ya programada de la cual existe una gran cantidad e incluso programar nuestra propia skin.

[[Units]] son las unidades de medida generales que el sistema necesita saber. Estas no deben ser alteradas por el usuario.

2.2 INSTALACIÓN

Para instalar de forma satisfactoria y práctica el software en nuestro micro controlador, seguiremos estos pasos.

Primero que todo, debemos descargar programas que nos ayudarán en la instalación.

Win32diskimager nos permite grabar una imagen dentro del dispositivo que queramos, en este caso, en el micro SD.

<https://sourceforge.net/projects/win32diskimager/>

Bitvise Tunnelier es un programa que nos permite y facilita la comunicación entre la Raspberry Pi y nuestro computador y viceversa.

<https://www.bitvise.com/ssh-client-download>

Putty, un software para tener dominio como cliente SSH (Secure Shell). Este nos permite comunicarnos con la Raspberry sin necesidad de transmitir ficheros. Este programa no necesita instalación, tan solo copiar el fichero en el escritorio.

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Raspbian es el sistema operativo que se utilizará en el micro controlador. Este puede ser descargado de la página oficial de Raspberry.

<https://www.raspberrypi.org/downloads/raspbian/>

Ya con estos cuatro programas descargados podemos proceder a grabar el fichero imagen del sistema operativo Raspbian en la tarjeta SD. Para hacer esto, se utiliza el programa win32diskimager. Una vez finalizado este proceso, expulsamos la tarjeta SD para comprobar que la grabación fue satisfactoria.

Cuando la grabación está hecha, la Raspberry Pi debe conectarse a una fuente de alimentación y se le debe conectar un cable Ethernet para conectarla directamente con el router.

Si la grabación fue exitosa, los led del micro controlador comienzan a parpadear después de unos segundos. Para continuar, debemos averiguar la dirección IP que el router ha asignado a nuestra Raspberry.

Con esa información que nos servirá más adelante, nos conectamos a la Raspberry con Putty y a través de él nos preguntarán por el nombre de usuario y contraseña.

Estos son los que vienen por defecto, “pi” y “raspberry” respectivamente.

De aquí en adelante tendremos que introducir comandos a la Raspberry por lo que lo que nosotros escribamos lo representaremos con letra en negrita mientras que la respuesta del sistema la representaremos en letra cursiva.

Primero le pediremos al sistema actualizar el sistema operativo Raspbian

pi@pi1 ~ \$ sudo apt-get update

sudo: unable to resolve host pi1

Get:1 http://archive.raspberrypi.org wheezy Release.gpg [490 B]

Get:2 http://archive.raspberrypi.org wheezy Release [7,227 B]

.../...

Fetched 13.7 MB in 1min 45s (130 kB/s)

Reading package lists... Done

```
pi@pi1 ~ $
```

Volvemos a repetir el commando, ya que es normal que los paquetes descargados necesiten una segunda actualización.

```
pi@pi1 ~ $ sudo apt-get update
```

```
sudo: unable to resolve host pi1
```

```
Reading package lists... Done
```

```
.../...
```

```
Need to get 78.5 MB of archives.
```

```
After this operation, 572 kB of additional disk space will be used.
```

```
Do you want to continue [Y/n]?
```

```
Y
```

Estos procesos antes mencionados toman su tiempo, pero una vez terminados podemos pasar a actualizar el firmware.

```
pi@pi1 ~ $ sudo rpi-update
```

Una vez terminada esta actualización debemos reiniciar la Raspberry con el comando

```
Sudo /etc/init.d/weewx restart
```

Nos conectamos nuevamente al micro controlador con Putty y entramos al directorio home/pi para descargar WeeWX.

```
cd /home/pi
```

```
wget http://www.weewx.com/downloads/weewx_3.8.2-1_all.deb
```

El commando `wget` nos permite descargar el software inmediatamente, siempre recordando estar en el directorio `home/pi`, de lo contrario la descarga no tendrá efecto alguno.

Por si acaso, escribimos un comando para cerciorarnos de que WeeWX no requiere algún otro paquete de nuestra Raspberry que aún no tenga instalada.

Si ese fuera el caso, el sistema nos respondería algo así:

```
sudo dpkg -i wee*.deb
```

```
sudo: unable to resolve host pi1
```

```
Selecting previously unselected package weewx.
```

```
(Reading database ... 64388 files and directories currently installed.)
```

```
.../...
```

```
dpkg: error processing weewx (install):
```

```
dependency problems leaving unconfigured
```

```
Errors were encountered while processing:
```

```
weewx
```

```
pi@pi1 ~ $
```

Para corregirlo debemos pedirle al sistema que descargue e instale este paquete

```
pi@pi - $ sudo apt-get update
```

```
sudo: unable to resolve host pi1
```

```
Hit http://archive.raspbian.org wheezy Release.gpg
```

```
Hit http://archive.raspbian.org wheezy Release
```

```
.../...
```

Reading package lists...

Done pi@pi1 ~ \$.

pi@pi - \$ sudo apt-get -f install

sudo: unable to resolve host pi1

Reading package lists... Done

.../...

Need to get 4,900 kB of archives.

After this operation, 17.2 MB of additional disk space will be used.

Do you want to continue [Y/n]?

Y

Con esta última actualización nuestro Raspberry esta lista y funcionando, nos empezará a pedir que la configuremos introduciendo los datos que se vieron al comienzo del capítulo, dando paso a su uso.

3. **WEATHER-UTIL**

Weather utility es una herramienta utilizada para obtener condiciones climáticas a través de la línea de comandos de nuestro micro controlador. Esta utilidad nos permite conectarnos a cualquier aeródromo en el mundo a través de su código METAR (Meteorologic Aerodrome Reports), NOAA (the USA National Oceanic and Atmospheric Administration) o NWS (the USA National Weather Service) aunque

nosotros utilizaremos el código OACI (Organización de Aviación Civil Internacional) y desde ahí tomar la información para exhibirla en nuestra línea de comandos.

También es posible obtener la previsión climática para varios días, pero por ahora esa opción solo está reservada para aeropuertos estadounidenses.

Weather utility puede ser descargado desde la página oficial de Raspbian a través del siguiente código:

sudo apt-get install weather-util

Para nuestro programa obtendremos los datos del Aeropuerto Carriel Sur ubicado en Avenida Jorge Rodríguez 5001, Talcahuano.

La sigla con la que se nombra al aeropuerto es **SCIE**.

```
pi@raspberrypi:~ $ weather scie
Current conditions at Concepcion, Chile (SCIE) 36-46S 073-03W
Last updated Feb 24, 2019 - 05:40 PM EST / 2019.02.24 2240 UTC
  Temperature: 62 F (17 C)
  Relative Humidity: 51%
  Wind: from the SW (220 degrees) at 18 MPH (16 KT) gusting to 31 MPH (27 KT)
```

Como podemos denotar, la utilidad nos arroja una información bastante completa, que incluye temperatura, humedad relativa, la velocidad y dirección del viento, la sensación térmica y las condiciones del cielo.

Aunque ya hemos logrado que funcione nuestra utilidad, ahora se requiere pasar al siguiente nivel y lograr automatizar este recurso, para ello tendremos que entender y profundizar ciertos conceptos, que son el de script, mailutils, ssmtp y cron, los cuales conoceremos a continuación.

3.1. **SCRIPT**

Script etimológicamente es una reducción de la palabra inglesa “manuscript” que significa “manuscrito hecho a mano” y un manuscrito es una serie de instrucciones que deben seguir los individuos que forman parte de una pieza de teatro, cine o programa televisivo.

De forma análoga, en informática un script es un documento que contiene un conjunto de órdenes almacenados en un archivo de texto, generalmente ligeros y que es ejecutado por lotes o línea a línea, en tiempo real por un intérprete.

Estos, dicho de otra forma son pequeños programas que no son compilados, es decir, necesitan un programa interprete que debe tomar el script y llevarlo al lenguaje de la máquina para que la información sea procesada y ejecutada por el ordenador.

Esto ocurre por la línea de comandos, la cual es la interfaz de usuario que permite, mediante instrucciones, comunicarse con el núcleo y lograr ejecutar dichas órdenes.

El fichero o script pasa a ser *Shell-script* cuando se le da el permiso de ejecución, logrando como resultado que este sea ejecutable.

El término *Shell* significa en simples términos el intérprete que nos permitirá comunicarnos con el núcleo, existen muchísimos intérpretes(*shell*) con distintas escrituras y comandos (bash, sh, ash, dash, ksh, etc) y es misión del usuario informar al sistema con cual se trabajará.

Para entender como se logró realizar el script, se expondrá la línea de códigos con la que esta fue creada y se explicará cada paso:

3.1.1. Código

```
pi@raspberrypi: ~$ mkdir bin
pi@raspberrypi: ~$ ln
pi@raspberrypi: ~$ cd bin
pi@raspberrypi: ~/bin nano 1
```

mkdir (nombre proveniente de “make directory” o crear directorio) nos permite crear nuevos directorios en nuestro sistema de archivos. En nuestra aplicación, el nuevo directorio creado recibió el nombre de *bin*.

ln nos permite crear un enlace entre un archivo especial y otro existente. En este caso nos enlaza con el directorio *bin* y crear aquí el script.

cd (“change directory” o cambiar directorio) es necesario para acceder a una ruta distinta a la que se está. Se le indica al programa que ingrese al directorio *bin*.

nano es el comando que nos permite leer, crear y editar un archivo. En este caso se crea un archivo o script con el nombre “1”.

```
#!/bin/bash

weather scie
```

Ya dentro del script, lo primero es informarle a la máquina con cual Shell se trabajara; en este caso, se decidió por el intérprete Bourne-Again Shell, más conocido como bash. Esto siempre se informa en la primera línea mientras que desde la segunda se puede describir la acción del script, en este caso, llamar a la utilidad weather.

Una vez terminado, se debe salir del programa de edición presionando **ctrl+x** lo cual nos preguntará si deseamos cambiar los cambios realizados y esto se confirma presionando la tecla **s**.

```
pi@raspberrypi: ~/bin ls -l
-rw-r--r-- 1
pi@raspberrypi: ~/bin chmod u+x 1
pi@raspberrypi: ~/bin ls -l
-rwxr--r-- 1
```

Ya con lo anterior realizado, volvemos a la ventana principal de comandos.

Se introduce el comando **ls -l** que nos permite visualizar los comandos hechos hasta ahora, arrojándonos en la siguiente línea la información del script nombrado “1”.

Si analizamos la información que nos da, se puede apreciar las primeras letras, **rw**, las cuales significan que ese script tiene permiso para ser leído(**r**) y escrito (**w**) por el sistema, lo cual nos conlleva a la siguiente línea, que es otorgarle el permiso de ejecutar.

Los permisos **chmod** (change mode) nos permiten cambiar los permisos de acceso para las carpetas y ficheros. A través de este le estamos informando al sistema que **u** (tú, haciendo referencia al script que se desea dar el permiso) + (añadir) **x** (execute/ejecutar) al script denominado **1**.

Con este permiso, el sistema puede ejecutar el script con solo su nombre y presionando enter, es por ello que se decidió por un nombre breve, dando la posibilidad de obtener los datos en solo 2 pasos.

3.2. MAILUTILS Y SSMTP

Mailutils es un paquete de herramientas creado para gestionar el correo electrónico a través de la consola, muy recomendado ya que tiene una amplia gama de usos dependiendo de con que MTA lo complementemos. MTA (Mail Transfer Agent, Agente de Transferencia de Correo) también conocido servidor de correo realiza una serie de procesos con la finalidad de transportar información entre los distintos usuarios. Otra gran ventaja de esta utilidad es el hecho de que trabaja con SMTP (Simple Mail Transfer Protocol, Protocolo Simple de Transferencia de Correo) el cual es un estándar oficial de internet dándole al usuario una mayor fiabilidad de su uso.

Para este proyecto, se utilizará el protocolo sSMTP, conocida también como la versión Slim, siendo esta más liviana y menos invasiva para la CPU, permitiéndonos seguir utilizando el microcontrolador mientras envía correos. Es importante puntualizar que esta versión solo permite enviar correos, no notifica al usuario si el correo enlazado tiene nueva correspondencia ni permite leerla.

Para instalar estos paquetes en nuestro ordenador, solo debemos ir a la línea de comandos e introducir las siguientes órdenes:

sudo apt-get install mailutils

sudo apt-get install ssmtp

Una vez instalados estos elementos, debemos de configurar sSMTP para poder informarle al sistema el correo que se enlazará con la utilidad, esta configuración es muy importante y delicada, ya que cualquier error nos negará la posibilidad de enviar el mail y nos informará de ello en la línea de comandos.

Para abrir la configuración se debe ingresar al fichero de configuración con el siguiente comando:

sudo nano /etc/ssmtp/ssmtp.conf

Este nos despliega el siguiente texto

```
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=hans.flavio.pinochet@gmail.com

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=gmail.com

# The full hostname
hostname=raspberrypi

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
FromLineOverride=YES
AuthUser=hans.flavio.pinochet@gmail.com
AuthPass=password del correo
UseSTARTTLS=YES
```

En la configuración existen 7 puntos clave los cuales deben ser llenados correctamente:

- Root, también conocido como postmaster, es el administrador de un servicio de correo y el cual es necesario ya que es donde se dirigen los errores en el procesamiento del correo electrónico.

- Mailhub es la parte del sistema que ayuda a acelerar la interacción con el mail. Aquí solo se debe editar el número después de los dos puntos, el cual da permiso al sistema para usar el correo.
- Hostname es el nombre del equipo el cual se está utilizando para enviar dichos mails, en este caso, se conserva el nombre predeterminado del sistema, raspberrypi.
- FromLineOverride le pregunta al usuario si quiere elegir la dirección de correo proveniente puesta en el sistema u otra, a lo cual siempre se recomienda no alterar este punto y dejarlo en YES.
- AuthUser es el correo donde nacen los emails a enviar.
- AuthPass es la contraseña del mismo correo, es importante señalar que no se encripta la clave por lo que la vuelve visible para cualquier usuario con acceso a la configuración.
- UseSTARTTLS es una extensión de los protocolos de comunicación de texto plano, este punto debe siempre ir incluido con YES.

Es importante denotar que los últimos 3 puntos no siempre aparecen en la configuración por lo que hay que incluirlos manualmente y que en ocasiones el correo de gmail no da acceso al microcontrolador por considerarlo un dispositivo no seguro, por lo que se aconseja ir a la configuración del gmail y habilitar el acceso de dispositivos no seguros.

Una vez realizado estos pasos llega el momento de probar si efectivamente el sistema tiene acceso a enviar correos, por lo que en la línea de comandos se ingresa el siguiente texto.

echo “probando” | mail -s “Titulo del correo” correo.destinatario@dominio.com

Ya comprobada la correcta conexión y solucionando cualquier error de configuración que se puede haber dado durante la configuración de este, se debe crear un código que nos permita enviar la información de la utilidad weather por correo.

Para ellos, debemos buscar la localización del script dentro de la Raspberry e incluirla dentro del correo. El producto final tiene la siguiente sintaxis

/home/pi/bin/1 | mail -s “Clima Actual” correo.destinatario@dominio.com

Con esta línea, a los pocos segundos el usuario receptor recibirá un correo con esta información:

Clima Actual



root <hans.flavio.pinochet@gmail.com>

14:30

Para: hanz-kingx@hotmail.cl; helmut.contreras@usm.cl

Current conditions at Concepcion, Chile (SCIE) 36-46S 073-03W

Last updated Mar 05, 2019 - 12:00 PM EST / 2019.03.05 1700 UTC

Temperature: 66 F (19 C)

Relative Humidity: 45%

Wind: from the SW (230 degrees) at 16 MPH (14 KT)

Logrando esto solo nos falta automatizar este proceso para que se haga de forma autónoma e independiente. Para este cometido se utilizó la aplicación Cron.

3.3. CRON

Cron es el programa que permite a los usuarios ejecutar automáticamente comando o scripts a una hora o fecha específica. Aunque es usado para tareas como respaldos, este puede ser usado para una amplia gama de servicios. Su nombre viene del griego chronos (tiempo.) En el sistema operativo, cron es un administrador regular de procesos en segundo plano (a esto se le denomina demonio). Los procesos que deben ejecutarse deben ser especificados en el fichero crontab.

Crontab es el archivo de texto donde se almacenan la lista de comandos que se deben ejecutar. Este es quien verifica la fecha y la hora en que el usuario ordena ejecutar el script o comando.

Para ver esta lista utilizamos el comando:

sudo crontab -l

Y para editarlo:

sudo crontab -e

Antes de crear alguna tarea, debemos entender la sintaxis con la que funciona cron.

Esta está formada por 5 asteriscos seguidos por el comando que se desea ejecutar.

*** * * * * /raíz/directorio/script.a.ejecutar.sh**

De izquierda a derecha los asteriscos representan:

- Minutos, de 0 a 59.
- Horas, de 0 a 23
- Día del mes, de 1 a 31
- Mes, de 1 a 12
- Día de la semana, de 0 a 6, siendo 0 el domingo.

Si se deja el asterisco sin reemplazarlo por un número, el sistema lo toma como que se quiere utilizar en todo momento, en otras palabras, cada minuto, cada hora, cada día del mes, correspondiendo el caso de dónde se mantenga el asterisco.

También existen palabras reservadas que nos permiten facilitar la ejecución del script.

- @reboot: se ejecuta una única vez al inicio.
- @hourly: cada hora.
- @daily/@midnight: una vez al día.
- @weewkly: una vez a la semana.
- @monthly: una vez al mes.
- @yearly/@annually: ejecutar cada año.

Ya con esta pequeña explicación pasamos al último paso, que consiste en escribir el comando que se editará en crontab. Después de un continuo proceso de ensayo y error, se logra crear el código encargado de enviar la información climática a través del correo, esto se realizará cada media hora.

```
*/30 * * * * /home/pi/bin/1 | mail -s "Clima Actual" correo.destinatario@dominio.com
```

Conclusión

Sintetizando lo visto en el capítulo 3, este nos permitió conocer diferentes formas de como el micro controlador puede obtener la información con la que trabaja, esto llevándonos a la fase final, que es la demostración de las capacidades del IoT.

El internet de las cosas es un tema que ha llegado para quedarse y evolucionar junto con la humanidad, es por ello que debemos aprovecharlo y sacarle el máximo provecho ya sea en nuestra vida rutinaria como en nuestro mundo laboral.

Con la meta de realizar un extenso estudio de este tópico, logramos conocer en profundidad desde el origen hasta las características que distinguen y componen el IoT.

Gracias a ellos, nos podemos hacer una idea concreta de sus distintas aplicaciones para lograr ver de qué forma nos beneficia el aplicarlo y de que formas más sutiles ya se está aplicando en nuestro entorno.

A través del estudio del micro controlador Raspberry Pi y el lenguaje de programación Python, se concretó la correcta manipulación sobre esta computadora de tamaño reducido, permitiendo una satisfactoria instalación del sistema operativo y del software WeeWX, esto nos permitió editar y configurar la aplicación para futuros usos.

A la hora de implementar todo lo investigado y profundizado, se obtuvo como producto final la aplicación del IoT a la hora de obtener los datos del clima, estos obtenidos de una fuente confiable y enviados a la comodidad del correo, a la hora y día que más cómodo le resulte al usuario.

Esto no solo se limita a un usuario, ya que se puede enviar la misma información a un sinnúmero de destinatarios, o crear un horario específico para cada uno si así se desea.

Resumiendo, podemos utilizar este ejemplo como pie inicial para darnos cuenta del gran potencial que tiene el Internet de las Cosas, un término que nos abre una infinidad de posibilidades y oportunidades, los cuales están solo limitadas por una persona, uno mismo.

Bibliografía

Cendón Bruno. El origen del IoT. [en línea] [16 de enero, 2017]
<<http://www.bcendon.com/el-origen-del-iot/>>

Contreras Lidia. Raspberry Pi. [en línea] [18 de Diciembre, 2013]
<<https://histinf.blogs.upv.es/2013/12/18/raspberry-pi/>>

Digital Guide. ARPANET: los primeros pasos del internet [en línea] [19 de marzo, 2018]
< www.ionos.es/digitalguide/paginas-web/desarrollo-web/arpanet-los-inicios-de-internet/>

F.M., Yúbal. Raspberry Pi 3 Model B+. [en línea]. [14 de Marzo, 2018].
<<https://www.xataka.com/ordenadores/llega-la-nueva-raspberry-pi-3-model-b-mismo-precio-pero-mas-velocidad-y-wifi-de-doble-banda>>

Master Magazine. Definición de ARPANET. [en línea] [s.a.]
<<https://sistemas.com/arpanet.php> >

Nattivos. El Internet de las cosas (IOT). [en línea][s.a.]
<<https://nattivos.com/el-internet-de-las-cosas-iota/>>

Raspberry Pi Foundation. FAQs [en línea] [s.a.]
 <<https://www.raspberrypi.org/documentation/faqs/>>

Rojas Parra, Juan Pablo. La historia de Python. [en línea]. [Marzo, 2018].
 <<https://platzi.com/blog/historia-python/>>

T.I., Manuel. Raspberry Pi 3+ una pequeña revolución. [en línea]. [14 de Marzo, 2018].
 <<https://rasberryparatorpes.net/hardware/raspberry-pi-3x-una-revolucion-a-medias/>>

T.I., Manuel. Sobre la Raspberry Pi 3 B+. [en línea]. [17 de Abril, 2018].
 <<https://rasberryparatorpes.net/hardware/sobre-la-raspberry-pi-3-bplus/>>

del Valle Hernández, Luis. Aplicaciones del IoT usos prácticos en el mundo real. [en línea] [2016] <<https://programarfacil.com/podcast/aplicaciones-del-iot-reales/>>

Ibex Tech. sSMTP to send emails. [en línea] [2014]
 <https://rasberry-projects.com/pi/software_utilities/email/ssmtp-to-send-emails>

Mail, envía correos electrónicos desde el terminal de tu equipo Gnu/Linux. [en línea]
 [27 de junio de 2018] <<https://ubunlog.com/mail-envia-correo-terminal>>

Concepción Certad, Claudio. Envía correos desde el Terminal con Mailutils y sSMTP en Ubuntu. [en línea] [14 de febrero de 2011]

<<http://fraterneo.blogspot.com/2011/02/envia-correos-desde-el-terminal-con.html>>

Programar tareas en Linux usando crontab. [en línea] [2015]

<<https://geekytheory.com/programar-tareas-en-linux-usando-crontab>>