UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TÍTULO DE TESIS

Plataforma para telemetría y telecontrol utilizando dispositivos con sistema operativo Android: Aplicación inmediata a la domótica

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:
Aldana Quintana Pedro Alejandro

ASESOR DE TESIS:
Ing. Segura Altamirano Segundo Francisco

LAMBAYEQUE - PERÚ
2017

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TÍTULO DE TESIS

Plataforma para telemetría y telecontrol utilizando dispositivos con sistema operativo Android: Aplicación inmediata a la domótica

PRESIDENTE DEL JURADO
Ing. Ramírez Castro Manuel

SECRETARIO DEL JURADO
Ing. Oblitas Vera Carlos Leonardo

VOCAL DEL JURADO
Ing. Nombera Lossio Martin Augusto

LAMBAYEQUE - PERÚ
2017

Dedicatoria

A quien fue, es y seguirá siendo mi mejor amigo, mi guía, mi mejor maestro y la inspiración para esta tesis y para mi vida, mi amado padre *Carlos Julio Aldana Ramírez*.

A mi amada esposa *Milagros Barturén de la Cruz*, por estar a mi lado con mis atinos y desatinos, por su amor y sonrisa, por haberme dado la dicha de tener dos hijas. Son ellas los tres luceros que iluminan mi vida.

A mi querida madre *Celia Isabel* por sus años de dedicación y amor y a mis estimados hermanos.

Agradecimientos

A Dios y a la vida, por existir.

A la escuela profesional de Ing. Electrónica de la UNPRG y a sus docentes por sus años dedicados a la formación de profesionales y por la educación que me dieron, de lo cual me siento profundamente agradecido.

A mis hijas por las innumerables interrupciones mientras realizaba esta tesis, ya que con cada interrupción veía sus sonrisas y su alegría lo que me alentaba a seguir y a mi esposa por limitar estas interrupciones.

Prefacio

La idea de operar pequeños equipos electrónicos de consumo, como celulares, PDAs, tablets, etc. para telemetría y telecontrol no es un concepto nuevo, pero es en el año 2010, donde la idea es sugerida por el Ing. Eduardo Cerna en el curso de Telecomunicaciones en la escuela profesional de Ing. Electrónica de la UNPRG.

El reto era de relativa simpleza, se nos pedía ingresar un uno o un cero (un bit) al celular y mostrarlo en pantalla. Nuestro grupo inicia el estudio de las soluciones posibles, nuestro trabajo en ese momento no alcanzó a cumplir ese pequeño reto, el mayor logro fue el de un grupo de compañeros que pudo tener respuestas de un equipo celular a comandos AT enviados por un microcontrolador.

Es este no logro el que permite iniciar la búsqueda de una solución al planteamiento del Ing. Eduardo Cerna, pero una solución más general y simple a la disponible, de usar comandos AT, que era prácticamente la única forma de lograr comunicación con celulares en esas fechas. Lo que finalmente lleva a la conclusión de que en la manipulación del software del equipo radica la respuesta y solución, es en esas fechas que se hace masivo el uso que terminales que soportaban juegos y aplicaciones java, específicamente J2ME. Se comienza el estudio de esta plataforma y la programación, ya con un relativo avance, este trabajo es dejado por temas de índole personal, solo bastó menos de dos años para que al querer retomarlo este enfoque fuera descartado al aparecer en el mercado los equipos con sistema operativo Android.

Era claro el futuro de esto o se dejaba ese camino y se adopta la nueva plataforma o simplemente no se hacía nada. Android llegó y con el este nuevo enfoque para la manipulación de equipos, un enfoque que plantea esta tesis.

Índice general

Índice de figuras

Índice de cuadros

Resumen

La presente tesis parte de un análisis de la situación actual y la tendencia del sistema operativo Android y las herramientas informáticas existentes, analiza como pueden ser estas usadas para aplicaciones en el campo de la electrónica. Es a partir de este análisis que se encuentra la necesidad de crear una plataforma que integre estas herramientas para ser usadas en el campo de la electrónica.

Se presenta la herramienta llamada CJAR la cual se define como una plataforma que hace uso de equipos con SO Android, de la informática y mínimo hardware para permitir su uso en telemetría y telecontrol, esta comunicación es a través de la red de Internet o de las redes celulares.

Se crea una aplicación en domótica que interactúa con CJAR. Aplicación en la cual se aplican métodos experimentales y teóricos que fundamenten su funcionamiento, esto con el objetivo de hacer de esta aplicación un modelo que ayude a desarrolladores he implementadores en su proceso de crear aplicaciones de electrónica que se integren eficientemente con la plataforma CJAR, se entregan algoritmos de comunicación y organización del código.

Finalmente se demuestra la transparencia, utilidad y funcionalidad de CJAR a través de su aplicación en domótica.

Abstract

This thesis starts from an analysis of the actual situation and the tendency the Android operating system and existing computing tools, analyzes how they can be used for applications in the electronics topics. It is from this analysis that the need to create a platform that integrates these tools to be used in the electronics topics.

The tool called CJAR is presented, which is defined as a platform that makes use of electronic equipment with Android OS, computing and minimum hardware to allow its use in telemetry and telecontrol, this communication it is through the Internet network or cellular networks.

It is creates an application in home automation that interacts with CJAR. Application in which experimental and theoretical methods are applied to support its operation, this with the aim of making this application a model that helps developers and implementors in their process of creating applications that are efficiently integrated with the platform CJAR, communication algorithms are given and how to organize the code.

Finally, the transparency, usefulness and functionality of CAR is demonstrated through its application in home automation.

Capítulo 1

Planteamiento del Problema Científico

1.1. Descripción del Problema Científico

Los dispositivos inteligentes llámense Smartphone, Tablets, SmartTV, SmartWear, Gafas inteligentes, etc. son ya una realidad, dispositivos como tablets y Smartphons tienen cuotas de mercado de hasta el 87 % de equipos producidos por los distintos fabricantes de equipos de cómputo y Smart devices con aproximadamente 2 106 048,000 el año 2014[1]. Estos dispositivos tienen entre sus características la capacidad de conexión a redes de datos vía wifi, redes celulares o redes privadas, es por tanto una tendencia la diversificación y generalización de estos dispositivos electrónicos. Estos dispositivos inteligentes hacen uso de un software que gestiona sus recursos físicos, un sistema operativo, este sistema operativo no es único para todos pero es claro las cuotas de mercado de los distintos sistemas operativos existentes, es allí donde suena con marcada diferencia el sistema operativo Android, el cual posee una cuota de mercado de 81.2 % con más de mil millones de dispositivos el 2014[2], esto no solo motivado por las cualidades y características del propio sistema como ser software libre, estable, configurable o programable sino también por el grupo de empresas que lo respaldan y usan, todas gigantes tecnológicas lideradas por su benefactor Google, entre estas de empresas encontramos a gigantes como Samsung, Huawei, Alcatel, HTC, Motorola, Qualcomm, LG, etc.

Las redes de comunicaciones están convergiendo a una única red de datos basado en los protocolos TCP/IP con capacidades de transmisión de datos hace algunos años inimaginables y la velocidad de despliegue de esta red única e integradora es de igual forma intensa esto es motivado por la existencia

[1]Fuente: Gartner, Inc.
[2]Fuente: International Data Corporation (IDC)

de un mercado creciente de consumidores de estos productos, solo hay que percatarse como por ejemplo la telefonía celular en nuestro país ya soporta redes 4G para operadoras como Movistar, Claro o Entel. Por tanto la velocidad de transmisión de datos en estas redes se va incrementando dando a si la posibilidad de que más dispositivos se integren a esta red convergente con enorme capacidad de transmisión de datos.

Tomando en cuenta los aspectos mencionados anteriormente tenemos ya a presente la masificación del uso de dispositivos inteligentes con capacidad de conexión a redes de datos de alta velocidad, tendencia en crecimiento. De igual forma los costos asociados van disminuyendo, tanto por el avance tecnológico como por las leyes del mercado ya que por ejemplo en nuestro país se tienen varios operadores no solo con sus redes físicas implementadas si también se tiene a operadores virtuales que generan competencia abaratando costos, esto aunado a la capacidad de innovación de los distintos proveedores de equipos telecomunicaciones como Ericsson, Huawei, Alcatel, Cisco, etc. disminuyen aún más los costos y aumentan la cantidad de usuarios de estos equipos inteligentes con conexiones a redes de datos. Es decir se tendrá cada vez más usuarios de equipos inteligentes con conexión a redes de datos y estos dispositivos van disminuyendo sus costos y aumentando sus capacidades.

Estos dispositivos inteligentes no solo tienen la capacidad de conexión a redes de datos si también integran diversos sensores que le permiten, bajo el control del sistema operativo, la posibilidad de interactuar con el medio y sobre todo tienen la capacidad de ser programados y utilizar su hardware. Existe por tanto ya en nuestro país, y en la mayoría de países, la posibilidad de utilizar estos dispositivos inteligentes con fines distintos a los habituales, con fines de Telemetría y Telecontrol.

Por tanto, planteamos el problema como *La ausencia de una plataforma de tipo general y de fácil manejo que nos permita utilizar las potencialidades de los equipos con S.O. Android en Telemetría y Telecontrol, haciendo transparente al usuario la dificultad de la informática asociada.*

1.2. Formulación del Problema Científico

Se formula el problema científico con la siguiente pregunta ¿Es posible el desarrollo de una plataforma o herramienta, a la que llamaremos CJAR, que haga uso de los equipos con S.O. Android para telemetría y telecontrol?

1.3. Objetivos de la Investigación

1.3.1. Objetivo General

Desarrollar una plataforma de tipo general y de fácil manejo que permita utilizar los equipos con S.O. Android en Telemetría y Telecontrol, haciendo transparente al usuario la dificultad de la informática asociada.

1.3.2. Objetivos Específicos

1. Desarrollar los programas para dispositivos Android que nos permitan la manipulación de su hardware, así como la base de datos y la aplicación web o interface que nos permita interactuar con estos dispositivos Android y su entorno a través del Internet.

2. Adquirir e implementar el hardware necesario para esta plataforma, buscando la minimización de costos y su fiabilidad.

3. Aplicar CJAR a la domótica, implementando esta aplicación de tal forma que sirva de modelo para desarrolladores e implementadores; entregando algoritmos y formas de organizar el código para estas aplicaciones.

4. Demostrar el buen desempeño de CJAR a través de la aplicación en domótica creada.

1.4. Justificación e Importancia

- La carencia de una plataforma de bajo costo y fiable que nos permitan utilizar las capacidades cada vez mayores de los llamados dispositivos inteligentes así como la necesidad de potenciar las capacidades de estos dispositivos pudiéndolos aplicar al campo de la Telemetría y Telecontrol.

- La dificultad que encuentran muchos desarrolladores e implementadores de electrónica para poder hacer uso de estos dispositivos por la cantidad de herramientas informáticas necesarias para poder manipularlos y acondicionarlos a sus proyectos, lo cual los limita o lo que

los desarrolladores informáticos implementen teniendo estos últimos en muchos casos una visión distinta a la requerida para aplicaciones de Telemetría y Telecontrol en Ing. Electrónica.

- Los resultados de la presente investigación permitirán la creación de CJAR, el cual consistirá en software desarrollado para los dispositivos Android, aplicaciones web y requerimientos mínimos de hardware para poder interactuar con el medio y de forma remota.
- CJAR facilitará el desarrollo de aplicaciones de Telemetría y Telecontrol en Ing. Electrónica, haciendo transparente para los desarrolladores e implementadores las dificultades del aparato informático que permite el desarrollo de esta plataforma.
- El presente trabajo mostrará que es posible el desarrollo de este tipo de plataformas, herramientas y procedimientos, sin incurrir en altas inversiones, utilizando software libre con hardware mínimo y de bajo costo.

Capítulo 2

Marco Teórico Conceptual

2.1. Antecedentes

2.1.1. Ideas previas

La existencia de equipos electrónicos de consumo con la posibilidad de poder ser manipulados para fines distintos a los que fueron diseñados no siempre estuvo presente, al inicio solo el fabricante disponía de la tecnología, conocimientos y autorización para la manipulación de los distintos equipos electrónicos de consumo, esto lo extendía en cierta medida a las empresas operadoras y organizaciones que compraban sus equipos para hacer uso directo de estos o vender servicios a terceros a través de estos.

Sin embargo el avance tecnológico empuja a las empresas a la integración de tus distintos equipos y sistemas en redes, por tanto se deben formular y cumplir estándares ya sean por consenso o de facto. Estándares que permitan la interoperatividad entre los dispositivos y sistemas, lo que implica la entrega de información de los equipos para poder interactuar con ellos a través de otros dispositivos.

Esto dio la posibilidad de usar estos equipos con fines distintos a los previstos originalmente, permitiendo el desarrollo de aplicaciones en campos como la Telemetría y Telecontrol usando equipos electrónicos de consumo.

Mencionaremos dos de los principales estándares que permitían manipular estos equipos, estándares que se antecedieron a la llegada de terminales modernos con nuevos estándares; también aremos mención de una alternativa a estas dos que también fué usada para luego mencionar algunos antecedentes ya desarrollados con un sistema operativo actual, Android.

2.1.2. Manipulación de dispositivos usando comandos AT

Una de formas más extendidas entre los años 90 y 2010 de manipular equipos celulares para telemetría y telecontrol era usando comandos AT.

Los comandos AT son un subconjunto de comandos Hayes, un tipo de órdenes que permiten controlar y configurar los módems desde un PC o un terminal. Todos comienzan por el comando AT, de ahí su nombre, que significa *atención*.[1]

Los equipos celulares disponibles en estas fechas podían comunicarse con una PC u otro terminal a través de comandos AT comportándose como modems y por tanto era posible manipular en cierta medida las capacidades de comunicación de estos equipos, algunos de estos comandos se muestran en el siguiente cuadro.

Comando	Función	Comando	Función
ATA	Contestar una llamada.	ATD	Inicia una llamada de datos, para llamada de voz escribir ; detrás del número.
ATH	Colgar una llamada.	ATZ	Resetea el módem.
AT+CBC	Devuelve el estado de batería.		
AT+CGMI	Devuelve el fabricante del equipo.	AT+CGSN	Responde con el IMEI.
AT+CGMM	Responde con el modelo del equipo.	AT+CGMW	Escribe un SMS en la memoria.
AT+CSQ	Devuelve el nivel de señal en recepción.	AT+CGMS	Envía un SMS.
AT+CPAS	Informa del estado del módem.	AT+CGMD	Borra uno o varios SMS.

Cuadro 2.1: Comandos AT

El uso de comandos AT para controlar las capacidades de comunicación de estos dispositivos era y es deficiente ya que es en general dependiente para cada fabricante y dentro del fabricante se tienen comandos para distintas familias de dispositivos. Se tiene igual que conocer las distintas respuestas que dan a estos comandos.
La cantidad de recursos que podían usarse del dispositivo era limita y haciéndose también los trabajos en Telemetría y Telecontrol dependientes de cada fabricante.

Sin embargo usando comandos AT se pudo utilizar equipos celulares para el envío datos a través de SMS o conexión a Internet

2.1.3. Manipulación de dispositivos usando J2ME

La llegada de las nuevas generaciones de telefonía demandó mayores velocidades de conexión de datos lo que implica mejoras no solo en la red sino también en el terminal, terminales con mayores prestaciones de hardware

[1] J. García, *Instalaciones de radiocomunicaciones*, España:Paraninfo, 2012 , pp. 171

que no se limiten a procesar SMSs y llamadas sino también puedan soportar multimedia.

Los distintos fabricantes desarrollaban en forma paralela al hardware los sistemas operativos y las aplicaciones que potenciaban el uso de sus equipos. Era en cierta medida difícil correr aplicaciones diseñadas por un fabricante en dispositivos de otro, era por tanto necesario una plataforma común que permitiera que las mismas aplicaciones corran en dispositivos de distintos fabricantes. Esta plataforma era J2ME.

Sun Microsystems fue una empresa informática adquirida en el año 2010 por Oracle Corporation, que desarrolló el lenguaje de programación Java y la plataforma Java, teniendo esta plataforma tres ediciones.

- Java Platform, Standard Edition o Java SE.
- Java Platform, Enterprise Edition o Java EE.
- Java 2 Micro Edition o Java ME.

La utilización de una u otra versión dependía del terminal donde se ejecutará. El siguiente gráfico nos brinda una idea de esto:

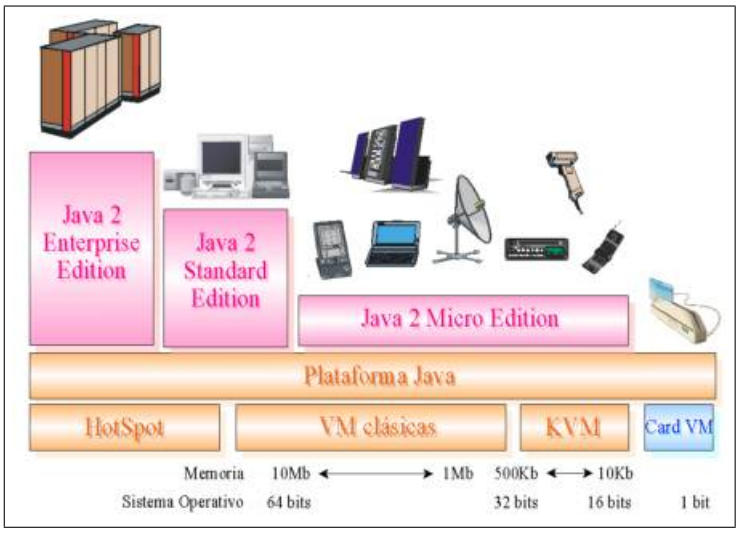


Figura 2.1: Arquitectura de la plataforma Java 2 de Oracle

Java 2 Platform, Micro Edition (J2ME) es una versión de Java que está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes.[2]

[2]S. Gálvez y L. Ortega, *Java a Tope: J2ME (Java 2 Micro Edition)*, España: Universidad de Málaga, 2002 , pp. 3

A diferencia de los comandos AT, la utilización de J2ME permite una mayor manipulación del hardware y menor dependencia del fabricante ya que se convirtió en un estándar en la industria.
El costo de usar esta plataforma para Telemetría y Telecontrol es dejar la relativa simpleza del uso de comandos AT por la programación en Java.

Uno de estos trabajos consistía en hacer uso de dispositivos con J2ME y manipular el puerto USB de estos, que en realidad funcionaria como puerto serial, para conectarlo a un micro y luego a un servidor web. Si bien el trabajo nunca vio la luz por falta de apoyo y cooperación profesional, se realizaron pruebas que confirmaban la posibilidad de usar este enfoque. Este trabajo inicial no pudo ser finalizado u aplicado, por temas de apoyo interprofesional, y sobre todo por la llegada de sistemas operativos como Android.

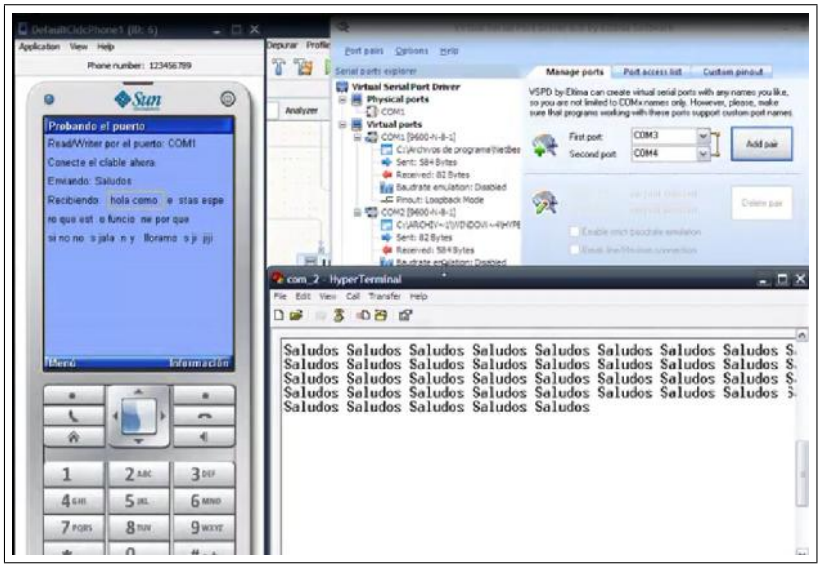


Figura 2.2: Programación y pruebas con J2ME

Al igual que utilizando comandos AT, usando J2ME se desarrollaron muchas aplicaciones, pero casi todas eras a nivel web-dispositivo, no web-dispositivo-planta, en otras palabras no de desarrollaban muchas aplicaciones, por lo menos en nuestro país, para Telemetría y Telecontrol usando J2ME, ya que precisamente el problema radicaba en la adquisición de datos de variables físicas externas al dispositivo.

2.1.4. Plataformas alternativas previas

Uno de los trabajos completamente funcionales que fue desarrollado con el mismo fin pero usando una plataforma distinta a las mencionadas es la

mostrada en la tesis doctoral de Jorge Muñoz Marí, del departamento de Ing. Electrónica de la universidad de Valencia del año 2003. Muños hace uso de un microcontrolador, un módulo bluetooth, un PDA programado en lenguaje C y una aplicación web para la recepción de datos

Se implementa un sistema de monitorización remota enfocado a pacientes que sufren algún trastorno cardíaco, fundamentalmente enfocado a pacientes post-infarto. Dicho sistema es capaz de analizar la señal de electrocardiograma por sí mismo y generar alarmas en caso de que se produzca algún peligro. Para completarlo, se implementa también un modelo de servidor que recoge los datos analizados por el sistema remoto y permite gestionarlos desde el Centro Médico.[3]

Este trabajo si bien guarda similitud en el enfoque con la presente tesis, las herramientas y algoritmos utilizados y desarrollados en el presente trabajo son completamente distintos e independientes a este.

2.1.5. Manipulación de dispositivos usando Android

Los antecedentes de trabajos en Telemetría y Telecontrol usando dispositivos con este sistema operativo no son muchos, la mayoría de trabajos son enfocados y desarrollados para entornos virtuales, es decir aplicaciones dispositivo-servidor, otros son aplicaciones o trabajos para obtener variables de planta pero no son remotizados, es decir dispositivo-planta y en otros trabajos los dispositivos son sólo usados para visualización.

Algunos trabajos que podemos mencionar son:

1. Tesis para optar el Título por Marta Alulema Quitaquis,Escuela Superior Politécnica de Chimborazo, Ecuador (2010).

   Estudio de la comunicación con comandos AT y microcontroladores caso práctico implementación de un prototipo sistema de gestión de alarma para viviendas con monitoreo mediante telefonía celular.

   Utiliza un microcontrolador con el que envía comandos AT a un equipo celular para poder a través de este enviar SMSs con información de sensores que detectan incendios o intromisiones.

2. Tesis para optar el Título por Alatorre Terán Sandra y Carbajall Peña Juan Raymundo, Instituto Politécnico Nacional, México (2011).

[3] J. Muñoz, *Arquitectura abierta escalable para monitorización domiciliaria: aplicación a pacientes con patologías cardíacas*, España: Universidad de Valencia, 2003 , pp. V

Diseño e implementación de un sistema de control vía bluetooth para la iluminación de un hogar basado en una aplicación de S.O. Android.

Utiliza un microcontrolador para la adquisición de datos, los que son enviados al módulo bluetooth y a través de este a un celular con un programa de control hecho en Android.

3. Tesis para optar el Título por Guerra Ruiz Feipe, Pontificia Universidad Católica del Perú, Perú (2013).

   Diseño de un sistema de control domónico y vídeo vigilancia supervisado por un teléfono Móvil.

   Propone una solución *Semi tradicional*, usando cámaras IP para el vídeo y microcontrolador para ingreso de datos de sensores a un controlador y luego a un modem, sólo usando el celular para visualización.

2.2. Estadísticas y Tendencias de Mercado

El mercado es el conjunto de compradores que necesitan o pueden necesitar los productos/servicios ofertados por la empresa. De esta definición se deriva que mercado actual es el que en un momento preciso demanda de un producto/servicio determinado. Y mercado potencial es el número máximo de compradores al cual se puede dirigir la oferta comercial de la empresa.[4]

Uno de los aspectos que determina el desarrollo de un producto/servicio y el enfoque que se le dará y es un determinante si se busca la acogida y posibilidades de crecimiento es el mercado y su tendencia No se puede sacar un producto a la vente u ofrecer un servicio si no hay mercado, no se puede pretender crecimiento sostenible o potenciar el producto/servicio si la tendencia de los componentes o aspectos de lo que depende el producto/servicio es decreciente, no se puede diseñar para la minoría si esta minoría no sostiene al diseñador no si el enfoque del trabajo a realizar es empresarial.

Es por estos aspectos que el estudio del mercado y su tendencia es fundamental y una guía adecuada de cómo, con qué y para quien diseñar un producto o servicio.

Una forma eficiente de hacerlo y mostrarlo es usando las estadísticas realizadas por fuentes confiables.

[4] J. Rivera Camino, *Dirección de marketing, fundamentos y aplicaciones*, España: ESIC Editores, 2012 , pp. 71-72

2.2.1. Estadísticas del mercado de dispositivos electrónicos inteligentes

Para un estudio global del mercado de dispositivos inteligentes tomaremos por referencia los estudios de la empresa Ericsson, en su Ericsson Mobility Report[5] del primer trimestre del año 2015. Esta empresa nos brinda una herramienta online que nos permite obtener las gráficas estadísticas según nuestro interés

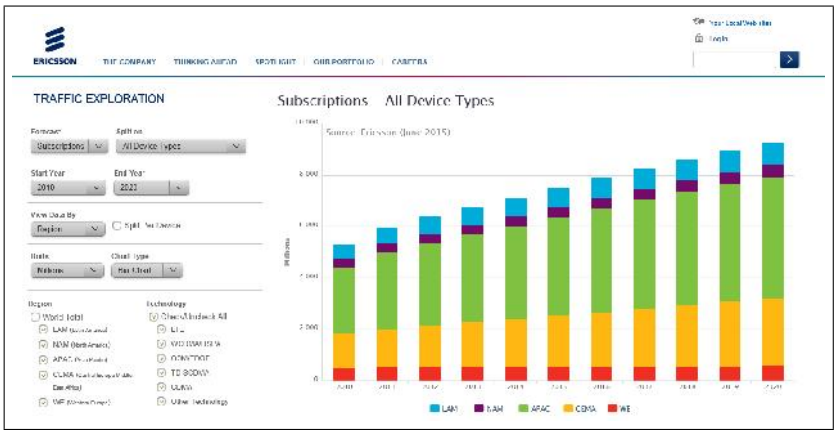


Figura 2.3: Traffic Exploration tool de Ericsson

El número total de suscriptores móviles hasta el primer trimestre del año 2015 fue cerca de 7.2 billones, incluyendo 108 millones de nuevos suscriptores para este periodo.

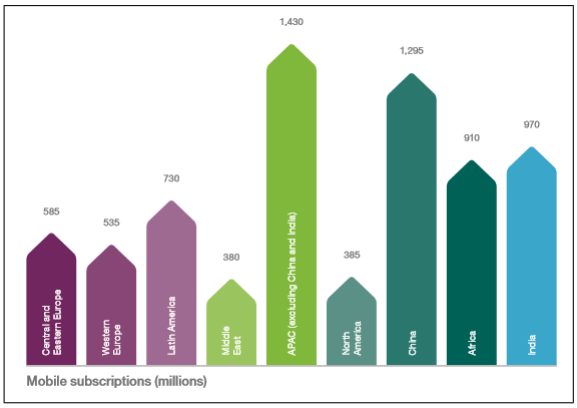


Figura 2.4: Suscriptores Móviles al Q1 2015

[5]Fuente: Ericsson Mobility Report

Se estima que el número total de suscriptores de teléfonos inteligentes se duplique para el 2020, donde el 70 % de la población mundial tendrá un smartphone.

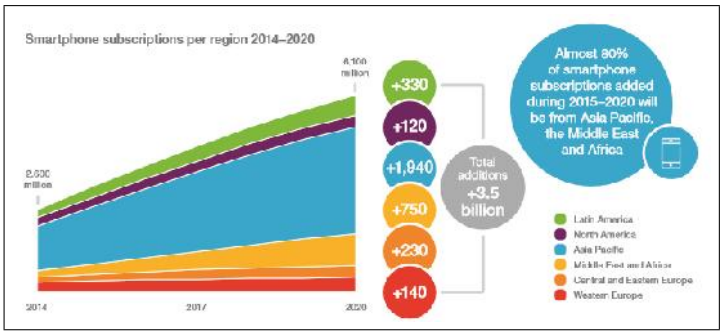


Figura 2.5: Suscriptores de teléfonos inteligentes 2014-2020

Siendo el número de dispositivos de 7.2 billones la penetración global de móviles es del 99 %, ya que la población mundial es aproximadamente de 7.3 billones al 2014[6].

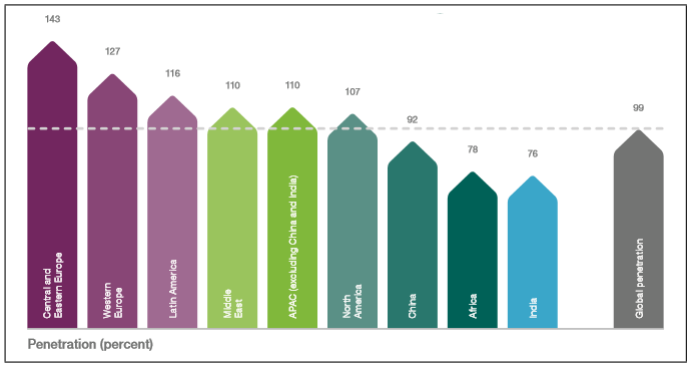


Figura 2.6: Penetración de móviles

[6]Fuente: Banco Mundial

Un cuadro comparativo de los distintos suscriptores con proyecciones al 2020.

Mobile subscription essentials	2013	2014	2020 forecast	CAGR 2014–2020	Unit
Worldwide mobile subscriptions	6,800	7,100	9,200	5%	million
> Smartphone subscriptions	1,800	2,600	6,100	15%	million
> Mobile PC, tablet and mobile router subscriptions	250	250	400	10%	million
> Mobile broadband subscriptions	2,200	2,900	7,700	20%	million
> Mobile subscriptions, GSM/EDGE-only	4,300	4,000	1,400	-15%	million
> Mobile subscriptions, WCDMA/HSPA	1,600	2,000	3,800	10%	million
> Mobile subscriptions, LTE	200	500	3,700	40%	million

Cuadro 2.2: Total de suscriptores móviles

Lo que nos indica que se tendrá 26 billones de dispositivos conectados a Internet

Las previsiones de Ericsson y otras fuentes nos indican que el número de dispositivos crecerá más rápido que el crecimiento de la población mundial, lo cual implica que el nivel de penetración irá en aumento.

Es de mencionar que otros fabricantes también proporcionan estadísticas sobre este mismo tema, como es el caso de Cisco Systems con su informe *Global Mobile Data Traffic Forecast*, un informe muy completo sobre dispositivos móviles y su conexión a Internet[7].

2.2.2. Estadísticas de acceso a internet de dispositos inteligentes

A escala mundial, podemos citar algunos resultados de los estudios de la empresa GlobalWebIndex[8] para el año 2014, que nos muestra el acceso a Internet de los dispositivos y su tendencia.

A la consulta *En el pasado mes, de los siguientes dispositivos, con cuales ha tenido acceso a Internet a través de un explorador o una aplicación?*

GlobalWebIndex indica que el 75 % de usuarios de smartphone encuestados acceden a servicios de Internet

[7]Fuente: Cisco Systems
[8]Fuente: GlobalWebIndex

Figura 2.7: Acceso de dispositivos a Internet

El siguiente cuadro nos muestra un histórico frente a la misma consulta tomando en cuenta el número de dispositivos conectados en millones de unidades.

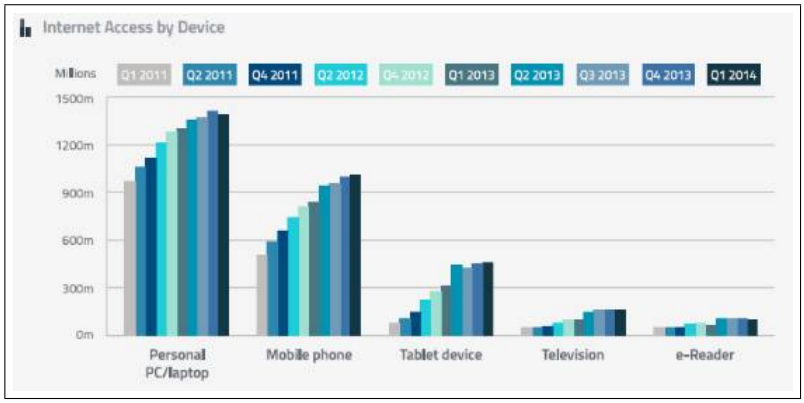


Figura 2.8: Acceso de dispositivos a Internet

Si tomamos en cuenta que las consultas se toman en periodos distintos para el mismo tamaño de muestra, entonces el acceso de teléfonos móviles y tablets a Internet se ha incrementado en los últimos tres años (2011-2014) en aproximadamente 10 % y 22 % respectivamente a nivel mundial.

Notamos un descenso en el acceso a Internet usando PC/Laptop para el primer trimestre del año 2014, lo cual no indica que se tenga menos acceso a Internet sino que este acceso se torna móvil, es decir usando dispositivos móviles como teléfonos celulares y tablets.

A la consulta *¿En un día típico, cuantas horas usas estos dispositivos?*

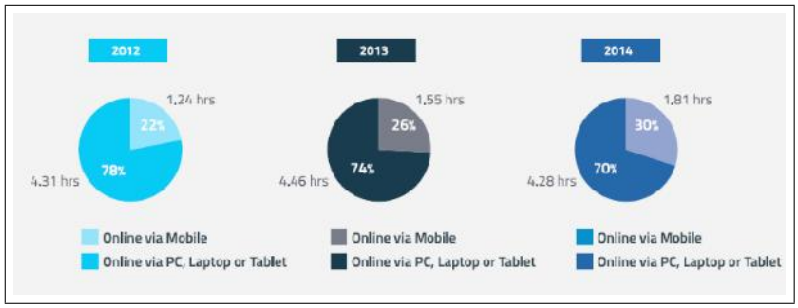


Figura 2.9: Uso diario de dispositivos

Este cuadro muestra claramente la tendencia de conectarse a Internet vía móviles, pero también se puede apreciar el mayor número de horas al día en que los usuarios permanece online, un promedio de 6.1 horas al día.

Haciendo una comparación con el tiempo que dedican a otros medios electrónicos sin conexión a Internet.

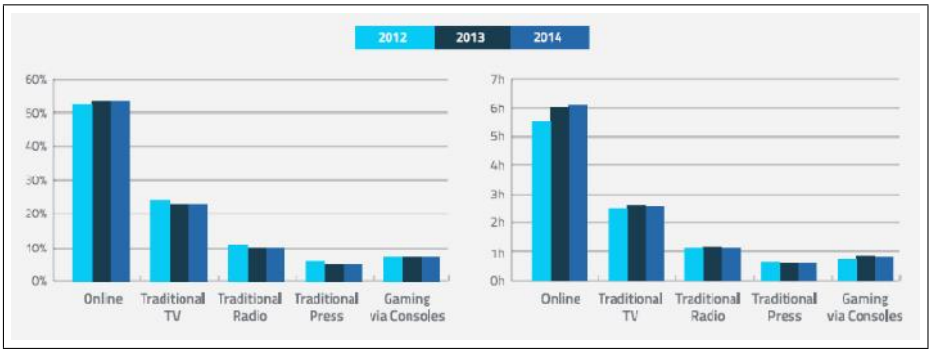


Figura 2.10: Uso diario de dispositivos

Es decir de las horas al día en el uso de equipos electrónicos de consumo más del 50 % son usando equipos electrónicos con conexción a Internet.

Para una comparación por país y el número de horas que permanecen online.

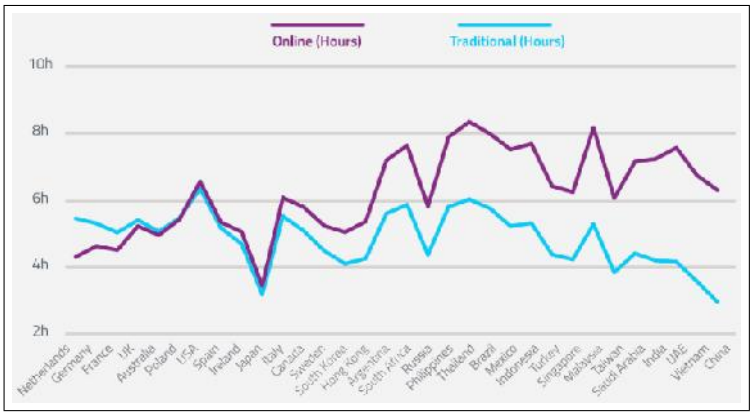


Figura 2.11: Horas online por país

Finalmente citaremos un cuadro de Ericsson Mobility Report, donde podemos apreciar el número de dispositivos inteligentes conectados a Internet desde el año 2010 con proyecciones al año 2020.

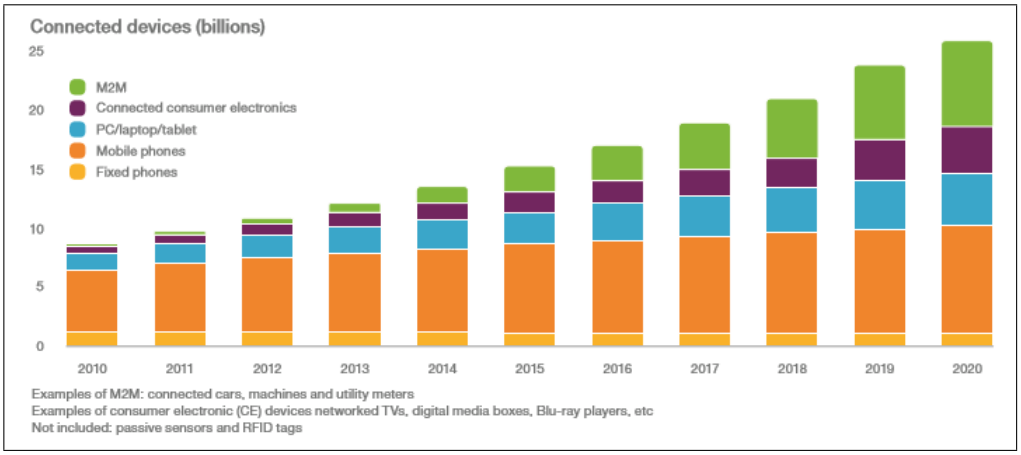


Figura 2.12: Dispositivos conectados a Internet

2.2.3. Estadísticas del mercado de sistemas operativos para móviles

Las siguientes estadísticas son tomadas de StatCounter Global Stats, considerándose teléfonos móviles y tablets[9].

Si bien la cuota de mercado de sistemas operativos para móviles se encuentra dividido, son dos los dominantes Android de Google y iOS de Apple. A escala mundial Android posee una cuota del 62 % mientras iOS el 24 %, dejando al sistema operativo de Windows con el 1.9 %, en el siguiente cuadro podemos apreciar claramente esto.

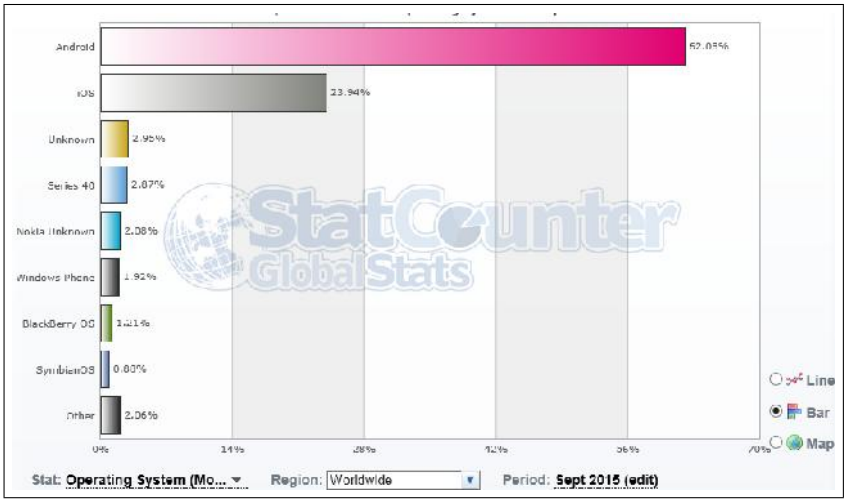


Figura 2.13: Top 8 SO de Móviles y Tablets - Septiembre 2015

Los países donde iOS tiene mayor cuota de mercado que Android son Canadá, Estados Unidos de Norte América, Islandia, Irlanda, Noruega, Suecia, Inglaterra, Japón, Australia, Nueva Zelanda, siendo dominante Android en el resto de países

[9]Fuente: StatCounter Global Stats

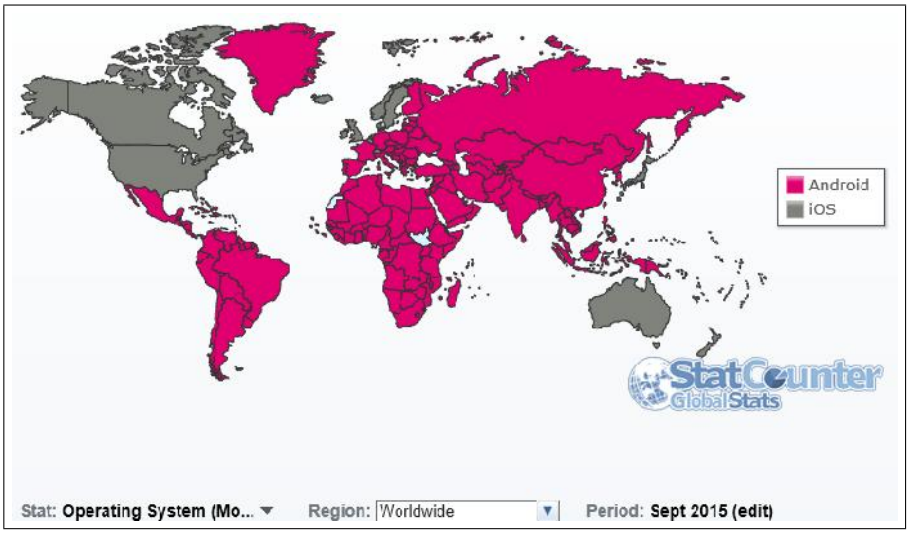


Figura 2.14: iOS vs Android por País, Septiembre 2015

En los últimos tres años ha pasado Android de tener el 27.55 % a tener el 58.24 % mientras que iOS ha pasado de tener el 35.77 % a tener el 27.68 %, otro sistema operativo que corre en terminales modernos es Windows Phone, el cual no ha podido despegar a pesar de los esfuerzos y nuevas propuestas de su creador.

Muchos de los sistemas operativos de los primeros terminales inteligentes van perdiendo terreno como es el caso Series 40 de Nokia, el cual desde Julio del 2014 Microsoft ya no desarrolla.

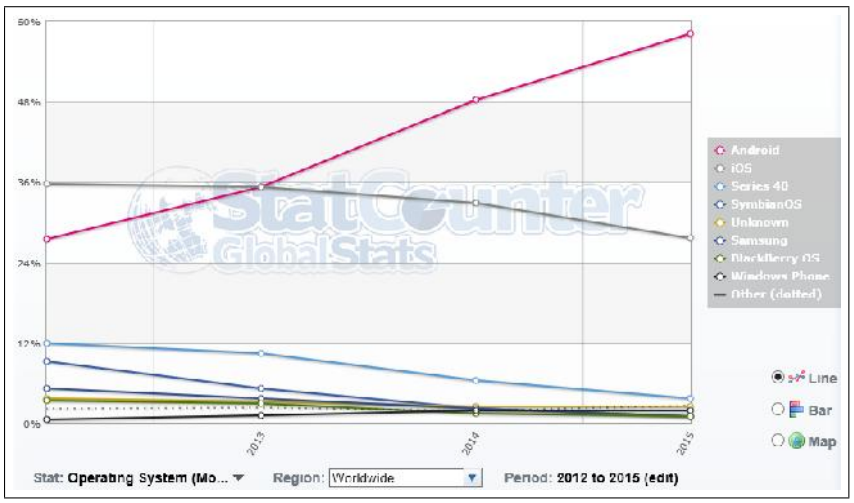


Figura 2.15: Evolució de SO para Móviles y Tablets, 2012 - 2015

En Sur América Android domina el mercado con el 76 % seguido de iOS con el 14 % y Windows Phone con el 4.9 %. Perú sigue la tendencia en Sur América teniendo Android el 73.6 % de su mercado de Móviles y Tablets.

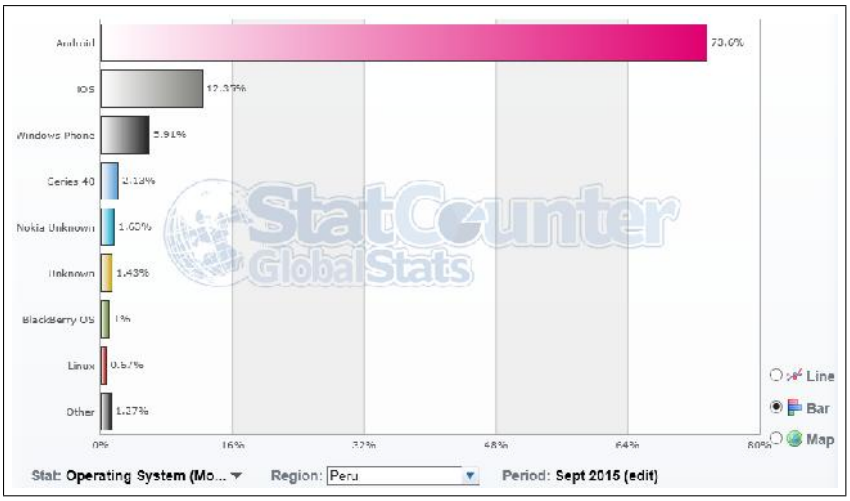


Figura 2.16: Top 8 SO de Móviles y Tablets en Perú, Septiembre 2015

ScientiaMobile provee un reporte con estadísticas sobre smartphone, tablet y teléfonos móviles excluyendo computadoras, laptops, Smart TV, Consolas de juegos, etc., en este informe podemos apreciar la clara ventaja de Android como sistema operativo en smartphone[10].

.

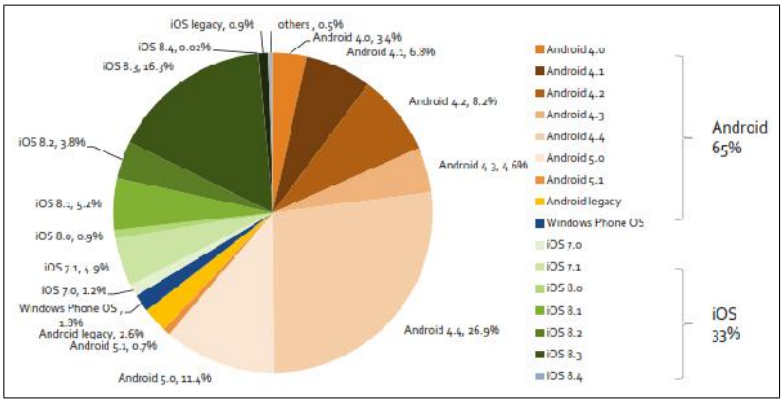


Figura 2.17: SO en smartphone, Abril-Junio del 2015

Básicamente Android domina en todos los mercados para smartphone, superando el 50 % en todos los continentes excepto en Oceanía, el sistema

[10]Fuente: Mobile Overview Report

operativo de Apple mantiene una alta cuota del mercado de smartphone en América del Norte, siendo en Sur América completamente dominante Android.

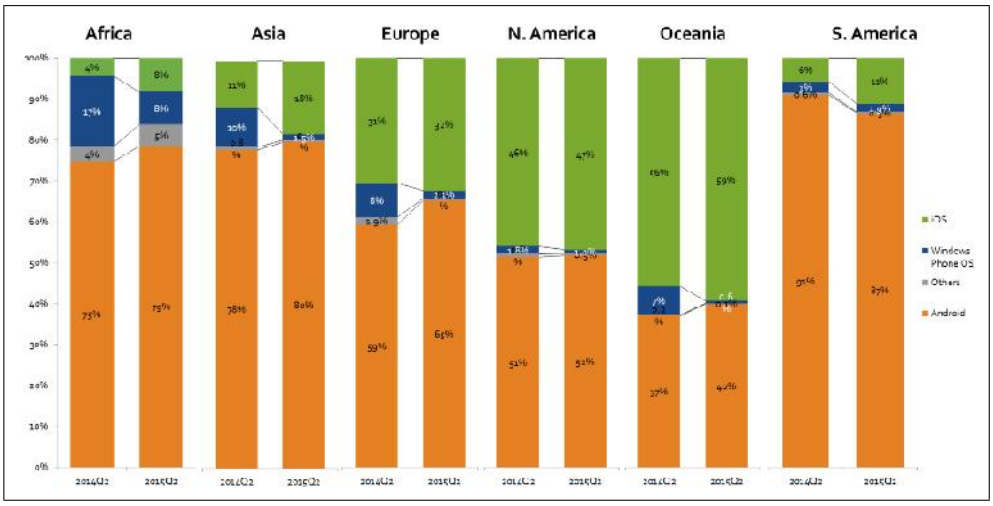
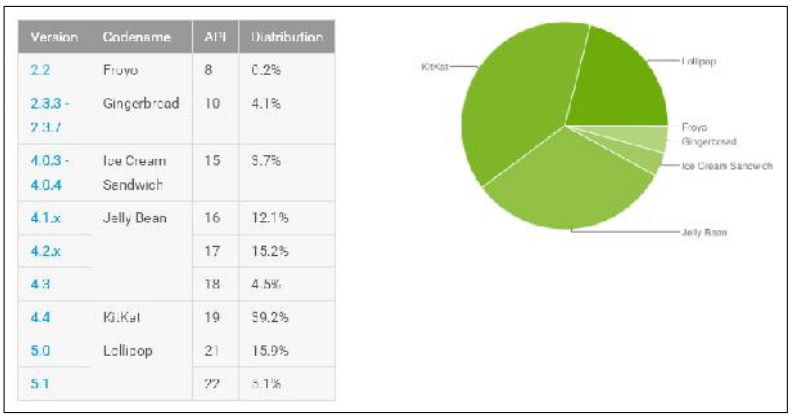


Figura 2.18: SO en smartphone por continente, Abril-Junio del 2015

Dentro del sistema operativo Android, se tiene diferentes versiones. En la página de desarrolladores de Android podemos tener una estadística del uso de estas versiones a Septiembre del 2015[11].



Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	4.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.7%
4.1.x	Jelly Bean	16	12.1%
4.2.x		17	15.2%
4.3		18	4.5%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	15.9%
5.1		22	5.1%

Figura 2.19: Uso de las versiones de Android, Septiembre 2015

[11]Fuente: Android Developer

2.3. Bases Teóricas

2.3.1. Dispositivos Inteligentes Móviles

Los dispositivos se diseñan generalmente en dos modalidades 'no inteligentes' o 'inteligentes'. Un dispositivo no inteligente no cuenta con capacidad propia de procesamiento o decisión; ha sido diseñado para una función específica y cumple con ella. Por otro lado, un dispositivo inteligente puede desempeñar otras funciones. Su capacidad depende de los requerimientos globales del sistema.[12]

Los dispositivos inteligentes modernos tienen capacidad de ejecutar programas diversos de forma paralela, procesamiento multimedia y en tiempo real, los dispositivos inteligentes móviles tienen prestaciones cada vez más cercanas a las computadoras de escritorio y laptops, con capacidad de conexión a distintos tipos de redes, como wifi, 2G, 3G, 4G o bluetooth, pueden funcionar en cierta medida de forma autónoma.

Dentro de los dispositivos inteligentes se consideran los teléfonos inteligentes o Smartphone, Tablets, SmartTV, SmartWear, Gafas inteligentes o las phablets.

Figura 2.20: Sony Smartwatch 2

Para poder apreciar la evolución de los dispositivos inteligentes móviles tomaremos algunos ejemplos fabricados en diferentes años de los llamados Smartphone. Se considera que el primer teléfono móvil inteligente es el IBM Simon Personal Comunicator, cuyo prototipo fue presentado el año 1992.

[12]E. Ramírez y M. Weiss, *Microprocessing Fundamentals, Hardware and Software*, United States of America: Glencoe/Mcgraw-Hill, 1980 , pp. 242

Figura 2.21: IBM Simon Personal Comunicator - 1992

Algunos datos sobre este equipo:

- Desarrollador: IBM
- Fabricante: Mitsubishi Electric Corp.
- Dimensiones: 8 pulgadas (200 mm)H, 2.5 pulgadas (64 mm)W, 1.5 pulgadas (38 mm)D
- Peso: 18 oz (510 g)
- Sistema operativo: Datalight ROM-DOS
- CPU: Vadem 16 MHz, 16 bits x86-compatible
- Memoria: 1 MB
- Almacenamiento de información: 1 MB
- Conectividad: 2400bps Compatible con Hayes módem 33-pin conector 9600bps Grupo 3 enviar y recibir fax, Puerto de conexión de la entrada Tipo PCMCIA 2

El primer teléfono al que realmente se le dio el nombre de smartphone fue el Ericsson R380, el primer aparato se comercializado el año 2000, era un teléfono GSM con sistema operativo Symbian OS.

Figura 2.22: Ericsson R380 - 2000

Otro hito en la historia de los teléfonos inteligentes lo marcó la empresa Apple con el iPhone en el año 2007. El primer equipo vendió más de 6 millones de unidades, contando con una pantalla de 3.5 pulgadas LCD táctil, 4, 8 y 16GB de memoria interna, una cámara de 2MP, memoria RAM de 128MB, contaba con conectividad 2G, Bluetooth y Wifi, corriendo el sistema operativo iPhone OS 3.1.3.

Figura 2.23: iPhone original o iPhone classic - 2007

El primer smartphone con sistema operativo Android fue el HTC Dream construido por HTP el año 2008 con sistema operativo Android 1.0.

Finalmente mencionaremos uno de los terminales más modernos a la fecha del fabricante Samsung, el Samsung Galaxy S6.

Figura 2.24: Samsung Galaxy S6 Edge - 2015

Algunos datos sobre este equipo:

- Fabricante: Samsung
- Dimensiones: 142.1 x 70.1 x 7 mm
- Peso: 132 g
- Sistema operativo: Android OS, v5.0 Lollipop
- CPU: Procesador Exynos 7420 octa-core 64 bits (4 x Cortex-A53@1.5GHz + 4 x Cortex-A57@2.1GHz), GPU Mali-T760
- Memoria: 3 GB
- Almacenamiento de información: Hasta 128 GB.
- Conectividad: 2G GSM, 3G WCDMA, 4G LTE Cat 6, NFC, Bluetooth 4.1, Wi-Fi ac, Ant+, GPS, IR.
- Cámara: Principal de 16 MP con estabilización, Vídeo UHD, Secundaria 5 MP

Se a mostrado en unos cuantos ejemplos la evolución de los Smartphones, donde se aprecia que en tan sólo 23 años han pasado de equipos como el IBM Simon al Samsung Galaxy S6, ver las características de ambos equipos nos da una idea de la evolución de los equipos electrónicos móviles inteligentes.

2.3.2. Android y Android Studio

Android es un sistema operativo móvil de código abierto propiedad de Google. Este se ejecuta en teléfonos inteligentes como el Nexus 5, tablets como el Nexus 7 y todo desde netbooks hasta autos.[13]

Android es un sistema operativo basado en una versión modificada de Linux. Fue originalmente desarrollado por una empresa del mismo nombre, Android Inc. En el 2002 como parte de su estrategia de entrar en el mundo de los móviles, Google compra Android y se hace cargo de su desarrollo, formándose para esto una alianza comercial la **Open Handset Alliance (OHA)**. La OHA se compone de alrededor de 80 empresas, tales como Samsung, HTC, SFR, Oracle, Asus, Qualcomm, etc.

Android es un completo stack de gestión de arranque o boot loader, driver de dispositivos y librerías en APIs (interfaces de programación de dispositivos), incluye aplicaciones y SDK (Kit de desarrollo de software). Android no

[13]Fuente: Google+

es un dispositivo en particular o alguna clase de dispositivos; Es una plataforma que puede ser usada y adaptada a diferentes configuraciones de hardware. Los teléfonos móviles son la clase principal de dispositivos potenciados con Android pero es también comúnmente usado en lectores electrónicos de libros, netbooks, tablets, relojes inteligentes, set-top-boxes, etc.

Sistema de capas de Android

Android se basa en un kernel Linux y se distribuye bajo la licencia **Apache License 2.0**. Separa la capa hardware de la capa lógica, es decir, cualquier teléfono Android puede ejecutar la misma aplicación y, de este modo, se puede crear un amplio abanico de posibilidades para los fabricantes, los usuarios y los desarrolladores. Se muestra a continuación las capas que conforman el sistema operativo Android.

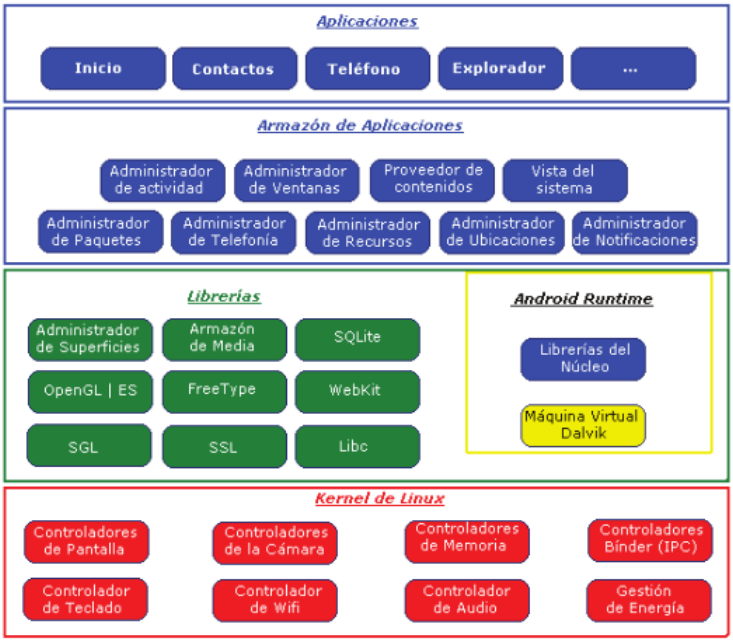


Figura 2.25: Sistema de capas de Android

El SO Android está dividido en cinco secciones en cuatro capas principales:

- **Kernel Linux**: Este el núcleo en que se basa Android. Esta capa contiene todos los drives de bajo nivel para varios de los componentes hardware de un dispositivo Android.

- **Librerías**: Estas librerías proveen todo el código principal de fábrica de un SO Android. Por ejemplo la librería WebKit provee todas las funcionalidades para un navegador web.
- **Android Runtime**: Está en la misma capa que las librerías, el Android Runtime proporciona un conjunto de librerías del núcleo para habilitar desarrolladores que escriban Apps Android usando el lenguaje de programación Java. Android Runtime también incluye la máquina virtual Dalvik, que permite que las aplicaciones Android corran en su propio proceso, con su propia instancia de la máquina virtual Dalvik (Las aplicaciones Android son copiladas dentro de ejecutables Dalvik). Dalvik es una máquina virtual diseñada específicamente para Android y optimizada para dispositivos móviles con baterías, militada memoria y CPU.
- **Armazón de Aplicaciones**:Expone las varias posibilidades del SO Android a los desarrolladores que pueden ser usados para sus aplicaciones.
- **Aplicaciones**: En esta capa superior, se encuentran las aplicaciones que lleva un dispositivo Android (Marcador de teléfono, Contactos, Navegador, etc.).

Versiones del SO Android

El nombre de cada versión de Android corresponde a un dulce diferente.

- Android 1.0 (Apple Pie)
- Android 1.1 (Banana Bread)
- Android 1.5 (Cupcake)
- Android 1.6 (Donut)
- Android 2.0 (Eclair)
- Android 2.2 (Froyo)
- Android 2.3 (Gingerbread)
- Android 3.0 (Honeycomb)
- Android 4.0 (Ice Cream Sandwich)
- Android 4.1 (Jelly Bean)
- Android 4.4 (KitKat)
- Android 5.0 (Lollipop)
- Android 6.0 (Marshmallow)

Componentes una Aplicación Android

- **Actividades(Activities):** Un actividad es un componente de la aplicación que provee una pantalla con la que le usuario puede interactuar para realizar algunas cosas, como hacer una llamada, tomar una foto o ver un mapa. Una actividad es implementada como una subclase de la clase **Activity**.

- **Servicios(Services):** Un servicio es un componente que se ejecuta en segundo plano para una larga operación o desempeña un trabajo para un proceso remoto, un servicio no contiene una interface de usuario, por ejemplo un servicio puede estar reproduciendo música mientras de ejecuta un App diferente o puede estar descargando datos del Internet sin bloquear la actividad del usuario con una interface. Un servicio es implementado como una subclase de de la clase **Service**.

- **Proveedores de contenidos(Content providers):** Un proveedor de contenidos gestiona un conjunto de datos de la aplicación. Se puede grabar datos en un archivo del sistema, una base de datos SQLite, en la web, o cualquier otro sitio para grabado persistente a la que la aplicación tenga acceso. A través del proveedor de contenidos otra aplicación puede consultar o modificar los datos (Si el proveedor de contenidos). Un proveedor de contenidos es implementado como una subclase de la clase **ContentProvider** y debe implementar un conjunto estándar de APIs que permitan a otras aplicaciones realizar lecturas o escrituras.

- **Receptores de Mensajes de Distribución(Broadcast receivers):** Un Receptores de Mensajes de Distribución es un componente que responde a los anuncios, broadcast, del sistema. La mayoría de estos roadcast receivers son originados por el sistema, por ejemplo un broadcast indica que la pantalla se a apagado, la batería esta baja, o una captura de pantalla. Las aplicaciones también pueden originar un broadcast, por ejemplo para dar a conocer a otra aplicación que una descarga de datos es finalizada y puede hacer uso de estos datos. Aunque los broadcast receivers no muestran una interface de usuario, ellos pueden crear una notificación en la barra de estado, para anunciar al usuario que un evento tipo broadcast a ocurrido. Un broadcast receivers se implementa como una subclase de **Broadcastreceivers**.

El Archivo Android Manifest

Antes que el SO Android pueda ejecutar un componente de una aplicación Android, el sistema necesita conocer el componente leyendo el AndroidManifest.xml de la aplicación. La aplicación necesita declarar todos los componentes en este archivo.

El Archivo Android Manifest hace una serie de cosas ademas de declarar los componentes de la aplicación, como:

- Identifica algún permiso que la aplición requiera, como acceso a Internet o lectura/escritura en los contactos del usuario.
- Declara el mínimo nivel de API que requiera la amplicación.
- Declara las características de software y hardware que requiere la aplicación, como una cámara, servicios bluetooth, o pantalla multitouch
- Y más

A continuación se muestra un ejemplo de un archivo Android Manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ejemplo.holamundo.holamundo" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Actividad2"
            android:label="@string/title_activity_actividad2" >
        </activity>
    </application>

</manifest>
```

Etiqueta Inicio Manifest

Se declara la aplicación, con su icono, tema, nombre...

Indica que es la inicial

Declaramos las actividades con el nombre de su clase

Fin Manifest

Estructura de un proyecto Android

Los proyectos Android tienen una estructura predefinida, donde los diferentes componentes están localizados y provee algunas convenciones sobre la conficuración para código fuente java, archivos layout, fuentes de texto, fuentes de imágenes, etc.

El archivo fuente java para un proyecto Android, es colocado en directorio superior *src*. Al mismo nivel se encuentra el directorio *gen*, el cual contiene código generado, este directorio es donde Android autogenera el R.Java.

R.java es una clase interna usado para instalar las fuentes. Las fuentes del proyecto de alojan en el directorio *res*.

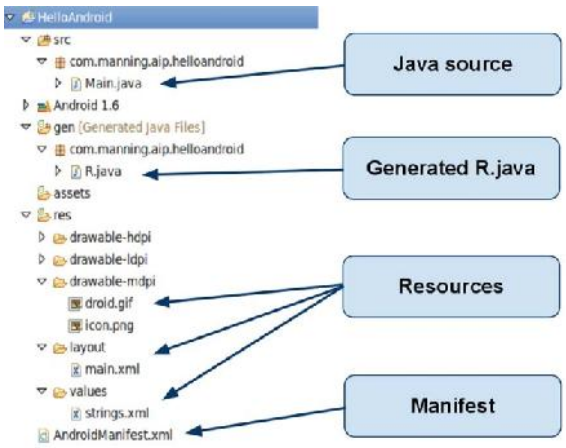


Figura 2.26: Estructura básica de un proyecto Android

Android Studio

Android Studio es el IDE(Entorno de Desarrollo Integrado) oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA.

Android Studio contiene una pantalla previa dinámica de las Actividades, con la posibilidad de agregar algunos componentes de las Actividades usando bloques predefinidos o código en los archivos layouts.

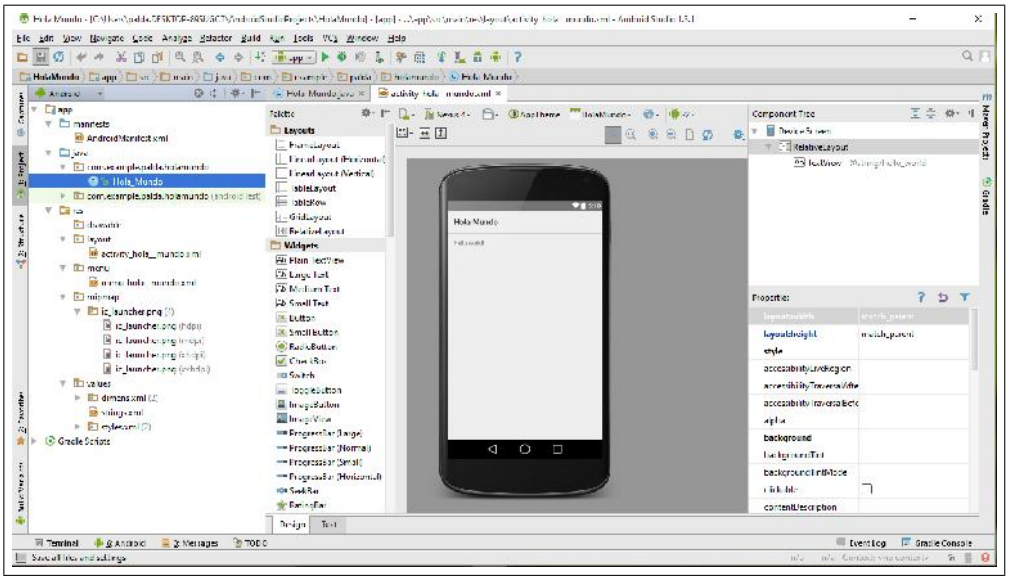


Figura 2.27: Android Studio

AVD Manager

AVD Manager provee una interface gráfica al usuario con la cual se puede crear dispositivos virtuales Android o ADVs(Android Virtual Devices), esto es requerido para la emulación de los programas hechos en Android.

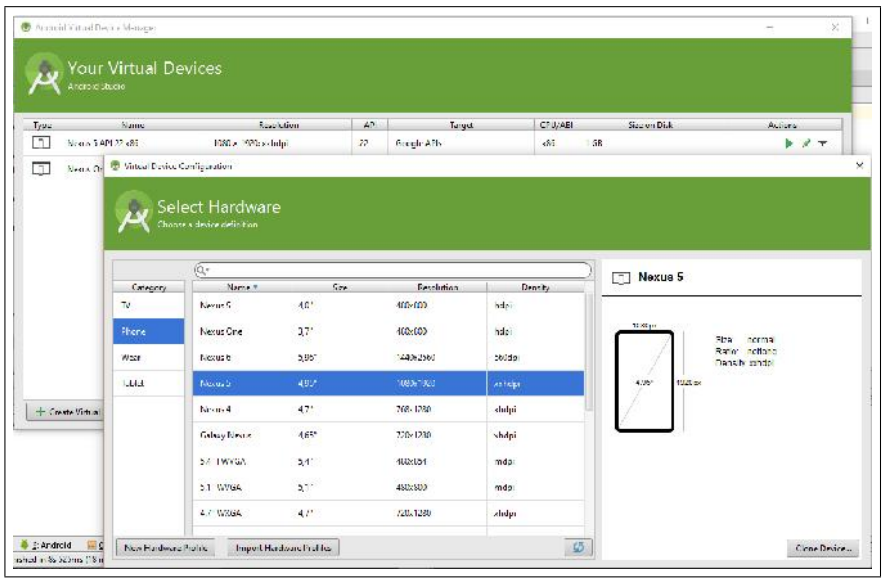


Figura 2.28: AVD Manager

Emulador Android

El Android SDK incluye dispositivos móviles emulados, los cuales son dispositivos móviles virtuales que corren en una computadora, esto permite el desarrollo de aplicaciones sin tener que tener un dispositivo real, sin embargo no están disponibles todas las opciones de configuración, por ejemplo no está disponible la simulación bluetooth.

Figura 2.29: Emulador Android

2.3.3. Servidores y Base de datos

Arquitectura Cliente/Servidor

Cliente/Servidor es una arquitectura de red en la que cada ordenador o proceso en la red es **cliente** o **servidor**. Normalmente los servidores son ordenadores potentes dedicados a gestionar unidades de discos(servidores de ficheros), datos (servidores de bases de datos), páginas web (servidores web) o incluso aplicaciones (servidores de aplicaciones), mientras que los clientes son máquinas menos potentes y usan los recursos que ofrecen los servidores.

Esta arquitectura implica la existencia de una relación entre procesos que solicitan servicios (**clientes**) y procesos que responden a los servicios (**servidores**)[14].

Servidor y cliente Web

El servidor web es una programa que está esperando permanentemente solicitudes de conexión mediante el protocolo **HTTP** por parte de los clientes web. En los sistemas Unix suele ser un *demonio* y en los sistemas Microsoft Windows un *servicio*.

El cliente web es un programa con el que interactúa el usuario para solicitar a un servidor web el envío de los recursos que desea obtener mediante **HTTP**.

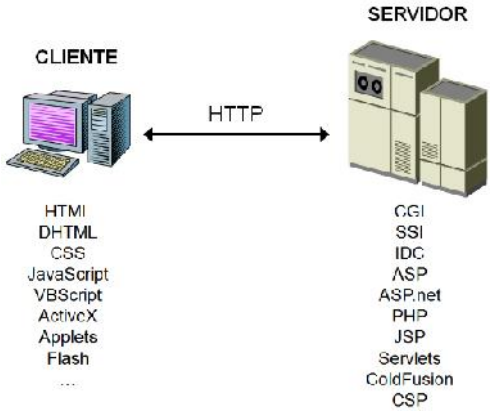


Figura 2.30: Servidor y cliente Web

[14]S. Luján, *Programación en Internet, Clientes Web*, España: Editorial Club Universitario, 2001, pp. 2

Base de Datos

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. La mayoría de bases de datos están en formato digital (electrónico), ya que éste ofrece un amplio rango de soluciones al problema de almacenar datos[15].

Las operaciones fundamentales que tiene que brindar una base de datos son las siguientes: Crear (**Create**), Leer (**Read**), Actualizar (**Update**) y Eliminar (**Delete**) de forma rápida. Además, la base de datos deberá permitir buscar información que sea consistente y válida.

La forma como se interactúa con la base de datos desde las aplicaciones web es usando un lenguaje de programación que pueda solicitar a la base de datos las operaciones mencionadas, como PHP, ASP, Java, etc.

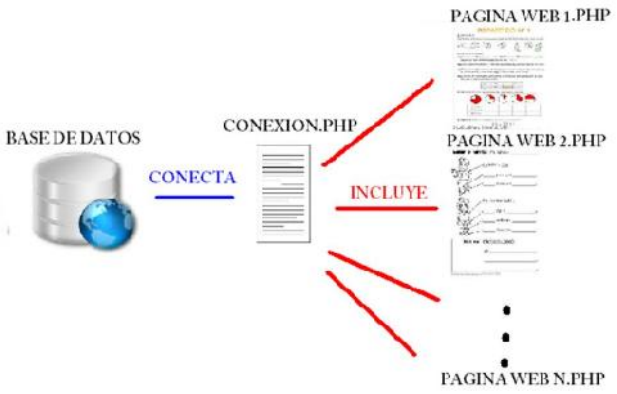


Figura 2.31: Página Web con acceso a una BD

2.3.4. Lenguajes de Programación

Un lenguaje de programación se define como: Lenguaje artificial que se utiliza para expresar programas de ordenador[16].

Cada ordenador según su diseño entiende' un cierto conjunto de instrucciones elementales (lenguaje máquina). No obstante, para facilitar la tarea

[15] J. Lopez, *Programación en tiempo real y bases de datos, Un enfoque práctico*, España: Universidad Pilitécnica de Catalunya, 2001, pp. 119

[16] J. Rodríguez, *Introducción a la Programación, Teoría y Práctica*, España: Editorial Club Universitario, 2008, pp. 4

de programar, se dispone también de lenguajes de alto novel más fáciles de manejar y que no dependen del diseño de cada ordenador.

Lenguaje de Programación JavaScript

En un principio el navegador Web solamente ofrecía al cliente la posibilidad de visualizar información de tipo estática (HTML), sin embargo han surgido diferentes tecnologías que permiten incorporar características multimedia a las páginas e interactuar con el usuario.

Una de las técnicas utilizadas para dotar de dinamismo a las páginas Web en la parte del cliente ha sido el uso de código incrustado dentro del código HTML estático. Estos fragmentos de código son interpretados en tiempo de ejecución en el cliente por el navegador Web y se denominan *Script*.

JavaScript es un lenguaje interpretado desarrollado por Netscape bajo el nombre de LiveScript y que tras formar la alianza SUN y Netscape pasó a tomar el nombre con el que hoy en día es conocido. El objetivo era crear un lenguaje de características similares, en lo que se refiere a sintaxis, a los lenguajes de alto novel pero de uso sencillo, sin necesidad de usar compiladores. Simplemente se introduce código JavaScript dentro de un documento HTML y el navegador se encargaba de interpretar este código[17].

A continuación se muestra un ejemplo de código JavaScript incrustado dentro de una página HTML, este código se encuentra entre las etiquetas <script>y </script>.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>

    <script type="text/javascript">
        function ShowText(str) {
            alert(str);
            return false;
        }
    </script>

</head>
<body>
    <form id="form2" runat="server">
    <div>
        <asp:Button ID="btnSubmit" runat="server" Text="Submit" />
    </div>
    </form>
</body>
</html>
```

[17]F. Maciá, *Administración de Servidores de Internet, De la Teoría a la Practica*, España: Publicaciones de la Universidad de Alicante, 2008, pp. 62

JavaScript y XML Asíncrono: AJAX

XML es un metalenguaje, es decir, un lenguaje para definir lenguajes de marcado. Para definir un lenguaje de marcado es necesario definir un DTD (*Document Tupe Definition)*. Un DTD es la gramática del lenguaje de marcado[18].

AJAX es el acrónimo de Aynchronous JavaScript And XML, no se trata de un lenguaje de programación, en realidad AJAX es una técnica de desarrollo de aplicaciones Web que combina un conjunto de tecnologías con el objeto de proveer al cliente una navegación ágil y rápida y convirtiendo el navegador en un entorno muy dinámico. AJAX usa tecnología del *objeto XMLHttpprequest*. Esto permite que de forma transparente al usuario, el cliente mantiene una comunicación asíncrona con el servidor en un segundo plano. Realiza peticiones GET y POST obteniendo un documento XML como resultado y usando DOM (Document Object Model) le permite leer datos y modificar la página, el usuario verá que la página cambia de aspecto sin la necesidad de re-cargarla

Las tecnologías básicas usadas para el desarrollo de aplicaciones Web con AJAX son:

- **XHTML y CSS:** Como estándares de presentación.
- **DOM:** Para mostrar e interactuar con la información.
- **XML y XSLT:** Manipulación e intercambio de datos(En la parte del servidor).
- **XMLHttpprequest:** Para el envío y la recepción de la información de forma asíncrona.
- **JavaScript:** Como enlace para gestionar todas las tecnologías anteriores.

Lenguaje de Programación Java

Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed frank y Mike Sheridan en Sun Microystems Inc en 1991. El desarrollo de la primera versión duró dieciocho meses y se le llamó 'Oak'. El motivo principal era un lenguaje independiente de la plataforma que se pudiera utilizar para crear software para diversos dispositivos electrónicos, como hornos microondas y controles remotos. Sin embargo, con la aparición de la World Wide web, Java ha sido impulsado al frente del diseño de los lenguajes de

[18] I. Ramos y M. Dolores, *Ingeniería de Software y Bases de Datos, Tendencias Actuales*, España: Ediciones de la Universidad de castilla-La Mancha, 2000, pp. 144

programación, ya que la red también exigía programas portables.

La *programación orientada a objetos es la base de java.* De hecho, todos los programas en Java son orientados a objetos, por esta razón es importante entender sus principios básicos.

La POO parece ser el paradigma de la programación actual, entrando a reemplazar las técnicas de programación estructurada que se desarrollaron a principios de los 70.

La POO basa su ideología en la 'funcionalidad empaquetada', donde los *Objetos* son entes abstractos que tienen un estado, un comportamiento y una identidad, en la POO no es importante para el programador la forma como es construido o funciona internamente un objeto, lo importante es que se comporte como se espera.

El siguiente código muestra el archivo Hola_Mundo.java de un proyecto Android, el popular Hola Mundo.

```
package com.example.palda.holamundo;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class Hola_Mundo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hola__mundo);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
}
```

Lenguaje de Programación Assembler

El lenguaje ensamblador es la representación simbólica de la codificación binaria de un computador, el lenguaje máquina. El lenguaje ensamblador es más próximo a los humanos que el lenguaje máquina ya que usa símbolos y no bits. Los símbolos del lenguaje ensamblador dan nombre a tiras de bits de aparición frecuente, como son los códigos de operación y los especificadores de registros; de esta manera, las personas pueden leerlos y recordarlos.

Además el lenguaje ensamblador permite a los programadores el uso de *etiquetas* para identificar y dar nombre a palabras de memoria que almacenan instrucciones o datos.

Se muestra a continuación un programa escrito en lenguaje ensamblador para una PC IBM (Procesador 8088 de Intel - 1981).

```
; Example of IBM PC assembly language
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32  PROC         ; procedure begins here
       CMP  AX,97   ; compare AX to 97
       JL   DONE    ; if less, jump to DONE
       CMP  AX,122  ; compare AX to 122
       JG   DONE    ; if greater, jump to DONE
       SUB  AX,32   ; subtract 32 from AX
DONE:  RET          ; return to main program
SUB32  ENDP         ; procedure ends here
```

De Programa Fuente a Código Ejecutable

Además de un compilador, se pueden necesitar otros programas para crear un programa objeto ejecutable. El programa objeto creado por un compilador puede estar en un lenguaje ensamblador el cual requiere ser traducido por un ensamblador a código máquina y después se enlaza a algunas rutinas de biblioteca para producir el código que realmente se ejecuta en una máquina.

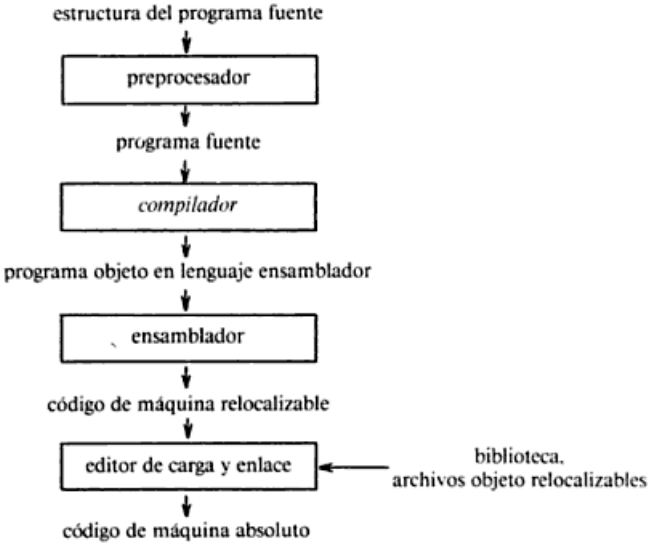


Figura 2.32: Sistema para procesamiento de un lenguaje

2.3.5. Generalidades de un Microcontrolador

Se llama microcontrolador a un sistema de microprocesador incluido todo él en un chip. Dentro de este chip están incluidos la CPU del procesador, memoria y elementos periféricos de forma que se puede realizar todo un sistema de control simplemente conectando elementos externos[19].

Los microcontroladores fueron diseñados para ejecutar tareas específicas en muy diversos campos, como automatisación, instrumentos electrónicos, equipos de comunicación, equipos médicos, electrodomésticos, juguetes, etc.

Los microprocesadores cuentan con una unidad de procesamiento o CPU el cual es un microprocesador diseñado para el procesamiento un programa específico, cuenta además con una memoria, recursos de entrada/salida y sistemas de buses de comunicación que interconectan estas unidades todo esto montado en un chip o circuito integrado.

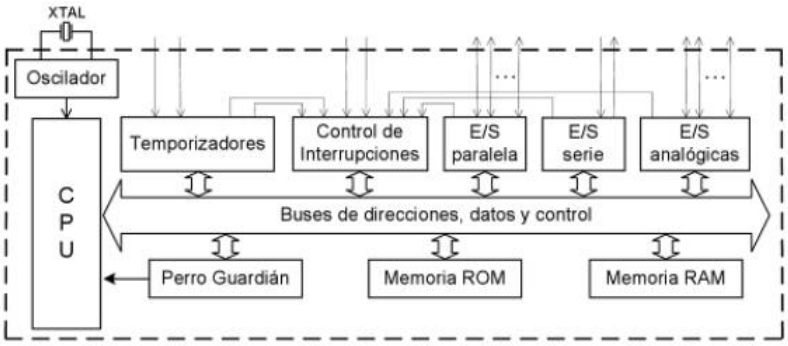


Figura 2.33: Esquema de bloques general de un microcontrolador

En el esquema podemos apreciar los siguientes elementos:

- **Oscilador**: Provee a la CPU de pulsos de sincronización de todas las operaciones, como la velocidad de ejecución de las instrucciones que procesará, puede tomar esta sincronización de una referencia de oscilación interna, dependiendo del modelo y fabricante, o externa como se muestra en el esquema, en este caso la referencia de oscilación externa es un cristal, XTAL.
- **CPU**: Es la unidad principal del microcontrolador, es un microprocesador que ejecutará un programa diseñado para una función específica,

[19]E. Santamaria, *Electrónica Digital y Microcontroladores*, España: Universidad Pontificia de Comillas, 1993 , pp. 257

posee una unidad aritmetica-lógica (ALU) para ejecutar operaciones binarias elementales, ejecutado las instrucciones una a una.

- **Perro Guardián**: O Watchgog Timer es el encargado de velar que el programa se esté ejecutando y no se haya estancado por ejemplo en un bucle infinito o error de hardware, posee un contador interno el cual aumenta continuamente tomando como referencia los pulsos del oscilador, si este registro no es limpiado por el programa el registro se desborda causando un reset, un reinicio del programa.

- **Temporizadores**: Los temporizadores pueden tomar pulsos provenientes del exterior o del oscilador interno e ir aumentando con esto un registro que al llegar completar un número N de pulsos causa la puesta en 0 o 1 según sea el caso de un bit de control el cual funciona como bandera que indica el termino del conteo o desbordamiento del registro.

- **Memoria**: Los microcontroladores disponen de varios tipos de memoria, ya sea ROM, RAM, SRAM, RAM no volátil o EEPROM, etc. El uso de una memoria depende del tipo de información del que haga uso, por ejemplo la memoria RAM es la memoria donde se carga temporalmente el programa a ejecutar, la memoria ROM viene grabada de fábrica con información indispensable para el funcionamiento del microcontrolador, la memoria EEPROM es una memoria no volátil donde se almacenan datos que se esperan conservar aún si se desconecta la alimentación.

- **Puertos de entrada/salida**: Son las vías por las cuales el microcontrolador interactúa con el medio ya sea recibiendo señales o enviándolas, son pines externos. Pueden recibir o enviar señales de tipo analógicas o digitales, establecer comunicaciones estándares como USB, seriales, etc. Estos pines pueden ser multiplexados internamente, lo cual permite cambiar su funcionalidad por software.

- **Control de Interrupciones**: Cuando algunos eventos ya sean internos o externos necesitan una atención inmediata por parte de la CPU, entonces de configuran para poder ejecutar interrupciones, se establecen prioridades y es este bloque el encargado de atenderlos según las prioridades programadas, por ejemplo se puede configurar para que se ejecute una interrupción al recibir un dato por el puerto serie del microcontrolador que obligue al CPU a la ejecución del código que procesa la lectura del dato recibido

2.3.6. Domótica

Tradicionalmente se suele llamar **domótica** a cualquier instalación en la que intervienen dispositivos de automatización de funciones, como encendido y apagado de luces, control de puesta en marcha de aparatos, control de climatización, supervisión de seguridad, etc. La palabra *domótica* realmente se refiere a la automatización del hogar, ya que está formada por la unión de *domo* ('hogar') y *automática*[20].

Una red de control domótica admite muy diversas soluciones tecnológicas, configuraciones, topologías, puede emplear diversos medios de transmisión y comunicarse a través de unos u otros protocolos. Pero todas estas configuraciones poseen una estructura común: la red *percibe* señales del exterior a través de sensores y actúa en consecuencia enviando señales a otros dispositivos denominados *actuadores* o salidas, como por ejemplo un relé, una electroválvula, etc. Sensores y actuadores se denominan genéricamente *nodos* del sistema domótico.

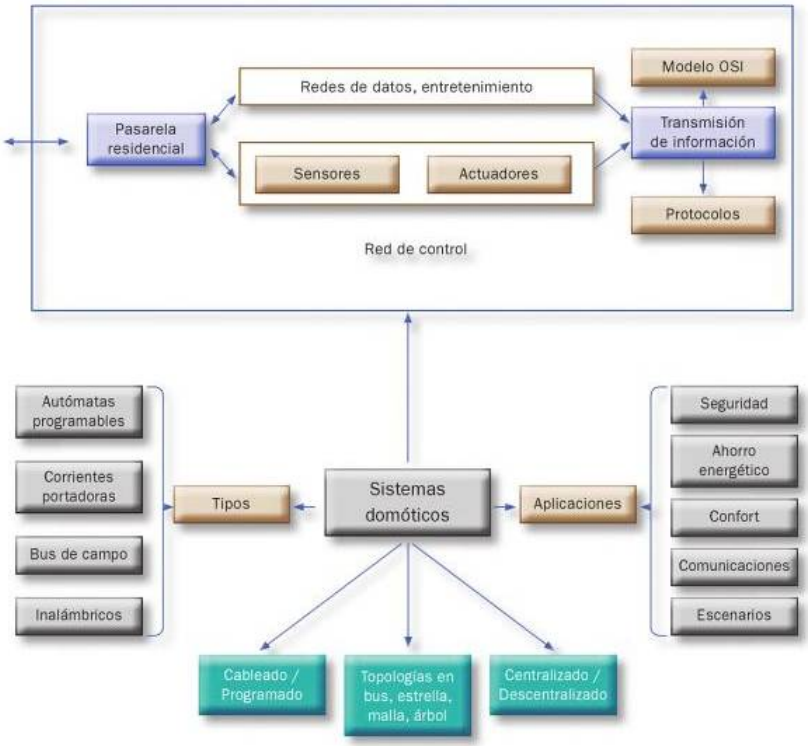


Figura 2.34: Domótica - Mapa conceptual

[20] R. Saavedra, *Automatización de viviendas y edificios*, España: Planeta DeAgostini Profesional y Formación S.L, 2009, pp. 10

2.4. Hipótesis

En el escenario actual es posible *desarrollar una plataforma de tipo general y de fácil manejo que permita utilizar las potencialidades de los equipos con S.O. Android en Telemetría y Telecontrol, haciendo transparente al usuario la dificultad de la informática asociada*, plataforma que llamaremos CJAR.

2.5. Definición de Términos y Conceptos

1. **Telemetría:** Sistema de medida de magnitudes físicas que permite transmitir está a un observador lejano.[21]

2. **Telecontrol:** Mando de un aparato, máquina o sistema, ejercido a distancia.[22]

3. **Dispositivo Inteligente:** Es un dispositivo electrónico, por lo general conectado a otros dispositivos o redes a través de diferentes protocolos como Bluetooth, NFC, Wi-Fi, 3G, X10, etc, que puede funcionar hasta cierto punto de forma interactiva y autónoma.[23]

4. **Software Libre:** Es el software que le da al usuario la libertad de compartirlo, estudiarlo y modificarlo. Llamamos a este software libre debido a que el usuario es libre.[24]

5. **Informática:** Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores.[25]

6. **CJAR:** En el presente proyecto se define como una plataforma que busca ser de tipo general y de fácil manejo que permita utilizar las potencialidades de los equipos con S.O. Android, los programas informáticos y componentes electrónicos adicionales mínimos en Telemetría y Telecontrol, haciendo transparente al usuario la dificultad de la informática asociada.

7. **Usuario Final e Intermedio:** En el contexto de este trabajo, usuario intermedio es el desarrollador o implementador que usando CJAR crea aplicaciones para terceros los que serían los usuarios finales.

[21] Fuente: Real Academia Española
[22] Fuente: Real Academia Española
[23] Fuente: Wikipedia
[24] Fuente: Free Software Foundation (FSF)
[25] Fuente: Real Academia Española

2.6. Operacionalización de Variables

OPERALIZACION DE VARIABLES						
Hipotesis:	La existencia de un numero cresiente de dispositivos inteligentes con capacidades cada vez mayores de procesamiento, configuracion, interaccion y conexion a redes a si como herramientas informaticas potentes y fiables a demas de componentes electronicos de bajo costo y gran disponibilidad hacen posible la creacion de CJAR.					
Tipo	**Variable**	**Definicion Conceptual**	**Definicion Operacional**	**Indicadores**	**Dato**	**Instrumento**
Independiente	Dispositivo inteligente	Dispositivo electronico, por lo general conectado a otros dispositivos o redes , que puede funcionar hasta cierto punto de forma interactiva y autonoma.	1.Equipo electronico con capacidad de procesamiento. 2.Equipo electronico programable. 3.Equipo electronico con conexion a red. 4.Equipo electronico con sensores integrados.	1.Tipo, arquitectura y velocidad del procesador. 2.Nivel al que pueden ser programado. 3.Tipos de conexion a red que soporta y velocidad de conexion. 4.Tipos de sensores que integra.	1. Dual-core de 1.3 GHz, Cuad - core de 1.2GHz, etc. 2. A nivel del sistema operativo Android. 3. Wifi, 2G, 3G, 4G con velocidad de conexion segun red y contratos. 4. Sensores de movimiento, proximidad, GPS, acelerometro, brujula, luz, NFC, giroscopio, grabedad, aceleracion lineal, presion, etc.	1.Hoja de datos 2. Wed of Android developer. 3. Hoja de datos 4.Hoja de datos
	Infraestructura de red	Infraestructura física a través de la cual se transporta la información desde la fuente hasta el destino.	1. Infraestructura fisica que posibilita la conexion a internet de los dispositivos inteligentes. 2.Infraestructura que proporciona una velocidad de conexion a internet a dispositivos inteligentes.	1.Cantidad de dispositivos inteligentes conectados a internet. 2.Velocidades de conexion a internet de los dispositivos inteligentes en infraestructuras de red disponibles.	1. -7.400 millones a nivel mundial, 2014 (Fuente: Cisco System) - 1.040,338 en el Peru, Dic. 2013 (Fuente: OSIPTEL) 2. -2G (GSM) hasta 14Kbps -3G (UMTS)hasta 2Mbps -4G (LTE) hasta 100Mbps	1. Web de los fabricantes de equipos, operadores y reguladores de Telecomunicaci ones y analistas. 2.Programas Android de test.
	Informatica	Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores.	1. Entornos de desarrollo integrados disponibles para programar las aplicaciones wed y Android de CJAR. 2.Lenguajes de programacion y las bibliotecas creadas para realizar programas informasticos. 3.Equipos computacionales necesarios para CJAR.	1. IDEs disponibles para el desarrollo de CJAR. 2.Lenguajes de programacion y bibliotecas disponibles para CJAR. 3.Lista de equipos de computo que utiliza CJAR.	1.Netbeans, Android Studio, Eclipse. 2.Java, PHP, JavaScript, CSS, Jquery, MySQL.. 3.Servidor Web.	1. Estadisticas de uso de los distintos IDEs. 2.Estadisticas de uso de los leguajes de programacion. 3.No aplica por ser dato necesario e independiente.
	Componente electronico	Dispositivo que forman parte de un circuito electrónico.	1.Hardware basico requerido para la implentacion de CJAR 2.Componentes adicionales para poder implementar un pequeño robot. 2.Disponivilidad de componentes electronicos para CJAR.	1.Lista de componentes electronicos utilizados en CJAR. 2.Lista de compontes utilizados para implementar el pequeño robot. 3.Proveedores de componentes electronicos.	1.Cable USB, fuente DC, modulo Bluetooth, placa con conectores e interconexiones. 2.Baterias, motores, engranajes, placa con componentes electronicos e interconexiones. 3. Locales: El Dial, Radiochat, etc. Internacionales: Microchip, Arduino, DigiKey, etc.	1.No aplica por ser dato necesario e independiente. 2.Objetivos del proyecto, asesor, tesista. 3.Listado de proveedores en paginas blancas o internet.
Interviniente	Tiempo	Magnitud fisica que mide la duracion y separacion entre acontecimientos.	1.Tiempo del que dispone y dedica el tesista a la investigacion y desarrollo de la tesis.	1. Cumplimiento del cronograma de actividades mostrado en el perfil de la tesis.	1. Tres meses para todas las etapas del proyecto (A ser demostrado).	1. Cronograma de la tesis.
	Dinero	Medio de cambio de curso legal.	1.Recursos economicos necesarios y suficientes para implementar de CJAR. 2.Recursos economicos necesarios y suficientes para implementar un pequeño robot con CJAR. 3.Recursos economicos adicionales para desarroyar y sustentar la tesis.	1. Monto en moneda nacional del costo de implemenar CJAR. 2. Monto en moneda nacional del costo de implemenar un pequeño robot con CJAR. 3. Monto en moneda nacional de costos adicionales.	1. S/ 200 (Nuevos soles) 2. S/500 (Nuevos soles) 3. S/300 (Nuevos soles)	1. Cuadro de costos de la tesis. 2.Cuadro de costos de la tesis. 3.Cuadro de costos de la tesis.
	Actitud del investifador	Voluntad para encarar las actividades del investigador.	1.Voluntad de investidar, profundizar, analizar y demostrar la hipotesis.	1.Cumplimiento del cronograma. 2.Resultado de la evaluacion del asesor y el jurado.	1. Dato a ser calculado. 2.Dato a ser calculado.	1.Cronograma de la tesis. 2. Jurado y asesor.
Dependiente	CJAR	Plataforma que puede integrar dispositivos inteligentes, herramientas informaticas y componentes electrónicos adicionales mínimos para Telemetría y Telecontrol.	1. Software desarrollado y hardware implementado que permite la utilizacion de dispositivos Android en el control y monitoreo a distancia de apliaciones en el area de Ing. Electronica.	a. Fiabilidad b.Disponibilidad de uso c.Escalabilidad d.Seguridad e.Dificultad de uso. f.Costo de uso. g.Soporte y respaldo.	a. En escala <0.0;1.0>, 0.7 b. En escala <0.0;1.0>, 0.7 c. En escala <0.0;1.0>, 0.5 d. En escala <0.0;1.0>, 0.6 e. En escala <0.0;1.0>, 0.2 f. En escala <0.0;1.0>, 0.3 g. En escala <0.0;1.0>, 0.5	1. Pruebas de desempeño, sustentacion de tesis, jurado de tesis, asesor.

Capítulo 3

Metodología

3.1. Tipo de Investigación

Investigación Tecnológica Física en el campo de la Ing. Electrónica y Formal en el área de programación computacional.

3.2. Diseño de Contrastación de Hipótesis

3.2.1. Modelo Lógico de Contrastación

- Sea X el conjunto de variables independientes, es decir X=$\{x_1, \ldots, x_n\}$ = $\{Dispositivo\ inteligente, Infraestructura\ de\ red, Informatica, Componente\ electronico\}$ y para el escenario actual se tiene un conjunto de valores definidos de X el cual identificaremos como X_a=$\{x_{1a}, \ldots, x_{na}\}$

- Sea Z el conjunto de variables intervinientes, es decir Z=$\{z_1, \ldots, z_n\}$ = $\{Tiempo, Dinero, Actitud\ del\ investigador\}$.

- Sea Y el conjunto de variables dependientes, es decir Y=$\{y_1, \ldots, y_n\}$ = $\{CJAR\}$ = $\{CJAR(Fiabilidad, Disponibilidad\ de\ uso, Escalabilidad, Seguridad, Dificultad\ de\ uso, Soporte\ y\ respaldo)\}$.

- Sea $I = I(X)$ la función *Investigación y Desarrollo*, la cual determirá la relación entre las variables independientes y las reglas de transformación para estas variables de tal forma que el producto de ejecutar $I(X)$ con las variables independientes x_i del proyecto de como resultado Y = $\{CJAR\}$.

Por tanto el trabajo consiste en definir y ejecutar $I(X)$ no como una función matemática pura si no como una metodología de la investigación y desarrollo, metodología que será desarrollada en la Tesis.

Si bien Z no interviene en la función I ya que por definición $I = I(X)$, si es un factor para CJAR, es decir condiciona la *Operatividad* de $I(X)$ y por tanto de su resultado que es CJAR. Si definimos correctamente a Z, es decir al conjunto de sus valores, podremos obtener CJAR a través de $I(X)$, llamemos a este conjunto de valores como *valores óptimos* identificándolos como Z_o=$\{z_{1o}, \ldots, z_{no}\}$.

CJAR tiene indicadores medibles y demostrables, lo que serían sus parámatros, es decir podríamos considerar CJAR como una variable con parámetros o $CJAR = CJAR(Fiabilidad, Disponibilidad\ de\ uso, Escalabilidad, Seguridad, Dificultad\ de\ uso, Soporte\ y\ respaldo)$. Los parámetros de CJAR pueden ser medibles mediante métodos estadísticos, por tanto podremos usar la estadística como herramienta de contrastación de los parámetros de CJAR que sería nuestra variable estadística.

De las definiciones dadas se puede establecer una relación lógica entre las distintas variables del proyecto, la hipótesis y el resultado.

$$(\text{Si } X = X_a \Rightarrow I(X) = CJAR) \Leftrightarrow (Z = Z_o)$$

Capítulo 4

Diseño e Implementación de CJAR

'El diseño no consiste en hacer bello lo útil, ni hacer bueno lo bello, ni hacer útil lo bueno. Un buen diseño encuentra su ciencia, su ética y su estética en su simple globalidad' – Jorge Wagensberg.

En el *Modelo Lógico de Contrastación*, en el capítulo tres se estableció $CJAR$ como una *Variable Dependiente*, producto de ejecutar la función *Investigación y Desarrollo* o $I(X)$, al conjunto de *Variables Independientes* en un escenario actual para el proyecto o X_a, con condiciones no necesariamente optimas pero si bajo cierto control, condiciones que son las *Variables Intervinientes* o Z_o.

$$(\text{Si } X = X_a \Rightarrow I(X) = CJAR) \Leftrightarrow (Z = Z_o)$$

Según este *Modelo Lógico de Contrastación* es necesario como primer paso garantizar que $Z = Z_o$ ya que sin esta condición no es posible crear $CJAR$. Puesto que este conjunto de variables dependen del investigador, son a *priori* considerados con valores óptimos o básicamente controlados lo cual tendrá que ser evaluado, confirmado o refutado a *posteriori* por el jurado, esto tomando en cuenta la tabla de *Operacionalización de Variables*.

El conjunto de variables independientes X y X_a a sido estudiado en el *Marco Teórico Conceptual* tanto en los antecedentes, las estadísticas y tendencias de mercado y en las bases teóricas.

Este capítulo se centra en la ejecución de $I(X_a)$ de tal forma que genere $CJAR$ tomando en cuenta los objetivos de la investigación dados en el capítulo uno, el escenario actual X_a y los parámetros de $CJAR$.

4.1. Consideraciones en el Diseño e Implementación de CJAR

Las siguientes consideraciones en el diseño e implementación de CJAR son en términos del Modelo Lógico de Contrastación los llamados *Parámetros de CJAR*.

1. **Mínimo costo**

   Podríamos separar este parámetro en *costo de uso* y *costo de implementación*.

   El *costo de implementación* en términos cuantitativos fue formulado en el presupuesto del capítulo dos, sin embargo podemos agregar que se busca costos mínimos de implementación, por ser para el tesista un proyecto *experimental*, cuya demanda del producto final no ha sido aún comprobada aunque existe un nicho de mercado. Los riesgos de una inversión en tiempo y dinero no justificado por los ingresos futuros son reales, pero también son reales los dividendos de comercializar el producto. Siendo además posible la utilización de plataformas y herramientas *free*, cuya fiabilidad es demostrada día a día en muchos campos de la tecnología.

   Por tanto, se busca generar el mínimo costo de implementación por los riesgos de usar recursos para crear un producto final que no ha sido aún comercializado y porque existen suficientes plataformas y herramientas free para su implementación.

   El *costo de uso* es el costo para el usuario, el costo que el usuario esté dispuesto a pagar y el desarrollador ofrecer, es por tanto es deseable conocer este punto de equilibrio, lo cual se puede lograr estudiando productos similares y/o estableciendo costos de prueba o diferenciados.

   Existen muchos modelos de negocio para este tipo de productos, se escogerá para CJAR uno combinación de estos.

2. **Independencia y Generalidad**

   Se busca la mayor independencia de CJAR tanto de fabricantes de software y hardware así como que se sustente en *free software*, no sólo por costos sin también para aumentar el mercado potencial de usuarios.

La generalidad de CJAR permitirá mayores aplicaciones por parte de los desarrolladores, se busca implementar interfaces y funcionalidades genéricas de tal forma que pueda ser usada en proyectos diversos.

3. **Disponibilidad, Fiabilidad y Auto recuperación**

   Que el producto esté disponible la mayor cantidad de tiempo posible, esto depende de la infraestructura que de soporte a CJAR y de las condiciones contratadas

   Fiabilidad es la probabilidad de que funcione correctamente el tiempo que esté disponible. Se deben implementar procedimientos y algoritmos que den garantía de una funcionalidad razonable y si deja de funcionar por algún motivo que exista la posibilidad de autodiagnóstico y autorrecuperación de tal forma que continúe en funcionamiento.

4. **Facilidad y Transparencia de uso**

   La implementación de CJAR implica el conocimiento de varios lenguajes de programación y plataformas informáticas, muchas de las cuales no son comúnmente manejadas por los desarrolladores de aplicaciones electrónicas, por tanto la complejidad de esto debe ser transparente al usuario intermedio, que es el desarrollador de aplicaciones en el campo de la electrónica, de tal forma que sólo se centre en la electrónica y software necesario de su aplicación para el usuario final, el cliente, teniendo en CJAR una herramienta de comunicación de fácil uso.

5. **Adaptabilidad**

   De forma independiente y no contradiciendo el parámetro *Independencia y Generalidad*, tanto CJAR Host como CJAR Web pueden ser modificadas a nivel de interfaces de usuario así como en funcionalidad, adaptándose a necesidades particulares, por ejemplo una interface Web (la interface Web de CJAR) diseñada con botones que indiquen derecha o izquierda, apagar o encender, etc. Estas interfaces y funcionalidades bajo demanda no implican mayor modificación del núcleo CJAR. Por tanto CJAR debe tener un núcleo cuyas funcionalidades e interfaces sean modificables son relativa simpleza.

6. **Escalabilidad**

   Dependiendo de la demanda, es necesario que se disponga de la posibilidad de aumento de prestaciones de CJAR, por ejemplo que permita lectura del GPS, acelerómetro, sensor de temperatura, mejoras en las

interfaces, etc. para aumentar el número de aplicaciones posibles.

La aparición de productos de similares prestaciones es probable, por tanto no se puede limitar el desarrollo futuro de CJAR, ya que en el caso de que productos de similares prestaciones a presente mejoren la oferta económica y no sea posible competir a ese nivel, se pueda agregar mayores prestaciones o funcionalidades para justificar los costos de uso.

7. **Seguridad**

   Refiriéndonos en este caso tanto a la confidencialidad de las aplicaciones para el usuario final, la seguridad en sus datos y su acceso así como a la integridad de CJAR, de su información de diseño interno y código creado para su implementación. Se requiere utilizar técnicas que impidan el acceso a información confidencial tanto de CJAR como de la etapa de transporte de información de las aplicaciones finales, de igual forma la utilización de herramientas que oculten o hagan difícil la interpretación o uso del código en que se fundamente CJAR, una de estas técnicas es *ofuscar el código* utilizable en el código JavaScript de CJAR Web.

8. **Comerciabilidad**

   Si el producto generado, CJAR, no es comerciable entonces no es posible más desarrollo de este, no por lo menos de forma aislada por el tesista, no pasará por tanto del ámbito académica al ámbito comercial, lo que implica que muchos de los parámetros mencionados no se podrán cumplir. De igual forma se limitaría el número de usuarios intermedios y finales, por no tener sustento económico la infraestructura que requiere CJAR para su funcionamiento y ampliación.

   Por tanto se debe estimar adecuadamente la inversión tanto en tiempo como en dinero para implementar CJAR y escoger el modelo de negocio más adecuado de tal forma que los costos no hagan inviable su comercialización.

9. **Soporte y Respaldo**

   El soporte técnico debe de consistir tanto en documentación del producto y aplicaciones de este, como también responder a las interrogantes de funcionamiento o errores que encuentren los usuarios finales y los desarrolladores o usuarios intermedios. Esto implica disposición de tiempo no sólo para atender las consultas y crear documentación sino también para corregir posibles errores de CJAR.

4.2. Elección de Plataformas y Herramientas

Las estadísticas y tendencias de mercado del capítulo dos permite apreciar que equipos inteligentes se producen más actualmente, su demanda, tendencia y los sistemas operativos que manejan.

Tomando esto en cuenta y los demás estudios realizados en el Marco Teórico así como los parámetros a cumplir en CJAR las elecciones son las siguientes:

- **Dispositivo inteligente:** *Celulares y tablets*, por su masificación, costos y prestaciones cada vez mayores.
- **Sistema Operativo del dispositivo inteligente:** *Android*, por su cuota de mercado, expectativas de crecimiento, soporte, respaldo y por ser software libre.
- **Lenguajes de programación:** Se usará el lenguaje *Java* por ser requerido en la programación en Android y por ser en si mismo una lenguaje rico, fiable y moderno.

  HTML5, CCS y JavaScript para programación de CJAR Web, por ser entandares modernos, con soporte en casi todos los navegadores. Estos lenguajes nos permites la creación de páginas web dinámicas e interactivas.

  PHP como lenguaje de programación del lado del servidor, para interactuar con la base de datos.

  En cuanto se muestre la aplicación en domótica de CJAR se usará un microcontrolador PIC, el cual será programado en lenguaje *assembler*.
- **Base de datos:** *MySQL* por ser la base de datos de código abierto de mayor aceptación mundial y permite la oferta económica de aplicaciones de bases de datos fiables, de alto rendimiento y fácilmente ampliables.
- **Entornos de Desarrollo Integrado (IDEs):** Para la programación en Android se usará el software oficial *Android Studio*, para la programacion de CJAR Web y PHP de usará *Netbeans*. Para la aplicación de CJAR a la domótica se programará un PIC con *MPLAB X*.
- **Hardware adicional:** Esta primera versión de CJAR requiere un *módulo Bluetooth/Serial*, este módulo puede ser de cualquier fabricante, se escoge en particular el módulo JY-MCU, por su funcionalidad transparente al usuario, disponibilidad en el mercado y bajo costo.

4.3. Estructura y Funcionalidad de CJAR

4.3.1. Estructura general de CJAR

La estructura general de CJAR se muestra en la Figura 4.1, donde se puede apreciar los distintos componentes que intervienen en su diseño e implementación.

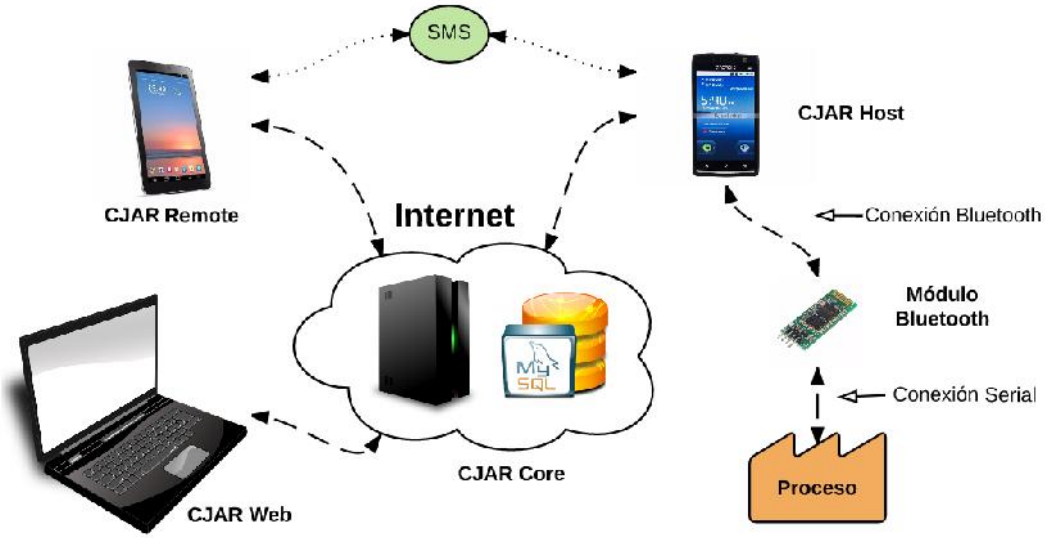


Figura 4.1: Estructura general de CJAR

- CJAR Host y CJAR Remote son programas hechos en Android para equipos con versiones de Android 2.2 (Froyo) a más.
- CJAR Web es una página web dinámica e interactiva.
- El módulo bluetooth es un conversor Bluetooth/Serial.
- CJAR Core consiste en un servidor web y una base de datos MySQL.
- La comunicación es bidireccional en todos los bloques.

Figura 4.2: Logo CJAR

4.3.2. Funcionalidad general de CJAR

En el capítulo dos se definió CJAR como una plataforma que busca ser de tipo general y de fácil manejo que permita utilizar las potencialidades de los equipos con S.O. Android, los programas informáticos y componentes electrónicos adicionales mínimos en Telemetría y Telecontrol, haciendo transparente al usuario la dificultad de la informática asociada.

CJAR permite adquirir y enviar información de un proceso, llegando al proceso vía una comunicación serial, esta comunicación serial es transformada en una señal de radio (Bluetooth) y transmitida a un equipo Android, el cual envía/recibe del Internet los datos para ser tratados en el otro extremo por el usuario usando una aplicación en Android o una aplicación web. De forma alternativa la comunicación con CJAR Host puede ser usando mensajes de texto (SMSs).

4.4. Implementación y Funcionalidad de los componentes

El dispositivo Android que ejecute CJAR Host a de poder intercambiar información con un proceso físico externo, para esto es posible usar en la plataforma Android básicamente dos medios USB y Bluetooth.

Los puntos a considerar para la elección de Bluetooth frente al USB, son los siguientes:

- La cantidad y la tasa de datos que se espera transmitir en aplicaciones hechas con CJAR es cubierta por una comunicación serial.
- Establecer una comunicación USB requiere que uno de los extremos sea Master, si esto de espera del dispositivo en que se ejecute CJAR Host, entonces este debe correr necesariamente Android 3.1 y superiores, pero esto tampoco implica que se pueda en todos los dispositivos Android 3.1 y superiores, ya que depende el fabricante si habilita esta funcionalidad a nivel de hardware y esto generalmente esta implementado en equipos de altas prestaciones y costos, lo cual encarece definitivamente el uso de CJAR.
- Si el dispositivo donde se ejecuta CJAR Host se espera sea esclavo, entonces en el otro extremo se debe tener un dispositivo con capacidad de USB Host, lo cual igualmente no es un dispositivo de bajo costo y complejidad, por ejemplo la familia de microcontroladores PIC 18FXX implementan USB pero en modo esclavo y aun así son de costos moderados, lo cual nuevamente encarece el uso de CJAR.

- El manejo de periféricos USB para muchos desarrolladores con microcontroladores no le es familiar y esto implica dificultar el desarrollo de las aplicaciones finales, reduciendo también el número de usuarios intermedios y finales.
- Este tipo de módulos requeridos es de bajo costo y de alta disponibilidad, con funcionamiento transparente.
- Si tomamos en cuenta los parámetros de CJAR, se busca el menor costo y facilidad de uso. Mientras se establezca una disponibilidad, confiabilidad y autorecuperación razonable, entonces es válido su uso.
- Se han hecho pruebas de funcionamiento de este módulo durante meses sin apagarlo, para estimar su durabilidad y funcionamiento, teniendo hasta el momento el módulo completamente funcional.

Si bien los puntos expuestos justifican el uso del módulo bluetooth frente al USB, esto no impide que a futuro se implemente esta funcionalidad en CJAR Host.

4.4.1. Módulo Bluetooth

Se usará el módulo JY-MCU, el cual requiere una alimentación de 3.6V a 6V, posee por defecto una contraseña (1234) que puede ser modificada por comandos transmitidos al pin TXD. El módulo convierte la señal bluetooth a serial TTL con niveles de voltaje de 3.3V y viceversa, por lo que puede ser conectado directamente a muchos microcontroladores. Posee un LED que que parpadea continuamente al ser energizado y se mantiene encendido si estable una conexión bluetooth con otro dispositivo.

Figura 4.3: Módulo Bluetooth

Los pasos para usar el módulo bluetooth en CJAR son simples.

1. Energizar el módulo con el voltaje adecuado.
2. Emparejar el módulo con el dispositivo que ejecutará CJAR Host usando la clave por defecto del módulo (1234) o la clave que se haya establecido.
3. Ejecutar CJAR Host, si el módulo es el único dispositivo emparejado con el dispositivo que ejecuta CJAR Host, entonces el led dejará de parpadear.
4. Enviar datos a CJAR a través del pin TXD y recibir datos por el pin RXD, los niveles son TTL a 3.3V.

4.4.2. CJAR Host

Tanto CJAR Host como CJAR Remote han sido programados usando Android Studio, el IDE oficial para desarrollo en Android.

CJAR Host es la aplicación Android principal que por un lado intercambia datos vía bluetooth con el módulo y por el otro establece una conexión HTTP para envío y recepción de datos con CJAR Web o CJAR Remote a través de CJAR Core o de forma alternativa puede intercambiar datos vía SMSs con un celular registrado en la aplicación que puede o no tener instalado CJAR Remote.

Para el desarrollo de este programa se han tenido que hacer pruebas en dispositivos reales, ya que el emulador de Android no soporta bluetooth.

Por ser esta aplicación fundamental para el funcionamiento de CJAR, es donde más se buscado cumplir el parámetro tres de CJAR, *Disponibilidad, fiabilidad y autorecuperación*, adicionando a esto la posibilidad de un testeo remoto y comandos remotos de control.

Actividad principal de CJAR Host

Iniciaremos una descripción de su diseño y funcionalidad mostrando la pantalla principal y las distintas opciones que implementa.

Figura 4.4: Actividad principal de CJAR Host

Opción de Registro

Para poder hacer uso del modo automático HTTP de CJAR Host, se requiere que el dispositivo este registrado en la base de datos con un usuario, contraseña y un correo de recuperación. El programa obtiene el IMEI (International Mobile System Equipment Identity) del dispositivo y lo envía junto con los datos del usuario para identificarlo de forma inequívoca en la red. Esto se implementa por temas de seguridad y control de los dispositivos.

Figura 4.5: Opción de Registro - CJAR Host

De igual forma el número de referencia se usará para el modo automático SMS de CJAR Host y como número del que se recibirán comandos vía SMS.

El registro de los datos del usuario en el servidor requiere obviamente conexión a Internet, pero registrar el número de referencia no. Este registro permite controlar cuando se inicia el uso de la aplicación en este dispositivo y para este usuario, de tal forma que según sea una versión gratuita, de extensión o de pago se habilite o deshabilite la funcionalidad de Modo Automático HTTP. Esto se explicará en más detalle en la sección *Modelo de negocio para CJAR*.

Durante el registro el servidor dará distintas respuestas, para indicar a usuario el estado del registro.

Opciones para Pruebas

CJAR Host dispone de cuatro opciones para realizar pruebas, tres de ellas son con conexión al módulo y la cuarta no.

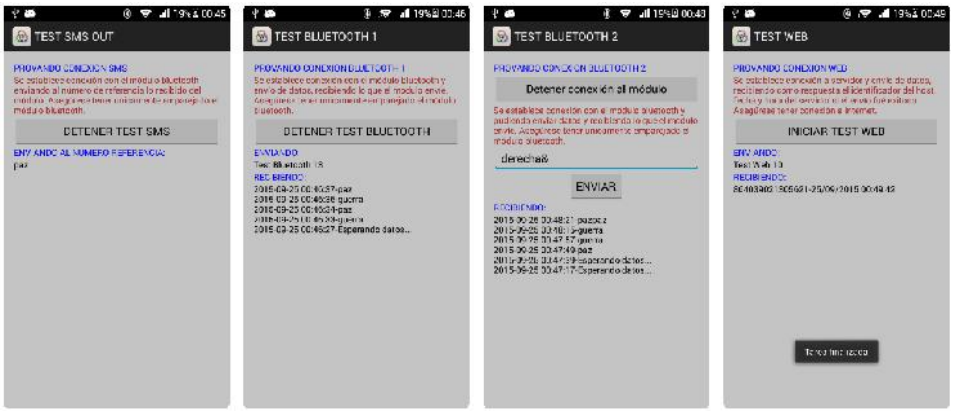


Figura 4.6: Opciones para Pruebas - CJAR Host

El Test SMS Out inicia una conexión al módulo esperando recibir algún dato, cualquiera sea el dato que reciba del módulo lo enviará por SMS al número de referencia sin pedir confirmación o dejar registro del mensaje. Esto requiere por tanto que se tenga registrado el número de referencia en el dispositivo.

Esta prueba nos permite saber si la comunicación con el módulo es correcta (por lo menos si se recibe datos del módulo) y si el dispositivo puede enviar mensajes, esto es útil ejecutar para el posterior uso de CJAR Host en Modo Automático SMS.

Los Test Bluetooth 1 y Test Bluetooth 2, establecen una conexión al módulo enviando en el caso de Test Bluetooth 1 el texto 'Test Bluetooth X', siendo X=1,2,3,4... y en el caso de Test Bluetooth 2 se puede enviar cualquier texto, en ambos casos se mostrará en un ScrollView (vista desplazante) lo que el módulo envíe

Estas dos pruebas nos permiten verificar la comunicación bidireccional con el módulo si este módulo está conectado al otro extremo con algún hardware que envíe/reciba datos serialmente por sus pines. Se puede hacer uso de un conversor de serial TTL a USB o de serial TTL a serial RS232 con un MAX232 y luego un cable adaptador RS232 a USB, para luego conectarlo a una computadora para que usando un software de gestión de puertos COM de hagan pruebas.

El Test Web requiere una conexión del dispositivo a Internet, CJAR Host envía CJAR Core el texto 'Test Web X' con X=1,2,3..., ademas de este dato se envía el IMEI del dispositivo; Si la conexión se establece CJAR Core responde enviando el IMEI recibido y la fecha y hora en que se van recibiendo los datos. CJAR Core guarda registro de estas pruebas.

Esta prueba nos permite confirmar la conexión http entre CJAR Host y CJAR y que ambos están funcionando correctamente. La prueba es útil para poder descartar algunos problemas que se puedan presentar en el Modo Automático HTTP de CJAR host.

Modos Automáticos HTTP y SMS

Los modos automáticos son las dos principales opciones de las que dispone CJAR Host para su función de mantener una comunicación continua con el módulo bluetooth (y a través de este con el Proceso) y con CJAR Web, CJAR Remote o un número registrado.

Se usa como intermediario CJAR Core si la comunicación es por Internet para el caso del Modo Automático HTTP y sin CJAR Core si la comunicación es por SMSs en el caso del Modo Automático SMS.

Ambos modos están diseñados para poder ejecutarse en segundo plano de forma indefinida en un proceso independiente (hilo en Java) de tal forma que su funcionamiento no depende de que se estén o no ejecutando otras aplicaciones. Los modos automáticos de CJAR Host no sólo se ejecutan en segundo plano sino en un componente de Android llamado *Servicio*, al cual el sistema prioriza su ejecución, para evitar de esta forma sea detenido por el SO Android, en el caso de que por algún motivo mayor el SO decida su

detención, por ejemplo por excesiva escasez de memoria, entonces este servicio será reiniciado por el SO apenas disponga de los recursos, en todo caso el inicio, la detención, reinicio o cambio entre modos automáticos es posible hacerlo remotamente, usando mensajes de texto o llamadas telefónicas del número registrado.

En caso que el bluetooth del dispositivo no se encuentre activado CJAR Host lo activará automáticamente sin pedir confirmación ni notificarlo.

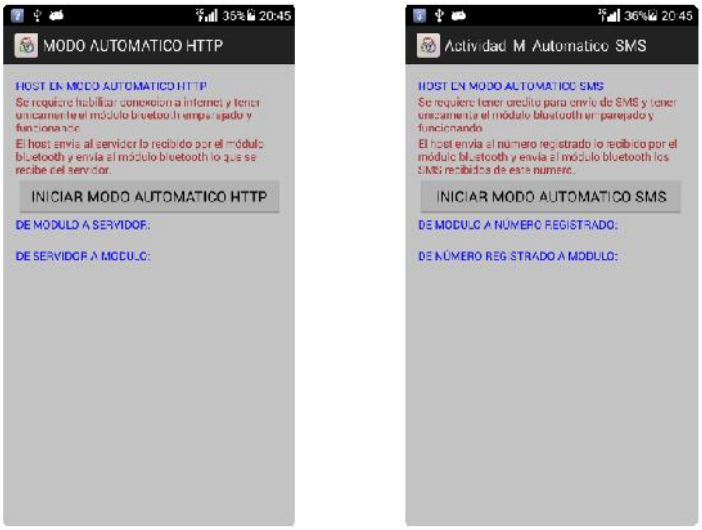


Figura 4.7: Modos Automáticos HTTP y SMS

Los modos automáticos muestran una notificación mientras se están ejecutando.

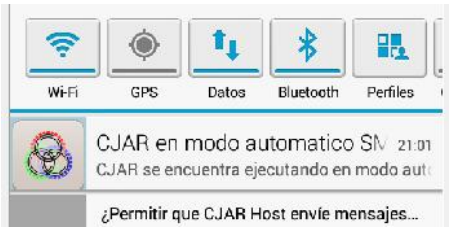


Figura 4.8: Notificación de Modo Automático

Si el número de referencia se encuentre registrado entonces para llamadas de este número se tendrían las siguientes respuestas de CJAR Host.

- Si no se encuentra ejecutando ningún modo automático, entonces se

iniciará en Modo Automático HTTP.

- Si se encuentra ejecutando algún modo automático, entonces se detendrá, esperará 5 segundos y se reiniciará en el mismo modo.

CJAR Host también responde a mensajes de texto enviados desde el número de referencia (el número registrado), si se encuentra en Modo Automático SMS y no es un comando lo enviará al módulo, si es un comando lo ejecutará y si no lo es y no se encuentra en Modo Automático SMS, no ejecutará acción alguna, en todos los casos lo notificará con un SMS al número registrado.

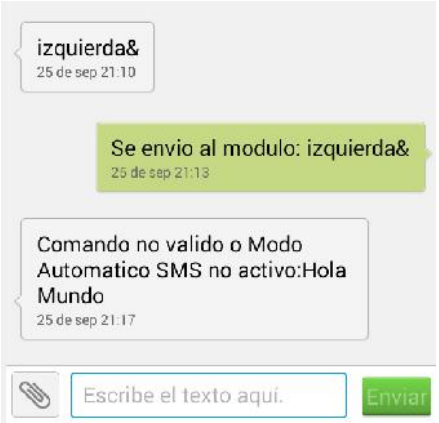


Figura 4.9: Notificaciones SMS de CJAR Host

La lista de comandos que reconoce esta versión de CJAR Host vía SMS son:

- CJARMODOHTTP (Iniciar CJAR Host en modo HTTP): Si ya lo está ejecutando entonces sólo lo notifica, sino iniciará en este modo o primero detendrá el modo SMS si lo está ejecutando antes de iniciarse
- CJARMODOSMS (Iniciar CJAR Host en modo SMS): Si ya lo está ejecutando entonces sólo lo notifica, sino iniciará en este modo o primero detendrá el modo HTTP si lo está ejecutando antes de iniciarse.
- CJARRESETHTTP (Reinicia en modo HTTP): Inicia en este modo si ningún modo se está ejecutando, si algún modo se está ejecutando primero los detendrá antes de iniciarse.
- CJARRESETSMS (Reinicia en modo SMS): Inicia en este modo si ningún modo se está ejecutando, si algún modo se está ejecutando primero los detendrá antes de iniciarse.
- CJARSTOP (Detener modos automáticos): Si ningún modo se está ejecutando entonces no hará nada, sino detendrá el modo en ejecución.

Cuando CJAR Host se encuentra ejecutando en Modo Automático HTTP puede cambiar el tipo de conexión a la red de Internet, conmutar entre la red celular o el acceso vía wifi. Cuando la red de datos esta activada y también wifi el SO Android da prioridad de conexión a wifi, CJAR Host activa o desactiva automáticamente el wifi del dispositivo para permanecer finalmente en la red donde sea posible no sólo conectarse sino también tener acceso a Internet

Al igual que CJAR responde a comandos enviados vía SMS desde el número registrado, también puede responder a comandos enviados por CJAR Web o CJAR Remote. La ejecución de estos comandos está en desarrollo pero ya se debe considerar algunas palabras reservadas para uso futuro. Es aquí donde radica uno de los mayores potenciales del uso de CJAR ya que a través de estos comandos se hará uso de todas las capacidades de las que dispone un dispositivo Android, como puede ser la lectura de sus distintos sensores.

Se muestran algunas de las palabras reservadas, tres comandos implementados y el resto para uso futuro de tal forma que los desarrolladores de aplicaciones las tengan en cuenta.

- CJARRESPONDE
- CJARCOORDENADAON
- CJARCOORDENADAOFF
- CJARIMAGEN
- CJARREINICIA
- CJAROTROS...

CJARRESPONDE, CJARCOORDENADAON y CJARCOORDENADA OFF son los tres comandos implementados en esta primera versión de CJAR. CJARRESPONDE es un comando que consulta a CJAR Host si se está ejecutando en Modo Automático HTTP, si se está ejecutando en este modo CJAR Host responderá 'Hola, el host me está ejecutando'. CJARCOORDENADAON solicita al dispositivo a través de CJAR Host el envío de sus coordenadas de forma periódica, CJARCOORDENADAOFF detiene este proceso.

CJAR Host en modo automático implementa algoritmos que buscan minimizar los posibles errores de conexión o errores que detengan su ejecución, de tal forma que pueda autorecuperarce o en todo caso estos errores puedan

ser manejados en cierta medida de forma remota. Es en CJAR Host donde se centraría gran parte del desarrollo futuro, primero implementando algoritmos adicionales que aumenten su fiabilidad y luego algoritmos que aumenten su funcionalidad.

Enlace a CJAR Web

Esta opción abre el explorador del dispositivo y lo direcciona a la página web de CJAR www.cjarperu.com

4.4.3. CJAR Remote

CJAR Host es la aplicación Android diseñada para interactuar con CJAR Host a través de CJAR Core.

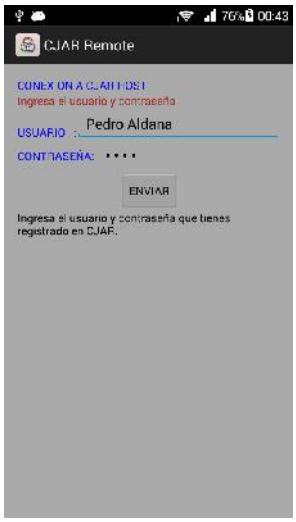


Figura 4.10: Actividad principal de CJAR Remote

CJAR Remote solicita el usuario y contraseña con que se registró el dispositivo que ejecuta CJAR Host, envía estos datos a CJAR Core y este responde con el IMEI del dispositivo que tiene registrado con estos datos, en el caso de usuarios de pago que tienen registrados varios equipos con el mismo usuario y contraseña CJAR Core responderá con todos los IMEIs que ha registrado. Si no es válido el usuario o contraseña lo informará, por ejemplo para el caso de un usuario no registrado daría el mensaje 'Usuario no registrado'.

En el caso de un usuario y contraseña válidos, CJAR Remote mostrará la lista de IMEIs, de los cuales podemos elegir uno para interactuar con el dispositivo al que corresponde este IMEI, esto si es que CJAR Host se está ejecutando en Modo Automático HTTP en ese dispositivo.

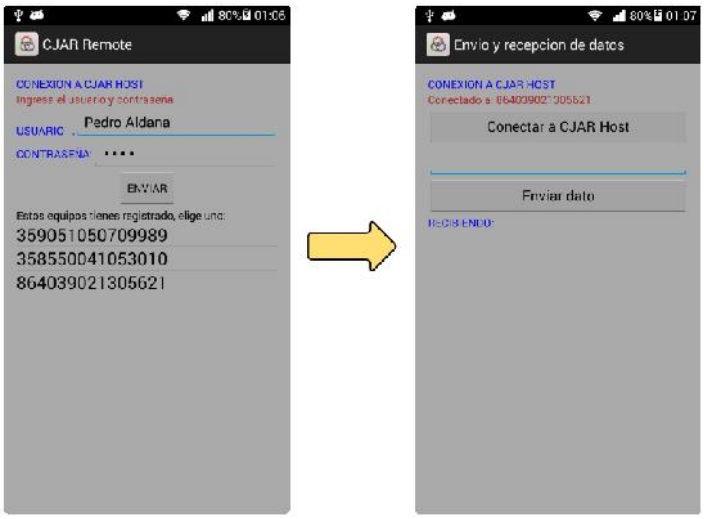


Figura 4.11: Selección del dispositivo en CJAR Remote

Aquí sólo hay que asegurarse tener conexión a Internet, pulsar en Conectar a CJAR Host y empezar a enviar y recibir datos.

4.4.4. CJAR Core

CJAR Core básicamente consiste en el servidor web, la base de datos y los programas hechos en lenguaje PHP para comunicar la base de datos con todos los demás componentes de CJAR.

En la base de datos se guarda registro de todos los usuarios que ejecutan CJAR Host, el tipo de usuario (gratis, de extensión o de pago), contraseñas, correos, coordenadas, fechas de registro de CJAR Host así como las fechas en que se ejecutan los Test de conexión Web y toda la información de la que hacen uso los demás componentes.

Los aspectos críticos en el uso de CJAR Core son la cantidad de datos mensual permitida y la velocidad de conexión, cuando más se requiera de estos dos parámetros, más costos demandará su implementación, mantenimiento y uso, esto es directamente proporcional a la cantidad de usuarios activos de CJAR, estos parámetros se deben dimensionar de acuerdo a la comerciabilidad de este producto/servicio.

4.4.5. CJAR Web

CJAR Web es la interface web de CJAR, en cuyo desarrollo se usaron tecnologías modernas, con gran aceptación y adopción por un gran número de desarrolladores de software, esto por tus capacidades presentes y a futuro, se usó en el desarrollo de CJAR Web: JavaScript, AJAX, CSS y HTML5. El uso de estas tecnologías permite hacer de CJAR Web una página dinámica, interactiva y autoconstruible.

CJAR Web se basa en los conceptos de modularidad, escalabilidad y generalidad. Generalidad de tal forma que la página pueda ser usada por la mayor cantidad de usuarios sin necesidad de hacer diseños para cada uno de ellos, es también autoconstruible según las necesidades del usuario y el tipo de cuenta que disponga ya que los distintos bloques de usuario que la conforman pueden ser agregados o retirados; permite además este diseño el aumento de sus funcionalidades a futuro agregando código sin hacer mayores modificaciones al código que sustenta esta primera versión.

Se muestra a continuación la página que carga por defecto, indicando algunas de sus bloques y opciones.

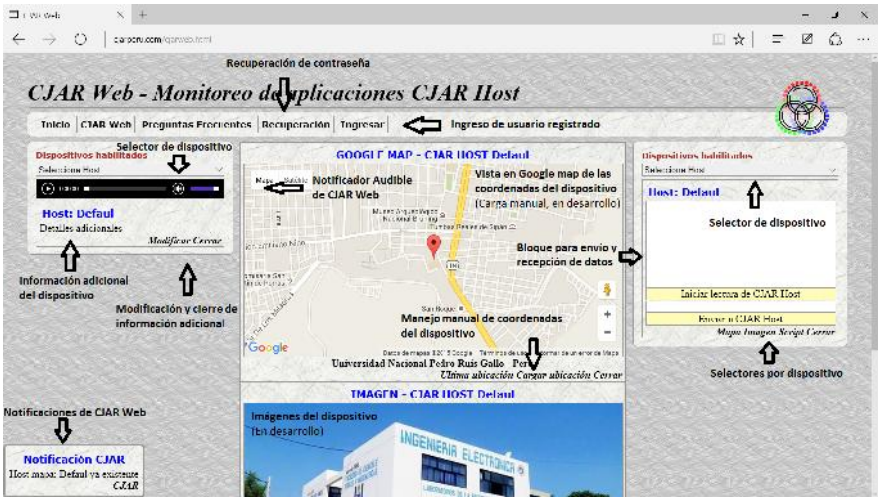


Figura 4.12: CJAR Web - Vista Inicial

Como se puede apreciar la página no muestra la opción de registro ya que al igual que CJAR Remote carga los datos y dispositivos del usuario según los registros hechos en CJAR Host. Esta página por defecto correspondería a un dispositivo cuyo IMEI sería 'Defaul', un valor no existente, estos bloques correspondientes a Defaul pueden ser cerrados.

En los *Selectores de Dispositivos* antes del ingreso sólo se encuentra el IMEI 'Defaul' que será reemplazado por los IMEIs reales registrados, los cuales se cargarán en los *Selectores de Dispositivos* en el momento del ingreso de los datos de Usuario en *Ingresar*.

Figura 4.13: CJAR Web - Ingreso y carga de IMEIs

Teniendo ya los IMEIs en los *Selectores de Dispositivos*, el usuario puede agregar los que considere necesarios o los que utilizará, simplemente seleccionando sus IMEIs, la página se 'construirá' de forma interactiva.

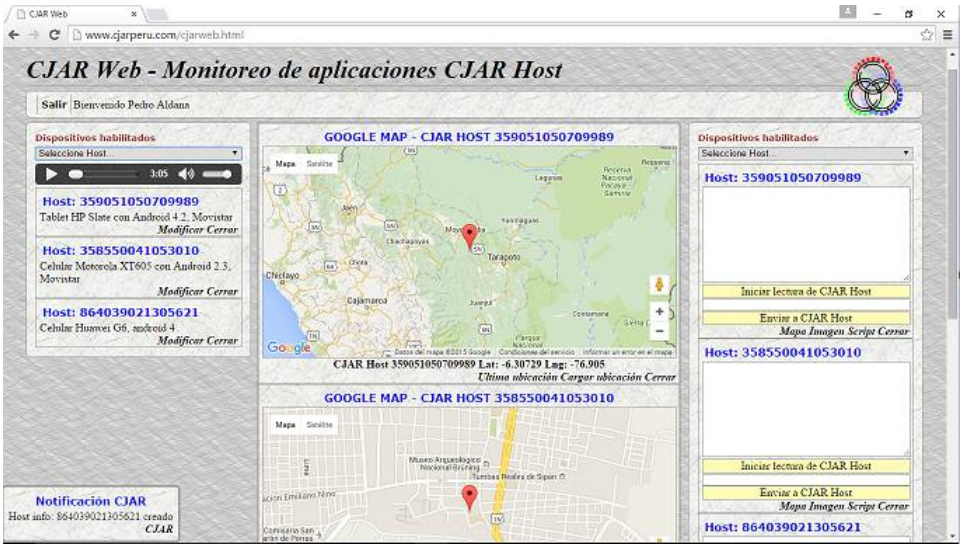


Figura 4.14: CJAR Web - Construcción de bloques de usuario

Cualquier bloque correspondiente a los dispositivos, identificados con sus correspondientes IMEIs, puede ser agregado o retirado en cualquier momen-

to. Se describe a continuación la funcionalidad de los distintos bloques de CJAR Web.

Selector de dispositivo

Es una lista desplegable, que guarda todos los IMEI registrados para un determinado usuario, estos datos son obtenidos de la base de datos en CJAR Core, los cuales fueron enviados en el momento del registro por CJAR Host. Existen dos *Selectores de dispositivos*, uno a la derecha y otro a la izquierda; el de la izquierda al ser seleccionado 'construirá' un bloque de *Información adicional del dispositivo*, mientras que al seleccionar el de la derecha se 'construirá' un *Bloque de envío y recepción de datos*, em ambos casos estos bloques estarán asociados al dispositivo con el IMEI seleccionado.

Notificaciones de CJAR Web

Muchos de los eventos producidos en CJAR Web necesitan ser notificados al usuario, ya sea la confirmación de una petición, de una construcción, un error, etc. es en este bloque donde se realizan estas notificaciones. El bloque por defecto está oculto y sólo aparece cuando quiere brindar información. Ademas de la función de informar eventos en el bloque de *Notificaciones de CJAR Web* es donde se modifica información del dispositivo, como por ejemplo las coordenadas, texto mostrado en el bloque de *Información adicional del dispositivo*, etc. Este bloque siempre se encontrará en la parte inferior izquierda del navegador, por encima de cualquier otro elemento.

Notificación audible de CJAR Web

Este bloque es un reproductor de audio que está diseñado para reproducir automáticamente un sonido (en realidad una canción) en el momento que ingrese un dato al *Bloque de envío y recepción de datos* de cualquier dispositivo. Se puede detener su reproducción con un botón en el reproductor o haciendo clic en cualquier ventana de ingreso de los datos en el *Bloque de envío y recepción de datos*; se puede de igual forma graduar el volumen de reproducción según de guste

Vista Google Map de coordenadas del dispositivo

En esta versión de CJAR Web muestra por defecto las coordenadas guardadas en la base de datos del dispositivo, teniendo en la parte inferior las opciones de cargar la última ubicación o modificar esta.
El ingreso o actualización de las coordenadas puede ser manual o de forma

automática enviando a CJAR Host el comando CJARCOORDENADAON, deteniendo el proceso con CJARCOORDENADAOFF.

Vista de imágenes del dispositivo

Bloque para desarrollo futuro; será asociado a imágenes que envíe CJAR Host provenientes de la cámara del dispositivo.

Información adicional del dispositivo

Este bloque guarda información que el usuario quiere mantener del dispositivo, como puede ser una descripción de este, algunas configuraciones, etc.; es un bloque cuyo texto es fácilmente editable y se guarda en la base de datos.

Bloque de envío y recepción de datos

Es el bloque principal de CJAR Web, mediante el cual se envía/recibe datos al módulo a través de CJAR Host, es un bloque cuya opción de lectura de datos es permanente, haciendo consultas cada tres segundos de forma asíncrona usando AJAX. Si se tienen varios *Bloques de envío y recepción de datos* haciendo lecturas, cada uno de ellos hará las consultas cada tres segundos, estos bloques funcionan tanto para la lectura como para la escritura de forma completamente independiente unos de otros.

Figura 4.15: CJAR Web -Bloque de envío y recepción de datos

Un detalle de estos bloques es que cada uno, por separado, cambia del color que tiene por defecto a un color rojo cuando llega algún dato prove-

niente de CJAR Host, pudiendo retornar al color por defecto haciendo clic en la ventada de ingreso de datos del correspondiente bloque.

4.4.6. Modo Automático HTTP y notificación al módulo

Un detalle importante a considerar durante el desarrollo de aplicaciones con CJAR usando el Modo Automático HTTP en CJAR Host, es que si bien se busca hacer la comunicación los más transparente posible es necesario de algún modo 'notificar' al usuario si es que los mensajes enviados vía Internet por el módulo bluetooth hacia CJAR Remote o CJAR Web están llegando correctamente, de esto se encarga CJAR Core y CJAR Host.

Al recibir los datos provenientes del módulo CJAR Core los guarda temporalmente en la base de datos para luego entregarlos a CJAR Remote o CJAR Web, es en el momento en que guarda estos datos que responde a CJAR Host con un mensaje de 'Operation Successful' o equivalente (esto depende del servidor) indicando que el dato llegó satisfactoriamente, a sido guardado y podrá ser accedido por CJAR Remote o CJAR Web, si se produce algún error responderá con un mensaje de 'Operation Failed' o equivalente enviándolo de igual forma a CJAR Host. CJAR Host interpreta estos mensajes del servidor, si recibe 'Operation Successful' y si él mismo no detecta errores mientras envía entonces enviará al módulo el mensaje **ok&**, pero si recibe 'Operation Failed' o él mismo detecta errores en el envío entonces enviará al módulo el mensaje **error&**. De esta forma el usuario es notificado si sus datos están siendo correctamente recibidos y guardados por CJAR Core y pueden ser accedidos por CJAR Remote o CJAR Web.

4.4.7. Transparencia de uso de CJAR

La función principal de esta primera versión de CJAR, es que CJAR Host se ejecute en uno de los modos automáticos indefinidamente permitiendo la conexión del módulo (y a través de este al proceso) con CJAR Web, CJAR Remote o el número registrado, esto de la forma más transparente posible para el usuario intermedio y final. El usuario sólo vería por un lado dos pines que transmiten/reciben a nivel TTL a 3.3V y por el otro una interfaz como lo son CJAR Remote y CJAR Web.

Se espera en las próximas versiones de CJAR hacer uso de todas las capacidades de los dispositivos con SO Android, siendo la cámara de estos dispositivos lo primero en ser agregado.

4.5. Posibles causas de error y soluciones en CJAR

Es recomendable primero ejecutar todas las opciones de prueba que se dispone en CJAR Host, se descarta de este modo muchas probables malas configuraciones durante el armado.

Sin embargo es posible encontrar entre las distintas causas de error las siguientes:

1. No tener en cuenta los mensajes ok& y error& enviados por CJAR.
2. Extensión del texto que se puede enviar por CJAR Web, CJAR Remote y CJAR Host en modo automático.

   Para cada caso el máximo número de caracteres: CJAR Web (20), CJAR Remote (39), CJAR Host (39).
3. Uso de palabras reservadas, comandos de CJAR, como parte de los mensajes.
4. La combinación de caracteres &# y #& es usada internamente por CJAR para la validar la integridad de los datos, se sugiere no usar estas combinaciones.
5. CJAR Host no responde a SMSs ni llamadas
   - Confirmar que el número registrado en CJAR Host corresponda al número que se está usando.
   - Registrar nuevamente CJAR Host, ya que puede que se visualice el número en la pantalla pero internamente el SO borrada estos datos.

6. El número de referencia para un dispositivo 1 que ejecuta CJAR Host es un dispositivo 2 que también ejecuta CJAR Host y tiene como número de referencia el número del dispositivo 1.

   Este escenario puede generar bucles en los mensajes SMS, ya que ambos dispositivos están configurados por CJAR Host para responder de forma automática.

4.6. Modelo de negocio para CJAR

Para mostrar con claridad el modelo a usar en CJAR, se hará primero la descripción de algunos modelos de negocio existentes y posibles para CJAR.

1. **Software de pago con tiempo indefinido (M1):** Un pago único por el software 'completo' por cada usuario o dispositivo.
2. **Software de pago por suscripción (M2):** Pago periódico (mensual o anual) por el uso del software 'completo' por cada usuario o dispositivo.
3. **Software de pago con periodo de prueba (M3):** Cualquiera de los dos modelos anteriores, con un periodo de uso gratuito del software 'completo', puede ser 15 días, un mes, un año, etc.
4. **Software gratis con servicios agregados de pago (M4):** Software 'completo' gratis de forma indefinida; las aplicaciones personalizadas (diseños a demanda), soporte técnico y asesoría son de pago.
5. **Software gratis con publicidad (M5):** Software 'completo' de forma indefinida con publicidad incrustada.
6. **Software básico gratis con funcionalidades completas de pago (M6):** Una parte del software es gratis y otra de pago. Se puede escoger la parte del software que sea de pago como la que tenga las mayores funcionalidades o/y las que demanden costos al desarrollador durante su utilización.

La elección de uno u otro modelo no sólo depende de maximizar los ingresos a presente, si no que además este modelo sea sostenible en el tiempo y sea fácilmente ajustable a las expectativas de la mayor cantidad de usuarios. Se debe también tener en cuenta la existencia o creación de software y/o hardware con similares o mejores prestaciones y costos; se considera también el estado actual del mercado y las tendencias.

Puesto que el único hardware que requiere esta versión de CJAR es un módulo bluetooth, este puede ser adquirido y escogido por el usuario localmente, por tanto el producto/servicio a vender es el Software. CJAR no será basado en uno de los modelos de forma exclusiva si no que será la combinación de algunos de los modelos mencionados

CJAR Host tendrá dos versiones, una será la *versión gratuita* la cual primero será M5, para luego de un tiempo de uso pasar a ser M6, quedando finalmente con publicidad. La parte de software que no estará disponible será el Modo Automático HTTP, ya que este modo hace uso de CJAR Core, el cual demanda costos de mantenimiento. La otra versión de CJAR Host será la *versión de pago*, la cual será M2 sin ningún tipo de publicidad y suscripción anual. Ambas versiones estarán disponibles en Google Play Store para su descarga.

CJAR Remote tendrá dos versiones, una estará disponible en Google Play Store y será M5 y la otra estará disponible en la página web de CJAR y será M4 sin publicidad.

CJAR Web tendrá dos versiones, para usuarios registrados como gratuitos la cual será M5 sin considerar el término 'completo' y a la vez será M6. La otra versión es para los usuarios de pago, la cual será M2 sin ningún tipo de publicidad, este pago se considera ya incluido en costo de la *versión de pago* de CJAR Host.

La versión gratuita de CJAR estaría disponible con la funcionalidad de Modo Automático HTTP de CJAR Host durante 30 días, pudiendo el usuario terminado este plazo solicitar una extensión del periodo gratuito con esta funcionalidad por dos meses más, para lo cual debe hacer un perfil de la aplicación que piensa desarrollar y enviar la solicitud por correo, entregando el trabajo final y la demostración de su funcionalidad en un plazo de 15 días (estos 15 días forman parte de los dos meses). Esta información formará parte de la documentación de ejemplos de CJAR, la cual estará disponible en su página web. De esta forma se busca fomentar el desarrollo de aplicaciones y compartir esta información y experiencia con otros usuarios.

Capítulo 5

Aplicación de CJAR a la domótica

5.1. Consideraciones en el Diseño de la Aplicación

Así como en el desarrollo de CJAR partimos de premisas o principios que regirán su funcionamiento y desarrollo, también para una aplicación de CJAR tendremos las siguientes consideraciones.

1. **Balance entre complejidad y simplicidad**
   La aplicación ha de ser lo suficientemente elaborada para mostrar a los desarrolladores e implementadores la posibilidad de que sus creaciones no simples sean compatibles a CJAR, pero de igual forma la aplicación no ha de ser tan compleja de tal forma que permita la fácil compresión de su funcionamiento e integración con CJAR que es en si mismo el motivo central del presente trabajo.

2. **Uso del paquete CJAR estándar**
   La aplicación creada no ha de exigir alguna modificación en el paquete CJAR estándar, paquete presentado en el capítulo 4, ya que es este paquete estándar el que será distribuido de forma masiva. Lo cual tampoco implica la no creación de paquetes personalizados de CJAR.

3. **Reutilización de código**
   Al ser esta una aplicación de CJAR y no CJAR es posible mostrar el código fuente que se genere; y al poder ser mostrado ha de buscarse que pueda ser reutilizado y entendido por desarrolladores de aplicaciones, muchos de los cuales tendrán dificultades en los primeros trabajos. Entregarles un código reutilizable permitirá eliminar en cierta medida la dificultad de integración a CJAR.

5.2. Estructura y Funcionalidad de la Aplicación

5.2.1. Estructura y funcionalidad general

La aplicación en domótica consiste en el monitoreo y control remoto de una variable analógica (temperatura) y de tres digitales (detección de puerta abierta, luz y alarma sonora) de una vivienda. Siendo además la variable analógica controlada por un sistema de control de tipo proporcional con realimentación negativa.

Se muestra a continuación un diagrama con la estructura general de la aplicación en domótica:

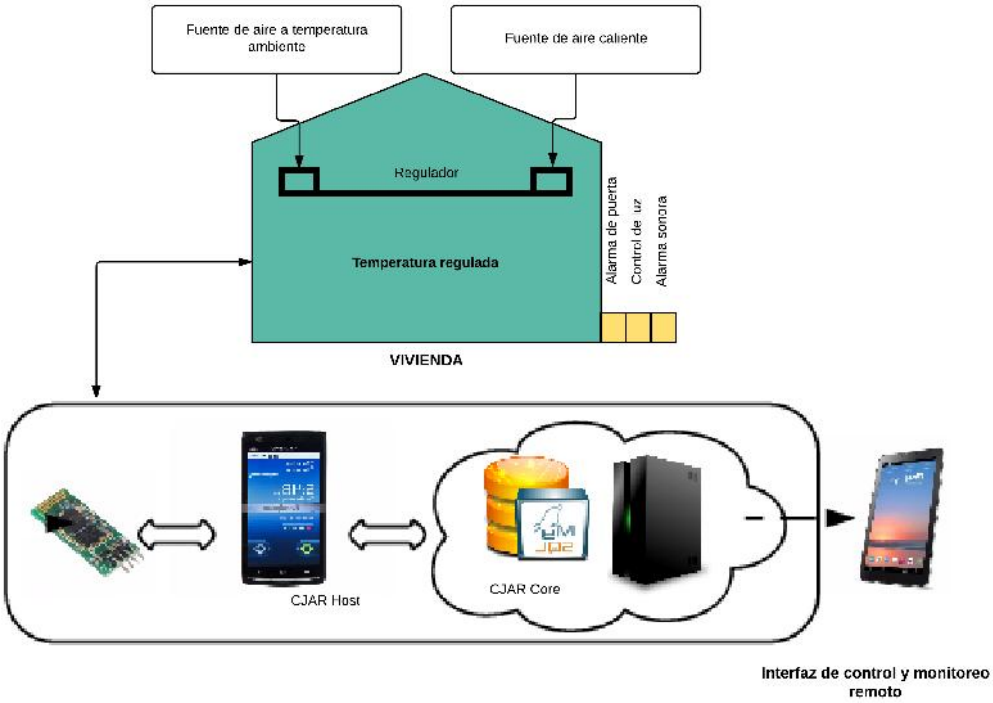


Figura 5.1: Estructura general de la aplicación en domótica

Si bien la aplicación cuenta con una 'Interfaz de control y monitoreo remoto' proporcionada por CJAR Web o CJAR Remote, también tiene un panel de control local, por lo tanto a menos que se indique lo contrario las funcionalidades pueden ser manejadas local o remotamente.

Las funcionalidades se detallan a continuación.

- Encendido y apagado de la luz y el proceso de control automático de temperatura.

- Habilitación o deshabilitación de la detección de puerta abierta.
- Conmutación entre el control local o remoto de la aplicación.
- Lectura remota del estado de todas las variables ya sean analógicas (temperaturas) o digitales.
- Reseteo del programa de control.
- Envío automático de mensaje de puerta abierta.
- Control automático de la temperatura de la vivienda con un sistema de control proporcional con realimentación negativa.
- Alarma sonora al detectarse puerta abierta.
- Colocación local de la temperatura de referencia (temperatura deseada).

5.2.2. Hardware y Maqueta

Se muestra a continuación los distintos componentes y sus conexiones.

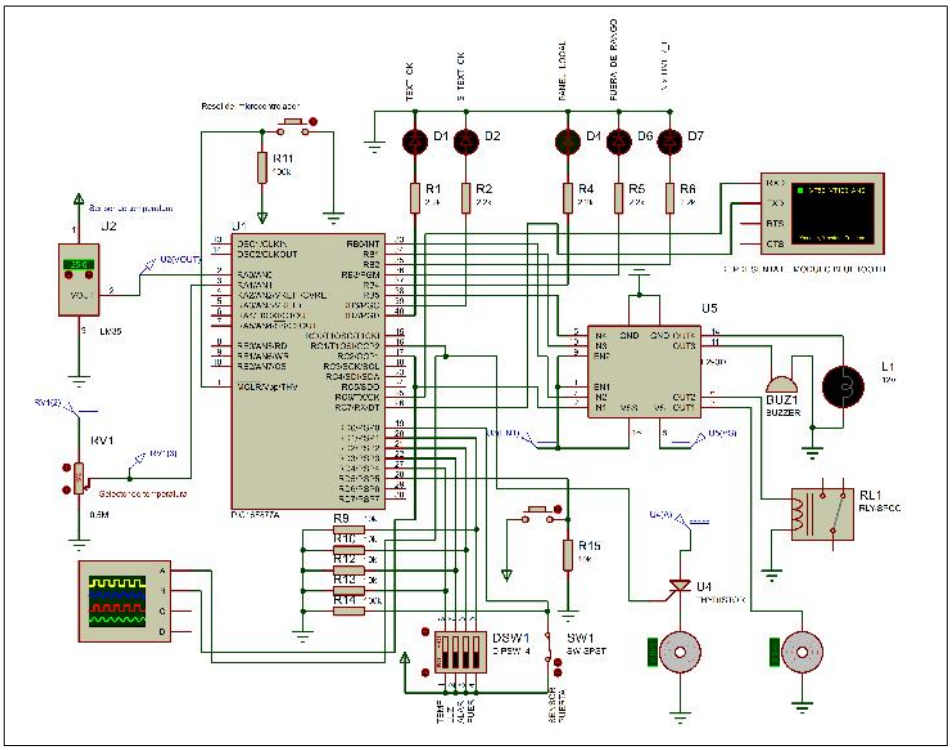
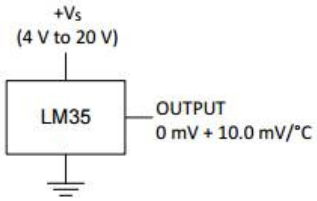


Figura 5.2: Componentes y conexiones de la aplicación

Como se puede apreciar, el hardware consiste en:

- Un microcontrolador PIC 16F877a el cual estará a cargo de la comunicación serial con el módulo bluetooth, control entradas y salidas digitales y la adquisición de voltajes analógicos a través del ADC (Analog Digital Converter), voltajes provenientes del selector de temperatura y del sensor de temperatura para ejecutar el programa de control automático de temperatura.

- Un sensor de temperatura LM35, que nos entrega un voltaje linealmente proporcional a la temperatura.



- Un potenciómetro que nos entrega un voltaje de referencia (y por tanto una temperatura de referencia), el cual puede entregar voltajes de $0V$ a $1{,}5V$.

- LEDs indicadores:

  -*Panel local*: Encendido indica control local, apagado indica que el control es remoto.

  -*Fuera de rango*: Se enciende si la diferencia entre las temperaturas del sensor y el selector excede cierto límite 'l', para esta aplicación $l = 9.^{o}$C.

  -$N.T_{Timer}$: El estado de este LED conmuta a la velocidad $N.T_{Timer}$, siendo $N \in \mathbf{N}$ y T_{Timer} la velocidad máxima conseguida con el Timer 1 del microcontrolador.

  -*Text_ ok* y *S_ Text_ ok*: Estos LEDs nos indican si los textos recibidos por el puerto USART del PIC corresponden o no a alguna palabra predefinida. *S_ Text_ ok* se mantiene encendido mientras llegue algún carácter perteneciente a una palabra predefinida, *Text_ ok* se enciende si se detecta el carácter de fin de texto & después de una secuencia correcta de caracteres y esta resulta en una palabra predefinida, y ambos LEDs se encienden si no se reconoce los caracteres o la palabra recibida.

- Dos pulsadores, uno para reset del microcontrolador y otro para conmutar entre control local y remoto.
- Switches para control local:

  -*Temp*: Permite o desactiva la ejecución del programa de control automático de temperatura.

  -*Luz*: Apago o encendido de la luz.

  -*Alar*: Permite o desactiva la alarma sonora.

  -*Puer*: Permite o desactiva la detección de puerta abierta.
- Integrado L293D, el cual consiste en 4 puentes H, los cuales repotencian las señales digitales del PIC para el manejo de las cargas. Este se alimenta de forma independiente con $5V$ para indicar el voltaje de control y con $12V$ como voltaje de salida a las cargas. Me muestra a continuación su diagrama funcional (Los diodos mostrados son internos).

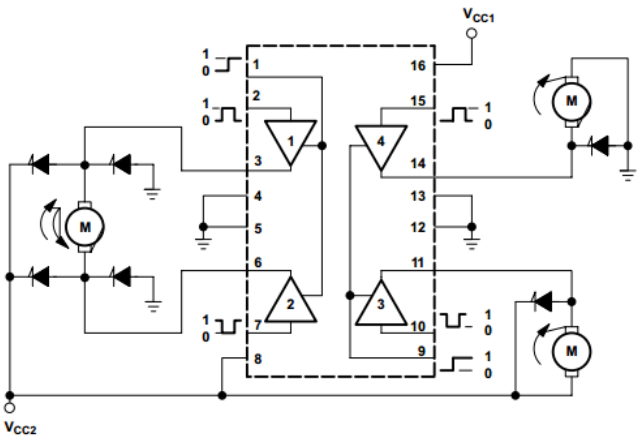


Figura 5.3: Diagrama funcional del integrado L293D

- Dos motores DC de $12V$ cuya velocidades son controladas por PWM (Pulse Width Modulation), uno de menor tamaño que permite el ingreso del aire caliente a la vivienda y el otro de mayor tamaño que permite el ingreso de aire a temperatura ambiente. Para mover el primero se usa el integrado L293D, mientras que para el segundo se usan el transistor de pequeña señal 2N3904 (en corte y saturación) con su Emisor conectado al Gateway del Thyristor BT151 como etapa de potencia.

- Un buzzer, el cual nos proporciona la alarma sonora.
- Una lámpara.
- Una relay, el cual es controlado por el PIC y proporciona de forma independiente alimentación a la fuente de calor, fuente que consiste en un arreglo de *Resistencias Cerámicas* y *Alambre Incandescente*.

Para la construcción de la maqueta se ha tenido en cuenta el flujo del aire tanto caliente como a temperatura ambiente, la simetría en el diseño y el aislamiento del sistema de control y comunicación con la fuente de energía principal y el ambiente cuya temperatura se desea controlar.
Se muestran a continuación imágenes de la maqueta con la identificación de las partes.

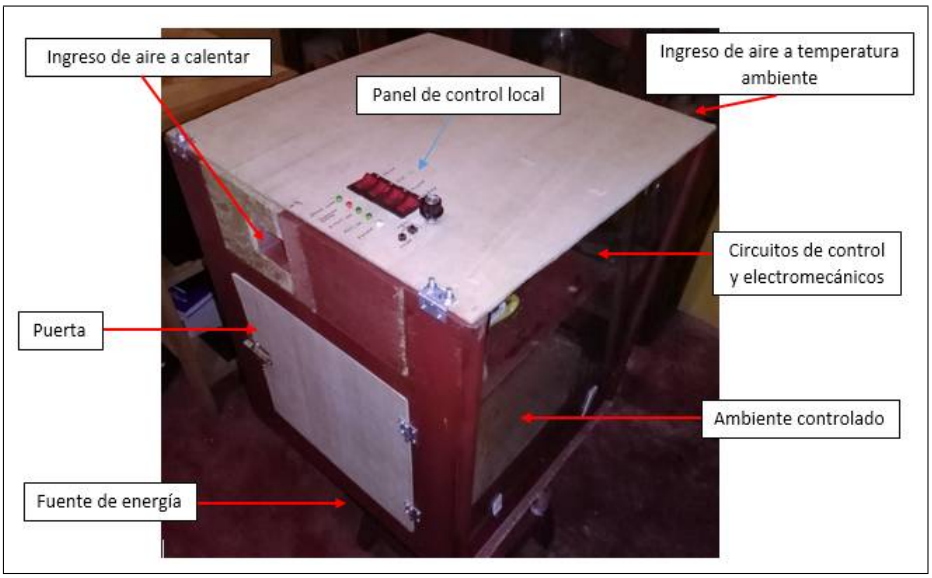


Figura 5.4: Vista en perspectiva de la maqueta

Figura 5.5: Panel de control local

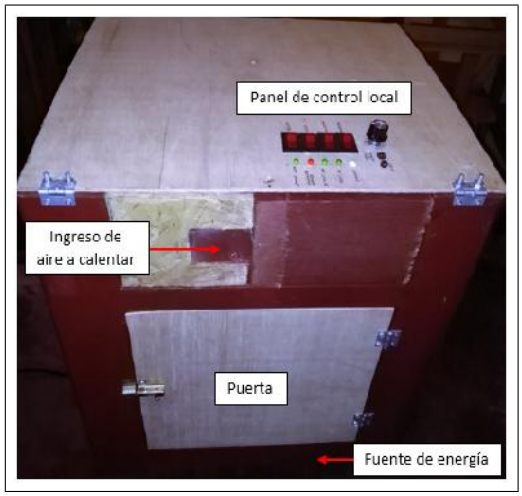


Figura 5.6: Vista frontal de la maqueta

El circuito de control y las partes electromecánicas se encuentran en la parte superior de la maqueta y esta a su vez está dividida en dos partes, aislando las *Resistencias Cerámicas*, el *Alambre Incandescente* y el aire caliente del sistema de control.

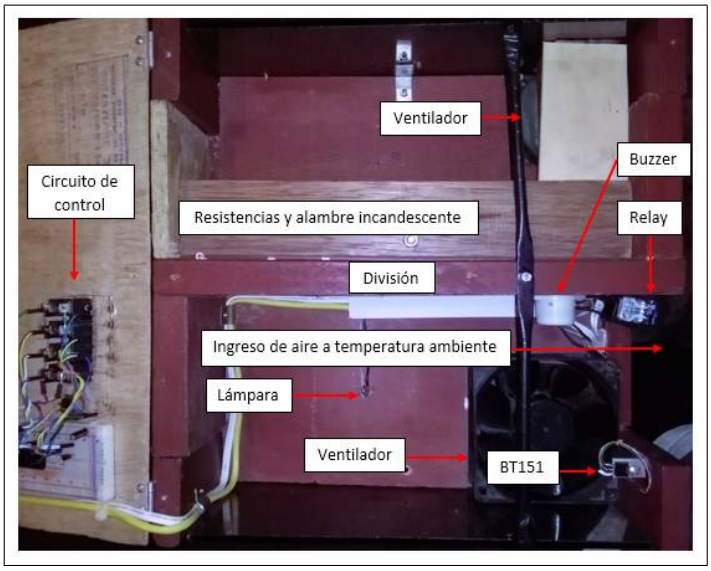


Figura 5.7: Vista superior interna de la maqueta

El circuito de control hace uso de tres voltajes; se esta tomando $12V$ de la fuente de alimentación el cual llega a la parte superior para alimentar el L293D al que están conectados los ventiladores, relay y buzzer, este voltaje

es convertido a $5V$ por un L7805CV para alimentar el PIC, LEDs, Switches de control, modulo bluetooth, sensor y switch de puerta y a $1{,}25V$ por un LM317T para alimentar el potenciómetro que da el voltaje (Temperatura) de referencia. Se usa un disipador de calor en el L7805CV.

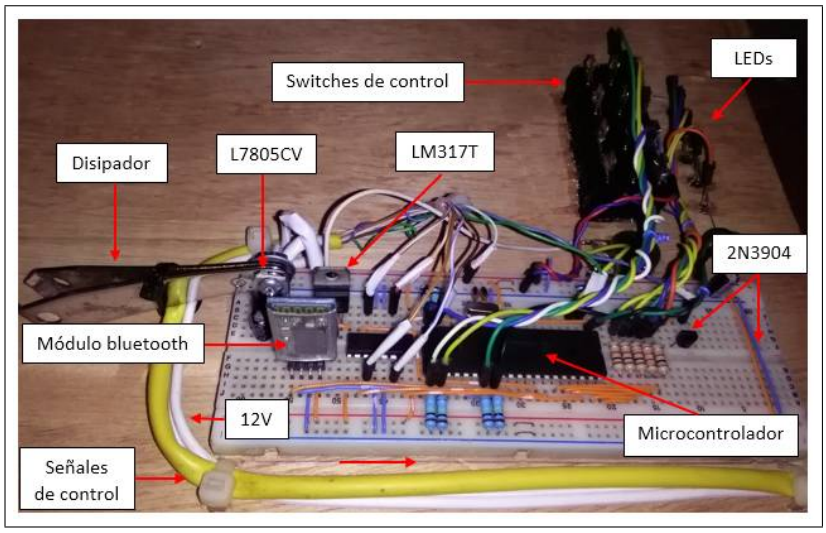


Figura 5.8: Circuito de control e indicadores

En la parte inferior se ubica la fuente de alimentación, la cual toma $220V$AC y nos entrega $12V$DC, este voltaje es distribuido independientemente al circuito de control y a través del relay a los elementos que calientan el aire, ambos en la parte superior.

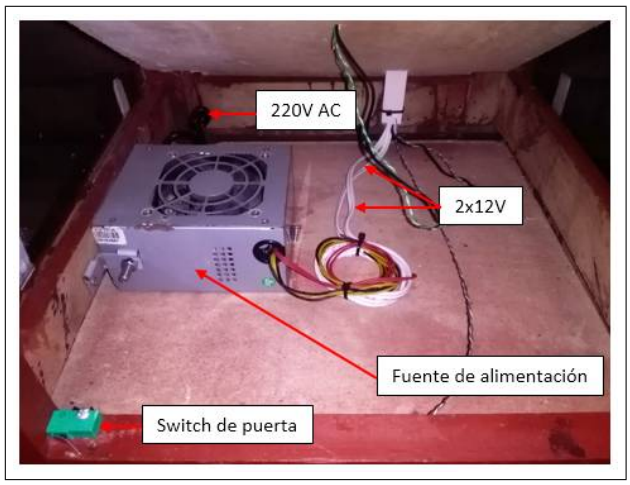


Figura 5.9: Vista de fuente de alimentación

En la parte interna encontramos el sensor de temperatura. Por fines demostrativos hemos colocado un potenciómetro que simule voltajes entregados por el sensor. La selección de uno u otro es a través de los conectores headers.

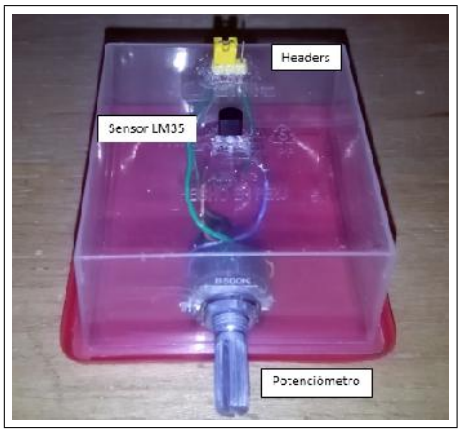


Figura 5.10: Vista del sensor LM35

5.2.3. Fundamentos del sistema de control de temperatura

En términos generales un sistema de control con realimentación consta de los siguientes bloques y señales en función del tiempo.

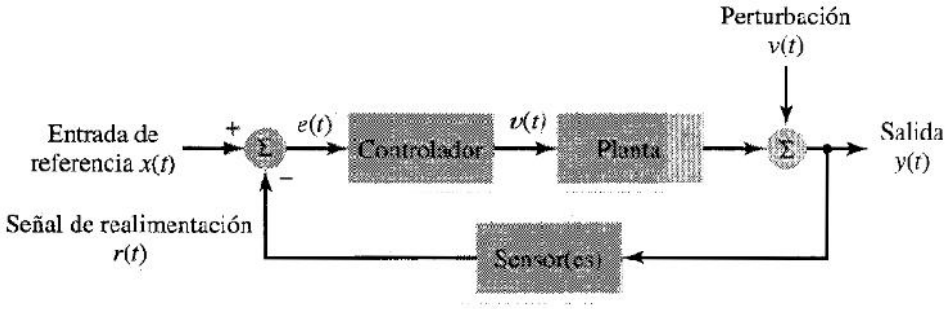


Figura 5.11: Diagrama de bloques de un sistema de control realimentado

El PIC engloba tanto el *Controlador* como la etapa de obtención de las señales $x(t)$ y $r(t)$ las cuales corresponden a los voltajes provenientes del potenciómetro de referencia y al sensor de temperatura respectivamente, entregando al final del procesamiento una señal de control $v(t)$ a los actuadores de la planta para obtener una temperatura deseada y(t), la cual puede variar por ejemplo por cambios en la temperatura ambiente o por la misma naturaleza del sistema de control; esto que causa las variaciones en la temperatura

deseada a la salida serían las perturbaciones $v(t)$.

Empezaremos nuestro análisis del sistema estudiando el comportamiento de la planta 'sin controlador'. Nuestra planta en términos físicos costa de una maqueta, que representa nuestra vivienda, la cual como se vio en la sección anterior hace uso de dos ventiladores para el ingreso del aire caliente o del aire a temperatura ambiente, de tal forma que se pueda cambiar la temperatura interior entre 8.ºC y 10.ºC por encima de la temperatura ambiente. Pero estos ventiladores no entrarían en funcionamiento si no hay una señal de arranque (voltaje), llamaremos a esta señal $v'(t)$, y si definimos el arranque de alguno de los motores en $t = t_0$ entonces $v'(t)$ puede ser definida como:

$$v'(t) = \begin{cases} 0V & \text{si } t < t_0 \\ 12V & \text{si } t \geq t_0 \end{cases} \tag{5.1}$$

Si esta es la señal de estrada para el sistema sin control, ¿cual es la señal de salida? o más claramente la temperatura de la maqueta. Para esto debemos conocer el comportamiento de la planta al estar presente $v'(t)$, obtendremos de forma experimental la salida 'sin controlar' $y'(t)$, para lo cual haremos mediciones en intervalos regulares (muestreo).
La siguiente tabla muestra las mediciones obtenidas experimentalmente para $y'(t)$ cuando $v'(t) = 12V$.

Incremento T Sensor (°C)	Incremento T Term. (°C)	Decremento T Term. (°C)	Tiempo (min)
22	23,9	31,4	1
22	24	28,7	2
24	24,2	27,7	3
24	24,8	26,5	4
26	25,5	26	5
26	26,2	25,7	6
28	26,9	25,4	7
28	27,6	25,3	8
30	28,1	25,1	9
30	28,7	25,1	10
30	29,3	24,9	11
30	29,7	24,8	12
32	30,2	24,8	13
32	30,6	24,6	14
32	31,1	24,6	15
32	31,4	24,6	16
32	31,8	24,5	17
34	32,1	24,4	18
34	32,4	24,4	19
34	32,8	24,4	20
0,9693361	0,9932609	-0,79436	Pearson "r"

Cuadro 5.1: Mediciones de temperatura sin sistema de control

Las mediciones son las obtenidas tanto por un termómetro digital (T Term) como por el ADC de microcontrolador con la lectura del sensor LM32 (T Sensor), también se muestra las mediciones en el decremento de la temperatura, durante todo el tiempo de medición uno de los ventiladores estuvo funcionando es decir $v'(t) = 12V$.
Las mediciones muestran una tendencia de tipo lineal entre el tiempo y el cambio de temperatura, pero ¿qué tan lineal se puede considerar?. Para estudiar esto haremos uso de un concepto estadístico llamado *Coeficiente de Pearson.*

Para estudiar la fuerza de asociación lineal entre dos variables; utilizaremos el coeficiente de correlación lineal de Pearson. La asociación será más elevada cuando más cerca de 1 esté el coeficiente, y menor cuando más cerca este de 0[1]. Existen fórmulas para calcular manualmente este coeficiente pero también existen herramientas informáticas, haciendo uso de Excel se obtiene este coeficiente para las tres columnas mostradas en la parte inferior del cuadro. Se puede apreciar la linealidad 'casi perfecta' ya que r es próximo a 1. Ya con la seguridad de que el comportamiento de $y'(t)$ es en esencia lineal podemos hacer uso del modelo de *Regreción Lineal* para $r = 0,993$, con lo cual obtenemos para $y'(t)$:

$$y'(t) = \begin{cases} 23,238.^{o}\text{C} & \text{si } t < t_0 \\ 0,5073t + 23,238.^{o}\text{C} & \text{si } t \geq t_0 \end{cases}$$

Se muestra a continuación el diagrama de dispersión para las mediciones hechas experimentalmente.

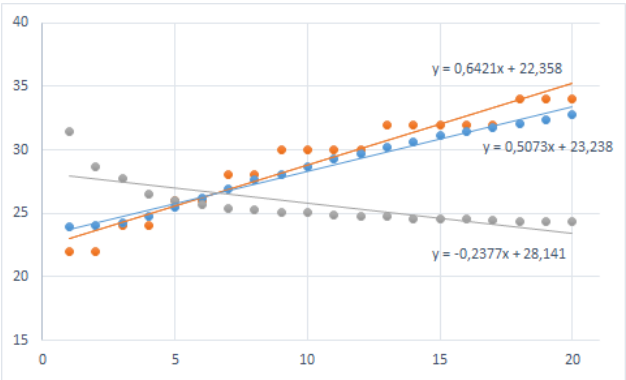


Figura 5.12: Diagrama de dispersión Temperatura (.ºC) Vs Tiempo (min)

En esencia lo que nos dice la ecuación anterior es que el aumento de la temperatura es lineal cuando encendemos el ventilador a plena potencia

[1]P. Juez y F. Diez, *Probabilidad y Estadística en medicina*, España: Ediciones Díaz de Santos S.A., 1997 , pp. 98

($v'(t) = 12V$) partiendo de la temperatura ambiente. Siendo este un sistema físico con energía limitada $y'(t)$ no puede aumentar de forma ilimitada, para algún valor suficientemente grande de t, $y'(t)$ ha de alcanzar su máximo valor, el cual experimentalmente podría considerarse el ultimo valor de la tabla para T Term. Por lo que la ecuación se redefine como:

$$y'(t) = \begin{cases} 23,238.^{o}\text{C} & \text{si } t < t_0 \\ 0,5073t + 23,238.^{o}\text{C} & \text{si } t_0 \leq t < t_\alpha \\ 32,8.^{o}\text{C} & \text{si } t > t_\alpha; \quad t_\alpha \approx 1200s \end{cases} \tag{5.2}$$

Si bien la ecuación (5,2) varía en función de la temperatura ambiente donde se tomen las mediciones y la fuente de calor usada (asumiendo el diseño geométrico de la maqueta constante), estas variaciones no serían en realidad importantes, lo importante de este estudio experimental es que podemos suponer una relación lineal entre el aumento de la temperatura y el tiempo, funcionando los ventiladores a plena potencia. Y si trabajamos en $t_0 \leq t < t_\alpha$ entonces podemos plantear la ecuación general de la temperatura de nuestra maqueta en función del tiempo, con los ventiladores a plena potencia, $v'(t) = 12V$, como:

$$y'(t) = y'(t_0) + mt \quad \text{si } t_0 \leq t < t_\alpha \tag{5.3}$$

Siendo $y'(t_0)$ la temperatura inicial del sistema y m la *velocidad de crecimiento* o pendiente de la recta de regresión lineal.

Los ventiladores cumplen una relación lineal entre el flujo y su velocidad de rotación, es decir:

$$\frac{\omega_1}{q_1} = \frac{\omega_2}{q_2} \quad \text{lo que es lo mismo} \quad q(t) = k_1\omega(t) \tag{5.4}$$

Es decir el flujo $q(t)$ de aire caliente o frio es proporcional a su velocidad de rotación $\omega(t)$. Este flujo de calor es lo que hace variar la temperatura en el interior de la maqueta, por tanto para variar la temperatura (controlar la temperatura) en el interior de la maqueta podemos hacer cambios en $\omega(t)$.

La *ley de enfriamiento de Newton* nos permite explicar la transferencia de calor por convección.

$$\frac{dQ(t)}{dt} = h(t)As(y(t) - y_\infty) \tag{5.5}$$

Donde $dQ(t)/dt$ es la velocidad de cambio durante la transferencia de calor, $h(t)$ es el coeficiente de convección, el cual depende de múltiples parámetros, entre ellos la velocidad del fluido y puesto que este pensamos variarlo en

el tiempo entonces $h = h(t)$, As es la superficie de contacto con el fluido, el cual es un aspecto geométrico que no variaría, $y(t)$ es la temperatura en la superficie de contacto o en el interior de la maqueta, notar que estamos colocando $y(t)$ y no $y'(t)$ ya que consideramos esta relación válida con o sin controlador, y_∞ es la temperatura del fluido lejos de la maqueta, la cual no es otra que la temperatura ambiente. Es en este punto donde se puede apreciar que la velocidad de cambio de $Q(t)$ expresada en (5,5) está relacionada con $q(t)$, para variaciones pequeñas en las temperaturas he intervalos cortos de tiempo podemos considerar:

$$\frac{dQ(t)}{dt} = k_2 q(t), \text{usando}(5{,}5) \Rightarrow \frac{dQ(t)}{dt} = k_1 k_2 \omega(t)$$

Remplazando este resultado en (5,6)

$$h(t)As(y(t) - y_\infty) = k_1 k_2 \omega(t)$$

Reordenando los términos tendremos.

$$y(t) = y_\infty + \frac{k_1 k_2}{As}\frac{\omega(t)}{h(t)}$$

Esta sería la ecuación si variamos la velocidad de rotación de los ventiladores, puesto que $h(t)$ depende de la velocidad del flujo y este de $\omega(t)$, entonces $h(t) = f(\omega(t))$ y tomando en cuenta que la temperatura lejos de la maqueta es igual a la temperatura ambiente y esta a su vez es la temperatura al inicio de las mediciones en $t = t_0$, entonces.

$$y(t) = y(t_0) + \frac{k_1 k_2}{As} F(\omega(t)), \quad \text{donde} \quad F(\omega(t)) = \frac{\omega(t)}{f(\omega(t))} \tag{5.6}$$

La ecuación (5,6) nos proporciona información de la temperatura obtenida cuando se entrega una señal de control $v(t)$ para variar $\omega(t)$. Siendo $h(t)$ un parámetro que estimamos no varíe de forma considerable, entonces $h(t) \approx k_3$, luego.

$$y(t) \approx y(t_0) + \frac{k_1 k_2}{Ask_3}\omega(t), \quad h(t) \approx k_3 \tag{5.7}$$

Para poder controlar $\omega(t)$, es decir la velocidad de rotación, podemos variar el voltaje o usar PWM (Pulse Width Modulation), una ventaja de usar PWM y no simplemente bajar el voltaje es que se mantiene el torque constante en los motores. Se define el *Ciclo de Trabajo* como:

$$CT(t) = \frac{\tau(t)}{T}$$

Siendo $\tau(t)$ el ancho de pulso, donde se aplica torque a los motores, T es el periodo de la señal.

En la aplicación se usa el PIC para realizar el PWM, donde $CT(t)$ y T son valores colocados en registros del microcontrolador, se fija T y se varía $CT(t)$. Puesto que $\omega(t)$ depende del torque aplicado de forma directamente proporcional, entonces $\omega(t) = k_4 CT(t)$. Lo que nos lleva a:

$$y(t) \approx y(t_0) + kCT(t), \quad k = \frac{k_1 k_2 k_4}{Ask_3}, h(t) \approx k_3 \tag{5.8}$$

Como se puede apreciar $CT(t)$ es la señal se control hacia los actuadores de la planta y recordando el diagrama de bloques presentado al inicio entonces $v(t) = CT(t)$. Para efectuar los cálculos siguientes tomaremos (5,8) como una ecuación y no como una aproximación.

Tomando en cuenta las ecuaciones (5,3), (5,7), lo analizado hasta el momento, así como las 'iteraciones' que se desean realizar para modificar $h(t)$. Entonces se muestra la siguiente gráfica que resume lo expuesto y permite definir nuevos parámetros, necesarios para hacer los ajusten en el $CT(t)$.

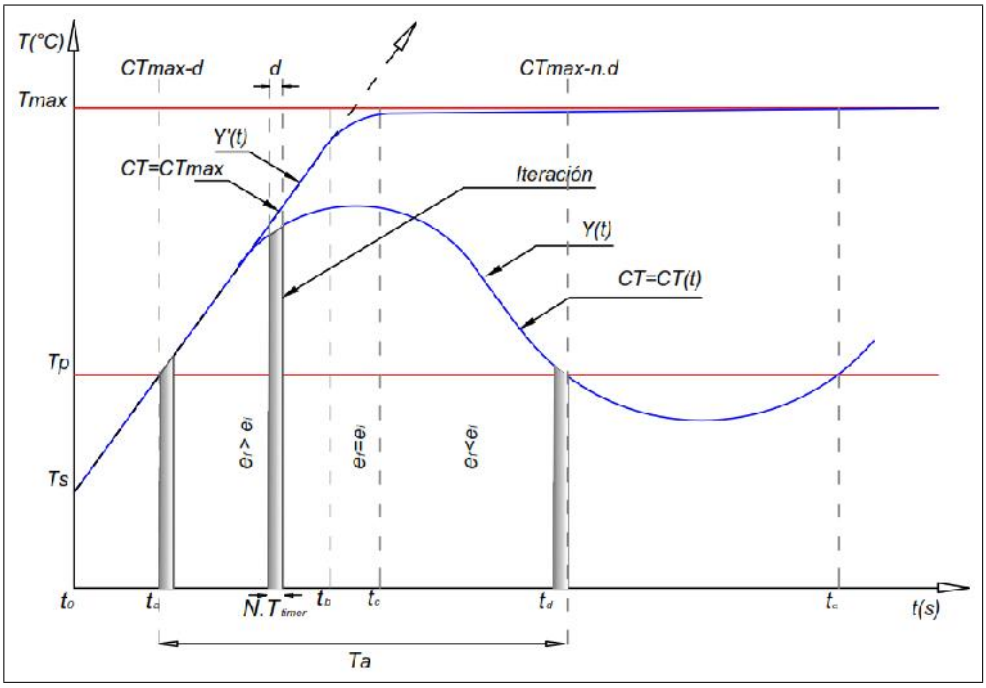


Figura 5.13: Temperaturas Vs Tiempo e iteraciones

Donde:

- CT_{max} es el máximo valor asignable a $CT(t)$.
- T_{timer} es el tiempo proporcionado por el temporizador *Timer 1*.

- $N.T_{timer}$ es el tiempo entre cada 'iteración', donde $N \in \mathbf{N}$.
- d es el decremento hecho a $CT(t)$ en una iteración.
- e_i y e_f, son los errores iniciales y finales en cada iteración respectivamente.
- T_p es la temperatura seteada por el potenciómetro de referencia, equivalente a $x(t)$ del diagrama de bloques.
- $T_s = T_s(t)$ es la temperatura detectada por el sensor de temperatura equivalente a $r(t)$ del diagrama de bloques.[2].
- T_{max} es la temperatura máxima alcanzable por el sistema.
- T_a es el *tiempo de ajuste*, el tiempo requerido para que las sucesivas iteraciones 'lleven' el valor de $y(t)$ a T_p.
- $y'(t)$ es la ecuación (5,3)
- $y(t)$ es la ecuación (5,8)

Como se puede apreciar el algoritmo consiste 'llevar' $T_s(t)$ al valor de referencia T_p usando el máximo valor posible de $CT(t)$, el cual es CT_{max}; una vez alcanzado este valor, se comienzan iteraciones que decrementan $CT(t)$ en d unidades cada $N.T_{timer}$ segundos. Apenas supere T_s a T_p, se ejecuta la primera iteración, siendo el número total de iteraciones n y el tiempo total empleado T_a.

De las definiciones dadas se deduce la siguiente ecuación.

$$T_a = (n-1)N.T_{timer}, \quad n, N \in \mathbf{N} \tag{5.9}$$

La ecuación (5,8) es válida para cualquier t y por tanto también es válida durante cualquier iteración. Si usamos esta ecuación para por ejemplo la h -ésima iteración, entonces.

$$y(h) = y(h-1) + kCT(h), \quad \text{para todo } h \in \mathbf{N} \tag{5.10}$$

[2]Cabe apreciar que T_p y T_s son considerados en la gráfica como temperaturas, en realidad son voltajes entregados al ADC de PIC, consideramos aquí ya transformados estos voltajes a sus equivalentes en temperatura.

Las iteraciones realizadas durante el proceso de ajuste son:

$$\begin{aligned} CT(1) &= CT_{max} - d \\ CT(2) &= CT_{max} - 2d \\ &\vdots \\ CT(m) &= CT_{max} - md \Rightarrow d = \frac{CT_{max} - CT(m)}{m} \qquad (5.11) \\ &\vdots \\ CT(n) &= CT_{max} - nd \Rightarrow d = \frac{CT_{max} - CT(n)}{n} \qquad (5.12) \end{aligned}$$

Siendo (5,11) la m-ésima iteración y representa cualquier ajuste durante todo el proceso, mientras que (5,12) es únicamente la última iteración, fijando un n, se tiene que $1 \leq m \leq n$.

Despejando $CT(h)$ de (5,10)

$$CT(h) = \frac{y(h) - y(h-1)}{k}, \quad \text{haciendo } h = m$$

$$\Rightarrow CT(m) = \frac{y(m) - y(m-1)}{k} \qquad (5.13)$$

Remplazando (5,13) en (5,11)

$$d = \frac{CT_{max}}{m} - \frac{y(m) - y(m-1)}{m.k} \qquad (5.14)$$

La ecuación (5,8) puede expresarse como:

$$k = \frac{\Delta y(t)}{CT(t)}$$

Siendo $\Delta y(t)$ la variación de la temperatura para un $CT(t)$ dado, puesto que k es constante, si $CT(t) = CT_{max} \Rightarrow \Delta y(t) = \Delta y_{max}$. Luego se tiene para k.

$$k = \frac{\Delta y_{max}}{CT_{max}} \qquad (5.15)$$

Remplazando (5,15) en (5,14)

$$d = \frac{CT_{max}}{m} - \frac{CT_{max}}{m\Delta y_{max}}[y(m) - y(m-1)], \quad 1 \leq m \leq n \qquad (5.16)$$

La ecuación (5,16) nos proporciona el decremento en cada iteración, durante todo el proceso de ajuste, sin embargo $y(m)$ y $y(m-1)$ son los valores a la salida de la planta (recordar el diagrama de bloques), y estos valores no los tenemos, el valor que si tenemos es el entregado por el sensor, que en

la Figura 5.13 es $T_s(t)$ y en el diagrama de bloques es $r(t)$. Luego debemos encontrar una relación entre $y(t)$ y $r(t)$. Esta relación puede tener muchas formas, sin embargo consideraremos esta relación lineal ya que usaremos un sensor lineal, es decir.

$$y(t) = pr(t), \quad p \text{ constante} \tag{5.17}$$

Aplicando (5,17) en (5,16).

$$d = \frac{CT_{max}}{m} - \frac{CT_{max}}{mp\Delta r_{max}}[pr(m) - pr(m-1)] \quad \Rightarrow$$

$$d = \frac{CT_{max}}{m} - \frac{CT_{max}}{m\Delta r_{max}}[r(m) - r(m-1)], \quad 1 \leq m \leq n \tag{5.18}$$

Se define el error $e(t)$ como la diferencia entre la entrada de referencia $x(t)$ y la lectura del sensor $r(t)$, luego

$$e(t) = x(t) - r(t)$$

Aplicando a la m -ésima iteración

$$e(m) = x(m) - r(m)$$

Aplicando a la $(m-1)$ -ésima iteración

$$e(m-1) = x(m-1) - r(m-1)$$

Restando ambos resultados y teniendo en cuenta que $x(m) = x(m-1)$ ya que el punto de referencia permanece constante.

$$r(n) - r(n-1) = -(e(n) - e(n-1))$$

Remplazando este resultado en la ecuación (5,18), se tiene finalmente que:

$$d = \frac{CT_{max}}{m} + \frac{CT_{max}}{m\Delta r_{max}}[e(m) - e(m-1)], \quad 1 \leq m \leq n \tag{5.19}$$

De similar forma se obtiene d en función de n.

$$d = \frac{CT_{max}}{n} + \frac{CT_{max}}{n\Delta r_{max}}[e(n) - e(n-1)], \qquad n \text{ fijo} \tag{5.20}$$

Si bien las ecuaciones (5,19) y (5,20) son en apariencia iguales, esto no es así, ya que en (5,19) m es un número que varia en cada iteración y $e(m) - e(m-1)$ es conocido en cada momento (son las lecturas de sensor antes y despues de cada iteración) y por tanto queda bien definido d;

mientras que en (5,20), n es un número fijo y si bien podemos escoger un n 'adecuado', no nos es posible conocer a priori el valor de los errores finales, luego (5,20) no se puede usar tal y como está planteada.

Si bien (5,19) esta bien definida en cada iteración, la dificultad de su uso radica en el aspecto computacional, ya que se usará un microcontrolador y se programará en assembler, las divisiones en assembler demandas muchos recursos, siendo el fin de esta aplicación la demostración de su buena interacción con CJAR entonces es la comunicación serial y los algoritmos asociados a esta la prioridad. Haciendo uso de la siguiente aproximación para calcular las iteraciones.

$$e(n) - e(n-1) \approx e(m) - e(m-1)$$

Remplazando esto en (5,20).

$$d \approx \frac{CT_{max}}{n} + \frac{CT_{max}}{n\Delta r_{max}}[e(m) - e(m-1)], \quad 1 \leq m \leq n, n \text{ fijo} \tag{5.21}$$

Que será la ecuación que se usará para calcular el decremento en las iteraciones. Las ecuaciones (5,21) y (5,9) serán implementadas en el programa del microcontrolador, en la etapa de control de temperatura.

5.2.4. Software y lógica de programación

Dentro de las consideraciones en el diseño de la aplicación, se tiene la *reutilización de código*, es decir la posibilidad de usar el código generado en otras aplicaciones sin mayores modificaciones. Si bien el programa en su totalidad no se puede aplicar a cualquier aplicación, si hay dentro del programa una estructura básica o 'columna vertebral' completamente reutilizable.

En realidad el mayor esfuerzo en la programación del código radicó en crear esta estructura básica, lo suficientemente *estable y adaptable* para sostener aplicaciones tan básicas como el CBP (Circuito Básico de Pruebas) o con relativa complejidad como la aplicación en domótica que se está presentando. ¿Pero cómo crear una estructura estable y adaptable?[3]. Conseguiremos que el programa sea *estable* si no 'forzamos' el hardware y los algoritmos implementados son consistentes, es decir que el resultado de la ejecución de un algoritmo no quede al azar si no bien definido, de igual forma programamos tomando en cuenta los límites que el fabricante impone al hardware, si bien esto último es casi obvio, no lo es su ejecución, para esto no sólo leemos el datasheet del fabricante sino que también forzamos con el código

[3]Es en este punto donde el autor de la presente tesis recuerda su responsabilidad en las opiniones vertidas. Durante todo el trabajo se ha respetado la referencia a los autores o a la fuente de información si esto correspondía

deliberadamente el hardware para ver cómo responde el programa, de tal forma que en la etapa de diseño se detecten errores presentes y potenciales de las aplicaciones, tener claras las 'reglas del juego' y transmitir estar reglas al diseñador o implementador de las aplicaciones con CJAR, minimiza los posibles errores en sus aplicaciones y junto a algoritmos consistentes hace estable esta estructura básica creada.

Para que el código o mejor dicho la estructura básica de este sea *adactable* sin que esto implique aminorar su alcance, usaremos una 'simetría creciente' en su diseño. Para comprender este concepto, haremos las siguientes definiciones.

- R_k: Representa un requerimiento de 'categoría' k, por ejemplo el requerimiento de atención a interrupciones, o de envío de caracteres por el USART.
- R_{ki}: Representa un elemento i del requerimiento de categoría k, por ejemplo interrupción por el timer o el carácter a ser enviado por el USART.
- M_k: Es el multiplexor lógico de los elementos de categoría k.
- P_k: Es el algoritmo que procesa los elementos de categoría k, recibe los datos de M_k.
- D_k: Es el desmultiplexor lógico, que toma los resultados de P_k y los deriva a las correspondientes salidas.
- S_k: Representa el conjunto de salidas a los requerimientos de categoría k.
- S_{ki}: Representa la salida al elemento i de categoría k.

Un programa con 'simetría creciente' tendría la siguiente estructura lógica:

$$
\begin{array}{cccccccccccccc}
R_{11} & R_{12} & \dots & R_{1p} & R_{21} & R_{22} & \dots & R_{2q} & \dots\dots & R_{n1} & R_{n2} & \dots & R_{nm} \\
 & R_1 & & & & R_2 & & & \dots\dots & & R_n & & \\
 & \downarrow & & & & \downarrow & & & & & \downarrow & & \\
M_1 + P_1 + D_1 & & & & M_2 + P_2 + D_2 & & & & \dots\dots & M_n + P_n + D_n & & & \\
 & \downarrow & & & & \downarrow & & & & & \downarrow & & \\
 & S_1 & & & & S_2 & & & \dots\dots & & S_n & & \\
S_{11} & S_{12} & \dots & S_{1p} & S_{21} & S_{22} & \dots & S_{2q} & \dots\dots & S_{n1} & S_{n2} & \dots & S_{nm}
\end{array}
$$

Como se puede apreciar la estructura es ordenada y simétrica, es decir puedo por ejemplo intercambiar el elemento R_{12} por $R_{1(p-1)}$ sin que se altere

la estructura del programa, o intercambiar todos los elementos de la columna donde encuentra R_2 por los elementos de la columna donde se encuentre R_n sin tampoco alterar la estructura lógica del programa. Intercambiar R_{12} por $R_{1(p-1)}$ implica que el algoritmo $P1$ que los procesa sea el mismo incluso para cualquier otro valor de R_{1k}, la posibilidad de intercambiar los elementos de ingreso sin modificar el algoritmo e incluso de intercambiar los algoritmos del programa sin modificar la estructura total es la simetría buscada.

Esta estructura lógica es fácilmente creciente por su misma simetría, ya que podemos variar los valores de $p, q...m$ y n según nuestras necesidades, esto sólo estaría limitado por las capacidades del hardware usado, en este caso del microcontrolador, variar los valores de $p, q...m$ y n en la práctica es ingresar nuevos datos e implementar nuevos algoritmos manteniendo la estructura lógica total del programa. Por lo expuesto es que el programa escrito en assembler se ha implementado lo más simétrico posible e independiente de la variación de los datos si estos son de la misma categoría.

Pero lograr esta simetría creciente tiene sus costos, implica buscar algoritmos eficientes y generales, implica costes computacionales. Para que el algoritmo o el programa en su totalidad sea considerado 'simétrico creciente', deben hacerse los intercambios ya mencionados y comprobar que el programa sigue funcionando con el mismo nivel de confianza. La estructura mostrada es una idealización pero como ya se mencionó esto es alcanzable en la medida de lo que el hardware lo permita.

Ya aclarado el enfoque usado para la programación del microcontrolador, se entrará de lleno al programa en sí y a los algoritmos implementados. No se expondrá detalles de cómo programar el PIC, sobre esto ya hay mucha bibliografía especializada, se centrará en explicar los algoritmos implementados y la interacción del programa con CJAR.

De los algoritmos a implementar ¿cuál de ellos es el más importante?, esta pregunta no es completa ya que debe ser planteada de la siguiente forma: ¿Cuál de ellos es el más importante para CJAR? y la respuesta naturalmente es **la comunicación** y esta comunicación se da a través del puerto USART del PIC y el puerto serie del módulo (por lo menos en la primera versión de CJAR Host).

La comunicación ha de ser bidireccional, y no ha de estar limitada a sólo caracteres, debe poder enviarse y recibirse cadenas de caracteres (palabras) y el número de estas palabras debe ser el mayor posible sin la necesidad de que tenga que existir alguna relación entre estas 'palabras'. Por ejemplo no se debe limitar a solo cadenas de caracteres de tres caracteres.

El envío de un carácter en el PIC es relativamente sencillo, basta con 'consultar' si el buffer está ocupado y si no lo está se coloca el carácter a enviar en el registro **TRXREG**, se muestra el fragmento de código que hace esto.

```
    ;Subrutina de envio de un caracter
Tx_Caracter     bsf     STATUS,RP0      ;RP0=0 RP1=0, nos ubicamos en el banco 1
bucle_tx_2      btfss   TXSTA,TRMT      ;Chequea buffer de Transmición
                goto    bucle_tx_2      ;Espera si esta ocupado
                bcf     STATUS,RP0      ;RP0=0 RP1=0, nos ubicamos en el banco 0
                movf    Dato_Tx_Rx,w    ;Envio del caracter guardado
                movwf   TXREG
                return                  ;Retorno de subrutina
```

Luego enviar una cadena de caracteres, es ejecutar esta subrutina de forma repetitiva, sólo se tendría que tener un mecanismo que indique el fin del texto a enviar, este mecanismo en el programa es un carácter de fin de texto, en este caso es &. Existe una subrutina que envía cadenas de caracteres usando de forma repetitiva el código anterior. Se muestra el detalle del carácter de fin de texto.

```
text_tx_4_      bsf     REG_AVISO,2     ;Indicamos el envio de texto
                movf    Apunt_Tx,w
                addwf   PCL,1
                DT"Variable ON","&"
```

Enviar cadenas de caracteres no implica mucha dificultad, sin embargo hay que implementar este procedimiento de tal forma que el algoritmo busque ser 'simétrico creciente', hay que considerar además que muchas de estas palabras enviadas son información que CJAR a de transmitir a la interfaz del usuario y se ha de tener de alguna forma una confirmación de que el mensaje llegó correctamente y si esto no fuese el caso enviar nuevamente la palabra. Esto se logra teniendo un registro de lo enviado y esperando algún mensaje de confirmación del envío o error durante el envío. Para esto de define el registro de usuario de 8 bist **REG_AVISO**, este registro está asociado a los distintos textos a enviar, si el valor de un bit es 0 entonces el texto asociado a este bit fue enviado y se confirmó la recepción por CJAR y el valor del bit es 1 en el caso contrario, la estructura del registro es el siguiente.

REG_AVISO	7	6	5	4	3	2	1	0

0: Texto enviado, 1: Texto no enviado.

Posición		Texto asociado
REG_AVISO[0]	→	Puerta abierta.
REG_AVISO[1]	→	Comando error.
REG_AVISO[2]	→	Variable ON.
REG_AVISO[3]	→	Variable OFF.
REG_AVISO[4]	→	Tem: XYZ.ºC, XYZ es un número.
REG_AVISO[5]	→	Comando ejecutado.

Pero ¿cómo es posible la confirmación de envío?, esto es posible ya que como se vio en la descripción de CJAR, este responde con *ok*& si el envío es satisfactorio y con *error*& si no lo es, estas palabras con 'detectadas' por el programa del PIC ya que están dentro de sus textos de ingreso predefinidos. Si se detecta un *ok*&, simplemente se limpia el registro REG_AVISO y si no se envían todos los textos donde su bit en REG_AVISO sea 1.

Para comprender el algoritmo implementado para la recepción de cadenas de caracteres, primero veremos cómo detectar un carácter. Es posible saber si un carácter ha llegado por el puerto serie leyendo de forma periódica el estado de la bandera **RCIF** del registro **PIR1**, si esta está en 1 entonces significa que hay un carácter a leer desde el registro **RCREG**. Pero esta forma de obtener un carácter es en cierta medida ineficiente, ya que no sabremos si un carácter llegó por el puerto USART hasta que el programa ejecute nuevamente el bloque de código de detección, se corre el riesgo de perder el carácter por la llegada de otros. La solución a esto se logra usando *interrupciones*, la interrupción es un mecanismo por el cual el PIC deja de ejecutar el normal funcionamiento secuencial del programa y salta a las líneas de código que se definen para atender la interrupción. Se usará la interrupción por recepción en el USART para procesar este carácter y que no se pierda.

```
;Interrupciones
Interrupcion    movwf   WREG_Temp       ;Salvamos contexto (WREG y STATUS)
                swapf   STATUS,W
                clrf    STATUS
                movwf   STATUS_Temp
                movf    PCLATH,W
                movwf   PCLATH_Temp
                clrf    PCLATH


                btfsc   PIR1,RCIF       ;¿La interrupción es causada por Rx de USART?
                goto    Inte_USART
                btfsc   PIR1,TMR1IF     ;¿La interrupción es causada por Timer 1?
                goto    Inte_Timer1
                goto    Otra_Inte       ;Interrupción causada por otra fuente

    ;Atención a la interrupción por Rx USART
Inte_USART      bcf     PIR1,RCIF       ;Limpiamos bandera de interrupción causada por Rx de USART
                movlw   06h             ;Chequeo de errores
                andwf   RCSTA,W
                btfss   STATUS,Z        ;¿Fue algún error encontrado?
                goto    rx_Error        ;Se produjo errores
    ;Identificación del mensaje recibido
                bcf     STATUS,RP0      ;RP0=0 RP1=0, nos ubicamos en el banco 0
```

Esto soluciona la lectura de un carácter, pero ¿qué pasa si es una cadena de caracteres o peor aún es una cadena de caracteres que se debe identificar y también dar una respuesta en función de esta cadena identificada?, el problema radica en que si bien podemos definir las palabras (cadenas de texto) que se consideren comandos reconocibles no nos es posible definir que texto llegará o cuantos caracteres tendrá y en todo caso es deseable poder procesar cualquier texto que ingrese, consideraremos esta posibilidad, sólo pediremos al usuario un carácter de fin de texto y que el texto no sea de longitud 0.

Programar en assembler implica que no se puede recibir la palabra completa si no carácter por carácter Se muestra el algoritmo de reconocimiento de texto recibido.

$$
\begin{array}{llllllll}
t_i & \leftarrow & t_0 & \leftarrow & t_1 \ldots & t_h & \leftarrow & t_{h+1} & \\
\downarrow & & & & & & & & \\
\text{TEXT[0]} & \text{Si } 0 & \longrightarrow & a_0 & a_1 & \ldots & a_p & a_{p+1} & ¿t_i = a_i? \\
\downarrow \text{Si } 1 & & & & & & & & \\
\text{TEXT[1]} & \text{Si } 0 & \longrightarrow & b_0 & b_1 & \ldots & b_q & b_{q+1} & ¿t_i = b_i? \\
\downarrow \text{Si } 1 & & & & & & & & \\
\vdots & & & & & & & & \\
\text{TEXT[n]} & \text{Si } 0 & \longrightarrow & z_0 & z_1 & \ldots & z_m & z_{m+1} & ¿t_i = z_i? \\
\downarrow \text{Si } 1 & & & & & & & & \\
¿\text{TEXT[k]}? & & & & & & & &
\end{array}
$$

Donde:

- $a_0 a_1 \ldots a_p, \quad b_0 b_1 \ldots b_q, \quad \ldots\ldots z_0 z_1 \ldots z_m$: Son las cadenas de caracteres (palabras) reconocibles por el programa, donde no necesariamente $p = q \ldots = m$.
- a_{p+1},b_{q+1},$\ldots z_{m+1}$: Son los caracteres de fin de texto de las palabras predefinidas, y como este carácter a de ser igual para todas las palabras predefinidas entonces $a_{p+1} = b_{q+1} \ldots = z_{m+1} = \&$.
- $t_0 t_1 \ldots t_h$: Son los caracteres del texto recibido, no necesariamente tienen que llegan juntos, pero si llegar en el orden correcto.
- t_{h+1}: Es el carácter final de la cadena de texto recibido. Si el texto recibido quiere tener la posibilidad de ser reconocido como una de las palabras predefinidas entonces $t_{h+1} = \&$ y $h \geq 0$
- TEXT: Es un registro o conjunto de registros auxiliares de $(n+1)$ posiciones (bist) y asocia cada posición a una de las palabras predefinidas. TEXT[k]= 0 indica que la palabra definida en la posición k no se ha descartado, si TEXT[k]= 1 el texto asociado se ha descartado.

- *Apuntador_ Rx*: Es una variable auxiliar que guarda temporalmente la posición del carácter que se ha de comparar.
- $a_i, b_i, \ldots, z_i t_i$: Son los caracteres en la posición i, $\quad i = 0, 1, 2 \ldots$.

En el inicio TEXT[k]= 0 para todo k, $\quad 0 \leq k \leq n$, es decir todos los textos pre definidos pueden ser comparados, TEXT[k] hace referencia a la palabra predefinida en la posición k; también al inicio *Apuntador_ Rx*= 0, *Apuntador_ Rx*= i y TEXT[k]= 1 indica que el caracter a ser comparado esta en la posición i de la palabra predefinida de la posición k, $\quad i = Max\{p+1, q+1, \ldots, m+1\}$ siempre que $i \leq h+1$.

Durante la primera comparación y para facilitar la descripción asumiremos que $t_0 \neq \&$. Al llegar t_0, el programa consulta el estado de TEXT[0] y puesto que al inicio TEXT[0]= 0, entonces t_0 se compara con a_0 si son iguales se activa un indicador de que el carácter es válido o pertenece a una de las palabras predefinidas, luego pasa a consultar el estado de TEXT[1], si no son iguales el programa hace TEXT[0]= 1 y pasa a comparar el estado de TEXT[1], puesto que TEXT[1] es también 0, entonces comparará t_0 con b_0, nuevamente si al realizar la comparación los caracteres son iguales entonces el programa activa el indicador de carácter reconocido y pasa a consultar el estado de TEXT[2] y si por el contrario resultaron ser diferentes, el programa coloca TEXT[1]= 1 y pasa a consultar el estado de TEXT[2], esto se repite hasta el consultar el estado en el registro de la n -ésimo palabra predefinida, es decir consultar el estado de TEXT[n] y compara t_0 con z_0, al hacer esta última comparación y dejar TEXT[n] en 1 o 0 según corresponda, pasa a consultar si algún TEXT[k]= 1, si para todo k, TEXT[k]= 1 entonces el carácter no se reconoce, se activa un indicador de carácter no reconocido, se coloca TEXT[k]= 0 para todo k y se pone el valor de *Apuntador_ Rx* en 0, aunque en esta primera comparación *Apuntador_ Rx* ya era 0, esto significa se reiniciarán las comparaciones asumiendo el carácter que llegue después como t_0. Si por el contrario TEXT[k]= 0 para algún k entonces se incrementa el valor de *Apuntador_ Rx* en 1, es decir *Apuntador_ Rx*=*Apuntador_ Rx*+1 y se dejan tal y como estaban los valores de TEXT[k] para todo k. Pero hemos asumido que $t_0 \neq \&$, ¿pero que pasa si $t_0 = \&$?. Para explicar esto se debe agregar que en cada comparación realizada y descrita anteriormente cuando t_0 es igual a alguno de de los caracteres $a_0, b_0, \ldots z_0$, entonces además de lo ya expuesto, consulta el valor de *Apuntador_ Rx* y si este es 0 entonces lo asumirá como un carácter no válido y hará indicado anteriormente cuando el carácter no es válido o reconocido como de una palabra predefinida. Si fuese el caso de que *Apuntador_ Rx*> 0 entonces no estaríamos en la comparación del primer carácter, y aquí estamos hablando de la primera comparación.

Una ves finalizada esta primera comparación de caracteres, sólo pasaríamos a la siguiente comparación si resultó TEXT[k]= 0 para algún k, se

compararía los caracteres de posición 1 ya que *Apuntador_ Rx* es ahora 1, en esta segunda comparación se repetirá el proceso descrito anteriormente simplemente que ya se descartan las comparaciones donde TEXT[k]= 1, nuevamente si se tiene el carácter &, se medirá la longitud de la cadena recidiva y puesto que ahora ya no será 0, entonces la cadena recibida será considerada válida, se activará un indicativo de texto reconocido y se ejecutara una subrutina que atenderá las peticiones asociadas a este texto reconocido. Nuevamente se coloca TEXT[k]= 0 para todo k y de hace *Apuntador_ Rx*= 0. Es decir se deja todo preparado para la detección de nuevos caracteres.

El proceso descrito anteriormente se repite en la tercera, cuarta,. . . i -ésima comparación hasta que se reconozca alguno de los textos predefinidos o se descarte lo que se reciba, el valor de *Apuntador_ Rx* va aumentando en cada comparación.

El algoritmo mostrado permite al diseñador de aplicaciones recibir cualquier texto, en cualquier momento (usa interrupciones), y de cualquier longitud sin más que definir un carácter común de fin de texto. Tanto el envío como recepción de texto tienen la característica de poder enviar o recibir cualquier texto (cadena de caracteres).

Para nuestra aplicación, el indicativo de carácter reconocido es poner RB6 en 1 y RB7 en 0, el indicativo de que se recibió una cadena de caracteres correcta y reconocible como una palabra predefinida es poner RB6 en 0 y RB7 en 1, el indicativo de carácter no reconocido es poner RB6 en 1 y RB7 en 1. Se muestra un fragmento de código.

```
                ;Esto se hará cuando se reciba texto pero no una palabra "clave"
                bsf     PORTB,RB6       ;Indicativo de texto no reconocido
                bsf     PORTB,RB7
                clrf    Apunt_Rx        ;Borramos apuntador de caracter recibido
                clrf    Texto_Aux_1     ;Habilitamos la comparación con todos los textos
                clrf    Texto_Aux_2     ;Habilitamos la comparación con todos los textos
;(Rx_n:)        .
;(Rx_n:)        .
;(Rx_n:)        .
;(Rx_n:)        clrf    Texto_Aux_(((n_max+7-f(n_max))/8))
                goto    fin_Inte        ;Retorno de interrupción
```

En el fragmento de código mostrado se puede apreciar que TEXT[k] corresponde a *Texto_ Aux_ 1* unido a *Texto_ Aux_ 1*, esto es necesario ya que estos variables definidas son posiciones de memoria de 8 bits cada uno, usar estas dos variables nos permite manejar hasta 16 palabras predefinidas, se han implementado 13 palabras predefinidas.

Como se puede apreciar el algoritmo implementado busca ser de naturaleza 'simétrica creciente', esto se aprecia en su propia descripción, pero también se puede apreciar en el fragmento código mostrado, para aumentar el número de textos predefinidos (reconocer más textos recibidos) basta con remplazar en todas las líneas cuyos comentarios inician con (Rx_n:) n por el valor siguiente al máximo existente, en el código este valor es $n\_max = 13$. Se aprecia también en este fragmento de código la función f aplicada a n_max, esta función nos permite transmitir de una forma 'compacta' al desarrollador lo que se ha de escribir en cada línea de código. Para definir f primero recordaremos la definición de la función máximo entero.

Si $x \in \mathbf{R}$, su máximo entero se denota $[|\ x\ |]$, y se define como:

$$[|\ x\ |] = n \iff n \leq x < n+1, \quad n \in \mathbf{Z}$$

El máximo entero de un número real x, es el mayor de todos los números enteros menores o iguales a x. Dada la definición de máximo entero de un número real, entonces la función f en las lineas de código comentadas en el programa se define como:

$$f(n) = (n-1) - 8[|\ (n-1)/8\ |], \quad n \in \mathbf{N} \tag{5.22}$$

Para la aplicación de han implementado 13 textos predefinidos, que son los comandos que el microcontrolador reconoce y ejecuta ciertas instrucciones en función de estas, dos de los textos predefinidos son *ok*& y *error*& que son enviados automáticamente por CJAR Host cuando se confirma el envío de los datos o cuando se detecta errores en el envío respectivamente.

Los 11 textos restantes tienen el siguiente formato.

$$T_i xyz T_f$$

Donde:

- T_i: Puede tomar dos valores, $T_i = G$ si lo que se realiza es una consulta del estado de una variable y $T_i = S$ si lo que se quiere es conmutar el estado de una variable o ejecutar una acción.
- xyz: Son nemotecnicos que describen a que variable va dirigido el comando.
- T_i: Es el carácter de fin de texto, en el programa es $T_i = \&$.

La lista de comandos completos es:

1. *Sres*&: Reseteo del microcontrolador.
2. *Gsen*&: Lectura del sensor de temperatura.

3. *Gsel&*: Lectura del selector de temperatura (potenciómetro).
4. *Gloc&*: Consulta si el control es local (ON) o el control es remoto (OFF).
5. *Sloc&*: Conmuta el estado del control, conmuta entre local y remoto.
6. *Gtem&*: Consulta si se está ejecutando el control de temperatura (ON), (OFF) si no se está ejecutando.
7. *Stem&*: Conmuta entre ejecutar el control de temperatura y detener el control de temperatura.
8. *Gluz&*: Consulta si la luz está prendida (ON) o apagada (OFF).
9. *Sluz&*: Conmuta entre encender y apagar la luz.
10. *Gpue&*: Consulta si la puerta está abierta (ON) o cerrada (OFF).
11. *Spue&*: Conmuta entre activar o desactivar la detección de puerta abierta.

Diseñado ya el aspecto más importante del programa del microcontrolador que es la comunicación, se implementa el algoritmo de control de temperatura. Para esto tomaremos en cuenta lo expuesto en la sección anterior y específicamente las ecuaciones (5,21) y (5,9).

Tomando en cuenta las mediciones realizadas y plasmadas en el cuadro 5,1, podemos considerar que $\Delta r_{max} = 8.^{o}$C.

En el programa para cambiar el ciclo de trabajo colocamos valores en los registros **CCPR1L** y **CCPR2L**, en CCPR1L si queremos variar ciclo de trabajo del ventilador que hace ingresar el aire caliente y en CCPR2L si queremos variar el ciclo de trabajo del ventilador de ingreso de aire a temperatura ambiente. La misma naturaleza del control de temperatura hace incompatible el funcionamiento de ambos ventiladores, por tanto si CCPR1L tiene un valor entonces CCPR2L es cero y viceversa. Si CT es el ciclo de trabajo entonces en el programa podemos variar este entre 0 y $256(2^8)$, luego $CT_{max} = 256$.

En la ecuación (5,9) escogeremos un valor de n de tal forma que el coeficiente de $[e(m) - e(m-1)]$ sea 1, esto nos facilitará el algoritmo; luego:

$$\frac{CT_{max}}{n\Delta r_{max}} = 1 \Rightarrow n = \frac{CT_{max}}{\Delta r_{max}} = \frac{256}{8} \Rightarrow n = 32$$

Remplazando estos valores en 5,21, tendríamos para el decremento:

$$d \approx 8 + [e(m) - e(m-1)], \quad 1 \leq m \leq n, n = 32$$

Puesto que en el assembler del microcontrolador no podemos propiamente restar un número mayor a un número menor, entonces redefinimos la ecuación anterior como:

$$d \approx \begin{cases} 8 + [e(m) - e(m-1)] & \text{si } e(m) > e(m-1) \\ 8 & \text{si } e(m) = e(m-1) \\ 8 - [e(m-1) - e(m)] & \text{si } e(m) < e(m-1) \end{cases} \tag{5.23}$$

La ecuación (5,23) se implementa en el programa y se toma siempre en cuenta que $d \geq 0$ y $0 \leq CT \leq 256$. Se muestra un fragmento del código que implementa $8 - [e(m-1) - e(m)]$.

```
Error_I_mas     movf    Error_F,w
                subwf   Error_I,f       ;Error_I = Error_I - Error_F
                bsf     STATUS,C        ;Para consultar si la resta es negativa
                sublw   d'8'            ;w = 8 - (Error_I - Error_F)
                btfss   STATUS,C
                movlw   0x00            ;Si la resta es negativa: w = 8 - (Error_I - Error_F) = 0
                ;Colocación del nuevo ciclo de trabajo
                bsf     STATUS,C        ;Para consultar si la resta es negativa
                subwf   CT_PWM,w        ;w = CT_PWM - d
                btfss   STATUS,C        ;Si la resta es negativa: CT_PWM = 0
                movlw   0x00
                movwf   CT_PWM

                movf    Error_F,w       ;Error_F -> Error_I
                movwf   Error_I
                return
```

Donde se puede apreciar que Error_F= $e(m)$ y Error_I= $e(m-1)$, finalizado el cálculo del decremento se asigna el valor final del error al valor inicial, ya que este sería el nuevo valor inicial para el siguiente cálculo.

Hasta este momento se tiene el decremento a realizar d y el número de decrementos $n = 32$, pero ¿durante cuánto tiempo se ejecutarán estos decrementos y cada cuánto tiempo?. Si nuevamente revisamos el cuadro 5,1 y sobre todo la recta de regresión lineal $y = 0{,}5073x + 23{,}238$ de la Figura 5,12 nos percatamos que cada dos minutos el incremento es de aproximadamente un grado, por tanto si queremos que las variaciones tiendan a corregirse ya en un grado, tendríamos que hacer el tiempo de ajuste en dos minutos, es decir $T_a = 120s$.

Despejando $N.T_{timer}$ de la ecuación (5.9) y reemplazando valores.

$$N.T_{timer} = \frac{T_a}{n-1} = \frac{120}{32-1} \Rightarrow N.T_{timer} = 3{,}871s \tag{5.24}$$

Es decir los decrementos se realizarían cada $3{,}871s$. Luego para realizar estos cálculos de tiempo se requiere usar un temporizador, y para esto haremos uso del timer uno del microcontrolador, lo denotaremos por T_{timer}. Para un oscilador de $4MHz$ como el que se está usando y un preescalador de 1/8, el máximo alcanzable por T_{timer} es de $0{,}524288s$, por lo cual no llega por si

sólo a darnos los conteos de cada 3,871s que se requieren, es por este motivo que multiplicaremos este tiempo N veces, donde $N \in \mathbf{N}$, luego escogeremos el tiempo del timer uno como su máximo posible, $T_{timer} = 0{,}524288s$.

Usando la ecuación (5,24) para calcular el valor de N.

$$N = \frac{3{,}871}{T_{timer}} = \frac{3{,}871}{0{,}524288} \Rightarrow N = 7{,}383, \quad \text{pero } N \in \mathbf{N} \Rightarrow N = 7$$

Puesto que N se ha aproximado a siete, entonces también cambiaría el tiempo de ajuste, y el tiempo entre cada decremento, haciendo uso de las ecuaciones (5,9) y (5,24), tendremos finalmente los siguientes valores.

$T_{timer} = 0{,}524288s$	Valor de conteo del timer uno.
$N = 7$	Número de veces que se repite el conteo del timer uno.
$N.T_{timer} = 3{,}67s$	Tiempo entre cada decremento.
$T_a = 113{,}77s$	Tiempo de ajuste.
$n = 32$	Número de decrementos del CT.

Se muestra a continuación la implementación de temporizador $N.T_{timer}$.

```
    ;Atención a la interrupción por Timer 1
Inte_Timer1     bcf     PIR1,TMR1IF     ;Limpiamos bandera de interrupción causada por Timer 1
                clrf    TMR1H           ;Inicialización de contador msb del timer 1 en 0
                clrf    TMR1L           ;Inicialización de contador lsb del timer 1 en 0

                decfsz  N_timer1,1
                goto    fin_Inte        ;Retorno de interrupción
```

Cabe mencionar que la relación entre la temperatura $y(t)$ y el voltaje entregado $r(t)$ por el sensor está dado por la ecuación (5.17), para el LM35 se tiene que $p = 100$. Pero este valor del voltaje no es el que usa el microcontrolador para sus cálculos aritméticos ya que primero pasa por el ADC.

Si $r_d(t)$ es el valor entregado por el ADC, entonces:

$$r_d(t) = 2014{,}6r(t) \tag{5.25}$$

Esto no modifica los resultados anteriores ya que el factor 2014,6 es común en el numerador y denominador del segundo término del primer miembro de la ecuación (5,21) ya que $e(n) - e(n-1) = -(r(n) - r(n-1))$ y por tanto de anulan.

Con respecto a la modificación del estado de variables digitales y sus lecturas, no amerita mayor explicación ya que no es más que activar o desactivar salidas digitales y consultar el estado de entradas digitales[4]. Sin embargo cabe la aclaración de que se usa un registro auxiliar **REG_CONFI** para

[4]La obtención del código fuente completo y comentado se puede hacer desde la pagina web de CJAR

guardar las configuraciones que se setean ya sea localmente por los switches o remotamente por palabras predefinidas, estas configuraciones guardadas son consultadas por el programa para ejecutar las acciones que se soliciten.

Se muestra a continuación el cuadro con los bits de configuración de REG_CONFI.

Posición	Estado	Indicativo
REG_CONFI[2][1]	00	No se obtiene el texto de las temperatura.s
	01	Optiene texto de la temp. del sensor.
	10	Optiene texto de la temp. del potenciómetro.
	11	Optiene textos de ambas temperaturas.
REG_CONFI[3]	0	Control local ON.
	1	Control local OFF.
REG_CONFI[4]	0	Permite detección de puerta abierta.
	1	No permite detección de puerta abierta.
REG_CONFI[5]	0	Permite sonido de alarma.
	1	Desactiva sonido de alarma.
REG_CONFI[6]	0	Luz prendida.
	1	Luz apagada
REG_CONFI[7]	0	Permite control de temperatura.
	1	Desactiva control de temperatura.

Por ejemplo el código que hace una consulta al bit 6 del registro REG_CONFI para saber si apagar o encender a luz es:

```
salto_e         btfsc   REG_CONFI,6      ;¿Prender o apagar luz?
                goto    salto_d - d'1'
                bsf     PORTB,RB5
                goto    salto_d
                bcf     PORTB,RB5

salto_d         btfsc   REG_CONFI,4      ;¿Detección de puerta permitido?
```

Capítulo 6

Resultados y Conclusiones

6.1. Resultados de la implementación

Las imágenes mostradas en este capítulo corresponden a la ejecución de CJAR en modo automático HTTP y su conexión a la aplicación en domótica.

Se muestra a continuación la ejecución de CJAR Web en Google Chrome.

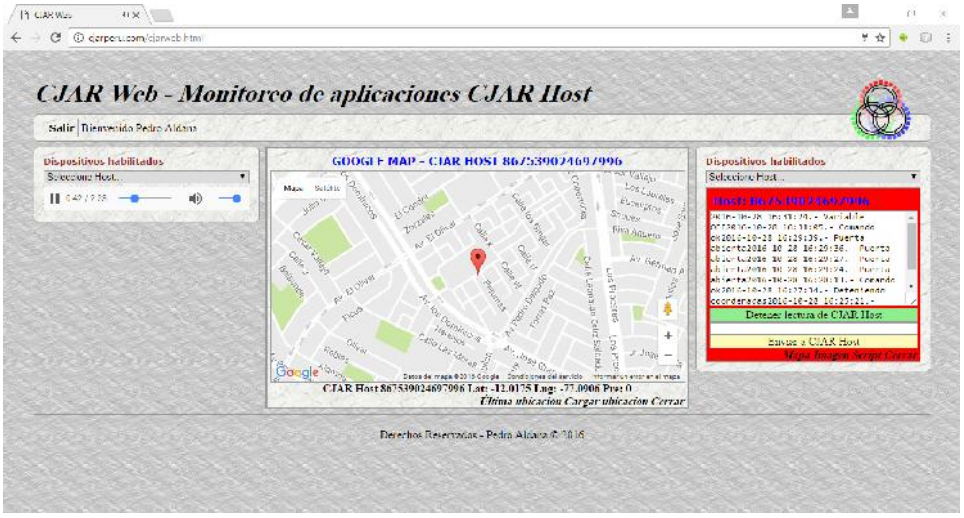
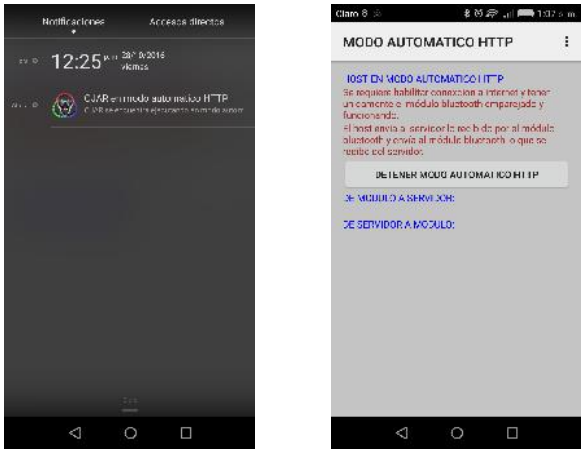


Figura 6.1: CJAR Web ejecutándose en Google Chrome

Como se puede apreciar, el microcontrolador está enviando información a través de CJAR y respondiendo a los comandos enviados, también se ejecutó los comandos para solicitar el envío de coordenadas por parte de CJAR Host, el cuadro en rojo y la reproducción del audio son indicativos del ingreso de datos a CJAR Web.

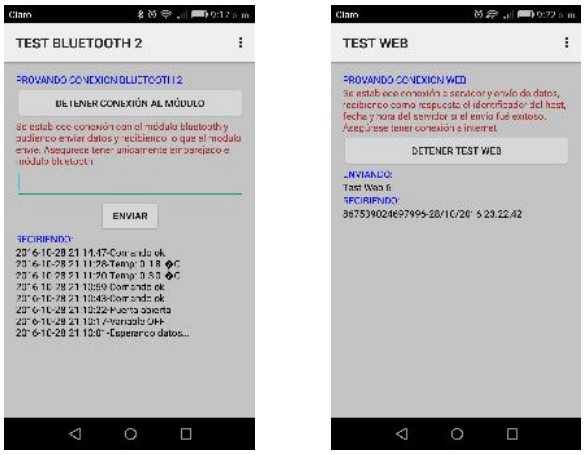
Las siguientes imágenes muestran la notificación en el host (celular) de CJAR ejecutándose en modo automático HTTP y en la imagen de la derecha se aprecia el ícono de GPS consultado.



(a) Notificación (b) GPS consultado

Figura 6.2: CJAR Host en ejecución

Previamente se realizaron pruebas de conexión local y remota.

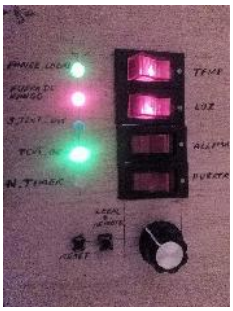


(a) Test Bluetooth 2 (b) Test Web

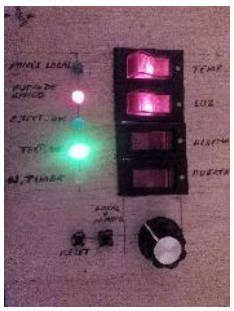
Figura 6.3: Pruebas de conexión

Las siguientes fotos muestran los elementos de control y monitoreo de la maqueta en pleno funcionamiento.

Figura 6.4: Maqueta en funcionamiento



(a) Control local



(b) Control remoto

Figura 6.5: Panel de control en maqueta

6.2. Conclusiones

1. Ha sido posible la implementación de CJAR como una plataforma que integra elementos informáticos y mínimo hardware para aplicaciones en el campo de la electrónica.

2. El desarrollador o implementador de aplicaciones ve de forma transparente el diseño y funcionamiento interno de CJAR, por lo que sólo ha de centrarse en el diseño de su aplicación

3. Las aplicaciones que se pueden desarrollar pueden ser elementales como el del CBP (Circuito Básico de Pruebas) o de relativa complejidad como la aplicación en domótica presentada.

4. El aspecto más crítico e importante para la integración de una aplicación con CJAR es la comunicación. Si el desarrollador implementa un programa, a de asegurar que sus algoritmos de comunicación serial sean eficientes.

5. CJAR puede interactuar con más de una aplicación a la vez por cada usuario y con múltiples usuarios al mismo tiempo, CJAR es de naturaleza escalable.

6. Se ha implementado una aplicación en domótica que interactua correctamente con CJAR.

7. La aplicación en domótica implementa un algoritmo de comunicación serial bit a bit, de naturaleza 'simétrica creciente' que puede ser usado en aplicaciones elementales así como en aplicaciones de mayor complejidad, algoritmo que en la aplicación presentada se ha escrito en lenguaje ensamblador.

Capítulo 7

Recomendaciones

1. Implementar de forma alternativa y complementaria la conexión vía USB.
2. Implementar los comandos adicionales en el Modo Automático HTTP de CJAR Host.
3. Implementar imágenes y vídeo en CJAR.
4. Lectura de los distintos sensores de los dispositivos Android.
5. Implementar la edición de scripts de usuario en CJAR Web.
6. Otros

Apéndice A

Circuito Básico de Pruebas (CBP) para CJAR

A.1. Descripción del CBP

El Circuito Básico de Pruebas es un circuito de gran utilidad durante la etapa de desarrollo de aplicaciones, ya que con este se aprecia con simplicidad el funcionamiento de CJAR, emplea hardware básico y de fácil adquisición.

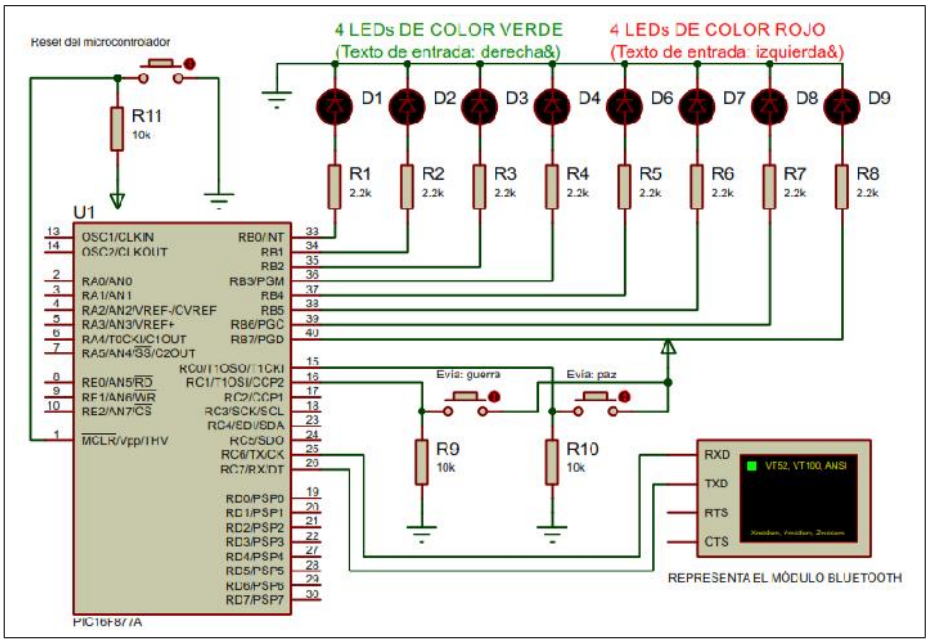


Figura A.1: Circuito básico de pruebas (CBP)

El microcontrolador PIC16f877a requiere un oscilador externo, el cual no aparece en el circuito pero es fundamental para que funcione. Durante la etapa de pruebas la conexión entre el microcontrolador y el módulo puede ser reemplazada, por ejemplo conectando el microcontrolador o el módulo a un MAX232 y luego al puerto serie de una computadora para visualizar, enviar y recibir datos de forma más cómoda, comprobando de esta forma la comunicación. La conexión entre este módulo y este microcontrolador puede ser directa ya que los niveles de voltaje son adecuados.

Se muestra a continuación este CBP (Circuito Básico de Pruebas) armado y funcionando, se puede apreciar un MAX232, esto se usó en etapas de prueba, pero en realidad este MAX no está conectado a nada, esto se puede apreciar en la foto.

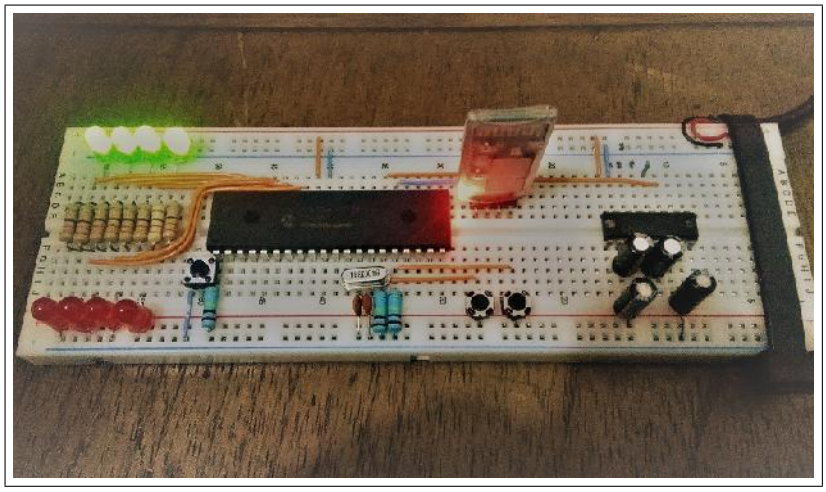


Figura A.2: CBP implementado

Apéndice B

Programa en Assembler para PIC usado en el CBP

B.1. Descripción del programa

1. El programa inicia con RB0 y RB7 encendidos.
2. Envía asincronamente por la transmisión USART la palabra 'paz' si presionas y luego sueltas RC0 y 'guerra' si presionas y luego sueltas RC1.
3. Puede emplearse para enviar cualquier texto para cualquier condición dada; sólo se repite consecutivamente para cada texto todo lo comentado como (Tx_m:).
4. Si recibe 'derecha&' y enciende RB0, RB1, RB2 y RB3; si recibe 'izquierda&' y enciende RB4, RB5, RB6 y RB7.
5. Puede recibir cualquier texto terminado en '&' por la recepción USART, colocarlo como palabra pre definida y ejecutar una instrucción asociada, sólo se repite consecutivamente para cada palabra pre definida todo lo comentado como (Rx_n:).
6. Mientras se reciba parte de un texto predefinido ("de", "derech", ïzq", ïzquier", etc) se encenderán RB0, RB2, RB4, RB6 y se apagarán RB1, RB3, RB5 y RB7. Si el texto se completa correctamente se ejecutará lo indicado en el punto (3), caso contrario se encenderá todo el puerto B, indicativo de texto no predefinido.
7. Para cualquier texto recibido no predefinido se encenderá todo el puerto B.

8. Todo aquel código que puede ser modificado, para a un proyecto particular sin afectar el uso del USART y de palabras predefinidas se le agrega (Mo:) en el comentario.

9. La palabras predefinidas de este programa pueden ser modificadas, por ejemplo puedes sustituir 'paz' por 'mundo feliz' o 'derecha' por 'no seas terco', etc., se sugiere hacer la modificación de las palabras en el código que se ejecuta y también en la tabla de al final para mantener su referencia.

10. Para los textos de ingreso: Siendo $[|\ x\ |]$ la función máximo entero de x, entonces $f(n) = (n-1) - 8[|\ (n-1)/8\ |], \quad n \in \mathbf{N}$; n_max el máximo valor de n, en este programa $n\_max = 2$.

11. La recepción es usando interrupciones.

12. El programa implementa la estructura básica para el uso de textos completos con el módulo USART, implementación de palabras predefinidas y ejecutar comandos asociados.

B.2. Extracto del programa

```
;Programa en Assembler PIC 16F877A para el CBP de CJAR
;Autor: Aldana Quintana Pedro Alejandro -- www.cjarperu.com

                LIST    p=16f877a       ;Directiva para definir microcontrolador
                INCLUDE <p16f877a.inc>  ;Incluye fichero de etiquetas
 __CONFIG _CP_OFF&_CPD_OFF&_WDTE_OFF &_BOREN_OFF&_PWRTE_OFF&_FOSC_XT&_WRT_OFF&_LVP_OFF&_DEBUG_OFF;   ;(Mo:) Directivas del microcontrolador.
;Definición de constantes y variables
Variables_20h   udata   0x20            ;Grupo de variables que empiezan en la dirección 20h
WREG_Temp       res     1               ;Guarda temporalmente WREG, usado en interrupciones
STATUS_Temp     res     1               ;Guarda temporarlmente STATUS, usado en interrupciones
PCLATH_Temp     res     1               ;Guarda temporarlmente PCLATH, usado en interrupciones
Selec_Mens_Tx   res     1               ;Asociado el mensaje a enviar por Tx de USART, se presenta una tabla al final
Selec_Mens_Rx   res     1               ;Asociado al mensaje recibido por RX de USART, se presenta una tabla al final
Apunt_Tx        res     1               ;Apunta los caracteres a enviar en USART
Apunt_Rx        res     1               ;Apunta los caracteres recibidos en USART
Dato_Tx_Rx      res     1               ;Auxiliar
WREG_Aux        res     1               ;Auxliar
Texto_Aux_1     res     1               ;(Rx:)Variables de ayuda para identificar que texto es valido o no, cada uno es valido hasta para 8 textos
;(Rx_n:)        .
;(Rx_n:)        .
;(Rx_n:)Texto_Aux_(((n_max+7-f(n_max))/8))      res     1

                org     0000h           ;Dirección de reset
                goto    Inicio
                org     0004h           ;Dirección de interrupción
                goto    Interrupcion

;Tablas antes de 255 o 0xFF
    ;Tablas de comparación de texto recibido
text_rx_1       movf    Apunt_Rx,w      ;Carga apuntador a w
                addwf   PCL,1           ;Salto w instrucciones adelante
```

B.3. Código fuente assembler y archivos generados

El conjunto de archivos completos de este programa pueden ser descargados de la página web de CJAR.

www.cjarperu.com

Apéndice C

Simulaciones en Proteus y MPLAB X IDE

Dos de las herramientas más potentes para trabajar con los microcontroladores PIC son Proteus y MPLAB X IDE, es posible integrar ambos programas.

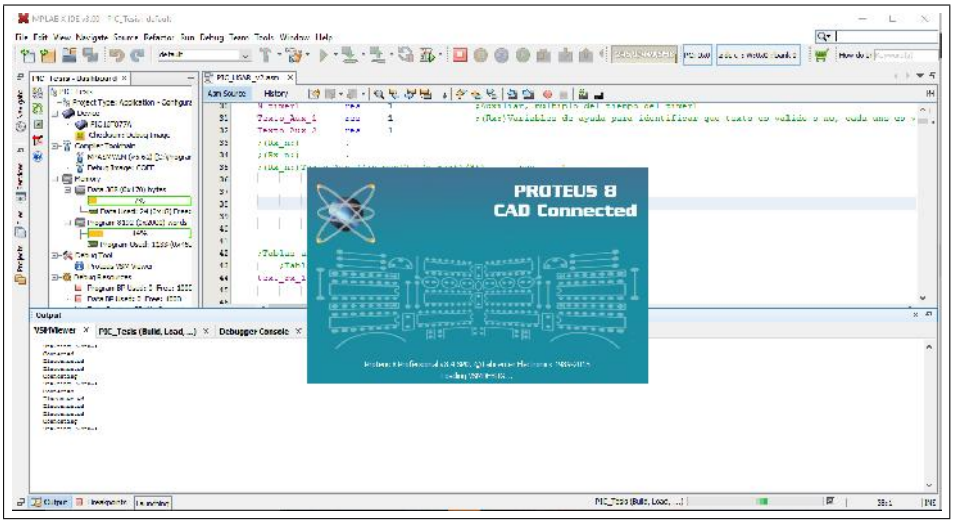


Figura C.1: Integración de Proteus y MPLAB X IDE

Se puede apreciar en la parte izquierda en el dashboard (tablero de instrucciones) que en Debug Tool está configurado Proteus VSM Viewer.

Las siguientes imágenes muestran las simulaciones de comunicación serial y el control de temperatura por PWM. Notar los valores en los registros

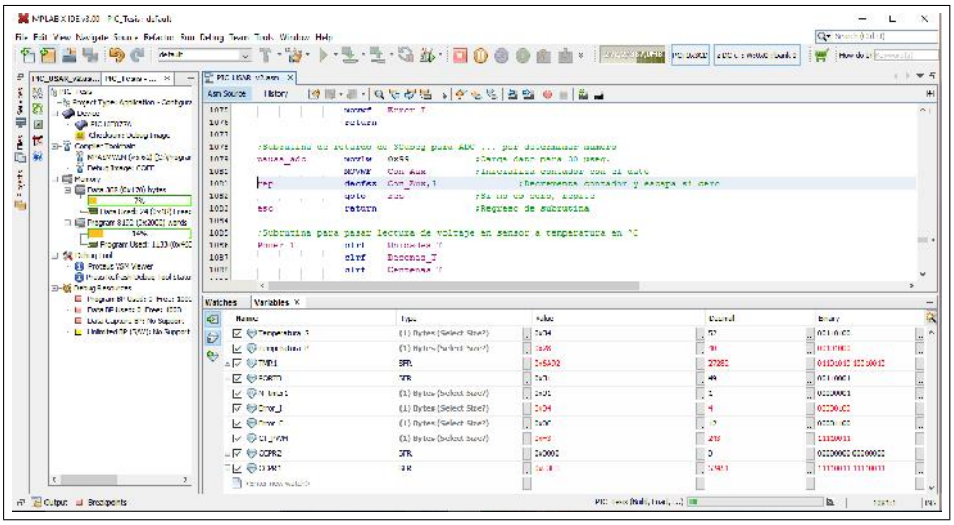


Figura C.2: Simulación de comunicación serial y PWM en MPLAB X IDE

Los números resaltados con color rojo son los que en la última ejecución del programa han cambiado sus valores, notar el valor cambiante en el timer. Para efectos de simulación y ver los cambios rápidamente se ha dado al múltiplo del timer el valor de uno.

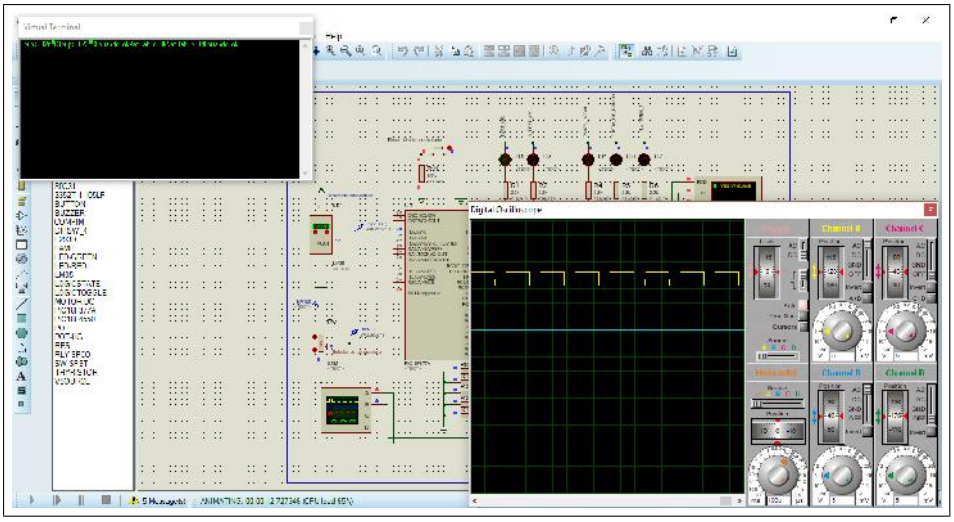


Figura C.3: Simulación de comunicación serial y PWM en Proteus

Bibliografía

[GCMA, 2014] GSMA Association (2014). Economía Móvil, América Latina 2014. *Informe anual de los operadores de todo el mundo.* www.gsma.com.

[Gartner, 2015] Gartner, Inc. (2015). Gartner Says Global Devices Shipments to Grow 2.8 Percent in 2015. *Empresa consultora y de investigación de las tecnologías de la información.* www.gartner.com.

[IDC, 2015] International Data Corporation (2015). Android and iOS Squeeze the Competition. *Empresa de analisis y consultoria, especializada en tecnologías de la información, telecomuniones y de consumo.* www.idc.com.

[OSIPTEL, 2013] Organismo Supervisor de Inversión Privada en Telecomunicaciones (2013). Dispositivos que acceden a Internet a Dic 2013. *Regulador y supervisr del mercado de servicios públicos de telecomunicaciones en el Perú.* www.osiptel.gob.pe.

[CISCO, 2015] Cisco Systems, Inc (2015). Global Mobile Data Traffic Forecast Update 2014 - 2019 White Paper. *Empresa global dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones.* www.cisco.com.

[Ericsson, Junio 2015] L. M. Ericsson (2015). Ericsson Mobility Report Q1 2015. *Compañía multinacional de Suecia dedicada a ofrecer equipos y soluciones de telecomunicaciones, principalmente en los campos de la telefonía, la telefonía móvil las comunicaciones multimedia e internet.* www.ericsson.com.

[ScientiaMobile, 2015] ScientiaMobile (2015). Mobile Overview Report April-June 2015. *ScientiaMobile es una compañia diseñadora del popular WURFL (Wireless Universal Resource FiLe) Un proyecto Open-Source, el cual es un reconosido standard de facto en el area de DDR (Device*

Description Repositories).
www.scientiamobile.com.

[Android Developers, 2015] Android Developers (2015). *Guia de desarrolladores y referencias.*
developer.android.com.

[Tomás, 2012] Tomás Gironés Jesús (2012). El Gran Libro de Android. *Alfaomega Grupo Editorial, S.A. de C.V., México.*

[Wei-Meng, 2011] Wei-Meng Lee(2011). Beginning Android Application Development. *Wiley Publishing, Inc., United States of America*

[Frank, Charlie and Robi, 2009] W. Frank Ableson, Charlie Collins and Robi Sen (2009). Unlocking Android a Developer's Guide. *Manning Publications Co., United States of America.*

[F. Maciá, 2008] F. Maciá (2008). Administración de Servidores de Internet, De la Teoría a la Practica. *Publicaciones de la Universidad de Alicante, España.*

[Santos, 2014] Santos Antonio Alva (2014). Operalización de la Variables.

[Microchip, 2003] Microchip Technology Inc. (2003). PIC16F87XA Data Sheet.