



**UNIVERSIDAD MAYOR DE SAN SIMÓN
FACULTAD DE CIENCIAS Y TECNOLOGÍA
CARRERA DE INGENIERÍA ELECTRÓNICA**



DISEÑO DE UN SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA AUTOMATIZACIÓN DE EDIFICACIONES

Proyecto de Grado, Presentado Para Optar al Diploma Académico de
Licenciatura en Ingeniería Electrónica.

PRESENTADO POR:

MIJAHIL DAVID HEREDIA HERBAS

TUTOR:

Ing. Boris Fernando Alfaro Bazán

**COCHABAMBA – BOLIVIA
Abril, 2022**

DEDICATORIA

Este trabajo está dedicado:

*A mi madre la Sra. Delina Herbas Varias
Que me enseño que el amor es estar dispuesto a
sacrificar tu felicidad por la felicidad de otros.*

AGRADECIMIENTOS

*En la noche que me envuelve,
negra, como un abismo insondable,
le doy gracias a mi Dios,
Por mi alma inconquistable.*

*En las garras de las circunstancias,
no me he lamentado, ni he llorado.
Bajo los golpes del destino,
mi cabeza ensangrentada jamás se ha postrado.*

*Más allá de este lugar de ira y llantos,
Donde yace el horror de las sombras,
La amenaza de los años me halla,
y me hallará sin temor.*

*Ya no importa que tan estrecho haya sido el camino,
ni cuantos castigos lleve mi espalda,
Soy el amo de mi destino,
Soy el capitán de mi alma.*

Henley, W (1888). Invictus. A book of verses. Londres: D. Nutt.

RESUMEN

Actualmente en Bolivia la mayoría de las edificaciones tienen una estructura básica con sistemas eléctricos simples, carecen totalmente de automatizaciones y mucho mas de sistemas que gestionen la edificación, provocando una relativa baja calidad de vida, edificaciones inseguras e inefficientes en la gestión de recursos, y finalmente existe una pésima administración de la información que se genera en las edificaciones.

Se diseña entonces un sistema de inteligencia artificial para la automatización de las edificaciones denominada SIA.

Y a través de la construcción de un prototipo domótico electrónico demostrativo y el desarrollo de un sistema inteligente que cuenta con asistencia virtual, SIA plantea resolver los problemas de seguridad e inefficiencia de ahorro de recursos, y mejorar la calidad de vida de los residentes, además de gestionar la información que generan las edificaciones.

Concluyendo de esta manera con la obtención de un sistema robusto, flexible, funcional y de muy bajo costo, capaz de centralizar mucha información generada por las edificaciones y controlar diferentes dispositivos que logran brindar un servicio que satisface las necesidades y problemáticas que existen en las edificaciones.

Palabras Clave: Domótica, inteligencia artificial, asistentes virtuales.

ÍNDICE

	Pág.
1 GENERALIDADES.....	2
1.1 INTRODUCCIÓN	2
1.2 ANTECEDENTES DEL TEMA GENERAL.....	2
1.3 FORMULACIÓN DEL PROBLEMA.....	3
1.4 JUSTIFICACIÓN DEL TEMA	7
1.5 OBJETIVOS	8
1.5.1 Objetivo general.	8
1.5.2 Objetivos específicos.....	8
1.6 ALCANCE Y LIMITACIONES.	8
1.6.1 Alcances.....	8
1.6.2 Limitaciones.	9
2 MARCO TEÓRICO.....	11
2.1 CONTENIDO DEL MARCO TEÓRICO.	11
2.2 DOMÓTICA E INMÓTICA.....	12
2.2.1 Tipos de arquitectura.....	12
2.2.2 Servicios y aplicaciones de la domótica.....	15
2.2.2.1 Confort.	15
2.2.2.2 Comunicación.....	16
2.2.2.3 Energía.	17
2.2.2.4 Seguridad.....	19
2.3 INTELIGENCIA ARTIFICIAL.	22
2.3.1 Tipos de inteligencia artificial.....	23
2.3.2 Agente racional.....	24
2.3.3 El futuro de la inteligencia artificial.	25
2.4 ASISTENTES VIRTUALES.....	26
2.4.1 Asistente personal inteligente.....	27
2.4.2 Servicios.	27
2.5 RECONOCIMIENTO DE VOZ.	28
2.5.1 Clasificación.	29

2.5.2	Aplicaciones	29
2.6	RASPBERRY	30
2.6.1	Microcomputadora.....	31
2.6.2	Otras computadoras de placa reducida.....	31
2.7	ARDUINO	32
2.7.1	Microcontroladores.....	33
2.7.2	Microprocesadores.....	33
2.7.2.1	Memoria	34
2.7.2.2	Memoria de programa	35
2.7.2.3	Memoria de datos.....	35
2.7.2.4	Unidades de entrada y salida.....	35
2.8	SENSORES Y ACTUADORES.....	35
2.8.1	Sensor	35
2.8.1.1	Características generales	36
2.8.1.2	Clasificación.....	36
2.8.1.3	Tipos de sensores	37
2.8.2	Actuador	39
2.8.2.1	Clasificación.....	39
3	INGENIERÍA Y DISEÑO DEL SISTEMA ELECTRÓNICO.....	42
3.1	ANÁLISIS DE REQUERIMIENTOS.....	42
3.2	DIAGRAMA DE BLOQUES DEL BÁSICO DEL SISTEMA	44
3.3	SELECCIÓN DEL SISTEMA EMBEBIDO.....	45
3.3.1	Selección de la estructura del sistema domótico	45
3.3.2	Selección del microcomputador	45
3.3.3	Selección de sensores, actuadores, interfaces de entrada y salida, microcontroladores y otros componentes.....	46
3.3.4	Distribución de los periféricos, sensores y actuadores	47
3.3.5	Selección del sistema de alimentación	49
3.4	CONSIDERACIONES EN EL DISEÑO DEL CIRCUITO.....	50
3.5	DIAGRAMA DE CIRCUITO DEL PROTOTIPO DEL SISTEMA DE INTELIGENCIA ARTIFICIAL	51

3.6	DIAGRAMA DE CONEXIÓN DEL PROTOTIPO DEL SISTEMA DE INTELIGENCIA ARTIFICIAL	52
3.7	DIAGRAMA PCB DE LA PLACA DEL SISTEMA DE INTELIGENCIA ARTIFICIAL	53
4	INGENIERÍA Y DESARROLLO DEL SOFTWARE	55
4.1	SELECCIÓN DEL SISTEMA OPERATIVO	55
4.2	SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN.....	56
4.2.1	Evaluación comparativa de lenguajes de programación.....	56
4.2.2	Selección y Justificación de la selección del lenguaje.	59
4.3	SELECCIÓN DEL ENTORNO DE DESARROLLO INTEGRADO (IDE).....	60
4.4	SELECCIÓN DEL GESTOR DE BASE DE DATOS.....	60
4.5	SELECCIÓN DEL MÓDULO DE RECONOCIMIENTO DE VOZ.....	60
4.6	DESARROLLO DEL AGENTE RACIONAL.....	61
4.6.1	Descripción R.E.A.S. del entorno.....	61
4.6.2	Descripción P>A (Percepción > Acción) del entorno.	62
4.7	DESARROLLO DEL SISTEMA.....	63
4.7.1	Adaptación de la metodología.....	63
4.7.2	Fase de inicio.....	64
4.7.2.1	Definición de riesgos.....	64
4.7.2.2	Requerimientos funcionales del sistema.	65
4.7.2.3	Requerimientos no funcionales del sistema.....	65
4.7.3	Fase de elaboración.	66
4.7.3.1	Modelo general de casos de uso.....	66
4.7.3.2	Componentes del sistema.....	68
4.7.3.3	Arquitectura del sistema.....	69
4.7.4	FASE DE CONSTRUCCIÓN	70
4.7.4.1	Diagrama UML del Sistema de Inteligencia Artificial (S.I.A.)	70
4.7.4.2	Especificación casos de uso – módulo SIA.....	71
4.7.4.3	Especificación casos de uso – módulo piso.	73
4.7.4.4	Especificación casos de uso – módulo apartamento.	74
4.7.4.5	Especificación casos de uso – módulo habitación.	76
4.7.4.6	Diagrama entidad relación de la base de datos.	77

4.7.4.7	Diagrama UML de la base de datos.	78
4.7.4.8	Diccionario de datos.....	78
4.7.5	Fase de transición.	82
4.7.5.1	Pruebas de integración.	82
4.7.5.2	Cambios en las edificaciones debido al sistema propuesto.....	85
4.7.5.3	Aspectos de implantación.....	86
5	EVALUACIÓN ECONÓMICA.	89
5.1	ESTRUCTURA FÍSICA DEL PROTOTIPO.	89
5.2	ACTUADORES Y PERIFÉRICOS DE SALIDA.....	90
5.3	SENSORES Y PERIFÉRICOS DE ENTRADA.	90
5.4	SISTEMA EMBEBIDO.....	91
5.5	COMPONENTES DESPRECIABLES.	93
5.6	PRESUPUESTO TOTAL.....	94
6	CONCLUSIONES Y RECOMENDACIONES.....	96
6.1	CONCLUSIONES.	96
6.2	RECOMENDACIONES.....	97
	BIBLIOGRAFÍA.....	99
	ANEXOS. 103	
	ANEXO A: Comandos por voz.	103
	ANEXO B: Historial de comandos de paquetes, módulos y librerías instaladas y/o usadas.	105
	ANEXO C: Código fuente.....	111
	ANEXO D: Fichas Técnicas.....	145

INDICE DE TABLAS

	Pág.
1 GENERALIDADES.....	2
2 MARCO TEÓRICO.....	11
Tabla 2.1. Contenido de la fundamentación teórica	11
Tabla 2.2. Comparación de asistentes con relación a la eficiencia de respuestas.....	28
Tabla 2.3. Tabla comparativa de microcomputadoras con placa reducida.	32
Tabla 2.4. Tipos y ejemplos de sensores electrónicos.	38
3 INGENIERÍA Y DISEÑO DEL SISTEMA ELECTRÓNICO.....	42
Tabla 3.1. Ficha técnica del microcomputador empleado.	46
Tabla 3.2. Lista de componentes seleccionados para el prototipo del sistema de inteligencia artificial. 46	
Tabla 3.3. Distribución de sensores, actuadores y otros componentes en la edificación demostrativa.....	48
4 INGENIERÍA Y DESARROLLO DEL SOFTWARE.	55
Tabla 4.1. Tabla de requerimientos de lenguajes de programación.	59
Tabla 4.2. Descripción REAS del entorno.....	61
Tabla 4.3. Tabla parcial de una función de agente sencilla para el sistema SIA.....	62
Tabla 4.4. Especificación de fases en función de artefactos.....	63
Tabla 4.5. Definición de riesgos.....	64
Tabla 4.6. Requerimientos funcionales del sistema.....	65
Tabla 4.7. Requerimientos no funcionales del sistema.....	66
Tabla 4.8. Módulos del sistema.....	68
Tabla 4.9. Casos de usos.....	71
Tabla 4.10. Especificación casos de uso – módulo piso.	73
Tabla 4.11. Especificaciones casos de uso – módulo apartamento.....	74
Tabla 4.12. Especificación casos de uso – módulo habitación.	76
Tabla 4.13. Diccionario de datos.	79
Tabla 4.14. Ficha técnica del consumo de recursos.....	87

5 EVALUACIÓN ECONÓMICA	89
 Tabla 5.1. Costos de materiales para la estructura.....	89
 Tabla 5.2. Costos de actuadores usados en el prototipo.....	90
 Tabla 5.3. Costos de sensores y periféricos de entrada usados en el prototipo.....	90
 Tabla 5.4. Costos de los materiales usados para la construcción de la tarjeta SIA	91
 Tabla 5.5. Costo del microcontrolador y dispositivos relacionados a su funcionamiento	91
 Tabla 5.6. Costo del microcomputador y dispositivos relacionados a su funcionamiento.....	92
 Tabla 5.7. Tabla estimativa de costos de componentes para un ordenador estándar.....	92
 Tabla 5.8. Tabla comparativa de precios.....	93
 Tabla 5.9 Componentes de costos despreciables.....	93
 Tabla 5.10 Costos totales.....	94
6 CONCLUSIONES Y RECOMENDACIONES.....	96
BIBLIOGRAFÍA	99
ANEXOS. 103	
 Tabla A.1. Lista de comandos.	103
 ANEXO D: Fichas Técnicas.....	145

ÍNDICE DE FIGURAS

	Pág.
1 GENERALIDADES.....	2
Figura 1.1: Árbol de problemas, causas y efectos.....	5
Figura 1.2: Árbol de soluciones.....	6
2 MARCO TEÓRICO.....	11
Figura 2.1: Esquema de arquitectura con periferia centralizada.....	13
Figura 2.2: Esquema de arquitectura con periferia descentralizada.....	13
Figura 2.3: Esquema de la arquitectura distribuida.....	15
Figura 2.4: Evolución de las estructuras hacia estructuras inteligentes.....	22
Figura 2.5: Diagrama de un agente basado en utilidades y modelos.....	25
Figura 2.6: Diagrama básico de un microprocesador.....	34
Figura 2.7: Diagrama en bloque de un sistema de computadora completo, con las memorias de datos y de programa como elementos externos al microprocesador.....	34
Figura 2.8: Esquema de clasificación de actuadores.....	40
3 INGENIERÍA Y DISEÑO DEL SISTEMA ELECTRÓNICO.....	42
Figura 3.1: Plano domicilio particular donde se instalaría el sistema de inteligencia artificial..	42
Figura 3.2: Plano de instalacion del sistema SIA en el domicilio particular.....	43
Figura 3.3: Diagrama de bloques del funcionamiento básico del sistema.....	44
Figura 3.4: Diagrama general de bloques del funcionamiento del sistema.....	44
Figura 3.5: Diagrama de circuito del transistor.....	50
Figura 3.4: Diagrama de circuito del prototipo SIA.....	51
Figura 3.5: Diagrama de conexión del prototipo SIA.....	52
Figura 3.7: Diagrama PCB de la placa de distribución SIA.....	53
4 INGENIERÍA Y DESARROLLO DEL SOFTWARE	55
Figura 4.1: Tiempo de respuesta Vs numero de nodos aplicación Hit Log.....	57
Figura 4.2: Numero de lineas de código vs lenguaje de programación..	58
Figura 4.3: Modelo general de casos de uso.....	67
Figura 4.4: Diagrama UML de S.I.A.....	70
Figura 4.5: Diagrama entidad-relación de la base de datos.....	77
Figura 4.6: Diagrama UML de la base de datos.....	78
Figura 4.7: Prueba de integración, primera interacción entre módulos I, II, III, IV	83

Figura 4.8: Prueba de integración, segunda interacción entre módulos I, II, III, IV.....	84
Figura 4.9: Prueba de integración, interacción entre módulos I, II, III.....	84
Figura 4.10: Prueba de integración, tercera interacción entre módulos I, II, III, IV.....	85

Capítulo 1

GENERALIDADES.

1 GENERALIDADES.

1.1 INTRODUCCIÓN.

El presente proyecto desarrolla un sistema híbrido (electrónico e informático), con la capacidad de administrar, controlar y automatizar edificaciones en general, abarcando funciones generales que cualquier edificación pueda llegar a tener como por ejemplo el encendido de luces.

Las bases de este proyecto están en las áreas de inmótica, domótica e inteligencia artificial, la interfaz entre el sistema y el usuario es a través de un asistente virtual el cual responde a comandos predeterminados, además de un sistema de mensajería a través de aplicaciones móviles que también son gestionados por el agente de inteligencia artificial que posee el sistema.

El sistema integra las tres funciones principales de la domótica: gestión energética, seguridad y comunicación. Usando distintos sensores, circuitos y programas detallados a lo largo este documento.

La idea de proponer un proyecto tan ambicioso se debe a que cada vez más el término “internet de las cosas” o IoT es más frecuente, esto porque hay más y más dispositivos y objetos conectados a Internet, ya no solo un ordenador o celular son dispositivos con acceso a la red sino también automóviles, lavadoras, televisores y otros dispositivos electrónicos, por ese motivo, puesto que el internet de las cosas seguirá desarrollándose es opinión del autor de este proyecto asumir que tanto la electrónica como la informática podrían acabar integrándose casi en una sola para desarrollar sistemas complejos e inteligentes en el área de la domótica, y prueba de que esto es posible es este mismo proyecto, un sistema domótico con inteligencia artificial.

1.2 ANTECEDENTES DEL TEMA GENERAL.

La inteligencia artificial se ha estado desarrollando de manera exponencial en las últimas décadas con implementaciones y aplicaciones sorprendentes, causando un impacto social, ambiental, económico y en otros ámbitos importantes.

Parte de esta inteligencia artificial es la creación de asistentes virtuales, actualmente muchas de las grandes Tech (Empresas tecnológicas) cuentan ya con asistentes virtuales como parte de sus sistemas operativos, o también de manera exclusiva como un sistema aparte como el caso de Alexa de Amazon. Entre estos asistentes se pueden mencionar a Siri de Apple, Cortana de Microsoft o el asistente de Google, y más empresas tecnológicas se unen al desarrollo de los asistentes virtuales como Samsung que tiene previsto lanzar al asistente virtual “Sam”.

Sin embargo, estos asistentes solo operan a un nivel informático, sus aplicaciones a un nivel de la domótica y la automatización aún tienen una luz tenue y con un amplio campo para desarrollar, solo Alexa de Amazon ha buscado expandirse a un nivel domótico.

Otro concepto fundamental a mencionar es el de la domótica, los cuales son sistemas automatizados capaces de recoger información proveniente de sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas aportando servicios de gestión energética, seguridad, bienestar y comunicación.

El desarrollo de esta tecnología ha evolucionado considerablemente en los últimos años y en la actualidad ofrece una oferta más consolidada.

En Bolivia ha crecido bastante la demanda de automatizaciones en edificaciones a un nivel de seguridad especialmente, con la implementación de cámaras, cercas electrificadas, alarmas, etc.

Sin embargo, son sistemas totalmente independientes uno de otros, las cercas eléctricas son totalmente independientes de las cámaras o los sensores, aun no se ha desarrollado un sistema domótico capaz de integrar todos estos sistemas, y procesar toda la información proveniente de estos en un solo agente domótico.

1.3 FORMULACIÓN DEL PROBLEMA.

Actualmente en Bolivia la mayoría de las edificaciones tienen una estructura básica con sistemas eléctricos simples donde están compuesto solamente por interruptores, luces y puntos de toma de corriente, carecen totalmente de automatizaciones y mucho más de sistemas que gestionen la edificación.

Esto provoca una baja calidad de vida de las personas con respecto a la que podría ser con ayuda de tecnologías domóticas, causando pérdidas de tiempo en actividades triviales y un bajo confort en la comodidad del hogar, también provoca que estas edificaciones sean inseguras y vulnerables a robos, incendios, fugas de gas, y otros accidentes, además que existe una ineficiencia energética que aumenta el costo de los servicios básicos y desperdicia recursos como el agua o la electricidad, y finalmente existe una pésima administración de la información que se genera en las edificaciones desaprovechando dicha información que podría servir para hacer retrospectivas y mejorar posteriormente diferentes aspectos de las mismas edificaciones.

Entonces se formula la siguiente pregunta:

¿Cómo mejorar las edificaciones para que mejoren la seguridad para los residentes, se optimice la distribución de recursos y mejore la calidad de vida de los habitantes a través de la tecnología?

Y la solución sería a través de un sistema de inteligencia artificial capaz de automatizar la edificación y gestionar los recursos y la información recolectada por las interfaces de entrada y salida.

Para llegar a esta conclusión se realizó un análisis del problema principal y la solución a través de un esquema de árbol de problemas y su respectivo árbol de soluciones:

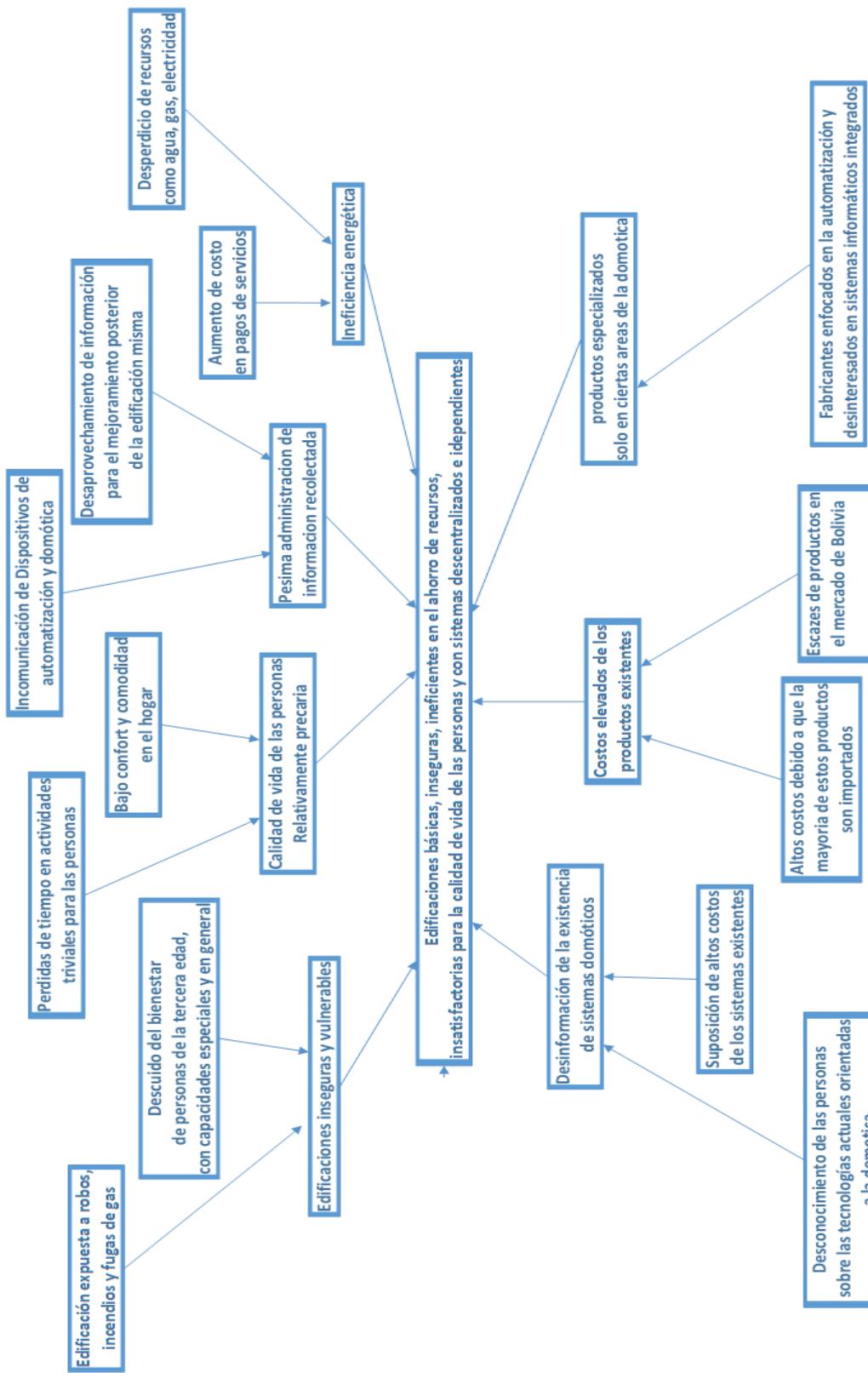


Figura 1.1 Árbol de problemas, causas y efectos.
Fuente: Elaboración propia.

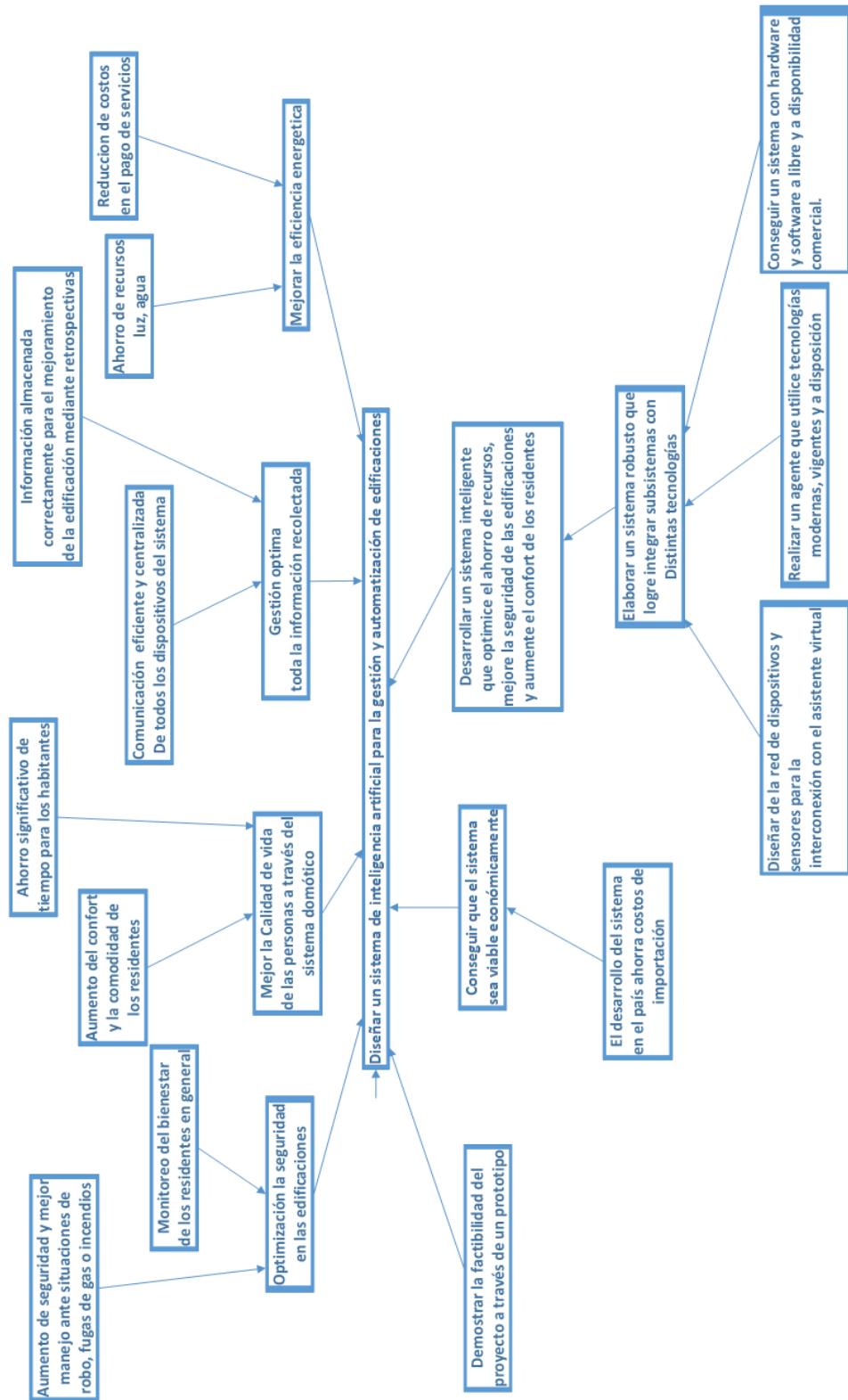


Figura 1.2 Árbol de soluciones.
Fuente: Elaboración propia.

1.4 JUSTIFICACIÓN DEL TEMA.

Con el avance de la tecnología áreas como la domótica, TICs e IoT (Internet of Things o internet de las cosas) cobran más fuerza con el desarrollo urbano, a la vez que se vive una época de la optimización de las cosas para el impacto ambiental.

Con este proyecto se integran todos estos conceptos con los conocimientos obtenidos en la carrera de ingeniería electrónica, en el cual la implementación de este en viviendas o edificaciones públicas podría mejorar la calidad de vida de los residentes aumentando su confort y seguridad, e incrementar el rendimiento de los servicios en las edificaciones.

La administración inteligente que este proyecto brinda, ahorra considerablemente los gastos en los servicios básicos (Agua, Luz, Gas).

Mediante análisis estadísticos mensuales del consumo general, detecta picos de consumo y los administrara de manera eficiente, controlando diferentes dispositivos o informando a los residentes.

De esta forma el ahorro eficiente de la energía ayudaría a reducir los niveles de dióxido de carbono (CO₂) y reducir el efecto invernadero, a la vez que genera un ahorro de costos de servicios para los usuarios residentes.

Además, el proyecto hace uso de las TICs lo cual generaría una relación estrecha con los usuarios.

1.5 OBJETIVOS.

1.5.1 Objetivo general.

Diseñar un sistema de inteligencia artificial para la automatización de edificaciones.

1.5.2 Objetivos específicos.

- Elaborar un hardware robusto para el sistema que logre integrar subsistemas con distintas tecnologías.
- Desarrollar un software inteligente que optimice el ahorro de recursos, mejore la seguridad de las edificaciones y aumente el confort de los residentes.
- Realizar un análisis de costos para conseguir que el sistema sea viable económicamente.
- Implementar un prototipo para demostrar la factibilidad del proyecto.

1.6 ALCANCE Y LIMITACIONES.

1.6.1 Alcances.

- Implementar un prototipo del sistema de inteligencia artificial capaz de gestionar, controlar y administrar una habitación a través de sensores, dispositivos electrónicos y programas interconectados con los cuales un usuario podrá interactuar.
- Controlar encendido y apagado algunos dispositivos electrónicos del hogar.
- Disponer de un asistente virtual para la interacción humano- máquina.
- Monitorear consumo de agua y temperatura a través de sensores.
- Gestionar la seguridad de la edificación.
- Administrar mediante programas estadísticos el consumo de energía y agua.

1.6.2 Limitaciones.

- La programación del asistente virtual responderá solo a preguntas y comandos que sean técnicos y/o estén dentro del propio diccionario del software del sistema.
- El prototipo posee un sistema no modular, está diseñado para la gestión de un apartamento con número específico de cuatro habitaciones o ambientes, (cocina, sala, baño y dormitorio), con sensores y actuadores determinados y detallados en el documento.
- El sistema necesita una conexión constante a internet para su funcionamiento.
- Para el manejo del sistema se requiere de una previa capacitación.

Capítulo 2

MARCO TEÓRICO.

2 MARCO TEÓRICO.

2.1 CONTENIDO DEL MARCO TEÓRICO.

El contenido del marco teórico en el proyecto está dispuesto en función de los objetivos específicos y actividades.

Tabla 2.1. Contenido de la fundamentación teórica

Objetivos Específicos	Actividades	Fundamentación Teórica
Desarrollar un software inteligente para el sistema que optimice el ahorro de recursos, mejore la seguridad de las edificaciones y aumente el confort de los residentes.	<ul style="list-style-type: none"> ➤ Realizar un estudio de las nuevas tecnologías Smart, internet de las cosas e inteligencia artificial. ➤ Desarrollar el software del sistema de I.A. ➤ Realizar un agente que utilice tecnologías modernas, vigentes y a disposición ➤ Conseguir un sistema con hardware y software a libre y a disponibilidad comercial. 	Elementos, técnicas y métodos de programación y estructura de datos: Se aplica para el desarrollo del software que manejara el sistema de I.A. Base de datos: Se aplica para la implementación de una base de datos en el sistema con el fin de almacenar la información recolectada. Inteligencia Artificial: Se implementa para el desarrollo de un software que gestione de manera eficiente la información y los servicios que brindará el sistema de inteligencia artificial (S.I.A.)
Diseñar y elaborar un hardware robusto para el sistema que logre integrar subsistemas con distintas tecnologías.	<ul style="list-style-type: none"> ➤ Definir los componentes de hardware y software para la implementación del sistema embebido. ➤ Diseñar e implementar el hardware. ➤ Diseñar la red de dispositivos y sensores para la interconexión de los sensores y actuadores con el sistema de I.A. 	Diseño digital: Se aplicará para la selección de dispositivos electrónicos. Control y Automatización: Se considera para el diseño de la estructura domótica que se integra a la red del sistema. Comunicación de datos, Redes y Sistemas operativos: Se aplica en la selección de la comunicación y diseño de la red para los distintos dispositivos conectados al sistema
Conseguir que el sistema sea viable económicaamente.	<ul style="list-style-type: none"> ➤ Realizar un análisis de costos. 	Preparación y evaluación de proyectos: Se considera y aplica para realizar el análisis de costos.
Demostrar la factibilidad del proyecto mediante un prototipo	<ul style="list-style-type: none"> ➤ Realizar un estudio de factibilidad del proyecto ➤ Implementar un prototipo del sistema de inteligencia artificial 	

Fuente: Elaboración propia.

2.2 DOMÓTICA E INMÓTICA.

Ambas palabras podrían definirse de manera general como la automatización de viviendas y edificaciones para satisfacer las necesidades humanas dentro de estas estructuras y mejorar su calidad de vida, brindando servicios como comunicación, seguridad, control y accesibilidad, gestión energética y confort.

La diferencia entre estas dos palabras es el distinto rango que abarcan, la domótica cubre el sector doméstico, mientras que la inmótica es para el sector terciario e industrial (residenciales, hoteles, zonas comunitarias, fábricas, etc.) otro término a mencionar de manera complementaria es la urbótica que abarca las ciudades (gestión de iluminación pública, gestión de semáforos, telecomunicaciones, medios de pago, etc.), en este documento se usará la palabra domótica para generalizar estos conceptos.

La domótica integra diferentes tecnologías, dispositivos y programas capaces de comunicarse entre ellos a través de un soporte de comunicaciones para realizar tareas domésticas que antes se hacían de manera manual.

2.2.1 Tipos de arquitectura.

Especifica el modo en que los diferentes elementos de control del sistema se van a ubicar. Se mencionan las siguientes:

- **Sistema de arquitectura centralizada.**

Existe una matriz central, un único sistema de control (como ser un ordenador) al cual los diferentes sensores, actuadores y otros dispositivos están conectados, si la unidad de control falla entonces se produce una caída de todo el sistema.

La ventaja de estos sistemas es que es relativamente económico, las desventajas es la gran cantidad de cableado que puede aparecer, la poca flexibilidad para expansiones, la dependencia total de los elementos a controlar y las complicadas interfaces hombre-máquina.

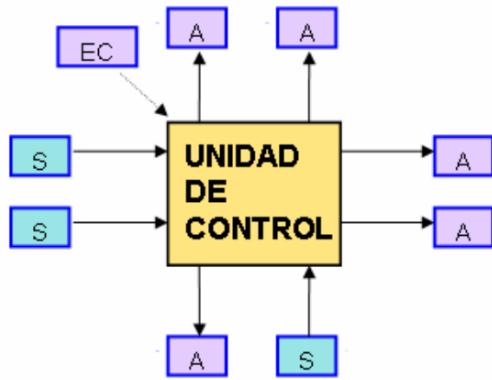


Figura 2.1: Esquema de arquitectura con periferia centralizada. Universidad de Oviedo (2007, p.15).

- **Sistema de arquitectura con periferia descentralizada.**

Este sistema está provisto de una única unidad de control central que gobierna la instalación, pero se dispone de algunos módulos que son capaces de recibir diferentes entradas y salidas (descentralizando la periferia de E/S) y transmitirlas a la unidad de control por medio de un bus. Estos módulos están carentes de toda capacidad de procesamiento y sólo soportan, como es lógico, un hardware de comunicaciones y las conexiones de módulos de E/S (digitales, analógicos, etc.),(Universidad de Oviedo,2007a).

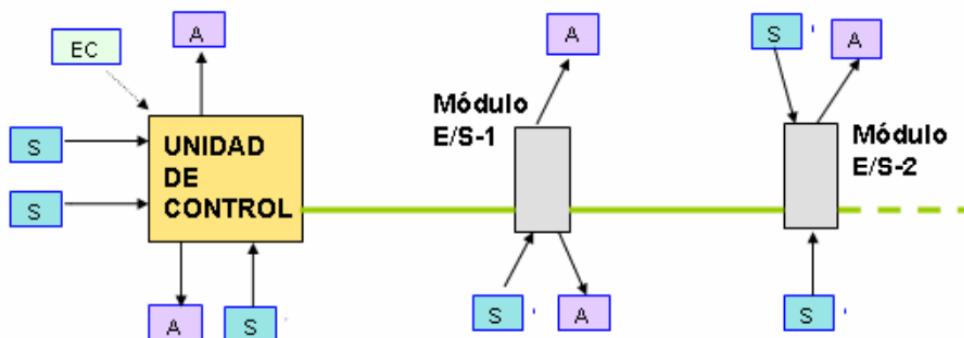


Figura 2.2: Esquema de arquitectura con periferia descentralizada. Universidad de Oviedo (2007, p.17).

Como se aprecia en la figura es bastante habitual que la unidad central también disponga de capacidad para direccionar E/S (conexión a sensores y actuadores) como una arquitectura centralizada. En este caso estamos ante una arquitectura híbrida. La principal ventaja de esta arquitectura es el importante ahorro de cableado que se consigue, así como una racionalización

mayor a la hora de plantear la instalación, teniendo en cuenta zonas, concentración de señales, etc. Tiene la ventaja de los sistemas distribuidos en cuanto a conexión y cableado pero la desventaja de que una caída de la unidad central provoca un fallo general del sistema, (Universidad de Oviedo,2007b).

- **Sistema de arquitectura distribuida.**

Sistema en que la unidad de control se sitúa próxima al elemento a controlar. Existen por tanto generalmente más de una unidad o elemento de control.

Esta arquitectura presenta algunas ventajas respecto a la arquitectura centralizada expuesta anteriormente, pues la tarea del control se reparte convenientemente entre diferentes elementos de control. Esto trae como consecuencia que el cableado se reduzca enormemente. La unión entre las diferentes unidades de control se puede hacer empleando alguno de los medios físicos existentes, y utilizando el protocolo adecuado al elemento de control. En esta arquitectura se permite la interrelación de sensores y actuadores asignados a diferentes elementos de control. Por lo tanto, a diferencia de la arquitectura centralizada, si existe algún fallo en alguna de las unidades de control que conforman la arquitectura distribuida, éste sólo va a afectar a los elementos que tenga unidos a su módulo y por tanto podrá seguir funcionando el sistema lo que lo hace un sistema más robusto frente a fallos, posee un fácil diseño de instalaciones y cumple con todos los requisitos (confort, seguridad, energía y comunicación) de un sistema domótico. La principal desventaja es que las unidades de control son varias y por tanto el coste debería de ser más alto ya que se están multiplicando elementos de control y comunicaciones en las mismas,(Universidad de Oviedo, 2007c).

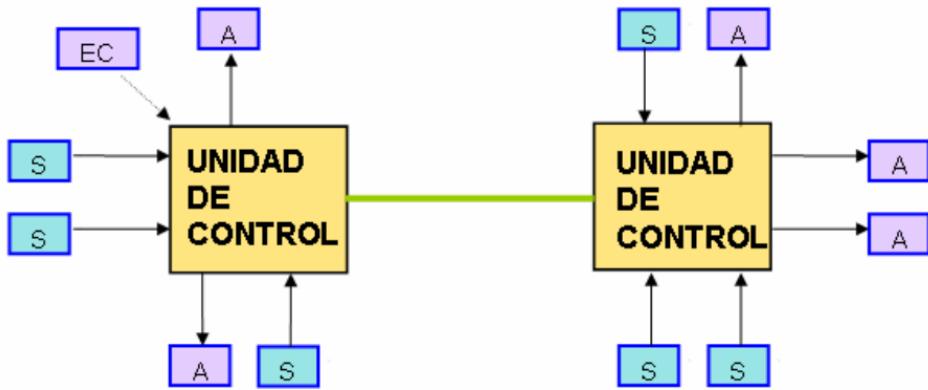


Figura 2.3: Esquema de la arquitectura distribuida. Universidad de Oviedo (2007, p.16).

2.2.2 Servicios y aplicaciones de la domótica.

Como se ha mencionado antes, los cuatro aspectos básicos y fundamentales que abarca la domótica son: confort, comunicación, energía y seguridad.

2.2.2.1 Confort.

El confort incluye los sistemas que contribuyen al bienestar, la comodidad y la reducción de trabajo doméstico, se enfoca principalmente en la climatización e iluminación.

Los sistemas domóticos deben ser acogedores para los residentes, sin embargo, se debe considerar que todos los equipos relacionados al confort consumen mucha energía como ser la calefacción y el aire acondicionado.

a) Iluminación.

Es necesario hacer un uso inteligente de la luz, adaptándolo a las necesidades de los usuarios y aunando, a su vez, un consumo energético lo más eficiente posible.

➤ Escenas de luz.

Gracias al empleo de programas específicos adaptados a cada situación, se pueden realizar escenas de luz, que consistirían en la memorización por parte del sistema de la iluminación que se elija para cada circunstancia de uso, por ejemplo, el control de luminosidad en un ambiente de acuerdo a la hora o a la cantidad de residentes (Universidad de Oviedo, 2007d).

➤ **Iluminación en factores externos.**

El control domótico de la iluminación puede adaptar el accionamiento de ésta dependiendo de variables como pueden ser: Detectores de presencia, detectores de luminosidad, alarmas técnicas, programación horaria.

b) Climatización.

La temperatura a la que se encuentre una habitación incide de gran forma en la actitud y la salud de las personas presentes, lo que lleva como consecuencia inmediata el traslado de esta incidencia a campos tan importantes como el rendimiento en el trabajo.

Suponiendo que estamos hablando de la climatización de un edificio no destinado a vivienda, ya que en un hogar el número de personas es más reducido y las relaciones entre ellas son muy distintas a las de un edificio con otros usos, todos los temas que están relacionados con el calor y las distintas formas de percibirlo tendrán una influencia que variará de una persona a otra.

Como estos factores son muy personales es necesario conocerlos de cerca para, al afrontar el diseño de una instalación de climatización, tenerlos en cuenta y darles una adecuada respuesta técnica.

Al igual que se comentó en el apartado de la iluminación, la climatización presenta la posibilidad de controlarse en función de variables externas como detectores de presencia, termostatos, programaciones horaria o estacional, u otros, (Universidad de Oviedo,2007e).

c) Sistemas de audio y video.

Los sistemas multimedia son fundamentales en las edificaciones para la intercomunicación de los residentes, dispositivos como timbres de voz, pantallas informativas y otros mejoran el confort de los residentes.

2.2.2.2 Comunicación.

En una edificación cabe distinguir la comunicación interior del edificio y la comunicación desde y hacia el exterior.

Las *comunicaciones internas* son las que se generan entre los diferentes dispositivos y sistemas, así como las que utilizan los usuarios con el sistema domótico (interfaces de usuario). La comunicación está íntimamente relacionada con las funcionalidades que se tengan definidas en el edificio puesto que debe dar la posibilidad de explotar el sistema, visualizando aquellos parámetros de interés y permitiendo la entrada de datos.

Se puede citar como comunicaciones internas las que puedan existir entre dispositivos dentro del edificio. Para establecer la comunicación entre elementos del mismo tipo se suelen emplear protocolos estándar o propietarios.

Las *comunicaciones exteriores* en un edificio o vivienda van encaminadas básicamente a campos que son la comunicación, telemetría, seguridad, automatización y entretenimiento.

Entre las posibilidades de comunicación destacan:

- Sistemas de comunicación en el interior.
- Megafonía, difusión de audio y video, intercomunicadores, etc.
- Red de Área Local doméstica (LAN).
- Educación, trabajo y compras a través de las TICs.
- Control de instalaciones domóticas mediante interfaces.
- Mensajes de alarma como fugas de gas, agua, etc.

2.2.2.3 Energía.

La gestión de la energía es de vital importancia en la automatización de las viviendas y los edificios, ya que la implantación de sistemas que estén encaminados a este criterio será bien acogida tanto por los usuarios como por las compañías suministradoras y los propios gobiernos y administraciones públicas.

Una edificación moderna, necesita una gestión que asegure que todos los recursos son empleados de forma eficiente y que se aprovechan las posibilidades que ofrece al máximo.

Para todo ello es necesario un exhaustivo conocimiento de toda la instalación, de forma que puedan ser tomadas las acciones necesarias para garantizar que se cumplen con los objetivos fijados. Los sistemas domóticos ofrecen la posibilidad de dar esa información de múltiples maneras. Con paneles o pantallas en cada zona se puede mostrar el funcionamiento de un área en concreto; centralizado en un PC puede estar monitorizado el edificio entero y con él extraer datos del periodo de tiempo que queramos (por ejemplo, consumo del agua en un determinado día, mes o año), así como almacenarlos para su posterior uso. Todo esto lo podemos hacer de forma directa, con un usuario administrador en el sistema o bien a través de sistemas de comunicación remota como la administración mediante el uso de celulares a través de internet.

La finalidad es satisfacer las necesidades del hogar a un mínimo coste, se pueden distinguir tres aspectos:

- **Regulación:** Mantener una magnitud regulada en función de un valor prefijado con la que se pueda obtener la evolución del consumo energético de la vivienda o edificio, por ejemplo, reducción del consumo en ausencia de individuos.
- **Programación:** Modificar en función del tiempo el nivel de un valor prefijado, como temperatura según horarios, días de la semana, etc.
- **Optimización:** El aprovechamiento de la energía y reducción de su consumo mediante el aprovechamiento de las franjas de tarificación de valle para hacer trabajar a aquellos equipos que lo permitan.

La detección de fuentes de pérdidas en los sistemas de aguas, gas y electricidad, por ejemplo, pérdidas en el sistema de climatización (suspensión de funcionamiento donde se detecten ventanas abiertas).

El uso racional de la energía es uno de los principales objetivos de las compañías eléctricas y de las autoridades. Se trata de que el usuario emplee estrategias orientadas a consumir sólo la energía necesaria evitando el “despilfarro”. Para conseguir este objetivo son necesarias varias condiciones: suministrar información al usuario y utilizar sistemas técnicos que permitan la regulación adecuada de los flujos energéticos. De esto último es de lo que se encarga la domótica/inmótica.

- **Control de accesos.**

Control de accesos a través de este subsistema podemos controlar los accesos al edificio o dentro de él a las zonas deseadas. Quedarse en un mero “portero electrónico”, no aportaría ventajas sobre otros métodos, pero la domótica ofrece la posibilidad de enlazar este subsistema con otros de forma que la gestión del edificio se haga más racional, (Universidad de Oviedo, 2007f).

Es posible encontrarse con las siguientes aplicaciones:

- El uso de la climatización y la iluminación en función del número de personas que hay en un edificio.
- Inicio de procesos asociados a la presencia de determinadas personas en la instalación.
- Cargas de diversos servicios que serán usados por las personas presentes: aparcamiento, alimentación, etc.

2.2.2.4 Seguridad.

Abarca tanto aquellos sistemas destinados a prevenir la intrusión como a las alarmas técnicas que corresponden a peligros derivados del mal funcionamiento de alguno de los sistemas de una edificación. Podemos hacer una división en los siguientes apartados:

- a) **Control de intrusión.**

La posibilidad de la presencia de personas no deseadas en una edificación hace necesaria la instalación de sistemas que prevean esta posibilidad y aporten soluciones eficaces.

La domótica ofrece estas funcionalidades aunándolas al resto de virtudes del sistema. No sólo se tendrá cubierta la gestión de alarmas, además ésta se podrá conectar con el resto del sistema domótico pudiendo conocer en cada instante el estado de la instalación y obtener información tanto local como de forma remota.

El sistema puede a su vez realizar algunas funciones cuando salte alguna alarma, como la conexión intermitente de la iluminación, el accionamiento de sirenas, el envío de señales

por teléfono, cierre de accesos, grabación de imágenes por medio de un circuito cerrado de televisión (CCTV), empleo de Internet, el uso de sensores volumétricos para detección de presencia, sensores magnéticos para la apertura de puertas y ventanas, sensores de hiperfrecuencia para detección de cristales rotos, etc.

b) Alarmas.

Una edificación moderna no puede prescindir de alarmas contra incendios que cubran todas las instalaciones. No solo se realizará una mera detección del fuego/humo, aportara además otros aspectos como son:

- Accionamiento de alarmas, tanto sonoras como visuales.
- Información a los servicios de emergencia.
- Cierre de puertas y elementos que puedan ayudar a la propagación del siniestro.
- Cortes de energía eléctrica.
- Envío de ascensores a la planta baja.

Como en el resto de aplicaciones de la domótica se puede tener un control a distancia y en este caso será el sistema el que informe, vía teléfono, a cualquier usuario que demande este servicio.

Otras funcionalidades están relacionadas con alarmas que puedan producirse por inundación, escapes de gas o fallo en el suministro eléctrico. El sistema típicamente debe detectar la alarma, actuar en consecuencia cortando las válvulas correspondientes y dando aviso al usuario por cualquiera de los métodos elegidos: señalización luminosa, acústica, telefónica, etc.

c) Seguridad personal.

Este aspecto es el encargado de cuidar el bienestar de los usuarios, encargado de realizar tareas como:

- Desactivación de enchufes de corriente para evitar contactos.
- Manipulación a distancia de interruptores en zonas húmedas.
- Detectores de fugas de gas o agua que cierren las válvulas de paso en caso de producirse escapes.
- Alumbrado automático en zonas de riesgo por detección de presencia.
- Alarmas de salud en caso de personas con necesidades especiales (personas de la tercera edad, personas con capacidades especiales) se dispone pulsadores cuya activación genera algún aviso a una central receptora, un familiar o un hospital para solicitar ayuda de urgencia.

2.2.1. Smart City e inteligencia ambiental.

Con la incorporación de sistemas de gestión técnica de instalaciones (clima, iluminación, control de electrodomésticos, alarmas técnicas, seguridad, control del agua, gestión de energía, control de accesos, etc.) los entornos evolucionan hacia sistemas automatizados (vivienda domótica, edificio inmótico, etc.).

Con la adición de nuevas tecnologías para la información y las telecomunicaciones, integrando ocio, multimedia, e-servicios, etc. se convierten en entornos digitales. Por último, con el uso de la inteligencia ambiental (integración sensorial, computación ubicua, sistemas multiagentes, interfaces multimodales, ...) el entorno puede pasar al escalón de “inteligente”.

Cuando se aplican estos conceptos en otros entornos más extensos, por agrupación de las entidades anteriores se puede llegar a la “Ciudad Inteligente” apuntando líneas de investigación de las próximas décadas.

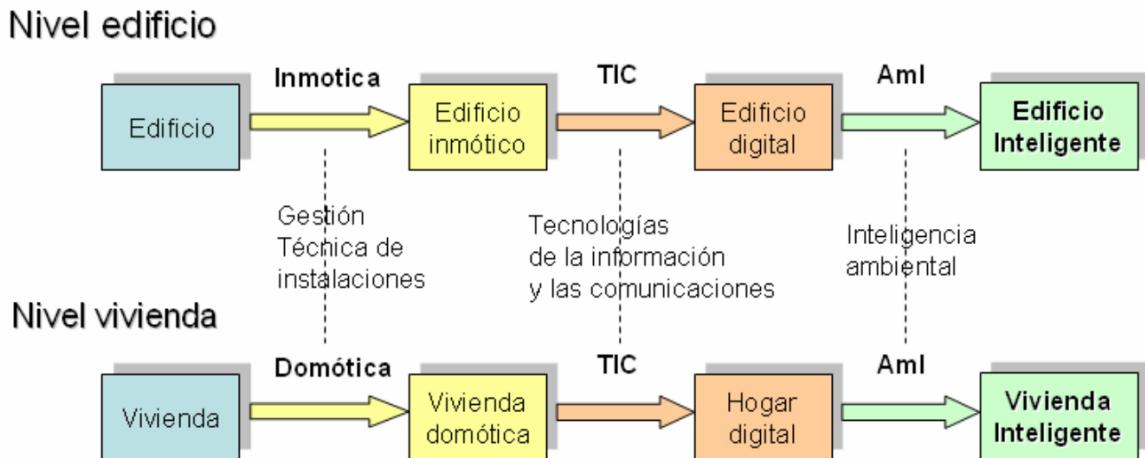


Figura 2.4: Evolución de las estructuras hacia estructuras inteligentes. Universidad de Oviedo (2007, p.44).

2.3 INTELIGENCIA ARTIFICIAL.

- **Inteligencia:** La inteligencia es la capacidad del ser humano de realizar procesos cognitivos, razonar, aprender, tomar decisiones y hacerse una idea determinada de la realidad.

El ser humano se ha autodenominado un “*ser inteligente*”, sin embargo, existe una gran variedad de seres vivos que también poseen inteligencia, pero en menor grado al del ser humano.

- **Artificial:** Cualquier cosa hecha por el ser humano sin la intervención de la naturaleza.

En algún punto de su historia el ser humano sintió la necesidad de comunicarse, para lo cual desarrolló un lenguaje, con el lenguaje desarrolló conceptos de las cosas que lo rodean y así se hizo una descripción del mundo en general a lo que podríamos definir como conocimiento. Pero el conocimiento en sí mismo no hace que el humano sea inteligente, sino la forma en que manipulamos ese conocimiento a su propio favor, analizando, razonando, cuestionando, aprendiendo, resolviendo problemas, sacando conclusiones, etc.

El concepto de la inteligencia artificial es amplio, abarca muchos campos y aún hay puntos muy discutidos y refutados, pero podría decirse que *la Inteligencia Artificial busca la representación en general de la inteligencia humana*.

Hace un tiempo un sistema capaz de realizar operaciones básicas de aritmética ya era considerado un sistema de inteligencia artificial porque ya poseía en cierta manera un grado de inteligencia lógica y cognitiva, sin embargo, ahora las computadoras son capaces de tomar decisiones, resolver problemas y aprender de ellos.

La inteligencia artificial es capaz de representar la inteligencia humana en la magnitud que es posible y que se le ha permitido, hay ciertas facultades que la inteligencia artificial puede representar de manera muy pertinente como evaluar, tomar decisiones, conocer, aprender, actuar, etc.

También existe una ambigüedad en otras facultades en las que el sistema puede representarla de manera objetiva pero no de manera subjetiva, por ejemplo, es posible que una máquina pueda emitir juicios si se le incorpora la constitución y el código penal en su base de datos, pero no puede formar juicios propios sobre si un individuo le es o no una buena persona. En resumen, puede hacer ciertas tareas de manera objetiva siempre y cuando se lo enseñe.

Y finalmente están las facultades humanas que no pueden realizar las máquinas como todo lo relacionado a la inteligencia emocional (emociones y sentimientos).

2.3.1 Tipos de inteligencia artificial.

a) Sistemas que piensan como humanos:

Se refiere a aquellos sistemas programados para razonar, tomar decisiones, resolver problemas, aprender, etc. De la misma manera que lo hace el ser humano.

b) Sistemas que actúan como humanos:

Esto abarca al campo de la robótica, todos aquellos entes programados para realizar tareas que los humanos realizan y hacerlo de manera eficiente.

c) Sistemas que piensan racionalmente:

Intentan emular el pensamiento lógico racional de los humanos, es decir, se investiga cómo lograr que las máquinas puedan percibir, razonar y actuar en consecuencia.

Los sistemas expertos se engloban en este grupo e intentan alcanzar el mejor resultado, esto no implica que el ser humano no sea racional, pero el humano no es perfecto, por ejemplo, no todo ser humano es el mejor jugador de ajedrez, pero una máquina con los algoritmos adecuados puede estar a un nivel incluso mayor que los mejores jugadores del mundo.

d) Sistemas que actúan racionalmente:

Son aquellos que pueden desempeñar sus trabajos de forma racional y eficiente, por ejemplo, el ser humano no es capaz de congelar cosas, pero un refrigerador si, y debe desempeñar su trabajo de manera que no deje descongelar los helados, a la vez que prevea que las verduras no se congelen y tampoco haga funcionar el motor más de lo debido, porque eso sería una ineficiencia en ahorro energético, este es la cualidad sobre la que avanza la inteligencia artificial, donde los actores en esta área son los agentes racionales.

2.3.2 Agente racional.

Son todos aquellos sistemas que actúan con la intención de alcanzar el mejor resultado o, cuando hay incertidumbre el mejor resultado esperado.

Para situaciones en las que el tiempo no es esencial, se necesitan respuestas reflejas, mientras que la deliberación basada en el conocimiento permite que el agente planifique con antelación. Un agente completo debe ser capaz de hacer las dos cosas, utilizando una *arquitectura híbrida*. Según Russell & Norvig (2004):

Un **agente** es cualquier cosa capaz de percibir su **medio ambiente** con la ayuda de **sensores** y actuar en ese medio utilizando **actuadores**.

Un agente humano tiene ojos, oídos y otros órganos sensoriales además de manos, piernas, boca y otras partes del cuerpo para actuar. Un agente robot recibe pulsaciones del teclado, archivos de información y paquetes vía red a modo de entradas sensoriales y actúa sobre el medio con mensajes en el monitor, escribiendo ficheros y enviando paquetes por la red. Se trabajará con la hipótesis general de que cada agente puede percibir sus propias acciones (pero no siempre sus efectos). (p.38).

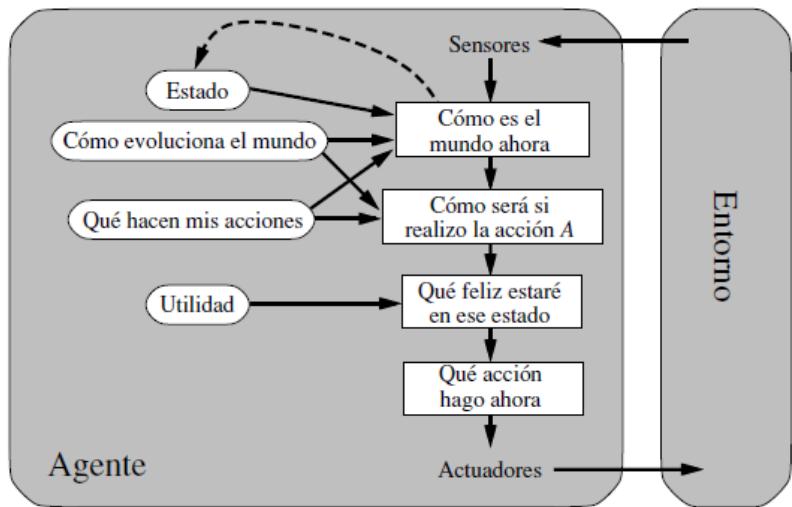


Figura 2.5: Diagrama de un agente basado en utilidades y modelos. Russell et al (2004,p.59).

2.3.3 El futuro de la inteligencia artificial.

Según Russell et al (2004):

La inteligencia artificial (IA) tienen como propósito general crear un agente racional con la capacidad de reflejos y alguna capacidad deliberativa, una variedad de formas de conocimiento y toma de decisiones, mecanismos de aprendizaje y de compilación para todas esas formas, métodos de controlar el razonamiento, y un gran almacén de conocimientos específicos del dominio. (p. 1105).

La IA ha buscado y considerado cuatro especificaciones para el desarrollo de sus agentes: la racionalidad perfecta, la racionalidad calculadora, la racionalidad limitada y la optimalidad limitada, pero solo la última mencionada es la que parece ofrecer la mejor esperanza de conseguir un fundamento teórico fuerte para la IA.

La optimalidad limitada es “la más realista” debido a que son realmente útiles en el mundo real, esta consiste en desempeñar una máxima eficiencia y utilidad con los recursos que se le otorga, se propone un concepto de optimalidad limitada como una tarea formal para la investigación de la IA que esté bien definida y que sea viable. La optimalidad limitada especifica *programas* óptimos en vez de *acciones* óptimas. Las acciones, después de todo, son generadas por programas, y es sobre los programas en donde tienen control los diseñadores.

Teniendo claro que se busca y sobre qué camino se avanza se puede indagar las áreas sobre las cuales la IA tiene un vasto campo de desarrollo a un futuro cercano, si bien la IA es aplicable a casi cualquier área en general, considerando la tecnología actual y otras áreas en desarrollo se reducirían a estos ámbitos:

- Comunicaciones (TICs)
- Robótica,
- Internet de las cosas
- Asistentes Virtuales
- Cibernética
- Ciudades Inteligentes

Existen temas éticos a tener en cuenta, como se cuestionaba Russell et al(2004).

Los computadores inteligentes son más potentes, pero ¿se va a utilizar ese poder para fines buenos o malos? Aquellos que se esfuerzan por desarrollar la IA tienen la responsabilidad de ver que el impacto de su trabajo es positivo. El alcance de este impacto dependerá del grado de éxito de la IA. Y ya se han obtenido éxitos modestos que han cambiado las formas de enseñar la informática (Stein, 2002) y las formas en que se practica el desarrollo del *software*. La IA ha hecho posibles aplicaciones nuevas tales como los sistemas de reconocimiento de voz, sistemas de control de inventarios, sistemas de vigilancia, robots y motores de búsqueda. Finalmente parece probable que un éxito en IA a gran escala, la creación de inteligencia en el nivel humano y más allá, cambiaría las vidas a una mayoría de la humanidad. La verdadera naturaleza de nuestro trabajo y de nuestro papel cambiaría, así como nuestro punto de vista de la inteligencia, la conciencia y el destino futuro de la raza humana. A este nivel, los sistemas IA podrían suponer una amenaza directa a la autonomía humana, la libertad, e incluso la supervivencia. Por estas razones, la investigación en IA no se puede divorciar de sus consecuencias éticas. (p.1106)

2.4 ASISTENTES VIRTUALES.

Un asistente virtual es un agente de software que ayuda a usuarios de sistemas computacionales, automatizando y realizando tareas con la mínima interacción hombre-máquina. La interacción que se da entre un asistente virtual y una persona, debe ser natural,

una persona se comunica usando la voz y el asistente virtual lo procesa, interpreta y responde de la misma manera.

2.4.1 Asistente personal inteligente.

Es un agente tipo software que puede realizar tareas u ofrecer servicios a un individuo. Estas tareas o servicios están basados en datos de entrada de usuario, reconocimiento de ubicación y la habilidad de acceder a información de una variedad de recursos en línea (como al clima o al tráfico, noticias, precios de acciones, horario del usuario, precios al por menor, etc.).

2.4.2 Servicios.

Los asistentes virtuales actualmente pueden proporcionar una amplia variedad de servicios, mencionado algunos:

- Proveer información sobre el tiempo, datos de Wikipedia o IMDb (Internet Movie Database), ajustar alarmas, listas de pendientes o de compras
- Reproducir música de servicios de *streaming* como Spotify y Pandora; reproducir estaciones de radio; leer audiolibros
- Reproducir videos, programas de televisión o películas en televisores, de fuentes como por ejemplo Netflix
- Comprar artículos, como por ejemplo desde Amazon
- Complementar y/o reemplazar servicios de atención al cliente por humanos.¹⁸ Un reporte estimó que un asistente automático en línea produjo una reducción de 30% en la carga de trabajo de un centro de llamadas.

Entre los principales asistentes virtuales que predominan esta área y están en auge tenemos a *Alexa* de Amazon, *Asistente de Google*, *Cortana* de Microsoft y *Siri* de Apple, cabe resaltar que solo Alexa y Asistente de Google son orientados a IoT.

El informe de asistentes virtuales del centro de tiflotécnica e innovación ONCE (2019) indica:

Al margen de cualquier valoración u opinión subjetiva, la empresa de investigación Loup Ventures, ha publicado un estudio comparativo de los cuatro principales asistentes virtuales, basado en una batería de 800 preguntas. Este estudio, realizado en el último trimestre de 2018, arroja el siguiente resultado:

Tabla 2.2. Comparación de asistentes con relación a la eficiencia de respuestas.

	Respuesta Correcta	Valoración de la pregunta
Asistente de Google	87,9%	100%
Siri	74,6%	99,6%
Alexa	72,5%	99,0%
Cortana	63,4%	99,4%

Nota: Información extraída del informe de asistentes virtuales del centro de tiflotécnica e innovación Once.

Tan solo un año atrás, en un estudio igual llevado a cabo por la misma empresa, el Asistente de Google pudo contestar correctamente al 81% de las preguntas, Siri el 52%, Alexa el 64% y Cortana el 56%.

Los resultados, por lo tanto, no han hecho más que confirmar la tendencia que, a tenor de los datos analizados en el presente documento hacían intuir: el Asistente de Google es el asistente virtual de mayor nivel de comprensión de las preguntas y aquél que ha podido contestar correctamente a la mayor cantidad de preguntas. (p.19).

2.5 RECONOCIMIENTO DE VOZ.

El *reconocimiento automático del habla* (RAH) o *reconocimiento automático de voz* es una disciplina de la inteligencia artificial que tiene como objetivo permitir la comunicación hablada entre seres humanos y computadoras. El problema que se plantea en un sistema de este tipo es el de hacer cooperar un conjunto de informaciones que provienen de diversas fuentes de conocimiento (acústica, fonética, fonológica, léxica, sintáctica, semántica y pragmática), en presencia de ambigüedades, incertidumbres y errores inevitables para llegar a obtener una interpretación aceptable del mensaje acústico recibido.

Un sistema de reconocimiento de voz es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En su desarrollo intervienen diversas disciplinas, tales como: la fisiología, la acústica, la lingüística, el procesamiento de señales, la inteligencia artificial y la ciencia de la computación. (Wikipedia, s.f.).

2.5.1 Clasificación.

Los sistemas de reconocimiento de voz pueden clasificarse según los siguientes criterios:

- **Entrenabilidad:** determina si el sistema necesita un entrenamiento previo antes de empezar a usarse.
- **Dependencia del hablante:** determina si el sistema debe entrenarse para cada usuario o es independiente del hablante.
- **Continuidad:** determina si el sistema puede reconocer habla continua o el usuario debe hacer pausas entre palabra y palabra.
- **Robustez:** determina si el sistema está diseñado para usarse con señales poco ruidosas o, por el contrario, puede funcionar aceptablemente en condiciones ruidosas, ya sea ruido de fondo, ruido procedente del canal o la presencia de voces de otras personas.
- **Tamaño del dominio:** determina si el sistema está diseñado para reconocer lenguaje de un dominio reducido (unos cientos de palabras ejemplo: reservas de vuelos o peticiones de información meteorológica) o extenso (miles de palabras). (Wikipedia, s.f.).

2.5.2 Aplicaciones.

Aunque en teoría cualquier tarea en la que se interactúe con un ordenador puede utilizar el reconocimiento de voz, actualmente las siguientes aplicaciones son las más comunes:

- **Dictado automático:** El dictado automático es, hasta hoy, el uso más común de las tecnologías de reconocimiento de voz. En algunos casos, como en el dictado de recetas médicas y diagnósticos o el dictado de textos legales, se usan corpus especiales para incrementar la precisión del sistema.
- **Control por comandos:** Los sistemas de reconocimiento de habla diseñados para dar órdenes a un computador ("Abrir Firefox", "cerrar ventana") se llaman Control por

comandos. Estos sistemas reconocen un vocabulario muy reducido, lo que incrementa su rendimiento.

- **Telefonía:** Algunos sistemas PBX permiten a los usuarios ejecutar comandos mediante el habla, en lugar de pulsar tonos. En muchos casos se pide al usuario que diga un número para navegar un menú.
- **Sistemas portátiles:** Los sistemas portátiles de tamaño reducido, como los relojes o los teléfonos móviles, tienen unas restricciones muy concretas de tamaño y forma, así que el habla es una solución natural para introducir datos en estos dispositivos.
- **Sistemas diseñados para discapacitados:** Los sistemas de reconocimiento de voz pueden ser útiles para personas con discapacidades que les impidan teclear con fluidez, así como para personas con problemas auditivos, que pueden usarlos para obtener texto escrito a partir del habla. Esto permitiría, por ejemplo, que los aquejados de sordera pudieran recibir llamadas telefónicas. (Wikipedia, s.f.).

2.6 RASPBERRY.

Raspberry Pi es un microcomputador de placa reducida, totalmente funcional y de bajo coste desarrollado en el Reino unido por la Raspberry Pi Foundation, con el objetivo de estimular la enseñanza de informática en las escuelas, sin embargo, también se han adoptado en oficinas, hogares, fábricas, centros de datos, etc.

Aunque no se indica expresamente si es hardware libre o con derechos de marca, en su web oficial explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo cualquiera puede convertirse en revendedor o distribuidor de las tarjetas Raspberry pi, por lo que da a entender que es un producto con propiedad registrada, manteniendo el control de la plataforma, pero permitiendo su uso libre tanto a nivel educativo como particular.

Wikipedia (s.f.).

En cambio, el software sí es de código abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos,

incluido una versión de Windows 10. En todas sus versiones, incluye un procesador Broadcom, memoria RAM, GPU, puertos USB, HDMI, Ethernet (el primer modelo no lo tenía), 40 pines GPIO (desde la Raspberry Pi 2) y un conector para cámara. Ninguna de sus ediciones incluye memoria, siendo esta en su primera versión una tarjeta SD y en ediciones posteriores una tarjeta MicroSD. Wikipedia (s.f.).

La fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora) y promueve principalmente el aprendizaje del lenguaje de programación Python. Otros lenguajes también soportados son Tiny BASIC, C, Perl y Ruby. Wikipedia (s.f.).

2.6.1 Microcomputadora.

La definición según un informe de la universidad politécnica de Madrid:

Es una computadora cuya unidad central es un microprocesador. Una microcomputadora es un sistema completo que lleva, además del microprocesador, una memoria y controladores de entrada/salida para conectarlo con periféricos exteriores todo en un mismo circuito integrado. (Saez, 1987, p.3).

Estas microcomputadoras tienen unidades de procesamiento y de almacenamiento, unidades de salida de visualización y de salida de audio, un teclado y todo ello puede colocarse sobre una mesa en el hogar o bien en la oficina.

2.6.2 Otras computadoras de placa reducida.

La **Raspberry Pi** es el mini ordenador más conocido del mundo. Con ella, los entusiastas con ganas, pueden desarrollar muchas utilidades, pero que sea el mini ordenador más famoso, no significa que sea el único microcomputador que los usuarios tengan a su alcance, en la siguiente tabla se verá otras alternativas y comparaciones con Raspberry pi 3B.

Tabla 2.3. Tabla comparativa de microcomputadoras con placa reducida.

	Raspberry pi 3 B+	ASUS Tinker Board	ODROID-C2	La frite
CPU	Broadcom BCM2837B0, Quadcore Cortex-A53 de 64 bits a 1,4 GH	Rockchip Quad Core RK3288 Quad core ARM Cortex-A17 a 1,8 GHz	PU Amlogic S905 (4 núcleos, 2 GHz)	Quad 64bit ARM Cortex-A53 CPU Cores a 1.2GHz
RAM	1 GB LPDDR2	2 GB DDR3	2 GB DDR3	512 MB o 1GB DDR4
LAN	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet	100 MB Fast Ethernet
WIFI	dual 2,4 GHz y 5 GHz. 802.11 b/g/n/ac	802.11 b/g/n	no	no
Bluetooth	4.2	4.0	no	no
GPIO	40 pines	40 pines	40 pines	40 pines
USB	4*USB 2.0	4*USB 2.0	4*USB 2.0	USB 2.0 Host + USB 2.0 OTG
HDMI	Tamaño Estándar	Tamaño Estándar	2.0 4K/60Hz	1.4 con 1080 output

Fuente: Elaboración propia.

2.7 ARDUINO.

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea. Utiliza el lenguaje de programación Arduino (basado en Wiring) y el Software Arduino (IDE) , basado en Processing.(Arduino, s.f.).

Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y el software también es de código abierto.(Arduino, s.f.).

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios.

Los diseños de las placas Arduino usan diversos **microcontroladores** y **microporcesadores**. Generalmente el hardware consiste de un microcontrolador Atmel AVR, conectado bajo la configuración de "sistema mínimo" sobre una placa de circuito impreso a la que se le pueden conectar placas de expansión (shields) a través de la disposición de los puertos de entrada y salida presentes en la placa seleccionada. Las shields complementan la funcionalidad del modelo de placa empleada, agregando circuitería, sensores y módulos de comunicación externos a la placa original. La mayoría de las placas Arduino pueden ser energizadas por un puerto USB o un puerto barrel Jack de 2.5mm. La mayoría de las placas Arduino pueden ser programadas a través del puerto serie que incorporan haciendo uso del Bootloader que traen programado por defecto. Las placas Arduino se programan mediante un computador, usando comunicación serie. (Wikipedia, s.f.).

2.7.1 Microcontroladores.

Un *microcontrolador* (abreviado **μC**, **UC** o **MCU**) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida

2.7.2 Microporcesadores.

Según Daniels (2011):

Los microporcesadores son circuitos integrados que contienen millones de transistores en su interior, los cuales crean circuitos complejos encargados de realizar diferentes tareas. También se los denomina *unidad de procesamiento central o CPU*, ya que muchos de ellos pueden actuar como el “cerebro” de un sistema computacional, administrando todas las tareas que este realice y llevando a cabo las operaciones con los datos.

Los microporcesadores están diseñados para interpretar y ejecutar las instrucciones que nosotros les indiquemos y que suelen ser operaciones simples, como sumar, restar, multiplicar y dividir. Pero también existen instrucciones lógicas, como **AND**, **OR**, **NOT**, entre otras. El listado de instrucciones recibe el nombre de *programa*, que las ejecuta una por una por medio del microporcesador.(pp.92,91).

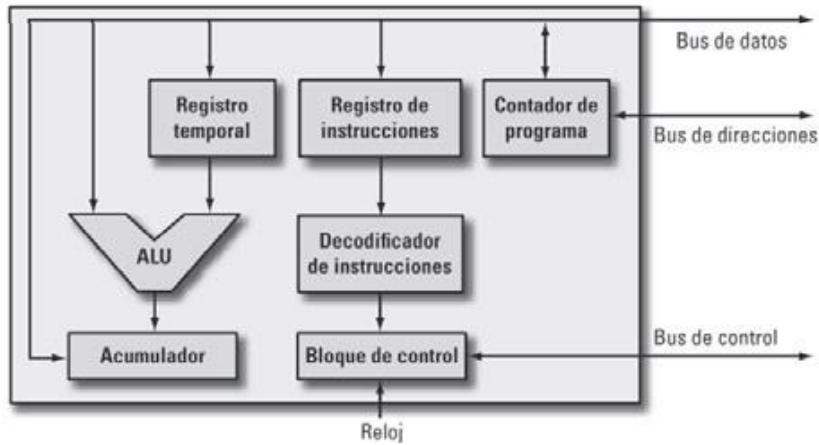


Figura 2.6: Diagrama básico de un microprocesador. Daniels (2011).

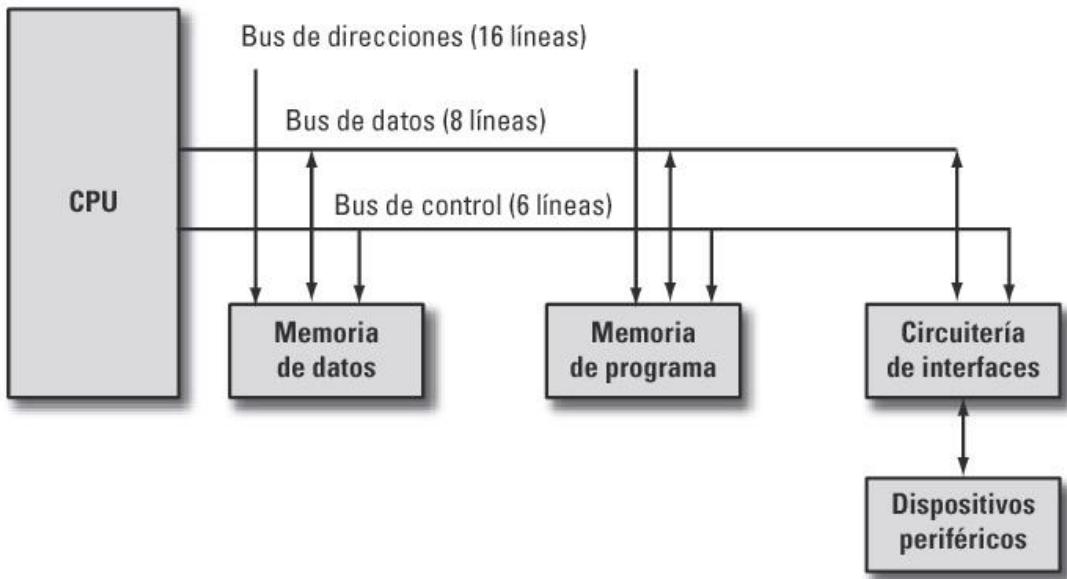


Figura 2.7: Diagrama en bloque de un sistema de computadora completo, con las memorias de datos y de programa como elementos externos al microprocesador. Daniels (2011).

2.7.2.1 Memoria.

La *memoria* es el dispositivo que retiene, memoriza o almacena datos informáticos durante algún período de tiempo. La memoria proporciona una de las principales funciones de la computación moderna: el almacenamiento de información y conocimiento. Es uno de los componentes fundamentales de la computadora, que interconectada a la unidad central de procesamiento (CPU) y los dispositivos entrada/salida.

2.7.2.2 Memoria de programa.

La memoria de programa podría explicarse de la siguiente manera:

Para realizar una tarea específica, un microprocesador necesita de un programa que le indique, instrucción por instrucción, cuáles son los pasos que debe cumplir. Este programa reside en una memoria externa al procesador llamada **memoria de programa**. Su característica principal es que no debe perder su contenido cuando el sistema carece de energía. (Daniels, 2011, p.96).

2.7.2.3 Memoria de datos.

Según Daniels (2011):

La **memoria de datos** es también una memoria externa al microprocesador, pero que se encarga de almacenar la información que precisa el procesador para ejecutar las operaciones que le indiquemos. El tipo de memoria que se emplea para los datos es la **RAM**, porque puede almacenar datos temporales que pueden ser escritos y leídos una infinidad de veces (p.96).

2.7.2.4 Unidades de entrada y salida.

Para que un procesador pueda comunicarse con el mundo externo necesita de unidades de entrada y de salida que codifiquen los mensajes para interpretarlos. A las unidades que funcionan como interfaz entre el mundo externo y el procesador se las llama periféricos. Los periféricos se comunican con el procesador mediante los buses de dirección, de datos y las señales de control. Existe dos formas de transmitir información entre un periférico externo y el procesador: en **paralelo** y en **serie**.

2.8 SENSORES Y ACTUADORES.

Los sensores y actuadores juegan un papel importante en la domótica, el desarrollo de la inteligencia artificial y otras áreas importantes de la electrónica, debido a que es equiparable a los sentidos y acciones de los seres vivos, además de que son elementos importantes para el control, obtención de información y desempeño de tareas en la industria.

2.8.1 Sensor.

Es un dispositivo con la capacidad de percibir magnitudes físicas, químicas (luz, temperatura, sonido, gases, etc.) u otras alteraciones de su entorno que varía sus propiedades con la

variación de dichas magnitudes físicas o alteraciones, el sensor traduce estas variaciones a otra magnitud que facilita las mediciones y la obtención de datos.

2.8.1.1 Características generales.

- **Rango de medida:** dominio en la magnitud medida en el que puede aplicarse el sensor.
- **Precisión:** es el error de medida máximo esperado.
- **Offset o desviación de cero:** valor de la variable de salida cuando la variable de entrada es nula. Si el rango de medida no llega a valores nulos de la variable de entrada, habitualmente se establece otro punto de referencia para definir el *offset*.
- **Linealidad o Correlación lineal:** relación entre valor entregado y valor representado.
- **Sensibilidad de un sensor:** Indica la mayor o menor variación de la señal de salida por unidad de la magnitud de entrada, cuanto mayor sea la variación de la señal de salida producida por una variación en la señal de entrada, el sensor es más sensible.
- **Resolución:** mínima variación de la magnitud de entrada que puede detectarse a la salida.
- **Rapidez de respuesta:** puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- **Derivas:** son otras magnitudes, aparte de la medida como magnitud de entrada, que influyen en la variable de salida. Por ejemplo, pueden ser condiciones ambientales, como la humedad, la temperatura u otras como el envejecimiento (oxidación, desgaste, etc.) del sensor.
- **Repetitividad:** error esperado al repetir varias veces la misma medida.

2.8.1.2 Clasificación.

Según el tipo de salida que proporcionan:

- **Analógicos:** Entregan una salida de nivel variable en función del parámetro que midan, por ejemplo, un sensor de temperatura de -20° a +50° con salida 0-10V.

- **Binarios:** Entregan un nivel ‘todo’ o ‘nada’ (1/0), por ejemplo, el estado de una puerta (abierta/cerrada).
- **Digitales:** Dan la información relativa a la medida con un protocolo de comunicaciones específico que el fabricante facilita: por ejemplo, el sensor de temperatura y humedad STH-11.

Según su estructura interna, tipo de sensor:

- **Pasivos:** No precisan de alimentación: Resistencias que cambian de valor según luz o temperatura.
- **Activos:** Tienen circuitos electrónicos que alimentar y necesitan una fuente de energía.

Según el tipo de parámetros que son capaces de detectar:

- **Mecánicos:** Detectan parámetros relacionados con acciones mecánicas, contactos, aceleración, etc.
- **Ambientales:** Medidas de temperatura, humedad, pluviometría, velocidad del viento, etc.
- **Químicos:** Niveles de CO₂, niveles de oxígeno, contaminación en el aire, azúcar en sangre, etc.

2.8.1.3 Tipos de sensores.

En la tabla 2.4 se indican algunos tipos y ejemplos de sensores.

Tabla 2.4. Tipos y ejemplos de sensores electrónicos.

Magnitud	Transductor	Característica
Posición lineal y angular	Potenciómetro	Analógica
	Encoder	Digital
	Sensor Hall	Digital
Desplazamiento y deformación	Galga extensiométrica	Analógica
	MagnetoestRICTIVOS	A/D
	Magnetoresistivos	Analógica
	LVDT	Analógica
Velocidad lineal y angular	Dinamo tacometrIca	Analógica
	Encoder	Digital
	Detector inductivo	Digital
	Servo-inclinómetros	A/D
	RVDT	Analógica
	Giróscopo	
Aceleración	Acelerómetro	Analógico
	Servo-acelerómetros	
Fuerza y par (deformación)	Galga extensiométrica	Analógico
	Sensor de fuerza	Analógico
	Sensor de par	Analógico
	Multicomponente	Analógico
Presión	Membranas	Analógica
	Piezoelectricos	Analógica
	Manómetros Digitales	Digital
Caudal	Turbina	Analógica
	Magnético	Analógica
Temperatura	Termopar	Analógica
	RTD	Analógica
	Termistor NTC	Analógica
	Termistor PTC	Analógica
	Bimetal - Termostato	I/O
Sensores de presencia	Inductivos	I/O
	Capacitivos	I/O
	Ópticos	I/O y Analógica
Sensores táctiles	Matriz de contactos	I/O
	Piel artificial	Analógica
Visión artificial	Cámaras de video	Procesamiento digital
	Cámaras CCD o CMOS	Procesamiento digital
Sensor de Proximidad	Sensor final de carrera	
	Sensor capacitivo	Analógica
	Sensor inductivo	Analógica
	Sensor fotoeléctrico	Analógica
Sensor acústico (presión sonora)	micrófono	Analógica
Sensores de acidez	ISFET	

Sensor de luz	fotodiode	Analógica
	Fotorresistencia	Analógica
	Fototransistor	Analógica
	Célula fotoeléctrica	Analógica
Sensores captura de movimiento	Sensores inerciales	

Nota: La información de esta tabla pertenece al sitio: <https://es.wikipedia.org/wiki/Sensor>

2.8.2 Actuador.

Un actuador es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover o “actuar” otro dispositivo mecánico. La fuerza que provoca el actuador proviene de tres fuentes posibles: Presión neumática, presión hidráulica, y fuerza motriz eléctrica. Dependiendo del origen de la fuerza el actuador se denomina “neumático”, “hidráulico” o “eléctrico”.

2.8.2.1 Clasificación.

Por lo regular los actuadores se clasifican en dos grandes grupos:

- **Por el tipo de energía utilizada:** Actuador neumático, hidráulico o eléctrico.
- **Por el tipo de movimiento que generan:** Lineal o rotatorio.

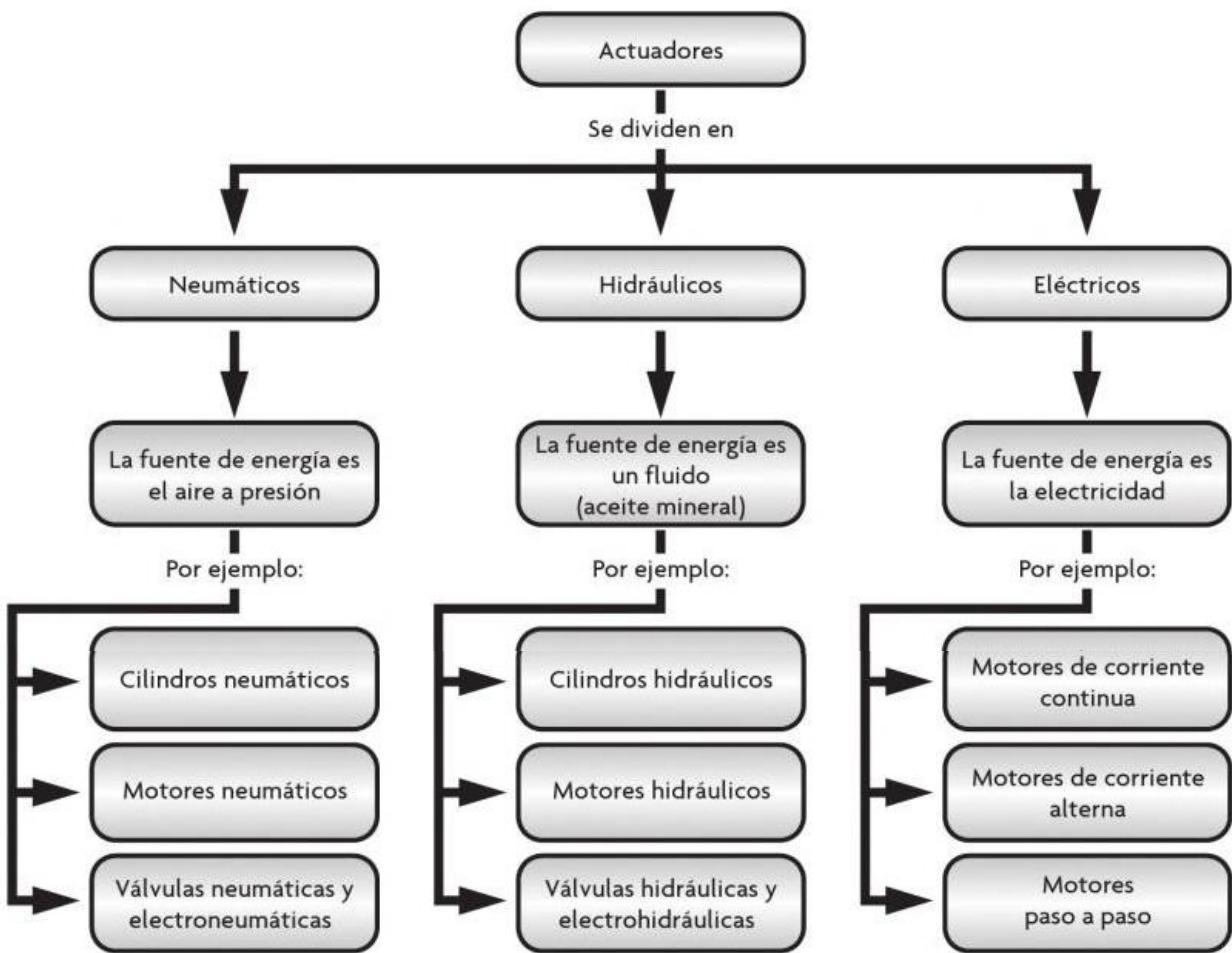


Figura 2.8: Esquema de clasificación de actuadores. Corona et al. (2014, p.26).

Capítulo 3

INGENIERÍA Y DISEÑO DEL SISTEMA ELECTRÓNICO

3 INGENIERÍA Y DISEÑO DEL SISTEMA ELECTRÓNICO.

3.1 ANÁLISIS DE REQUERIMIENTOS.

Se analiza los requerimientos de materiales (sensores, actuadores y otros dispositivos electrónicos) para un prototipo de sistema electrónico para un domicilio particular, considerando que este cuenta con una sala, un dormitorio, un baño y una cocina.



Figura 3.1: Plano domicilio particular donde se instalaría el sistema de inteligencia artificial.

Fuente: Elaboración propia.

PLANO DE INSTALACION

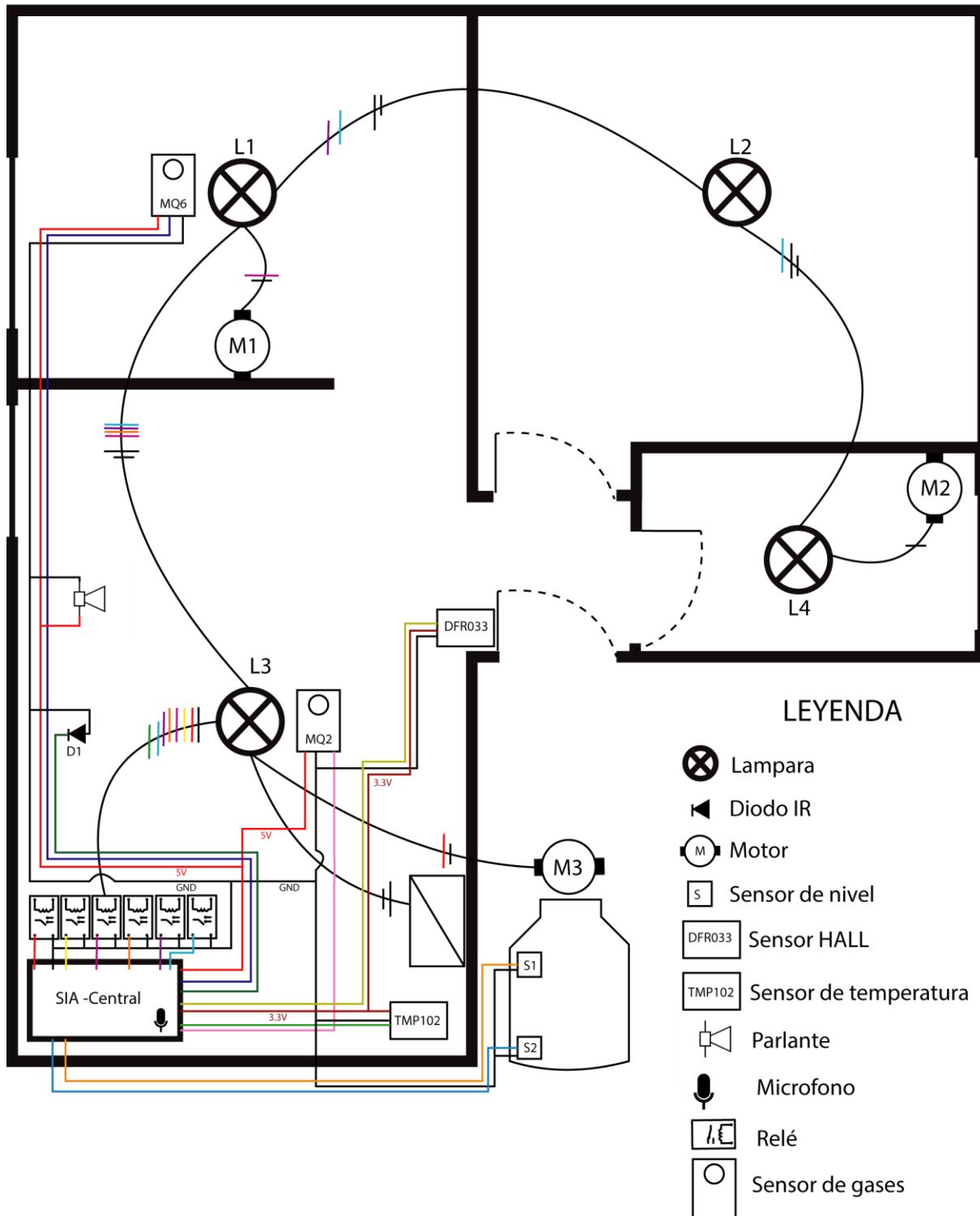


Figura 3.2: Plano de instalacion del sistema SIA en el domicilio particular.

Fuente: Elaboración propia.

3.2 DIAGRAMA DE BLOQUES DEL BÁSICO DEL SISTEMA.

La figura 3.2 muestra las interacciones que realizan tanto los sensores como los usuarios con el microcomputador.

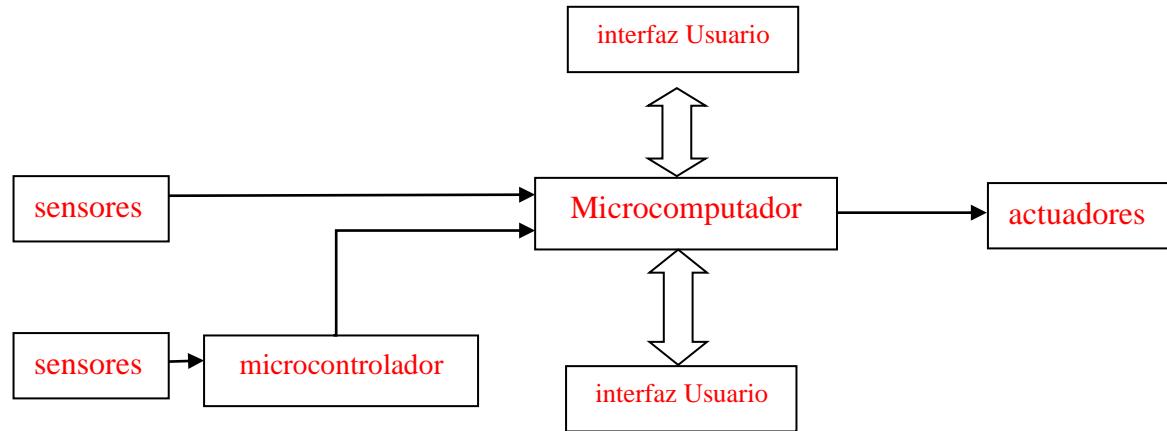


Figura 3.3: Diagrama de bloques del funcionamiento básico del sistema.

Fuente: Elaboración propia.

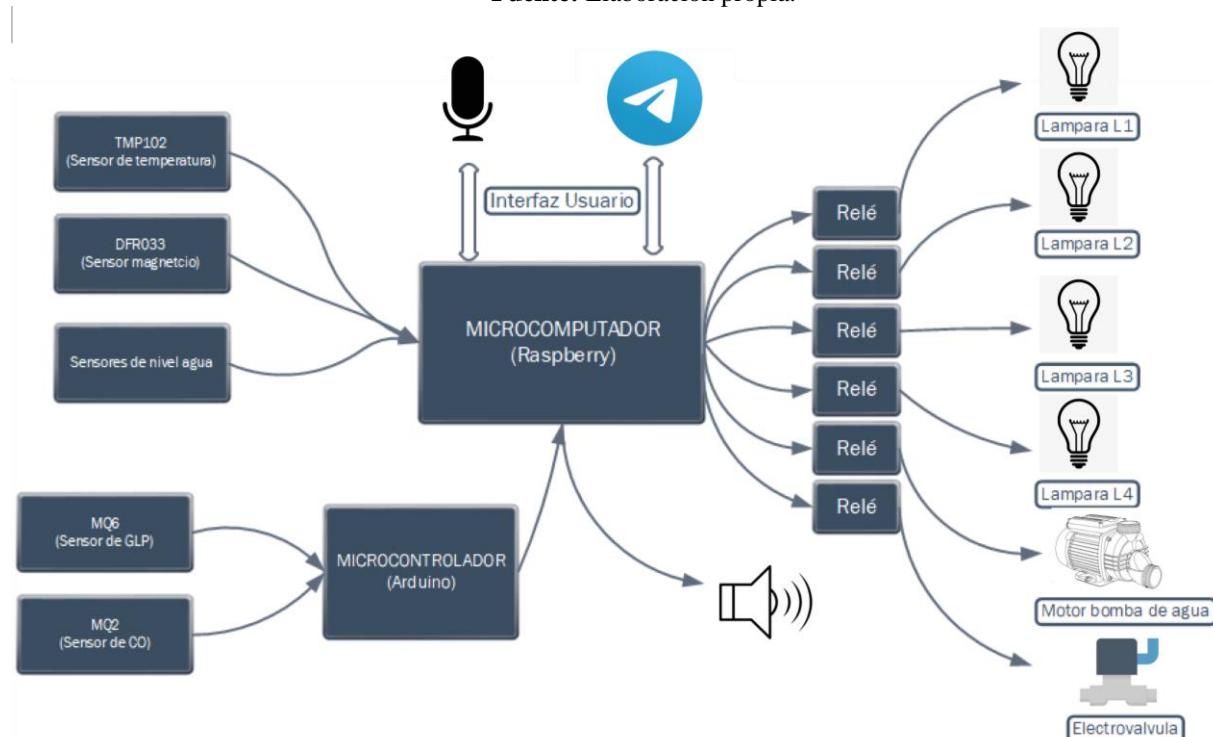


Figura 3.4: Diagrama general de bloques del funcionamiento del sistema.

Fuente: Elaboración propia.

3.3 SELECCIÓN DEL SISTEMA EMBEBIDO.

3.3.1 Selección de la estructura del sistema domótico.

Preponderando el factor económico se utiliza como matriz central un único ordenador definiendo así la estructura de un sistema domótico centralizado.

3.3.2 Selección del microcomputador.

Se realizó un análisis previo de los microcomputadores existentes y disponibles y se decidió que en el proyecto se trabajará con Raspberry pi 4 con 4Gb de RAM por los siguientes motivos:

- Es un microcomputador que posee una expansión de 40 pines digitales como interfaces de entrada y salida, suficientes para interactuar con los distintos sensores y actuadores que se usan en el prototipo, además de que maneja distintos protocolos de comunicación necesarios para interactuar con los dispositivos implementados.
- El microcomputador tiene el tamaño de una tarjeta de crédito, lo que lo hace un producto versátil y eficiente en ahorro de espacio a comparación de otros computadores.
- Es de muy bajo costo comparando con otras microcomputadoras y cumple por demás con los requerimientos necesarios para almacenar el software del sistema (almacenamiento, memoria, red, protocolos de comunicación) .
- Tiene una gran comunidad de desarrolladores y de soporte técnico a nivel mundial que lo respaldan, lo que conlleva al desarrollo de más librerías de software para su uso en distintos proyectos y una constante actualización de las falencias en el sistema.
- Su sistema operativo principal (Raspbian) es una distribución de Linux amigable con el usuario, y que incluye muchas aplicaciones orientadas a la educación y al desarrollo de software.
- Python, que es el lenguaje principal usado en la Raspberry, es fácil de aprender y muy poderoso.

Tabla 3.1. Ficha técnica del microcomputador empleado.

Tarjeta	Raspberry pi 4
Sistema Operativo	Raspbian (raspberrypi 4.19.97-v7l+)
Procesador	Broadcom BCM2711 de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
Almacenamiento	SD card Kingston 16 Gb
RAM	3.8 Gb
Conectividad	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
GPU	VideoCore VI
Expansión	Cabezal GPIO de 40 pines

Fuente: Elaboración propia.

3.3.3 Selección de sensores, actuadores, interfaces de entrada y salida, microcontroladores y otros componentes.

Para este proyecto se seleccionó los siguientes componentes debido a su costo, disponibilidad en el mercado nacional y compatibilidad con la tarjeta Raspberry Pi.

Tabla 3.2. Lista de componentes seleccionados para el prototipo del sistema de inteligencia artificial.

Nombre	Características	Descripción Justificativa	Cantidad
Sensor MQ6	5VDC	Sensor analógico para la detección de monóxido de carbono (CO)	1
Sensor MQ2	5VDC	Sensor analógico para la detección de Gas Licuado Particular (GLP)	1
Sensor DFR0033	3.3VDC	Sensor digital magnético de efecto Hall	1
Sensor TMP102	3.3VDC	Sensor de temperatura digital por comunicación I2C	1
Sensor ZP5210.	$V_{rup}=220\text{DC}$; $I_{rup}=1\text{A}$	Sensor vertical digital de detección de nivel de agua.	2
Diodo emisor infrarrojo	5mm, 1,5V	Opera como actuador que trabaja conjuntamente con dispositivos con receptores infrarrojos	1
Transistor 2n222a	$\beta \cong 1000$	Usado para elevar la potencia de la señal enviada al diodo infrarrojo	1

Micrófono Trust GXT 212 Mico	USB/JACK	Interfaz de entrada principal entre el usuario y el sistema	1
Parlantes Estéreo Genius	128 kbps, entrada JACK	Interfaz de salida principal entre el sistema y el usuario, compatibilidad con cualquier parlante estéreo de entrada JACK.	1
Monitor BENQ	Tamaño:28" HDMI Res: 1920*1080	Interfaz de salida para las configuraciones, cualquier monitor HDMI es compatible.	1
Electroválvula	220[VAC] 50[hz] 0.02-0.8[Mpa]	Actuador para la salida de agua desde el tanque.	1
Motor de drenaje	220[VAC] 50[hz] 0.21[A] 3000 [rpm]	Actuador para la carga de agua al tanque.	1
Relé	5VDC/250VAC 10[A]	Actuador con la función de energizar o desenergizar equipos y/o electrodomésticos de corriente alterna (220VAC) mediante señales digitales.	6
Lámpara Espiral Fluorescente	220[VAC] 50/60[hz] 20[W] 850[Lm]	Se seleccionó esta lámpara debido a su bajo costo, bajo consumo de electricidad y alta disponibilidad en el mercado.	4
Socket para lampara espiral		Pieza metálica o de plástico en la que se encaja el casco de un foco para conectarlo a la electricidad	4
Cable bipolar	Nº12; 20[A]	Esta es la medida más eficiente entre el costo y el trabajo al cual estará sometido el cable.	12m
Resistencias	(220,330,10k)[Ω]	Se usan para complementar algunos circuitos para su funcionamiento correcto.	9
Arduino	Mega2560	Empleado solo con la finalidad de convertir la señal analógica a señal digital y enviar la información a través de un puerto serial.	1

Fuente: Elaboración propia.

3.3.4 Distribución de los periféricos, sensores y actuadores.

En la tabla 3.3 muestra la distribución de los sensores y actuadores en los distintos ambientes de la edificación demostrativa.

Tabla 3.3.Distribución de sensores, actuadores y otros componentes en la edificación demostrativa.

Ambiente.	Componentes.
Habitación.	Relé (cumple la función de interruptor de energía).
	Lámpara y socket para la iluminación de los ambientes.
Baño	Relé (cumple la función de interruptor de energía).
	Lámpara y socket para la iluminación de los ambientes.
	Electroválvula como llave de paso en la ducha.
Cocina	Relé (cumple la función de interruptor de energía).
	Lámpara y socket para la iluminación de los ambientes.
	Electroválvula como llave de paso del lavadero.
	Sensor de GLP MQ2 (Sensor detector de gas)
SALA	Relé (cumple la función de interruptor de energía).
	Lámpara y socket para la iluminación de los ambientes.
	Sensor de monóxido de carbono MQ6 (sensor contra incendios)
	Sensor magnético Hall para el monitoreo de la puerta principal.
	Diodo emisor infrarrojo con la función de controlar la televisión.
APARTAMENTO	Micrófono para la interacción de los usuarios con el asistente virtual.
	Parlantes para la comunicación del asistente virtual con los usuarios
PISO	Motor de drenaje para el bombeo del agua hacia el tanque.
	Sensores de nivel de agua para el control de agua en el tanque.
	Cableado general.

Fuente: Elaboración propia.

3.3.5 Selección del sistema de alimentación

Según las especificaciones técnicas otorgadas por el fabricante, para el óptimo funcionamiento de la tarjeta Raspberry Pi la alimentación es mediante una fuente conmutada de 5V, 3A mínimamente, la marca es indistinta, si bien esta tarjeta puede compartir alimentación a otros dispositivos a través de sus pines se optó por usar otras fuentes externas, debido a la cantidad de sensores y al consumo de corriente que tendría los mismos, tan solo la tarjeta Arduino usada en el prototipo consume 50 mA aproximadamente, además los dispositivos conectados a los puertos USB (como ser el micrófono) pueden llegar a consumir hasta 500 mA, esta información y el consumo de corriente de los distintos sensores están detallados en las fichas técnicas adjuntadas en los anexos.

Entonces para los distintos sensores empleados en el prototipo y la tarjeta Arduino se usa otra fuente de 5V, 3A o 3.3V compartiendo GND con la tarjeta Raspberry.

La tarjeta Arduino está alimentada con 5V y tiene comunicación con la Raspberry Pi mediante el cable serial RS232/USB.

Finalmente, los motores y lámparas están alimentados con voltaje domiciliario de baja tensión (220 VAC).

3.4 CONSIDERACIONES EN EL DISEÑO DEL CIRCUITO.

Para el diseño del circuito se usan transistores 2N222, los transistores tienen dos funciones principales, una como interruptor y otra como amplificador.

Para este proyecto se hará uso de los transistores en el circuito con la funcionalidad de interruptores, pues una vez que el Raspberry Pi mande una señal eléctrica, colocando uno de sus puertos GPIO en 1, el transistor que reciba la señal dará paso a la corriente de otra fuente externa, activando de esta manera el relé que controla los dispositivos de 220 VAC.

La razón del uso de transistores en este proyecto es debido a que el Raspberry no suministra suficiente corriente para activar los relés.

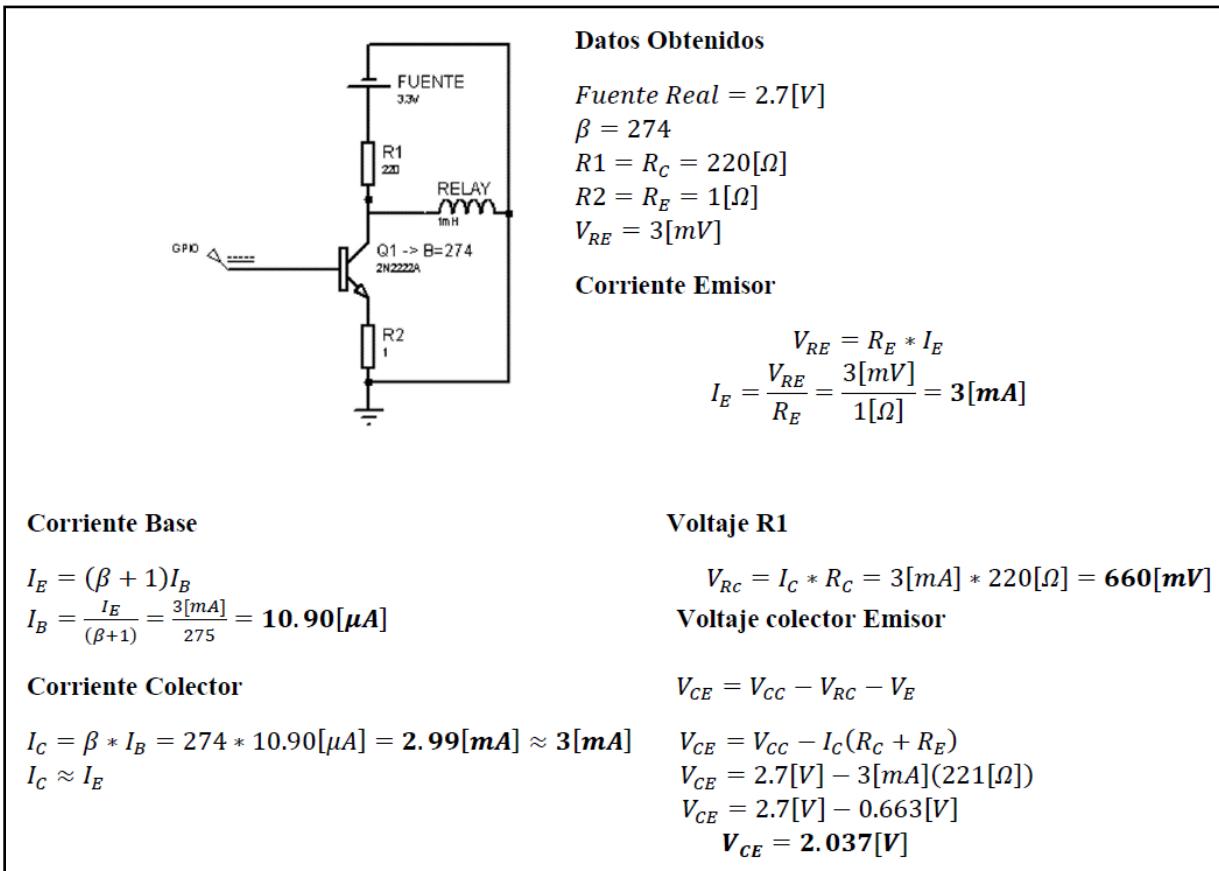


Figura 3.5: Diagrama de circuito del transistor.

Fuente: Elaboración propia.

3.5 DIAGRAMA DE CIRCUITO DEL PROTOTIPO DEL SISTEMA DE INTELIGENCIA ARTIFICIAL.

La figura 3.4 muestra el circuito que se diseñó e implementó en el prototipo del sistema de inteligencia artificial S.I.A.

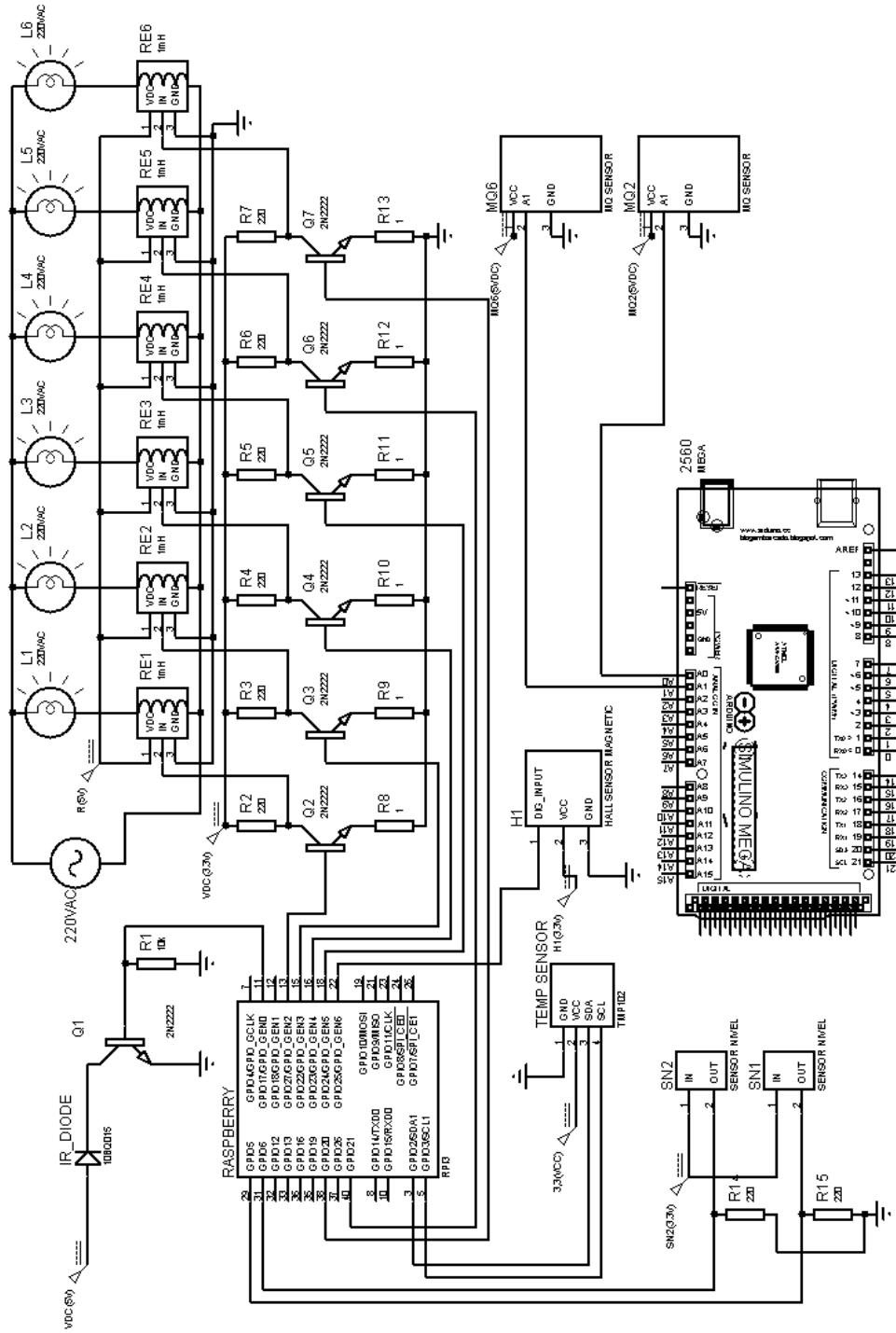


Figura 3.5: Diagrama de circuito del prototipo SIA.

Fuente: Elaboración propia.

3.6 DIAGRAMA DE CONEXIÓN DEL PROTOTIPO DEL SISTEMA DE INTELIGENCIA ARTIFICIAL.

La figura 3.5 muestra las conexiones que existen entre los diferentes dispositivos empleados para el prototipo S.I.A. y que cumple con el diseño de circuito mostrado en la figura 3.4.

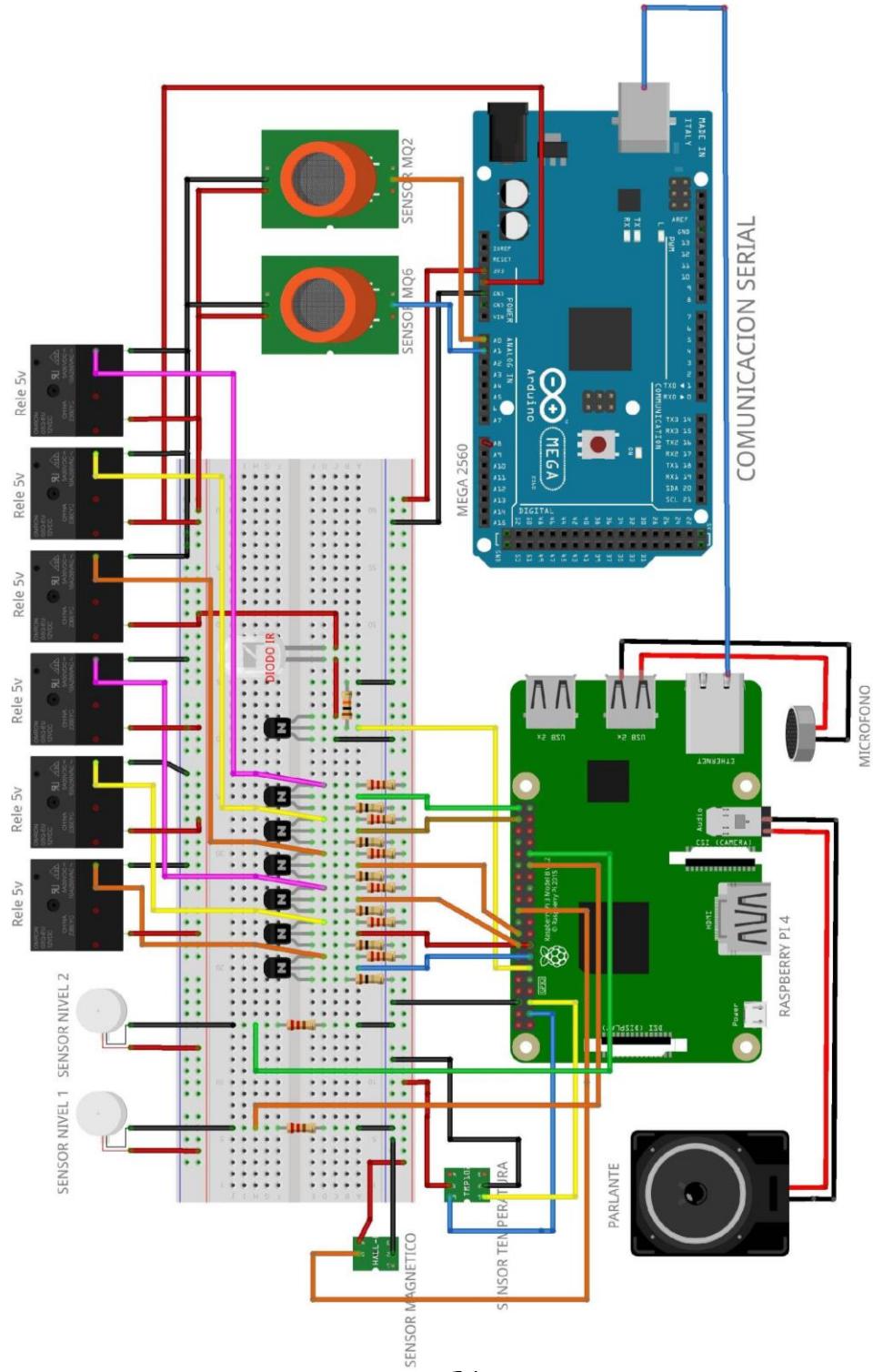


Figura 3.6: Diagrama de conexión del prototipo SIA.
Fuente: Elaboración propia.

3.7 DIAGRAMA PCB DE LA PLACA DEL SISTEMA DE INTELIGENCIA ARTIFICIAL.

Para la implementación del circuito en el sistema es necesario desarrollar una tarjeta de expansión para las conexiones de los sensores y actuadores con el microcontrolador y la Raspberry Pi, la figura 3.6 muestra el diseño de esta tarjeta de expansión la cual se denominará como “tarjeta de distribución SIA”, la tarjeta tiene una dimensión de 10X10[cm].

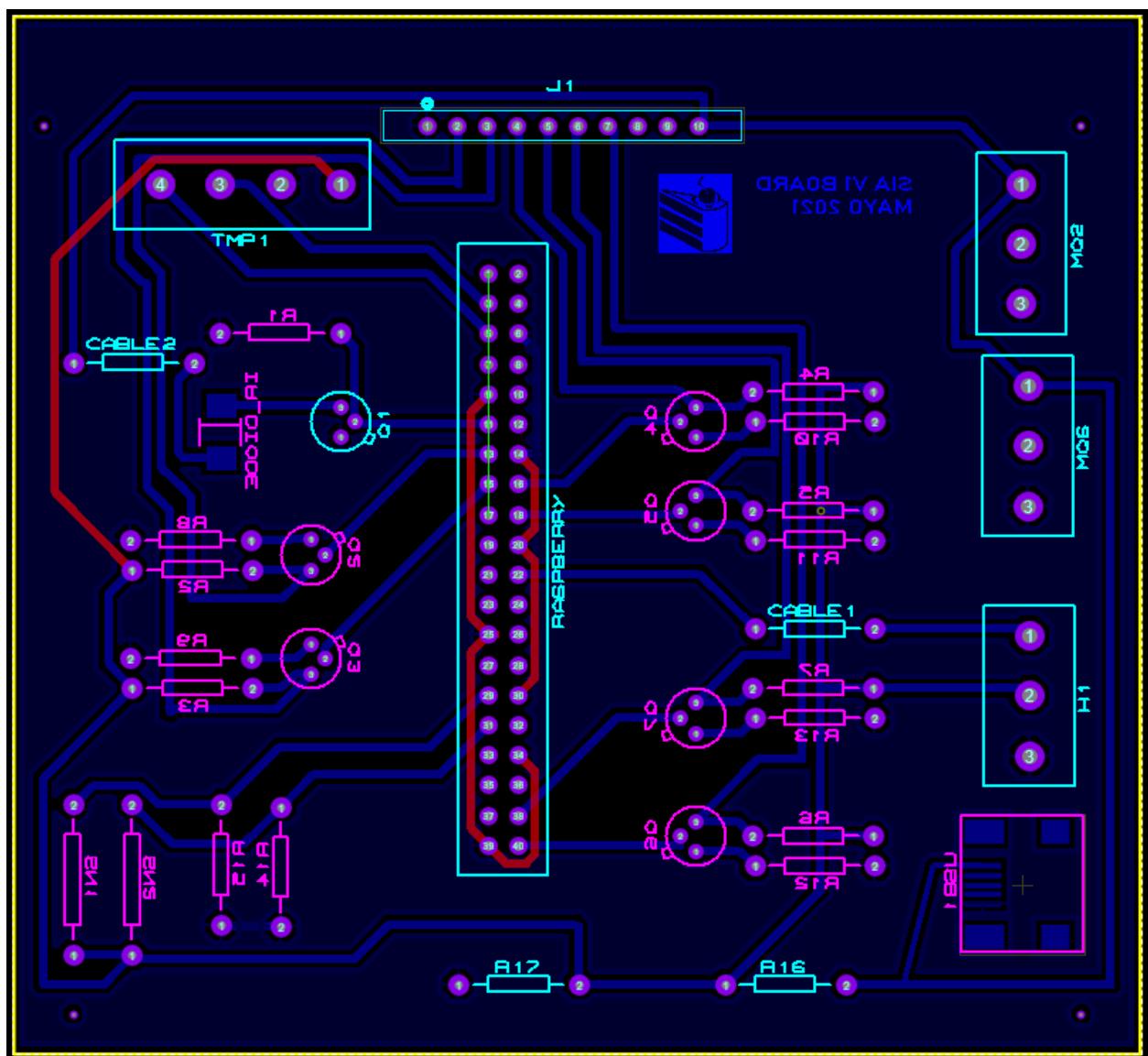


Figura 3.7: Diagrama PCB de la placa de distribución SIA.
Fuente: Elaboración propia.

Capítulo 4

INGENIERÍA Y DESARROLLO DEL SOFTWARE

4 INGENIERÍA Y DESARROLLO DEL SOFTWARE.

4.1 SELECCIÓN DEL SISTEMA OPERATIVO.

Debido a que la tarjeta Raspberry Pi 4 tiene un sistema operativo dedicado, se usara dicho sistema operativo el cual es Raspbian que es a la vez una derivación de Debían, donde Debian es una distribución de Linux.

Al usar una distribución de Linux se puede destacar las siguientes ventajas:

- Linux es muy robusto, estable y rápido: Ideal para servidores y aplicaciones distribuidas. A esto se añade que puede funcionar en máquinas *humildes*: Linux puede correr servicios en un x86 a 200 MHz con calidad
- Linux es libre: Esto implica no sólo la gratuidad del software, sino también que Linux es modificable y que Linux tiene una gran cantidad de aplicaciones libres en Internet. Todo ello arropado por la inmensa documentación de Linux que puede encontrarse en la Red.
- Linux ya no está restringido a personas con grandes conocimientos de informática: Los desarrolladores de Linux han hecho un gran esfuerzo por dotar al sistema de asistentes de configuración y ayuda, además de un sistema gráfico muy potente. Distribuciones Linux como Red Hat/Fedora tienen aplicaciones de configuración similares a las de Windows.
- El manejo de la memoria de Linux evita que los errores de las aplicaciones detengan el núcleo de Linux.
- Linux es multitarea y multiusuario: Esta característica imprescindible está en Unix desde su concepción, pero le llevó a Microsoft más de 20 años ofrecerlo en su sistema operativo de consumo.
- Casi cualquier aplicación Unix puede usarse bajo Linux.
- Las libertades de copia y modificación permiten usar GNU/Linux para facilitar servicios sin depender de terceros.
- Raspbian en particular es uno de los sistemas operativos más estables en la actualidad.
- Casi no existen los malwares o virus para el sistema operativo Raspbian.

Sin embargo, existen también algunas desventajas de las cuales se pueden resaltar las siguientes:

- Linux no cuenta con una empresa que lo respalde, por lo que no existe un verdadero soporte como el de otros sistemas operativos.
- Documentación y terminología muy técnica.
- Requiere consulta, lectura e investigación en lista, foros o en bibliografía dedicada al tema.
- Muchas distribuciones e idiomas.
- El inglés es el idioma estándar para las líneas de comandos y los mensajes del sistema.
- No todas las versiones cuentan con asistencia a largo plazo.
- Windows es incompatible con Linux: Este punto es difícil de explicar: no quiere decir que no podamos tener instalados ambos Sistemas (que es relativamente fácil de hacer).
- En la mayoría de distribuciones Linux hay que conocer nuestro Hardware a la hora de instalar.

4.2 SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN.

4.2.1 Evaluación comparativa de lenguajes de programación.

Para la selección adecuada del lenguaje de programación se toma en cuenta los siguientes aspectos que son requeridos e indispensables para el avance y desarrollo del proyecto, aspectos los cuales el lenguaje de programación seleccionado debe satisfacer.

En primer lugar, se debe tomar en cuenta que el lenguaje de programación debe estar orientado al desarrollo de una aplicación ejecutable para el sistema operativo Raspberry.

Por tanto, se descarta todo lenguaje que no cumpla este primer objetivo como por ejemplo JavaScript o PHP, que son lenguajes orientados al desarrollo web o lenguajes como Kotlin o Swift que son lenguajes orientados al desarrollo de aplicaciones móviles.

También se requiere que el lenguaje sea orientado a objetos para que el código sea robusto, flexible y modular, se escoge por preferencia del autor tres lenguajes de alto nivel: Java, Python y C++.

Se selecciona estos tres lenguajes por la amplia documentación, soporte y una vasta comunidad que lo respalda.

A continuación, se muestra en la figura 4.1 el tiempo de respuesta de los tres lenguajes, de una evaluación, análisis y comparación del rendimiento de programas de procesamiento masivo.

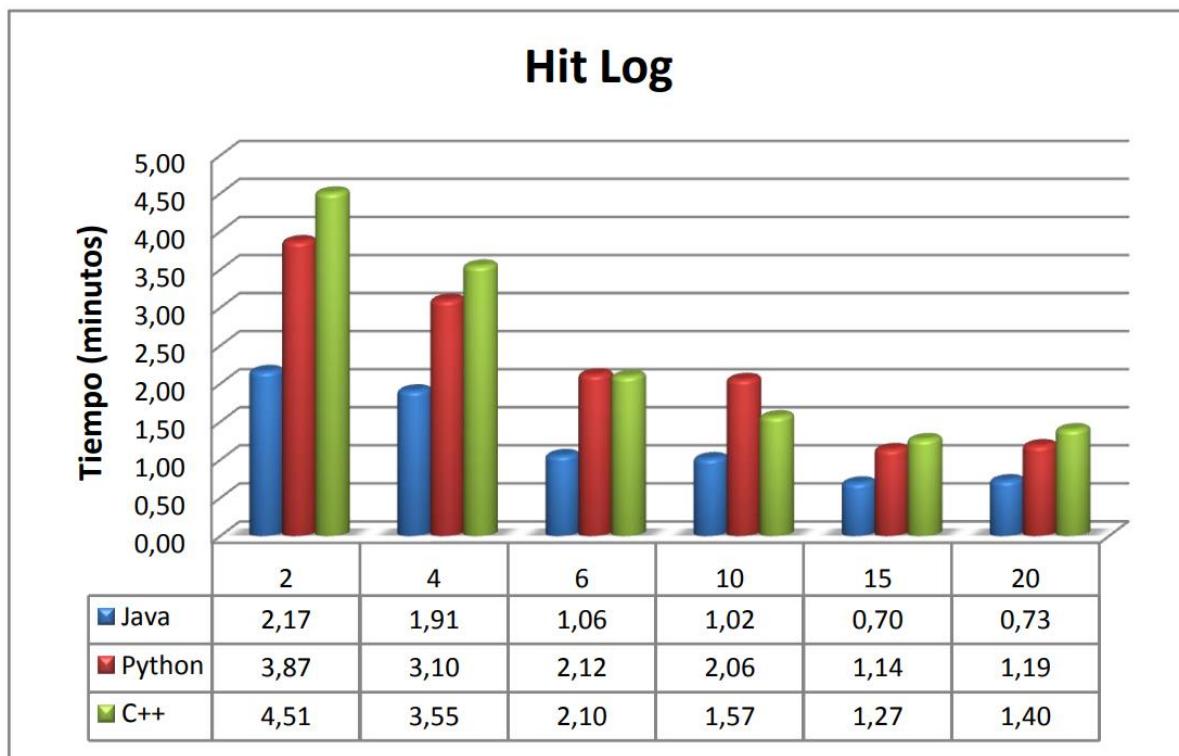


Figura 4.1: Tiempo de respuesta Vs numero de nodos aplicación Hit Log. Escuela superior politecnica del Litoral (2010, p.27).

También en la figura 4.2 se muestra la cantidad código que usa cada lenguaje de programación.

Líneas de Código vs Lenguaje

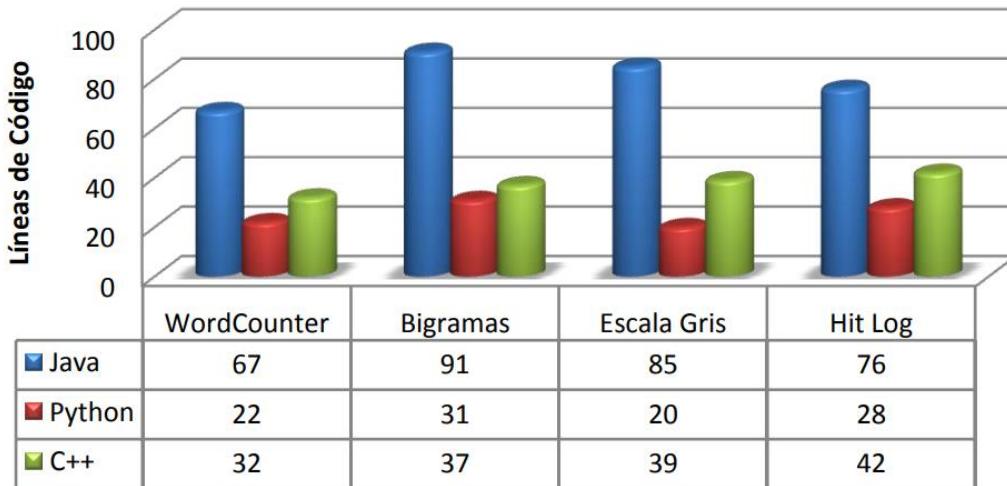


Figura 4.2: Numero de lineas de código vs lenguaje de programación. Escuela superior politecnica del Litoral (2010, p.27).

Es comprensible que Python tenga una cantidad mínima de líneas, según Marzal et al (2014):

Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: Un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. (Python llega a ser considerado por muchos un lenguaje de programación de muy alto nivel).

Python es muy legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación (sección 1.3.6).

Tanto Python, como Java y C++ son lenguajes con los cuales se puede desarrollar software orientado a la inteligencia artificial.

Concluyendo obtenemos la siguiente tabla:

Tabla 4.1. Tabla de requerimientos de lenguajes de programación.

	Lenguajes		
Características Requeridas	Java	Python	C++
Lenguaje con una amplia documentación, soporte y una vasta comunidad que lo apoya.	Si	Si	Si
Lenguaje de programación orientado a objetos.	Si	Si	Si
Fácil interacción con hardware y periféricos de entrada y salida.	No	Si	No
Sintaxis simple.	No	Si	No
Lenguaje de programación de alto nivel.	Si	Si	Si.
Amplia gama de librerías enfocadas hacia la inteligencia artificial y machine learning.	Si	Si	Si

Fuente: Elaboración propia.

4.2.2 Selección y Justificación de la selección del lenguaje.

Concluyendo y tomando en cuenta la tabla 4.1 el lenguaje seleccionado que cumple con todos los requerimientos y por preferencia del autor es Python, por ende, es el lenguaje seleccionado para el desarrollo del software para el sistema de inteligencia artificial.

Más allá de los puntos mencionados anteriormente se selecciona Python por varios motivos.

Python es un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

La sintaxis de Python es tan sencilla y cercana al lenguaje natural que los programas elaborados en Python parecen pseudocódigo.

Python es quizás el mejor candidato en cuanto a lenguajes de programación si queremos centrarnos en Inteligencia Artificial, es el lenguaje a la vanguardia de la investigación de IA, el que encontrará la mayor cantidad de marcos de aprendizaje automático y aprendizaje profundo.

Este lenguaje ya se encuentra preinstalado en varias distribuciones del sistema operativo Linux, entre los cuales se encuentra Raspbian, sistema operativo que se seleccionó para el desarrollo del software, además, es recomendación de la fundación Raspberry usar Python

para el desarrollo de proyectos con sus microcomputadores, ya que posee variedad de librerías que facilitan la interacción con sensores y periféricos de entrada y salida.

Python es un poderoso lenguaje de programación que es fácil de usar, fácil de leer y escribir y, con Raspberry Pi, le permite conectar su proyecto al mundo real. La sintaxis de Python es limpia, con énfasis en la legibilidad y utiliza palabras clave estándar en inglés.(Raspberry Pi Foundation, s.f.)

4.3 SELECCIÓN DEL ENTORNO DE DESARROLLO INTEGRADO (IDE).

Para la selección del IDE se tomó en cuenta dos entornos de desarrollo más populares y disponibles para el sistema operativo Raspbian, los cuales son IDLE Python y Thonny.

Ambos vienen preinstalados en el sistema operativo, sin embargo, para el desarrollo del software se usó Thonny.

Thonny es fácil de usar y viene con Python 3.7 incorporado, ofrece una interfaz simple y limpia, a diferencia de IDLE Python, Thonny viene con un depurador para ayudar a detectar y corregir errores en el código. Tiene características como evaluación de expresiones, explicación del alcance, resaltado de sintaxis y finalización de código, que agregan conveniencia y mejoran la experiencia de codificación.

4.4 SELECCIÓN DEL GESTOR DE BASE DE DATOS.

Se utiliza MariaDB SQL, debido a que posee una licencia pública, permitiendo la utilización del programa, consulta y modificación de su código fuente, es fácil de personalizar y adaptar a necesidades concretas, añadiendo que es portable y compatible con diferentes sistemas operativos entre ellos: Windows, Mac Os y Linux.

4.5 SELECCIÓN DEL MÓDULO DE RECONOCIMIENTO DE VOZ.

Para el reconocimiento de voz del asistente virtual se utiliza el módulo **Google Speech Recognition** debido a que tiene compatibilidad con Linux y Python, además, tiene la opción de reconocimiento de lenguaje español, es bastante robusto y estable, y su librería es de código

abierto, como única desventaja se podría mencionar que se necesita de una conexión constante a internet.

4.6 DESARROLLO DEL AGENTE RACIONAL.

Para el desarrollo del agente es necesario especificar el entorno de trabajo, para esto se usa dos tipos de descripción:

- R.E.A.S (Rendimiento, Entorno, Actuadores, Sensores).
- P -> A (Percepción - Acción)

4.6.1 Descripción R.E.A.S. del entorno.

Tabla 4.2. Descripción REAS del entorno.

Tipo de agente	Medida de rendimiento	Entorno	Actuadores	Sensores
Sistema de inteligencia Artificial Domótico	<ul style="list-style-type: none"> • Tiempo • Precisión • Confort • Fluidez • Seguridad • Costos 	Edificaciones (Casas y /o edificios)	<ul style="list-style-type: none"> • Monitor • Parlantes • Válvulas • Celular • Motor • Diodo Infrarrojo • Luces 	<ul style="list-style-type: none"> • Micrófono • Teclado • Sensor de gas • Sensor de incendios • Sensor magnético de puertas • Sensor de nivel de agua • Sensor de temperatura • Celular

Fuente: Elaboración propia.

4.6.2 Descripción P>A (Percepción > Acción) del entorno.

Un agente tomará una decisión en un momento dado dependiendo de la secuencia completa de percepciones hasta ese instante.

En términos matemáticos se puede decir que el comportamiento del agente viene dado por la función del agente que proyecta una percepción dada en una acción.

Tabla 4.3. Tabla parcial de una función de agente sencilla para el sistema SIA.

Secuencia de percepciones.	Acción.
Percibe orden por voz.	Enciende luz. Apaga luz. Indica la temperatura. Activa Seguridad. Reproduce música. Programa Alarma. Genera y envía reportes al teléfono móvil. Controla el televisor. Enciende válvula de agua.
Percibe intrusos en la edificación.	Informa a los usuarios.
Percibe auxilio por voz.	Informa a familiares.
Percibe Gas.	Activa alarma, informa a los usuarios.
Percibe Humo.	Activa alarma, informa a los usuarios.
Percibe ambiente libre de humo y/o gas.	Desactiva alarma.
Percibe tanque de agua vacío.	Activa bomba de agua.
Percibe tanque de agua lleno.	Desactiva bomba.
Para cualquiera de estas percepciones.	Genera logs (registros).

Fuente: Elaboración Propia

4.7 DESARROLLO DEL SISTEMA.

4.7.1 Adaptación de la metodología.

Se implementa la metodología R.U.P. (por sus siglas en inglés de Rational Unified Process) o proceso de desarrollo unificado, es un proceso de desarrollo de software ,constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP está orientada a un equipo de trabajo que manejan roles definidos, sin embargo, es preferencia del autor usar esta metodología debido a la estructura organizada y disciplinada que posee la metodología y a la amplia documentación que tiene a disposición en la universidad, además, es una metodología ideal para cualquier persona que inicie en el desarrollo de software.

Tabla 4.4. Especificación de fases en función de artefactos.

Fases	Artefactos
Inicio	Definición y gestión de riesgos. Identificación de requerimientos.
Elaboración	Modelo de casos de uso del sistema. Lista priorizada de requerimientos del sistema. Arquitectura del sistema.
Construcción	Modelo de casos de uso. Especificación de casos de uso. Lista especificada de tareas. Pruebas unitarias. Pruebas de aceptación. Hoja de trabajo.
Transición	Pruebas de integración del sistema. Pruebas de aceptación del sistema.

Fuente: Elaboración propia.

4.7.2 Fase de inicio.

4.7.2.1 Definición de riesgos.

La tabla 4.4 muestra la valoración de los riesgos que presenta el sistema donde cada riesgo es la descripción de un problema potencial con la posibilidad de ocurrir.

Tabla 4.5. Definición de riesgos.

Riesgo	Categoría	Probabilidad de impacto	Impacto	Gestión de riesgo
Desarrollo erróneo de interfaz	Técnico-Diseño	Alta	Crítico	Prototipos, escenarios, análisis de tareas.
Deficiencia en componentes	Técnico-Construcción	Alta	Crítico	Inspecciones, análisis de compatibilidad.
Desarrollo de funciones erróneas	Proyecto - Requerimientos	Moderada	Serio	Análisis de planificación y organización, pruebas con participación de usuarios, manuales preliminares.
Se proponen cambios en requerimientos que requieren rehacer el diseño	Proyecto - Requerimientos	Moderado	Serio	Los nuevos requerimientos no deben desviarse de los objetivos principales del proyecto.
El tiempo requerido para el desarrollo del software está subestimado	Proyecto - Planificación	Alta	Serio	Análisis de la planificación del proyecto.

Fuente: Elaboración propia.

4.7.2.2 Requerimientos funcionales del sistema.

La tabla 4.6 muestra la descripción de los requerimientos funcionales del sistema que están en función al identificador que especifica la identificación funcional de cada requerimiento y su descripción es el requerimiento como tal.

Tabla 4.6. Requerimientos funcionales del sistema.

Nro.	Identificador.	Descripción.
1	RF01	Configuración inicial.
2	RF02	Registrar residentes y sus respectivos roles.
3	RF03	Controlar sensores y actuadores.
4	RF04	Gestionar servicios y sus respectivos consumos.
5	RF05	Interactuar con los usuarios.
6	RF06	Generar registros de actividades.
7	RF07	Generar registros de residentes.
8	RF08	Generar gráficas de servicios.

Fuente: Elaboración propia.

4.7.2.3 Requerimientos no funcionales del sistema.

La tabla 4.6 muestra la descripción de los requerimientos no funcionales del sistema, están en función al identificador que especifica la identificación funcional de cada requerimiento y su descripción es el requerimiento no funcional.

Tabla 4.7. Requerimientos no funcionales del sistema.

Nro.	Identificador	Descripción.
1	RNF01	Sistema operativo Linux
2	RNF02	El sistema necesita conexión constante a internet
3	RNF03	El sistema es codependientes de todos los sensores, actuadores y periféricos de entrada y salida.

Fuente: Elaboración propia.

4.7.3 Fase de elaboración.

En esta fase se especifica la comunicación y el comportamiento del sistema, mediante su interacción con los usuarios identificados, los cuales interactúan a con el sistema principalmente a través de la voz, también se diseña la arquitectura del sistema.

4.7.3.1 Modelo general de casos de uso.

En la figura 4.3 se muestra un esquema del modelo general de los casos de uso del sistema de inteligencia artificial, el cual sirve para especificar la comunicación y el comportamiento de la aplicación, el modelo cuenta con tres actores: Administrador, usuario residente y el asistente virtual.

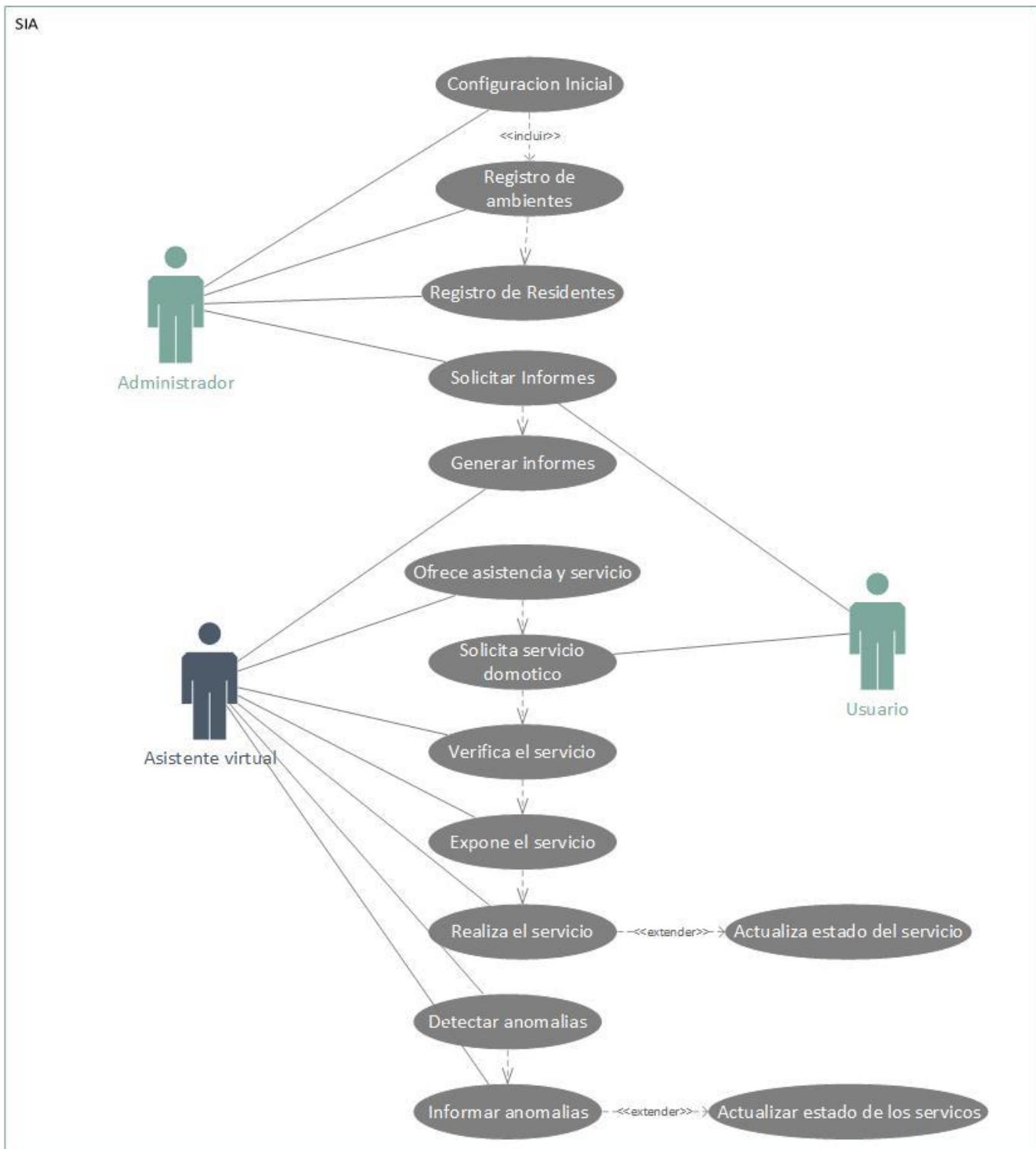


Figura 4.3: Modelo general de casos de uso.
Fuente: Elaboración propia.

4.7.3.2 Componentes del sistema.

El proyecto se compone de cuatro módulos detallados en la siguiente tabla.

Tabla 4.8. Módulos del sistema.

Módulos del sistema.		
Nro.	Módulo.	Descripción.
1	SIA	Es el módulo principal que permite la creación de usuarios, almacena la información de residentes, además de la posterior gestión de los mismos, también se encarga de la recolección y almacenamiento de información de los otros módulos.
2	PISO	Permite la creación de apartamentos y la gestión de servicios del piso y envía la información al módulo principal.
3	APARTAMENTO	Módulo de creación y registro de residentes, habitaciones y/o ambientes y envió de información al módulo principal.
4	HABITACION	Gestiona y controla los sensores instalados y distribuidos en los ambientes, e interactúa directamente con las peticiones de los residentes.

Fuente: Elaboración propia.

4.7.3.3 Arquitectura del sistema.

Originalmente el sistema está planificado para manejar una estructura cliente-servidor, con la siguiente lógica:

En una edificación cualquiera el servidor principal debe encontrarse en un ambiente particular donde pueda trabajar de manera óptima y segura.

En el caso de un edificio, por ejemplo, el servidor debe trabajar en el cuarto de máquinas donde solo personal autorizado pueda acceder a él.

El servidor almacena la base de datos y el programa que gestiona la información que recolecta de las máquinas cliente, además que administra la información de los residentes y de los servicios.

Las máquinas cliente (raspberry) deben de estar distribuidos uno por apartamento de edificio, o de no tratarse de un piso que tenga apartamentos por ejemplo el parqueo, entonces se distribuye una máquina cliente por piso.

Las máquinas clientes deben estar prestas a interactuar con los usuarios, controlar los sensores y actuadores, y enviar toda la información de su actividad al servidor.

En el caso de una casa lo más conveniente es que el servidor se encuentre el estudio o de no haber uno que se ubique en un lugar fijo con un ambiente frío y de temperatura constante, además, si se trata de una casa pequeña que no posea pisos superiores se puede manejar el cliente-servidor en una sola máquina.

Para este prototipo se usa esta última opción.

4.7.4 FASE DE CONSTRUCCIÓN.

4.7.4.1 Diagrama UML del Sistema de Inteligencia Artificial (S.I.A.).

En la figura 4.4 se muestra el diagrama UML del modelo de programación orientado a objetos que se manejó para el desarrollo del módulo del sistema S.I.A.

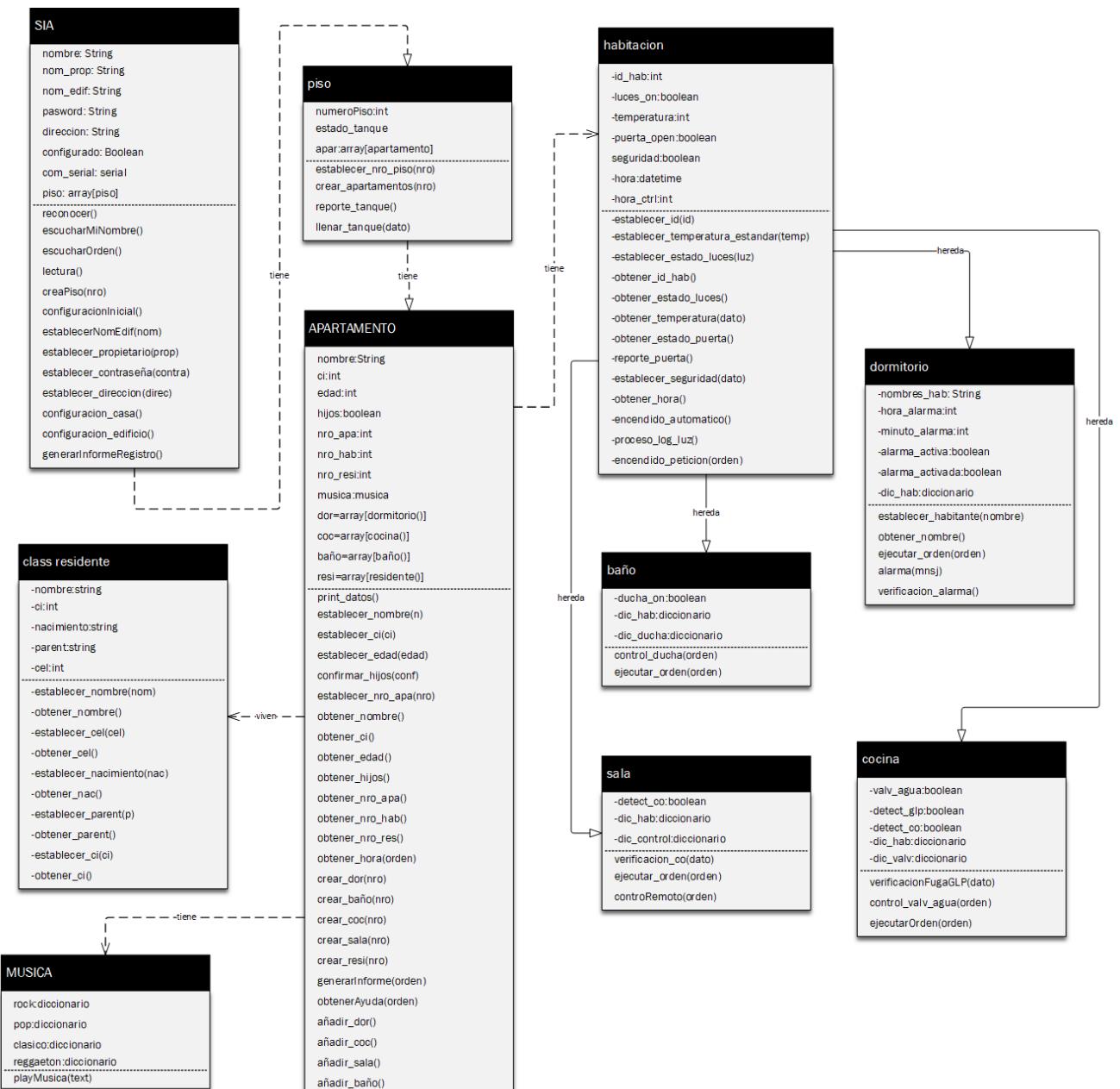


Figura 4.4: Diagrama UML de S.I.A.
Fuente: Elaboración Propia.

4.7.4.2 Especificación casos de uso – módulo SIA.

A continuación, en la tabla 4.9 se muestra con más detalle los diferentes casos de uso que se presentan en el módulo SIA.

Tabla 4.9. Casos de usos.

Nombre del caso de uso	Configuración inicial.
Actor principal	Administrador.
Descripción	Permite crear una cuenta del propietario en el sistema y definir el tipo de edificación.
Precondición	Iniciar el sistema.
Flujo Normal	1: El administrador inicia el sistema enciendo el ordenador. 2: El administrador ingresa el nombre del propietario y la respectiva contraseña. 3: El administrador ingresa el tipo de edificación (domicilio particular o edificio)
Postcondición	El registro del propietario y la selección de la edificación se realizó correctamente.
Criterios de aceptación	El propietario y el administrador deben tener acceso al sistema.

Nombre del caso de uso	Crear pisos.
Actor principal	Administrador.
Descripción	Permite crear la cantidad de pisos que hay en el edificio.
Precondición	Iniciar el sistema, tipo de edificación seleccionada.
Flujo Normal	1: Una vez seleccionado el tipo de edificación se presentan dos casos. 2: El administrador ingresa la cantidad de pisos que hay en el edificio. 3: El sistema asigna automáticamente un piso si se trata de un domicilio particular.
Postcondición	El registro del número de pisos se realizó correctamente.
Criterios de aceptación	El administrador debe poder acceder a las funciones de todos los pisos creados.

Nombre del caso de uso	Escuchar petición.
Actor principal	Administrador, usuario.
Descripción	Escucha la petición de servicio del usuario.
Precondición	Configuración inicial concluida.
Flujo Normal	<p>1: El usuario pronuncia el nombre del asistente virtual.</p> <p>2: El asistente virtual responde al nombre y ofrece su atención.</p> <p>3: El usuario solicita un servicio domótico.</p> <p>4: El asistente virtual busca el servicio.</p> <p>5: El asistente virtual encuentra el servicio y ejecuta la orden del usuario.</p> <p>6: El asistente virtual espera nuevamente escuchar su nombre.</p>
Postcondición	El servicio domótico se ejecutó correctamente.
Criterios de aceptación	Los servicios domóticos ejecutados deben ser los mismos que los usuarios solicitaron.

Nombre del caso de uso	Generar Informes.
Actor principal	Administrador.
Descripción	Genera un informe general de los registros de todas las actividades del edificio.
Precondición	Configuración inicial concluida.
Flujo Normal	<p>1: El administrador solicita un informe general.</p> <p>2: El asistente virtual genera el informe en un formato PDF.</p> <p>3: El asistente virtual envía el informe al celular del administrador solicitante.</p>
Postcondición	El documento llegó correctamente al móvil del usuario.
Criterios de aceptación	Solo los administradores pueden acceder a este servicio de manera regular.

Fuente: Elaboración propia.

4.7.4.3 Especificación casos de uso – módulo piso.

En la tabla 4.10 se muestran los diferentes casos de uso que se presentan en la clase “piso”.

Tabla 4.10. Especificación casos de uso – módulo piso.

Nombre del caso de uso	Crear apartamentos.
Actor principal	Administrador.
Descripción	Permite crear la cantidad de apartamentos que hay en un piso del edificio.
Precondición	Iniciar el sistema, tener pisos creados.
Flujo Normal	1: Una vez creado los pisos el administrador selección un piso. 2: El administrador ingresa la cantidad de apartamentos que hay en el piso seleccionado.
Postcondición	Los apartamentos se crearon correctamente.
Criterios de aceptación	El administrador debe poder acceder a las funciones de todos los apartamentos creados.

Nombre del caso de uso	Control de servicios.
Actor principal	Asistente virtual.
Descripción	Controla que los servicios operen de manera regular.
Precondición	Configuración inicial concluida
Flujo Normal	1: El asistente virtual verifica que los servicios sean suministrados de manera regular 2: El asistente encuentra una irregularidad en un servicio. 3: El asistente informa al administrador sobre la irregularidad. 4: El asistente ejecuta acciones programadas, predefinidas y limitadas.
Postcondición	El asistente informó y ejecutó las acciones programadas de manera exitosa.
Criterios de aceptación	Los residentes deben tener los servicios constantemente minimizando los períodos de tiempo en mantenimiento.

Fuente: Elaboración propia.

4.7.4.4 Especificación casos de uso – módulo apartamento.

En la tabla 4.11 se muestran los diferentes casos de uso que se presentan en la clase “apartamento”.

Tabla 4.11. Especificaciones casos de uso – módulo apartamento.

Nombre del caso de uso	Crear habitaciones.
Actor principal	Administrador.
Descripción	Permite crear la cantidad de habitaciones que hay en el apartamento de un piso.
Precondición	Iniciar el sistema, tener apartamentos creados.
Flujo Normal	1: Una vez creado los apartamentos el administrador selecciona un apartamento. 2: El administrador ingresa la cantidad de habitaciones que hay en el apartamento seleccionado. 3: Registra las habitaciones con el tipo de uso correspondiente.
Postcondición	Las habitaciones se crearon y registraron correctamente.
Criterios de aceptación	El administrador y los usuarios residentes deben poder acceder a las funciones de las habitaciones creadas por apartamento.

Nombre del caso de uso	Crear Residentes.
Actor principal	Administrador.
Descripción	Permite crear y registrar los residentes que viven en un apartamento.
Precondición	Iniciar el sistema, tener apartamentos creados.
Flujo Normal	1: Una vez creado los apartamentos el administrador selecciona un apartamento. 2: Ingresa la cantidad de residentes que hay en el apartamento seleccionado. 3: Registra al propietario del apartamento. 4: Registra al resto de residentes incluyendo el parentesco con el propietario
Postcondición	La creación y registro de los residentes se realizó correctamente.
Criterios de aceptación	El administrador puede ingresar a la información de los residentes.

Nombre del caso de uso	Generar informe de residentes.
Actor principal	Administrador.
Descripción	Genera un informe de los residentes que habitan en un apartamento del edificio.
Precondición	Configuración inicial concluida.
Flujo Normal	1: El administrador solicita un informe de residentes. 2: El asistente virtual genera el informe en un formato PDF. 3: El asistente virtual envía el informe al celular del administrador solicitante.
Postcondición	El documento llegó correctamente al móvil del usuario.
Criterios de aceptación	Solo los administradores pueden acceder a este servicio de manera regular.

Nombre del caso de uso	Generar librería de música.
Actor principal	Sistema.
Descripción	Crea un módulo de repertorio de música para los residentes.
Precondición	Tener apartamentos creados.
Flujo Normal	1: Una vez creado un apartamento el sistema crea un repertorio de música con distintos géneros.
Postcondición	El módulo se creó correctamente.
Criterios de aceptación	Los residentes deben poder solicitar canciones al asistente virtual y escucharlas.

Fuente: Elaboración propia.

4.7.4.5 Especificación casos de uso – módulo habitación.

En la tabla 4.12 se muestran los diferentes casos de uso que se presentan en la clase “habitación”.

Tabla 4.12. Especificación casos de uso – módulo habitación.

Nombre del caso de uso	Escuchar petición.
Actor principal	Usuario residente.
Descripción	Escucha la petición de servicio del usuario.
Precondición	Configuración inicial concluida.
Flujo Normal	1: El usuario pronuncia el nombre del asistente virtual. 2: El asistente virtual responde al nombre y ofrece su atención. 3: El usuario solicita un servicio domótico. 4: El asistente virtual busca el servicio. 5: El asistente virtual encuentra el servicio y ejecuta la orden del usuario. 6: El sistema verifica el estado de las peticiones de los usuarios residentes. 7: El asistente virtual espera nuevamente escuchar su nombre.
Postcondición	El servicio domótico se ejecutó correctamente.
Criterios de aceptación	Los servicios domóticos ejecutados deben ser los mismos que los usuarios solicitaron.

Nombre del caso de uso	Control de sensores y actuadores.
Actor principal	Sistema.
Descripción	Gestiona y controla los sensores y actuadores.
Precondición	Configuración inicial concluida.
Flujo Normal	1: El sistema recolecta la información de los sensores. 2: El sistema encuentra irregularidades en la información recolectada. 3: El asistente virtual informa a los usuarios residentes sobre las irregularidades.
Postcondición	El control de los sensores se realiza de manera exitosa.
Criterios de aceptación	EL asistente virtual debe informar las irregularidades de un apartamento solo a los residentes de ese apartamento y al administrador.

Fuente: Elaboración propia.

4.7.4.6 Diagrama entidad relación de la base de datos.

La figura 4.5 muestra el diagrama entidad-relación (ER) de la base de datos del sistema, donde la base de datos se encuentra modelado en su primera forma.

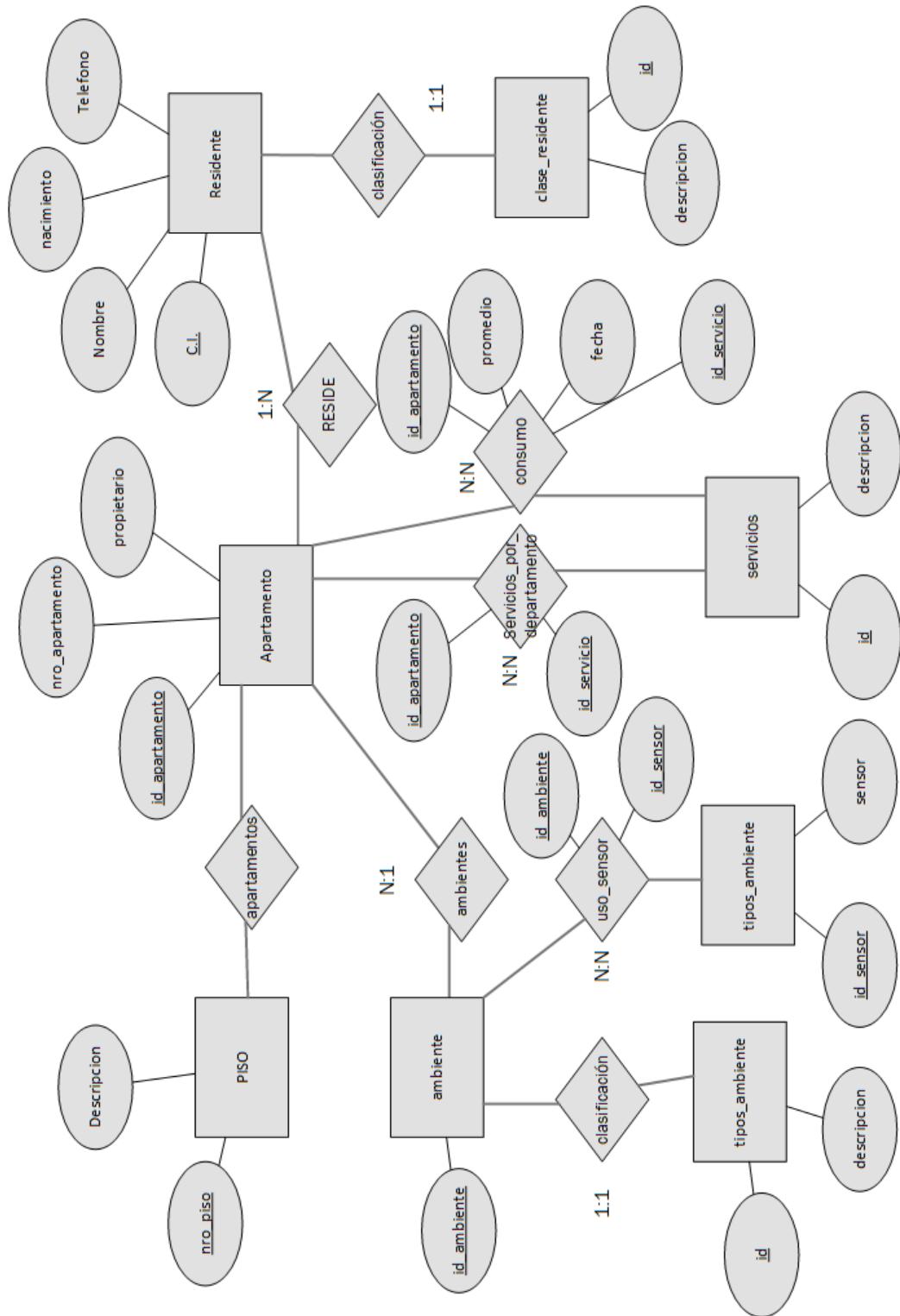


Figura 4.5: Diagrama entidad-relación de la base de datos.

Fuente: Elaboración propia.

4.7.4.7 Diagrama UML de la base de datos.

La figura 4.6 muestra el diagrama UML de la base de datos del sistema, donde la base de datos se encuentra modelado en su tercera forma.

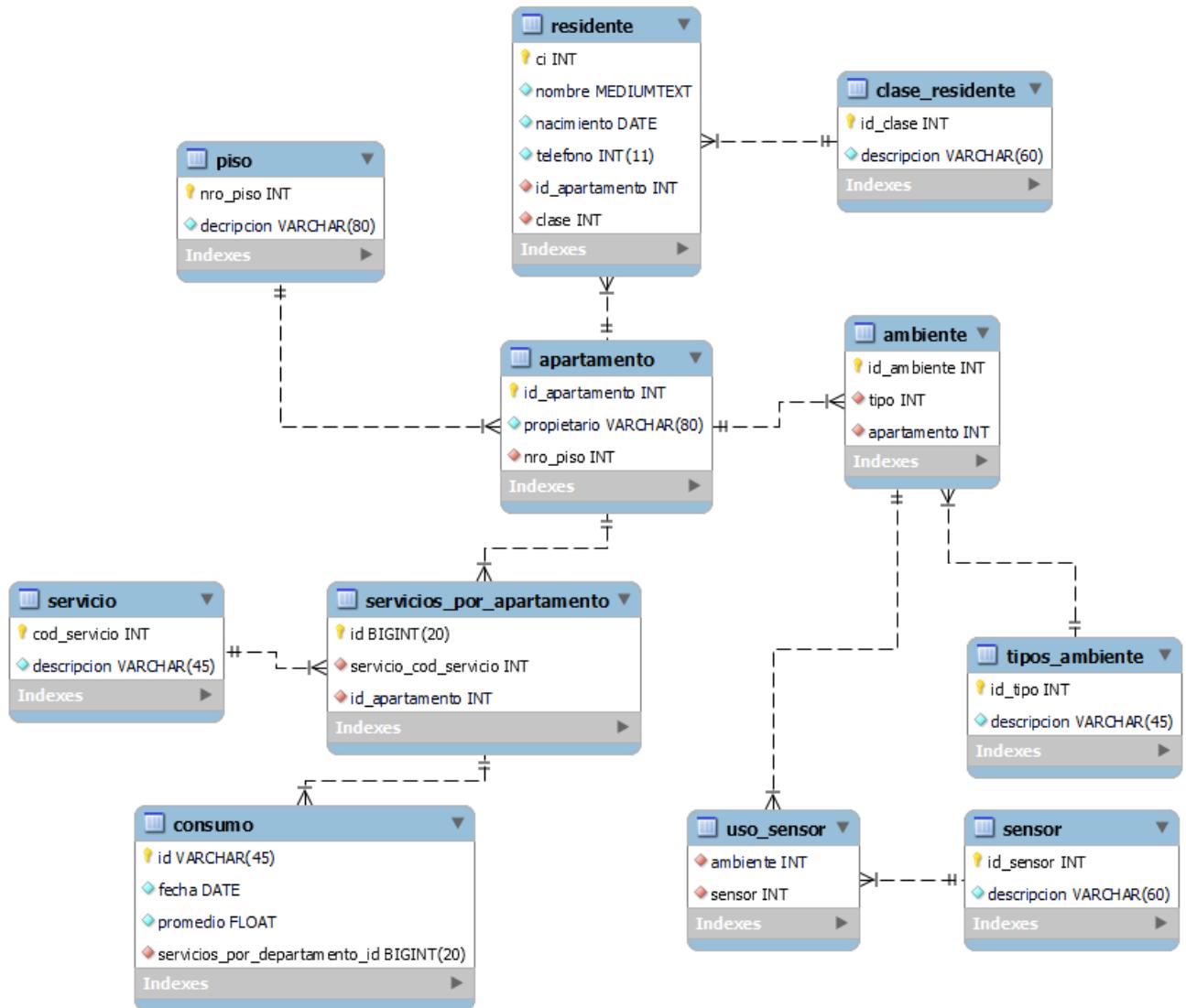


Figura 4.6: Diagrama UML de la base de datos.

Fuente: Elaboración propia.

4.7.4.8 Diccionario de datos.

A continuación, se presenta el diccionario de datos en tablas organizadas, con el nombre de la tabla, su descripción y otros datos necesarios. Los cuales se tomaron del modelo ER (Entidad-Relación) de la base de datos.

Tabla 4.13. Diccionario de datos.

Tabla: piso				
Descripción: almacena los datos principales del piso				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	nro_piso	Número entero único que identifica al piso.	INT
2		descripción	Describe la funcionalidad del piso con respecto al edificio.	VARCHAR(80)

Tabla: apartamento				
Descripción: almacena los datos principales del apartamento.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id_apartamento	Número entero único que identifica al apartamento.	INT
2		propietario	Nombre del propietario del apartamento.	VARCHAR(80)
3		nro_piso	Dependencia de piso.	INT

Tabla: clase_residente				
Descripción: Almacena los tipos de parentesco que un residente puede tener con respecto al propietario.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id	Número entero único que identifica a la clase de residente.	INT
2		descripción	Describe el parentesco del residente con respecto al propietario.	VARCHAR(60)

Tabla: residente				
Descripción: almacena los datos personales de los residentes.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	ci	Número entero único que identifica al apartamento.	INT
2		nombre	Nombre del residente.	MEDIUM TEXT
3		nacimiento	Fecha de nacimiento.	DATE
4		teléfono	Número de teléfono celular.	INT(11)
5		id_apartamento	Número de apartamento al cual pertenece el residente	INT
6		clase	Parentesco del residente con respecto al propietario	INT

Tabla: ambiente				
Descripción: Almacena los datos principales de un ambiente.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id_ambiente	Número entero único que identifica al ambiente.	INT
2		tipo	Número relacionado con el tipo de ambiente al cual pertenece.	INT
3		apartamento	Número de apartamento al cual pertenece el ambiente	INT

Tabla: tipos_ambiente				
Descripción: Almacena los tipos de ambiente que puede existir (baño, cocina, etc.).				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id_tipo	Número entero único que identifica al tipo de ambiente.	INT
2		descripción	Describe la funcionalidad del ambiente.	VARCHAR(80)

Tabla: uso_sensor				
Descripción: Almacena los sensores que existen en un ambiente específico del edificio.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1		ambiente	Número relacionado a un ambiente del edificio.	INT
2		sensor	Número relacionado a un sensor en particular	INT

Tabla: sensor				
Descripción: Almacena los tipos de sensores que puede haber en la edificación.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id_sensor	Número entero único que identifica al tipo de sensor.	INT
2		descripción	Descripción breve del tipo de sensor.	VARCHAR(60)

Tabla: servicio				
Descripción: Almacena los tipos de servicio que puede haber en la edificación (gas, agua, electricidad, etc).				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id	Número entero único que identifica al tipo de servicio.	INT
2		descripción	Describe brevemente el tipo de servicio.	VARCHAR(45)

Tabla: servicio_por_apartamento				
Descripción: Almacena los servicios que existen en un apartamento específico del edificio.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id	Número entero único que identifica la relación del servicio respecto al apartamento.	INT
2		cod_servicio	Número relacionado a un servicio específico.	INT
3		id_apartamento	Número relacionado a un apartamento específico.	INT

Tabla: consumo				
Descripción: Almacena el consumo por apartamento de un servicio en un determinado mes.				
Nro.	Clave.	Dato.	Descripción.	Tipo(longitud).
1	primaria	id	Número entero único que identifica al consumo de un servicio en un determinado mes.	INT
2		fecha	Mes en el cual un servicio fue consumido.	DATE
3		promedio	Consumo promedio de un servicio en un mes	FLOAT
4		servicios_dep	Número relacionado a un servicio por departamento específico	INT

Fuente: Elaboración propia.

4.7.5 Fase de transición.

4.7.5.1 Pruebas de integración.

Esta herramienta se utiliza para especificar la comunicación y el comportamiento del sistema mediante sus iteraciones. Las pruebas de integración son realizadas para verificar que los módulos interactúen entre sí de manera apropiada después de haber sido integrados.

En la figura 4.7 se muestra la prueba de integración de crear una habitación. La secuencia de acciones realizadas durante las pruebas de integración consiste en agregar a la base de datos dicha habitación.

Se inicia en la iteración I, donde se crea el piso, se registra el piso con los datos principales y una vez registrado se pasa a la iteración II donde el piso crea el apartamento y el administrador registra los datos principales, entonces se pasa a la iteración III donde el apartamento crea una habitación, registra y almacena su respectivo id y el tipo de funcionalidad que tendrá la habitación.

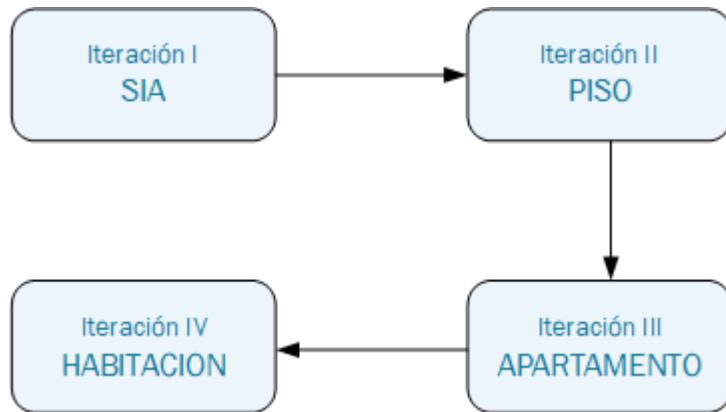


Figura 4.7: Prueba de integración, primera interacción entre módulos I, II, III, IV.

Fuente: Elaboración propia.

La figura 4.8 muestra la prueba de integración cuando el administrador solicita un servicio domótico.

En la iteración I El administrador solicita un servicio para una habitación al asistente virtual, el módulo 1 verifica el piso donde se requiere dicho servicio y manda la información a ese piso.

En la iteración II el piso verifica el apartamento que requiere el servicio y manda la información a ese apartamento.

En la iteración III el apartamento verifica la habitación que requiere el servicio y manda la información a esa habitación.

Finalmente, en la iteración IV la habitación ejecuta el servicio y devuelve el mensaje de haber concretado exitosamente el servicio a su módulo superior, y este devuelve a su superior, hasta llegar al primer módulo donde almacena las actividades en la base de datos y/o los logs.

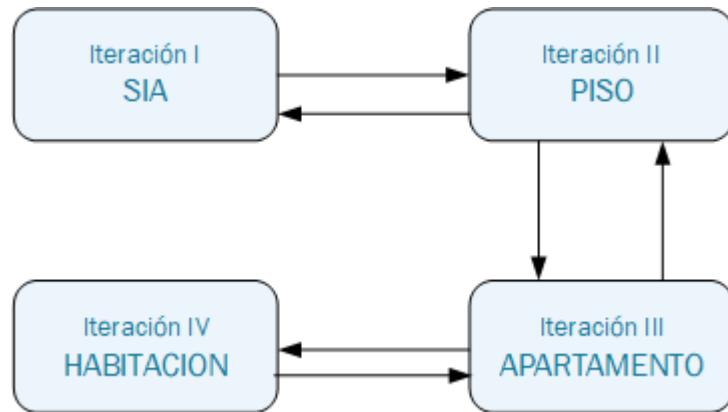


Figura 4.8: Prueba de integración, segunda interacción entre módulos I, II, III, IV.

Fuente: Elaboración propia.

La figura 4.9 muestra la prueba de integración cuando el piso detecta la falta de un servicio básico.

En la iteración II el piso detecta la falta de un servicio básico e informa a el módulo principal (SIA) y también a los apartamentos, donde la iteración III que son los apartamentos informan a los residentes la falta del servicio a través del asistente virtual y en la iteración I se almacena la información y la fecha de la irregularidad para posteriores análisis.

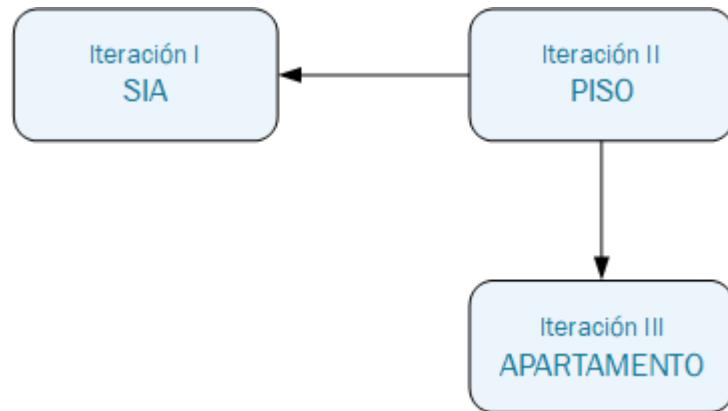


Figura 4.9: Prueba de integración, interacción entre módulos I, II, III.

Fuente: Elaboración propia.

La figura 4.10 muestra la prueba de integración cuando un sensor de una habitación detecta una irregularidad.

Comienza en la iteración IV donde el sensor detecta una irregularidad en una habitación y el asistente virtual informa sobre la irregularidad a los residentes y la habitación pasa su dirección y la información de la irregularidad al módulo apartamento.

En la iteración III el módulo apartamento envía su dirección y la información de la irregularidad al módulo piso.

En la iteración II el módulo piso envía su dirección y la información de la irregularidad al módulo SIA.

Finalmente, en la iteración I el módulo SIA almacena la información de la irregularidad, la fecha del suceso y la dirección de donde proviene en la base de datos y/o en logs, posteriormente se informa al administrador de la irregularidad a través del asistente virtual.

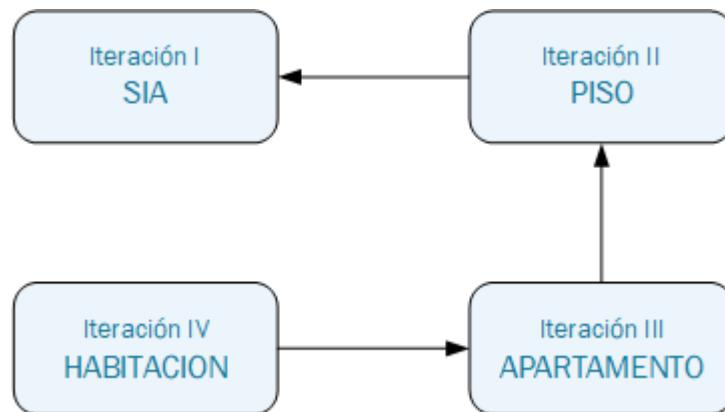


Figura 4.10: Prueba de integración, tercera interacción entre módulos I, II, III, IV.
Fuente: Elaboración propia.

4.7.5.2 Cambios en las edificaciones debido al sistema propuesto.

➤ Posibles dificultades:

- Falta de conocimiento de los comandos a los cuales responde el asistente virtual.
- Falla en los sensores que provocaría la desinformación de irregularidades.
- Manejo de información muy personal de los residentes a cargo de un administrador.

- Falta de presupuesto para contratar a un administrador o encargado de sistema para el mantenimiento y soporte del sistema.
- Falta de asignación de un presupuesto o de recursos para la implantación del sistema.
- Resistencia al cambio.

➤ **Beneficios del sistema.**

- Mejor calidad de vida para los residentes.
- Mayor control en el manejo de información del edificio.
- Eficiencia en el uso de recursos y servicios.
- Aumento de la seguridad en el edificio.
- Ahorro de tiempo para los residentes y el administrativo.

4.7.5.3 Aspectos de implantación.

Para implementar el sistema propuesto en una edificación es necesario determinar los siguientes procesos:

• **Acondicionamiento de las instalaciones.**

Acondicionar las instalaciones donde será implantado el sistema teniendo condiciones óptimas para la puesta de sensores, actuadores, hardware y software que permitan la operatividad del sistema.

• **Migración y pruebas.**

Obtener los datos reales que maneja la edificación y los residentes, desarrollar una serie de pruebas que permitan la verificación y corrección de errores.

- **Capacitación del personal.**

Capacitar al personal administrativo y a los residentes sobre el uso y manejo del sistema y los comandos de interacción con el asistente virtual.

- **Puesta en marcha.**

Es un proceso que sirve para que las distintas partes de un sistema involucrado en el proyecto se desenvuelvan según las especificaciones del cliente, que en este caso en particular sería el propietario del edificio.

Para esto se realizó un prototipo integrado de una edificación donde se instaló el software del sistema de manera conjunta con el hardware para su posterior utilización.

- **Ficha técnica del consumo de recursos.**

Finalmente se presenta una tabla mostrando los recursos consumidos por el software SIA en el sistema operativo Raspbian.

Tabla 4.14. Ficha técnica del consumo de recursos.

Peso del software SIA.	58KB
Uso de memoria RAM.	23MB
Peso inicial de la base de datos.	328.125KB
Peso diario de los registros.	2304KB

Fuente: Elaboración propia.

Capítulo 5

EVALUACIÓN ECONÓMICA.

5 EVALUACIÓN ECONÓMICA.

En este capítulo se analizará mediante tablas la valoración económica del conjunto de componentes, software y equipamiento que fueron previstos para la implementación del prototipo del proyecto.

5.1 ESTRUCTURA FÍSICA DEL PROTOTIPO.

Todos los materiales e insumos complementarios necesarios para la construcción de la estructura física se adquirieron del mercado local.

La tabla 5.1 describe los materiales usados para la implementación de la parte estructural del proyecto, con los precios que se obtienen a nivel de consumidor final en el mercado local.

Tabla 5.1. Costos de materiales para la estructura.

Material.	Cantidad [u].	Precio [Bs].
Tablero de melamina 80X60	1	40
Lampara fluorescente espiral.	4	56
Distribuidor de cables	1	25
Socket de lámpara	4	20
Acido perclorato férrico	50 [g]	15
Tornillos 4X12 [mm]	16	4
Cable bipolar n°12	12	30
Monto Total:		190

Fuente: Elaboración propia.

5.2 ACTUADORES Y PERIFÉRICOS DE SALIDA.

La tabla 5.2 muestra un detalle de los actuadores y periféricos de salida usados en el prototipo, incluye algunas características principales y su respectivo precio.

Tabla 5.2. Costos de actuadores usados en el prototipo.

Material.	Cantidad [u].	Precio [Bs].
Motor de drenaje 220[VAC] 50[hz] 0.21[A] 3000 [rpm]	1	450
Válvula solenoide 220[VAC] 50[hz] 0.02-0.8[Mpa]	1	75
Modulo Relé de 8 canales 220 VAC/ 5VDC	1	80
Total:		605

Nota: La información de algunos precios de esta tabla fueron extraídos del sitio:
<https://tienda.sawers.com.bo/index.php?route=product/search&search=valvula%20solenoid>

Fuente: Elaboración propia.

5.3 SENSORES Y PERIFÉRICOS DE ENTRADA.

La tabla 5.3 muestra un detalle de los sensores y periféricos de entrada usados en el prototipo, incluye algunas características principales y su respectivo precio.

Tabla 5.3. Costos de sensores y periféricos de entrada usados en el prototipo.

Material.	Cantidad [u].	Precio [Bs].
Sensor MQ6 5VDC	1	35
Sensor MQ2 5VDC	1	35
Sensor DFR0033 3.3VDC	1	30
Sensor TMP102 3.3VDC	1	28
Micrófono Trust GXT 212 Micro USB/JACK	1	200
Sensor de nivel de agua vertical	2	36
Adaptador USB/JACK	1	25
Total:		389

Nota: La información de algunos precios de esta tabla pueden consultarse en el sitio:

<https://tienda.sawers.com.bo/>

Fuente: Elaboración propia.

5.4 SISTEMA EMBEBIDO.

En esta sección se muestra el precio de los componentes que integran el sistema embebido, donde se considera el microcomputador, el microcontrolador y la tarjeta de distribución SIA.

Tabla 5.4. Costos de los materiales usados para la construcción de la tarjeta SIA.

Material.	Cantidad [u.]	Precio [Bs.]
Baquelita de cobre.	1	10
Resistencia 200 [Ω]; $\frac{1}{2}$ [w]	8	2.40
Resistencia 1 [Ω]; $\frac{1}{2}$ [w]	6	1.80
Resistencia 10 [$k\Omega$]; $\frac{1}{2}$ [w]	1	0.50
Transistor 2n222 NPN	7	10.50
Diodo emisor infrarrojo 5 mm; 1.5 V	1	2.50
Bornera de conectores de 3 entradas	3	12
Bornera de conectores de 4 entradas	1	4.5
Espadín macho de 40 pines	1	4
Total:		48.20

Nota: La información de algunos precios de esta tabla pueden consultarse en el sitio:

<https://tienda.sawers.com.bo/>

Fuente: Elaboración propia.

Tabla 5.5. Costo del microcontrolador y dispositivos relacionados a su funcionamiento.

Material.	Cantidad [u.]	Precio [Bs.]
Arduino Mega 2560	1	90
Fuente de poder 5V 2 ^a	1	30
Cable USB serial	1	10
Total:		130

Nota: La información de algunos precios de esta tabla pueden consultarse en el sitio:

<https://tienda.sawers.com.bo/>

Fuente: Elaboración propia.

Cabe mencionar que el sistema podría funcionar de manera regular con un microcontrolador Arduino nano reduciendo su costo en aproximadamente un 60 %. Se utilizó Arduino mega debido a que ya se tenía a disposición y no era necesaria la compra del microcontrolador para el proyecto.

Tabla 5.6. Costo del microcomputador y dispositivos relacionados a su funcionamiento.

Material.	Cantidad [u].	Precio [Bs].
Raspberry Pi 4 (4GB)	1	790
Memoria SD Kingston (16GB)	1	70
Fuente de poder 5V 3 ^a	1	50
Extensión GPIO adaptador	1	30
Total:		940

Nota: La información de algunos precios de esta tabla pueden consultarse en los sitios:

<https://tienda.sawers.com.bo/>

<https://tecbolivia.com/index.php/tienda-virtual>

Fuente: Elaboración propia.

A continuación, en la tabla 5.7 se muestra una estimación de costos de un ordenador convencional estándar de escritorio para realizar posteriores comparaciones.

Tabla 5.7. Tabla estimativa de costos de componentes para un ordenador estándar.

Material.	Marca	Cantidad [u].	Precio [\$].
Tarjeta madre B365M PRO	MSI	1	90
Procesador I3-540	INTEL	1	65
Memoria RAM 4G DDR4	Mushkin Essential	1	34
Disco HDD Western Digital 1TB	WD	1	60
Fuente de poder 900 W	DELUX	1	26
Case tamaño media torre	S/M	1	51
Total [Bs]	2249.27	Total [\$]	326

Nota: La información de algunos precios de esta tabla pueden consultarse en la app:

<https://play.google.com/store/apps/details?id=com.msrgcomputers.xdevsbo>

Fuente: Elaboración propia

Se realiza una comparación del costo del microcomputador (Raspberry pi) usado como máquina cliente con respecto al costo estimativo de un ordenador estándar que podría usarse como máquina cliente.

Se puede notar que el precio del Raspberry pi es muy bajo a comparación del ordenador estándar, además de que satisface exitosamente los requerimientos del sistema, concluyendo así el ahorro significativo que se tuvo al seleccionar dicho microcomputador, dando paso a la vez a una mejor viabilidad económica del proyecto.

A continuación, en la tabla 5.8 se presentan los precios de algunos componentes usados en el proyecto con otros componentes similares alternativos que se encuentran en el mercado nacional.

Tabla 5.8. Tabla comparativa de precios.

Componentes usados en el proyecto	Precio[Bs]	Componentes alternativos.	Precio[Bs]	Diferencia.
Ordenador cliente Raspberry	940	Ordenador cliente estándar	2249.27	1309.27
Sensor de monóxido de carbono MQ-6	35	Sensor de humo fotoeléctrico Mircom	325	290
Sensor de GLP MQ-2	35	Sensor de gas natural Orvibo	295	260
Sensor magnético DFR0033	30	Contacto magnético inalámbrico Everspring	154	124
Sensor de temperatura TMP102	28	Sensor de temperatura Orvibo	260	232

Nota: La información de algunos precios de esta tabla pueden consultarse en el sitio:

<https://tienda.sawers.com.bo/>

<https://www.digicorp.com.bo/>

Fuente: Elaboración propia.

5.5 COMPONENTES DESPRECIABLES.

En la tabla 5.9 se muestra los componentes, dispositivos y materiales que se usaron en el prototipo, sin embargo, sus costos son despreciables debido a que ya se tenía a disponibilidad antes del inicio del proyecto y/o no se adquirieron exclusivamente para el desarrollo del prototipo del proyecto.

Tabla 5.9 Componentes de costos despreciables.

Materiales.
Televisor de rayos catódicos.
Parlantes.
Monitor.
Teclado.
Mouse.
Cubetas de agua.
Cable UTP.
Cable nº16 multifilar.
Herramientas en general.

Fuente: Elaboración propia

5.6 PRESUPUESTO TOTAL.

Finalmente, en la tabla 5.10 se muestran los costos totales y la sumatoria de estos.

Tabla 5.10 Costos totales.

Descripción.	Precio [Bs].
Materiales para la estructura.	190
Actuadores y periféricos de salida.	605
Sensores y periféricos de entrada.	389
Tarjeta de distribución SIA.	48.20
Costo del microcontrolador y sus componentes.	130
Costo del microcomputador y sus componentes.	940
Costo total del prototipo:	2302.2

Fuente: Elaboración propia

Capítulo VI

CONCLUSIONES Y RECOMENDACIONES.

6 CONCLUSIONES Y RECOMENDACIONES.

6.1 CONCLUSIONES.

Por lo desarrollado en el presente proyecto del sistema de inteligencia artificial para la automatización de edificaciones se puede concluir lo siguiente:

- El prototipo fue realizado de manera exitosa obteniendo un sistema funcional e inteligente que optimiza el uso de recursos y/o servicios básicos, aumenta la seguridad de la edificación demostrativa, y presta servicios que mejoran el confort y la calidad de vida de los usuarios, demostrando a la vez la factibilidad del proyecto propuesto, sin embargo, más allá del prototipo demostrativo no se pudo implementar el sistema en un ambiente real, lo cual lo hace sujeto a fallas por comunicación de datos a largas distancias y fallas de los dispositivos por exposición a ambientes pesados.
- Se consiguió desarrollar un sistema robusto y modular, que cuenta con una base de datos expansible, un software de gestión flexible y ordenada, un asistente virtual para la atención e interacción con los residentes y se logró integrar dispositivos, sensores y actuadores de diferentes tecnologías y protocolos de comunicación.
- Si bien se logró integrar dispositivos de diferentes tecnologías, es necesario hacer una breve investigación del dispositivo y su compatibilidad con los diferentes componentes del sistema.
- El asistente virtual presenta latencia de respuesta debido a carencias en el hardware y dependencia del asistente de este mismo hardware, para la corrección de esta falencia es necesario desarrollar una tarjeta amplificadora, lo que implica un proyecto independiente y no procedente a este proyecto.
- Se demostró la viabilidad económica del proyecto, exponiendo los bajos precios de los componentes seleccionados y comparando algunos componentes con otros productos equiparables en características, pero con un costo mayor a los componentes usados en este proyecto.

6.2 RECOMENDACIONES.

- Se propone hacer énfasis en la investigación de redes neuronales y síntesis de habla, temas procedentes del área de inteligencia artificial.
- Es pertinente investigar de manera más profunda las tecnologías IoT actuales y nuevas, y sus aplicaciones para mejorar los sistemas domóticos.
- Resulta importante desarrollar tarjetas de expansión para el manejo masivo de sensores, actuadores y otros periféricos, ya que el microcomputador usado en este proyecto tenía una limitada cantidad de puertos para la interacción con periféricos.
- Analizar de manera exhaustiva los diferentes protocolos de comunicación vigentes para implementar a un sistema que gestione sensores, actuadores y otros periféricos, y lograr que sea más estable y eficiente.
- Se recomienda plantear proyectos para la gestión y centralización de información de varias edificaciones, incorporándose de esta manera en el área creciente de Smart Cities o ciudades inteligentes.

BIBLIOGRAFÍA.

BIBLIOGRAFÍA.

Russell S. y Norvig P. (2004) Inteligencia artificial. Un enfoque moderno. (Traducción de: Corchado J y Rubio F.). España, Madrid: Editorial Pearson Educación S.A.

Benchimol D. (2011) Microcontroladores. Argentina, Buenos Aires: Editorial Fox Andina.

Boylestad R. (2009) Electrónica. Teoría de circuitos y dispositivos electrónicos. (Traducción de: Navarro R.). México, México DF: Editorial Pearson Educación S.A.

Tocci R., Widmer S. y Moss G. (2007) Sistemas digitales. Principios y aplicaciones (Traducción de: Romero A.). México, México DF: Editorial Pearson Educación S.A.

Corona L., Abarca G. y Mares J. (2014) Sensores y actuadores. Aplicaciones con Arduino. México, México DF: Grupo editorial Patria S.A.

Gonzales R. (s.f.) Python para todos. España: Creative Commons Reconocimiento 2.5 España.

Marzal V., García I. y García P. (2014) Introducción a la programación con Python 3. España: Editorial de la Universidad Jaume I.

Wikipedia.(s.f.). Reconocimiento del Habla. Recuperado el 10 de agosto del 2021 de https://es.wikipedia.org/wiki/Reconocimiento_del_habla#Usos_y_aplicaciones

Wikipedia. (s.f.). Raspberry Pi. Recuperado el 10 de agosto del 2021 de https://es.wikipedia.org/wiki/Raspberry_Pi

Wikipedia.(s.f.). Arduino. Recuperado el 11 de agosto del 2021 de <https://es.wikipedia.org/wiki/Arduino>

Wikipedia.(s.f.). Sensor. Consultado el 11 de agosto del 2021 de <https://es.wikipedia.org/wiki/Sensor>

ONCE Centro de tiflotecnia e innovación. (2019). Informe sobre asistentes virtuales. <https://www.once.es/cti/biblioteca/Accesibilidad/Asistentes%20virtuales/Informe%20Asistentes%20Virtuales.pdf>

Universidad de Oviedo. (2007). Automatización Integral de Edificios. <http://isa.uniovi.es/docencia/AutomEdificios/transparencias/Generalidades2.pdf>

Universidad Politécnica de Madrid. (2016). Microcomputadoras y microprocesadores. Fernando Zaes Vaca. http://oa.upm.es/22244/1/Micropocesadores_y_Microcomputadoras.pdf

Arduino. (s.f.). Introducción. Consultado el 11 de agosto del 2021 de <https://www.arduino.cc/en/guide/introduction>

Ciberaula. (s.f.). Ventajas y desventajas de Linux. Consultado el 13 de agosto del 2021 de https://linux.ciberaula.com/articulo/ventajas_inconvenientes_linux/

Tecnoloco. (s.f.). 5 grandes IDEs de Raspberry Pi para programadores y estudiantes. Consultado el 10 de agosto del 2021 de <https://tecnoloco.istocks.club/5-grandess-ide-de-raspberry-pi-para-programadores-y-estudiantes/2021-04-29/>

Instructables circuits. (s.f.). Creating a Raspberry Pi universal remote with LIRC. Consultado el 15 de mayo del 2021 de <https://www.instructables.com/Creating-a-Raspberry-Pi-Universal-Remote-With-LIRC/>

Dev Kimchi. (s.f.). Turning Raspberry Pi into remote controller. Consultado el 15 de mayo del 2021 de <https://devkimchi.com/2020/08/12/turning-raspberry-pi-into-remote-controller/>

StackExchange. (s.f.) Raspberry Pi 3 not lirc not running/working. Consultado el 15 de mayo del 2021 de <https://raspberrypi.stackexchange.com/questions/81876/raspberry-pi-3-not-lirc-not-running-working>

Medium. (s.f.). How to create a Telegram bot, and send messages with python. Consultado el 06 de junio del 2021 de https://medium.com/@ManHay_Hong/how-to-create-a-telegram-bot-and-send-messages-with-python-4cf314d9fa3e

Telegram. (s.f.). Bots: An introduction for developers. Consultado el 06 de junio del 2021 de <https://core.telegram.org/bots#inline-mode>

Github. (s.f.). Google speech API example. Consultado el 20 de marzo del 2020 de <https://gist.github.com/mertyildiran/957b8c9f7631f6ab7f21>

Python Package Index. (s.f.). Pyserial 3.0 documentation. Consultado el 20 de agosto del 2020 de <https://pythonhosted.org/pyserial/#>

PIBITS. (s.f.). TMP102 sensor and raspberry pi python example. Consultado el 10 de agosto del 2021 de <http://www.pibits.net/code/tmp102-sensor-and-raspberry-pi-python-example.php>

ESPOL. (s.f.). Evaluación, análisis y comparación del rendimiento de programas de procesamiento masivo implementados usando lenguajes de programación Java, Python y C++ sobre la plataforma Hadoop para clústeres de varios tamaños. Consultado el 15 de noviembre del 2021 de <https://www.dspace.espol.edu.ec/handle/123456789/43527>

Keep coding, Tech School. (s.f). Como programar inteligencia artificial [5 lenguajes]. consultado el 17 de noviembre de 2021 de <https://keepcoding.io/blog/como-programar-inteligencia-artificial/>

CampusMVP. (05 de agosto del 2021). Los 4 mejores lenguajes de programación para inteligencia artificial/ machine learning. <https://www.campusmvp.es/recursos/post/los-4-mejores-lenguajes-de-programacion-para-inteligencia-artificial-machine-learning.aspx>

Raspberry Pi Foundation. (s.f.). Documentacion de Raspberry. Consultado el 18 de noviembre de 2021 de <https://www.raspberrypi.com/documentation/computers/os.html#python>

ANEXOS.

ANEXOS.

ANEXO A: Comandos por voz.

Para el inicio de cualquier servicio antes debe mencionarse el nombre del asistente virtual que es “Alejandra” una vez que Alejandra confirme que lo escucho y está atenta, usted puede pronunciar los comando que se muestran en la lista.

Tabla A.1. Lista de comandos.

Dormitorio.	Servicio.
Enciende la luz del dormitorio	Luz
Prende la luz del dormitorio	Luz
Luces dormitorio	Luz
Encender luz dormitorio	Luz
Apaga la luz del dormitorio	Luz
Apagar la luz del dormitorio	Luz
Fuera luces dormitorio	Luz
Despiértame a las “hora” y “minuto”	Despertador
Temperatura	Temperatura
Baño	
Enciende la luz del baño	Luz
Prende la luz del baño	Luz
Luces baño	Luz
Encender luz baño	Luz
Apaga la luz del baño	Luz
Apagar la luz del baño	Luz
Fuera luces baño	Luz
Ducha	Aqua
Enciende la ducha	Aqua
Abre la ducha	Aqua
Apaga la ducha	Aqua
Cierra la ducha	Aqua
Cocina	
Enciende la luz de la cocina	Luz
Prende la luz de la cocina	Luz
Luces cocina	Luz
Encender luz cocina	Luz
Apaga la luz de la cocina	Luz
Apagar la luz de la cocina	Luz
Fuera luces cocina	Luz
Abre la llave de la cocina	Aqua
Enciende la llave de la cocina	Aqua

Agua cocina	Aqua
Apaga la llave de la cocina	Aqua
Cierra la llave de la cocina	Aqua
Sala	
Enciende la luz de la cocina	Luz
Prende la luz de la cocina	Luz
Luces cocina	Luz
Encender luz cocina	Luz
Apaga la luz de la cocina	Luz
Apagar la luz de la cocina	Luz
Fuera luces cocina	Luz
Cambia de canal	Televisión
Otro canal	Televisión
Cambia de canal atrás	Televisión
Baja canal	Televisión
Sube el volumen	Televisión
Aumenta el volumen	Televisión
Baja el volumen	Televisión
Disminuye el volumen	Televisión
Activar la seguridad	Seguridad
Desactivar la seguridad	Seguridad
Apartamento	
Hora	Tiempo
Dame la hora	Tiempo
Qué hora es	Tiempo
Qué hora tienes	Tiempo
Genera un informe de los residentes	Información
Genera un informe de los habitantes	Información
Manda un informe de los residentes	Información
Manda un informe de los habitantes	Información
Genera un informe del consumo de agua	Informacion
Informes habitantes	Información
Informes residentes	Información
Estoy herido	Auxilio
Auxilio	Auxilio
Pide ayuda	Auxilio
Administración	
Genera un informe de actividades	Información
Genera un informe general	Información
manda un informe de actividades	Información
manda un informe general	Información
informe general	Información
registro de actividades	Información
informe de actividades	Información

Música.	
Comando.	Artista-Título canción.
Nirvana	Nirvana - Smells Like Teen Spirit
Después de ti	CODA 3 - DESPUÉS DE TI
Hada y mago	Rata Blanca - La Leyenda del Hada y El Mago
Mujer amante	Rata Blanca - Mujer amante
Religión	R.E.M. - Losing My Religion
Subidón	Fey - Subidon
Montaner	Ricardo Montaner - Dejame llorar
Eclipse	LISSETTE - Eclipse Total Del Amor
Hijo de la luna	Mecano - Hijo de la luna
Sera	Ricardo Montaner - Sera
Adán y Eva	Paulo Londra – Adán y Eva
Desconocidos	Mau y Ricky, Manuel Turizo, Camilo - Desconocidos
Por perro	Sebastián Yatra - Por Perro
Sebastián	Sebastián Yatra - Por Perro
Se preparó	Ozuna - Se preparo
Otra vez	Zion & Lennox ft. J Balvin - Otra Vez

ANEXO B: Historial de comandos de paquetes, módulos y librerías instaladas y/o usadas.

```

> sudo apt-get install audacity
> sudo apt-get install update
> sudo apt-get install python3
> sudo apt-get install pyDev
> sudo apt-get install idle
> sudo pip-3.2 install SpeechRecognition
> sudo pip install SpeechRecognition
> pip install SpeechRecognition
> pip search "speech recognition"
> sudo pip3 install SpeechRecognition
> install pyaudio
> install apt pyaudio
> python -m speech_recognition
> pip install pyaudio
> sudo pip install pyaudio
> sudo pip3 install pyaudio
> python pip install python-pyaudio
> sudo apt-get install python-pyaudio python3-pyaudio
> pip install SpeechRecognition
> python

```

```
> sudo apt-get install portaudio19-dev python-all-dev python3-all-dev && sudo pip3
install      pyaudio
> python -m speech_recognition
> python -m speech_recognition
> sudo apt-get install flac
> python -m speech_recognition
> recognizer_instance.recognize_google
> sudo vi recognizer_instance.recognize_google
> recognizer_instance.recognize_sphinx
> recognizer_instance.recognize_google
> python -m speech_recognition
> sudo apt-get install python-pygame
> sudo apt-get install spider
> pip install playsound
> sudo pip install playsound
> pip install python-telegram-bot
> sudo pip install python-telegram-bot
> pip3 install python-telegram-bot
> pip install python-telegram-bot
> sudo apt-get install vim
> sudo pip install pyttsx3
> install pyttsx3
> pip install pyttsx3
> pip install pyttsx3
> sudo apt-get install espeak
> sudo apt-get install arduino
> sudo apt-get install lirc
> sudo apt-get install lirc3
> sudo apt-get install lirc
> sudo vim modules
> sudo vim lircd.conf
> sudo vim hardware.conf
> sudo /etc/init.d/lirc stop
> sudo /etc/init.d/lirc start
> sudo vim /etc/modprobe.d/ir-remote.conf
> sudo modprobe lirc_rpi
> sudo modprobe
> sudo apt-get install lirc
> mode2 -d /dev/lirc0
> sudo /etc/init.d/lircmd restart
> sudo /etc/init.d/lircd restart
> cd /dev/lirc0
> sudo vim /etc/lirc/lircd.conf
> sudo mv /etc/lirc/lircd.conf /etc/lirc/lircd_orig.conf
```

```
> sudo vim /etc/lirc/lircd.conf
> sudo cp /etc/lirc/lircd_orig.conf /etc/lirc/lircd.conf
> sudo vim /etc/lirc/lircd.conf
> sudo /etc/lirc/lircd restart
> sudo /etc/init.d/lircd restart
> cd /etc/lirc/
> irrecord --list-namespace
> irsend
> irsend LIST
> irsend LIST lircd.conf
> lirc --version
> lircd --version
> sudo /etc/init.d/lircd status
> sudo /etc/init.d/lircd status -l
> sudo /etc/init.d/lircd -l status
> sudo /etc/init.d/lircd status
> sudo -l /etc/init.d/lircd status
> sudo /etc/init.d/lircd status
> sudo /etc/init.d/lircd restart
> sudo /etc/init.d/lircd status
> sudo vim /etc/lirc/lirc_options.conf
> sudo vim /etc/lirc/hardware.conf
> sudo reboot
> sudo /etc/init.d/lircd estado
> sudo /etc/init.d/lircd estatus
> sudo /etc/init.d/lircd status
> irsend LIST lircd
> irsend LIST
> irsend list "" ""
> irsend list SONY-TV ""
> irsend SEND_ONCE SONY-TV KEY_VOLUMEUP
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend SEND_ONCE sony-tv KEY_CHANNELDOWN
> irsend SONY-TV KEY_CHANNELDOWN
> sudo /etc/init.d/lircd restart
> irsend SONY-TV KEY_CHANNELDOWN
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> sudo systemctl restart lircd
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> sudo /etc/init.d/lircd status
> sudo /etc/init.d/lircd restart
> sudo /etc/init.d/lircd status
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> sudo /etc/init.d/lircd status
```

```
> sudo /etc/init.d/lircd restart
> sudo /etc/init.d/lircd status
> sudo /etc/init.d/lirc stop
> sudo /etc/init.d/lircd stop
> mode2 -d /dev/lirc0
> irrecord -d /dev/lirc0 ~/lircd.conf
> sudo irrecord -d /dev/lirc0 ~/lircd.conf
> sudo /etc/init.d/lircd start
> sudo irrecord -d /dev/lirc0 ~/lircd.conf
> sudo /etc/init.d/lirc stop
> mode2 -d / dev / lirc0
> mode2 -d /dev/lirc0
> sudo irrecord -d /dev/lirc0 ~/lircd.conf
> sudo /etc/init.d/lircd status
> sudo /var/run/lirc/lircd
> sudo vim /boot/overlays/README
> sudo vim /boot/config.txt
> sudo apt update
> sudo apt install lirc
> /etc/lirc/lirc_options.conf
> sudo vim /etc/lirc/lirc_options.conf
> sudo systemctl reboot
> ls -l /dev/lirc0
> lsmod | grep lirc
> lsmod | egrep lirc
> lsmod | grep lirc
> systemctl status lircd.service
> systemctl status lircd.socket
> sudo systemctl start lircd socket
> sudo systemctl start lircd.service
> irrecord -n -d /dev/lirc0 ~/lircd.conf
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> sudo /etc/init.d/lircd status
> sudo systemctl restart lircd
> sudo /etc/init.d/lircd status
> irw
> sudo vim /etc/lirc/hardware.conf
> sudo rm /etc/lirc/hardware.conf
> sudo vim /etc/lirc/hardware.conf
> irsend list ""
> irsend list "" ""
> irsend list SONY-TV ""
> sudo /etc/init.d/lircd status
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
```

```

> sudo su -c "grep '^deb ' /etc/apt/sources.list | sed 's/^deb/deb-src/g' >
  /etc/apt/sources.list.d/deb-src.list"
> sudo apt install -y vim devscripts dh-exec doxygen expect libasound2-dev libftdi1-dev
  libsystemd-dev libudev-dev libusb-1.0-0-dev libusb-dev man2html-base portaudio19-
  dev socat xsltproc python3-yaml dh-python libx11-dev python3-dev python3-
  setuptools
> mkdir ~/lirc-src
> cd ~/lirc-src
> apt source lirc
> wget https://raw.githubusercontent.com/neuralassembly/raspi/master/lirc-gpio-ir-
  0.10.patch
> patch -p0 -i lirc-gpio-ir-0.10.patch
> cd lirc-0.10.1
> debuild -uc -us -b
> cd ~/lirc-src
> sudo apt install -y ./liblirc0_0.10.1-6.2~deb10u1_armhf.deb ./liblircclient0_0.10.1-
  6.2~deb10u1_armhf.deb ./lirc_0.10.1-6.2~deb10u1_armhf.deb
> sudo vim /etc/lirc/lirc_options.conf
> sudo systemctl restart lircd
> sudo vim /boot/config.txt
> sudo shutdown -r 0
> sudo systemctl stop lircd
> irrecord --list-namespace
> irrecord -d /dev/lirc1 ~/lircd.conf
> cd /etc/lirc/
> sudo vim sony.lircd.conf
> sudo vim /etc/lirc/lirc_options.conf
> sudo systemctl start lircd
> irsend """
> irsend list """
> irsend list SONY-TV """
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend list SONY-TV """
> irsend SEND_ONCE SONY-TV KEY_CHANNELUP
> irsend list SONY-TV """
> irsend SEND_ONCE SONY-TV KEY CHANNELDOWN
> sudo vim /etc/lirc/lirc_options.conf
> sudo vim /boot/config.txt
> irsend SEND_ONCE SONY-TV KEY CHANNELDOWN
> sudo vim /boot/config.txt
> irsend SEND_ONCE SONY-TV KEY CHANNELDOWN
> irsend SEND_ONCE SONY-TV KEY CHANNELDOWN
> irsend list SONY-TV """

```

```
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend list SONY-TV ""
> irsend SEND_ONCE SONY-TV KEY_TV
> sudo /etc/init.d/lircd status
> irsend SEND_ONCE SONY-TV KEY_TV
> sudo vim /etc/modules
> sudo /etc/init.d/lirc stop
> sudo vim /etc/lirc/lirc_options.conf
> ls -l /dev/lirc0
> lsmod | grep lirc
> systemctl status lircd.service
> systemctl status lircd.socket
> sudo systemctl restart lircd
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN -t
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> sudo vim /dev/lirc0
> cd /etc/lirc/
> sudo vim lircd.conf
> sudo vim sonytv.lircd.conf
> sudo mv sonytv.lircd.conf /etc/lirc/lircd.conf
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend list "" ""
> irsend list SONY_V164G/TV ""
> irsend list SONY_V164G/TV KEY_VOLUMEUP
> irsend SEND_ONCE SONY_V164G/TV KEY_VOLUMEUP
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend list SONY_V164G/TV ""
> irsend SEND_ONCE SONY_V164G/TV CH_+
> irsend SEND_ONCE SONY_V164G/TV KEY_1
> irsend SEND_ONCE SONY_V164G/TV KEY_3
> irsend SEND_ONCE SONY_V164G/TV CH_+
> sudo vim sonytv.lircd.conf
> sudo mv sonytv.lircd.conf /etc/lirc/lircd.conf
> irsend list "" ""
> irsend list SONY-TV ""
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
> irsend SEND_ONCE SONY-TV SONY-TV
> irsend SEND_ONCE SONY-TV KEY_TV
> irsend SEND_ONCE SONY-TV KEY_CHANNELDOWN
```

```

> irsend SEND_ONCE SONY-TV KEY_T
> irsend SEND_ONCE SONY-TV KEY_KEY_1
> irsend SEND_ONCE SONY-TV KEY_1
> irsend SEND_ONCE SONY-TV KEY_3
> irsend SEND_ONCE SONY-TV KEY_1
> irsend list SONY-TV ""
> sudo apt-get install python3-smbus
> uname -a
> uname
> free
> df
> df -h
> free -h
> pip install opencv-python
> pip install pyautogui
> pip install pyperclip
> pip install selenium
> sudo pip3 install selenium
> chromium-browser --version
> chromium-browser --version
> sudo pip3 install fpdf

```

ANEXO C: Código fuente.

Módulo SIA.

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import datetime
import time
import pymysql
import speech_recognition as sr
import RPi.GPIO as GPIO
import pygame
from pygame.locals import *
import subprocess
import serial
import sensorTemp
import sensorHall
import tanque1
import tanque2
import consumoAgua
import botTelegram
import registrosSIA
import pdf

```

```

import ventana
import threading
import sys
from PyQt5 import QtWidgets, QtCore, QtGui

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setmode(GPIO.BCM)
GPIO.setup(24,GPIO.OUT)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
GPIO.setup(22,GPIO.OUT)
GPIO.setup(20,GPIO.OUT)
GPIO.setup(21,GPIO.OUT)

#Aqui se establece la conexion con la base de datos
db = pymysql.connect(host="localhost",
                      user="root",
                      password="Zorro1530",
                      database="sia")
cursor = db.cursor()

class habitacion:
    """Contiene todas las funciones basicas de una habitacion general"""
    def __init__(self,id_hab):
        self.__id_hab=id_hab
        self.__luces_on=False
        self.t = sensorTemp.sensorTemperatura()
        self.t.valDefine()
        self.__temperatura = int(self.t.devuelveTemperatura())
        #self.__puerta_open = sensorHall.magnetismo()
        self.__seguridad = False
        self.__ventilador_on = False
        self.__hora= datetime.datetime.now().strftime("%H"+":"+"%M"+":"+"%S")
        self.__hora_ctrl = int(datetime.datetime.now().strftime("%H"))
        self.__dic_h =
        {"00":"/home/pi/SIAuidos/ceroh.mp3","01":"/home/pi/SIAuidos/unah.mp3","02":"/home/pi/SIAuidos/dosh.mp3","03":"/home/pi/SIAuidos/tresh.mp3"
,"04":"/home/pi/SIAuidos/cuatroh.mp3","05":"/home/pi/SIAuidos/cincoh.mp3","06":"/home/pi/SIAuidos/seish.mp3","07":"/home/pi/SIAuidos/sieteh.mp3"
,"08":"/home/pi/SIAuidos/ochoh.mp3","09":"/home/pi/SIAuidos/nueveh.mp3","10":"/home/pi/SIAuidos/diezh.mp3","11":"/home/pi/SIAuidos/onceh.mp3"
,"12":"/home/pi/SIAuidos/doceh.mp3","13":"/home/pi/SIAuidos/13h.mp3","14":"/home/pi/"

```

```

SIAaudios/14h.mp3","15":"/home/pi/SIAaudios/15h.mp3","16":"/home/pi/SIAaudios/16h.mp
3"
,"17":"/home/pi/SIAaudios/17h.mp3","18":"/home/pi/SIAaudios/18h.mp3","19":"/home/pi/SI
Aaudios/20h.mp3","20":"/home/pi/SIAaudios/20h.mp3","21":"/home/pi/SIAaudios/21h.mp3"
,"22":"/home/pi/SIAaudios/22h.mp3","23":"/home/pi/SIAaudios/23h.mp3","24":"/home/pi/SI
Aaudios/24h.mp3","25":"/home/pi/SIAaudios/25h.mp3","26":"/home/pi/SIAaudios/26h.mp3"
,"27":"/home/pi/SIAaudios/27h.mp3","28":"/home/pi/SIAaudios/28h.mp3","29":"/home/pi/SI
Aaudios/29h.mp3","30":"/home/pi/SIAaudios/30h.mp3"}
```

```

def establecer_id(self,id_hab):
    self.__id_hab=id_hab
def establecer_temperatura_estandar(self,temp):
    self.__temperatura=temp
def establecer_estado_luces(self,luz):
    self.__luces_on = luz
def obtener_id_hab(self):
    return self.__id_hab
def obtener_estado_luces(self):
    return self.__luces_on
def obtener_temperatura(self, dato):
    if dato == "temperatura":
        print("la temperatura ambiente es ",self.__temperatura," grados centigrados")
        registrosSIA.saveLog(self.__hora+":consulta temperatura,"
temp:"+str(self.__temperatura))
        if self.__temperatura < 30:
            pygame.mixer.music.load("/home/pi/SIAaudios/temp1.mp3")
            pygame.mixer.music.play(1)
            time.sleep(3)
            pygame.mixer.music.load(self.__dic_h[str(self.__temperatura)])
            pygame.mixer.music.play(1)
            time.sleep(1.5)
            pygame.mixer.music.load("/home/pi/SIAaudios/temp2.mp3")
            pygame.mixer.music.play(1)
        return self.__temperatura
    def obtener_estado_puerta(self):
        return sensorHall.magnetismo()

    def reporte_puerta(self):
        if self.__seguridad == True:
            if self.obtener_estado_puerta() == False:
                print("la puerta principal esta abierta")
                botTelegram.enviarMensajeTelegram("la puerta principal esta abierta")
                botTelegram.enviarMensajeTelegram("hora: "+self.__hora)
```

```

botTelegram.enviarMensajeGrupal("hora: "+self.__hora+"\nla puerta principal esta
abierta")
registrosSIA.saveLog(self.__hora+" :Puerta principal abierta")

def establecer_seguridad(self,dato):
    if dato == "activar seguridad" or dato == "activa la seguridad":
        self.__seguridad = True
        print("seguridad activada")
        registrosSIA.saveLog(self.__hora+":seguridad activada")
    elif dato == "desactivar seguridad" or dato == "desactiva la seguridad":
        self.__seguridad = False
        print("seguridad desactivada")
        registrosSIA.saveLog(self.__hora+"seguridad desactivada")

def obtener_hora(self):
    return datetime.datetime.now().strftime("%H"+":"+"%M"+":"+"%S")

def encendidoAutomatico(self):
    #Enciende de manera automatica la luz en una determinada hora
    if self.__hora_ctrl >= 18 or self.__hora_ctrl < 6:
        self.__luces_on = True

def proceso_log_luz(self,dato):
    self.__resp = ""
    if dato == 22:
        self.__resp = "Dormitorio"
    elif dato == 23:
        self.__resp = "Baño"
    elif dato == 24:
        self.__resp = "Cocina"
    elif dato == 27:
        self.__resp = "Sala"
    return self.__resp

def encendido_peticion(self,orden,dato):
    #Enciende la luz si es que se le ordena encenderlo
    self.__mensaje =""
    if orden == "off" and self.__luces_on == False:
        pygame.mixer.music.load("/home/pi/SIAaudios/luzReoff.mp3")
        pygame.mixer.music.play(1)
        self.__mensaje = "las luces ya estan apagadas"
        registrosSIA.saveLog(self.__hora+":"+self.proceso_log_luz(dato)+"-orden de apagado
denegado")
    elif orden == "off" and self.__luces_on == True:
        self.establecer_estado_luces(False)
        pygame.mixer.music.load("/home/pi/SIAaudios/luzOff.mp3")

```

```

        pygame.mixer.music.play(1)
        GPIO.output(dato,GPIO.LOW)
        self.__mensaje = "luces apagadas"
        registrosSIA.saveLog(self.__hora+":"+self.proceso_log_luz(dato)+"-luces apagadas")
    elif orden == "on" and self.__luces_on == False:
        self.establecer_estado_luces(True)
        pygame.mixer.music.load("/home/pi/SIAaudios/luzOn.mp3")
        pygame.mixer.music.play(1)
        print(self.__luces_on)
        GPIO.output(dato,GPIO.HIGH)
        self.__mensaje = "luces encendidas"
        registrosSIA.saveLog(self.__hora+":"+self.proceso_log_luz(dato)+"-luces
encendidas")
    elif orden == "on" and self.__luces_on == True:
        pygame.mixer.music.load("/home/pi/SIAaudios/luzReon.mp3")
        pygame.mixer.music.play(1)
        self.__mensaje = "las luces ya estan encendidas"
        registrosSIA.saveLog(self.__hora+":"+self.proceso_log_luz(dato)+"-orden de
encendido denegado")
        print(self.__luces_on)
    return self.__mensaje

class dormitorio(habitacion):
    pass
    """la clase correspondiente a funciones especificas de un dormitorio"""
    def __init__(self,id_hab,nombres_hab):
        habitacion.__init__(self,id_hab)
        self.__nombres_hab = nombres_hab
        self.__horaAlarma = 0
        self.__minutoAlarma = 0
        self.__alarmaActiva = False
        self.__alarmaActivada = False
        self.__dic_hab = {"enciende la luz del dormitorio":"on","prende la luz del
dormitorio":"on","luces dormitorio":"on",
;"encender luz dormitorio":"on","apaga la luz del dormitorio":"off",
;"apagar la luz del dormitorio":"off","fueras luces dormitorio":"off" }

        self.__dic_h =
{"00":"/home/pi/SIAaudios/ceroh.mp3","01":"/home/pi/SIAaudios/unah.mp3","02":"/home/pi
/SIAaudios/dosh.mp3","03":"/home/pi/SIAaudios/tresh.mp3"
,"04":"/home/pi/SIAaudios/cuatroh.mp3","05":"/home/pi/SIAaudios/cincoh.mp3","06":"/home
/pi/SIAaudios/seish.mp3","07":"/home/pi/SIAaudios/sieteh.mp3"
,"08":"/home/pi/SIAaudios/ochoh.mp3","09":"/home/pi/SIAaudios/nueveh.mp3","10":"/home/
pi/SIAaudios/diezh.mp3","11":"/home/pi/SIAaudios/onceh.mp3"

```

```

,"12":"/home/pi/SIAuidos/doceh.mp3","13":"/home/pi/SIAuidos/13h.mp3","14":"/home/pi/
SIAuidos/14h.mp3","15":"/home/pi/SIAuidos/15h.mp3","16":"/home/pi/SIAuidos/16h.mp
3"
,"17":"/home/pi/SIAuidos/17h.mp3","18":"/home/pi/SIAuidos/18h.mp3","19":"/home/pi/SI
Auidos/20h.mp3","20":"/home/pi/SIAuidos/20h.mp3","21":"/home/pi/SIAuidos/21h.mp3"
,"22":"/home/pi/SIAuidos/22h.mp3","23":"/home/pi/SIAuidos/23h.mp3","24":"/home/pi/SI
Auidos/24h.mp3","25":"/home/pi/SIAuidos/25h.mp3","26":"/home/pi/SIAuidos/26h.mp3"
,"27":"/home/pi/SIAuidos/27h.mp3","28":"/home/pi/SIAuidos/28h.mp3","29":"/home/pi/SI
Auidos/29h.mp3","30":"/home/pi/SIAuidos/30h.mp3" }

def establecer_habitante(self,nombres):
    self.__nombres_hab = nombres
def obtener_nombres(self):
    return self.__nombres_hab

def ejecutarOrden(self,orden):
    self.__respuesta = self.__dic_hab.get(orden,"nada en habitacion")
    print(self.encendido_peticion(self.__respuesta,22))
    print(self.obtener_estado_luces())

def Alarma(self,mnsj):
    """Esta funcion define la hora de la alarma"""

    if(mnsj[0:17] == "Despiértame a las"):
        if(mnsj[20:21]==":"):
            self.__horaAlarma = int(mnsj[18:20])
            self.__minutoAlarma = int(mnsj[21:23])
        else:
            print(mnsj[18:19])
            self.__horaAlarma = int(mnsj[18:19])
            self.__minutoAlarma = int(mnsj[20:22])
        print("Alarma")
        print("hora: "+str(self.__horaAlarma))
        print("minuto: "+str(self.__minutoAlarma))
        registrosSIA.saveLog(self.obtener_hora()+":Alarma activada para
las:"+str(self.__horaAlarma)+":"+str(self.__minutoAlarma))
        self.__alarmaActivada = True
        pygame.mixer.music.load("/home/pi/SIAuidos/activacionAlarm.mp3")
        pygame.mixer.music.play(1)
        time.sleep(2)
        self.__ha = None #variables auxiliares
        self.__ma = None

```

```

if self.__minutoAlarma > 30:
    self.__ha = self.__horaAlarma+1
    self.__ma = 60-self.__minutoAlarma
    if self.__ha < 10:
        pygame.mixer.music.load(self.__dic_h["0"+str(self.__ha)])
        pygame.mixer.music.play(1)
        time.sleep(1)
    else:
        pygame.mixer.music.load(self.__dic_h[str(self.__ha)])
        pygame.mixer.music.play(1)
        time.sleep(1)
    pygame.mixer.music.load("/home/pi/SIAaudios/menosh.mp3")
    pygame.mixer.music.play(1)
    time.sleep(1)
else:
    self.__ha = self.__horaAlarma
    self.__ma = self.__minutoAlarma
    if self.__ha < 10:
        pygame.mixer.music.load(self.__dic_h["0"+str(self.__ha)])
        pygame.mixer.music.play(1)
        time.sleep(1)
    else:
        pygame.mixer.music.load(self.__dic_h[str(self.__ha)])
        pygame.mixer.music.play(1)
        time.sleep(1)
    time.sleep(1)
    pygame.mixer.music.load("/home/pi/SIAaudios/horash.mp3")
    pygame.mixer.music.play(1)
    time.sleep(1)
if self.__ma < 10:
    self.__ma = "0"+str(self.__ma)
else:
    self.__ma = str(self.__ma)
pygame.mixer.music.load(self.__dic_h[self.__ma])
pygame.mixer.music.play(1)
time.sleep(1)
pygame.mixer.music.load("/home/pi/SIAaudios/minutosh.mp3")
pygame.mixer.music.play(1)

def verificacionAlarma(self):
    self.__h = int(datetime.datetime.now().strftime("%H")) and
int(datetime.datetime.now().strftime("%H"))
    self.__m = int(datetime.datetime.now().strftime("%H")) and
int(datetime.datetime.now().strftime("%M"))
    if self.__alarmaActivada == True:
        if self.__h == self.__horaAlarma:
            if self.__m == self.__minutoAlarma:

```

```

        if self.__alarmaActiva == False:
            self.__alarmaActiva = True
            pygame.mixer.music.load("/home/pi/SIAaudios/desperta.mp3")
            pygame.mixer.music.play(1)
            time.sleep(5)
            pygame.mixer.music.load("/home/pi/SIAmusic/musicaDespertar.mp3")
            pygame.mixer.music.play(1)
        elif self.__m > self.__minutoAlarma:
            self.__alarmaActiva = False
            self.__alarmaActivada = False
            self.__minutoAlarma = 0
            self.__horaAlarma = 0

```

```

class baño(habitacion):
    pass
    def __init__(self,id_ba):
        habitacion.__init__(self,id_ba)
        self.__ducha_on = False
        self.__dic_hab = { "enciende la luz del baño":"on","prende la luz del baño":"on","luces
baño":"on"
                           ,"encender luz baño":"on","apaga la luz del baño":"off",
                           "apagar la luz del baño":"off","fuera luces baño":"off" }

        self.__dic_ducha = { "ducha":"on","enciende la ducha":"on","abre la ducha":"on"
                           ,"apaga la ducha":"off","cierra la ducha":"off" }

    def control_ducha(self,orden):
        self.__respuesta = self.__dic_ducha.get(orden,None)
        self.__mensaje =""
        if self.__respuesta == "off" and self.__ducha_on == False:
            pygame.mixer.music.load("/home/pi/SIAaudios/llaveReoff.mp3")
            pygame.mixer.music.play(1)
            self.__mensaje = "la ducha ya esta apagada"
            registrosSIA.saveLog(self.obtener_hora()+":ducha-orden de apagado denegado")
        elif self.__respuesta == "off" and self.__ducha_on == True:
            self.__ducha_on = False
            pygame.mixer.music.load("/home/pi/SIAaudios/llaveCerrar.mp3")
            pygame.mixer.music.play(1)
            GPIO.output(20,GPIO.LOW)
            self.__mensaje = "ducha apagada"
            registrosSIA.saveLog(self.obtener_hora()+":ducha apagada")
        elif self.__respuesta == "on" and self.__ducha_on == False:
            self.__ducha_on = True
            pygame.mixer.music.load("/home/pi/SIAaudios/llaveAgua.mp3")
            pygame.mixer.music.play(1)

```

```

        GPIO.output(20,GPIO.HIGH)
        self.__mensaje = "ducha encendida"
        registrosSIA.saveLog(self.obtener_hora() + ":ducha encendida")
    elif self.__respuesta == "on" and self.__ducha_on == True:
        pygame.mixer.music.load("/home/pi/SIAaudios/lлавeReon.mp3")
        pygame.mixer.music.play(1)
        self.__mensaje = "la ducha ya esta encendida"
        registrosSIA.saveLog(self.obtener_hora() + ":ducha-orden de encendido denegado")
    print(self.__ducha_on)
    return self.__mensaje

def ejecutarOrden(self,orden):
    self.__respuesta = self.__dic_hab.get(orden,None)
    print(self.encendido_peticion(self.__respuesta,23))
    print(self.obtener_estado_luces())

class cocina(habitacion):
    pass
    """Funciones basicas especificas de una cocina"""
    def __init__(self,id_co):
        habitacion.__init__(self,id_co)
        self.__valv_agua = False
        self.__detect_glp = False
        self.__detect_co = False
        self.__dic_hab = {"enciende la luz de la cocina":"on","prende la luz de la cocina":"on","luces cocina":"on",
                         "encender luz cocina":"on","apaga la luz de la cocina":"off",
                         "apagar la luz de la cocina":"off","fuerza luces cocina":"off"}

        self.__dic_valv = {"abre la llave de la cocina":"on","enciende la llave de la cocina":"on",
                           "agua cocina":"on",
                           "apaga la llave de la cocina":"off",
                           "cierra la llave de la cocina":"off"}

    def verificacionFugaGLP(self, dato):
        if dato[0] == 'g':
            self.__pos = 3
            self.__d = 0
            if dato[self.__pos] == ':':
                self.__d = int(dato[self.__pos+1:-2])
            else:
                self.__pos = self.__pos+1
                if dato[self.__pos] == ':':
                    self.__d = int(dato[self.__pos+1:-2])

        print(self.__d)
        if self.__d > 150:

```

```

        self.__detect_glp = True
        registrosSIA.saveLog(self.obtener_hora()+":Deteccion de fuga de gas")
        botTelegram.enviarMensajeTelegram("Deteccion de fuga de gas")
        botTelegram.enviarMensajeTelegram("hora: "+self.obtener_hora())
        botTelegram.enviarMensajeGrupal("hora: "+self.obtener_hora()+"\nDeteccion de
fuga de gas")
        pygame.mixer.music.load("/home/pi/SIAaudios/fugaGas.mp3")
        pygame.mixer.music.play(1)
        time.sleep(6)
        pygame.mixer.music.load("/home/pi/SIAaudios/ALARMA.mp3")
        pygame.mixer.music.play(1)
        time.sleep(1)
    elif self.__d < 150 and self.__detect_glp == True:
        self.__detect_glp = False
        registrosSIA.saveLog(self.obtener_hora()+":Ambiente estable, fuga de gas
controlada")
        botTelegram.enviarMensajeTelegram("Ambiente estable, fuga de gas controlada")
        botTelegram.enviarMensajeTelegram("hora: "+self.obtener_hora())
        botTelegram.enviarMensajeGrupal("hora: "+self.obtener_hora()+"\nAmbiente
estable, fuga de gas controlada")
        pygame.mixer.music.load("/home/pi/SIAaudios/estable.mp3")
        pygame.mixer.music.play(1)

def control_valv_agua(self,orden):
    self.__respuesta = self.__dic_valv.get(orden,None)
    self.__mensaje =""
    if self.__respuesta == "off" and self.__valv_agua == False:
        pygame.mixer.music.load("/home/pi/SIAaudios/llaveReoff.mp3")
        pygame.mixer.music.play(1)
        self.__mensaje = "la llave ya esta cerrada"
        registrosSIA.saveLog(self.obtener_hora()+":llave-orden de apagado denegado")
    elif self.__respuesta == "off" and self.__valv_agua == True:
        self.__valv_agua = False
        pygame.mixer.music.load("/home/pi/SIAaudios/llaveCerrar.mp3")
        pygame.mixer.music.play(1)
        GPIO.output(20,GPIO.LOW)
        self.__mensaje = "llave cerrada"
        registrosSIA.saveLog(self.obtener_hora()+":llave cerrada")
    elif self.__respuesta == "on" and self.__valv_agua == False:
        self.__valv_agua = True
        pygame.mixer.music.load("/home/pi/SIAaudios/llaveAgua.mp3")
        pygame.mixer.music.play(1)
        GPIO.output(20,GPIO.HIGH)
        self.__mensaje = "llave abierta"
        registrosSIA.saveLog(self.obtener_hora()+":llave abierta")
    elif self.__respuesta == "on" and self.__valv_agua == True:
        pygame.mixer.music.load("/home/pi/SIAaudios/llaveReon.mp3")

```

```

        pygame.mixer.music.play(1)
        self.__mensaje = "la llave ya esta abierta"
        registrosSIA.saveLog(self.obtener_hora() + ":llave-orden de encendido denegado")
        print(self.__valv_agua)
        return self.__mensaje

    def ejecutarOrden(self, orden):
        self.__respuesta = self.__dic_hab.get(orden, "nada en cocina")
        print(self.encendido_peticion(self.__respuesta, 27))
        print(self.obtener_estado_luces())

    class sala(habitacion):
        pass
        """funciones basicas especificas que pueden haber en una sala"""
        def __init__(self, id_sala):
            habitacion.__init__(self, id_sala)
            self.__detect_co = False
            self.__dic_hab = {"enciende la luz de la sala": "on", "prende la luz de la sala": "on", "luces sala": "on",
                             "encender luz sala": "on", "apaga la luz de la sala": "off",
                             "apagar la luz de la sala": "off", "fuera luces sala": "off" }

            self.__dic_control = {"cambia de canal": "chUp", "otro canal": "chUp", "cambia de canal atras": "chD", "baja canal": "chD",
                                  "sube el volumen": "v+", "aumenta el volumen": "v+", "baja el volumen": "v-",
                                  "disminuye el volumen": "v-" }

        def verificacionCO(self, dato):
            if dato[0] == 'c':
                self.__pos = 3
                self.__d = 0
                if dato[self.__pos] == ':':
                    self.__d = int(dato[self.__pos+1:-2])
                else:
                    self.__pos = self.__pos+1
                    if dato[self.__pos] == ':':
                        self.__d = int(dato[self.__pos+1:-2])

                print(self.__d)
                if self.__d > 50:
                    self.__detect_co = True
                    registrosSIA.saveLog(self.obtener_hora() + ":Deteccion de monoxido de carbono, posible incendio")
                    botTelegram.enviarMensajeTelegram("Deteccion de monoxido de carbono, posible incendio")
                    botTelegram.enviarMensajeTelegram("hora: " + self.obtener_hora())

```

```

botTelegram.enviarMensajeGrupal("hora: "+self.obtener_hora()+"\nDeteccion de
monoxido de carbono, posible incendio")
    pygame.mixer.music.load("/home/pi/SIAaudios/alarmaIncendio.mp3")
    pygame.mixer.music.play(1)
    time.sleep(6)
    pygame.mixer.music.load("/home/pi/SIAaudios/ALARMA.mp3")
    pygame.mixer.music.play(1)
    time.sleep(1)
elif self.__d < 10 and self.__detect_co == True:
    self.__detect_co = False
    registrosSIA.saveLog(self.obtener_hora()+":ambiente estable, posible incendio
controlado")
    botTelegram.enviarMensajeTelegram("Ambiente estable, posible incendio
controlado")
    botTelegram.enviarMensajeTelegram("hora: "+self.obtener_hora())
    botTelegram.enviarMensajeGrupal("hora: "+self.obtener_hora()+"\nambiente
estable, posible incendio controlado")
    pygame.mixer.music.load("/home/pi/SIAaudios/estable.mp3")
    pygame.mixer.music.play(1)

def ejecutarOrden(self,orden):
    self.__respuesta = self.__dic_hab.get(orden,"nada en sala")
    print(self.encendido_peticion(self.__respuesta,24))
    print(self.obtener_estado_luces())

def controlRemoto(self,orden):
    self.respuesta = self.__respuesta = self.__dic_control.get(orden,None)
    self.__control = False
    if self.respuesta == "chUp":
        subprocess.call("irsend SEND_ONCE SONY-TV KEY_CHANNELUP",shell=True)
        self.__control = True
    elif self.respuesta == "chD":
        subprocess.call("irsend SEND_ONCE SONY-TV
KEY CHANNELDOWN",shell=True)
        self.__control = True
    elif self.respuesta == "v+":
        subprocess.call("irsend SEND_ONCE SONY-TV KEY_VOLUMEUP",shell=True)
    elif self.respuesta == "v-":
        subprocess.call("irsend SEND_ONCE SONY-TV
KEY_VOLUMEDOWN",shell=True)

    if self.__control == True:
        pygame.mixer.music.load("/home/pi/SIAaudios/cambioCanal.mp3")
        registrosSIA.saveLog(self.obtener_hora()+":tv-cambio de canal")
        pygame.mixer.music.play(1)

class musica():

```

```

def __init__(self):
    self.__rock={"nirvana":"/home/pi/SIAmusic/nirvana.mp3","después de
ti":"/home/pi/SIAmusic/Coda 3-Despues de ti.mp3","Hada y mago":"/home/pi/SIAmusic/rata
blanca - la leyenda del hada y el mago.mp3",
    "Mujer amante":"/home/pi/SIAmusic/Rata Blanca - Mujer
Amante.mp3","religión":"/home/pi/SIAmusic/REM Losing My Religion.mp3"}
    self.__pop={"Zain":"/home/pi/SIAmusic/ZAYN-
PILLOWTALK.mp3","Onion":"/home/pi/SIAmusic/on&on.mp3","body":"/home/pi/SIAmusi
c/Loud Luxury-Body.mp3",
    "Jonas":"/home/pi/SIAmusic/fastCar.mp3","stay":"/home/pi/SIAmusic/Kygo-
Stay.mp3"}
    self.__clasico={"Subidon":"/home/pi/SIAmusic/Fey-
Subidon.mp3","Montaner":"/home/pi/SIAmusic/DEJAME
LLORAR.mp3","eclipse":"/home/pi/SIAmusic/Eclipse Total Del Amor.mp3",
    "Hijo de la luna":"/home/pi/SIAmusic/Mecano-Hijo de la
luna.mp3","Sera":"/home/pi/SIAmusic/sera.mp3"}
    self.__reggaeton={"Adan y Eva":"/home/pi/SIAmusic/Adan y Eva.mp3","Adán y
Eva":"/home/pi/SIAmusic/Adan y
Eva.mp3","desconocidos":"/home/pi/SIAmusic/Desconocidos.mp3","otra
vez":"/home/pi/SIAmusic/Otra Vez.mp3",
    "por perro":"/home/pi/SIAmusic/POR
PERRO.mp3","Sebastián":"/home/pi/SIAmusic/POR
PERRO.mp3","Sebastian":"/home/pi/SIAmusic/POR PERRO.mp3",
    "se preparó":"/home/pi/SIAmusic/Se Preparó.mp3"}
def playMusica(self,text):
    print(text)
    self.__nombreCancion="" #el metodo get nos permitira evitar una excepcion cuando no
encuentre una clave
    if self.__rock.get(text,"nada")!= "nada":
        self.__nombreCancion=self.__rock.get(text,"nada")
    elif self.__pop.get(text,"nada")!= "nada":
        self.__nombreCancion=self.__pop.get(text,"nada")
    elif self.__clasico.get(text,"nada")!= "nada":
        self.__nombreCancion=self.__clasico.get(text,"nada")
    elif self.__reggaeton.get(text,"nada")!= "nada":
        self.__nombreCancion=self.__reggaeton.get(text,"nada")

    if self.__nombreCancion != "":
        registrosSIA.saveLog(":Reproduccion de cancion:"+self.__nombreCancion)
        pygame.mixer.music.load(self.__nombreCancion)
        pygame.mixer.music.play(1)

```

class residente:

"""En un apartamento viven familias o cualquier numero de personas
esas personas tienen datos importantes para la gestion de una edificacion"""

```

def __init__(self,nombre,ci,nacimiento,parent,celular):
    self.__nombre = nombre
    self.__ci = ci
    self.__nacimiento = nacimiento
    self.__parent = parent #es el parentezco que tienen con el propietario
    self.__cel = celular

def establecer_cel(self,cel):
    self.__cel = cel
def obtener_cel(self):
    return self.__cel

def establecer_ci(self,ci):
    self.__ci = ci
def obtener_ci(self):
    return self.__ci

def establecer_nacimiento(self,nac):
    self.__nacimiento = nac
def obtener_nacimiento(self):
    return self.__nacimiento

def establecer_parent(self,p):
    self.__parent = p
def obtener_parent(self):
    return self.__parent

def establecer_nombre(self,n):
    self.__nombre = n
def obtener_nombre(self):
    return self.__nombre

def obtener_datos(self):
    print(self.__nombre,self.__ci,self.__nacimiento,self.__parent,self.__cel,sep = "\n")

```

```

class apartamento:
    """Aqui maneja menos funciones electronicas y mas funciones administrativas"""
    def __init__(self,nombre,ci,edad,hijos,nro_apo):
        #se define nombre del propietario(string),edad(int),carnet(int),hijos(True),
        #numero del apartamento(int)
        self.__nombre = nombre
        self.__ci = ci
        self.__edad = edad
        self.__hijos = hijos #Esta variable verifica si tiene hijos o no
        self.__nro_apo = nro_apo
        self.__nro_hab = 0

```

```

self.__nro_resi = 0
self.musica=musica()
self.dor = []
self.baño = []
self.coc = []
self.sala = []
self.resi = []

def print_datos(self):
    print(self.__nombre,self.__ci,self.__edad,self.__hijos,self.__nro_apo,self.__nro_hab,sep='\n')

def establecer_nombre(self,nom):
    self.__nombre = nom
def establecer_ci(self,ci):
    self.__ci=ci
def establecer_edad(self,edad):
    self.__edad=edad
def confirmar_hijos(self,conf):
    self.__hijos = conf
def establecer_nro_apo(self,nro):
    self.__nro_apo = conf

def obtener_nombre(self):
    return self.__nombre
def obtener_ci(self):
    return self.__ci
def obtener_edad(self):
    return self.__edad
def obtener_hijos(self):
    return self.__hijos
def obtener_nro_apo(self):
    return self.__nro_apo
def obtener_nro_hab(self): #devuelve el numero de habitaciones
    return self.__nro_hab
def obtener_nro_res(self): #devuelve el numero de residentes
    return self.__nro_resi

def obtener_hora(self, orden):
    self.__hora= datetime.datetime.now().strftime("%H"+":"+"%M"+":"+"%S")
    self.__h = int(datetime.datetime.now().strftime("%H"))
    self.__min = int(datetime.datetime.now().strftime("%M"))
    self.__dic_hora = { "hora":self.__hora,"Dame la hora":self.__hora,"qué hora es":self.__hora,"qué hora tienes":self.__hora}

```

```

    self.__dic_h =
    {"00":"/home/pi/SIAaudios/ceroh.mp3","01":"/home/pi/SIAaudios/unah.mp3","02":"/home/pi
    /SIAaudios/dosh.mp3","03":"/home/pi/SIAaudios/tresh.mp3"
    ,"04":"/home/pi/SIAaudios/cuatroh.mp3","05":"/home/pi/SIAaudios/cincoh.mp3","06":"/home/
    /pi/SIAaudios/seish.mp3","07":"/home/pi/SIAaudios/sieteh.mp3"
    , "08": "/home/pi/SIAaudios/ochoh.mp3", "09": "/home/pi/SIAaudios/nueveh.mp3", "10": "/home/
    /pi/SIAaudios/diezh.mp3", "11": "/home/pi/SIAaudios/onceh.mp3"
    , "12": "/home/pi/SIAaudios/doceh.mp3", "13": "/home/pi/SIAaudios/13h.mp3", "14": "/home/pi/
    SIAaudios/14h.mp3", "15": "/home/pi/SIAaudios/15h.mp3", "16": "/home/pi/SIAaudios/16h.mp
    3"
    , "17": "/home/pi/SIAaudios/17h.mp3", "18": "/home/pi/SIAaudios/18h.mp3", "19": "/home/pi/SI
    Aaudios/20h.mp3", "20": "/home/pi/SIAaudios/20h.mp3", "21": "/home/pi/SIAaudios/21h.mp3"
    , "22": "/home/pi/SIAaudios/22h.mp3", "23": "/home/pi/SIAaudios/23h.mp3", "24": "/home/pi/SI
    Aaudios/24h.mp3", "25": "/home/pi/SIAaudios/25h.mp3", "26": "/home/pi/SIAaudios/26h.mp3"
    , "27": "/home/pi/SIAaudios/27h.mp3", "28": "/home/pi/SIAaudios/28h.mp3", "29": "/home/pi/SI
    Aaudios/29h.mp3", "30": "/home/pi/SIAaudios/30h.mp3" }
    if self.__dic_hora.get(orden,"nada")!="nada":
        print("Son las "+self.__dic_hora.get(orden,"nada"))
        registrosSIA.saveLog(self.baño[0].obtener_hora()+":consulta hora")
        pygame.mixer.music.load("/home/pi/SIAaudios/sonlash.mp3")
        pygame.mixer.music.play(1)
        time.sleep(1)
        if self.__min > 30:
            print(str(self.__h+1))
            if self.__h+1 < 10:
                pygame.mixer.music.load(self.__dic_h["0"+str(self.__h+1)])
                pygame.mixer.music.play(1)
                time.sleep(1)
            else:
                pygame.mixer.music.load(self.__dic_h[str(self.__h+1)])
                pygame.mixer.music.play(1)
                time.sleep(1)
            pygame.mixer.music.load("/home/pi/SIAaudios/menosh.mp3")
            pygame.mixer.music.play(1)
            time.sleep(1)
            self.__min = 60 - self.__min
        else:
            pygame.mixer.music.load(self.__dic_h[datetime.datetime.now().strftime("%H")])
            pygame.mixer.music.play(1)
            time.sleep(1)
            pygame.mixer.music.load("/home/pi/SIAaudios/horash.mp3")

```

```

        pygame.mixer.music.play(1)
        time.sleep(1)
        if self.__min < 10:
            self.__min = "0"+str(self.__min)
        else:
            self.__min = str(self.__min)
        print(str(self.__min))
        pygame.mixer.music.load(self.__dic_h[self.__min])
        pygame.mixer.music.play(1)
        time.sleep(1)
        pygame.mixer.music.load("/home/pi/SIAuidos/minutosh.mp3")
        pygame.mixer.music.play(1)

```

#En esta funcion llenamos la relacion entre sensores y ambiente de la base de datos.

```

def sensorxambiente(self):
    self.__tup1 = []
    self.__tup2 = []
    self.ps = 0
    cursor.execute("select * from ambiente")
    self.datos = cursor.fetchall()
    for data in self.datos:
        self.__tup1.append(data[0])
        self.__tup2.append(data[1])
    while (self.ps < len(self.__tup2)):
        self.__i = str(self.__tup1[self.ps])
        if self.__tup2[self.ps] == 1:
            cursor.execute("insert into uso_sensor values("+self.__i+",2)")
            db.commit()
        elif self.__tup2[self.ps] == 2:
            cursor.execute("insert into uso_sensor values("+self.__i+",2)")
            db.commit()
            cursor.execute("insert into uso_sensor values("+self.__i+",3)")
            db.commit()
            cursor.execute("insert into uso_sensor values("+self.__i+",4)")
            db.commit()
            cursor.execute("insert into uso_sensor values("+self.__i+",8)")
            db.commit()
        elif self.__tup2[self.ps] == 3:
            cursor.execute("insert into uso_sensor values("+self.__i+",1)")
            db.commit()
            cursor.execute("insert into uso_sensor values("+self.__i+",2)")
            db.commit()
            cursor.execute("insert into uso_sensor values("+self.__i+",6)")
            db.commit()
        elif self.__tup2[self.ps] == 4:
            cursor.execute("insert into uso_sensor values("+self.__i+",5)")
            db.commit()

```

```

        cursor.execute("insert into uso_sensor values("+self.__i+",6)")
        db.commit()
        cursor.execute("insert into uso_sensor values("+self.__i+",7)")
        db.commit()
        self.ps=self.ps+1
    
```

#en estos metodo creamos la cantidad de dormitorios, baños, cocinas y sala que puede tener un apartamento

```

def crear_dor(self,nro):
    self.copy = 0
    while self.copy < nro:
        self.dor.append(dormitorio(self.copy,""))
        cursor.execute("insert into ambiente(tipo,apartamento)
values(1,"+str(self.__nro_ap)+")")
        db.commit()
        self.copy = self.copy+1
    self.__nro_hab = self.__nro_hab + nro
    
```

```

def crear_baño(self,nro):
    self.copy = 0
    while self.copy < nro:
        self.baño.append(baño(self.copy))
        cursor.execute("insert into ambiente(tipo,apartamento)
values(4,"+str(self.__nro_ap)+")")
        db.commit()
        self.copy = self.copy+1
    self.__nro_hab = self.__nro_hab+nro
    
```

```

def crear_coc(self,nro):
    self.copy = 0
    while self.copy < nro:
        self.coc.append(cocina(self.copy))
        cursor.execute("insert into ambiente(tipo,apartamento)
values(3,"+str(self.__nro_ap)+")")
        db.commit()
        self.copy = self.copy+1
    self.__nro_hab = self.__nro_hab+nro
    
```

```

def crear_sala(self,nro):
    self.copy = 0
    while self.copy < nro:
        self.sala.append(sala(self.copy))
        cursor.execute("insert into ambiente(tipo,apartamento)
values(2,"+str(self.__nro_ap)+")")
        db.commit()
        self.copy = self.copy+1
    
```

```

self.__nro_hab = self.__nro_hab+nro

def crear_resi(self,nro):
    self.copy = 0
    while self.copy < nro:
        self.resi.append(residente("",0,"","",0))
        self.copy = self.copy+1
    self.__nro_resi = nro

#este metodo genera un informe pdf de todos los residentes y lo envia a telegram
def generarInforme(self,orden):
    self.__dic_inf = {"genera un informe de los residentes":"ok","genera un informe de los habitantes":"ok",
                      "manda un informe de los residentes":"ok","manda un informe de los habitantes":"ok",
                      "informe habitantes":"ok",
                      "informe residentes":"ok"}
    self.__respuesta = self.__dic_inf.get(orden,None)
    if self.__respuesta == "ok":
        pygame.mixer.music.load("/home/pi/SIAaudios/regResidentes.mp3")
        pygame.mixer.music.play(1)
        self.__doc = open("informeResidentes.txt",'w')
        self.__p = pdf.PDF()
        self.__doc.writelines("Lista de residentes")
        self.__cont = 0
        while self.__cont < self.__nro_resi:
            self.__doc.writelines("\n"+ "Residente numero "+str(self.__cont+1))
            self.__doc.writelines("\n"+self.resi[self.__cont].obtener_nombre())
            self.__doc.writelines("\n"+str(self.resi[self.__cont].obtener_ci()))
            self.__doc.writelines("\n"+self.resi[self.__cont].obtener_nacimiento())
            self.__doc.writelines("\n"+self.resi[self.__cont].obtener_parent())
            self.__doc.writelines("\n"+str(self.resi[self.__cont].obtener_cel()))
            self.__doc.writelines("\n")
            self.__cont = self.__cont + 1
        self.__doc.close()
        self.__p.crearPDF("informeResidentes.txt","INFORME DE RESIDENTES")
        registrosSIA.saveLog(self.baño[0].obtener_hora()+":Solicitud de informe de residentes")
        botTelegram.enviarDocumento("Envio de registro de habitantes",'siaRegistro.pdf')

def obtenerAyuda(self,orden):
    self.__res = self.baño[0].obtener_hora()+":Alguien pide auxilio de manera urgente en la casa, puede estar en problemas"
    self.__dic_help = {"estoy herido":self.__res,"auxilio":self.__res,
                      "pide ayuda":self.__res}
    if orden in self.__dic_help:
        pygame.mixer.music.load("/home/pi/SIAaudios/auxilio.mp3")
        pygame.mixer.music.play(1)

```

```

botTelegram.enviarMensajeGrupal(self.__dic_help[orden])

#los siguientes metodos son para añadir o quitar un elemento desde un dormitorio hasta

def añadir_dor(self,nro):
    self.dor.append(dormitorio(self.copy,""))

def añadir_baño(self,nro):
    self.baño.append(baño(0))

def añadir_coc(self,nro):
    self.coc.append(cocina(0))

def añadir_sala(self,nro):
    self.sala.append(dormitorio(0))

class piso:
    """el piso administrara la cantidad de habitaciones que existe y el control de algunas
valvulas"""
    def __init__(self):
        self.__nro_piso=0
        self.__termico = True
        self.__valvula_agua = True
        self.__estadoTanque = None
        self.__valvula_gas = True
        self.__vol_tanque = 3.458 #litros
        self.__nro_apar = 0
        self.apar = []

    def establecer_nro_piso(self,nro):
        self.__nro_piso=nro

    def control_termico(self,orden):
        self.__termico = orden
    def obtener_estado_termico(self):
        return self.__termico

    def control_valv_agua(self,orden):
        self.__valvula_agua = orden
    def obtener_estado_valv_agua(self):
        return self.__valvula_agua

    def control_valv_gas(self,orden):
        self.__valvula_gas = orden
    def obtener_estado_valv_gas(self):
        return self.__valvula_gas

```

```

def crear_apartamentos(self,nro):
    self.copy = 0
    self.base_nro = self.__nro_piso*10;
    while self.copy < nro:
        self.apar.append(apartamento("",0,0,False,self.base_nro+self.copy+1))
        self.__id=self.base_nro+self.copy+1
        cursor.execute("insert into apartamento(id_apartamento,nro_piso)
values("+str(self.__id)+","+str(self.__nro_piso)+")")
        db.commit()
        self.__id2 = (self.__id*10)+1
        cursor.execute("insert into servicios_por_apartamento
values("+str(self.__id2)+",1,"+str(self.__id)+")")
        db.commit()
        cursor.execute("insert into servicios_por_apartamento
values("+str(self.__id2+1)+",2,"+str(self.__id)+")")
        db.commit()
        cursor.execute("insert into servicios_por_apartamento
values("+str(self.__id2+2)+",4,"+str(self.__id)+")")
        db.commit()
        self.copy = self.copy+1
    self.__nro_apa = nro

def reporte_tanque(self):
    if(tanque1.estado()==True and tanque2.estado()==False):
        print("el tanque esta vacio")
        registrosSIA.saveLog(self.apar[0].baño[0].obtener_hora()+":tanque vacio")
        self.__estadoTanque = False
        pygame.mixer.music.load("/home/pi/SIAaudios/llevarTanque.mp3")
        pygame.mixer.music.play(1)
        self.llenarTanque(True)
        GPIO.output(21,GPIO.HIGH)
    elif(tanque1.estado()==False and tanque2.estado()==True):
        if self.__estadoTanque == False:
            self.__estadoTanque = True
            print("el tanque esta lleno")
            registrosSIA.saveLog(self.apar[0].baño[0].obtener_hora()+":tanque lleno")
            self.llenarTanque(False)
            pygame.mixer.music.load("/home/pi/SIAaudios/tanqueLleno.mp3")
            pygame.mixer.music.play(1)
            GPIO.output(21,GPIO.LOW)

def llenarTanque(self, dato):
    if dato==True:
        print("llenando tanque")
        registrosSIA.saveLog(self.apar[0].baño[0].obtener_hora()+":llenando tanque")
    elif dato == False:
        print("parando llenado")

```

```

        self.fechaActual = datetime.datetime.today().strftime('%Y-%m-%d')
        cursor.execute("insert into consumo values('SA-
"+str(datetime.datetime.today())+"','"+self.fechaActual+"','"+str(self.__vol_tanque)+",112)")
        db.commit()
        registrosSIA.saveLog(self.apar[0].baño[0].obtener_hora()+"tanque lleno")

def reporteConsumos(self,orden):
    self.__dic_cons = {"genera un informe del consumo de agua":"ok","genera un informe
del agua":"ok","informe agua":"ok",
                      "manda un informe del agua":"ok","manda un informe del servicio de
agua":"ok","informe de consumo de agua":"ok",
                      "informe del consumo de servicio de agua":"ok"}
    self.__respuesta = self.__dic_cons.get(orden,None)
    if self.__respuesta == "ok":
        pygame.mixer.music.load("/home/pi/SIAaudios/regConAgua.mp3")
        pygame.mixer.music.play(1)
        self.__tup1 = []
        self.__tup2 = []
        self.ps = 0
        cursor.execute("select * from consumo")
        self.datos = cursor.fetchall()
        for data in self.datos:
            self.__tup1.append(data[1])
            self.__tup2.append(data[2])
        self.__mes = str(self.__tup1[0])[0:7]
        self.__comp = str(self.__tup1[0])[8:10]
        self.__total = 0
        while (self.ps < len(self.__tup1)):
            self.__aux = str(self.__tup1[self.ps])[8:10]
            if self.__aux == self.__comp:
                self.__total = self.__total + float(self.__tup2[self.ps])
            else:
                consumoAgua.addConsDia(int(self.__comp),self.__total)
                self.__comp = str(self.__tup1[self.ps])
                self.__total = 0
            self.ps = self.ps+1
        consumoAgua.addConsDia(int(self.__comp),self.__total)
        consumoAgua.addConsDia(int(self.__comp)+1,0)
        consumoAgua.crearPDF(self.__mes)
        registrosSIA.saveLog(self.apar[0].baño[0].obtener_hora()+"Solicitud de informe de
consumo de agua")
        botTelegram.enviarDocumento("Envio de informe de
consumo",'consumoAgua'+self.__mes+'.pdf')
        botTelegram.enviarDocGrupal("Envio de informe de
consumo",'consumoAgua'+self.__mes+'.pdf')

```

#Aqui es donde comienza la mas importante de las clases, la matriz de todo, S.I.A. que significa sistema de inteligencia artificial, no solo es un asistente virtual
#Si no que tambien es capaz de realizar tareas de domotica e inmotica,

class sia:

```
    """Aquí se debe controlar todo el sistema"""
    def __init__(self):
        self.__nombre = "Alejandra"
        self.__nom_prop = ""
        self.__nom_edif = ""
        self.__pasword = ""
        self.__direccion = ""
        self.__configurado = False
        self.__com_serial =
serial.Serial('/dev/ttyUSB0',baudrate=9600,bytesize=8,parity='N',stopbits=1)
        self.piso = []
        self.__msnWindow = "" #aqui se guarda el mensaje para la interfaz grafica
        pygame.init()

    def reconocer(self):
        self.__mens=""
        self.__r = sr.Recognizer()
        with sr.Microphone() as source:
            print("te escucho")
            try:
                self.__audio = self.__r.listen(source,2) #te escuchara por durante 2 segundos
            except sr.WaitTimeoutError as e:
                self.__mens = "tiempo"

        if self.__mens != "tiempo": #existe una excepcion si es que se quitara este if, porfavor no
            guitar
            try:
                self.__mens = self.__r.recognize_google(self.__audio, language = "es-ES")

            except sr.UnknownValueError:
                self.mens = "lo siento no pude entenderte"

        return self.__mens
```

#Sia tiene un nombre por defecto, su nombre es Alejandra, como en cualquier conversacion, primero debes mencionar el nombre de la persona o en este caso del agente, esto para evitar mencionar alguna palabra clave durante una conversacion con otras personas y
#SIA ejecute alguna funcion

```
def escucharMiNombre(self):
```

```

if self.__configurado == True: #esta variable se mantiene en true cuando esta configurado
    while True:
        self.__escuchado = self.reconocer()
        if self.__escuchado == self.__nombre:
            print(self.__escuchado)
            pygame.mixer.music.load("/home/pi/SIAuidos/resp1.wav")
            pygame.mixer.music.play(1)
            self.escucharOrden()
        else:
            print(self.lectura())
            self.piso[0].apar[0].dor[0].verificacionAlarma()
            self.piso[0].apar[0].coc[0].verificacionFugaGLP(self.lectura()[0])
            self.piso[0].apar[0].sala[0].verificacionCO(self.lectura()[1])
            self.piso[0].apar[0].sala[0].reporte_puerta()
            self.piso[0].reporte_tanque()

    else:
        self.configuracion_inicial()

def escucharOrden(self):
    while True:
        self.__escuchado = self.reconocer()
        print(self.__escuchado)
        self.piso[0].apar[0].musica.playMusica(self.__escuchado)
        self.piso[0].apar[0].dor[0].ejecutarOrden(self.__escuchado)
        self.piso[0].apar[0].baño[0].ejecutarOrden(self.__escuchado)
        self.piso[0].apar[0].coc[0].ejecutarOrden(self.__escuchado)
        self.piso[0].apar[0].sala[0].ejecutarOrden(self.__escuchado)
        self.piso[0].apar[0].baño[0].control_ducha(self.__escuchado)
        self.piso[0].apar[0].coc[0].control_valv_agua(self.__escuchado)
        self.piso[0].apar[0].sala[0].establecer_seguridad(self.__escuchado)
        self.piso[0].apar[0].sala[0].controlRemoto(self.__escuchado)
        self.piso[0].apar[0].obtener_hora(self.__escuchado)
        self.piso[0].apar[0].generarInforme(self.__escuchado)
        self.piso[0].apar[0].obtenerAyuda(self.__escuchado)
        self.piso[0].apar[0].dor[0].Alarma(self.__escuchado)
        self.piso[0].apar[0].dor[0].obtener_temperatura(self.__escuchado)
        self.piso[0].reporteConsumos(self.__escuchado)
        self.generarInformeRegistros(self.__escuchado)
        #print(self.piso[0].apar[0].dor[0].obtener_estado_luces())
        self.escucharMiNombre()

def lectura(self):
    self.__respuestaS1 = self.__com_serial.readline().decode()
    time.sleep(0.2)
    self.__respuestaS2 = self.__com_serial.readline().decode()
    self.__com_serial.flushInput()

```

```

        return (self.__respuestaS1,self.__respuestaS2)

def crear_pisos(self,nro):
    self.copy = 0
    self.desc = ""
    if nro <= 1:
        self.desc = 'planta baja'
    else:
        self.desc = 'piso '+str(nro)
    while self.copy < nro:
        self.piso.append(piso())
        self.piso[self.copy].establecer_nro_piso(self.copy+1)
        self.copy = self.copy+1
        cursor.execute("insert into piso(descripcion) values(\""+self.desc+f"\")")
        db.commit()
    self.__nro_piso = nro

def configuracion_inicial(self):
    pygame.mixer.music.load("/home/pi/SIAaudios/mensaje1.wav")
    pygame.mixer.music.play(1)
    print(""", """ , "¡¡BIENVENIDO A LA CONFIGURACION MANUAL ORIENTADA!!",
          "El sistema se ha iniciado por primera vez asi que comenzaremos ingresando el",
          "nombre del propietario",
          "y su respectiva contraseña", "", "Ingrese el nombre completo del propietario",sep='\n')
    self.__dato = input()
    self.establecer_propietario(self.__dato)

    self.__nro_try = 3
    self.__copy = ""
    while self.__dato != self.__copy:
        if self.__nro_try < 0:
            pygame.mixer.music.load("/home/pi/SIAaudios/mensAdv3.wav")
            pygame.mixer.music.play(1)
            print("numero de intentos superados","estableciendo contraseña por",
                  "defecto","contraseña: 1234",sep='\n')
            self.__dato = "1234"
            break
        pygame.mixer.music.load("/home/pi/SIAaudios/msj2.wav")
        pygame.mixer.music.play(1)
        print("Ingrese una nueva contraseña")
        self.__dato = input()
        self.__copy = self.__dato
        pygame.mixer.music.load("/home/pi/SIAaudios/msj3.wav")
        pygame.mixer.music.play(1)
        print("Repita la contraseña")
        self.__dato = input()
        self.__nro_try = self.__nro_try - 1

```

```

self.__establecer_contraseña(self.__dato)
pygame.mixer.music.load("/home/pi/SIAuidos/msj4.wav")
pygame.mixer.music.play(1)
print("Ingrese el tipo de edificacion","1:Domicilio particular"," Edificacion pequeña
donde habita una sola familia",
      "2:Otros"," Edificacion construida para albergar varias familias, como ser
residencial,condominio,edificio,etc",sep='\n')
self.__dato = input()
while True:
    if self.__dato == "1":
        break
    elif self.__dato == "2":
        break
    else:
        pygame.mixer.music.load("/home/pi/SIAuidos/msjAdv4.wav")
        pygame.mixer.music.play(1)
        print("por favor ingrese un numero de las opciones mostradas")
        self.__dato = input()

if self.__dato == "1":
    self.__configuracion_casa()
elif self.__dato == "2":
    self.__configuracion_edificio()

pygame.mixer.music.load("/home/pi/SIAuidos/msjpres.wav")
pygame.mixer.music.play(1)
print(" "," ","Se ha finalizado la configuracion",
      "Hola mi nombre es Alejandra y soy tu asistente virtual, di mi nombre si es que
necesitas algo",sep='\n')
self.__configurado = True
self.escucharMiNombre()

def establecer_nom_edif(self,nom):
    #metodo que define el nombre de la edificacion
    self.__nom_edif = nom

def establecer_propietario(self,prop):
    self.__nom_prop = prop

def __establecer_contraseña(self,contra):
    self.__pasword = contra

def establecer_direccion(self,direc):
    self.__direccion = direc

```

```

#define la configuracion inicial de la casa
def __configuracion_casa(self):
    self.crear pisos(1)
    self.piso[0].crear_apartamentos(1)
    self.piso[0].apar[0].establecer_nombre(self.__nom_prop)
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_1.wav")
    pygame.mixer.music.play(1)
    print(" ","Configuracion de domicilio particular iniciada",
          "sr: "+self.__nom_prop+" ingrese su carnet de identidad",sep='\n')
    self.__dato = int(input())
    cursor.execute("insert into residente(ci,nombre,id_apartamento,clase)
values("+str(self.__dato)+","+self.__nom_prop+",11,1)")
    db.commit()
    self.piso[0].apar[0].establecer_ci(self.__dato)
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_2.wav")
    pygame.mixer.music.play(1)
    print("por favor indique la cantidad de habitantes que viven dentro de esta casa
incluyendose a usted")
    self.__dato = int(input())
    self.piso[0].apar[0].crear_resi(self.__dato)
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_3.wav")
    pygame.mixer.music.play(1)
    print("Acabemos añadiendo algunos datos personales","Ingrese su fecha de nacimiento
(Formato aaaa-mm-dd)",sep='\n')
    self.__fn = input()
    cursor.execute("update residente set nacimiento = '"+self.__fn+"' where
ci="+str(self.piso[0].apar[0].obtener_ci()))
    db.commit()
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_4.wav")
    pygame.mixer.music.play(1)
    print("Ingrse su numero de celular")
    self.__cel = int(input())
    cursor.execute("update residente set telefono = "+str(self.__cel)+" where
ci="+str(self.piso[0].apar[0].obtener_ci()))
    db.commit()
    cursor.execute("update apartamento set propietario = '"+self.__nom_prop+"' where
id_apartamento=11")
    db.commit()
    self.piso[0].apar[0].resi[0] =
residente(self.__nom_prop,self.piso[0].apar[0].obtener_ci(),self.__fn,"Propietario",self.__cel)
    print("Estos son sus datos personales")
    self.piso[0].apar[0].resi[0].obtener_datos()
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_6.wav")
    pygame.mixer.music.play(1)
    print("¿Desea registrar al resto de los habitantes?"," 1:si"," 2:no",sep='\n')
    self.__dato = input()
    if self.__dato == "1":

```

```

self.__nro = self.piso[0].apar[0].obtener_nro_res()
self.__cont = 1
while self.__cont < self.__nro:
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_7.wav")
    pygame.mixer.music.play(1)
    print(" ", "registrando al siguiente habitante", sep='\n')
    print("Ingrese un nombre")
    self.__n = input()
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_8.wav")
    pygame.mixer.music.play(1)
    print("Ingrese el carnet de identidad correspondiente al nombre")
    self.__ci = int(input())
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_9.wav")
    pygame.mixer.music.play(1)
    print("Ingrese la fecha de nacimiento (Formato aaaa-mm-dd)")
    self.__fn = input()
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_10.wav")
    pygame.mixer.music.play(1)
    print("Ingrese el parentezco con el propietario")
    self.__p = input()
    self.__clase_p = 0
    if(self.__p=="esposo" or self.__p=="esposa"):
        self.__clase_p = 2
    elif(self.__p=="madre" or self.__p=="mama"):
        self.__clase_p = 3
    elif(self.__p=="padre" or self.__p=="papa"):
        self.__clase_p = 4
    elif(self.__p=="hermano" or self.__p=="hermana"):
        self.__clase_p = 5
    elif(self.__p=="primo" or self.__p=="prima"):
        self.__clase_p = 6
    elif(self.__p=="tio" or self.__p=="tia"):
        self.__clase_p = 7
    elif(self.__p=="suegro" or self.__p=="suegra"):
        self.__clase_p = 8
    elif(self.__p=="pareja"):
        self.__clase_p = 9
    elif(self.__p=="inquilino"):
        self.__clase_p = 10
    else:
        self.__clase_p = 11
    pygame.mixer.music.load("/home/pi/SIAaudios/msj5_11.wav")
    pygame.mixer.music.play(1)
    print("Ingrese un numero de celular relacionado a esta persona")
    self.__cel = int(input())
    self.piso[0].apar[0].resi[self.__cont] =
residente(self.__n,self.__ci,self.__fn,self.__p,self.__cel)

```

```

        cursor.execute("insert into residente
values("+str(self.__ci)+","+self.__n+","+self.__fn+","+str(self.__cel)+",11,"+str(self.__clase
_p)+"")
db.commit()
pygame.mixer.music.load("/home/pi/SIAuidos/msj5_12.wav")
pygame.mixer.music.play(1)
print(" ", "Estos son los datos registrados", " ",sep='\n')
self.piso[0].apar[0].resi[self.__cont].obtener_datos()
self.__cont = self.__cont + 1

pygame.mixer.music.load("/home/pi/SIAuidos/msj7.wav")
pygame.mixer.music.play(1)
print("¡¡¡MUY BIEN!!!, ahora seguiremos con la configuracion de la infraestructura",
      "Ingrese la cantidad de DORMITORIOS que tiene su casa",sep='\n')
self.__dato = int(input())
self.piso[0].apar[0].crear_dor(self.__dato)
pygame.mixer.music.load("/home/pi/SIAuidos/msj8.wav")
pygame.mixer.music.play(1)
print("Ingrese la cantidad de BAÑOS que hay en la casa")
self.__dato = int(input())
self.piso[0].apar[0].crear_baño(self.__dato)
pygame.mixer.music.load("/home/pi/SIAuidos/msj9.wav")
pygame.mixer.music.play(1)
print("Ingrese la cantidad de COCINAS que hay en la casa")
self.__dato = int(input())
self.piso[0].apar[0].crear_coc(self.__dato)
pygame.mixer.music.load("/home/pi/SIAuidos/msj10.wav")
pygame.mixer.music.play(1)
print("Ingrese la cantidad de SALAS que hay en la casa")
self.__dato = int(input())
self.piso[0].apar[0].crear_sala(self.__dato)
self.piso[0].apar[0].sensorxambiente()

def __configuracion_edificio(self):
    print("Se comenzara definiendo la cantidad de pisos que existe en este edificio",
          "indique la cantidad de pisos que existe en el edificio",sep='\n')
    self.__p = int(input())
    self.crear_pisos(self.__p)
    print("Indique la cantidad de apartamentos que tiene cada piso")
    self.__contador = 0
    while self.__p > self.__contador:
        print("indique la cantidad de apartamentos que existe en el piso: ",self.__contador+1)
        self.__a = int(input())
        self.piso[self.__contador].crear_apartamentos(self.__a)
        self.__contador = self.__contador+1
    print("Ha terminado la configuracion inicial del edificio",

```

" si desea personalizar cada apartamento en particular o registrar a habitantes en algun apartamento",

"solamente pronuncie la palabra: CONFIGURACION y se le mostrara una serie de opciones")

```
def generarInformeRegistros(self,orden):
    self.__dic_inf = { "genera un informe de activades":"ok","genera un informe general":"ok",
                      "manda un informe de actividades":"ok","manda un informe general":"ok",
                      "informe general":"ok",
                      "registro de actividades":"ok","informe de actividades":"ok" }
    self.__respuesta = self.__dic_inf.get(orden,None)
    if self.__respuesta == "ok":
        pygame.mixer.music.load("/home/pi/SIAaudios/regGeneral.mp3")
        pygame.mixer.music.play(1)
        registrosSIA.saveLog(self.piso[0].apar[0].baño[0].obtener_hora() +":Solicitud de informe de actividades general")
        self.__fecha = datetime.datetime.now().strftime("%d-%m-%y")
        self.__nombre ="sia"+self.__fecha+".log"
        self.__p = pdf.PDF()
        self.__p.crearPDF(self.__nombre,"INFORME DE REGISTRO DE ACTIVIDADES")
        botTelegram.enviarDocumento("Envio de registro de actividades",'siaRegistro.pdf')

    def setMensaje(self,dato):
        self.__msnWindow = dato

    def getMensaje(self):
        return self.__msnWindow

s = sia()
s.escucharMiNombre()
```

Modulo consumoAgua.

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

#diagrama de lineas
#Definir los datos
x1 = []
y1 = []

def addConsDia(dia,con): #
    x1.append(dia)
    y1.append(con)

def retFig(x,y,mes):
    fig = plt.figure()
```

```

plt.bar(x,y, label='Agua', linewidth = 4, color = 'blue')

#Definir titulo y nombre de ejes
plt.title('Servicio de agua '+mes)
plt.ylabel('Consumo en litros')
plt.xlabel('dias')
#Mostrar leyenda cuadricula y figura
plt.grid()
#plt.show()
return fig

def crearPDF(mes):
    fig1 = retFig(x1,y1,mes)
    pp = PdfPages('consumoAgua'+mes+'.pdf')
    pp.savefig(fig1)
    pp.close()

```

Módulo SensorHall.

```

import RPi.GPIO as GPIO
from time import sleep # this lets us have a time delay (see line 15)
GPIO.setmode(GPIO.BCM) # set up BCM GPIO numbering
GPIO.setup(25, GPIO.IN) # set GPIO25 as input (button)

def magnetismo():
    respuesta = None
    if GPIO.input(25): # if port 25 == 1
        respuesta = True # set port/pin value to 1/HIGH/True
    else:
        respuesta = False # set port/pin value to 0/LOW/False
    sleep(0.1) # wait 0.1 seconds
    return respuesta

```

Módulo SensorTemp.

```

import time
import smbus

class sensorTemperatura:
    def __init__(self):
        self.i2c_ch = 1
        # TMP102 address on the I2C bus
        self.i2c_address = 0x48
        # Register addresses
        self.reg_temp = 0x00
        self.reg_config = 0x01

    # Calculate the 2's complement of a number
    def twos_comp(self, val, bits):
        if (val & (1 << (bits - 1))) != 0:
            val = val - (1 << bits)
        return val

    # Read temperature registers and calculate Celsius
    def read_temp(self):

```

```

# Read temperature registers

self.val = self.bus.read_i2c_block_data(self.i2c_address, self.reg_temp, 2)
# NOTE: val[0] = MSB byte 1, val [1] = LSB byte 2
#print ("!shifted val[0] = ", bin(val[0]), "val[1] = ", bin(val[1]))

self.temp_c = (self.val[0] << 4) | (self.val[1] >> 4)
#print (" shifted val[0] = ", bin(val[0] << 4), "val[1] = ", bin(val[1] >> 4))
#print (bin(temp_c))

# Convert to 2s complement (temperatures can be negative)
self.temp_c = self.twos_comp(self.temp_c, 12)

# Convert registers value to temperature (C)
self.temp_c = self.temp_c * 0.0625

return self.temp_c

# Initialize I2C (SMBus)

# Read the CONFIG register (2 bytes)

#print("Old CONFIG:", self.val)

# Set to 4 Hz sampling (CR1, CR0 = 0b10)
def valDefine(self):
    self.bus = smbus.SMBus(self.i2c_ch)
    self.val = self.bus.read_i2c_block_data(self.i2c_address, self.reg_config, 2)
    self.val[1] = self.val[1] & 0b00111111
    self.val[1] = self.val[1] | (0b10 << 6)
    # Write 4 Hz sampling back to CONFIG
    self.bus.write_i2c_block_data(self.i2c_address, self.reg_config, self.val)
    # Read CONFIG to verify that we changed it
    self.val = self.bus.read_i2c_block_data(self.i2c_address, self.reg_config, 2)
    print("New CONFIG:", self.val)

# Print out temperature every second
def devuelveTemperatura(self):
    self.temperature = self.read_temp()
    #print(round(self.temperature, 2), "C")
    return self.temperature

```

Módulo Tanque1.

```

import RPi.GPIO as GPIO
from time import sleep  # this lets us have a time delay (see line 15)
GPIO.setmode(GPIO.BCM)  # set up BCM GPIO numbering
GPIO.setup(5, GPIO.IN)  # set GPIO25 as input (button)

def estado():
    respuesta = None
    if GPIO.input(5): # if port 25 == 1
        respuesta = True      # set port/pin value to 1/HIGH/True
    else:

```

```

    respuesta = False      # set port/pin value to 0/LOW/False
    sleep(0.1)      # wait 0.1 seconds
    return respuesta

```

Módulo Tanque 2.

```

import RPi.GPIO as GPIO
from time import sleep  # this lets us have a time delay (see line 15)
GPIO.setmode(GPIO.BCM)  # set up BCM GPIO numbering
GPIO.setup(6, GPIO.IN)  # set GPIO25 as input (button)

def estado():
    respuesta = None
    if GPIO.input(6): # if port 25 == 1
        respuesta = True      # set port/pin value to 1/HIGH/True
    else:
        respuesta = False      # set port/pin value to 0/LOW/False
        sleep(0.1)      # wait 0.1 seconds
    return respuesta

```

Módulo botTelegram.

```

import requests

def telegram_bot_sendtext(bot_message):

    bot_token = '1837548266:AAHuE157PauOzTHDaggwsMrIU4BJkM8kKL0'
    bot_chatID = '400386487'
    send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id=' + bot_chatID +
    '&parse_mode=Markdown&text=' + bot_message

    response = requests.get(send_text)

    return response.json()

def enviarMensajeTelegram(mensaje):
    test = telegram_bot_sendtext(mensaje)

def enviarDocumento(mensaje,archivo):
    bot_token = '1837548266:AAHuE157PauOzTHDaggwsMrIU4BJkM8kKL0'
    bot_chatID = '400386487'
    response =
    requests.post('https://api.telegram.org/bot'+bot_token+'/sendDocument',files={'document':(archivo,open(archivo,'rb')),'thumb':(archivo,open(archivo,'rb'))},data={'chat_id':bot_chatID,'caption':mensaje})
    return response.json()

def enviarMensajeGrupal(bot_message):

    bot_token = '1837548266:AAHuE157PauOzTHDaggwsMrIU4BJkM8kKL0'
    bot_grupoID = '-441604449'
    send_text = 'https://api.telegram.org/bot' + bot_token + '/sendMessage?chat_id=' + bot_grupoID +
    '&parse_mode=Markdown&text=' + bot_message

    response = requests.get(send_text)

```

```
return response.json()
```

Módulo RegistrosSia.

```
import logging
import datetime

fecha = datetime.datetime.now().strftime("%d-%m-%Y")
nombre = "sia"+fecha+".log"
logging.basicConfig(level=logging.DEBUG, filename=nombre)

def saveLog(mensaje):
    logging.info(fecha + ":" + mensaje)
```

Módulo PDF.

```
from fpdf import FPDF

class PDF(FPDF):
    pass

    def logo(self, name, posx, posy, ancho, alto):
        self.image(name, posx, posy, ancho, alto)

    def texto(self, name):
        with open(name, 'rb') as xy:
            txt = xy.read().decode('latin-1')
        self.set_xy(10.0,40.0)
        self.set_text_color(0,0,0)
        self.set_font('Arial', 'B', 12)
        self.multi_cell(0,6,txt)

    def titulo(self, title):
        self.set_xy(0,0)
        self.set_text_color(0,0,0)
        self.set_font('Arial', 'B', 20)
        self.cell(w=150, h=40, align='C', txt=title, border=0)

    def Header(self, titu):
        self.set_font('Arial', 'B', 20)
        self.cell(80);
        self.cell(30,10,'SIA',0,1,'C')
        self.set_font('Arial', 'B', 14)
        self.cell(80);
        self.cell(30,10,titu,0,1,'C')
        self.ln(10);

    def crearPDF(self, name, titu):
        self.add_page()
        self.logo('cake.jpg', 5, 5, 20, 20)
        #self.titulo(titu)
        self.Header(titu)
        self.texto(name)
        self.set_author("David Heredia")
        self.output('siaRegistro.pdf', 'F')
```

ANEXO D: Fichas Técnicas.

Digital magnetic sensor SKU DFR0033



Introduction

What's the best way to detect a magnet? Use another magnet. But if it's not sensitive enough, you may have to feel it by yourself. Right? This magnetic sensor knows whether there is a magnetic object nearby or not. And it correctly tells you through digital output. See below picture for a quick demo!

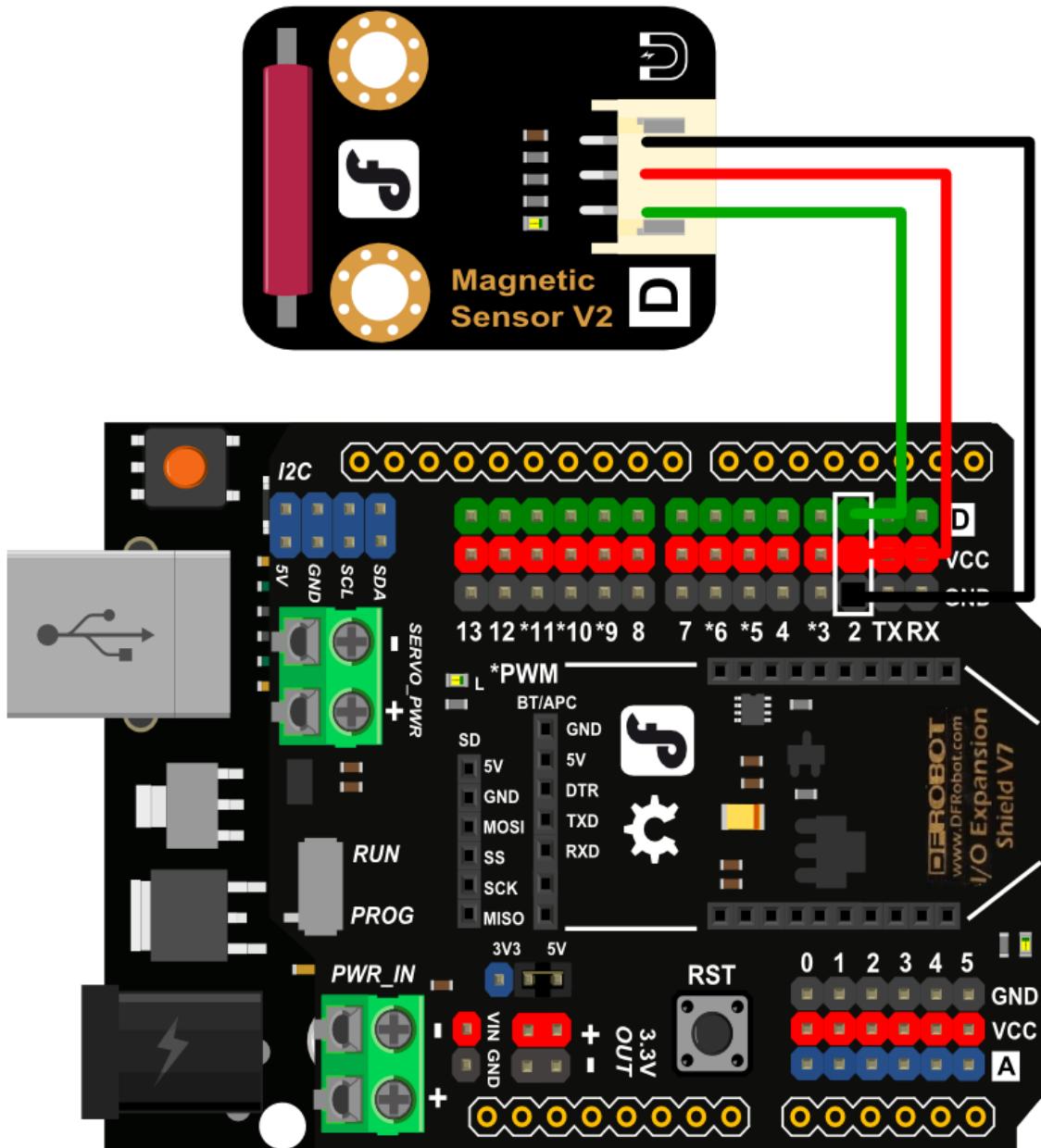
Specification

- Supply Voltage: 3.3V to 5V
- Indicator LED on board
- Interface: Digital
- Size: 22x30mm

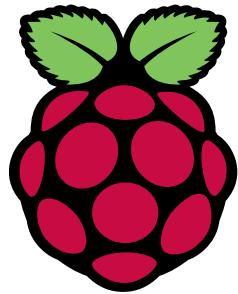
Sample Code

```
int ledPin = 13;           // choose the pin for the LED
int inputPin = 2;          // choose the input pin
int val = 0;               // variable for reading the pin status
void setup() {
    pinMode(ledPin, OUTPUT); // declare LED as output
    pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
    val = digitalRead(inputPin); // read input value
    if (val == HIGH) {          // check if the input is HIGH
        digitalWrite(ledPin, LOW); // turn LED OFF
    } else {
        digitalWrite(ledPin, HIGH); // turn LED ON
    }
}
```

Connection Diagram



DATASHEET



Raspberry Pi 4 Model B

Release 1

June 2019

Copyright 2019 Raspberry Pi (Trading) Ltd. All rights reserved.



Table 1: Release History

Release	Date	Description
1	21/06/2019	First release

The latest release of this document can be found at <https://www.raspberrypi.org>



Contents

1	Introduction	5
2	Features	6
2.1	Hardware	6
2.2	Interfaces	6
2.3	Software	7
3	Mechanical Specification	7
4	Electrical Specification	7
4.1	Power Requirements	9
5	Peripherals	9
5.1	GPIO Interface	9
5.1.1	GPIO Pin Assignments	9
5.1.2	GPIO Alternate Functions	10
5.1.3	Display Parallel Interface (DPI)	11
5.1.4	SD/SDIO Interface	11
5.2	Camera and Display Interfaces	11
5.3	USB	11
5.4	HDMI	11
5.5	Audio and Composite (TV Out)	11
5.6	Temperature Range and Thermals	11
6	Availability	12
7	Support	12



List of Figures

1	Mechanical Dimensions	7
2	Digital IO Characteristics	8
3	GPIO Connector Pinout	9



List of Tables

1	Release History	1
2	Absolute Maximum Ratings	8
3	DC Characteristics	8
4	Digital I/O Pin AC Characteristics	8
5	Raspberry Pi 4 GPIO Alternate Functions	10



1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with significantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is available with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.



2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
 - Up to 6x UART
 - Up to 6x I2C
 - Up to 5x SPI
 - 1x SDIO interface
 - 1x DPI (Parallel RGB Display)
 - 1x PCM
 - Up to 2x PWM channels
 - Up to 3x GPCLK outputs



2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
 - Recent Linux kernel support
 - Many drivers upstreamed
 - Stable and well supported userland
 - Availability of GPU functions using standard APIs

3 Mechanical Specification

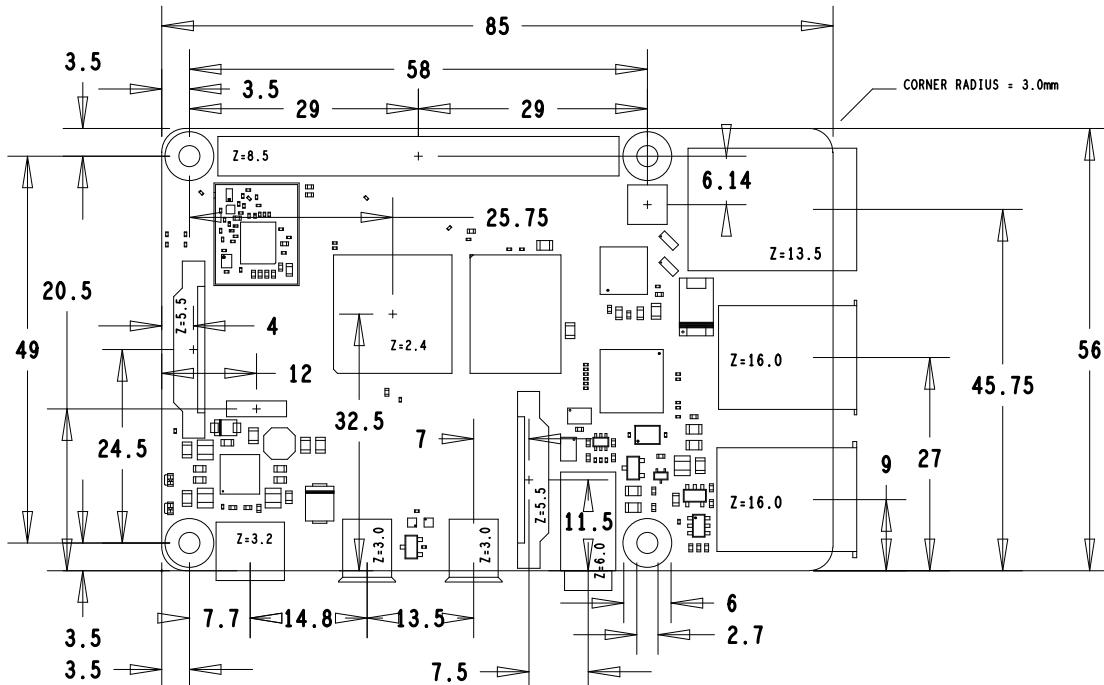


Figure 1: Mechanical Dimensions

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.



Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage ^a	$VDD_IO = 3.3V$	-	-	TBD	V
V_{IH}	Input high voltage ^a	$VDD_IO = 3.3V$	TBD	-	-	V
I_{IL}	Input leakage current	$TA = +85^\circ C$	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage ^b	$VDD_IO = 3.3V, IOL = -2mA$	-	-	TBD	V
V_{OH}	Output high voltage ^b	$VDD_IO = 3.3V, IOH = 2mA$	TBD	-	-	V
I_{OL}	Output low current ^c	$VDD_IO = 3.3V, VO = 0.4V$	TBD	-	-	mA
I_{OH}	Output high current ^c	$VDD_IO = 3.3V, VO = 2.3V$	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	$k\Omega$
R_{PD}	Pulldown resistor	-	TBD	-	TBD	$k\Omega$

^a Hysteresis enabled

^b Default drive strength (8mA)

^c Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time ^a	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time ^a	-	TBD	-	ns

^a Default drive strength, $CL = 5pF$, $VDD_IO = 3.3V$

Table 4: Digital I/O Pin AC Characteristics

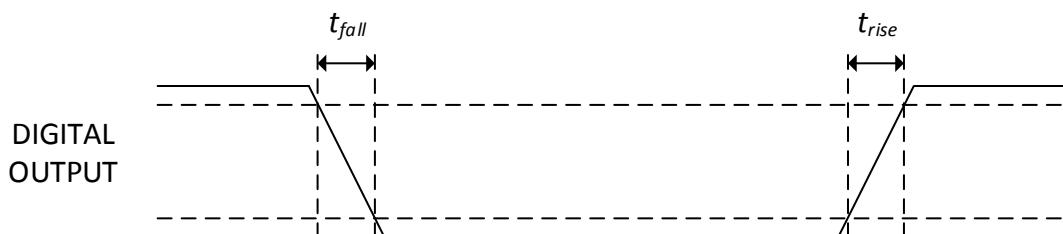


Figure 2: Digital IO Characteristics



4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

5 Peripherals

5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

5.1.1 GPIO Pin Assignments

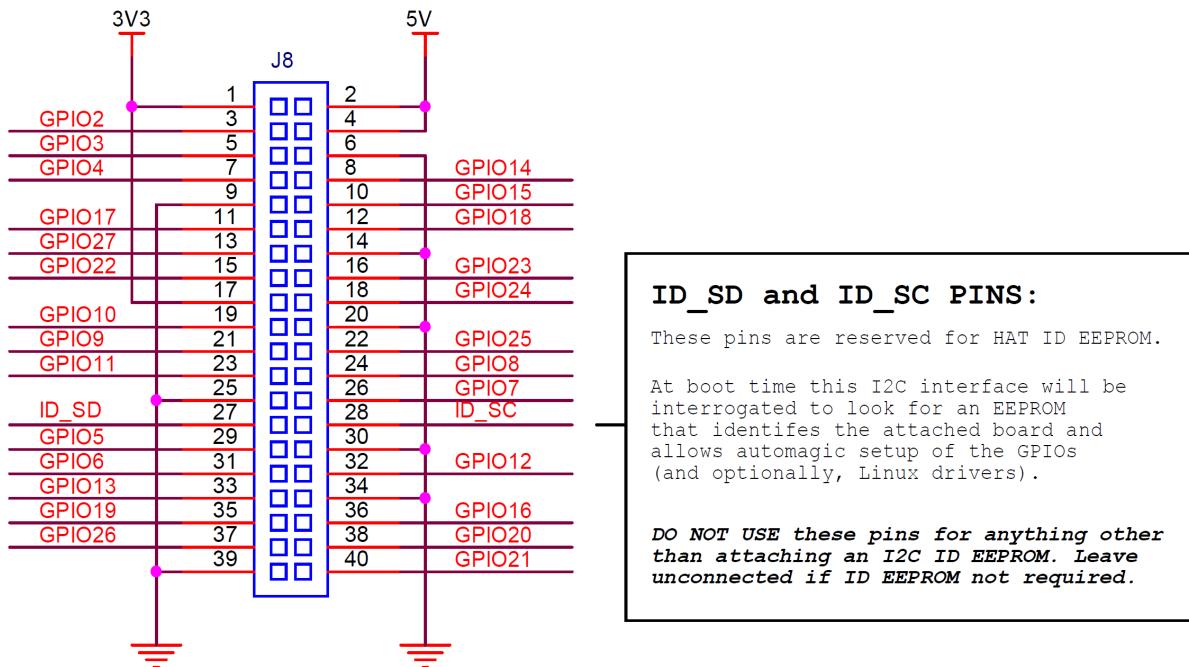


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.



5.1.2 GPIO Alternate Functions

GPIO	Pull	Default					
		ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPLD0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPLD1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPLD2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPLD3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPLD4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPLD5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPLD6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPLD7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPLD8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPLD9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPLD10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPLD11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPLD12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPLD13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPLD14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPLD15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPLD16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPLD17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPLD18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPLD19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPLD20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPLD21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPLD22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPLD23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the [hardware documentation](#) section of the website.



5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which supports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high temperature at full performance, further cooling may be needed.

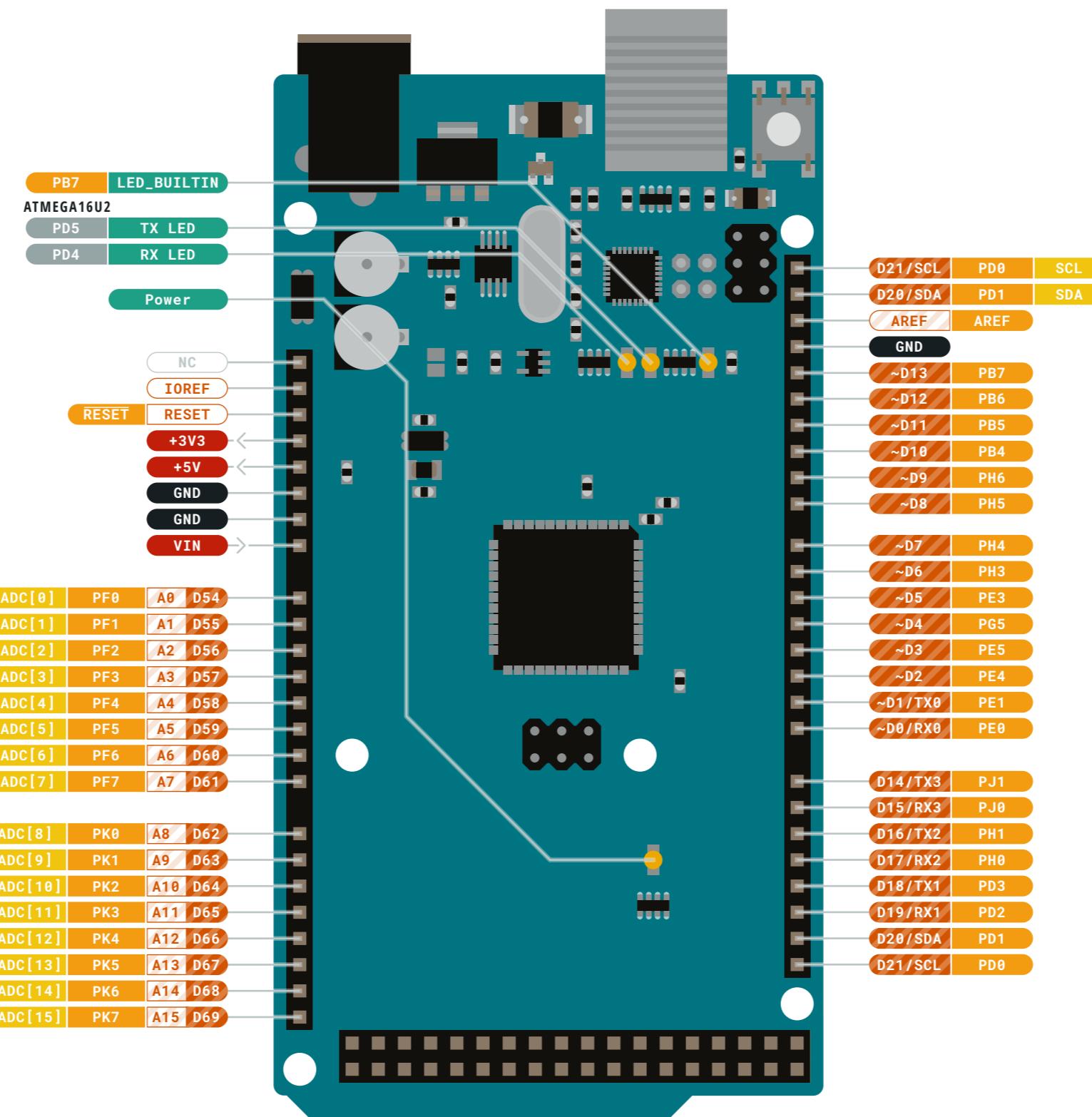


6 Availability

Raspberry Pi guarantee availability Pi4B until at least January 2026.

7 Support

For support please see the hardware documentation section of the [Raspberry Pi website](#) and post questions to the [Raspberry Pi forum](#).



- Ground
- Power
- LED
- Internal Pin
- SWD Pin

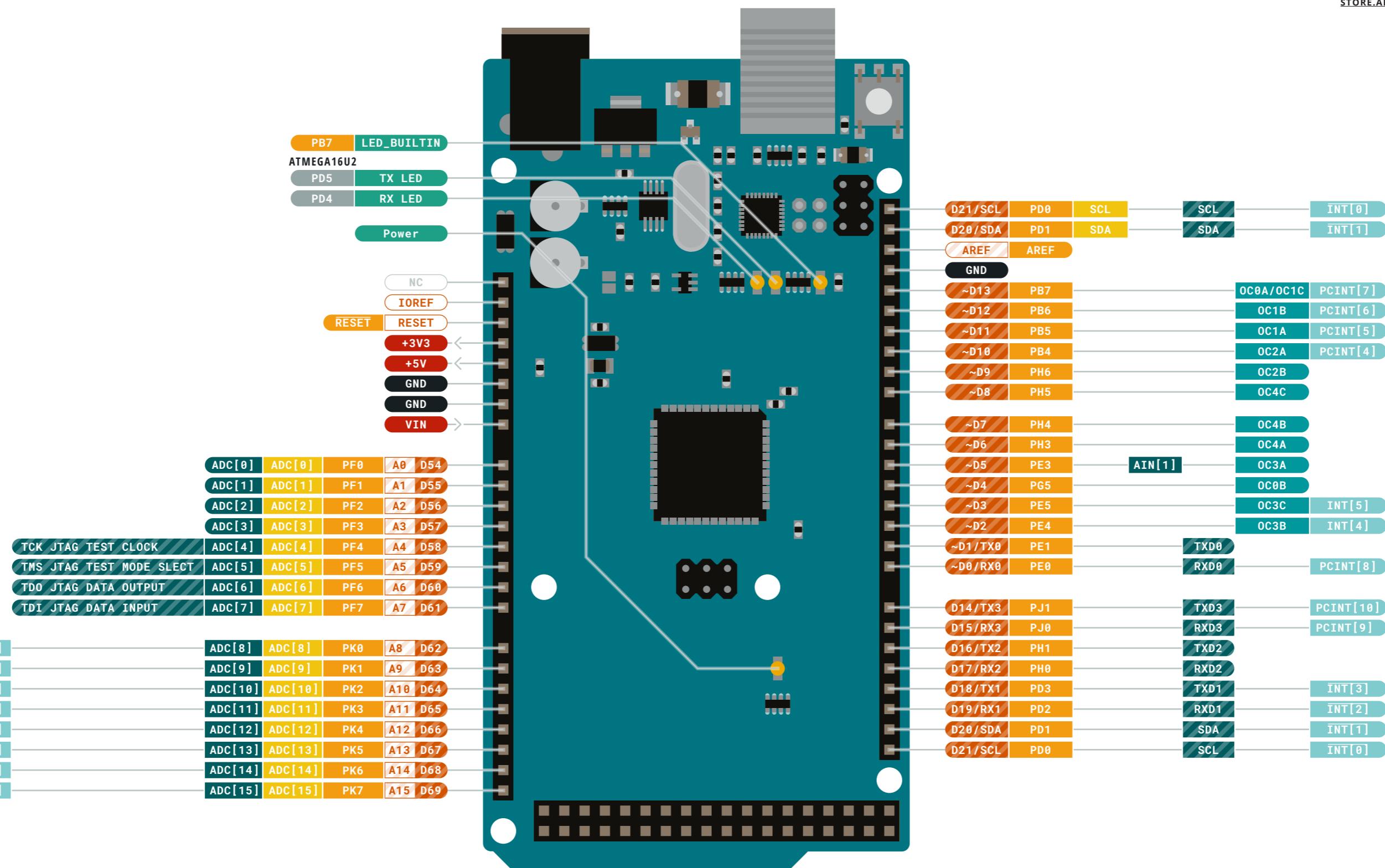
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default

- ! **MAXIMUM** current per I/O pin is 20mA
! **MAXIMUM** current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

Last update: 16/12/2020





- Ground
- Power
- LED
- Internal Pin
- SWD Pin

- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default
- Analog
- Communication
- Timer
- Interrupt
- Sercom

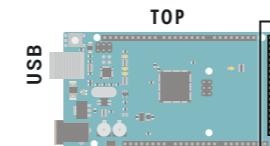
- ⚠ **MAXIMUM** current per I/O pin is 20mA
 ⚠ **MAXIMUM** current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

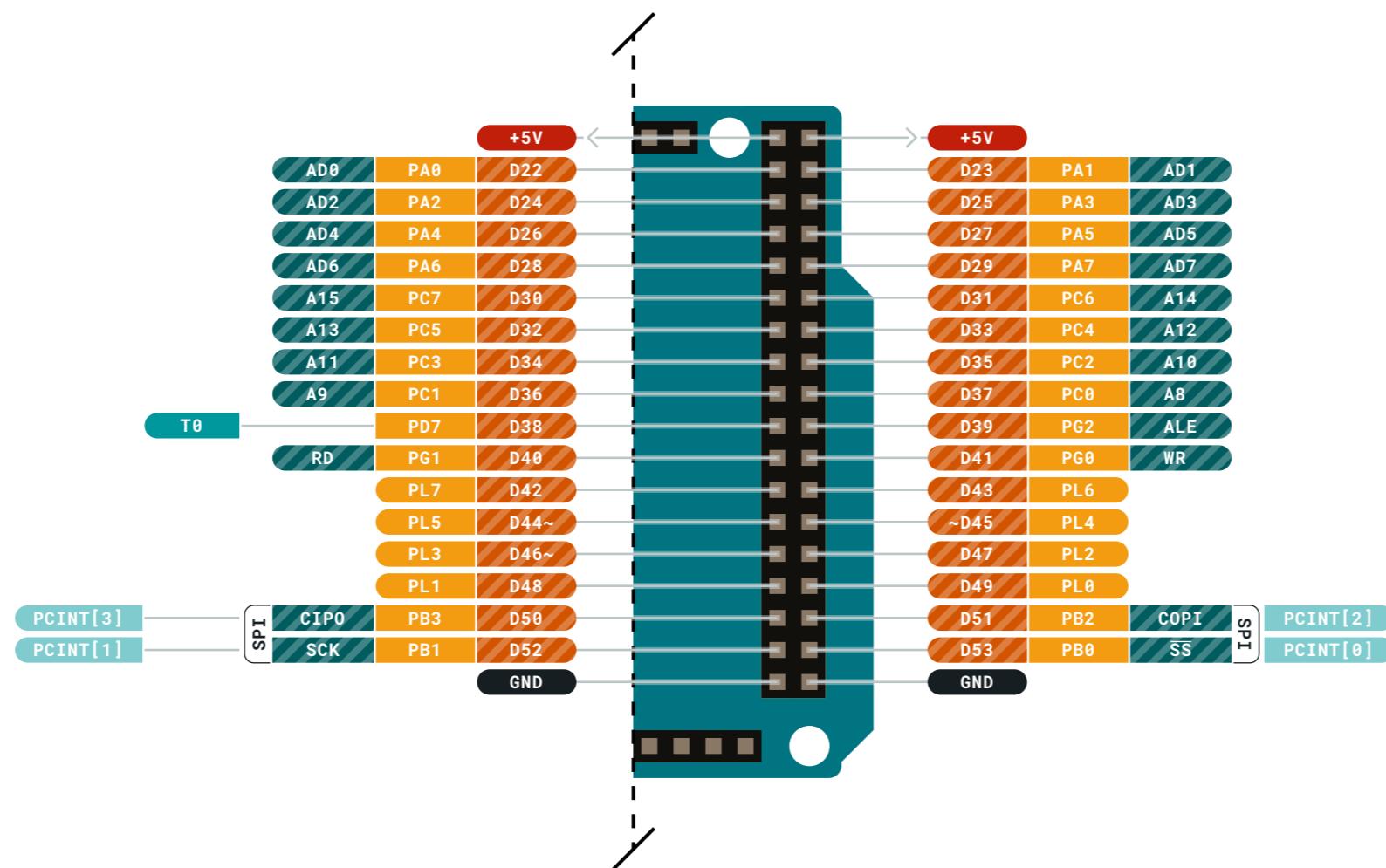
Last update: 16/12/2020



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.



Digital pins D22-D53



	Ground
	Power
	LED
	Internal Pin
	SWD Pin

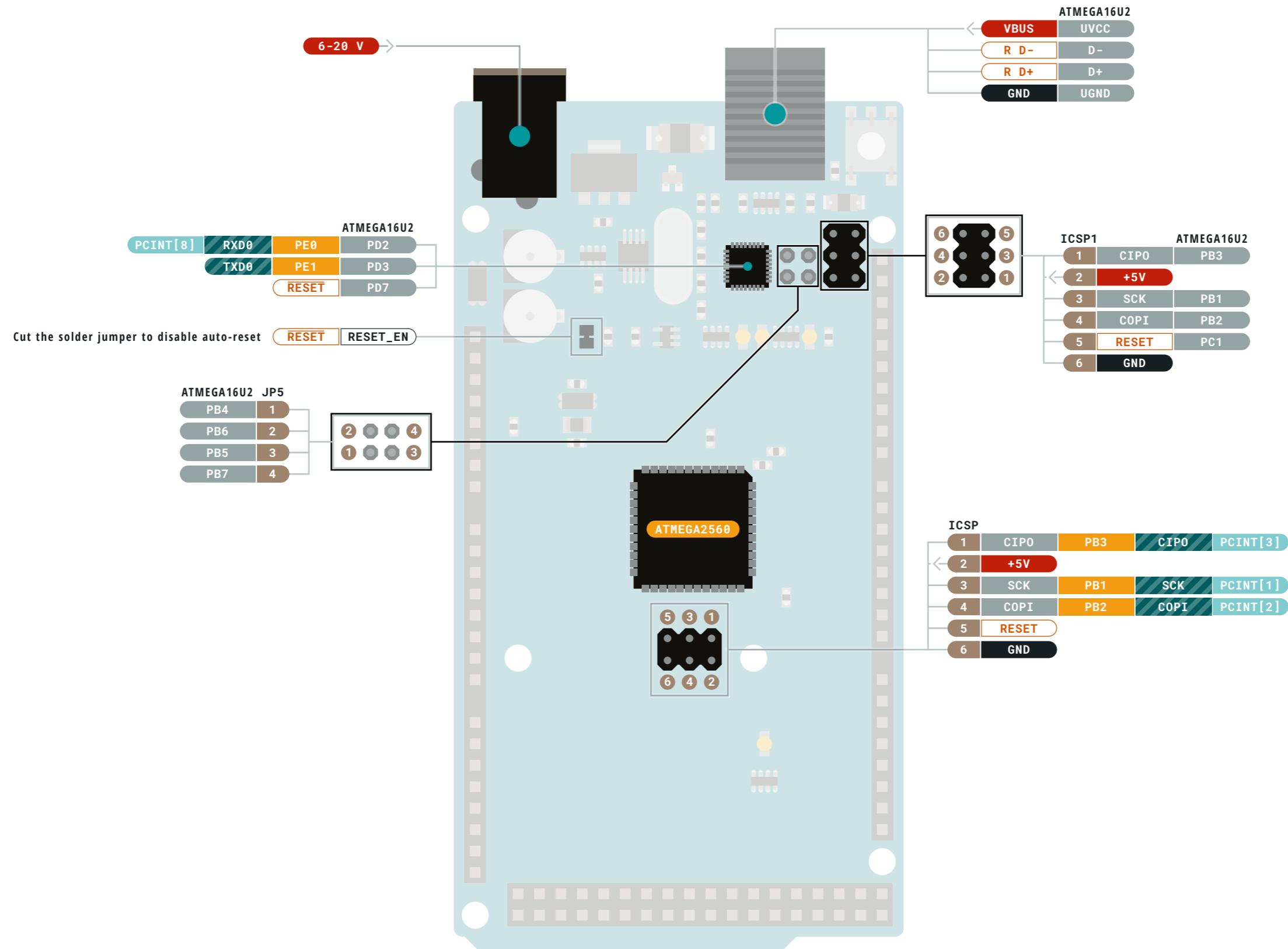
	Digital Pin
	Analog Pin
	Other Pin
	Microcontroller's Port
	Default

	Analog
	Communication
	Timer
	Interrupt
	Sercom

- MAXIMUM** current per I/O pin is 20mA
- MAXIMUM** current per +3.3V pin is 50mA

VIN 6-20 V input to the board.
 NOTE: CIPO/COPI have previously been referred to as MISO/MOSI





- Ground
- Power
- LED
- Internal Pin
- SWD Pin

- Digital Pin
- Analog
- Communication
- Other Pin
- Microcontroller's Port
- Default
- Analog Pin
- Timer
- Interrupt
- Sercom

- MAXIMUM** current per I/O pin is 20mA
- MAXIMUM** current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

NOTE: CIPO/COPI have previously been referred to as MISO/MOSI

ARDUINO.CC

Last update: 16/12/2020



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Tech specs

MICROCONTROLLER	ATmega2560
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	101.52 mm
WIDTH	53.3 mm
WEIGHT	37 g

JQC-3FF

SUBMINIATURE HIGH POWER RELAY



File No.:R50034671



File No.:E133481



File No.:CQC02001001953



Features

- * Extremely low cost
- * SPST-NO & SPDT configuration
- * Subminiature, standard PCB layout
- * Sealed IP67 and Flux proof types available

CONTACT DATA

Contact Arrangement	1A	1C
Initial Contact Resistance Max.		100mΩ (at 1A 6VDC)
Contact Material		Silver Alloy
Contact Rating (Res. Load)	10A 277VAC	7A 250VDC 10A 277VAC
Max. switching voltage		277VAC/30VDC
Max. switching current	15A	10A
Max. switching power		2770VA 210W
Mechanical life		1 x 10 ⁷ OPS
Electrical life		1 x 10 ⁵ OPS

COIL DATA

Nominal Voltage VDC	Pick-up Voltage VDC	Drop-out Voltage VDC	Max. allowable Voltage VDC(at 25°C)	Coil Resistance Ω
5	3.80	0.5	6.5	70 ± 10%
6	4.50	0.6	7.8	100 ± 10%
9	6.80	0.9	11.7	225 ± 10%
12	9.00	1.2	15.6	400 ± 10%
18	13.5	1.8	23.4	900 ± 10%
24	18.0	2.4	31.2	1600 ± 10%
48	36.0	4.8	62.4	4500 ± 10%

CHARACTERISTICS

Initial Insulation Resistance	100MΩ ,500VDC
Dielectric Strength	Between coil and contacts 1500VAC, 1min
	Between open contacts 750VAC, 1min
Operate time (at nomi. Volt.)	Max. 10ms
Release time (at nomi. Volt.)	Max. 5ms
Temperature rise (at nomi. Volt.)	Max. 60°C
Shock Resistance	Functional 98 m/s ² (10g)
	Destructive 980 m/s ² (100g)
Vibration Resistance	1.5mm, 10 to 55Hz
Humidity	35% to 85%RH
Ambient temperature	-40°C to +85°C
Termination	PCB
Unit weight	Approx. 10g
Construction	Sealed IP67 & Flux proof

SAFETY APPROVAL RATINGS

UL	1 Form C	10A 277 VAC
		10A 120VAC
TÜV	1 Form A	1/2 HP 125/250VAC
		10A 277VAC
TÜV	1 Form C	TV-5 120VAC
		15A 125VAC
TÜV	1 Form A	120VAC 125VAC
		1/2hp,125VAC
TÜV	1 Form C	8A 250VAC
		12A 125VAC cos phi=1
TÜV	1 Form A	5A 250VAC cos phi=1
		10A 277VAC
TÜV	1 Form C	12A 125VAC cos phi=1
		5A 250VAC cos phi=1

COIL

Coil power	0.36W*48VDC : 0.51W*
------------	----------------------



HONGFA RELAY

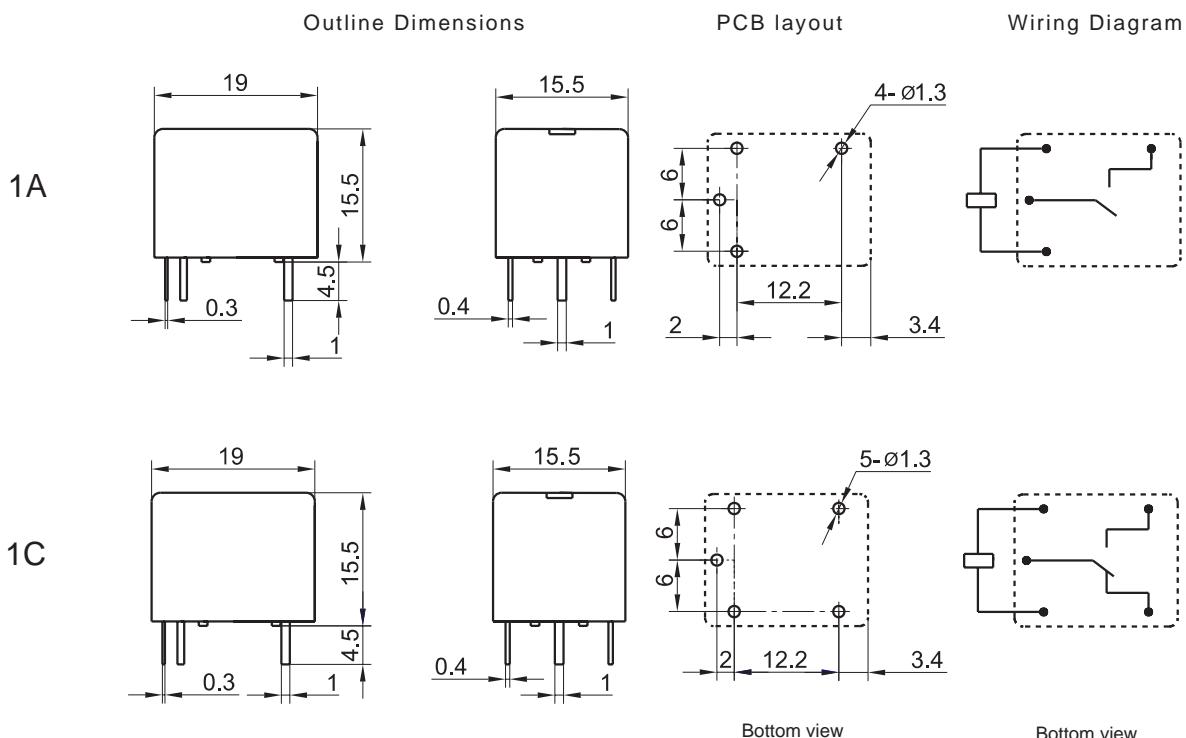
ISO9001*ISO/TS16949 *ISO14001 CERTIFIED

VERSION: EN02-20040601

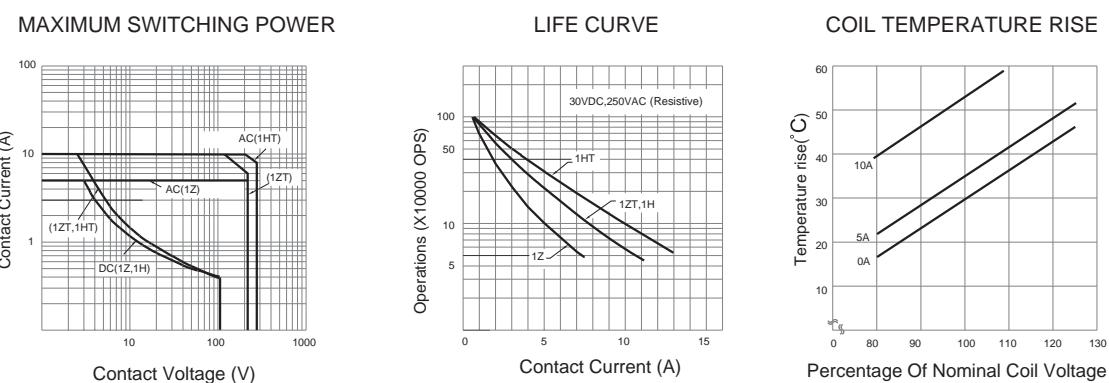
ORDERING INFORMATION

	JQC-3FF / 012	1H	S	T	F
Type					
Coil voltage	5, 6, 9, 12, 18, 24, 48VDC				
Contact arrangement	1H:1 Form A (SPST-NO) 1Z:1 Form C (SPDT)				
Structure	S: Sealed IP67 Nil: Flux proof				
Contact Material	T: AgSnO ₂ Nil: AgCdo				
Insulation System	F: Class F 155°C Nil: Class B 130°C				

OUTLINE DIMENSIONS, WIRING DIAGRAM AND PC BOARD LAYOUT

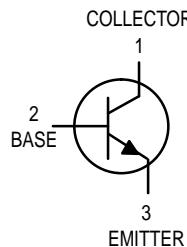


CHARACTERISTICS CURVE

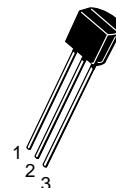


Amplifier Transistors

NPN Silicon



P2N2222A



CASE 29-04, STYLE 17
TO-92 (TO-226AA)

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Collector-Emitter Voltage	V_{CEO}	40	Vdc
Collector-Base Voltage	V_{CBO}	75	Vdc
Emitter-Base Voltage	V_{EBO}	6.0	Vdc
Collector Current — Continuous	I_C	600	mAdc
Total Device Dissipation @ $T_A = 25^\circ\text{C}$ Derate above 25°C	P_D	625 5.0	mW mW/ $^\circ\text{C}$
Total Device Dissipation @ $T_C = 25^\circ\text{C}$ Derate above 25°C	P_D	1.5 12	Watts mW/ $^\circ\text{C}$
Operating and Storage Junction Temperature Range	T_J, T_{Stg}	-55 to +150	$^\circ\text{C}$

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	200	$^\circ\text{C}/\text{W}$
Thermal Resistance, Junction to Case	$R_{\theta JC}$	83.3	$^\circ\text{C}/\text{W}$

ELECTRICAL CHARACTERISTICS ($T_A = 25^\circ\text{C}$ unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
OFF CHARACTERISTICS				

Collector-Emitter Breakdown Voltage ($I_C = 10 \text{ mA}\text{dc}$, $I_B = 0$)	$V_{(BR)CEO}$	40	—	Vdc
Collector-Base Breakdown Voltage ($I_C = 10 \mu\text{A}\text{dc}$, $I_E = 0$)	$V_{(BR)CBO}$	75	—	Vdc
Emitter-Base Breakdown Voltage ($I_E = 10 \mu\text{A}\text{dc}$, $I_C = 0$)	$V_{(BR)EBO}$	6.0	—	Vdc
Collector Cutoff Current ($V_{CE} = 60 \text{ Vdc}$, $V_{EB(\text{off})} = 3.0 \text{ Vdc}$)	I_{CEX}	—	10	nAdc
Collector Cutoff Current ($V_{CB} = 60 \text{ Vdc}$, $I_E = 0$) ($V_{CB} = 60 \text{ Vdc}$, $I_E = 0$, $T_A = 150^\circ\text{C}$)	I_{CBO}	— —	0.01 10	μAdc
Emitter Cutoff Current ($V_{EB} = 3.0 \text{ Vdc}$, $I_C = 0$)	I_{EBO}	—	10	nAdc
Collector Cutoff Current ($V_{CE} = 10 \text{ V}$)	I_{CEO}	—	10	nAdc
Base Cutoff Current ($V_{CE} = 60 \text{ Vdc}$, $V_{EB(\text{off})} = 3.0 \text{ Vdc}$)	I_{BEX}	—	20	nAdc

P2N2222A
ELECTRICAL CHARACTERISTICS ($T_A = 25^\circ\text{C}$ unless otherwise noted) (Continued)

Characteristic	Symbol	Min	Max	Unit
ON CHARACTERISTICS				
DC Current Gain ($I_C = 0.1 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$) ($I_C = 1.0 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $T_A = -55^\circ\text{C}$) ($I_C = 150 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$) ⁽¹⁾ ($I_C = 150 \text{ mA}_\text{dc}$, $V_{CE} = 1.0 \text{ V}_\text{dc}$) ⁽¹⁾ ($I_C = 500 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$) ⁽¹⁾	h_{FE}	35 50 75 35 100 50 40	— — — — 300 — —	—
Collector-Emitter Saturation Voltage ⁽¹⁾ ($I_C = 150 \text{ mA}_\text{dc}$, $I_B = 15 \text{ mA}_\text{dc}$) ($I_C = 500 \text{ mA}_\text{dc}$, $I_B = 50 \text{ mA}_\text{dc}$)	$V_{CE(\text{sat})}$	— —	0.3 1.0	V_dc
Base-Emitter Saturation Voltage ⁽¹⁾ ($I_C = 150 \text{ mA}_\text{dc}$, $I_B = 15 \text{ mA}_\text{dc}$) ($I_C = 500 \text{ mA}_\text{dc}$, $I_B = 50 \text{ mA}_\text{dc}$)	$V_{BE(\text{sat})}$	0.6 —	1.2 2.0	V_dc

SMALL-SIGNAL CHARACTERISTICS

Current-Gain — Bandwidth Product ⁽²⁾ ($I_C = 20 \text{ mA}_\text{dc}$, $V_{CE} = 20 \text{ V}_\text{dc}$, $f = 100 \text{ MHz}$)	f_T	300	—	MHz
Output Capacitance ($V_{CB} = 10 \text{ V}_\text{dc}$, $I_E = 0$, $f = 1.0 \text{ MHz}$)	C_{obo}	—	8.0	pF
Input Capacitance ($V_{EB} = 0.5 \text{ V}_\text{dc}$, $I_C = 0$, $f = 1.0 \text{ MHz}$)	C_{ibo}	—	25	pF
Input Impedance ($I_C = 1.0 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$)	h_{ie}	2.0 0.25	8.0 1.25	kΩ
Voltage Feedback Ratio ($I_C = 1.0 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$)	h_{re}	— —	8.0 4.0	$\times 10^{-4}$
Small-Signal Current Gain ($I_C = 1.0 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$)	h_{fe}	50 75	300 375	—
Output Admittance ($I_C = 1.0 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$) ($I_C = 10 \text{ mA}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $f = 1.0 \text{ kHz}$)	h_{oe}	5.0 25	35 200	μmhos
Collector Base Time Constant ($I_E = 20 \text{ mA}_\text{dc}$, $V_{CB} = 20 \text{ V}_\text{dc}$, $f = 31.8 \text{ MHz}$)	$r_b' C_C$	—	150	ps
Noise Figure ($I_C = 100 \mu\text{A}_\text{dc}$, $V_{CE} = 10 \text{ V}_\text{dc}$, $R_S = 1.0 \text{ k}\Omega$, $f = 1.0 \text{ kHz}$)	N_F	—	4.0	dB

SWITCHING CHARACTERISTICS

Delay Time	$(V_{CC} = 30 \text{ V}_\text{dc}$, $V_{BE(\text{off})} = -2.0 \text{ V}_\text{dc}$, $I_C = 150 \text{ mA}_\text{dc}$, $I_{B1} = 15 \text{ mA}_\text{dc}$) (Figure 1)	t_d	—	10	ns
Rise Time		t_r	—	25	ns
Storage Time	$(V_{CC} = 30 \text{ V}_\text{dc}$, $I_C = 150 \text{ mA}_\text{dc}$, $I_{B1} = I_{B2} = 15 \text{ mA}_\text{dc}$) (Figure 2)	t_s	—	225	ns
Fall Time		t_f	—	60	ns

1. Pulse Test: Pulse Width $\leq 300 \mu\text{s}$, Duty Cycle $\leq 2.0\%$.
2. f_T is defined as the frequency at which $|h_{fe}|$ extrapolates to unity.

SWITCHING TIME EQUIVALENT TEST CIRCUITS

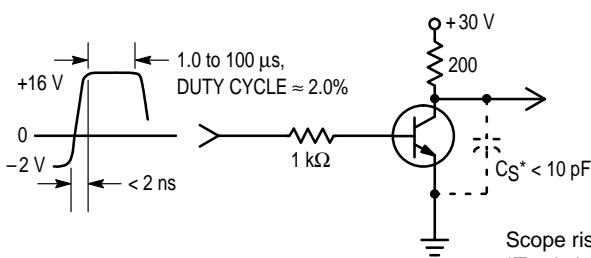


Figure 1. Turn-On Time

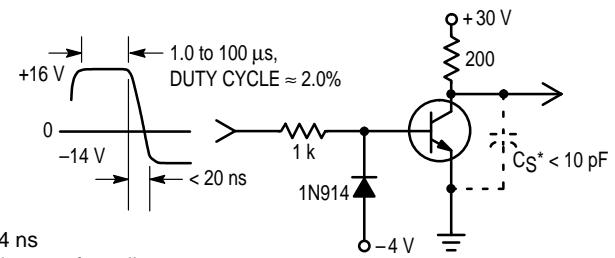


Figure 2. Turn-Off Time

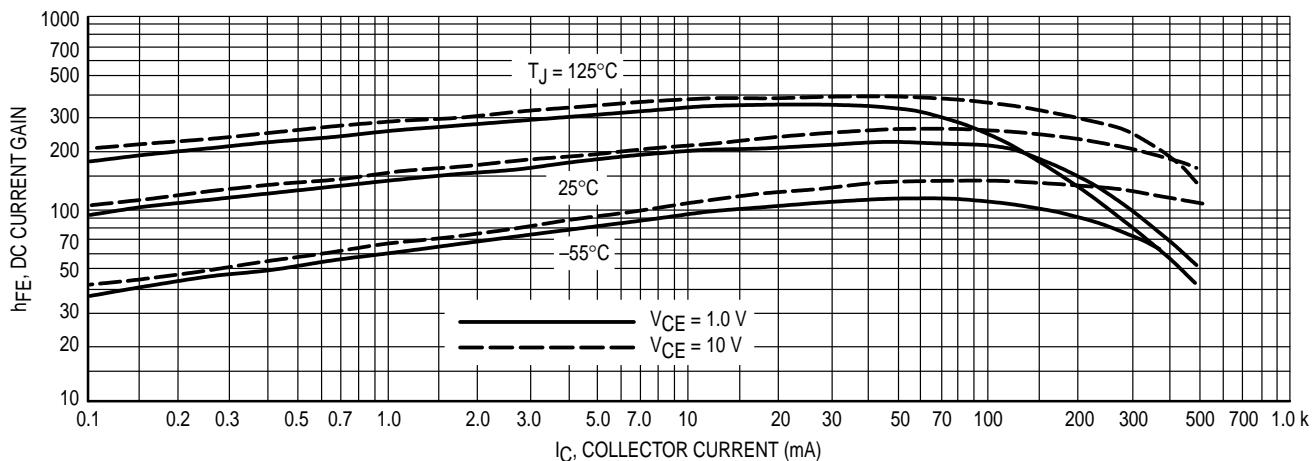


Figure 3. DC Current Gain

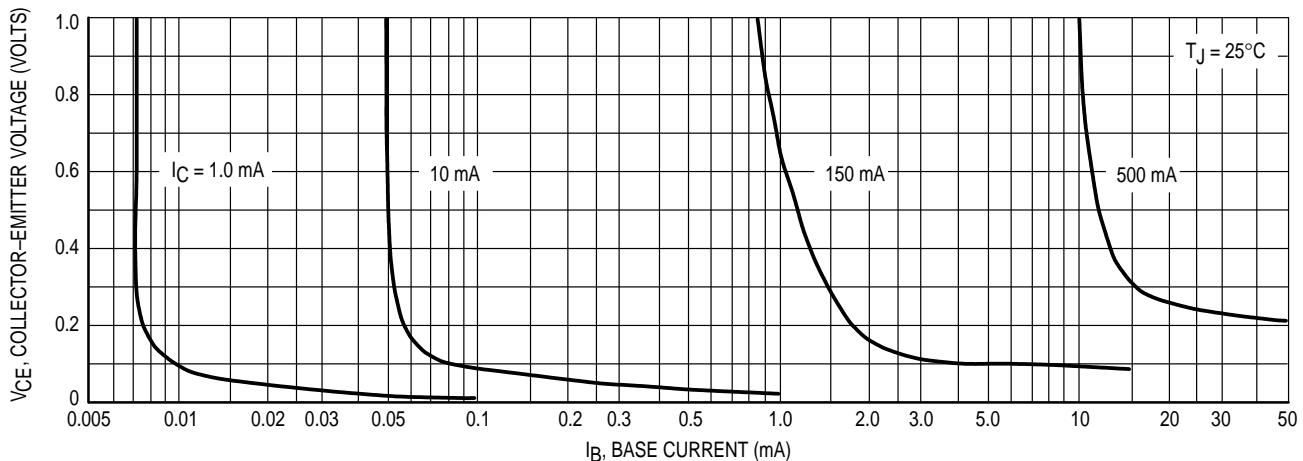


Figure 4. Collector Saturation Region

P2N2222A

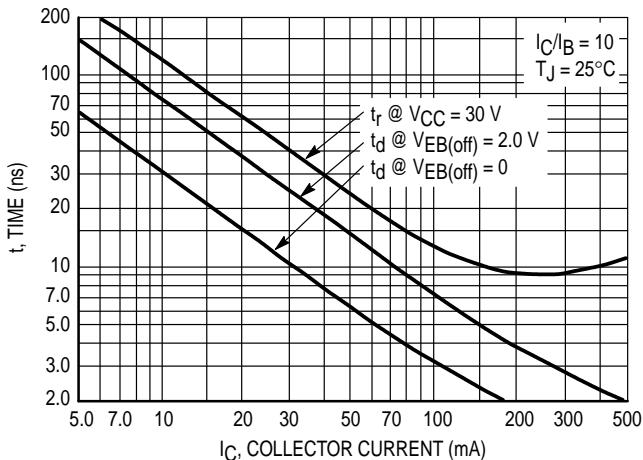


Figure 5. Turn-On Time

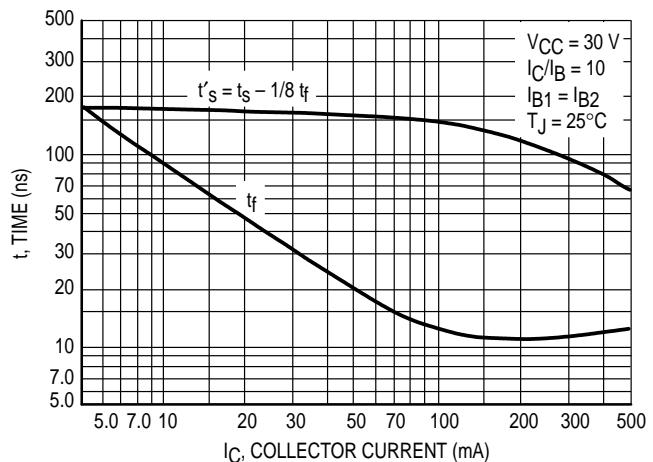


Figure 6. Turn-Off Time

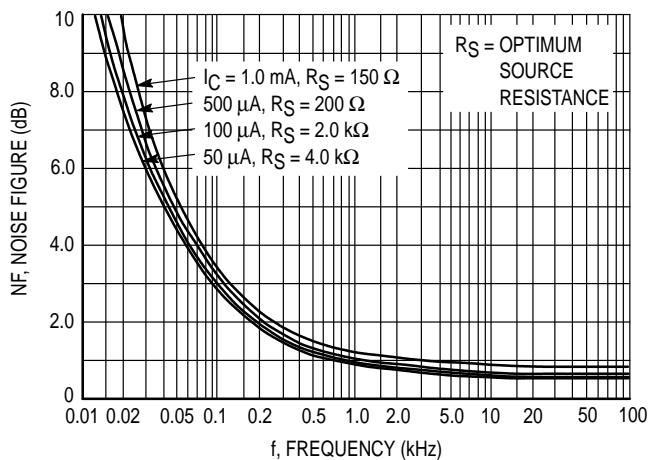


Figure 7. Frequency Effects

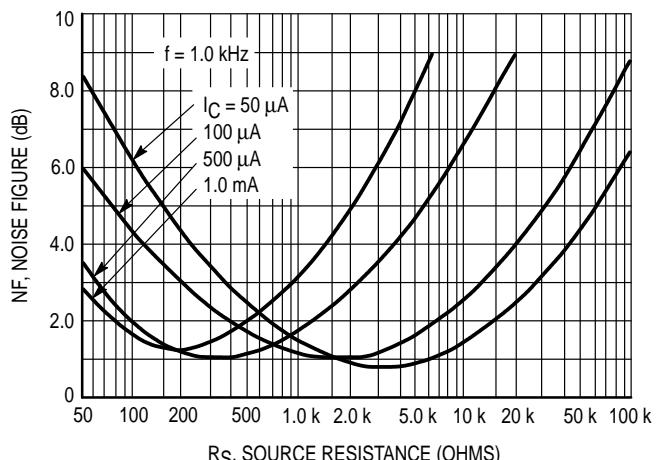


Figure 8. Source Resistance Effects

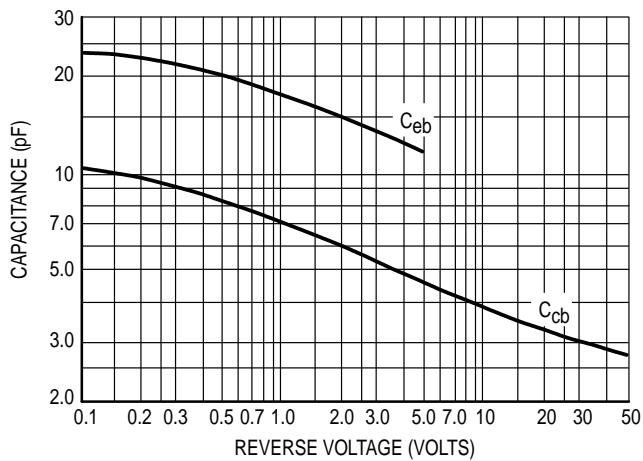


Figure 9. Capacitances

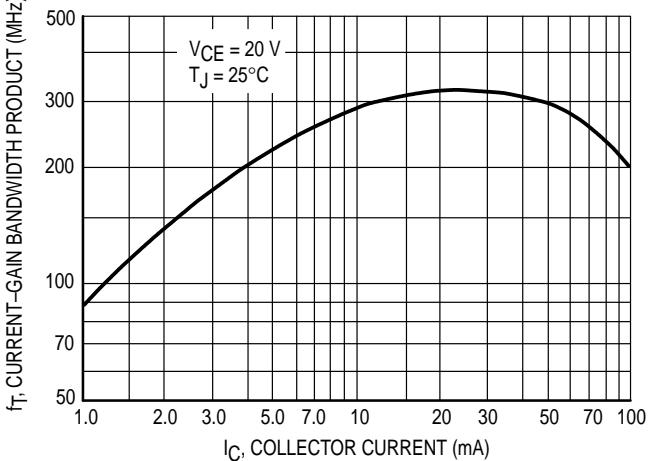


Figure 10. Current-Gain Bandwidth Product

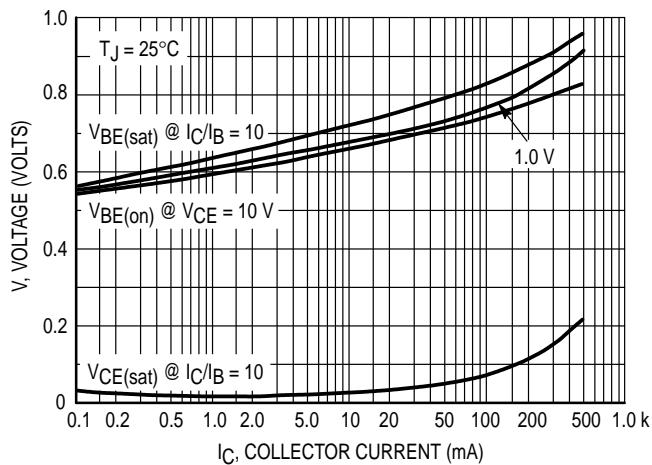


Figure 11. "On" Voltages

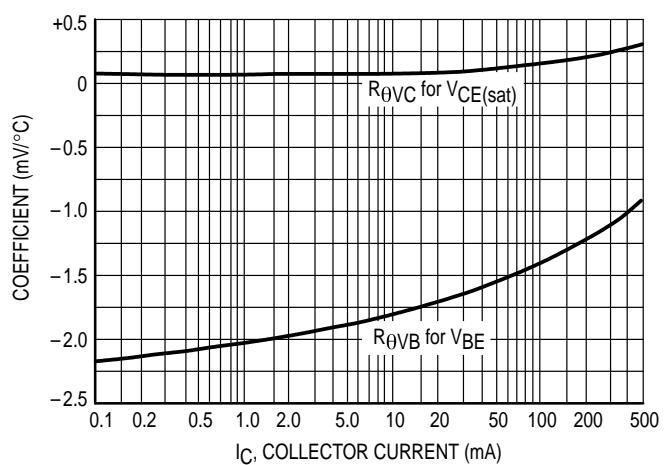
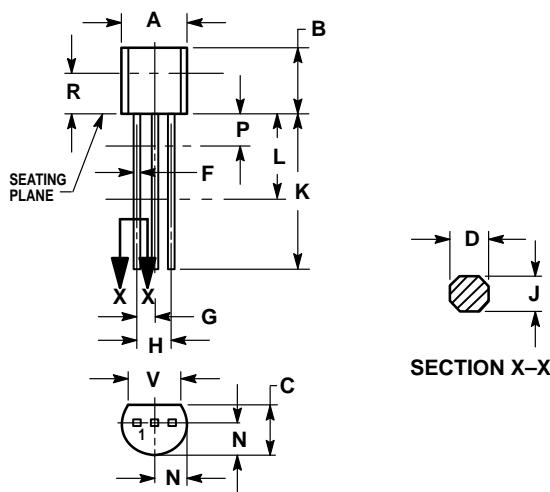


Figure 12. Temperature Coefficients

PACKAGE DIMENSIONS



NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. CONTOUR OF PACKAGE BEYOND DIMENSION R IS UNCONTROLLED.
4. DIMENSION F APPLIES BETWEEN P AND L. DIMENSION D AND J APPLY BETWEEN L AND K MINIMUM. LEAD DIMENSION IS UNCONTROLLED IN P AND BEYOND DIMENSION K MINIMUM.

DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.175	0.205	4.45	5.20
B	0.170	0.210	4.32	5.33
C	0.125	0.165	3.18	4.19
D	0.016	0.022	0.41	0.55
F	0.016	0.019	0.41	0.48
G	0.045	0.055	1.15	1.39
H	0.095	0.105	2.42	2.66
J	0.015	0.020	0.39	0.50
K	0.500	—	12.70	—
L	0.250	—	6.35	—
N	0.080	0.105	2.04	2.66
P	—	0.100	—	2.54
R	0.115	—	2.93	—
V	0.135	—	3.43	—

**CASE 029-04
(TO-226AA)
ISSUE AD**

STYLE 17:
PIN 1. COLLECTOR
2. BASE
3. Emitter

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447 or 602-303-5454

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center,
3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-81-3521-8315

MFAX: RMFAX0@email.sps.mot.com – **TOUCHTONE** 602-244-6609
INTERNET: <http://Design-NET.com>

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298



MQ-2 Semiconductor Sensor for Combustible Gas

Sensitive material of MQ-2 gas sensor is SnO_2 , which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-2 gas sensor has high sensitivity to LPG, Propane and Hydrogen, also could be used to Methane and other combustible steam, it is with low cost and suitable for different application.

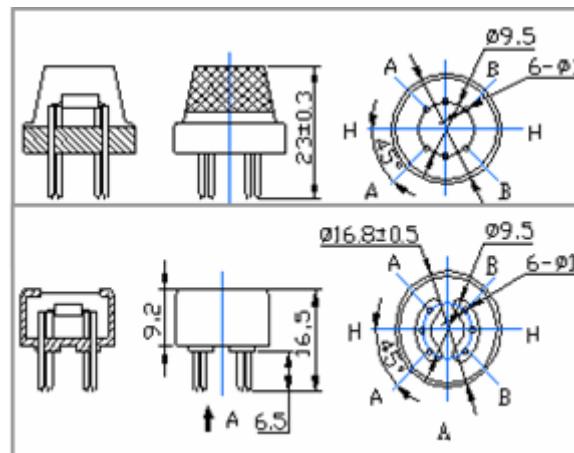
Character

- *Good sensitivity to Combustible gas in wide range
- * High sensitivity to LPG, Propane and Hydrogen
- * Long life and low cost
- * Simple drive circuit

Application

- * Domestic gas leakage detector
- * Industrial Combustible gas detector
- * Portable gas detector

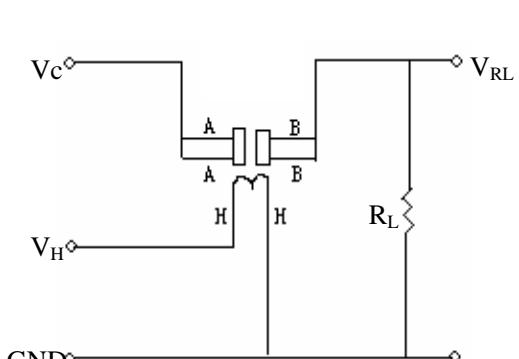
Configuration



Technical Data

Model No.			MQ-2
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Combustible gas and smoke
Concentration			300-10000ppm (Combustible gas)
Circuit	Loop Voltage	V_c	$\leq 24V$ DC
	Heater Voltage	V_H	$5.0V \pm 0.2V$ AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	$31\Omega \pm 3\Omega$ (Room Temp.)
	Heater consumption	P_H	$\leq 900\text{mW}$
	Sensing Resistance	R_s	$2K\Omega - 20K\Omega$ (in 2000ppm C_3H_8)
	Sensitivity	S	$R_s(\text{in air})/R_s(1000\text{ppm isobutane}) \geq 5$
	Slope	α	$\leq 0.6(R_{5000\text{ppm}}/R_{3000\text{ppm CH}_4})$
Condition	Tem. Humidity	$20^\circ\text{C} \pm 2^\circ\text{C}; 65\% \pm 5\% \text{RH}$	
	Standard test circuit	$V_c: 5.0V \pm 0.1V;$ $V_H: 5.0V \pm 0.1V$	
	Preheat time	Over 48 hours	

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage (V_H) and test voltage (V_C). V_H used to supply certified working temperature to the sensor, while V_C used to detect voltage (V_{RL}) on load resistance (R_L) whom is in series with sensor. The sensor has light polarity, V_C need DC power. V_C and V_H could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable R_L value is needed:
Power of Sensitivity body (P_s):
$$P_s = V_c^2 \times R_s / (R_s + R_L)^2$$

Resistance of sensor(Rs): $Rs = (V_c/VRL - 1) \times RL$

Sensitivity Characteristics

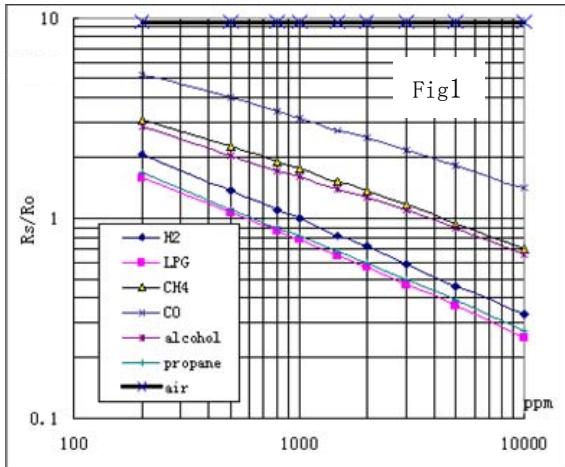


Fig.1 shows the typical sensitivity characteristics of the MQ-2, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm Hyrogen. All test are under standard test conditions.

Influence of Temperature/Humidity

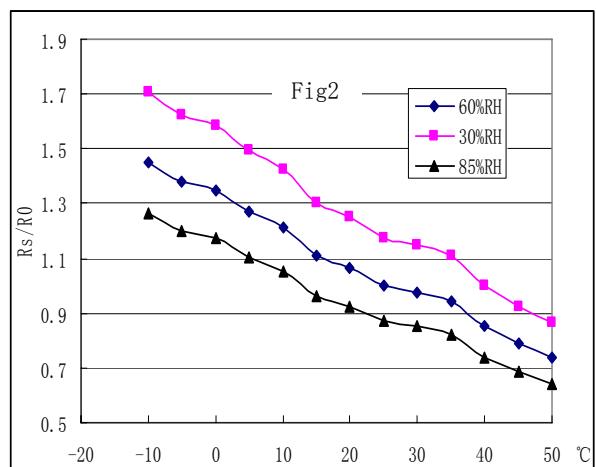
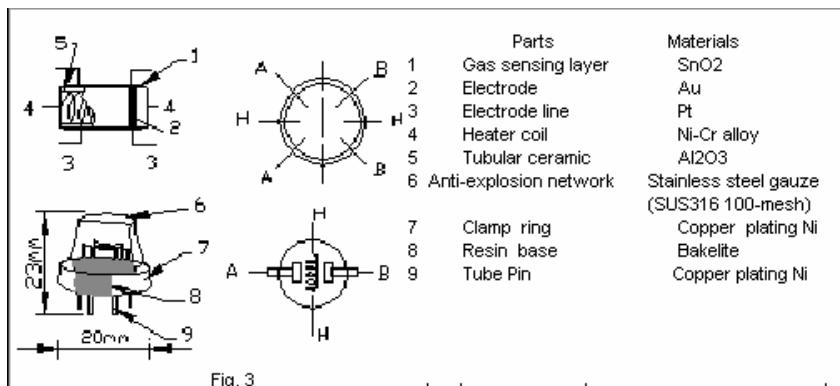


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 1000ppm Butane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Methane, 20°C/65%RH

Structure and configuration



Structure and configuration of MQ-2 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO_2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-2 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixture, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H₂Sz, SO_X, Cl₂, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

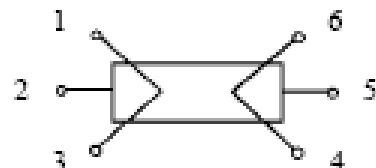
Do avoid icing on sensor's surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor's sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, it will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stability before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then reprise. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine

2.7.2 Speed: 1-2 Meter/ Minute

2.7.3 Warm-up temperature: 100±20°C

2.7.4 Welding temperature: 250±10°C

2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.

MQ-6 Semiconductor Sensor for LPG

Sensitive material of MQ-6 gas sensor is SnO_2 , which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-6 gas sensor has high sensitivity to Propane, Butane and LPG, also response to Natural gas. The sensor could be used to detect different combustible gas, especially Methane, it is with low cost and suitable for different application.

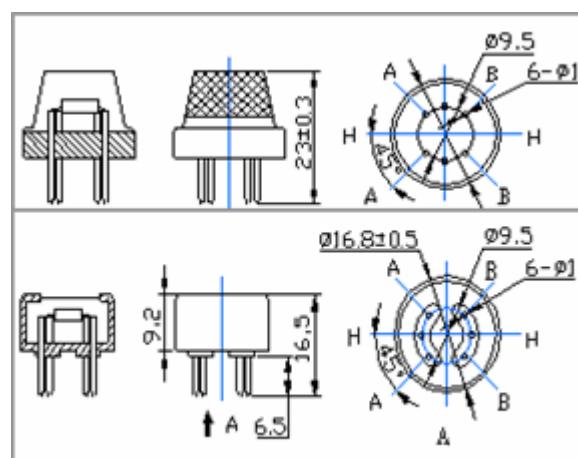
Character

- * Good sensitivity to Combustible gas in wide range
- * High sensitivity to Propane, Butane and LPG
- * Long life and low cost
- * Simple drive circuit

Application

- * Domestic gas leakage detector
- * Industrial Combustible gas detector
- * Portable gas detector

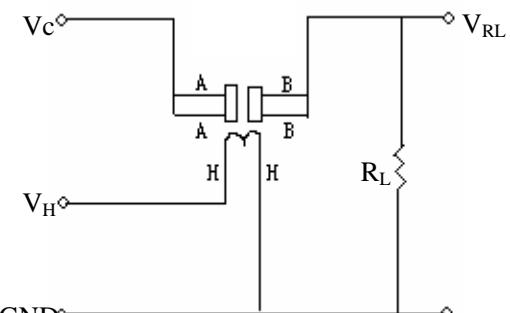
Configuration



Technical Data

Model No.			MQ-6			
Sensor Type			Semiconductor			
Standard Encapsulation			Bakelite (Black Bakelite)			
Detection Gas			Isobutane, Butane, LPG			
Concentration			300-10000ppm (Butane, Propane, LPG)			
Circuit	Loop Voltage	V_c	$\leq 24\text{V DC}$			
	Heater Voltage	V_H	$5.0\text{V}\pm 0.2\text{V AC or DC}$			
	Load Resistance	R_L	Adjustable			
Character	Heater Resistance	R_H	$31\Omega\pm 3\Omega$ (Room Tem.)			
	Heater consumption	P_H	$\leq 900\text{mW}$			
	Sensing Resistance	R_s	$2\text{K}\Omega\text{-}20\text{K}\Omega$ (in 2000ppm C_3H_8)			
	Sensitivity	S	$R_s(\text{in air})/R_s(1000\text{ppm } \text{C}_4\text{H}_{10}) \geq 5$			
	Slope	α	$\leq 0.6 (R_{2000\text{ppm}}/R_{1000\text{ppm LPG}})$			
Condition	Tem. Humidity	$20^\circ\text{C}\pm 2^\circ\text{C}; 65\%\pm 5\%\text{RH}$				
	Standard test circuit	$V_c: 5.0\text{V}\pm 0.1\text{V}; V_H: 5.0\text{V}\pm 0.1\text{V}$				
	Preheat time	Over 48 hours				

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage (V_H) and test voltage (V_c). V_H used to supply certified working temperature to the sensor, while V_c used to detect voltage (V_{RL}) on load resistance (R_L) whom is in series with sensor. The sensor has light polarity, V_c need DC power. V_c and V_H could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable R_L value is needed:
Power of Sensitivity body(P_s):

$$Ps = Vc^2 \times Rs / (Rs + RL)^2$$

Resistance of sensor (Rs): $Rs = (Vc/VRL - 1) \times RL$

Sensitivity Characteristics

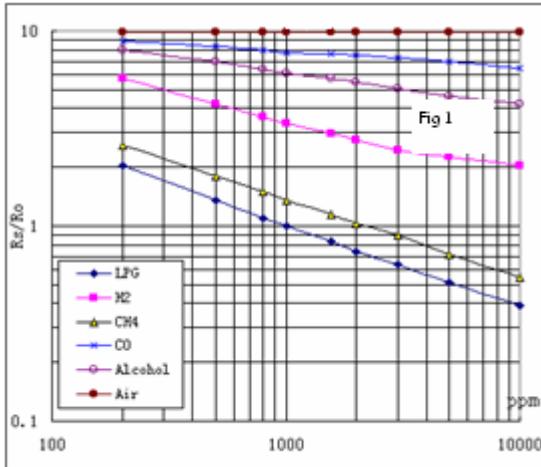


Fig.1 shows the typical sensitivity characteristics of the MQ-6, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm LPG. All test are under standard test conditions.

Influence of Temperature/Humidity

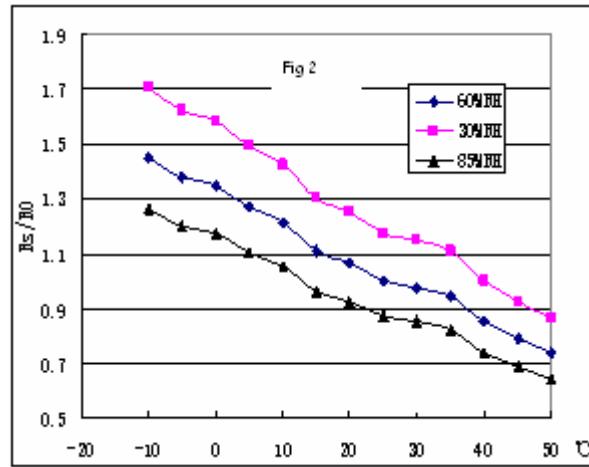
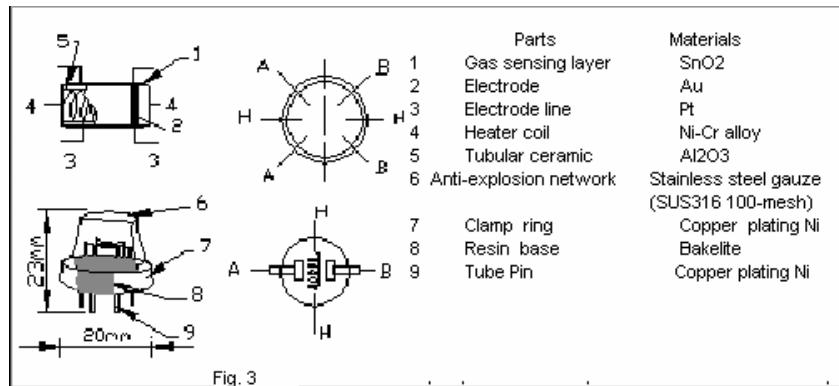


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 1000ppm Methane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Propane, 20°C/65%RH

Structure and configuration



Structure and configuration of MQ-6 gas sensor is shown as Fig. 3, sensor composed by micro Al₂O₃ ceramic tube, Tin Dioxide (SnO₂) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-4 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixture, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H₂Sz, SO_X, Cl₂, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

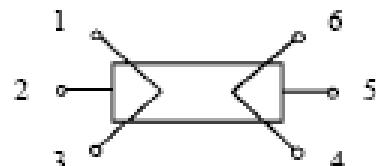
Do avoid icing on sensor's surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor's sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, it will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stability before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then reprise. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine

2.7.2 Speed: 1-2 Meter/ Minute

2.7.3 Warm-up temperature: 100±20°C

2.7.4 Welding temperature: 250±10°C

2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.



44 FARRAND STREET
BLOOMFIELD, NJ 07003
(973) 748-5089

NTE3020 thru NTE3024 Light Emitting Diode (LED)

Description:

The NTE3020 through NTE3024 LEDs offer a variety of lens effects and color availability. The Red (NTE3020) source color device is made with Gallium Arsenide Phosphide on Gallium Arsenide Red Light Emitting Diode. The High Efficiency Red (NTE3022) and Orange (NTE3023) source color devices are made with Gallium Arsenide Phosphide on Gallium Phosphide Orange Light Emitting Diode. The Green (NTE3024) source color device device is made with Gallium Phosphide on Gallium Phosphide Green Light Emitting Diode. The Yellow (NTE3021) source color device is made with Gallium Arside Phosphide on Gallium Phosphide Yellow Light Emitting Diode.

Features:

- Low Power Consumption
- High Efficiency
- IC Compatible/Low Current Requirements
- Versatile mounting on P.C. board or panel
- Reliable and Rugged

Absolute Maximum Ratings: ($T_A = +25^\circ\text{C}$ unless otherwise specified)

Power Dissipation, P_D

NTE3020	80mW
NTE3021	60mW
NTE3022	100mW
NTE3023	100mW
NTE3024	100mW

Peak Forward Current (1/10 Duty Cycle, 0.1ms Pulse Width), $I_F(\text{Peak})$

NTE3020	200mA
NTE3021	80mA
NTE3022	120mA
NTE3023	120mA
NTE3024	120mA

Absolute Maximum Ratings (Cont'd): ($T_A = +25^\circ\text{C}$ unless otherwise specified)

Continuos Forward Current, I_F

NTE3020	40mA
Derate Linearly Above 25°C	0.5mA/ $^\circ\text{C}$
NTE3021	20mA
Derate Linearly Above 25°C	0.25mA/ $^\circ\text{C}$
NTE3022	30mA
Derate Linearly Above 25°C	0.4mA/ $^\circ\text{C}$
NTE3023	30mA
Derate Linearly Above 25°C	0.4mA/ $^\circ\text{C}$
NTE3024	30mA
Derate Linearly Above 25°C	0.4mA/ $^\circ\text{C}$

Reverse Voltage, V_R 5V

Operating Temperature Range, T_A -55° to $+100^\circ\text{C}$

Storage Temperature Range, T_{stg} -55° to $+100^\circ\text{C}$

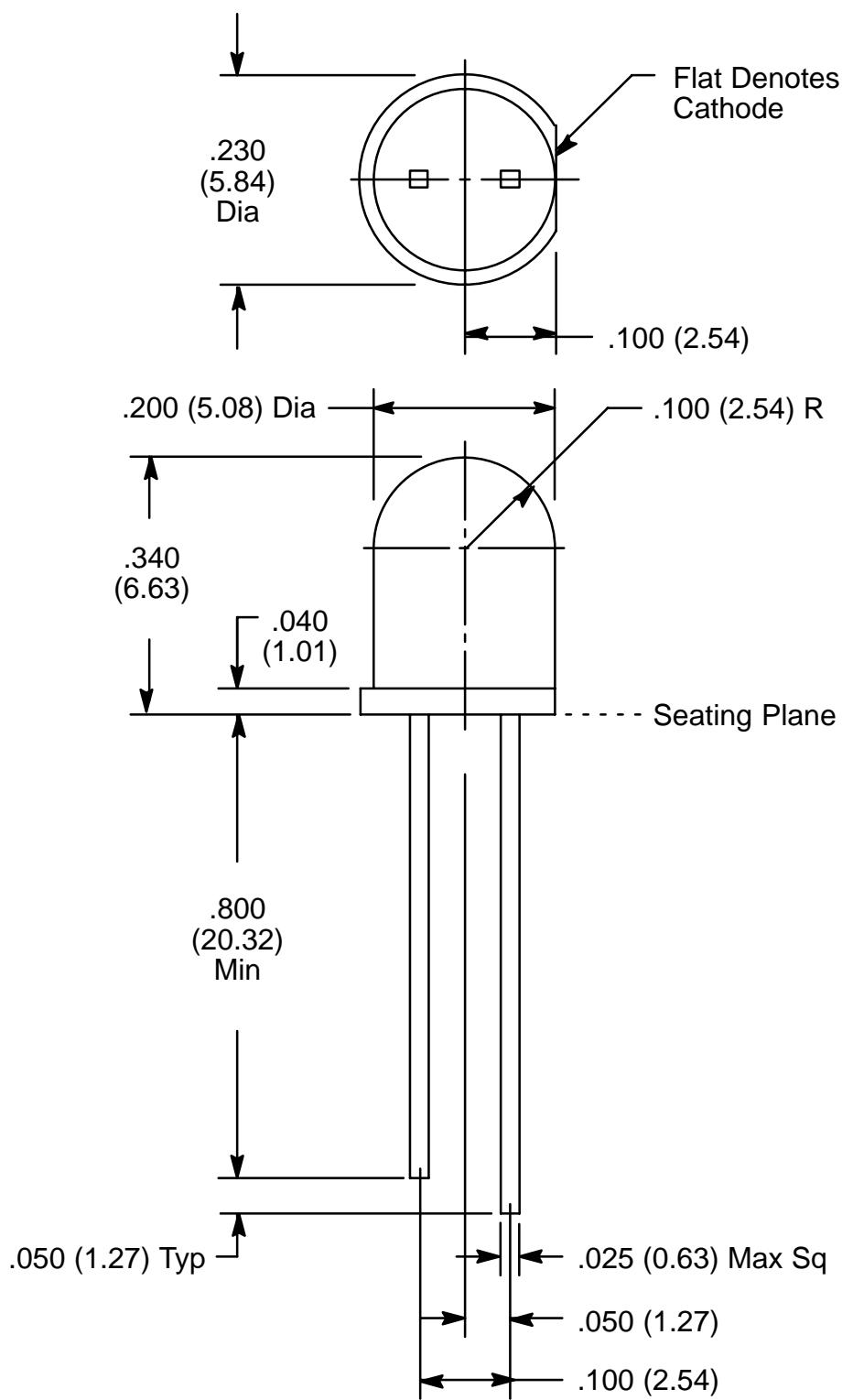
Lead Temperature (During Soldering, .063 in. (1.6mm) from Body for 5sec), T_L $+260^\circ\text{C}$

Electrical/Optical Characteristics: ($T_A = +25^\circ\text{C}$ unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Luminous Intensity NTE3020 All Other Devices	I_V	$I_F = 10\text{mA}$, Note 1	0.3 2.5	0.8 8.7	— —	mcd
Viewing Angle	$2\Theta^{1/2}$	Note 2	—	36	—	deg.
Peak Emission Wavelength NTE3020 NTE3021 NTE3022, NTE3023 NTE3024	λ_P		— — — —	655 585 635 565	— — — —	nm
Spectral Line Half Width NTE3020 NTE3021 NTE3022, NTE3023 NTE3024	$\Delta\lambda$		— — — —	24 35 40 30	— — — —	nm
Forward Voltage NTE3020 NTE3021 NTE3022 NTE3023 NTE3024	V_F	$I_F = 20\text{mA}$	— — — — —	1.7 2.1 2.0 2.0 2.1	— 2.8 — 2.8 2.8	V
Reverse Current	I_R	$V_R = 5\text{V}$	—	—	100	μA
Capacitance NTE3020 NTE3021 NTE3022, NTE3023 NTE3024	C	$V_F = 0$, $f = 1\text{MHz}$	— — — —	30 15 20 15	— — — —	pF

Note 1. Luminous intensity is measured with a light sensor and filter combination that approximates the CIE (Commission Internationale De L'Eclairage) eye-response curve.

Note 2. $\Theta^{1/2}$ is the off-axis angle at which the liminous intensity is half the axial luminous intensity.



Tolerance $\pm .010 (.254)$

This datasheet has been downloaded from:

www.DatasheetCatalog.com

Datasheets for electronic components.

TMP102 Low-Power Digital Temperature Sensor With SMBus and Two-Wire Serial Interface in SOT563

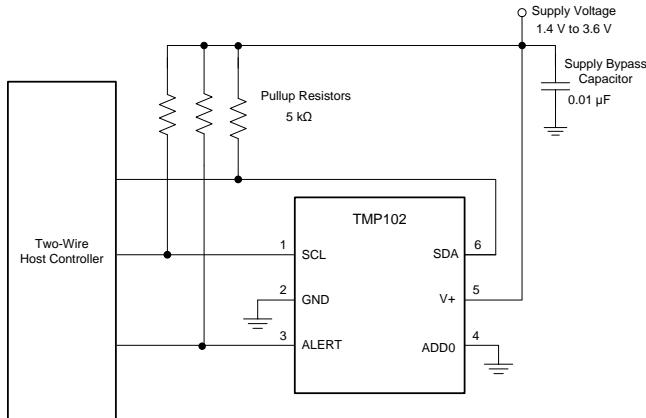
1 Features

- SOT563 Package (1.6-mm × 1.6-mm) is a 68% Smaller Footprint than SOT-23
- Accuracy Without Calibration:
 - 2.0°C (max) from -25°C to 85°C
 - 3.0°C (max) from -40°C to 125°C
- Low Quiescent Current:
 - 10-µA Active (max)
 - 1-µA Shutdown (max)
- Supply Range: 1.4 to 3.6 V
- Resolution: 12 Bits
- Digital Output: SMBus™, Two-Wire, and I²C Interface Compatibility
- NIST Traceable

2 Applications

- Portable and Battery-Powered Applications
- Power-supply Temperature Monitoring
- Computer Peripheral Thermal Protection
- Notebook Computers
- Battery Management
- Office Machines
- Thermostat Controls
- Electromechanical Device Temperatures
- General Temperature Measurements:
 - Industrial Controls
 - Test Equipment
 - Medical Instrumentations

Simplified Schematic



3 Description

The TMP102 device is a digital temperature sensor ideal for NTC/PTC thermistor replacement where high accuracy is required. The device offers an accuracy of $\pm 0.5^\circ\text{C}$ without requiring calibration or external component signal conditioning. Device temperature sensors are highly linear and do not require complex calculations or lookup tables to derive the temperature. The on-chip 12-bit ADC offers resolutions down to 0.0625°C .

The 1.6-mm × 1.6-mm SOT563 package is 68% smaller footprint than an SOT-23 package. The TMP102 device features SMBus™, two-wire and I²C interface compatibility, and allows up to four devices on one bus. The device also features an SMBus alert function. The device is specified to operate over supply voltages from 1.4 to 3.6 V with the maximum quiescent current of 10 µA over the full operating range.

The TMP102 device is ideal for extended temperature measurement in a variety of communication, computer, consumer, environmental, industrial, and instrumentation applications. The device is specified for operation over a temperature range of -40°C to 125°C.

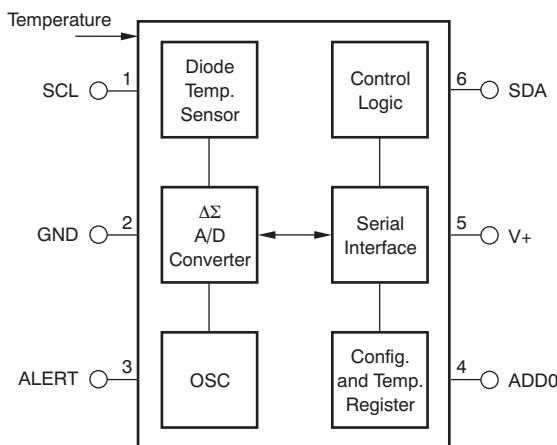
The TMP102 production units are 100% tested against sensors that are NIST-traceable and are verified with equipment that are NIST-traceable through ISO/IEC 17025 accredited calibrations.

Device Information⁽¹⁾

PART NUMBER	PACKAGE	BODY SIZE (NOM)
TMP102	SOT563 (6)	1.60 mm × 1.20 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Block Diagram



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

Table of Contents

1 Features	1	7.4 Device Functional Modes.....	13
2 Applications	1	7.5 Programming.....	14
3 Description	1	8 Application and Implementation	20
4 Revision History.....	2	8.1 Application Information.....	20
5 Pin Configuration and Functions	3	8.2 Typical Application	20
6 Specifications.....	3	9 Power Supply Recommendations	22
6.1 Absolute Maximum Ratings	3	10 Layout.....	22
6.2 Handling Ratings.....	3	10.1 Layout Guidelines	22
6.3 Recommended Operating Conditions	4	10.2 Layout Example	22
6.4 Thermal Information	4	11 Device and Documentation Support	23
6.5 Electrical Characteristics.....	4	11.1 Documentation Support	23
6.6 Timing Requirements	5	11.2 Community Resources.....	23
6.7 Typical Characteristics	6	11.3 Trademarks	23
7 Detailed Description	7	11.4 Electrostatic Discharge Caution	23
7.1 Overview	7	11.5 Glossary	23
7.2 Functional Block Diagram	7	12 Mechanical, Packaging, and Orderable Information	23
7.3 Feature Description.....	7		

4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

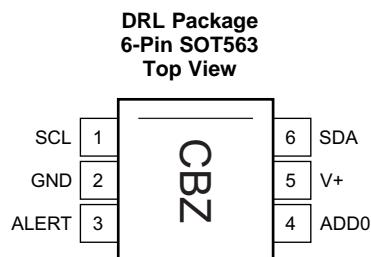
Changes from Revision E (April 2015) to Revision F	Page
• Added TI Design	1
• Added NIST Features bullet	1
• Added last paragraph of <i>Description</i> section	1

Changes from Revision D (December 2014) to Revision E	Page
• Changed the MAX value for the Supply voltage from 3.6 to 4 in the <i>Absolute Maximum Ratings</i> table	3
• Changed MIN, TYP, and MAX values for the Temperature Accuracy (temperature error) parameter	4
• Changed the frequency from 2.85 to 3.4 MHz in the POWER SUPPLY section of the <i>Electrical Characteristics</i> table	5
• Changed the Temperature Error vs Temperature graph in the <i>Typical Characteristics</i> section	6
• Changed the Temperature Error at 25°C graph in the <i>Typical Characteristics</i> section	6

Changes from Revision C (October 2012) to Revision D	Page
• Added <i>Handling Rating</i> table, <i>Feature Description</i> section, <i>Device Functional Modes</i> , <i>Application and Implementation</i> section, <i>Power Supply Recommendations</i> section, <i>Layout</i> section, <i>Device and Documentation Support</i> section, and <i>Mechanical, Packaging, and Orderable Information</i> section	3
• Changed parameters in <i>Timing Requirements</i>	5

Changes from Revision B (October 2008) to Revision C	Page
• Changed values for <i>Data Hold Time</i> parameter in <i>Timing Requirements</i>	11

5 Pin Configuration and Functions



Pin Functions

PIN		I/O	DESCRIPTION
NO.	NAME		
1	SCL	I	Serial clock. Open-drain output; requires a pullup resistor.
2	GND	—	Ground
3	ALERT	O	Overtemperature alert. Open-drain output; requires a pullup resistor.
4	ADD0	I	Address select. Connect to GND or V+
5	V+	I	Supply voltage, 1.4 V to 3.6 V
6	SDA	I/O	Serial data. Open-drain output; requires a pullup resistor.

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)⁽¹⁾

	MIN	MAX	UNIT
Supply Voltage		4	V
Input Voltage ⁽²⁾	-0.5	3.6	V
Output voltage		3.6	V
Operating temperature	-55	150	°C
Junction temperature		150	°C
Storage temperature, T _{stg}	-60	150	°C

- (1) Stresses above these ratings may cause permanent damage. Exposure to absolute maximum conditions for extended periods may degrade device reliability. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those specified is not supported.
- (2) Input voltage rating applies to all TMP102 input voltages.

6.2 Handling Ratings

		VALUE	UNIT
V _(ESD)	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001 ⁽¹⁾	±2000	V
	Charged-device model (CDM), per JEDEC specification JESD22-C101 ⁽²⁾	±1000	
	Machine model (MM)	±200	

- (1) Level listed above is the passing level per ANSI, ESDA, and JEDEC JS-001. JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
- (2) Level listed above is the passing level per EIA-JEDEC JESD22-C101. JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	NOM	MAX	UNIT
V ₊	Supply voltage	1.4	3.3	3.6	V
T _A	Operating free-air temperature	-40		125	°C

6.4 Thermal Information

	THERMAL METRIC ⁽¹⁾	TMP102	UNIT
	DRL (SOT563)		
	6 PINS		
R _{θJA}	Junction-to-ambient thermal resistance	200	°C/W
R _{θJC(top)}	Junction-to-case (top) thermal resistance	73.7	°C/W
R _{θJB}	Junction-to-board thermal resistance	34.4	°C/W
Ψ _{JT}	Junction-to-top characterization parameter	3.1	°C/W
Ψ _{JB}	Junction-to-board characterization parameter	34.2	°C/W

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, [SPRA953](#).

6.5 Electrical Characteristics

At T_A = 25°C and V_S = 1.4 to 3.6 V, unless otherwise noted.

PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
TEMPERATURE INPUT						
Range			-40		125	°C
Accuracy (temperature error)	-25°C to 85°C			±0.5	±2	°C
	-40°C to 125°C			±1	±3	
vs supply			0.2	0.5		°C/V
Resolution			0.0625			°C
DIGITAL INPUT/OUTPUT						
Input capacitance			3			pF
V _{IH}	Input logic high		0.7 × (V ₊)		3.6	V
V _{IL}	Input logic low		-0.5	0.3 × (V ₊)		V
I _{IN}	Input current	0 < V _{IN} < 3.6 V		1		µA
V _{OL}	Output logic	SDA	V ₊ > 2 V, I _{OL} = 3 mA	0	0.4	V
			V ₊ < 2 V, I _{OL} = 3 mA	0	0.2 × (V ₊)	
	ALERT		V ₊ > 2 V, I _{OL} = 3 mA	0	0.4	
			V ₊ < 2 V, I _{OL} = 3 mA	0	0.2 × (V ₊)	
Resolution			12			Bit
Conversion time			26	35		ms
Conversion modes	CR1 = 0, CR0 = 0		0.25			Conv/s
	CR1 = 0, CR0 = 1		1			
	CR1 = 1, CR0 = 0 (default)		4			
	CR1 = 1, CR0 = 1		8			
Timeout time			30	40		ms

Electrical Characteristics (continued)

At $T_A = 25^\circ\text{C}$ and $V_S = 1.4$ to 3.6 V , unless otherwise noted.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
POWER SUPPLY					
Operating supply range		+1.4		+3.6	V
I_Q Average quiescent current	Serial bus inactive, CR1 = 1, CR0 = 0 (default)		7	10	μA
	Serial bus active, SCL frequency = 400 kHz		15		
	Serial bus active, SCL frequency = 3.4 MHz		85		
I_{SD} Shutdown current	Serial bus inactive	0.5	1		μA
	Serial bus active, SCL frequency = 400 kHz		10		
	Serial bus active, SCL frequency = 3.4 MHz		80		
TEMPERATURE					
Specified range		-40		125	$^\circ\text{C}$
Operating range		-55		150	$^\circ\text{C}$

6.6 Timing Requirements

See the [Timing Diagrams](#) section for additional information.

		V+	FAST MODE			HIGH-SPEED MODE			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
$f(\text{SCL})$	SCL operating frequency	V+	0.001		0.4	0.001		2.85	MHz
$t_{(\text{BUF})}$	Bus-free time between STOP and START condition	See Figure 7	600			160			ns
$t_{(\text{HDSTA})}$	Hold time after repeated START condition. After this period, the first clock is generated.		600			160			ns
$t_{(\text{SUSTA})}$	repeated start condition setup time		600			160			ns
$t_{(\text{SUSTO})}$	STOP condition setup time		600			160			ns
$t_{(\text{HDDAT})}$	Data hold time		100	900	25	25	105		ns
$t_{(\text{SUDAT})}$	Data setup time		100			25			ns
$t_{(\text{LOW})}$	SCL-clock low period	V+, see Figure 7	1300			210			ns
$t_{(\text{HIGH})}$	SCL-clock high period	See Figure 7	600			60			ns
$t_{(\text{FD})}$	Data fall time	See Figure 7		300			80		ns
$t_{(\text{RD})}$	Data rise time	See Figure 7		300					ns
		SCLK $\leq 100 \text{ kHz}$, see Figure 7			1000				ns
$t_{(\text{FC})}$	Clock fall time	See Figure 7		300			40		ns
$t_{(\text{RC})}$	Clock rise time	See Figure 7		300			40		ns

6.7 Typical Characteristics

At $T_A = 25^\circ\text{C}$ and $V+ = 3.3\text{ V}$, unless otherwise noted.

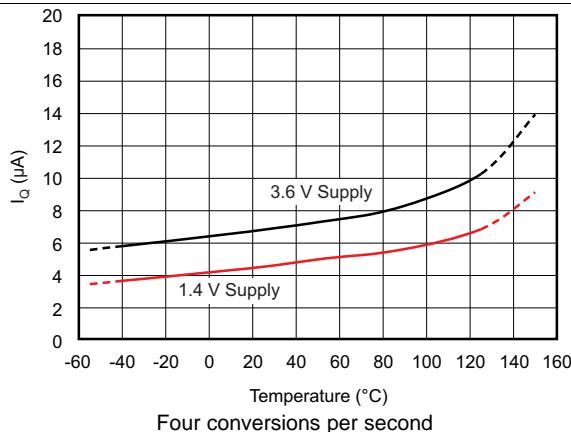


Figure 1. Average Quiescent Current vs Temperature

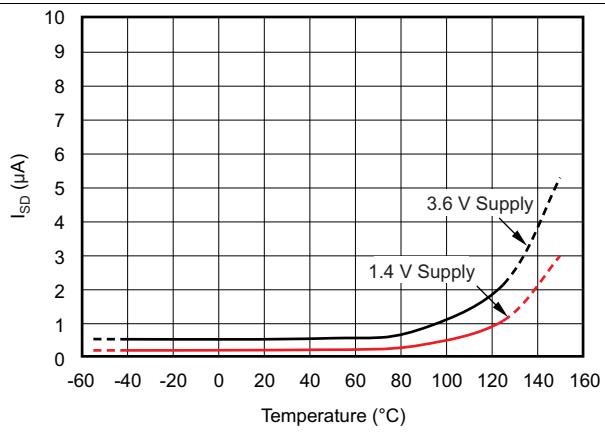


Figure 2. Shutdown Current vs Temperature

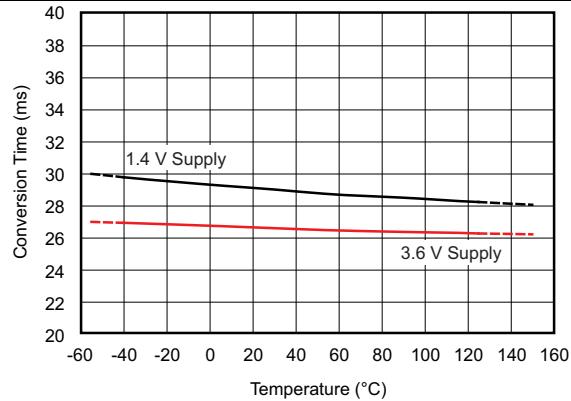


Figure 3. Conversion Time vs Temperature

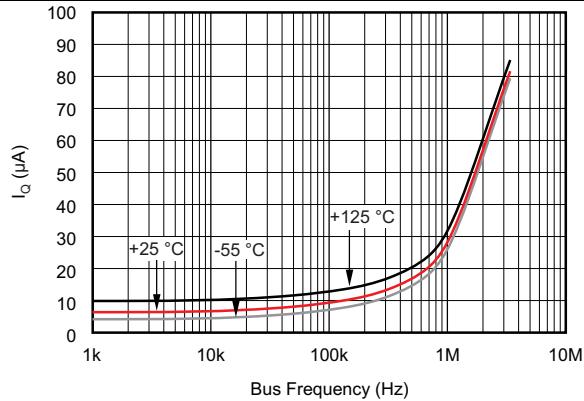


Figure 4. Quiescent Current vs Bus Frequency
(Temperature at 3.3-V Supply)

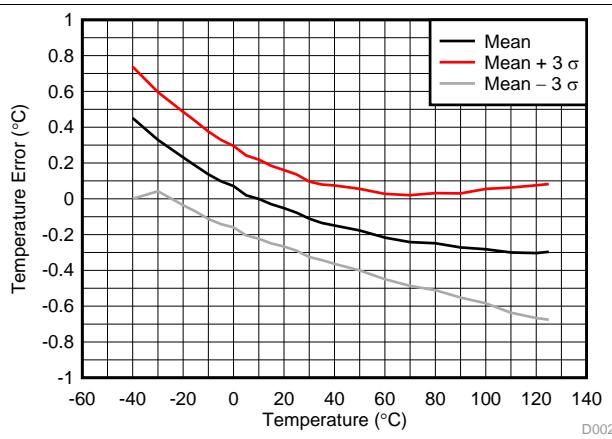


Figure 5. Temperature Error vs Temperature

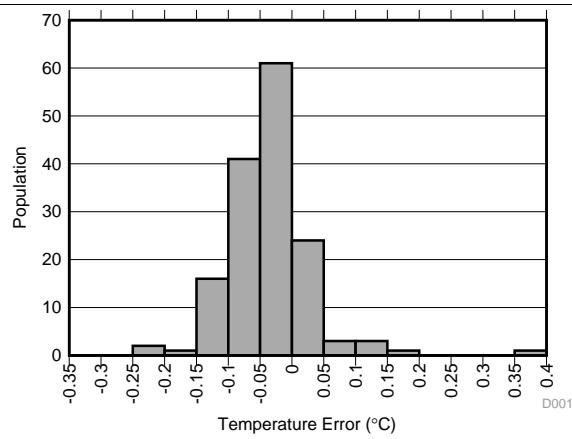


Figure 6. Temperature Error at 25°C

7 Detailed Description

7.1 Overview

The TMP102 device is a digital temperature sensor that is optimal for thermal-management and thermal-protection applications. The TMP102 device is two-wire, SMBus and I²C interface-compatible. The device is specified over an operating temperature range of -40°C to 125°C. See [Functional Block Diagram](#) for a block diagram of the TMP102 device.

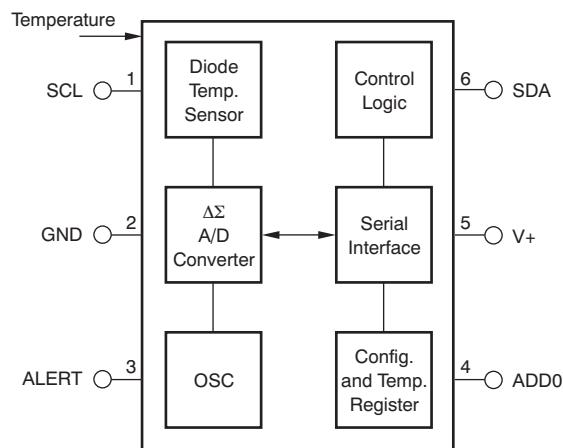
The temperature sensor in the TMP102 device is the chip itself. Thermal paths run through the package leads as well as the plastic package. The package leads provide the primary thermal path because of the lower thermal resistance of the metal.

An alternative version of the TMP102 device is available. The TMP112 device has highest accuracy, the same micro-package, and is pin-to-pin compatible.

Table 1. Advantages of TMP112 versus TMP102

DEVICE	COMPATIBLE INTERFACES	PACKAGE	SUPPLY CURRENT	SUPPLY VOLTAGE (MIN)	SUPPLY VOLTAGE (MAX)	RESOLUTION	LOCAL SENSOR ACCURACY (MAX)	SPECIFIED CALIBRATION DRIFT SLOPE
TMP112	I ² C SMBus	SOT563 1.2 x 1.6 x 0.6	10 µA	1.4 V	3.6 V	12 bit 0.0625°C	0.5°C: (0°C to 65°C) 1°C: (-40°C to 125°C)	Yes
TMP102	I ² C SMBus	SOT563 1.2 x 1.6 x 0.6	10 µA	1.4 V	3.6 V	12 bit 0.0625°C	2°C: (25°C to 85°C) 3°C: (-40°C to 125°C)	No

7.2 Functional Block Diagram



7.3 Feature Description

7.3.1 Digital Temperature Output

The digital output from each temperature measurement is stored in the read-only temperature register. The temperature register of the TMP102 device is configured as a 12-bit, read-only register (configuration register EM bit = 0, see the [Extended Mode \(EM\)](#) section), or as a 13-bit, read-only register (configuration register EM bit = 1) that stores the output of the most recent conversion. Two bytes must be read to obtain data and are listed in [Table 8](#) and [Table 9](#). Byte 1 is the most significant byte (MSB), followed by byte 2, the least significant byte (LSB). The first 12 bits (13 bits in extended mode) are used to indicate temperature. The least significant byte does not have to be read if that information is not needed. The data format for temperature is summarized in [Table 2](#) and [Table 3](#). One LSB equals 0.0625°C. Negative numbers are represented in binary two's-complement format. Following power-up or reset, the temperature register reads 0°C until the first conversion is complete. Bit D0 of byte 2 indicates normal mode (EM bit = 0) or extended mode (EM bit = 1), and can be used to distinguish between the two temperature register data formats. The unused bits in the temperature register always read 0.

Feature Description (continued)

Table 2. 12-Bit Temperature Data Format⁽¹⁾

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	HEX
128	0111 1111 1111	7FF
127.9375	0111 1111 1111	7FF
100	0110 0100 0000	640
80	0101 0000 0000	500
75	0100 1011 0000	4B0
50	0011 0010 0000	320
25	0001 1001 0000	190
0.25	0000 0000 0100	004
0	0000 0000 0000	000
-0.25	1111 1111 1100	FFC
-25	1110 0111 0000	E70
-55	1100 1001 0000	C90

(1) The resolution for the Temp ADC in Internal Temperature mode is 0.0625°C/count.

Table 2 does not list all temperatures. Use the following rules to obtain the digital data format for a given temperature or the temperature for a given digital data format.

To convert positive temperatures to a digital data format:

1. Divide the temperature by the resolution
2. Convert the result to binary code with a 12-bit, left-justified format, and MSB = 0 to denote a positive sign.

Example: (50°C) / (0.0625°C / LSB) = 800 = 320h = 0011 0010 0000

To convert a positive digital data format to temperature:

1. Convert the 12-bit, left-justified binary temperature result, with the MSB = 0 to denote a positive sign, to a decimal number.
2. Multiply the decimal number by the resolution to obtain the positive temperature.

Example: 0011 0010 0000 = 320h = 800 × (0.0625°C / LSB) = 50°C

To convert negative temperatures to a digital data format:

1. Divide the absolute value of the temperature by the resolution, and convert the result to binary code with a 12-bit, left-justified format.
2. Generate the two's complement of the result by complementing the binary number and adding one. Denote a negative number with MSB = 1.

Example: (|-25°C|) / (0.0625°C / LSB) = 400 = 190h = 0001 1001 0000

Two's complement format: 1110 0110 1111 + 1 = 1110 0111 0000

To convert a negative digital data format to temperature:

1. Generate the two's compliment of the 12-bit, left-justified binary number of the temperature result (with MSB = 1, denoting negative temperature result) by complementing the binary number and adding one. This represents the binary number of the absolute value of the temperature.
2. Convert to decimal number and multiply by the resolution to get the absolute temperature, then multiply by -1 for the negative sign.

Example: 1110 0111 0000 has two's compliment of 0001 1001 0000 = 0001 1000 1111 + 1

Convert to temperature: 0001 1001 0000 = 190h = 400; 400 × (0.0625°C / LSB) = 25°C = (|-25°C|); (|-25°C|) × (-1) = -25°C

Table 3. 13-Bit Temperature Data Format

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	HEX
150	0 1001 0110 0000	0960
128	0 1000 0000 0000	0800
127.9375	0 0111 1111 1111	07FF
100	0 0110 0100 0000	0640
80	0 0101 0000 0000	0500
75	0 0100 1011 0000	04B0
50	0 0011 0010 0000	0320
25	0 0001 1001 0000	0190
0.25	0 0000 0000 0100	0004
0	0 0000 0000 0000	0000
-0.25	1 1111 1111 1100	1FFC
-25	1 1110 0111 0000	1E70
-55	1 1100 1001 0000	1C90

7.3.2 Serial Interface

The TMP102 device operates as a slave device only on the two-wire bus and SMBus. Connections to the bus are made through the open-drain I/O lines, SDA and SCL. The SDA and SCL pins feature integrated spike suppression filters and Schmitt triggers to minimize the effects of input spikes and bus noise. The TMP102 device supports the transmission protocol for both fast (1 kHz to 400 kHz) and high-speed (1 kHz to 2.85 MHz) modes. All data bytes are transmitted MSB first.

7.3.3 Bus Overview

The device that initiates the transfer is called a *master*, and the devices controlled by the master are called *slaves*. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions.

To address a specific device, a START condition is initiated, indicated by pulling the data-line (SDA) from a high to low logic level when SCL is high. All slaves on the bus shift in the slave address byte on the rising edge of the clock, with the last bit indicating whether a read or write operation is intended. During the ninth clock pulse, the slave being addressed responds to the master by generating an acknowledge and by pulling SDA pin low.

A data transfer is then initiated and sent over eight clock pulses followed by an acknowledge bit. During the data transfer the SDA pin must remain stable when SCL is high, because any change in SDA pin when SCL pin is high is interpreted as a START signal or STOP signal.

When all data have been transferred, the master generates a STOP condition indicated by pulling SDA pin from low to high, when the SCL pin is high.

7.3.4 Serial Bus Address

To communicate with the TMP102, the master must first address slave devices via a slave address byte. The slave address byte consists of seven address bits, and a direction bit indicating the intent of executing a read or write operation.

The TMP102 features an address pin to allow up to four devices to be addressed on a single bus. [Table 4](#) describes the pin logic levels used to properly connect up to four devices.

Table 4. Address Pin and Slave Addresses

DEVICE TWO-WIRE ADDRESS	A0 PIN CONNECTION
1001000	Ground
1001001	V+
1001010	SDA
1001011	SCL

7.3.5 Writing and Reading Operation

Accessing a particular register on the TMP102 device is accomplished by writing the appropriate value to the pointer register. The value for the pointer register is the first byte transferred after the slave address byte with the R/W bit low. Every write operation to the TMP102 device requires a value for the pointer register (see [Figure 8](#)).

When reading from the TMP102 device, the last value stored in the pointer register by a write operation determines which register is read by a read operation. To change the register pointer for a read operation, a new value must be written to the pointer register. This action is accomplished by issuing a slave address byte with the R/W bit low, followed by the pointer register byte. No additional data are required. The master then generates a START condition and sends the slave address byte with the R/W bit high to initiate the read command. See [Figure 7](#) for details of this sequence. If repeated reads from the same register are desired, continually sending the Pointer Register bytes is not necessary because the TMP102 remembers the Pointer Register value until it is changed by the next write operation.

Register bytes are sent with the most significant byte first, followed by the least significant byte.

7.3.6 Slave Mode Operations

The TMP102 can operate as a slave receiver or slave transmitter. As a slave device, the TMP102 never drives the SCL line.

7.3.6.1 Slave Receiver Mode

The first byte transmitted by the master is the slave address, with the R/W bit low. The TMP102 then acknowledges reception of a valid address. The next byte transmitted by the master is the pointer register. The TMP102 then acknowledges reception of the pointer register byte. The next byte or bytes are written to the register addressed by the pointer register. The TMP102 acknowledges reception of each data byte. The master can terminate data transfer by generating a START or STOP condition..

7.3.6.2 Slave Transmitter Mode

The first byte transmitted by the master is the slave address, with the R/W bit high. The slave acknowledges reception of a valid slave address. The next byte is transmitted by the slave and is the most significant byte of the register indicated by the pointer register. The master acknowledges reception of the data byte. The next byte transmitted by the slave is the least significant byte. The master acknowledges reception of the data byte. The master terminates data transfer by generating a *Not-Acknowledge* on reception of any data byte, or generating a START or STOP condition.

7.3.7 SMBus Alert Function

The TMP102 device supports the SMBus alert function. When the TMP102 device operates in Interrupt Mode (TM = 1), the ALERT pin can be connected as an SMBus alert signal. When a master senses that an ALERT condition is present on the ALERT line, the master sends an SMBus alert command (0001 1001) to the bus. If the ALERT pin is active, the device acknowledges the SMBus alert command and responds by returning the slave address on the SDA line. The eighth bit (LSB) of the slave address byte indicates if the ALERT condition was caused by the temperature exceeding T_{HIGH} or falling below T_{LOW} . For POL = 0, the LSB is low if the temperature is greater than or equal to T_{HIGH} ; this bit is high if the temperature is less than T_{LOW} . The polarity of this bit is inverted if POL = 1. See [Figure 10](#) for details of this sequence.

If multiple devices on the bus respond to the SMBus alert command, arbitration during the slave address portion of the SMBus alert command determines which device clears the ALERT status. The device with the lowest two-wire address wins the arbitration. If the TMP102 device wins the arbitration, its ALERT pin inactivates at the completion of the SMBus alert command. If the TMP102 device loses the arbitration, its ALERT pin remains active.

7.3.8 General Call

The TMP102 device responds to a two-wire general call address (000 0000) if the eighth bit is 0. The device acknowledges the general call address and responds to commands in the second byte. If the second byte is 0000 0110, the TMP102 device internal registers are reset to power-up values. The TMP102 device does not support the general address acquire command.

7.3.9 High-Speed (HS) Mode

In order for the two-wire bus to operate at frequencies above 400 kHz, the master device must issue an HS-Mode master code (0000 1xxx) as the first byte after a START condition to switch the bus to high-speed operation. The TMP102 device does not acknowledge this byte, but switches the input filters on SDA and SCL and the output filters on SDA to operate in HS-mode, allowing transfers of up to 2.85 MHz. After the HS-Mode master code has been issued, the master transmits a two-wire slave address to initiate a data transfer operation. The bus continues to operate in HS-Mode until a STOP condition occurs on the bus. Upon receiving the STOP condition, the TMP102 device switches the input and output filters back to fast-mode operation..

7.3.10 Timeout Function

The TMP102 device resets the serial interface if SCL is held low for 30 ms (typ) between a start and stop condition. The TMP102 device releases the SDA line if the SCL pin is pulled low and waits for a start condition from the host controller. To avoid activating the time-out function, maintaining a communication speed of at least 1 kHz for SCL operating frequency is necessary..

7.3.11 Timing Diagrams

The TMP102 device is two-wire, SMBus, and I²C-interface compatible. [Figure 7](#), [Figure 8](#), [Figure 9](#), and [Figure 10](#) list the various operations on the TMP102 device. Parameters for [Figure 7](#) are defined in the [Timing Requirements](#) table. The bus definitions are defined as follows:

Acknowledge Each receiving device, when addressed, is obliged to generate an acknowledge bit. A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable low during the high period of the Acknowledge clock pulse. Setup and hold times must be taken into account. On a master receive, the termination of the data transfer can be signaled by the master generating a *not-acknowledge* (1) on the last byte that has been transmitted by the slave.

Bus Idle Both SDA and SCL lines remain high.

Data Transfer The number of data bytes transferred between a START and a STOP condition is not limited and is determined by the master device. The TMP102 device can also be used for single byte updates. To update only the MS byte, terminate the communication by issuing a START or STOP communication on the bus.

Start Data Transfer A change in the state of the SDA line, from high to low, when the SCL line is high, defines a START condition. Each data transfer is initiated with a START condition.

Stop Data Transfer A change in the state of the SDA line from low to high when the SCL line is high defines a STOP condition. Each data transfer is terminated with a repeated START or STOP condition.

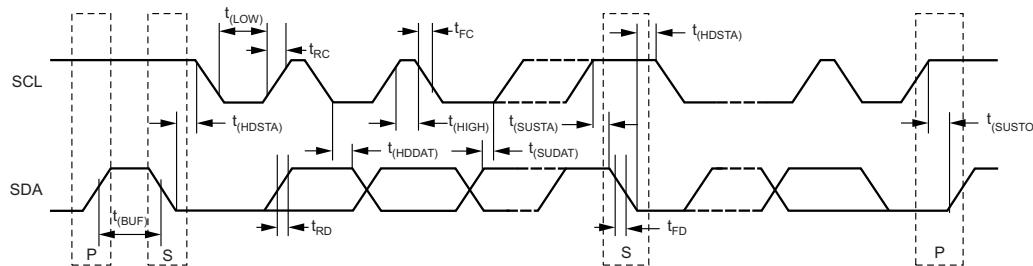
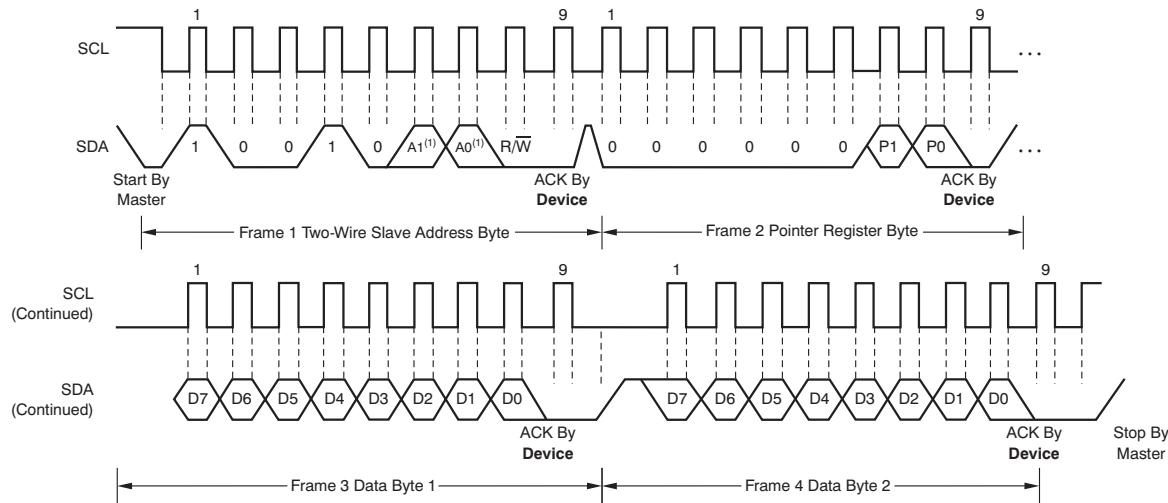
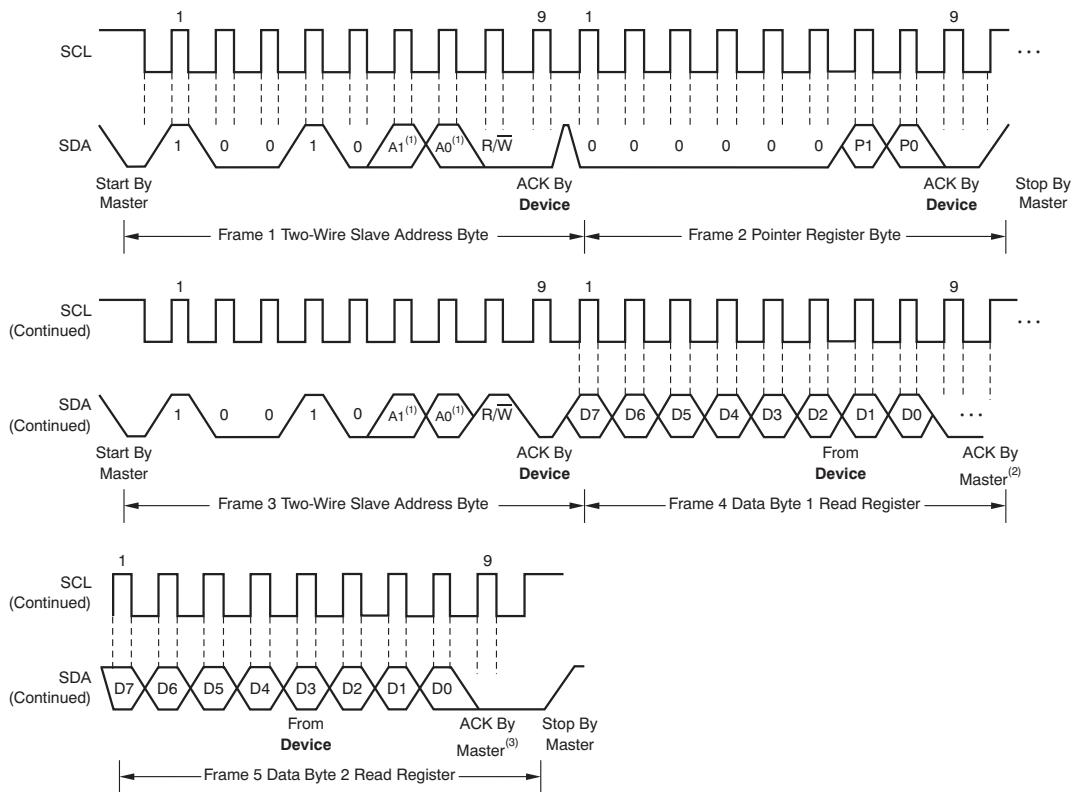


Figure 7. Two-Wire Timing Diagram



NOTE: (1) The value of A0 and A1 are determined by the ADD0 pin.

Figure 8. Two-Wire Timing Diagram for Write Word Format

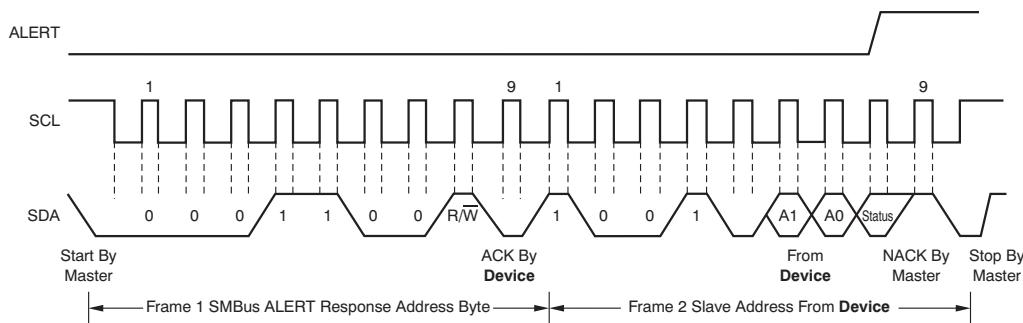


NOTE: (1) The value of A0 and A1 are determined by the ADD0 pin.

(2) Master should leave SDA high to terminate a single-byte read operation.

(3) Master should leave SDA high to terminate a two-byte read operation.

Figure 9. Two-Wire Timing Diagram for Read Word Format



NOTE: (1) The value of A0 and A1 are determined by the ADD0 pin.

Figure 10. Timing Diagram for SMBus Alert

7.4 Device Functional Modes

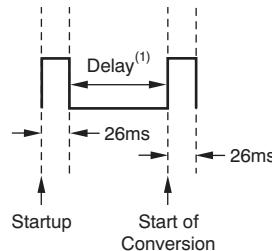
7.4.1 Continuos-Conversion Mode

The default mode of the TMP102 device is continuos conversion mode. During continuos-conversion mode, the ADC performs continuos temperature conversions and stores each results to the temperature register, overwriting the result from the previous conversion. The conversion rate bits, CR1 and CR0, configure the TMP102 device for conversion rates of 0.25 Hz, 1 Hz, 4 Hz, or 8 Hz. The default rate is 4 Hz. The TMP102 device has a typical conversion time of 26 ms. To achieve different conversion rates, the TMP102 device makes a conversion and then powers down to wait for the appropriate delay set by CR1 and CR0. [Table 5](#) lists the settings for CR1 and CR0.

Table 5. Conversion Rate Settings

CR1	CR0	CONVERSION RATE
0	0	0.25 Hz
0	1	1 Hz
1	0	4 Hz (default)
1	1	8 Hz

After power-up or general-call reset, the TMP102 immediately starts a conversion, as shown in [Figure 11](#). The first result is available after 26 ms (typical). The active quiescent current during conversion is 40 μ A (typical at +27°C). The quiescent current during delay is 2.2 μ A (typical at +27°C).



(1) Delay is set by CR1 and CR0.

Figure 11. Conversion Start

7.4.2 Extended Mode (EM)

The Extended-Mode bit configures the device for Normal mode operation (EM = 0) or Extended mode operation (EM = 1). In Normal mode, the Temperature Register and high- and low-limit registers use a 12-bit data format. Normal mode is used to make the TMP102 device compatible with the [TMP75](#) device.

Extended mode (EM = 1) allows measurement of temperatures above 128°C by configuring the Temperature Register, and high- and low-limit registers for 13-bit data format.

7.4.3 Shutdown Mode (SD)

The Shutdown-mode bit saves maximum power by shutting down all device circuitry other than the serial interface, reducing current consumption to typically less than 0.5 μ A. Shutdown mode enables when the SD bit is 1; the device shuts down when current conversion is completed. When SD is equal to 0, the device maintains a continuous conversion state.

7.4.4 One-Shot/Conversion Ready (OS)

The TMP102 device features a one-shot temperature measurement mode. When the device is in Shutdown Mode, writing a 1 to the OS bit starts a single temperature conversion. During the conversion, the OS bit reads '0'. The device returns to the shutdown state at the completion of the single conversion. After the conversion, the OS bit reads 1. This feature reduces power consumption in the TMP102 device when continuous temperature monitoring is not required.

As a result of the short conversion time, the TMP102 device achieves a higher conversion rate. A single conversion typically takes 26 ms and a read can take place in less than 20 μ s. When using One-Shot Mode, 30 or more conversions per second are possible.

7.4.5 Thermostat Mode (TM)

The thermostat-mode bit indicates to the device whether to operate in comparator mode ($TM = 0$) or Interrupt mode ($TM = 1$).

7.4.5.1 Comparator Mode ($TM = 0$)

In Comparator mode ($TM = 0$), the Alert pin is activated when the temperature equals or exceeds the value in the $T_{(HIGH)}$ register and remains active until the temperature falls below the value in the $T_{(LOW)}$ register. For more information on the comparator mode, see the [High- and Low-Limit Registers](#) section.

7.4.5.2 Interrupt Mode ($TM = 1$)

In Interrupt mode ($TM = 1$), the Alert pin is activated when the temperature exceeds $T_{(HIGH)}$ or goes below $T_{(LOW)}$ registers. The Alert pin is cleared when the host controller reads the temperature register. For more information on the interrupt mode, see the [High- and Low-Limit Registers](#) section.

7.5 Programming

7.5.1 Pointer Register

[Figure 12](#) illustrates the internal register structure of the TMP102 device. The 8-bit Pointer Register of the device is used to address a given data register. The Pointer Register uses the two least-significant bytes (LSBs) (see [Table 15](#) and [Table 16](#)) to identify which of the data registers must respond to a read or write command. [Table 6](#) identifies the bits of the Pointer Register byte. During a write command, P2 through P7 must always be '0'. [Table 7](#) describes the pointer address of the registers available in the TMP102 device. The power-up reset value of P1 and P0 is 00. By default, the TMP102 device reads the temperature on power up.

Programming (continued)

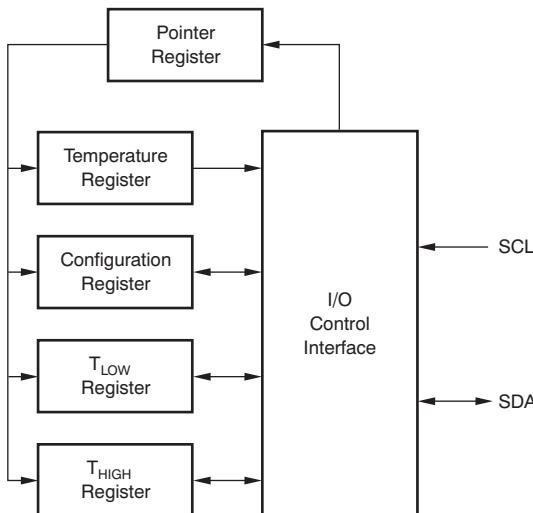


Figure 12. Internal Register Structure

Table 6. Pointer Register Byte

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Bits	

Table 7. Pointer Addresses

		REGISTER
0	0	Temperature Register (Read Only)
0	1	Configuration Register (Read/Write)
1	0	T _{LOW} Register (Read/Write)
1	1	T _{HIGH} Register (Read/Write)

7.5.2 Temperature Register

The Temperature Register of the TMP102 is configured as a 12-bit, read-only register (Configuration Register EM bit = 0, see the [Extended Mode](#) section), or as a 13-bit, read-only register (Configuration Register EM bit = 1) that stores the output of the most recent conversion. Two bytes must be read to obtain data, and are described in [Table 8](#) and [Table 9](#). Note that byte 1 is the most significant byte, followed by byte 2, the least significant byte. The first 12 bits (13 bits in Extended mode) are used to indicate temperature. The least significant byte does not have to be read if that information is not needed.

Table 8. Byte 1 of Temperature Register⁽¹⁾

D7	D6	D5	D4	D3	D2	D1	D0
T11 (T12)	T10 (T11)	T9 (T10)	T8 (T9)	T7 (T8)	T6 (T7)	T5 (T6)	T4 (T5)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 9. Byte 2 of Temperature Register⁽¹⁾

D7	D6	D5	D4	D3	D2	D1	D0
T3	T2	T1	T0	0	0	0	0
(T4)	(T3)	(T2)	(T1)	(T0)	(0)	(0)	(1)

(1) Extended mode 13-bit configuration shown in parenthesis.

7.5.3 Configuration Register

The Configuration Register is a 16-bit read/write register used to store bits that control the operational modes of the temperature sensor. Read/write operations are performed MSB first. [Table 10](#) and [Table 11](#) list the format and the power-up or reset value of the configuration register. For compatibility, [Table 10](#) and [Table 11](#) correspond to the configuration register in the TMP75 device and TMP275 device (for more information see the device data sheets, [SBOS288](#) and [SBOS363](#), respectively). All registers are updated byte by byte.

Table 10. Byte 1 of Configuration and Power-Up or Reset Format

D7	D6	D5	D4	D3	D2	D1	D0
OS	R1	R0	F1	F0	POL	TM	SD
0	1	1	0	0	0	0	0

Table 11. Byte 2 of Configuration and Power-Up or Reset Format

D7	D6	D5	D4	D3	D2	D1	D0
CR1	CR0	AL	EM	0	0	0	0
1	0	1	0	0	0	0	0

7.5.3.1 Shutdown Mode (SD)

The Shutdown-mode bit saves maximum power by shutting down all device circuitry other than the serial interface, reducing current consumption to typically less than 0.5 μ A. Shutdown mode enables when the SD bit is 1; the device shuts down when current conversion is completed. When SD is equal to 0, the device maintains a continuous conversion state.

7.5.3.2 Thermostat Mode (TM)

The Thermostat mode bit indicates to the device whether to operate in Comparator mode (TM = 0) or Interrupt mode (TM = 1). For more information on comparator and interrupt modes, see the [High- and Low-Limit Registers](#) section.

7.5.3.3 Polarity (POL)

The polarity bit allows the user to adjust the polarity of the ALERT pin output. If the POL bit is set to 0 (default), the ALERT pin becomes active low. When the POL bit is set to 1, the ALERT pin becomes active high and the state of the ALERT pin is inverted. The operation of the ALERT pin in various modes is illustrated in [Figure 13](#).

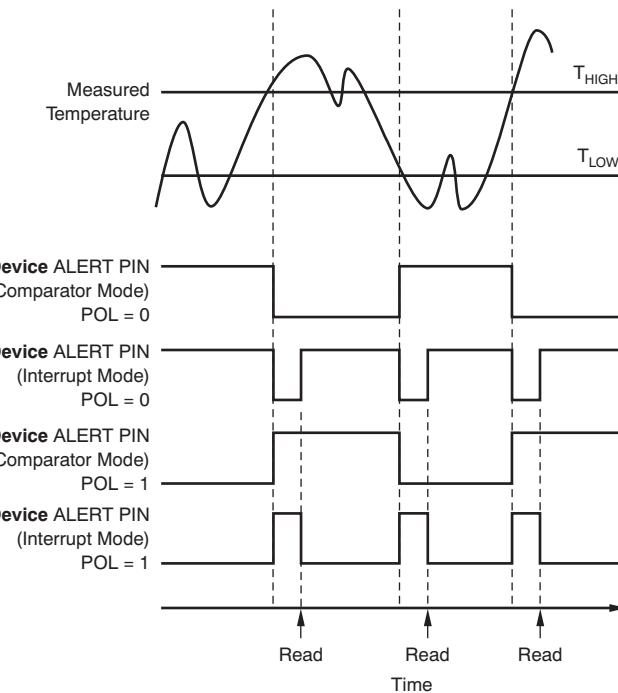


Figure 13. Output Transfer Function Diagrams

7.5.3.4 Fault Queue (F1/F0)

A fault condition exists when the measured temperature exceeds the user-defined limits set in the T_{HIGH} and T_{LOW} registers. Additionally, the number of fault conditions required to generate an alert may be programmed using the fault queue. The fault queue is provided to prevent a false alert as a result of environmental noise. The fault queue requires consecutive fault measurements in order to trigger the alert function. [Table 12](#) defines the number of measured faults that may be programmed to trigger an alert condition in the device. For T_{HIGH} and T_{LOW} register format and byte order, see the [High- and Low-Limit Registers](#) section.

Table 12. TMP102 Fault Settings

F1	F0	CONSECUTIVE FAULTS
0	0	1
0	1	2
1	0	4
1	1	6

7.5.3.5 Converter Resolution (R1/R0)

The converter resolution bits, R1 and R0, are read-only bits. The TMP102 converter resolution is set at device start-up to 11 which sets the temperature register to a 12 bit-resolution.

7.5.3.6 One-Shot (OS)

When the device is in Shutdown Mode, writing a 1 to the OS bit starts a single temperature conversion. During the conversion, the OS bit reads '0'. The device returns to the shutdown state at the completion of the single conversion. For more information on the one-shot conversion mode, see the [One-Shot/Conversion Ready \(OS\)](#) section.

7.5.3.7 EM Bit

The Extended-Mode bit configures the device for Normal Mode operation ($EM = 0$) or Extended Mode operation ($EM = 1$). In normal mode, the temperature register, high-limit register, and low-limit register use a 12-bit data format. For more information on the extended mode, see the [Extended Mode \(EM\)](#) section.

7.5.3.8 Alert (AL Bit)

The AL bit is a read-only function. Reading the AL bit provides information about the comparator mode status. The state of the POL bit inverts the polarity of data returned from the AL bit. When the POL bit equals 0, the AL bit reads as 1 until the temperature equals or exceeds $T_{(HIGH)}$ for the programmed number of consecutive faults, causing the AL bit to read as 0. The AL bit continues to read as 0 until the temperature falls below $T_{(LOW)}$ for the programmed number of consecutive faults, when it again reads as 1. The status of the TM bit does not affect the status of the AL bit.

7.5.3.9 Conversion Rate (CR)

The conversion rate bits, CR1 and CR0, configure the TMP102 device for conversion rates of 0.25 Hz, 1 Hz, 4 Hz, or 8 Hz. The default rate is 4 Hz. For more information on the conversion rate bits, see [Table 5](#).

7.5.4 High- and Low-Limit Registers

The temperature limits are stored in the $T_{(LOW)}$ and $T_{(HIGH)}$ registers in the same format as the temperature result, and their values are compared to the temperature result on every conversion. The outcome of the comparison drives the behavior of the ALERT pin, which operates as a comparator output or an interrupt, and is set by the TM bit in the configuration register.

In Comparator mode (TM = 0), the ALERT pin becomes active when the temperature equals or exceeds the value in $T_{(HIGH)}$ and generates a consecutive number of faults according to fault bits F1 and F0. The ALERT pin remains active until the temperature falls below the indicated $T_{(LOW)}$ value for the same number of faults.

In Interrupt mode (TM = 1), the ALERT pin becomes active when the temperature equals or exceeds the value in $T_{(HIGH)}$ for a consecutive number of fault conditions (as shown in [Table 5](#)). The ALERT pin remains active until a read operation of any register occurs, or the device successfully responds to the SMBus Alert Response address. The ALERT pin will also be cleared if the device is placed in Shutdown mode. When the ALERT pin is cleared, it becomes active again only when temperature falls below $T_{(LOW)}$, and remains active until cleared by a read operation of any register or a successful response to the SMBus Alert Response address. When the ALERT pin is cleared, the above cycle repeats, with the ALERT pin becoming active when the temperature equals or exceeds $T_{(HIGH)}$. The ALERT pin can also be cleared by resetting the device with the General Call Reset command. This action also clears the state of the internal registers in the device, returning the device to Comparator mode (TM = 0).

Both operational modes are represented in [Figure 13](#). [Table 13](#) through [Table 16](#) describe the format for the $T_{(HIGH)}$ and $T_{(LOW)}$ registers. Note that the most significant byte is sent first, followed by the least significant byte. Power-up reset values for $T_{(HIGH)}$ and $T_{(LOW)}$ are: $T_{(HIGH)} = +80^\circ\text{C}$ and $T_{(LOW)} = +75^\circ\text{C}$. The format of the data for $T_{(HIGH)}$ and $T_{(LOW)}$ is the same as for the Temperature Register.

Table 13. Byte 1 Temperature Register $T_{(HIGH)}$ ⁽¹⁾

D7	D6	D5	D4	D3	D2	D1	D0
H11 (H12)	H10 (H11)	H9 (H10)	H8 (H9)	H7 (H8)	H6 (H7)	H5 (H6)	H4 (H5)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 14. Byte 2 Temperature Register $T_{(HIGH)}$

D7	D6	D5	D4	D3	D2	D1	D0
H3 (H4)	H2 (H3)	H1 (H2)	H0 (H1)	0 (H0)	0 (0)	0 (0)	0 (0)

Table 15. Byte 1 Temperature Register $T_{(LOW)}$ ⁽¹⁾

D7	D6	D5	D4	D3	D2	D1	D0
L11 (L12)	L10 (L11)	L9 (L10)	L8 (L9)	L7 (L8)	L6 (L7)	L5 (L6)	L4 (L5)

(1) Extended mode 13-bit configuration shown in parenthesis.

Table 16. Byte 2 Temperature Register Low

D7	D6	D5	D4	D3	D2	D1	D0
L3 (L4)	L2 (L3)	L1 (L2)	L0 (L1)	0 (L0)	0 (0)	0 (0)	0 (0)

8 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

8.1 Application Information

The TMP102 device is used to measure the PCB temperature of the board location where the device is mounted. The programmable address options allow up to four locations on the board to be monitored on a single serial bus.

8.2 Typical Application

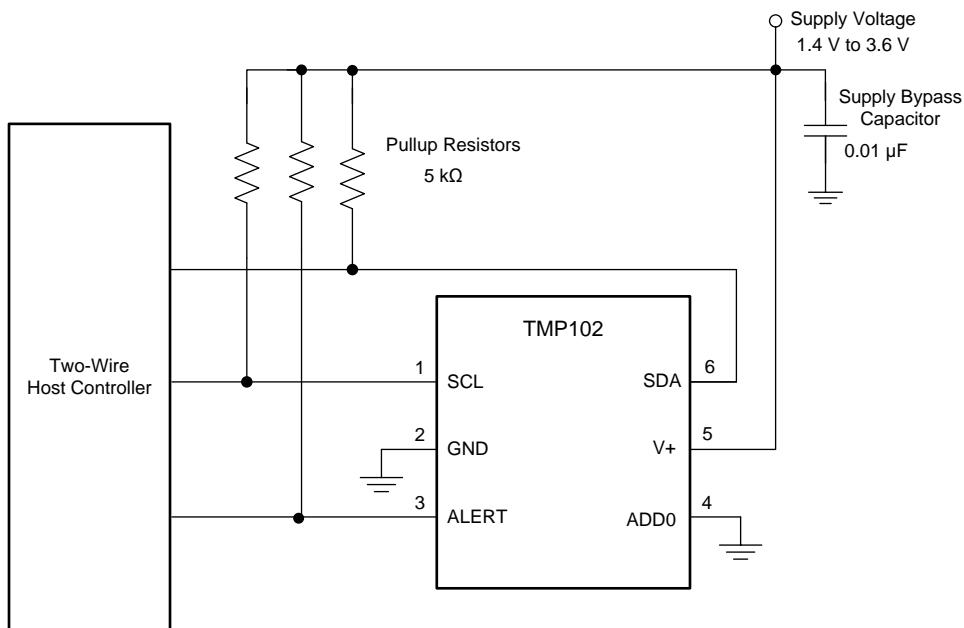


Figure 14. Typical Connections

8.2.1 Design Requirements

The TMP102 device requires pullup resistors on the SCL, SDA, and ALERT pins. The recommended value for the pullup resistors is 5-kΩ. In some applications the pullup resistor can be lower or higher than 5 kΩ but must not exceed 3 mA of current on any of those pins. A 0.01-μF bypass capacitor on the supply is recommended as shown in [Figure 14](#). The SCL and SDA lines can be pulled up to a supply that is equal to or higher than V+ through the pullup resistors. To configure one of four different addresses on the bus, connect the ADD0 pin to either the GND, V+, SDA, or SCL pin.

8.2.2 Detailed Design Procedure

Place the TMP102 device in close proximity to the heat source that must be monitored, with a proper layout for good thermal coupling. This placement ensures that temperature changes are captured within the shortest possible time interval. To maintain accuracy in applications that require air or surface temperature measurement, care must be taken to isolate the package and leads from ambient air temperature. A thermally-conductive adhesive is helpful in achieving accurate surface temperature measurement.

Typical Application (continued)

The TMP102 device is a very low-power device and generates very low noise on the supply bus. Applying an RC filter to the V+ pin of the TMP102 device can further reduce any noise that the TMP102 device might propagate to other components. $R_{(F)}$ in [Figure 15](#) must be less than 5 k Ω and $C_{(F)}$ must be greater than 10 nF.

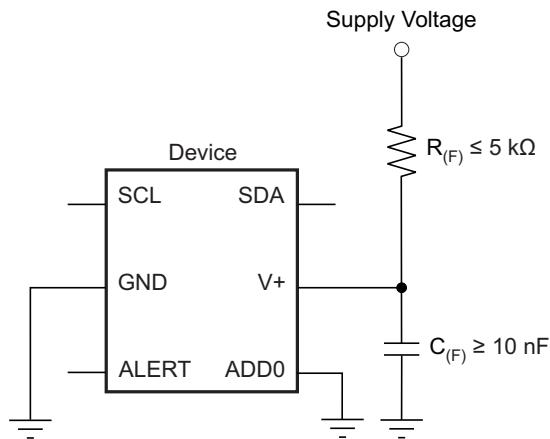


Figure 15. Noise Reduction Techniques

8.2.3 Application Curve

[Figure 16](#) shows the step response of the TMP102 device to a submersion in an oil bath of 100°C from room temperature (27°C). The time-constant, or the time for the output to reach 63% of the input step, is 0.8 s. The time-constant result depends on the printed circuit board (PCB) that the TMP102 device is mounted. For this test, the TMP102 device was soldered to a two-layer PCB that measured 0.375 inch × 0.437 inch.

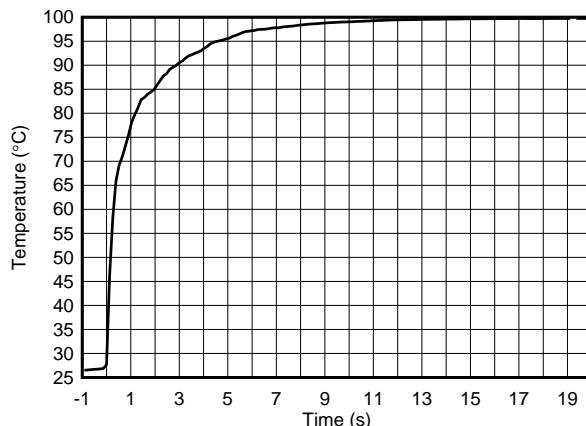


Figure 16. Temperature Step Response

9 Power Supply Recommendations

The TMP102 device operates with power supply in the range of 1.4 to 3.6 V. The device is optimized for operation at 3.3-V supply but can measure temperature accurately in the full supply range.

A power-supply bypass capacitor is required for proper operation. Place this capacitor as close as possible to the supply and ground pins of the device. A typical value for this supply bypass capacitor is 0.01 μ F. Applications with noisy or high-impedance power supplies may require additional decoupling capacitors to reject power-supply noise.

10 Layout

10.1 Layout Guidelines

Place the power-supply bypass capacitor as close as possible to the supply and ground pins. The recommended value of this bypass capacitor is 0.01 μ F. Additional decoupling capacitance can be added to compensate for noisy or high-impedance power supplies. Pull up the open-drain output pins (SDA, SCL and ALERT) through 5-k Ω pullup resistors.

10.2 Layout Example

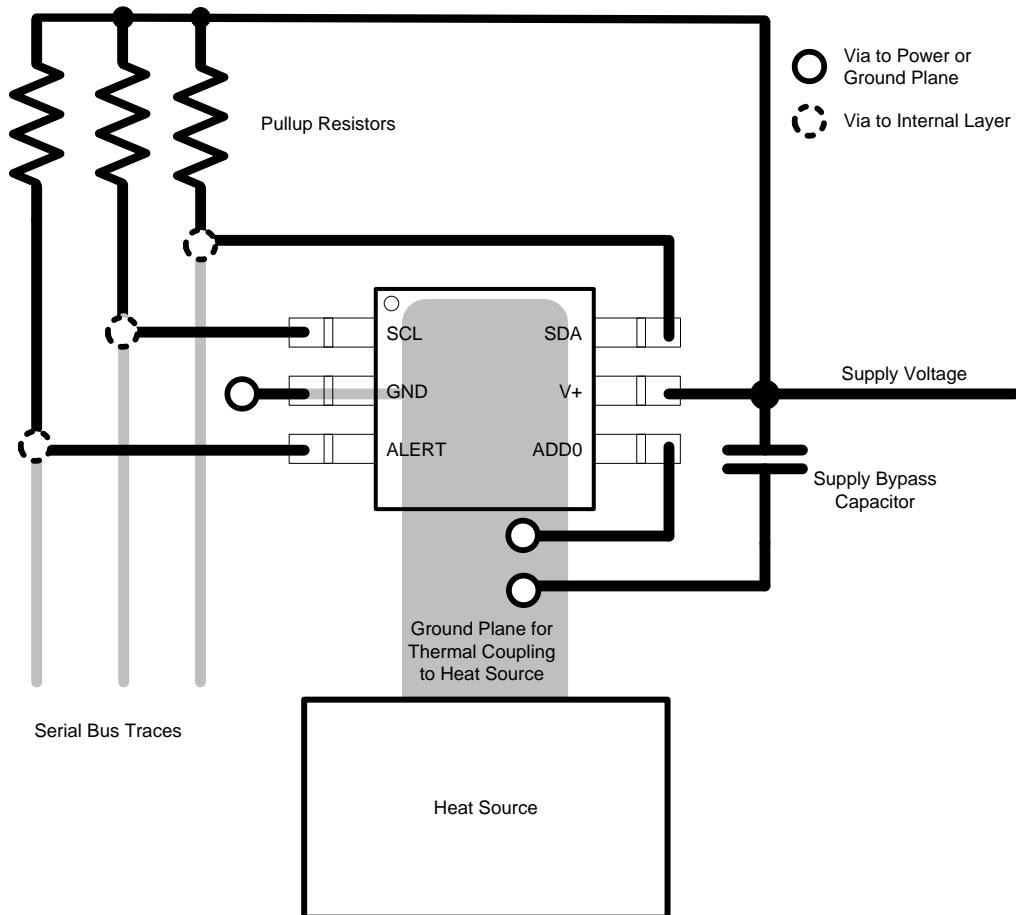


Figure 17. TMP102 Layout Example

11 Device and Documentation Support

11.1 Documentation Support

11.1.1 Related Documentation

For related documentation see the following:

- TMP175, TMP75 Data Sheet, [SBOS288](#)
- TMP275 Data Sheet, [SBOS363](#)
- *Capacitive Touch Operated Automotive LED Dome Light with Haptics Feedback* Design Guide

11.2 Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's [Terms of Use](#).

TI E2E™ Online Community *TI's Engineer-to-Engineer (E2E) Community.* Created to foster collaboration among engineers. At [e2e.ti.com](#), you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Design Support *TI's Design Support* Quickly find helpful E2E forums along with design support tools and contact information for technical support.

11.3 Trademarks

E2E is a trademark of Texas Instruments.

SMBus is a trademark of Intel, Inc.

All other trademarks are the property of their respective owners.

11.4 Electrostatic Discharge Caution



These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

11.5 Glossary

[SLYZ022](#) — *TI Glossary.*

This glossary lists and explains terms, acronyms, and definitions.

12 Mechanical, Packaging, and Orderable Information

The following pages include mechanical, packaging, and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
TMP102AIDRLR	ACTIVE	SOT	DRL	6	4000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	CBZ	Samples
TMP102AIDRLRG4	ACTIVE	SOT	DRL	6	4000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	CBZ	Samples
TMP102AIDRLT	ACTIVE	SOT	DRL	6	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	CBZ	Samples
TMP102AIDRLTG4	ACTIVE	SOT	DRL	6	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	CBZ	Samples

(1) The marketing status values are defined as follows:

ACTIVE: Product device recommended for new designs.

LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.

NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

PREVIEW: Device has been announced but is not in production. Samples may or may not be available.

OBsolete: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check <http://www.ti.com/productcontent> for the latest availability information and additional product content details.

TBD: The Pb-Free/Green conversion plan has not been defined.

Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.

Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.

Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

OTHER QUALIFIED VERSIONS OF TMP102 :

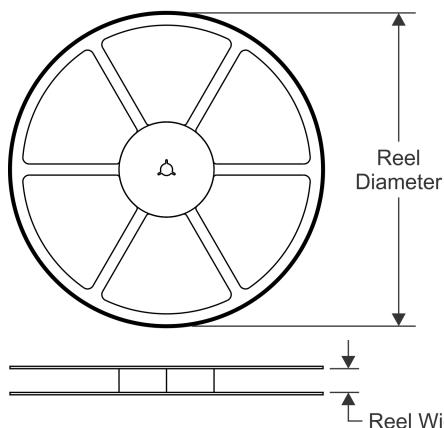
- Automotive: [TMP102-Q1](#)

NOTE: Qualified Version Definitions:

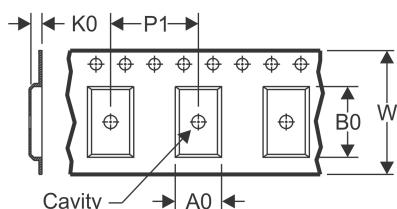
- Automotive - Q100 devices qualified for high-reliability automotive applications targeting zero defects

TAPE AND REEL INFORMATION

REEL DIMENSIONS

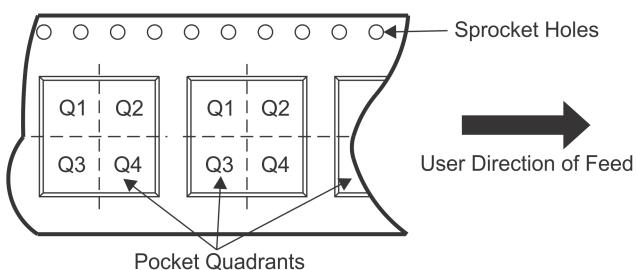


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

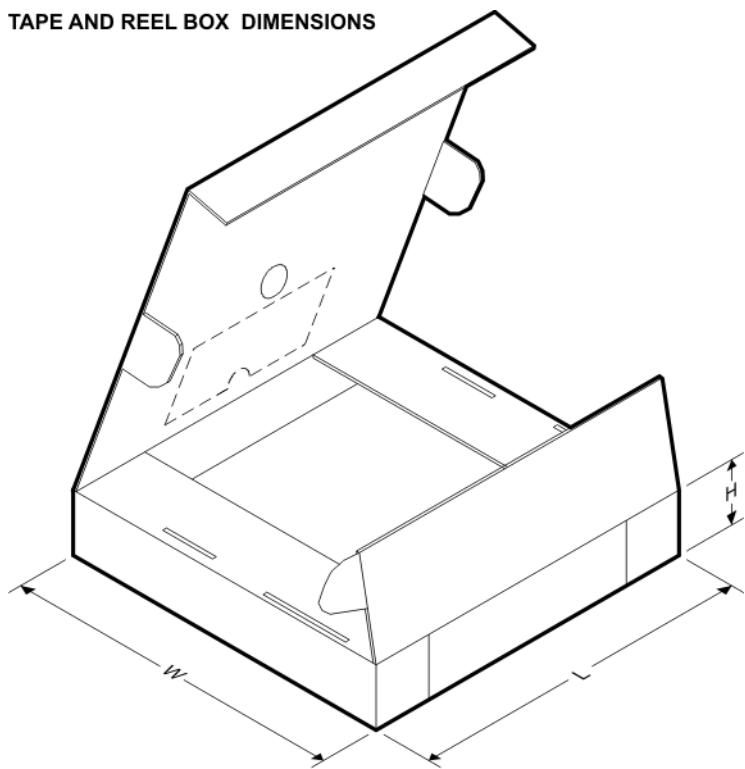
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
TMP102AIDRLR	SOT	DRL	6	4000	180.0	8.4	1.98	1.78	0.69	4.0	8.0	Q3
TMP102AIDRLT	SOT	DRL	6	250	180.0	9.5	1.78	1.78	0.69	4.0	8.0	Q3

TAPE AND REEL BOX DIMENSIONS

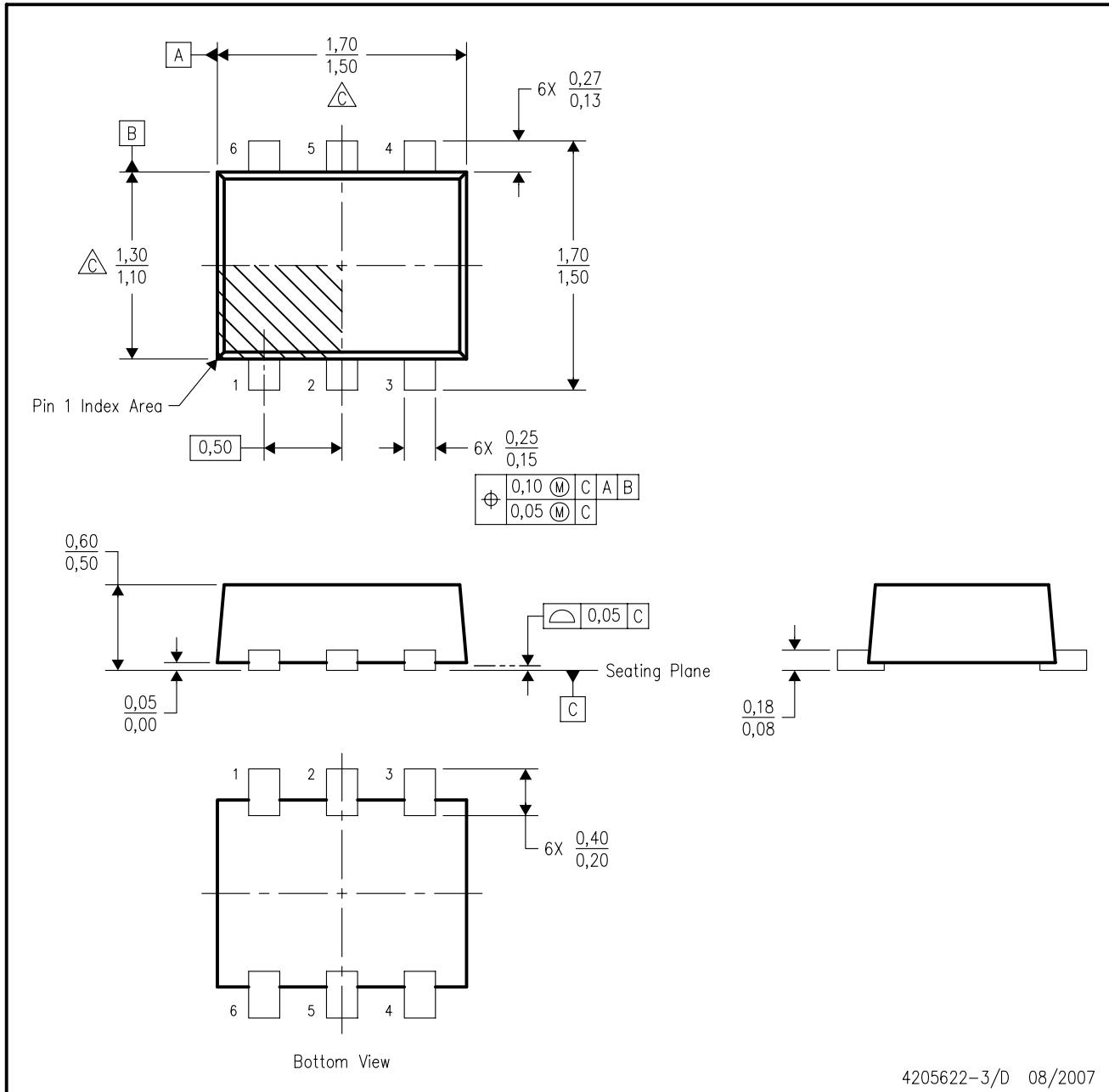


*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
TMP102AIDRLR	SOT	DRL	6	4000	202.0	201.0	28.0
TMP102AIDRLT	SOT	DRL	6	250	184.0	184.0	19.0

DRL (R-PDSO-N6)

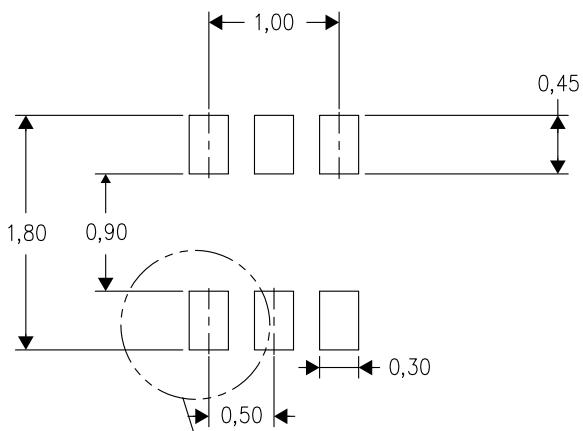
PLASTIC SMALL OUTLINE



DRL (R-PDSO-N6)

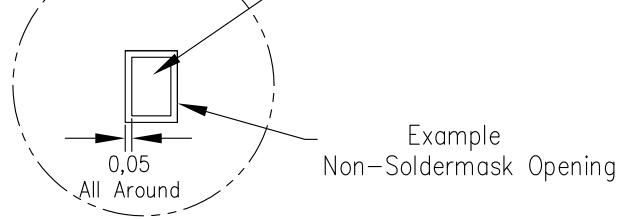
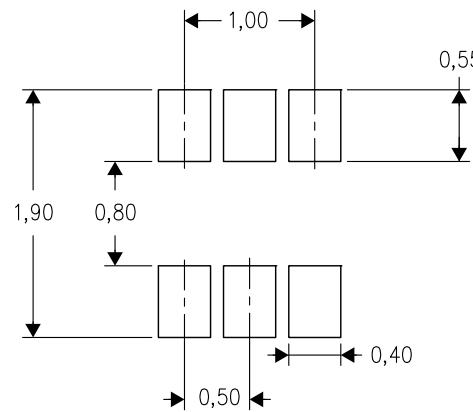
PLASTIC SMALL OUTLINE

Example Board Layout



Example Non-Soldermask Defined Pad

Example Pad Geometry

Example Stencil Design
(Note E)

4208207-3/E 06/12

- NOTES:
- All linear dimensions are in millimeters.
 - This drawing is subject to change without notice.
 - Publication IPC-7351 is recommended for alternate designs.
 - Customers should contact their board fabrication site for minimum solder mask web tolerances between signal pads.
 - Maximum stencil thickness 0,127 mm (5 mils). All linear dimensions are in millimeters.
 - Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC 7525 for stencil design considerations.
 - Side aperture dimensions over-print land for acceptable area ratio > 0.66. Customer may reduce side aperture dimensions if stencil manufacturing process allows for sufficient release at smaller opening.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products	Applications		
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	TI E2E Community	
OMAP Applications Processors	www.ti.com/omap	e2e.ti.com	
Wireless Connectivity	www.ti.com/wirelessconnectivity		