Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES

José Rodrigo de León Velásquez
Asesorado por la Inga. Ingrid Salomé Rodríguez de Loukota

Guatemala, noviembre de 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSÉ RODRIGO DE LEÓN VELÁSQUEZ
ASESORADO POR LA INGA. INGRID RODRÍGUEZ DE LOUKOTA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, NOVIEMBRE DE 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NOMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Ángel Roberto Sic García
VOCAL II	Ing. Pablo Christian De León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Jurgen Andoni Ramírez Ramírez
VOCAL V	Br. Óscar Humberto Galicia Núñez
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADORA	Inga. Ingrid Salomé Rodríguez de Loukota
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
EXAMINADOR	Ing. Armando Alonso Rivera Carrillo
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 17 de abril de 2015.

José Rodrigo de León Velásquez

Guatemala 22 de septiembre de 2017

Ingeniero
Julio Cesar Solares Peñate
Coordinador del Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Apreciable Ingeniero Solares.

Me permito dar aprobación al trabajo de graduación titulado "**Diseño de recinto controlado a bajo coste para ambientes especiales**", del señor **José Rodrigo de León Velásquez**, por considerar que cumple con los requisitos establecidos.

Por tanto, el autor de este trabajo de graduación y, yo, como su asesora, nos hacemos responsables por el contenido y conclusiones del mismo.

Sin otro particular, me es grato saludarle.

Atentamente,

Inga. Ingrid Rodríguez de Loukota
Colegiada 5,356
Asesora

Ingrid Rodríguez de Loukota
Ingeniera en Electrónica
colegiado 5356

NIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 55. 2017.

5 de OCTUBRE 2017.

Señor Director
Ing. Otto Fernando Andrino González
Escuela de Ingeniería Mecànica Elèctrica
Facultad de Ingenierìa, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES, del estudiante José Rodrigo de León Velásquez, que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Julio César Solares Peñate
Coordinador de Electrónica

sro

NIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 55. 2017.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; JOSÉ RODRIGO DE LEÓN VELÁSQUEZ titulado: DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES, procede a la autorización del mismo.

Ing. Otto Fernando Andrino González

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 16 DE OCTUBRE 2,017.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.D.542.2017

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica al trabajo de graduación titulado: **DISEÑO DE RECINTO CONTROLADO A BAJO COSTE PARA AMBIENTES ESPECIALES,** presentado por el estudiante universitario: **José Rodrigo de León Velásquez,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, noviembre, de 2017

/cc

ACTO QUE DEDICO A:

Mi madre	Emilia Velásquez, por todo el amor, esfuerzo, apoyo moral y apoyo económico recibido, además de las sabias enseñanzas que hicieron que culminara mi carrera satisfactoriamente.
Mi hermana	Claudia de León, por toda la ayuda que me ha brindado, por su amor y su apoyo durante y antes de mi carrera.
Mis sobrinos	Gabriel Cumes y Valery Cumes, por el cariño recibido, compartir muchas experiencias y estar conmigo en los buenos momentos.
Mi tío	Chris de León, por su incondicional apoyo económico e influencia durante toda mi carrera, y por sus consejos que hicieron que concluyera mis estudios a tiempo.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser mi casa de estudios y hogar durante muchos años, y también por la formación profesional.
Facultad de Ingeniería	Por ser el lugar que me dio muchos conocimientos, me ayudó a cumplir mis objetivos e influyó para ser un mejor profesional.
Laboratorio de Electrónica	Por haberme dado la oportunidad de trabajar y utilizar las instalaciones para las necesidades que requería en todo momento.
Mis amigos de la Facultad	Por haber compartido experiencias, aventuras, sacrificios y mucho esfuerzo para la realización de este proyecto.
Ing. Ingrid Rodríguez	Por su excelente asesoría en mi trabajo de graduación y su valioso tiempo dedicado.
Ing. Byron Arrivillaga	Por su apoyo y conocimientos compartidos.
Todos los ciudadanos responsables de Guatemala	Quienes, con su honrado trabajo, pagan sus impuestos. Gracias a ellos pude realizar mis estudios superiores.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES VII
LISTA DE SÍMBOLOS XI
GLOSARIO XIII
RESUMEN XVII
OBJETIVOS XIX
INTRODUCCIÓN XXI

1. FUNDAMENTOS Y TECNOLOGÍAS APLICADAS 1
1.1. Conceptos básicos 1
1.1.1. Monitorizar 1
1.1.2. Sensor 1
1.1.3. Clasificación de los sensores 1
1.1.4. Señal y lógica binaria 2
1.1.5. Lógica y niveles digitales 2
1.1.6. Nivel lógico TTL 3
1.1.7. Nivel lógico CMOS 3.3V 3
1.1.8. Circuitos convertidores de niveles 5
1.1.9. Microcontrolador 5
1.1.10. Periféricos 5
1.1.11. Periférico de comunicación UART 6
1.1.12. Periférico de comunicación I2C 6
1.1.13. Periférico de comunicación SPI 7
1.1.14. Periférico de modulación PWM 8
1.1.15. Protocolo de comunicación 9
1.1.16. Protocolo 1-Wire 10

1.1.17. Señalización 1-Wire 10
1.1.18. Protocolo serial asíncrono UART 12
1.1.19. Definición del protocolo serial UART 13
1.1.20. Lenguaje de programación 15
1.1.21. Lenguajes de bajo nivel 16
1.1.22. Lenguajes de alto nivel 16
1.1.23. Lenguajes compilados 17
1.1.24. Fases de compilación 17
1.1.25. Lenguajes interpretados 18
1.1.26. Paradigmas de programación 18
1.1.27. Programación estructurada o imperativa 18
1.1.28. Programación modular 19
1.1.29. Programación orientada a objetos 19
1.1.30. Ambiente de desarrollo integrado 20
1.1.31. Base de datos 20
1.1.32. Servidor 21
1.2. Dispositivos y tecnologías empleadas 21
1.2.1. Componentes físicos y circuitos 21
1.2.2. Raspberry Pi 21
1.2.3. Características de hardware 22
1.2.4. Fuente de alimentación 23
1.2.5. Tarjeta de memoria SD 24
1.2.6. GPIO 25
1.2.7. Microcontrolador Arduino 28
1.2.8. Arduino Mega 28
1.2.9. Sensores de bajo costo seleccionados 30
1.2.10. Sensor de humedad 30
1.2.11. Sensor de sonido 31
1.2.12. Micrófonos *electret* 32

1.2.13. Módulo detector de sonido 33
1.2.14. Sensor de vibración 34
1.2.15. Sensor de temperatura 36
1.2.16. Sensor de gases 40
1.2.17. Aplicaciones de software 41
1.2.18. Lenguaje C 41
1.2.19. Python 43
1.2.20. PHP 44
1.2.21. HTLM 44
1.2.22. Sistema operativo Raspbian 44
1.2.23. Servidor *web* Apache 44
1.2.24. Servidor de transmisión de video 45

2. CONFIGURACIÓN DE SOFTWARE 47
2.1. Preparando el ambiente de trabajo en Raspberry Pi 47
2.1.1. Instalación de un sistema operativo 47
2.1.2. Instalación y configuración de servidor LAMP 53
2.2. Transmisión de video 55

3. DISEÑO Y PARÁMETROS DE LOS DISPOSITIVOS ELECTRÓNICOS 57
3.1. Comparador de voltaje LM393 58
3.1.1. Parámetros del comparador LM393 59
3.2. Sensor de flama 61
3.3. Sensor de temperatura DS18B20 62
3.4. Módulo DHT11 65
3.4.1. Explicando el protocolo de comunicación 67
3.4.2. Parámetros y condiciones de trabajo 71
3.4.3. Recomendaciones ante influencias externas 71

3.5. Sensor de vibración....73
3.5.1. SW-520D: sensor de inclinación....73
3.5.2. Sensor de golpe....74
3.6. Sensor de sonido....74
3.6.1. Circuito para micrófono *electret*....74
3.6.2. Parámetros del micrófono *electret*....76
3.6.3. Módulo: sensor de sonido....76
3.7. Sensor de gas MQ....77
3.7.1. Sensor MQ-2....79
3.7.2. Parámetros del sensor MQ-2....83

4. DISEÑO DE UN SISTEMA DE MONITOREO....85
4.1. Implementación del diseño de hardware....86
4.1.1. Fuente de alimentación externa para alimentar los sensores....87
4.1.2. Diseño de sistema sonoro al superar el nivel de referencia....88
4.2. Implementación del diseño de software....89
4.2.1. Implementación de los niveles de referencia....89
4.2.2. Desarrollo de comandos para la comunicación bidireccional....89
4.2.3. Definición del tiempo de adquisición de muestras....93
4.2.4. Lógica de programación del microcontrolador Arduino....93
4.2.5. Lógica de programación del sistema de control Raspberry Pi....95
4.2.6. Enfoque de intercomunicación Python y PHP....96
4.2.7. Implementación de un sistema dedicado a almacenar la información....98

4.2.8. Desarrollo de una interfaz *web* 99
4.2.9. Implementación de gráficos de los valores en el tiempo 100

CONCLUSIONES 103
RECOMENDACIONES 105
BIBLIOGRAFÍA 107
APÉNDICES 111

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Nivel lógico TTL 3
2. Nivel lógico CMOS 3.3V 4
3. Conexión entre dos dispositivos UART 6
4. Esquema de conexión I^2C 7
5. Conexión SPI con múltiples esclavos 8
6. Ciclo de trabajo de una onda cuadrada 9
7. Tiempo de inicialización 1-Wire 11
8. Intervalo de escritura y lectura de información 1-Wire 12
9. Encapsulación de un paquete de transmisión UART 14
10. Paquete de transmisión UART 8N1 14
11. Diagrama de temporización UART 8N1 15
12. Raspberry Pi 2 modelo B 22
13. Marcas de clase para tarjetas de memoria SD 24
14. Comparación entre las marcas de clase de una tarjeta SD 25
15. *Pinout* Raspberry Pi v1 revisión 2 26
16. *Pinout* Raspberry Pi V2 27
17. Módulo DHT11 31
18. Micrófono *electret* 32
19. Módulo detector de sonido 33
20. Sensor SW-520D de funcionamiento 35
21. Sensor SW-520D en módulo 35
22. Módulo sensor de golpe 36

23. Gráfica de longitud de onda vs sensibilidad espectral relativa de un fototransistor 37
24. Fototransistor. Módulo IR detector de flama 38
25. Código ROM 64-*bit* para el sensor DS18B20 39
26. Módulo sensor de gas MQ-2 41
27. Fases de compilación del microcontrolador 43
28. Configuración inicial en Raspberry Pi 50
29. Sensor de gas MQ3 57
30. Sensor MQ3 en un módulo 58
31. Circuito de referencia para LM393 60
32. Curva de aplicación LM393 60
33. Circuito equivalente al módulo sensor de flama 61
34. Múltiples sensores DS18B20 conectados al bus de datos 62
35. Ejemplo de conexión del sensor DS18B20 en Arduino 63
36. Circuito típico de aplicación para el módulo DHT11 66
37. Circuito DHT11 con resistencia *pull up* 67
38. Inicio de transmisión entre microcontrolador y DHT11 68
39. Representación de cada *bit* según el módulo DHT11 69
40. Diagrama temporizado de la señales en el módulo DHT11 70
41. Trama DHT11 recibida, mostrada en un analizador lógico 70
42. Circuito simple SW-520D con protección antirebote 73
43. Circuito equivalente de sensor de golpe 74
44. Circuito *electret* básico 75
45. Circuito de amplificación *electret* 75
46. Circuito equivalente de módulo de sonido 77
47. Circuito equivalente de módulo MQ 78
48. Divisor de voltaje entre sensor MQ y R_L 79
49. Curvas características para el sensor MQ-2 80
50. Regresión potencial para el gas CO del sensor MQ-2 81

51. Dependencia de parámetros ambientales del sensor MQ-2 84
52. Propuesta de sistema de monitoreo 85
53. Diagrama de interconexión entre los circuitos 87
54. Fuente de alimentación externa para los sensores 88
55. Muestra del comando *STATUS* en un monitor serial 90
56. Muestra del comando *getPoint* en un monitor serial 91
57. Muestra del comando *setPoint_H* en un monitor serial 91
58. Muestra del comando *setPoint_T* en un monitor serial 92
59. Muestra del comando *setPoint_G* en un monitor serial 93
60. Aproximación al flujo del programa del microcontrolador 95
61. Enfoque de comunicación Python-PHP 97
62. Niveles de referencia en el sistema 98
63. Interfaz *web* simple 100
64. Gráfica en el tiempo de la temperatura en la interfaz *web* 101
65. Gráfica en el tiempo de la humedad en la interfaz *web* 101
66. Gráfica en el tiempo del gas CO en la interfaz *web* 102

TABLAS

I. Especificaciones de hardware para diferentes versiones de Raspberry Pi 22
II. Especificaciones de Arduino Mega 29
III. Condiciones de operación recomendadas para LM393 59
IV. Límites de operación SW-520D 73
V. Especificaciones del micrófono condensador *electret* 76
VI. Valores de la curva del gas CO para el sensor MQ-2 79
VII. Especificaciones de sensor MQ-2 83
VIII. Características sensitivas de sensor MQ-2 84
IX. Variables y sensores establecidos 86

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperio
MSB	Bit más significativo
LSB	Bit menos significativo
CI	Circuito integrado
AC	Corriente alterna
DC	Corriente directa
LPG	Del inglés, gas licuado de petróleo
IDE	Entorno de desarrollo integrado
GPIO	Entradas/salidas de propósito general
ºC	Grados Celsius
Hz	Hertz
I2C	Inter-integrated circuit
kHz	Kilohertz
RAM	Memoria de acceso aleatorio
m	Metro
PWM	Modulación por ancho de pulsos
nm	Nanómetro
PPM	Partes por millón
IR	Radiación infrarroja
CMOS	Semiconductor complementario de óxido metálico
SPI	*Serial peripheral interface*
OS	Sistema operativo
SoC	*System on a chip*

GND	Tierra
CPU	Unidad central de procesamiento
GPU	Unidad de procesamiento gráfico
UART	*Universal asynchronous receiver-transmitter*
USB	*Universal serial bus*
V	Voltio

GLOSARIO

Amperio Unidad de medida de intensidad de corriente eléctrica.

ASCII Codificación de caracteres basada en el alfabeto latino.

Checksum Suma de verificación utilizada para comprobar integridad de los datos.

CMOS Familia lógica utilizada en la construcción de circuitos integrados.

Código máquina Conjunto de códigos que el microcontrolador puede interpretar.

Ethernet Estándar de transmisión IEEE802.3 para redes de área local.

Filesystem El sistema de archivos se encarga de cómo almacenar y recuperar la información en el sistema operativo.

Firmware Programa que controla directamente los circuitos de los dispositivos.

Hardware	Todo elemento electrónico y mecánico de un sistema informático.
***HUB* USB**	Dispositivo que permite compartir puertos USB mediante la conexión de únicamente un bus.
I2C	Tipo de bus serial para la comunicación entre diferentes circuitos integrados.
IDE	Un programa informático que consiste básicamente en un editor de texto, un compilador y un depurador de código.
LPG	Mezcla de gases presentes en el gas natural o petróleo.
Multiplataforma	Término para los programas informáticos que pueden ejecutarse en diferentes sistemas operativos.
Pinout	Pequeño diagrama que indica la disposición y función de los pines de un circuito integrado.
Pull down	Resistencia que polariza a positivo de la fuente de alimentación DC.
Pull up	Resistencia que polariza a referencia o negativo de la fuente de alimentación DC.

Raspberry Pi	Computadora diminuta y de bajo costo desarrollada por la fundación Raspberry Pi.
Script	Es un archivo que contiene instrucciones en un lenguaje de programación para ser interpretado y ejecutado.
Sensor	Dispositivo que detecta magnitudes físicas al estar en contacto y cambia ciertos parámetros en función de la medición.
Sistema embebido	Sistemas de cómputo programados para realizar tareas específicas.
Sistema operativo	Conjunto de programas que permiten y controlan los procesos básicos y el funcionamiento de una computadora.
SoC	Integración de la mayoría de módulos de un sistema informático en un solo circuito integrado.
Software	Parte lógica de un sistema informático, como las aplicaciones informáticas.
SPI	Bus serial estándar usado para la comunicación entre circuitos integrados.
Token	Conjunto de caracteres que tienen un significado léxico.

Transductor	Dispositivo encargado de convertir un tipo de energía en otro diferente.
TTL	Familia lógica y tecnología de construcción de circuitos integrados.
Voltio	Unidad de medida de potencial eléctrico.

RESUMEN

El presente trabajo se enfoca en el desarrollo de un sistema de monitoreo dentro de un recinto cerrado donde existen factores especiales como la acumulación de gases o alta probabilidad de incendios. La propuesta de construcción es de bajo costo, el diseño se enfoca en el uso de dispositivos electrónicos que, en conjunto, puedan ser construidos parcial o totalmente siguiendo el análisis mostrado.

El primer capítulo describe los conceptos fundamentales utilizados en el desarrollo del sistema, como señales, niveles de voltaje, protocolos de comunicación y dispositivos usados. Por su parte, el segundo capítulo se enfoca en la configuración y uso de las herramientas de software necesarias para desarrollar el sistema de monitoreo.

En el tercer capítulo se analizan las curvas, los parámetros recomendados por los fabricantes, los circuitos y sensores para desarrollar el hardware del sistema de monitoreo. Y en el cuarto capítulo se desarrolla el sistema de monitoreo, se describe la lógica de control y el enfoque de intercomunicación entre los distintos componentes, como una propuesta para que otras personas puedan construirlo y modificarlo a conveniencia.

OBJETIVOS

General

Diseñar un sistema de monitoreo para un recinto controlado, a bajo costo, capaz de medir parámetros ambientales y adaptarse a la industria en Guatemala y países en vías de desarrollo.

Específicos

1. Describir el funcionamiento de los dispositivos de bajo costo a utilizar en la construcción del sistema de monitoreo económico.

2. Describir el programa que un microcontrolador ejecuta para mantener un constante monitoreo del recinto controlado.

3. Definir una interfaz que comunique los procesos generados por el microcontrolador y los procesos generados por un servidor en la red.

4. Definir una interfaz *web* para poder monitorear el sistema a través de cualquier dispositivo que tenga un navegador *web.*

INTRODUCCIÓN

En Guatemala y otros países en vías de desarrollo las industrias enfrentan un problema con la seguridad laboral, debido a que no cuentan con un plan de prevención. Las medidas anticipadas de precaución producen cierto gasto económico y la atención prestada al tema de seguridad no es del interés correspondiente. Por tal razón, el presente trabajo presenta una propuesta para desarrollar un sistema de monitoreo como guía de construcción, incluyendo análisis en los parámetros de los dispositivos, capaz de detectar ciertas variables que conllevan detectar un peligro, como un incendio, pero a un bajo costo económico.

El sistema de monitoreo se basa en la utilización de uno o más microcontroladores, un conjunto de sensores según la utilización y una computadora que comparte la información recopilada a través de la red y a través del protocolo HTTP, es decir, los datos están centralizados y pueden ser accesibles desde múltiples dispositivos. La integración e intercomunicación entre las partes del sistema se discuten a profundidad más adelante.

La elección de uno u otro tipo de sensores se debe analizar. Un sensor económico solo indicará medición aproximada y es posible que haya que calibrarlo para ser usado, mientras que un sensor de una categoría superior tiene una sensibilidad mayor, mediciones más precisas y, en muchos casos, calibrado de fábrica, pero a un precio mucho más elevado.

1. FUNDAMENTOS Y TECNOLOGÍAS APLICADAS

1.1. Conceptos básicos

1.1.1. Monitorizar

Es observar el rumbo de uno o varios parámetros para detectar posibles perturbaciones o comprender ciertos fenómenos que ayudan a controlar y supervisar una situación general o específica. Un sistema de monitorización es el conjunto de todos los componentes que poseen características propias y que en conjunto realizan una o varias tareas específicas, siempre supervisadas por seres humanos. Monitorización en un recinto se refiiere a la observación de un entorno delimitado por un área específica, denominada área de vigilancia, de donde se desea obtener información relacionada con las variables de interés.

1.1.2. Sensor

Un sensor se puede definir como todo dispositivo diseñado para captar información del exterior, como magnitudes físicas, y transformarla en una magnitud que sea capaz de cuantificarse, analizarse y manipularse. Los sensores son un tipo de transductores que proporcionan diferentes tipos de salidas, generalmente salidas eléctricas y ópticas.

1.1.3. Clasificación de los sensores

Entre los tipos de sensores más utilizados según la variable física medida están:

- Sensores mecánicos: miden fenómenos mecánicos como posición, desplazamiento, tensión, fuerza, torsión, presión y vibración.
- Sensores eléctricos: se utilizan para medir voltaje, corriente, carga y conductividad.
- Sensores magnéticos: tienen la capacidad de detectar objetos magnéticos; se utilizan para medir campo, flujo y permeabilidad magnética. Muchas veces se utilizan en lugar de algunos tipos de sensores mecánicos, como en el caso de un final de carrera, para evitar el contacto directo.
- Sensores térmicos: se utilizan para detectar cambios de temperatura, flujo y conductividad; entre los sensores térmicos se encuentran los termistores, los detectores térmicos de resistencia RDT y el termopar o termocupla.

1.1.4. Señal y lógica binaria

La lógica binaria trata con un tipo de variable, denominada variable binaria, mediante el álgebra de Boole. En este tipo de variable existen únicamente dos tipos de estados posibles que pueden ser representados con distintos símbolos, pero los más comunes son ‘1’ y ‘0’ o alto (*HIGH)* y bajo (*LOW)*. Es la base de todos los sistemas digitales y, por ende, de los microprocesadores, microcontroladores y todo sistema de cómputo en general.

1.1.5. Lógica y niveles digitales

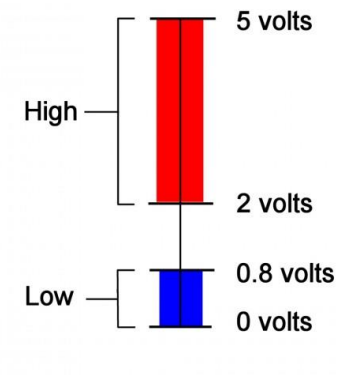
Un nivel lógico se define como el rango de estados finitos que cualquier señal digital puede tener. Existen estándares conocidos como familias lógicas

que definen los rangos de voltajes que se pueden utilizar e interpretar de manera correcta.

1.1.6. Nivel lógico TTL

La lógica transistor transistor (TTL) es un tipo de tecnología que se utiliza para la construcción de circuitos electrónicos digitales y es una familia lógica que define rangos de voltajes para trabajar con lógica binaria digital. Los voltajes que comprenden de 0 V a 0,8 V se interpretan como bajos, de 2 V a 5V se interpretan como altos, y todos aquellos valores que estén entre 0,8 V a 2 V están un rango indeterminado. Una representación gráfica se observa en la figura 1:

Figura 1. **Nivel lógico TTL**



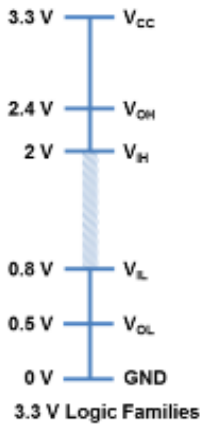
Fuente: *Nivel lógico TTL.* http://usercontent2.hubimg.com/8722637_f520.jpg. Consulta: diciembre de 2016.

1.1.7. Nivel lógico CMOS 3.3V

Los semiconductores complementarios de óxido metálico (CMOS) son una tecnología de construcción de circuitos integrados y también una familia lógica

que define una variedad de niveles de voltajes. Debido a los avances tecnológicos y a los requerimientos de bajo consumo de potencia se han creado niveles lógicos como lo es el nivel lógico CMOS a 3.3V.

Figura 2. **Nivel lógico CMOS 3.3V**



Fuente: *CMOS 3.3V.* https://cdn.sparkfun.com/. Consulta: diciembre de 2016.

- **Compatibilidad TTL CMOS 3.3V**

Cuando existen dos familias lógicas en un mismo circuito se debe tener en consideración la compatibilidad entre los dispositivos que obedecen a cada familia. Un dispositivo que funciona a 3.3V puede enviar información a un dispositivo TTL (5V), el cual sí lo interpretará de forma correcta (véanse las figuras 1 y 2), pero en sentido contrario se pueden producir problemas por la tensión más elevada que reciba el dispositivo de lógica 3.3V, es decir, al recibir este 5V podría dañarse.

1.1.8. Circuitos convertidores de niveles

Son circuitos diseñados específicamente para realizar la conversión de niveles de voltaje entre la lógica de distintos dispositivos, en general son una interfaz situada en el medio de dos mecanismos diferentes. Actualmente existe una amplia variedad de circuitos integrados en el mercado que tienen la capacidad de convertir niveles de tensión en forma bidireccional, es decir, enviar o recibir información *bit* a *bit* que se convierte al nivel de tensión entre uno y otro. Un ejemplo es el circuito integrado de Texas Instruments TXB010X, en donde la última X indica el número de *bits* que puede utilizar al mismo tiempo.

1.1.9. Microcontrolador

Es un dispositivo encapsulado dentro de un circuito integrado que posee un procesador, memoria y periféricos de entrada y salida programables, en otras palabras, es un computador con todos sus componentes dentro de un pequeño CI. Se utiliza para el control de una tarea específica, ya que después de programarse el microcontrolador solo podrá realizar la tarea para la cual ha sido diseñado. Son el núcleo de los sistemas embebidos, se utilizan en gran variedad de aplicaciones, como en los sistemas de control que gobiernan ciertos motores, sistemas de automatización, procesadores digitales de señales, máquinas de oficina e incluso en juguetes.

1.1.10. Periféricos

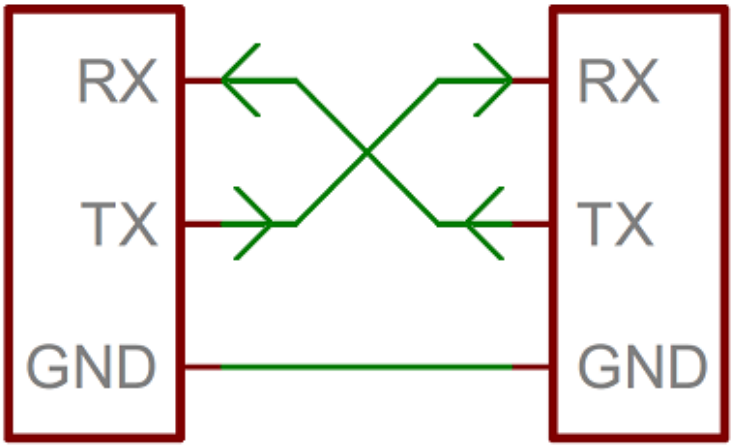
Son dispositivos hardware que están conectados a la unidad central de procesos de un microcontrolador y sirven como interfaz entre el mundo exterior y el microcontrolador. Existen diferentes tipos de periféricos, entre ellos:

periféricos de entradas y salidas, periféricos de almacenamiento y periféricos de comunicación.

1.1.11. Periférico de comunicación UART

Transmisor Receptor Asíncrono Universal (*Universal Asynchronous Receiver-Transmitter*). Es un dispositivo usado para comunicación en serie y que además cuenta con sus propios protocolos de comunicación. La conexión se realiza entre dos dispositivos, como se muestra en la figura 3:

Figura 3. **Conexión entre dos dispositivos UART**



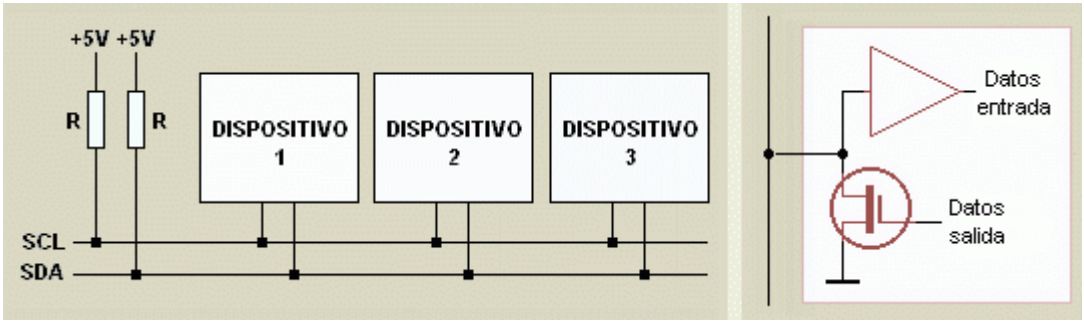
Fuente: *Conexión UART*. https://learn.sparkfun.com/tutorials/serial-communication. Consulta: diciembre de 2016.

1.1.12. Periférico de comunicación I2C

Bus diseñado originalmente por Philips Semiconductors, es un tipo de comunicación serial síncrona. I^2C (*Inter Integrated Circuit*) se usa para conectar muchos circuitos integrados y dispositivos en general. Cada dispositivo puede

actuar como maestro iniciando la transferencia de información. En su forma más básica usa tres cables SCL (*system clock*), el reloj de sincronización, SDA (*system data*), transferencia de información o línea de datos y GND o nivel de referencia común a todos los dispositivos. El esquema de conexión del I^2C se muestra en la figura 4:

Figura 4. **Esquema de conexión I^2C**



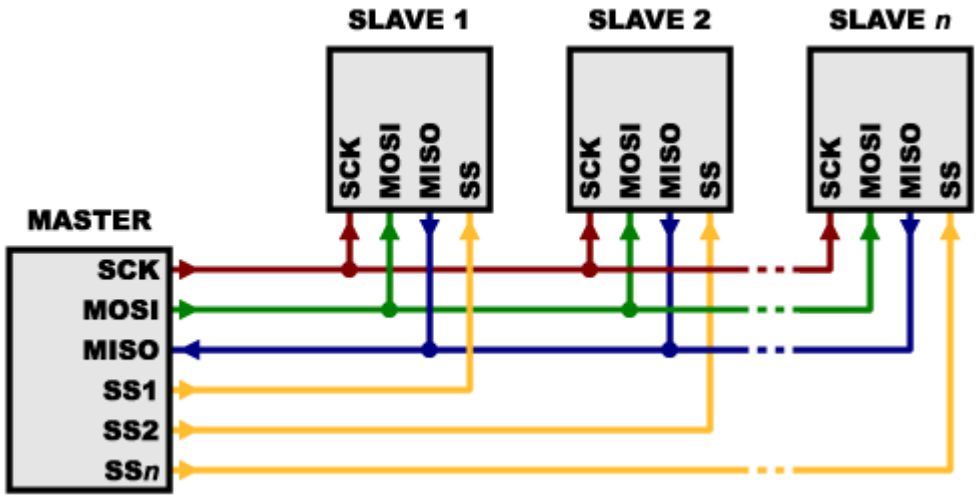
Fuente: *Esquema I^2C.* http://robots-argentina.com.ar/Comunicacion_busI2C.htm. Consulta: diciembre de 2016.

1.1.13. Periférico de comunicación SPI

La interfaz de bus SPI (*Serial Peripheral Interface*) es un estándar de comunicación serial síncrona utilizado para transmitir y recibir información utilizando su propio protocolo de comunicación. Comúnmente se usa entre microcontroladores y periféricos como sensores. La transmisión y sincronización se hace por medio de cuatro señales: SCLK (*clock*), es el reloj que sincroniza todo el sistema de comunicación, y por cada pulso de reloj que realiza se envía o recibe un *bit* de información; MOSI (*Master Output Slave Input*), es la línea que se usa cuando el maestro envía datos al esclavo; MISO (*Master Input Slave Output*), es la línea utilizada por el esclavo para enviar

información al maestro; SS/Select es una línea utilizada cuando hay múltiples esclavos, indica el momento en el cual cada esclavo se activa para recibir o transmitir información. La conexión a través de SPI se muestra en la figura 5:

Figura 5. **Conexión SPI con múltiples esclavos**



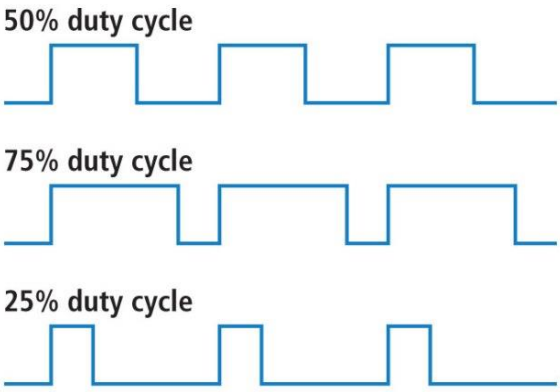
Fuente: *Conexión SPI.* https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
Consulta: diciembre de 2016.

1.1.14. Periférico de modulación PWM

La modulación por ancho de pulsos PWM (*Pulse Width Modulation)* es otro periférico disponible en la mayoría de microcontroladores, es una modulación que aplica una técnica de cambiar el ancho de pulso, ya sea para transmitir información a través de un medio de comunicación o controlar la cantidad de energía que se entrega a una carga, y por esta razón tiene amplio rango de aplicaciones donde se utiliza. El ciclo de trabajo se define como el tiempo que la señal se mantiene en estado alto respecto de un período completo. En forma matemática se expresa así: $D = \tau/T$, donde D es el ciclo de

trabajo, τ el tiempo en alto y T el período de la señal. En la figura 6 se representan tres ciclos de trabajo diferentes:

Figura 6. **Ciclo de trabajo de una onda cuadrada**



Fuente: *Ciclo de trabajo de una onda cuadrada.* https://learn.sparkfun.com/tutorials/pulse-width-modulation. Consulta: diciembre de 2016.

1.1.15. Protocolo de comunicación

Para realizar un intercambio de información entre diferentes dispositivos de un sistema, es necesario crear un conjunto de reglas formales que rijan la forma de transmitir y recibir información, entre otros factores. Todo sistema de comunicación empleado en la actualidad utiliza formatos bien definidos, inclusive estandarizados, que abarcan las partes que incluye el mensaje, los niveles de voltajes u otra magnitud eléctrica, tiempos de recepción y transmisión, detección y corrección de errores, la encriptación y seguridad, etc. Algunos protocolos de comunicación son: TCP/IP, UART, SPI e I2C.

1.1.16. Protocolo 1-Wire

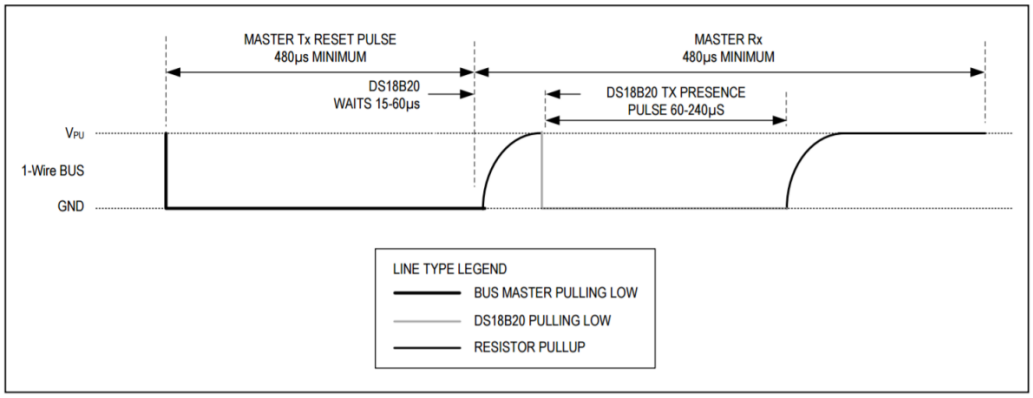
Es un protocolo de comunicación serial propietario, desarrollado originalmente por la compañía Dallas Semiconductors y posteriormente adquirida por Maxim Integrated Products. Se basa en un bus (una línea de comunicación) con un dispositivo maestro y varios esclavos. Para que el bus pueda tener un estado alto debe colocarse una resistencia *pull up*. El bus es bidireccional pero solo puede ser usado para transmitir o recibir en diferente momento, lo que lo convierte en un tipo de transmisión *half-duplex*. Toda transmisión empieza con el *bit* menos significativo LSB.

En principio, los dispositivos esclavos pueden depender únicamente de la alimentación del bus a través de la resistencia *pull up*, sin necesidad de ser conectados a V_{CC}, pero sí es necesario conectarlos a referencia GND; a ese tipo de alimentación se le llama “alimentación parásita”. La alimentación parásita debe tener ciertos cuidados debido a la energía entregada, no obstante, para evitar cualquier inconveniente, se puede utilizar la alimentación normal.

1.1.17. Señalización 1-Wire

Debe tomarse en cuenta el reinicio del bus (*resetpulse*): se trata de todas las comunicaciones iniciales con un reinicio del bus. El pulso de *reset* debe permanecer 480μs mínimo en estado bajo. Por otro lado está el pulso de presencia (*presence pulse*): luego que el maestro envía el pulso de *reset*, los esclavos tardan en responder entre 15 a 60 μs, emitiendo un pulso en bajo entre 60 a 240 μs. El diagrama temporizado se muestra en la figura 7:

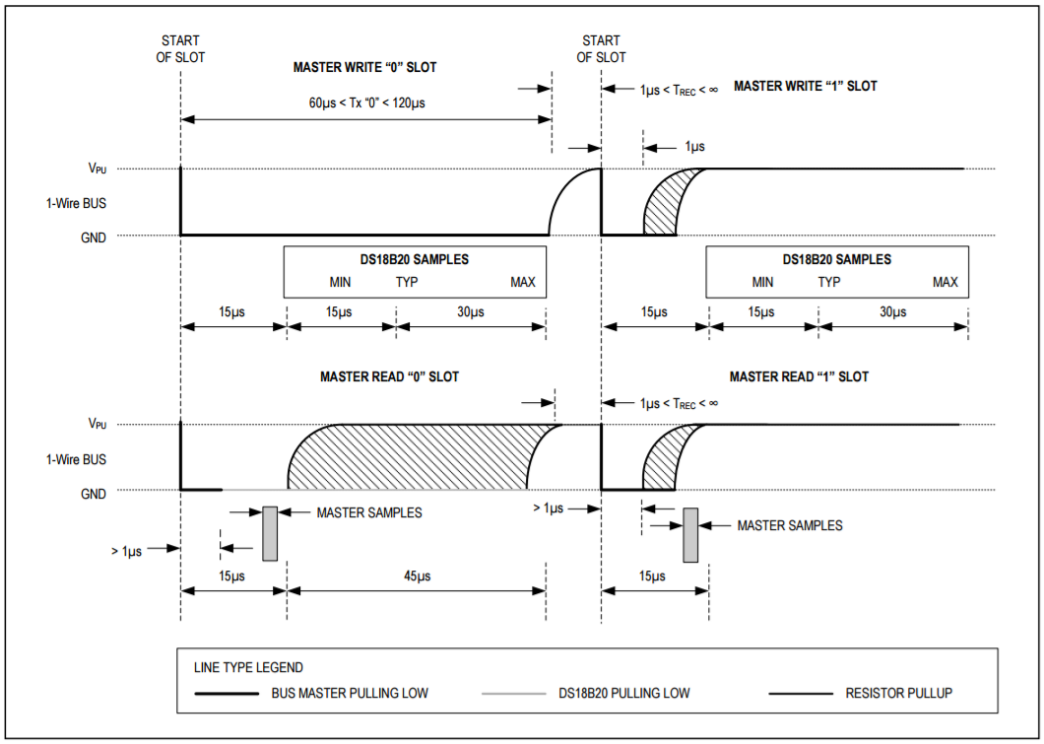
Figura 7. **Tiempo de inicialización 1-Wire**



Fuente: *Maxim integrated datasheet.*
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
Consulta: diciembre de 2016.

- Intervalos de envío de información: todos los intervalos de transmisión deben tener un mínimo de 60 μs y 1 μs de recuperación. Para transmitir un *bit* 1 en “estado lógico alto” el maestro pone el bus en bajo y luego debe liberar el bus en un lapso de 15 μs. Para generar un *bit* 0 en “estado lógico bajo” el maestro pone el bus en bajo y luego debe mantenerlo así durante el tiempo que dure el intervalo de transmisión.
- Intervalos de recepción de información: los esclavos solo pueden transmitir cuando el maestro solicita el intervalo de recepción. Todos los intervalos de recepción tienen un mínimo de 60 μs y 1 μs de recuperación. Un intervalo de recepción es iniciado por el maestro colocando el bus en bajo por un mínimo de 1 μs y luego liberándolo. El proceso de recepción y transmisión se ilustra en la figura 8:

Figura 8. **Intervalo de escritura y lectura de información 1-Wire**



Fuente: *Maxim intergrated datasheet.*
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Consulta: diciembre de 2016.

1.1.18. Protocolo serial asíncrono UART

Existen muchos protocolos seriales, entre los que destaca el protocolo de comunicación serial síncrono/asíncrono USART, que se encuentra embebido en la gran mayoría de microcontroladores. Cuando se utiliza de forma asíncrona se denomina UART (transmisor receptor asincrónico universal). UART se basa en niveles lógicos TTL a 5 V o incluso a 3.3 V. Existen más definiciones del protocolo serial, como el RS-232, con niveles de voltaje -25 V a 25 V, disponible en las computadoras antiguas con el típico conector DB-9.

1.1.19. Definición del protocolo serial UART

Dos dispositivos se intercambian información de forma *full-duplex* utilizando la misma configuración:

- *Bits* de datos: son los *bits* de información pura que emite el transmisor y son interpretados por el receptor; los *bits* de datos pueden ser entre 5 a 9 *bits* transmitidos. La configuración por defecto son 8 *bits* de información (para transmitir caracteres ASCII).
- *Bits* de sincronización: son los *bits* de inicio y parada que identifican el inicio y final de una trasmisión. El *bit* de inicio es siempre uno y los *bits* de parada pueden ser uno o dos. Para comenzar una transmisión el *bit* de inicio se representa por el bus de datos en estado inactivo y en estado alto, y a continuación puesto en estado lógico bajo por un período de transmisión. Un *bit* de parada utilizando un solo *bit* se representa colocando el bus en estado alto de forma indefinida (bus en estado inactivo) hasta el inicio de la siguiente transmisión.
- *Bit* de paridad: es un *bit* designado para realizar una comprobación de errores en la transmisión. Puede ser configurado de dos formas: la primera es par, que indica que si la cantidad de ‘1’ en los *bits* de datos es impar, el *bit* de paridad se coloca en ‘1’, en otro caso ‘0’; la segunda es impar, que indica que si la cantidad de ‘1’ en los *bits* de datos es par, el *bit* de paridad se coloca en ‘1’, en otro caso ‘0’.
- Taza de transferencia: se refiere a la velocidad de transmisión de la información medida en *bits* por segundo (b/s) o baudios, donde 1b/s = 1 *baudio* para este caso. La tasa de transferencia es

conocida mayormente por su nombre en inglés, *baudrate,* además la velocidad de trasmisión está definida con ciertas velocidades para la mayoría de microcontroladores 1200, 2400, 4800, 9600, 19200, 38400 y 115200. Por defecto es 9600 *baudios.*

Un concepto de las comunicaciones seriales es la encapsulación y se refiere a unir los *bits* de sincronización, datos, paridad, etc., para crear un paquete de transmisión. En el caso de los *bits* de datos, el orden de transmisión es a partir del *bit* menos significativo, LSB usualmente. Ver la figura 9:

Figura 9. **Encapsulación de un paquete de transmisión UART**

Fuente: *Paquete UART*. https://learn.sparkfun.com/. Consulta: diciembre de 2016.

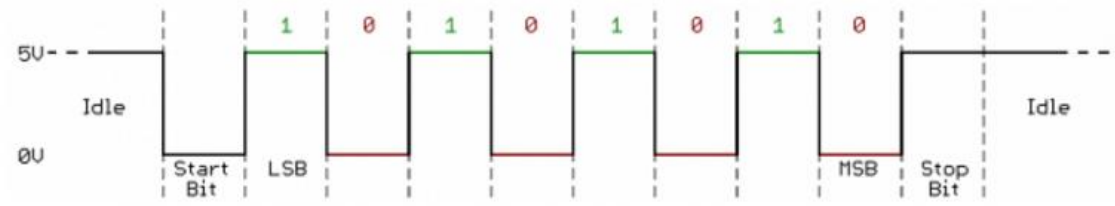
La configuración por defecto y más usada se conoce como UART 8N1, que indica: 8 *bit* de datos, sin paridad (*None*) y un *bit* de parada (*stop*). La figura 10 muestra una trama o paquete de transmisión como ejemplo:

Figura 10. **Paquete de transmisión UART 8N1**

Fuente: *Paquete UART 8N1.* https://learn.sparkfun.com/. Consulta: diciembre de 2016.

Un diagrama de temporización ayuda a entender el funcionamiento a bajo nivel de las señales TTL transmitidas, donde un estado lógico alto representa un *bit* “1” y un estado lógico bajo represita un *bit* “0”. Ver la figura 11:

Figura 11. **Diagrama de temporización UART 8N1**



Fuente: *Diagrama de temporización UART 8N1*. https://learn.sparkfun.com/
Consulta: diciembre de 2016.

Para una correcta implementación de dos dispositivos conectados por serial UART, ambos deben poseer la misma configuración, utilizar los mismos niveles de tensión y realizar una conexión cruzada, es decir, la transmisión de un dispositivo al receptor de otro dispositivo. El período de transmisión de un *bit* es inversamente proporcional a la velocidad de transmisión.

$$T_b = \frac{1}{baudios}$$

1.1.20. Lenguaje de programación

Programación es la acción de escribir código capaz de indicarle a una máquina, como la computadora, una serie de instrucciones a ejecutarse. Un lenguaje es un sistema de comunicación que posee forma, contenido y uso definido. Un lenguaje de programación es una herramienta diseñada para crear

programas capaces de controlar el comportamiento de una máquina, tanto lógico como físico, y expresar todo tipo de algoritmos de manera precisa. Consiste en un conjunto de reglas de sintaxis, semántica y símbolos que, a su vez, definen el significado de sus elementos y sus estructuras.

1.1.21. Lenguajes de bajo nivel

Un lenguaje de programación de bajo nivel es un tipo de lenguaje donde el nivel de abstracción entre el hardware y el lenguaje es reducido. Las instrucciones de un lenguaje de bajo nivel ejercen un control directo al hardware. Se utiliza este tipo de lenguajes para programar tareas de alto rendimiento en tiempo real y tareas críticas para el sistema operativo. El lenguaje de programación más representativo en esta categoría es Ensamblador (*Assembler*).

1.1.22. Lenguajes de alto nivel

Un lenguaje de programación de alto nivel es un lenguaje que se acerca a la forma del pensamiento humano, por lo tanto, se refiere a un nivel de abstracción más elevado que los lenguajes de bajo nivel. Con este tipo de lenguajes se reduce la interacción con los registros y las direcciones de memoria y pila, en cambio se orientan más a trabajar con conceptos informáticos tales como: variables, objetos, rutinas, funciones y aritmética compleja. Lenguajes de programación de alto nivel son C++, Java y Python, entre otros.

1.1.23. Lenguajes compilados

Un compilador es una herramienta utilizada para compilar, es decir, toma el código fuente escrito en un lenguaje de programación de alto nivel y lo traduce a un lenguaje que es comprensible por las computadoras, el cual se denomina código objeto (o código máquina). Un claro ejemplo de lenguaje compilado es C.

1.1.24. Fases de compilación

- Análisis léxico: en esta fase se verifica que todas las cadenas pertenezcan al lenguaje; se analiza símbolo por símbolo, indicando el *token* por cada uno de los elementos reconocidos, o bien, indicará un error en caso de no reconocer.
- Análisis de sintaxis: en esta fase se verifica la estructura de las expresiones con base en la gramática, como en el caso de una expresión matemática mal formada.
- Análisis semántico: en esta fase se dota de significado a las expresiones analizadas en las dos fases anteriores, para esto se utilizan rutinas auxiliares y se explora el árbol de sintaxis abstracta con el fin de detectar los errores, para comprobar restricciones de tipo y otras limitaciones semánticas y preparar la generación de código.
- Generación de código intermedio: se utiliza cuando el objetivo del compilador es producir código eficiente, ya que para hacerlo así se requiere una gran cantidad del análisis de las propiedades del código objetivo, facilitándose mediante el uso del código intermedio.

- Optimización de código: esta fase consiste en mejorar el código intermedio con el objetivo de reducir el número de líneas de código y provocar una ejecución más rápida del programa.
- Generación de código: aquí se genera código ensamblador o código máquina que el procesador es capaz de comprender.

1.1.25. Lenguajes interpretados

Este tipo de lenguajes no necesitan un código que deba ser compilado, ya que las instrucciones son interpretadas en tiempo real utilizando *scripts* por un intérprete, pero con resultados lentos frente a los lenguajes compilados. Entre los principales programas de este tipo se encuentran sus principales representantes: Java, Perl, Python, Ruby, PHP, Matlab y Bash, además de contar con la ventaja de que pueden ser multiplataforma y ejecutarse en dos sistemas operativos distintos.

1.1.26. Paradigmas de programación

Un paradigma de programación es una forma particular de diseñar soluciones y resolver problemas, como un modelo a seguir. El paradigma de programación más utilizado actualmente se llama programación orientada a objetos.

1.1.27. Programación estructurada o imperativa

Paradigma de programación más utilizado en el pasado y muy utilizado en la actualidad. Está orientado a mejorar la calidad, claridad y tiempo de desarrollo; utiliza subrutinas y estructuras secuenciales. Este paradigma es soportado por lenguajes como C, Basic y Pascal. El teorema de programa

estructurado indica que la programación estructurada puede ser implementada con tres estructuras de control únicamente:

- Secuencia: ejecutar una instrucción tras otra en orden específico.
- Selección: seleccionar una instrucción de un conjunto ejecutable según el valor de una condición *booleana.*
- Iteración: ejecución de un conjunto de instrucciones mientras una variable *booleana* sea verdadera. Se le conoce mejor con el nombre de ciclos o bucles.

1.1.28. Programación modular

Es una evolución de la programación estructurada y se basa en el hecho de resolver problemas dividiendo en módulos o subprogramas. Logra solucionar problemas de programación más grandes y complejos que la programación estructurada. En la práctica se pueden utilizar funciones o procedimientos para crear módulos, sin confundir esto con la programación modular, ya que en general significa utilizar un subprograma para resolver un subproblema. Entre los lenguajes que soportan programación modular están Ruby, Java y Perl.

1.1.29. Programación orientada a objetos

La programación orientada a objetos o POO es una forma de programar que se aproxima a la manera en que se expresan las cosas en el mundo real. El enfoque de este paradigma se basa en darle más importancia a los datos. La programación orientada a objetos se realiza mediante la organización del código mediante el conjunto de datos con sus funciones; los objetos incluyen datos y sus funciones.

1.1.30. Ambiente de desarrollo integrado

Del inglés *Integrated Development Environment* (IDE), es una herramienta que consiste básicamente en un editor de código fuente, un compilador y un depurador. Puede poseer también un constructor de interfaz gráfica y un autocompletado de código. Proporciona servicios integrales para facilitar el desarrollo de aplicaciones; Netbeans y Eclipse, son un claro ejemplo de IDE. Un IDE es capaz de soportar varios lenguajes de programación; es en sí un entorno de desarrollo completo. Componentes principales:

- Editor de texto
- Compilador
- Depurador
- Intérprete
- Autocompletado de código
- Soporte para diversos lenguajes de programación
- Herramientas de automatización

1.1.31. Base de datos

Es un sistema diseñado para almacenar una gran cantidad de datos de forma organizada, que luego pueden ser encontrados con facilidad y rapidez. Las bases de datos son administrados por programas llamados gestores de bases de datos, que tienen el control total del sistema, a estos se les puede pedir almacenar información relacionada o realizar consulta de información requerida.

1.1.32. Servidor

Es un tipo de ordenador que ejecuta ciertas aplicaciones software que suministran algún tipo de servicio a otros dispositivos como las computadoras. A cualquier dispositivo que utiliza los servicios de un servidor se le llama cliente. Cualquier computadora se puede utilizar como servidor en principio (cumpliendo los requisitos mínimos especificados), pero es deseable en la mayoría de los casos que posean altos requerimientos de hardware, puesto que va a suministrar servicios a muchos clientes.

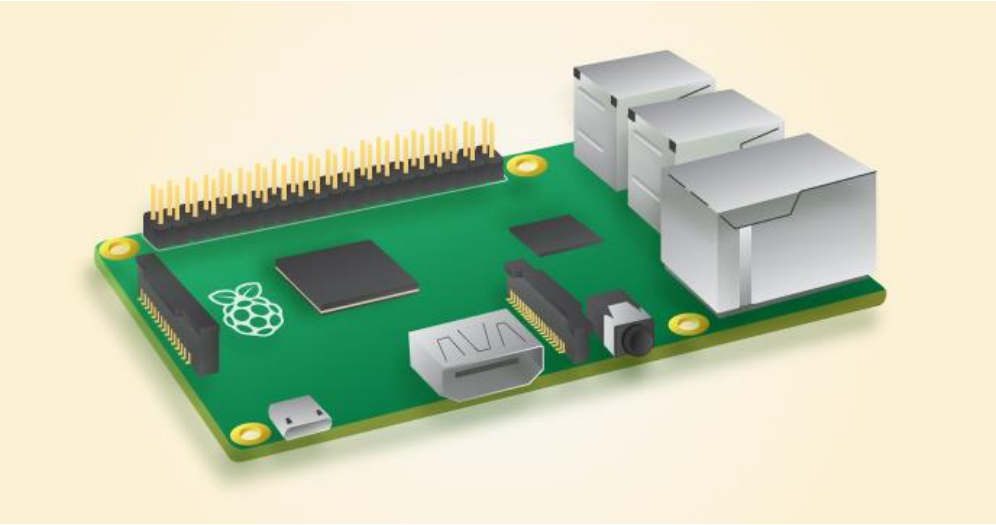
1.2. Dispositivos y tecnologías empleadas

1.2.1. Componentes físicos y circuitos

1.2.2. Raspberry Pi

Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito (figura 12) desarrollada por la Fundación Raspberry Pi en el Reino Unido. Es capaz de realizar cualquier cosa como se esperaría de una computadora de escritorio. Fue diseñada con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Figura 12. **Raspberry Pi 2 modelo B**



Fuente: *Raspberry Pi 2 modelo B.* https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. Consulta: enero de 2017.

1.2.3. Características de hardware

Periódicamente, la Fundación Raspberry Pi ha ido liberando nuevas versiones del hardware con características superiores, como mayor potencia computacional y nuevas tecnologías implementadas.

Tabla I. **Especificaciones de hardware para diferentes versiones de Raspberry Pi**

	Modelo A	**Modelo A+**	**Modelo B**	**Modelo B+**	**Versión 2 Modelo B+**
SoC	BCM2835	BCM2835	BCM2835	BCM2835	BCM2835
CPU	ARM11 V6 700 MHz	ARM11 V6 700 MHz	ARM11 V6 700 MHz	ARM11 V6 700 MHz	ARM11 V7 CORTEX A7 4 núcleos 900 MHz

GPU	BROADCOM VIDEOCORE IV 250 MHz. OPENGL ES 2.0	BROADCOM VIDEOCORE IV 250 MHz. OPENGL ES 2.0	BROADCOM VIDEOCORE IV 250 MHz. OPENGL ES 2.0	BROADCOM VIDEOCORE IV 250 MHz. OPENGL ES 2.0	BROADCOM VIDEOCORE IV 250 MHz. OPENGL ES 2.0
Memoria RAM	256 MB LPDDR SDRAM 400 MHz	256 MB LPDDR SDRAM 400 MHz	512 MB LPDDR SDRAM 400 MHz	512 MB LPDDR SDRAM 400 MHz	1 GB LPDDR2 SDRAM 450 MHz
Puertos USB	1	1	2	4	4
GPIO	26 pines	40 pines	26 pines	40 pines	40 pines
Vídeo	HDMI 1.4 1920X1200	HDMI 1.4 1920X1200	HDMI 1.4 1920X1200	HDMI 1.4 1920X1200	HDMI 1.4 1920X1200
Disco Duro	SD	MicroSD	SD	MicroSD	MicroSD
Ethernet 10/100	No disponible	No disponible	Sí	Sí	Sí
Tamaño (mm)	85,60X56,5	65X56,5	85,60X56,5	85,60X56,5	85,60X56,5

Fuente: *Especificaciones de hardware*. http://www.raspberryshop.es/hardware-raspberry-pi.php. Consulta: enero de 2017.

1.2.4. Fuente de alimentación

Una Raspberry Pi necesita una fuente de alimentación de 5V y 2A, recomendados en su versión 2. Cabe destacar que el consumo de corriente varía dependiendo de qué tipo de trabajo realice y los periféricos que tenga conectados, siendo en este caso 2A máximos en la versión 2, y 1A en la versión 1, debiendo mantener un voltaje constante de 5V.

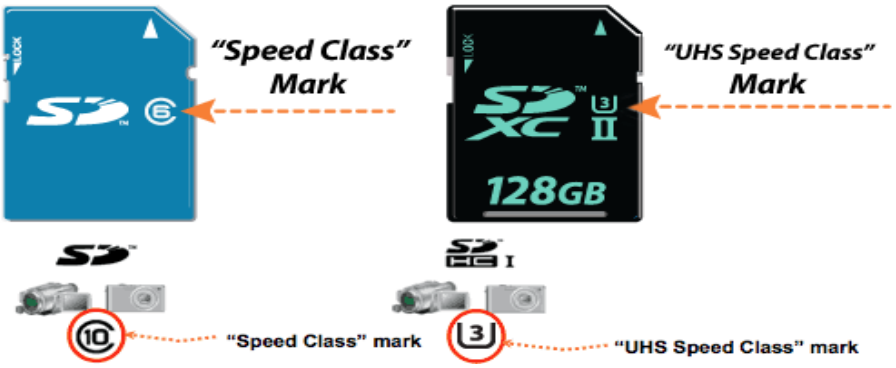
Muchas veces se necesitan más puertos USB de los que trae Raspberry Pi, para lo cual se le coloca un *HUB* USB, que funciona como extensor de conexiones. Se debe tener cuidado pues muchos de estos *HUB* no tienen una protección que evita una corriente de retorno a la Raspberry Pi. Para evitar

daños se recomienda adquirir un *HUB* de marca reconocida o que en sus especificaciones indique algún tipo de protección explícita.

1.2.5. Tarjeta de memoria SD

Una tarjeta de memoria SD (*Secure Digital*) es una pequeña tarjeta de memoria *flash* que ha sido diseñada para tener un gran espacio de almacenamiento y un tamaño reducido. Son utilizadas en muchos dispositivos electrónicos como teléfonos y computadoras. A través del tiempo han sido mejoradas al poseer mayor espacio de almacenamiento y velocidad de transmisión (velocidad de escritura/lectura). La asociación SD Card define la velocidad de clase (*Speed Class* estándar*)* con dos tipos de indicadores diferentes: el primero posee la marca de clase 2, 4, 6 y 10; la segunda posee la marca de clase UHS 1 y 3 o U1 y U3 (ver la figura 13).

Figura 13. **Marcas de clase para tarjetas de memoria SD**



Fuente: *Marcas de clase para tarjetas SD.* https://www.sdcard.org/developers/overview/speed_class/index.html. Consulta: enero de 2017.

A continuación se muestra una imagen comparativa entre los dos tipos de velocidad clase y algunas de sus aplicaciones:

Figura 14. **Comparación entre las marcas de clase de una tarjeta SD**

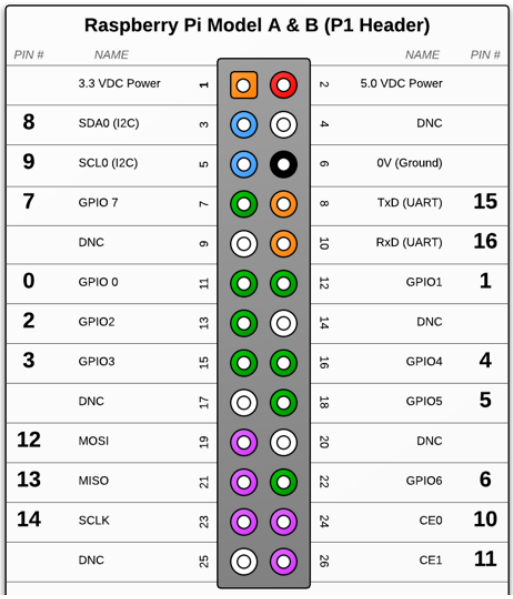
	Mark	Minimum Serial Data	SD Bus Mode	Application
UHS Speed Class	U3	30MB/s	UHS-II UHS-I	4K2K Video Recording
	U1	10MB/s		Full HD Video Recording HD Still Image Continuous Shooting
Speed Class	CLASS 10	10MB/s	High Speed	
	CLASS 6	6MB/s	Normal Speed	HD and Full HD Video Recording
	CLASS 4	4MB/s		
	CLASS 2	2MB/s		Standard Video Recording

Fuente: *Comparación entre marcas de tarjetas SD.*
https://www.sdcard.org/developers/overview/speed_class/index.html.
Consulta: enero de 2017.

1.2.6. GPIO

Entradas y Salidas de Propósito General (*General Purpose Input/Output*) es un puerto que posee un conjunto de pines que se pueden programar como entradas y salidas e incluso como funciones extras (SPI, UART, PWM, etc). Es una característica importante y atractiva que tiene Raspberry Pi, que además de ser una computadora, agrega este bus de expansión muy parecido a los puertos de un microcontrolador general, aunque menos eficiente, ya que el procesador de Raspberry Pi también debe compartir los recursos para el sistema operativo. En la figura 15 se aprecia la configuración de pines (*pinout*) de la versión original Raspberry Pi, que cuenta con 26 pines en total.

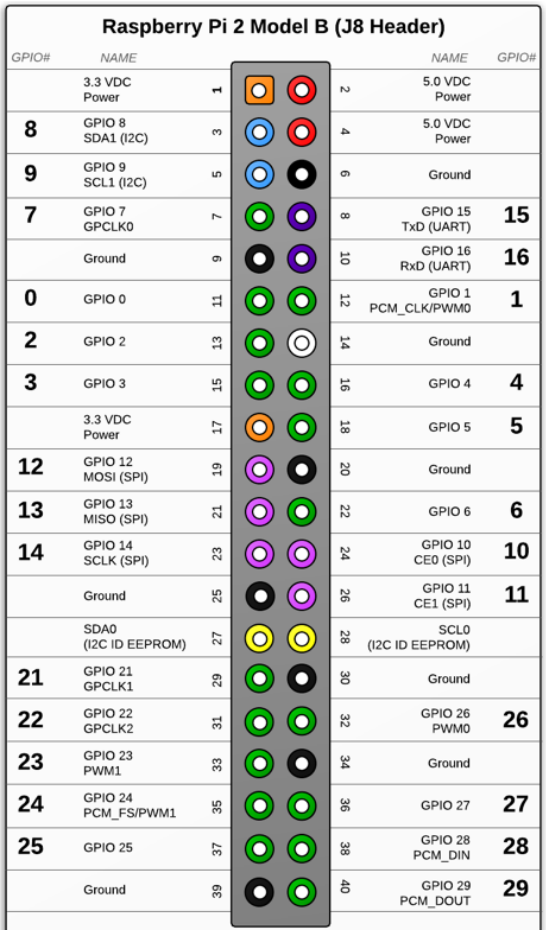
Figura 15. ***Pinout* Raspberry Pi v1 revisión 2**



Fuente: *Pinout de Raspberry Pi.* http://pi4j.com/pins/model-b-rev2.html. Consulta: enero de 2017.

La versión 2 cuenta con mayor cantidad de pines disponibles, en total 40, y se observa en la figura 16:

Figura 16. ***Pinout* Raspberry Pi V2**



Fuente: *Pinout de Raspberry Pi 2*. http://pi4j.com/pins/model-2b-rev1.html.Consulta: enero de 2017.

Es importante mencionar, respecto al GPIO de Raspberry Pi, que utiliza lógica de 3.3V, cualquier señal digital que reciba mayor a esa tensión, como en el caso de un microcontrolador que trabaja con voltaje de 5V, dañará permanentemente el dispositivo. Para evitar cualquier tipo de avería se usa un circuito convertidor de niveles de voltaje.

1.2.7. Microcontrolador Arduino

Arduino es una plataforma de código abierto y fácil uso de hardware y software que consiste en una placa con un microcontrolador con entradas y salidas. Además de un entorno de desarrollo integrado, tiene por objetivo facilitar el uso de la electrónica. Arduino es fácil incluso para los principiantes. Sus características son: económico, multiplataforma, entorno de programación limpio, capacidad para extender hardware y software compatibles. Aunque Arduino no necesariamente es el microcontrolador más eficiente, es uno de los más populares, cuenta con abundante información de utilización y es usado tanto en proyectos simples como los complicados sistemas de control industriales.

- Arduino IDE: es un entorno de desarrollo integrado para crear aplicaciones en Arduino. El lenguaje de programación es muy parecido a C/C++, pero está basado en Processing (que a su vez está basado en Java). Trae muchas librerías para utilizar el propio hardware, pantallas LCD, motores, sensores, módulo serial, Wifi, o incluso se pueden agregar librerías externas para tener compatibilidad con mayor cantidad de hardware.

1.2.8. Arduino Mega

El microcontrolador Arduino Mega se basa en otro microcontrolador ATmega1280 de 8 *bit* con arquitectura de conjunto de instrucciones reducidas RISC. Un resumen de sus especificaciones principales se muestra en la tabla II:

Tabla II. **Especificaciones de Arduino Mega**

Parámetro	Descripción
Microcontrolador	ATmega1280
Voltaje de operación	5 V
Cantidad de pines digitales E/S	54 (15 pueden ser PWM)
Entradas analógicas	16
Corriente DC en cada pin E/S	40 mA
Memoria Flash	128 KB (8 KB bootloader)
Memoria SRAM	8 KB
Memoria EEPROM	4 KB
Velocidad de Reloj	16 MHz

Fuente: *Arduino Mega*. https://www.arduino.cc. Consulta: enero de 2017.

La alimentación se realiza a través del puerto USB al conectarse a una computadora (con protección contra sobrecorrientes) o a través de una fuente de alimentación externa, debido a que cuenta con un regular de voltaje. La alimentación externa puede estar entre los rangos de 6 V a 20 V, pero es recomendable utilizar una fuente externa entre 9 V a 12 V. También cuenta con un pin designado “VIN” para alimentarse con una fuente regulada de 5 V.

Cabe mencionar que cuenta adicionalmente con cuatro puertos seriales UART por hardware en pines específicos o se pueden emular mediante software en cualquier pin digital. Las entradas analógicas pueden ser comparadas con un voltaje de referencia distinto a GND o VCC mediante un pin de entrada analógica llamado “AREF”.

1.2.9. Sensores de bajo costo seleccionados

1.2.10. Sensor de humedad

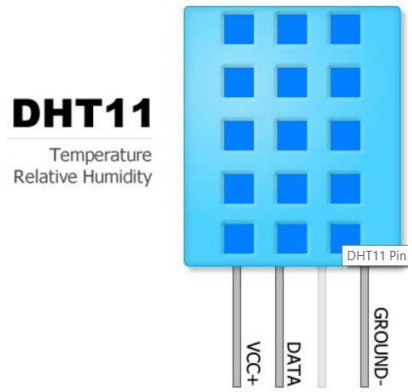
El principio de medición de un sensor de humedad de tipo capacitivo es a través de cambio en la constante dieléctrica. Se utilizan ciertos materiales, como polímeros entre dos electrodos conductores. Naturalmente la constante dieléctrica cambia a medida que absorbe agua de forma proporcional a la humedad relativa del ambiente circundante. El sensor de temperatura y humedad DHT11 es un módulo compuesto que contiene una salida digital calibrada para humedad y temperatura. El módulo consta de un pequeño sensor de humedad capacitivo, un sensor tipo NTC para medir temperatura y un pequeño microcontrolador de 8 *bit.* Entre las aplicaciones se encuentran (según la hoja de especiaciones):

- Equipo de medición e inspección
- Bienes de consumo
- Industria automotriz
- Control automático
- *Dataloggers* (registrador de datos)
- Estaciones meteorológicas
- Regulador de humedad
- Aplicaciones médicas y otras mediciones de humedad y control

Un aviso proveniente de la hoja de especificaciones del dispositivo indica claramente que no se debe usar como el dispositivo principal para seguridad o paradas de emergencia, de hecho en este contexto es utilizado únicamente como dispositivo de prevención en conjunto con el resto de dispositivos aquí descritos. Entre sus características se encuentran: bajo costo de adquisición,

estabilidad duradera, medición de temperatura y humedad en el mismo dispositivo, una fuerte disposición ante interferencias y una calibración precisa de fábrica. La figura 17 muestra la apariencia del módulo.

Figura 17. **Módulo DHT11**



Fuente: Módulo DHT11. http://steve.zazeski.com/arduino-dht11-temperature-and-humidity-data-logger/. Consulta: enero de 2017.

1.2.11. Sensor de sonido

Un sensor de sonido es un dispositivo que capta un rango de frecuencias audibles por el oído humano, conociendo que los seres humanos pueden escuchar rangos de frecuencias comprendidos entre los 20 Hz a 20 kHz. Un micrófono es un transductor electroacústico que convierte la energía acústica en energía eléctrica. El rango de frecuencias muchas veces está más limitado, por ejemplo a frecuencias máximas de 2 kHz para la voz humana, dependiendo la aplicación que se le de.

1.2.12. Micrófonos *electret*

Los micrófonos *electret* son útiles para detectar el sonido del recinto que está bajo observación, puesto que captan muy bien el rango de frecuencias audibles y poseen una sensibilidad aceptada. Son tan comunes que están dentro de los muchos dispositivos considerados como de uso cotidiano, tales como teléfonos celulares o *laptop*. La apariencia física del micrófono se muestra en la figura 18. Los circuitos para utilizarlo se describen más adelante.

Figura 18. **Micrófono *electret***

Fuente: *Micrófono electrect.* https://www.cooking-hacks.com/
Consulta: enero de 2017.

Ventajas de los micrófonos *electret:*

- Su precio es muy bajo
- Sensibilidad aceptable
- No es difícil de conseguir
- Rango de frecuencias amplio

Desventajas de los micrófonos *electret:*

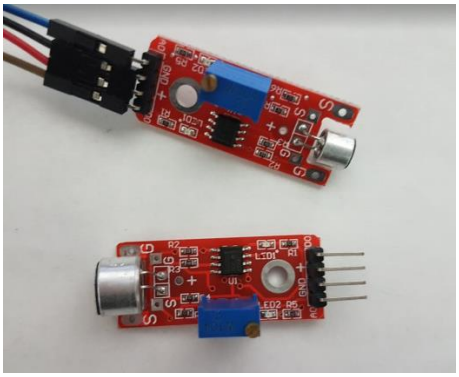
- La sensibilidad disminuye a frecuencias más elevadas.

- El nivel de potencia de salida es muy bajo. En algunas aplicaciones es necesario usar un circuito preamplificador.
- Alcanza un nivel de saturación muy rápido si la intensidad de volumen es alto.

1.2.13. Módulo detector de sonido

El módulo detector de sonido que se muestra en la siguiente figura incorpora el sensor de sonido *electret* junto al resto de circuitos, para conectarlo a un microcontrolador directamente. Permite realizar dos tipos de lecturas: una digital para establecer el potenciómetro en un valor fijo *setpoint* que, al superar el umbral, enviará un estado lógico al microcontrolador; y una medición analógica, que se conecta al ADC interno del microcontrolador para obtener valores cuantizados de los niveles de voltaje.

Figura 19. **Módulo detector de sonido**



Fuente: elaboración propia.

La ventaja de utilizar el módulo frente al sensor *electret* es que el circuito ya está montado en una placa y se puede conectar directamente el microcontrolador.

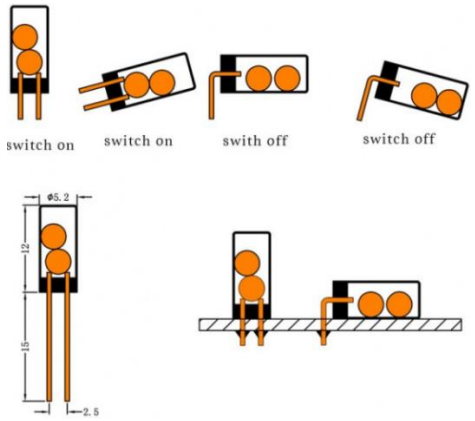
1.2.14. Sensor de vibración

- Sensor de inclinación

Los sensores de inclinación conocidos en inglés como *tilt sensors o* sensores *tilt,* son un tipo de sensor que detecta la inclinación y lo hace a través de colocarlo en una ubicación donde se define un plano de referencia imaginario con un conjunto de coordenadas asociadas al dispositivo que, cuando existe un movimiento, detectará si existe inclinación. Actualmente estos sensores se construyen utilizando pequeñas bolas conductoras conocidas como doble esfera, estas actúan dentro de un cilindro conductor o bien utilizando un contacto en un extremo o en el centro, mientras el otro contacto está ubicado en un extremo o en el centro. Las pequeñas esferas, al existir una inclinación, cerrarán el circuito tocando los contactos.

El sensor de inclinación SW-520D proporciona una salida que se puede interpretar como digital, ya que funciona abriendo y cerrando un circuito, dos estados que indican inclinación o su ausencia. No es capaz de detectar los ángulos de inclinación asociados, únicamente si existe inclinación a partir de alcanzar cierto umbral. Un microcontrolador puede detectar el sensor de forma muy sencilla, no obstante, al ser un sensor mecánico, tiene asociado un rebote (*debouncing)*. El problema se soluciona al utilizar un circuito antirebote.

Figura 20. **Sensor SW-520D de funcionamiento**



Fuente: Sensor SW-520D. http://www.prometec.net/tilt-switch/. Consulta: enero de 2017.

El comportamiento del sensor de la figura 20 es igual al de un interruptor cualquiera, pero en este ámbito es útil para medir las “vibraciones bruscas” en un entorno donde se desee tener control. En el mercado se encuentra disponible también en módulo, como se muestra en la figura 21.

Figura 21. **Sensor SW-520D en módulo**

Fuente: *Sensor SW-520D en módulo.* http://www.electroschematics.com/12124/arduino-tilt-sensor-experiment/. Consulta: enero de 2017.

- Sensor de golpe

Es un sensor capaz de percibir los golpes que se le dan; si está anclado sobre una superficie detectará las vibraciones sobre esa superficie. La construcción consta de un poste conductor en el centro, rodeado de un conductor tipo resorte que lo rodea completamente. Si se produce un golpe lo suficientemente fuerte para que los conductores se toquen se cerrará el circuito y circulará una pequeña corriente, de esa forma es posible realizar detecciones de vibración (ver la figura 22).

Figura 22. **Módulo sensor de golpe**

Fuente: Módulo sensor de golpe. http://www.gearbest.com/development-boards/pp_22511.html. Consulta: febrero de 2017.

1.2.15. Sensor de temperatura

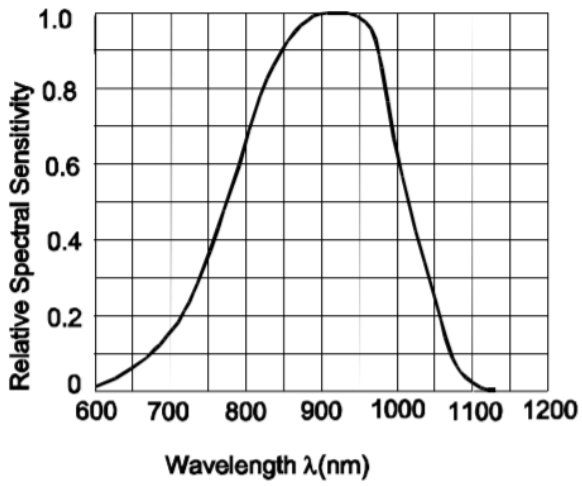
La temperatura es un fenómeno físico que indica la medida de energía térmica en una sustancia o en el ambiente. La temperatura se mide en grados Celsius, Fahrenheit o Kelvin. Para detectar la temperatura en el ambiente que se ha designado, se utilizarán varios sensores para relacionar la información respecto a la temperatura.

- Detección de llamas

Los sensores de flama tienen como función principal detectar la presencia de llamas debido al fuego. Una manera de detectar llamas es a través de la combustión de un compuesto inflamable, en combinación de oxígeno que necesariamente produce la emisión de luz que un sensor óptico puede detectar.

Se utilizará un sensor óptico que corresponde con un fototransistor con filtro IR. Los filtros infrarrojos están ajustados a medida entre 760-1100nm, ángulo medio de sensibilidad de 60º y una distancia de detección aproximada de 80cm máximo (ver la figura 23).

Figura 23. **Gráfica de longitud de onda vs sensibilidad espectral relativa de un fototransistor**



Fuente: *Longitud de onda versus sensibilidad espectral.* https://www.kingbrightusa.com/images/catalog/spec/WP3DP3BT.pdf. Consulta: febrero de 2017.

Este tipo de sensor de flama por sí solo no es suficiente como para catalogarlo de auténtico dispositivo de seguridad contra el fuego, siempre será necesario combinarlo con otros tipos de sensores para que sea efectivo. El módulo del sensor de flama se presenta en la figura 24, construido con un circuito que se analiza más adelante.

Figura 24. **Fototransistor. Módulo IR detector de flama**

Fuente: *Fototransistor.* http://www.inventionhardware.com/product/flame-sensor-module/#prettyPhoto. Consulta: febrero de 2017.

- Termómetro digital DS18B20

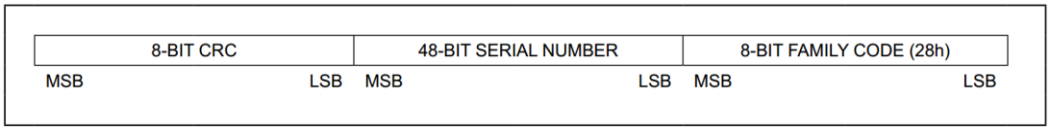
La hoja de especificaciones de este dispositivo lo describe como un termómetro digital con resolución programable de 9 a 12 *bit* que está calibrado en grados Celsius. Si se desea utilizar otro sistema de medida se deben aplicar las fórmulas de conversión de temperatura. Entre las aplicaciones descritas se encuentran:

- Controles termostáticos
- Sistemas industriales
- Termómetros

- Sistemas térmicos sensibles

El protocolo de comunicación usado es 1-Wire, lo que permite conectar más de un dispositivo en el mismo bus de comunicación. Para reconocer cada uno de forma individual en el bus sensor tiene escrito un código de 64 *bit* único (ROM *code*), lo que permite conectar un número ilimitado de sensores según la hoja de especificaciones. Los 8 *bit* menos significativos corresponden a indicar la familia del dispositivo DS18B20 con el valor hexadecimal 24h; 56 *bit* para el código serial y los 8 *bit* más significativos es la comprobación de error de redundancia cíclica CRC, calculados a partir de los primeros 56 *bit* del código único (ver la figura 25).

Figura 25. **Código ROM 64-*bit* para el sensor DS18B20**



Fuente: *Maxim integrated datasheet.* http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf Consulta: febrero de 2017.

Existen dos formas de alimentar el sensor, una es la manera común de alimentar al dispositivo, ya que cuenta con los terminales V_{DD} y GND; otra forma adicional es llamada alimentación parásita (*parasitepower*). Esta forma de alimentar el sensor no necesita conectar el terminal V_{DD} y se utiliza la misma línea de comunicación para energizar todo el circuito integrado, reduciendo los pines utilizados a únicamente dos.

No se utilizará la alimentación parásita debido a ciertos inconvenientes que presenta el sensor con la energía que consume al realizar ciertas tareas,

por ejemplo, después que el sensor comienza el proceso de conversión analógica-digital, utiliza una corriente mayor que en estado inactivo, por consiguiente se debe utilizar una resistencia *pull up* que pueda garantizar dicha corriente (*strong pull up*), de lo contrario existe el riesgo de pérdida de energía en el sensor y una medición imprecisa o fallo temporal, mientras que, alimentado en modo normal, no causa preocupaciones.

1.2.16. Sensor de gases

Los sensores utilizados para medir gases son de la serie MQ y se encuentran disponibles en el mercado, generalmente en módulos que simplifican en cierta medida su uso. Los sensores MQ son capaces de medir diferentes gases, unos con mayor precisión y otros con menor precisión, por ejemplo, en el caso del sensor de gas MQ-3 es sensible al alcohol, LPG y bencina, entre otros. La característica del MQ-3 es que es altamente sensible al alcohol y poco sensible a la bencina.

Los sensores MQ, al detectar la presencia de gases, varían su resistencia interna. Se define la resistencia del sensor R_s como "la resistencia interna del sensor a distintas concentraciones de gases"[1]. Además se define R_o como el valor constante bajo ciertas condiciones de trabajo, la definición es diferente para cada sensor MQ.

- Sensor MQ-2

Conocido también como sensor de gases combustibles, es utilizado en la detección de fugas de gases en mercados de consumo e industria. Es capaz de

[1] MQ-2 Gas Sensor Datasheet. p. 2.

detectar isobunato (metilpropano), propano, metano, alcohol, hidrógeno, humo y LPG. Su apariencia se aprecia en la figura 26:

Figura 26. **Módulo sensor de gas MQ-2**

Fuente: *Módulo sensor de gas MG-2.* https://rydepier.wordpress.com/2015/07/02/mq2-gas-sensor-and-arduino/. Consulta: febrero de 2017.

Se define R_o para el sensor MQ-2 como “la resistencia del sensor cuando está expuesto a un ambiente de 1 000 PPM de H_2 (dihidrógeno) en aire limpio”[2].

1.2.17. Aplicaciones de software

1.2.18. Lenguaje C

Es un lenguaje de programación de propósito general con estilo de programación imperativo. Soporta programación estructurada, destacada por la eficiencia de código que logra. Se trata de un lenguaje que posee estructuras típicas de lenguajes de alto nivel, pero además es importante mencionar que posee estructuras que permiten un control a un nivel muy bajo, muchos compiladores permiten mezclar entre código ensamblador y C. Es un lenguaje

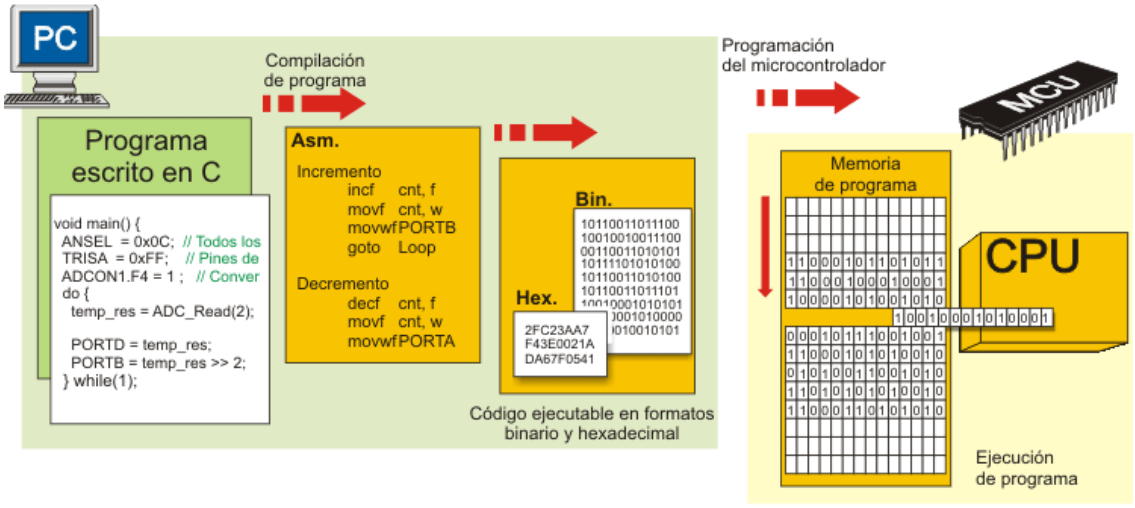
[2] MQ-2 Gas Sensor Datasheet. p. 2.

de nivel medio y portable, es decir, es independiente del hardware si se tiene el cuidado de utilizar las bibliotecas estándar y no funciones específicas de cada sistema operativo.

En el ámbito de los microcontroladores, C es el lenguaje más eficiente después de ensamblador, al permitir crear programas en mucho menor tiempo y con gran eficiencia de código máquina. Las características de C son muy útiles para programar microcontroladores. Es un lenguaje compilado, por lo que la ejecución de instrucciones es más rápida que en otros tipos de sistemas no compilados. C permite realizar operaciones, tanto con *bits* como con *bytes*, operaciones lógicas y desplazamientos, está estandarizado en el llamado ANSI C.

Muchos compiladores para microcontroladores en C permiten la manipulación directa desde C o traen *plug-in* para insertar código de bajo nivel en ensamblador y dar un control total del hardware del microcontrolador. Las fases de compilación en general para un microcontrolador se pueden observar en la siguiente figura:

Figura 27. **Fases de compilación del microcontrolador**



Fuente: *Fases de compilación del microcontrolador.* http://www.mikroe.com/chapters/view/80/capitulo-2-programacion-de-los-microcontroladores/. Consulta: febrero de 2017.

1.2.19. Python

Es un lenguaje de programación interpretado, *tipado* dinámico, fuertemente *tipado* y multiplataforma; sintaxis simple, clara y limpia. Python es un potente lenguaje que posee muchas librerías para realizar diversas tareas que van desde controlar los GPIO en una Rasberry Pi hasta utilizar *frameworks* para trabajar del lado un servidor *web*. Python es promovido como el lenguaje de programación principal por la Fundación Raspberry Pi, para su dispositivo del mismo nombre. No obstante, Python no siempre es el más adecuando en cuanto a programación a bajo nivel o aplicaciones en que el rendimiento sea crítico (tiempos muy precisos).

1.2.20. PHP

Es un lenguaje de programación de propósito general interpretado pero usado principalmente para desarrollo *web* en el lado servidor. PHP puede generar resultados en una página *web* con lenguaje de etiquetas HTML y da la capacidad de gestionar contenido dinámico.

1.2.21. HTLM

Es un lenguaje de etiquetas de hipertexto. Es el lenguaje estándar para crear páginas *web*, los navegadores *web* analizan el código HTML y construyen (renderizan) la página *web* que el usuario común ve. Todos los navegadores son capaces de comprender HTML, por lo que naturalmente es multiplataforma. Para dar estilo y formato a las páginas HTML se utiliza un lenguaje de diseño gráfico denominado las hojas de estilos en cascada CSS, el cual permite realizar páginas HTML muy vistosas.

1.2.22. Sistema operativo Raspbian

Para utilizar la Raspberry Pi es necesario utilizar un sistema operativo capaz de soportar su hardware. Raspbian basado en Linux es el sistema operativo oficial de la Fundación Raspberry, aunque no es el único, existen otras distribuciones que se pueden utilizar, como por ejemplo Fedora ARM, Arch Linux ARM o Windows 10 IoT Core Edition.

1.2.23. Servidor *web* Apache

Un servidor *web* surge de la necesidad de crear un estándar para comunicar información, el resultado es el protocolo HTTP, capaz de realizar

conexiones bidireccionales, síncronas o asíncronas entre un cliente y un servidor. Existen varios servidores *web* utilizados, Apache es el más popular y usado. Entre sus ventajas se encuentran: multiplataforma, modular, soporte y documentación amplias. Apache permite trabajar con diferentes lenguajes de programación como PHP, Perl o Python.

1.2.24. Servidor de transmisión de video

A la acción de transmitir contenido multimedia se le conoce como *streaming* (emisión o difusión). Para transmitir video se debe contar con una conexión hacia el dispositivo emisor de contenido, un ancho de banda de la red suficiente (y calidad de servicio QoS, si es posible), para evitar demasiado retraso en la recepción de información. A diferencia de otros servicios en Internet que utilizan *códec* especiales para comprimir y enviar la información, como Youtube o Netflix, en este caso se utilizará un software de línea de comandos y código abierto llamado “mjpg-streamer”.

Esta utilidad fue desarrollada originalmente para dispositivos embebidos con una limitada cantidad de recursos, como la memoria RAM y procesador. Es por esta razón que es ideal para la aplicación que se va a desarrollar. Mjpg-streamer genera imágenes en formato JPEG que obtiene de alguna fuente de entrada, como una cámara *web,* y las envía para crear la ilusión de video; es capaz de soportar diversos complementos conocidos como *plugin*, que soportan múltiples dispositivos de entrada. Para visualizar la transmisión se utiliza algún navegador *web* (como Google Chrome o Firefox, en otros) o software capaz de recibir transmisiones (como VLC media player).

2. CONFIGURACIÓN DE SOFTWARE

2.1. Preparando el ambiente de trabajo en Raspberry Pi

2.1.1. Instalación de un sistema operativo

El primer paso es la elección de un sistema operativo compatible con Raspberry Pi, a partir de eso se realiza la instalación de lSO, para que tenga el control total. El sistema operativo seleccionado es “Raspbian”, porque es una de las distribuciones más estables basadas en Debian, con actualizaciones frecuentes y optimizadas para el hardware Raspberry Pi, y es de código libre y gratis. Para poder instalar el sistema operativo se necesita de una tarjeta SD o microSD, una recomendación útil es utilizar una tarjeta de clase 10, la cual garantiza una velocidad de lectura/escritura a 10MB/s, aumentando la eficiencia del microordenador.

Tomando como referencia a un usuario promedio que utiliza un sistema operativo basado en Microsoft Windows, se procede con la instalación de Raspbian, acto previo se descarga el archivo de imagen del sistema operativo mencionado anteriormente de fuentes oficiales. Luego se formatea y se transfiere la imagen del sistema hacia la tarjeta de memoria SD. Se utilizan dos herramientas:

- Formateo de la unidad de memoria. La herramienta que se utiliza para dar formato a cualquier tarjeta de memoria SD se llama SD Formatter. Esta herramienta la recomienda la propia asociación SD (SD Association) para usarla en lugar de las herramientas que

proveen los sistemas operativos, ya que estos utilizan un formateo genérico que resulta en la pérdida de rendimiento óptimo de las tarjetas de memoria SD. El formato que utiliza es FAT32.

- Escritura del archivo de imagen en la unidad de memoria. Para trasladar el contenido de la imagen hacia la unidad de memoria SD se realiza con la herramienta “win32diskimager”, la cual copia el contenido completo de la imagen del sistema en la memoria.

El siguiente paso consiste en colocar la tarjeta SD en Raspberry Pi y encenderla. No se conectará ningún monitor, teclado o *mouse* en Raspberry Pi, por lo tanto se realiza una conexión remota desde una computadora estándar hacia la Raspberry Pi. Para realizar dicha conexión remota se necesita estar dentro de una red LAN. En la primera iniciación del sistema operativo Raspbian está habilitada por defecto la obtención de una dirección IP a través del protocolo DHCP. A continuación se describen dos formas para lograr una conexión remota de forma satisfactoria:

- Utilizando un *router.* La computadora desde donde se está trabajando y la Raspberry Pi deben conectarse al *router* (configurado de forma que tenga habilitado DHCP), entonces ambos reciben una dirección IP distinta. La dirección IP de interés es la que tiene la Raspberry Pi.
- Si se puede acceder al *router*, entonces se verifica en la lista de direcciones otorgadas la última dirección IP asignada, pues intencionalmente se conecta la Raspberry Pi de último.
- Si no es posible acceder al *router*, quizás por razón de no ser el administrador o simplemente no poseer la contraseña de ingreso al dispositivo, se utiliza alguna herramienta capaz de escanear

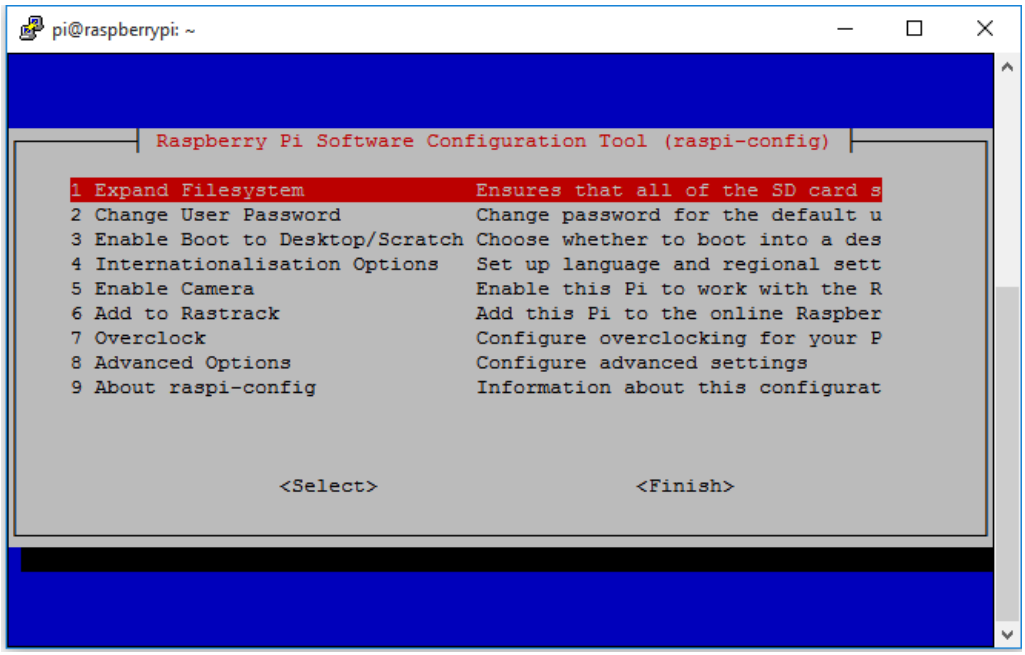
direcciones IP en la red a la cual se está conectado, por simplicidad se utiliza “Advanced IP Scanner”.

- Utilizando Windows ICS (conexión compartida a Internet). Este método consiste en habilitar un servidor DHCP en Windows, que proporciona una red 192.168.137.0/24 (Windows 7, 8 y 10); 192.168.0.0/24 (Windows XP). Para poder activar ICS se accede al panel de control, luego a redes, luego a configuración de adaptadores, se deja un adaptador Ethernet libre para conectarlo a la Raspberry Pi y con el otro adaptador, ya sea un adaptador Wifi o Ethernet, se comparte la conexión y se selecciona el adaptador donde se encuentra conectada la Raspberry Pi. Por último, se utiliza Advanced IP Scanner para encontrar la dirección IP de la Raspberry Pi.

Con la dirección IP obtenida, resta conectarse remotamente utilizando el protocolo SSH, para ese fin se utiliza Putty, que permite ejecutar comandos de forma remota como si se estuviera directamente en la consola de Raspbian.

Naturalmente, al conectarse mediante la consola de Putty, para acceder a Raspbian en la Raspberry Pi, se debe realizar la autenticación que consiste en un nombre de usuario y una contraseña, por defecto son: pi y “aspberry, respectivamente (sin comillas y todo en minúscula). Para realizar la primera configuración se utiliza el comando como súper usuario: “sudo raspi-config”, como se muestra en la figura 28:

Figura 28. **Configuración inicial en Raspberry Pi**



Fuente: elaboración propia, utilizando Putty.

- *Expand Filesystem.* Literalmente expande el sistema de archivos en la totalidad de la tarjeta SD. Usualmente una imagen del sistema consiste en 2GB de información, cuando se copia el contenido de la imagen en una tarjeta SD de mayor capacidad queda una porción sin utilizar. Para poder utilizar la totalidad de la tarjeta se expande el sistema de archivos. Después de ejecutar esta herramienta aparece un mensaje que indica que el sistema de archivos se expandirá en el siguiente reinicio del sistema.
- *Change User Password*. Por defecto la contraseña del usuario pi es raspberry; con esta opción se puede cambiar por otra contraseña personal, lo que es una opción recomendada para evitar que cualquier intruso acceda al sistema.

- *Enable Boot to Desktop/Scratch*. Esta opción permite iniciar el sistema con la consola o en el entorno gráfico, como sucede en Windows o MAC OS. El inicio en modo consola de texto viene por defecto y se dejará intencionalmente así, para evitar el consumo de recursos ligado al entorno gráfico.
- *Internationalisation Options*. A través de esta opción se puede cambiar la zona horaria, el idioma del teclado y el idioma para Raspberry Pi.
- *Enable Camera*. Si se posee el módulo especial de cámara para Raspberry Pi se habilita esta opción que garantiza al menos 128MB de memoria RAM dedicada al GPU (unidad de procesamiento gráfico).
- *Add to Rastrack*. Es un servicio que permite a la comunidad Raspberry agregar a un mapa su localidad, para que pueda ser vista por todos. El servicio está localizado en “rastrack.co.uk”.
- *Overclock*. Sirve para aumentar la velocidad de reloj, la propia Raspberry Pi indica que se debe utilizar una velocidad moderada para evitar inestabilidades o reducción del tiempo de vida útil.
- *Advance Options*. Configura las opciones avanzadas del sistema:

  - OverScan: en monitores y TV antiguos que utilizan la opción para que la pantalla aparezca de forma correcta.
  - Hostname: nombre visible que tiene Raspberry Pi cuando está conectada a la red.
  - Memory Split: cambia la cantidad de memoria disponible para la GPU.
  - SSH: habilita o deshabilita el acceso remoto por línea de comandos utilizando el protocolo SSH en la Raspberry Pi.

- Device Tree: sirve para administrar ciertos recursos y carga de módulos.
- SPI: habilita o deshabilita la interfaz de comunicación SPI, útil cuando se tienen dispositivos que utilizan este protocolo de comunicación.
- I2C: como en el caso anterior, habilita o deshabilita la interfaz I2C.
- Serial: habilita o deshabilita los mensajes del kernel a través de la conexión serial.
- Audio: selecciona la salida de audio por HDMI o conector de audio estándar de 3,5 mm.
- Update: actualiza la herramienta a la versión más reciente.

- *About raspi-config.* Despliega un mensaje que describe la utilidad de utilizar la herramienta raspi-config.

Al terminar de configurar la Raspberry Pi con las opciones antes mencionadas, se reinicia el sistema para que los cambios surtan efecto, entonces se está preparado para empezar el trabajo.

- En resumen, estas son las partes esenciales para la instalación de Raspbian:

  - Descargar la imagen del sistema operativo Raspbian del sitio oficial.
  - Realizar un formateo de la memoria SD.
  - Transferir la imagen a la tarjeta SD.
  - Primer inicio de Raspberry Pi:
    - Encontrar dirección IP.
    - Conectarse de forma remota con Putty.

- Configurar Raspbian de acuerdo a las necesidades del proyecto.
- Actualizar el sistema a la versión más reciente.

- Instalación de un entorno gráfico remoto. Es opcional pero es útil muchas veces cuando no se está familiarizado con la consola de Linux (ver la sección de apéndices para más información).

2.1.2. Instalación y configuración de servidor LAMP

LAMP se refiere a la utilización en conjunto de Linux-Apache-MySQL-PHP como las herramientas para un servidor, utilizando la gran estabilidad de Linux como sistema operativo, el servidor *web* Apache como uno de los más utilizados y populares, la base de datos MySQL y el lenguaje de *scripts* PHP.

En este contexto Linux es representado por Raspbian, pues está basado en ese sistema operativo. Para instalar el resto de herramientas LAMP se utiliza el gestor de aplicaciones APTITUDE para descargar e instalar los programas utilizando la consola de Linux.

Para el caso de Apache se ejecuta el comando APTITUDE en modo súper usuario con el parámetro de instalación:

sudo apt-get install apache2 -y

El directorio raíz donde están ubicados todos archivos para mostrar la página *web* en Raspbian es *"/var/www/html/".* Para poder observar la página *web* de prueba, se accede mediante un navegador *web* y la dirección URL es *"http://localhost/"* si se accede desde la Raspberry Pi, o en una computadora

cualquiera en la misma red, utilizando la dirección IP de la Raspberry Pi por ejemplo *"http://192.168.1.25".*

Un inconveniente desde el punto de vista del usuario son los permisos de Linux para ciertos directorios del sistema; en el caso el directorio *"/var/*"* está protegido para que solo el usuario *root* pueda realizar cualquier modificación. Una opción es utilizar el usuario *root* para realizar las modificaciones y otra consiste en cambiar los permisos sobre los archivos en esos directorios mediante el comando *sudo chown* e incluir al usuario del sistema *pi.*

Para instalar PHP se utiliza el gestor de aplicaciones APTITUDE mediante el siguiente comando:

sudo apt-get install php5 libapache2-mod-php5 -y

Para que el intérprete de PHP entre en funcionamiento se debe renombrar los archivos con la extensión *.php, por ejemplo en el caso del archivo *index.html* se reemplaza por *index.php*, además los archivos deben tener código en PHP.

Por último, para finalizar la instalación de LAMP se debe instalar la base de datos relacional MySQL. También se utiliza el gestor de aplicaciones APTITUDE y se instala mediante el comando:

sudo apt-get install mysql-server php5-mysql -y

En un momento de la instalación la base de datos preguntará por el ingreso de una contraseña para el usuario *root* de MySQL.

2.2. Transmisión de video

La aplicación *mjpg-streamer* no viene instalada por defecto en Raspbian, se debe proceder a descargarla y compilarla. Para realizar dicha acción se crea una carpeta en el usuario que se tiene en Raspbian, por defecto es pi. En su directorio *Home* se puede crear una carpeta que contiene todos los archivos necesarios para utilizar *mjpg-streamer*. Se procede a descargar de la fuente oficial y a compilarlo con el comando de Linux *make*.

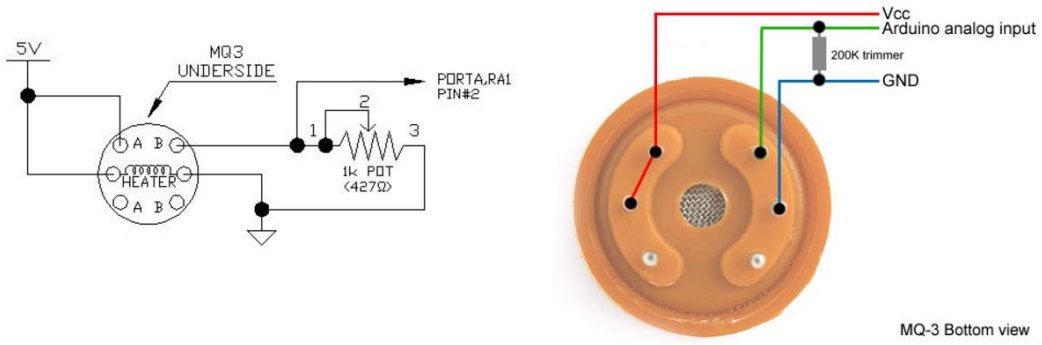
Ahora es momento de lidiar con la fuente de entrada, es decir, la cámara *web.* En Raspbian, cuando se conecta un periférico como una cámara en algún puerto USB, es capaz de reconocerlo y colocar un *driver* genérico para ese dispositivo (para cualquier dispositivo que necesite un *driver* especial, se tendrá que instalar para hacer funcionar al dispositivo), pero se utilizarán cámaras *web* que funcionen con *drivers* genéricos. En la carpeta del sistema de Raspbian "*Dev*" se encuentran innumerables dispositivos, las cámaras *web* aparecen generalmente como "*VideoY*", donde Y es algún número entero a partir de cero que indica que la cámara esa conectada al sistema. Por ejemplo si hay conectada solo una cámara aparece como "*Video0*".

Ahora ya se puede utilizar para realizar la tarea de *streaming* de video, al conectar la cámara *web.* Se procede a iniciar el servicio de *streaming* mediante la línea de comandos de Linux o incluso se puede llamar desde un lenguaje de programación, como Python, para automatizar tareas. Para poder observar la transmisión se debe tener un equipo conectado en la misma red o estar en una red que alcance al servidor que transmite y conocer la dirección IP y el puerto de transmisión del emisor de video, Raspberry Pi. Las imágenes podrán ser vistas con baja latencia y se pueden modificar parámetros como la calidad de video y la compresión de imagen.

3. DISEÑO Y PARÁMETROS DE LOS DISPOSITIVOS ELECTRÓNICOS

Muchos de los dispositivos mostrados en este trabajo vienen por defecto en un módulo propio, en otras palabras, el sensor está montado en una placa con todos los dispositivos necesarios para hacerlo funcionar. Trae pines que se conectan al microcontrolador para hacer más sencilla su utilización. Está claro que se pueden adquirir sensores sin ningún módulo (ver la figura 29), y cada desarrollador puede realizar el circuito que lo pone en funcionamiento a gusto propio e incluso crear sus propios PCB y distribuirlos.

Figura 29. **Sensor de gas MQ3**

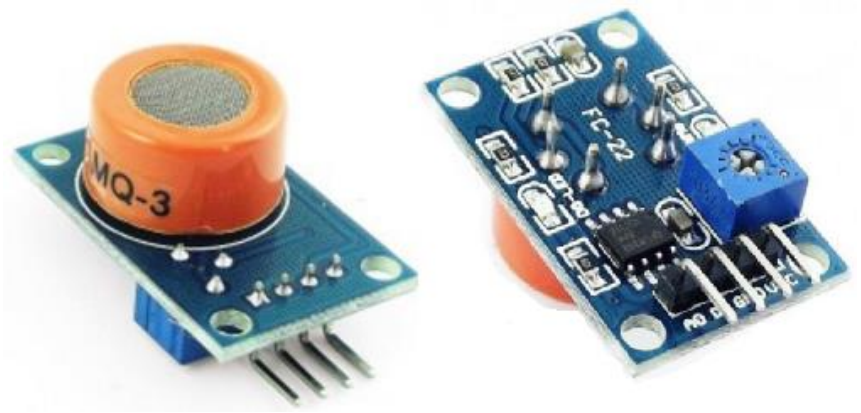


Fuente: *Sensor de gas MQ3*. http://teslabem.com/sensor-de-alcohol-mq3.html
Consulta: marzo de 2017.

Por ejemplo, el sensor de gas MQ-3 mostrado en la figura 30 viene soldado en un módulo acompañado por un potenciómetro, un comparador de

voltaje LM393, varias resistencias y LED de indicación. Todo el módulo es un circuito que permite utilizar el sensor de forma más amigable.

Figura 30. **Sensor MQ3 en un módulo**



Fuente: *Sensor MQ3 en un módulo.* http://www.naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html. Consulta: febrero de 2017.

3.1. Comparador de voltaje LM393

Muchos de los módulos aquí presentados contienen un circuito integrado LM393 que corresponde a un comparador de voltaje, por lo que es útil un análisis de su diseño. El LM393 es un circuito integrado que contiene dos comparadores de voltaje de precisión independientes, se diseñó para ser compatible con las lógicas TTL y CMOS, además las salidas de los comparadores son de colector abierto, dando la posibilidad de conectarlas entre sí directamente; para obtener un potencial alto se debe utilizar una resistencia *pull up.* Entre sus aplicaciones están:

- Conversores análogo-digital simples
- Generados de pulso y onda cuadrada

- Generadores de retardo temporal
- Amplio rango de uso para VCO
- Comparación de voltajes

3.1.1. Parámetros del comparador LM393

Se recomienda utilizar el dispositivo dentro de los límites recomendados por el fabricante. La tabla III muestra un resumen de los voltajes de alimentación recomendados:

Tabla III. **Condiciones de operación recomendadas para LM393**

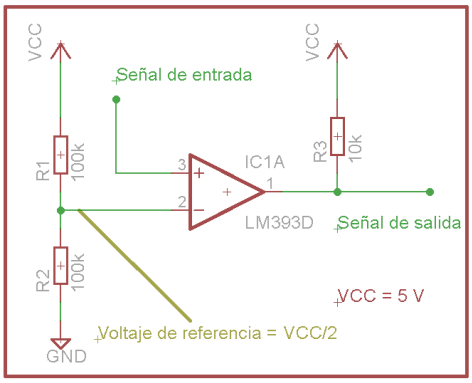
Parámetro	Valor	Unidad de medida
Voltaje de alimentación simple	$2.0 - 36$	V
Voltaje de alimentación dual	$\pm 1.0 - \pm 18$	V

Fuente: elaborado con base en *Datasheet* de Texas Instruments. Consulta: febrero de 2017.

Para realizar la gráfica del tiempo de respuesta del LM393 el fabricante definió[3] el circuito de operación muy simple que se muestra en la figura 31. En la entrada no inversora (IN+) se coloca una señal triangular con periodo T, mientras en la entrada inversora (IN-) se coloca un potencial de referencia con valor de $V_{CC}/2$.

[3] Texas Instruments Inc. *Datasheet LM393. P. 12.* Revisión: diciembre de 2014.

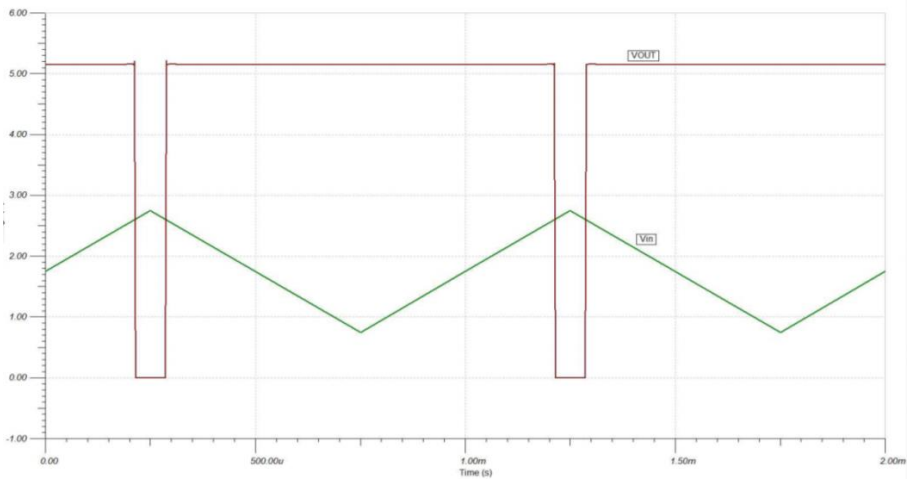
Figura 31. **Circuito de referencia para LM393**



Fuente: elaboración propia, utilizando Eagle. Con base en el *Datasheet LM393.*

La figura 32 muestra el tiempo de reacción en la salida del comparador, la salida es modificada cada vez que se supera o disminuye sobre umbral $V_{CC}/2$. La salida es invertida debido a que la salida del comparador está conectada a un transistor NPN dentro del circuito integrado.

Figura 32. **Curva de aplicación LM393**

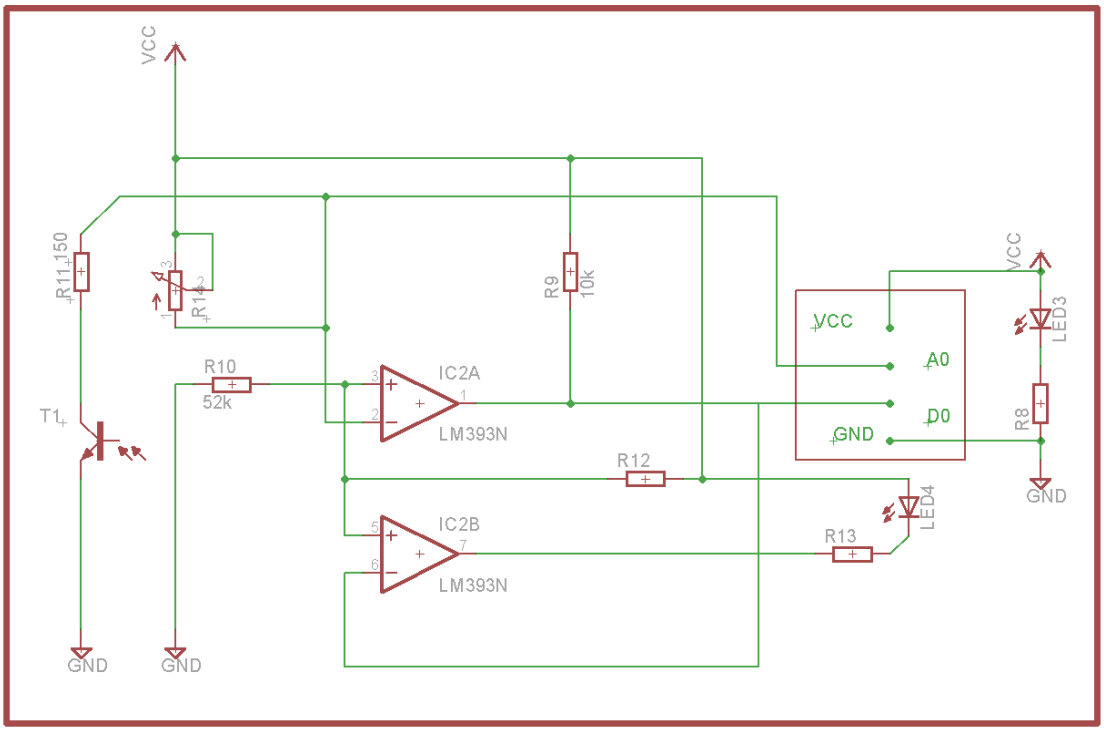


Fuente: Texas Instruments Inc. *Datasheet*. P. 12. Consulta: octubre de 2014.

3.2. Sensor de flama

El sensor de flama de la figura 24 se encuentra montado sobre un módulo que incluye: un fototransistor con filtro IR, varias resistencias, dos led de indicación, un potenciómetro de precisión y un comparador de voltaje LM393; el conjunto permite conectar el módulo directamente a cualquier microcontrolador de 5 V (ver la figura 33).Cuando el sensor detecta luz infrarroja se activa la señal que llega al comparador de voltaje y este envía un pulso (en estado alto o bajo, dependiendo de la entrada) al microcontrolador a través del pin etiquetado como “D0”, haciendo referencia a una lectura digital.

Figura 33. **Circuito equivalente al módulo sensor de flama**

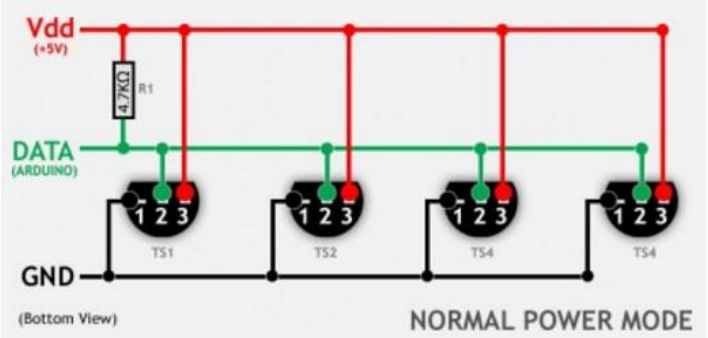


Fuente: elaboración propia, utilizando el programa Eagle.

3.3. Sensor de temperatura DS18B20

El termómetro digital, como se menciona antes, funciona a través del protocolo de comunicación 1-Wire y es posible conectar uno o múltiples sensores en la misma línea de comunicación si así se desea (ver la figura 34).

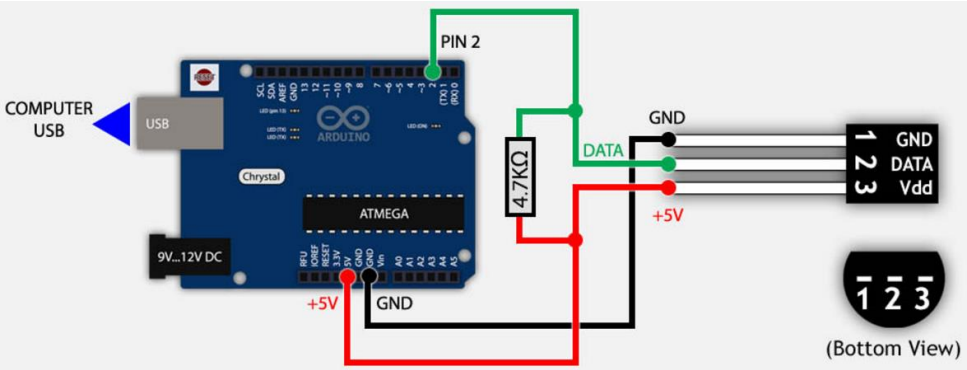
Figura 34. **Múltiples sensores DS18B20 conectados al bus de datos**



Fuente: *Sensores conectados al bus de datos.*
http://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/
Consulta: marzo de 2017.

A través del protocolo se descubre cuántos y cuáles son los dispositivos conectados al bus de datos. Es posible crear una pequeña red ubicando los sensores de forma estratégica y medir la temperatura promedio del recinto donde están montados. Cabe mencionar que el pin DATA o DQ es de tipo drenador abierto (*open-drain*) por ser MOSFET, es decir, la salida puede estar en estado bajo GND pero no en estado alto V_{DD} a menos que se conecte una resistencia *pull up* a V_{DD}. La hoja de especificaciones indica que se debe utilizar una resistencia con valor aproximado de 5 $k\Omega$. La figura 35 muestra la resistencia conectada al circuito.

Figura 35. **Ejemplo de conexión del sensor DS18B20 en Arduino**



Fuente: *Conexión del sensor DS18B20.* http://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/. Consulta: marzo de 2017.

El sensor siempre está configurado como esclavo y el microcontrolador siempre se reconoce como maestro. Se define la secuencia de transacciones como la serie de pasos que se deben seguir para obtener información del sensor DS18B20:

- Inicialización. Esta etapa consiste en descubrir el número de dispositivos que están conectados y listos para operar.
- Comando ROM (ROM *command*). Después de la etapa anterior, en esta se descubren los dispositivos conectados y de qué tipo son, como en el caso del DS18B20. Se puede llamar a un dispositivo específico entre los varios que estén en la misma red, usando código de 64 *bit* escrito en la memoria ROM de cada sensor. Existen cinco comandos ROM de 8 *bit* de longitud, el maestro para poder seleccionar una función de comando antes debe elegir el comando ROM apropiado. Los cinco comandos

ROM y sus valores hexadecimales asociados se listan a continuación:

- *Search Rom* [F0h]. Este proceso identifica los dispositivos conectados empleando todo el tiempo necesario para lograrlo; el *master* aprende todos los códigos ROM de cada sensor. Si solo un dispositivo está conectado se puede ignorar el comando y utilizar *Read Rom* en su lugar.
- *Read Rom* [33h]. Este comando sirve para leer el código ROM y evitar la búsqueda generada con el comando *Search Rom.* Solo se debe usar cuando hay un solo dispositivo esclavo conectado al bus de datos, sino una colisión de datos ocurrirá porque todos los dispositivos tratarán de transmitir su información.
- *Match Rom* [55h]. Se usa para llamar a un dispositivo específico utilizando su código ROM.
- *Skip Rom* [CCh]. Se utiliza para comunicarse con todos los dispositivos conectados al bus, sin necesidad de utilizar el código ROM.
- *Alarm Search* [ECh]. Funciona de la misma manera que *Search Rom,* con la diferencia que solo los dispositivos con la bandera *flag alarm* activada responderán. Después de cada conversión de temperatura el sensor automáticamente realiza una comparación de los niveles de temperatura con un nivel que puede ser programado, e indicar si es menor, igual o mayor que el valor de temperatura establecido; esta capacidad se denomina *alarm signaling*.

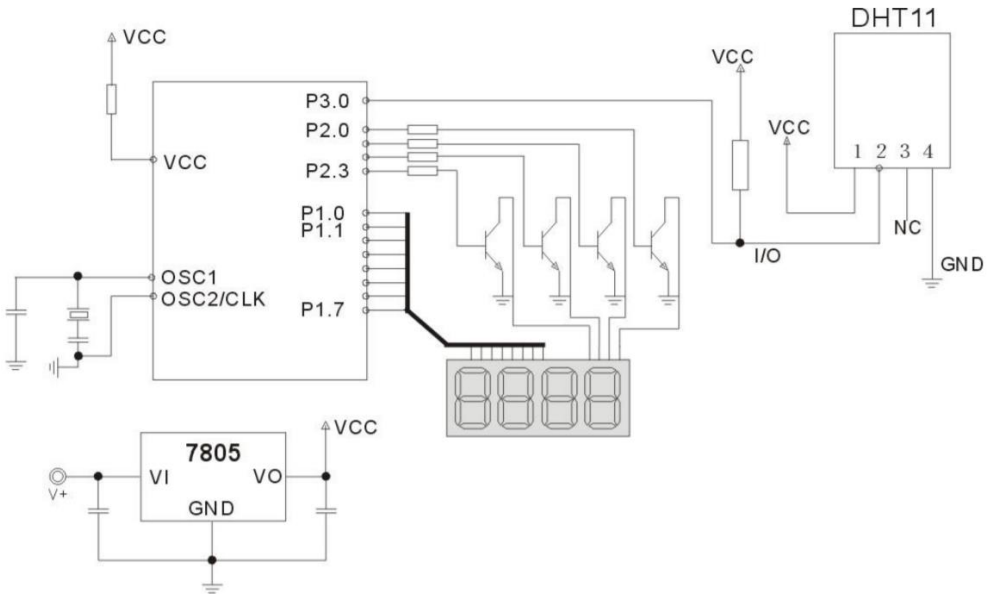
- Función de comando (*function command*). Es la última etapa y, como en el caso anterior, consta de varios comandos y sus respectivos valores en formato hexadecimales. Se muestran algunos de sus comandos a continuación:

    - *Conversion T* [44h]. Inicia el proceso de conversión de temperatura.
    - *Write Scratchpad* [4Eh]. Permite al microcontrolador (o maestro) escribir en la memoria interna del sensor (llamada *Scratchpad*) tres *bytes* de datos.
    - *Read Scratchpad* [BEh]. Permite al maestro leer el contenido de la memoria interna del sensor.
    - *Copy Scratchpad* [48h]. Permite copiar el contenido de ciertos registros en la memoria EEPROM del sensor.
    - *Read Power Supply* [B4h]. Permite conocer si algún dispositivo está utilizando la fuente de alimentación parásita.

3.4. Módulo DHT11

Un circuito de aplicación general se muestra en la figura 36 y consta de un regular de voltaje a 5 V de la serie 78XX, un microcontrolador y una serie de visualizadores (*display* de 7 segmentos) para mostrar la temperatura y/o humedad.

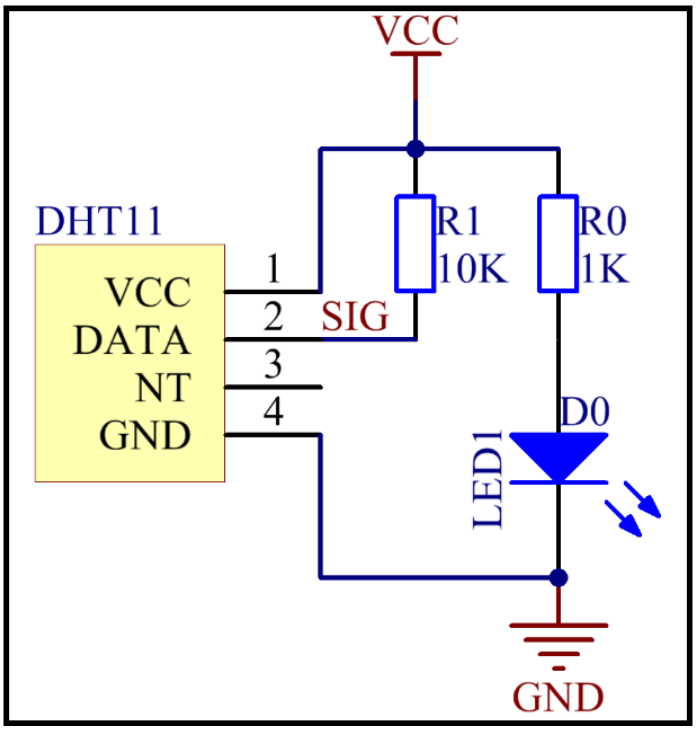
Figura 36. **Circuito típico de aplicación para el módulo DHT11**



Fuente: *Datasheet DHT11.*
https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. Consulta: marzo de 2017.

Para que el dispositivo funcione de forma adecuada la salida del pin DATA, que es de tipo drenador abierto, requiere una resistencia *pull up* de 10 kΩ aproximadamente. El módulo DHT11 que se utiliza está montado en una placa con los componentes que se muestran en la figura 37 e incluye un LED que indica que está conectado a una fuente de alimentación y operando.

Figura 37. **Circuito DHT11 con resistencia *pull up***



Fuente: *DHT con pull up.* https://www.sunfounder.com/learn/Sensor-Kit-v2-0-for-Arduino/lesson-11-humiture-detection-sensor-kit-v2-0-for-arduino.html. Consulta: marzo de 2017.

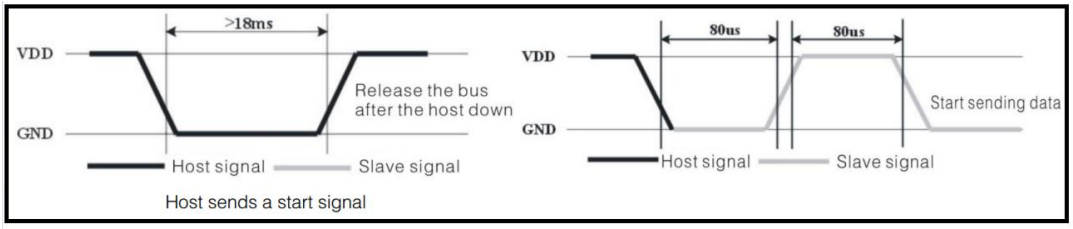
3.4.1. Explicando el protocolo de comunicación

El módulo funciona con un protocolo de comunicación definido para esa serie de dispositivos. Trata de ser de fácil implementación en cualquier microcontrolador. La hoja de datos define cómo se “lee” el dispositivo a través de sus señales de temporización. Si el módulo está apagado y se enciende, se debe esperar algunos pocos segundos para alcanzar el nivel de estabilización.

Un pin del microcontrolador se debe configurar como salida digital, seguidamente ponerse en estado bajo por al menos 18 ms; ahora el mismo pin

se debe configurar como entrada digital, la resistencia *pull up* pondrá la señal en estado alto y el módulo empieza la comunicación, colocándose 80 μs en estado bajo, seguido 80 μs en estado alto. En la figura 38, la señal *host* (*host signal*) representa al microcontrolador y la señal esclavo (*slave signal*) representa al módulo DHT11.

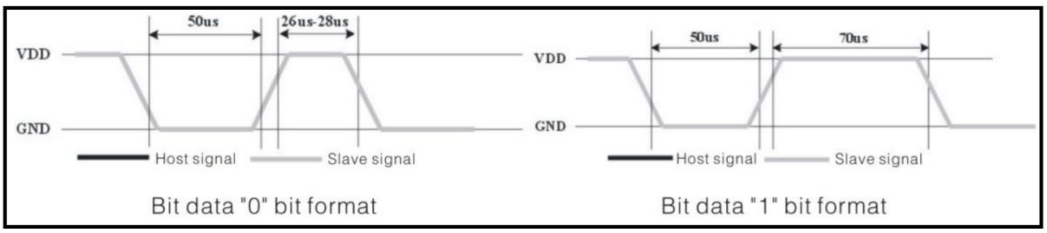
Figura 38. **Inicio de transmisión entre microcontrolador y DHT11**



Fuente: *Inicio de transmisión.* https://akizukidenshi.com/download/ds/aosong/DHT11.pdf
Consulta: marzo de 2017.

El módulo envía 40 *bit* de información al microcontrolador, la forma particular de interpretar cada *bit* es: un bit 0 comienza en un estado bajo 50 μs y un estado alto 26 a 28 μs. Un *bit* 1 comienza en un estado bajo de 50 μs y un estado alto de 70 μs (observar la figura 39).

Figura 39. **Representación de cada *bit* según el módulo DHT11**



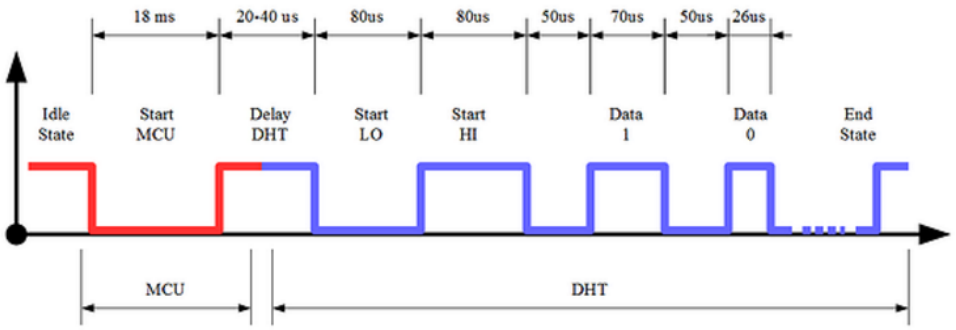
Fuente: *Bits según DHT11.* https://akizukidenshi.com/download/ds/aosong/DHT11.pdf
Consulta: marzo de 2017.

Todo el proceso de comunicación se puede resumir en cinco pasos:

- 1. En el estado desocupado (microcontrolador y módulo) la señal se mantiene en estado alto debido a la resistencia *pull up.*
- 2. El microcontrolador coloca la señal en estado bajo por lo menos 18 ms.
- 3. El microcontrolador libera el pin (se configura como entrada de alta impedancia) y la señal naturalmente se pone en estado alto.
- 4. El módulo lleva la señal 80 us en estado bajo y luego 80 us en estado alto.
- 5. El módulo transmite 40 *bit* de datos al microcontrolador, luego ambos dispositivos quedan en estado desocupado y la señal regresa a estado alto.

Un diagrama temporizado de todo el proceso que lleva el protocolo de comunicación para el módulo DHT11 se describe en la figura 40. El módulo siempre seguirá tomando muestras del ambiente y solo las enviará cuando el microcontrolador las solicite.

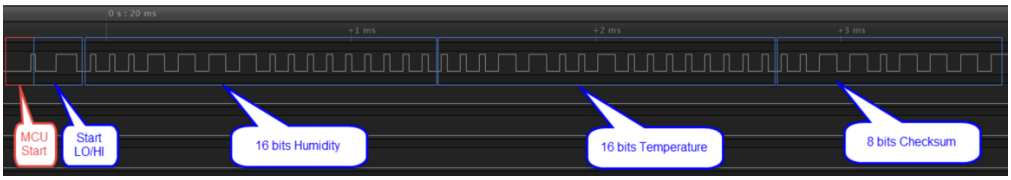
Figura 40. **Diagrama temporizado de la señales en el módulo DHT11**



Fuente: *Diagrama temporizado.* https://mcuoneclipse.com/2015/03/27/using-the-dht11-temperaturehumidity-sensor-with-a-frdm-board/. Consulta: marzo de 2017.

La transmisión de 40 bit de datos es formateada de la siguiente manera: 8 *bit* de humedad para la parte entera, 8 *bit* de humedad para la parte decimal (16 *bit* se utilizan para representar el valor de humedad); 8 *bit* de temperatura para la parte entera, 8 *bit* de temperatura para la parte decimal (16 *bit* se utilizan para representar el valor de temperatura); y los 8 *bit* sobrantes representan la comprobación de error *checksum*. La figura 41 ayuda a comprender el formato. El *bit* más significativo es enviado primero.

Figura 41. **Trama DHT11 recibida, mostrada en un analizador lógico**



Fuente: *Trama DHT11.* https://mcuoneclipse.com/2015/03/27/using-the-dht11-temperaturehumidity-sensor-with-a-frdm-board/. Consulta: marzo de 2017.

Siempre se debe realizar la comprobación de error y comparar con la comprobación de error recibida del sensor; debe dar el mismo resultado, una comprobación de error diferente es una indicación clara de que la información enviada y recibida no es igual, por lo tanto debe ser descartada y volverse a solicitar. La fórmula para realizar la comprobación de errores es: *checksum* = 8 *bit* humedad (entera) + 8 *bit* humedad (fraccional) + 8 *bit* temperatura (entera) + 8 *bit* temperatura (fraccional). Se muestra un ejemplo para realizar la comprobación de error de una medición hipotética transmitida:

00110101 + 00000000 + 00011000 + 00000000 = 0100 1101 (binario).

35h + 18h = 4Dh (hexadecimal).

53 + 24 = 77 (decimal).

3.4.2. Parámetros y condiciones de trabajo

El módulo DHT11 funciona con una fuente de alimentación 3 V a 5 V, al momento de encenderlo se debe esperar un segundo por lo menos para que alcance un nivel de estabilización y poder realizar una comunicación satisfactoria con el dispositivo. Es recomendable colocar un capacitor como filtro de fuente de alimentación entre V_{DD} y GND, lo más cerca posible de los pines de alimentación del dispositivo. El uso prolongado en condiciones de trabajo no recomendado deteriora con mayor rapidez el dispositivo.

3.4.3. Recomendaciones ante influencias externas

Evitar colocar el dispositivo en entornos de mucha condensación o muy secos durante un período de tiempo prolongado. Evadir ciertos tipos de gases como dióxido de azufre, ácido clorhídrico o cualquier otro tipo de gas ácido u oxidante. En cuanto a la concentración química de gases: la dispersión de

químicos en la capa de detección de humedad produce una disminución en la sensibilidad de la medición. En caso de que esto suceda existe un procedimiento para autocalibrar el sensor nuevamente, no obstante una alta concentración de gases por contaminación química provoca un daño permanente en la capa sensitiva del dispositivo.

Cuando se trata del proceso de restauración, si por cualquier razón el sensor estuvo expuesto por un período prolongado en ambiente de contaminantes químicos en condiciones de trabajo extremas, se puede volver a autocalibrar el dispositivo de forma acelerada, colocándolo por dos horas en un entorno de humedad seco: 10 % RH o menor a una temperatura de 45 ºC, luego llevarlo a un entorno de humedad de 70 % RH y una temperatura 20 a 30 ºC por más de cinco horas. La influencia térmica por la variación constante de temperatura debida al calor que emiten los dispositivos electrónicos, y los disipadores de calor en la misma placa donde esté montado el sensor, pueden provocar un cambio en el valor de la medición. Se recomienda colocar el sensor lo sufrientemente lejos de la influencia térmica de otros dispositivos electrónicos y montar el sensor en un placa independiente al resto de circuito que contenga disipadores de calor. Almacenar el dispositivo se recomienda bajo las siguientes condiciones:

- Temperatura 10 a 40 ºC
- Humedad 60 % RH o menor

En cuanto a la influencia de la radiación ultravioleta, la exposición prolongada a la luz solar puede degradar el dispositivo prematuramente.

3.5. Sensor de vibración

3.5.1. SW-520D: sensor de inclinación

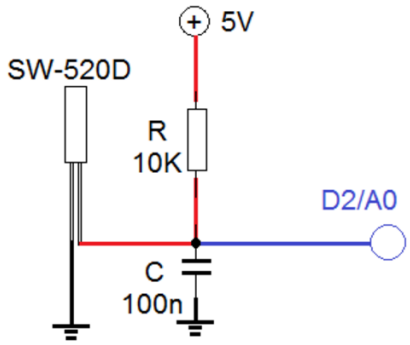
Debido al sencillo principio que utiliza el sensor de inclinación SW-520D, en la figura 42 se muestra un circuito simple que elimina el rebote producido y se puede conectar directamente al microcontrolador. El sensor también está disponible en módulo, en un circuito idéntico al de la figura mencionada antes, excepto por algunos módulos donde el capacitor no está incluido. Utilizar este sensor requiere conocer los parámetros recomendados para su funcionamiento, los cuales se muestran en la tabla IV:

Tabla IV. **Límites de operación SW-520D**

Parámetro	Especificación
El ángulo máximo de inclinación	$\pm$90º
Corriente máxima	25 mA
Voltaje máximo de operación	12 V
Temperatura máxima de operación	70 ºC

Fuente: Datasheet SW-520D.

Figura 42. **Circuito simple SW-520D con protección antirebote**

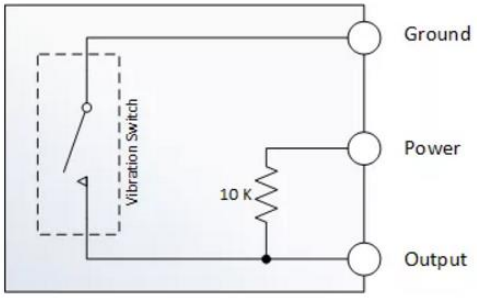


Fuente: *Circuito simple SW-520D.* http://www.electroschematics.com/12124/arduino-tilt-sensor-experiment/. Consulta: marzo de 2017.

3.5.2. Sensor de golpe

El circuito de la figura 43 muestra la forma simple de su construcción, trabaja de forma parecida a un simple interruptor, con la diferencia que se activa cada vez que se produce un golpe sobre él. Para conectar el circuito al microcontrolador se debe conectar un pin a GND y el otro hacia el pin designado del microcontrolador, añadiendo una resistencia de *pull up* para que siempre haya un estado alto o bajo. Es posible conectar de otra manera, intercambiando GND por la fuente de alimentación V_{DD} y la resistencia *pull up* por una resistencia *pull down*.

Figura 43. **Circuito equivalente de sensor de golpe**



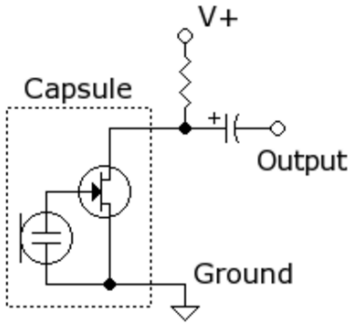
Fuente: *Circuito equivalente de sensor de golpe*. http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/ky-002-arduino-vibration-shake-sensor-manual-and-tutorial/ Consulta: marzo de 2017.

3.6. Sensor de sonido

3.6.1. Circuito para micrófono *electret*

El circuito más simple que se utiliza con el micrófono *electret* se muestra en la figura 44. Consta de una resistencia *pull up* y un capacitor de desacoplo. El problema de utilizar el circuito es que no se puede implementar en cualquier aplicación. La potencia de salida del dispositivo *electret* suele ser muy baja.

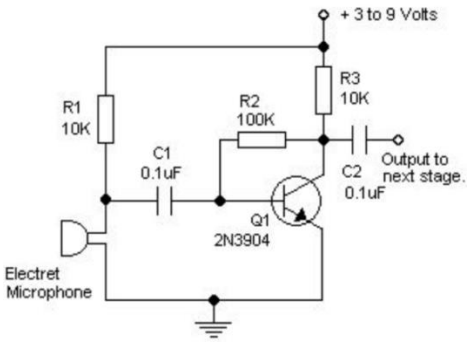
Figura 44. **Circuito *electret* básico**



Fuente: *Circuito electret básico*. http://www.eevblog.com/forum/beginners/electret-microphone-circuits/. Consulta: marzo de 2017.

Un circuito amplificador soluciona el problema, siendo el acople entre el micrófono *electret* y el circuito de aplicación. Un circuito sencillo con un preamplificador basado en un transistor de unión bipolar se muestra en la figura 45. El circuito, además, incluye resistencias, capacitores de acoplo y desacoplo.

Figura 45. **Circuito de amplificación *electret***



Fuente: *Circuito de amplificación electret.* http://www.instructables.com/id/Pre-amp-to-electret-mic/. Consulta: marzo de 2017.

3.6.2. Parámetros del micrófono *electret*

En la tabla V se describen las especificaciones, según el fabricante, del micrófono usado para realizar las mediciones de sonido. No es recomendable superar los límites máximos de operación del micrófono.

Tabla V. **Especificaciones del micrófono condensador *electret***

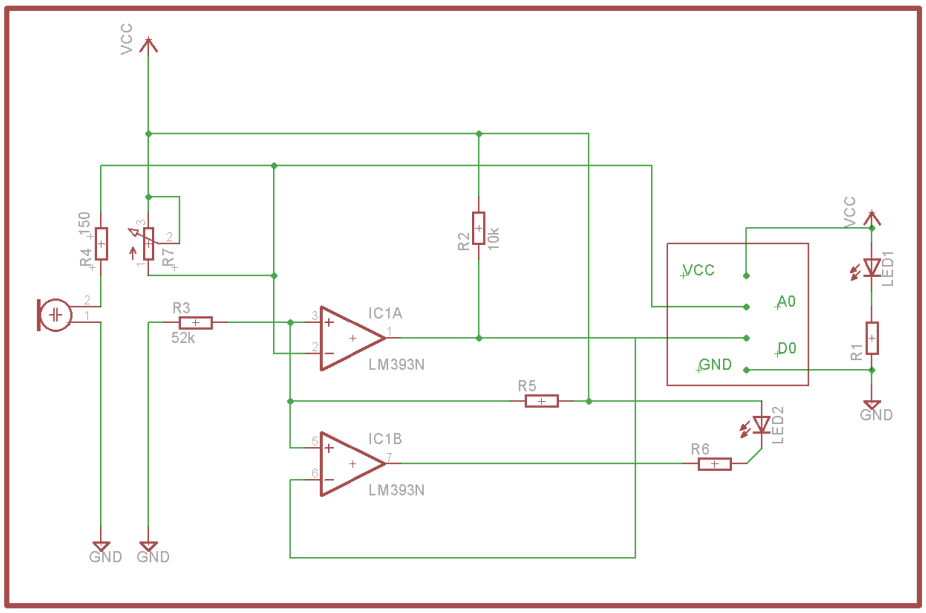
Parámetro	Valor	Descripción adicional
Directividad	Omnidireccional	
Sensibilidad	$-44 \pm 2\ dB$	
Voltaje de operación (DC)	$3\ V$ (normal), $10\ V$ (máximo)	$f = 1\ kHz, 1\ Pa$
Impedancia de salida (Z_O)	$2.2\ k\Omega$	
Frecuencia	$20 - 20\,000\ Hz$	
Consumo de corriente I_{DSS}	$0.5\ mA$ máximo	$V_S = 3\ V\ (DC)$; $R_L = 2.2\ k\Omega$
Relación señal a ruido (S/N)	$60\ dBA$	$f = 1\ kHz, 1\ Pa$
Temperatura de operación	$-20 \sim +70\ ºC$	

Fuente: *Datasheet micrófono electret.* https://tkkrlab.nl/wiki/Arduino_KY-038_Microphone_sound_sensor_module. Consulta: marzo de 2017.

3.6.3. Módulo: sensor de sonido

El módulo de sonido de la figura 19 (capítulo 1) incorpora los circuitos necesarios para conectarlo directamente al microcontrolador, tanto de forma digital como analógica. La figura 46 muestra un circuito equivalente al módulo de sonido. El comparador LM393 estabiliza la señal proveniente de la fuente, es decir, el sensor de sonido se vale del potenciómetro para establecer un nivel de referencia de salida.

Figura 46. **Circuito equivalente de módulo de sonido**



Fuente: elaboración propia, utilizando el programa Eagle.

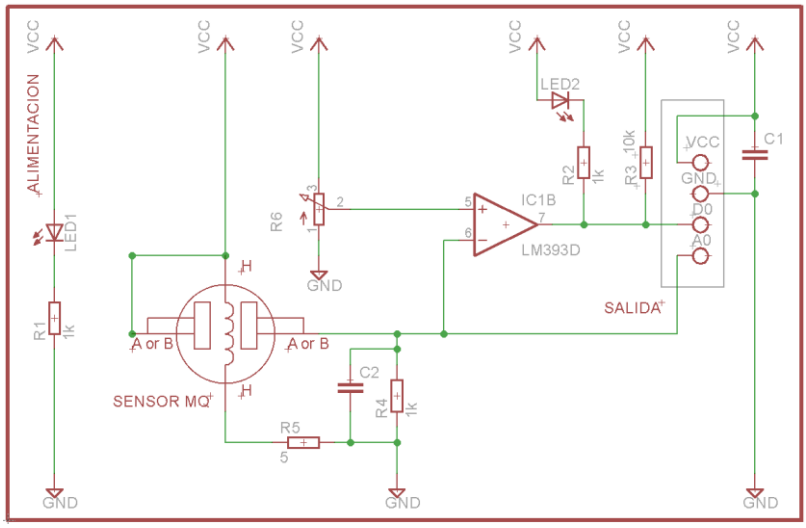
3.7. Sensor de gas MQ

Una característica de la línea de sensores MQ es la dificultad para calibrar los dispositivos. La calibración debe realizarse según el *datasheet* en presencia de una cantidad conocida del gas en cuestión, unidades mg/L o PPM, además la medición cambia si cambian algunos parámetros como la humedad y la temperatura. A menos que se cuente con un laboratorio para realizar la calibración, existe otra manera un poco más simple para determinar la cantidad de un gas: a través de correlación.

Mediante el método de regresión se estima la curva de cada sensor MQ utilizado y se obtiene la ecuación para relacionar la medición del sensor con la presencia del gas en cuestión. De esta forma los resultados de las mediciones

son aproximadas, pero suficientes para dar indicaciones de niveles peligrosos sobre ciertos gases de interés. La curva característica de cada sensor indica una relación no lineal y entre más puntos se tengan para realizar la regresión mejor aproximados serán los resultados. Un circuito equivalente para los módulos de los sensores MQ se presenta en la figura 47:

Figura 47. **Circuito equivalente de módulo MQ**

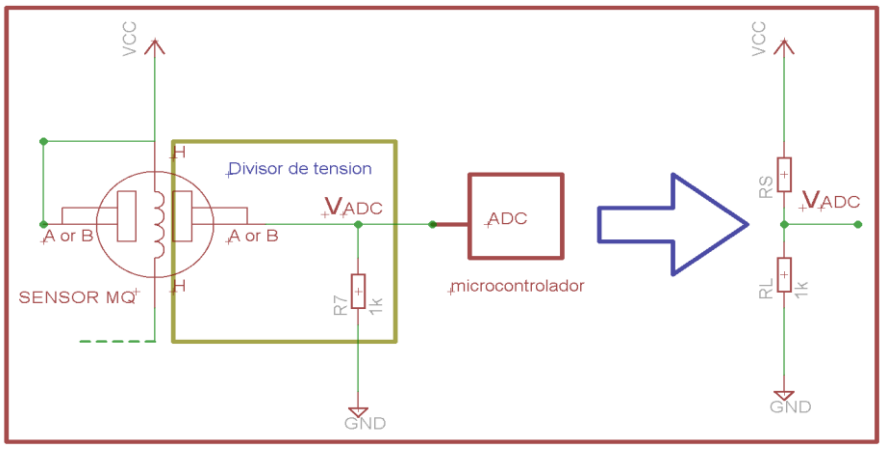


Fuente: elaboración propia, utilizando el programa Eagle.

El voltaje analógico que mide el microcontrolador es tomado del divisor de voltaje que forma la resistencia (del sensor MQ) R_s y (la resistencia de carga) R_L, que generalmente tiene un valor de 1 kΩ. La figura 48 ayuda a comprender la idea. Al variar la concentración de gas presente en el aire también variará el valor R_s (de forma no lineal) y afectará el valor del divisor de tensión V_{ADC} de la siguiente forma:

$$V_{ADC} = \frac{R_L}{R_L + R_s} V_{CC}$$

Figura 48. **Divisor de voltaje entre sensor MQ y R_L**



Fuente: elaboración propia, utilizado el programa Eagle.

3.7.1. Sensor MQ-2

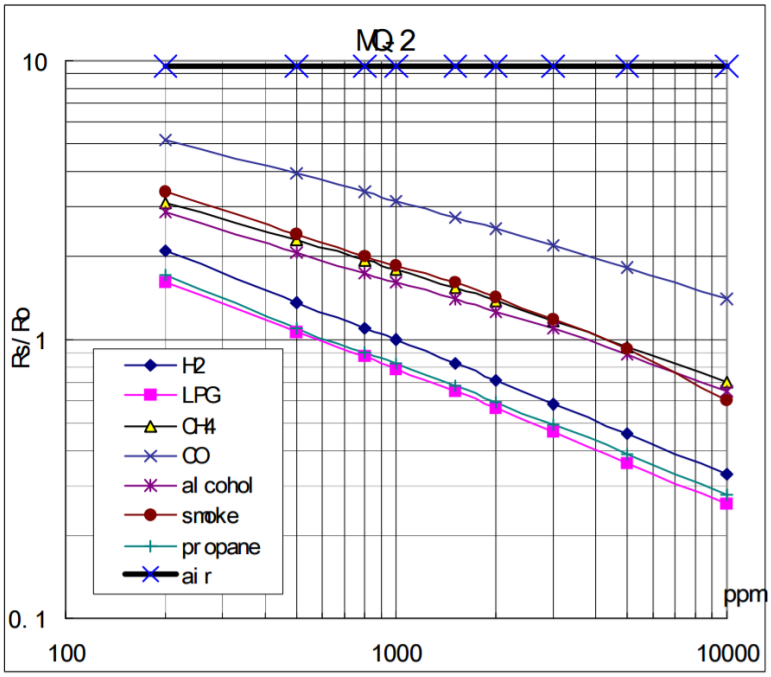
A partir de las curvas características del sensor MQ-2 que se muestra en la figura 49, se obtienen los valores para monóxido de carbono (CO) y se presentan en la tabla VI:

Tabla VI. **Valores de la curva del gas CO para el sensor MQ-2**

R_s/R_o (adimensional)	Partes por millón (PPM)
5.2	200
3.9	500
3.5	800
3.2	1 000
2.5	2 000
2.2	3 000
1.8	5 000
1.5	10 000

Fuente: elaboración propia.

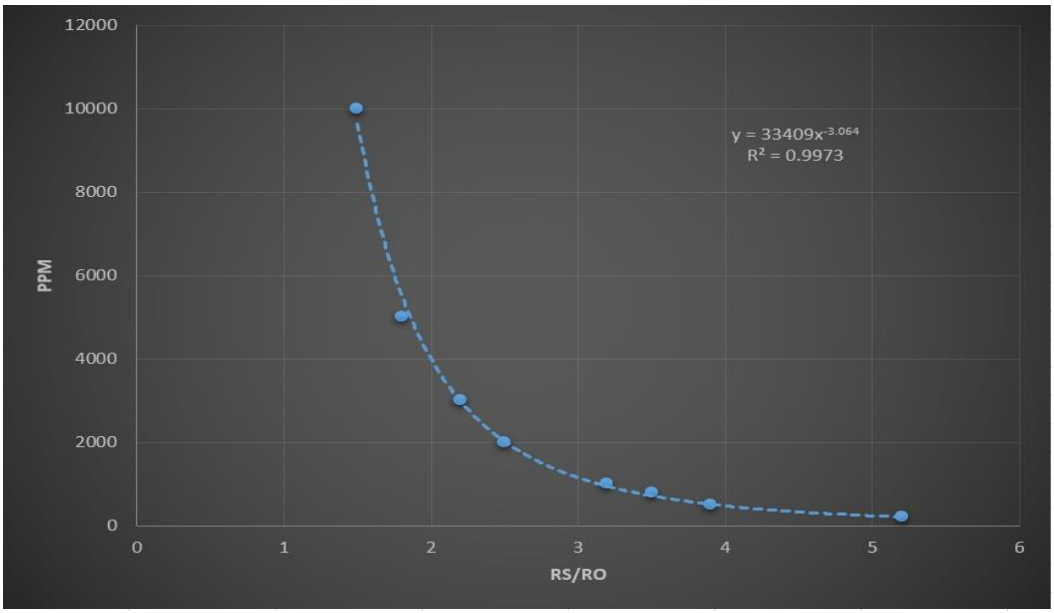
Figura 49. **Curvas características para el sensor MQ-2**



Fuente: *Sensor MQ-2 datasheet.*
http://sandboxelectronics.com/files/SEN-000004/MQ-2.pdf. Consulta: abril de 2017.

Con la información resultante se construye una regresión potencial, la cual es la que mejor se ajusta a los valores en la gráfica, para así obtener la ecuación que relaciona las variables de interés. Para realizar la gráfica de la figura 50 se hace la sustitución $y = PPM$ y $x = \left(\frac{R_S}{R_o}\right)$.

Figura 50. **Regresión potencial para el gas CO del sensor MQ-2**



Fuente: elaboración propia, utilizado el programa Excel.

La ecuación es:

$$PPM = 33\,409\left(\frac{R_s}{R_o}\right)^{-0.3064}$$

Con un coeficiente de determinación:

$$R^2 = 0.9973$$

El único valor que no se conoce es R_o y, según la definición, se debe encontrar bajo ciertos parámetros ambientales. No obstante, una forma muy aproximada de encontrar un valor teórico para R_o es utilizar la figura 50 y observar que los valores que saturan el sensor son cercanos a los 10 000 PPM, además los cambios grandes del eje vertical PPM producen cambios pequeños

en el eje horizontal R_s/R_o, por lo que el error se reduce. Entonces se expone al sensor MQ-2 a una concentración grande de CO, se despeja la ecuación para R_o y se obtiene un valor aproximado:

$$R_o = \left.\frac{R_s}{\left(\frac{PPM}{33\ 409}\right)^{-3.064}}\right|_{PPM \to 10\ 000}$$

Para obtener R_s es necesario observa el circuito de la figura 48, se despeja el divisor de voltaje para R_s y se obtiene:

$$R_s = R_L\left(\frac{V_{cc}}{V_{ADC}} - 1\right) = 1k\Omega\left(\frac{5V}{V_{ADC}} - 1\right)$$

Donde V_{ADC} es el voltaje obtenido del conversor analógico digital del microcontrolador, $V_{cc} = 5\,V$ y $R_L = 1\,k\Omega$ son el voltaje de alimentación y la resistencia (que junto a R_s hacen el divisor de tensión), respectivamente, del módulo que contiene al sensor. Para un convertidor analógico digital con resolución de 10 *bit*, V_{ADC} se obtiene:

$$V_{ADC} = \frac{V_{CC}}{2^{10} - 1}(ADC) = \frac{5\,V}{1023}(ADC)$$

Donde el valor ADC representa el voltaje cuantizado del convertidor analógico digital entre 0 y 1023 (2^{10}-1), es decir 1024 niveles de cuantización.

3.7.2. Parámetros del sensor MQ-2

Para utilizar el dispositivo de forma correcta es necesario conocer y aplicar las condiciones de trabajo necesarias. La tabla VII muestra las especificaciones necesarias para poner en funcionamiento el sensor:

Tabla VII. **Especificaciones de sensor MQ-2**

Símbolo	Parámetro	Condición	Comentarios
V_C	Voltaje del circuito	5.0 V	AC o DC
V_H	Voltaje del calentador	5.0 V	AC o DC
R_L	Resistencia de carga	Ajustable	
R_H	Resistencia del calentador	33.0 Ω	

Fuente: Hanwei Electronics.*MQ-2 datasheet.*
http://sandboxelectronics.com/files/SEN-000004/MQ-2.pdf. Consulta: abril de 2017.

Las características sensitivas de la tabla VIII son necesarias si se desea implementar cálculos, como en el análisis previo. La tabla siguiente muestra el rango en el que varía la resistencia interna del sensor:

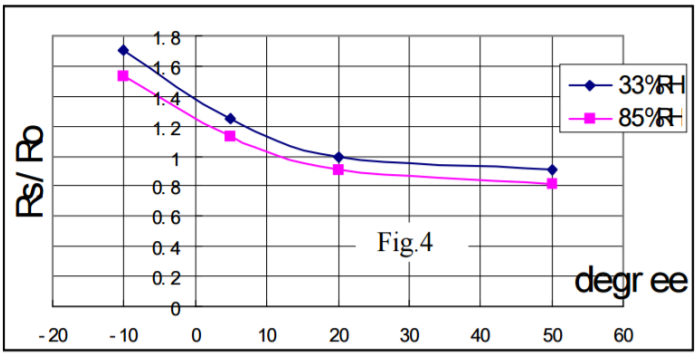
Tabla VIII. **Características sensitivas de sensor MQ-2**

Símbolo	Parámetro	Valor	Comentarios
R_s	Resistencia interna del sensor	$3k - 30k\ \Omega$	Alcance de detección de concentración:
Condición de detección	Temperatura: 20ºC Humedad: 65% V_c=V_H=5 V		
Tiempo de precalentado	24 horas aproximadas		

Fuente: Hanwei Electronics.*MQ-2 datasheet.* http://sandboxelectronics.com/files/SEN-000004/MQ-2.pdf. Consulta: abril de 2017.

Por último, se muestra en la figura siguiente la dependencia que muestra el sensor MQ-2 a dos parámetros ambientales: temperatura y humedad. El rango más estable se determina entre 20º C a 50º C, porque se observa una pendiente más constante respecto al resto de la curva.

Figura 51. **Dependencia de parámetros ambientales del sensor MQ-2**



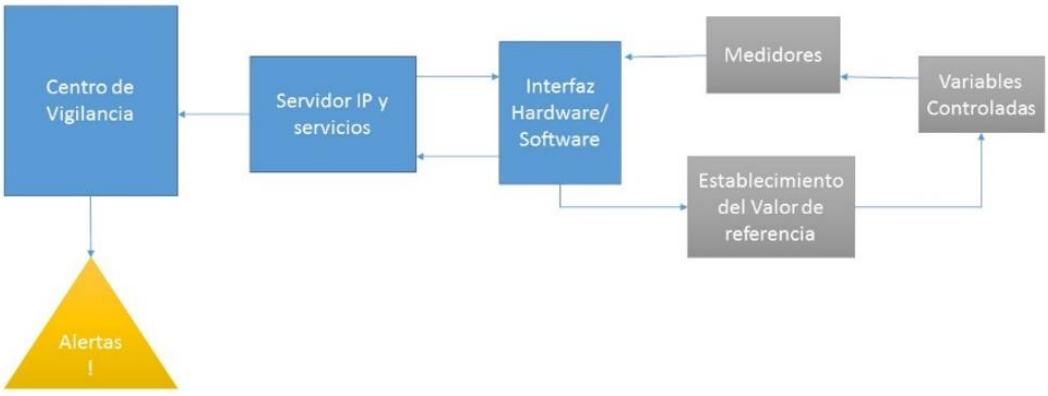
Fuente: Hanwei Electronics.*MQ-2 datasheet.* http://sandboxelectronics.com/files/SEN-000004/MQ-2.pdf. Consulta: mayo de 2017.

4. DISEÑO DE UN SISTEMA DE MONITOREO

El modelo de diseño para el sistema de monitoreo y control se basa en la medición continua de las variables de interés y en la comparación de las mediciones obtenidas con los valores de referencia que no se deben superar.

Para lograr el objetivo se utiliza un centro de vigilancia que tenga la capacidad de mostrar alertas cuando los niveles críticos ocurran en las mediciones. El centro de vigilancia está conectado a una red de área local para que cualquier dispositivo de red que ejecute un navegador *web* pueda utilizarse para visualizar la información. Los sensores son los dispositivos encargados de realizar las mediciones constantes de las variables de interés. Entre los sensores y el centro de control existe una interfaz encargada de tomar todas las muestras y enviarlas. La figura 52 ilustra el modelo propuesto de diseño:

Figura 52. **Propuesta de sistema de monitoreo**



Fuente: elaboración propia.

4.1. Implementación del diseño de hardware

El centro de vigilancia/control está basado en una computadora Raspberry Pi 2, configurada como servidor que brinda todos los servicios necesarios tanto para controlar el sistema completo como para presentar la información visual de forma agradable. Está claro que las mediciones se obtienen a partir de los sensores y la configuración de los niveles de referencia la establece el usuario a través de la interfaz *web*. Una base de datos almacena todas las mediciones realizadas para mantener un récord de la vigilancia. Se seleccionan las variables y sensores para tomar muestras del ambiente. La tabla IX indica las variables seleccionadas y los sensores con que se obtienen los valores.

Tabla IX. **Variables y sensores establecidos**

Variable ambiental	Sensor seleccionado
Temperatura	DS18B20 / DHT11
Humedad	DHT11
Sonido	Micrófono Electret
Vibración	Sensor de inclinación Tilt
Gas	Sensor de Gas MQ-2

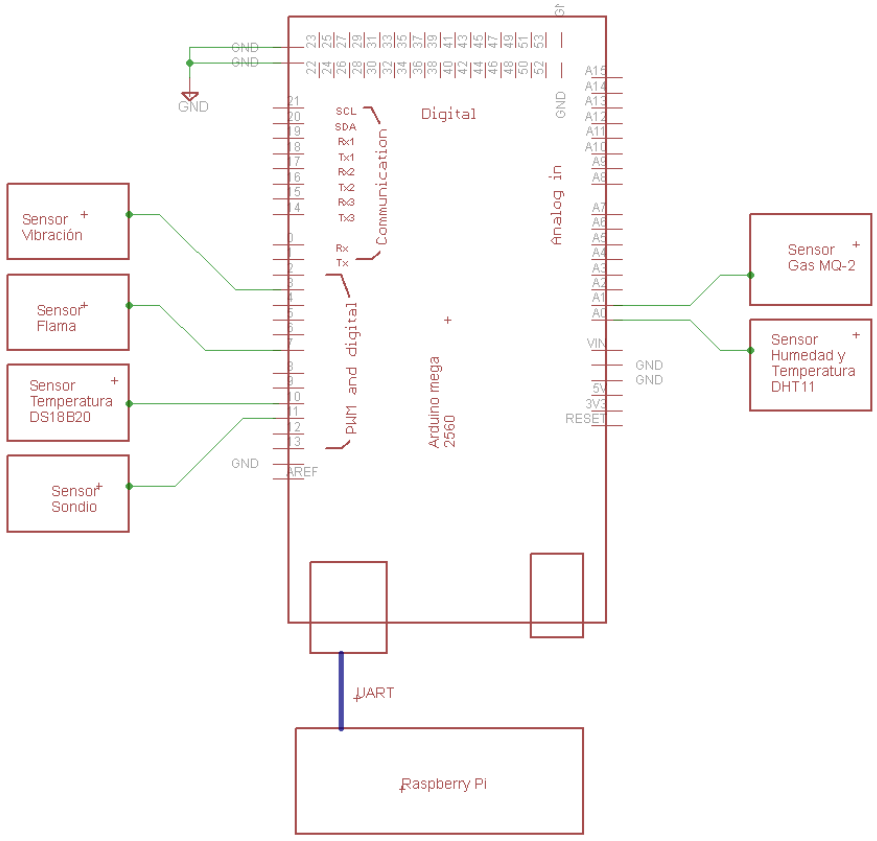
Fuente: elaboración propia.

El sistema que mantiene en ejecución el proceso de medición dedicado es el microcontrolador Arduino, que actúa como la interfaz descrita más arriba.

Arduino seguirá tomando las muestras de los sensores indefinidamente y la Raspberry Pi solicitará la información de forma periódica, analizará y procesará cada dato recibido y lo almacenará en una base de datos. También se activará un servicio de transmisión de video y este conjunto de información será mostrado en una página *web* accesible para toda la red alcanzable.

Resulta oportuno indicar que debe existir una buena comunicación bidireccional entre Arduino y Raspberry Pi. Para lograrlo se utiliza el protocolo de comunicación serial UART, disponible en ambos dispositivos de forma embebida. La figura 53 muestra la interconexión en los circuitos descritos:

Figura 53. **Diagrama de interconexión entre los circuitos**



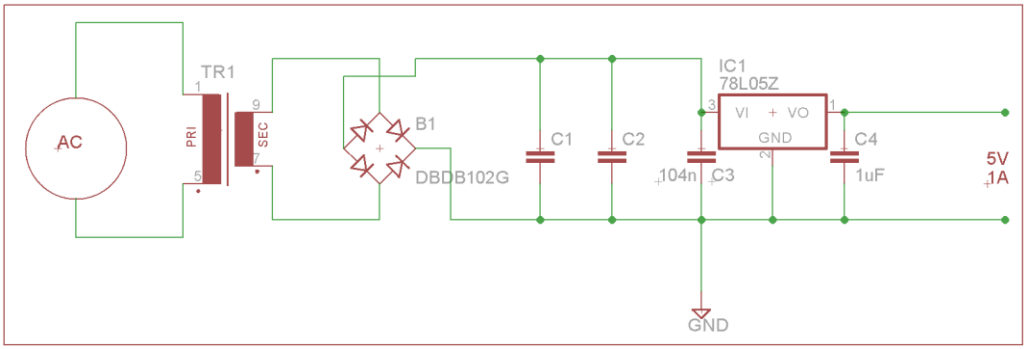
Fuente: elaboración propia.

4.1.1. Fuente de alimentación externa para alimentar los sensores

Se utiliza una fuente de alimentación externa para alimentar los sensores, la razón principal de la elección es que, en conjunto, los sensores pueden

consumir más energía de la que puede entregar Arduino, lo que deriva en una posible inestabilidad del microcontrolador. Para evitar ese problema la fuente de alimentación suministra la energía suficiente. De hecho, cualquier fuente de alimentación regulada de 5 V de 1 A es suficiente. La figura 54 muestra un circuito típico para una fuente de voltaje utilizando un regulador LM7805.

Figura 54. **Fuente de alimentación externa para los sensores**



Fuente: elaboración propia, utilizando el programa Eagle.

4.1.2. Diseño de sistema sonoro al superar el nivel de referencia

Otro mecanismo de prevención de niveles críticos es la utilización de una fuente sonora que emita un sonido mientras niveles por encima del nivel de referencia se alcancen. Un *buzzer* piezoeléctrico es un transductor electroacústico utilizado como mecanismo de señalización. Los *buzzer* no necesitan un circuito extra para funcionar, son capaces de emitir sonido al conectarlo en un pin del microcontrolador que emita ciertas frecuencias.

4.2. Implementación del diseño de software

4.2.1. Implementación de los niveles de referencia

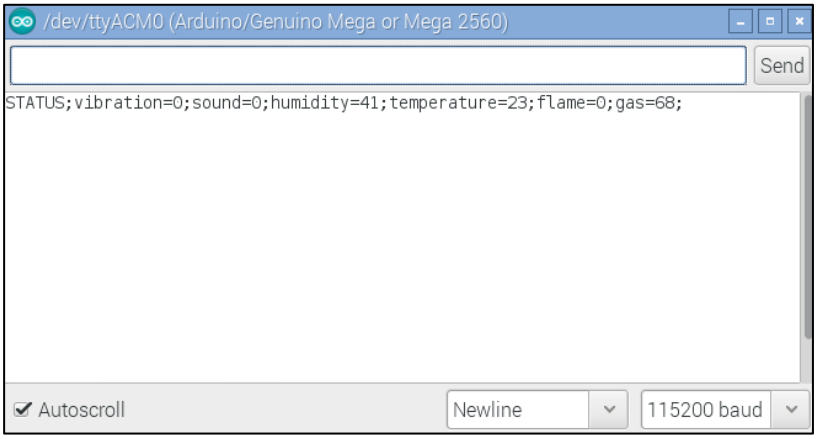
Se definen los niveles de referencia como *setPoints* para los sensores que van a ser monitoreados. Los niveles de referencia son los niveles establecidos por el usuario y representan los niveles límites del valor de un sensor.

4.2.2. Desarrollo de comandos para la comunicación bidireccional

Como se mencionó anteriormente, el protocolo de comunicación entre Raspberry Pi y Arduino es UART, a una velocidad de transmisión de 115200 *baudios*. Para lograr una comunicación bidireccional se definen varios comandos para ejecutar operaciones específicas:

- *STATUS*: se utiliza para obtener los valores de todos los sensores conectados al microcontrolador en el siguiente orden: vibración, sonido, humedad, temperatura, flama y gas. El microcontrolador, al recibir el comando *STATUS**n* a través de UART, enviará de vuelta la recopilación actual de los valores de cada sensor en el orden especificado. La figura 55 utiliza el monitor serial de Arduino para simular la acción del comando *STATUS*:

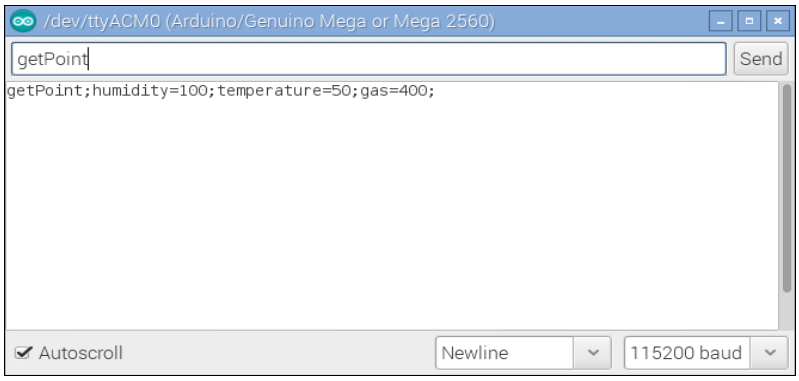
Figura 55. **Muestra del comando *STATUS* en un monitor serial**



Fuente: elaboración propia.

- *getPoint*: el comando se utiliza para obtener el valor actual de los niveles de referencia que tiene Arduino. Los valores de referencia son para el sensor de humedad, sensor de temperatura y sensor de gas. El microcontrolador, al recibir el comando *getPoint\n* a través de UART, enviará de vuelta la recopilación actual de los valores de referencia de los sensores en el orden especificado. La figura 56 utiliza el monitor serial de Arduino para simular la acción del comando *getPoint*:

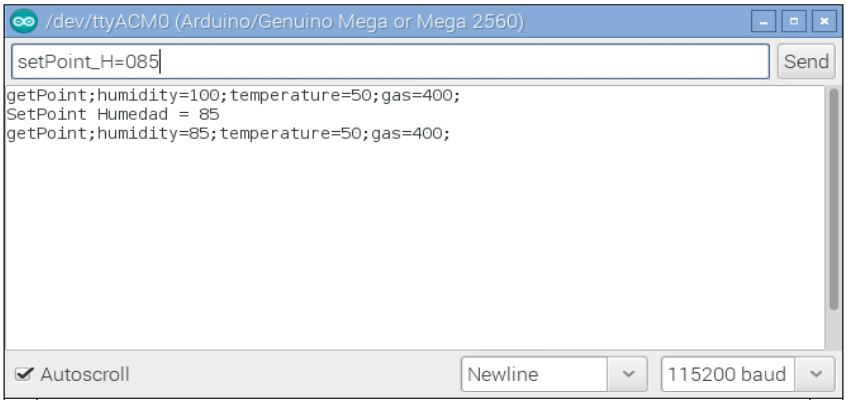
Figura 56. **Muestra del comando *getPoint* en un monitor serial**



Fuente: elaboración propia.

- *setPoint_H*: el comando ordena a Arduino cambiar el valor de referencia de humedad por uno definido por el usuario, por ejemplo para cambiar el valor de referencia de humedad a 85 %, Arduino debe recibir *setPoint_H=085\n*, donde el formato de los números es siempre de 3 cifras. En la figura 57 se muestra un ejemplo:

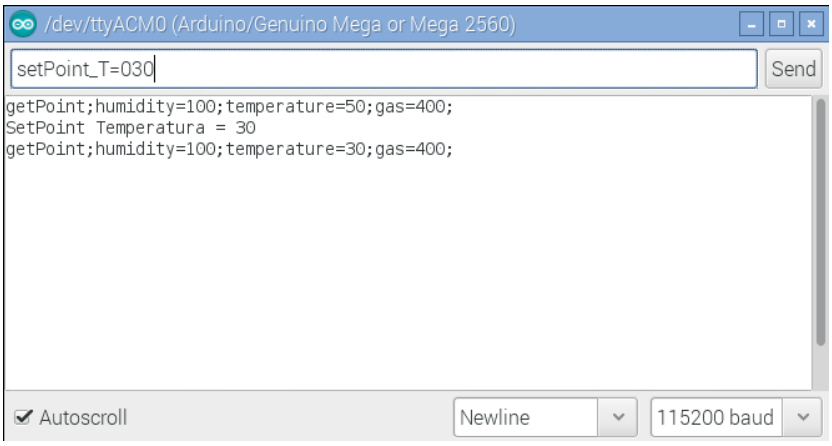
Figura 57. **Muestra del comando *setPoint_H* en un monitor serial**



Fuente: elaboración propia.

- *setPoint_T*: el comando ordena a Arduino cambiar el valor de referencia de temperatura por uno definido por el usuario, por ejemplo, para cambiar el valor de referencia de temperatura a 30 ºC Arduino debe recibir *setPoint_T=030\n*, donde el formato de los números es siempre de 3 cifras. En la figura 58 se muestra un ejemplo:

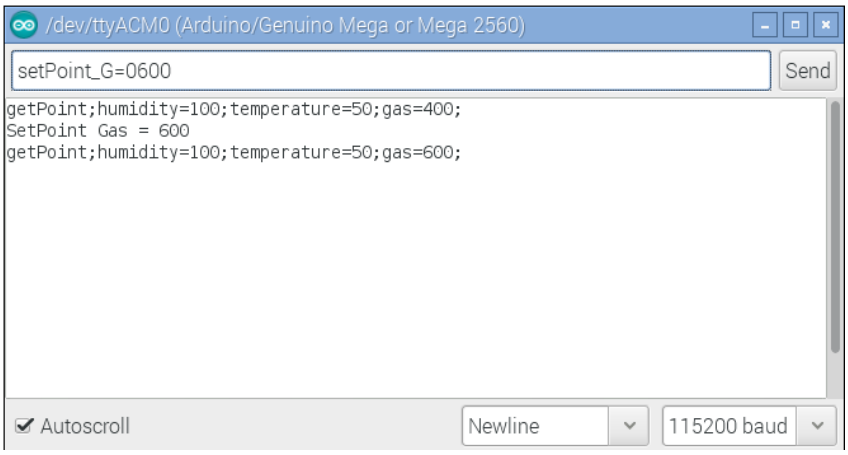
Figura 58. **Muestra del comando *setPoint_T* en un monitor serial**



Fuente: elaboración propia.

- *setPoint_G*: el comando ordena a Arduino cambiar el valor de referencia del gas por uno definido por el usuario, por ejemplo, para cambiar el valor de referencia de gas a 600, Arduino debe recibir *setPoint_G=0600\n"* donde el formato de los números es siempre de 4 cifras. En la figura 59 se muestra un ejemplo:

Figura 59. **Muestra del comando *setPoint_G* en un monitor serial**



Fuente: elaboración propia.

4.2.3. Definición del tiempo de adquisición de muestras

Es necesario fijar un tiempo para adquirir las muestras de los sensores y guardarlos de forma permanente. Para evitar tener una cantidad excesiva de datos en un corto período de tiempo se establece un tiempo entre cada medición llamado “tiempo de muestreo”, por defecto de un minuto.

4.2.4. Lógica de programación del microcontrolador Arduino

El microcontrolador tiene dos tareas dedicadas a realizar en ciertos instantes de tiempo:

- Mantener el control sobre todos los sensores conectados.
- Recibir órdenes mediante comandos y responder solicitudes utilizando el protocolo de comunicación serial.

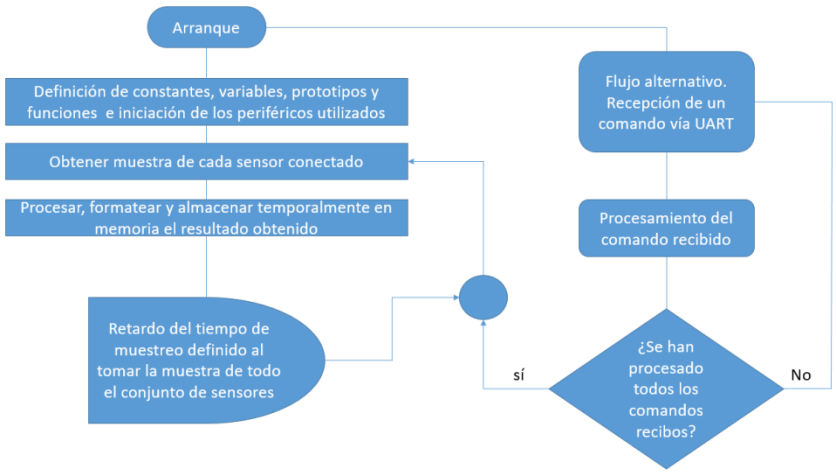
Por las dos consideraciones anteriores, se realiza un enfoque de programación para lograr ambos objetivos. Mantener el control sobre los sensores implica ir revisando cada sensor conectado al microcontrolador y almacenar temporalmente los valores obtenidos en un espacio de memoria. Ese espacio de memoria se modifica cada vez que se obtengan nuevas mediciones.

Si la medición de un sensor no se puede obtener correctamente, se establece en su lugar una condición de “ERROR_CONDITION = -101”, que implica que no existe una medición para el sensor por algún error en su lectura, debido a que el programa siempre espera un valor definido para una medición, de manera tal que, si no se obtiene un valor específico, la ejecución del programa puede terminar inesperadamente y producir efectos indeseados.

Por otro lado, recibir órdenes mediante comandos y responder solicitudes implica la comunicación bidireccional, por un lado Raspberry Pi envía órdenes específicas y por otro Arduino debe procesarlas y responderlas. Los dos objetivos clave de mantener control de los sensores y la comunicación bidireccional requieren definir el flujo del programa que se ejecuta en Arduino.

De los anteriores planteamientos se deduce la ejecución del programa, que consiste en repetir de forma indefinida la porción de código que controla los sensores y activar una interrupción cada vez que se reciba información mediante UART. La interrupción cambia el flujo del programa para leer el comando recibido y ejecutar la orden, luego retornando al flujo anterior. Una aproximación al flujo del programa se muestra en la figura 60:

Figura 60. **Aproximación al flujo del programa del microcontrolador**



Fuente: elaboración propia.

4.2.5. Lógica de programación del sistema de control Raspberry Pi

El sistema de control Raspberry Pi se divide en dos etapas, una es responsable de interactuar con el microcontrolador y la otra es la interfaz de interacción con el usuario. La primera etapa está diseñada para comunicarse con el microcontrolador y recibir la medición de los sensores que previamente están almacenados temporalmente en Arduino. Es necesario procesar y formatear los datos recibidos para guardarlos permanentemente en la base de datos MySQL. Se usa Python para controlar esta etapa. Las principales tareas son:

- Enviar comandos a Arduino.
- Recibir las mediciones de los sensores.
- Leer los niveles de referencia de Arduino.
- Cambiar los niveles de referencia de Arduino.

- Almacenar la información en una base de datos.
- Leer o escribir los niveles de referencia en un archivo de configuración.

La segunda etapa es responsable de presentar la información. Se programa en PHP. Las principales tareas son:

- Crear la interfaz *web.*
- Mostrar los niveles de referencia actuales.
- Cambiar los niveles de referencia por los que establece el usuario.
- Mostrar las gráficas de las mediciones de los sensores.
- Mostrar alertas en los niveles de las mediciones elevadas.
- Mostrar la transmisión de video en tiempo real.

Al final, se observa un pequeño problema, al notar que, cuando se usan dos lenguajes de programación, ¿cómo hacer que se comuniquen entre sí?

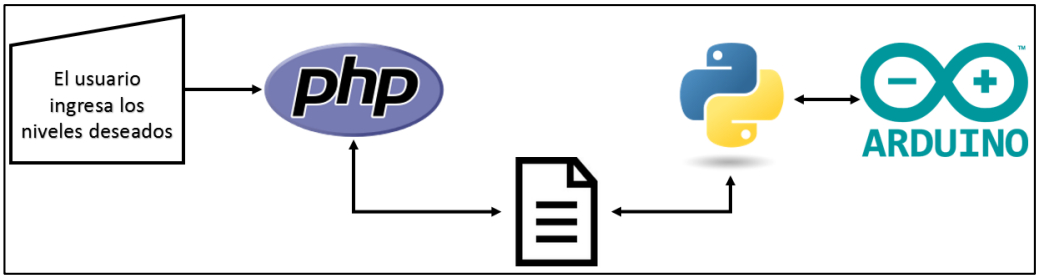
4.2.6. Enfoque de intercomunicación Python y PHP

Existen varias maneras de compartir información entre ambos lenguajes de programación, PHP y Python; por ejemplo, utilizando *sockets*, pero requiere un poco más de trabajo. Un enfoque de comunicación simple es el manejo de archivos (*file handling*) y consiste en crear un archivo de configuración que contenga la información que se desea compartir, en este caso son los niveles de referencia. PHP escribe los valores y Python los lee y establece el Arduino. El nombre que recibe el archivo no es de mayor importancia pero se debe definir uno y por conveniencia se llama *config-file.txt*, almacenado en el directorio /*home/pi/config-file.txt*.

Adicionalmente, se debe definir el contenido interior del archivo de configuración, que consiste en usar el nombre del parámetro ambiental seguido del signo igual = y el valor numérico que representa el nivel de referencia máximo. Por ejemplo: *humidity=70 temperature=50 gas=600*. Para que ambos lenguajes de programación entiendan cuál parámetro están leyendo se utiliza una expresión regular para encontrar los valores. Por ejemplo, en Python una búsqueda para la humedad específicamente se realiza *humidity=\d+*, donde *\d+* significa buscar una coincidencia para uno o más números. En el caso de PHP, para encontrar todos los valores de números en una sola expresión regular se utiliza: */\d+/i*.

La figura 61 ilustra el proceso para realizar la comunicación bidireccional entre Python y PHP, donde la información que comparten son los niveles de referencia máximos. El proceso inicia en PHP, cuando el usuario ingresa los valores límite, y finaliza en Arduino, que monitorea estos niveles constantemente.

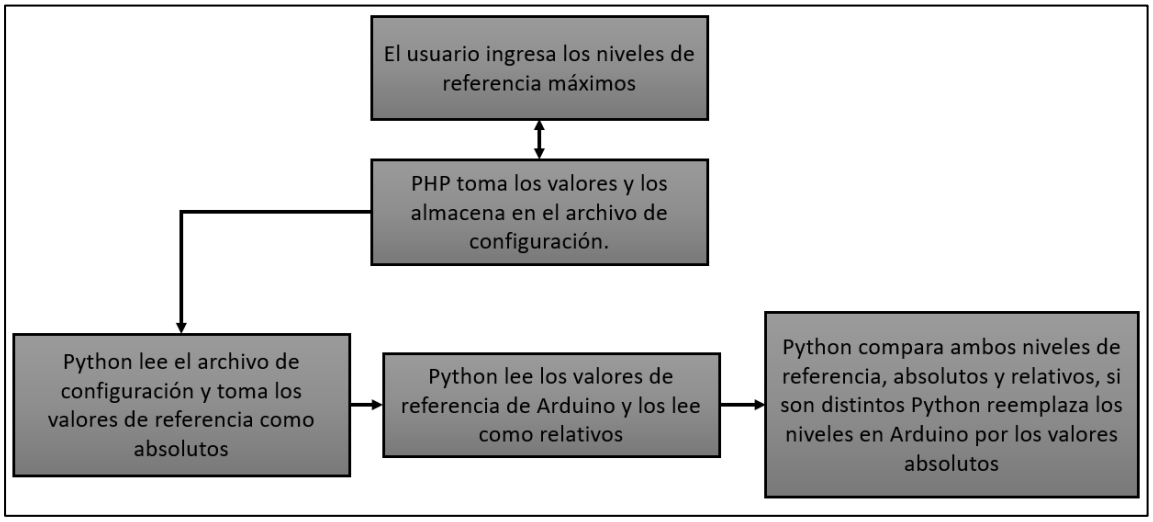
Figura 61. **Enfoque de comunicación Python-PHP**



Fuente: elaboración propia, utilizando los logos oficiales. Python. https://www.python.org/. PHP: http://php.net/. Arduino: http://arduino.cc

Con el propósito de comprender el proceso, el diagrama de bloques de la figura 62 ilustra la secuencia que establece los niveles de referencia en el sistema completo:

Figura 62. **Niveles de referencia en el sistema**



Fuente: elaboración propia.

Para que PHP y Python puedan escribir el archivo deben tener permisos de escritura: *chmod 777 config-file.txt*.

4.2.7. Implementación de un sistema dedicado a almacenar la información

Se utiliza la base de datos relacional MySQL para almacenar las muestras que se toman cada período de adquisición de muestras, para mantener un registro del comportamiento de ambiente vigilado que se plasma en una gráfica en el tiempo de los valores de los sensores. La potencia de la base de datos permite realizar consultas al mismo tiempo desde múltiples dispositivos si así se

desea. ¿Por qué limitarse a utilizar un dispositivo cuando el sistema permite que en varios se pueda mostrar la información al mismo tiempo, por uno o más usuarios interesados en el monitoreo?

4.2.8. Desarrollo de una interfaz *web*

Una de las mejores maneras de mostrar resultados es a través de una interfaz *web*, la razón principal es que es multiplataforma y no se depende de ninguna plataforma de hardware o software específico, sabiendo que múltiples dispositivos pueden conectarse al sistema de control y vigilar el recinto al mismo tiempo. Es posible usar cualquier dispositivo compatible con un navegador *web*, además ayuda a reducir costos al no tener que gastar en equipos especiales, sino más bien a reutilizar los dispositivos que se poseen.

Para el desarrollo de la interfaz gráfica *web*, se usan las herramientas PHP, HTML y CSS; las dos últimas son para crear la interfaz gráfica y darle un estilo agradable a la vista del usuario que vigila el sistema. PHP se usa para controlar el sistema *web* y dar lógica al sistema. La figura 63 muestra la interfaz más simple que se puede construir para presentar la información.

Figura 63. **Interfaz *web* simple**

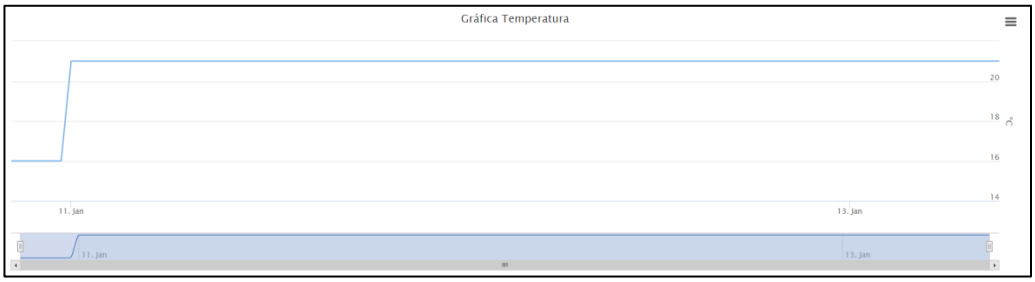
Fuente: elaboración propia.

4.2.9. Implementación de gráficos de los valores en el tiempo

Las gráficas que se elaboran son las de los sensores para los que se definió un nivel de referencia, por lo tanto, existen gráficas para la humedad, la temperatura y el gas. Se utiliza la herramienta HighChart, que permite realizar gráficas utilizando el lenguaje de programación JavaScript en un pequeño bloque de código. Los valores a graficar se obtienen de la base de datos que previamente contiene toda la información disponible. El criterio para realizar gráficas es tomar los últimos cien datos de cada sensor. Resulta oportuno

mantener un registro visual de las mediciones de los sensores, para mantener un seguimiento a través del tiempo y observar el comportamiento de las variables ambientales, esa es la principal razón de agregar las gráficas al sistema de control. En la figura 64 se muestra un ejemplo de la gráfica de temperatura constante:

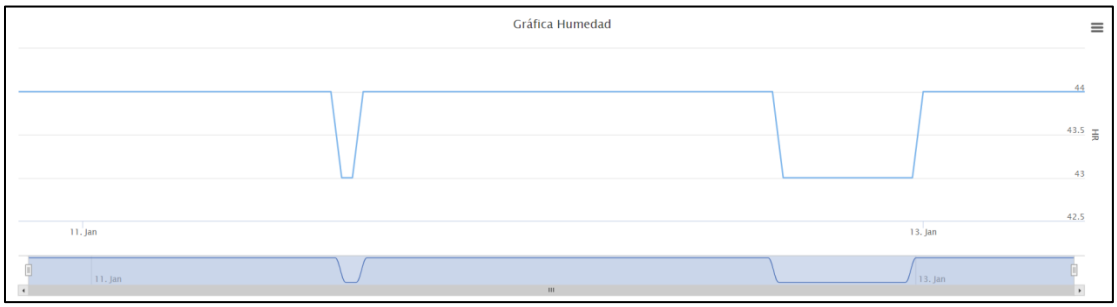
Figura 64. **Gráfica en el tiempo de la temperatura en la interfaz *web***



Fuente: elaboración propia.

La figura 65 muestra el comportamiento constante que tiene la humedad, las caídas bruscas en este ejemplo se generaron debido a que se apagó el sensor en ciertos instantes de tiempo para mostrar la variación que pueda existir en un momento dado.

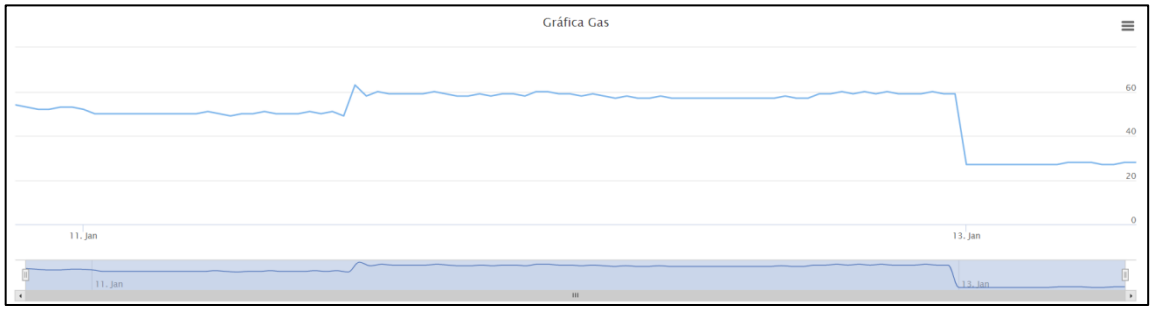
Figura 65. **Gráfica en el tiempo de la humedad en la interfaz *web***



Fuente: elaboración propia.

La figura 66 ejemplifica una medición de monóxido de carbono, en que los niveles en el eje vertical pueden variar de 0 a 1023:

Figura 66. **Gráfica en el tiempo del gas CO en la interfaz *web***



Fuente: elaboración propia.

CONCLUSIONES

1. Se determinó que los dispositivos económicos presentados en este trabajo pueden ser usados en el desarrollo del sistema de monitoreo bajo la región de operación establecida.

2. Utilizando un pequeño programa en el microcontrolador Arduino se logró mantener una monitorización constante del recinto.

3. A través de la creación de una interfaz intermedia entre microcontrolador y servidor de red, el sistema de monitoreo puede realizar una comunicación bidireccional para poder compartir la información entre ambos.

4. La creación de una interfaz *web* permite utilizar el sistema para monitorear a través de múltiples dispositivos tales como computadoras, teléfonos inteligentes y similares.

RECOMENDACIONES

1. Para la persona que vigila utilizando el sistema de monitoreo: es importante tener en cuenta que es un dispositivo de seguridad que ayudará a prevenir problemas suscitados, no obstante, la responsabilidad principal siempre será la del vigilante humano o el sistema de seguridad principal del recinto.

2. Para la persona que construye el sistema de monitoreo: el sistema puede ser modificado a conveniencia. Agregar o restar sensores siempre implica modificar un poco del programa de control.

3. Tomar en cuenta el consumo que el conjunto de sensores necesita.

4. Leer la hoja de especificaciones del fabricante del dispositivo utilizado.

BIBLIOGRAFÍA

1. ARDUINO*. Arduino*. [en línea]. <https://www.arduino.cc>. [Consulta: diciembre de 2016].

2. _____. *A DHT11 Class for Arduino*. [en línea]. <http://playground.arduino.cc/Main/DHT11Lib>. [Consulta: mayo de 2017].

3. _____*. MQ Gas sensors.* [en línea]. <http://playground.arduino.cc/Main/MQGasSensors>. [Consulta: mayo de 2017].

4. BAILIN ELECTRONICS. *SW-520 Roll Ball Switch Datasheet*. [en línea]. <http://funduino.de/DL/SW-520D.pdf>. [Consulta: abril de 2017].

5. BOYLESTAD, Robert L. *Introducción al análisis de circuitos.* 10a ed. México: Pearson Educación, 2004. 1248 p.

6. BOYLESTAD, Robert L.; NASHESLSKY, Louis. *Electrónica: teoría de circuitos y dispositivos electrónicos*. 8a ed. México: Pearson Educación, 2003. 1040 p.

7. CARLETTI, Eduardo. *Comunicación bus I2C. Descripción y funcionamiento*. [en línea]. <http://robots-argentina.com.ar/Comunicacion_busI2C.htm>. [Consulta: de enero de 2017].

8. CONVERSE, Tim; PARK, Joyce; MORGAN, Clark. *PHP5 and MySQL® Bible*. 3a ed. Estados Unidos: Wiley Publishing, Inc, 2004. 1042 p.

9. DEITEL, Harvey M.; DEITEL, Paul J. *Cómo programar en C/C++ y Java.* 4a ed. México: Pearson Educación, 2004. 1152 p.

10. *DHT 11 Humidity & Temperature Sensor Datasheet.* [en línea]. <http://www.micro4you.com/files/sensor/DHT11.pdf>. [Consulta: mayo de 2017].

11. *Electret Condenser Microphone Datasheet*. [en línea]. <https://cdn-shop.adafruit.com/datasheets/CMA-4544PF-W.pdf>. [Consulta: mayo de 2017].

12. GONZÁLEZ, Raúl. *Python para todos*. [en línea]. <http://mundogeek.net/tutorial-python/>. [Consulta: marzo de 2017].

13. Instituto Técnico de Capacitación y Productividad. *Consulta de detección de necesidades de capacitación en el área de seguridad industrial*. [en línea]. <http://www.intecap.edu.gt/oml/images/pdfsdocumentos/CNC-10.pdf>. [Consulta: abril de 2017].

14. KINGBRIGHT. *Phototransistor*. [en línea]. <https://www.kingbrightusa.com/images/catalog/spec/WP3DP3BT.pdf>. [Consulta: mayo de 2017].

15. MAXIM INTEGRATED PRODUCTS. *DS18B20 Programmable resolution 1-Wire digital thermometer Datasheet*. [en línea]. <http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. [Consulta: mayo de 2017].

16. *MQ-2 gas sensor datasheet.* [en línea]. <http://sandboxelectronics.com/files/SEN-000004/MQ-2.pdf>. [Consulta: mayo de 2017].

17. *Serial communication*. [en línea]. <https://learn.sparkfun.com/tutorials/serial-communication>. [Consulta: diciembre de 2016].

18. *Serial peripheral interface (SPI)*. [en línea]. <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>. [Consulta: diciembre de 2016].

19. Texas Instruments. *LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators Datasheet*. [en línea]. <http://www.ti.com/lit/ds/symlink/lm393-n.pdf>. [Consulta: mayo de 2017].

20. The Raspberry Pi Foundation. *Raspberry Pi.* [en línea]. <https://www.raspberrypi.org>. [Consulta: diciembre de 2016].

APÉNDICES

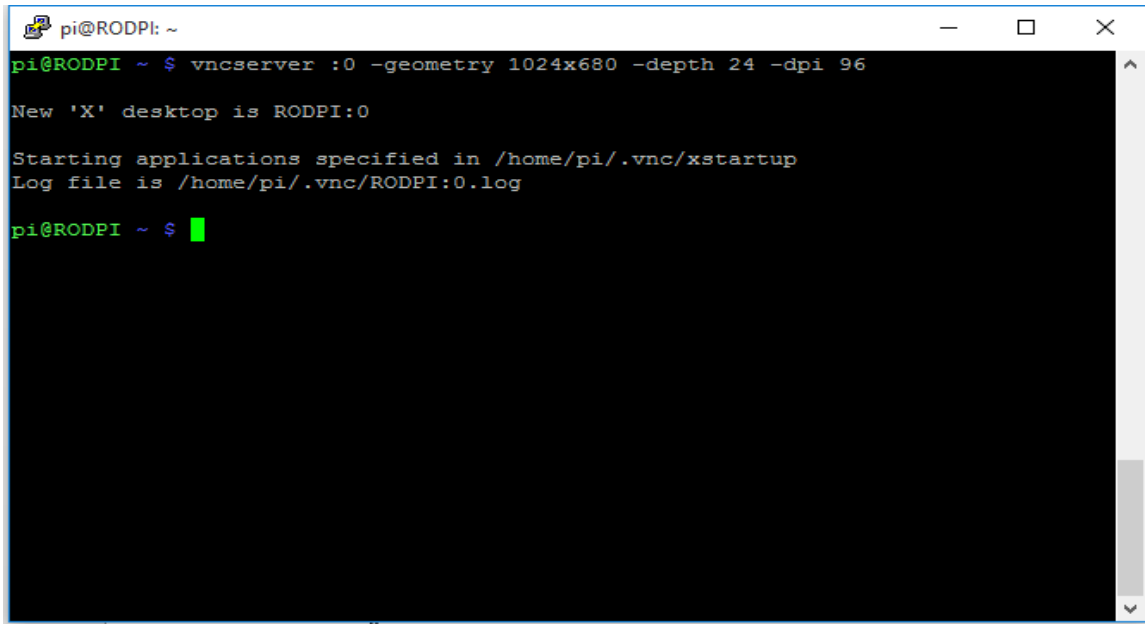
Apéndice 1. Instalación de entorno gráfico remoto

Se utiliza *tightvnc* server en el lado del servidor (Raspberry Pi) y VNC Viewer en el lado del cliente (desde donde se accede a la Raspberry Pi).

sudo apt-get install tightvncserver

Para poder acceder al entorno gráfico remoto se debe activar el servicio correspondiente con el siguiente comando:

vncserver :0 –geometry 1240x680 –depth 24 –dpi 96

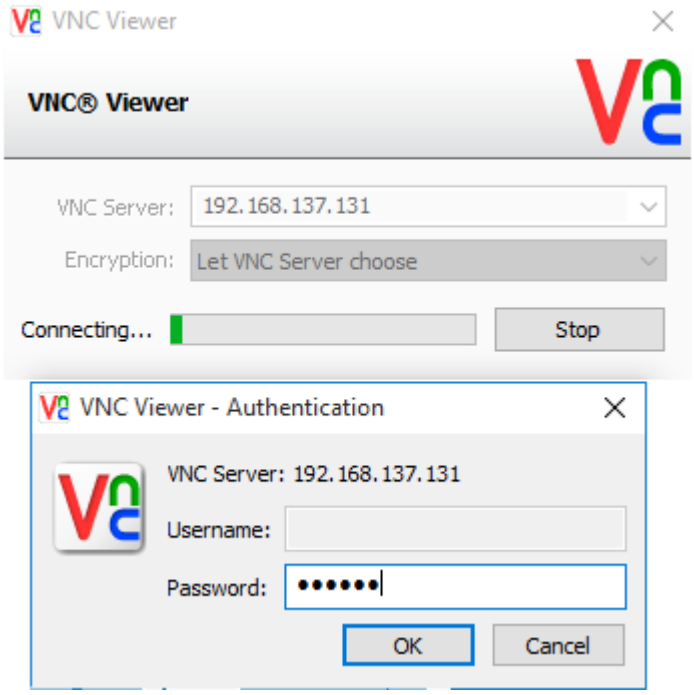


Fuente: elaboración propia.

Continuación de Apéndice 1.

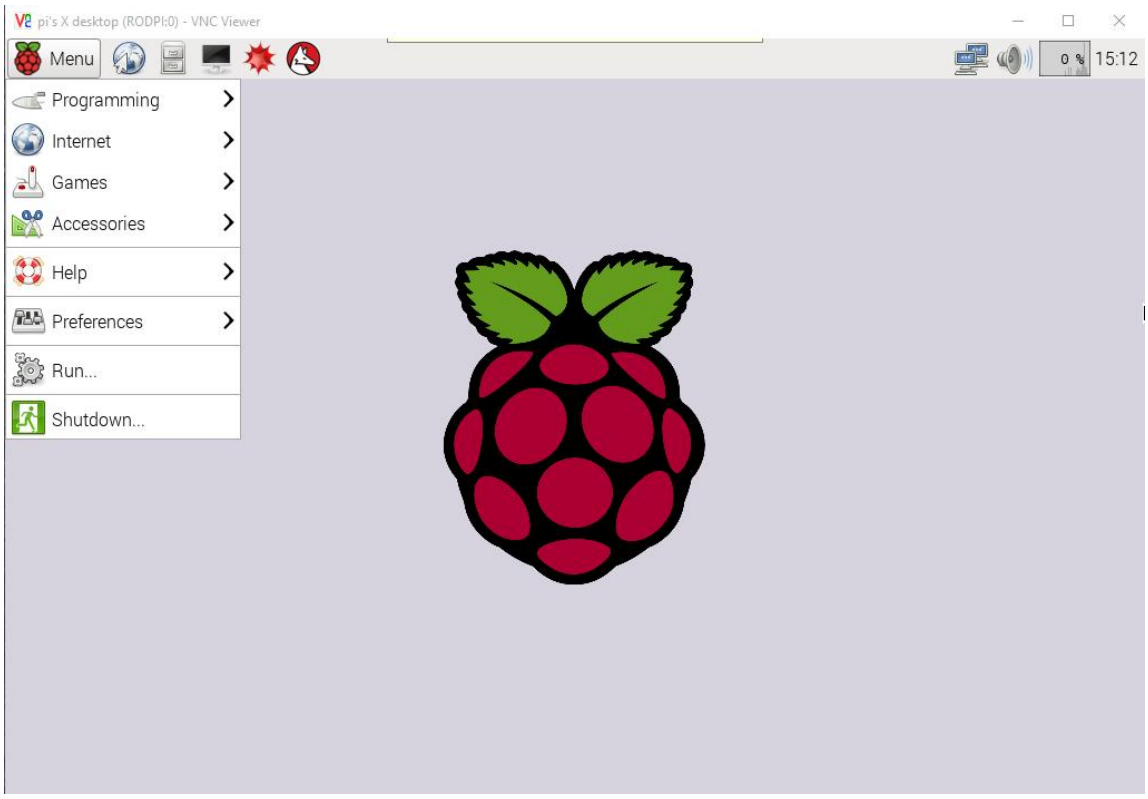
Donde ":0" indica el *display* número cero, "*geometry*" indica la resolución de pantalla ancho x alto, "*depth*" la profundidad de color en *bits* por *pixel,* en un número entre 8 y 32, y "dpi" para tamaño de fuentes de texto desplegadas. Para más información ingresar al sitio *web*: http://www.tightvnc.com/vncserver.1.php.

La primera ejecución preguntará por una contraseña para acceder a la interfaz gráfica; se debe colocar una. Utilizando un cliente VNC Viewer desde Windows se accede al entorno gráfico, como si se estuviera delante de la Raspberry Pi.



Fuente: elaboración propia.

Continuación de Apéndice 1.



Fuente: elaboración propia.

Por último, para cerrar el servicio generado y ahorrar recursos se utiliza el siguiente comando:

```
vncserver -kill :0
```

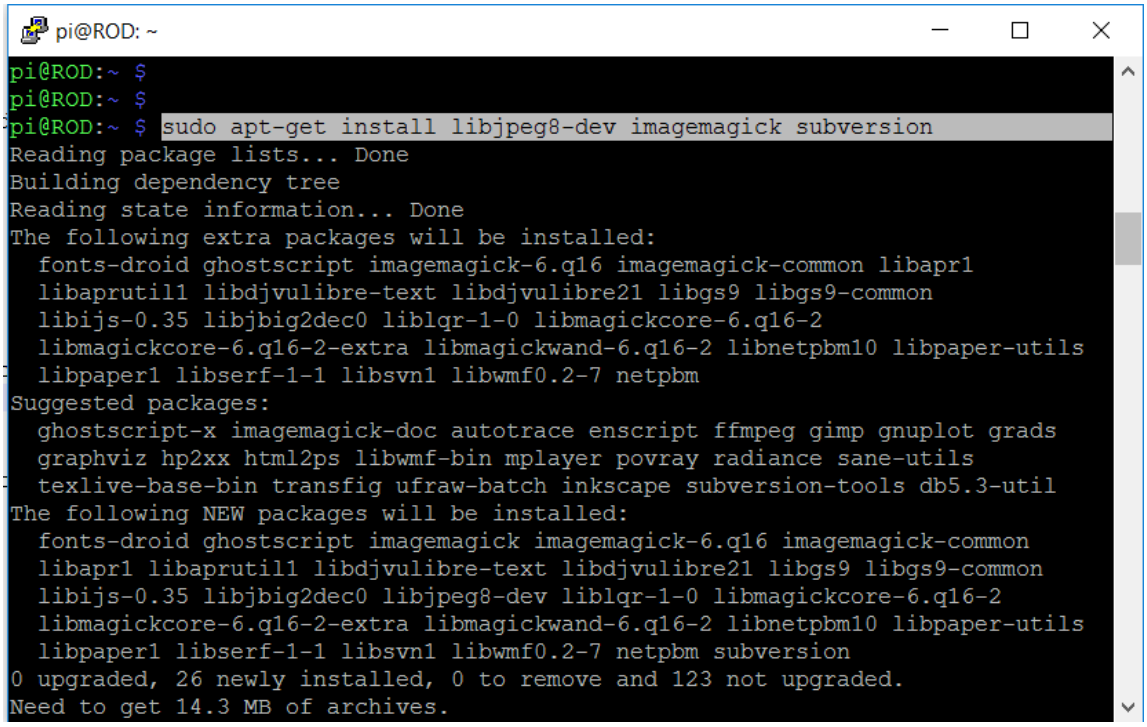
Apéndice 2. Instalación de MJPG-STREAMER

En el lado servidor se instalará esta utilidad que permite realizar una transmisión de video simple a través de una cámara *web.* Es recomendable consultar las fuentes oficiales de donde provienen para utilizar las instrucciones

Continuación de Apéndice 2.

de instalación. A través de la fuente de Github, se recomienda tener instalador *cmake* y libjpeg8-dev.

sudo apt-get install libjpeg8-dev imagemagick subversion



Fuente: elaboración propia.

*libjpeg**8*** es una utilidad para el manejo de archivos JPEG. *imagemagick* es una aplicación en línea de comandos para la creación o la manipulación de imágenes en múltiples formatos. *subversión*, también conocido como SVN, es una herramienta para el control de versiones de desarrollo de aplicaciones para software, aquí se utilizará para descargar el código fuente de *mjpg-streamer* desde la consola de Linux, no para llevar un control de versiones.

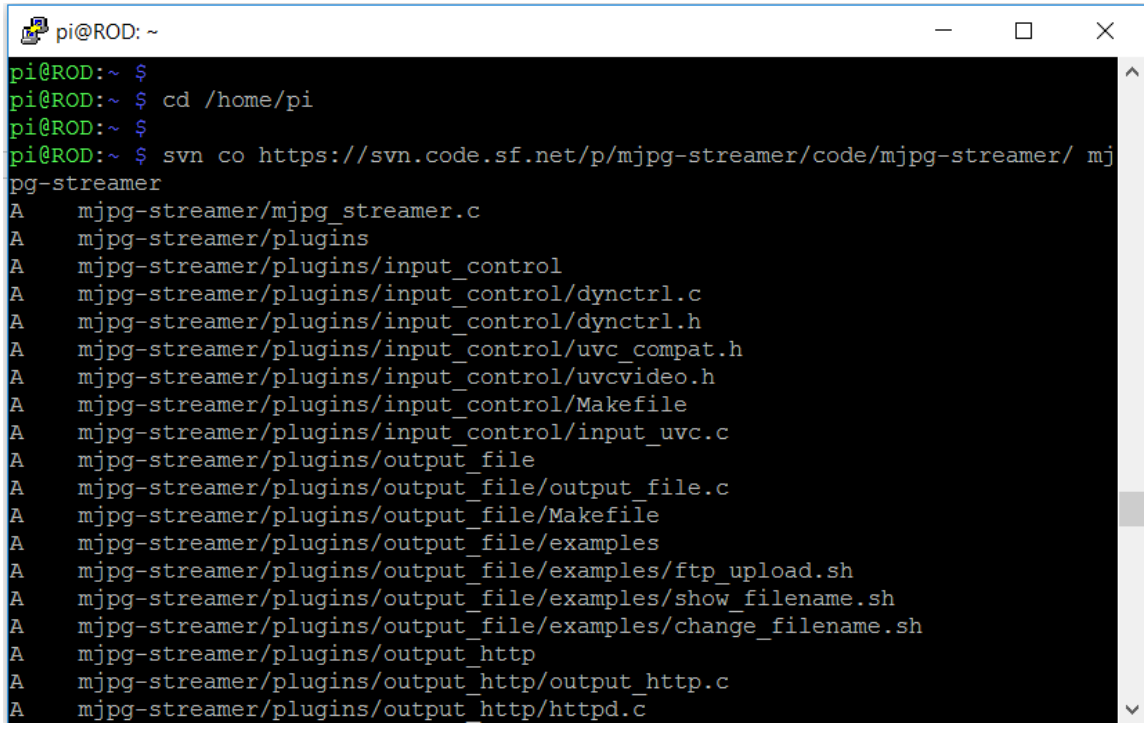
Continuación de Apéndice 2.

Es importante mencionar que también está disponible en el sitio de colaboración de software Sourceforge.net. Se puede acceder a través de este url: https://sourceforge.net/projects/mjpg-streamer/, o en su defecto se deberá buscar el código fuente oficial.

Ahora se procederá a descargar el código fuente para compilarlo, la ubicación de descarga es indiferente, por conveniencia se creará en/home/pi/.

cd /home/pi

svn co https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer



Fuente: elaboración propia.

Continuación de Apéndice 2.

Ya descargado, se compilará el código. Ahora se debe entrar a la carpeta descargada y ejecutar el comando para compilar el código fuente:

```
cd mjpg-streamer
make
```

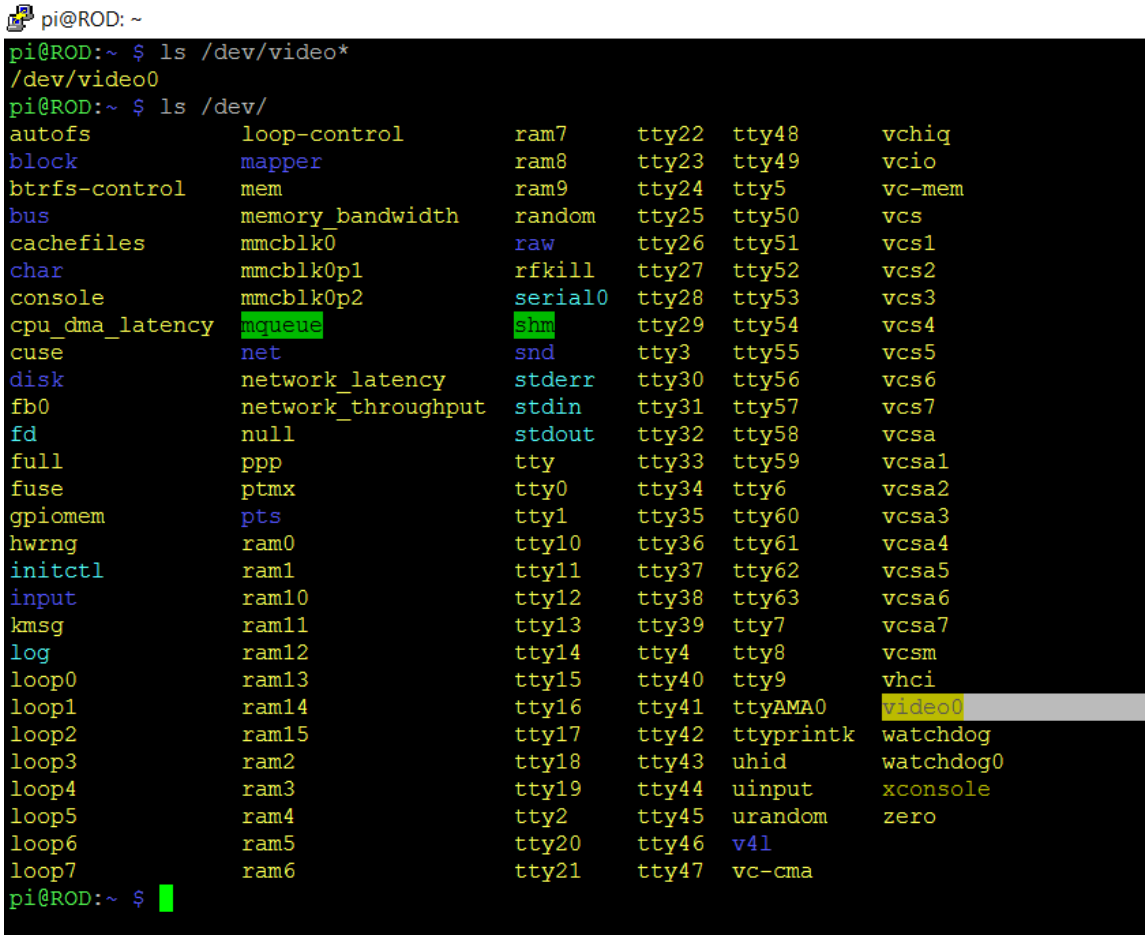
Para poder realizar la transmisión se necesita de una cámara *web*. Se conecta una cámara *web* y se revisa cómo aparece en el listado de archivos de dispositivos de hardware conectados a Raspberry Pi, eso en el directorio */dev.*

```
ls /dev
```

O bien, como se sabe que es un dispositivo cámara *web:*

```
ls /dev/video*
```

Continuación de Apéndice 2.



Fuente: elaboración propia.

Aparece el dispositivo *video0* indicando que Raspbian sí detecta la cámara *web*. El “0” es porque solo hay conectada una cámara. Ahora, ya es posible realizar una transmisión de video ininterrumpidamente. Pero antes se debe ejecutar *mjpg-streamer*. Se ingresa a la carpeta que tiene el ejecutable:

cd /home/pi/mjpg-streamer

Continuación de Apéndice 2.

El comando más básico para iniciar la transmisión es:

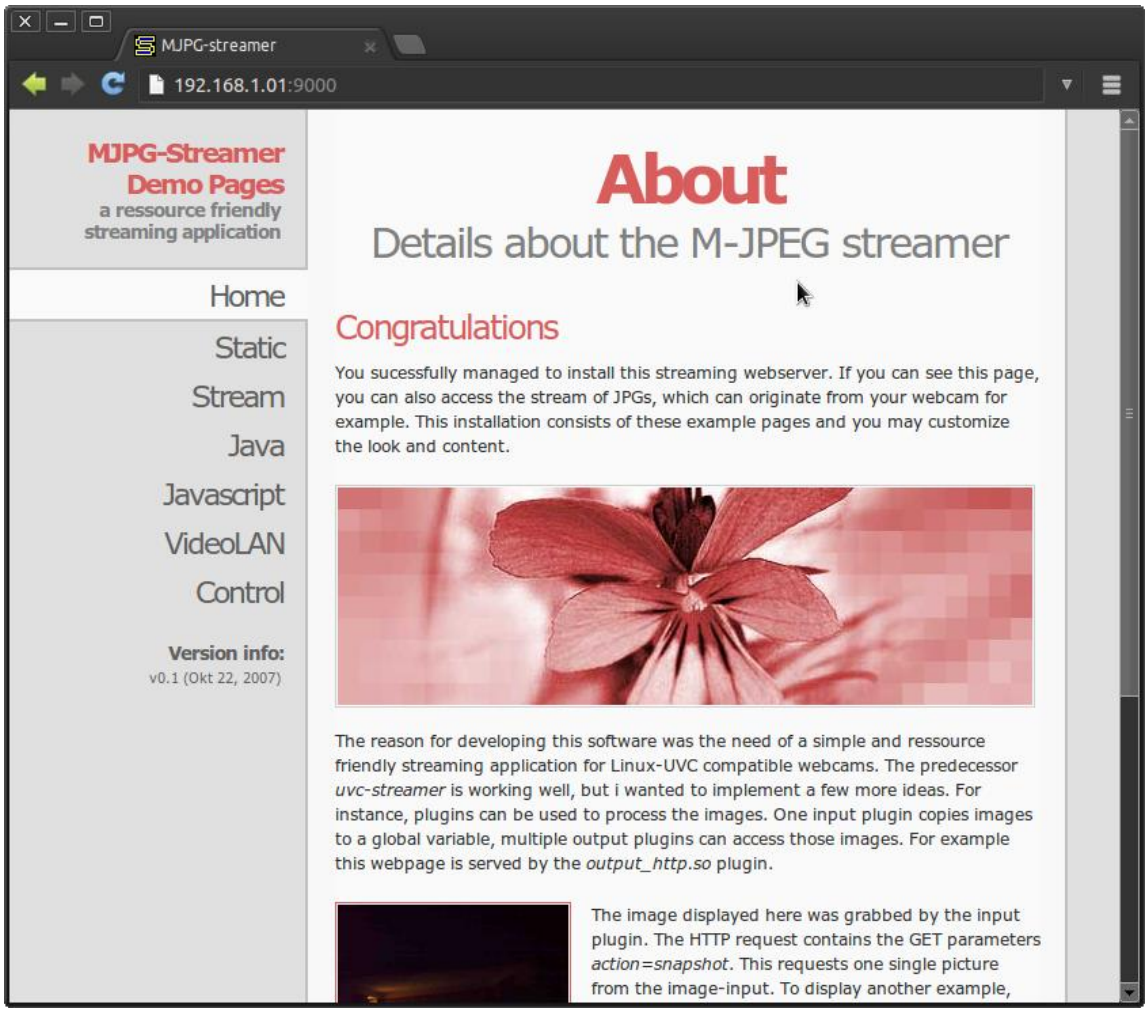
```
./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y" -o "./output_http.so -w
./www"
```

Como se indicó antes, */dev/video0* hace referencia directamente al dispositivo cámara *web.* Además, existen más parámetros que se pueden configurar, como establecer el puerto por donde se realiza la transmisión o los *frames* del video. Por ejemplo, se establece el puerto de transmisión 9090, 30 *frames* por segundo y calidad de codificación de JPEG a 20.

```
./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y -f 30 -q 20" -o
"./output_http.so -w ./www -p 9090"
```

Para visualizar la transmisión de video se requiere de cualquier computadora o teléfono que tenga un navegador *web* instalado y se ingresa a la url de la Raspberry Pi y el puerto, por ejemplo, en una red LAN *192.168.43.109:9090*.

Continuación de Apéndice 2.



Fuente: https://miguelmota.com/blog/raspberry-pi-camera-board-video-streaming//mjpg-streamer-demo-pages-high-res.png

Apéndice 3. Riegos típicos inherentes a cada sector industrial

Zapatería

La industria del calzado se dedica a todo lo relacionado con el calzado para pie, desde el diseño hasta la comercialización del producto. La industria del calzado se puede componer de diferentes empresas en la cadena de producción:

- Empresa de suministro de materias primas
- Empresa de fabricación del producto
- Empresa de distribución y ventas

Hay muchos establecimientos donde se realizan las ventas de calzado de marcas reconocidas, poseen sistemas de seguridad adecuados tanto para la integridad del personal, en el caso de extintores de incendios o detectores de humo, como para la protección contra robo del producto; en el caso de alarmas de seguridad o contratación de empresas de seguridad, este tipo de establecimientos se caracteriza por estar dentro de un centro comercial.

Por otro lado, existen establecimientos independientes que venden los productos a los consumidores de la misma manera, pero en muchos casos carecen de todo tipo de seguridad y es posible realizar las consideraciones necesarias para adaptar el sistema de seguridad del recinto controlado. Suponiendo un caso en donde no hay ningún tipo de seguridad, el objetivo es utilizar el dispositivo para la detección de intrusiones y la detección de humo que alerte de forma temprana el peligro. En este caso el sistema de seguridad no necesita ningún módulo extra, es capaz de transmitir video, de detectar

sonido, vibraciones y un sensor de humo. Deberá seleccionar la ubicación más adecuada dentro del recinto.

Industrial Textil

La industria textil se encarga de realizar los procesos y la producción de todo tipo de fibras, en sentido estricto está separada de la industria del calzado, no obstante en reportes económicos suelen incluirse en conjunto. La industria produce un producto de consumo en masa, por lo tanto genera empleo para muchas personas; hay fábricas textiles que no tiene un ambiente adecuado en temas de seguridad para tener a una población de trabajadores, la pérdida económica y la integridad de las personas puede verse afectada cuando suceda un desastre.

La materia prima usada para la industria textil son las fibras de tipo natural, como las fibras animal o vegetal, y fibras de tipo sintético que provienen de productos químicos, entre otras. Las fibras se consideran producto inflamable, además que se acumulan en gran cantidad en la mayoría de espacios. Algunos de los riegos asociados al sector textil son:

- Incendios.
- Elevada cantidad de materiales inflamables en el área de trabajo.
- Concentración de fibras suspendidas en el aire.
- Una cantidad insuficiente o falta completa de extintores.
- Ningún sistema de detección de humo o alarmas instaladas.
- Falta de creación de planes de emergencia y evacuación para instruir a los trabajadores.

Sin bien son varios los problemas en este tipo de sectores y no es posible cubrir todos, el sistema de monitoreo controlado de recintos puede actuar como un sistema de alerta temprana.

Panificadora

Las panificadoras son el lugar donde se fabrica y vende el pan de muchos tipos; las panificadoras y panaderías son susceptibles a incendios, si no se tiene el debido cuidado, hay materiales y materias primas inflamables tales como las grasas, tanques que almacenan el producto, los embalajes y los hornos. Además existen otros riesgos asociados a esta industria, como las posibles explosiones debido a los hornos de pan y robo en horas sin actividad humana. Se recomienda tener extintores para mínimas situaciones de emergencia, además de detectores de gas en el área para una detección temprana.

Recicladoras

Reciclar es la reutilización de materiales en lugar de desecharlos, es recopilar materiales que pueden ser procesados para volver a utilizarse en un ciclo de vida completo, de forma que se evita la creación de los mismos tipos de materiales, que en el proceso de crearlos contaminan el medio ambiente. Gases como el gas del diésel pueden ser perjudiciales para la salud, a largo plazo pueden producir cáncer pulmonar. Riesgos típicos como incendios y explosiones pueden existir en estos ambientes. Algunos materiales reciclables son:

- Plásticos
- Vidrio
- Madera
- Papel
- Dispositivos eléctricos
- Neumáticos

Los plásticos son otro tipo de material peligroso, los gases de la quema de plástico tienen partículas tóxicas para la salud y contaminantes para el medio ambiente, especialmente en el caso de las recicladoras que producen compuestos químicos e incineran basura.