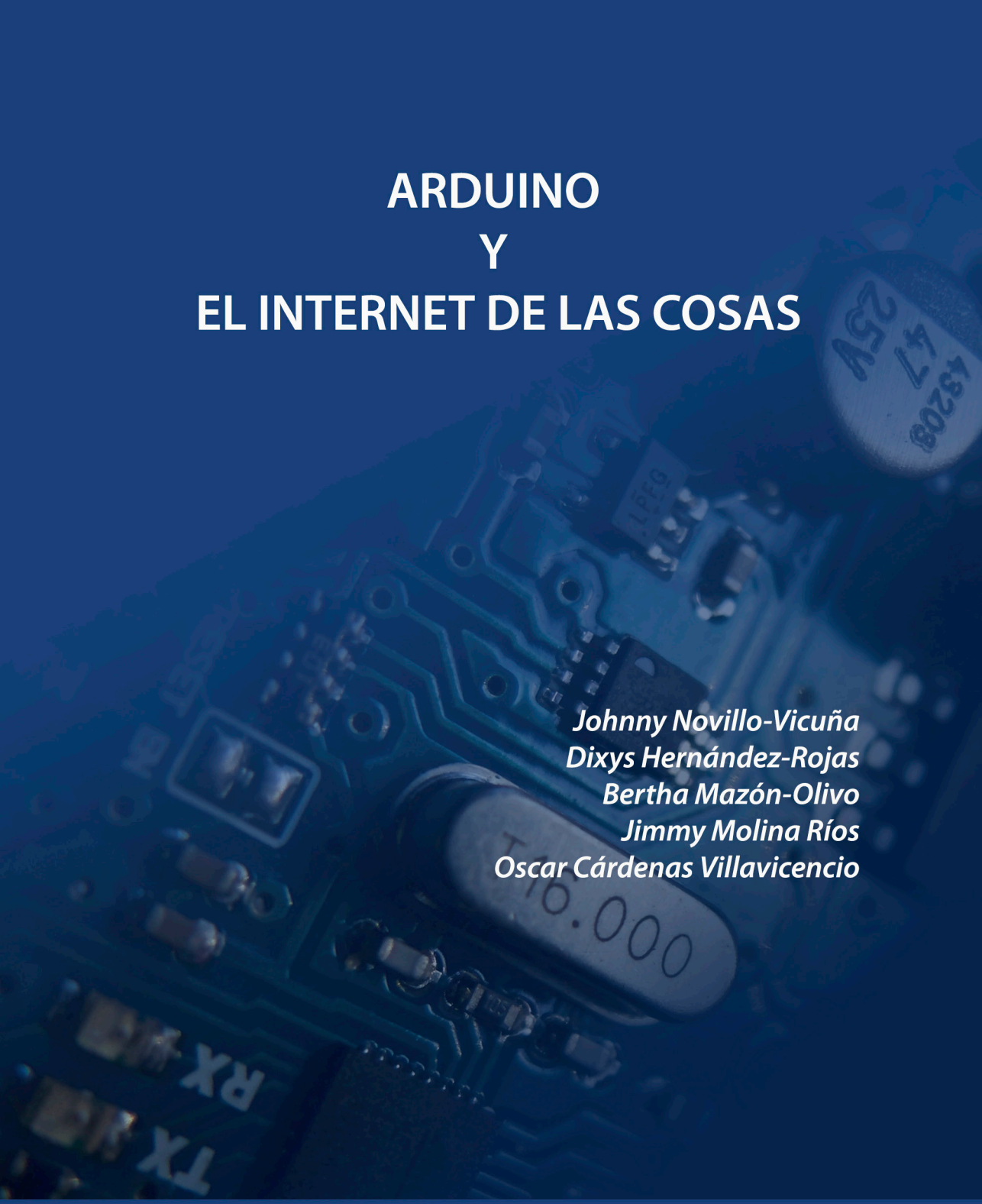
ARDUINO
Y
EL INTERNET DE LAS COSAS



Johnny Novillo-Vicuña
Dixys Hernández-Rojas
Bertha Mazón-Olivo
Jimmy Molina Ríos
Oscar Cárdenas Villavicencio

Ingeniería y Tecnología

3ciencias

ARDUINO
Y
EL INTERNET DE LAS COSAS

Johnny Novillo-Vicuña
Dixys Hernández Rojas
Bertha Mazón Olivo
Jimmy Molina Ríos
Oscar Cárdenas Villavicencio

Editorial Área de Innovación y Desarrollo,S.L.

ÁREA DE INNOVACIÓN Y DESARROLLO, S.L.

C/ Els Alzamora, 17- 03802- ALCOY (ALICANTE) info@3ciencias.com

Primera edición: **octubre 2018**

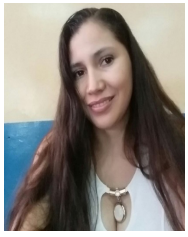
ISBN: **978-84-949151-8-5**

DOI: http://dx.doi.org/10.17993/IngyTec.2018.45

AUTORES

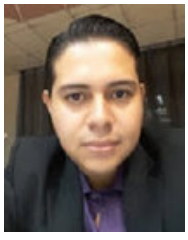
Johnny Novillo Vicuña. Ingeniero Eléctrico y Magister en Educación Superior, de nacionalidad ecuatoriana. Realizó sus estudios superiores de Ingeniería Eléctrica, en la "Universidad de Cuenca", y su postgrado en Educación Superior, en la Universidad Tecnológica "San Antonio de Machala – Universidad de Ciego de Ávila". En la actualidad, continúa desempeñando su cargo como docente titular e investigador en la Universidad Técnica de Machala, Gerente de SYSE Cía. Ltda., y ejerce su profesión de Ingeniero Eléctrico, en la construcción y consultoría de obras eléctricas para empresas públicas y privadas. Candidato a Doctor en Tecnologías de la Información y Comunicación en la Universidade A Coruña, España.

Dixys L. Hernandez Rojas. Ingeniero Electrónico y Máster en Electrónica, doble nacionalidad Cubana - Española. Sus estudios superiores de Ingeniería y postgrado de Ingeniería Electrónica y Máster en Electrónica respectivamente, fueron realizados en su alma mater de la "Universidad Central Marta Abreu de Las Villas". En la actualidad, vive en la ciudad de Machala, provincia de El Oro, desempeñando su cargo como docente titular e investigador en la Universidad Técnica de Machala. Candidato a Doctor en Tecnologías de la Información y Comunicación en redes móil la Universidade A Coruña, España.



Bertha Eugenia Mazón Olivo. Ecuatoriana, Ingeniera en Sistemas y Magíster en Informática Aplicada por la Escuela Superior Politécnica de Chimborazo (ESPOCH). Candidata a Doctora en Tecnologías de la Información y Comunicación en la Universidade A Coruña, España. Actualmente ejerce como docente-investigador de la Unidad Académica de Ingeniería Civil de la Universidad Técnica de Machala, en áreas de Inteligencia de Negocios, Sistemas Distribuidos e Internet de las Cosas.

Jimmy Rolando Molina Ríos. Nació en Machala, Provincia de El Oro – Ecuador, el 14 de Septiembre de 1984. Es Ingeniero de Sistemas de la Universidad Técnica de Machala, Magister en Docencia y Gerencia en Educación Superior de la Universidad de Guayaquil. Actualmente, ejerce como docente a tiempo completo, y desempeña el cargo de coordinador de las carreras de ingeniería de Sistemas y Tecnologías de la Información en la Universidad Técnica de Machala.



Oscar E. Cardenas Villavicencio. Ecuatoriano, Ingeniero de Sistemas y Magister en Telecomunicaciones. Sus estudios superiores de Ingeniería de Sistemas los cursó en la "Universidad Técnica de Machala", en la ciudad de Machala, y su maestría en Telecomunicaciones en la Universidad Católica de Santiago de Guayaquil, en la ciudad de Guayaquil. Actualmente es docente a tiempo completo en la Universidad Técnica de Machala.

ÍNDICE

INTRODUCCIÓN 13
CAPÍTULO I: INTRODUCCIÓN AL ARDUINO Y EL INTERNET DE LAS COSAS 15
1.1. Objetivos 15
1.2. Introducción 15
1.3. Arduino y el Internet de las Cosas 15
1.4. Conceptos previos 16
1.4.1. Internet de las Cosas (IoT) 16
1.4.2. Hardware Libre 17
1.4.3. Arduino 19
1.4.3.1. Historia 20
1.5. Plataformas Arduino 20
1.5.1. Arduino Mega 2560 20
1.5.2. Arduino ADK 21
1.5.3. Arduino YUN 21
1.5.4. Arduino UNO 22
1.5.5. Arduino Duemilanove 23
1.5.6. Arduino Nano 23
1.5.7. Arduino Leonardo 24
1.5.8. Arduino Micro 24
1.5.9. Arduino Esplora 25
1.5.10. Arduino Mini 26
1.5.11. Arduino BT 26
1.5.12. Arduino Pro 27
1.5.13. Arduino Pro Mini 27
1.5.14. Arduino Yun Linux 28
1.5.15. Arduino Duemilanove 28
1.5.16. Arduino Diecimila 29
1.5.17. Arduino Nano A Tmega168 29
1.5.18. Lilypad Arduino ATmega 328V 30
1.5.19. Arduino Fio 30
1.5.20. Arduino Pro 31
1.5.21. Arduino Pro mini 31
1.5.22. LilyPad Arduino 32
1.5.23. Arduino Gemma 32
1.6. Tipos de conectividad entre Arduino y el computador 33
1.6.1. Comunicación Serial 33
1.6.2. Comunicación en Paralelo 35
1.7. Protocolos de comunicación 35
1.7.1. MQTT 35
1.7.2. CoAP 36
1.7.3. Rest API 37
1.7.4. XMPP 38

CAPÍTULO II: PRÁCTICAS DE ARDUINO ORIENTADAS AL INTERNET DE LAS COSAS 39
2.1. Objetivo 39
2.2. Introducción 39
2.3. Adquisición de datos en tiempo real de sensores utilizando protocolo de comunicaciones MQTT, Ethernet Shield y Arduino uno 39
2.3.1. Objetivos 39
2.3.2. Descripción 40
2.3.3. Materiales 40
2.3.4. Desarrollo 41
2.3.4.1. Diagrama esquemático de la actividad 41
2.3.4.2. Descripción del funcionamiento del circuito 41
2.3.5. Implementación del software 42
2.3.5.1. Instalación de la máquina virtual, Virtual Box 4.3.0 42
2.3.5.2. Importar y configurar la máquina virtual del sistema operativo Débian donde se encuentra pre-configurado el servidor bróker 44
2.3.5.3. Instalación de Arduino IDE 47
2.3.5.4. Instalación de Node JS 52
2.3.5.5. Instalación del editor de textos "Sublime Text" 55
2.3.6. Implementación del hardware 57
2.3.6.1. Conexión de placa Arduino UNO con la Ethernet Shield 57
2.3.6.2. Conexión para grabar código de programación (ver código de programación) a la tarjeta Arduino 60
2.3.6.3. Conexión del cable de red a la tarjeta Arduino 61
2.3.6.4. Configuración de la tarjeta de red de la computadora 61
2.3.7. Detalles de configuración y ejecución de la práctica 63
2.4. Letrero led con conexión desde un Smartphone Vía Bluetooth 64
2.4.1. Objetivos 64
2.4.2. Descripción 64
2.4.2.1. Descripción de las matrices LED 65
2.4.3. Materiales 67
2.4.4. Desarrollo 68
2.4.4.1. Implementación del hardware 68
2.4.4.2. Implementación del software 71
2.4.5. Detalles de configuración y ejecución de la práctica 76
2.5. Control de estado de un actuador mediante la red social twitter haciendo uso del protocolo MQTT 76
2.5.1. Objetivos 76
2.5.2. Descripción 76
2.5.3. Materiales 77
2.5.4. Desarrollo 78
2.5.4.1. Diagrama esquemático 78
2.5.4.2. Descripción de funcionamiento de circuito 78
2.5.5. Implementación del software 78

2.5.5.1. Instalación de la máquina Virtual Box 4.3.0 78
2.5.5.2. Importar y configurar la máquina virtual del sistema operativo Débian donde se encuentra pre-configurado el servidor bróker 81
2.5.5.3. Instalación de Arduino IDE 84
2.5.5.4. Instalación de Node JS 86
2.5.6. Detalles de configuración y ejecución de la práctica 88
REFERENCIAS BIBLIOGRÁFICAS 91
ANEXOS 93

ÍNDICE DE TABLAS

Tabla 1. ARDUINO MEGA 2560. 20
Tabla 2. ARDUINO ADK. 20
Tabla 3. ARDUINO YUN. 21
Tabla 4. ARDUINO UNO. 21
Tabla 5. ARDUINO DUEMILANOVE. 22
Tabla 6. ARDUINO NANO. 22
Tabla 7. ARDUINO LEONARDO. 23
Tabla 8. Arduino Micro. 23
Tabla 9. ARDUINO ESLORA. 24
Tabla 10. ARDUINO MINI. 25
Tabla 11. ARDUINO BT. 25
Tabla 12. ARDUINO PRO. 26
Tabla 13. ARDUINO PRO MINI. 26
Tabla 14. ARDUINO YUN LINUX. 27
Tabla 15. ARDUINO DUEMILANOVE. 27
Tabla 16. ARDUINO DIECIMILA. 28
Tabla 17. ARDUINO NANO CON PLACA ATMEGA 168. 28
Tabla 18. ARDUINO LILYPAD. 29
Tabla 19. ARDUINO FIO. 29
Tabla 20. ARDUINO PRO ATMEGA328. 30
Tabla 21. ARDUINO PRO. 30
Tabla 22. ARDUINO LILYPAD ATMEGA 168. 31
Tabla 23. ARDUINO GEMMA. 31

ÍNDICE DE FIGURAS

Figura 1. Internet de las cosas. 15
Figura 2. Logotipo de OSHWA. 16
Figura 3. Logotipo de Arduino. 18
Figura 4. Arduino Mega 2560. 20
Figura 5. Arduino ADK. 20
Figura 6. Arduino YUN. 21
Figura 7. Arduino Uno. 21
Figura 8. Arduino Duemilanove. 22
Figura 9. Arduino Nano modelo1. 22
Figura 10. Arduino Nano modelo2. 22
Figura 11. Arduino Leonardo. 23
Figura 12. Arduino Micro. 23
Figura 13. Arduino Esplora. 24
Figura 14. Arduino Mini. 25
Figura 15. Arduino BT. 25
Figura 16. Arduino Pro. 26
Figura 17. Arduino Pro Mini. 26
Figura 18. Arduino Yun Linux. 27
Figura 19. Arduino Duemilanove ATmega168. 27
Figura 20. Arduino Diecimila. 28
Figura 21. Arduino nano ATmega168. 28
Figura 22. Arduino LilyPad ATmega328. 29
Figura 23. Arduino Fio. 29
Figura 24. Arduino Pro ATmega328. 30
Figura 25. Arduino Pro. 30
Figura 26. Arduino LilyPad ATmega168. 31
Figura 27. Arduino Gemma. 31
Figura 28. Ethernet. 32
Figura 29. Wireless SD Shield. 32
Figura 30. USB host Shield. 33
Figura 31. Arduino Proto Shield. 33
Figura 32. Protocolo MQTT. 34
Figura 33. Protocolo CoAp. 35
Figura 34. Protocolo Rest API. 36
Figura 35. Protocolo XMPP. 37

INTRODUCCIÓN

Este texto de Arduino y el internet de las cosas, ayudará a sus lectores a conceptualizar definiciones básicas que permitan fortalecer el aprendizaje y conocimiento de la importancia que tiene Arduino en el internet de las cosas.

Los temas considerados en el desarrollo del texto han sido referenciados por diferentes fuentes, y seleccionados gracias a diversas experiencias adquiridas durante el desarrollo de la cátedra de Microprocesadores en la Universidad Técnica de Machala.

El texto consta de dos capítulos, los cuales se lo han dividido con la finalidad de mejorar el aprendizaje del lector; el primer capítulo trata sobre la Introducción al Arduino y el Internet de las cosas; y el segundo capítulo abarca las prácticas de Arduino orientadas al internet de las cosas. A continuación, se explica brevemente el contenido de cada uno de los capítulos:

- **CAPÍTULO I – Introducción al Arduino y el internet de las cosas**: Este capítulo comprende seis temáticas: Hardware Libre, Arduino, Historia de Arduino, Plataforma de Arduino, Tipos de Conectividad entre Arduino y el computador y Protocolos de comunicación.
- **CAPÍTULO II Prácticas de Arduino orientadas al internet de las cosas:** En este capitulo.se abarcarán tres prácticas: “Adquisición de datos en tiempo real de sensores utilizando protocolo de comunicaciones MQTT, Ethernet Shield Y Arduino uno”, “Letrero led con conexión desde un Smartphone vía Bluetooth”, “Control de estado de un actuador mediante la red social Twitter haciendo uso del protocolo MQTT” .

Conforme realice la lectura, se encontrará con símbolos que le ayudarán a entender mejor el tema, evaluar sus conocimientos y reforzar lo aprendido mediante fuentes alternas para complementar ciertos temas. Entre los símbolos establecidos se encuentran:

Sabías que...

Presenta un enfoque que los autores realizan sobre una temática relacionada al tema central o sobre algún dato curioso propicio en el contenido.

Resuelve

Preguntas de reflexión que le ayudarán a reforzar su conocimiento del tema.

Bibliografía y fuentes electrónicas

Bibliografía y fuentes electrónicas que le ayudarán a complementar el tema tratado.

CAPÍTULO I: INTRODUCCIÓN AL ARDUINO Y EL INTERNET DE LAS COSAS

1.1. Objetivos

- Contextualizar a los lectores acerca de la importancia que tiene Arduino en el internet de las cosas.
- Detallas las características de las diferentes plataformas de Arduino existentes hasta la presente fecha.
- Explicar los protocolos de comunicación que se utilizan para la aplicación de Arduino y el internet de las cosas.

1.2. Introducción

El internet de las cosas es uno de los conceptos que han marcado tendencia en la actualidad, gracias a todos los beneficios que brinda su aplicación en los hogares.

En la actualidad la tendencia hacia el Internet de las cosas (Iot: Internet of things), ha permitido que la mayoria de artefactos electrónicos de uso cotidiano, tales como celulares, tablets, computadoras, refrigeradoras, etc, puedan comunicarse con la nube y generar reportes.

El presente capitulo brinda al lector información teórica completa sobre el Arduino y el Internet de las cosas, así como su importancia, aplicación, plataformas, tipo de conexión y los protocolos que utilizan.

1.3. Arduino y el Internet de las Cosas

La humanidad en el presente vive un momento de transición muy importante para su desarrollo, la integración de circuitos electrónicos en sus actividades cotidianas. Los circuitos electrónicos son cada vez más útiles dentro de nuestro diario vivir, desde servirnos para poner la hora en una alarma de radio, preparar un café exactamente a nuestro gusto o simplemente escoger nuestra transacción en un cajero automático, hasta llegar a la navegación automática de aviones, automóviles y barcos.

La importancia que tienen estos objetos en nuestro estilo de vida actual contrasta fuertemente con el conocimiento que la población en general tiene de los mismos. Las personas promedio ven los circuitos microprocesados como algo incomprensible y cuya programación e implementación está fuera de su alcance. Es por esta razón que se llevan adelante ciertas iniciativas que buscan vincular a la población general con el desarrollo de hardware y software de manera sencilla y dinámica, estos son las iniciativas de hardware y software libre.

Existen muchas empresas de hardware libre pero la más sobresaliente es sin duda la empresa de hardware libre italiana llamada Arduino. Arduino facilita la conexión e implementación de hardware en placas impresas que contienen un microprocesador y brinda también el software necesario para realizar una programación sencilla; permitiendo de esta manera a la población en general poder crear sus propios artefactos para satisfacer necesidades específicas.

Estas iniciativas han tenido una gran acogida en los últimos años, ya que permiten a personas con ideas frescas e innovadoras, poder materializar sus anhelos de manera sencilla y con la posibilidad de brindar al mundo un aporte importante y que pueda tener un impacto gigantesco en el estilo de vida.

Arduino pone a disposición del público en general los diagramas completos de sus circuitos impresos y de todos los componentes por ellos desarrollados y además los comercializa, ya implementados, a un costo muy asequible.

La comunicación de Arduino y el almacenamiento de información histórica de los datos recabados se ha vuelto un aspecto muy relevante en la actualidad, en especial con la visión del internet de las cosas (IoT- Internet of things), donde se busca que todos los objetos que se posean conexión a alguna red puedan recibir y enviar información sobre el funcionamiento y estado del dispositivo.

Para poder realizar esta titánica tarea se han desarrollado distintas formas de comunicación, tanto cableada como inalámbrica, serial y paralela. Teniendo cada una de estas sus beneficios y problemas individuales.

En el presente capítulo se llevará a cabo un análisis de las principales formas y protocolos de comunicación utilizados en la actualidad para permitir la comunicación entre un computador y una placa Arduino.

1.4. Conceptos previos

1.4.1. Internet de las Cosas (IoT)

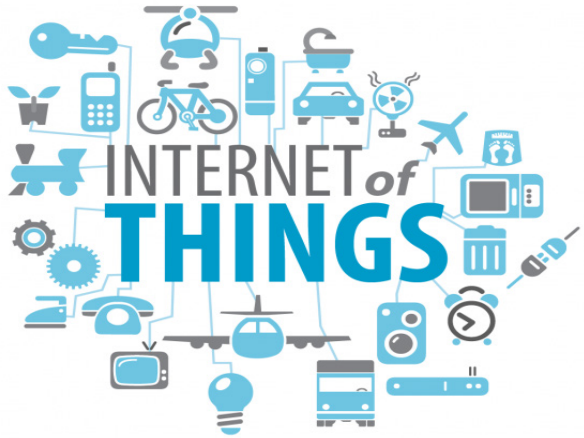


Figura 1. Internet de las cosas.

El mundo está cambiando, anteriormente los objetos solamente se comunicaban para dar informes de sus sensores, pero en la actualidad los objetos se comunican entre ellos, intercambian información y toman acciones de acuerdo a la información recibida.

Este intercambio y procesamiento de información mediante el uso de redes es conocido como Internet de las cosas (IoT).

Ejemplos de la aplicación del internet de las cosas:

- Cuidado infantil
- Rastreo de desempeño deportivo
- Domótica
- Rastreo de objetos
- Vigilancia
- Smart city
- Smart agriculture
- Smart buildings
- Smart banking

El internet de las cosas es la red de dispositivos que se encuentran interconectados entre sí, o con internet; esta red permite que los dispositivos compartan información entre ellos, tanto recabada por sensores o realizada por actuadores además de permitir manipulación remota de los mismos.

Sabías que...

Internet de las cosas. *El concepto de interconectar dispositivos aparece en la década de los 70s, pero el término "internet de las cosas" apareció en 1999 acuñado por Kevin Ashton mientras laboraba en Procter & Gamble.*

Preguntas de reflexión

¿Cuáles son las siglas de internet de las cosas?
¿Qué es internet de las cosas?

Bibliografía y fuentes electrónicas

Adrian McEwen, H. C. (2014). *Internet de las cosas: la tecnología revolucionaria que todo lo conecta.*

CLÚSTER ICT-AUDIOVISUAL DE MADRID (2013). *Internet de las cosas: Objetos interconectados y dispositivos inteligentes.* Madrid.

Pisani, F. (2016). *Internet Industrial: Máquinas inteligentes en un mundo de sensores.* Barcelona: Editorial Planeta.

1.4.2. Hardware Libre

Figura 2. Logotipo de OSHWA.

El hardware libre es un concepto que ha sido desarrollado en los años recientes, que defiende la idea de permitir acceso completo a diseño de hardware sin licencia de propiedad, abaratando a gran medida sus costos y permitiendo a su vez la modificación para mejora del hardware en cuestión. En el año 2010, OSHWA (Open

Source Hardware Association) estableció los principios por los que se regirá el Open Hardware.

Pese a los monumentales esfuerzos realizados en los años siguientes, aún no se ha logrado definir adecuadamente los conceptos mediante los cuales se regirá el hardware libre, ya que en la mayoría de los casos los circuitos implementados utilizan otros que son hardware con licencia de propietario, razón por la cual es muy difícil encasíllar un hardware como completamente libre.

Según OSHW Statement of principales 1.0, el hardware libre debe estar disponible de manera pública para todos los que quieran estudiarlo, modificarlo o crearlo y el hardware creado basado en este diseño podrá ser creado y distribuido sin ninguna restricción. En general, se considera una buena práctica la distribución de los modelos en software de diseño que sea open source para maximizar la capacidad de visualización que la población pueda tener sobre el mismo.

En los principios 1.0 establecidos se debe respetar los siguientes criterios:

1. Documentación
El hardware que se creará bajo esta licencia debe incluir documentación completa en forma de fichas de diseño y debe permitir realizar modificaciones en caso de ser necesarias y poder redistribuir el diseño original o modificado.

2. Alcance
El diseño puede ser liberado parcial o totalmente por lo que debe estar claramente especificado que partes del diseño son liberadas.

3. Programas informáticos necesarios
En caso de requerir un programa informático para operar, el hardware deberá proveerse de manera íntegra o bien detallar claramente mediante pseudocódigo como el mismo deberá ser desarrollado.

4. Obras derivadas
En caso de cualquier obra realizada bajo la licencia de un diseño que sea open hardware, esta se distribuirá bajo los mismos términos. La licencia deberá permitir que se vendan, distribuyan o fabriquen productos.

5. Libre redistribución
Los productos realizados bajo el diseño open software podrán ser vendidos libremente sin restricciones ni pagos por derecho de autor.

6. Atribución
Podría requerirse que se cite el nombre de los autores originales en la distribución del fichero original o de los ficheros derivados.

7. No discriminación a personas o grupos
Todo hardware libre debe estar al alcance de todas las personas de la misma manera.

8. No discriminación a campos de aplicación
La licencia permite a cualquier persona y propósito que requiera. No se puede limitar a un campo específico.

9. Distribución de licencia
La licencia permanecerá vigente no importa cuántas veces se distribuya el diseño.

10. La licencia no será específica para un producto
En caso de modificación o uso del diseño de hardware libre original esté mantiene su licencia vigente.

11. La licencia no deberá restringir otro hardware o software
La licencia no restringe el uso de hardware o software privativo ni libre.

12. La licencia será neutra en términos tecnológicos
Aparte de la plataforma de Arduino, existe otra llamada RaspberryPi, en ambas plataformas se puede controlar las entradas y salidas de los mismos, pero con la diferencia de que el Arduino es utilizado en circuitos de automatización como la robótica, entre otros, mientras que el Raspberry es un ordenador pequeño y simple, su uso es versátil ya que se utiliza en diferentes campos como: Computación general, Enseñanza de programación, Plataforma de proyectos, etc. Es decir, el Raspberry es muy importante para estimular la enseñanza en el área de ciencias computacionales.

1.4.3. Arduino

Figura 3. Logotipo de Arduino.

Arduino es una empresa de desarrollo de hardware, que tiene como objetivo principal el diseño y manufactura de circuitos electrónicos en circuitos impresos que incorporan un microcontrolador y el entorno de desarrollo para realizar la programación de cada placa de manera sencilla.

La empresa busca permitir a los usuarios tener todas las facilidades necesarias para poder construir aparatos interactivos, es decir, que no se centren únicamente en realizar una serie de movimientos o acciones específicas, sino que también puedan realizar procesamiento de señales y activar sus actuadores para realizar las actividades necesarias, brindando de esta manera una adecuada respuesta a las mismas. Es por esto, que es fundamental crear un puente de comunicación que permita al usuario que acciones debe recibir señales la placa y cómo reaccionar ante cada una de ellas.

Para la tarea de comunicación Arduino cuenta con varios puertos de comunicación serial digital y análogo, mismos que permiten comunicación directa mediante la interacción con el dispositivo o mediante el envío de señales por bluetooth, wifi, ethernet, entre otros medios.

Arduino cuenta con un catálogo muy amplio de placas con microprocesadores y shields que simplifican la conexión de los circuitos necesarios para el desarrollo de los artefactos que el usuario requiera crear.

1.4.3.1. Historia

Arduino nació en el instituto IVRAE de la mano de Massimo Banzi en el año 2005, esta plataforma era una placa de circuitos conectados a un microcontrolador al que únicamente podían conectársele resistencias y leds y que no contaba con un lenguaje para su control. Fue creada ante la necesidad de material de educación para los estudiantes del instituto, debido a los elevados costos de las placas en aquella época. Tras su éxito inicial se sumaron al equipo Harmando Barragán y David Mellis quienes desarrollaron un entorno de programación para la placa, mismo que fue bautizado como Wiring, además se sumaron al equipo David Cuartielles y Tom Igoe, quienes mejoraron la interfaz de hardware, agregando peurtos USB y los microcontroladores necesarios para poder interpretar y almacenar las instrucciones que enviaba Wiring.

Sabías que...

Origen del nombre de Arduino

- El nombre de Arduino viene del nombre del Bar di Re Arduino (Bar del rey Arduino), ubicado en Ivrea en la ciudad de Turín en la región Piedmont del noroeste de Italia, lugar donde Massimo Banzi solía reunirse con el equipo de desarrollo.

- Las placas Arduino tuvieron que cambiar su nombre a Genuino por conflictos legales ante la internacionalización del producto, actualmente ambas marcas coexisten y la única diferencia es su procedencia: Arduino para estados unidos y Genuino para el resto del mundo.

Bibliografía y fuentes electrónicas

Aranda, D. (2014). *ELECTRÓNICA 3: Plataformas Arduino y Raspberry Pi.* Buenos Aires: Manual USERS.

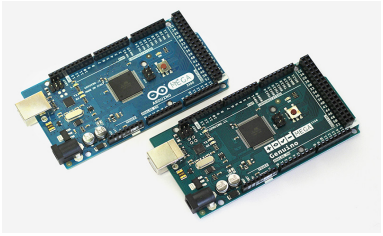
Nicolas GOILAV, G. L. (2016). *Arduino: Aprender a desarrollar para crear objetos inteligentes.* Barcelona: Ediciones ENI.

1.5. Plataformas Arduino

1.5.1. Arduino Mega 2560

Arduino Mega 2560 es la placa con mayor cantidad de E/S permitiendo la creación de circuitos extremadamente grandes, y a su vez ofrece soporte para la mayoría de los shields de la familia Arduino. Arduino mega 2560 reemplaza a Arduino Mega tradicional.

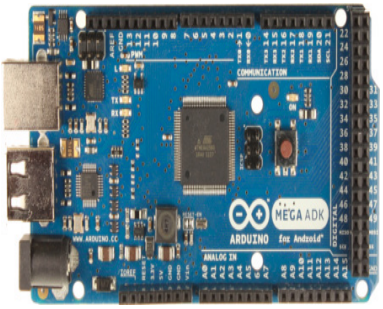
Tabla 1. ARDUINO MEGA 2560.

ARDUINO MEGA 2560		
 **Figura 4.** Arduino Mega 2560.	***Microcontrolador:***	ATmega2560
	Voltaje de Operación:	5V
	Pines digitales:	28
	Pines PWM:	15
	Pines de entradas análogas:	11
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	256 KB
	Memoria SRAM:	256 KB
	Memoria EEPROM:	4 KB
	Velocidad de reloj:	16 MHZ

1.5.2. Arduino ADK

Arduino ADK está basado en Arduino 2560, con la diferencia que tienen una resistencia de 8u2 hwb a tierra, haciendo más fácil ponerlo en modo DFU.

Tabla 2. ARDUINO ADK.

ARDUINO ADK		
 **Figura 5.** Arduino ADK.	***Microcontrolador:***	ATmega2561
	Voltaje de Operación:	5V
	Pines digitales:	54
	Pines PWM:	0
	Pines de entradas análogas:	16
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	256 KB
	Memoria SRAM:	8 KB
	Memoria EEPROM:	4 KB
	Velocidad de reloj:	16 MHZ

1.5.3. Arduino YUN

Arduino YUN presenta la particularidad de permitir la comunicación con LunixDistribution on board, lo cual permite utilizarlo como una computadora en red con la facilidad de implementación de Arduino.

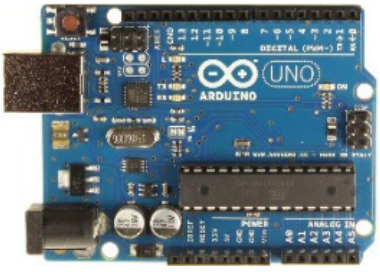
Tabla 3. ARDUINO YUN.

ARDUINO YUN		
 **Figura 6.** Arduino YUN.	***Microcontrolador:***	ATmega32U4
	Voltaje de Operación:	5V
	Pines digitales:	20
	Pines PWM:	7
	Pines de entradas análogas:	12
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2.5 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.4. Arduino UNO

Arduino uno es la placa ideal para iniciar en el desarrollo de hardware debido a que es la placa más ampliamente documentada en la familia Arduino.

Tabla 4. ARDUINO UNO.

ARDUINO UNO		
 **Figura 7.** Arduino Uno.	***Microcontrolador:***	ATmega328P
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.5. Arduino Duemilanove

Arduino Duemilanove, cuyo nombre significa 2009 en celebración del año en el que fue lanzado, es una placa basada en ATmega328.

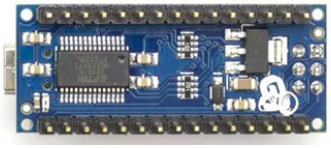
Tabla 5. ARDUINO DUEMILANOVE.

ARDUINO DUEMILANOVE		
 **Figura 8.** Arduino Duemilanove.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	0
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	51 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.6. Arduino Nano

Arduino Nano es una placa cuya principal característica es su reducido tamaño sin sacrificar sus componentes y facilitando la implementación del mismo en protoboards.

Tabla 6. ARDUINO NANO.

ARDUINO NANO		
 **Figura 9.** Arduino Nano modelo1.  **Figura 10.** Arduino Nano modelo2.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	1
	Pines de entradas análogas:	8
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.7. Arduino Leonardo

Arduino Leonardo difiere de todos los demás Arduinos basados en ATmega32u4 en que tiene comunicación USB integrada, permitiendo que este se instale los drivers necesarios al conectar al computador, tal como lo haría un teclado o un mouse.

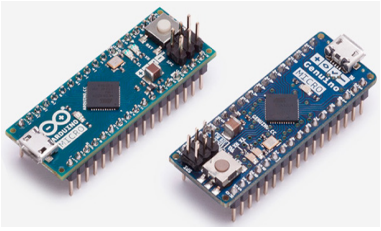
Tabla 7. ARDUINO LEONARDO.

ARDUINO LEONARDO		
 **Figura 11.** Arduino Leonardo.	***Microcontrolador:***	ATmega32u4
	Voltaje de Operación:	5V
	Pines digitales:	20
	Pines PWM:	7
	Pines de entradas análogas:	12
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2.5 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.8. Arduino Micro

Es la placa más pequeña de la familia Arduino, tiene todas las características del Arduino Leonardo, pero en un tamaño mucho más reducido para facilitar su conexión a protoboards.

Tabla 8. Arduino Micro.

ARDUINO MICRO		
 **Figura 12.** Arduino Micro.	***Microcontrolador:***	ATmega32u4
	Voltaje de Operación:	5V
	Pines digitales:	20
	Pines PWM:	7
	Pines de entradas análogas:	12
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2.5 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.9. Arduino Esplora

Arduino Esplora está basado en la placa Arduino Leonardo y tiene todos los circuitos básicos necesarios para la comunicación con la placa montada sobre la misma, facilitando el uso de esta para personas que desean ir directamente a la aplicación práctica sin necesidad de aprender cómo realizar conexiones de sensores.

Tabla 9. ARDUINO ESLORA.

ARDUINO ESPLORA		
 **Figura 13.** Arduino Esplora.	***Microcontrolador:***	ATmega32u4
	Voltaje de Operación:	5V
	Pines digitales:	-
	Pines PWM:	-
	Pines de entradas análogas:	-
	Corriente DC por cada pin I/O:	-
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2.5 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

Sabías que...

La placa Arduino ESPLORA fue retirado del mercado porque tuvo poca acogida.

Bibliografía y fuentes electrónicas

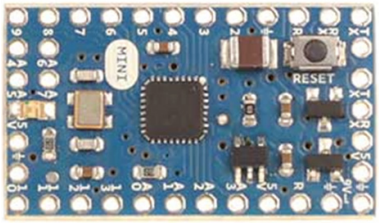
Aranda, D. (2014). *ELECTRÓNICA 3: Plataformas Arduino y Raspberry Pi.* Buenos Aires: Manual USERS.

Nicolas GOILAV, G. L. (2016). *Arduino: Aprender a desarrollar para crear objetos inteligentes.* Barcelona: Ediciones ENI.

1.5.10. Arduino Mini

Es una placa muy pequeña basada en ATmega328 diseñada para ser utilizada en protoboards para minimizar espacio.

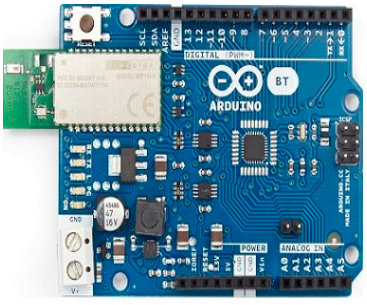
Tabla 10. ARDUINO MINI.

ARDUINO MINI		
 **Figura 14.** Arduino Mini.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	-
	Pines de entradas análogas:	8
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.11. Arduino BT

Es una placa basada en ATmega168 que incluye dentro de la misma un módulo bluetooth Bluegiga WT11 mediante el cual puede ser programado.

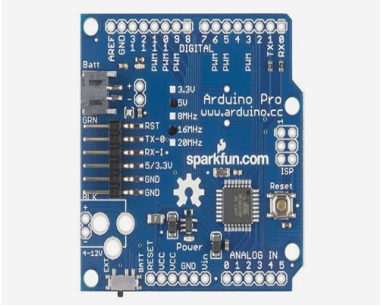
Tabla 11. ARDUINO BT.

ARDUINO BT		
 **Figura 15.** Arduino BT.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	8
	Pines PWM:	6
	Pines de entradas análogas:	-
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	500 mA
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.12. Arduino Pro

Está placa está diseñada para ser implementado de manera semi permanente, es decir, no se recomienda su reconexión sin una razón de fuerza mayor. Su reparto de pines es compatible con la mayor parte de shields de Arduino.

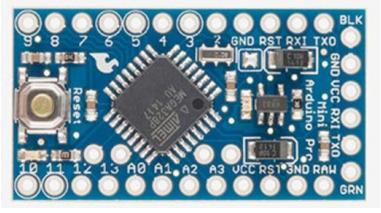
Tabla 12. ARDUINO PRO.

ARDUINO PRO		
 **Figura 16.** Arduino Pro.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.13. Arduino Pro Mini

Posee las mismas características de Arduino Pro pero en un tamaño menor y operativo a 3.3v y 5v.

Tabla 13. ARDUINO PRO MINI.

ARDUINO PRO MINI		
 **Figura 17.** Arduino Pro Mini.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

1.5.14. Arduino Yun Linux

Arduino yun basada en microprocesador Atheros le permite a la placa utilizar Linino, una distribución especial basada en Linux.

Tabla 14. ARDUINO YUN LINUX.

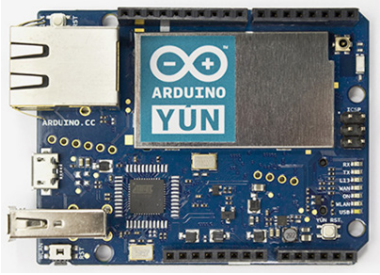
ARDUINO YUN LINUX		
	***Microcontrolador:***	Atheros AR933
	Voltaje de Operación:	3.3V
	Pines digitales:	20
	Pines PWM:	7
	Pines de entradas análogas:	12
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	16 KB
	Memoria SRAM:	2.5 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	16 MHZ

Figura 18. Arduino Yun Linux.

1.5.15. Arduino Duemilanove

Arduino Duemilanove, cuyo nombre significa 2009 en celebración del año en el que fue lanzado, es una placa basada en ATmega168.

Tabla 15. ARDUINO DUEMILANOVE.

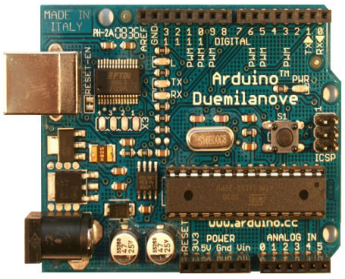
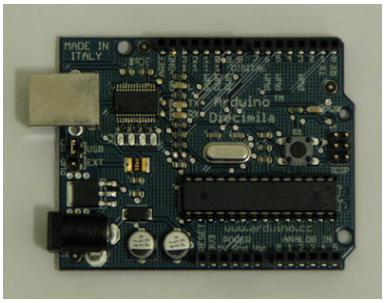
ARDUINO DUEMILANOVE		
	***Microcontrolador:***	ATmega168
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	0
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	16 MHZ

Figura 19. Arduino Duemilanove ATmega168.

1.5.16. Arduino Diecimila

Arduino Diecimilla, nombrado así en celebración de las 10.000 unidades de Arduino creadas.

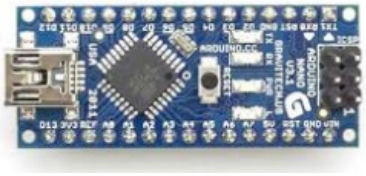
Tabla 16. ARDUINO DIECIMILA.

ARDUINO DIECIMILA		
 **Figura 20.** Arduino Diecimila.	***Microcontrolador:***	ATmega168
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	-
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	50 mA
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	16 MHZ

1.5.17. Arduino Nano ATmega168

Arduino Nano basado en una placa ATmega168.

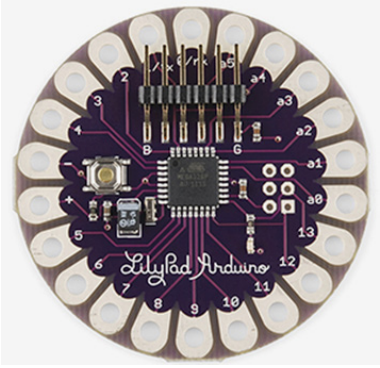
Tabla 17. ARDUINO NANO CON PLACA ATMEGA 168.

ARDUINO NANO		
 **Figura 21.** Arduino nano ATmega168.	***Microcontrolador:***	ATmega168
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	
	Pines de entradas análogas:	8
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	16 MHZ

1.5.18. Lilypad Arduino ATmega 328V

Lilypad fue diseñado y desarrollado por Leah Buechley y SparkFun Electronics, está orientada a obtener el máximo potencial con el menor consumo posible basado en un procesador ATmega328V.

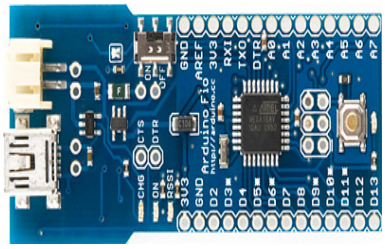
Tabla 18. ARDUINO LILYPAD.

ARDUINO LILYPAD		
 **Figura 22.** Arduino LilyPad ATmega328.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	9 MHZ

1.5.19. Arduino Fio

Arduino fio es una placa diseñada para comunicaciones inalámbricas.

Tabla 19. ARDUINO FIO.

ARDUINO FIO		
 **Figura 23.** Arduino Fio.	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	3.3 V
	Pines digitales:	8
	Pines PWM:	6
	Pines de entradas análogas:	8
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	8 MHZ

1.5.20. Arduino Pro

Mismas características de Arduino Pro pero de 3.3v.

Tabla 20. ARDUINO PRO ATMEGA328.

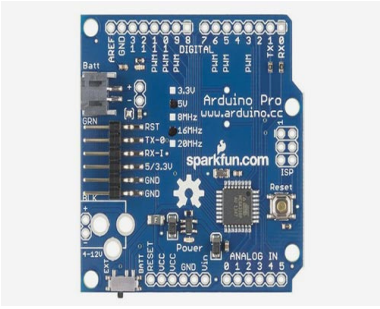
ARDUINO PRO		
	***Microcontrolador:***	ATmega328
	Voltaje de Operación:	3.3V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	8 MHZ

Figura 24. Arduino Pro ATmega328.

1.5.21. Arduino Pro mini

Mismas características de Arduino Pro mini pero de 3.3v.

Tabla 21. ARDUINO PRO.

ARDUINO PRO		
	***Microcontrolador:***	ATmega329
	Voltaje de Operación:	3.3V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	32 KB
	Memoria SRAM:	2 KB
	Memoria EEPROM:	1 KB
	Velocidad de reloj:	8 MHZ

Figura 25. Arduino Pro.

1.5.22. LilyPad Arduino

Lilypad basado en un procesador ATmega168V de 5v.

Tabla 22. ARDUINO LILYPAD ATMEGA 168.

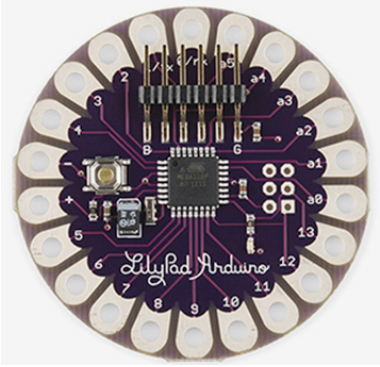
ARDUINO LILYPAD		
	***Microcontrolador:***	ATmega168
	Voltaje de Operación:	5V
	Pines digitales:	14
	Pines PWM:	6
	Pines de entradas análogas:	6
	Corriente DC por cada pin I/O:	40 mA
	Corriente DC en el pin de 3.5V:	-
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	8 MHZ

Figura 26. Arduino LilyPad ATmega168.

Sabías que...

Placa Arduino Lilypad

- Por su atractivo diseño es ampliamente utilizado para Wearables como collares, guantes, zapatos y ropa.

1.5.23. Arduino Gemma

Es una placa desarrollada e implementada por Adafruit y basada en ATtiny85.

Tabla 23. ARDUINO GEMMA.

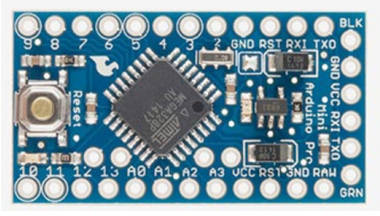
ARDUINO GEMMA		
	***Microcontrolador:***	ATmega168
	Voltaje de Operación:	3.3 V
	Pines digitales:	3
	Pines PWM:	2
	Pines de entradas análogas:	1
	Corriente DC por cada pin I/O:	20 mA
	Corriente DC en el pin de 3.5V:	
	Memoria Flash:	16 KB
	Memoria SRAM:	1 KB
	Memoria EEPROM:	0.5 KB
	Velocidad de reloj:	8 MHZ

Figura 27. Arduino Gemma.

1.6. Tipos de conectividad entre Arduino y el computador

1.6.1. Comunicación Serial

La comunicación serial o secuencial es el envío de datos bit a bit en un único canal o bus. La mayor ventaja que ofrece la comunicación en serie es que necesita un número muy reducido de lineas de transmisión o bus único por el cual se realizará la transferencia de la información. La comunicación en serie dado que posee un canal único compensa la falta de ancho de palabra con una frecuencia mucho más elevada.

Algunos shields utilizados para comunicación serial son:

Ethernet

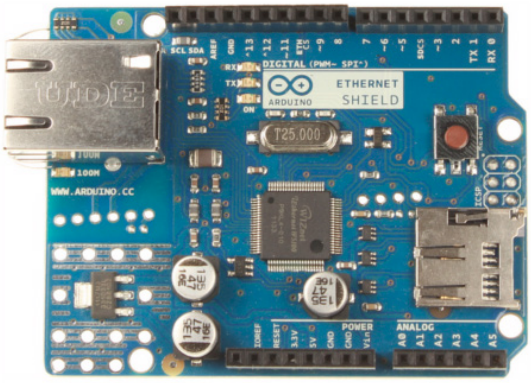


Figura 28. Ethernet.

Permite conectarse a una red mediante un cable RJ45 y posee un slot de tarjetas micro-SD para almacenamiento de archivos.

Wireless SD

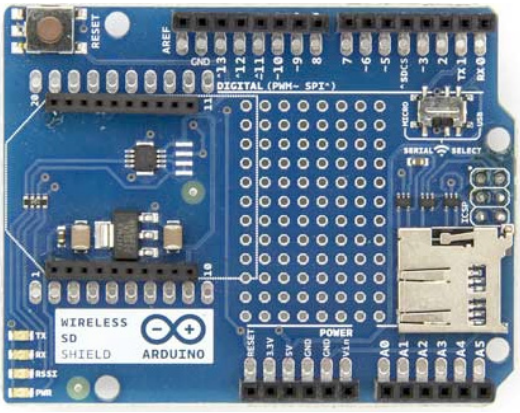


Figura 29. Wireless SD Shield.

Permite comunicación inalámbrica a 100 pies, posee un slot para tarjetas SD.

USB host Shield

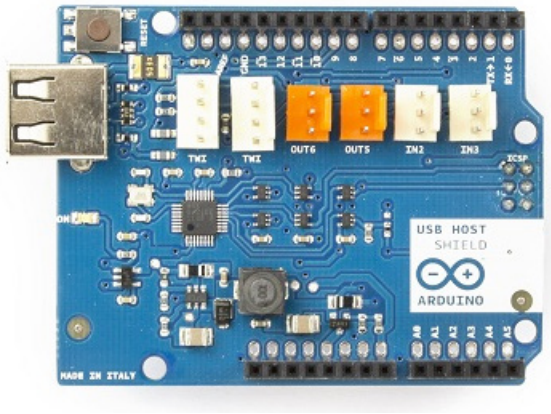


Figura 30. USB host Shield.

Permite conectar dispositivos USB y comunicarlos con la placa, posee una librería llama TinkerKit para una rápida conexión al módulo.

Otros Shield
Arduino proto Shield

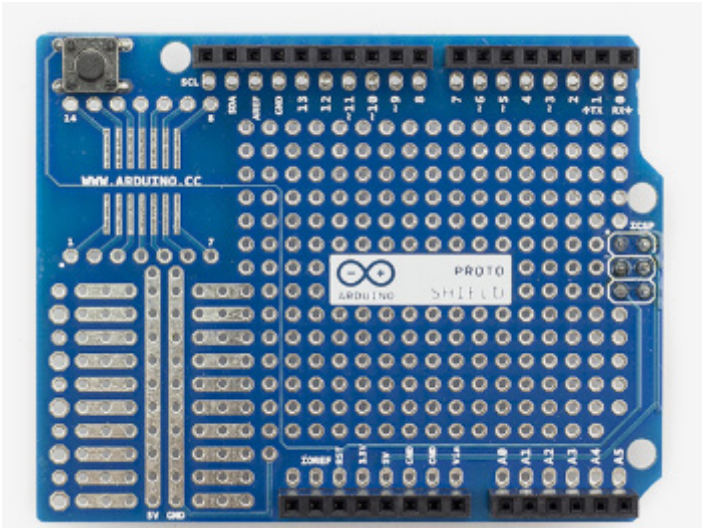


Figura 31. Arduino Proto Shield.

Permite crear circuitos a la medida e inclusive crear un shield a la medida.

Arduino provee una serie de shields (placas que se peuden conectar a las placas Arduino) para permitir aumentar sus funcionalidades o integrar circuitos de manera completa a la misma.

Los principales Shields utilizados son:

- Xbee: Permite la conexión de varias placas de manera inalámbrica

utilizando un módulo Maxtream Xbee Zigbee y puede reemplazar las conexiones USB.

- Motor Control v1.1: Permite la conexión y control de motores DC.
- RedFly-Shield: provee conectividad WiFi/ WLAN.
- Módulo MicroSD: Permite leer y escribir datos en memorias MicroSD.
- Shield de matriz de leds 8X8.

Para ver la lista completa de shields oficiales y no oficiales compatibles con Arduino visitar: shieldlist.org

1.6.2. Comunicación en Paralelo

A diferencia de la comunicación serial, la comunicación en paralelo posee una mayor cantidad de canales o buses de datos por los cuales enviará la información, esto quiere decir que requiere una menor latencia para el envío de información para alcanzar un alto rendimiento, pero sacrifica espacio físico para lograr este cometido. En la actualidad la comunicación en paralelo está cayendo en desuso debido a la alta latencia que ha logrado consolidar la comunicación serial y la necesidad de miniaturizar los componentes.

1.7. Protocolos de comunicación

Los protocolos de comunicación poseen un papel vital dentro del desarrollo de proyectos que requieren comunicación constante con el servidor o con los dispositivos, es por eso que se desarrollaron protocolos diseñados específicamente para facilitar la transmisión de información de manera rápida y ligera.

Para poder realizar procesos de comunicación eficientes se desarrolló el MQTT un protocolo desarrollado específicamente para el IoT.

1.7.1. MQTT

Figura 32. Protocolo MQTT.

MQTT es un protocolo completamente creado en 1999 por el Dr. Andy Stanford-Clark de IBM y Arrlen Nipper de Arcom. Es un protocolo abierto, de fácil implementación, además sencillo y ligero. es ideal para responder a las necesidades del IoT dado que está adaptado para utilizar un mínimo de espacio en el ancho de banda, lo cual lo hace ideal para ser usado en redes inalámbricas, MQTT es ampliamente utilizado en una gran variedad de artefactos tanto industriales como particulares.

Entre las principales características de MQTT se pueden notar:

- Bajo consumo energético.
- Rápido y ligero para el envío y recepción de información.

- Requiere pocos recursos para su adecuado funcionamiento.

MQTT está basado en una estructura suscriptor/publicador donde el servidor (suscriptor) llamado broker recibe los datos de los publicadores y en caso de ser necesario, enviará datos a los publicadores únicamente bajo petición del broker. Ningún mensaje puede jamás poseer un tamaño superior a 256 Mb.

Principios de MQTT:

1. Los dispositivos de campo deben poseer apoyo satelital o celular para la transferencia de información, cada kb importa, el tráfico es muy costoso.
2. Permitir comunicación en doble vía en redes con problemas para acelerar la transmisión de información.
3. Mantener el consumo energético lo más bajo posible para obtener máximo rendimiento de dispositivos con baterías.
4. Los dispositivos pueden estar en estado pasívo (Dormidos) pero no más del 95% de tiempo.

Sabías que...

Protocolo MQTT

Es el más utilizado en proyectos de internet de las cosas dado su bajo consumo de recursos.

1.7.2. CoAP

Figura 33. Protocolo CoAp.

CoAP (Constrained Application Protocol) es un protocolo de transferencia especializado en la red para el uso de nodos y redes compactadas en el IoT. este protocolo está diseñado para transferencias máquina a máquina (M2M) tales como Smart Energy o Smart Building.

Las características de coAp son:

1. **Modelo de Reposo para los dispositivos pequeños.**

Los dispositivos solo envían mensajes en caso de ser necesario mediante métodos como GET, PUT, POST o DELETE.

2. **Hecho para millones de nodos de todo tipo.**

CoAp permite a microcontroladores desde 10 Kb de ram y 100 Kb de espacio direccionable conectarse de manera adecuada.

3. **Transferencia de habilidades existentes (HTTP COAP).**

CoAP utiliza un sistema muy similar a HTTP por lo que un programador que posea experiencia HTTP puede fácilmente adaptarse a desarrollo COAP.

4. **Seguridad.**

CoAp por defecto implementa parámetros de seguridad que equivalen a llaves RSA de 3072-bit.

5. **Fácil integración.**

CoAP comparte características con HTTP por lo que pueden ser conectados de manera sencilla.

6. **Descubrimiento integrado.**

Cada nodo con CoAP tendrá un modelo de identificación para permitir a los demás nodos descubrirlos de manera sencilla.

7. **Escoger modelos de datos.**

CoAP puede tener distintos tipos de modelos mismos que podrán integrar XML, JSON, CBOR o cualquier formato de datos.

1.7.3. Rest API

Figura 34. Protocolo Rest API.

REST (Representational State Transfer) es un estilo de arquitectura para sistemas hipermedia distribuidos. REST es un estilo híbrido derivado de varias arquitecturas orientadas a la red.

Las restricciones establecidas son:

1. **Interfaz uniforme**

Esta restricción define que la interfaz existente entre clientes y servidores debe mantenerse igual en todos los nodos, lo cual permite que pese al desenvolvimiento individual de cada nodo la información se transfiera de igual manera permitiendo conectividad todo el tiempo.

2. **Sin estado**

Para minimizar el ancho de banda ocupado por una sesión abierta convencional, el sistema REST implementa un envío único de información en el que se dan todas las indicaciones necesarias al nodo receptor para el adecuado tratamiento de la información sin necesidad que este último tenga que comunicar el estado del proceso de la información.

3. **Cacheable**

Las comunicaciones pueden o no almacenar un caché de las mismas para ser utilizadas en futuras transacciones minimizando el tiempo de procesamiento.

4. **Cliente servidor**

El cliente y el servidor poseen estructuras internas distintas, centradas a la tarea de cada uno de estos.

5. **Sistema en capas**

REST permite la implementación de servidores intermediarios para mejorar la escalabilidad del sistema activando el balance de carga.

1.7.4. XMPP

Figura 35. Protocolo XMPP.

XMPP (Extensible Messaging and Presence Protocol) es un conjunto de tecnologías de código abierto para proveer a los dispositivos de mensajería en tiempo real. Presencia y mensajería a varios receptores, transmisión de voz y video, es considerado como un protocolo usable en IoT pero tiene como principal desventaja el alto consumo energético debido a la presencia que los dispositivos deben mantener en la red.

Sus principales características son:

1. **Abierto**

XMPP es de código abierto, público y gratuito para la implementación necesaria y cambios que se puedan realizará para adaptarlo a necesidades específicas.

2. **Descentralizado**

Cada persona puede correr su propio servidor XMPP por lo que permite tener control absoluto sobre las comunicaciones.

3. **Seguro**

Los servidores XMPP pueden ser aislados de los demás servidores. Además, implementa el sistema SASL y TLS.

4. **Extensible**

Se puede añadir funcionalidades nuevas utilizando XML dado que XMPP es de código abierto.

5. **Flexible**

Los servidores XMPP son extremadamente flexibles ya que permiten incluso compartir archivos.

CAPÍTULO II: PRÁCTICAS DE ARDUINO ORIENTADAS AL INTERNET DE LAS COSAS

2.1. Objetivo

- Dar a conocer aplicaciones prácticas con la utilización de Arduino orientadas al internet de las cosas.
- Brindar al lector una guía práctica del paso a paso, que permita replicar de forma fácil y sencilla cada una de las prácticas desarrolladas.
- Ejemplificar cada uno de los materiales a utilizar para el desarrollo de las prácticas.

2.2. Introducción

El presente capitulo busca mostrar de manera completa y detallada ejemplos prácticos aplicados del correcto uso de una interfaz de Arduino y el internet de las cosas, para poder obtener y enviar información mediante diversos protocolos.

La aplicación de los conocimientos teóricos en el campo práctico y el desarrollo de hardware es en general un tópico de gran dificultad para personas con poca práctica, quienes quizás puedan incluso tener temor ante el mismo.

En los últimos años se ha difundido la idea del uso de software y hardware libre para impulsar el desarrollo de nuevas aplicaciones de los mismos de manera gratuita y brindando a los usuarios finales una experiencia sencilla y muy gratificante.

Es por eso que se han elegido plataformas de desarrollo OpenSource y Hardware Libre Arduino para permitirle al lector incursionar dentro del emocionante mundo del desarrollo de soluciones.

2.3. Adquisición de datos en tiempo real de sensores utilizando protocolo de comunicaciones MQTT, Ethernet Shield y Arduino uno

2.3.1. Objetivos

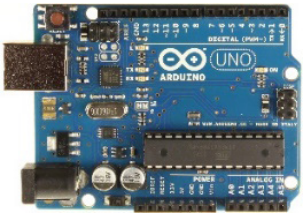
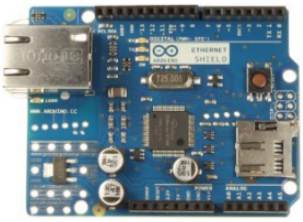
- Poner en práctica el protocolo mqtt bajo la modalidad suscriptor-publicador.
- Lectura de sensores (humedad y temperatura) garantizando tiempos de respuesta inmediatos.
- Complementar las funcionalidades del Arduino UNO con la ayuda de la tarjeta Ethernet Shield para tener acceso a la red.

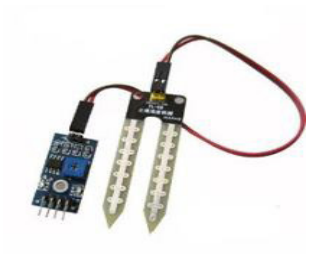
2.3.2. Descripción

El presente proyecto hace uso del protocolo MQTT y lenguaje de programación nodejs:

- MQTT es un protocolo de comunicación que trabaja máquina a máquina (M2M) / protocolo de comunicación. Está diseñado como un protocolo extremadamente ligero para mensajería. Es útil para las conexiones donde se requiere limitar el ancho de banda de red.
- Nodejs es un intérprete JavaScript del lado del servidor que cambia la noción de cómo debería trabajar un servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en solo una máquina física.

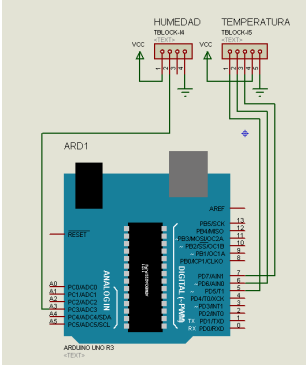
2.3.3. Materiales

DESCRIPCIÓN	REPRESENTACIÓN
HARDWARE	
Arduino UNO	
Tarjeta Ethernet Shield	
Sensor de temperatura de agua 556	

Sensor de humedad	
Cable de red directo	
Cable USB Arduino	

2.3.4. Desarrollo

2.3.4.1. Diagrama esquemático de la actividad



2.3.4.2. Descripción del funcionamiento del circuito

La presente práctica implementa el uso del protocolo de comunicación mqtt en una WSL (Wireless Network Sensor) haciendo uso de la tecnología Arduino UNO.

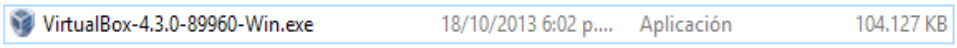
- Al tener el WSN funcionando, se evidencia el funcionamiento del protocolo mqtt bajo la modalidad suscriptor – publicador.
- El "sensor de humedad" capta el porcentaje de humedad de la tierra, luego envía dicha información a la placa Arduino UNO, mismo que se encarga de procesarla y la publicarla mediante el protocolo mqtt, lo cual va ser visualizado en el cliente Web.

- El “sensor de temperatura” capta la temperatura, luego envía dicha información a la placa Arduino UNO, mismo que se encarga de procesarla y la publicarla mediante el protocolo mqtt, lo cual va ser visualizado en el cliente Web.

2.3.5. Implementación del software

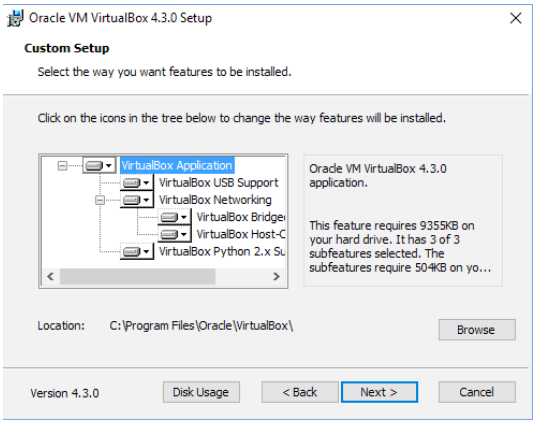
2.3.5.1. Instalación de la máquina virtual, Virtual Box 4.3.0.

Paso 1: Doble click en el instalador que se encuentra en la carpeta de recursos.

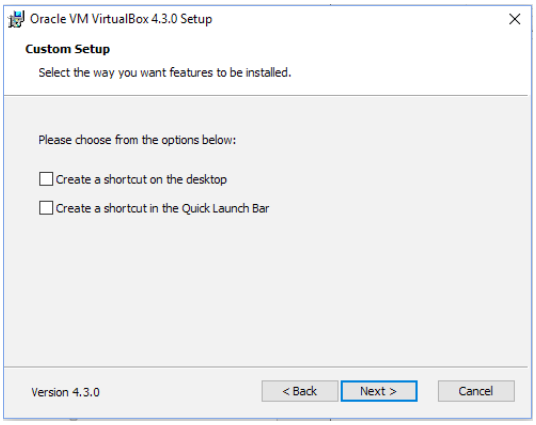


Paso 2: Click en siguiente (Next).

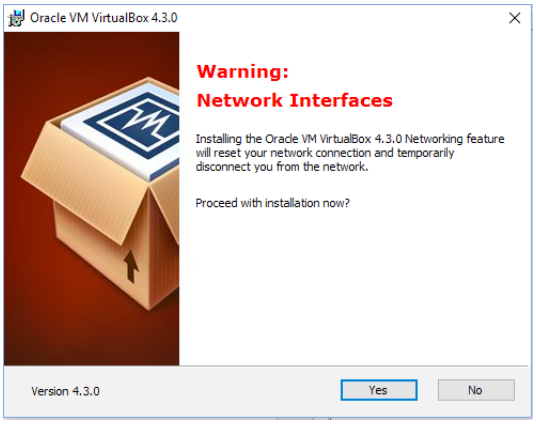
Paso 3: Click en siguiente (Next).



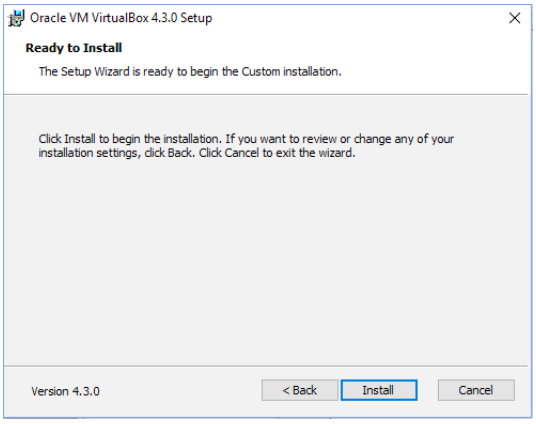
Paso 4: Click en siguiente (Next).



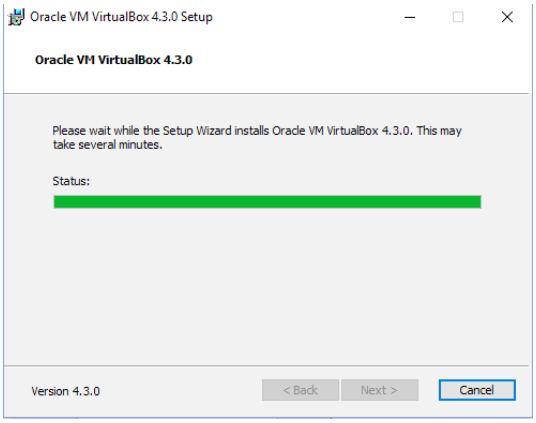
Paso 5: Click en Si (Yes).



Paso 6: Click en Install (Instalar).



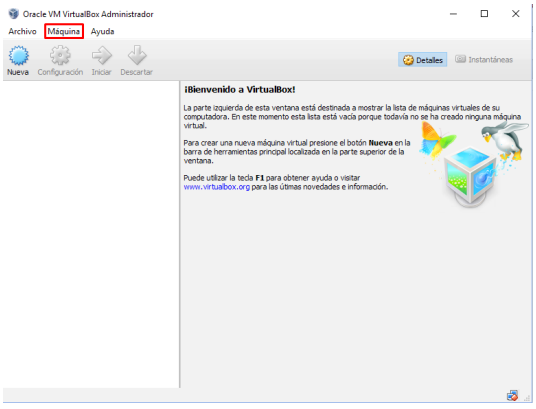
Paso 7: Esperamos que la barra de estado complete su progreso. Click en siguiente (Next).



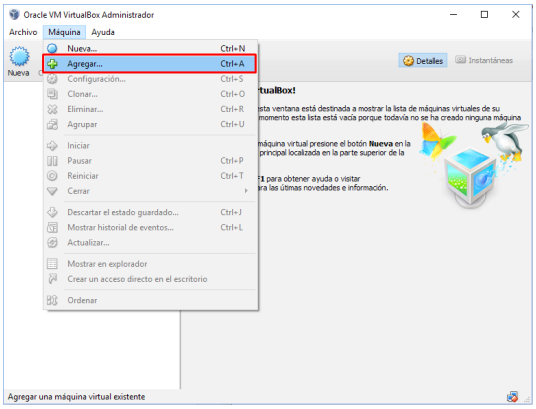
Paso 8: Marcar opción de pantalla para que inicie la máquina virtual y click en Finalizar (Finish).

2.3.5.2. Importar y configurar la máquina virtual del sistema operativo Débian donde se encuentra pre-configurado el servidor bróker

Paso 1: Una vez abierta la interfaz de Virtual Box, en la barra superior ubicar la opción Máquina y dar click.

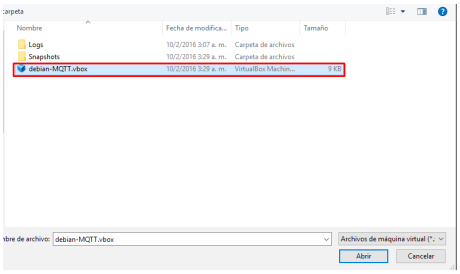


Paso 2: Entre las opciones desplegadas ubicar la opción Agregar, dar click.

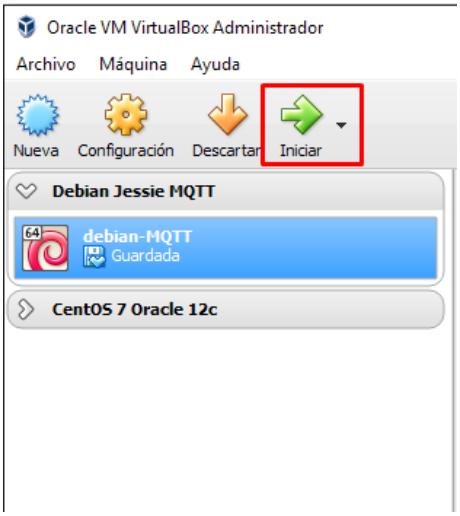


Paso 3: En la carpeta de recursos ubicar la siguiente carpeta. Dar doble click.

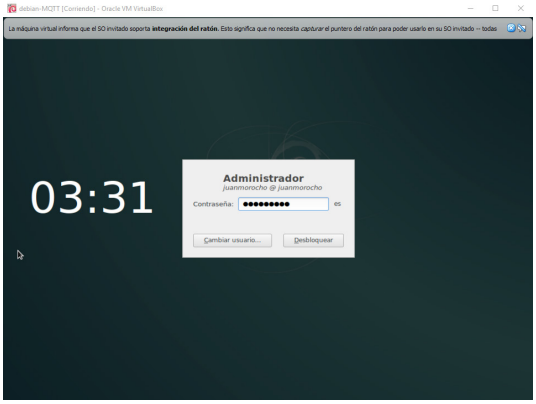
Paso 4: Elegir la máquina virtualizada con el sistema operativo Debian que contiene el servidor bróker. Click en Abrir.



Paso 5: Seleccionar la máquina virtualizada y click en la opción superior Iniciar.

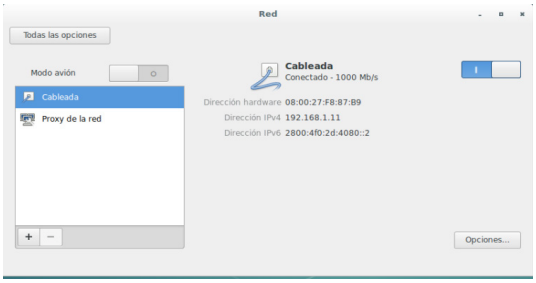


Paso 6: Escribir contraseña juan12345. Click en desbloquear.

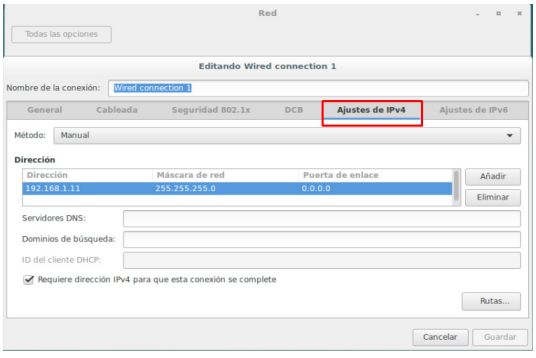


Paso 7: Configurar dirección IP de la máquina virtual. Click en preferencia de red.

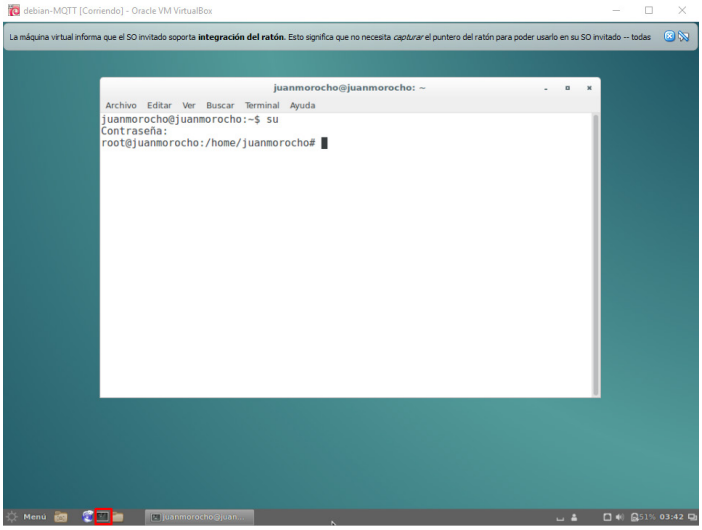
Paso 8: Click en opciones.



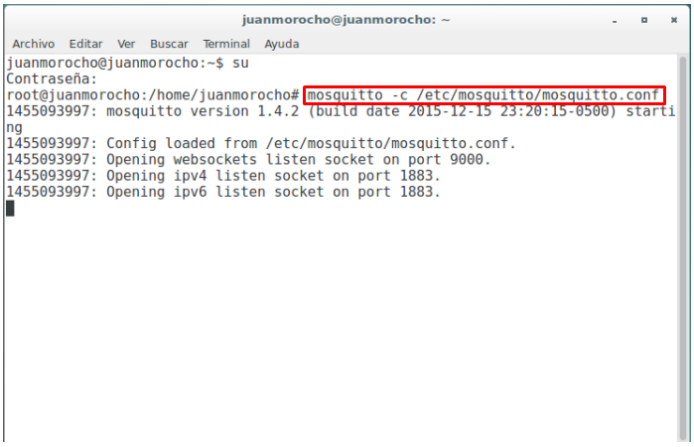
Paso 9: Ubicarse en la pestaña de Ajustes de IPv4 e ingresar dirección IP a utilizar. Finalmente, click en Guardar.



Paso 10: Abrir la terminal, ingresar como usuario root y escribir la contraseña: 12345678. Presionar Enter.



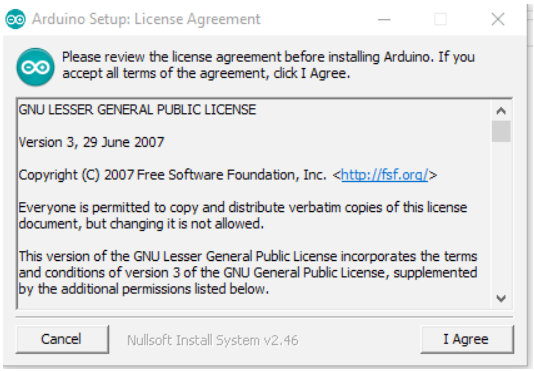
Paso 11: Escribir la siguiente línea de comandos en la terminal. Presionar Enter.



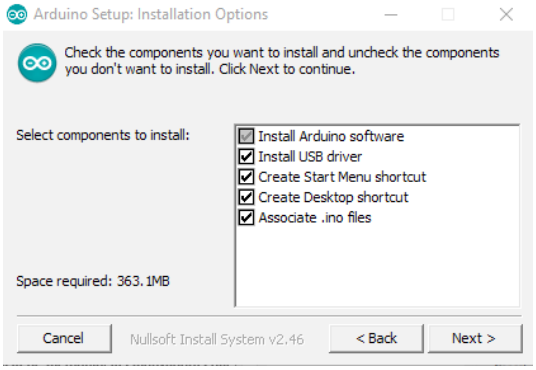
2.3.5.3. Instalación de Arduino IDE

Paso 1: Ubicar en la carpeta de recursos el instalador del IDE Arduino y dar doble click.

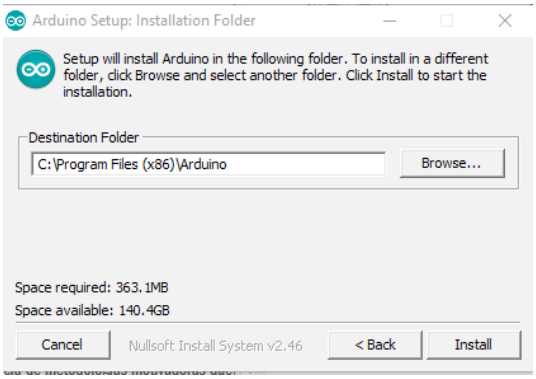
Paso 2: Aceptar las condiciones. Click en Acepto (I Agree).



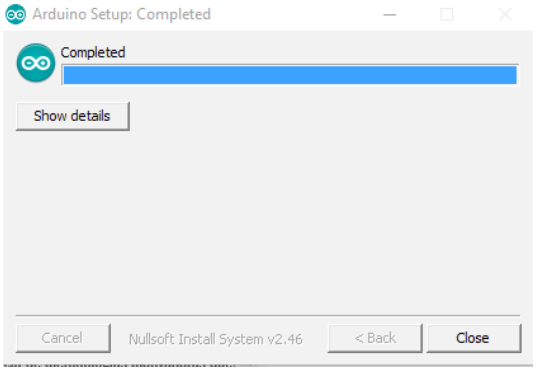
Paso 3: Click en Siguiente (Next).



Paso 4: Click en Instalar (Install).

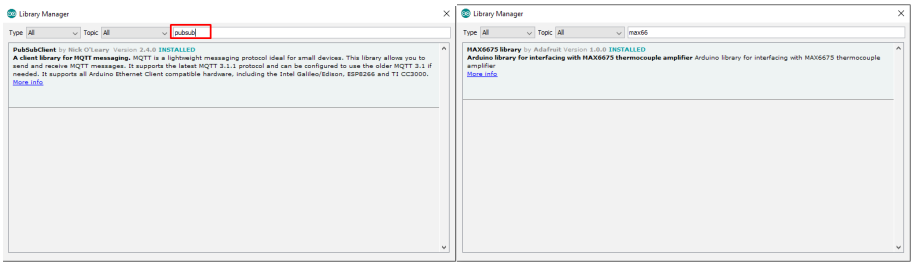


Paso 5: Esperar que la barra de estado de instalación finalice y click en Cerrar (Close).



Paso 6: Posterior a la instalación del IDE, instalar las librerías necesarias desde la opción Programa, Include Library, Manage Library.

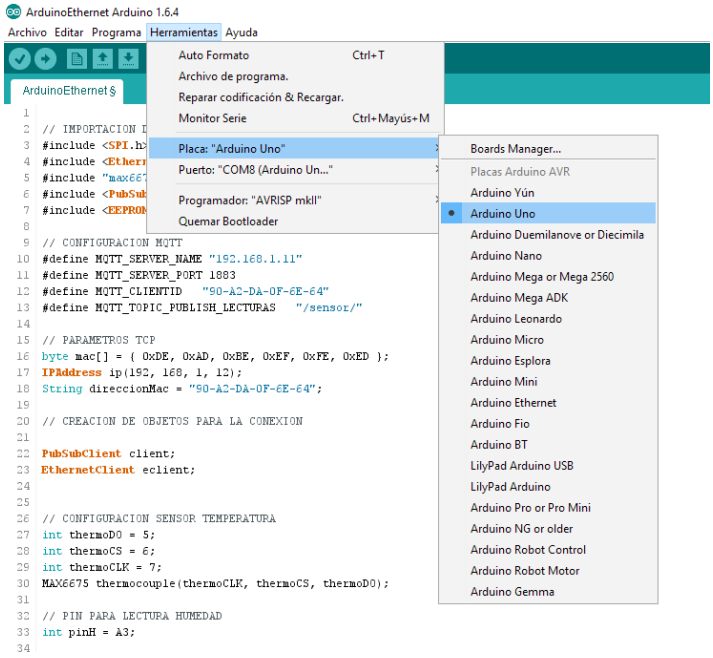
Paso 7: Buscar las librerías a instalar y click en Instalar (install).



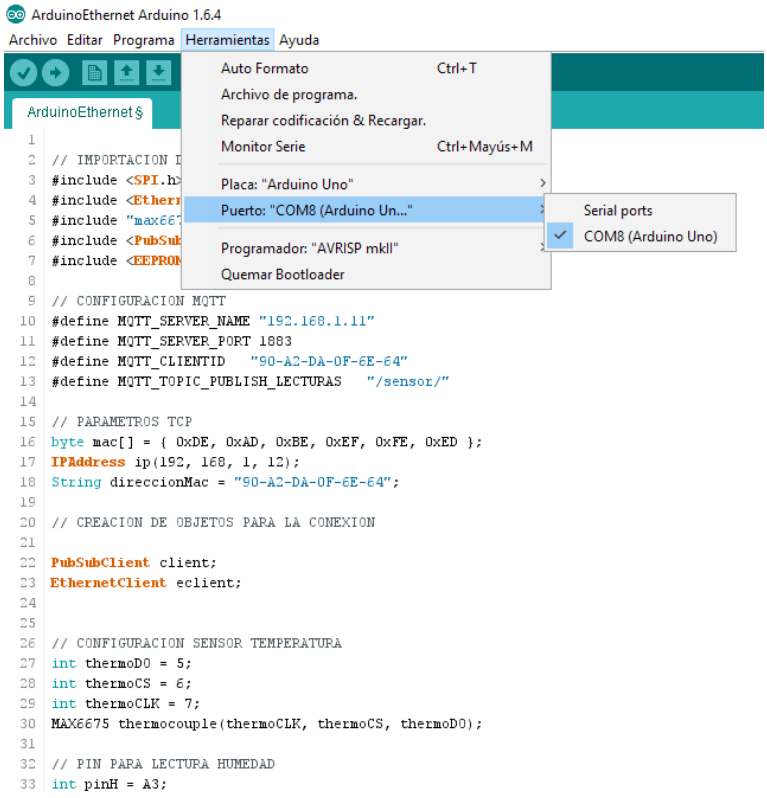
Paso 8: Ahora se puede empezar a programar.

NOTA: Ver código de programación en Anexos. (Anexo 1)

Paso 9: Dirigirse a la opción herramientas y ubicar la opción Placa, seleccionar Arduino UNO.



Paso 10: Dirigirse a la opción herramientas y ubicar la opción Puerto, seleccionar el puerto en que se encuentre conectado mediante cable la tarjeta Arduino a la computadora.

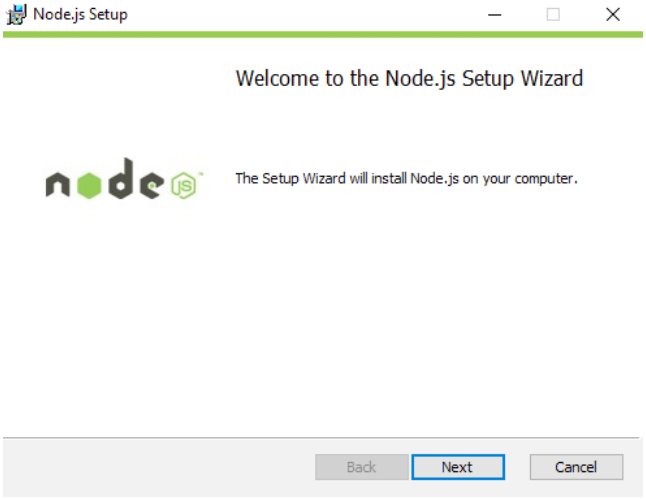


Paso 11: Grabar código en la tarjeta Arduino UNO.

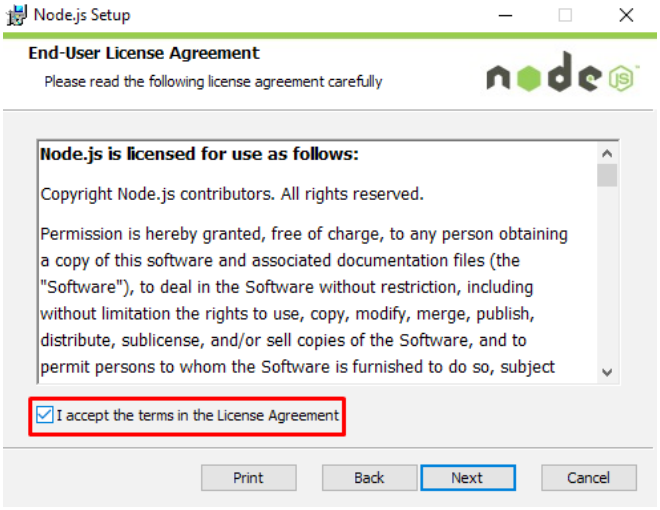
2.3.5.4. Instalación de Node JS

Paso 1: Ubicar en la carpeta de recursos el instalador de Node JS y dar doble click.

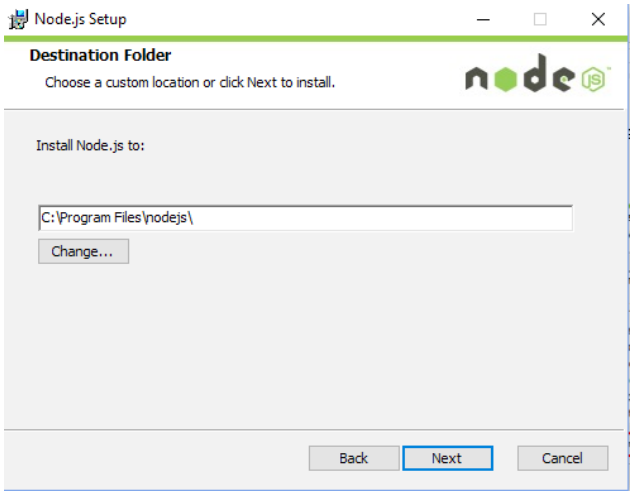
Paso 2: Click en Siguiente (Next).



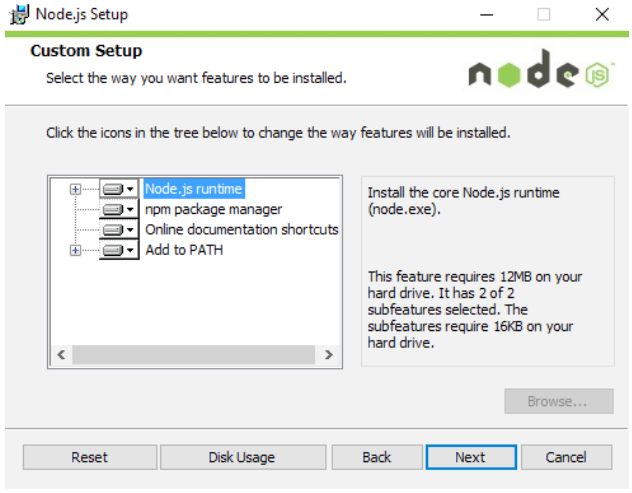
Paso 3: Seleccionar la casílla para aceptar las condiciones de instalación y click en botón Siguiente (Next).



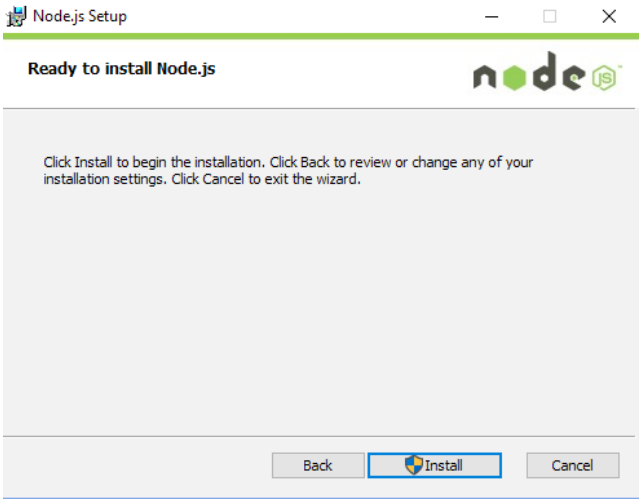
Paso 4: Click en el botón siguiente (Next).



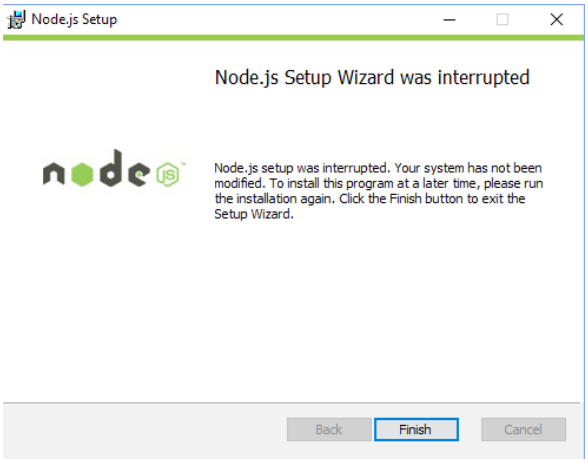
Paso 5: Click en botón Siguiente (Next).



Paso 6: Click en Instalar (Install).



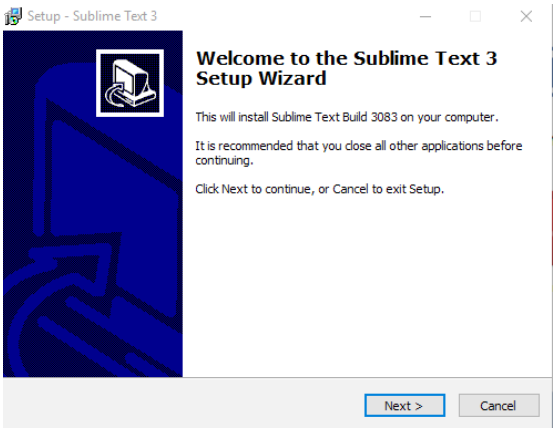
Paso 7: Click en Finalizar (Finish).



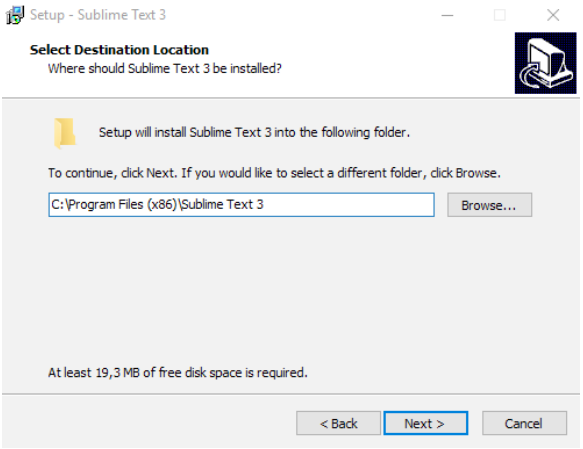
2.3.5.5. Instalación del editor de textos "Sublime Text"

Paso 1: En la carpeta de recursos ubicar el instalador de Sublime Text, y doble click.

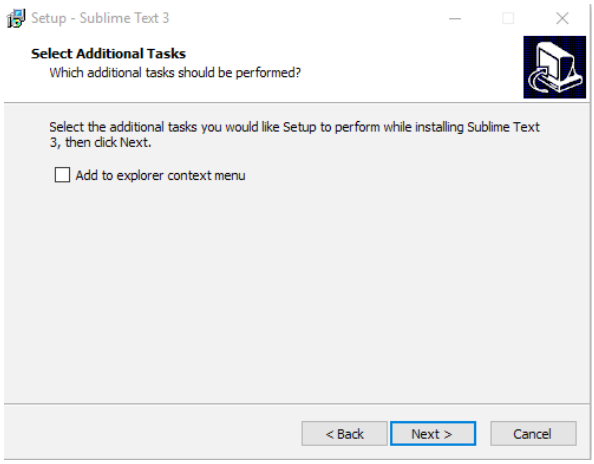
Paso 2: Click en Siguiente (Next).



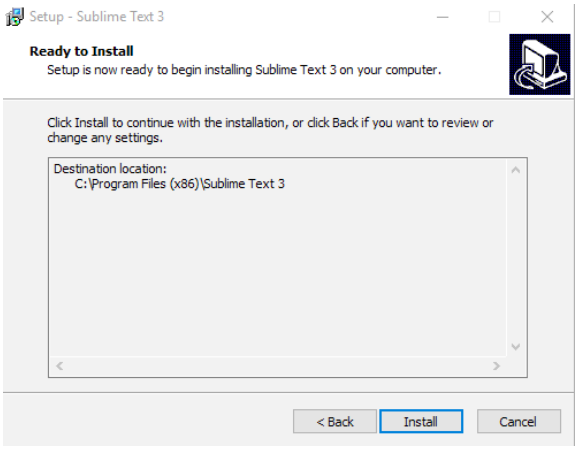
Paso 3: Click en Siguiente (Next).



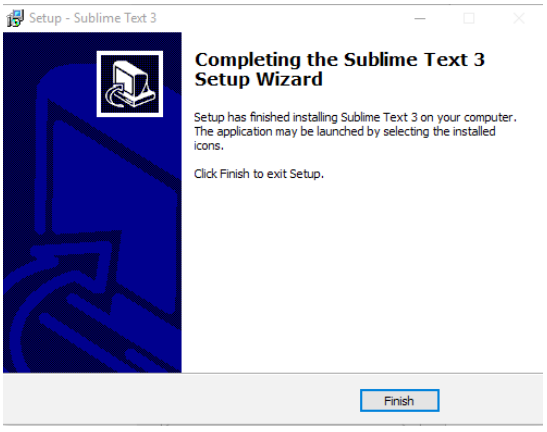
Paso 4: Click en Siguiente (Next).



Paso 5: Click en Instalar (Install).



Paso 6: Click en Finalizar (Finish).



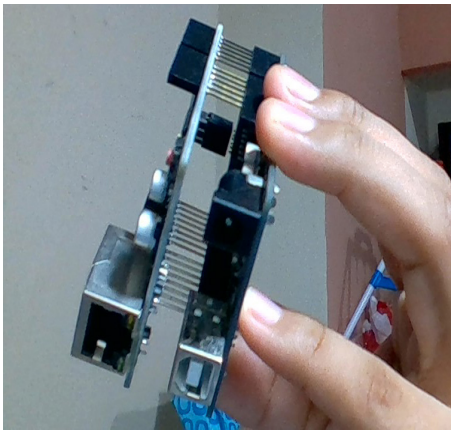
Paso 7: Una vez abierta la aplicación se puede empezar a programar.

NOTA: Ver código de programación en Anexos. (Anexo 2)

2.3.6. Implementación del hardware

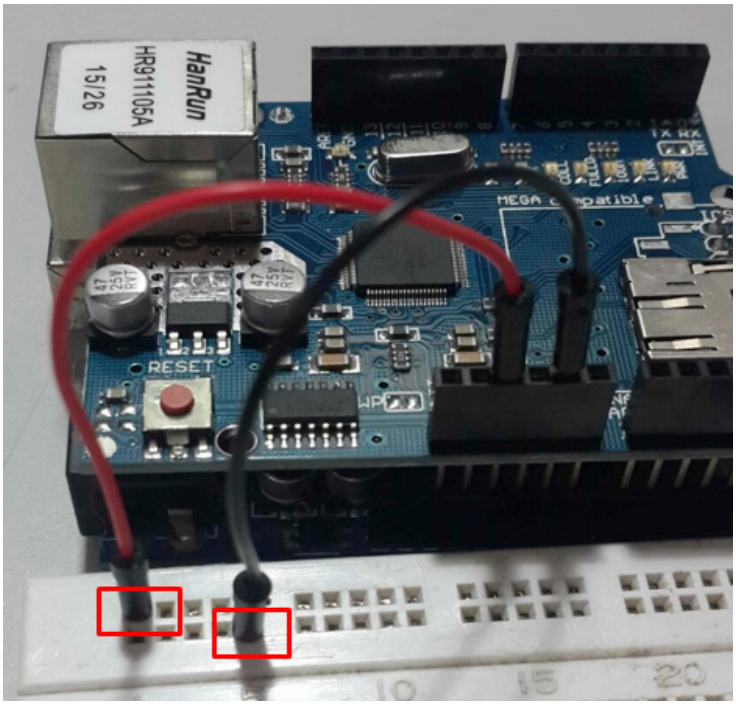
2.3.6.1. Conexión de placa Arduino UNO con la Ethernet Shield

Paso 1: Ubicar los pines del Ethernet Shield en las ranuras de la placa Arduino UNO.

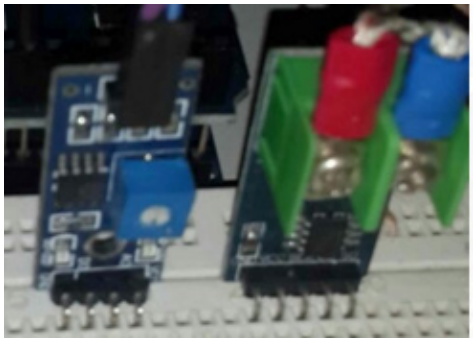


Conexión de sensores con la tarjeta Arduino UNO.

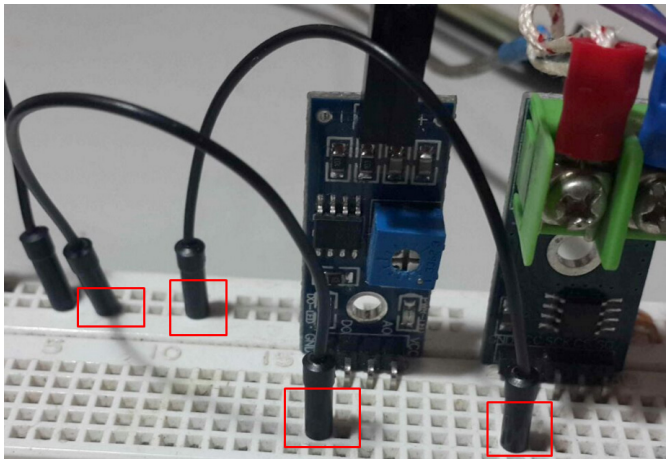
Paso 1: Pin GND del Arduino y pin VCC 5 Voltios conectar al protoboard.



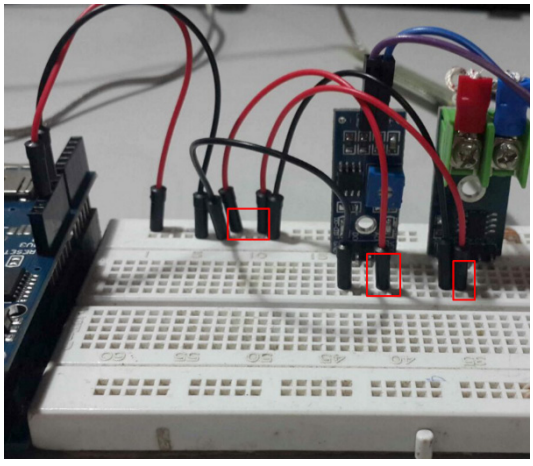
Paso 2: Conectar el sensor de humedad y el sensor de temperatura al protoboard.



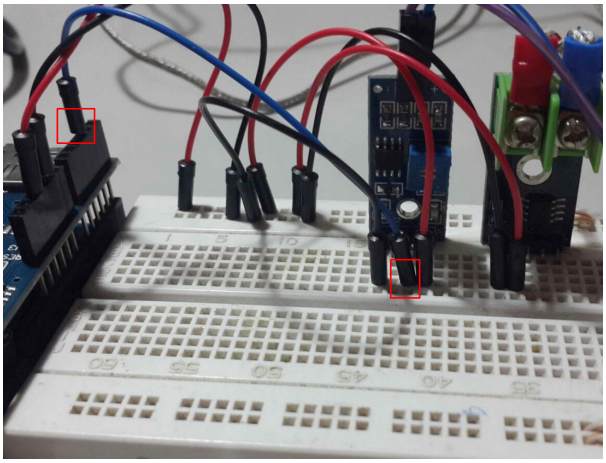
Paso 3: Conectar del pin GND a los sensores de temperatura y humedad.



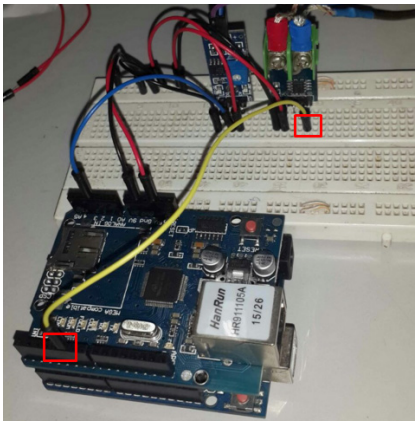
Paso 4: Conectar el pin VCC a los sensores de temperatura y humedad.



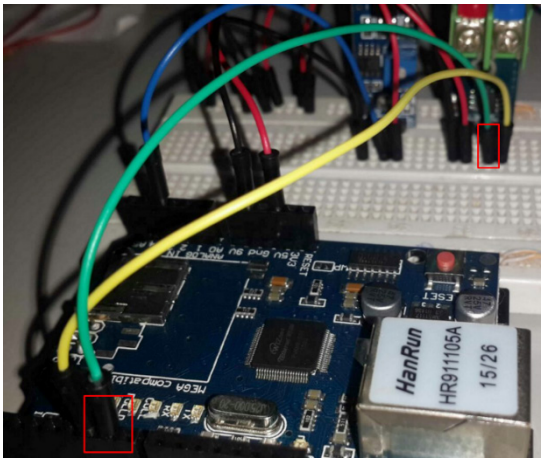
Paso 5: Conectar el pin A3 del Arduino al pin AO del sensor de humedad.



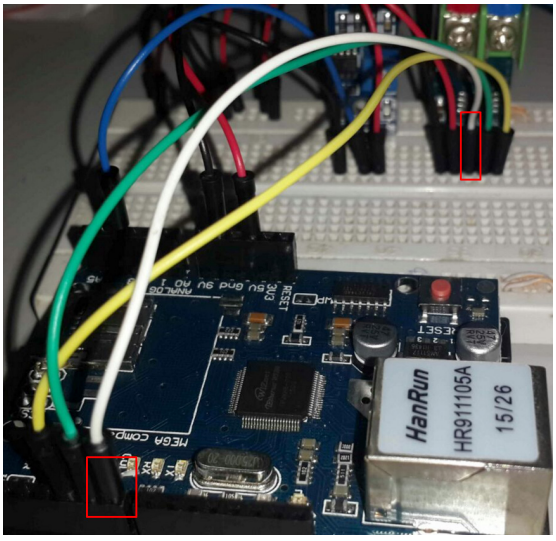
Paso 6: Conectar el pin 5 del Arduino al pin S0 del sensor de temperatura.



Paso 7: Conectar el pin 6 del Arduino al pin CS del sensor de temperatura.



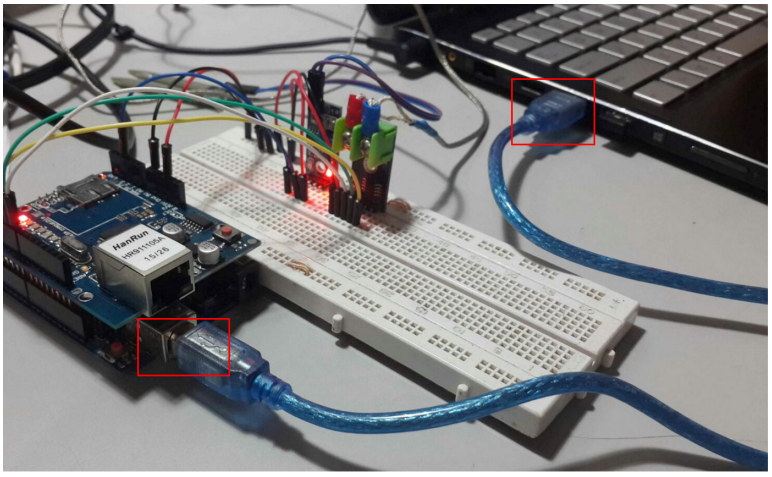
Paso 8: Conectar el pin 7 del Arduino al pin SCK del sensor de temperatura.



2.3.6.2. Conexión para grabar código de programación (ver código de programación) a la tarjeta Arduino

Previo a la grabación del código Arduino es necesario realizar la siguiente conexión.

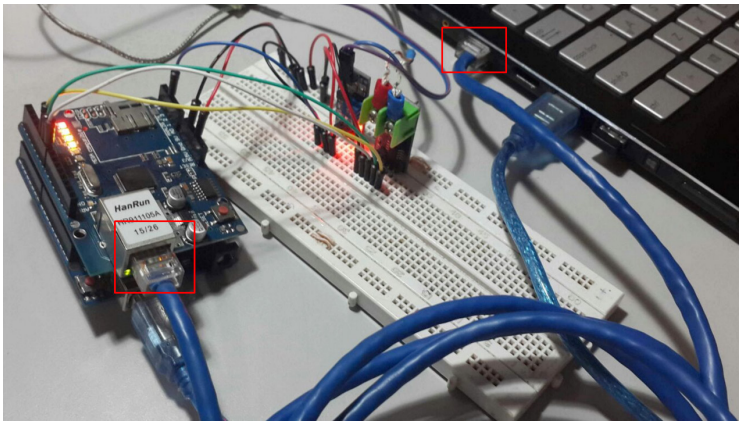
Paso 1: Conectar cable USB de entre el Arduino y una computadora.



Paso 2: Haciendo uso del Arduino IDE, escribir código de programación de la tarjeta Arduino UNO.

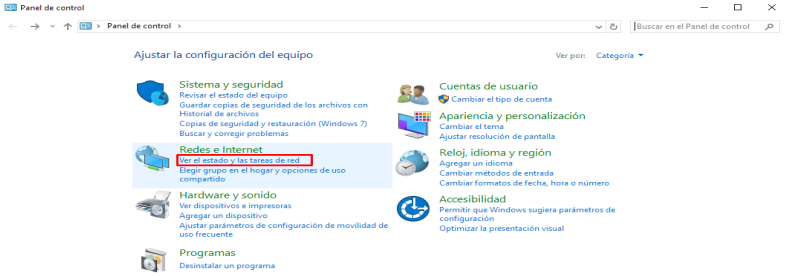
2.3.6.3. Conexión del cable de red a la tarjeta Arduino

Paso 1: Conectar el cable de red entre el Ethernet Shield y el puerto de red de la computadora.

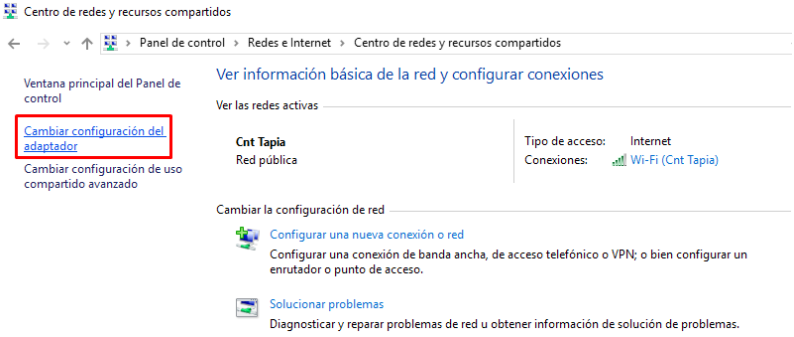


2.3.6.4. Configuración de la tarjeta de red de la computadora

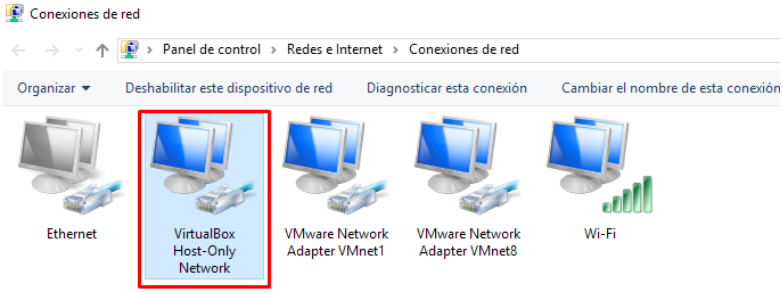
Paso 1: Ingresar al panel de control. Click en la opción Ver estado de tarjeta de red.



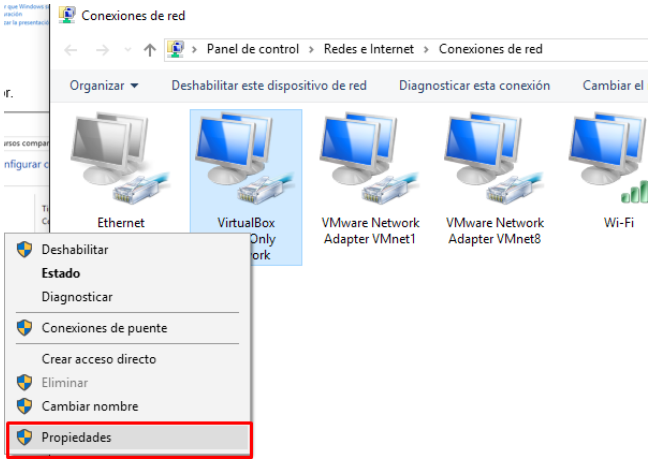
Paso 2: Click en Cambiar configuración de adaptador.



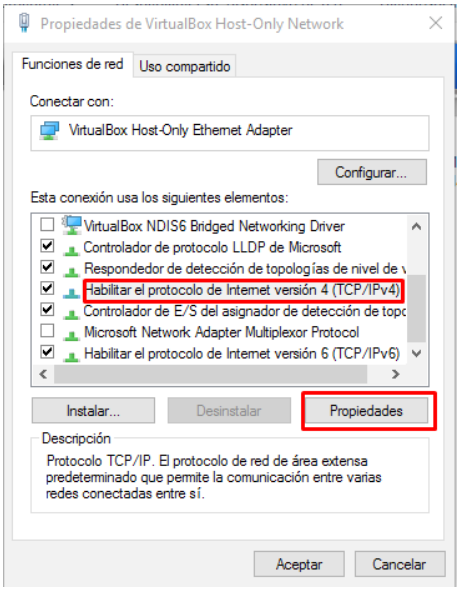
Paso 3: Ubicarse en Virtual Box Network.



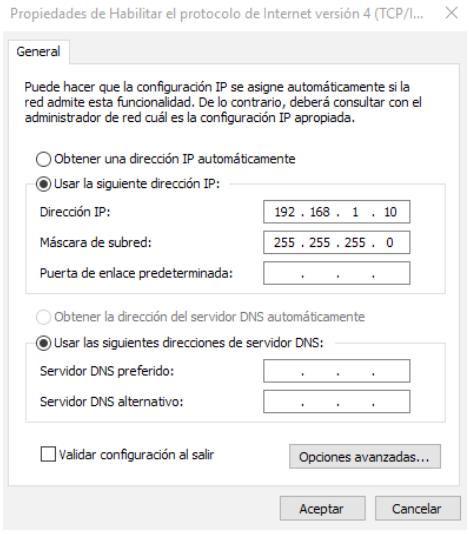
Paso 4: Click derecho y click en la opción Propiedades.



Paso 5: Click en Habilitar el protocolo de internet versión 4. Click en propiedades.

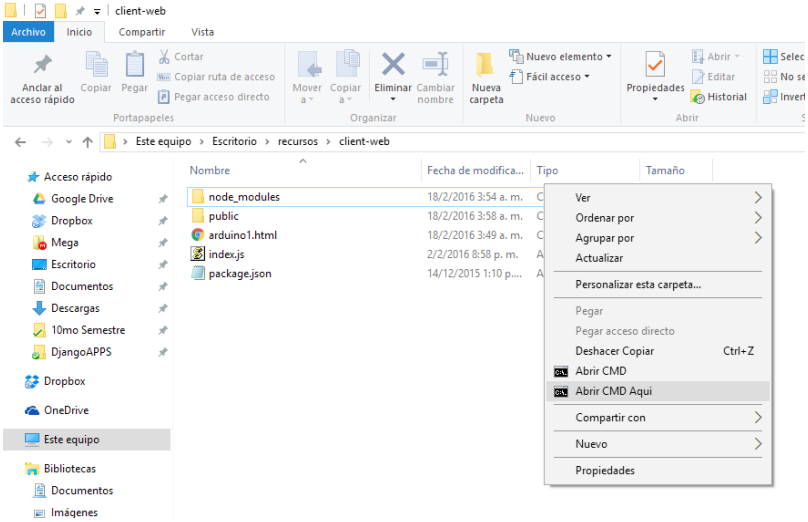


Paso 6: Realizar la siguiente configuración de direccionamiento IP. Click en Aceptar.



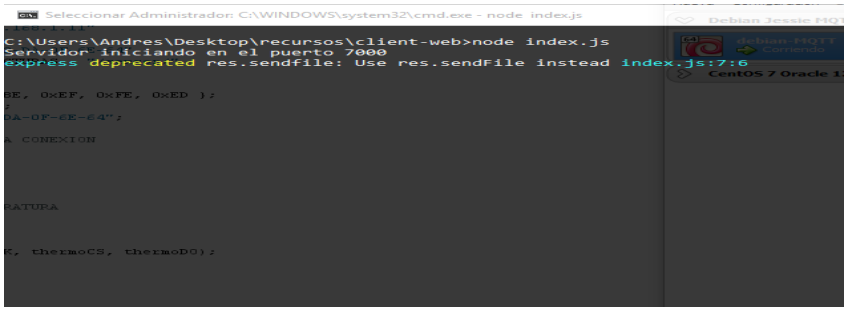
2.3.7. Detalles de configuración y ejecución de la práctica

Ir a la carpeta del cliente Web ubicado en la carpeta de recursos, click derecho y opción Abrir CMD Aquí, en una opción de permisos click en opción SI.



Para instalar dependencias de node js escribir el siguiente comando: node install y presionar Enter.

Para poner en marcha el cliente Web, ir a la carpeta del cliente Web ubicado en la carpeta de recursos, click derecho y opción Abrir CMD Aquí, en una opción de permisos click en opción SI, escribir el comando: node index.js y presionar Enter.



En el navegador de preferencia escribir la ruta: localhost:7000 y presionar Enter.

Se puede visualizar los estados de los sensores tanto de temperatura como de humedad.

2.4. Letrero led con conexión desde un Smartphone Vía Bluetooth

2.4.1. Objetivos

- A través de una aplicación Android enviar un mensaje a una matriz 8x8 en tiempo real.
- Aprender el manejo del módulo Arduino Uno R3, la Matriz 8x8 y el módulo de Bluetooth HC-05.

2.4.2. Descripción

La presente práctica hace uso del Módulo Bluetooth HC-05, el integrado MAX7219CNG y la Matriz 8x8 led:

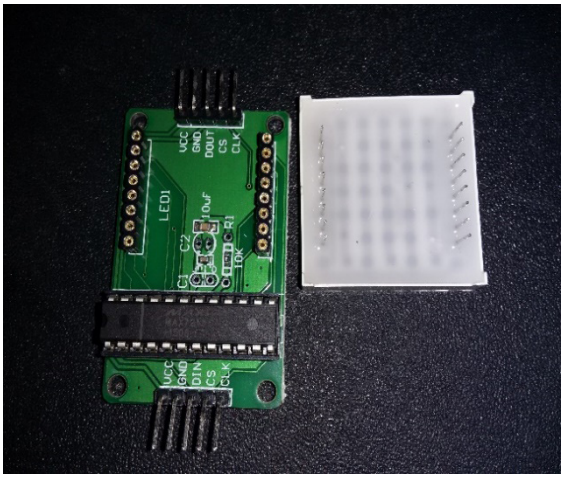
1. La Matriz de LEDs de 8×8, compuesta por 64 LEDs de 3mm cada uno color rojo.
2. Está montado con cátodo común dentro de un encapsulado sólido de fibra de vidrio como protección para los LEDs.
3. Trabaja con el circuito integrado MAX7219 de Maxim son para la conducción, ya sea individuales 64 LEDs, o hasta 8 dígitos de 7 segmentos.

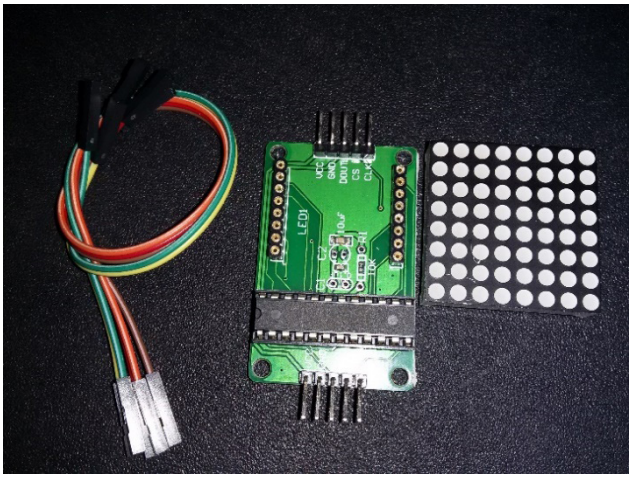
4. Los controladores implementan un SPI interfaz esclava compatible que puede ser controlado desde el Arduino o cual otro microcontrolador utilizando sólo 3 de los pines de salida digitales.

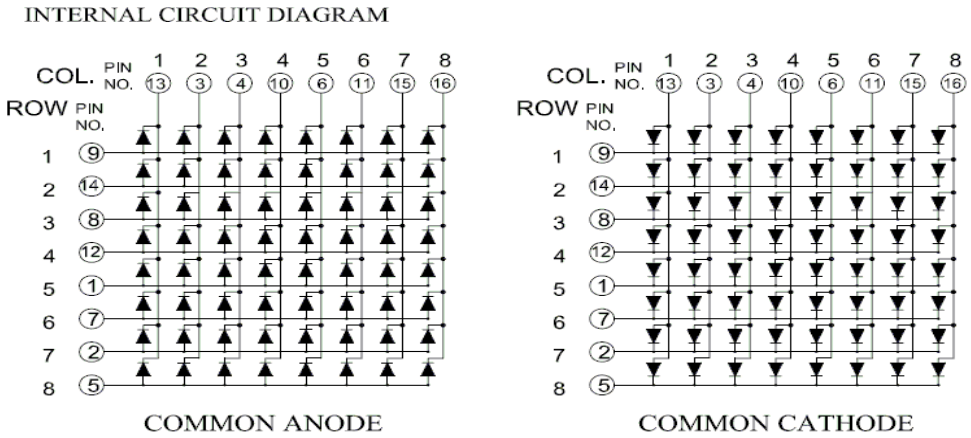
El módulo de bluetooth es importante, por el cual recibe el texto enviado por el Smartphone, para que se visualice en las matrices de LEDs de 8x8.

2.4.2.1. Descripción de las matrices LED

Las matrices LED comercializan en multitud de formatos y colores. Desde las de un solo color, a las que tienen varios colores posibles, e incluso las hay de una matriz RGB.

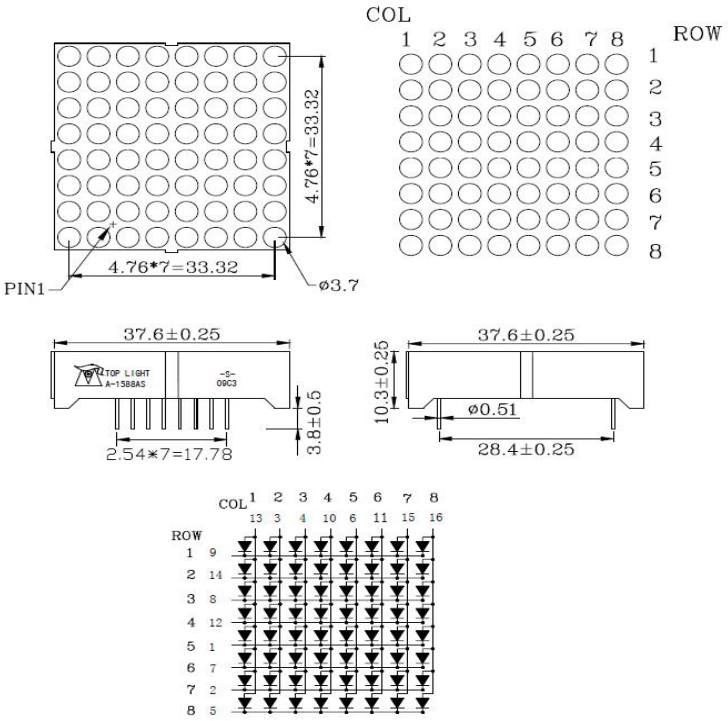






Por lo demás, son diodos LED totalmente normales, organizados en forma de matriz, que tendremos que multiplexar para poder iluminar uno u otro punto. Este componente se presenta con dos filas de 8 pines cada una, que se conectan a las filas y las columnas.

- Si los diodos se unen por el positivo, se dice que son matrices de Ánodo común y se une por el negativo decimos que son de Cátodo común.



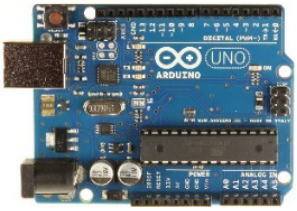
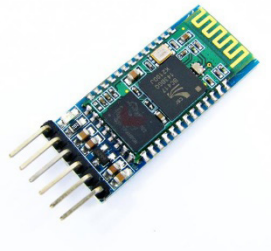
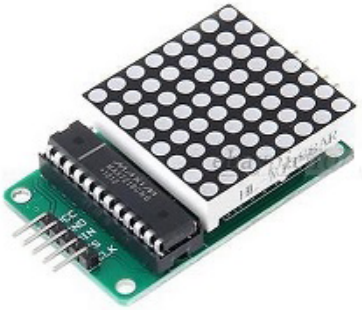
Nota: Teniendo la primera matriz conectada al Arduino, el resto de matrices se conectan en cascada con los pines de salida que se encuentra en la parte superior, con la misma configuración de la primera matriz.

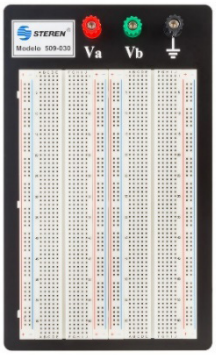
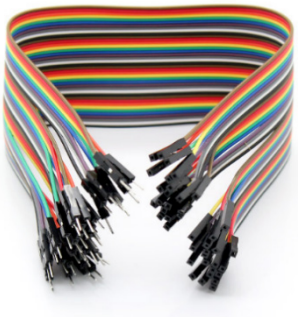
Matriz	13	3	4	10	6	11	15	16
Arduino	2	3	4	5	6	7	8	9

Y las filas:

Matriz	9	14	8	12	1	7	2	5
Arduino	10	11	12	13	A0	A1	A2	A3

2.4.3. Materiales

DESCRIPCIÓN	***REPRESENTACIÓN***
HARDWARE	
(1) Arduino Uno R3	
(1) Modulo Bluetooth HC-05	
(3) Matriz 8x8	

(1) Protoboard	
(10) Cables jumper Macho – Macho	
(15) cables Jumper Hembra – Macho	

2.4.4. Desarrollo

2.4.4.1. Implementación del hardware

Para realizar la práctica planteado es necesario ubicar algunos JUMPER de tal manera que permita el uso de los componentes necesarios para esta práctica.

Paso 1: El módulo Arduino Uno R3 puede utilizar una de dos fuentes de alimentación.

a. Fuente de alimentación USB desde PC a través del cable USB.
b. Fuente de alimentación EXT desde un Jack DC.

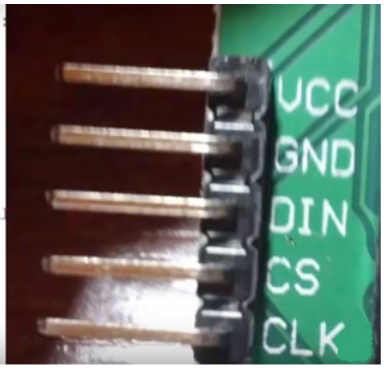
Paso 2: Conectamos el Arduino a la matriz LED 8x8 con la siguiente numeración de pines.

Conexiones del Arduino al Modulo MAX7219:

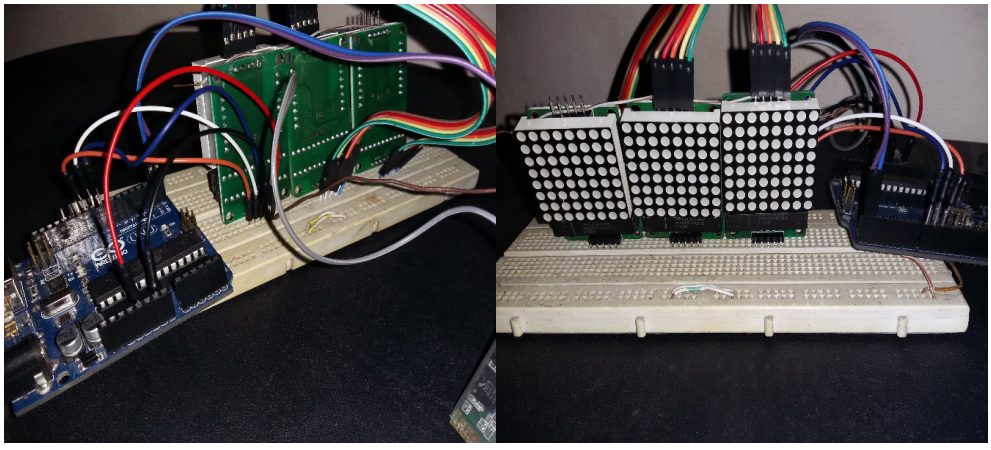
ARDUINIO	MAX7219
10	CLK
9	CS
8	DIN
GND	GND
5V	VCC

Conexion de la cascada de MAX7219(1) al MAX7219(2)

MAX7219(1)	MAX7219(2)
CLK	CLK
CS	CS
DOUT	DIN
GND	GND
VCC	VCC



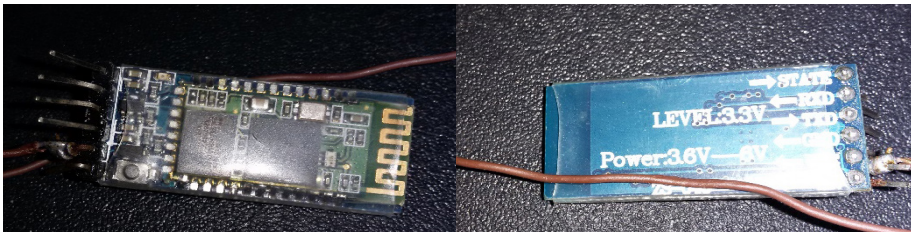
Paso 3: Ya teniendo correctamente conectado el Arduino a la matriz de leds 8x8 nos quedará así:



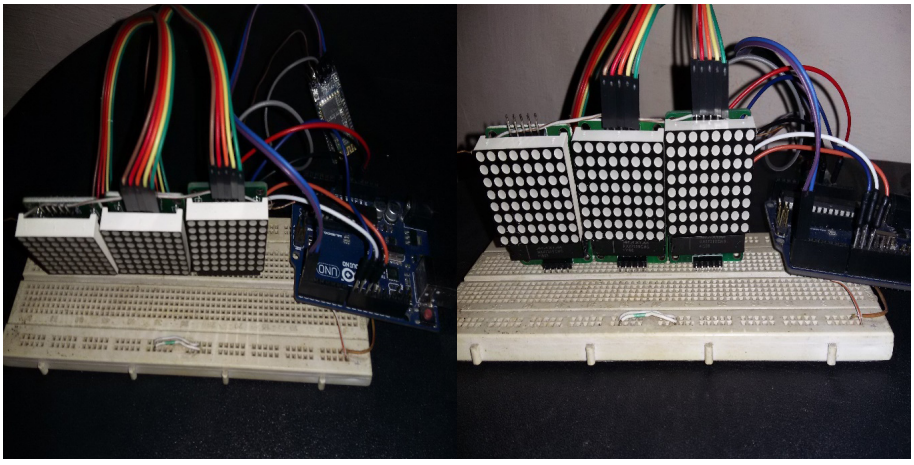
Paso 4: Aquí podemos observar el módulo de Bluetooth HC-05, en los pines nos muestra la siguiente numeración.

Arduino UNO R3	BLUETOOTH HC-05
VCC	VCC
GND	GND
TX->1	RXD
RX->2	TXD

Paso 5: Conectados correctamente podemos visualizar de la siguiente manera.

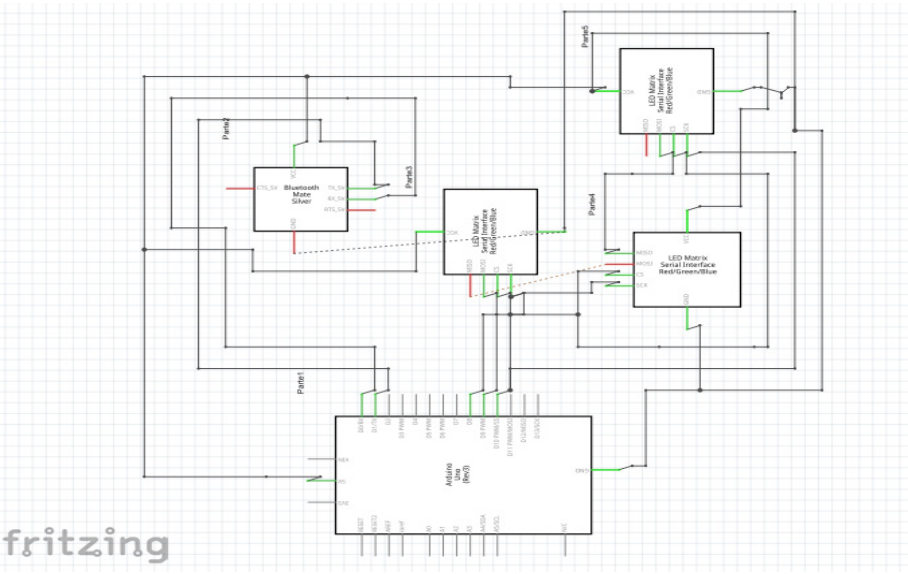


Paso 6: Como podemos observar ya tenemos todos los componentes conectados al Arduino para así visualizar el mensaje en las matrices 8x8.



Paso 7: Para su mayor entendimiento del circuito le mostramos el Diagrama circuital.

Diagrama Circuital



2.4.4.2. Implementación del software

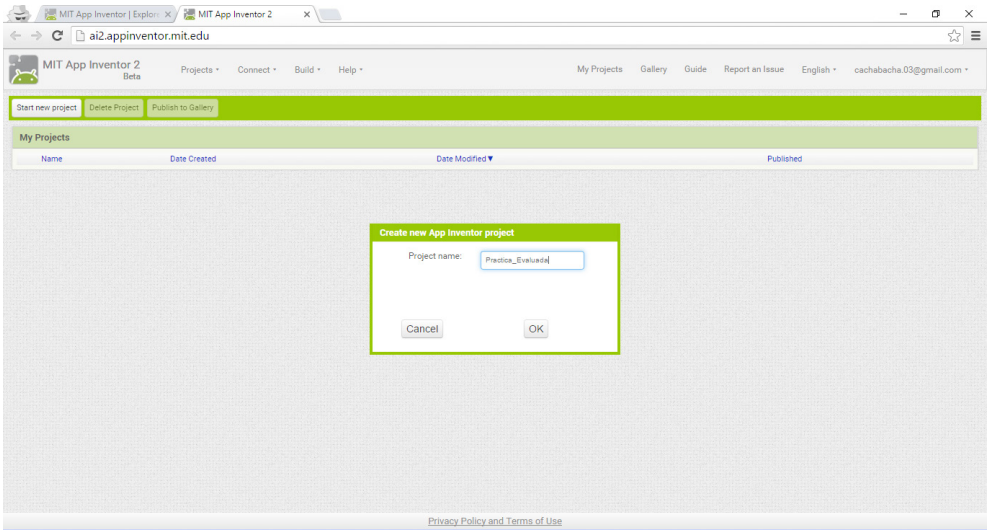
Para utilizar el IDE Arduino se recomienda para esta práctica utilizar la versión 1.0.5 y la librería MaxMatrix.

Ven Anexo 1: Programación en Arduino

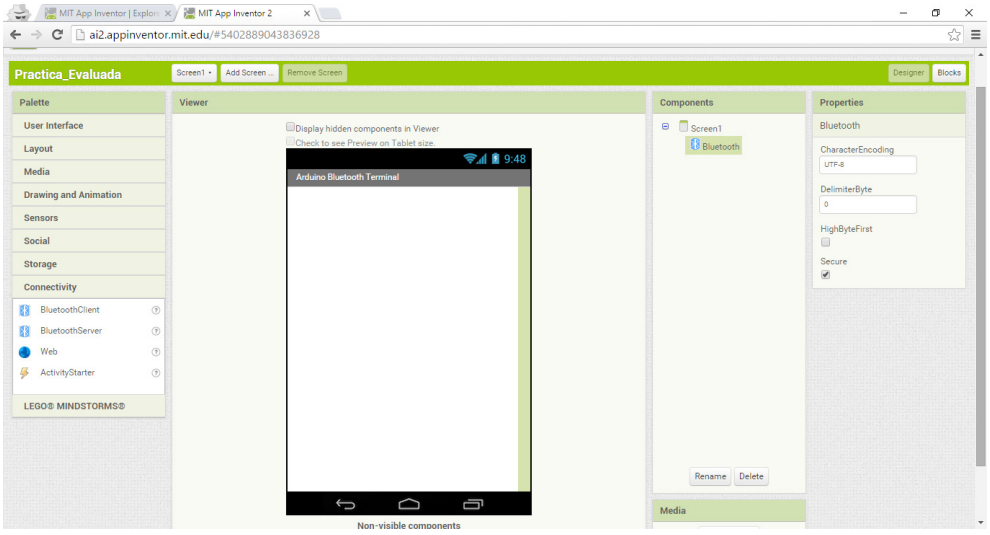
Programación en Android (apk) utilizando la herramienta de programación grafica app inventor 2:

Lo primero que vamos a hacer es ir a la página http://appinventor.mit.edu/ y dar clickk en el botón Create apps ¡, nos pedirá una cuenta en gmail e ingresamos.

Una vez allí hacemos click en Start new Project.



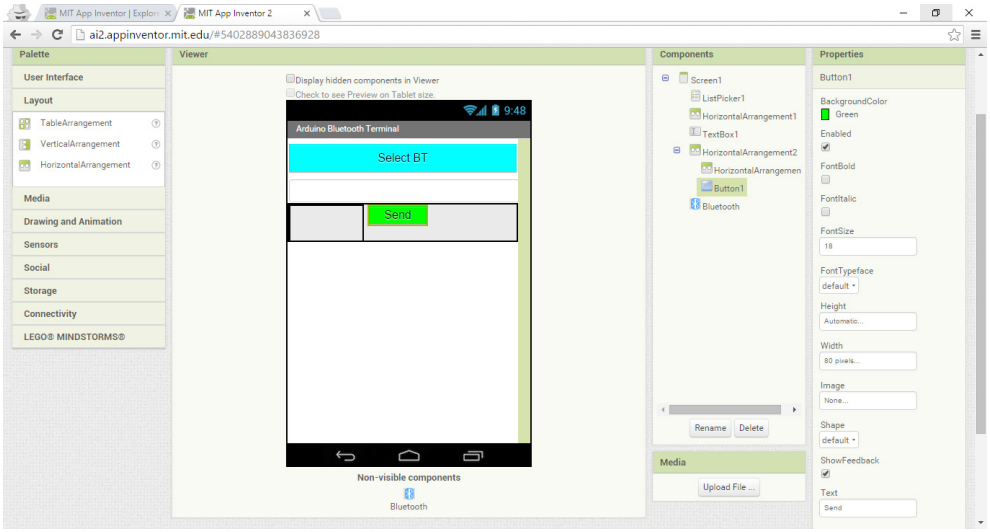
Una vez creado el proyecto se mostrará la pantalla donde desplegaremos nuestro proyecto tenemos dos partes muy importantes el "Designer" y el "Blocks". El Designer es donde nos encargaremos de poner todos los componentes que tendrá nuestro programa y el apartado de "Blocks" es donde se programará, pero no de la forma habitual sino utilizando "bloques" de una manera muy fácil.



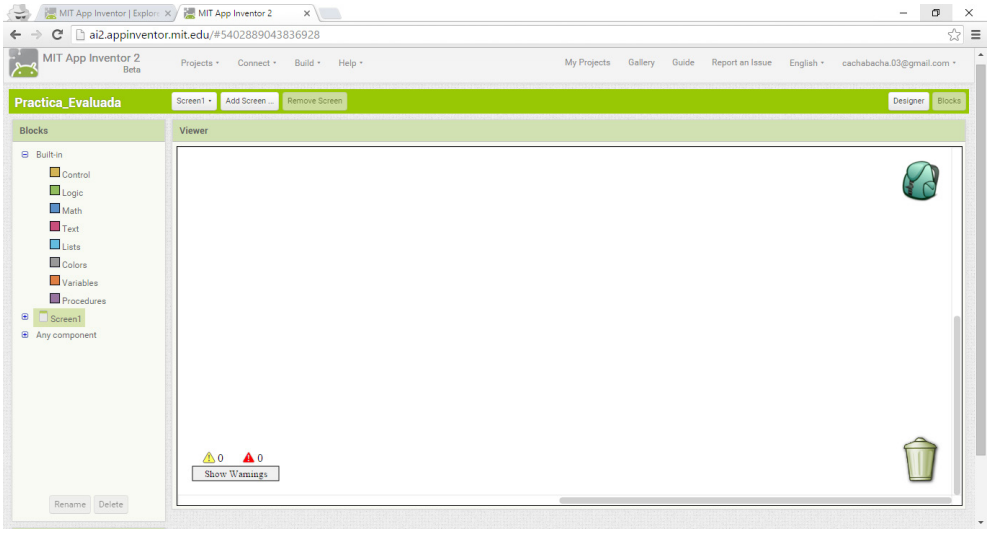
El primer paso será agregar nuestros componentes que son:

- BluetoohtClient (Connectivity-> BluetoohtClient) - con este componente buscaremos los clientes Bluetooth conectados a nuestro celular.
- ListPicker (User Interface-> ListPicker).- permitirá la selección de los clientes Bluetooth disponibles.

- TextBox (User Interface-> TextBox).- donde se escribirá el texto que se mostrará en las pantallas LED 8x8.

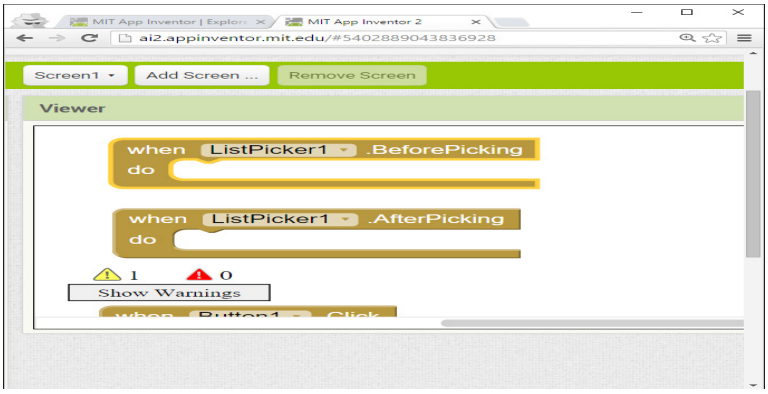


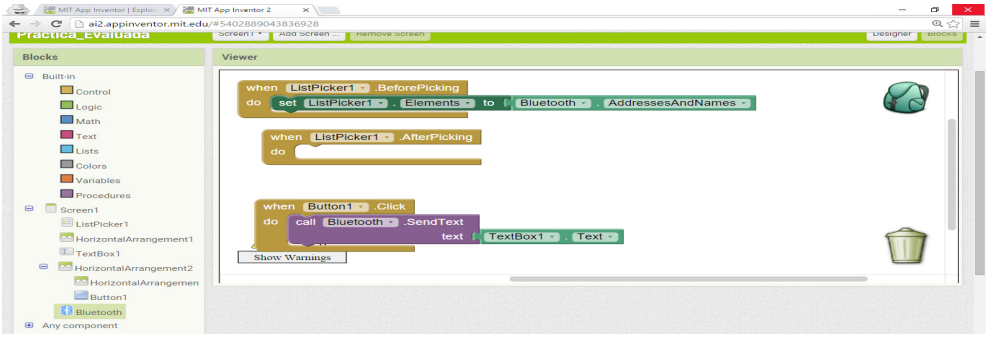
Luego para la programación de estos componentes iremos a “Blocks”. Esta es la pantalla que maneja esta sección.



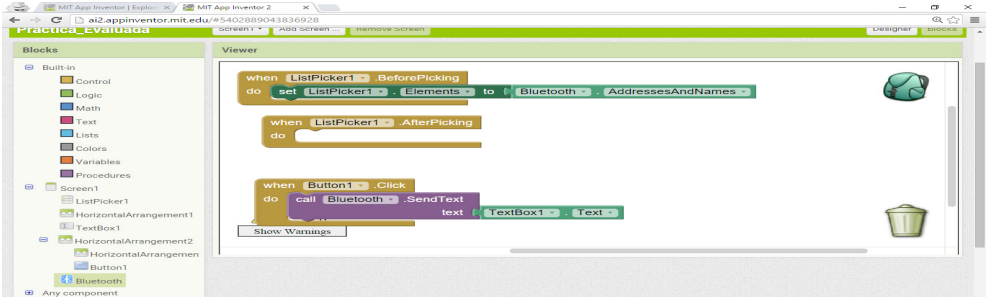
En este paso iniciaremos la programación por bloques, lo primero que se hará es seleccionar el componente ListPicker siguiendo la siguiente lógica:

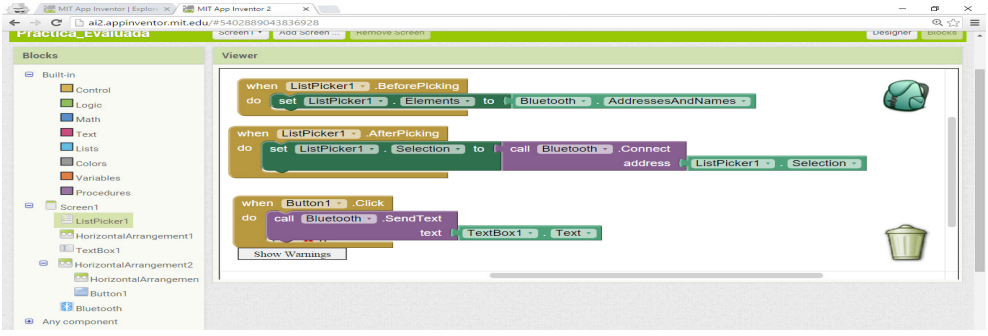
- Antes de que se Oprimido (Before Picking) al ListPicker le estableceremos todos los módulos Bluetooth que vaya encontrando nuestro Smartphone vía BluetoothClient.



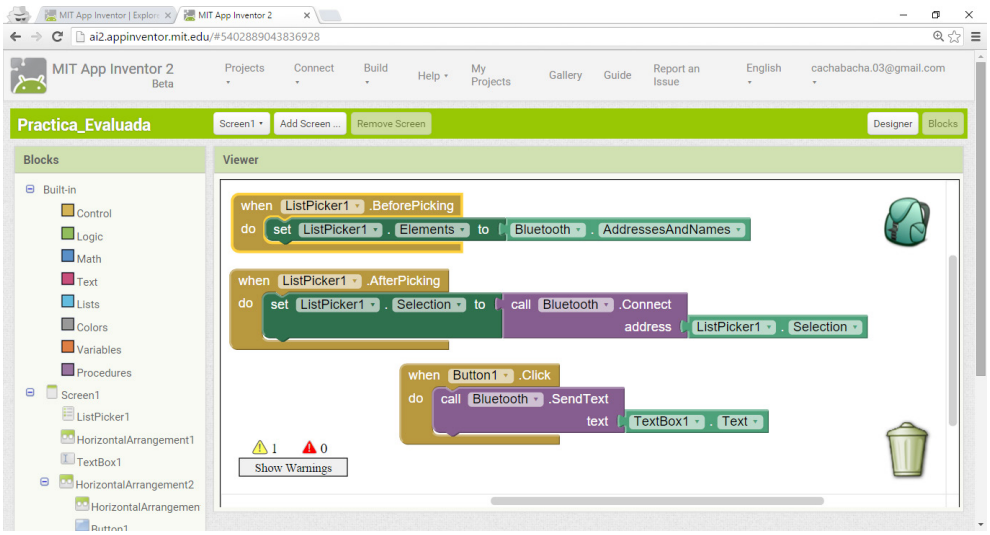


- Despues de oprimirlo (After Picking) que se conecte al módulo Bluetooth seleccionado por medio del ListPicker.

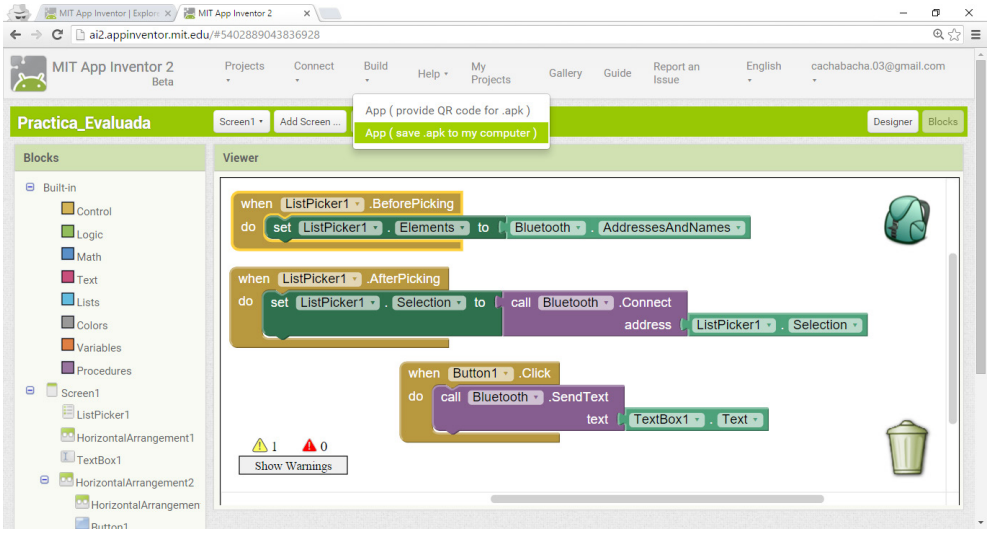




Con ésta programación ya tenemos la conexión a Bluetooth, ahora solo necesitamos que al hacer click en el botón “Send” se envíe la información que contiene el componente TextBox por medio del Bluetooth, de la siguiente manera.



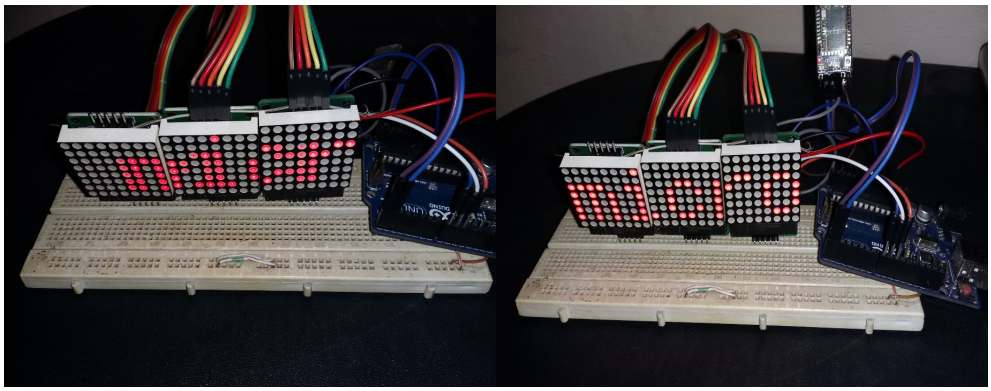
Guardaremos el Proyecto y construiremos (Build) nuestro archivo instalador (.apk), desde la pestaña superior click en Build-> App(save .apk to my computer) y listo ya tendremos nuestra aplicación lista para usarla en nuestro Smartphone.



NOTA: Para un mejor entendimiento del funcionamiento de la práctica se recomienda leer todos los comentarios puestos durante la programación, ahí se explica de manera detallada todo el proceso.

2.4.5. Detalles de configuración y ejecución de la práctica

Teniendo cargado el código podemos observar cómo nos muestra en las imágenes, el mensaje en las matrices de LEDs de 8x8 que a su vez ha enviado el mensaje un Smartphone vía Bluetooth.



2.5. Control de estado de un actuador mediante la red social twitter haciendo uso del protocolo MQTT

2.5.1. Objetivos

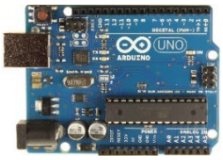
- Aprender el uso del protocolo MQTT.
- Controlar a través del Nodejs un actuador.

2.5.2. Descripción

El presente proyecto hace uso del protocolo MQTT y lenguaje de programación nodejs:

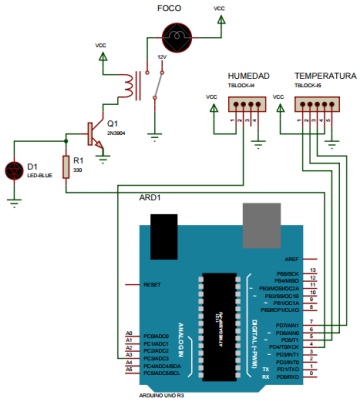
- MQTT es un protocolo de comunicación que trabaja máquina a máquina (M2M) / protocolo de comunicación. Está diseñado como un protocolo extremadamente ligero para mensajería. Es útil para las conexiones donde se requiere limitar el ancho de banda de red.
- Nodejs es un intérprete Javascript del lado del servidor que cambia la noción de cómo debería trabajar un servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sólo una máquina física.

2.5.3. Materiales

DESCRIPCIÓN	*REPRESENTACIÓN*
HARDWARE	
Arduino UNO	
Cable de red directo	
Cable USB Arduino	
SOFTWARE	
Virtual Box	
Máquina virtualizada con servidor bróker.	
Node JS 5.5 (64 bits)	
Arduino IDE 1.6	

2.5.4. Desarrollo

2.5.4.1. Diagrama esquemático



2.5.4.2. Descripción de funcionamiento de circuito

Por medio de la placa Arduino se implementará un sistema que permita el encendido y apagado de un motor haciendo uso de la red social Twitter.

La presente práctica implementa el uso del protocolo de comunicación mqtt en una WSL (Wireless Network Sensor) haciendo uso de la tecnología Arduino UNO.

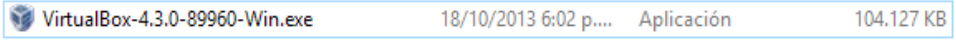
- Al tener el WSN funcionando se evidencia el funcionamiento del protocolo mqtt bajo la modalidad suscriptor – publicador.

El motor o el foco conectado a través del pin 4 de la placa Arduino, recibe las señales que se publican en el twitter y que pertenezcan al código, si pertenece, éste se encenderá al publicar los estados.

2.5.5. Implementación del software

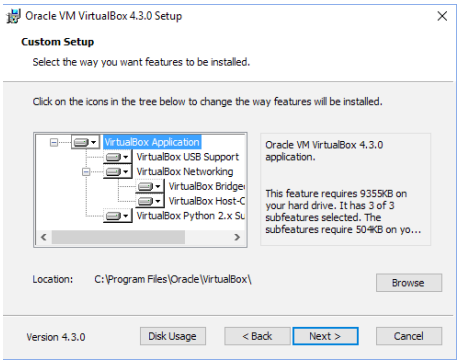
2.5.5.1. Instalación de la máquina Virtual Box 4.3.0.

Paso 1: Doble click en el instalador que se encuentra en la carpeta de recursos.

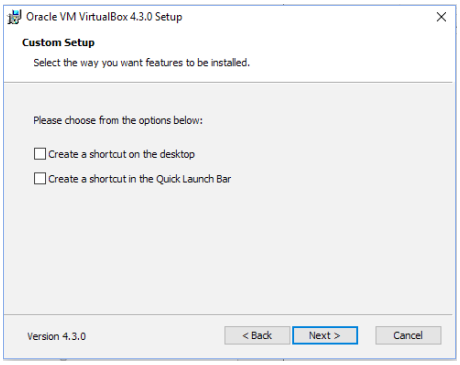


Paso 2: Click en siguiente (Next).

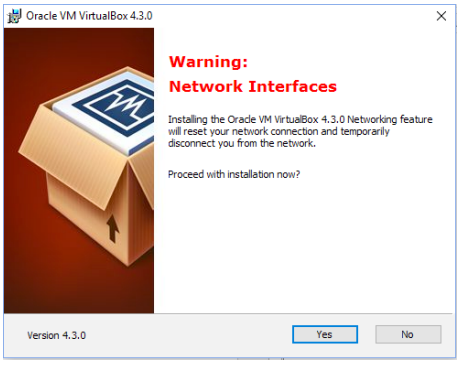
Paso 3: Click en siguiente (Next).



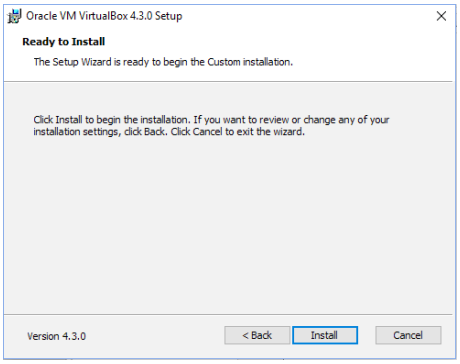
Paso 4: Click en siguiente (Next).



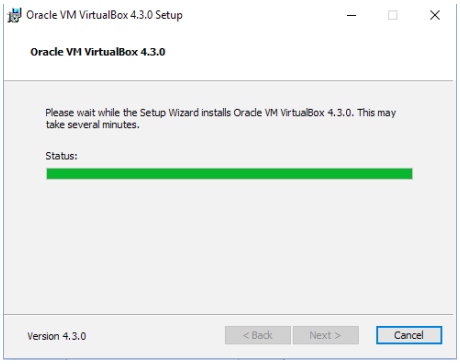
Paso 5: Click en Si (Yes).



Paso 6: Click en Install (Instalar).



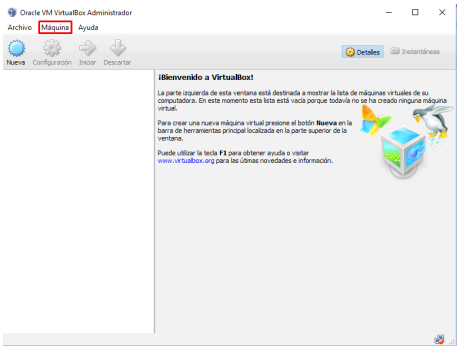
Paso 7: Esperamos que la barra de estado complete su progreso. Click en Next.



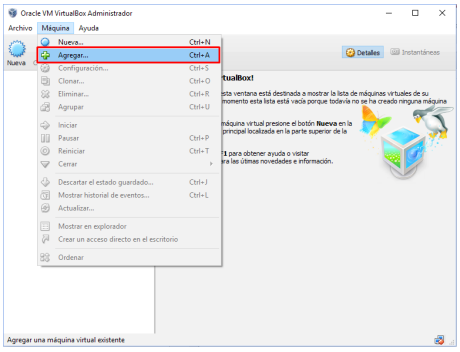
Paso 8: Marcar opción de pantalla para que inicie la máquina virtual y Click en Finalizar (Finish).

2.5.5.2. Importar y configurar la máquina virtual del sistema operativo Débian donde se encuentra pre-configurado el servidor bróker

Paso 1: Una vez abierta la interfaz de Virtual Box, en la barra superior ubicar la opción Máquina y dar click.

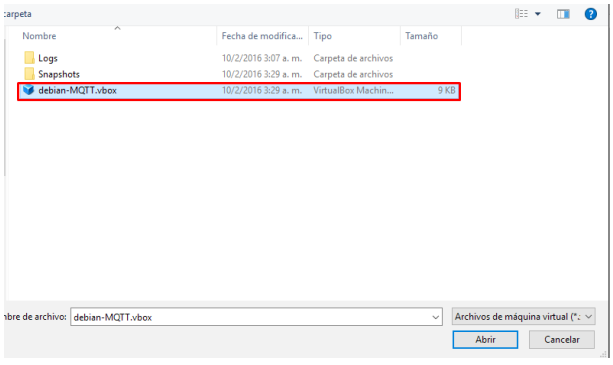


Paso 2: Entre las opciones desplegadas, ubicar la opción Agregar y dar click.

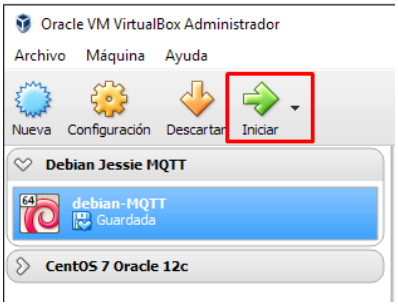


Paso 3: En la carpeta de recursos ubicar la siguiente carpeta. Dar doble click.

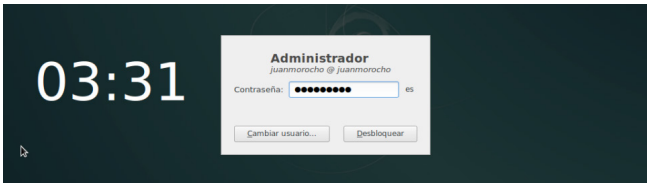
Paso 4: Elegir la máquina virtualizada con el sistema operativo Debian, que contiene el servidor broker. Click en Abrir.



Paso 5: Seleccionar la máquina virtualizada y click en la opción superior Iniciar.

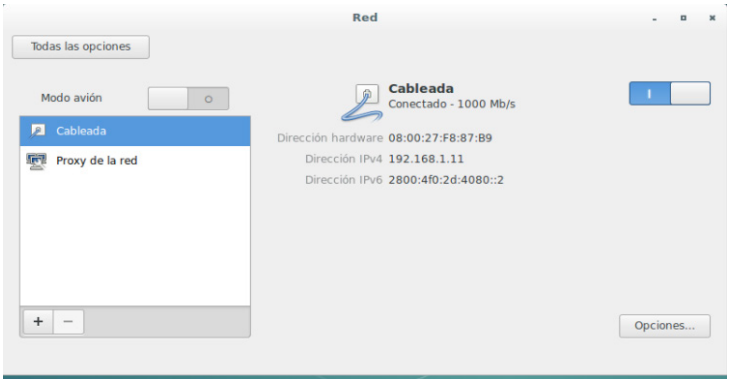


Paso 6: Escribir contraseña *juan12345*. Click en desbloquear.

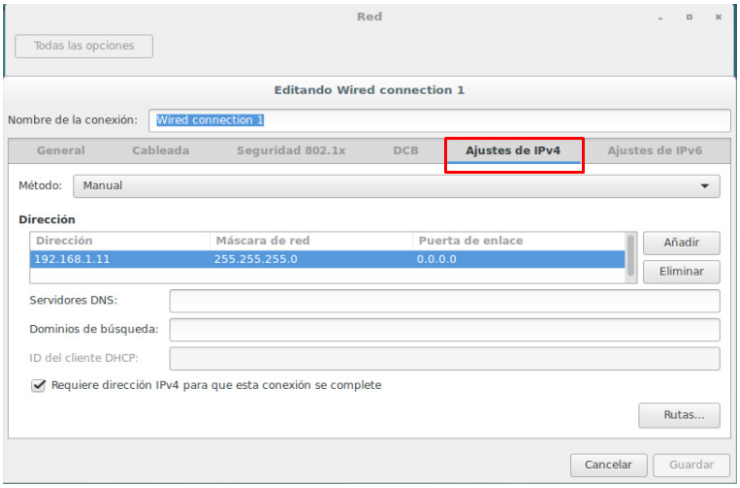


Paso 7: Configurar dirección IP de la máquina virtual. Click en preferencia de red.

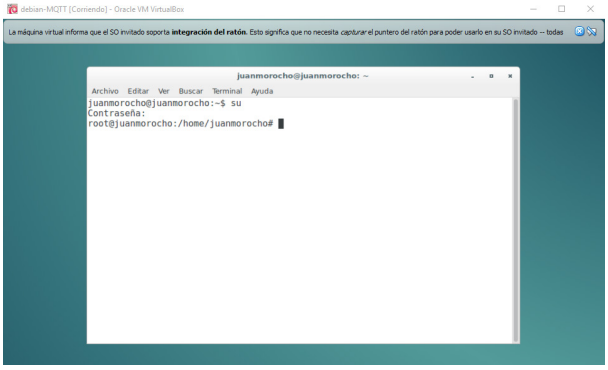
Paso 8: Click en opciones.



Paso 9: Ubicarse en la pestaña de Ajustes de IPv4 e ingresar dirección IP a utilizar. Finalmente, click en Guardar.



Paso 10: Abrir la terminal, ingresar como usuario root y escribir la contraseña: 12345678. Presionar Enter.

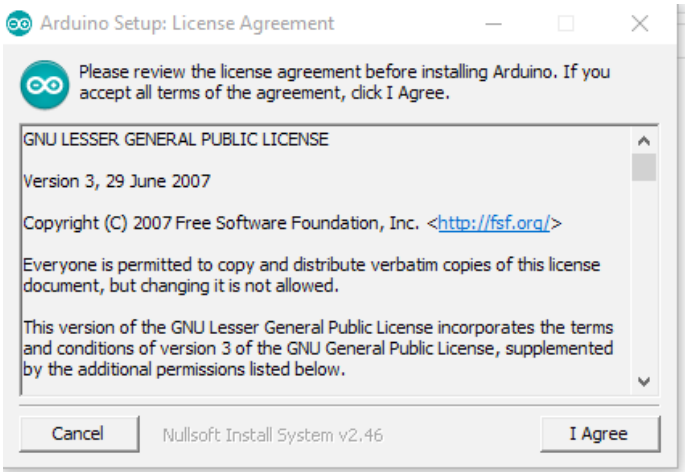


Paso 11: Escribir la siguiente línea de comandos en la terminal. Presionar Enter.

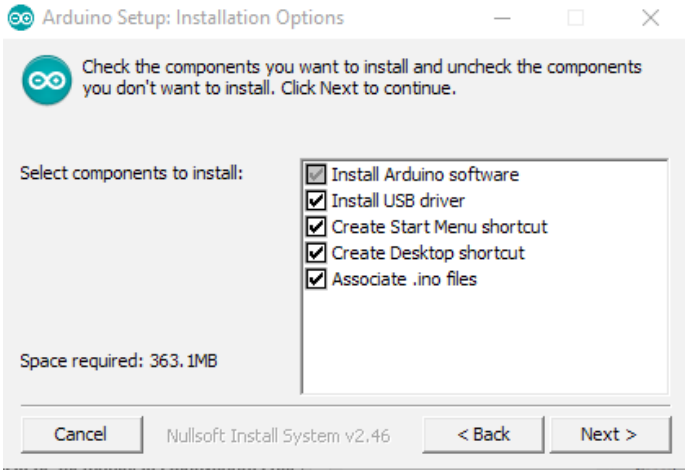
2.5.5.3. Instalación de Arduino IDE

Paso 1: Ubicar en la carpeta de recursos el instalador del IDE Arduino y dar doble click.

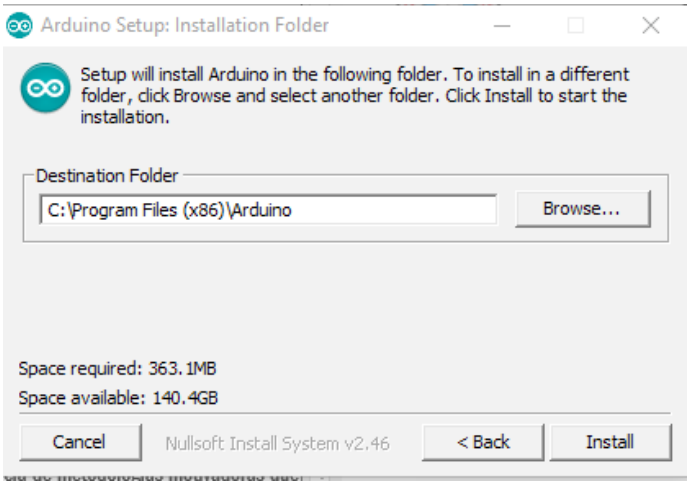
Paso 2: Aceptar las condiciones. Click en Acepto (I Agree).



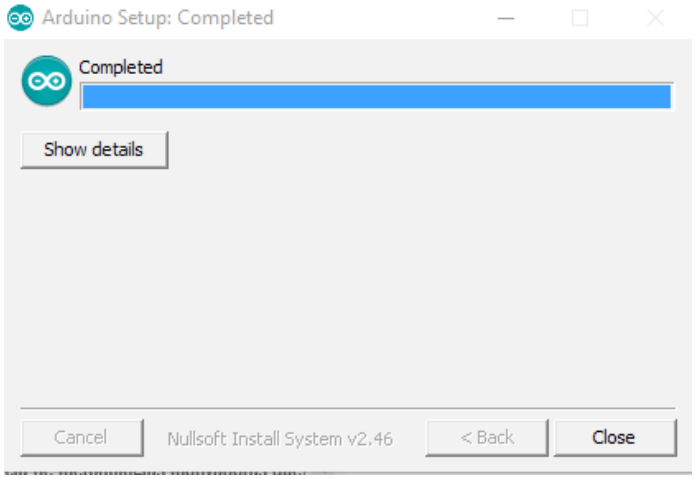
Paso 3: Click en Siguiente (Next).



Paso 4: Click en Instalar (Install).



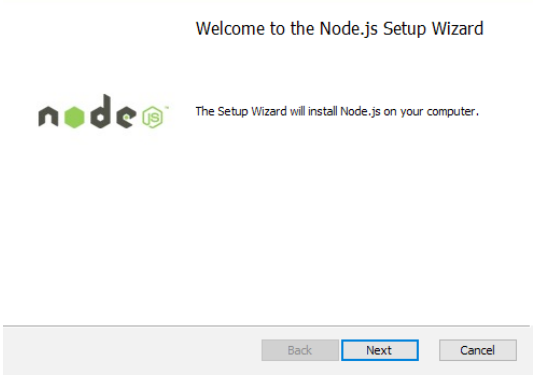
Paso 5: Esperar que la barra de estado de instalación finalice y click en Cerrar (Close).



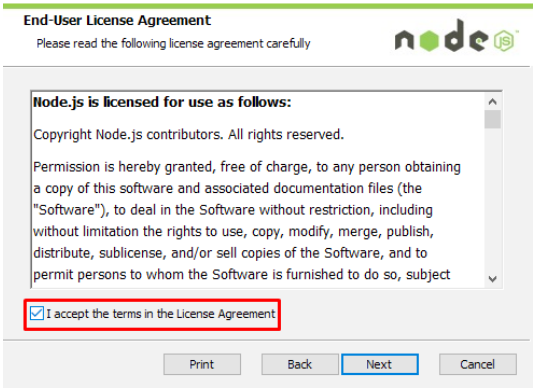
2.5.5.4. Instalación de Node JS

Paso 1: Ubicar en la carpeta de recursos el instalador de Node JS y dar doble click.

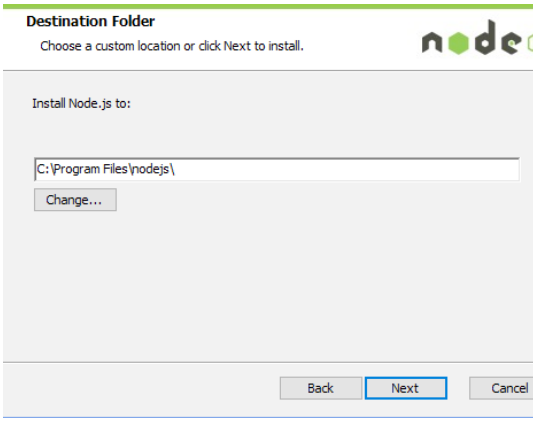
Paso 2: Click en Siguiente (Next).



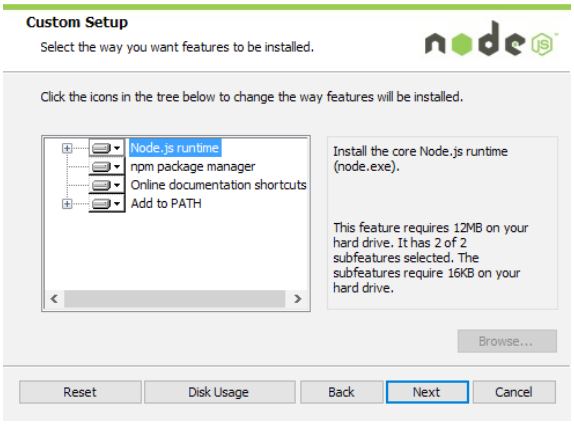
Paso 3: Seleccionar la casílla para aceptar las condiciones de instalación y click en botón Siguiente (Next).



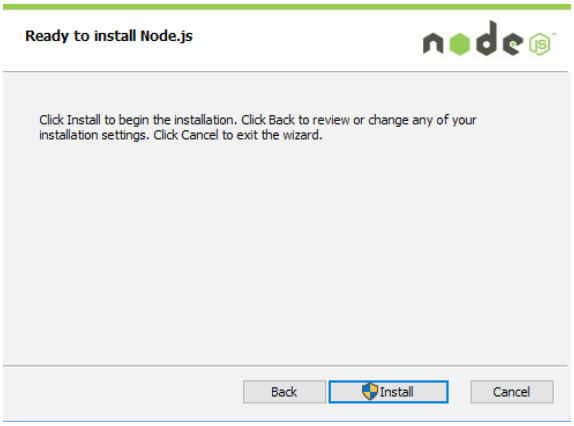
Paso 4: Click en el botón siguiente (Next).



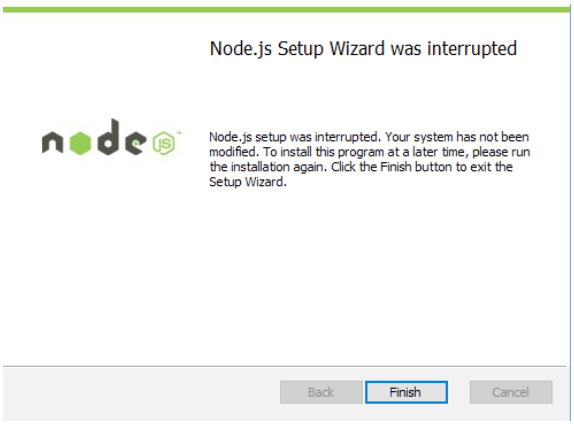
Paso 5: Click en botón Siguiente (Next).



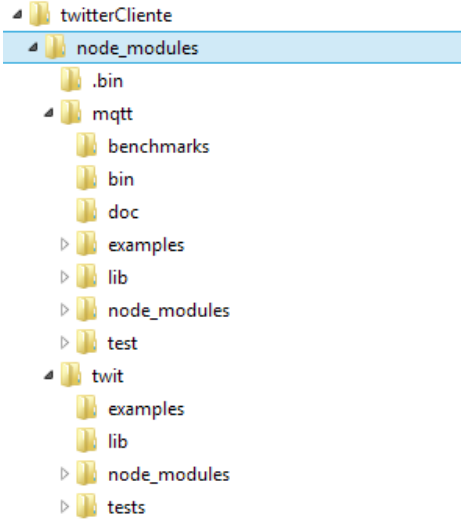
Paso 6: Click en Instalar (Install).



Paso 7: Click en Finalizar (Finish).



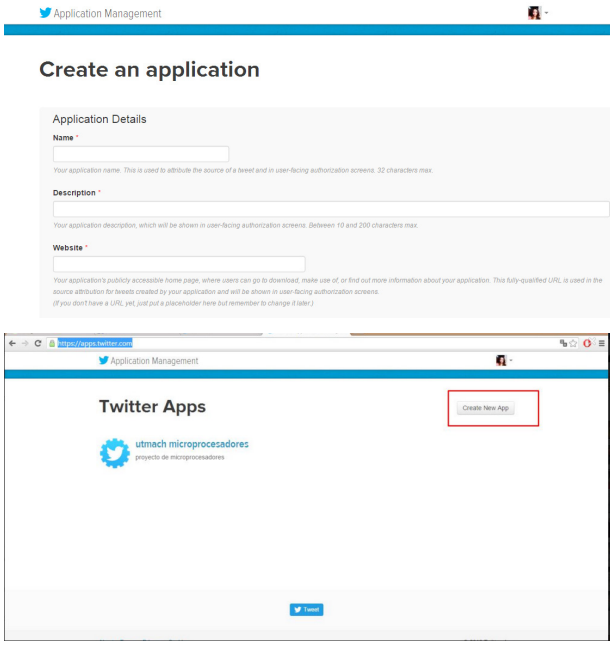
Paso 8. Abrimos el cmd y ejecutamos en la consola npm install y se nos creará la carpeta node_modules:



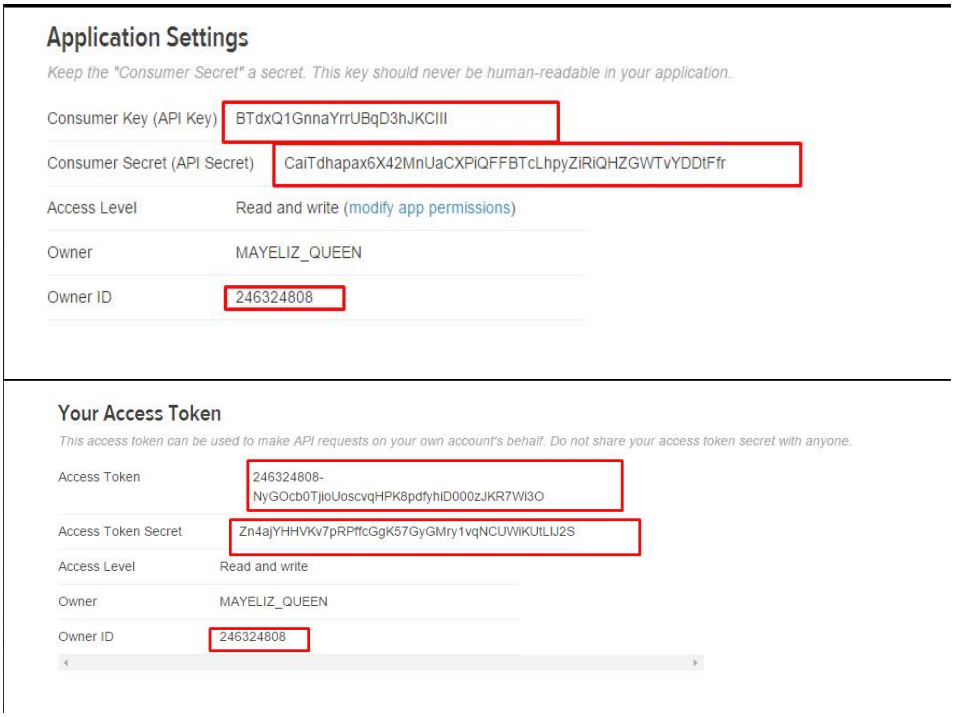
Nota: Ver en Anexo 1: Escribimos el código nodejs.

2.5.6. Detalles de configuración y ejecución de la práctica

Paso 1: Creamos una aplicación en Twitter.



Paso 2: Escribimos los permisos en la aplicación nodejs.



Paso 3: Ejecutamos.

REFERENCIAS BIBLIOGRÁFICAS

McEwen, A. y Cassimally, H. (2014). *Internet de las cosas: la tecnología revolucionaria que todo lo conecta.* Murcia, España: Anaya.

Aranda, D. (2014). *ELECTRÓNICA 3: Plataformas Arduino y Raspberry Pi.* Buenos Aires, Argentina: Manual USERS.

ICT Audiovisual. (2013). *Internet de las cosas: Objetos interconectados y dispositivos inteligentes.* Madrid, España: Madrid Network. Recuperado de: http://www.madrid-network.org/red/audiovisual

Goilav, N. y Loi, G. (2016). *Arduino: Aprender a desarrollar para crear objetos inteligentes.* Barcelona, España: Ediciones ENI.

Pisani, F. (2016). *Internet Industrial: Máquinas inteligentes en un mundo de sensores.* Barcelona, España: Editorial Planeta.

ANEXOS

Capítulo II

Practica 1

Anexo 1. Código de programación de la tarjeta Arduino

```
// IMPORTACION DE LIBRERIAS
#include <SPI.h>
#include <Ethernet.h>
#include “max6675.h”
#include <PubSubClient.h>
#include <EEPROM.h>

// CONFIGURACION MQTT
#define MQTT_SERVER_NAME “192.168.1.11”
#define MQTT_SERVER_PORT 1883
#define MQTT_CLIENTID   “90-A2-DA-0F-6E-64”
#define MQTT_TOPIC_PUBLISH_LECTURAS   “/sensor/”

// PARAMETROS TCP
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 12);
String direccionMac = “90-A2-DA-0F-6E-64”;

// CREACION DE OBJETOS PARA LA CONEXION

PubSubClient client;
EthernetClient eclient;

// CONFIGURACION SENSOR TEMPERATURA
int thermoDO = 5;
int thermoCS = 6;
int thermoCLK = 7;
MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);
```

```
// PIN PARA LECTURA HUMEDAD
int pinH = A3;

char message_buff[100];

void setup() {

  pinMode(A3, INPUT);
  Serial.begin(9600);
  while (!Serial) {
    ;
  }

  // CONEXION A LA RED
  Serial.println("Obteniendo IP");
  /*if (Ethernet.begin(mac) == 0) {
    Serial.println("Fallo al Obtener IP por DHCP");
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());
  } else {
    Serial.print("IP Dinamica: ");
    Serial.println(Ethernet.localIP());
  }*/
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());
  // CONEXION AL SERVIDOR MQTT
     client  =  PubSubClient(MQTT_SERVER_NAME,  1883,
callback, eclient);
}
void loop() {
  // VERIFICACION DE CONEXION AL SERVIDOR MQTT
```

```
  if (!client.connected()) {
    Serial.println("Conectando servidor MQTT");
    if (client.connect(MQTT_CLIENTID)) {
      publicar_lecturas();
    } else {
      Serial.println("No se pudo conectar al servidor
MQTT");
    }
  } else {
    publicar_lecturas();
  }
  client.loop();

  delay(500);
}
void publicar_lecturas() {
  int sensorValue = thermocouple.readCelsius();
  int sensorValue1 = map(analogRead(pinH), 0, 1024, 0,
100);
  sensorValue1 = 100 - sensorValue1;
  String json = buildJson(sensorValue, sensorValue1);
  char jsonStr[200];
  json.toCharArray(jsonStr, 200);
 client.publish(MQTT_TOPIC_PUBLISH_LECTURAS, jsonStr);
 Serial.println("Publicando al topico : " + String(MQTT_
TOPIC_PUBLISH_LECTURAS));
  Serial.println(jsonStr);
}
String convierteIp(IPAddress address)
{
  return String(address[0]) + "." +
         String(address[1]) + "." +
         String(address[2]) + "." +
         String(address[3]);
```

```
}
void callback(char* topic, byte* payload, unsigned int
length) {
}
String buildJson(int sensorValue, int sensorValue1) {
  String data = "{";
  data += "\"device\": {";
  data += " \"id\": \"";
  data += MQTT_CLIENTID;
  data += "\",";
  data += "\"temperatura\": ";
  data += "\"";
  data += (int)sensorValue;
  data += "\"";
  data += ",";
  data += "\"humedad\": ";
  data += "\"";
  data += (int)sensorValue1;
  data += "\"";
  data += ",";
  data += "\"bomba\": ";
  data += "\"";
  data += (int)1;
  data += "\"";
  data += ",";
  data += "\"ult\": ";
  data += "\"";
  data += (int)1;
  data += "\"";
  data += "}";
  data += "}";
  return data;
}
```

Anexo 2. Código de programación del cliente Web.

```
<!DOCTYPE html>
<html lang="es" ng-app="app">
<head>
    <title>UTMACH - Microprocesadores</title>
    <meta charset="utf-8"/>
        <meta name="viewport" content="initial-
scale=1.0,maximum-scale=1.0,user-scalable=no">
      <link rel="shortcut icon" href="/img/logos/
logo_utmach.ico">
    <!-- bootstrap framework -->
      <link href="/assets/bootstrap/css/bootstrap.
min.css" rel="stylesheet" media="screen">
    <!-- custom icons -->
    <!-- font awesome icons -->
   <link href="/assets/icons/font-awesome/css/font-
awesome.min.css" rel="stylesheet" media="screen">
    <!-- ionicons -->
   <link href="/assets/icons/ionicons/css/ionicons.
min.css" rel="stylesheet" media="screen">
    <!-- flags -->
   <link rel="stylesheet" href="/assets/icons/flags/
flags.css">
    <!-- page specific stylesheets -->
    <!-- nvd3 charts -->
    <link rel="stylesheet" href="/assets/lib/novus-
nvd3/nv.d3.min.css">
    <!-- owl carousel -->
     <link rel="stylesheet" href="/assets/lib/owl-
carousel/owl.carousel.css">
    <!-- main stylesheet -->
 <link href="/assets/css/style.css" rel="stylesheet"
media="screen">
    <!-- google webfonts -->
   <link href='http://fonts.googleapis.com/css?fami
```

```
ly=Source+Sans+Pro:400&amp;subset=latin-ext,latin'
rel='stylesheet' type='text/css'>
    <!-- moment.js (date library) -->
     <script src="assets/lib/moment-js/moment.min.
js"></script>
    <!--INICIO de checkbox switch-->
        <link  href="bootstrap-switch-master/dist/
css/bootstrap3/bootstrap-switch.min.css"
rel="stylesheet">
  <script type="text/javascript" src="/js/mqttws31.
js"></script>
   <script type="text/javascript" src="/js/jquery-
2.1.3.min.js"></script>
    <script type="text/javascript">
      var client;
      var brokerIp = "192.168.1.11";
      var interactive_plot1;
      var interactive_plot2;
      var options = {
        grid: {
          borderColor: "green",
          borderWidth: 1,
          tickColor: "green"
        },
        series: {
         shadowSize: 0, // Drawing is faster without
shadows
          color: "orange"
        },
        yaxis: {
          min: 0,
          max: 100
        },
        lines: {
          show: true
```

```
        },
        xaxis: {
          show: true
        }
      }
       var data1 = [], totalPoints = 60;//x=tiempo
60 segundos
      function getRandomData1(valor_dato) {
        if (data1.length > 0)
          data1 = data1.slice(1);
        while (data1.length < totalPoints) {
         var prev = data1.length > 0 ? data1[data1.
length - 1] : 50,
          y = prev + Math.random() * 10 - 5;

          data1.push(valor_dato);//pone el dato
        }

         // Zip the generated y values with the x
values
        var res = [];
        for (var i = 0; i < data1.length; ++i) {
          res.push([i, data1[i]]);
        }
        return res;
      }
       var data2 = [], totalPoints = 60;//x=tiempo
60 segundos
      function getRandomData2(valor_dato) {
        if (data2.length > 0)
          data2 = data2.slice(1);
        while (data2.length < totalPoints) {
         var prev = data2.length > 0 ? data2[data2.
length - 1] : 50,
```

```
            y = prev + Math.random() * 10 - 5;

            data2.push(valor_dato);//pone el dato
          }

           // Zip the generated y values with the x
values
          var res = [];
          for (var i = 0; i < data2.length; ++i) {
            res.push([i, data2[i]]);
          }
          return res;
        }

        doConnect();

        //funcion para suscribirse al broker
        function doSubscribe() {
          var topic = "/sensor/";
          client.subscribe(topic);
          $("#id_conexion").removeClass('stat_down');
          $("#id_conexion").addClass('stat_up');
          console.log("se suscribio");
        }
        //funcion que se conecta al broker
        function onConnect() {
          console.log("entro a conexion exitosa");
          doSubscribe();
        }
        //funcion cuando se pierde la conexion
        function onConnectionLost(responseObject) {
          if (responseObject.errorCode !== 0){
            $("#id_conexion").removeClass('stat_up');
```

```
          $("#id_conexion").addClass('stat_down');
        }
      }
         //funcion la cual permite inicializar la
conexion

      //funcion que recibe los mensajes del broker
      function onMessageArrived(message) {
              console.log("llego dato: "+message.
payloadString);
       var json = JSON.parse(message.payloadString);
        var identificador = json["device"].id;
        var dato1 = json["device"].temperatura;//el
id del valor
        var dato2 = json["device"].humedad;
        var dato3 = json["device"].areador;
        var dato4 = json["device"].foco;
                                  interactive_plot1.
setData([getRandomData1(dato1)]);
        interactive_plot1.draw();
                                  interactive_plot2.
setData([getRandomData2(dato2)]);
        interactive_plot2.draw();
         $('#id_medidor_tem').data('easyPieChart').
update(dato1);
        $('#id_med_tem_da').html(dato1);
         $('#id_medidor_hum').data('easyPieChart').
update(dato2);
        $('#id_med_hum_da').html(dato2);

          // if(dato3==0) $('#id_estado_areador').
html("Off");
          // if(dato3==1) $('#id_estado_areador').
html("On");
```

```
            //  if(dato4==0)  $('#id_estado_foco').
html("Off");
            //  if(dato4==1)  $('#id_estado_foco').
html("On");


      }
      function doConnect() {
        console.log('intentando conectarse');
       client = new Messaging.Client(brokerIp, 9000,
"cliente_web");
        client.onConnect = onConnect;
        client.onMessageArrived = onMessageArrived;
        client.onConnectionLost = onConnectionLost;
        client.connect(
          {onSuccess:onConnect}
       );//si se realizo la conexion con exito salta
esta funcion
      }

      //funcion para desuscribirse del broker
      function doUnsubscribe() {
          var topic = "/sensor/";
          client.unsubscribe(topic);
      }
      //funcion que permite publicar en el broker
      // function doSend(act) {
      //   var publicar = "/actuador/"
      //   var valor_publicar="";
      //   if(act=='areador')
      //   {
      //      if($("#id_areador").prop('checked'))
{
        //      valor_publicar="ON1"
        //    }else{
```

```
        //      valor_publicar="OF1"
        //    }
      //   }
      //   else if(act=='foco')
      //   {
      //      if($("#id_foco").prop('checked')){
        //      valor_publicar="ON2"
        //    }else{
        //      valor_publicar="OF2"
        //    }
      //   }

       //    message = new Messaging.Message(valor_
publicar);
      //   message.destinationName = publicar;
      //   client.send(message);
      // }
      //funcion que desconecta del broker
      function doDisconnect() {
          client.disconnect();
      }
       // funcion que da a conocer si el navegagor
funciona con websockets
     function WebSocketTest()
        {
          if ("WebSocket" in window)
          {
           alert("WebSockets supported here!\r\n\r\
nBrowser: " + navigator.userAgent );
          }
          else
          {
          // the browser doesn't support WebSockets
            alert("WebSockets NOT supported here!\
```

```
r\n\r\nBrowser: " + navigator.userAgent );
          }
        }

      </script>
</head>

<body style="background-color: #24282b;background-
image: url(/img/fondo.png);"  >

       <header   class="navbar   navbar-fixed-top"
role="banner">
    <div class="container-fluid">
      <div class="navbar-header">
        <a href="#" class="navbar-brand">
        <img src="assets/img/blank.gif" alt="Logo">
        </a>
      </div>
      <ul class="nav navbar-nav navbar-right">
        <li class="lang_menu">
         <a href="#" style="text-decoration:none" >
                <span  class="ion-ios7-information-
outline"></span> <span>Informacion</span>
          </a>
        </li>
      </ul>
    </div>
  </header>
    <!-- side navigation -->
    <nav id="side_nav">
      <ul>
        <li style="background-color:orange;">
          <a href="/" style="color:black">
          <span class="ion-ios7-infinite-outline"></
```

```
span>
            <span class="nav_title">Arduino</span>
          </a>
        </li>
      </ul>
    </nav>
    <!-- main content -->
    <div id="main_wrapper">
      <div class="page_content">
        <div class="container-fluid">
          <div class="row">
            <div class="col-xs-12">
              <div class="row">
                <div class="col-xs-12">
                   <div class="panel panel-default"
style="background-color:transparent;color:white">
                  <div id="id_conexion" class="stat_
box stat_down">
                     <div class="stat_ico visible-
lg"><img  src="/img/logos/escuela_informatica.png"
height= "50"></div>
                        <div class="stat_content"
style="margin-top:10px">
                          <span class="stat_count
visible-lg"   style="color:orange;">Monitoreo   y
Control</span>
                          <span class="stat_name"
style="text-align:right" >Estado Conexion</span>
                     </div>
                   </div>
                 </div>
               </div>
             </div>
           </div>
         </div>
```

```
          <div class="row">
             <div class="col-xs-12 col-lg-6 col-md-6
col-sm-12">
               <div class="panel" style="background-
color:fff;">
                 <div class="panel-header">
                <h3 class="panel-title"><i class="fa
fa-bar-chart-o"></i> Sensor de Temperatura</h3>
                 </div>
                 <div class="panel-body">
                           <div  id="id_grafico_temp"
style="height: 200px;"></div>
                 </div>
               </div>
             </div>

             <div class="col-xs-12 col-lg-6 col-md-6
col-sm-12">
               <div class="panel">
                 <div class="panel-header">
                <h3 class="panel-title"><i class="fa
fa-bar-chart-o"></i> Sensor de Humedad</h3>
                 </div>
                 <div class="panel-body">
              <div id="id_grafico_hum" style="height:
200px;"></div>
                 </div>
               </div>
             </div>
           </div>

           <div class="row">
              <div class="col-xs-6 col-lg-6 col-md-6
col-sm-6">
               <div class="panel" style="background-
```

```
color:transparent;color:white”>
                <div class=”panel-heading”>Sensor de
Temperatura</div>
                 <div class=”panel-body”>
          <div id=”id_medidor_tem” class=”easy_chart
easy_chart_opens pull-left” data-percent=”0”><span
id=”id_med_tem_da” style=”color:orange”>0</span></
div>
                <div class=”easy_chart_desc visible-
lg”>
                     <h4>Temperatura</h4>
                   <p>La temperatura ideal debe ser
minimo 20°C y maximo 30°C</p>
                  </div>
                 </div>
               </div>
             </div>
              <div class=”col-xs-6 col-lg-6 col-md-6
col-sm-6”>
               <div class=”panel” style=”background-
color:transparent;color:white”>
                  <div class=”panel-heading”>Sensor
de Humedad</div>
                <div class=”panel-body”>
          <div id=”id_medidor_hum” class=”easy_chart
easy_chart_opens pull-left” data-percent=”0”><span
id=”id_med_hum_da” style=”color:orange”>0</span></
div>
                <div class=”easy_chart_desc visible-
lg”>
                     <h4>Humedad</h4>
                       <p>La humedad ideal debe ser
minimo 40% maximo 60% </p>
                  </div>
                 </div>
               </div>
```

```
            </div>
          </div>
          <!-- <div class="row">
            <div class="col-xs-12 col-lg-6 col-md-6
col-sm-12">
          <div class="panel" style="background-colo
r:transparent;color:white;border:1px solid orange">
                <div class="panel-header">
             <h3 class="panel-title"><i class="ion-
loading-b"></i>   Areador<span   class="stat_name"
id="id_estado_areador"    style="float:right"    ></
span></h3>
                </div>
                <div class="panel-body">
                  <div class="row">
                      <div class="col-xs-12 col-lg-
offset-4 col-lg-4 col-md-4 col-sm-4">
                     Accion: <input id="id_areador"
type="checkbox"    onchange    ="doSend('areador')"
name="actuador1"/>
                    </div>
                  </div>
                </div>
              </div>
            </div>

            <div class="col-xs-12 col-lg-6 col-md-6
col-sm-12">
          <div class="panel" style="background-colo
r:transparent;color:white;border:1px solid orange">
                <div class="panel-header">
               <h3 class="panel-title"><i class="fa
fa-lightbulb-o"></i>  Foco<span  class="stat_name"
id="id_estado_foco"  style="float:right"  ></span></
h3>
                </div>
```

```
                <div class="panel-body">
                  <div class="row">
                     <div class="col-xs-12 col-lg-
offset-4 col-lg-4 col-md-4 col-sm-4">
                          Accion: <input id="id_
foco" type="checkbox" onchange ="doSend('foco')"
name="actuador2"/>
                   </div>
                  </div>
                </div>
              </div>
            </div>

          </div> -->
        </div>
      </div>
    </div>
  <!-- jQuery -->
    <script src="assets/js/jquery.min.js"></script>
    <!-- easíng -->
     <script src="assets/js/jquery.easíng.1.3.min.
js"></script>
    <!-- bootstrap js plugins -->
    <script src="assets/bootstrap/js/bootstrap.min.
js"></script>
    <!-- top dropdown navigation -->
    <script src="assets/js/tinynav.js"></script>
    <!-- perfect scrollbar -->
     <script src="assets/lib/perfect-scrollbar/min/
perfect-scrollbar-0.4.8.with-mousewheel.min.js"></
script>

    <!-- common functions -->
   <script src="assets/js/tisa_common.js"></script>
```

```
    <!-- style switcher -->
       <script  src="assets/js/tisa_style_switcher.
js"></script>

  <!-- page specific plugins -->

    <!-- nvd3 charts -->

    <!-- FLOT CHARTS -->
       <script  src="assets/lib/flot/jquery.flot.min.
js"></script>
     <!-- FLOT RESIZE PLUGIN - allows the chart to
redraw when the window is resized -->
     <script src="assets/lib/flot/jquery.flot.resize.
min.js"></script>
    <!-- FLOT PIE PLUGIN - also used to draw donut
charts -->
    <script src="assets/lib/flot/jquery.flot.pie.min.
js"></script>
    <!-- FLOT CATEGORIES PLUGIN - Used to draw bar
charts -->
          <script   src="assets/lib/flot/jquery.flot.
categories.min.js"></script>
    <!-- clndr -->
  <script src="assets/lib/underscore-js/underscore-
min.js"></script>
     <script src="assets/lib/CLNDR/src/clndr.js"></
script>
    <!-- easy pie chart -->
      <script  src="assets/lib/easy-pie-chart/dist/
jquery.easypiechart.min.js"></script>
    <!-- owl carousel -->
        <script  src="assets/lib/owl-carousel/owl.
carousel.min.js"></script>
    <!-- dashboard functions -->
      <script  src="bootstrap-switch-master/dist/js/
```

```
bootstrap-switch.min.js"></script>

    <script type="text/javascript">

      $(function () {

          interactive_plot1 = $.plot("#id_grafico_
temp", [getRandomData1()], options);
       interactive_plot2 = $.plot("#id_grafico_hum",
[getRandomData2()], options);
       });
      $("#id_foco").bootstrapSwitch({
         labelText:'<i class="fa fa-lightbulb-o"></
i>',
        onText:'On',
        offText:'Off',
        onColor:'primary',
        offColor:'danger',
        handleWidth:'30'
      });
      $("#id_areador").bootstrapSwitch({
        labelText:'<i class="ion-loading-b"></i>',
        onText:'On',
        offText:'Off',
        onColor:'primary',
        offColor:'danger',
        handleWidth:'30'
      });
    </script>
    <!-- newsletter functions -->
      <script src="assets/js/apps/tisa_newsletter.
js"></script>
</body>
</html>
```

NOTA: Dentro de la carpeta proporcionada denominada “recursos” se encuentran los instaladores necesarios para la instalación e importación de la práctica.

Práctica 2

Anexo 1: Programación en Arduino

```
#include <MaxMatrix.h>
#include <avr/pgmspace.h>

/**
PROGMEM guarda datos en la memoria de programa en lugar de guardarla en el SRAM donde normalmente debería ir
**/
PROGMEM unsigned char const CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // “
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // ‘
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000,
```

```
B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000,
B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000,
B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000,
B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001,
B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110,
B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000,
B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110,
B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110,
B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111,
B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001,
B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000,
B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111,
B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110,
B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110,
B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000,
B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000,
B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000,
B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000,
B00000000, // =
```

```
  3, 8, B01000100, B00101000, B00010000, B00000000,
  B00000000, // >
  4, 8, B00000010, B01011001, B00001001, B00000110,
  B00000000, // ?
  5, 8, B00111110, B01001001, B01010101, B01011101,
  B00001110, // @
  4, 8, B01111110, B00010001, B00010001, B01111110,
  B00000000, // A
  4, 8, B01111111, B01001001, B01001001, B00110110,
  B00000000, // B
  4, 8, B00111110, B01000001, B01000001, B00100010,
  B00000000, // C
  4, 8, B01111111, B01000001, B01000001, B00111110,
  B00000000, // D
  4, 8, B01111111, B01001001, B01001001, B01000001,
  B00000000, // E
  4, 8, B01111111, B00001001, B00001001, B00000001,
  B00000000, // F
  4, 8, B00111110, B01000001, B01001001, B01111010,
  B00000000, // G
  4, 8, B01111111, B00001000, B00001000, B01111111,
  B00000000, // H
  3, 8, B01000001, B01111111, B01000001, B00000000,
  B00000000, // I
  4, 8, B00110000, B01000000, B01000001, B00111111,
  B00000000, // J
  4, 8, B01111111, B00001000, B00010100, B01100011,
  B00000000, // K
  4, 8, B01111111, B01000000, B01000000, B01000000,
  B00000000, // L
  5, 8, B01111111, B00000010, B00001100, B00000010,
  B01111111, // M
  5, 8, B01111111, B00000100, B00001000, B00010000,
  B01111111, // N
  4, 8, B00111110, B01000001, B01000001, B00111110,
  B00000000, // O
  4, 8, B01111111, B00001001, B00001001, B00000110,
  B00000000, // P
```

```
4, 8, B00111110, B01000001, B01000001, B10111110,
B00000000, // Q
4, 8, B01111111, B00001001, B00001001, B01110110,
B00000000, // R
4, 8, B01000110, B01001001, B01001001, B00110010,
B00000000, // S
5, 8, B00000001, B00000001, B01111111, B00000001,
B00000001, // T
4, 8, B00111111, B01000000, B01000000, B00111111,
B00000000, // U
5, 8, B00001111, B00110000, B01000000, B00110000,
B00001111, // V
5, 8, B00111111, B01000000, B00111000, B01000000,
B00111111, // W
5, 8, B01100011, B00010100, B00001000, B00010100,
B01100011, // X
5, 8, B00000111, B00001000, B01110000, B00001000,
B00000111, // Y
4, 8, B01100001, B01010001, B01001001, B01000111,
B00000000, // Z
2, 8, B01111111, B01000001, B00000000, B00000000,
B00000000, // [
4, 8, B00000001, B00000110, B00011000, B01100000,
B00000000, // \ backslash
2, 8, B01000001, B01111111, B00000000, B00000000,
B00000000, // ]
3, 8, B00000010, B00000001, B00000010, B00000000,
B00000000, // hat
4, 8, B01000000, B01000000, B01000000, B01000000,
B00000000, // _
2, 8, B00000001, B00000010, B00000000, B00000000,
B00000000, // `
4, 8, B00100000, B01010100, B01010100, B01111000,
B00000000, // a
4, 8, B01111111, B01000100, B01000100, B00111000,
B00000000, // b
4, 8, B00111000, B01000100, B01000100, B00101000,
B00000000, // c
```

```
  4, 8, B00111000, B01000100, B01000100, B01111111,
  B00000000, // d
  4, 8, B00111000, B01010100, B01010100, B00011000,
  B00000000, // e
  3, 8, B00000100, B01111110, B00000101, B00000000,
  B00000000, // f
  4, 8, B10011000, B10100100, B10100100, B01111000,
  B00000000, // g
  4, 8, B01111111, B00000100, B00000100, B01111000,
  B00000000, // h
  3, 8, B01000100, B01111101, B01000000, B00000000,
  B00000000, // i
  4, 8, B01000000, B10000000, B10000100, B01111101,
  B00000000, // j
  4, 8, B01111111, B00010000, B00101000, B01000100,
  B00000000, // k
  3, 8, B01000001, B01111111, B01000000, B00000000,
  B00000000, // l
  5, 8, B01111100, B00000100, B01111100, B00000100,
  B01111000, // m
  4, 8, B01111100, B00000100, B00000100, B01111000,
  B00000000, // n
  4, 8, B00111000, B01000100, B01000100, B00111000,
  B00000000, // o
  4, 8, B11111100, B00100100, B00100100, B00011000,
  B00000000, // p
  4, 8, B00011000, B00100100, B00100100, B11111100,
  B00000000, // q
  4, 8, B01111100, B00001000, B00000100, B00000100,
  B00000000, // r
  4, 8, B01001000, B01010100, B01010100, B00100100,
  B00000000, // s
  3, 8, B00000100, B00111111, B01000100, B00000000,
  B00000000, // t
  4, 8, B00111100, B01000000, B01000000, B01111100,
  B00000000, // u
  5, 8, B00011100, B00100000, B01000000, B00100000,
  B00011100, // v
```

```
5, 8, B00111100, B01000000, B00111100, B01000000,
B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000,
B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100,
B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000,
B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000,
B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000,
B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000,
B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100,
B00000000, // ~
};

int data = 8;    // DIN pin del modulo MAX7219
int load = 9;    // CS pin del modulo MAX7219
int clock = 10;  // CLK pin del modulo MAX7219

int maxInUse = 5;    //Cambie este valor dependiendo
del tipo de modulo MAX7219 que use

MaxMatrix m(data, load, clock, maxInUse); // Define
el modulo

byte buffer[10];

char mensaje1[] = "..    "; //Escriba el mensaje a
desplegar
//char mensaje2[] = "";
char msj[] =" ";

/**
```

```
Función inicial, aquí es donde se inicializan las
variables, librerías, pins etc.
esta funcion solo se ejecuta una vez después que el
Arduino ha sido encendido o reseteado.
**/
void setup(){
  pinMode(8,OUTPUT);       //Conexion a DIN
  pinMode(9,OUTPUT);       //Conexion a CS
  pinMode(10,OUTPUT);      //Conexion a CLK
  m.init();               // inicializa el modulo
  m.setIntensity(5);     // intencidad de los puntos
de la matriz, entre 1-5
   Serial.begin(9600);      // inicializa el puerto
serial
}
/**
Función que se ejecuta repetidamente "loop"
permitiendo al programa responder a cada momento
**/
void loop()
{
  byte c;
  int contador = 0;
  while (Serial.available()>0)// Lee el mensaje que
llega por el puerto serial
  {
    c = Serial.read();
    msj[contador] = char(c);
    contador++;
  }

  m.shiftLeft(false, true);
```

```
   // Despliega los mensajes almacenados en las
variables
  imprimirString(mensaje1, 40);
  imprimirString(msj, 100);
}

void imprimirChar(char c, int shift_speed){     //
Imprime caracteres
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(maxInUse*8, 0, buffer);
  m.setColumn(maxInUse*8 + buffer[0], 0);

  for (int i=0; i<buffer[0]+1; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}

void imprimirString(char[] s, int shift_speed)//
Imprime cadena de caracteres
{
  while (*s != 0){
    imprimirChar(*s, shift_speed);
    s++;
  }
}
```

Práctica 3

Anexo 1: Código nodejs.

```
//Programa que perimte leer los mensajes que publica en twitter un usuario determinado
// y publicar a un topico determinado si el mensaje publicado coincide con mensajes
// que estan establecidos en este programa.
var Twit = require('twit');
var mqtt = require('mqtt');
// direccion en la cual esta el broker
var serverMqtt = '192.168.1.11';
//Topicos a los cuales se van a publicar
var topico = '/actuador/';
//var topico2 = 'utmach/ArduinoWifi/motor';
var mensaje1 = 'ON1';
var mensaje2 = 'ON2';
var mensaje3 = 'OF1';
var mensaje4 = 'OF2';
//var mensaje_off = 'OFF';
//puerto en el cual esta el broker
var puerto = 1883;

//variables del apiKey de twitter
// agregar sus key y Access_Token  o no podran leer los mensajes de twitter
var T = new Twit({
   consumer_key : 'BTdxQ1GnnaYrrUBqD3hJKCIIl'
   , consumer_secret : 'CaiTdhapax6X42MnUaCXPiQFFBTcLhpyZiRiQHZGWTvYDDtFfr'
   , access_token : '246324808-NyGOcb0TjioUoscvqHPK8pdfyhiD000zJKR7Wi3O'
   , access_token_secret : 'Zn4ajYHHVKv7pRPffcGgK57GyGMry1vqNCUWi
```

```
KUtLlJ2S'
})

console.log('|-------------------------------------
-----|')
console.log('|                                     |')
console.log('|  MQTT  CLIENTE  conectado  atraves  de
TWITTER |')
console.log('|                                     |')
console.log('|-------------------------------------
-----|')
console.log('')

//crea un nuevo cliente MQTT
var client = mqtt.createClient(puerto, serverMqtt);
client.options.reconnectPeriod = 0;

//Busca al usuario con el ID 14585199 en twitter
var  stream  =  T.stream('statuses/filter',  {  follow:
'246324808' });

//cualquier  mensaje  que  publique  lo  compara  y  si
coincide publica el topico con el mensaje
stream.on('tweet', function (tweet) {
  console.log(tweet.text + ' (' + tweet.user.screen_
name  + ')');
  //si el mensaje coincide con GalileoON publica al
bkoker sobre el topico especificado.
  if (tweet.text.match('EnciendeFoco')){
    console.log('mensaje coincide');
     client.publish(topico, mensaje2);
    console.log('Publicando mensaje: '+topico+' ON2'
)
    }
```

```
  //si el mensaje coincide con GalileoOFF publica al
bkoker sobre el topico especificado.
    else if(tweet.text.match('EnciendeMotor')){
      console.log('mensaje coincide');
      client.publish(topico, mensaje1);
         console.log('Publicando mensaje: '+topico+'
ON1');
    }
     //si el mensaje coincide con MegaON publica al
bkoker sobre el topico especificado.
    else if (tweet.text.match('ApagaMotor')){
      console.log('mensaje coincide');
      client.publish(topico, mensaje3);
         console.log('Publicando mensaje: '+topico+'
OF1');
    }
     //si el mensaje coincide con MegaOFF publica al
bkoker sobre el topico especificado.
    else if (tweet.text.match('ApagaFoco')){
      console.log('mensaje coincide');
      client.publish(topico, mensaje4);
         console.log('Publicando mensaje: '+topico+'
OF2');
    }
    else {
        console.log('mensaje no coincide');
   }
});
}
//si el mensaje coincide con GalileoOFF publica al
bkoker sobre el topico especificado.
else if(tweet.text.match('EnciendeMotor')){
console.log('mensaje coincide');
client.publish(topico, mensaje1);
console.log('Publicando mensaje: '+topico+' ON1');
```

```
}
//si el mensaje coincide con MegaON publica al bkoker
sobre el topico especificado.
else if (tweet.text.match('ApagaMotor')){
console.log('mensaje coincide');
client.publish(topico, mensaje3);
console.log('Publicando mensaje: '+topico+' OF1');
}
//si el mensaje coincide con MegaOFF publica al bkoker
sobre el topico especificado.
else if (tweet.text.match('ApagaFoco')){
console.log('mensaje coincide');
client.publish(topico, mensaje4);
console.log('Publicando mensaje: '+topico+' OF2');
}
else {
console.log('mensaje no coincide');
}
});
//si el mensaje coincide con GalileoOFF publica al
bkoker sobre el topico especificado.
else if(tweet.text.match('EnciendeMotor')){
console.log('mensaje coincide');
client.publish(topico, mensaje1);
console.log('Publicando mensaje: '+topico+' ON1');
}
//si el mensaje coincide con MegaON publica al bkoker
sobre el topico especificado.
else if (tweet.text.match('ApagaMotor')){
console.log('mensaje coincide');
client.publish(topico, mensaje3);
console.log('Publicando mensaje: '+topico+' OF1');
}
//si el mensaje coincide con MegaOFF publica al bkoker
```

```
sobre el topico especificado.
else if (tweet.text.match('ApagaFoco')){
console.log('mensaje coincide');
client.publish(topico, mensaje4);
console.log('Publicando mensaje: '+topico+' OF2');
}
else {
console.log('mensaje no coincide');
}
```

Anexo 2: Programación

```
// IMPORTACION DE LIBRERIAS
#include <SPI.h>
#include <Ethernet.h>
#include "max6675.h"
#include <PubSubClient.h>
#include <EEPROM.h>
// CONFIGURACION MQTT
#define MQTT_SERVER_NAME "192.168.1.11"
#define MQTT_SERVER_PORT 1883
#define MQTT_CLIENTID   "90-A2-DA-0F-6E-64"
#define MQTT_TOPIC_SUBSCRIBE  "/actuador/"
#define MQTT_TOPIC_PUBLISH_LECTURAS   "/sensor/"
// PARAMETROS TCP
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 20);
String direccionMac = "90-A2-DA-0F-6E-64";

// CREACION DE OBJETOS PARA LA CONEXION
PubSubClient client;
EthernetClient eclient;
// PIN PARA ACTUADOR
int pinFoco = 3;
```

```
// DECLARACION DE VARIABLES GLOBALES
char message_buff[100];
void setup() {
  pinMode(pinAreador, OUTPUT);
  pinMode(pinFoco, OUTPUT);
  // INCIALIZACION DE CONSOLA SERIAL
  Serial.begin(9600);
while (!Serial) {
    ; // wait for serial
port to connect. Needed for Leonardo only
  }
  // CONEXION A LA RED
  Serial.println("Obteniendo IP");
  /*if (Ethernet.begin(mac) == 0) {
    Serial.println("Fallo al Obtener IP por DHCP");
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());
  } else {
    Serial.print("IP Dinamica: ");
    Serial.println(Ethernet.localIP());
  }*/
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());

// CONEXION AL SERVIDOR MQTT
    client = PubSubClient(MQTT_SERVER_NAME, 1883,
callback, eclient);
}
void loop() {
  // VERIFICACION DE CONEXION AL SERVIDOR MQTT
  if (!client.connected()) {
```

```
Serial.println("Conectando servidor MQTT");
    if (client.connect(MQTT_CLIENTID)) {
        suscribir();
        publicar_lecturas();
    } else {
      Serial.println("No se pudo conectar al servidor
MQTT");
// DECLARACION DE VARIABLES GLOBALES
char message_buff[100];

void setup() {

  pinMode(pinFoco, OUTPUT);
  // INCIALIZACION DE CONSOLA SERIAL
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for
Leonardo only
  }
  // CONEXION A LA RED
  Serial.println("Obteniendo IP");
  /*if (Ethernet.begin(mac) == 0) {
    Serial.println("Fallo al Obtener IP por DHCP");
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());
  } else {
    Serial.print("IP Dinamica: ");
    Serial.println(Ethernet.localIP());
  }*/
    Ethernet.begin(mac, ip);
    Serial.print("IP Manual: ");
    Serial.println(Ethernet.localIP());
```

```
    // CONEXION AL SERVIDOR MQTT
    client  =  PubSubClient(MQTT_SERVER_NAME,  1883,
callback, eclient);
}
void loop() {
  // VERIFICACION DE CONEXION AL SERVIDOR MQTT
  if (!client.connected()) {
    Serial.println("Conectando servidor MQTT");
    if (client.connect(MQTT_CLIENTID)) {
        suscribir();
        publicar_lecturas();
    } else {
      Serial.println("No se pudo conectar al servidor
MQTT");
    }
  } else {
      //suscribir();
      publicar_lecturas();
  }
  client.loop();
  delay(500);
}
// METODO PARA PUBLICACION DE LECTURAS AL SERVIDOR
MQTT
void publicar_lecturas() {
  int sensorValue = thermocouple.readCelsius();
   int sensorValue1 = map(analogRead(pinH), 0, 1024,
0, 100);
  sensorValue1 = 100 - sensorValue1;
  String json = buildJson(sensorValue, sensorValue1);
  char jsonStr[200];
  json.toCharArray(jsonStr, 200);
         client.publish(MQTT_TOPIC_PUBLISH_LECTURAS,
jsonStr);
```

```
 Serial.println("Publicando al topico : "+ String(MQTT_
TOPIC_PUBLISH_LECTURAS));
}
  } else {
      //suscribir();
      publicar_lecturas();
  }
  client.loop();
  delay(500);
}
// METODO PARA PUBLICACION DE LECTURAS AL SERVIDOR
MQTT
void publicar_lecturas() {
  int sensorValue = thermocouple.readCelsius();
   int sensorValue1 = map(analogRead(pinH), 0, 1024,
0, 100);
  sensorValue1 = 100 - sensorValue1;
  String json = buildJson(sensorValue, sensorValue1);
  char jsonStr[200];
  json.toCharArray(jsonStr, 200);
         client.publish(MQTT_TOPIC_PUBLISH_LECTURAS,
jsonStr);
 Serial.println("Publicando al topico : "+ String(MQTT_
TOPIC_PUBLISH_LECTURAS));
  //Serial.println(MQTT_TOPIC_PUBLISH_LECTURAS);
  Serial.println(jsonStr);
}
String convierteIp(IPAddress address)
{
  return String(address[0]) + "." +
         String(address[1]) + "." +
         String(address[2]) + "." +
         String(address[3]);
}
```

```
void suscribir() {
    client.subscribe(MQTT_TOPIC_SUBSCRIBE);
  Serial.println("Suscrito al topico : " + String(MQTT_
TOPIC_SUBSCRIBE));
    //Serial.println(MQTT_TOPIC_SUBSCRIBE);
}

void callback(char* topic, byte* payload, unsigned
int length) {
  int i;
  for (i = 0; i < length; i++) {
    message_buff[i] = payload[i];
    //Serial.print(payload[i]);
  }
  message_buff[i] = '\0';
  String msgString = String(message_buff);
  Serial.print("Recibiendo : ");
  Serial.println(msgString);
  if ( msgString.equals("ON1" )) {
    Serial.println("Encendiendo Areador");
    digitalWrite(pinAreador,HIGH);
  }
  if ( msgString.equals("OF1" )) {
    Serial.println("Apagando Areador");
    digitalWrite(pinAreador,LOW);
    }
  if ( msgString.equals("ON2" )) {
    Serial.println("Encendiendo Foco");
    digitalWrite(pinFoco,HIGH);
     }
  if ( msgString.equals("OF2" )) {
    Serial.println("Apagando Foco");
    digitalWrite(pinFoco,LOW);
```

```
      }
  }
  String buildJson(int sensorValue, int sensorValue1) {
    String data = "{";
    data += "\"device\": {";
    data += " \"id\": \"";
    data += MQTT_CLIENTID;
    data += "\",";
    data += "\"temperatura\": ";
    data += "\"";
    data += (int)sensorValue;
    data += "\"";
    data += ",";
    data += "\"humedad\": ";
    data += "\"";
    data += (int)sensorValue1;
  data += "\"";
    data += ",";
    data += "\"areador\": ";
    data += "\"";
    data += (int)0;
    data += "\"";
    data += ",";
    data += "\"foco\": ";
    data += "\"";
    data += (int)1;
    data += "\"";
    data += "}";
    data += "}";
    return data;
  }
    //Serial.println(MQTT_TOPIC_PUBLISH_LECTURAS);
    Serial.println(jsonStr);
```

```
}
String convierteIp(IPAddress address)
{
  return String(address[0]) + “.” +
         String(address[1]) + “.” +
         String(address[2]) + “.” +
         String(address[3]);
}
void suscribir() {
    client.subscribe(MQTT_TOPIC_SUBSCRIBE);
  Serial.println(“Suscrito al topico : “ + String(MQTT_
TOPIC_SUBSCRIBE));
    //Serial.println(MQTT_TOPIC_SUBSCRIBE);
}
void callback(char* topic, byte* payload, unsigned
int length) {
  int i;
  for (i = 0; i < length; i++) {
    message_buff[i] = payload[i];
    //Serial.print(payload[i]);
  }
  message_buff[i] = ‘\0’;
  String msgString = String(message_buff);
  Serial.print(“Recibiendo : “);
  Serial.println(msgString);
  if ( msgString.equals(“ON1” )) {
    Serial.println(“Encendiendo Areador”);
    digitalWrite(pinAreador,HIGH);
  }
  if ( msgString.equals(“OF1” )) {
    Serial.println(“Apagando Areador”);
    digitalWrite(pinAreador,LOW);
    }
```

```
    if ( msgString.equals("ON2" )) {
      Serial.println("Encendiendo Foco");
      digitalWrite(pinFoco,HIGH);
      }
    if ( msgString.equals("OF2" )) {
      Serial.println("Apagando Foco");
      digitalWrite(pinFoco,LOW);
       }
  }
  String buildJson(int sensorValue, int sensorValue1) {
    String data = "{";
    data += "\"device\": {";
    data += " \"id\": \"";
    data += MQTT_CLIENTID;
    data += "\",";
    data += "\"temperatura\": ";
  data += "\"";
    data += (int)sensorValue;
    data += "\"";
    data += ",";
    data += "\"humedad\": ";
    data += "\"";
    data += (int)sensorValue1;
    data += "\"";
    data += ",";
    data += "\"areador\": ";
    data += "\"";
    data += (int)0;
    data += "\"";
    data += ",";
    data += "\"foco\": ";
    data += "\"";
    data += (int)1;
```

```
  data += "\"";
  data += "}";
  data += "}";
  return data;
}
void loop() {
  // VERIFICACION DE CONEXION AL SERVIDOR MQTT
  if (!client.connected()) {
    Serial.println("Conectando servidor MQTT");
    if (client.connect(MQTT_CLIENTID)) {
        suscribir();
        publicar_lecturas();
    } else {
      Serial.println("No se pudo conectar al servidor
MQTT");
    }
  } else {
      //suscribir();
      publicar_lecturas();
  }
  client.loop();
  delay(500);
}
// METODO PARA PUBLICACION DE LECTURAS AL SERVIDOR
MQTT
void publicar_lecturas() {
  int sensorValue = thermocouple.readCelsius();
   int sensorValue1 = map(analogRead(pinH), 0, 1024,
0, 100);
  sensorValue1 = 100 - sensorValue1;
  String json = buildJson(sensorValue, sensorValue1);
  char jsonStr[200];
  json.toCharArray(jsonStr, 200);
         client.publish(MQTT_TOPIC_PUBLISH_LECTURAS,
```

```
jsonStr);
 Serial.println("Publicando al topico : "+ String(MQTT_
TOPIC_PUBLISH_LECTURAS));
  //Serial.println(MQTT_TOPIC_PUBLISH_LECTURAS);
  Serial.println(jsonStr);
}
String convierteIp(IPAddress address)
{
  return String(address[0]) + "." +
         String(address[1]) + "." +
         String(address[2]) + "." +
         String(address[3]);
}
void suscribir() {
    client.subscribe(MQTT_TOPIC_SUBSCRIBE);
  Serial.println("Suscrito al topico : " + String(MQTT_
TOPIC_SUBSCRIBE));
    //Serial.println(MQTT_TOPIC_SUBSCRIBE);
}
void callback(char* topic, byte* payload, unsigned
int length) {
  int i;
  for (i = 0; i < length; i++) {
    message_buff[i] = payload[i];
    //Serial.print(payload[i]);
  }
  message_buff[i] = '\0';
  String msgString = String(message_buff);
  Serial.print("Recibiendo : ");
  Serial.println(msgString);
  if ( msgString.equals("ON1" )) {
    Serial.println("Encendiendo Areador");
    digitalWrite(pinAreador,HIGH);
  }
```

```
  if ( msgString.equals("OF1" )) {
    Serial.println("Apagando Areador");
    digitalWrite(pinAreador,LOW);
    }
  if ( msgString.equals("ON2" )) {
    Serial.println("Encendiendo Foco");
    digitalWrite(pinFoco,HIGH);
     }
  if ( msgString.equals("OF2" )) {
    Serial.println("Apagando Foco");
    digitalWrite(pinFoco,LOW);
}
}
String buildJson(int sensorValue, int sensorValue1) {
  String data = "{";
  data += "\"device\": {";
  data += " \"id\": \"";
  data += MQTT_CLIENTID;
  data += "\",";
  data += "\"temperatura\": ";
  data += "\"";
  data += (int)sensorValue;
  data += "\"";
  data += ",";
  data += "\"humedad\": ";
  data += "\"";
  data += (int)sensorValue1;
  data += "\"";
  data += ",";
  data += "\"areador\": ";
  data += "\"";
  data += (int)0;
  data += "\"";
```

```
    data += ",";
    data += "\"foco\": ";
    data += "\"";
    data += (int)1;
    data += "\"";
    data += "}";
    data += "}";
    return data;
}
String convierteIp(IPAddress address)
{
    return String(address[0]) + "." +
           String(address[1]) + "." +
           String(address[2]) + "." +
           String(address[3]);}
void suscribir() {
    client.subscribe(MQTT_TOPIC_SUBSCRIBE);
  Serial.println("Suscrito al topico : " + String(MQTT_
TOPIC_SUBSCRIBE));
    //Serial.println(MQTT_TOPIC_SUBSCRIBE);}
void callback(char* topic, byte* payload, unsigned
int length) {
  int i;
  for (i = 0; i < length; i++) {
    message_buff[i] = payload[i];
    //Serial.print(payload[i]);
  }
  message_buff[i] = '\0';
  String msgString = String(message_buff);
  Serial.print("Recibiendo : ");
  Serial.println(msgString);
   if ( msgString.equals("ON2" )) {
    Serial.println("Encendiendo Foco");
```

```
    digitalWrite(pinFoco,HIGH);
  }
  if ( msgString.equals("OF2" )) {
    Serial.println("Apagando Foco");
    digitalWrite(pinFoco,LOW);
     } }
String buildJson(int sensorValue, int sensorValue1) {
  String data = "{";
  data += "\"device\": {";
  data += " \"id\": \"";
  data += MQTT_CLIENTID;
  data += "\",";
  data += "\"temperatura\": ";
  data += "\"";
  data += (int)sensorValue;
  data += "\"";
  data += ",";
  data += "\"humedad\": ";
  data += "\"";
  data += (int)sensorValue1;
  data += "\"";
  data += ",";
  data += "\"areador\": ";
  data += "\"";
  data += (int)0;
  data += "\"";
  data += ",";
  data += "\"foco\": ";
  data += "\"";
  data += (int)1;
  data += "\"";
data += "}";
  data += "}";
```

```
  return data;
}
String buildJson(int sensorValue, int sensorValue1) {
  String data = "{";
  data += "\"device\": {";
  data += " \"id\": \"";
  data += MQTT_CLIENTID;
  data += "\",";
    data += "\"areador\": ";
  data += "\"";
  data += (int)0;
  data += "\"";
  data += ",";
  data += "\"foco\": ";
  data += "\"";
  data += (int)1;
  data += "\"";
  data += "}";
  data += "}";
  return data;
}
```

Ingeniería y Tecnología

3ciencias