Trabajo de Inserción Profesional

Informe Final

*Título:*

Integración de tecnología para control y automatización de espacio físico*.*

*Alumno:*

*Martin Alejandro Melo.*

*Director:*

*Ing. José Luis Di Biase.*

*Carrera:*

*Tecnicatura en Programación Informática.*



Contenido

[Introducción 4](#_Toc410745893)

[Contexto 4](#_Toc410745894)

[Problemática 4](#_Toc410745895)

[Objetivos 5](#_Toc410745896)

[Resumen 7](#_Toc410745897)

[Desarrollo de la solución propuesta 8](#_Toc410745898)

[Descripción/Contexto General 8](#_Toc410745899)

[Características de la solución desarrollada 9](#_Toc410745900)

[Hardware 10](#_Toc410745901)

[Componente base 11](#_Toc410745902)

[Sensores específicos 12](#_Toc410745903)

[Componentes específicos 12](#_Toc410745904)

[Diagramas de conexión 13](#_Toc410745905)

[Firmware 13](#_Toc410745906)

[PubSubClient 14](#_Toc410745907)

[ArduinoJson 14](#_Toc410745908)

[Software 14](#_Toc410745909)

[Front-End 15](#_Toc410745910)

[Back-End 16](#_Toc410745911)

[Middleware 17](#_Toc410745912)

[Interconexión 18](#_Toc410745913)

[Protocolo de comunicación 19](#_Toc410745914)

[Conclusiones 23](#_Toc410745915)

[*Anexos* 24](#_Toc410745916)

[Anexo Hardware 24](#_Toc410745917)

[Arduino 24](#_Toc410745918)

[Diagramas del Hardware 26](#_Toc410745919)

[Sensor de temperatura 28](#_Toc410745920)

[Anexo Software 28](#_Toc410745921)

[M2M 28](#_Toc410745922)

[MQTT 28](#_Toc410745923)

[Mosca 29](#_Toc410745924)

[Socket.IO 29](#_Toc410745925)

[¿Qué es JSON? 30](#_Toc410745926)

[MongoDB 33](#_Toc410745927)

[Express 33](#_Toc410745928)

[Node-Schedule 33](#_Toc410745929)

[Malhar-angular 33](#_Toc410745930)

[NodeJS 34](#_Toc410745931)

[MeanJS 34](#_Toc410745932)

[Anexo Extra 35](#_Toc410745933)

[Licencia MIT 35](#_Toc410745934)

[Internet Of Things 36](#_Toc410745935)

[Automatización y control de los espacios físicos 37](#_Toc410745936)

[Referencias 38](#_Toc410745937)

[Referencias Adicionales 39](#_Toc410745938)

# ***Introducción***

## Contexto

En la actualidad es cada vez más común y frecuente ver dispositivos inteligentes en nuestras vidas. Desde Smartphones hasta robots de servicios o domésticos, los cuales realizan tareas diarias la cuales pueden ser realizadas por una persona. Todos estos dispositivos se encuentran aislados entre ellos, es decir, no se comunican entre sí o con un dispositivo central. Si estos estuviesen comunicados o controlados por un sistema central, se podrían obtener mejores resultados.

## Problemática

El desafío del trabajo es poder centralizar información e interactuar con múltiples dispositivos y sensores electrónicos localizados remotamente.

No solo los dispositivos no están en el mismo espacio físico, sino que quien interactúa y los controla potencialmente está también en otro lugar.

Finalmente la idea es poder recolectar datos generados por los dispositivos, visualizarlos en tiempo real, agendar acciones futuras y poder cambiar su estado a partir de una aplicación web en internet.

# ***Objetivos***

La empresa en la que actualmente me desempeño programando necesita tener una herramienta que sirva de control y automatización de ciertos espacios físicos (oficina, depósito, zona común, etc.). En este momento estos controles son llevados a cabo por empleados que los realizan diariamente cuando algún superior se lo indica dándole además las instrucciones necesarias para su realización.

Se necesita tener acceso a través de internet a un sitio web con la información de los distintos controles que se tengan los espacios físicos para corroborar estados, efectuar acciones o configurar tareas automatizadas para que se realicen según un calendario.

Los principales controles realizados son los siguientes:

* Iluminación: Se desea poder controlar la iluminación de determinados sectores de forma automática, es decir configurar su encendido y apagado determinando un calendario para el mismo o manualmente.
* Climatización: Esto sería idéntico que para la iluminación ya que determinados lugares del recinto cuentan con diferentes rangos necesarios de humedad y temperatura (por ejemplo: un depósito de documentos en papel o una habitación con servidores).
* Empleados: Se desea poder controlar la llegada y partida de los empleados al recinto a través de alguna forma de control como un lector de huellas digital.
* Seguridad: Se desea poder conocer el estado de los distintos puntos de acceso(puertas) al recinto.

Estos 4 controles principales se llevan a cabo diariamente y son de una importancia vital para la empresa.

La elaboración de este trabajo tuvo como objetivo principal desarrollar una herramienta que automatice todos los aspectos mencionados en la sección anterior. Dicha herramienta es intuitiva y de fácil acceso, a su vez esta es distribuida como software libre bajo la licencia MIT.

# ***Resumen***

En la actualidad es cada vez más común y frecuente ver dispositivos inteligentes en nuestras vidas. Desde Smartphones hasta robots de servicios o domésticos, los cuales realizan tareas diarias la cuales pueden ser realizadas por una persona. Todos estos dispositivos se encuentran aislados entre ellos, es decir, no se comunican entre sí o con un dispositivo central. Si estos estuviesen comunicados o controlados por un sistema central, se podrían obtener mejores resultados.

El desafío del trabajo es poder centralizar información e interactuar con múltiples dispositivos y sensores electrónicos localizados remotamente.

No solo los dispositivos no están en el mismo espacio físico, sino que quien interactúa y los controla potencialmente está también en otro lugar.

La solución propuesta es la de desarrollar una herramienta la cual logre unir la electrónica y la programación para formar una solución capaz de simplificar tareas que día a día se realizan de forma presencial. Como por ejemplo verificar el estado de una puerta, encender o apagar luces, obtener diferentes datos a través de sensores que se encuentren en el recinto, etc. Todos estos se va a unir utilizando protocolos de comunicación entre los dispositivos y el programa central con el cual se van a comunicar. El usuario que utilice el sistema se podrá comunicar a través de un sitio web(Front-End).

# ***Desarrollo de la solución propuesta***

## Descripción/Contexto General

Este proyecto se basó en los conceptos de Automatización y control de los espacios físicos y también en el concepto de Internet of Things para poder obtener una herramienta útil.

El Internet de las cosas o Internet Of Things es un término que se relacionan mucho con la web. Pero también resulta un concepto un poco abstracto para los no iniciados pese a que ha estado ganando bastante popularidad en el último tiempo. La idea que intenta representar queda bastante bien ilustrada por su nombre, cosas cotidianas que se conectan a Internet, pero que en realidad se trata de mucho más que eso.

La Domotica es un conjunto de sistemas capaces de automatizar una vivienda, que pueden estar integrados por medio de redes cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

## Características de la solución desarrollada

Considerando los [objetivos](#_Objetivos) descriptos en este mismo documento, se creó una aplicación que cuenta con las siguientes funcionalidades:

* Encontrar y comunicarse con los distintos dispositivos físicos conectados a mi aplicación.
* Recolectar información de los distintos sensores que posea un dispositivo.
* Crear, editar y listar tareas automáticas, las cuales se ejecutaran en un determinado calendario y realizaran una acción determinada. Como por ejemplo cambiar el estado de una luz determinada.
* Crear, editar y listar configuraciones de climatización para los distintos sensores de un dispositivo.
* Crear, editar y listar Widgets, los cuales representan la información que se obtiene de los distintos dispositivos.
* Crear, editar y listar dispositivos, los cuales se utilizan para obtener información de los dispositivos físicos.
* Crear, editar y listar secciones, las cuales representan el lugar físico donde se encuentran los sensores.
* Personalizar la pagina principal con los distintos widgets.
* Registrarse para tener un perfil de usuario.
* Editar el perfil del usuario.
* Cambiar la clave del usuario.
* Obtener la lista de los controles de horario de llegada o partida del personal realizados.
* Obtener la lista de los empleados registrados.

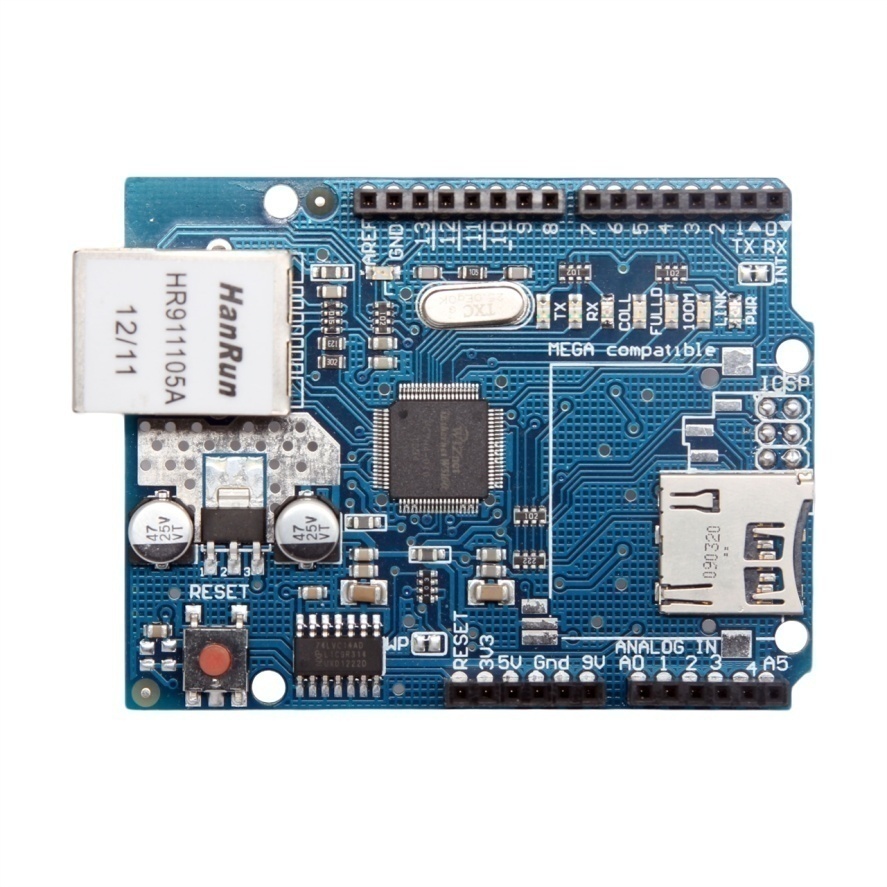
## Hardware

El hardware es la parte que está formado por los componentes eléctricos y/o electrónicos que son necesarios para hacer que el sistema funcione. El mismo se encarga de suministrar información al sistema y a su vez al cual se le podrá pedir realizar distintos tipos de acciones estará basado en la plataforma Arduino.

### Componente base

Los componentes de base necesarios para que el sistema funcione son:

* Arduino Mega2560: Es utilizado como hardware principal al cual se tendrá conectado distintos tipos de circuitos y sensores.
* Ethernet Shield con chipset Wiznet W510: El cual es utilizado para la comunicación con la aplicación que utilizara el usuario.



A los componentes anteriores se pueden añadir los sensores específicos los cuales se utilizan para extraer información del ambiente.

### Sensores específicos

El sensor de temperatura utilizado en el circuito impreso es un LM35DZ.

También se utilizaron ciertos componentes específicos para simular sensores de puerta o para controlar el cambio de estado de otros componentes electrónicos como las luces.

### Componentes específicos[[1]](#footnote-2)

Los componentes específicos utilizados son los siguientes:

* ULN2803. Es un array de Darlington que se utiliza para controlar las luces que se encuentran conectadas al circuito de las luces. Su hoja de datos es la siguiente: [http://www.ti.com/lit/ds/symlink/uln2803.pdf](http://www.ti.com/lit/ds/symlink/uln2803a.pdf).
* Botones de estado normal abierto. Cada botón se utiliza para simular apertura de puertas.
* Resistencia 1K ohm. Se utilizan para eliminar el efecto "antena" de los cables que se conectan a las entrada del Arduino, estas resistencia van junto a los botones.

### Diagramas de conexión

Se agregan en el [Anexo](#_Diagramas_del_Hardware) los esquemas de conexión de los componentes específicos que conforman el hardware.(Diagrama eléctrico y esquema en protoboard)

El primero de ellos pertenece al esquema de conexión de los componentes al micro controlador. La segunda imagen pertenece al esquema de conexión de los componentes utilizando un protoboard[[2]](#footnote-3).

## Firmware

El firmware es el software encargado de manejar directamente los dispositivos a nivel eléctrico.

Está programado en el lenguaje C y está grabado en la memoria del micro controlador.

En el mismo se incluyen las siguientes librerías:

* PubSubClient.
* ArduinoJson.

### PubSubClient

Esta librería permite la suscripción y publicación de mensajes contra un servidor que soporte el protocolo MQTT. Y es utilizado para enviar y recibir información desde los micro controladores.

### ArduinoJson

Permite el parseo y utilización del texto con formato JSON que se le envía al micro controlador con las instrucciones a realizar.

## Software

La aplicación tiene 3 partes bien diferenciadas ( Back-End, Middlewarey Front-End ). Cada uno de estos componentes cumple un rol determinado y son independientes. De manera que puedan funcionar los componentes en diferentes equipos.

Tanto el Back-End como el Front-End se encuentran construidas a partir del generador de código MeanJS. El cual permite generar aplicaciones web utilizando el Stack de aplicaciones (MongoDB, Express, AngularJS y NodeJS).Se utilizo esta herramienta ya que permite reutilizar funcionalidades comunes de los desarrollo web, lo cual ayudo a minimizar el tiempo de desarrollo.

### Front-End

El Front-End es el componente que hace de interfaz web a partir de la cual el usuario interactúa. Está desarrollado íntegramente con Javascript en particular con el framework AngularJS como base y múltiples bibliotecas adicionales.

En este componente se puede realizar lo siguiente:

* Crear, editar y listar tareas automáticas, las cuales se ejecutaran en un determinado calendario y realizaran una acción determinada. Como por ejemplo cambiar el estado de una luz determinada.
* Crear, editar y listar configuraciones de climatización para los distintos sensores de un dispositivo.
* Crear, editar y listar Widgets, los cuales representan la información que se obtiene de los distintos dispositivos.
* Crear, editar y listar dispositivos, los cuales se utilizan para obtener información de los controladores físicos.
* Crear, editar y listar secciones, las cuales representan el lugar físico donde se encuentran los sensores.
* Personalizar la pagina principal con los distintos widgets.
* Registrarse para tener un perfil de usuario.
* Editar el perfil del usuario.
* Cambiar la clave del usuario.
* Obtener la lista de los controles de horario de llegada o partida del personal realizados.
* Obtener la lista de los empleados registrados.

### Back-End

El Back-End es el componente encargado de servir las páginas web y de la persistencia de datos.   
Está desarrollado utilizando NodeJS.

Las funcionalidades de este componente son las siguientes:

* Provee las páginas web que el Front-End mostrara al usuario.
* Almacenar en base de datos los registros de control de llegada o partida de los empleados.
* Almacenar en base de datos la información de widgets creados por el usuario.

También se utilizan las siguientes librerías y Frameworks:

* MongoDB: Es el motor de la base de datos.
* Mongoose: Utilizado para administrar la base de datos.
* Express: Utilizado para unir el Back-End con el Front-End ya que es el que provee una vía de comunicación entre ambos puntos.
* node-schedule: Utilizada para llevar a cabo las tareas automáticas que se creen en la aplicación.

### Middleware

El Middleware es la parte de la aplicación que permite la comunicación entre los distintos dispositivos y el Front-End. Hace de nexo, intercambiando datos entre el Front-End y los distintos dispositivos de control.

Por ejemplo cuando el usuario desea encender una luz, el middleware será el encargado de informarle al Dispositivo de llevar a cabo esta acción.

También fue desarrollado con NodeJS.

Además se utilizan dos protocolos de comunicación:

* Mqtt: Se utilizo Mosca Servidor de mensajes que permite la comunicación entre las distintas aplicaciones
* Websockets: Se utilizó Socket.io para crear un servidor de mensajes web que habilita la comunicación directa entre el Front-End y los dispositivos.
* Mosca: En la aplicación se utiliza como servidor de mensajes principal para permitir la comunicación entre todos los servicios.

Esta parte está alojada en el mismo servidor que el Back-End, ya que no es necesaria una separación física. Pero esta se puede realizar de ser necesaria.

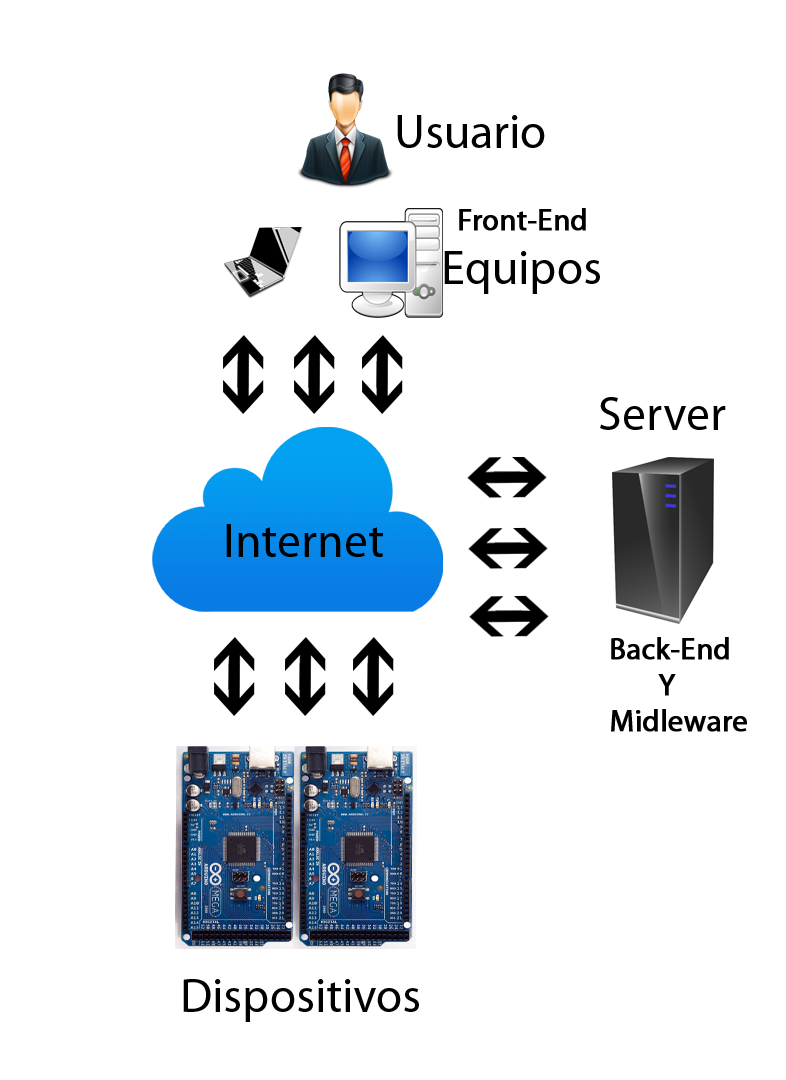
## Interconexión

Las distintas partes de la aplicación se comunican utilizando un protocolo diferente para cada parte.

El Front-End el cual de entre las diversas funcionalidades que posee tiene la cualidad de "comunicarse" con los controladores, esta función no se lleva a cabo directamente. El flujo de comunicación se da entre 3 partes y de la siguiente manera:

1. El Front-End demanda la información de un sensor o la petición de realizar una acción determinada.
2. El Middleware recibe esta petición y se la envía al controlador que está a la espera de estas peticiones.
3. El controlador recibe, procesa y contesta si es necesario la petición hacia el Middleware, el cual se lo devuelve al Front-End si es que está a la espera de una repuesta.

El siguiente grafico describe la conexión del sistema.



### Protocolo de comunicación

El protocolo de comunicación utilizado se basa en el envió de pequeñas porciones de texto con el formato JSON. Que contienen pequeña información de las acciones que se desean realizar.

#### Contenidos de los mensajes

Para esta aplicación se creó una estructura básica y sencilla para poder comunicarse e interactuar con los dispositivos y viceversa.

La estructura para comunicarse con los dispositivos está formada de la siguiente manera:

{

"command": "comando",

"destino": "destino",

"datos": {}

}

Donde el "comando " es el tipo de petición a realizar, pudiendo ser "accion" o "exponerServicios". Siendo la primera la cual informa que el dispositivo va a tener que realizar una acción como por ejemplo encender una luz o enviar información de un sensor, Y la segunda para que el dispositivo envié la información de los servicios que provee el mismo.

El "destino" es para marcar que tipo de petición se desea como por ejemplo si es iluminación, sensores o climatización. Si el comando fue "exponerServicios" este campo sirve para denotar por cual servicio se está preguntando, para este comando el campo puede tener los siguientes valores: "luz", "sensor" y "todo".

El campo datos es otro json que depende de la sección de destino, ya que la información es diferente para cada sección. Para la sección "Iluminación" el campo "datos" tendrá la siguiente estructura:

{

"id": "número de luz",

"estado": "valor"

}

Donde "id" tendrá el número que pertenece a una luz que el dispositivo tiene asignado. Y "estado" será "on" o "off" siendo el primero para denotar el encendido y el segundo para denotar el apagado de la luz.

Para la sección "sensores" el campo "datos" tendrá la siguiente estructura:

{

"id": "tipo de sensor",

"posicion": "nombre de la posición"

}

Donde "id" pueden ser "temperatura" o "puertas", siendo el primero para pedir información de sensores que son numéricos(por ejemplo un sensor de temperatura ambiental con un rango de valores entre -50 y 150 grados Celsius) y el segundo para sensores que estados de forma booleana, por ejemplo un sensor de apertura donde se desea tener solo 2 valores posibles(true para denotar un estado y false para denotar otro, pudiendo ser el primero para puerta cerrada y el segundo para puerta abierta).

El campo "posicion" se utiliza para denotar el sensor del cual se desea obtener la información.

La respuesta de los dispositivos son únicamente 2, siendo una de ellas para responder a las peticiones de información de un sensor donde el valor de la respuesta es el valor resultado(un número o un boolean).

La segunda respuesta es a la petición de exponer los servicios. Esta ultima puede tener 3 formas diferentes dependiendo la petición realizada al dispositivo. La primera forma es en respuesta a la petición con el valor de sección "todo" la cual si bien está implementado no se utiliza, y tendrá la siguiente estructura:

{

"suscripto": "nombre del dispositivo",

"tipos": [

{

"nombre": "luz"

},

{

"nombre": "sensor"

}

]

}

La segunda forma es en respuesta a la petición con el valor de sección "luz" y tendrá la siguiente estructura:

{

"luz": [

{

"nombre": "nombre que identifica la luz",

"topico": "valor numérico correspondiente a la luz"

},

]

}

La tercer forma es en respuesta a la petición con el valor de sección "sensor" y tendrá la siguiente estructura:

{

"sensor": [

{

"nombre": "nombre del sensor",

"topico": "tópico destino",

"tipo": "tipo de datos que se obtiene del sensor"

}

]

}

La información obtenida de las respuesta a la petición de exponer servicios se utiliza para saber cómo interactuar con el dispositivo.

# ***Problemas resueltos***

Durante el desarrollo temprano de la aplicación es donde tuve más problemas relacionados a la forma de uso de las distintas herramientas que forman la aplicación.

La forma de comunicación entre las distintas partes de la aplicación y los distintos dispositivos fue un problema bastante complicado de resolver, ya que al no tener una misma herramienta para comunicar todas las partes que forman la aplicación hizo que sea necesario un punto de unión para llevar a cabo la comunicación. Esta unión la lleva a cabo el Middleware.

Por último, no lo considero un problema, pero todas las herramientas utilizadas en la aplicación fueron completamente nuevas para mí. Esto hizo que sea un poco más complejo desarrollar, haciendo que quizás existan cosas que se hubieran podido desarrollar en menos tiempo o de manera más sencilla.

# ***Conclusiones***

Luego de haber estado trabajando en este proyecto durante los últimos meses, hice una presentación interna en la empresa para verificar si se habían alcanzado los objetivos propuestos. Tuve una devolución satisfactoria en donde expresaban su voluntad para utilizar este proyecto como base para próximos productos.

Este proyecto sirvió para mostrar un prototipo usable e innovador, movilizando pocos recursos. Igualmente mencionaron detalles para mejorar la usabilidad de la interfaz, de todas formas desde el desarrollo se hizo hincapié en mostrar un prototipo funcional de la aplicación.

Finalmente, este desarrollo está abierto a sugerencias que aporten a mejorarlo, al ser un proyecto de código abierto, cualquiera puede colaborar con código que resuelva errores o agregue nuevas funcionalidades. Más información en: https://github.com/MartinMelo/DomiticTIP

# ***Anexos***

## Anexo Hardware

### Arduino

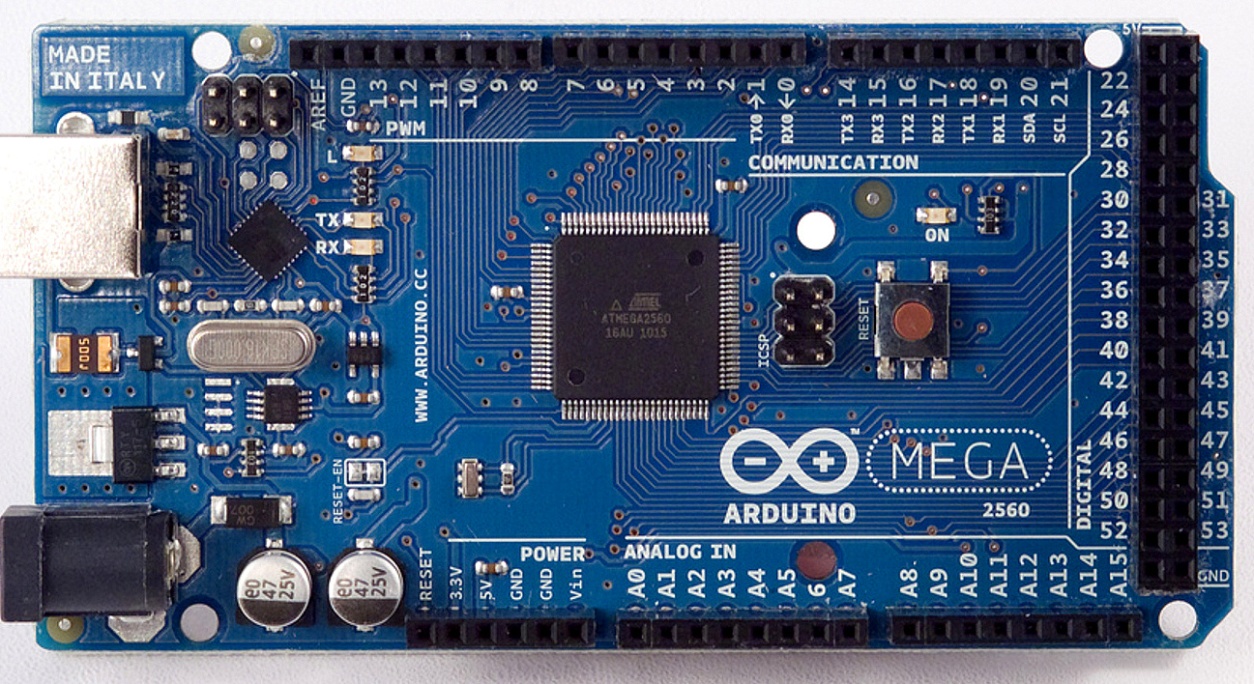
#### Como nace

La creación de Arduino se inicio en el año 2005 y fue un proyecto para los estudiantes de un instituto en Ivrea Italia. Este proyecto surgió de la necesidad de contar con un dispositivo de bajo costo y que se pueda utilizar con cualquier sistema operativo.

#### ¿Qué es?

*Arduino* es una plataforma de hardware libre, basada en una placa con un micro controlador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un micro controlador [Atmel AVR](http://es.wikipedia.org/wiki/AVR). La siguiente imagen pertenece una placa Arduino Mega2650.



Actualmente existen muchos entornos de desarrollos para crear programas para Arduino, los cuales se encuentran programados en diferentes lenguajes como C, Java, y Python.

La sintaxis del lenguaje de programación Arduino es una versión simplificada de C/C++.

#### ¿Qué resuelve?

Se utilizó para crear un dispositivo capaz de poder comunicarse en ambos sentidos con la aplicación que utiliza el usuario y que a su vez se pueda utilizar para efectuar distintas acciones basándose en la información que se le suministre a través de un protocolo específico. A su vez, el mismo está conectado a distintos circuitos electrónicos que le proveen efectuar las acciones anteriormente mencionadas y también la obtención de información de los sensores que posea.

### Diagramas del Hardware





### Sensor de temperatura

El modelo del sensor es LM35DZ y el fabricante es Texas Instruments. Posee un rango de medición de -55° a 150° Celsius. El mismo puede ser reemplazado por otro sensor de similares características. Su hoja de datos es la siguiente:

<http://www.ti.com.cn/cn/lit/ds/symlink/lm35.pdf>

## Anexo Software

### M2M

M2M (Machine to Machine o Máquina a Máquina) es un concepto genérico que se refiere al intercambio de información o comunicación en formato de datos entre dos máquinas remotas.

### MQTT

[Message Queue Telemetry Transport (MQTT)](http://mqtt.org/) es un protocolo de conectividad enfocado a M2M (machine-to-machine) y al IOT (Internet of Things) ya que se ha diseñado para ser un protocolo de mensajería extremadamente ligero basado en TCP.

Es útil para conexiones con sitios remotos donde se posee una alta latencia y el ancho de banda es muy importante.

Una característica muy importante es que al ser un protocolo tan ligero existen clientes y servidores MQTT en diversos lenguajes.

### Mosca

Es un bróker de mensajería, con lo cual es un mecanismo mediador de la comunicación entre aplicaciones y/o dispositivos, permitiendo minimizar el grado de conocimiento mutuo que estos necesitan tener, para poder intercambiar mensajes, implementando así efectivamente su [desacoplamiento](http://es.wikipedia.org/wiki/Dise%C3%B1o_estructurado#Acoplamiento).

El propósito del bróker es recibir los mensajes entrantes desde las aplicaciones y llevar a cabo determinadas acciones con ellas.

### Socket.IO

[*Socket.io*](http://socket.io/) es una librería que permite manejar eventos en tiempo real mediante una conexión TCP y todo ello en Javascript. Es realmente potente y permite hacer todo tipo de aplicaciones en tiempo real, como por ejemplos servicios de mensajería/chats. Ya que permite la comunicación bidireccional entre clientes web y servidor. Esta librería está compuesta de 2 partes, la primera de ellas es la que corre en el navegador del cliente y la otra de ellas es la que corre en el servidor de la aplicación. Ambas partes son casi idénticas y ambas se basan en el manejo de eventos.

*Socket.io* utiliza principalmente el protocolo de Websockets, aunque también permite utilizar polling manteniendo la misma interfaz. A su vez también contiene características extra entre las cuales se incluye la posibilidad de realizar broadcast a múltiples sockets, almacenar información asociada a cada cliente y procesar entrada/salida de forma asincrónica.

### ¿Qué es JSON?

JSON (Javascript Object Notation - Notación de Objetos de Javascript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del [Lenguaje de Programación Javascript](http://javascript.crockford.com/), [Standard ECMA-262 3rd Edition - Diciembre 1999](http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf).

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, Javascript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

* Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
* Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un*objeto* es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma).

Un*arreglo* es una colección de valores. Un arreglo comienza con [ (corchete izquierdo) y termina con ] (corchete derecho). Los valores se separan por , (coma).

Un *valor* puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o true o false o null, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse.

Una *cadena de caracteres* es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una *cadena de caracteres* es parecida a una cadena de caracteres C o Java.

Un *número* es similar a un número en C o Java, excepto que no se usan los formatos octales y hexadecimales.

Los espacios en blanco pueden insertarse entre cualquier par de símbolos.

Un pequeño ejemplo de json es el siguiente:

{

"nombre": "cofre simple",

"descripcion": "soy un cofre simple de madera",

"cantidad": 3,

"elementos": [

{

"nombre": "Espada de diamantes",

"id": 276

},

{

"nombre": "Disco Strad",

"id": 2264

},

{

"nombre": "mesa de encantamiento",

"id": 116

}

]

}

### MongoDB

Es un sistema de [base de datos](http://es.wikipedia.org/wiki/Base_de_datos) [NoSQL](http://es.wikipedia.org/wiki/NoSQL) orientado a documentos, desarrollado bajo el concepto de [código abierto](http://es.wikipedia.org/wiki/C%C3%B3digo_abierto).

En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo [JSON](http://es.wikipedia.org/wiki/JSON) con un esquema dinámico (formato [BSON](http://es.wikipedia.org/wiki/BSON)), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

### Express

Es un Framework para NodeJS diseñado para construir tanto sitios web simples como complejos y a su vez aplicaciones web hibridas.

### Node-Schedule

Es un planificador de tareas flexibles tanto cron-like como non-cron-like para NodeJS. El cual permite programar tareas para realizar funciones arbitrarias en fechas específicas, con reglas de recurrencia opcionales. Utiliza un solo timer en un momento dado (en lugar de reevaluar próximos trabajos cada segundo/minuto).El mismo es MIT-License.

### Malhar-angular

Librería que provee el uso de componentes y directivas para utilizar Dashboard con Widgets en el Front-End.

### NodeJS

Es un entorno de programación para el Back-End basado en el lenguaje de programación [ECMAScript](http://es.wikipedia.org/wiki/ECMAScript), asincrónico, con [Entrada/Salida](http://es.wikipedia.org/wiki/I/O) de datos en una [arquitectura orientada a eventos](http://es.wikipedia.org/wiki/Programaci%C3%B3n_dirigida_por_eventos) y basado en el motor [V8](http://es.wikipedia.org/wiki/V8_(motor_JavaScript)) de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, [servidores web](http://es.wikipedia.org/wiki/Servidor_web).

### MeanJS

Es una solución Javascript full stack que ayuda a crear aplicaciones web de producción rápidas, robustas y fáciles de mantener usando MongoDB, Express, AngularJS y Node.js.

## Anexo Extra

### Licencia MIT

Esta licencia permite reutilizar el software así licenciado tanto para ser [software libre](http://es.wikipedia.org/wiki/Software_libre) como para ser [software no libre](http://es.wikipedia.org/wiki/Software_no_libre), permitiendo no liberar los cambios realizados al programa original.

También permite licenciar dichos cambios con [licencia BSD](http://es.wikipedia.org/wiki/Licencia_BSD), [GPL](http://es.wikipedia.org/wiki/Licencia_p%C3%BAblica_general_de_GNU), u otra cualquiera que sea compatible (es decir, que cumpla las cláusulas de distribución).

Con esta licencia se tiene software libre. Ejemplos en los que podría interesar su aplicación serían las [licencias duales](http://es.wikipedia.org/wiki/Licencia_dual), si se pretende difundir un [estándar](http://es.wikipedia.org/wiki/Norma_(tecnolog%C3%ADa)) mediante una implementación de referencia, o si simplemente se pretende que el producto sea Libre sin mayores consideraciones.

### Internet Of Things



El Internet de las cosas o Internet Of Things es un término que se relacionan mucho con la web. Pero también resulta un concepto un poco abstracto para los no iniciados pese a que ha estado ganando bastante popularidad en el último tiempo. La idea que intenta representar queda bastante bien ilustrada por su nombre, cosas cotidianas que se conectan a Internet, pero que en realidad se trata de mucho más que eso.  
  
Para entender de qué va el internet de las cosas debemos también comprender que sus fundamentos no son en lo absoluto nuevos. Desde hace unos 30 años que se viene trabajando con la idea de hacer un poco más interactivos todos los objetos de uso diario. Planteos como el hogar inteligente o casa del mañana, han evolucionado antes de que nos demos cuenta en el hogar conectado para entrar al internet de las cosas.  
  
El Internet de las cosas potencia objetos que antiguamente se conectaban mediante circuito cerrado, como comunicadores, cámaras, sensores, y demás, y les permite comunicarse globalmente mediante el uso de la red de redes.  
  
Si tuviéramos que dar una definición del internet de las cosas probablemente lo mejor sería decir que se trata de una red que interconecta objetos físicos valiéndose de la red. Los mentados objetos se valen de sistemas embebidos, o lo que es lo mismo, hardware especializado que le permite no solo la conectividad a internet, sino que además programa eventos específicos en función de las tareas que le sean dictadas remotamente (por ejemplo desde nuestro celular mientras trabajamos en la oficina).

### Automatización y control de los espacios físicos

Es también conocida como Domotica, la cual es un conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

# ***Referencias***

[1]Paulo Novais, Davy Preuveneers, Juan Manuel Corchado Rodríguez; Ambient Intelligence - Software and Applications: 2nd International Symposium on Ambient Intelligence, Springer, 2011.

[2] Rubén Saavedra Silveira, Automatización de viviendas y edificios, Ediciones CEAC, 2009.

[3] James Gerhart, Home Automation and Wiring, McGraw Hill Professional, 1999.

[4] Jamari Harmon, The Home Automation, Bookpubber, 2014.

[5] David Flanagan, JavaScript: The Definitive Guide, O'Reilly Media, Inc., 2011.

[6] Gary Shelly, Denise Woods, HTML: Introductory Concepts and Techniques, Cengage Learning, 2008.

[7] Brian W. Kernighan, Dennis M. Ritchie, El Lenguaje de Progra C, Pearson Educación, 1991.

## Referencias Adicionales

http://www.wisegeek.com/what-is-domotics.htm

https://angularjs.org/

http://jquery.com/

http://www.json.org/json-es.html

http://nodejs.org/

https://github.com/bblanchon/ArduinoJson

http://knolleary.net/arduino-client-for-mqtt/

1. Todos estos componentes son de bajo costo y pueden ser adquiridos en cualquier local donde vendan componentes electrónicos [↑](#footnote-ref-2)
2. Es un tablero con orificios conectados eléctricamente entre sí, en el cual se pueden insertar componentes electrónicos y cables para el armado y prototipado de circuitos electrónicos y sistemas similares. [↑](#footnote-ref-3)