# Capítulo 8 - Análisis y selección de tecnologías para desarrollo del SAR

En capítulos anteriores analizamos diversas tecnologías hardware y software relacionadas con la robótica. Dentro de las tecnologías hardware investigamos las plataformas Arduino (Capítulo 3 – Arduino) y Raspberry Pi (Capítulo 4 – Raspberry Pi). De las diversas herramientas en tecnologías software para aplicaciones móviles, investigamos aquellas que permiten desarrollar una aplicación que interactúe con el hardware mencionado.

En este capítulo examinamos dichas tecnologías para concluir cuales son las que integran el SAR. Para ello, realizamos diversas comparativas sobre características, ventajas y desventajas de cada una de las plataformas. Además, se presentan las problemáticas surgidas al relacionar los componentes.

## **8.1 Primer análisis**

El análisis realizado de las distintas tecnologías de hardware y software, basándose en factores como rendimientos, tiempos de respuesta, consumo energético, portabilidad y compatibilidad, nos permitió seleccionar los elementos que componen el SAR.

El hardware estudiado y utilizado a lo largo del desarrollo, fue el de la familia Arduino y Raspberry Pi.

Dentro de la plataforma Arduino se probaron las versiones Arduino Uno, Arduino Mega y Arduino Nano, siendo estos dos últimos los utilizados en el SAR. Además, se experimentaron con variados módulos, sensores y actuadores compatibles con esta familia como la cámara OV7670, ESP8266, Bluetooth, DHT11, entre otros. Los ensayos realizados con estos componentes se encuentran en el anexo de casos pruebas de módulos, sensores y actuadores.

En el caso de Raspberry Pi utilizamos la versión Pi 3 modelo B, en conjunto con la cámara compatible para esta SBC.

El software investigado para el desarrollo de aplicaciones móviles fue mencionado en los capítulos 5 y 6 (Capítulo 5 - Aplicaciones Móviles y Capítulo 6 – Stack MEAN). La idea era encontrar compatibilidad entre los elementos del hardware, utilizando el software como interfaz entre ellos.

## **8.2 Selección tecnologías hardware**

### 8.2.1 ¿Por qué Arduino?

Como se abordó en el capítulo 3 (Capítulo 3 – Arduino), siendo una arquitectura hardware pensada para hobbistas, diseñadores y personas no relacionadas con la electrónica ni la programación a bajo nivel, Arduino permite una curva de aprendizaje relativamente plana más corta y la facilidad de conexión de los distintos componentes la hace muy atractiva, para encarar distintos proyectos con diversos niveles de complejidad.

Las placas Arduino, fueron pensadas mayormente para el uso del control de sensores y actuadores utilizando un microcontrolador, esto genera una mejor transición desde la electrónica discreta a la electrónica programable.

Dentro de la plataforma Arduino, se seleccionó Arduino UNO, donde se elaboraron distintos prototipos simples con *protoboard*, desde la manipulación de actuadores con motores hasta la toma de datos de distintos sensores como temperatura, humedad, obstáculos, entre otros. Se encontró en la placa Arduino UNO una baja disponibilidad de pines E/S para la cantidad de sensores/actuadores y módulos que se requerían conectar. Por tal motivo, se decidió ampliar la cantidad de pines optando por la placa Arduino Mega. Esta última, otorga mayor cantidad de pines, sin expandir la cantidad de memoria ni procesamiento. Este detalle devino en la dificultad a la hora de la programación, por contar con pocas interrupciones hardware, forzando a la utilización de consulta periódica o *pooling* en el bucle principal (loop).  Estos problemas surgieron a la hora de conectar el módulo de la cámara OV7670 y el módulo WiFi ESP8266, los cuales requerían una alta cantidad de pines y nivel de cómputo.

Teniendo en cuenta el párrafo anterior, es que se opto del traspaso de una plataforma que trabaja con un microcontrolador a un computador, resultando ser la Raspberry Pi

### 8.2.2 ¿Por qué Raspberry?

En el capítulo 4 (Capítulo 4 – Raspberry Pi) se mencionó y analizó el computador de placa reducida (SBC) Raspberry Pi, en el cual se pudo apreciar las características fundamentales del mismo. Esta plataforma se diseñó, primordialmente, con fines didácticos para la educación secundaria, es por esto, que su costo es relativamente bajo. Al contar, la misma, con todas las capacidades básicas de una computadora portátil de hoy en día con su respectivo microprocesador (bastante potente para nuestras necesidades), memorias y puertos físicos (como el USB, HDMI, microSD, entre otros); y la posibilidad de instalar un sistema operativo totalmente funcional y con interfaz gráfica (en este caso Raspbian), es que se seleccionó como centro de administración y control del SAR.

Además, cuenta con pines GPIO para las conexión y manipulación de distintos módulos (como actuadores y sensores), aunque como se analiza en el apartado siguiente, se delegó en placas Arduino Mega y Arduino Nano las funcionalidades de control y sensado, exceptuando la conexión y procesamiento de imágenes, delegadas a la cámara de Raspberry Pi v2 y las comunicaciones inalámbricas proporcionadas por los módulos wifi y bluetooth integrados a este computador.

### 8.2.3 Comparativa entre Arduino Mega, Arduino Nano y Raspberry Pi 3 Model b

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Arduino Mega | Arduino Nano | Raspberry Pi3 Model B |
| Microcontrolador/  Microprocesador | ATmega 1280 - 16Mhz 8bits | ATmega328 – 16Mhz 8bits | Quad Core 1.2GHz Broadcom BCM2837 64bit |
| Tensión | 5v | 5v | 5v |
| Memoria | 128 KB (Bootloader 4KB) | 32 KB (Bootloader 2KB) | 1 GB |
| Digital I/O | 54, 15 PWM | 22, 6 PWM | 40 GPIO |
| Analog I/O | 16 | 8 |
| Interfaces | USB x 1(energía) | USB x 1 (energia) | USB x 4, HDMI, CSI, DSI, MicroSD, WLAN y BLE, microUSB (Energía) |

Dada la comparativa entre las tecnologías, se decide utilizar las placas Arduinos para el control de módulos, sensores y actuadores. La Raspberry Pi quedó seleccionada para la captura de imágenes y como servidor dedicado. Tanto la Arduino Nano como la Mega se conectan a la Raspberry a través de sus interfaces USB.

¿Cuáles son los beneficios de esta combinación? Se podría utilizar únicamente Raspberry para la elaboración del SAR, pero existen numerosos beneficios que proporciona las placas Arduino frente a la Raspberry y son:

* Menor costo de adquisición del producto en caso de fallos energéticos.
* Mayor flexibilidad y facilidad en la conexión con distintos componentes electrónicos.
* Mayor compatibilidad, con los módulos arduino-compatible y la familia adafruit
* Buen tiempo de respuesta de I/O.
* Alta confiabilidad en la lectura de sensores y en los valores de manipulación de actuadores.

Varios de estos beneficios se deben a que Arduino no posee un sistema operativo, sino un único programa que se ejecuta indefinidamente (LOOP) sin necesidad de correr algún software auxiliar que lo dispare o ejecutando como servicio; logrando concentrar su poder de procesamiento en el único programa definido.  La ejecución de servicio genera retrasos propios de sistemas operativos.

### 8.2.4 Cámara V2 de Raspberry Pi

La cámara V2 de Raspberry, es una cámara exclusiva de esta plataforma la cual se conecta al puerto CSI de cualquier modelo de este computador (desde la Raspberry Pi 1 hasta el modelo actual, ósea, la Raspberry Pi 3), lo cual permite obviar la conexión pin a pin y abstraernos de la comunicación y procesamiento de la cámara. Como se comentó en el capítulo 4, es una cámara de alta definición de 8 megapíxeles, suficiente para el objetivo que se pretende con el desarrollo del SAR. Esto soluciono las problemáticas que se nos presentaron a la hora de probar la cámara OV7670 con Arduino; como el poder de procesamiento de imágenes y transmisión de las mismas (inalámbricamente) hacia otro dispositivo tal como una PC o un dispositivo móvil (en nuestro caso smartphones).

### 8.2.5 Módulos de Arduino

Dentro de los módulos, sensores y actuadores de Arduino que se probaron y/o se utilizan, se encuentran:

*Utilizados en el sar:*

* El módulo GPS, será utilizado para determinar la ubicación geográfica del SAR (Geolocalización)
* Sensor de temperatura KY-001(-55° a +125°)
* Sensor ultrasonido HC-SR04 para determinar presencia de objetos a determinadas distancia y tratar de evitar el impacto con los mismos
* Motores CC para la movilidad del SAR dentro del ambiente

*Ensayados y no seleccionados:*

* El módulo wifi ESP8266 y el módulo Bluetooth HC-05, no se utilizarán debido a que la Raspberry Pi3 Model B, brinda su funcionalidad.
* El módulo Acelerómetro MMA7361.
* Servomotor sg90.
* Sensor de evasión de obstáculos KY032.
* Sensor de golpe KY-031.
* Sensor de llamas KY-026.

## **8.3 Selección tecnologías software**

### La selección del software, necesario para el desarrollo del SAR, se basa en los siguientes requerimientos:

### Generar un mayor nivel de abstracción, mediante librerías basadas en JavaScript, para la comunicación con el hardware.

### Utilizar un Sistema Operativo de base (en nuestro caso Raspbian), en vez de una rutina corriendo en un microcontrolador.

### Tener los recursos necesarios para desplegar un servidor.

### Contar con la posibilidad de comunicar las plataformas Arduino al servidor mediante un protocolo estándar.

### Aprovechar las herramientas de Raspbian para realizar la comunicación y captura de imágenes.

### Necesitar el desarrollo de una aplicación móvil para el control inalámbrico del SAR.

### Almacenar datos para la generación de estadísticas.

### Permitir el acceso a más de un cliente a los datos alojados en él SAR.

### Para cumplir con los requisitos mencionados anteriormente se escogieron las siguientes de tecnologías.

### Como se mencionó en un apartado anterior (8.2.1 ¿Por qué Arduino?) en cuanto a las dificultades que surgieron al tratar de utilizar la cámara OV7670 con el Arduino Mega, es que se decidió adquirir la Raspberry Pi 3 modelo B. Esta plataforma cuenta con una cámara (mencionada en el apartado 4.6 Accesorios para Raspberry Pi) como accesorio, la cual permite la captura de imágenes mediante aplicativos compatibles con Debian, como consecuencia también compatible con Raspbian.

Se opto por la instalación de Raspbian en la Raspberry, porque es el sistema operativo oficialmente soportado por la fundación[[1]](#endnote-1). La placa Arduino se encuentra pensada para desarrollos donde existe un único proceso de ejecución principal. Esto genera limitaciones en cuanto a memoria, procesamiento y almacenamiento como, por ejemplo:

* Para poder almacenar gran cantidad de datos es necesario contar con un módulo para memorias SD. Este módulo funciona con comunicación serial.
* El almacenamiento disponible no permite el uso de bases de datos.
* Posee una cantidad limitada de interrupciones por hardware (2 en Arduino Uno y Nano, 6 en el caso de Arduino Mega). No cuenta con la posibilidad de interrupciones por software.
* La detección de nuevo valores en sensores se realiza por pooling. Esto significa, consultar el estado de cada uno de los sensores.
* No es suficiente el tiempo de transmisión de imágenes dado el nivel de procesamiento para almacenar bytes en un buffer y retransmitirlos. Este fue el caso de la captura de valores de la cámara y su retransmisión via serie (cable). Además, de no alcanzar los FPS (frame per seconds) necesarios para una visualización fluida (al menos 10 FPS) ver anexo, se nos dificultó lograr el objetivo de la transmisión de las imágenes por medios inalámbricos (Bluetooth y Wifi).

Por otro lado, la Raspberry al ser un computador que permite la instalación de un sistema operativo, facilitó resolver varias de las dificultades, antes mencionadas, que sucedieron con los microcontroladores.

En los repositorios de Raspbian encontramos una aplicación denominada Motion. La cual surgió para satisfacer la videovigilancia a través de cámaras web. En nuestro caso, nos permitió la captura de imágenes con la cámara de Raspberry abstrayéndonos de la comunicación entre la aplicación y el hardware.

Dadas las capacidades inalámbricas que posee la Raspberry es que se pudo configurar en modo ad-hoc. Esto quiere decir, crear una red inalámbrica wifi (con una ssid y contraseña) sin depender de ningún punto de acceso (como un router, punto de acceso, etc.) y permitiendo la conexión de diversos hosts, donde cada uno obtiene su respectiva ip por medio de dhcp.

Al comenzar el desarrollo de esta tesina, y teniendo en cuenta que íbamos a trabajar únicamente con la familia Arduino, decidimos diseñar una aplicación móvil nativa. Dado que la única comunicación que existía entre un posible cliente y el SAR era por datos *raw* enviados por bluetooth o wifi.

Para diseñar esta app se investigaron distintas herramientas (explicadas en el Capítulo 5 - Aplicaciones Móviles), sin embargo, al cambiar las prestaciones hardware y tener un sistema operativo, se decidió cambiar la arquitectura del software del SAR.

Esta nueva arquitectura genero un cambio en la aplicación, o sea, se pasó del desarrollo de una app nativa, para Android, a una app web. Esto permitió crear una única aplicación que puede ser consumida por distintos dispositivos que accedan a la red LAN[[2]](#footnote-1) del SAR.

Para producir la app web se necesitó de un grupo de tecnologías que satisfagan los siguientes puntos:

* Contar con la posibilidad de almacenar datos de los sensores y acciones realizadas mediante una base de datos.
* Tener una interfaz de fácil comunicación con el servidor.
* Tener la capacidad de desplegar a demanda la app desde una red LAN.
* Diseñar una app, utilizando herramientas de frontend, para el renderizado en el cliente.
* Permitir que esta app utilice las acciones http para la comunicación con el servidor.

Teniendo en cuenta los puntos anteriores examinamos distintas herramientas de desarrollo de aplicaciones web móviles (como se vieron en el Capítulo 5 - Aplicaciones Móviles). Estas tecnologías son: Cordova, IntelXDK, Meteor y MEAN.

Tanto Cordova como IntelXDK fueron descartadas dado que se prefirió un grupo de herramientas, compatibles entre ella, y estables (stack de desarrollo de software).

Por otro lado, se trato de incursionar en Meteor, realizando aplicaciones sobre arquitecturas Intel x86/x64. Se diseño un prototipo funcional de la aplicación, pero al migrar la misma a la arquitectura ARM (en nuestro caso la de Raspberry Pi) tuvimos inconvenientes dado que este framework no se encontraba soportado oficialmente para esta arquitectura. Existía un fork, pero no tuvimos éxito en la integración de las tecnologías, que integran Meteor, a pesar de lograr compilarlo.

Meteor se encuentra conformado por el siguiente conjunto de herramientas: Mongo, Blaze, Meteor, Iron (explicadas en **¡Error! No se encuentra el origen de la referencia.**). Al presentarse la dificultad, antes mencionada, es que seleccionamos el stack MEAN el cual resultó ser compatible con el desarrollo avanzado hasta el momento, hecho con Meteor.

El stack MEAN (visto Capítulo 6 – Stack MEAN) compuesto por Mongo, Express, Angular y Node. La migración de la aplicación tanto *frontend* como *backend*, desarrollada con Meteor, fue dispuesta de la siguiente forma: El procesamiento de templates, captura y gestión de eventos, realizada en Blaze, se trasladó a Angular 4+. El servidor Meteor se codificó en Node. El manejo de rutas y REST desarrollado en Iron se migró a Express. En cuanto a las colecciones de datos se mantuvieron en Mongo.

Otro desafío que se presentó, era comunicar el servidor (realizado con Node) corriendo en Raspbian (dentro de la Raspberry) con las placas Arduino Mega y Arduino Nano. Avanzando sobre NPM, se encontraron dos librerías estables para la comunicación de Node y Arduino. Estas librerías son Cylon[[3]](#footnote-2) y Johnny-five. La librería Cylon utiliza el paradigma de programación declarativo, en cambio, Johnny-five el funcional. Este último fue el seleccionado por mantener el mismo estilo de codificación que el stack MEAN, compatibilidad con los componentes de Arduino y poseer una versión estable de serialport compatible con la arquitectura ARM.

Para poder realizar la comunicación entre Node y las placas Arduino, Johnny-five se vale de un protocolo de comunicación estándar (

Introducción

En éste capítulo, se analizará el stack MEAN y sus componentes. El mismo está compuesto por un conjunto de tecnologías respetando el acrónimo como sigue: Mongo, Express, Angular y Node. Además, otros complementos, como Compodoc (documentador), bibliotecas y framework aplicados a la vista como Bootstrap y JQuery.

## **6.1 ¿Qué es MEAN?**

“Se denomina MEAN, o MEAN *stack*, a un conjunto de capas de software para el desarrollo de aplicaciones, dónde la característica predominante es el uso del lenguaje de programación popularizado como JavaScript”.



Ilustración 31 - Acrónimo MEAN

## **6.2 Componentes de MEAN**

### 6.2.1 MongoDB

Es un sistema de base de datos NoSQL, es decir, que almacena los datos en estructuras o “documentos”, los cuales están estructurados como objetos bajo la notación JSON (Notación simple de objeto tipo JavaScript), en vez de filas y agrupdos en colecciones en contraposición a las tablas de un RDBMS. Debido a la ausencia de comprobación de integridad referencial tine un alto desempeño .

### 6.2.2 Express

Es un paquete de NodeJS que ofrece una interface mínima para manejo de solicitudes o peticiones HTTP. Ofrece también un sistema simple de enrutamiento (Routing). Dentro del MEAN *stack* opera del lado del servidor, también conicido como *back-end*.

### 6.2.3 Angular

Es un framework orientado a crear aplicaciones web, basado en el sub-lenguaje TypeScript (JavaScript con verificación de tipos de dato *ahead of time*), mantenido por Google, enfocado en aplicaciones web de una sola página o SPA. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), con el objetivo de que el desarrollo y las pruebas sean más sencillas.

### 6.2.4 Node

En MEAN, Node, es la plataforma encargada del funcionamiento del servidor. Se trata de un intérprete de JavaScript enfocado en la programación del lado del servior.

Utiliza el motor de ejecución de JavaScript de Google, denominado V8, y presenta una arquitectura orientada a eventos, en conjunto con una serie de APIs no-bloqueantes (asíncronas) que le proporcionan un rendimiento y una escalabilidad muy elevadas. Esta característica se debe a una librería en C, multiplataforma, que proporciona soporte de E/S asíncronas basada en bucles de eventos, se encuentra diseñada específicamente para el uso en NodeJS y es llamada LibUV (Unicornio Velocirraptors).

Si bien Node se puede utilizar para crear cualquier tipo de aplicación, dado que incorpora un módulo para poder actuar como un servidor web dentro de su biblioteca standard, es especialmente popular para crear aplicaciones web.

Actualmente lo emplean para sus aplicaciones multitud de empresas de todos los ámbitos, pero especialmente de Internet. Desde aplicaciones web, aplicaciones en línea de comandos, scripts para administración de sistemas, aplicaciones de red, etc. Es aconsejable su utilización en aplicaciones como: chats, APIs REST, entrada de datos concurrentes y en espera (escritura en DB bloqueante), procesamiento parcial de archivos, transmisión de datos, proxy, aplicaciones como corredores de bolsa (tiempo real), visualización de interacciones, etc.

Node es utilizado en la construcción rápida y escalable de aplicaciones de red, debido a que es capaz de manejar un gran número de conexiones simultáneas con alto rendimiento, lo que equivale a una alta escalabilidad. La idea principal de Node.js es el uso no-bloqueante, event-driven I/O, permanecer ligero y eficiente en la superficie del uso intensivo de datos en tiempo real de las aplicaciones que se ejecutan en dispositivos distribuidos (Como podemos apreciar en la comparativa entre los servidores tradicionales y Node Ilustración 40 Comparativa de servidores tradicionales y Nodejs).

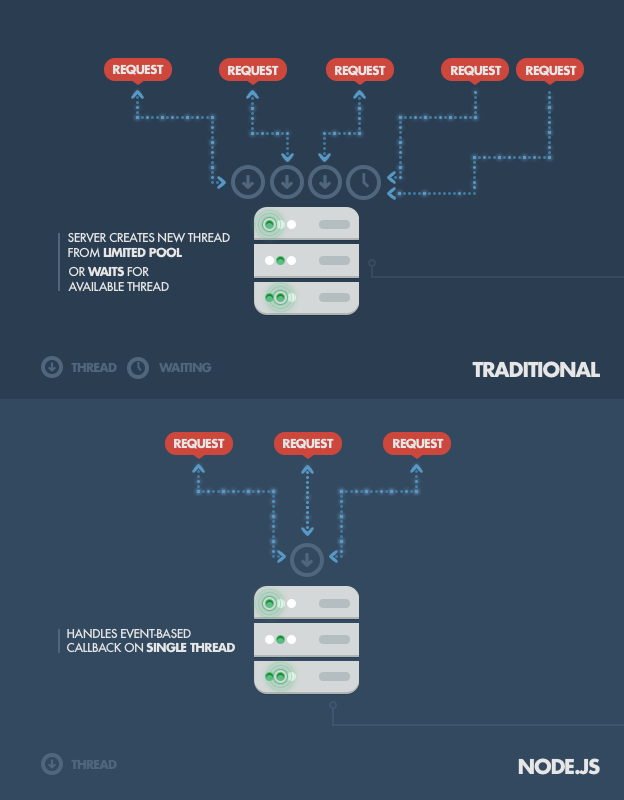


Ilustración 40 Comparativa de servidores tradicionales y Nodejs

## **6.3 Otros complementos**

### 6.3.1 Twitter Bootstrap

Es uno de los frameworks más populares que integra HTML, CSS, y JS para el desarrollo de proyectos adaptables a diferentes medios de reproducción (Responsive). Falta

### 6.3.2 Compodoc

Documentador JavaScript estático.

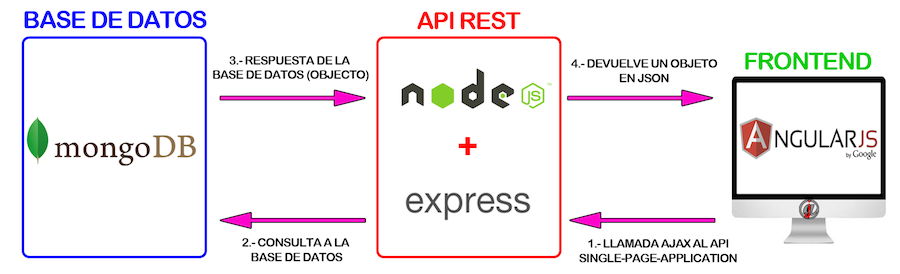


Ilustración 33 - Arquitectura de interacción MEAN

### 6.3.3 JSON

Es el acrónimo de JavaScript Simple Object Notation. Se trata de un mecanismo de *marshaling*, que permite transmitir en formato de cadenas de texto objetos (o estructuras complejas) que pueden ser luego des-*marshalizadas* para recuperar los objetos origniales.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función JSON.parse(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

Podemos decir que en MEAN, JSON es el formato de transferencia de datos entre todas las capas: navegador, servidor web y servidor de datos”

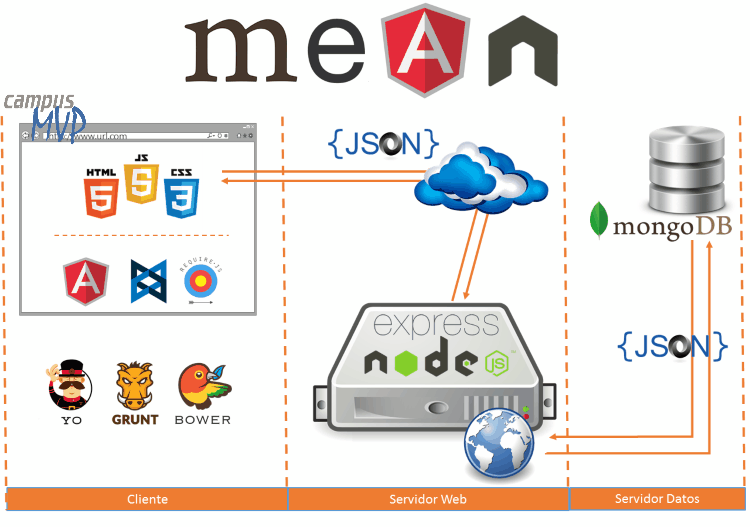


Ilustración 35 - Json pegamento de tecnologías

Capítulo 7 – Librería Johnny-five y el protocolo Firmata) denominado Firmata.

Finalmente, se definió una secuencia de eventos, que permiten capturar los valores sensados para poder insertarlos en las colecciones de Mongo. Con estas colecciones se generan las estadísticas requeridas por los objetivos de esta tesina.

1. <https://www.raspberrypi.org/downloads/raspbian/> [↑](#endnote-ref-1)
2. LAN: Red de área local, red de computadoras que abarca un área reducida como una casa, un departamento o un edificio.

   <https://es.wikipedia.org/wiki/Red_de_%C3%A1rea_local> [↑](#footnote-ref-1)
3. [↑](#footnote-ref-2)