

**Universidad Nacional de la Patagonia San Juan Bosco**

**Facultad de Ingeniería**

**Tesina de grado:**

|  |
| --- |
| Desarrollo y construcción de un sistema autónomo robótico administrado por una aplicación web para exploración |

**Alumnos:**

**Mansilla Fernando Damián**

**Schlapp Agustín Pablo**

**Tutor**:

**Lic. Defossé Nahuel**

**Trelew**

**Año 2018**

Índice

[Capítulo 1 - Introducción 7](#_Toc504153874)

[**1.1 Objetivo general** 7](#_Toc504153875)

[1.1.1 Objetivos específicos 7](#_Toc504153876)

[1.1.2 Metodología 7](#_Toc504153877)

[**1.2 Motivación** 8](#_Toc504153878)

[**1.3 Desarrollos Propuestos** 9](#_Toc504153879)

[**1.4 Resultados Esperados** 9](#_Toc504153880)

[Capítulo 2 - La robótica 10](#_Toc504153881)

[**2.1 ¿Qué es la robótica?** 10](#_Toc504153882)

[**2.2 Estructura física de los robots** 12](#_Toc504153883)

[2.2.1 Poliarticulados 12](#_Toc504153884)

[2.2.2 Móviles 12](#_Toc504153885)

[2.2.3 Androides 13](#_Toc504153886)

[2.2.4 Zoomórficos 13](#_Toc504153887)

[2.2.5 Híbridos 13](#_Toc504153888)

[**2.3 Distintas tecnologías para la robótica educativa** 14](#_Toc504153889)

[**2.4 Microcontroladores y computadora de placa reducida (SBC)** 14](#_Toc504153890)

[**2.5. Comunicación entre distintas plataformas de cómputo** 16](#_Toc504153891)

[**2.6 ¿Qué es un SAR (Sistema Autónomo Robótico)?** 17](#_Toc504153892)

[**2.7 La robótica en la educación** 17](#_Toc504153893)

[Capítulo 3 – Arduino 19](#_Toc504153894)

[**3.1 ¿Qué es Arduino?** 19](#_Toc504153895)

[**3.2 ¿Qué es Processing?** 19](#_Toc504153896)

[**3.3** 21](#_Toc504153897)

[**Fritzing** 21](#_Toc504153898)

[**3.4 ¿Qué es Wiring?** 21](#_Toc504153899)

[**3.5 Entonces Arduino es…** 21](#_Toc504153900)

[**3.6 Plataforma Arduino** 22](#_Toc504153901)

[**3.7 Distintas plataformas para Arduino** 23](#_Toc504153902)

[**3.8 Diferencias entre distintas placas de la familia Arduino** 24](#_Toc504153903)

[**3.9 ¿Por qué usar Arduino?** 25](#_Toc504153904)

[3.9.1 La comunidad 25](#_Toc504153905)

[3.9.2 La sencillez del lenguaje de programación 26](#_Toc504153906)

[3.9.3 Es hardware de bajo costo 26](#_Toc504153907)

[**3.10 Incorporación de Arduino en las escuelas** 26](#_Toc504153908)

[3.10.1 Las tres erres 27](#_Toc504153909)

[**3.11 Actuadores y sensores** 27](#_Toc504153910)

[**3.12 Actuadores en el SAR** 28](#_Toc504153911)

[**3.13 Sensores en el SAR** 29](#_Toc504153912)

[**3.14 Módulos o *shields* en el SAR** 30](#_Toc504153913)

[Capítulo 4 – Raspberry Pi 31](#_Toc504153914)

[**4.1 ¿Qué es Raspberry Pi?** 31](#_Toc504153915)

[**4.2 Especificaciones técnicas de las distintas versiones** 31](#_Toc504153916)

[**4.3 Entrada/Salida de propósito general (GPIO)** 32](#_Toc504153917)

[**4.4 Sistemas Operativos compatibles** 33](#_Toc504153918)

[**4.5 ¿Qué es Python?** 34](#_Toc504153919)

[**4.5 Python con Raspberry Pi** 34](#_Toc504153920)

[**4.6 Accesorios para Raspberry Pi** 36](#_Toc504153921)

[**4.7 ¿Por qué elegir Raspberry Pi?** 37](#_Toc504153922)

[Capítulo 5 - Aplicaciones Móviles 39](#_Toc504153923)

[Capítulo 6 – Stack MEAN 51](#_Toc504153924)

[Capítulo 7 – Librería Johnny-five y el protocolo Firmata 57](#_Toc504153925)

**[7.1 ¿Qué es Johnny-five?](#_Toc504153926)** [57](#_Toc504153926)

**[7.2 Instalación](#_Toc504153927)** [57](#_Toc504153927)

**[7.3 Arduino Firmata](#_Toc504153928)** [58](#_Toc504153928)

**[7.4 Instalación Firmata](#_Toc504153929)** [59](#_Toc504153929)

[Capítulo 8 - Análisis y selección de tecnologías para desarrollo del SAR 61](#_Toc504153930)

**[8.1 Primer análisis](#_Toc504153931)** [61](#_Toc504153931)

**[8.2 Selección tecnologías hardware](#_Toc504153932)** [61](#_Toc504153932)

[8.2.1 ¿Por qué Arduino? 61](#_Toc504153933)

[8.2.2 ¿Por qué Raspberry? 62](#_Toc504153934)

[8.2.3 Comparativa entre Arduino Mega, Arduino Nano y Raspberry Pi 3 Model b 63](#_Toc504153935)

[8.2.4 Cámara V2 de Raspberry Pi 64](#_Toc504153936)

[8.2.5 Módulos de Arduino 64](#_Toc504153937)

**[8.3 Selección tecnologías software](#_Toc504153938)** [65](#_Toc504153938)

[La selección del software, necesario para el desarrollo del SAR, se basa en los siguientes requerimientos: 65](#_Toc504153939)

[ Generar un mayor nivel de abstracción, mediante librerías basadas en JavaScript, para la comunicación con el hardware. 65](#_Toc504153940)

[ Utilizar un Sistema Operativo de base (en nuestro caso Raspbian), en vez de una rutina corriendo en un microcontrolador. 65](#_Toc504153941)

[ Tener los recursos necesarios para desplegar un servidor. 65](#_Toc504153942)

[ Contar con la posibilidad de comunicar las plataformas Arduino al servidor mediante un protocolo estándar. 65](#_Toc504153943)

[ Aprovechar las herramientas de Raspbian para realizar la comunicación y captura de imágenes. 65](#_Toc504153944)

[ Necesitar el desarrollo de una aplicación móvil para el control inalámbrico del SAR. 65](#_Toc504153945)

[ Almacenar datos para la generación de estadísticas. 65](#_Toc504153946)

[ Permitir el acceso a más de un cliente a los datos alojados en él SAR. 65](#_Toc504153947)

[Para cumplir con los requisitos mencionados anteriormente se escogieron las siguientes de tecnologías. 65](#_Toc504153948)

[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. 65](#_Toc504153949)

[Capítulo 9 – Arquitectura y Ensamblado del SAR 69](#_Toc504153950)

**[9.1 Componentes](#_Toc504153951)** [69](#_Toc504153951)

[72](#_Toc504153952)

**[9.2 Estructura](#_Toc504153953)** [72](#_Toc504153953)

[9.2.1 Diseño 72](#_Toc504153954)

[9.2.2 Los 4 niveles 73](#_Toc504153955)

[Capítulo 10 – Desarrollo del SAR 77](#_Toc504153956)

[Esquema general del SAR (lógico /físico) 77](#_Toc504153957)

[Desarrollo de la aplicación (front-end) 77](#_Toc504153958)

[Desarrollo del servidor (back-end) 77](#_Toc504153959)

[Esquema de la arquitectura lógica 78](#_Toc504153960)

[Capturas de pantalla de la aplicación 79](#_Toc504153961)

[Funcionamiento de la App 80](#_Toc504153962)

[Como se construyen las estadísticas 80](#_Toc504153963)

[FALTA DETERMINAR LAS MUESTRAS 80](#_Toc504153964)

[Parámetros de configuración 80](#_Toc504153965)

**[FALTA FINALIZARLA](#_Toc504153966)** [80](#_Toc504153966)

[Ilustración 1 - Esquema básico de un robot 11](#_Toc504153967)

[Ilustración 2 - Ejemplo de robot poliarticulado 12](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153968)

[Ilustración 3 - Ejemplo de robot móvil 12](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153969)

[Ilustración 4 - Androide Asimo de Honda 13](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153970)

[Ilustración 5 - Robot Zoomórfico caminador 13](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153971)

[Ilustración 6 - Robot móvil-poliarticulado 13](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153972)

[Ilustración 7 - Arquitectura de un microcontrolador 15](#_Toc504153973)

[Ilustración 8 - Logo Arduino 19](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153974)

[Ilustración 9 - Logo de Processing 19](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153975)

[Ilustración 10 - Wiring IDE 20](#_Toc504153976)

[Ilustración 11 - C++ Blink ejemplo 20](#_Toc504153977)

[Ilustración 12 - Entorno Fritzing 21](#_Toc504153978)

[Ilustración 13 - Ejemplo serie 22](#_Toc504153979)

[Ilustración 14 - Niveles de entrada a la plataforma Arduino 23](#_Toc504153980)

[Ilustración 15 - Arduino Uno 24](#_Toc504153981)

[Ilustración 16 - Logotipo comunidad open-source de Arduino 26](#_Toc504153982)

[Ilustración 17- Representación actuadores y sensores 28](#_Toc504153983)

[Ilustración 18 - Actuadores y sensores compatibles con Arduino 29](#_Toc504153984)

[Ilustración 19- Representación de sensores 30](#_Toc504153985)

[Ilustración 20 - Logo oficial de Raspberry Pi 31](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153986)

[Ilustración 21 - Raspberry Pi 2 y sus GPIOs 32](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153987)

[Ilustración 22 - Interfaces de Raspberry Pi 33](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153988)

[Ilustración 23 - Logo de Python 34](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153989)

[Ilustración 24 - Menú de Raspbian 34](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153990)

[Ilustración 25 - Cámara Raspberry Pi V2 36](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153991)

[Ilustración 26 - Pantalla táctil de Raspberry Pi 36](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153992)

[Ilustración 27 - Adafruit Prototyping Pi 36](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153993)

[Ilustración 28 - Pidrive 37](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153994)

[Ilustración 29 - Pi TFT 37](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Capitulos\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx#_Toc504153995)

[Ilustración 44 - Página oficial de Johnny-Five 57](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504153996)

[Ilustración 45 - Remote Wiring de Windows 58](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504153997)

[Ilustración 46 - IDE de Arduino 59](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504153998)

[Ilustración 47 - Código StandardFirmata 59](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504153999)

[Ilustración 48 - Código ConfigurableFirmata 59](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154000)

[Ilustración 49 - Esquema de conexión de componentes 69](#_Toc504154001)

[Ilustración 50 - Raspberry Pi 3 69](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154002)

[Ilustración 51 - Arduino Mega 70](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154003)

[Ilustración 52 - Arduino Nano 70](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154004)

[Ilustración 53 - Motores CC 70](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154005)

[Ilustración 54 - Sensor de ultrasonido 70](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154006)

[Ilustración 55 - Porta pilas 70](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154007)

[Ilustración 56 - Módulo Puente H 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154008)

[Ilustración 57 - Mini-protoboard 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154009)

[Ilustración 58 - Sensor de Temperatura 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154010)

[Ilustración 59 - MQ7 CO 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154011)

[Ilustración 60 - GPS 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154012)

[Ilustración 61 - Cámara V2 71](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154013)

[Ilustración 62 - Panel Solar Power Bank 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154014)

[Ilustración 65 - Diseño estructura nivel 3 con SketchUp 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154015)

[Ilustración 66 - Diseño estructura nivel 4 con SketchUp 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154016)

[Ilustración 63 - Diseño estructura nivel 1 con SketchUp 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154017)

[Ilustración 64 - Diseño estructura nivel 2 con SketchUp 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154018)

[Ilustración 67 - Impresión 3D del nivel 1 72](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154019)

[Ilustración 68 - Nivel 2 descubierto 73](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154020)

[Ilustración 69 - RM Vista Lateral 73](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154021)

[Ilustración 70 - Esquema general del SAR 77](#_Toc504154022)

[Ilustración 71 - Logo PM2 77](file:///C:\\Users\\Agustin\\Documents\\Agustin\\Universidad\\Tesina\\Proyecto%20de%20tesina\\sar\\Capitulos\\Tesina%20Mansilla-Schlapp%20Correccion%20total%20Enero%202018.docx.docx" \l "_Toc504154023)

[Ilustración 72 - Arquitectura lógica del SAR 78](#_Toc504154024)

[Ilustración 73 - Aplicación Web 79](#_Toc504154025)

# Capítulo 1 - Introducción

## **1.1 Objetivo general**

Se pretende desarrollar un prototipo de un Sistema Autónomo Robótico (SAR), gestionado por un software definido como agente inteligente (que responda al modelo basado en objetivos[[1]](#footnote-1))para la exploración y análisis del medio ambiente.

### 1.1.1 Objetivos específicos

* Desarrollar una aplicación web multiplataforma que mediante comunicación inalámbrica permita el control del Robot móvil.
* Ensamblar un robot móvil integrando las plataformas Arduino y Raspberry Pi con diversos módulos y software.
* Investigar protocolos existentes y evaluar la necesidad de diseño de protocolos de comunicación para el control y procesamiento de datos entre el microcontrolador y la aplicación.
* Ensamblar físicamente e integrar a nivel de software los distintos componentes (sensores y actuadores) al SAR.
* Extender la aplicación para interactuar con la información que brinda el SAR de los sensores.

### 1.1.2 Metodología

El SAR será creado mediante las plataformas Arduino y Raspberry Pi. El robot poseerá motores como actuadores para desplazarse sobre la superficie a explorar y diversos sensores que permitan tomar muestras del ambiente explorado. Todos estos componentes se ensamblarán sobre distintas piezas estructurales para conformar el robot móvil o RM.

El RM estará en un estado receptivo, donde se le otorga el control a una aplicación web, la cual contará con una interfaz de usuario que facilitará la comunicación con el SAR. La aplicación permitirá manipular el desplazamiento del RM sobre la superficie y obtener las muestras del ambiente según se soliciten, en otras palabras, la lectura de los sensores.

La comunicación entre el SAR y la aplicación se realizará por medio de señales inalámbricas de radiofrecuencia. Se mantendrá una arquitectura de diseño denominada cliente/servidor, donde el cliente es el dispositivo que ejecuta la aplicación y el servidor es el SAR.

## **1.2 Motivación**

Las nuevas tendencias de hardware comomicrocontroladores, Smartphones y nuevos dispositivos programables, requieren contar con un nuevo esquema de diseño donde se puedan integrar las distintas tecnologías relacionadas (robótica, redes, plataformas móviles, etc.) en un área de conocimiento específica, para lograr una integración de saberes y disminuir la curva de aprendizaje de personas que se introducen en estas temáticas.

Para esto se necesita incursionar en el investigación y desarrollo en los ámbitos de la computación, control, mecánica y electrónica. Los cuales dieron paso a la robótica como técnica que combina diversas disciplinas, logrando un alto impacto en la sociedad en diversos ámbitos.

En la actualidad es muy popular la utilización de teléfonos móviles inteligentes (*smartphones*). De estos dispositivos, un segmento mayoritario se basa en el sistema operativo Android, presentado por Google en el 2007.

Android está basado en Linux y utiliza Java como lenguaje de desarrollo de aplicaciones. Por otro lado, Arduino, introducido en el año 2005, es una plataforma de hardware libre para electrónica orientado a la computación física (Phisical Computing).

Arduino aprovecha ciertas características de C++ para permitir el desarrollo de pequeños programas o sketches con conocimientos básicos de programación y electrónica. Esta simplicidad, sumado al bajo coste de las placas ha otorgado a la plataforma una gran popularidad.

Tanto Java como C++ han sido lenguajes utilizados en las actividades de laboratorio de varias cátedras de la Licenciatura por lo cual consiste en una motivación para llevar a cabo esta tesina.

Los nuevos avances en interoperabilidad de las distintas plataformas de las áreas de robótica y programación tanto en hardware como software, brindan un excelente recurso en materia de educación de nivel medio y superior permitiendo agilidad en el desarrollo de proyectos educativos con escaso conocimiento en dichas áreas. Es por ello que se necesita un estándar o prototipo de dónde partir, que se encuentre testeado con una biblioteca de funciones inmersas en el mismo y una arquitectura moldeable a distintas temáticas. Este prototipo base es el denominado SAR que se quiere desarrollar. En síntesis, el objetivo del SAR es crear un instrumento didáctico para la comprensión e incentivación de los alumnos en las distintas áreas mencionadas (robótica e informática).

## **1.3 Desarrollos Propuestos**

* Diseño y desarrollo del software necesario para el funcionamiento del SAR.
* Ensamblado de un prototipo hardware basado en Arduino y Raspberry Pi, integrado por distintos módulos compatibles con dichas plataformas.
* Diseño y desarrollo de una aplicación web que permita controlar el RM cuya interfaz integre la visualización de valores recolectados por los sensores integrados al SAR y generación de estadísticas a partir de estos datos.
* Selección de un medio de comunicación inalámbrica (Radiofrecuencia) que permita la interrelación entre la aplicación móvil y el SAR.

## **1.4 Resultados Esperados**

Al finalizar la tesina esperamos haber construido el robot móvil a partir de la integración de las diversas plataformas previamente mencionadas, conformando el denominado SAR.

Se espera aportar conocimiento significativo para futuros proyectos que requieran la utilización de protocolos de comunicación inalámbricos entre aplicaciones móviles y microcontroladores.

Tanto el desarrollo del software como el hardware serán liberados para contribuir a un mejor proceso de enseñanza de la informática y robótica en principio en el nivel medio.

Un resultado esperable es que el SAR en su conjunto sea fácilmente extensible y por lo tanto se prevé que otros continúen la evolución del producto y sea utilizado como base para nuevos proyectos relacionados con la robótica y aplicaciones móviles.

Otro resultado esperado es que los anexos referentes a la utilización de módulos sean de utilidad para la enseñanza de electrónica en nivel medio.

# Capítulo 2 - La robótica

 En este capítulo se va a abordar el concepto de la robótica desde el punto de vista de su utilidad en áreas relacionadas con la informática, para el ámbito educativo. Se introducen diversas estructuras robóticas, como también distintas plataformas que facilitan la aplicación de esta ciencia, dando soporte didáctico, en la actualidad. Además, se distinguen los conceptos de microcontrolador y computadora de placa reducida, detallando ventajas, desventajas y formas de comunicación de cada uno de ellos. Finalmente, se define que es un sistema autónomo robótico (el cual, como se mencionó en el capítulo anterior, es el desarrollo propuesto por esta tesina) concluyendo con el impacto de la robótica en la educación.

## **2.1 ¿Qué es la robótica?**

A lo largo de la historia el ser humano ha sentido fascinación por las máquinas que puedan imitar las figuras y movimientos de seres animados. El poder desarrollar sistemas electromecánicos que simulen o realicen actividades típicas de seres vivos, ofrece la sensación de tener un propósito propio, lo cual fue un motivador para su estudio.

A este tipo de maquinaria se la denomina Robot. Según la RIA (Robotic Industries Association):

*“Un robot es un manipulador funcional reprogramable, capaz de mover material, piezas, herramientas o dispositivos especializados mediante movimientos variables programados, con el fin de realizar tareas diversas.”*

Una de las grandes diferencias entre los robots y el resto de las máquinas es la versatilidad que adquieren los mismos al poder variar su propósito modificando su programación. Todas las tareas que realizan los robots están basadas en la manipulación de su entorno.

Se le considera robótica a la ciencia y técnica encargada del diseño, construcción y aplicación de robots. Esta ciencia involucra diversas disciplinas tales como la mecatrónica, electrónica, mecánica, e informática, entre otras.

Actualmente la robótica ha ido evolucionando rápidamente, dando lugar a innovaciones tecnológicas destacadas para la historia de la humanidad, logrando un alto impacto socio-económico. Hoy en día, la robótica no es solo utilizada en los ámbitos industriales o militares, sino que podemos ver a robots en variadas áreas como por ejemplo en la medicina o en la educación.



Ilustración - Esquema básico de un robot

En la imagen (Ilustración 1 - Esquema básico de un robot) se puede apreciar el esquema básico del funcionamiento de un robot, detallando los componentes que pueden tener (Actuadores, sensores y un sistema de control).

La robótica está constituida por tres grandes temas como lo son; la *percepción*, la *planificación* y la *manipulación*. En conjunto permiten el desarrollo de robots con un gran índice de autonomía, logrando acciones básicas que realiza un ser humano al ejecutar ciertas tareas. Cuando una persona ha detectado una necesidad, los primeros pasos que realiza es estudiar su entorno con alguno de sus cinco sentidos (*percepción*); luego toma la decisión de realizar acciones con determinados movimientos (*planificación*) para que, finalmente, las ejecute de modo secuencial (*manipulación*).

Podemos identificar elementos y acciones relacionados con cada etapa de la secuencia antes descripta:

Percepción:

* Sensores
* Tratamiento de información
* Procesamiento de información

Planificación:

* Trayectorias
* Tareas
* Planificación de tareas
* Toma de decisiones

Manipulación:

* Mecánica
* Actuadores
* Sistema de control
* Sistema de programación

## **2.2 Estructura física de los robots**

La estructura es definida por el tipo de configuración general de las distintas piezas que conforman al Robot. Es difícil establecer una clasificación estricta de los mismos que resista un análisis riguroso. La subdivisión de los Robots, con base en su arquitectura, se podría hacer dentro de alguno de los siguientes grupos: poliarticulados, móviles, androides, zoomórficos e híbridos.

### 2.2.1 Poliarticulados

Se les denomina robots poliarticulados a aquellos que en su mayoría son sedentarios o de desplazamientos muy limitados y tanto su forma como configuración pudiera ser muy diversa. En este grupo entrarían aquellos robots estructurados para mover sus componentes terminales (Ej.: sus actuadores) en un espacio determinado de trabajo con una simetría específica. Ejemplos, podrían ser los robots industriales, cartesianos y/o manipuladores. En la ilustración anterior (Ilustración 2 - Ejemplo de robot poliarticulado) se muestra un brazo robótico como ejemplo de un robot poliarticulado.

Ilustración - Ejemplo de robot poliarticulado

### 2.2.2 Móviles

Estos robots se caracterizan, primordialmente, por su capacidad de desplazamiento. Su forma, por lo general, se basa en diseños típicos de vehículos como los automóviles. Su objetivo prioritario suele ser recorrer un determinado camino guiándose por la información de su entorno, obtenida a través de sus sensores. Pueden ser dotados de un cierto nivel de inteligencia (gracias a su programación) e incluso sortear obstáculos. En la imagen (Ilustración 3 - Ejemplo de robot móvil) se visualiza un robot móvil que cuenta con 4 ruedas y motores para su desplazamiento, y a su vez con un brazo manipulado por servo motores.

Ilustración - Ejemplo de robot móvil

### 2.2.3 Androides



Se les llama androide a los robots que intentan simular y/o reproducir la forma y comportamiento cinemático de seres vivos. Todavía no cuentan con alguna aplicación práctica específica, sino más que, para el estudio y la experimentación. La imagen (Ilustración 4 - Androide Asimo de Honda) muestra el androide ASIMO creado por la compañía japonesa Honda en el año 2000.

Ilustración - Androide Asimo de Honda

### 2.2.4 Zoomórficos

Los Robots zoomórficos, se caracterizan principalmente por sus sistemas de locomoción que tienen como objetivo imitar a los diversos seres vivos, como se puede apreciar en la imagen (Ilustración 5 - Robot Zoomórfico caminador) un robot con forma canina. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción se suelen distinguir entre dos categorías principales: caminadores y no caminadores. El grupo de los no caminadores está muy poco evolucionado. Los Robots zoomórficos caminadores multípedos son muy numerosos y están siendo objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenales, pilotados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos Robots apuntan a su utilización en el campo de la exploración espacial y en el estudio de los volcanes.

Ilustración - Robot Zoomórfico caminador

### https://lh4.googleusercontent.com/Iop1qqdMsk7UnEMkQs6-v938nAD7qo8OVTlpS-kQ6kgmjNjhegpQ9YcBiHqOy3RBTBYb5whkIafhH6t6Bfsxk6ALuxxxNW5ErbhPGpIyAI2Y3ZQJCFjVwj3AkZABWm4fRvTY4zdO2.2.5 Híbridos

Los robots híbridos se les consideran a aquellos a los cuales es difícil clasificar dentro de las mencionadas anteriormente o bien es la combinación de algunas de ellos. En esta imagen (Ilustración 6 - Robot móvil-poliarticulado), se puede observar un robot móvil con variados actuadores para la manipulación de objetos y que además su forma es similar a la de un escorpión.

Ilustración - Robot móvil-poliarticulado

## **2.3 Distintas tecnologías para la robótica educativa**

Sin duda alguna, en los últimos años, las arquitecturas más destacadas para la enseñanza y desarrollo de robótica a nivel educativo han sido las plataformas **Arduino**[[2]](#footnote-2)y **Raspberry Pi[[3]](#footnote-3)**. Gracias a su costo accesible y disponibilidad de versiones, estas tecnologías son utilizadas en las diversas disciplinas relacionadas con la robótica educativa. En el caso de Arduino, presenta una notable ventaja dentro de este ámbito dado que la compañía que lo fábrica (del homónimo Arduino) libera su hardware y a su vez ofrece una amplia variedad de modelos para usos múltiples (se brindará más detalle sobre esta tecnología en el siguiente capítulo). Por otro lado, Raspberry Pi es un computador reducido creado con el objetivo de la enseñanza de la informática, cuenta con notables capacidades de procesamiento en relación a su bajo costo.

La gran ventaja de estas arquitecturas con respecto a las que se mencionan a continuación, es su gran soporte y compatibilidad, dada la amplia comunidad que las utiliza.

Existen otras tecnologías para el desarrollo de la robótica tales como; la plataforma **Intel Galileo**, similar a Raspberry Pi pero desarrollada por Intel, es también un computador reducido certificado por Arduino que integra la arquitectura Intel X86; **BeagleBone**, es una placa computadora de hardware libre diseñada como plataforma de evaluación y de prototipos para ingenieros profesionales; **Nanode**, es un placa de microcontrolador de código abierto, similar a Arduino, que cuenta con un módulo Wifi incorporado, su objetivo es el de la experimentación en Iot (Internet de las cosas).

## **2.4 Microcontroladores y computadora de placa reducida (SBC)**

Un **microcontrolador** es un circuito integrado programable, por lo general montado sobre una PCB (placa de circuito impreso), con la capacidad de ejecutar órdenes cargadas en su memoria. Su velocidad de procesamiento es limitada comparada con un CPU dado que su objetivo es el de funcionar como controlador. Son utilizados en periféricos informáticos, electrodomésticos, control de sistemas mecánicos, etc.

Puede ser muy común pensar que un microcontrolador es igual a un microprocesador, pero esto no es así, de hecho, difieren en muchos aspectos. La principal diferencia es su funcionalidad, dado que, para utilizar un microprocesador en alguna aplicación real, se debe conectar con diversos componentes tales como memorias o buses de transmisión de datos.

Aunque el microprocesador se considera una máquina de computación poderosa, no está preparado para la comunicación con los dispositivos periféricos que se le conectan. Para que el microprocesador se comunique con algún periférico, debe interactuar con un microcontrolador (cómo por ejemplo en el caso de un mouse, disco rígido o una cámara web). Por ende, se puede decir que, el CPU requiere del microcontrolador para la comunicación con el resto del hardware. Así era en el principio y esta práctica sigue vigente en la actualidad.

Por otro lado, al microcontrolador se lo diseña de tal manera que tenga todos los componentes integrados en el mismo chip, como se puede apreciar en la siguiente imagen (Ilustración 7 - Arquitectura de un microcontrolador). No necesita de otros componentes especializados para su operación, porque todos los circuitos necesarios, que de otra manera correspondan a los periféricos, ya se encuentran incorporados. De esta forma se ahorra tiempo y espacio al momento de su utilización.

Es por estas razones que han tenido grandes repercusiones para el desarrollo de la robótica.

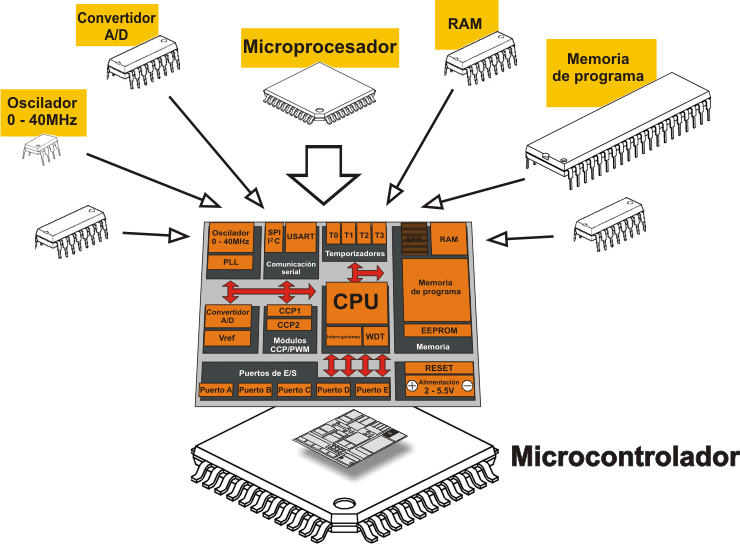


Ilustración - Arquitectura de un microcontrolador

Una **computadora de placa reducida** (SBC, *Single Board Computer*),en cambio, es una computadora completa que integra todos los componentes necesarios, que definen a la misma, en un solo circuito (la placa madre o *motherboard*) con la particularidad de que la misma es de un tamaño mucho más reducido que el de una computadora tradicional. Ejemplos típicos de este tipo de computadoras son las plataformas Arduino y Raspberry Pi.

En el caso de Arduino, dentro de su placa se integra un microcontrolador para el procesamiento de sus órdenes programadas, en cambio, Raspberry Pi integra un microprocesador con capacidades de ejecutar un sistema operativo con interfaz gráfica.

## **2.5. Comunicación entre distintas plataformas de cómputo**

Existen diversos medios de comunicación entre las PCs y las SBCs o de dispositivos de cómputo entre sí, a continuación, se listan algunos de ellos:

* Serial: La comunicación serie o serial es una interfaz de comunicación de datos digitales que nos permite establecer transferencia de información entre varios dispositivos. Un puerto es el nombre genérico con que denominamos a las interfaces, físicas o virtuales, que permite esta comunicación entre dispositivos. Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). Las placas Arduino actuales cuenta con un puerto USB para realizar este tipo de comunicación y es su principal interfaz para conectarlos a una PC donde cargar la secuencia de órdenes que luego ejecutará.
* Inalámbricas: Los medios de comunicación inalámbricos, para computadoras, han evolucionado de forma exponencial desde su aparición. Su gran ventaja, como su nombre lo dice, es que no necesitan de un medio de propagación físico (como los cables) para la transmisión de los datos, sino que, para el envío de los mismo se utiliza la modulación de ondas electromagnéticas a través del espacio. Existen diversos tipos, con grandes diferencias en cuanto a velocidades y rangos de alcance. En cuanto para la robótica podemos encontrar dispositivos que nos permitan conectar computadoras de placas reducidas con diversos computadores por medio de:
  + *Radiofrecuencia*: Existen módulos compatibles con Arduino, como el módulo de radiofrecuencia RF 433Mhz, que nos permiten conectar dos dispositivos de este tipo entre sí de forma inalámbrica a través de radiofrecuencia.
  + *Infrarrojo*: Las redes de luz infrarroja están limitadas por el espacio, se utilizan por lo general en dispositivos que se encuentran en un mismo espacio físico como un cuarto o un piso. Utilizan luz infrarroja tanto como para la transmisión como para la recepción de datos.
  + *Bluetooth*: Es una especificación industrial que permite crear redes inalámbricas de área personal (WPAN), mediante un enlace de radiofrecuencia que trabaja en la banda ISM (Industrial Scientific and Medical) de 2.4 GHz posibilitando la transmisión de voz y datos.
  + *Wifi*: Este mecanismo de comunicación inalámbrica es el más popular entre computadoras de hoy en día. A su vez, es una marca de la Alianza Wi-fi la cual certifica que los dispositivos cumplan con los estándares IEEE 802.11 vigentes relacionados a redes inalámbricas de área local.

## **2.6 ¿Qué es un SAR (Sistema Autónomo Robótico)?**

Se le considera SAR o sistema autónomo robótico a aquellos robots que presentan cierto grado de autonomía (Inteligencia artificial[[4]](#footnote-4)) y que, cuentan con la capacidad de testear su entorno (por medio de sensores) para decidir qué acciones realizar (por medio de actuadores). Por ende, se puede decir que, son sistemas dinámicos que consisten en un controlador electrónico acoplado a un cuerpo mecánico.

En el desarrollo propuesto por esta tesina, se diseñó y armó un sistema autónomo robótico móvil que posee un cierto grado de inteligencia, pero a su vez, permite ser manipulado desde una aplicación web.

## **2.7 La robótica en la educación**

En educación pueden diferenciarse dos tipos de uso de la programación y la robótica como apoyo en la clase: por un lado, la robótica y la programación como elemento educacional, y por otro, como elemento social.

Como elemento educacional, consiste en un conjunto de elementos físicos o de programación que motivan a los estudiantes a construir, programar, razonar de manera lógica y crear nuevas interfaces o dispositivos.

Mientras que, por otro lado, la programación y la robótica también es utilizada como elemento social, por ejemplo, a modo de juego o gamificación, de forma que sistemas autónomos o semiautónomos interactúan con humanos u otros agentes físicos o software en roles como entrenador, compañero, dispositivo tangible o registro de información.

El desarrollo de actividades educacionales basadas en robots o en programación pueden incrementar el compromiso y motivación por el aprendizaje en otras áreas como literatura o historia a través del juego. Aún más, su uso puede mejorar el desarrollo ético, emocional y social en base al impacto que, por ejemplo, un robot con atribuciones sociales puede causar en los niños.

Otro beneficio, es su potencial educativo para niños con necesidades especiales tanto en las áreas cognitivas como psicosociales. La escalabilidad de las propuestas educativas basadas en robots, y su enorme potencial motivador, lo hacen especialmente útil en programas de refuerzo y de educación especial.

Una de las grandes controversias en estas áreas, es sobre los materiales que deben utilizarse en el aula. Algunos investigadores, como Cecilio Angulo (Profesor de la Universitat Politècnica de Catalunya y director del Grupo de Investigación en Ingeniería del Conocimiento), afirman que los dispositivos tangibles aumentan el nivel de inmersión porque los estudiantes están manipulando las cosas en un mundo real. Sin embargo, podemos encontrar otros estudios que entienden que los dispositivos no tangibles, como los elementos de programación, atraen más y evitan limitaciones a causa de la necesidad de un cuerpo físico en el espacio real. Por tanto, lo que parece lógico es un enfoque híbrido entre robótica y programación, donde una fusión entre lo físico y lo virtual proporciona más flexibilidad a los docentes y a los estudiantes.

La robótica y la programación en conjunto brindan una experiencia de aprendizaje particular respecto a otras áreas, porque la posibilidades ofrecidas por la utilización de computadoras se localiza no solo en una pantalla, sino también, en objetos tangibles, que comparten con los interesados en un espacio físico con la posibilidad de afectar su entorno. Aprender a través de la robótica aumenta el compromiso de los alumnos en actividades basadas en la manipulación, el desarrollo de habilidades motoras, la coordinación ojo-mano y una forma de entender las ideas abstractas. Además, las actividades basadas en robots proporcionan un contexto apropiado para el comportamiento cooperativo y el trabajo en equipo.[[5]](#endnote-1)

En Argentina, existen distintos centros de estudios relacionados con la robótica educativa, uno de los más renombrados es RoboGroup. Esta es una empresa nacional dedicada al diseño, fabricación y capacitación en robótica, que, según la misma, su objetivo es insertar la robótica como sistema interdisciplinario de aprendizaje en las entidades educativas de todos los niveles de nuestro país. Anualmente organiza campeonatos de robots para alumnos de colegios primarios y secundarios llamados Roboliga.

Flata resumen

# Capítulo 3 – Arduino

Flata intro

## **3.1 ¿Qué es Arduino?**

Arduino es una plataforma y compañía, del mismo nombre, de electrónica "open-source" o de código abierto cuyos principios son contar con software y hardware fáciles de usar y que cualquiera pueda fabricar y mejorrar. Es decir, se propone como una plataforma sencilla de aprender para realizar proyectos interactivos para público no necesariamente con conocmientos técnicos.

Ilustración - Logo Arduino

Arduino se trata de una SBC con entradas y salidas, analógicas y digitales, la cual es programda bajo un entorno de desarrollo, inspirado en el entorno de programación **Processing**. En la imagen (Ilustración 8 - Logo Arduino) se puede ver el logo oficial de la compañía.

## **3.2 ¿Qué es Processing?**

**Processing**es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. En la imagen (Ilustración 9 - Logo de Processing) se puede apreciar su logo.

Uno de los objetivos expresos de Processing es el de actuar como herramienta para que artistas, diseñadores visuales y miembros de otras comunidades ajenos a la programación, aprendan las bases de la misma a través de una realimentación gráfica inmediata y visual de los resultados obtenidos de su experiencia de programación.

Ilustración - Logo de Processing

El lenguaje de Processing se basa en Java, aunque hace uso de una sintaxis simplificada y de un una biblioteca sencilla para generación de gráficos.

## FALTA EJEMPLO DE PROCESSING PARA CONECTAR CONCEPTOS CON WIRING

## Wiring

Wiring es un framework basado en Processing, enfocado en la programación de microcontroladores en vez de programación gráfica. Proveé una librería de C/C++ la cual simplifica operaciones comunes como el manejo de entrada/salida. Los programas de Wiring están escritos en C/C++, pese a que sus usuarios sólo necesiten definir dos funciones para hacer un programa ejecutable:

setup() – una función ejecutada sólo una vez en el arranque de la placa, la cual puede ser usada para definir los ajustes iniciales de un entorno.

loop() – una función llamada repetidamente hasta que la placa es apagada.

Como podemos apreciar en la siguiente ilustración (Ilustración 10 - Wiring IDE) hacer un blink a un led es muy sencillo dado la abstracción que nos otorga la librería. Un blink es un parpadeo de un led conectado a la placa. Se lo considera el “hola mundo” de Arduino.

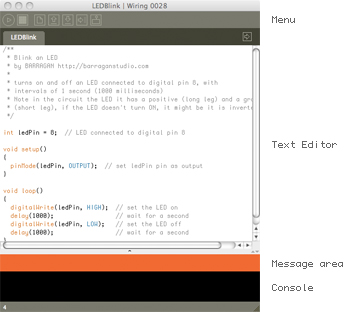


Ilustración - Wiring IDE

Por otro lado, trabajando con C++ sin librerías quedaría como en la siguiente figura (Ilustración 11 - C++ Blink ejemplo) haciendo el mismo blink, programando directamente con la biblioteca AVR (control de los puertos de entrada y salida) y en C.[[6]](#endnote-2)



Ilustración 11 - C++ Blink ejemplo

## **3.3** **Fritzing**

El entorno Fritzing ayuda a los diseñadores y artistas a documentar sus prototipos interactivos y dar paso en la creación de prototipos físicos al producto real. Como podemos apreciar en la siguiente ilustración (Ilustración 12 - Entorno Fritzing), permite arrastrar componentes y generar un sketch.

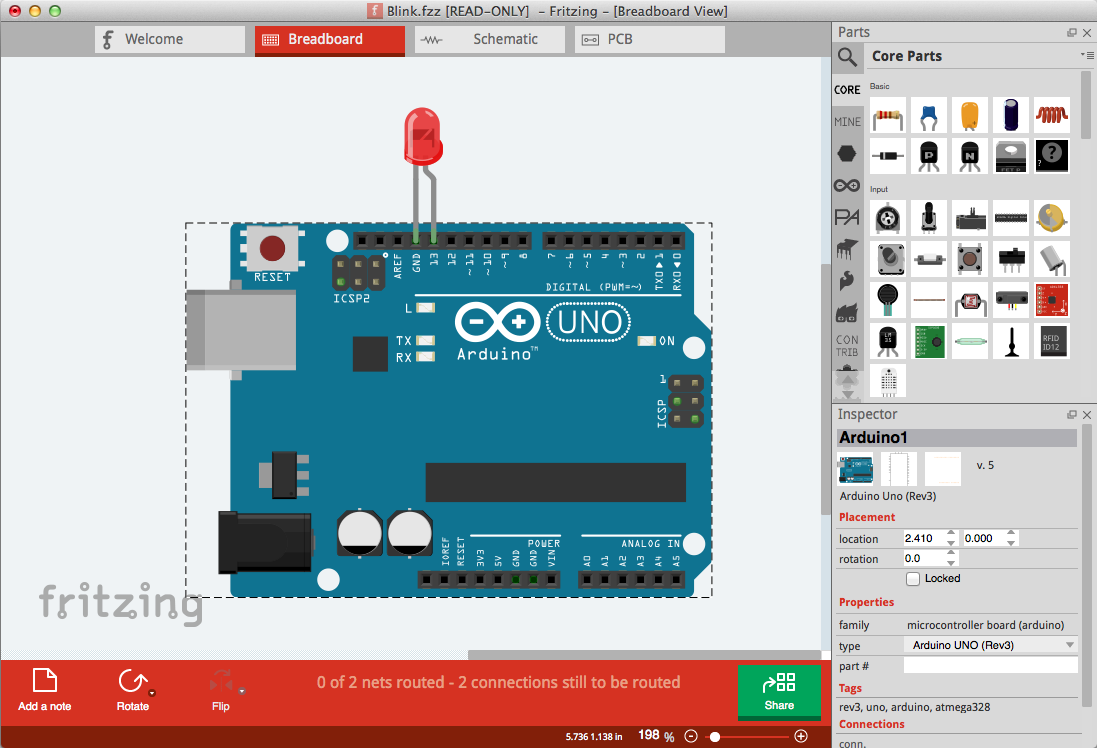


Ilustración - Entorno Fritzing

## **3.4 ¿Qué es Wiring?**

Wiring es una plataforma de prototipado electrónico de fuente abierta compuesta de un lenguaje de programación, un entorno de desarrollo integrado (IDE), y un microcontrolador.

Esta plataforma permite escribir software para controlar dispositivos conectados a la tarjeta electrónica para crear toda clase de objetos interactivos, espacios o experiencias físicas que sienten y responden al mundo físico.

Este proceso se llama *sketching* con hardware; se explora una gran cantidad de ideas de forma muy rápida, se seleccionan las más interesantes, se afinan y producen prototipos en un proceso iterativo.

## **3.5 Entonces Arduino es…**

* Una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.
* Una plataforma de hardware abierto que facilita la programación de un microcontrolador. Los microcontroladores nos rodean en nuestra vida diaria, usan los sensores para escuchar el mundo físico y los actuadores para interactuar con el mismo. Los microcontroladores leen sobre los sensores y escriben sobre los actuadores.

## **3.6 Plataforma Arduino**

La plataforma consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida los cuales pueden conectarse a placas de expansión (*shields*), que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

Los puertos serie que cuentan las distintas plataformas Arduino presentan puertos de serie como UART. La UART (universally asynchronous receiver/transmitter) es una unidad que incorporan ciertos procesadores, encargada de realizar la conversión de los datos a una secuencia de bits y transmitirlos o recibirlos a una velocidad determinada.

Por otro lado, también opera en nivel TTL (transistor-transistor logic). Esto significa que la comunicación se realiza mediante variaciones en la señal entre 0V y Vcc (donde Vcc suele ser 3.3V o 5V). Por el contrario, otros sistemas de transmisión emplean variaciones de voltaje de -Vcc a +Vcc (por ejemplo, los puertos RS-232 típicamente varían entre -13V a 13V).

Como podemos observar en la siguiente ilustración (Ilustración 13 - Ejemplo serie), se realiza una comunicación serie a (9600 bps) imprimiendo un contador. La zona marcada con rojo, es un botón que al presionarlo nos permite acceder a la terminal y ver el flujo serie seteando el clock correspondiente.

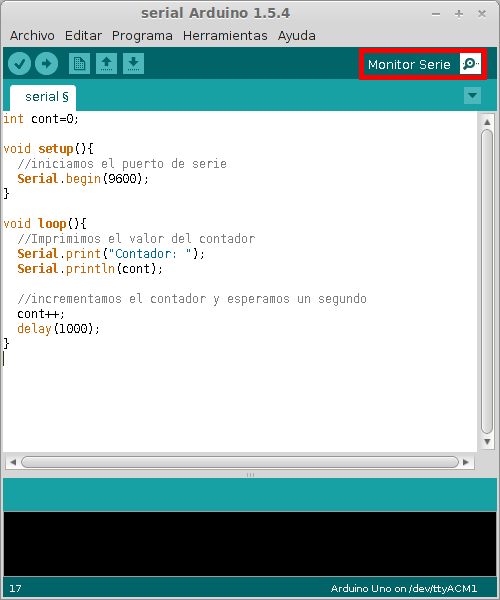


Ilustración 13 - Ejemplo serie

## **3.7 Distintas plataformas para Arduino**

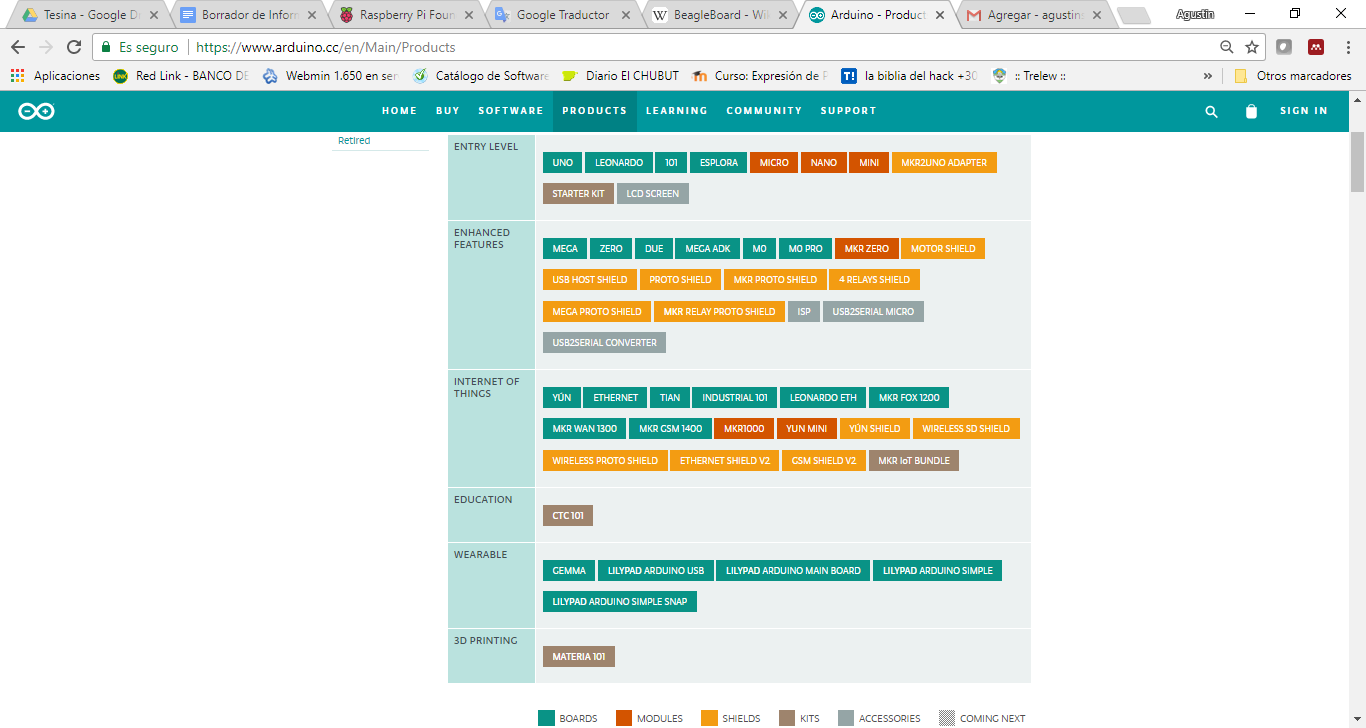


Ilustración - Niveles de entrada a la plataforma Arduino

Existe una gran variedad de productos Arduino, la compañía los cataloga, como se puede aprecias en la imagen anterior (Ilustración 14 - Niveles de entrada a la plataforma Arduino), en distintos niveles según su utilidad[[7]](#endnote-3):

* Nivel de entrada: Son los más sencillos de utilizar, ideales para comenzar con la plataforma Arduino y realizar proyectos sencillos.
* Características mejoradas: Estas plataformas poseen características superiores, con respecto a las del nivel de entrada, están pensadas para proyectos más avanzados o de respuesta más rápida.
* Internet de las cosas: Estas placas vienen incorporadas con componentes que permitan realizar trabajos relacionados con la IoT (Internet de las cosas)[[8]](#footnote-5) mediante la incorporación de hardware de conectividad.
* Educación: En este caso, Arduino, ofrece un kit con herramientas y más de 25 proyectos, orientados a la educación, para realizar con sus plataformas.
* Usables: Estas plataformas están pensadas para “agregarle algo de electrónica” a prendas de vestir.
* Impresión 3D: Arduino ofrece una impresora 3D nombrada como Materia 101.

El hardware Arduino más sencillo consiste en una placa con un microcontrolador y una serie de puertos de entrada y salida. Los microcontroladores de 8 bits de AVR más utilizados en estas placas son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste, aunque también se dispone de microcontroladores ARM, cómo el caso del CortexM3 de 32 bits. A pesar de que ARM y AVR son plataformas diferentes, al utilizar la IDE de Arduino, los programas se compilan y luego se ejecutan sin cambios en cualquiera de las plataformas. En la imagen (Ilustración 15 - Arduino Uno) se visualiza la distribución física de puertos y componentes de la versión Arduino Uno R3.

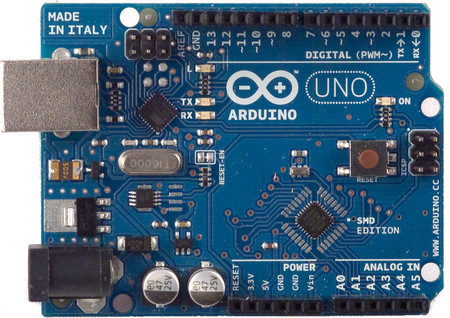


Ilustración - Arduino Uno

## **3.8 Diferencias entre distintas placas de la familia Arduino**

Una primera diferenciación entre los distintos modelos de Arduino la encontraremos en el voltaje o tensión de alimentación de las placas. Las basadas en CortexM3 operan con un voltaje de 3,3 voltios, mientras que la mayor parte de las placas basadas en AVR utilizan una tensión de 5 voltios. Esto de todas formas no es un factor decisivo en la elección de una placa, dado que existen conmutadores de tensión en muchos actuadores y sensores compatibles.

Los usos posibles que se le pueden dar a un Arduino, en forma general son:

* Aquellos en los que el Arduino es utilizado como microcontrolador, tiene un programa descargado desde un ordenador y funciona de forma independiente de éste, y controla y alimenta determinados dispositivos y toma decisiones de acuerdo al programa descargado e interactúa con el mundo físico gracias a sensores y actuadores.
* La placa Arduino hace de interfaz entre un ordenador (como podría ser una Raspberry Pi) u otro dispositivo, que ejecuta una determinada tarea, para traducir dicha tarea en el mundo físico a una acción (actuadores).

## **3.9 ¿Por qué usar Arduino?**

### 3.9.1 La comunidad

La comunidad Arduino se desarrolla y enriquece a partir del trabajo con la placa, de la experimentación, de la producción de conocimiento en torno a ella, y estas habilidades se comparten dentro de la comunidad, pudiendo cualquier persona tener acceso a ellas.

La página principal de Arduino da soporte por medio de secciones y dedica un blog histórico con novedades y proyectos que se encuentran en desarrollo.

Además, se crean sitios como Arduino playground (<http://playground.arduino.cc/>) que es una wiki donde todos los usuarios de Arduino pueden contribuir. Es el lugar donde publicar y compartir código, diagramas de circuitos, guías, manuales, cursos. Es la base de datos de conocimiento de la comunidad de Arduino. Este sitio a su vez tiene soporte a distintos lenguajes como el español (<https://playground.arduino.cc/Es/Es>).

Por otro lado, existe Arduino Hub, un lugar donde se comparten los proyectos, dando los distintos pasos para reproducirlo.

El manifiesto de la comunidad Arduino dice (traducción al español):

“Apoyar al ecosistema de hardware y software de open source Arduino, haciendo que los productos electrónicos sean abiertos y participativos.

Servir como un evangelista para Arduino, expandir el ecosistema de código abierto a estudiantes, fabricantes, desarrolladores, diseñadores, ingenieros y empresas dentro de sus comunidades locales.

Construir una red global de comunidades que diseñen y codifiquen proyectos, intercambien ideas, organicen actividades de colaboración y dicten cursos oficiales de Arduino, independientemente de su edad, sexo, idioma y capacidad técnica”[[9]](#footnote-6)

La siguiente imagen (Ilustración 16 - Logotipo comunidad open-source de Arduino) muestra el logotipo oficial de la comunidad open-source de Arduino.



Ilustración 16 - Logotipo comunidad open-source de Arduino

### 3.9.2 La sencillez del lenguaje de programación

Programar la placa es muy sencillo y accesible, y la ayuda por parte de la comunidad lo hace aún más fácil. Como se mostró en las ilustraciones (Ilustraciones 10 y 11), podemos apreciar librerías con un alto nivel de abstracción en cuanto al acceso a bajo nivel.

La definición de puertos con la sentencia a alto nivel, seteando los puertos del 1 al 7 en un Arduino UNO como salida sería:

PinMode(1,OUTPUT) ;

PinMode(2,OUTPUT) ;

…;

PinMode(7,OUTPUT) ;

en contraposición con:

DDRD = B11111110; [[10]](#footnote-7) // sets Arduino pins 1 to 7 as outputs, pin 0 as input

### 3.9.3 Es hardware de bajo costo

Lo único que “vale” en la placa son sus componentes, ya que no debemos pagar el costo de la licencia de su creador, por el hecho de ser hardware libre.

## **3.10 Incorporación de Arduino en las escuelas**

Gracias a la plataforma Arduino y su comunidad, se permite una vinculación con la dinámica de las escuelas a través de la creación de una red de trabajo colaborativo. Esto conlleva a:

* La utilización de medios multimediales, para conformar cursos y capacitaciones.
* Creación de proyectos articulares entre distintos espacios curriculares
* Uso de diversas herramientas que son Arduino-compatibles y gratuitas.
* Costos relativamente bajos en la adquisición de las distintas plataformas.

Todos estos puntos anteriores favorecen la introducción en el aula y clubes.

Por otro parte, la sencillez del lenguaje de programación de la placa, permite la rápida utilización por alumnos y docentes, no necesariamente del ámbito de la informática y la electrónica. Esto contribuye a la construcción colectiva del conocimiento, promoviendo la interdisciplinariedad escolar, donde docentes de distintas áreas articulan para crear proyectos.

Utilizando clubes de ciencia o proyectos específicos permite que el trabajo se apoye sobre un modelo pedagógico de aprendizaje en proceso, donde el sujeto que aprende es participante activo de ese proceso, desde la concepción de la idea hasta el producto final, incorporando conocimientos técnicos específicos.

Este tipo de actividades educativas hacen que la tecnología y su uso se pongan al servicio de la creatividad, el juego, la experimentación y la invención, con la posibilidad de ser adaptado al contexto en el que se inserta. Además, proporcionar la recuperación de la tecnología obsoleta existente en ellas como se describe en la siguiente sección.

### 3.10.1 Las tres erres

Las tres erres (reducir, reutilizar, reciclar) es una regla para cuidar el medio ambiente, específicamente para reducir el volumen de residuos o basura generada.

Cuando hablamos de reducir lo que estamos diciendo es que se debe tratar de simplificar el consumo de los productos directos.

Al decir **reutilizar**, nos estamos refiriendo a poder volver a utilizar los objetos y darles la mayor utilidad posible antes de que llegue el momento de desecharlos.

Por otro lado, **reciclar** consiste en el proceso de someter los materiales a una transformación en el cual se puedan volver a utilizar.

Esta definición se pretende aplicar en las escuelas haciendo un proceso de clasificación, selección y desoldando componentes electrónicos de placas en desuso y materiales que se han desechado en las instituciones o en hogares de los alumnos.

## **3.11 Actuadores y sensores**

Un **actuador** es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de una acción con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control, como por ejemplo un LED.

Un **sensor** es un objeto capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser, por ejemplo: intensidad lumínica, temperatura, distancia, aceleración, inclinación, presión, desplazamiento, fuerza, torsión, humedad, movimiento, pH, etc.

En conjunto, los sensores y actuadores, permiten la creación de distintos tipos de artefactos, que posibilitan comunicarse con el ambiente que los rodea, modificándolo (actuadores) o recibir estímulos (sensores).

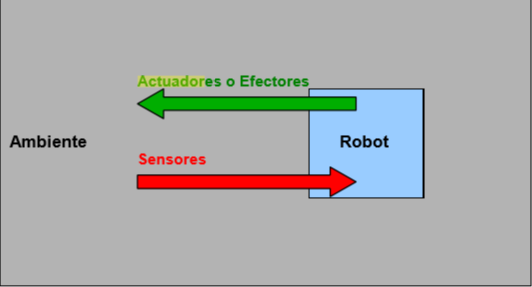


Ilustración - Representación actuadores y sensores

En esta imagen (Ilustración 17- Representación actuadores y sensores) se representa la toma de datos un robot puede capturar del ambiente que lo rodea mediante sus diversos sensores, y a su vez como podría interactuar con el mismo mediante actuadores.

## **3.12 Actuadores en el SAR**

La electrónica industrial ha generado estandarización en el campo de los sensores y actuadores, muchos de estos últimos con buen soporte en Arduino. Precisamente en el SAR se utilizarán:

* Motores de corriente continua
  + Para el desplazamiento del robot móvil
* LED
  + Para indicar estados del RM

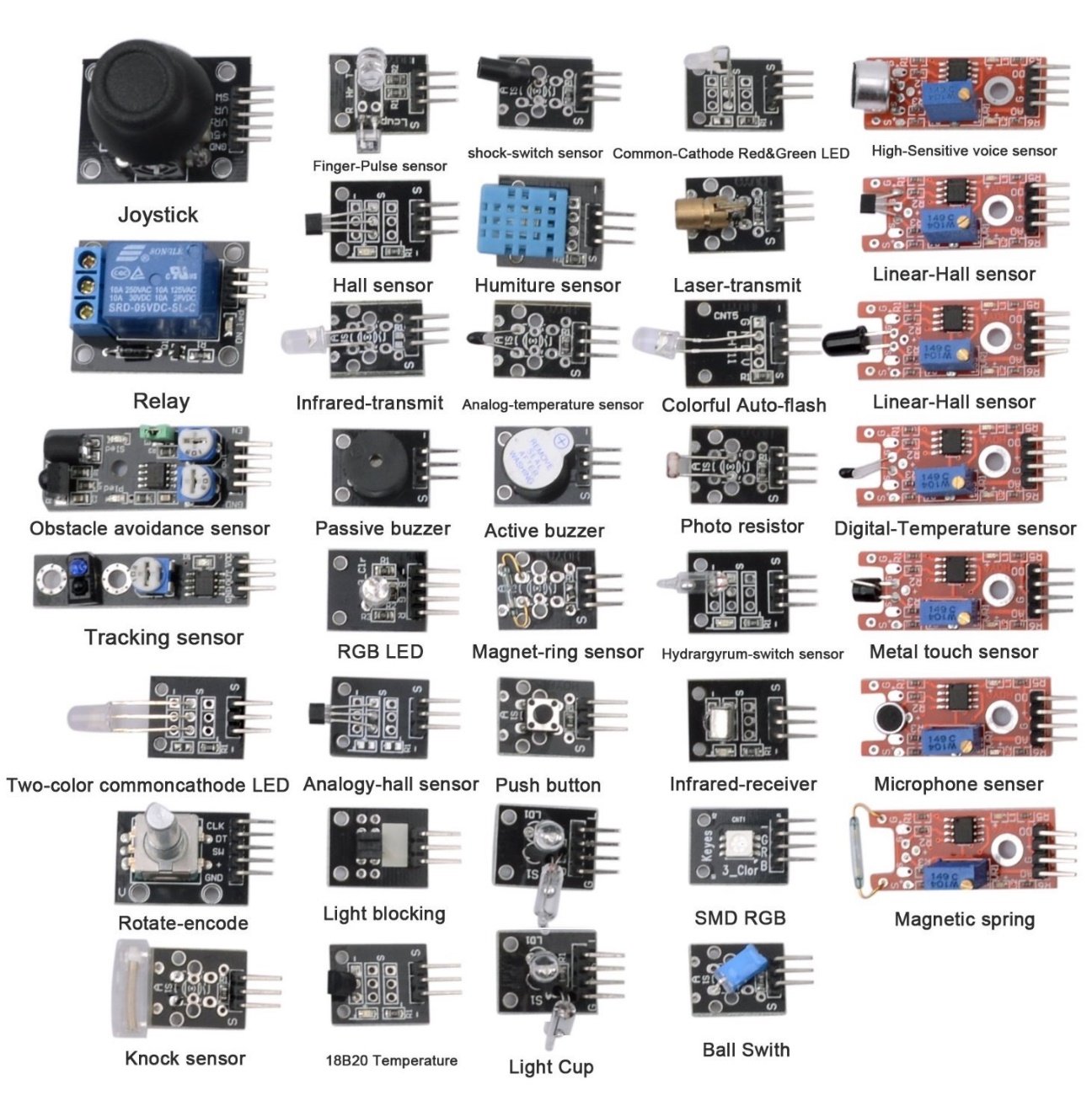


Ilustración - Actuadores y sensores compatibles con Arduino

## **3.13 Sensores en el SAR**

El SAR utiliza los siguientes sensores:

* Sensor ultrasónico HC-SR04
  + Para detectar objetos, y distancia entre el RM y elementos del ambiente
* Sensor de Temperatura KY-001
  + Incorporado para analizar la temperatura del ambiente
* Sensor de presencia de gases MQ-7
  + Detección de monóxido de carbono

Algunos de los sensores y actuadores se pueden apreciar en la ilustración anterior (Ilustración 18 - Actuadores y sensores compatibles con Arduino).

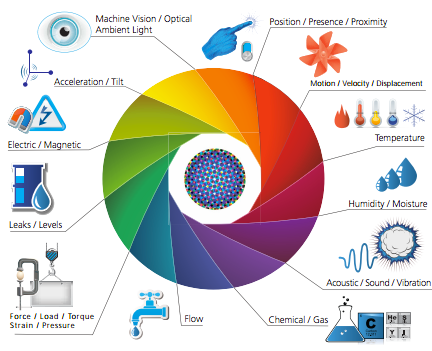


Ilustración - Representación de sensores

En esta imagen (Ilustración 19- Representación de sensores) se pueden apreciar los distintos factores de un entorno que pueden ser evaluadores con sensores mencionados anteriormente.

## **3.14 Módulos o *shields* en el SAR**

El SAR utiliza[[11]](#footnote-8):

* MotorShield L298
  + Para administración del puente H y gestión de los motores de CC
* Módulo bluetooth HC-05
  + Para la comunicación con dispositivos compatibles (móviles y/o computadoras)
  + Envío de órdenes
* Módulo GPS NEO-6
  + Para la geolocalización del RM
* Módulo ESP8266
  + Conectividad y transferencia de datos vía WIFI
  + Activación del Ad-hoc

A lo largo del desarrollo de la tesina se fueron implementando diversos casos de pruebas sobre los sensores, actuadores y módulos especificados en esta sección. Las pruebas se encuentran anexas en este documento.

# Capítulo 4 – Raspberry Pi

## **4.1 ¿Qué es Raspberry Pi?**

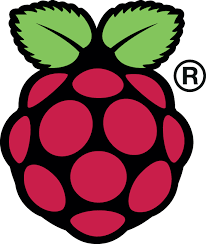
**Raspberry Pi** es un computador de placa reducida (SBC) desarrollado en Reino Unido por la Fundación Raspberry Pi. Su lanzamiento fue el 29 de febrero del 2012 con el *Raspberry Pi 1 Modelo A*. Su costo es relativamente bajo en relación a sus especificaciones técnicas (alrededor de U$D 25), dado que su objetivo primordial es el de estimular la enseñanza de la informática en las escuelas. Su logo oficial, como se muestra en la imagen (Ilustración 20 - Logo oficial de Raspberry Pi), no es más que una frambuesa.

Ilustración - Logo oficial de Raspberry Pi

## **4.2 Especificaciones técnicas de las distintas versiones**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Raspberry Pi 1 Modelo A** | | **Raspberry Pi 1 Modelo B** | | **Raspberry Pi 1 Modelo B+** | **Raspberry Pi 2 Modelo B** | **Raspberry Pi 3 Modelo B** |
| **SoC** | Broadcom BCM2835 ([CPU](https://es.wikipedia.org/wiki/CPU) + [GPU](https://es.wikipedia.org/wiki/GPU) + [DSP](https://es.wikipedia.org/wiki/Procesamiento_digital_de_se%C3%B1ales) + [SDRAM](https://es.wikipedia.org/wiki/SDRAM) + puerto USB) | | | | | Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + Puerto USB) | Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + Puerto USB |
| **CPU** | ARM 1176JZF-S a 700 MHz (familia ARM11) | | | | | 900 MHz quad-core ARM Cortex A7 | 1.2GHz 64-bit quad-core ARMv8 |
| **Juego de instrucciones** | RISC de 32 bits | | | | | | |
| **GPU** | Broadcom [VideoCore](https://es.wikipedia.org/w/index.php?title=VideoCore&action=edit&redlink=1" \o "VideoCore (aún no redactado)) IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC | | | | | | |
| **Memoria SDRAM** | 256 MiB compartidos con la GPU | 512 MiB compartidos con la GPU, desde el 15 de octubre del 2012 | | | | 1 GB compartidos con la GPU | |
| **Puertos USB 2.0** | 1 | 2 | | 4 | | | |
| **Entradas de vídeo** | Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la Fundación Raspberry Pi | | | | | | |
| **Salidas de vídeo** | Conector RCA (PAL y NTSC), HDMI (rev 1.3 y 1.4), interfaz DSI para panel LCD | | | | | | |
| **Salidas de audio** | Conector de 3.5 mm, HDMI | | | | | | |
| **Almacenamiento integrado** | SD, MMC, ranura para SDIO | | | MicroSD | | | |
| **Conectividad de red** | Ninguna | 10/100 Ethernet (RJ45) via hub USB | | | | | 10/100 Ethernet (RJ45) vía hub USB, Wifi 802.11n, Bluetooth 4.1 |
| **Periféricos de bajo nivel** | 8 x GPIO, SPI, I2C, UART | | | | | 17 x GPIO y un bus HAT ID | |
| **Consumo energético** | 500 mA (2.5 W) | 700 mA (3.5 W) | | 600 mA (3.0 W) | | 800 mA (4.0 W) | |
| **Fuente de alimentación** | 5 V vía Micro USB o GPIO header | | | | | | |
| **Dimensiones** | 85.60mm × 53.98mm | | | | | | |
| **SO soportados** | GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, SUSE Linux Enterprise Server for ARM.  RISC OS | | | | | | |

## **4.3 Entrada/Salida de propósito general (GPIO)**

Se le llama GPIO (En inglés, *General Purpose Input/Output*) a un conjunto de pines genérico integrado a una placa o chip electrónico sin un fin específico, sino que, su “comportamiento” queda sujeto al usuario de dicha placa según algún tipo de lógica previamente cargada.

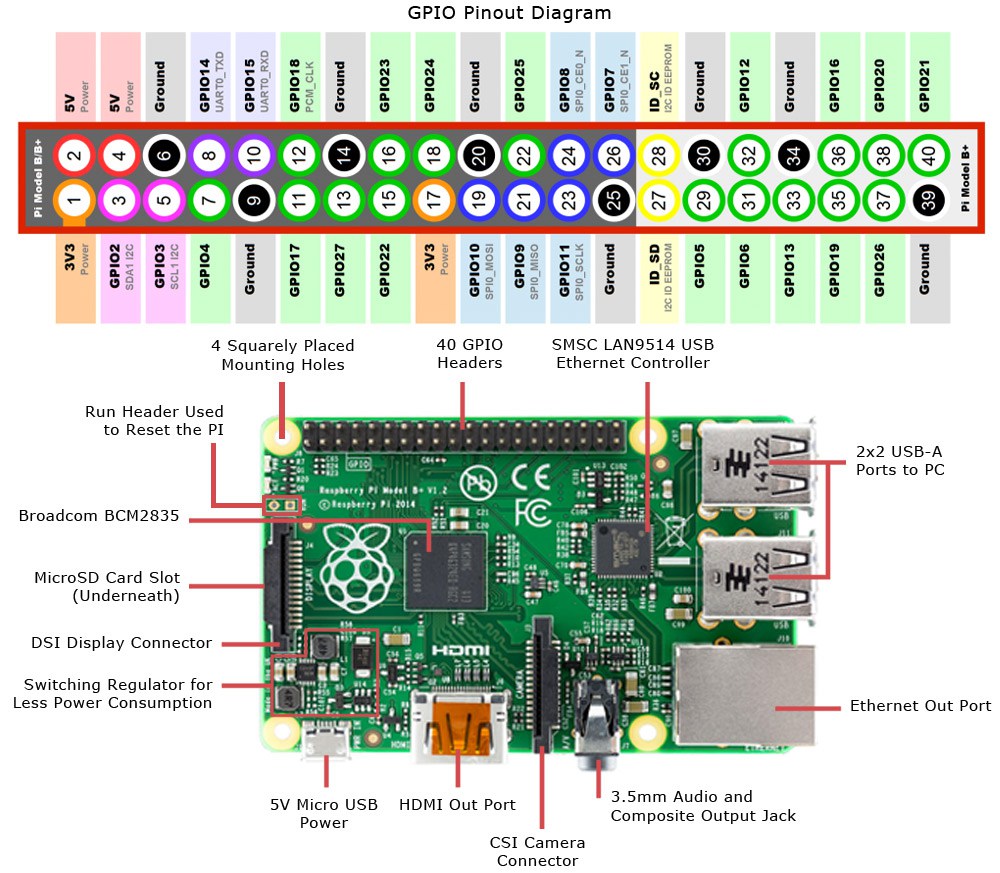
En la imagen (Ilustración 21 - Raspberry Pi 2 y sus GPIOs) se puede ver la Raspberry Pi 2 Modelo B de características bastante similares, en general, a la versión 3 de esta plataforma (utilizada en el desarrollo de esta tesina) y en detalle sus diversas interfaces. Un poco más arriba se pueden apreciar los distintos pines del tipo GPIO con los que cuenta esta plataforma (40 pines en total tanto la versión 2 como la 3)[[12]](#endnote-4).

Ilustración - Raspberry Pi 2 y sus GPIOs

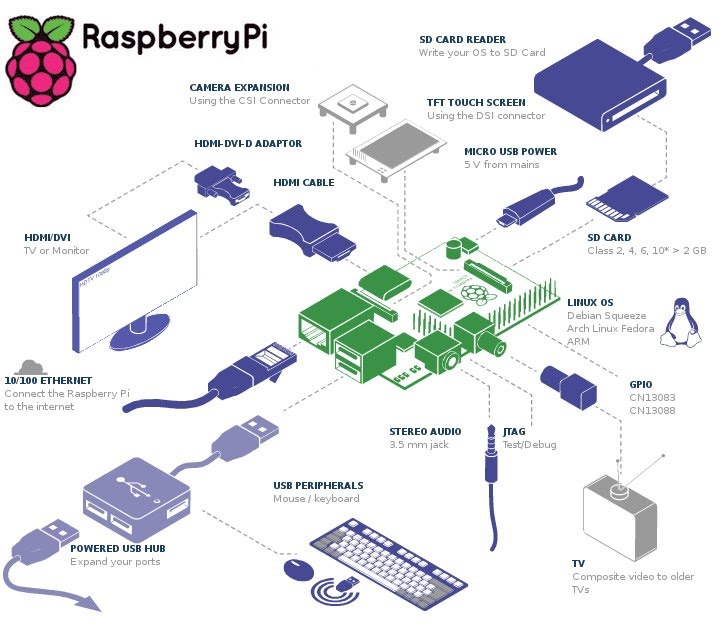
La siguiente imagen (Ilustración 22 - Interfaces de Raspberry Pi) ilustra los distintos periféricos que se pueden conectar a este computador.

Ilustración - Interfaces de Raspberry Pi

## **4.4 Sistemas Operativos compatibles**

Los computadores Raspberry Pi utilizan en su mayoría sistemas operativos basados en GNU/Linux compatibles con el mismo, alguno de ellos son los siguientes:

* Arch Linux
* Android
* Debian Whezzy
* Ubuntu Mate
* Google Chromium OS
* Raspbian

Este último (Raspbian), es una distribución derivada del sistema operativo Debian, la cual fue modificada y optimizada para el hardware de Raspberry Pi. Es la distribución por defecto recomendada por la Fundación Raspberry Pi para utilizarse en dicho computador.

Por otro lado, también existe una versión de Windows 10 desarrollada específicamente para sistemas embebidos, denominada IoT Core, compatible con esta plataforma (en particular con las Raspberrys Pi 2 y 3).

## **4.5 ¿Qué es Python?**

Se le llama Python a un lenguaje de programación multiparadigma y multiplataforma desarrollado en el año 1991 por Guido Van Rossum. A partir de marzo del 2001 la Python Software Foundation es la encargada de administrar este lenguaje. Su misión es la de fomentar el desarrollo de la comunidad Python.

Ilustración - Logo de Python

Este lenguaje de programación posee las siguientes características principales:

* Es de código abierto.
* Es multiparadigma, permite los estilos de programación orientado a objetos, imperativo y funcional.
* Permite otros paradigmas, a parte de los mencionados, con el uso de extensiones.
* Es interpretado.
* Utiliza tipado dinámico.
* Extensible.

## **4.5 Python con Raspberry Pi**

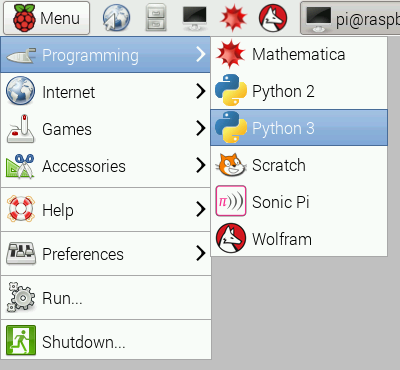
A Python se lo considera un lenguaje de programación fácil de aprender, además de ser muy popular y potente, es un lenguaje serio, usado en distintos ámbitos profesionales. Como se identificó en el apartado anterior, es un lenguaje de código abierto y multiplataforma, por lo que se puede utilizar en cualquier sistema con total libertad e incluso con fines comerciales si se quisiese.

Ilustración - Menú de Raspbian

Dicho esto, y por otros motivos, es que la Raspberry Pi Foundation lo ha seleccionado como el lenguaje de programación “estándar” para su plataforma Raspberry Pi. Lo que no significa que la misma se deba programar solo en Python. Raspberry Pi es una computadora que corre un sistema operativo, lo cual le da la facultad de correr diversos programas y por ende variados lenguajes.

Python es un lenguaje interpretado, que quiere decir que un “interprete” va leyendo las instrucciones y ejecutándolas en tiempo real. Un intérprete es un software encargado de convertir nuestras sentencias a código máquina. Puede instalarse cualquiera de los muchos intérpretes que existen para tu máquina, ya sea la Raspberry Pi o un ordenador personal.

El sistema operativo Raspbian, desarrollado específicamente para la Raspberry Pi, viene con las versiones de Python 2 y 3 pre- instaladas, con sus respectivos intérpretes.

Como ya se mencionó, la Raspberry Pi 3 cuenta con una cantidad de 40 pines del tipo GPIO. Con los cuales, al igual que en la plataforma Arduino, se pueden conectar y manipular un número considerable de actuadores y sensores.

Cabe aclarar que todos estos pines son del tipo digital, por lo que, este computador no cuenta con pines del tipo analógico. Para la lectura de sensores que devuelvan valores analógicos se debe utilizar un convertidor externo o un Arduino.

Para manipular estos pines, existe una librería nombrada como RPi.GPIO que nos permite, mediante Python, configurarlos rápidamente.

Como se puede apreciar en la Ilustración 21 - Raspberry Pi 2 y sus GPIOs podemos distinguir entre dos tipos de identificación de estos pines, según su orden físico en la placa o según su posición correspondiente a su conexión física a su CPU (en Raspberry Pi, Broadcom), estas formas de identificación se llaman BCM y BOARD respectivamente.

El número que identifica al pin en el modo BCM es el que esta después de la palabra GPIO (se puede apreciar en la Ilustración 21 - Raspberry Pi 2 y sus GPIOs), el índice para identificar los pines en modo BOARD son los que están dentro del círculo. Es importante mencionar que la identificación de los pines en modo BCM cambio entre la revisión 1 y la revisión 2 del hardware.

## **4.6 Accesorios para Raspberry Pi**

Para poder operar la placa **Raspberry Pi**, es necesario contar con ciertos accesorios, como una fuente de alimentación de al menos 1A, un cable HDMI, una tarjeta de memoria microSD con el Sistema Operativo y un adaptador WIFI o un cable RJ45 para poder conectarla en red. Además, ya sea por estética o por protección existen variados gabinetes o carcasas para su resguardo.

Algunos de los accesorios más comunes compatibles para esta plataforma son los siguientes:

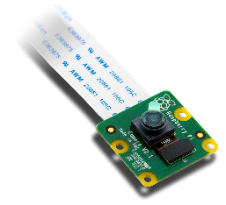
* *Cámara para Raspberry Pi V2*: Es una cámara de alta definición (HD) que se puede conectar a cualquier modelo de Raspberry para la captura de imágenes o videos en HD. Esta cámara posee un sensor de imagen IMX219PQ de Sony, el cual ofrece imágenes de video de alta velocidad y alta sensibilidad, además con enfoque fijo puede llegar a una resolución de hasta 8 megapíxeles. En la imagen (Ilustración 25 - Cámara Raspberry Pi V2) se puede apreciar esta cámara.

Ilustración - Cámara Raspberry Pi V2

* *Pantalla táctil LCD para Raspberry Pi de 7”:* Es la pantalla táctil oficial de la plataforma (Ilustración 26 - Pantalla táctil de Raspberry Pi). Se trata de una pantalla táctil LCD capacitiva multitáctil (de hasta 10 puntos de contacto). El display de 7 pulgadas posee una resolución de 800x480 píxeles con una velocidad de refresco de 60 fps ( fotogramas por segundo) y color RGB de 24 bits. Se conecta a través de una placa adaptadora que se ocupa de la conversión de potencia y señal. Sólo se requieren dos conexiones a la Pi; la de energía a través del puerto GPIO del Pi y un cable de cinta que se conecta al puerto DSI (Display Serial Interface) presente en todo modelo de Raspberry Pi.

Ilustración - Pantalla táctil de Raspberry Pi

* *Kit de Placa de prototipado de Pi de Adafruit (Adafruit Prototyping Pi Plate Kit):* Se trata de una placa que se encastra en la parte superior de las Raspberry Pi, en la cual se pueden soldar componentes en su área de GPIO (entrada/salida de propósito general) y además cuenta en su centro con un área de protoboard[[13]](#footnote-9). En la imagen (Ilustración 27 - Adafruit Prototyping Pi) se puede ver esta placa empalmada sobre una Raspberry Pi

Ilustración - Adafruit Prototyping Pi

* *Western digital Pidrive:* Es un disco rígido (Ilustración 28 - Pidrive) exclusivo para esta plataforma, de una capacidad de 314 GB, creado por la marca homónima. Cuenta con una interfaz de conexión USB para comunicarse con la Raspberry Pi.

Ilustración - Pidrive

* *Pi TFT:* Es una pequeña pantalla táctil de 2.8 pulgadas del tipo resistiva (Ilustración 29 - Pi TFT), que se encastra en la parte superior del Raspberry. Su resolución es de 320x240 y color de 16 bits. Se le pueden soldar 4 botones de forma opcional para su manipulación.

Ilustración - Pi TFT

## **4.7 ¿Por qué elegir Raspberry Pi?**

Al igual que lo que se mencionó en el capítulo 3 con respecto a Arduino, la plataforma Raspberry Pi presenta una serie de ventajas, con respecto a otras arquitecturas similares, que se describen a continuación:

* **Comunidad**: Existe una vasta comunidad en variadas partes del mundo que trabaja, da soporte y utiliza esta plataforma para diversos proyectos[[14]](#footnote-10), que dado esto, se expanden con él tiempo. A su vez, como se mostró en el apartado anterior, se cuenta con una serie de accesorios que facilitan su uso.
* **Bajo costo**: Como se mencionó con anterioridad, esta SBC se puede conseguir a un bajo costo teniendo en cuenta las prestaciones que posee.
* **Desarrollada con finalidad educativa**: Como ya se comentó anteriormente, según sus creadores, esta plataforma fue desarrollada con fines educativos y existe una comunidad que constantemente aporta lo necesario para trabajar con ella en el aula.
* **Interfaces y GPIO:** Cuenta con una variedad de interfaces para la conexión de distintos periféricos (HDMI, USB, Ethernet, Wifi, Bluetooth) y a su vez, los modelos más actuales (la versión 3), vienen con 40 pines del tipo GPIO, lo que lo convierte en un SBC muy versátil en cuanto a su utilidad.
* **Prestaciones**: Explicado todo lo anterior en este capítulo, podemos concluir que esta plataforma cumple con las prestaciones necesarias pretendidas en el desarrollo de esta tesina.

# Capítulo 5 - Aplicaciones Móviles

## **Introducción**

Como veremos en este capítulo, las aplicaciones móviles se encuentran distribuidas en distintos sistemas operativos como Android, iOS, Windows, entre otros en menor medida. Estos aplicativos permiten interactuar en un mundo en los que los dispositivos móviles toman mayor relevancia en el mercado de Internet, por la alta demanda de usuarios. Entre la diversidad de aplicaciones se encuentran las nativas, las webs y las híbridas. Cada uno de éstas, cuentan con distintas metodologías de desarrollo, pero enfocadas a un mismo paradigma. Por un lado, HTML5 como tecnología emergente empuja al desarrollo de aplicaciones web y por otro, las apps nativas sobre Android y IOS (principalmente) reinan en rendimiento. La brecha entre las ventajas y desventajas entre las App Nativas y las webs, genera mayor interés sobre las App Híbridas, tomando ventajas sobre cada partido. Con el advenimiento de tecnologías cómo Cordova, IntelXDK, Ionic y la salida de las sombras del HTML5, la comunidad de desarrolladores comienza a apostar a estos framework que se apoyan en tecnologías de Front-End para el desarrollo de aplicaciones móviles. [[15]](#footnote-11)

## **5.1 ¿Qué son las aplicaciones móviles?**

“Una aplicación móvil es un programa que se puede descargar y ejecuta desde un dispositivo móvil”

## **5.2 Las App’s**

“Una aplicación móvil o *App*es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permite al usuario efectuar una tarea concreta de cualquier tipo: profesional, de ocio, educativas, de acceso a servicios, etc; Facilitando las gestiones o actividades a desarrollar”

Al ser aplicaciones residentes en los dispositivos están escritas están escritas mayormente en Java (Android), Objective-C (IOS) y Windows Phone (C#). Su funcionamiento y recursos se encaminan a aportar una serie de ventajas tales como:

Ilustración - Aplicaciones móviles

* Un acceso más rápido y sencillo a la información necesaria sin necesidad de los datos de autenticación en cada acceso.
* Un almacenamiento de datos personales que, a priori, es de una manera segura. Esto quiere decir (Android) que utiliza un sistema de permisos heredado de Linux y conocido como la Sandbox. Este sistema limita en gran medida el acceso al sistema de archivos e impide que los procesos puedan acceder a los recursos de otros procesos, como la memoria y la CPU. Sin embargo, existen procesos con privilegios elevados que pueden acceder a recursos del dispositivo sin problemas, principalmente servicios del sistema y aplicaciones preinstaladas en los dispositivos.[[16]](#endnote-5) Los permisos son declarados en un archivo denominado AndroidManifest.xml . Estos permisos definen los recursos del dispositivo que van a utilizar y que el usuario debe aceptar antes de instalarlas, como el acceso a la cámara, leer o escribir en el almacenamiento externo, el acceso a internet, etc. Android impide que las apps utilicen recursos no manifestados, provocando un fallo en las mismas o su cierre, aunque esto solo sucede durante el desarrollo de las aplicaciones y no en su distribución.
* Una gran versatilidad en cuanto a su utilización o aplicación práctica. Dada la facilidad que presenta a la hora de la utilización del usuario, dado que necesita instalar la aplicación y mantenerla actualizada. No necesita de un navegador para poder acceder
* La atribución de funcionalidades específicas.Esto se debe a que las app nativas acceden a recursos hardware con mayor facilidad (cámara, contactos, memoria, notificaciones push, etc.)
* Mejorar la capacidad de conectividad y disponibilidad de servicios y productos (usuario-usuario, usuario-proveedor de servicios).

### 5.2.1 Las Web Apps

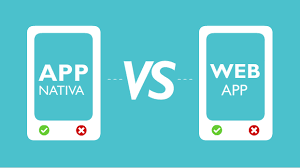
****Una Web App es una versión de una página web optimizada y adaptable a un grán número de dispositivos móviles independientemente del sistema operativo que utilice. Esta optimización es posible gracias al características provistas por lenguaje de marcado HTML5, combinado con hojas de estilo e cascada CSS3, que permiten proveer adaptabilidad, denominada en inglés “*Responsive Web Design*”. El diseño web responsivo es una filosofía de diseño y desarrollo donde el objetivo es adaptar la apariencia de las páginas webs al dispositivo(visualización) que se utiliza para visitarlas. Se caracteriza porque los layout y los contenidos multimediales son fluidos y se utiliza código media-queries de CSS3. [[17]](#footnote-12)

Ilustración - App nativa vs Web App

### 5.2.2 Ventajas de las Web-App:

* No ocupa espacio de memoria de almacenamiento en los dispositivos (no es una App).
* No requiere actualizaciones ya que al ser una página web siempre se accede a la última versión.
* No consume recursos dado que no instala servicios en segundo plano y además no consume espacio dado que no es necesario instalar la aplicación para su uso.
* En líneas generales la implementación de una Web App es más económica que el de una App.

### 5.2.3 Desventajas de las Web-Apps

* No permite la promoción y distribución a través de los *markets* (Google Play, Nokia Store, App Store, Windows Phone Apps)
* Requiere de una conexión entre el cliente y el servidor (por ejemplo por internet o una WLAN).
* Menor usabilidad, al ofrecer un acceso muy limitado a los elementos y capacidades hardware del dispositivo. (geolocalización, notificaciones “push”).
* Carece de un icono de lanzamiento específico.
* Necesitan de un espacio web.
* No funcionan en segundo plano (multitarea)

Ilustración – WebApps – Diseño multipropósito

## **5.3 Sistemas operativos para dispositivos móviles**

Un sistema operativo móvil o SO móvil es un conjunto de programas de bajo nivel que permite la abstracción de las peculiaridades del hardware específico del teléfono móvil y provee servicios a las aplicaciones móviles, que se ejecutan sobre él. Al igual que los PCs que utilizan Windows, Linux o Mac OS, los dispositivos móviles ejecutan sistemas operativos como Android, iOS o Windows Phone, entre otros.

A medida que los teléfonos móviles crecen en popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia. La cuota de mercado de sistemas operativos móviles en el primer trimestre de 2016 era el siguiente:

* Android 84,1 %
* iOS 14,8 %
* Windows Phone 0,7 %
* BlackBerry OS 0,2 %
* Otros 0,2 %

Android tiene la mayor cuota, desde enero 2011, con más de la mitad del mercado, experimentó un creciente aumento y en solo dos años (2009 a comienzos de 2011) ha pasado a ser el SO móvil más utilizado.

Es por esto, que en principio se pensó desarrollar una App para operar el SAR para esta plataforma.

## **5.4 Android**

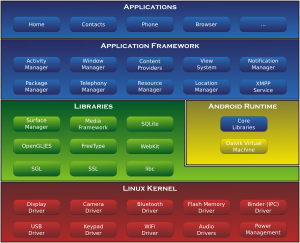
Es sin duda el líder del mercado móvil en sistemas operativos, Se encuentra basado en Linux, diseñado originalmente para cámaras fotográficas profesionales, luego fue vendido a Google y modificado para ser utilizado en dispositivos móviles como los teléfonos inteligentes y luego en tablets. Actualmente se encuentra en desarrollo para usarse en netbooks y PCs. Debido a la gran variedad de dispositivos que ejecutan Android, la *Open Handset Alliance*, compuesta por 84 compañías de hardware, software y telecomunicaciones, se dedicada al desarrollo de estándares abiertos para celulares, ayudado en gran medida a la masificación del SO de Google, hasta el punto de ser usado por empresas como HTC, LG, Samsung, Motorola entre otros.

Ilustración - Arquitectura de Android



Las aplicaciones para Android se escriben y desarrollan en Java, aunque con unas APIs propias, por lo que las aplicaciones escritas en Java para PC y demás plataformas ya existentes no son compatibles con este sistema.

Ilustración - Logo de Android

## **5.5 Aplicaciones móviles multiplataforma**

### 5.5.1 Diferencias entre aplicaciones y web móviles

Las aplicaciones nativas y comparten la pantalla del teléfono con las webs móviles conviven en mismo dispositivo. Mientras las primeras tienen que ser descargadas e instaladas antes de usar, a una web puede accederse simplemente usando Internet y un navegador compatible con la aplicación web. Pero, no todas pueden verse correctamente desde una pantalla generalmente más pequeña que la de un ordenador de escritorio.

Las que se adaptan especialmente a un dispositivo móvil como se mencionó anteriormente se denominan “web responsiva” (5.2.1 Las Web Apps) y son ejemplo del “diseño líquido” (adaptativo), debido a que el contenido toma la forma del contenedor. , mostrando la información según sea necesario.

Antes de la existencia del CSS3, como no había tecnología para poder crear sitios elásticos que se vieran bien en cualquier dimensión de pantalla, los desarrolladores de web estaban obligados a crear diferentes versiones de las páginas web. Gracias a las utilidades de CSS3 como las queries, las páginas pueden reaccionar ante distintas circunstancias como la anchura de la pantalla, ventana donde están aquellos elementos a maquetar. Web responsive se denomina a todas aquellas técnicas (no solamente redimensionado de pantalla) que permiten la adaptabilidad del contenido a los dispositivos terminales.

### 5.5.2 App Nativas

Una App nativa es aquella que se desarrolla de forma específica para un determinado sistema operativo, utilizando un *Software Development Kit*o SDK. Cada una de las plataformas, Android, iOS o Windows Phone, tienen un SDK diferente, por lo que si se desea que una App esté disponible en todas las plataformas se deberán de crear varias versiones, cada una con el lenguaje de la plataforma.

* Las apps para iOS se desarrollan con lenguaje Objective-C o Swift.
* Las apps para Android se desarrollan con lenguaje Java o Kotlin
* Las apps en Windows Phone se desarrollan en C# o lenguajes *managed* que se ejecuten sobre el CLR de .Net.

Cuando hablamos de desarrollo móvil casi siempre nos estamos refiriendo a Apps nativas. La principal ventaja con respecto a las app webs y app híbridas, los otros dos tipos, es la posibilidad de acceder a la totalidad de las características del hardware ofrecidas por el móvil, por ejemplo cámara, GPS, agenda de contactos, almacenamiento y la capacidad ser ejecutada sin necesidad de conectividad a internet. Además, las Apps son promocionadas por medio de las tiendas de aplicaciones, mientras que las Web Apps deben ser descargadas accedidas por el usuario a través de una URL. Las App nativas presentan un mayor rendimiento y aprovechamiento del hardware de cada smartphone o dispositivo móvil.

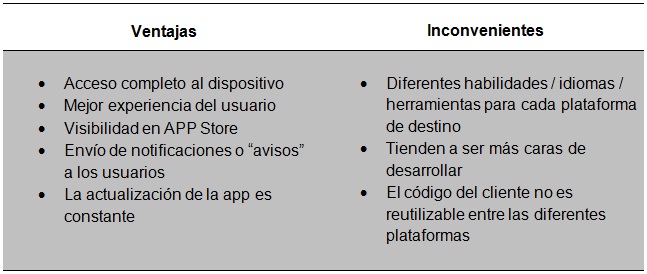


Ilustración - Cuadro comparativo nativas

### 5.5.3 Desarrollo de Web Apps

Una aplicación web o Web App se desarrollada con los lenguajes HTML, Javascript y CSS que revisten de gran popularidad en la actualidad.

En el contexto de aplicaciones móviles, su principal ventaja con respecto un desarrollo nativo es la posibilidad de programar independientemente del SO en el que se usará la aplicación. De esta forma se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL.

La diferencia mayor con una aplicación nativa es que carece del proceso de instalación, pero con la contraparte de no poder estar visibles en la tienda de aplicaciones y por ende, la promoción y comercialización debe realizarse de forma independiente. Carecen también de la capacidad de accederse desde el lanzador del dispositivo, pero en algunas plataformas, este inconveniente puede suplirse con la creación de un acceso directo o link.

Las Web Apps móviles son una opción atractiva si el objetivo es adaptar la web a formato móvil.



Ilustración - Cuadro comparativo - Aplicaciones Web

### 5.5.4 Aplicaciones Híbridas

Una aplicación híbrida es una combinación de las dos anteriores. Las apps híbridas se desarrollan con lenguajes propios de las Web Apps, es decir, HTML, Javascript y CSS, pero también dan la posibilidad de acceder a gran parte de las características del hardware del dispositivo. La principal ventaja es que es posible empaquetarla y distribuirla en la tienda de aplicaciones de cada SO.

PhoneGap es uno de los frameworks más utilizados por los programadores para el desarrollo multiplataforma de aplicaciones híbridas. Otro ejemplo de herramienta para desarrollar apps híbridas es Cordova. Tanto PhoneGap como Apache Cordova, son los frameworks más utilizados por los programadores para el desarrollo multiplataforma de aplicaciones híbridas.[[18]](#footnote-13) También existe la opción de servirse de cross-platforms frameworks, para desarrollar una web app.

### 5.5.5 Aplicación híbrida: Web App

Cosiste en diseñar la aplicación como si fuera una Web App para ser ejecutada en el propio navegador del cliente. Sin embargo, ésa es precisamente su gran virtud, pero también su principal inconveniente. Nos encontramos con que el código es válido para todos los dispositivos, pero tanto la experiencia de usuario y como la apariencia no están a la par de una App nativa. Phonegap es el framework más popular en esta tecnología.

### 5.5.6 Aplicación híbrida: app interpretada

La aplicación interpretada significa que la aplicación es programada y luego cada terminal la traduce a su propio lenguaje de programación. Facilita el desarrollo de aplicaciones y reduce el esfuerzo considerablemente. Aunque el resultado no es idéntico a la nativa, la apariencia es bastante buena, y en muchas ocasiones puede ser la solución al problema del desarrollo de aplicaciones multiplataforma.

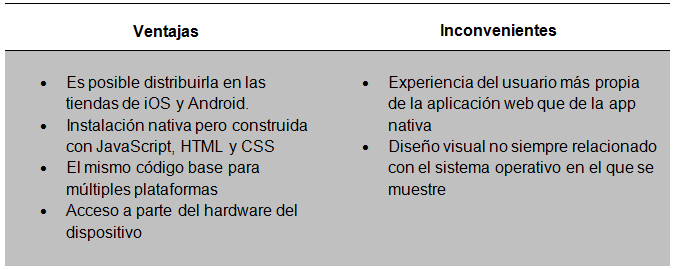


Ilustración - Comparativa aplicaciones híbridas

## **5.6 Investigación en Desarrollo de Apps y WebApps**

Al momento de seleccionar el tipo de App o Web App se investigó en las distintas tecnologías para seleccionar la adecuada

Ilustración - Herramientas para desarrollo de apps

### 5.6.1 Android Studio

Android Studio es el IDE oficial para el desarrollo de aplicaciones para Android, basado en la tecnología IntelliJ IDEA . Además del editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece la posibilidad de instalarse en los distintos sistemas operativos como UN/Linux, MS Windows, MacOs. Además incluye características[[19]](#footnote-14) como:

* Integración de ProGuard(reducción de código, eliminación de atributos, clases, métodos sin utilizar)
* Firma de aplicaciones
* Renderizado en tiempo real
* Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso
* Editor de diseño, con posibilidad de arrastrar y soltar elementos
* Herramientas Lint (detección de problemas de rendimiento, usabilidad, compatibilidad de versiones, etc)
* Soporte para crear diseños en Android Wear (Sistema operativo para dispositivos corporales de Android)
* Google Cloud Platform
* Dispositivos virtuales para simular las aplicaciones

funciones que aumentan la productividad durante la compilación de apps para Android. Provee también emuladores para diferentes plataformas de hardware, destinados a la prueba de Apps.

### 5.6.2 App Inventor

Es un entorno de desarrollo de software creado por Google Labs para la elaboración de aplicaciones destinadas al sistema operativo Android. El usuario puede, de forma visual y a partir de un conjunto de herramientas básicas, ir enlazando una serie de bloques para crear la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web desde la web la aplicación generada. Las aplicaciones creadas con App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.

Con Google MIT App Inventor, se espera un incremento importante en el número de aplicaciones para Android debido a dos grandes factores: la simplicidad de uso, que facilitará la aparición de un gran número de nuevas aplicaciones; y Google Play, el centro de distribución de aplicaciones para Android donde cualquier usuario puede distribuir sus creaciones libremente.

Otra gran cualidad es la posibilidad de insertarlo en la educación dado su programación por medio de bloques gráficos, que resultan ser muy intuitivos en aquellas personas que se introducen.

Unas primeras apps para el SAR fueron realizadas bajo esta plataforma, de dicha experiencia se pudo concluir que tiene una baja curva de aprendizaje.

5.6.3 Tecnologías del lado del cliente - *Open Web Stack* (HTML, CSS y JS)

### 5.6.3.1 HTML

 Es un [lenguaje de marcado](https://es.wikipedia.org/wiki/Lenguaje_de_marcado) para la elaboración de [páginas web](https://es.wikipedia.org/wiki/P%C3%A1gina_web). Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.

### 5.6.3.2 CSS

Como se mencionó anteriormente (Sección 5.5.1), CSS (Hoja de estilos en cascada) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

### 5.6.3.3 JS

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,​ basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, ​ aunque existe una forma de JavaScript del lado del servidor.

### 5.6.3.4 ¿Qué es SASS?

Es un metalenguaje de estilos en cascada. Es un lenguaje que preprocesado produce CSS, denominado SassScript. Existen dos formatos SCSS (Sintaxis con bloques) y SASS (Sintaxis con indentación). CSS3 consiste en una serie de selectores y pseudo-selectores que agrupan las reglas que son aplicadas.

SassScript proporciona los mecanismos necesarios para ofrecer variables, nesting (anidamientos), mixins, y herencia de los selectores.

Las variables Permite reutilizar valores que podemos manejar desde un solo sitio de forma sencilla y centralizada

Un mixin permite aprovechar un trozo de nuestro código al que podemos llamar repetidamente y que encapsula en una o más líneas de código.

La sintaxis de .sass y .scss no puede ser interpretada directamente por los navegadores, por ende es necesario la compilación.

### 5.6.3.5 Angular JS

Es un framework MVC (Model, View, Controller) de JavaScript para el Desarrollo Web Front End que permite crear aplicaciones SPA Single-Page Applications (una única página).

### 5.6.3.6 MVC

El MVC (Model-View-Controller o Modelo-Vista-Controlador), es un patrón de diseño que separa los datos, la lógica y las interfaces de usuario. Como su nombre indica, está separado en tres componentes: Modelo, Vista y Controlador. Está basado en la ideología de separación de conceptos.

* Modelo: Es la capa encargada de los datos, es decir, la que se encarga de hacer peticiones a las bases de datos para enviar o recibir información.
* Vista: Se trata del código que nos permitirá presentar los datos que el modelo nos proporciona.
* Controlador: Es la capa que sirve de enlace entre la vista y el modelo. Envía comandos al modelo para actualizar su estado, y a la vista correspondiente para cambiar su presentación.

### 5.6.4 Cordova

Apache Cordova es un entorno de desarrollo de aplicaciones móviles, originalmente creado por Nitobi y comprado por Adobe [[20]](#footnote-15). Más tarde fue liberado como Apache Cordova. Permite, construir aplicaciones para dispositivos móviles utilizando CSS3, HTML5, y Javascript. Las aplicaciones resultantes son híbridas, lo que significa que no son ni una aplicación móvil nativa o App (porque toda la representación gráfica se realiza vía vistas Web en vez del framework de UI nativo) ni puramente basadas en web (porque no son solo aplicaciones web, sino que están empaquetadas como aplicaciones para su distribución y tienen acceso a las APIs nativas del dispositivo).

### 5.6.5 Intel XDK

Es un kit de desarrollo creado por Intel para crear aplicaciones nativas para los teléfonos celulares y las tabletas que utilizan tecnologías web como HTML5, CSS y JavaScript. Las aplicaciones se compilan mediante un servicio on-line. Hace uso de la plataforma Cordova para crear aplicaciones cross-platform, enfocado en el segmento de Apps para IoT. Posee un emulador, pervisualización de apliciones mediante el scan de un código QR, Drag and Drop y soporte de plantillas. Brinda soporte para Android, iOS, Windows Phone, entre otras plataformas.

### 5.6.6 Ionic

Es una Framework, Open Source y de distribución gratuita, para el desarrollo de aplicaciones híbridas, inicialmente pensado para móviles y tablets, basadas en HTML5, CSS y JS. Está construido con Sass y optimizado con AngularJS.

### 5.6.6 Meteor

Es una plataforma para crear aplicaciones web en tiempo real construida sobre Node.js. Meteor se localiza entre la base de datos de la aplicación y su interfaz de usuario y se encarga que las dos partes estén sincronizadas.

Meteor puede compartir código JavaScript entre el cliente y en el servidor.

### 5.6.7 Meteor y Cordova

Existe una integración del framework Meteor con Cordova, que permtite que una aplicación web crada con Meteor, sea ejecutarla en un dispositivo iOS o Android de forma híbrida, es decir, se ejecuta en una vista web incrustada en una aplicación nativa en lugar de en un navegador móvil del usuario. Un beneficio importante de empaquetar su aplicación web como una aplicación de Cordova es que todos sus recursos que erían descargados desde la web se incluyen con la aplicación, asegurando una velocidad de carga mayor, beneficiando a los usuarios con conexiones lentas. Otra característica es la compatibilidad con *hot code push*, que le permite actualizar la aplicación en los dispositivos de los usuarios sin pasar por el proceso habitual de revisión de la tienda de aplicaciones. Cordova también permite el acceso a ciertas características nativas a través de una arquitectura de complementos. Los complementos permiten utilizar funciones que normalmente no están disponibles para aplicaciones web, como acceder a la cámara del dispositivo o al sistema de archivos local, interactuar con lectores de código de barras o NFC.

# Capítulo 6 – Stack MEAN

## **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[[21]](#footnote-16)) 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[[22]](#footnote-17) y es llamada LibUV (Unicornio Velocirraptors).

Ilustración 2 - Logo del motor V8

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).

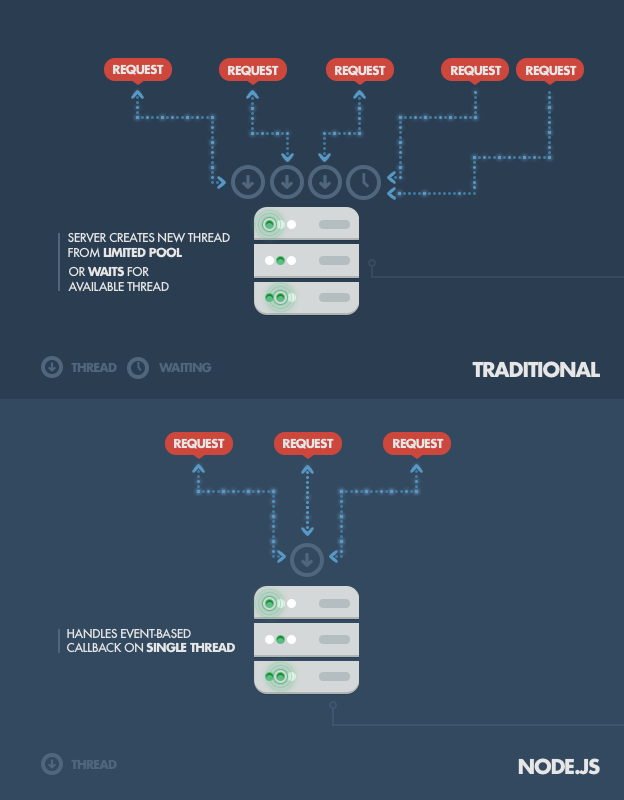


Ilustración Comparativa de servidores tradicionales y Nodejs

## **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.

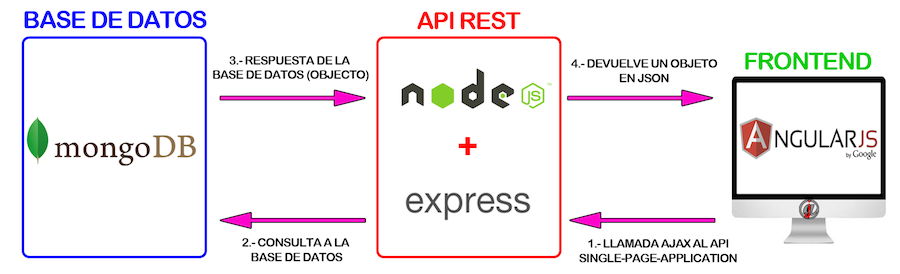


Ilustración 33 - Arquitectura de interacción MEAN

### 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.

Ilustración 4 - Logo de JSON

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

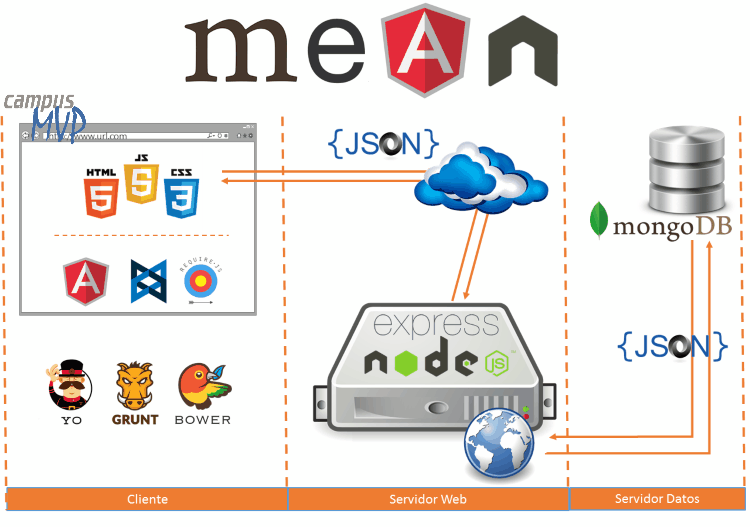


Ilustración 35 - Json pegamento de tecnologías

# Capítulo 7 – Comunicación NodeJS con Arduino

Este capítulo introduce sobre un framework denominado Johnny-five el cual es utilizado para la comunicación entre la aplicación web y las placas Arduinos que componen al SAR. Además, se explica en detalle la utilidad de un protocolo, llamado Firmata, que es cargado en cada uno de los Arduinos.

## **7.1 ¿Qué es Johnny-five?**

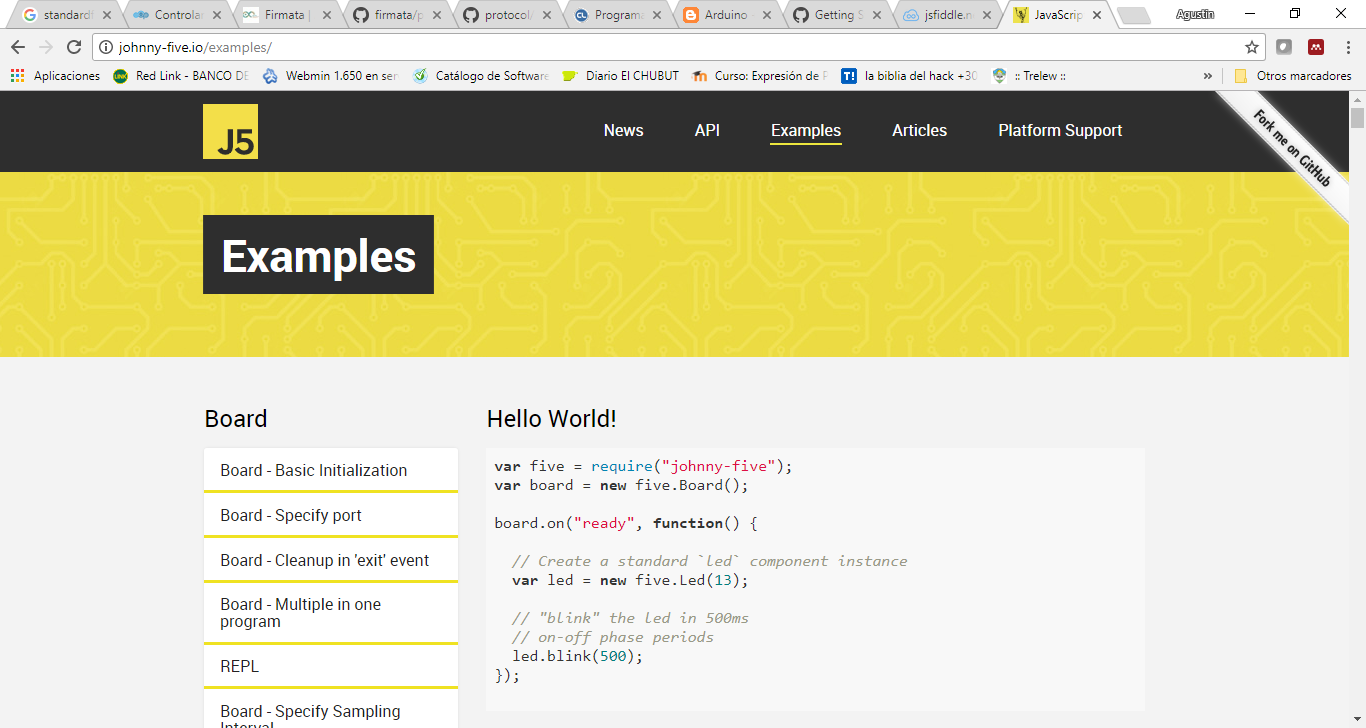
Johnny-five, más conocido como J5, es un framework de programación robótica basado en javascript lanzado por la compañía de tecnología web (Ilustración 42 - Página oficial de Johnny-Five[[23]](#footnote-18)) Bocoup en el 2012, es de código abierto, y l cuenta con una gran variedad de desarrolladores, ingenieros y colaboradores que están constantemente perfeccionando y agregando características nuevas a esta herramienta.

Ilustración - Página oficial de Johnny-Five

Es compatible con la mayoría de los microcontroladores y SBC más populares como, los utilizados en el desarrollo de esta tesina, Arduino y Raspberry Pi (ambos en todas sus versiones). Además de placas como BeagleBone, SparkFun, ChipKit, Intel-Galileo, entre otras.

Posee una librería muy completa, con compatibilidad a una vasta cantidad de sensores y actuadores, y ejemplos útiles de usos e implementación de los mismos. Dichos ejemplos se encuentran disponibles en su página oficial con el esquema de conexión de los componentes a las distintas plataformas compatibles.

## **7.2 Instalación**

Para poder utilizar Johnny-Five, se debe contar con el Node.js (visto en el apartado **¡Error! No se encuentra el origen de la referencia.**).

La instalación de este módulo es sencilla, dado que la misma se puede agregar al paquete de librerías que se obtienen al crear un proyecto de node.js. Para ello se debe ejecutar el siguiente comando, en una consola, dentro del directorio del proyecto:

npm install Johnny-five

Como se puede apreciar, se utiliza el comando npm que no es más que el gestor de módulos y aplicaciones de node.js

## **7.3 Arduino Firmata**

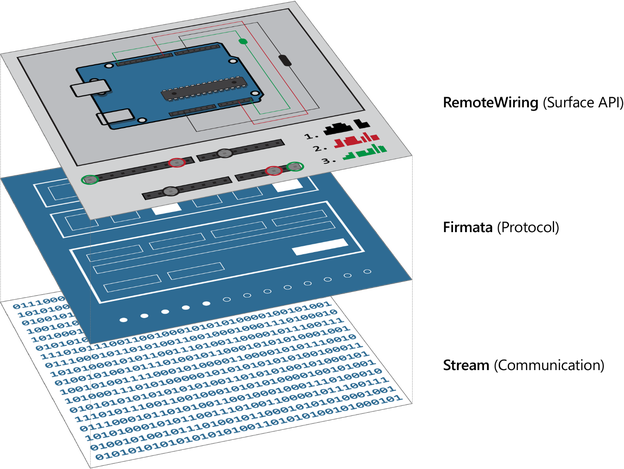


Ilustración - Remote Wiring de Windows

Firmata es un protocolo genérico utilizado para la comunicación con microcontroladores desde software instalado en una computadora. Este protocolo se puede implementar en cualquier arquitectura de microcontroladores, así como en cualquier paquete de software.

El objetivo de firmata es permitir controlar completamente un micricontrolador de forma remota (Ilustración 43 - Remote Wiring de Windows), por ejemplo Arduino, desde un programa instalado en una computadora, sin escribir código de Arduino[[24]](#endnote-6).

Ventajas:

* Nuestro programa no está limitado por la memoria RAM y Flash de Arduino
* El software de control se puede programar en cualquier lenguaje, no solo C++, siempre que tenga soporte para firmata. Por ejemplo: Firmata: Processing, Visual Basic, Perl, C#, PHP, Java, **JavaScript**, Ruby y Python.

Desventajas:

* Programas más restringidos, es posible que algunas operaciones complejas no sea posible hacerlas. Por ejemplo, el uso de interrupciones.
* El microcontrolador no es autónomo, es decir, siempre debe estar conectado al computador para poder recibir comandos.

## **7.4 Instalación Firmata**

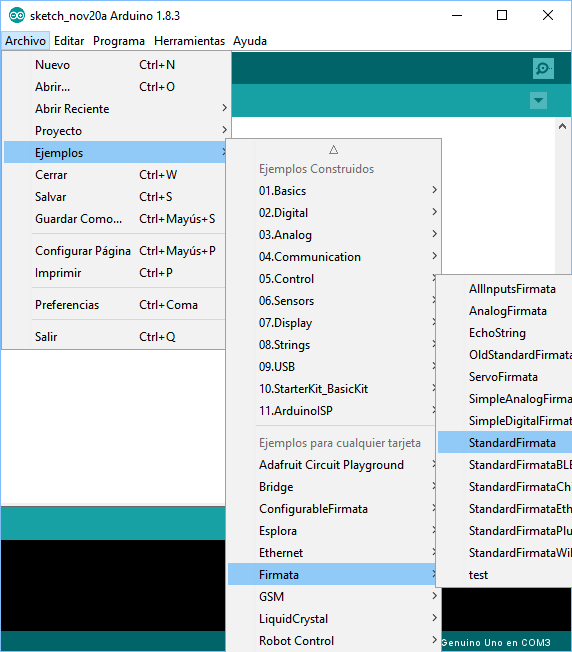
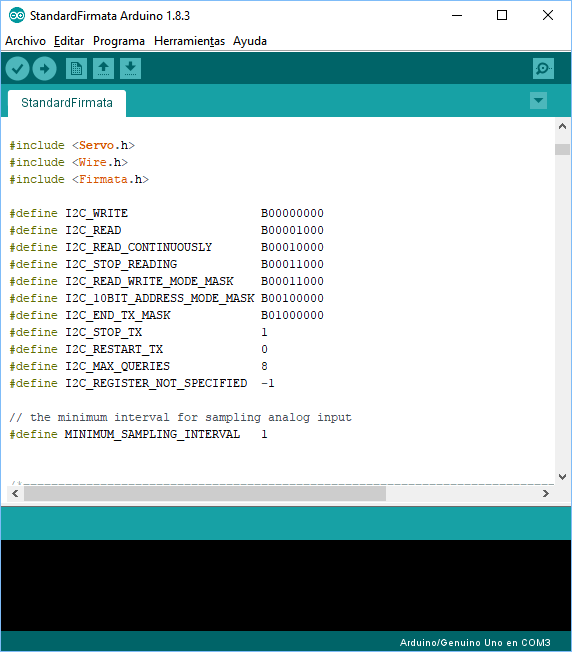
Con la instalación del IDE Arduino en una computadora se incluyen librerías y ejemplos que permiten manipular diversos componentes que se conecten a la plataforma. Dentro de estos ejemplos de códigos se encuentran los del protocolo Firmata.

Ilustración - IDE de Arduino

Para instalar Firmata se necesita tener conectada la placa Arduino a la computadora a través de un puerto USB. Luego de ello se debe seleccionar desde el IDE de Arduino (Ilustración 44 - IDE de Arduino) el código Firmata, dentro de su respectiva librería, según los dispositivos que se requieran comunicar con el mismo. Para ello se debe ir a Archivo 🡪 Ejemplos 🡪 Firmata.

En nuestro caso se utilizaron dos códigos Firmata:



**StandardFirmata**: Es, como su nombre lo indica, el estándar del protocolo que permite la comunicación con la mayoría de los componentes compatibles con Arduino. En nuestro caso, es el utilizado para cargarlo dentro del Arduino Mega para manipular la mayoría de sensores y actuadores del SAR.

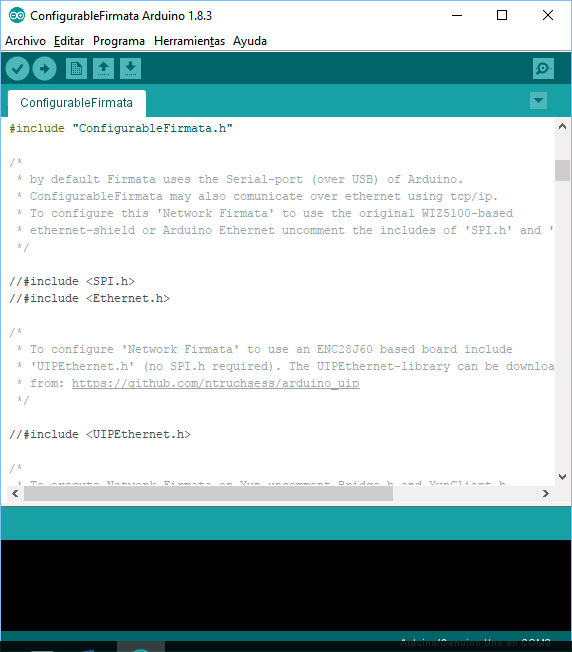


Ilustración - Código StandardFirmata

**ConfigurableFirmata**: Esta versión personalizada del protocolo[[25]](#footnote-19), es la utilizada para captar la temperatura mediante el sensor para dicho fin conectado al Arduino Nano. Permite separar las características del protocolo en clases individuales, haciendo más sencillo mezclar las características estándar del protocolo con otras personalizadas.

Ilustración - Código ConfigurableFirmata

# 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[[26]](#endnote-7). 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[[27]](#footnote-20) 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[[28]](#footnote-21) 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).

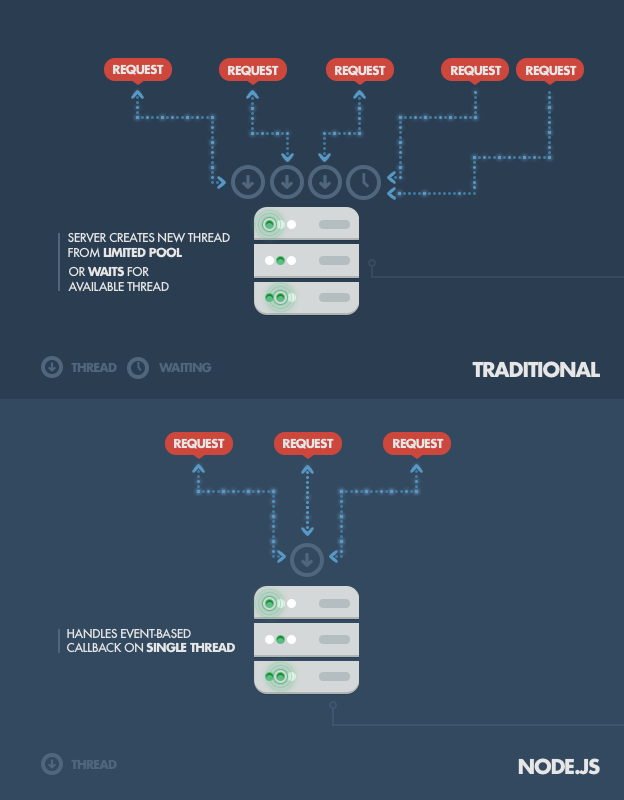


Ilustración 40 Comparativa de servidores tradicionales y Nodejs

## **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.

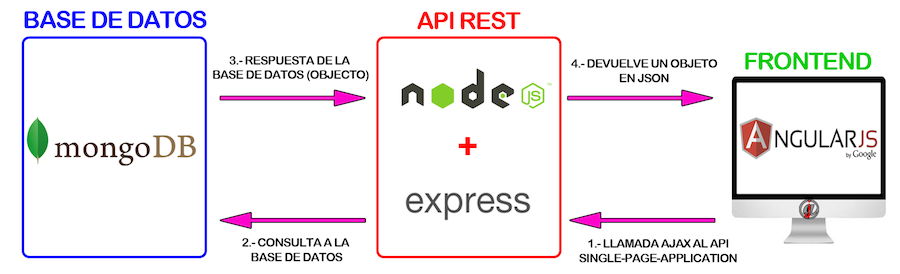


Ilustración 33 - Arquitectura de interacción MEAN

### 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”

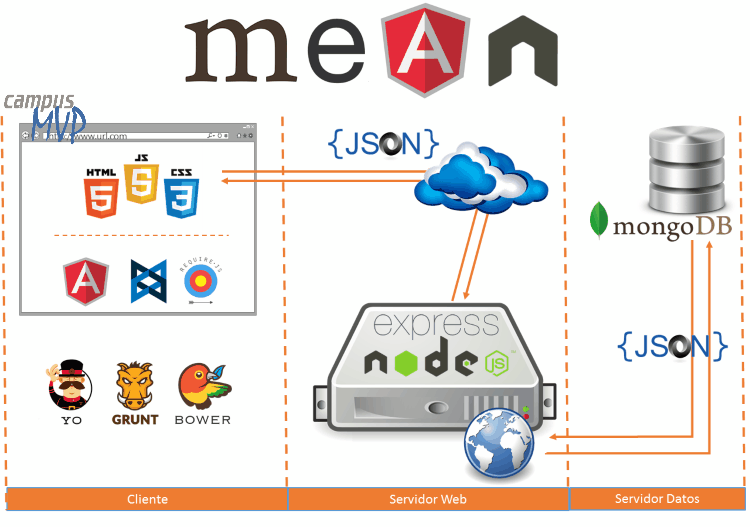


Ilustración 35 - Json pegamento de tecnologías

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.

# Capítulo 9 – Arquitectura y Ensamblado del SAR

Para el desarrollo de esta tesina se procedió a armar un prototipo del SAR mediante un robot móvil, el cual cuenta con una variedad de actuadores y sensores que le permiten interactuar con el entorno que lo rodea. En este capítulo, se describen los diversos componentes del SAR, con sus respectivas funciones dentro del mismo, así como también, los procedimientos que se llevaron a cabo para construirlo. Se muestran los distintos niveles en los cuales se estructura y como se disponen los elementos dentro de él. En la siguiente imagen (Ilustración 47 - Esquema de conexión de componentes) se puede apreciar la arquitectura de conexión de los componentes que integran al SAR.

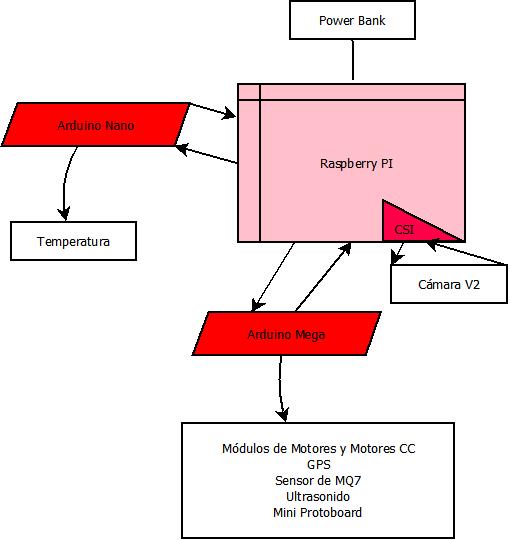


Ilustración - Esquema de conexión de componentes

## **9.1 Componentes**

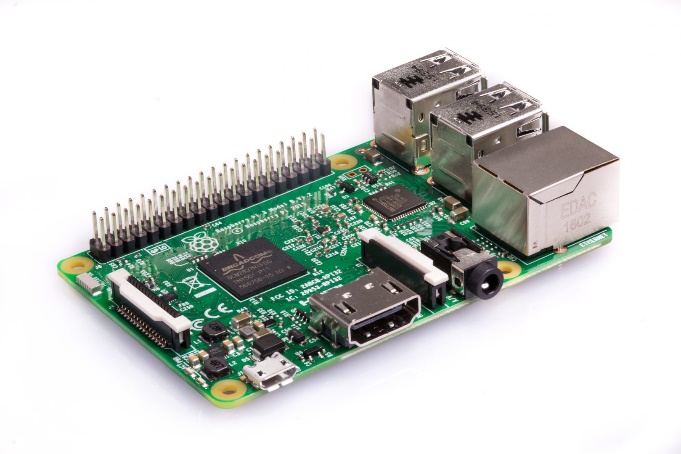
**Una Raspberry Pi 3 model B**: Componente principal del SAR, es el servidor del mismo, encargado de almacenar la aplicación web y recibir las peticiones de los clientes para luego mandar las ordenes a las placas Arduino. Cuenta con una tarje microSD donde almacena el sistema operativo Raspbian el cual se ejecuta al encenderla y permite correr la aplicación desarrollada.

Ilustración - Raspberry Pi 3

**Un Arduino Mega**: Es el principal controlador del SAR, en él se conectan todos los sensores y actuadores (a excepción del sensor de temperatura). Funciona como intermediario entre la Raspberry y el resto de los componentes, dado que recibe todas las ordenes de ejecución de la misma. En su memoria, se encuentra almacenada una versión del protocolo Firmata nombrada como StandarFirmata (dado por la librería Firmata de Arduino, ***Anexo X***) necesaria para establecer la comunicación con los comandos enviados desde Javascript por la aplicación web.

Ilustración - Arduino Mega

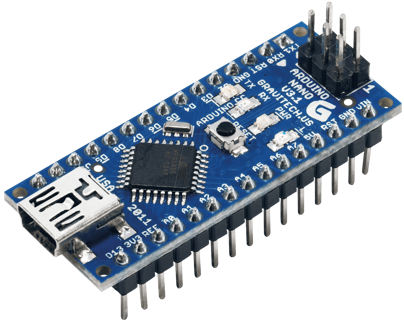
**Un Arduino Nano**: Esta versión de Arduino es la que se encarga de capturar la temperatura obtenida por el sensor DS18B20. Se debió optar por el uso de otro Arduino, dado que para la captura de temperaturas y el envío de los datos a la Raspberry mediante JavaScript se necesita una versión particular del protocolo Firmata, nombrada como ConfigurableFirmata (***Anexo X1***).

Ilustración - Arduino Nano

**Cuatro motores DC (corriente continua de 3v a 6v) con caja reductora:** Estos motores, en conjunto con cuatro ruedas de plástico cubiertas con una goma cada una, son los que permiten darle la movilidad al SAR.

Ilustración - Motores CC

**Tres sensores ultrasónicos HC-SR04**: Los sensores ultrasónicos, se utilizan para determinar la presencia de algún objeto a una distancia menor a 20 centímetros, tanto al frente del SAR como en sus laterales. Al identificar un objeto a una distancia menor a la mencionada, se bloquea el avance del robot en la dirección en donde se encuentre dicho objeto.

Ilustración - Sensor de ultrasonido

**Dos portas pilas AA x4 con sus respectivas pilas recargables:** Utilizados para alimentar de corriente eléctrica a los 4 motores.

Ilustración - Porta pilas

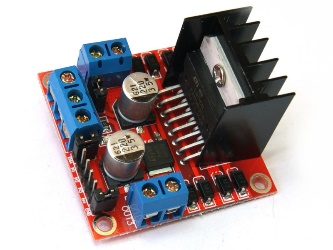
**Dos puentes H L298N:** Son los intermediarios entre el Arduino Mega y los motores, cada uno de ellos se encarga de la manipulación de dos motores.

Ilustración - Módulo Puente H



**Una mini Protoboard:** Utilizada como extensión de pines, más que nada para los pines GND y 5v de la placa Arduino Mega.

Ilustración - Mini-protoboard

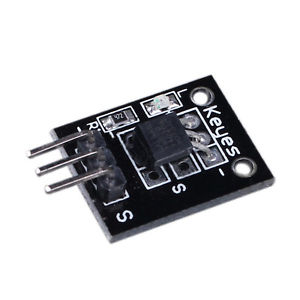
**Un sensor de temperatura DS18B20 montado sobre una placa KY-001:** Este módulo es el encargado de sensar la temperatura, se encuentra conectado al Arduino Nano.

Ilustración - Sensor de Temperatura

**Un sensor de monóxido de carbono MQ-7:** El sensor de monóxido, conectado al Arduino Mega, detecta la ausencia o presencia de dicho gas.

Ilustración - MQ7 CO

**Un GPS GY-GPS6MV2:** Con este módulo de GPS obtenemos toda la información necesaria con respecto a la Geolocalización del SAR (latitud, longitud, punto cardinal, velocidad, orientación, fecha y hora).

Ilustración - GPS

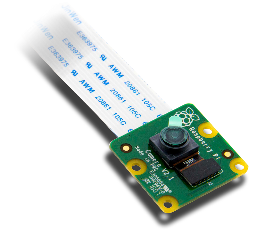
**

Ilustración - Cámara V2

**Cámara de Raspberry Pi V2:** Esta cámara, exclusiva de Raspberry, es la utilizada para captar con señal de video en tiempo real (mediante el software motion) el entorno que rodea al SAR.

**PowerBank Malibu de 20Ah con panel solar:** Funciona como batería del SAR, provee de corriente eléctrica a la Raspberry y por ende a los arduinos conectados a ella.

## 

Ilustración - Panel Solar Power Bank

## **9.2 Estructura**

### 9.2.1 Diseño

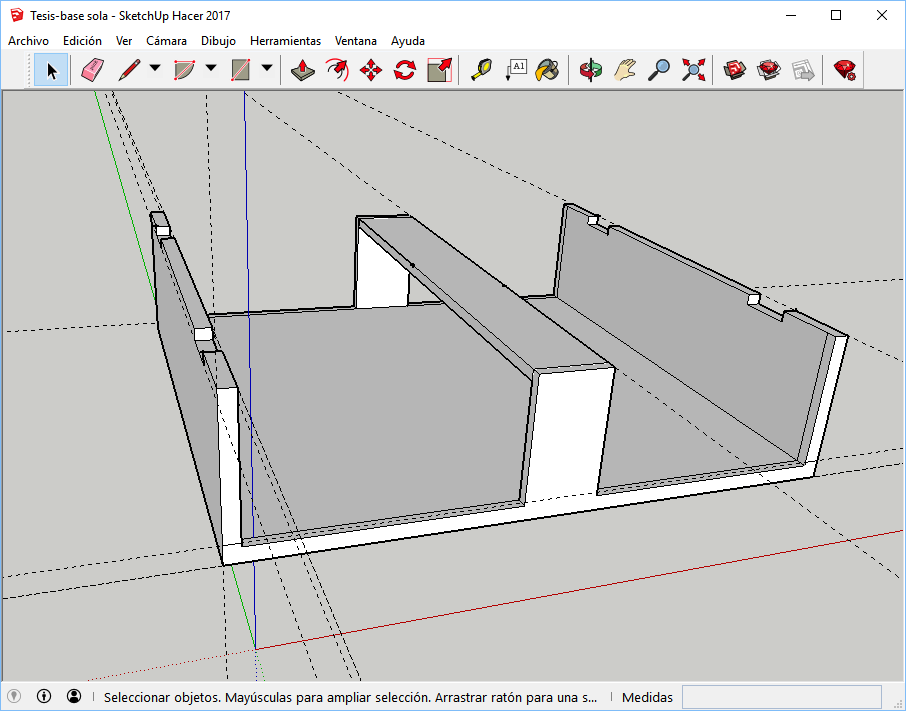
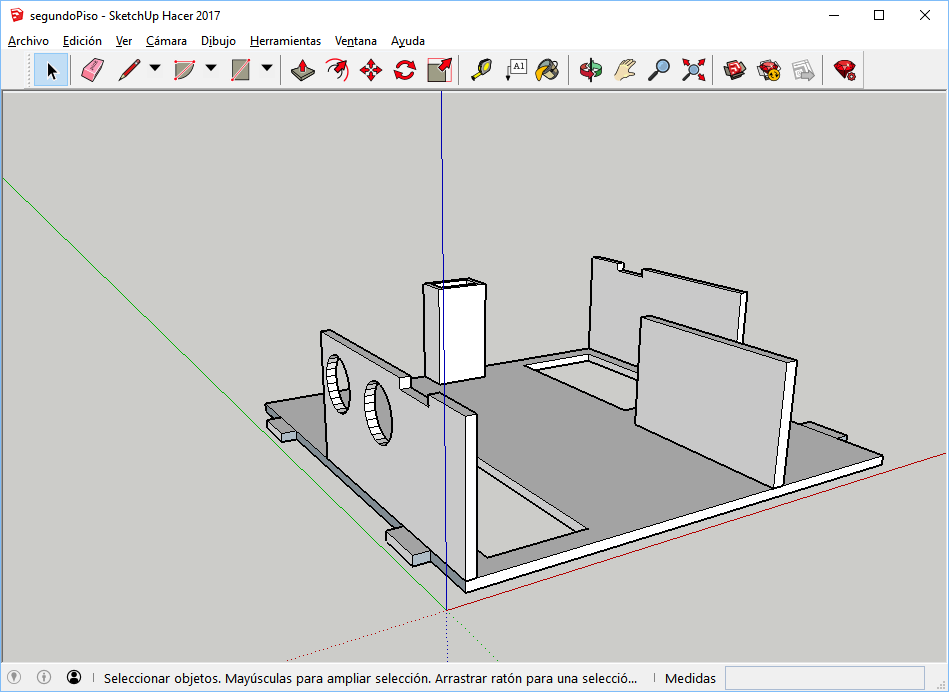
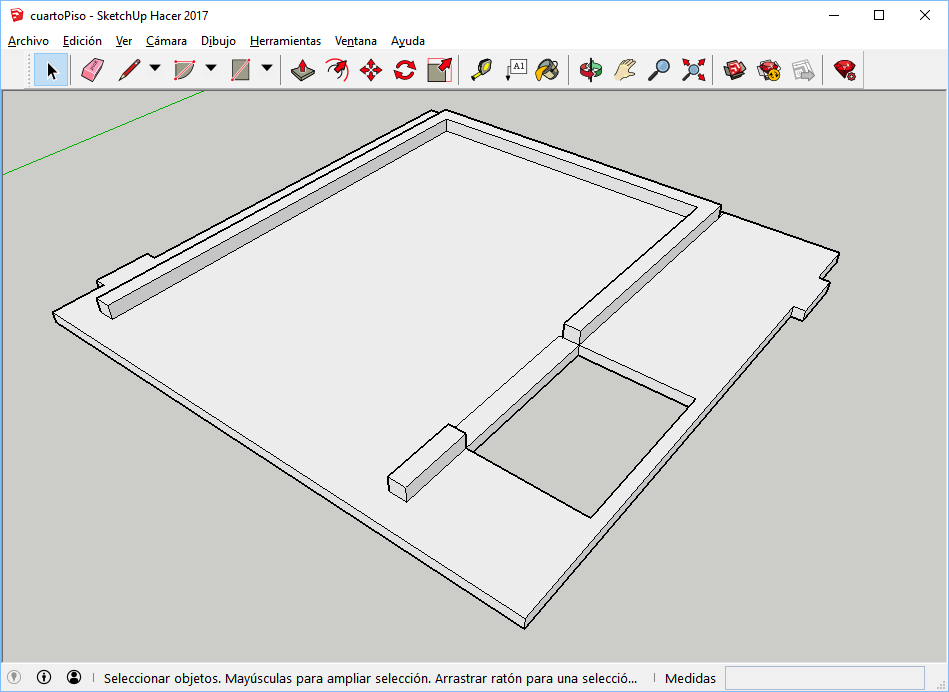
Para el armado de la estructura se procedió a diseñar en 3D las distintas piezas por medio del entorno de diseño gráfico SketchUp 2017. Se tomaron medidas de los distintos componentes y en base a ellas decidimos dividir el gabinete del SAR en cuatro niveles.

Ilustración - Diseño estructura nivel 3 con SketchUp

Ilustración - Diseño estructura nivel 4 con SketchUp

Ilustración - Diseño estructura nivel 1 con SketchUp

Ilustración - Diseño estructura nivel 2 con SketchUp

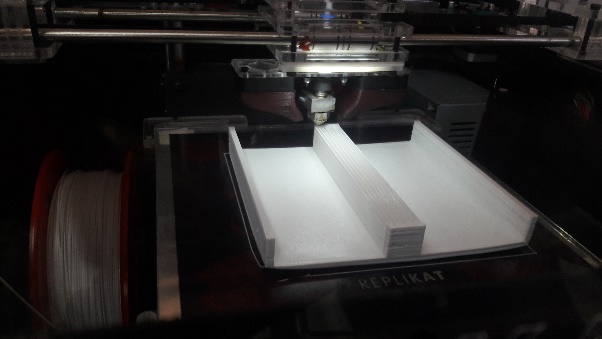
Una vez armados los modelos de los distintos niveles, fuimos imprimiendo los mismos mediante una impresora 3D.

Ilustración - Impresión 3D del nivel 1

### 9.2.2 Los 4 niveles

**Nivel 1**: El primer nivel es en donde se instalaron los motores, con distintas piezas estructurales metálicas diseñadas exclusivamente para dicha función, además se encuentran los dos puentes H L298N conectados a cada par de motores respectivamente. Cada motor cuenta con su rueda de plástico.

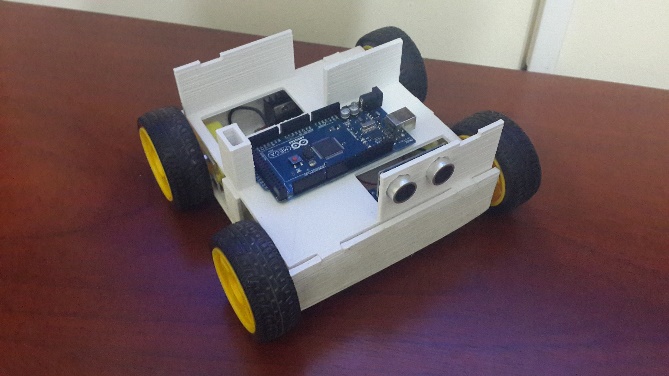
**Nivel 2**: En este nivel se adhirió con tornillos el Arduino Mega y la mini protoboard cada uno con sus respectivas conexiones, además del porta pilas utilizados para la alimentación de los motores. En su frente se colocó uno de los sensores ultrasónicos HC-SR04 que verifica la presencia de objetos en la parte delantera del SAR.

Ilustración - Nivel 2 descubierto

**Nivel 3**: En el tercer nivel se encuentra la Raspberry Pi y el Arduino Nano, en conjunto con variados sensores: 2 sensores HC-SR04, ubicados uno en cada lateral para verificar objetos en dichos lugres, 1 sensor de monóxido de carbono MQ-7 y la cámara de Raspberry en el frente.

Ilustración - RM Vista Lateral

**Nivel 4**: El nivel superior es el que se equipa con la batería portátil solar y que se conecta directamente a la Raspberry, además se encuentra a la vista el GPS y el sensor de temperatura DS18B20. Este último conectado al Arduino Nano.

## **9.3 Esquemas de conexión de componentes Arduino**

En las siguientes imágenes se pueden apreciar los esquemas básicos de conexión, a la plataforma Arduino, de los diversos actuadores, sensores y módulos que componen al SAR (antes descriptos).

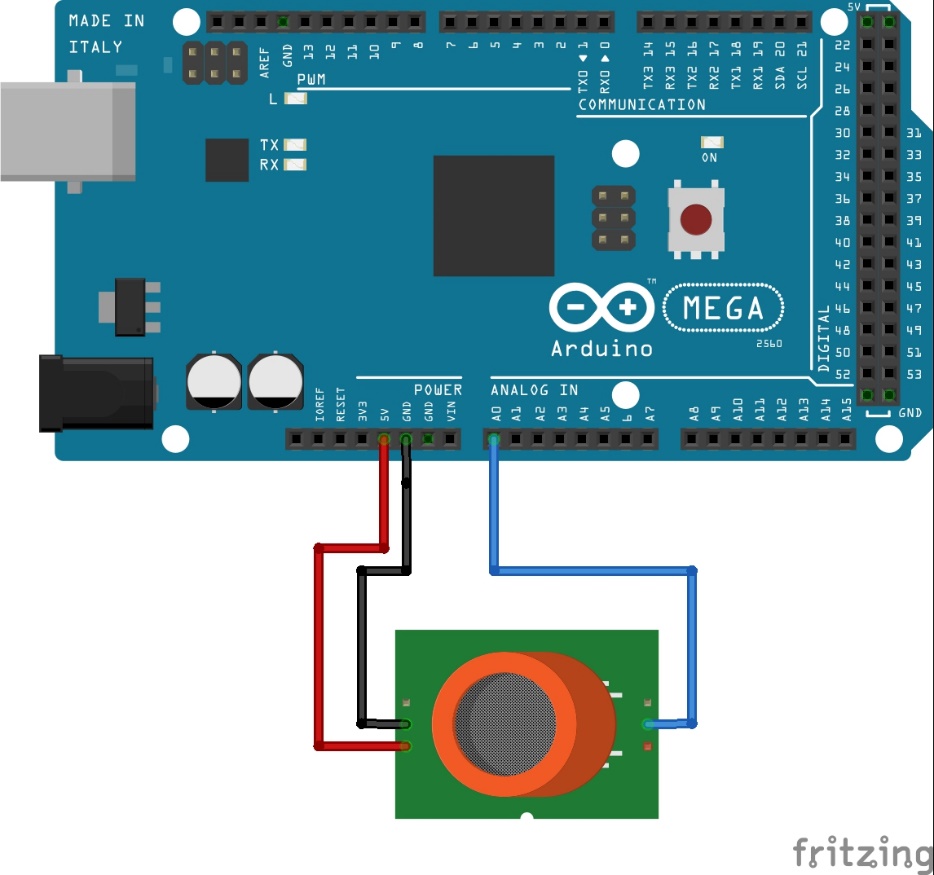


Ilustración - Esquema de conexión de sensor de monoxido MQ-7 a Arduino Mega

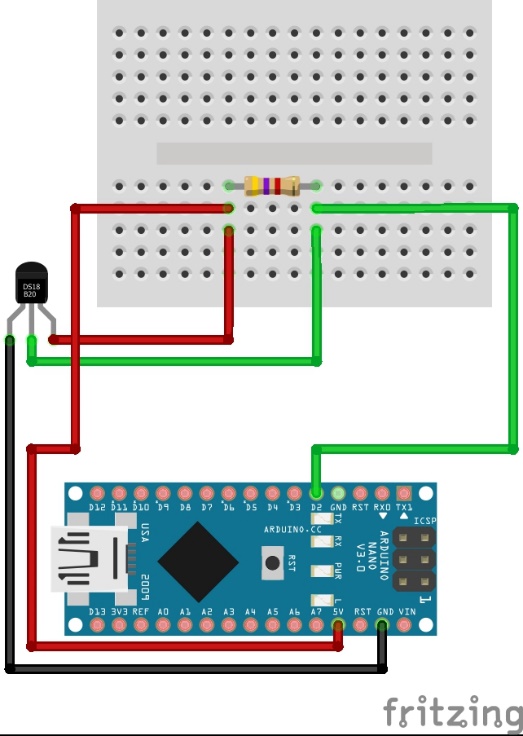


Ilustración - Esquema de conexión de sensor de temperatura DS18D20 a Arduino Nano

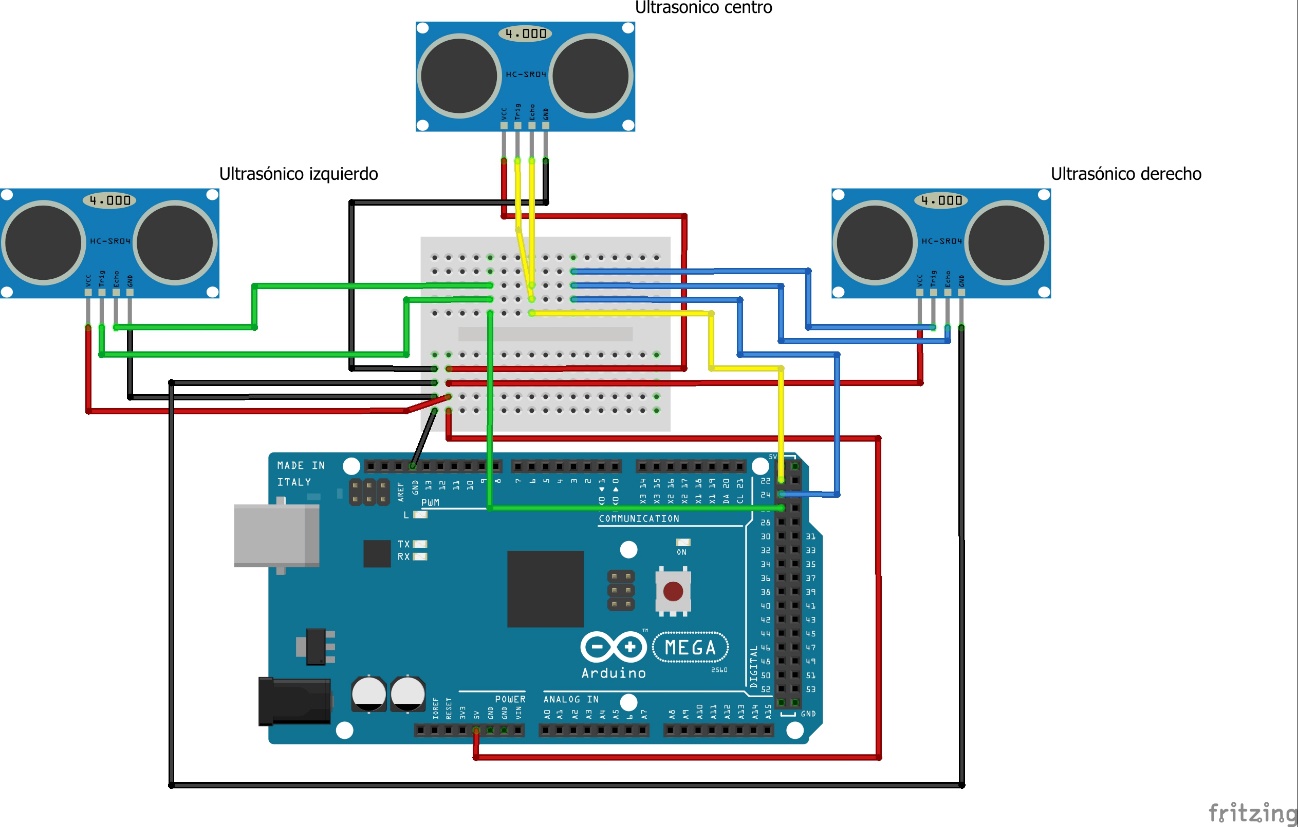


Ilustración - Esquema de conexión de sensores ultrasónicos HC-SR04 con Arduino Mega

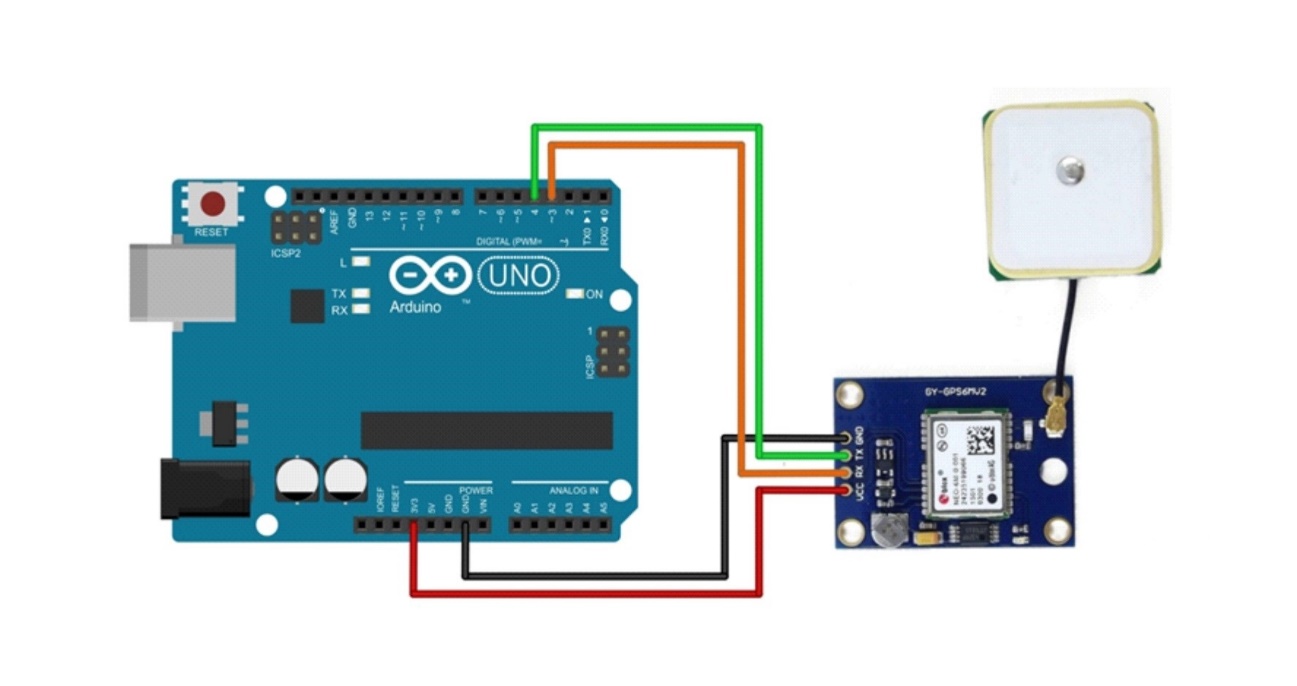


Ilustración - Esquema de conexión de módulo GPS con Arduino UNO

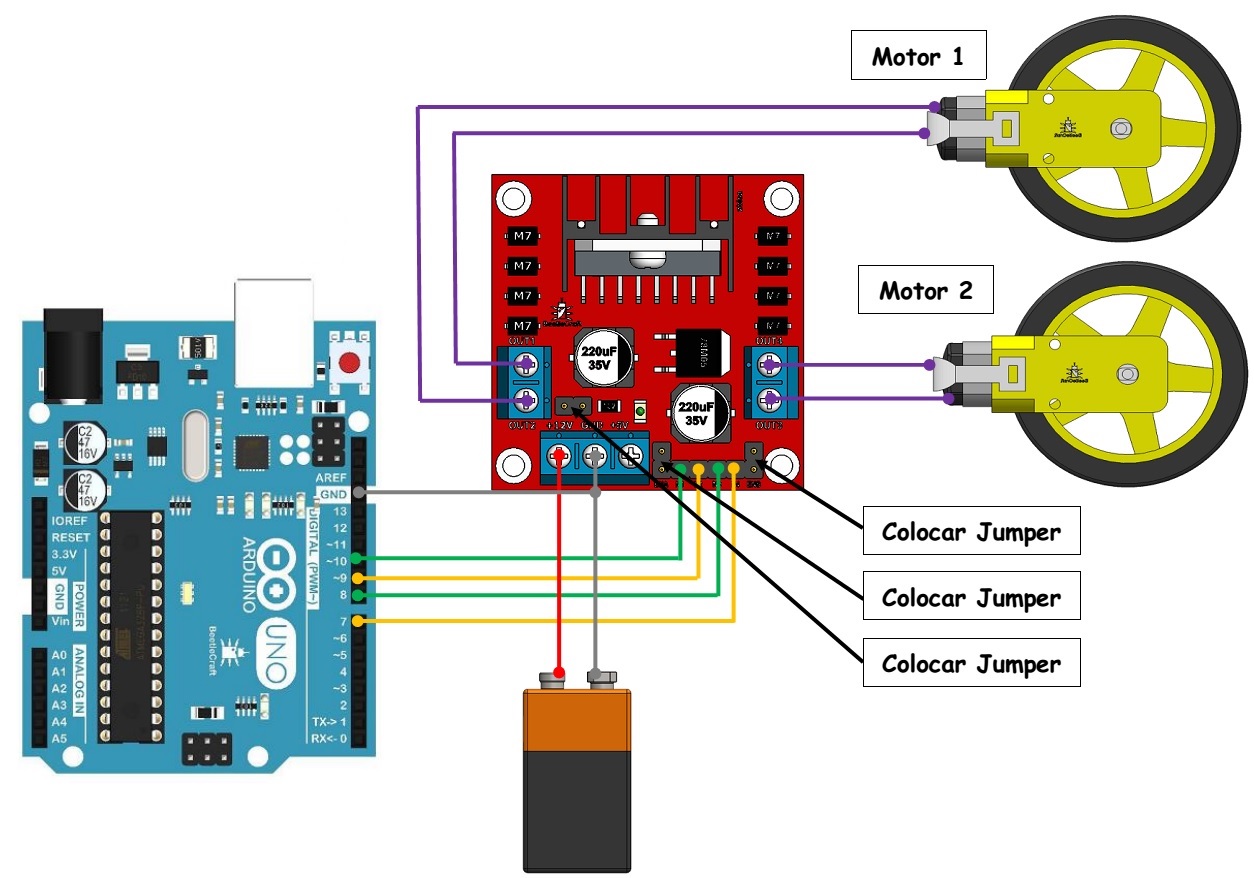


Ilustración - Esquema de conexión de puente H y motores con Arduino UNO

# Capítulo 10 – Desarrollo del SAR

El desarrollo del SAR se descompone en varios niveles de capas. Por un lado, existen dos esquemas muy diferenciados el lógico y el físico.

El esquema físico se compone de los dispositivos electrónicos que controlan los actuadores, efectores y sensores. Además de los microcontroladores (Arduino Mega, Arduino Nano) y la microcomputadora Raspberry Pi 3.

El esquema lógico se compone del sistema operativo Raspbian y una aplicación web desarrollada bajo la arquitectura cliente/servidor respetando el conjunto de herramientas MEAN. Además contiene un administrador de servicios para Node denominado PM2, y un controlador de cámaras de video conectadas al Sistema Operativo llamado Motion. (Enrique, 2016)

Este esquema a su vez se encuentra organizado en dos unidades funcionales llamadas Back-end y Front-end.

## Esquema general del SAR (lógico /físico)

Ilustración - Esquema general del SAR

## Desarrollo de la aplicación (front-end)

El front-end se encuentra desarrollado en Angular 4+, contando con los siguientes esquemas:

(Pegar esquemas del Compodoc)

Al conectarnos al servidor, desde un cliente (Browser) nos descarga la aplicación embebida, y la comunicación con el servidor sucede a través de Json.

## Desarrollo del servidor (back-end)

El back-end, construido en Node y Express, almacena los estáticos del front-end y los suministra al conectarse un cliente. Los directorios del proyecto son: COLOCAR DIRECTORIOS

Express filtra con las rutas AGREGAR RUTAS resolviendo y respondiendo las distintas consultas realizadas por la App en Angular.

Node es administrado por PM2 (administrador de procesos para JavaScript) el cual inicia el servidor, automáticamente al arrancar Raspbian, controlando y monitoreándolo.

Ilustración - Logo PM2

MOSTRAR PM2 EN FUNCIONAMIENTO

## Esquema de la arquitectura lógica

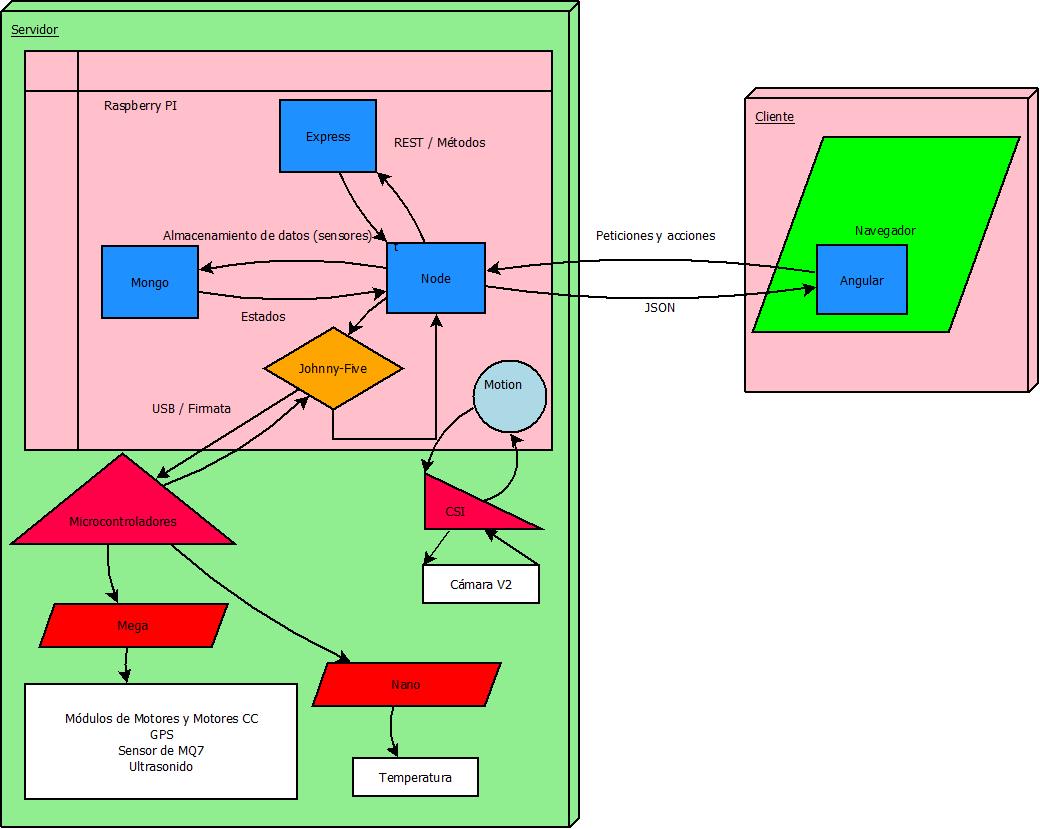


Ilustración - Arquitectura lógica del SAR

## Capturas de pantalla de la aplicación

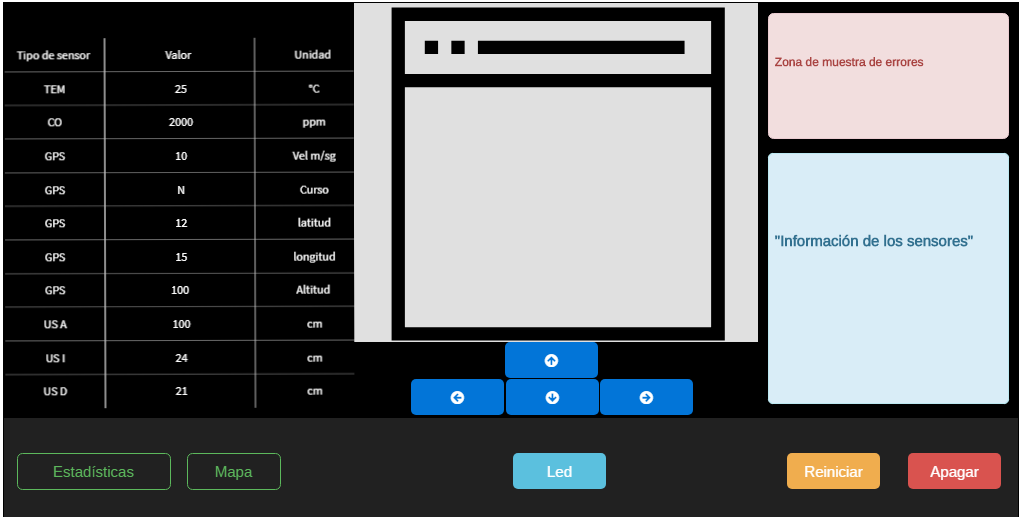


Ilustración - Aplicación Web

## Funcionamiento de la App

FALTA FINALIZARLA

## Como se construyen las estadísticas

## FALTA DETERMINAR LAS MUESTRAS

## Parámetros de configuración

Dentro de esta sección agregar MOTION, PM2, Ad-hoc

## **FALTA FINALIZARLA**

Referencias bibliográficas

1. Agente basado en objetivos: “Almacena información del estado del mundo, así como del conjunto de objetivos que intenta alcanzar, y que es capaz de seleccionar la acción que eventualmente lo guiará hacia la consecución de sus objetivos” [Inteligencia Artificial un enfoque moderno. Person. Stuart Russell, Peter Norving 2da Ed. Pág. 57] [↑](#footnote-ref-1)
2. “Arduino nace como una solución para los diseñadores…”” Donde más se está potenciando es en la educación…” Matías Scovotti, director pedagógico y co-fundador de Educabot. <http://www.telam.com.ar/notas/201704/184406-robotica-arduino-day.html> [↑](#footnote-ref-2)
3. <http://www.educaciontrespuntocero.com/noticias/raspberry-pi-educacion/34377.html> [↑](#footnote-ref-3)
4. Inteligencia Artificial: “Es la inteligencia exhibida por máquinas… una máquina ‘inteligente’ ideal es un agente racional flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea.” <https://es.wikipedia.org/wiki/Inteligencia_artificial> [↑](#footnote-ref-4)
5. <https://www.upc.edu/latevaupc/usos-y-beneficios-robotica-las-aulas/> [↑](#endnote-ref-1)
6. http://comoprogramarpic.blogspot.com.ar/2012/06/programando-un-atmel-mi-primer-programa.html [↑](#endnote-ref-2)
7. <https://www.arduino.cc/en/Main/Products> [↑](#endnote-ref-3)
8. Internet de las cosas: “Concepto que se refiere a la interconexión de objetos cotidianos con Internet. “ <https://es.wikipedia.org/wiki/Internet_de_las_cosas> [↑](#footnote-ref-5)
9. <https://www.arduino.cc/en/aug/> [↑](#footnote-ref-6)
10. <https://www.arduino.cc/en/Reference/PortManipulation> [↑](#footnote-ref-7)
11. El uso de algunos de estos módulos queda en forma tentativa, dado que existen también en la Raspberry y su uso puede ser complementario. [↑](#footnote-ref-8)
12. <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/> [↑](#endnote-ref-4)
13. Protoboard: O placa de pruebas en castellano, se le llama así a un tablero con orificios que se encuentran conectados eléctricamente entre si siguiendo un determinado patrón. Es utilizado para la conexión de componentes electrónicos. [↑](#footnote-ref-9)
14. En el sitio oficial se encuentra disponible una sección en donde la comunidad puede compartir distintas experiencias y novedades sobre esta plataforma (<https://www.raspberrypi.org/community/>). Por otro lado, cuenta con un área exclusiva donde se puede obtener distinto material didáctico, con proyectos para realizar por ejemplo con alumnos (<https://projects.raspberrypi.org/en/projects>). [↑](#footnote-ref-10)
15. <https://www.developereconomics.com/graphs/de11>

    <https://www.xatakamovil.com/mercado/desarrollo-de-aplicaciones-moviles-i-asi-esta-el-mercado> [↑](#footnote-ref-11)
16. https://source.android.com/security/overview/kernel-security [↑](#endnote-ref-5)
17. https://es.wikipedia.org/wiki/Dise%C3%B1o\_web\_adaptable [↑](#footnote-ref-12)
18. <https://es.wikipedia.org/wiki/PhoneGap>

    <https://es.wikipedia.org/wiki/Apache_Cordova> [↑](#footnote-ref-13)
19. https://es.wikipedia.org/wiki/Android\_Studio# [↑](#footnote-ref-14)
20. <https://es.wikipedia.org/wiki/Apache_Cordova>

    http://cordova.apache.org/ [↑](#footnote-ref-15)
21. https://en.wikipedia.org/wiki/Libuv [↑](#footnote-ref-16)
22. https://github.com/libuv/libuv/wiki/Projects-that-use-libuv [↑](#footnote-ref-17)
23. Sitio oficial de Johnny-five: <http://johnny-five.io/> [↑](#footnote-ref-18)
24. <https://github.com/firmata/arduino> [↑](#endnote-ref-6)
25. Se puede obtener del siguiente sitio web <https://github.com/firmata/ConfigurableFirmata> [↑](#footnote-ref-19)
26. <https://www.raspberrypi.org/downloads/raspbian/> [↑](#endnote-ref-7)
27. 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-20)
28. [↑](#footnote-ref-21)