



**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.....	10
1.1 Objetivo general.....	10
1.1.1 Objetivos específicos.....	10
1.1.2 Metodología.....	10
1.2 Motivación.....	11
1.3 Desarrollos Propuestos.....	12
1.4 Resultados Esperados	12
Capítulo 2 - La robótica.....	13
2.1 ¿Qué es la robótica?	13
2.2 Estructura física de los robots	15
2.2.1 Poliarticulados.....	15
2.2.2 Móviles.....	15
2.2.3 Androides.....	16
2.2.4 Zoomórficos	16
2.2.5 Híbridos	16
2.3 Distintas tecnologías para la robótica educativa	17
2.4 Microcontroladores y computadora de placa reducida (SBC)	17
2.5. Comunicación entre distintas arquitecturas de cómputo	19
2.5.1 Formas de comunicación.....	19
2.5.2 Tipos de medios de transmisión.....	19
2.6 SAR (Sistema Autónomo Robótico)	20
2.7 La robótica en la educación.....	20
2.7 Diseño conceptual del SAR.....	22
Resumen	23
Capítulo 3 – Arduino.....	24
3.1 Arduino	24
3.2 Historia	24
3.2.1 Wiring.....	25
3.2.2 Processing.....	27
3.2.3 Fritzing.....	28
3.3 Características generales de la plataforma	28
3.4 Distintas plataformas para Arduino.....	30
3.5 Aplicaciones	32
3.6 Motivaciones para su uso	32
3.6.1 La comunidad.....	32
3.6.2 Sencillez de programación.....	33

3.6.3 Hardware económico.....	33
3.7 Incorporación de Arduino en las escuelas.....	34
3.7.1 Las tres erres	34
3.8 Actuadores y sensores.....	35
3.9 Actuadores en el SAR.....	35
3.10 Sensores en el SAR.....	36
3.11 Módulos o <i>shields</i> en el SAR.....	37
Resumen	38
Capítulo 4 – Raspberry Pi	39
4.1 Raspberry Pi	39
4.2 Especificaciones técnicas de las distintas versiones.....	39
4.3 Entrada/Salida de propósito general (GPIO)	40
4.4 Sistemas Operativos compatibles.....	42
4.5 Accesorios para Raspberry Pi	43
4.6 Ventajas del uso de Raspberry Pi	44
Resumen	46
Capítulo 5 - Aplicaciones Móviles	47
5.1 Las Aplicaciones móviles.....	47
5.1.1 Las Web Apps	48
5.1.2 Ventajas de las Web-App:.....	48
5.1.3 Desventajas de las Web-Apps.....	49
5.2 Sistemas operativos para dispositivos móviles	49
5.3 Android.....	50
5.4 Aplicaciones móviles multiplataforma.....	51
5.4.1 Diferencias entre aplicaciones y web móviles	51
5.4.2 App Nativas	51
5.4.3 Desarrollo de Web Apps.....	52
5.4.4 Aplicaciones Híbridas	53
5.4.5 Creación de una Aplicación híbrida	53
5.4.6 Aplicación híbrida: app interpretada	54
5.5 Entornos y herramientas para el desarrollo.....	54
5.5.1 Android Studio.....	55
5.5.2 App Inventor	56
5.5.3 Tecnologías del lado del cliente - Open Web Stack (HTML, CSS y JS).....	56
5.5.3.1 HTML	56
5.5.3.2 CSS	56
5.5.3.3 JS	56

5.5.3.4 SASS.....	57
5.5.3.5 Angular JS.....	57
5.5.4 Cordova	57
5.5.5 Intel XDK.....	57
5.5.6 Ionic.....	58
5.5.6 Meteor.....	58
5.5.7 Meteor y Cordova.....	58
Resumen.....	59
Capítulo 6 – Stack MEAN.....	60
6.1 ¿Qué es MEAN?	60
6.2 Componentes de MEAN	61
6.2.1 MongoDB.....	61
6.2.2 Express.....	61
6.2.3 Angular	61
6.2.4 NodeJS.....	61
6.3 Otros complementos	64
6.3.1 Twitter Bootstrap.....	64
6.3.2 Compodoc.....	64
6.3.3 JSON	64
6.3.3 JQuery.....	65
Resumen	66
Capítulo 7 – Comunicación NodeJS con Arduino.....	67
7.1 Johnny-five	67
7.2 Instalación	67
7.3 Arduino Firmata	68
7.4 Surgimiento y funcionamiento de Firmata	68
7.5 Métodos de librería Firmata en Arduino	69
7.5.1 Métodos de propósito general.....	69
7.5.2 Métodos para el envío de mensajes.....	70
7.5.3 Métodos para la recepción de mensajes.....	70
7.5.4 Otros métodos.....	70
7.6 Instalación de Firmata en Arduino	71
Resumen	74
Capítulo 8 - Análisis y selección de tecnologías para el desarrollo del SAR.....	75
8.1 Primer análisis	75
8.2 Selección de tecnologías hardware	76
8.2.1 Razones para la elección de Arduino	76

8.2.2 Razones para la elección de Raspberry Pi.....	76
8.2.3 Comparativa entre Arduino Mega, Arduino Nano y Raspberry Pi 3 Model b	77
8.2.4 Beneficios del complemento Arduino/Raspberry	77
8.2.5 Cámara V2 de Raspberry Pi.....	78
8.2.6 Módulos de Arduino	78
8.3 Selección tecnologías software	79
Resumen	82
Capítulo 9 – Arquitectura y Ensamblado del SAR	83
9.1 Componentes.....	83
9.2 Estructura	86
9.2.1 Diseño.....	86
9.2.2 Los 4 niveles.....	87
9.3 Esquemas de conexión de componentes Arduino	88
Resumen	91
Capítulo 10 – Desarrollo del SAR	92
10.1 Estructura de la aplicación (<i>front-end</i>).....	92
10.2 Desarrollo del servidor (<i>back-end</i>).....	93
10.3 Esquema de la arquitectura lógica.....	94
10.4 Funcionamiento de la App	95
10.5 Puesta en producción del SAR	97
10.5.1 Configuración de Raspberry como AP.....	97
10.5.2 Configuración del servicio Motion.....	100
10.5.3 Instalación del gestor de procesos PM2.....	102
Resumen	103
Capítulo 11 – Conclusión y trabajos futuros	104
11.1 Conclusión final	104
11.1.1 Ensamblar un robot móvil integrando las plataformas Arduino y Raspberry Pi con diversos módulos y software.....	104
11.1.2 Desarrollar una aplicación web multiplataforma que mediante comunicación inalámbrica permita el control del Robot móvil.....	104
11.1.3 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.	105
11.1.4 Ensamblar físicamente e integrar a nivel de software los distintos componentes (sensores y actuadores) al SAR.....	105
11.1.5 Extender la aplicación para interactuar con la información que brinda el SAR de los sensores.....	105
Anexo de casos de pruebas.....	107
Servomotor SG90.....	107

Código sg90-01-funcionamiento	108
Pruebas en el sensor de Monóxido de Carbono.....	109
Código MQ7-01-funcionamiento.....	110
Caso de prueba N 1 Módulo WIFI ESP8266 Velocidad.....	111
Caso de prueba N 2 Módulo WIFI ESP8266 Velocidad.....	113
Caso de prueba Módulo WIFI ESP8266 Velocidad y configuración AP.....	115
Código comandosAT-configuracionWifi.ino	117
Caso de prueba N 3 Módulo WIFI ESP8266 Velocidad.....	118
Código pruebaVelocidad6-configuracionWifi	122
Caso de prueba Módulo GPS	125
Código GPS-NEO6-01Conectividad	127
Caso de prueba Módulo microSD Card Adapter.....	128
Código microSD-01-LeerEscribir	130
Caso de prueba Integración WIFI y Cámara	132
Caso de prueba Cámara OV 7670.....	134
Código OV7670	136
Caso de prueba Módulo Bluetooth HC05-01.....	151
Comunicación Bluetooth.ino.....	153
Anexo de códigos	154
Códigos del lado del servidor.....	154
Código StandarFirmata utilizado en el Arduino MEGA	154
Código ConfigurableFirmata utilizado en el Arduino NANO.....	173
Código Servidor Node (server.js)	184
Código API Express (api.js)	185
Código Manejo de Arduino Mega y Arduino Nano (placas.js)	188
Código de Apagar y reiniciar (apagar.js).....	195
Códigos del lado del cliente.....	196
Código de Servicio Angular (servicio.ts)	196
Código de appComponent.ts Angular.....	198
Código de appModule (rutas) [Extracto]	199
Glosario	200
Ampere.....	200
AP	200
API	200
Back-End.....	200
Daemon.....	200
Datos raw.....	200

DHCP	200
DOM	200
Framework	201
Front-End	201
Host	201
HTML	201
HTTP	201
IDE	201
Inteligencia Artificial	201
Internet	202
IoT	202
IP	202
LAN	202
Lenguaje de programación	202
LESS	202
Linux	202
Marshaling	202
Navegador web	203
Protoboard	203
Open Source	203
Query	203
Resolución de pantalla	203
Template	203
UART	203
WIFI	203
Bibliografía	204

Ilustración 1 - Esquema básico de un robot.....	14
Ilustración 2 - Ejemplo de robot poliarticulado.....	15
Ilustración 3 - Ejemplo de robot móvil	15
Ilustración 4 - Androide Asimo de Honda.....	16
Ilustración 5 - Robot Zoomórfico caminador	16
Ilustración 6 - Robot móvil-poliarticulado	16
Ilustración 7 - Arquitectura de un microcontrolador	18
Ilustración 8 - Esquema conceptual orientado a servicios.....	22
Ilustración 9 - Logo de Arduino.....	24
Ilustración 10 – Código de Blink en Wiring IDE.....	26
Ilustración 11 - C++ Blink ejemplo.....	26
Ilustración 12 - Logo de Processing.....	27
Ilustración 13 - Processing ejemplo.....	28
Ilustración 14 - Entorno Fritzing.....	28
Ilustración 15 - Ejemplo serie	29
Ilustración 16 - Niveles de entrada a la plataforma Arduino	30
Ilustración 17 - Arduino Uno	31
Ilustración 18 - Logotipo comunidad open-source de Arduino.....	33
Ilustración 19- Representación actuadores y sensores	35
Ilustración 20 - Actuadores y sensores compatibles con Arduino.....	36
Ilustración 21- Representación de sensores	37
Ilustración 22 - Logo oficial de Raspberry Pi	39
Ilustración 23 - Raspberry Pi 2 y sus GPIOs.....	41
Ilustración 24 - Interfaces de Raspberry Pi	42
Ilustración 25 - Cámara Raspberry Pi V2.....	43
Ilustración 26 - Pantalla táctil de Raspberry Pi.....	43
Ilustración 27 - Adafruit Prototyping Pi	43
Ilustración 28 - Pidrive	44
Ilustración 29 - Pi TFT.....	44
Ilustración 30 - Aplicaciones móviles.....	47
Ilustración 31 - App nativa vs Web App.....	48
Ilustración 32 – WebApps – Diseño multipropósito.....	49
Ilustración 33 - Arquitectura de Android	50
Ilustración 34 - Logo de Android.....	50
Ilustración 35 - Cuadro comparativo - Aplicaciones nativas	52
Ilustración 36 - Cuadro comparativo - Aplicaciones Web	53
Ilustración 37 - Comparativa aplicaciones híbridas.....	54
Ilustración 38 - Herramientas para desarrollo de apps	54
Ilustración 39 - Acrónimo MEAN.....	60
Ilustración 40 - Arquitectura de interacción MEAN	60
Ilustración 41 - Logo del motorV8.....	61
Ilustración 42 Comparativa de servidores tradicionales y NodeJS	63
Ilustración 43 - Logo de JSON	64
Ilustración 44 - Json pegamento de tecnologías.....	65
Ilustración 45 - Sitio web oficial de Johnny-Five (http://johnny-five.io/).....	67
Ilustración 46 – Firmata como interfaz.....	68
Ilustración 47 - IDE de Arduino.....	71
Ilustración 48 - Código StandardFirmata	72
Ilustración 49 - Código ConfigurableFirmata.....	73
Ilustración 50 - Esquema de conexión de componentes	83
Ilustración 51 - Raspberry Pi 3.....	83

Ilustración 52 - Arduino Mega.....	84
Ilustración 53 - Arduino Nano.....	84
Ilustración 54 - Motores CC.....	84
Ilustración 55 - Sensor de ultrasonido.....	84
Ilustración 56 - Porta pilas.....	85
Ilustración 57 - Módulo Puente H.....	85
Ilustración 58 - Mini-protoboard	85
Ilustración 59 - Sensor de Temperatura	85
Ilustración 60 - MQ7 CO.....	85
Ilustración 61 - GPS.....	85
Ilustración 62 - Cámara V2	86
Ilustración 63 - Panel Solar Power Bank	86
Ilustración 64 - Diseño estructura nivel 3 con SketchUp	86
Ilustración 65 - Diseño estructura nivel 4 con SketchUp	86
Ilustración 66 - Diseño estructura nivel 1 con SketchUp	86
Ilustración 67 - Diseño estructura nivel 2 con SketchUp	86
Ilustración 68 - Impresión 3D del nivel 1	87
Ilustración 69 - Nivel 2 descubierto.....	87
Ilustración 70 - RM Vista Lateral	87
Ilustración 71 - Esquema de conexión de sensor de monóxido MQ-7 a Arduino Mega.....	88
Ilustración 72 - Esquema de conexión de sensor de temperatura DS18D20 a Arduino Nano....	88
Ilustración 73 - Esquema de conexión de sensores ultrasónicos HC-SR04 con Arduino Mega..	89
Ilustración 74 - Esquema de conexión de módulo GPS con Arduino UNO	89
Ilustración 75 - Esquema de conexión de puente H y motores con Arduino UNO.....	90
Ilustración 76 - Esquema general del SAR.....	92
Ilustración 77 - Módulos Angular.....	92
Ilustración 78 - Backend.....	93
Ilustración 79 - Arquitectura lógica del SAR.....	94
Ilustración 80 - Aplicación web.....	95
Ilustración 81 – Estadísticas de temperaturas.....	96
Ilustración 82 - Estadísticas de monóxido.....	96
Ilustración 83 - Aplicación web - Otras opciones.....	97
Ilustración 84 - Software de configuración de Raspberry.....	100
Ilustración 85 - Monitor de PM2.....	102
Ilustración 86 - Keymetrics.....	102

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 objetivos¹) para la exploración y análisis del medio ambiente.

1.1.1 Objetivos específicos

- Ensamblar un Robot Móvil integrando las plataformas Arduino y Raspberry Pi con diversos módulos y software.
- Desarrollar una aplicación web multiplataforma que mediante comunicación inalámbrica permita el control del Robot Móvil.
- Investigar y evaluar protocolos de comunicación para la recolección de datos y control entre microcontroladores y aplicaciones web.
- Integrar sensores al robot móvil y escribir el software, utilizando el protocolo seleccionado, para la transmisión de las medidas y presentación en la aplicación web.

1.1.2 Metodología

El SAR se creará 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

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]

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 como microcontroladores, 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 la 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 (Physical 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. [1]

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 [2] (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.

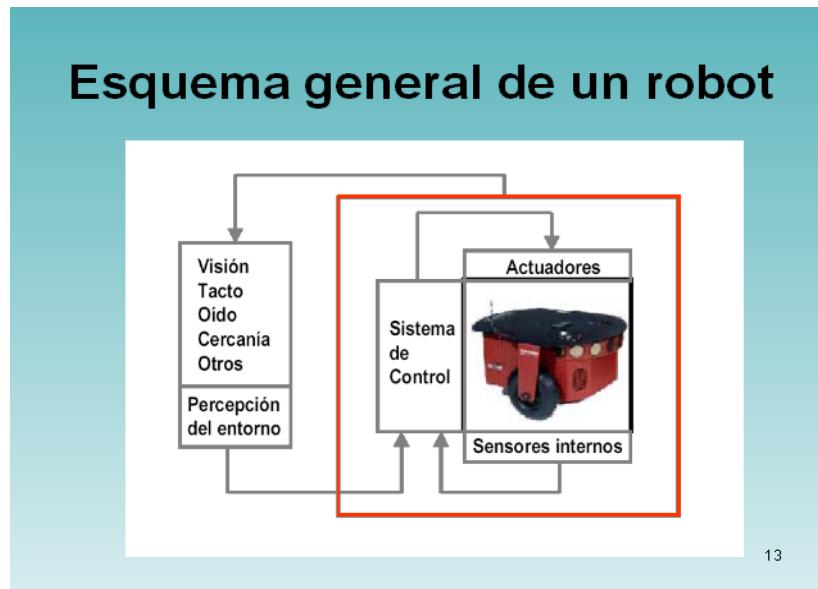


Ilustración 1 - Esquema básico de un robot

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 2 - 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 3 - 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 4 - 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 5 - Robot Zoomórfico caminador

2.2.5 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 6 - 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 **Arduino²** y **Raspberry Pi**. 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 fabrica (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 - **Capítulo 3 – Arduino**). 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. [3]

Existen otras tecnologías para el desarrollo de la robótica, como por ejemplo, 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. La **BeagleBone**, es una placa computadora de hardware libre diseñada como plataforma de evaluación y de prototipos para ingenieros profesionales. La **Nanode**, es una 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**.

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 (como por ejemplo en el caso de un mouse, disco rígido o una cámara web). Por ende, se puede

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>

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.

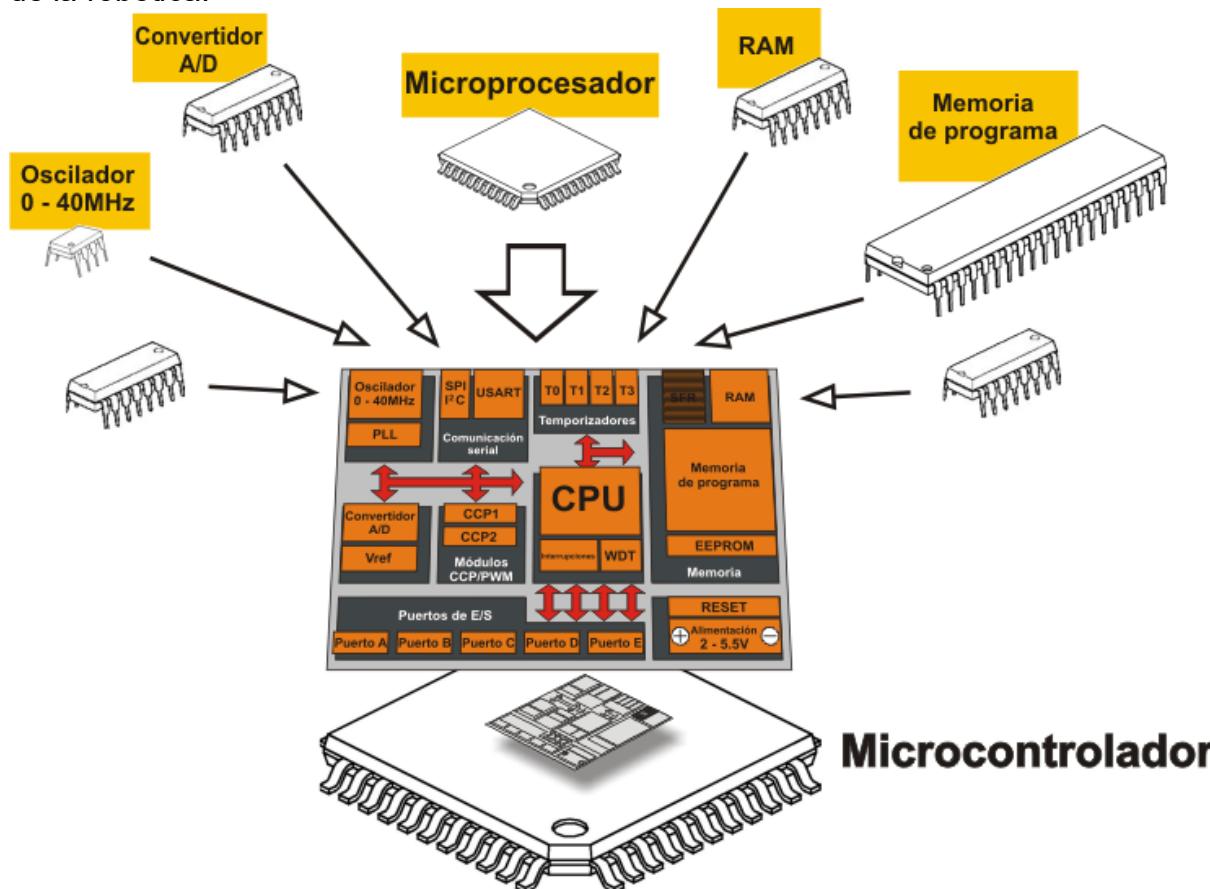


Ilustración 7 - Arquitectura de un microcontrolador

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 arquitecturas de cómputo

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

2.5.1 Formas de comunicación

- *Paralelo*: La comunicación paralela, es un método para transmitir muchos *packs* de múltiples dígitos en binarios (bits) de manera simultánea.
- *Serial*: La comunicación serie o serial es una interfaz de comunicación de datos digitales que nos permite establecer transferencia de información entre varios dispositivos. Es un método donde el proceso de envío de datos se realiza de un bit a la vez, en forma secuencial, sobre un canal de comunicación o un *bus*. Un puerto es el nombre genérico con que denominamos a las interfaces, físicas o virtuales, que permite esta comunicación entre dispositivos. Dado que es una comunicación serie, 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.

2.5.2 Tipos de medios de transmisión

- **Alámbricas**: Los medios de comunicación alámbricos son aquellos en los que se basan en la transmisión de información a través de un conductor que transporta corriente eléctrica.
- **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 cumplen con los estándares IEEE 802.11 vigentes relacionados a redes inalámbricas de área local.

2.6 SAR (Sistema Autónomo Robótico)

Se le considera SAR o sistema autónomo robótico a aquellos robots que presentan cierto grado de autonomía (**Inteligencia Artificial**) 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 las posibilidades ofrecidas por la utilización de computadoras se localizan 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. [4]

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. [5]

2.7 Diseño conceptual del SAR

Como podemos apreciar en la figura (**Ilustración 8 - Esquema conceptual orientado a servicios**), el SAR cuenta con una estructura similar, a nivel arquitectónico, al de un robot. El sistema de control (SC) es el encargado de gestionar las comunicaciones para acceder a los sensores, actuadores y módulos. Además, tiene la capacidad de atender solicitudes de clientes que se conectan con el SAR. El SC administra servicios, que proporciona a los clientes conectados. Estos servicios son:

- Almacenamiento por medio de una base de datos. Todos los valores de los sensores y módulos son almacenados cada vez que sucede un cambio en su lectura.
- Servicio WEB. Este servicio, permite almacenar la aplicación cliente que es desplegada cuando el cliente se conecta con el SAR. Además, permite la interacción posterior entre el cliente y el SC.
- Comunicación con los sensores, actuadores y módulos:
 - Lectura de sensores
 - Acciones sobre los actuadores
 - Lectura de valores proporcionados por los módulos.
- Transmisión de imagen y video en tiempo real, al cliente.
- Generación de punto de acceso inalámbrico.

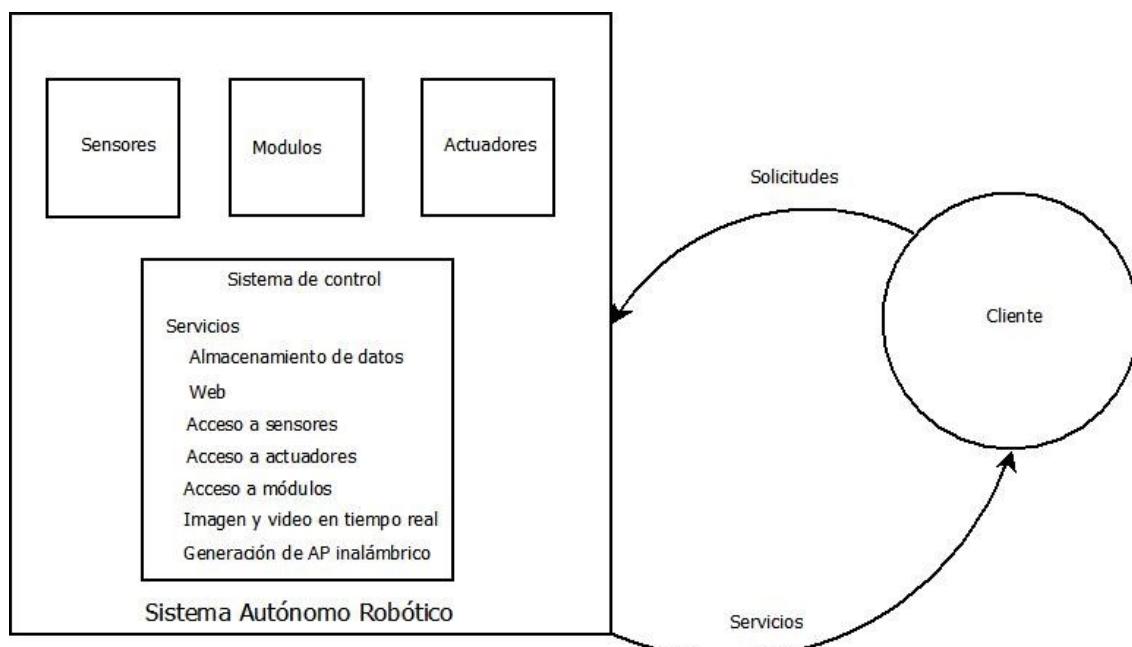


Ilustración 8 - Esquema conceptual orientado a servicios

Resumen

En este capítulo se abordó la definición de robot como:

"Un manipulador funcional reprogramable, capaz de mover materiales, piezas, herramientas o dispositivos especializados mediante movimientos variables programados, con el fin de realizar tareas diversas" y la robótica como "la ciencia y técnica que estudia a los robots, encargada del diseño, construcción y aplicabilidad de los mismos".

Se definió además que un robot cuenta con actuadores, sensores y un sistema de control. Están diseñados en base a tres grandes funcionalidades: la percepción, la planificación y la manipulación; y se clasifican en poliarticulados, móviles, androides, zoomórficos e híbridos.

Luego se mencionaron que las razones por las cuales Arduino y Raspberry Pi se han popularizado en el diseño y construcción de robots en el ámbito de la enseñanza fueron su facilidad de uso, bajo costo, materiales provistos por la comunidad, en comparación con Intel Galileo, BeagleBone, Nanode, entre otras.

Posteriormente, se analizaron los conceptos de microcontrolador y SBC (computadora de placa reducida) y los mecanismos de comunicación.

Dado que la propuesta del SAR está enfocada en el ambiente educativo, se mencionó que la robótica tiene doble impacto como elemento educacional y elemento social.

Estos conceptos serán de utilidad para entender el desarrollo propuesto de esta tesina: la construcción de un SAR (Sistema Autónomo Robótico).

Capítulo 3 – Arduino

En este capítulo conoceremos qué es la plataforma Arduino, sus comienzos y otras tecnologías que colaboraron en el desarrollo de la misma. Además, analizaremos características de la placa, examinando capacidades técnicas como el microcontrolador, memoria y medios de comunicación.

También veremos el abanico de placas producidas por la compañía, sus especificaciones técnicas, similitudes y diferencias. Por otro lado, se examinarán diversos sensores, actuadores y módulos compatibles con la plataforma Arduino. Por último, se comentará la aplicación en las instituciones educativas y la utilización de Arduino en el SAR.

3.1 Arduino

Arduino es una plataforma y compañía, del mismo nombre, de electrónica "**Open Source**" o de código abierto cuyo objetivo es brindar hardware y software de fácil utilización. Es decir, se propone como una plataforma sencilla con una curva de aprendizaje baja para realizar proyectos interactivos para público no necesariamente con conocimientos técnicos.

Arduino se trata de una SBC (2.4

Microcontroladores y computadora de placa reducida (SBC)) con entradas y salidas, analógicas y digitales, la cual es programada bajo un entorno de desarrollo, basado en el entorno de programación inspirado en **Processing** y en la estructura de programación **Wiring**. En la imagen (**Ilustración 9 - Logo de Arduino**) se puede ver el logo oficial de la compañía.



Ilustración 9 - Logo de Arduino

3.2 Historia

Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia). Dado que se utilizaba el microcontrolador BASIC Stamp, cuyo costo era alto para los fines educativos, es que, se comienza el proyecto Arduino. El nombre del proyecto viene del nombre del Bar di Re Arduino (Bar del Rey Arduino), donde Massimo Banzi empezaba a desarrollarlo. [6]

Un estudiante, Hernando Barragán, es quien desarrolló la tarjeta electrónica Wiring, el **Lenguaje de programación** y la plataforma de desarrollo. Una vez concluida dicha plataforma, los investigadores trabajaron para hacerlo más ligero, más económico y de mayor alcance a la comunidad de hardware y código abierto.

Posteriormente, Google colaboró en el desarrollo del Kit Android ADK (*Accessory Development Kit*), una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android para que el teléfono controle luces, motores y sensores conectados a Arduino.

Para la producción en serie de la primera versión se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul, debía ser *Plug and Play* y que trabajara con todas las plataformas informáticas tales como MacOSX, Windows y GNU/*Linux*.

3.2.1 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.

Wiring toma de Processing la **IDE** y el concepto de *sketch*, pero enfocado en la programación de microcontroladores en vez de programación gráfica. Provee 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 – Código de Blink en 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.

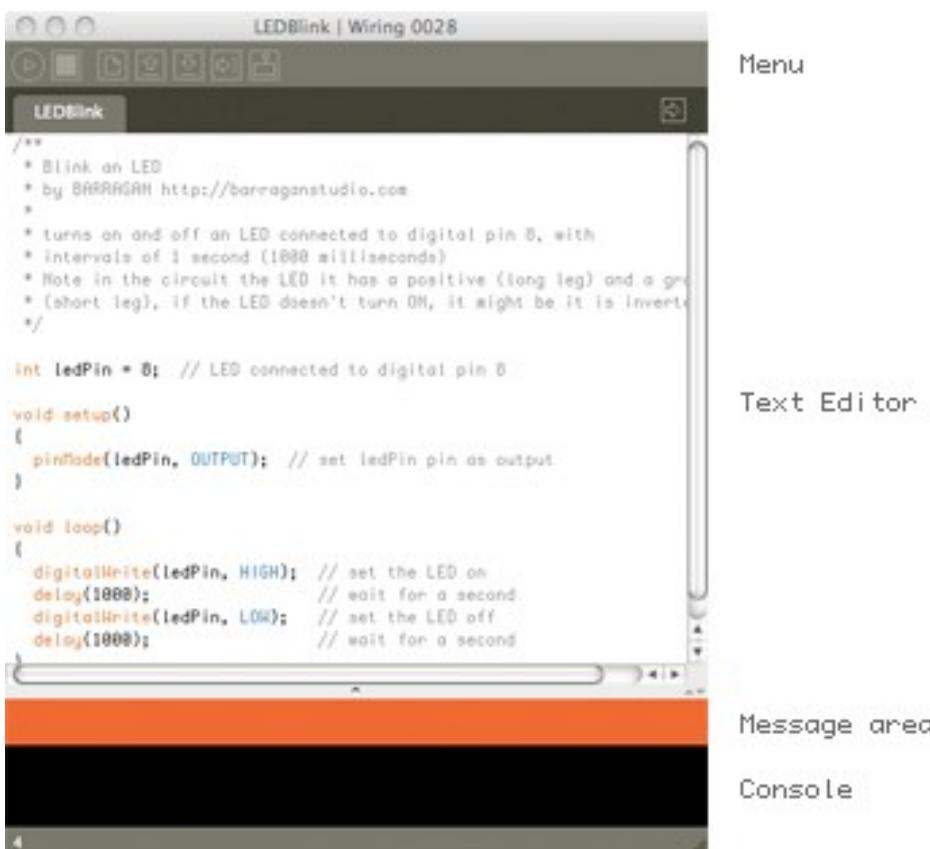


Ilustración 10 – Código de Blink en Wiring IDE

Para exemplificar la interfaz de programación que provee Wiring al usuario en contraposición a la utilización de la **API** del fabricante pude observarse como ejemplo el código en el lenguaje C++ de la siguiente ilustración (**Ilustración 11 - C++ Blink ejemplo**) el cual puede ser escrito de la forma dada en la ilustración anterior (**Ilustración 10 – Código de Blink en Wiring IDE**). [7]

```

1 #include <avr/io.h>
2 #include <util/delay.h>
3
4 int main(void) {
5
6     DDRB = (1<<PB5); //Pin B5 Salida
7
8     for(;;){ // while(1)
9         PORTB |= (1<<PB5); //Ponemos el pin en alto
10        _delay_ms(1000);
11        PORTB &= ~(1<<PB5); //Ponemos el pin en bajo
12        _delay_ms(1000);
13    }
14
15    return 0;
16 }

```

Ilustración 11 - C++ Blink ejemplo

3.2.2 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 12 - Logo de Processing**) se puede apreciar su logo.



Ilustración 12 - Logo de Processing

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.

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

Más adelante podemos apreciar un extracto de código de Processing viendo la similitud con el código Arduino. Al correr este ejemplo podemos observar como renderiza visualmente el código en el visor (**Ilustración 13 - Processing ejemplo**)

```
void setup() {
    size(480, 120);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(255);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```



Ilustración 13 - Processing ejemplo

3.2.3 Fritzing

El entorno de software 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 14 - Entorno Fritzing**), permite arrastrar componentes y generar un sketch. Fritzing es creado bajo los principios de Processing y Arduino, y permite a los usuarios, documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación.

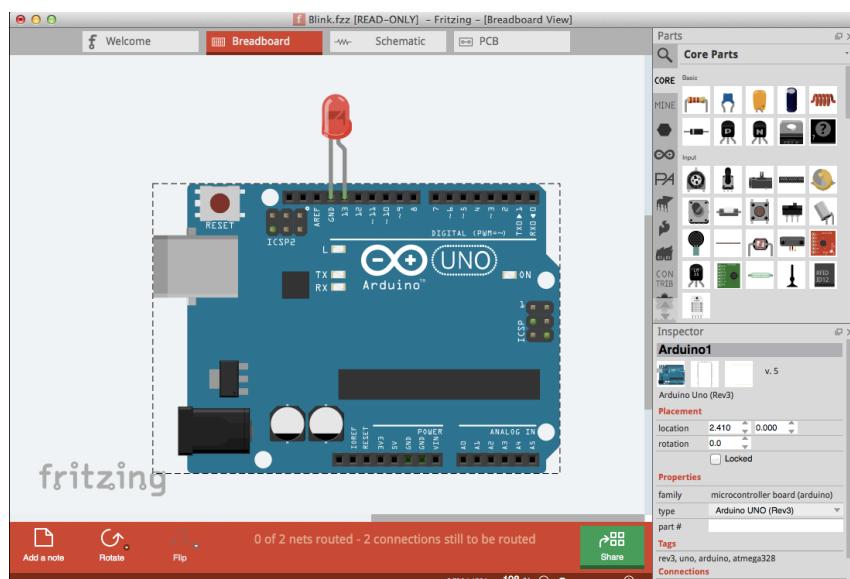


Ilustración 14 - Entorno Fritzing

3.3 Características generales de la plataforma

- ✓ 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.
- ✓ Arduino es 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 él mismo. Los microcontroladores leen sobre los sensores y escriben sobre los actuadores.

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.

Las placas Arduino además incluyen puertos serie, uno de ellos asociado a la conexión USB a la computadora a través de una **UART**.

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 15 - Ejemplo serie**), se realiza una comunicación serie a 9600 bps (baudios por segundo) 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.



The screenshot shows the Arduino IDE interface with the title bar "serial Arduino 1.5.4". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for upload, refresh, file, and other functions. A red box highlights the "Monitor Serie" button in the top right corner of the toolbar. The code editor window contains the following sketch:

```

serial
int cont=0;

void setup(){
  //iniciamos el puerto de serie
  Serial.begin(9600);
}

void loop(){
  //Imprimimos el valor del contador
  Serial.print("Contador: ");
  Serial.println(cont);

  //incrementamos el contador y esperamos un segundo
  cont++;
  delay(1000);
}

```

The bottom status bar shows "17" on the left and "Arduino Uno on /dev/ttyACM1" on the right.

Ilustración 15 - Ejemplo serie

3.4 Distintas plataformas para Arduino

ENTRY LEVEL	UNO	LEONARDO	101	ESPLORA	MICRO	NANO	MINI	MKR2UNO ADAPTER
	STARTER KIT	LCD SCREEN						
ENHANCED FEATURES	MEGA	ZERO	DUE	MEGA ADK	MO	MO PRO	MKR ZERO	MOTOR SHIELD
	USB HOST SHIELD	PROTO SHIELD	MKR PROTO SHIELD	4 RELAYS SHIELD				
	MEGA PROTO SHIELD	MKR RELAY PROTO SHIELD	ISP	USB2SERIAL MICRO				
	USB2SERIAL CONVERTER							
INTERNET OF THINGS	YUN	ETHERNET	TIAN	INDUSTRIAL 101	LEONARDO ETH	MKR FOX 1200		
	MKR WAN 1300	MKR CSM 1400	MKR1000	YUN MINI	YUN SHIELD	WIRELESS SD SHIELD		
	WIRELESS PROTO SHIELD	ETHERNET SHIELD V2	CSM SHIELD V2	MKR IoT BUNDLE				
EDUCATION	CTC 101							
WEARABLE	GEMMA	LILYPAD ARDUINO USB	LILYPAD ARDUINO MAIN BOARD	LILYPAD ARDUINO SIMPLE				
	LILYPAD ARDUINO SIMPLE SNAP							
3D PRINTING	MATERIA 101							



Ilustración 16 - Niveles de entrada a la plataforma Arduino

Existe una gran variedad de productos Arduino, la compañía los cataloga, como se puede apreciar en la imagen anterior (**Ilustración 16 - Niveles de entrada a la plataforma Arduino**), en distintos niveles según su utilidad [8]:

- 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* 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, como 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 17 - Arduino Uno**) se visualiza la distribución física de puertos y componentes de la versión Arduino Uno R3.

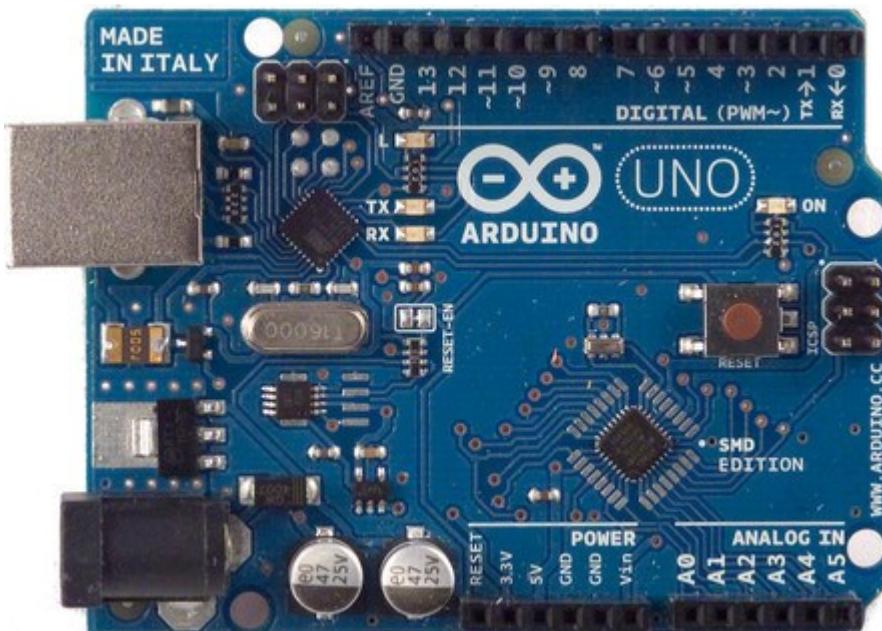


Ilustración 17 - Arduino Uno

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.

3.5 Aplicaciones

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

- Utilizarlo como microcontrolador, con un programa descargado desde una computadora y funcionando de forma independiente, recibiendo lecturas de sensores y realizando acciones sobre actuadores en función de las entradas y el programa.
- Ídem anterior pero conectado a una computadora (que también podría ser una SBC como Raspberry Pi).

En el siguiente apartado se enumeran una serie de razones por las cuales utilizar esta plataforma.

3.6 Motivaciones para su uso

3.6.1 La comunidad

Arduino cuenta con una gran comunidad, cuyas actividades se centran en la experimentación, publicación de resultados y proyectos, y organización de eventos. 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 evangelizador 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]

Dentro de la página oficial se brinda soporte por medio de documentación, foros y la publicación de un blog con novedades y proyectos relevantes que se encuentran en desarrollo.

Además, se han creado sitios como Arduino Playground, que consiste en 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 una de las bases de datos de conocimiento de la comunidad de Arduino. [10]

Este sitio a su vez tiene soporte de distintos lenguajes como el español. [11] Otro ejemplo de las actividades de la comunidad es el sitio Arduino Hub, un lugar donde se comparten los proyectos, dando los distintos pasos para reproducirlo.

El Arduino Day, o cumpleaños de Arduino, es una celebración mundial que se lleva a cabo una vez al año en diversos puntos del mundo. Este evento es organizado por la comunidad de Arduino y/o sus fundadores. En él se desarrollan diferentes talleres, charlas y concursos, entre otras actividades, relacionadas con la plataforma.

La siguiente imagen (**Ilustración 18 - Logotipo comunidad open-source de Arduino**) muestra el logotipo oficial de la comunidad **Open Source** de Arduino.



Ilustración 18 - Logotipo comunidad open-source de Arduino

3.6.2 Sencillez de programación

Gracias a la reutilización de las ideas de Wiring, Arduino provee un alto nivel de abstracción con respecto al hardware.

Por ejemplo, para establecer como salida los puertos 1 al 7, se puede utilizar el siguiente fragmento de código:

```
PinMode (1, OUTPUT) ;
PinMode (2, OUTPUT) ;
...
PinMode (7, OUTPUT) ;
```

En contraposición con:

```
DDRD = B11111110;[12]// sets Arduino pins 1 to 7 as outputs, pin 0 as input
```

3.6.3 Hardware económico

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

3.7 Incorporación de Arduino en las escuelas

Las diversas características y motivaciones hacen atractiva a la plataforma Arduino para su incorporación en las escuelas.

Esto ha llevado la creación de proyectos articulares entre distintos espacios curriculares.

Dentro de las principales características que han promovido esta tendencia, se encuentra la sencillez del **Lenguaje de programación** que permite que alumnos y docentes, no necesariamente del ámbito de la informática y la electrónica, puedan utilizarlo. Como consecuencia, contribuye a la construcción colectiva del conocimiento, promoviendo la interdisciplinariedad escolar, permitiendo la colaboración de docentes de distintas áreas y la cooperación en la articulación de proyectos.

Desde el punto de vista pedagógico, del proceso de aprendizaje, este tipo de actividades permiten al sujeto que aprende, ser participante activo. Desde la concepción de la idea hasta el producto final, incorporando gradualmente 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.

3.7.1 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 desoldado de componentes electrónicos de placas en desuso y materiales que se han desechado en las instituciones o en hogares de los alumnos.

3.8 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 acción para activar un elemento final de control, como por ejemplo un LED.

Por otro lado, 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).

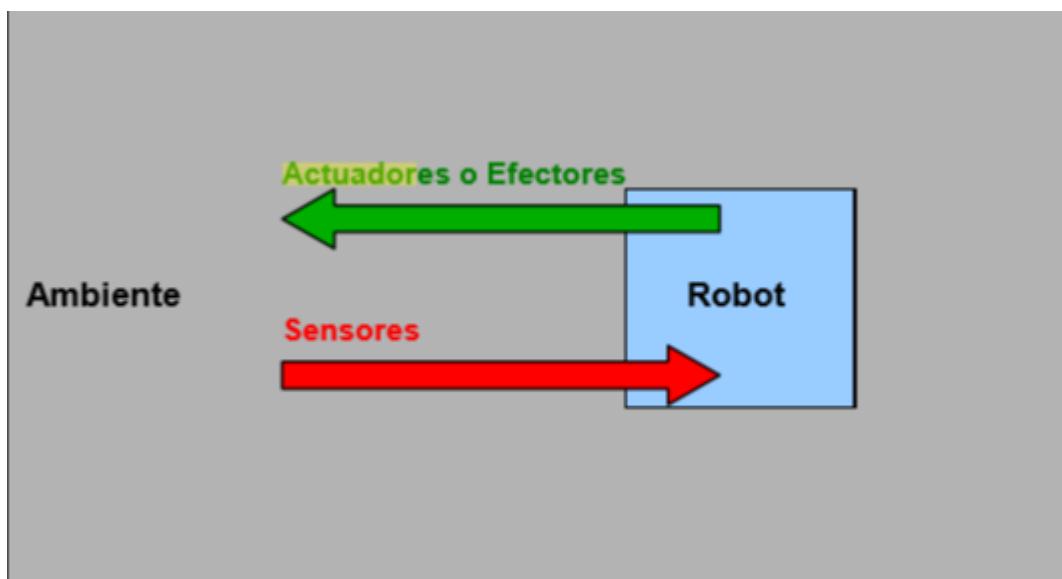


Ilustración 19- Representación actuadores y sensores

En esta imagen (**Ilustración 19- Representación actuadores y sensores**) se representan los datos que un robot puede capturar de su ambiente por medio de diversos sensores, y a su vez como podría interactuar con el mismo mediante actuadores.

3.9 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 utilizan:

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

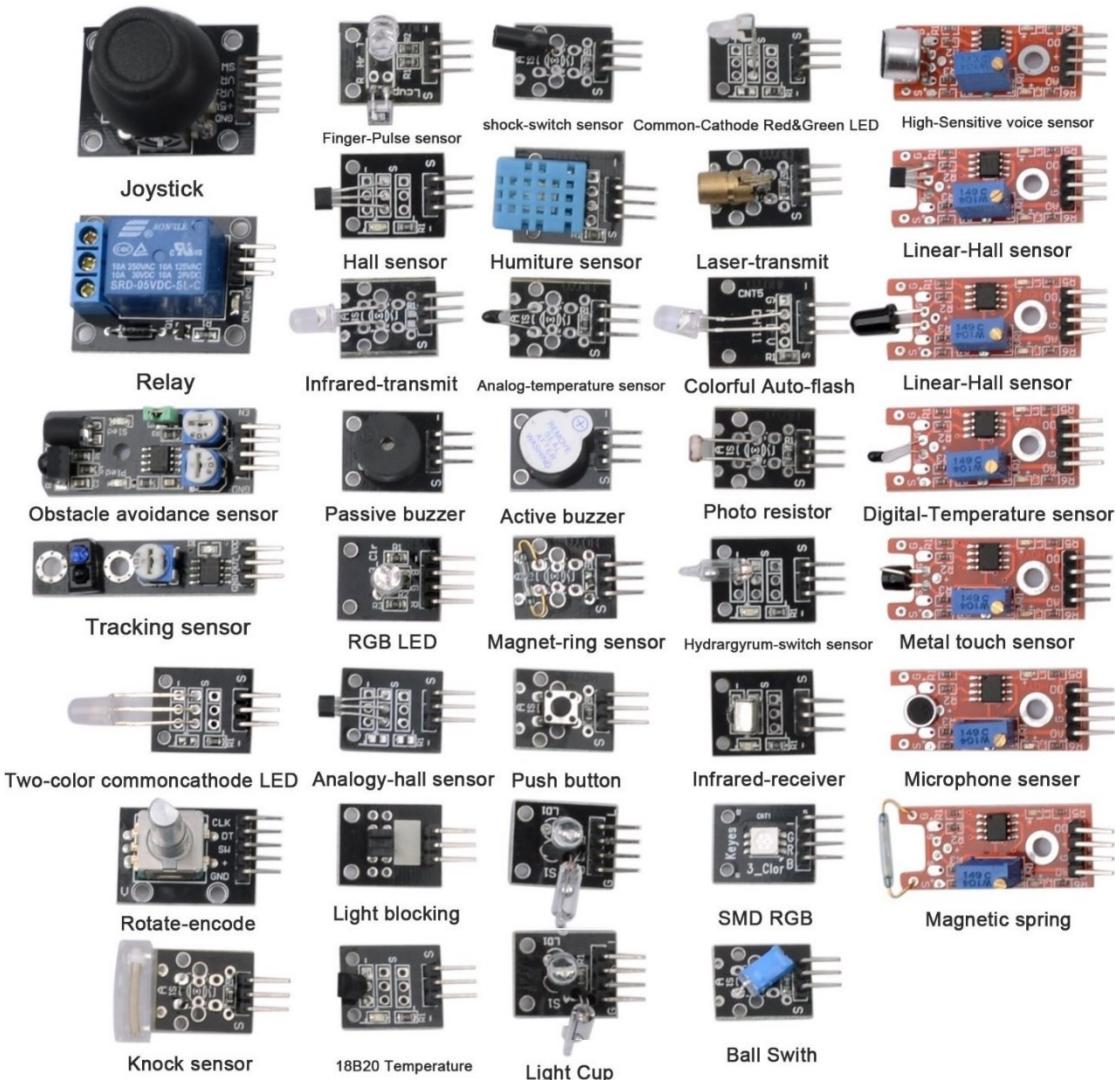


Ilustración 20 - Actuadores y sensores compatibles con Arduino

3.10 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 20 - Actuadores y sensores compatibles con Arduino**).

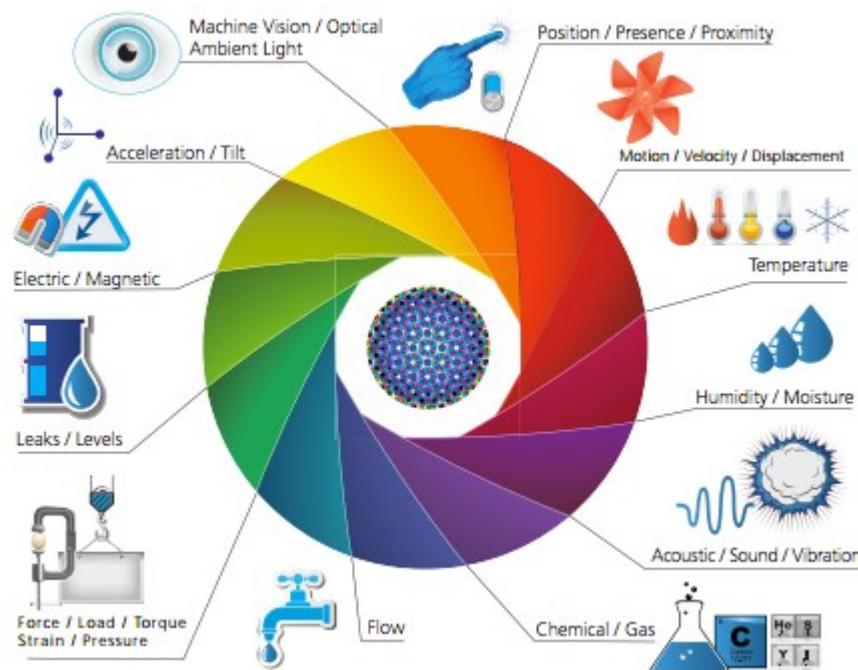


Ilustración 21- Representación de sensores

En esta imagen (**Ilustración 21- Representación de sensores**) se pueden apreciar los distintos factores de un entorno que pueden ser evaluados con sensores mencionados anteriormente.

3.11 Módulos o *shields* en el SAR

El SAR utiliza³:

- 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 modo **AP**

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. (**Anexo de casos de pruebas**)

³ El uso de algunos de estos módulos queda en forma tentativa, dado que existen también en la Raspberry y su uso puede ser complementario.

Resumen

Como vimos en el presente capítulo, Arduino es una plataforma electrónica **Open Source**, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. A su vez facilita la programación de un microcontrolador, este último lee sobre los sensores y escribe sobre los actuadores.

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.

Por otro lado, un sensor es un objeto capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Además, existen módulos que integran sensores y actuadores con un micro controlador.

Capítulo 4 – Raspberry Pi

En este capítulo se va a analizar y detallar el SBC Raspberry Pi, el cual tomó un papel fundamental en el desarrollo del SAR, siendo el mismo el centro de mando del robot móvil. Se detallan las especificaciones técnicas de las principales versiones de esta plataforma, donde se puede apreciar la evolución, en cuanto al hardware, que ha ido teniendo.

Por otro lado, se introduce el concepto de GPIO, que no son más que pines de Entrada/Salida de propósito general, para la conexión de diversos sensores, módulos y/o actuadores que se deseen comunicar, en este caso, con la Raspberry Pi.

Además, se presentan variados sistemas operativos y accesorios complementarios compatibles con Raspberry Pi. Dentro de los accesorios se describe la cámara V2 de esta plataforma utilizada en el SAR.

Para finalizar el capítulo se describen una serie de ventajas que presenta esta plataforma con respecto a otras similares.

4.1 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

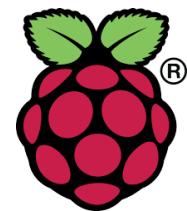


Ilustración 22 - Logo oficial de Raspberry Pi

objetivo primordial es el de estimular la enseñanza de la informática en las escuelas. Su logo oficial, como se muestra en la imagen (**Ilustración 22 - Logo oficial de Raspberry Pi**) no es más que una frambuesa.

4.2 Especificaciones técnicas de las distintas versiones

En la siguiente tabla se puede observar la evolución de las diversas versiones de Raspberry Pi, más populares, a lo largo del tiempo. [13]

	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 + GPU + DSP + 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 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, I ² C, 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)
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 genéricos integrados 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.

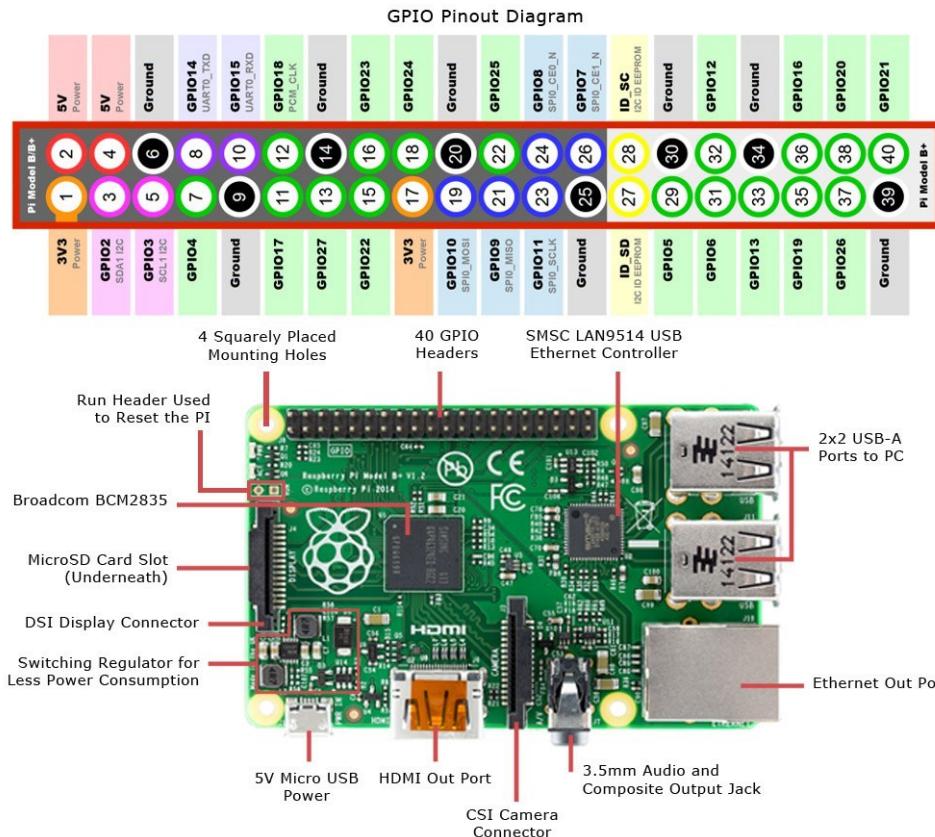


Ilustración 23 - Raspberry Pi 2 y sus GPIOs

En la imagen (**Ilustración 23 - 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 tesis) 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). [14]

La siguiente imagen (**Ilustración 24 - Interfaces de Raspberry Pi**) ilustra los distintos periféricos que se pueden conectar a este computador.

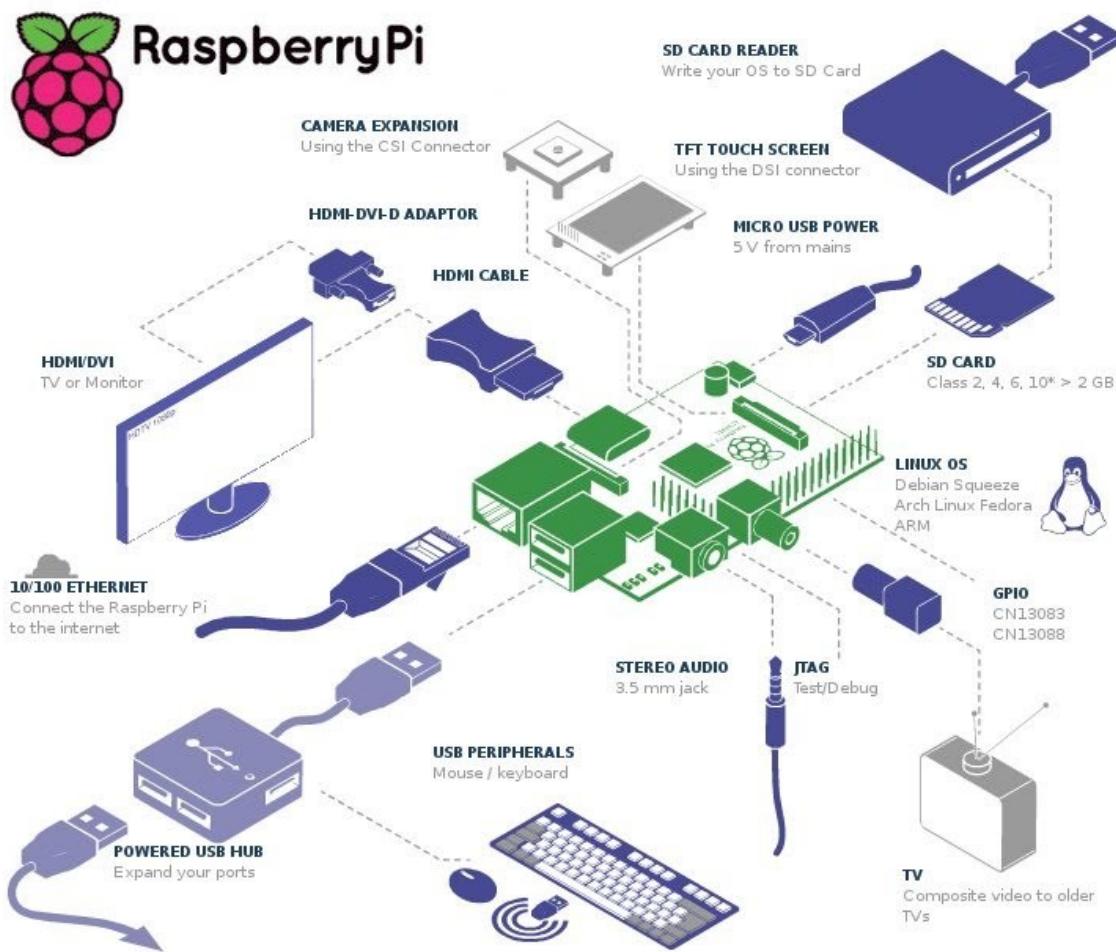


Ilustración 24 - Interfaces de Raspberry Pi

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 Wheezy
- 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 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 (**Ampere**), 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 carcasa 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 25 -
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 26 - 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 encasta 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**.



Ilustración 27 - Adafruit
Prototyping Pi

En la imagen (**Ilustración 27 - Adafruit Prototyping Pi**) se puede ver esta placa empalmada sobre una Raspberry 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 28 - 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 de pantalla** es de 320x240 y color de 16 bits. Se le pueden soldar 4 botones de forma opcional para su manipulación.



Ilustración 29 - Pi TFT

4.6 Ventajas del uso de Raspberry Pi

Al igual que lo que se mencionó en el capítulo 3 (**Capítulo 3 – Arduino**) sobre 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 proyectos⁴, que dado esto, se expanden con el 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.

⁴ 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>).

- **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.

Resumen

En este capítulo se habló sobre el computador de placa reducida (SBC) Raspberry Pi, especificando las fichas técnicas de las versiones más populares de la plataforma. Además, se explicó el concepto de GPIO detallando los que integran a las Raspberry.

Por otro lado, se revisaron diversos sistemas operativos que funcionan en esta plataforma, además de variados accesorios que sirven de complemento para la utilización más “amigable” de la misma.

Finalmente se describieron ventajas del uso de la Raspberry Pi, en relación con otras plataformas destinadas al mismo objetivo.

Capítulo 5 - Aplicaciones Móviles

En este capítulo se verá que sistemas operativos se utilizan en plataformas móviles. Se revisarán, como los dispositivos móviles toman mayor relevancia en el mercado de **Internet**, debido a que sus aplicaciones son de alta demanda por parte de los usuarios. Se analizarán los modos de construcción de aplicaciones diferenciando las nativas, las webs y las híbridas. Al respecto, HTML5 como tecnología de desarrollo emergente, permite la utilización de los conocimientos de aplicaciones web y, por otro lado, las apps nativas ofrecen un mayor rendimiento. La brecha entre estas dos técnicas de desarrollo, deviene en las App Híbridas, tomando ventajas de cada modalidad. El advenimiento de tecnologías como Cordova, IntelXDK, Ionic y la popularización de HTML5, ha logrado que la comunidad de desarrolladores comience a apostar a estos **Frameworks** basados en tecnologías de **Front-End** para el desarrollo de aplicaciones móviles. [15] [16]

5.1 Las Aplicaciones móviles

“Una aplicación móvil o App (**Ilustración 30 - Aplicaciones móviles**) 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”. [17]

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

- 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 aislado para cada aplicación. En el caso de Android, basado en **Linux**, este concepto se lo denomina **Sandboxing**, limitando 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.
- Una gran flexibilidad en cuanto a su utilización, dada la facilidad que presenta a la hora del manejo por parte del usuario, al ser necesario solo instalarla y teniendo como ventaja que el sistema operativo se encarga de mantenerla actualizada.



Ilustración 30 - Aplicaciones móviles

- Acceso a funcionalidades específicas del dispositivo. Esto se debe a que las App nativas acceden directamente a recursos hardware (cámara, contactos, memoria, notificaciones *push*, etc.).
- Mejorar la conectividad y disponibilidad de servicios y productos entre usuarios, y usuarios con proveedores de servicios).

5.1.1 Las Web Apps

Una Web App (**Ilustración 31 - App nativa vs Web App**) es una versión de una página web optimizada y adaptable a un gran número de dispositivos móviles independientemente del sistema operativo que utilice. Esta optimización es posible gracias a características provistas por lenguaje de marcado HTML5, combinado con hojas de estilo en cascada CSS3, que permiten proveer adaptabilidad, denominada en inglés “*Responsive Web Design*”. El diseño web responsive 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 (**Ilustración 32 – WebApps – Diseño multipropósito**). Se caracteriza porque los *layout* y los contenidos multimediales son fluidos y se utiliza código media-queries de CSS3. [18]



Ilustración 31 - App nativa vs Web App

5.1.2 Ventajas de las Web-App:

- No ocupa espacio de memoria de almacenamiento en los dispositivos que la ejecutan.
- 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 resto de los tipos de Apps.

5.1.3 Desventajas de las Web-Apps

- No permite la promoción y distribución a través de los *markets* o tienda de aplicaciones (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 **LAN**).
- Menor usabilidad, al ofrecer un acceso muy limitado a los elementos y capacidades hardware del dispositivo.
- Carece de un ícono de lanzamiento específico.
- Necesitan de un espacio web.
- No funcionan en segundo plano (multitarea)



Ilustración 32 – WebApps – Diseño multipropósito

5.2 Sistemas operativos para dispositivos móviles

Al igual que en una computadora, las aplicaciones que se han mencionado se ejecutan sobre un sistema operativo móvil (SO). Se componen de un conjunto de programas de bajo nivel que permiten la abstracción de las peculiaridades del hardware específico del aparato y proveen servicios a las aplicaciones. Al igual que los dispositivos de computación tradicionales dónde se utilizan Windows, **Linux** o Mac OS, en el caso de los móviles los SO son Android, iOS o Windows Phone, entre otros.

A medida que los dispositivos móviles crecen en popularidad, sus SO adquieren mayor importancia. La cuota de mercado de sistemas operativos móviles en el primer trimestre de 2016 fue el siguiente sobre una base de 6600 millones de dispositivos:

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

Android tiene la mayor cuota, desde enero del 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 en el desarrollo de una App para operar el SAR sobre esta plataforma.

5.3 Android

Se encuentra basado en **Linux (Ilustración 33 - Arquitectura de Android)**, diseñado originalmente para cámaras fotográficas profesionales, luego fue vendido a Google y modificado para ser utilizado en dispositivos móviles como los teléfonos inteligentes y posteriormente 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 ha dedicada al desarrollo de estándares abiertos para celulares, ayudando en gran medida a la masificación del SO de Google, hasta el punto de que estos estándares son usados por empresas como HTC, LG, Samsung, Motorola entre otros.

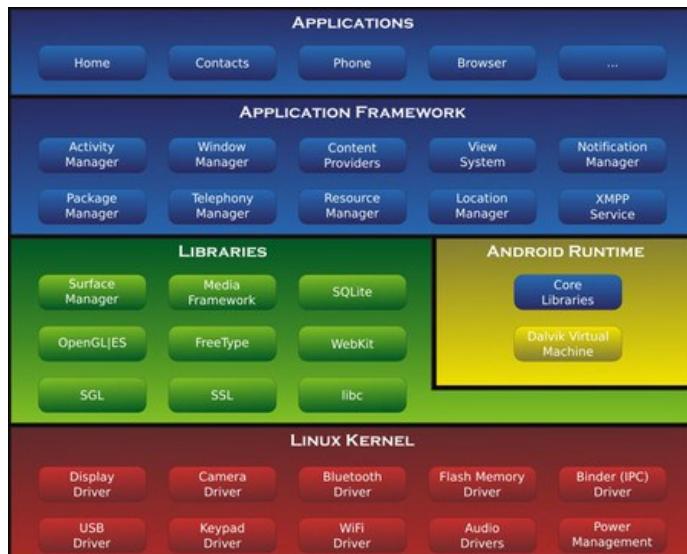


Ilustración 33 - Arquitectura de Android



Ilustración 34 - Logo de Android

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

En la imagen se puede apreciar el logo oficial de Android.

5.4 Aplicaciones móviles multiplataforma

5.4.1 Diferencias entre aplicaciones y web móviles

Una aplicación móvil debe ser descargada e instalada para ser usada, mientras que una web puede accederse simplemente teniendo conexión a **Internet** y un navegador compatible. Pero estas últimas siempre pueden presentarse correctamente desde una pantalla generalmente más pequeña que la de un ordenador de escritorio.

Las “webs responsivas” (**5.1.1 Las Web Apps**) son como un subconjunto de las aplicaciones web y utilizan conceptos como el “diseño líquido” para que su contenido aproveche la forma del contenedor.

Previo a la existencia del CSS3, se carecía de tecnología para poder crear sitios “elásticos”, es decir, que su disposición se adapte a cualquier dimensión y relación de aspecto de pantalla, por lo tanto, los desarrolladores de web estaban obligados a crear diferentes versiones de las páginas web. CSS3 provee mecanismos como las consultas de medio de presentación (**media Querys**) para que las páginas puedan reaccionar ante distintas circunstancias como el cambio de ancho de la pantalla (como cuando ocurre una rotación). En conclusión, **Web responsive** se denomina a todas aquellas técnicas (no solamente redimensionado de pantalla) que permiten la adaptabilidad del contenido a los dispositivos terminales.

5.4.2 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 disponible a través del proveedor del dispositivo. 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 una para cada SO, implicando la utilización no solo de múltiples **APIs**, sino también de distintos lenguajes según 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.

Las aplicaciones nativas, como se mencionó anteriormente, tienen acceso a las características específicas de hardware, además de la capacidad de ser ejecutadas sin necesidad de conectividad a **Internet**. Por otro lado, estas Apps son promocionadas por medio de las tiendas de aplicaciones, que facilitan su descarga y ofrecen un mejor rendimiento que las alternativas de desarrollo. En la siguiente imagen (**Ilustración 35 - Cuadro comparativo - Aplicaciones nativas**) se mencionan algunas ventajas e inconvenientes que se presentan con este tipo de aplicaciones.

Ventajas	Inconvenientes
<ul style="list-style-type: none"> • Acceso completo al dispositivo • Mejor experiencia del usuario • Visibilidad en APP Store • Envío de notificaciones o “avisos” a los usuarios • La actualización de la app es constante 	<ul style="list-style-type: none"> • Diferentes habilidades / idiomas / herramientas para cada plataforma de destino • Tienden a ser más caras de desarrollar • El código del cliente no es reutilizable entre las diferentes plataformas

Ilustración 35 - Cuadro comparativo - Aplicaciones nativas

5.4.3 Desarrollo de Web Apps

Una aplicación web o Web App es 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 a 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.

A continuación (**Ilustración 36 - Cuadro comparativo - Aplicaciones Web**), algunas ventajas e inconvenientes de este tipo de aplicaciones.

Ventajas	Inconvenientes
<ul style="list-style-type: none"> • El mismo código base reutilizable en múltiples plataformas • Proceso de desarrollo más sencillo y económico • No necesitan ninguna aprobación externa para publicarse (a diferencia de las nativas para estar visibles en app store) • El usuario siempre dispone de la última versión • Pueden reutilizarse sitios “responsive” ya diseñados 	<ul style="list-style-type: none"> • Requiere de conexión a internet • Acceso muy limitado a los elementos y características del hardware del dispositivo • La experiencia del usuario (navegación, interacción..) y el tiempo de respuesta es menor que en una app nativa • Requiere de mayor esfuerzo en promoción y visibilidad

Ilustración 36 - Cuadro comparativo - Aplicaciones Web

5.4.4 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.

Tanto PhoneGap como Apache Cordova, son los **Frameworks** más utilizados por los programadores para el desarrollo multiplataforma de aplicaciones híbridas. [19] [20]

5.4.5 Creación de una Aplicación híbrida

Consiste en diseñar la aplicación como si fuera una Web App para ser ejecutada en el navegador del cliente. La facilidad de desarrollo debe ser balanceada con una experiencia de usuario y apariencia en principio inferior a una aplicación nativa.

5.4.6 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.

En la siguiente ilustración (**Ilustración 37 - Comparativa aplicaciones híbridas**), ventajas e inconvenientes de las aplicaciones híbridas

Ventajas	Inconvenientes
<ul style="list-style-type: none"> • Es posible distribuirla en las tiendas de iOS y Android. • Instalación nativa pero construida con JavaScript, HTML y CSS • El mismo código base para múltiples plataformas • Acceso a parte del hardware del dispositivo 	<ul style="list-style-type: none"> • Experiencia del usuario más propia de la aplicación web que de la app nativa • Diseño visual no siempre relacionado con el sistema operativo en el que se muestre

Ilustración 37 - Comparativa aplicaciones híbridas

5.5 Entornos y herramientas para el desarrollo

En el contexto de esta tesina, al momento de seleccionar el tipo de aplicación móvil, dentro de los tipos antes mencionados, se analizaron los entornos de desarrollo y herramientas para el desarrollador que a continuación se describen (**Ilustración 38 - Herramientas para desarrollo de apps**).



Ilustración 38 - Herramientas para desarrollo de apps

5.5.1 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 **Linux**, MS Windows, MacOs. Además, incluye características 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

Provee también emuladores para diferentes plataformas de hardware, destinados a la prueba de Apps. [21]

5.5.2 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 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. [22]

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

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

A continuación, se mencionan tecnologías utilizadas tanto para aplicaciones web como, Web Apps y aplicaciones híbridas.

5.5.3.1 HTML

Es un lenguaje de marcado para la elaboración de páginas 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.

5.5.3.2 CSS

Como se mencionó anteriormente (**5.4.3 Desarrollo de Web Apps**), 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.

5.5.3.3 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.

5.5.3.4 SASS

Es un metalenguaje de estilos en cascada. Es un lenguaje que, pre-procesado 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 permiten reutilizar valores que podemos manejar desde un solo sitio de forma sencilla y centralizada

Un mixin permite aprovechar un fragmento de código al que podemos llamar repetidamente, evitando repetición.

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

5.5.3.5 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).

5.5.4 Cordova

Apache Cordova es un entorno de desarrollo de aplicaciones móviles, originalmente creado por Nitobi y comprado por Adobe. 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 nativa (debido a que la representación gráfica se realiza con vistas Web) ni puramente basadas en web (están empaquetadas como aplicaciones para su distribución y tienen acceso a las **APIs** nativas del dispositivo en lenguaje JavaScript). [23] [20]

5.5.5 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,

pre-visualización de aplicaciones mediante el scan de un código QR, Drag and Drop y soporte de plantillas. Brinda soporte para Android, iOS, Windows Phone, entre otras plataformas.

5.5.6 Ionic

Es un **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.

5.5.6 Meteor

Es una plataforma para crear aplicaciones webs 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.

5.5.7 Meteor y Cordova

Existe una integración del **Framework** Meteor con Cordova, que permite que una aplicación web creada con Meteor, sea ejecutada en un dispositivo iOS o Android de forma híbrida. Un beneficio importante de empaquetar la aplicación web como una aplicación de Cordova es que todos sus recursos no deben ser descargados desde la web, 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.

Resumen

Como se vio en este capítulo, las aplicaciones móviles son aplicaciones informáticas diseñadas para ser ejecutadas en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permite al usuario efectuar una tarea con mayor versatilidad que con una computadora de escritorio.

Se detallaron ventajas y desventajas sobre cada tipo de aplicación y su forma de desarrollo; como, por ejemplo, las Apps nativas no requieren de conectividad a **Internet** en comparación a las Apps web. Las Apps hibridas poseen ventajas agregadas de las otras dos.

Capítulo 6 – Stack MEAN

En este capítulo, se analizará el stack MEAN y sus componentes. El mismo está compuesto por un conjunto de tecnologías que responden al siguiente acrónimo: MongoDB (acceso a datos), Express (**Framework** web, **Back-End**), Angular (**Framework** web, **Front-End**) y NodeJS (plataforma de aplicación web). Además, otros complementos, como Compodoc (documentador), bibliotecas y **Framework** aplicados a la vista (o **Front-End**) 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”. Más adelante, se visualiza el logotipo de este stack de tecnologías (**Ilustración 39 - Acrónimo MEAN**). [24]



Ilustración 39 - Acrónimo MEAN

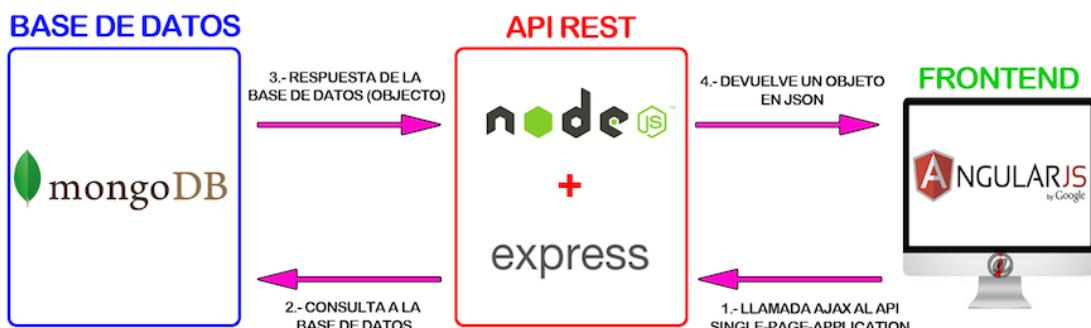


Ilustración 40 - Arquitectura de interacción MEAN

6.2 Componentes de MEAN

Como podemos apreciar en la imagen (**Ilustración 40 - Arquitectura de interacción MEAN**) se puede observar que componentes interactúan entre si dentro del *stack*. En esta sección se ampliará cada tecnología.

En un primer momento la aplicación Angular se encuentra almacenada en un servidor WEB. Cuando es transferida al cliente (**Navegador web**) comienza la comunicación entre el componente Angular y el servidor NodeJS, por medio de una **API** REST implementada sobre el **Framework** Express. Los requerimientos producidos por los *endpoints* de la **API** provocan que NodeJS realice las consultas tanto de lectura como escritura sobre MongoDB. El formato de presentación de datos es JSON (**6.3.3 JSON**) para todos los requerimientos.

6.2.1 MongoDB

Es un sistema de base de datos NoSQL, en el cual la información se almacena documentos en vez de filas en una tabla. Cada documento se trata de una estructura con formato JSON (notación simple de objeto tipo JavaScript). Estos documentos son agrupados en colecciones en contraposición a las tablas de un RDBMS. Debido a la ausencia de comprobación de integridad referencial tiene un alto desempeño.

6.2.2 Express

Es un paquete de NodeJS que ofrece una interface mínima para manejo de solicitudes o peticiones **HTTP**. Uno de sus componentes principales se trata de un sistema de enrutamiento (Routing), que asocia URLs con funciones. Dentro del MEAN *stack* opera del lado del servidor, también conocido 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 proponer un diseño de aplicaciones basadas en navegadores webs, utilizando el patrón Modelo Vista Controlador (MVC), facilitando el desarrollo y las pruebas.

6.2.4 NodeJS

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

Utiliza el motor de ejecución de JavaScript de Google, denominado V8 (**Ilustración 41 - Logo del motor V8**), y presenta una arquitectura orientada a eventos, en



Ilustración 41 - Logo del motor V8

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 llamada LibUV (Unicornio Velociraptors), que proporciona soporte de E/S asíncronas basada en bucles de eventos. [25]

Si bien NodeJS se puede utilizar para crear cualquier tipo de aplicación, dado que incorpora un módulo de servidor web dentro de su biblioteca estándar, es especialmente popular para crear aplicaciones web. Por lo cual lo ha popularizado entre empresas que se dedican a servicios basados en **Internet**. Su uso no se encuentra limitado a web, sino que también existen aplicaciones de línea de comandos, scripts para administración de sistemas, aplicaciones de red, etc. Su utilización es recomendada en aplicaciones concurrentes por E/S como: chats, **APIREST**, entrada de datos concurrentes, aplicaciones cuya interacción sea con servicios bloqueantes como escritura en RDBMS, procesamiento de archivos, transmisión de datos, proxies, aplicaciones como corredores de bolsa (tiempo real), visualización de interacciones, etc. La principal razón de su utilización en la construcción y escalabilidad de aplicaciones de red, es su capacidad de afrontar la concurrencia mediante el procesamiento de eventos de manera no bloqueante (también conocido como, *event-driven I/O*). En la siguiente imagen podemos apreciar la comparativa entre los servidores tradicionales y NodeJS (**Ilustración 42 Comparativa de servidores tradicionales y NodeJS**).

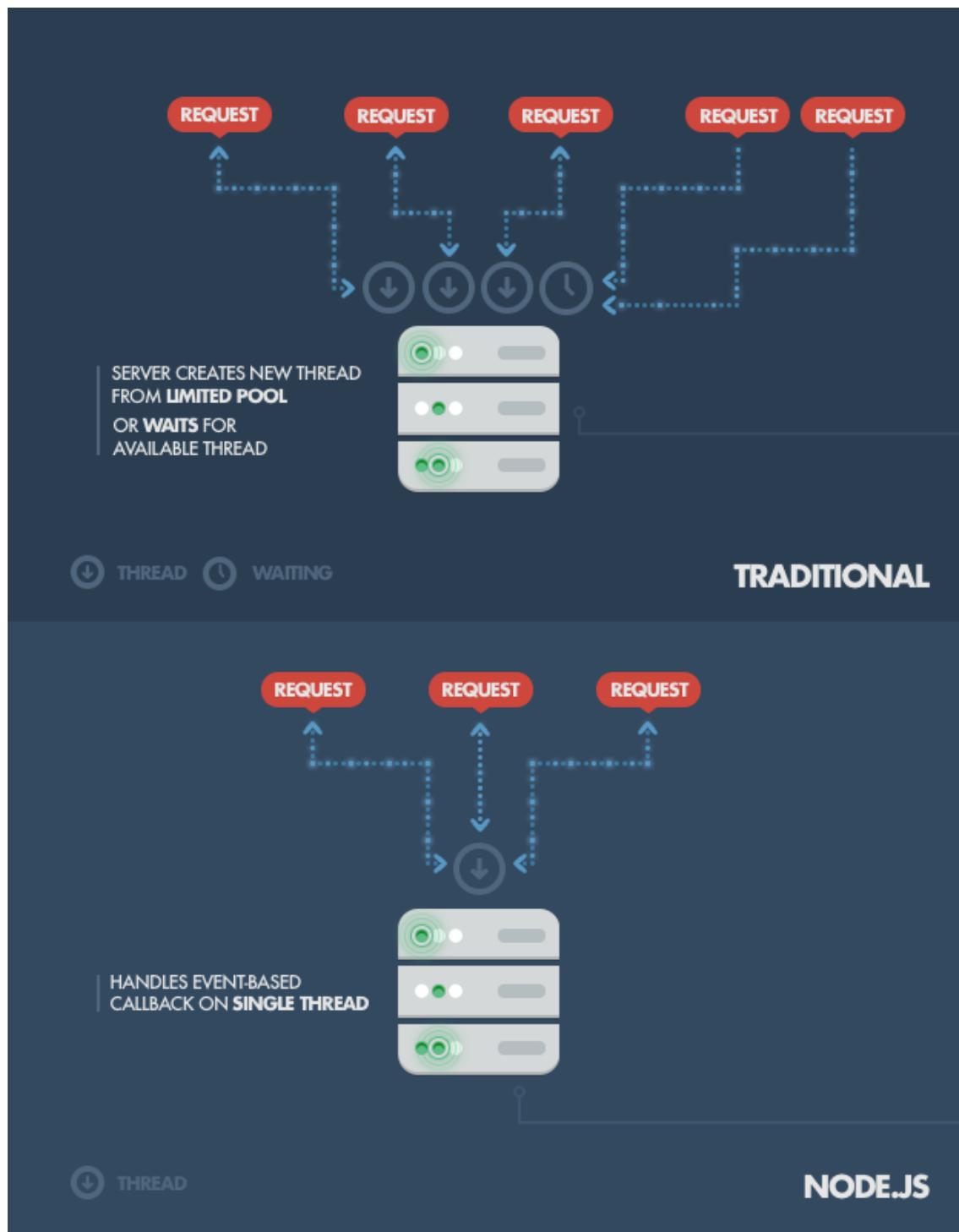


Ilustración 42 Comparativa de servidores tradicionales y NodeJS

6.3 Otros complementos

6.3.1 Twitter Bootstrap

Se trata de uno de los **Frameworks** más populares que integra **HTML**, CSS, y JS para el desarrollo de aplicaciones web del lado del cliente adaptables, es decir, que su presentación aproveche los diferentes medios de reproducción (Responsive).

Dentro de las ventajas que presenta este **Framework** son:

- Facilita un sistema de maquetado por columnas.
- Cuenta con el soporte de una amplia comunidad.
- Admite la reconfiguración y recopilación mediante lenguajes como **LESS**.

6.3.2 Compodoc

Se trata de un generador de documentación, compatible con todas las definiciones de **API** de Angular. Genera contenidos estáticos, “responsivos” y provee de un sistema de búsqueda basado en lunr.js para indexar los componentes, módulos, servicios y modelos.

6.3.3 JSON

Es el acrónimo de JavaScript Simple Object Notation, en la imagen (**Ilustración 43 - Logo de JSON**) se puede ver su logo oficial. 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-marshallizadas para recuperar los objetos originales.

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 él. 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**.

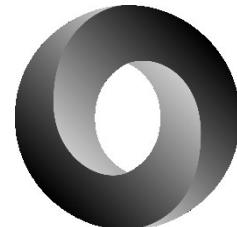


Ilustración 43 - Logo de JSON

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 44 - Json pegamento de tecnologías**).

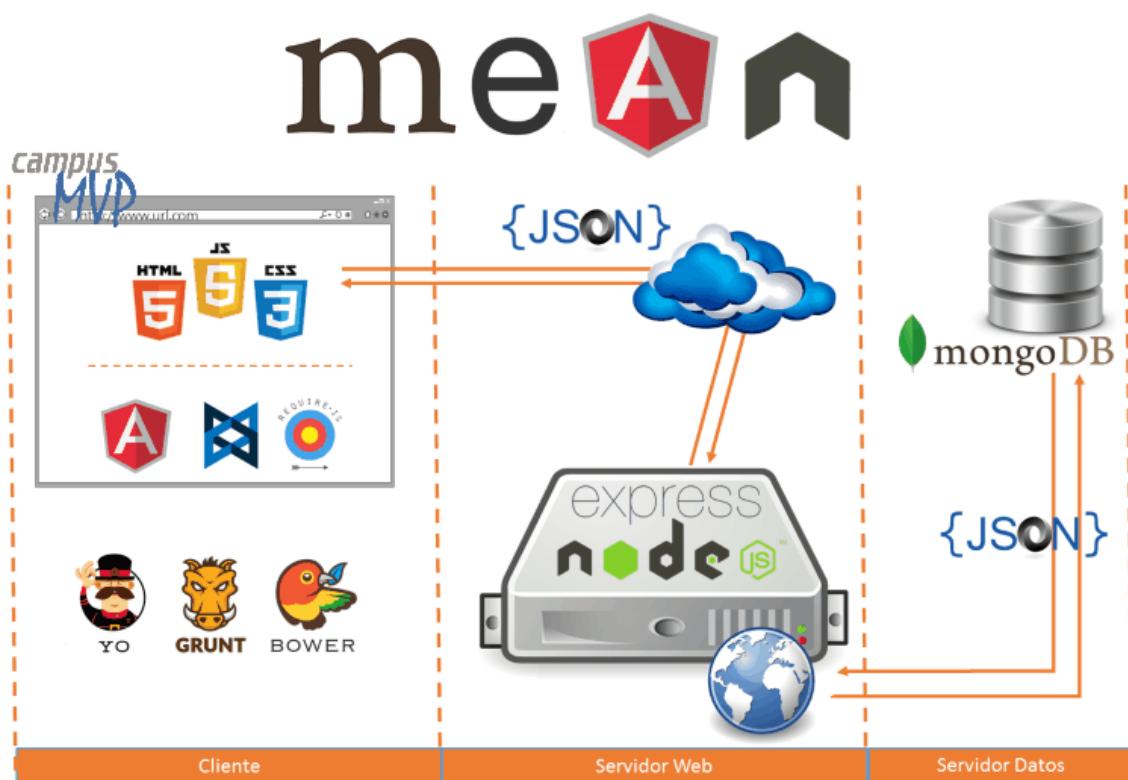


Ilustración 44 - Json pegamiento de tecnologías

6.3.3 JQuery

Es una biblioteca multiplataforma de JavaScript del lado del cliente, que permite simplificar la manera de interactuar con los documentos **HTML**, manipular el árbol **DOM**, manejar eventos, desarrollar animaciones y agregar interacción mediante la simplificación de la utilización de AJAX.

Resumen

En este capítulo, se vió el concepto de MEAN y sus componentes. El mismo está compuesto por un conjunto de tecnologías respetando el acrónimo como sigue: MongoDB, Express, Angular y NodeJS y que todas ellas se comunican mediante el formato JSON.

Finalmente, se analizaron distintas herramientas complementarias como Compodoc (documentador), **Frameworks** y bibliotecas orientados a la vista como Bootstrap y JQuery.

Capítulo 7 – Comunicación NodeJS con Arduino

Este capítulo introduce sobre un **Framework** denominado Johnny-five el cual es utilizado para las comunicaciones entre la aplicación web y las placas Arduinos que componen al SAR. Además, se explica en detalle el protocolo subyacente a esta librería, conocido como Firmata, que en esta tesina es implementado en cada uno de los Arduinos.

7.1 Johnny-five

Johnny-five, más conocido como J5, es un **Framework** de programación robótica basado en JavaScript lanzado por la compañía Bocoup en el 2012, bajo licencia abierta, y que ha logrado ser adoptado tanto por desarrolladores como por ingenieros. Estos

no solo son usuarios, sino que, como comunidad también proveen soporte, mejoras y nuevas características al **Framework**. Una captura de pantalla de su sitio se encuentra en la **Ilustración 45 - Sitio web oficial de Johnny-Five (<http://johnny-five.io/>)**.

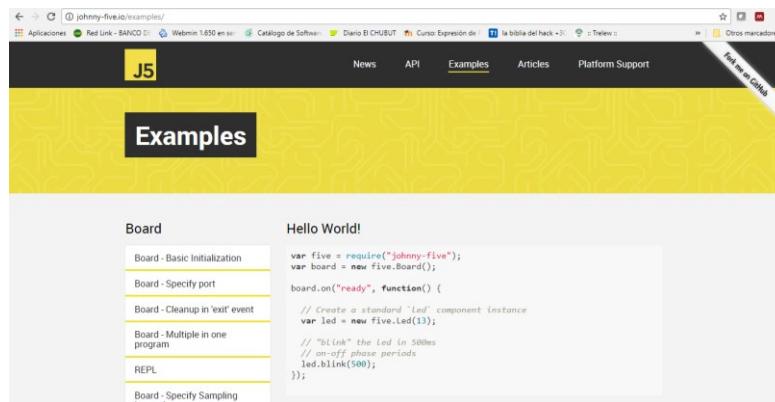


Ilustración 45 - Sitio web oficial de Johnny-Five (<http://johnny-five.io/>)

Es compatible con la gran mayoría de microcontroladores y SBC más populares, incluyendo los utilizados en el desarrollo de esta tesina: Arduino y Raspberry Pi (ambos en todas sus versiones). Dentro de la variedad de dispositivos soportados se encuentran las placas como BeagleBone, SparkFun, ChipKit, Intel-Galileo, entre otras.

Su librería posee compatibilidad a una gran cantidad de sensores y actuadores, así como ejemplos de utilización. Éstos se encuentran disponibles en su página oficial, complementado con esquemas de conexión de los componentes, para cada plataforma soportada.

7.2 Instalación

Para poder utilizar Johnny-Five, se debe contar con NodeJS (visto en el apartado **6.2.4 NodeJS**). Se realiza a través del gestor de paquetes de NodeJS, llamado npm, que permite la gestión de proyectos y administración de software de terceros.

Dentro del directorio del proyecto se debe ejecutar la siguiente orden:

```
npm install Johnny-five
```

7.3 Arduino Firmata

Firmata se trata de un protocolo serial, genérico orientado a la comunicación entre microcontroladores y una computadora. Al ser genérico, se puede implementar en cualquier arquitectura de microcontroladores, así como también las bibliotecas para utilizarlo desde la computadora se pueden implementar en cualquier lenguaje.



Ilustración 46 – Firmata como interfaz

Su objetivo es permitir controlar completamente un microcontrolador de forma remota (**Ilustración 46 – Firmata como interfaz**), eliminando la necesidad de la escritura de código específico para cada microcontrolador. [26]

Podemos enumerar las siguientes ventajas:

- El programa no está limitado por la memoria RAM ni Flash de Arduino.
- El software de control se puede programar en cualquier lenguaje que tenga soporte para Firmata. Algunos lenguajes son, por ejemplo: Firmata: Processing, Visual Basic, Perl, C#, PHP, Java, **JavaScript** (a través de Jhonny-Five u otras bibliotecas), Ruby, Python, etc.

Sin embargo, podemos enumerar también una serie de Desventajas:

- Programas más restringidos, operaciones específicas como manejo pormenorizado de interrupciones no es posible.
- El microcontrolador deja de ser autónomo, es decir, depende de la conexión al computador para poder recibir comandos.

7.4 Surgimiento y funcionamiento de Firmata

El protocolo surge con el objetivo de hacer del microcontrolador una extensión del entorno de desarrollo, quitando del foco la programación embebida y permitiendo el uso de lenguajes de mayor nivel de abstracción o familiaridad con el desarrollador.

Su creación data del año 2006 como una demostración para Arduino por Hans-Chistoph Steiner, mientras trabajaba en un proyecto musical con dispositivos MIDI interconectados a varios Arduinos, que contaban con varios sensores. La problemática de Hans era que debía replicar en cada microcontrolador cada cambio realizado.

Actualmente la implementación de referencia se encuentra en la biblioteca de Arduino, incluida a partir de la versión 0012 y está disponible para cualquier placa compatible.

Firmata expone la **API** de Arduino en la computadora conectada al microcontrolador, permitiendo la operación remota. A través del puerto serie se codifican los mensajes utilizando la codificación del protocolo MIDI (Musical Instrument Digital Interface), un protocolo orientado a la comunicación de dispositivos musicales con computadoras.

Estos mensajes se conforman de bytes de comando seguidos de bytes de datos. Los bytes de comando son 8 bits y los bytes de datos son 7 bits. Por ejemplo, el mensaje MIDI Channel Pressure (Comando: 0xD0) tiene 2 bytes de longitud, en Firmata se utiliza para habilitar informes para un puerto digital (colección de 8 pines). Si bien Firmata respeta la cantidad de bytes de cada comando MIDI que reemplaza, hace uso de los mensajes de Midi System Exclusive (Sysex) para comunicaciones dónde se necesite un mensaje de longitud arbitraria.

Firmata permite operar tanto entradas y salidas analógicas, como digitales. Soporta más de 16 pines analógicos con una resolución de 14 bits y más de 128 pines digitales.

Por ser de código abierto, se han implementado diferentes versiones con características específicas para dar soporte a gran variedad de funcionalidades más allá de la operación de E/S analógica y digital. La versión original, Standard_Firmata, se incluye dentro de las versiones del entorno oficial de Arduino y Wiring e incluye soporte para las siguientes características [27]:

- Entradas y salidas analógicas
- Salidas PWM
- Comutación entre entradas y salidas analógicas
- Control de Servomotores
- Matrices de LEDs
- Comunicación I2C

7.5 Métodos de librería Firmata en Arduino

La librería Firmata de Arduino cuenta con un conjunto de métodos, relacionados con la **API** de Arduino (como se describió en el apartado anterior). A continuación, se detallan los más relevantes [28]:

7.5.1 Métodos de propósito general

- **`begin(long)`**: Comienza la librería, es posible utilizar otra velocidad diferente a la velocidad por defecto que es 57600 baudios. También es posible iniciar el protocolo Firmata desde otro Stream que no sea el que viene por defecto que es Serial.
- **`printVersion()`**: Envía la versión del protocolo al ordenador.
- **`blinkVersion()`**: Parpadea la versión de protocolo en el “build in LED”, generalmente el pin 13.

- **printFirmwareVersion()**: Envía la versión de firmware y su versión al ordenador.
- **setFirmwareVersion(byte major, byte minor)**: Configura la versión del firmware.
- **setFirmwareNameAndVersion(const char *name, byte major, byte minor)**: Configura nombre y versión del firmware.

7.5.2 Métodos para el envío de mensajes

- **sendAnalog(byte pin, int value)**: Envía el valor del pin analógico.
- **sendDigitalPort(byte portNumber, int portData)**: Envía el valor de un puerto digital de 8 bits.
- **sendString(const char* string)**: Envía un string a una computadora.
- **sendString(byte command, byte bytec, byte *bytev)**: Envía un string a la computadora usando un tipo de comando.
- **sendSysex(byte command, byte bytec, byte* bytev)**: Envía un comando un con array de bytes
- **write(byte c)** – Envía un byte al stream de datos.

7.5.3 Métodos para la recepción de mensajes

- **available()**: Comprueba si hay algún mensaje entrante en el buffer.
- **processInput()**: Procesa los mensajes entrantes que hay en el buffer, mandado los datos a cualquiera de las funciones de callback registradas.
- **attach(byte command, callbackFunction myFunction)**: Registrar una función a un tipo de mensaje entrante.
- **detach(byte command)** : Desvincular la función del tipo de mensaje

7.5.4 Otros métodos

- **sendValueAsTwo7bitBytes(int value)**: Escribe el valor como 2 bytes.
- **startSysex(void)**: Comienza mensaje sysex.
- **endSysex(void)**: Finaliza mensaje sysex.

7.6 Instalación de Firmata en Arduino

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. 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 47 - 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.

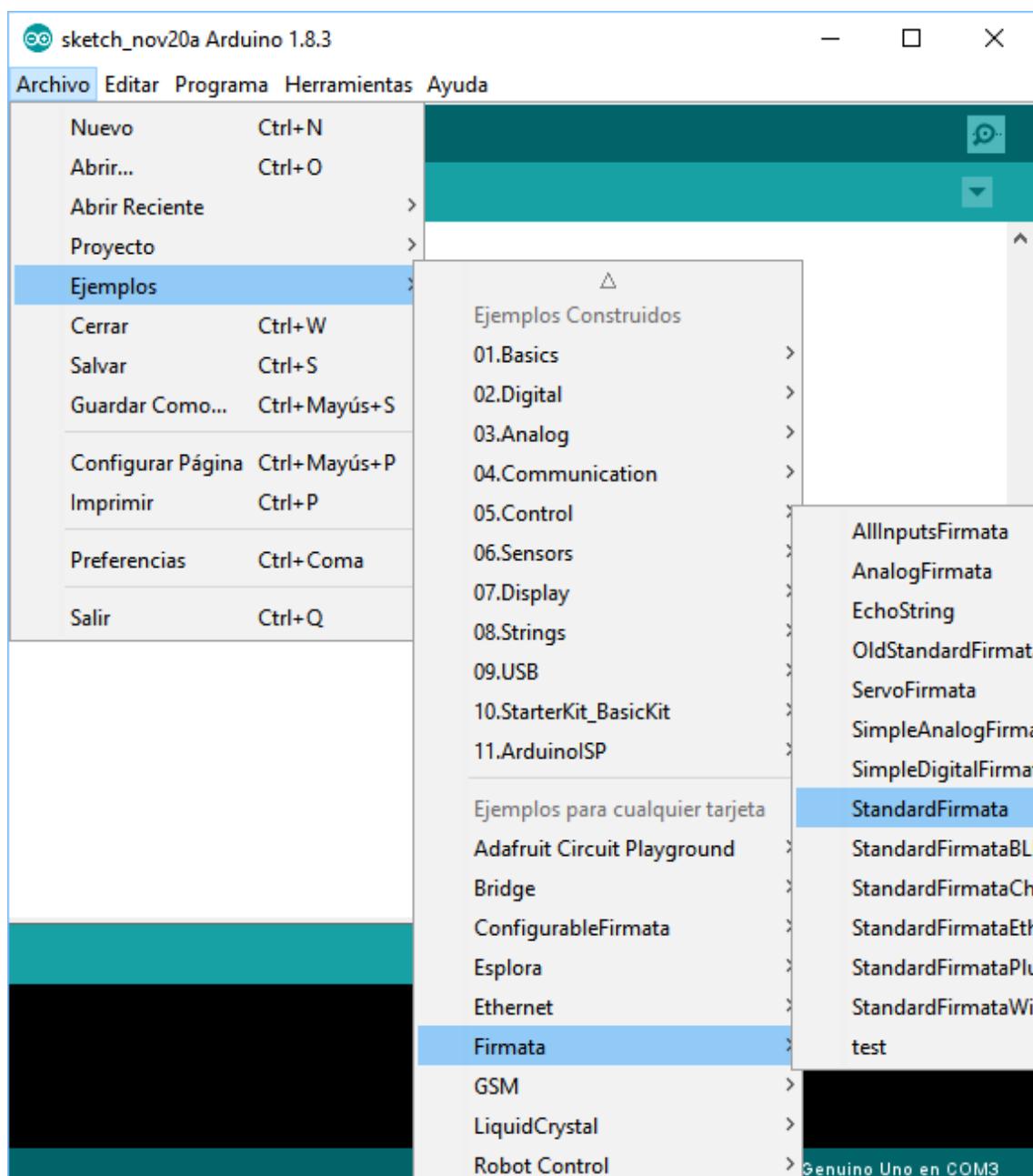
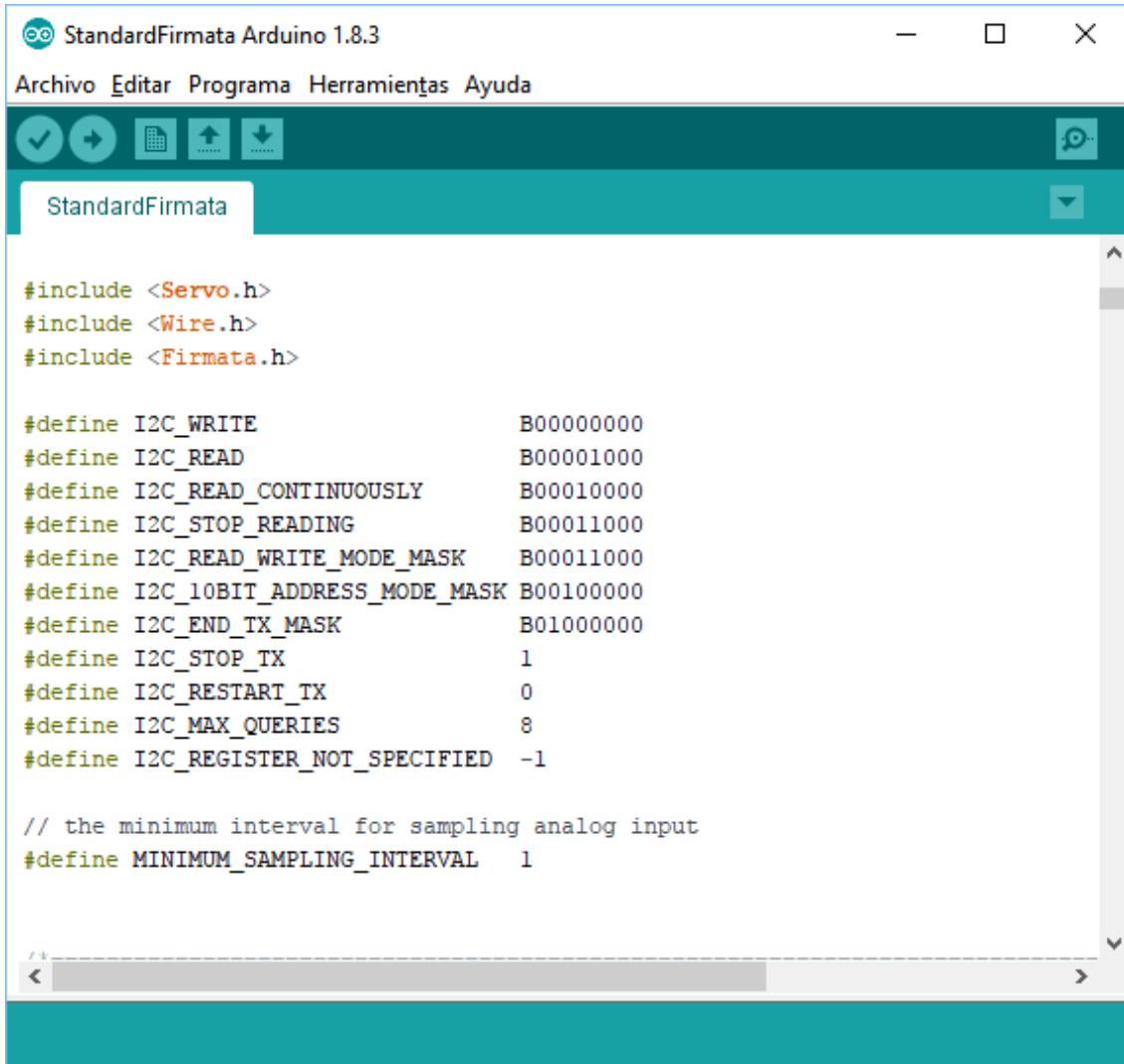


Ilustración 47 - IDE de Arduino

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. (**Código StandardFirmata utilizado en el Arduino MEGA**). En la siguiente imagen (**Ilustración 48 - Código StandardFirmata**) se puede apreciar el código, de este protocolo, abierto desde el **IDE** de Arduino.



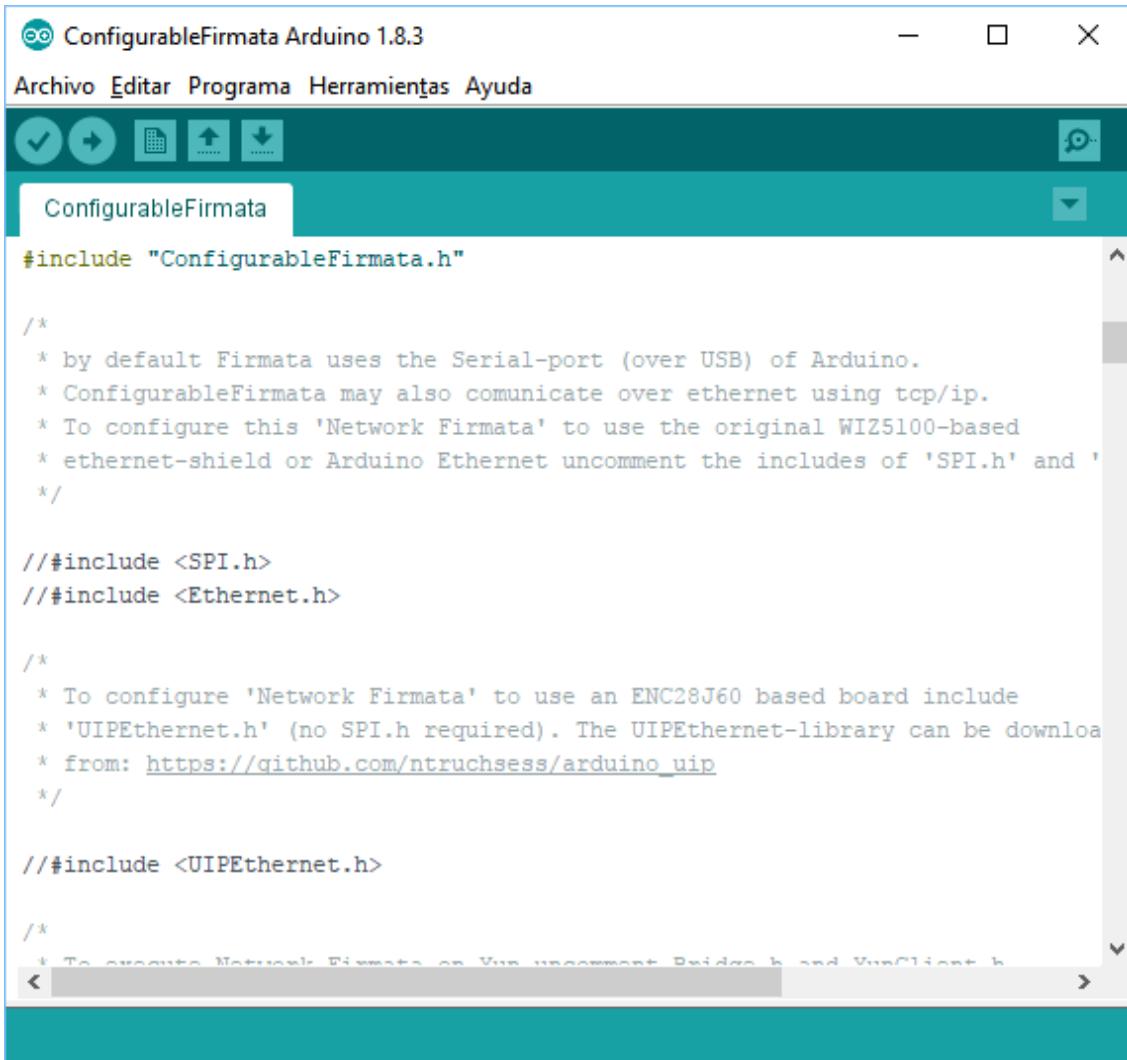
```
#include <Servo.h>
#include <Wire.h>
#include <Firmata.h>

#define I2C_WRITE B00000000
#define I2C_READ B00001000
#define I2C_READ_CONTINUOUSLY B00010000
#define I2C_STOP_READING B00011000
#define I2C_READ_WRITE_MODE_MASK B00011000
#define I2C_10BIT_ADDRESS_MODE_MASK B00100000
#define I2C_END_TX_MASK B01000000
#define I2C_STOP_TX 1
#define I2C_RESTART_TX 0
#define I2C_MAX_QUERIES 8
#define I2C_REGISTER_NOT_SPECIFIED -1

// the minimum interval for sampling analog input
#define MINIMUM_SAMPLING_INTERVAL 1
```

Ilustración 48 - Código StandardFirmata

ConfigurableFirmata: Esta versión personalizada del protocolo⁵, es la utilizada para captar la temperatura mediante el sensor para dicho fin conectado a un Arduino Nano. Permite separar las características del protocolo en clases individuales, haciendo más sencillo mezclar las características estándar del protocolo con otras personalizadas. (**Código ConfigurableFirmata utilizado en el Arduino NANO**). En la siguiente imagen (**Ilustración 49 - Código ConfigurableFirmata**) se puede apreciar el código, de este protocolo, abierto desde el **IDE** de Arduino.



```
#include "ConfigurableFirmata.h"

/*
 * by default Firmata uses the Serial-port (over USB) of Arduino.
 * ConfigurableFirmata may also communicate over ethernet using tcp/ip.
 * To configure this 'Network Firmata' to use the original WIZ5100-based
 * ethernet-shield or Arduino Ethernet uncomment the includes of 'SPI.h' and '
 */

//#include <SPI.h>
//#include <Ethernet.h>

/*
 * To configure 'Network Firmata' to use an ENC28J60 based board include
 * 'UIPEthernet.h' (no SPI.h required). The UIPEthernet-library can be download
 * from: https://github.com/ntruchsess/arduino\_uip
 */

//#include <UIPEthernet.h>

/*
 * To comment Network Firmata in View comment Board and Variant in

```

Ilustración 49 - Código ConfigurableFirmata

⁵ Se puede obtener del siguiente sitio web
<https://github.com/firmata/ConfigurableFirmata>

Resumen

En este capítulo, se abordó la utilidad en el SAR de un **Framework** denominado Johnny-five, el cual es utilizado para la programación robótica y está basado en JavaScript, utilizando Firmata.

A continuación, se definió este protocolo mencionando ventajas y desventajas en su utilización. Se explicó su surgimiento y su funcionamiento en general, además del formato de sus mensajes. Finalmente se vio como instalar y/o cargar este protocolo en la familia de placas Arduino y los dos códigos utilizados en el desarrollo de esta tesina (StandardFirmata y ConfigurableFirmata)

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

En capítulos anteriores, se analizaron diversas tecnologías hardware y software relacionadas con la robótica. Dentro de las tecnologías hardware se investigaron las plataformas Arduino (**3.1 Arduino**) y Raspberry Pi (**Capítulo 4 – Raspberry Pi**). De las diversas herramientas en tecnologías software para aplicaciones móviles, se investigaron aquellas que permiten desarrollar una aplicación que interactúe con el hardware mencionado.

En este capítulo se examinan dichas tecnologías, para concluir cuales son las que integran el SAR. Para ello, se realizaron 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, basados en factores como rendimientos, tiempos de respuesta, consumo energético, portabilidad y compatibilidad, permitió la selección de 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 (**Anexo de casos de pruebas**).

En el caso de Raspberry Pi, se optó por 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 de tecnologías hardware

8.2.1 Razones para la elección de 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 baja y la disponibilidad de componentes conectables lo hace muy atractivo, para afrontar proyectos con diversos niveles de complejidad.

Gracias a que existe buen soporte de placas Arduino para el uso en control mediante en sensores y actuadores basado en microcontrolador, es atractivo en el contexto de una transición desde electrónica discreta hacia la programable.

Dentro de la plataforma Arduino, se seleccionó la placa Arduino UNO, sobre la cual 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, distancia, 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 deseaban 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.

A razón de estos problemas, se optó por el traspaso de una plataforma que trabaja con un microcontrolador, a una basada en un computador, siendo elegida la Raspberry Pi.

8.2.2 Razones para la elección de Raspberry Pi

En el capítulo 4 (**Capítulo 4 – Raspberry Pi**) se mencionó y analizó el computador de placa reducida (SBC) Raspberry Pi, plataforma diseñada primordialmente con fines didácticos por lo que su costo es relativamente bajo. Al contar con todas las capacidades básicas de una computadora portátil de hoy en día con su respectivo microprocesador (de potencia suficiente para las necesidades del SAR), 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 la conexión y manipulación de distintos módulos (como actuadores y sensores), aunque como se analiza en el apartado siguiente, se delegó en la placas Arduino Mega y Arduino Nano las funcionalidades de control y sensado, exceptuando la conexión y procesamiento de imágenes, encomendadas 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 (energía)	USB x 4, HDMI, CSI, DSI, MicroSD, WLAN y BLE, microUSB (Energía)

Dada la comparativa entre las tecnologías, es que se decidió utilizar las placas Arduinos (versiones Nano y Mega) para el control de módulos de hardware como sensores y actuadores y la SBC Raspberry Pi dedicada a la captura de imágenes y ejecución de servidor web. La conexión entre Arduino y Raspberry se realiza a través de sus interfaces USB.

8.2.4 Beneficios del complemento Arduino/Raspberry

Si bien podría pensarse que sería suficiente Raspberry para la elaboración del SAR, deben considerarse los siguientes beneficios que proporciona Arduino:

- Menor costo del producto para sustitución ante fallos.
- Mayor compatibilidad, con los módulos arduino-compatibles (como los de la familia Adafruit)
- Buen tiempo de respuesta para E/S.
- 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. A diferencia de Raspberry, al carecer Arduino de un sistema operativo, no existen retrasos inesperados propios de la arquitectura con protección de memoria y paginación ni tampoco los de la política de programación de tareas (*scheduling*).

8.2.5 Cámara V2 de Raspberry Pi

Como se mencionó anteriormente, se delegó la tarea de captura de imágenes a Raspberry, a través de la cámara exclusiva de esta plataforma en su versión V2. Esta se conecta al puerto CSI de cualquier modelo de este SBC, lo cual permite obviar la conexión pin a pin (como ocurre con cámaras como la OV7660) y no es necesario controlar la comunicación y captura de imágenes. Como se comentó en el capítulo 4 (**4.5 Accesorios para Raspberry Pi**), es una cámara de alta definición de 8 megapíxeles, suficiente para el objetivo que se pretendió con el desarrollo del SAR. Esto solucionó 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 el caso de esta tesina smartphones). En los casos de pruebas (**Caso de prueba Integración WIFI y Cámara - Caso de prueba Cámara OV 7670**) se describen las dificultades mencionadas.

8.2.6 Módulos de Arduino

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

Utilizados en el SAR:

- El módulo GPS, es 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 determinada distancia y tratar de evitar el impacto con los mismos
- Motores CC (corriente continua) para la movilidad del SAR dentro del ambiente

Ensayados y no seleccionados:

- El módulo wifi ESP8266 y el módulo Bluetooth HC-05, no se utilizaron debido a que la Raspberry Pi3 Modelo 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:

- Nivel de abstracción alto, logrado mediante librerías basadas en JavaScript, para la comunicación con el hardware (J5).
- Utilizar un Sistema Operativo de base (en este caso Raspbian), en vez de una rutina corriendo en un microcontrolador.
- Contar con recursos necesarios para desplegar un servidor web.
- Disponer de la posibilidad de comunicar las plataformas Arduino al servidor mediante un protocolo bien conocido.
- Utilizar las herramientas de SO del SBC para realizar la comunicación y captura de imágenes por sobre la captura manual de frames.
- Desarrollar de una aplicación móvil para el control inalámbrico del SAR.
- Almacenar datos para la generación de estadísticas
- Permitir el acceso multi-cliente a los datos alojados en el SAR.

Se optó por la instalación de Raspbian en la Raspberry, porque es el sistema operativo oficialmente soportado por la fundación [29]. Como se mencionó en un apartado anterior (**8.2.5 Cámara V2 de Raspberry Pi**) 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 un accesorio que funciona como cámara (mencionada en el apartado **4.5 Accesorios para Raspberry Pi**) tal se tratase de una webcam.

En los repositorios de Raspbian se encontró una aplicación denominada Motion. La cual está orientada a videovigilancia a través de cámaras web. En el caso del SAR, permitió la captura de imágenes en forma de *streaming*.

Dentro de las dificultades afrontadas con Arduino que devinieron en la delegación de funciones a Raspberry encontramos, que las placas están orientadas a programas dónde existe un único bucle de ejecución principal. En el caso del control de una cámara, no es suficiente el tiempo de transmisión de imágenes dado el nivel de procesamiento para almacenar bytes en un buffer y ser retransmitidos, tanto en serie (cable) como en forma inalámbrica (requiriéndose shields de comunicación), no alcanzando los FPS (cuadros por segundo anexo **Caso de prueba Cámara OV 7670**) necesarios para una visualización mínimamente fluida (al menos 10 FPS). Por otro lado, para poder almacenar gran cantidad de datos es necesario contar con un módulo para memorias SD. Entre otras desventajas de las placas, poseen una cantidad limitada de interrupciones por hardware (2 en Arduino Uno y Nano, 6 en el caso de Arduino Mega), resultando en la detección de nuevo valores en sensores mediante pooling.

Ante estas limitaciones Raspberry gracias a su hardware y ser un computador que permite la instalación de un sistema operativo, facilitó resolver varias de las

dificultades antes mencionadas. Se destacan, por ejemplo, las capacidades inalámbricas que permitieron configurar la SBC en modo **AP** (AP, o punto de acceso en español). Esto quiere decir, crear una red inalámbrica **WIFI** (con una SSID y contraseña) sin depender de ningún dispositivo de red externo (como por ejemplo un router inalámbrico) y permitiendo la conexión de diversos **Host**, donde cada uno obtiene su respectiva dirección **IP** por medio de **DHCP**.

Al comienzo del desarrollo, y teniendo en cuenta que inicialmente se pensó en trabajar únicamente con la familia Arduino, se había considerado en 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** (el compendio de tecnologías relacionadas con esta App, se abordaron en el **Capítulo 5 - Aplicaciones Móviles**); pero al mejorarse las prestaciones hardware y tener un sistema operativo, se decidió cambiar la arquitectura del software del SAR.

Esta nueva arquitectura generó un cambio en la aplicación, o sea, se pasó del desarrollo de una app nativa, para Android, a una app Web, permitiendo crear una única aplicación que puede ser consumida por distintos dispositivos que accedan a la red **LAN** 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 comunicación sencilla con el servidor.
- Tener la capacidad de desplegar a demanda la app desde una red **LAN**.
- Diseñar una app, utilizando herramientas de **Front-End**, para el renderizado en el cliente, para una mejor experiencia de usuario, basada en requerimientos **HTTP** para la comunicación con el servidor.

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 trató 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 (Raspberry Pi) se encontraron inconvenientes dado que este **Framework** no se encontraba soportado oficialmente para esta arquitectura. Existe un *fork*, pero no se tuvo éxito en la integración de las tecnologías.

Finalmente se seleccionó el stack MEAN el cual resultó ser compatible con el desarrollo avanzado hasta el momento, hecho con Meteor (MEAN se encuentra detallado en el **Capítulo 6 – Stack MEAN**). La migración de la aplicación tanto **Front-End** como **Back-End**, desarrollada con Meteor, fue dispuesta de la siguiente forma: El procesamiento de **Template**, 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, fue comunicar el proceso servidor, ejecutándose en Raspberry (como un proceso en Raspian), con las placas Arduino Mega y Arduino Nano. Dentro de los paquetes disponibles en NPM, se encontraron dos librerías estables para la comunicación de Node y Arduino. Estas librerías son Cylon y Johnny-five. La librería Cylon utiliza el paradigma de programación declarativo, en cambio, Johnny-five es orientado a *callbacks*. Este último fue el seleccionado por mantener el mismo estilo de codificación que el stack MEAN, compatibilidad con los componentes de Arduino (gracias a estar basado en Firmata) y, por, sobre todo, poseer una versión estable de serialport compatible con la arquitectura ARM. El protocolo Firmata, base de Johnny-five se analizó en el **Capítulo 7 – Comunicación NodeJS con Arduino**.

Finalmente se agregó el código necesario para que las mediciones tomadas por Jhony-Five de los sensores sean almacenadas en colecciones de MongoDB. A partir de éstas se generan las estadísticas requeridas por los objetivos de esta tesina.

Resumen

En este capítulo se analizaron las diversas tecnologías tanto de hardware como de software utilizadas en el SAR, justificando la selección de cada una de ellas y los ensayos realizados para concluir en su utilización.

Se explicó el porqué del uso de Arduino y Raspberry Pi como plataformas de base para la manipulación del robot móvil. Las ventajas del uso de la cámara v2 de Raspberry, y las problemáticas que se presentaron al probar la cámara para Arduino OV7670.

Por otro lado, en cuanto al software seleccionado, se detallaron los requerimientos necesarios para el desarrollo del SAR. Dentro de los mismos, se destacó el aprovechamiento de las capacidades brindadas por el Sistema Operativo de Raspberry, seleccionándose Raspbian para tal fin. Finalmente se describen las diversas tecnologías que se probaron a lo largo del desarrollo, resultando MEAN el satck elegido para realizar la aplicación web.

Capítulo 9 – Arquitectura y Ensamblado del SAR

El prototipo del SAR está compuesto de un robot móvil, dotado de 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, sus funcionalidades y los procedimientos que se llevaron a cabo para construirlo. Se detalla su estructura y la disposición de sus componentes.

En la siguiente imagen (**Ilustración 50 - Esquema de conexión de componentes**) se puede apreciar la arquitectura de conexión de los componentes que integran al SAR.

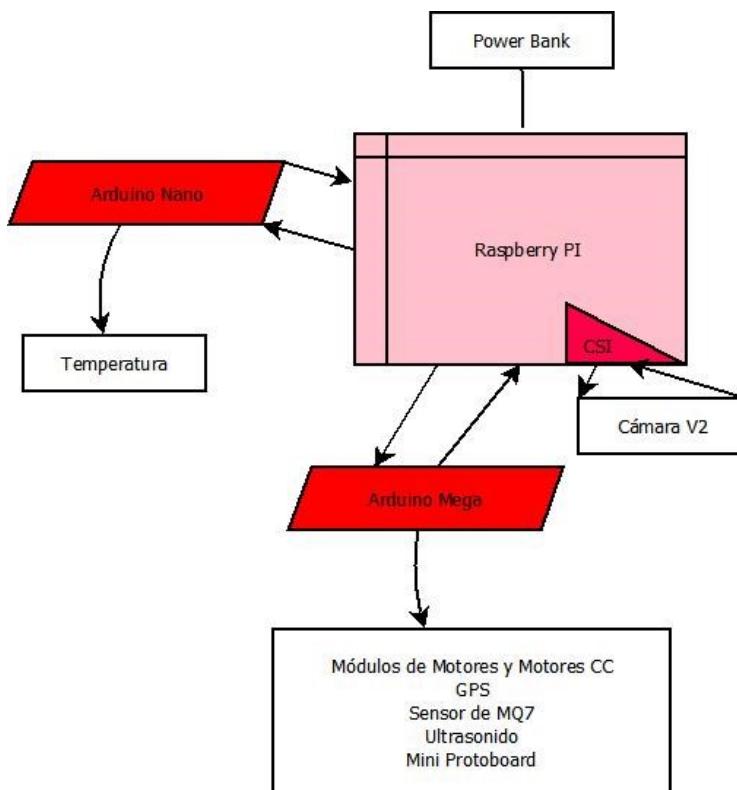


Ilustración 50 - Esquema de conexión de componentes

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 que se traducen en órdenes a las placas Arduino. Cuenta con una tarjeta microSD donde almacena el sistema SO Raspbian, el cual, tras el encendido ejecuta la aplicación desarrollada. En la imagen (**Ilustración 51 - Raspberry Pi 3**)



Ilustración 51 - Raspberry Pi 3

se puede apreciar esta SBC.



Ilustración 52 - Arduino Mega

Arduino Mega: Es el microcontrolador de mayor capacidad 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 órdenes 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, **Código StandarFirmata utilizado en el Arduino MEGA**) necesaria para establecer la comunicación con los comandos enviados desde Javascript por la aplicación web. En la imagen (**Ilustración 52 - Arduino Mega**) se puede distinguir esta placa.

Un Arduino Nano: Esta versión de Arduino (**Ilustración 53 - Arduino Nano**) 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 (**Código ConfigurableFirmata utilizado en el Arduino NANO**).

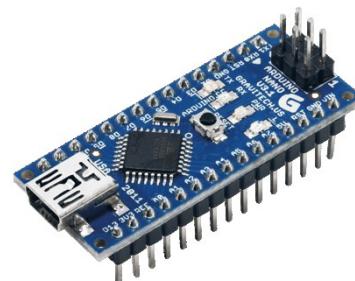


Ilustración 53 - Arduino Nano

Cuatro motores DC (corriente continua de 3v a 6v) con caja reductora: Estos motores (**Ilustración 54 - Motores CC**), en conjunto con cuatro ruedas de plástico cubiertas con una goma cada una, son los que permiten darle la movilidad al SAR.



Ilustración 54 - Motores CC



Ilustración 55 - Sensor de ultrasonido

Tres sensores ultrasónicos HC-SR04: Los sensores ultrasónicos (**Ilustración 55 - Sensor de ultrasonido**), 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.

Dos portas pilas AA x4 con sus respectivas pilas recargables: Utilizados para alimentar de corriente eléctrica a los 4 motores. En la imagen (**Ilustración 56 - Porta pilas**), el modelo utilizado.



Ilustración 56 - 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. En la imagen (**Ilustración 57 - Módulo Puente H**) se puede apreciar el modelo de ellos.

Ilustración 57 - Módulo Puente H

Una mini Protoboard: Esta **Protoboard** (**Ilustración 58 - Mini-protoboard**), fue agregada como extensión de pines, dónde se conectan pines GND y 5v de la placa Arduino Mega.



Ilustración 58 - Mini-protoboard



Ilustración 59 - Sensor de Temperatura

Un sensor de temperatura DS18B20 montado sobre una placa KY-001: Este módulo (**Ilustración 59 - Sensor de Temperatura**) es el encargado de sensar la temperatura, se encuentra conectado al Arduino Nano.

Un sensor de monóxido de carbono MQ-7: El sensor de monóxido (**Ilustración 60 - MQ7 CO**), conectado al Arduino Mega, detecta la ausencia o presencia de dicho gas.



Ilustración 60 - MQ7 CO

Un GPS GY-GPS6MV2: Con este módulo de GPS (**Ilustración 61 - GPS**) obtenemos toda la información necesaria con respecto a la Geolocalización del SAR (latitud, longitud, punto cardinal, velocidad, orientación, fecha y hora).



Ilustración 61 - GPS



Cámara de Raspberry Pi V2: Esta cámara (**Ilustración 62 - Cámara V2**), 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.

Ilustración 62 - Cámara V2

PowerBank Malibu de 20Ah con panel solar: Este cargador portátil (**Ilustración 63 - Panel Solar Power Bank**) funciona como batería del SAR, provee de corriente eléctrica a la Raspberry y por ende a los arduinos conectados a ella.



Ilustración 63 - Panel Solar Power Bank

9.2 Estructura

9.2.1 Diseño

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

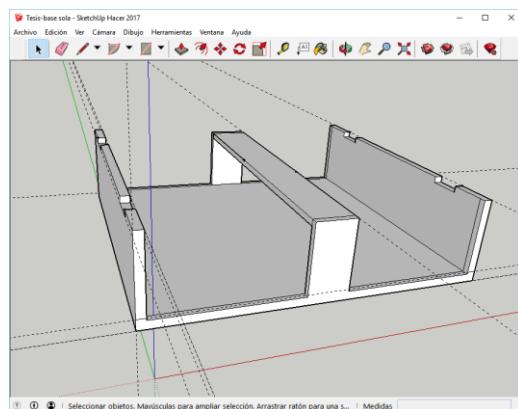


Ilustración 66 - Diseño estructura nivel 1 con SketchUp

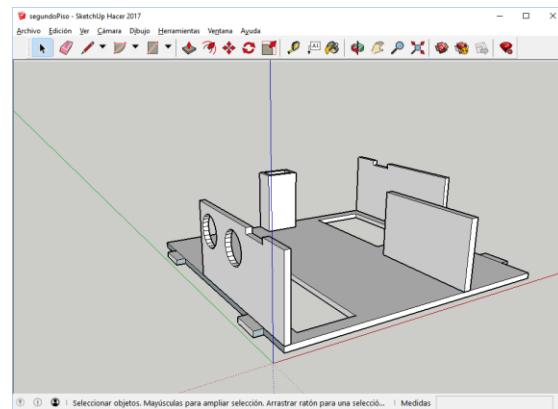


Ilustración 67 - Diseño estructura nivel 2 con SketchUp

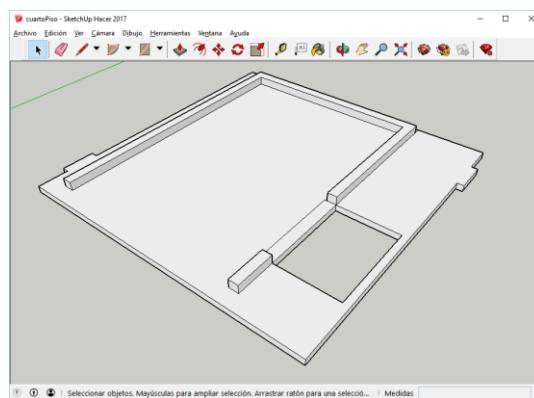


Ilustración 65 - Diseño estructura nivel 4 con SketchUp

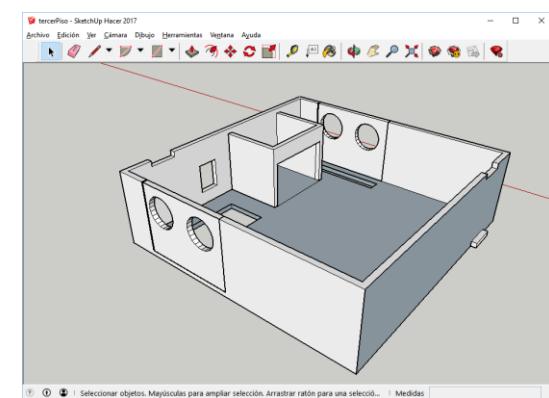


Ilustración 64 - Diseño estructura nivel 3 con SketchUp

En las imágenes anteriores se pueden apreciar los modelos diseñados (**Ilustración 64 - Diseño estructura nivel 3 con SketchUp**, **Ilustración 65 - Diseño estructura nivel 4 con SketchUp**, **Ilustración 66 - Diseño estructura nivel 1 con SketchUp** e **Ilustración 67 - Diseño estructura nivel 2 con SketchUp**)

Una vez completado el proceso de modelado se procedió a la impresión mediante una impresora 3D (**Ilustración 68 - Impresión 3D del nivel 1**). A continuación, se detallan los niveles.

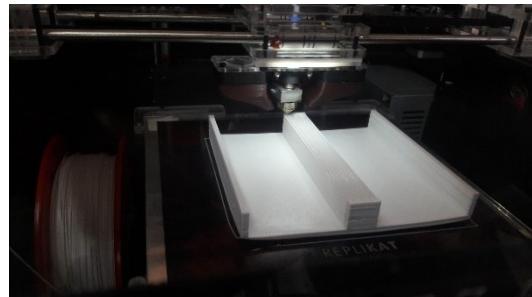


Ilustración 68 - Impresión 3D del nivel 1

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.

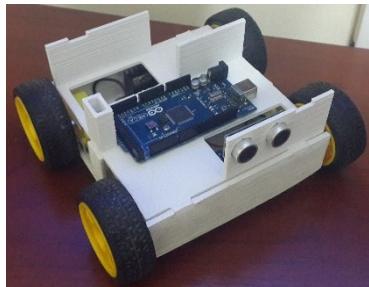


Ilustración 69 - Nivel 2 descubierto

Nivel 2: En este nivel (**Ilustración 69 - Nivel 2 descubierto**) se sujetó mediante tornillos el Arduino Mega y la mini **Protoboard**, dónde se realizó la interconexión de los componentes. Además, se encuentra en este nivel el porta pilas utilizado para la alimentación de los motores. En el frente se colocó uno de los sensores ultrasónicos HC-SR04 que verifica la presencia de objetos en la parte delantera del SAR.

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, un sensor de monóxido de carbono MQ-7 y la cámara de Raspberry en el frente.

Nivel 4: El nivel superior (**Ilustración 70 - RM Vista Lateral**) es el que se equipa con la batería portátil con carga 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.



Ilustración 70 - RM Vista Lateral

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).

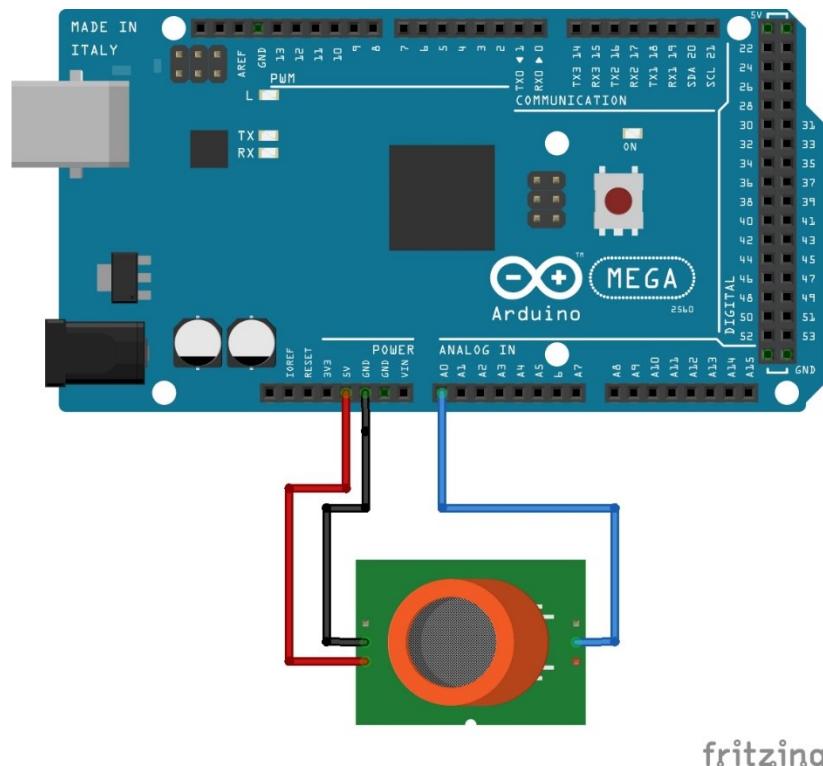


Ilustración 71 - Esquema de conexión de sensor de monóxido MQ-7 a Arduino Mega

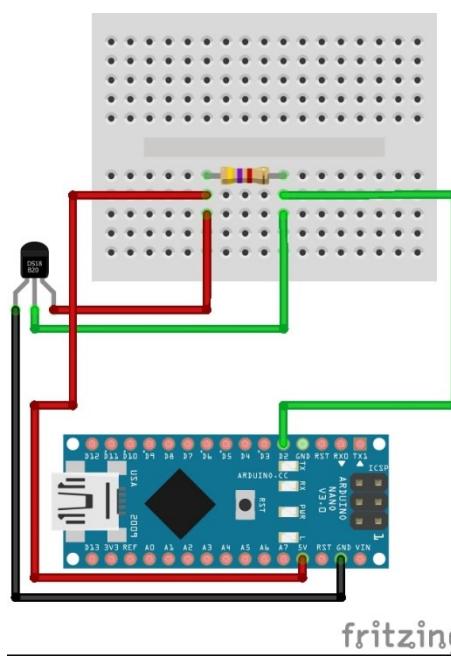


Ilustración 72 - Esquema de conexión de sensor de temperatura DS18D20 a Arduino Nano

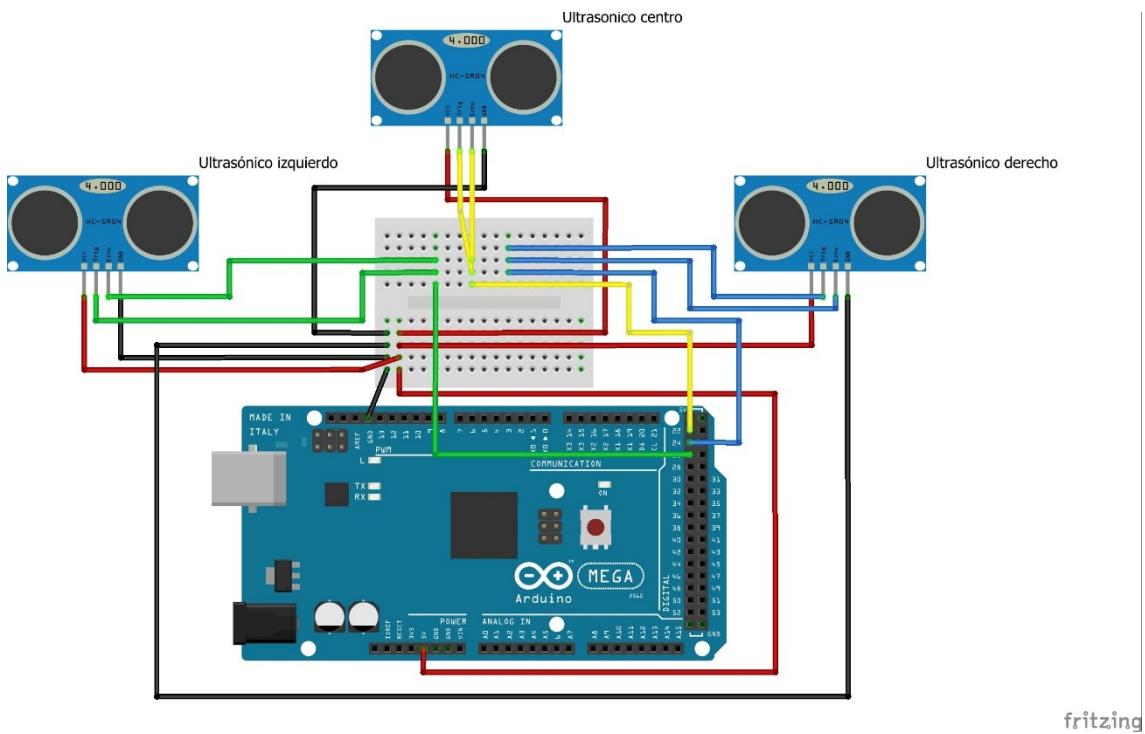


Ilustración 73 - Esquema de conexión de sensores ultrasónicos HC-SR04 con Arduino Mega

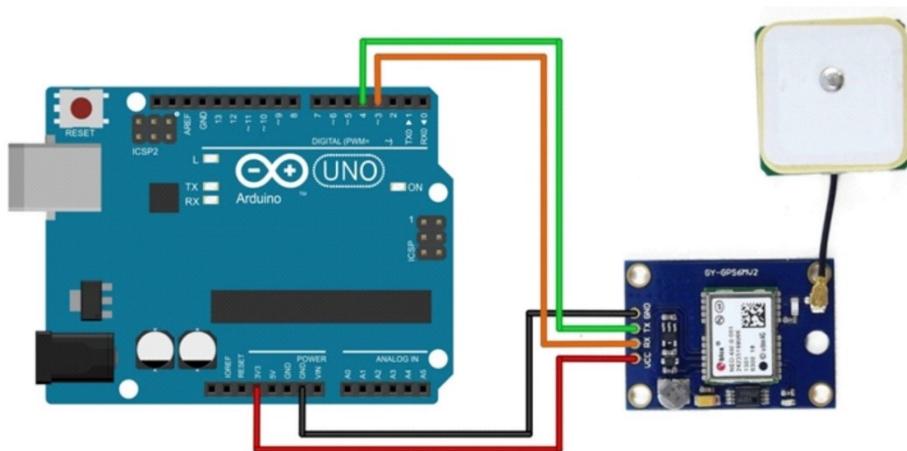


Ilustración 74 - Esquema de conexión de módulo GPS con Arduino UNO

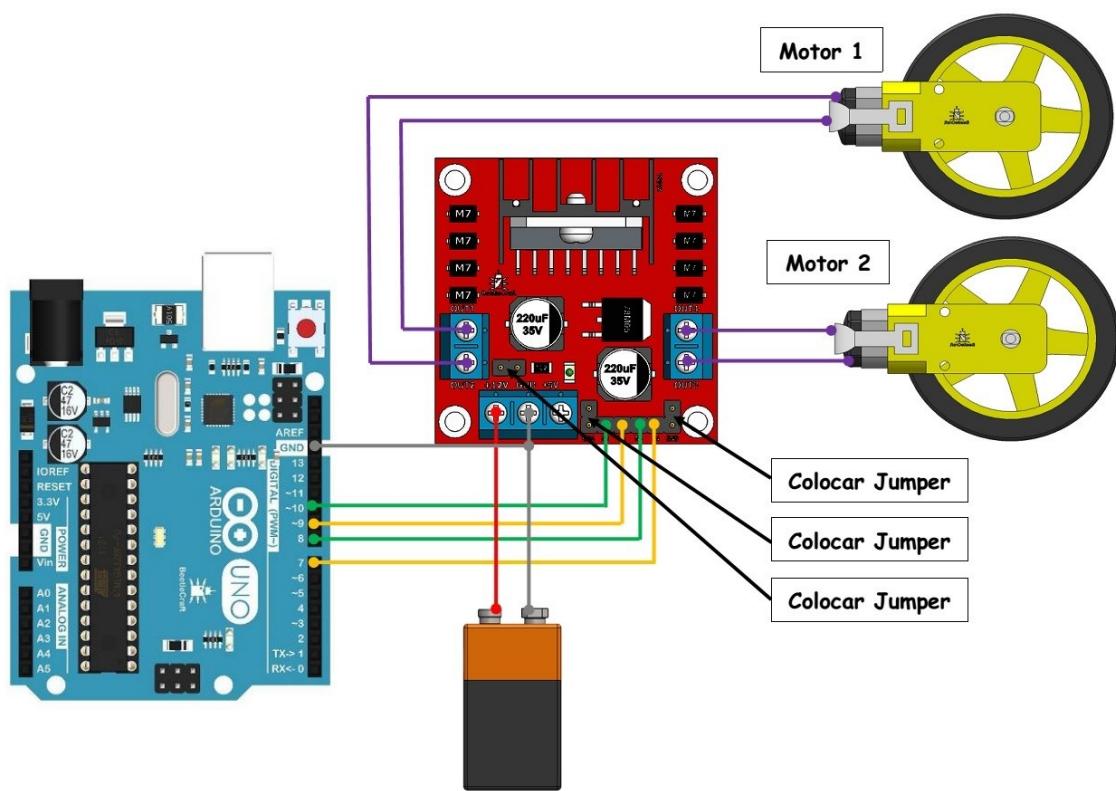


Ilustración 75 - Esquema de conexión de puente H y motores con Arduino UNO

Resumen

En este capítulo se pudo apreciar la arquitectura del SAR, para ello se describieron los distintos componentes con los que cuenta. La Raspberry Pi cumple un rol fundamental dentro del robot, dado que es el centro de control del mismo. Contiene la App que permite la manipulación del SAR.

Por otro lado, se vio que, tanto el Arduino Mega como el Nano sirven de intermediarios entre la Raspberry y el resto de los componentes (sensores, módulos y actuadores). Para ello se les debió cargar un protocolo denominado Firmata. Además, se detallaron los esquemas de conexión básicos entre un Arduino y los demás elementos del SAR.

Finalmente, se describió la estructura física del robot y los cuatro niveles con los que cuenta, además de que componentes contiene cada uno.

Capítulo 10 – Desarrollo del SAR

El desarrollo del SAR se descompone en varios niveles de capas. A su vez, existen dos esquemas muy diferenciados el lógico y el físico (**Ilustración 76 - Esquema general del SAR**).

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 mediante la arquitectura cliente/servidor bajo el conjunto de herramientas MEAN. Además, su funcionamiento está controlado por un administrador de servicios para Node denominado PM2. El SAR también hace uso de Motion, un controlador de cámaras de video, sobre el SO Raspbian. Este esquema a su vez se encuentra organizado en dos unidades funcionales llamadas **Back-End** y **Front-End**.

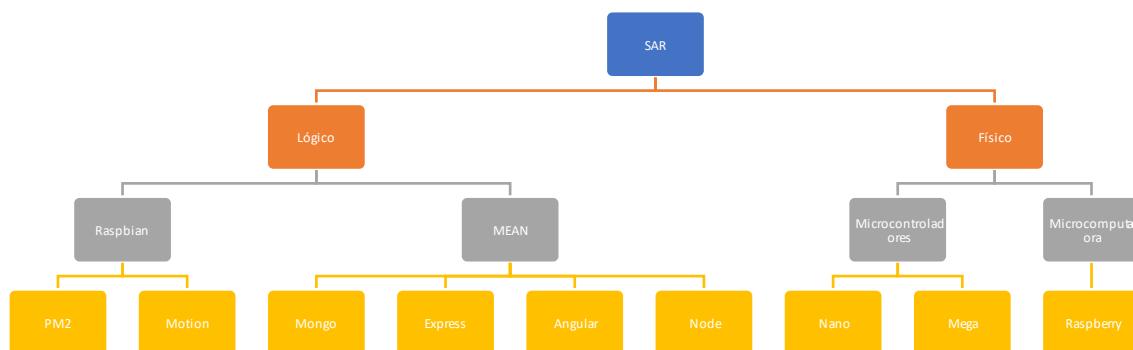


Ilustración 76 - Esquema general del SAR

10.1 Estructura de la aplicación (front-end)

El **Front-End** se encuentra desarrollado en Angular 4+, contando con los siguientes esquemas (**Ilustración 77 - Módulos Angular**):

Al momento de la conexión desde un navegador hacia el servidor web, se obtiene la aplicación embebida, y esta inicia la comunicación nuevamente hacia el servidor intercambiando mensajes codificados en JSON.

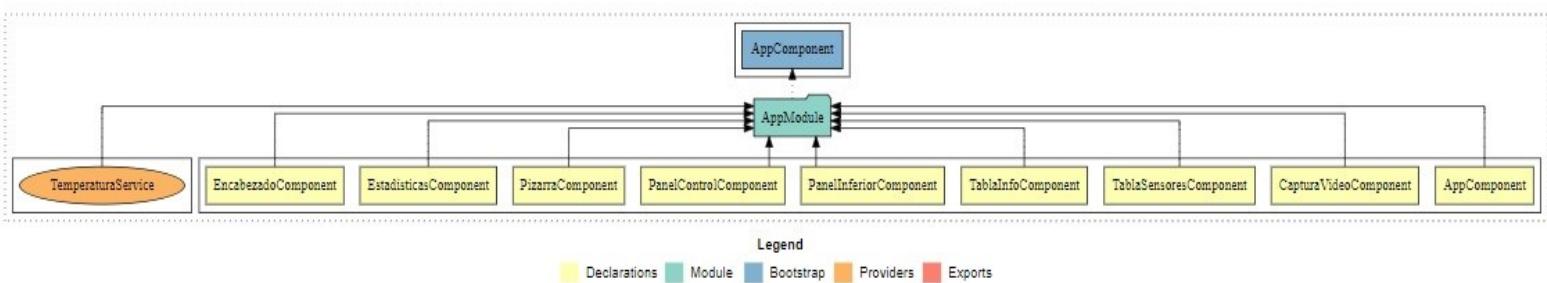
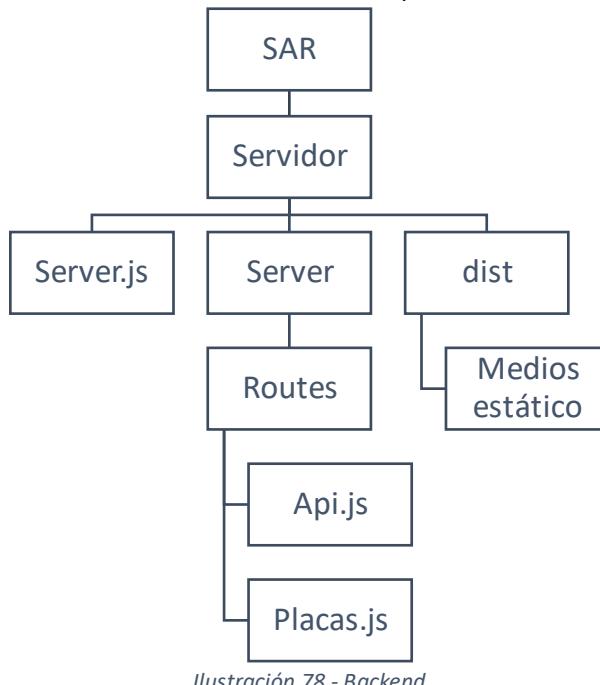


Ilustración 77 - Módulos Angular

10.2 Desarrollo del servidor (*back-end*)

El **Back-End**, fue construido sobre el **Framework** web Express, y aprovechando de NodeJS la capacidad para servir medios estáticos del **Front-End** (JS, CSS, **HTML** no generados dinámicamente). Los directorios del proyecto se pueden observar en la ilustración (Ilustración 78 - Backend)



En el directorio Routes, se encuentran dos archivos: Api.js y Placas.js. El primero se encarga de gestionar las llamadas por métodos **HTTP**, hacia **endpoints REST**. Por otro lado, Placas.js se encarga de gestionar la conexión a los periféricos de la Raspberry, que son la Arduino NANO y la Arduino MEGA.

Además, en el directorio dist, se almacena el resultado de la compilación de la aplicación Angular **Front-End**, que consisten en archivos JavaScript, **HTML** y CSS3. Por otro lado, existe un archivo server.js encargado del arranque del servidor escuchando el puerto 3000, a través de las **APIs** estándar de Node.

La conectividad a MongoDB es realizada a través del driver MongoClient utilizado en el archivo Api.js.

Las **endpoints REST**, atendidos por Express, son:

- Temperaturas → Api.js, devolviendo temperaturas almacenadas en la BD.
- Monóxidos → Api.js, devolviendo monóxidos almacenadas en la BD.
- Monóxido → placas.js, se obtiene el valor actual de monóxido en el ambiente.
- Apagar → Api.js, apagando la Raspberry desde el sistema operativo.
- Reiniciar → Api.js, reiniciando la Raspberry desde el sistema operativo.

- Arriba, Abajo, Izquierda, Derecha, Stop → placas.js, ejecutando un orden a los motores para desplazarse o detenerse.
- Ultrasonido → placas.js, recupera valores de distancia medida por ultrasonido.
- GPS → placas.js, recupera valores como altura, velocidad, curso, fecha y coordenadas.

En el anexo de códigos, se encuentran los mencionados anteriormente:

- **Código Servidor Node (server.js)**
- **Código API Express (api.js)**
- **Código Manejo de Arduino Mega y Arduino Nano (placas.js)**

10.3 Esquema de la arquitectura lógica

El funcionamiento de la aplicación se basa en la comunicación entre el **Front-End** (cliente) con el **Back-End** (servidor). En el siguiente gráfico (**Ilustración 79 - Arquitectura lógica del SAR**) se puede apreciar la arquitectura lógica cliente/servidor que posee el SAR, detallando todos los componentes, tanto de software como de hardware, que intervienen para que el sistema funcione.

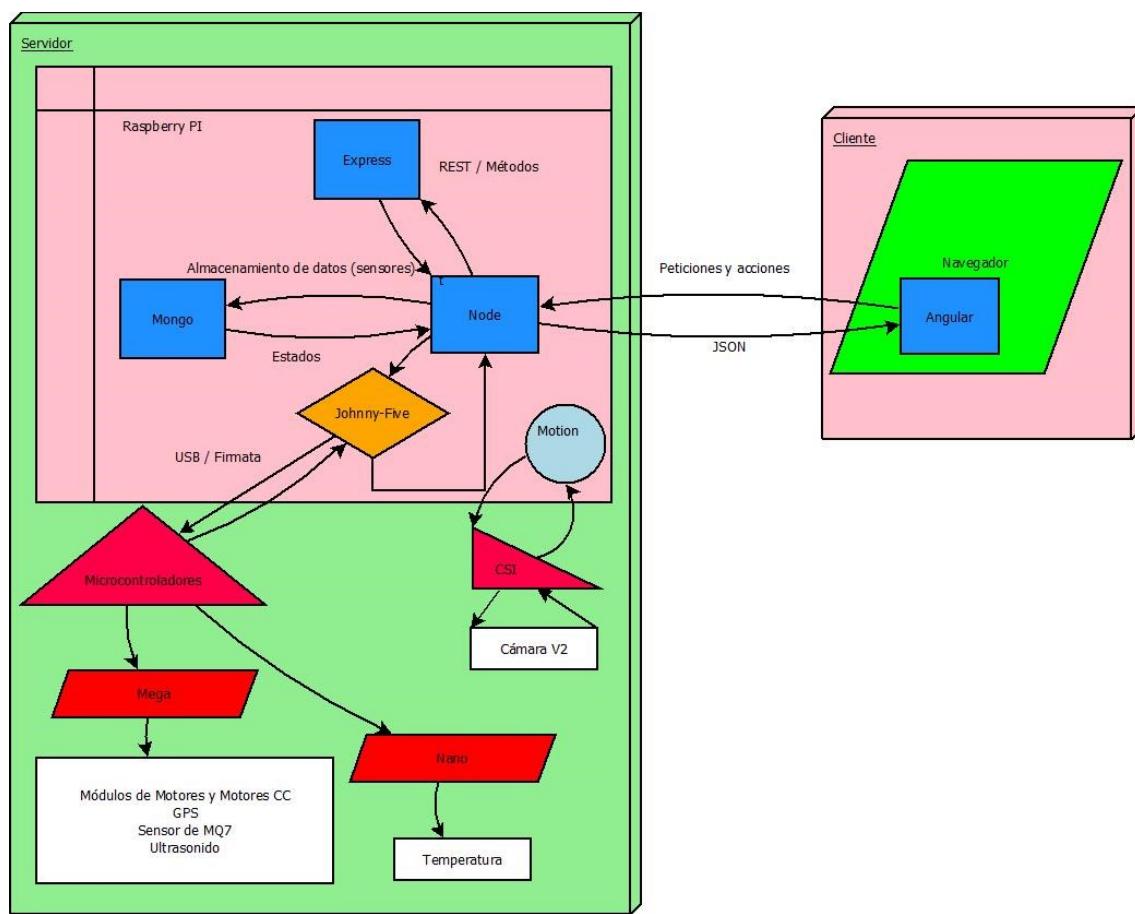
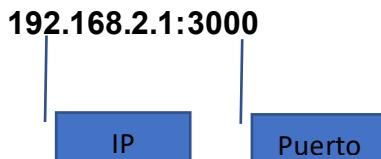


Ilustración 79 - Arquitectura lógica del SAR

10.4 Funcionamiento de la App

Para poder acceder a la aplicación del SAR y controlar el robot móvil, se debe conectar al punto de acceso (**AP**) de red **WIFI** que genera la Raspberry denominada “SAR”, con su respectiva contraseña.

Una vez conectado a la red, se debe acceder mediante un **Navegador web** a la URL:



Automáticamente se desplegará la interfaz que se puede apreciar en la siguiente ilustración (**Ilustración 80 - Aplicación web**).

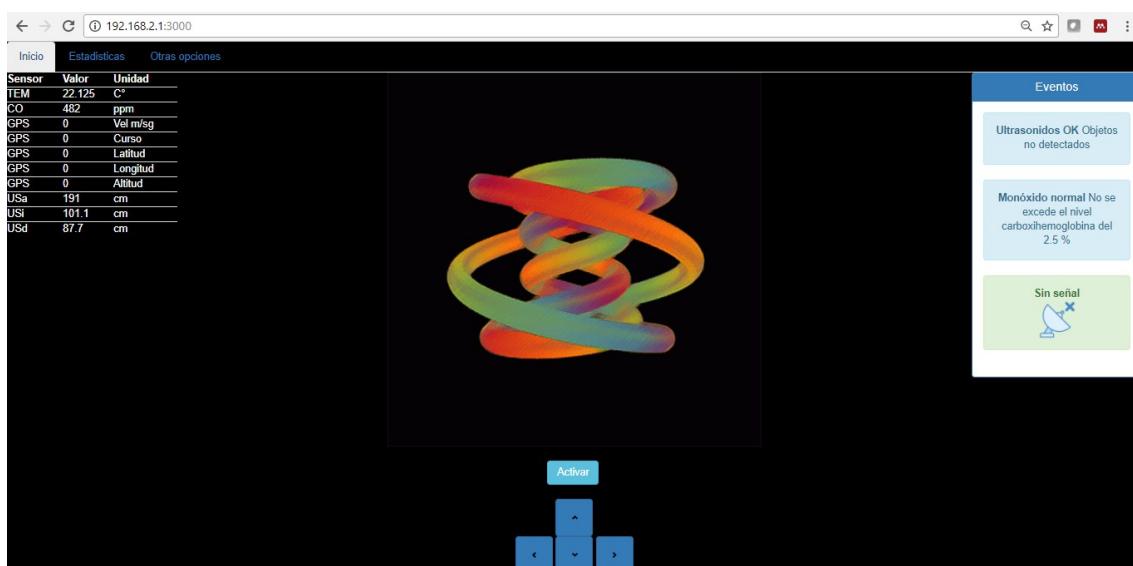


Ilustración 80 - Aplicación web

Como se puede apreciar, la interfaz cuenta con un menú, con tres solapas. Éstas son: Inicio, Estadísticas y Otras opciones.

La solapa de inicio, se encuentra dividida en tres partes. En el panel izquierdo, se observa una tabla de valores obtenidos por los sensores. Estos valores se actualizan en tiempo real. En el panel central se encuentra, un botón para activar/desactivar la visualización de video en tiempo real. Por debajo, de dicho botón, existe un conjunto de botones, que permiten controlar el movimiento y la dirección del robot móvil. En el panel derecho, se puede ver una sección de “novedades” la cual muestra los distintos estados de los sensores con mensajes de advertencia y/o información al usuario.

En la solapa estadísticas, se visualizan dos diagramas estadísticos. El primero es de temperatura (**Ilustración 81 – Estadísticas de temperaturas**), que a través de unos selectores se puede indicar la fecha de inicio y de fin. De esta forma se confecciona una gráfica, al cliquear en obtener, siempre y cuando existan datos almacenados en el período seleccionado. La gráfica muestra en

las columnas, los días y en las filas, las horas del día desde las 00:00 hs hasta las 23:00 hs. En cada una de las intersecciones fecha/hora se muestra el promedio de temperaturas en un rango de 1 hr. En la primera columna, se genera una media de los valores que se encuentran en la misma fila adyacente de las fechas seleccionadas.



Ilustración 81 – Estadísticas de temperaturas

En el segundo (**Ilustración 82 - Estadísticas de monóxido**), se toman valores en tiempo real del valor de monóxido de carbono presente en el ambiente. Los mismos se representan según el horario y el valor obtenido en PPMV (Partes Por Millón en Volumen). Por otro lado, se permite colocar el máximo de valores a visualizar.

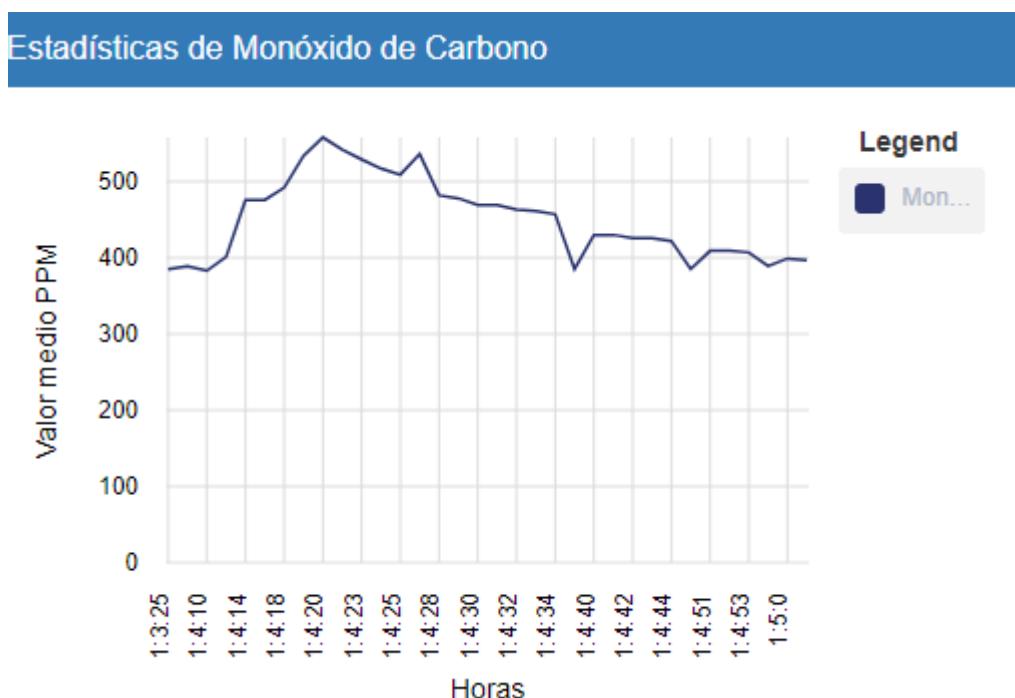


Ilustración 82 - Estadísticas de monóxido

En la solapa otras opciones (**Ilustración 83 - Aplicación web - Otras opciones**), visualizamos dos botones, que permiten apagar o reiniciar el sistema operativo, por ende, el robot móvil.

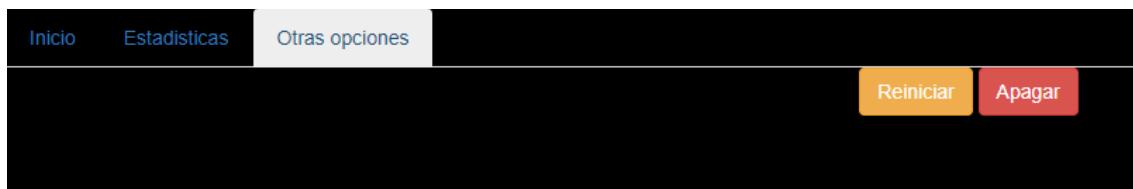


Ilustración 83 - Aplicación web - Otras opciones

10.5 Puesta en producción del SAR

Para desplegar el SAR se necesitó de software y configuraciones adicionales que a continuación se describen.

10.5.1 Configuración de Raspberry como AP

El SBC Raspberry Pi 3 se encuentra dotado de un módulo **WIFI** que permite la configuración como modo Access Point (**AP**). Para que diversos dispositivos se puedan conectar al SAR, se realizó la configuración de dicho módulo de la siguiente manera. [30]

Primeramente, se instalaron, los programas necesarios para generar el punto de acceso (**AP**), **hostapd** y **isc-dhcp-server**. Introduciendo, en una terminal de Raspbian, los siguientes comandos:

```
sudo apt-get install hostapd isc-dhcp-server
```

A continuación, se configuro el archivo que permite la configuración de **DHCP**, accediendo al mismo de la siguiente manera:

```
sudo gedit /etc/dhcp/dhcpd.conf
```

Dentro del mismo se comentaron dos líneas que definen el nombre de dominio, dado que no se utilizaron:

```
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;
```

Se descomentó la línea que a continuación se resalta, que define a la Raspberry como servidor **DHCP**:

```
# If this DHCP server is the official DHCP server for the local
```

```
# network, the authoritative directive should be uncommented.

authoritative;
```

Finalmente, la última configuración realizada con este archivo, fue la de definición de la subred para la **LAN**, donde se configuraron los siguientes parámetros:

```
subnet 192.168.2.0 netmask 255.255.255.0 {

    range 192.168.2.10 192.168.2.30;

    option broadcast-address 192.168.2.255;

    option routers 192.168.2.1;

    default-lease-time 600;

    max-lease-time 7200;

    option domain-name "local";

    option domain-name-servers 8.8.8.8, 8.8.4.4;

}
```

El siguiente archivo a modificar fue el **isc-dhcp-server**, para ello se accedió al mismo de la siguiente manera:

```
sudo gedit /etc/default/isc-dhcp-server
```

En él se configuró la interfaz de las Raspberry que funcionara como servidor **DHCP**, en nuestro caso sobre la interfase wlan0, agregando la misma en la siguiente línea:

```
INTERFACES="wlan0"
```

Como es común para **Host** dónde se ejecutan servidores, se estableció el acceso a la aplicación web del SAR mediante una dirección **IP** estática, definida sobre la interfaz wlan0.

Primero se debió desactivar la interfaz wlan0, de la siguiente manera:

```
sudo ifdown wlan0
```

Luego se modificaron los parámetros de configuración de wlan0 accediendo al siguiente archivo:

```
sudo gedit /etc/network/interfaces
```

En él se agregó la **IP** estática definida para el SAR y se comentaron las tres últimas líneas, que definían la configuración manual de la interfaz:

```
allow-hotplug wlan0

iface wlan0 inet static
    address 192.168.2.1
    netmask 255.255.255.0

#iface wlan0 inet manual
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Luego de guardado el archivo anterior se habilitó asignando la **IP** mencionada con el siguiente comando en la terminal:

```
sudo ifconfig wlan0 192.168.2.1
```

Para finalizar la configuración **AP**, se procedió a la modificación del siguiente archivo:

```
sudo gedit /etc/hostapd/hostapd.conf
```

El cual permite la definición de los ajustes básicos de la red **WIFI** a transmitir, como el SSID, contraseña de acceso, interfaz y demás opciones que a continuación se detallan:

```
interface=wlan0
ssid=sar
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=sartesis2017
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Para que se inicie el modo **AP** al arrancar el sistema operativo se iniciaron los siguientes servicios:

```
sudo service hostapd start
sudo service isc-dhcp-server start
```

Luego, se habilitaron al arranque del SO:

```
sudo update-rc.d hostapd enable
sudo update-rc.d isc-dhcp-server enable
```

10.5.2 Configuración del servicio Motion

Como ya se mencionó anteriormente (**8.3 Selección tecnologías software**), el software seleccionado para la captura de imágenes por parte de la cámara de Raspberry es Motion. Para poder utilizar dicho programa se tuvo que realizar lo siguiente:

Primero, se habilitó el uso de la cámara de Raspberry. Para ello, se accedió a las herramientas de configuración de Raspberry con el siguiente comando en una terminal de Raspbian:

```
sudo raspi-config
```

Dentro de la lista de opciones (**Ilustración 84 - Software de configuración de Raspberry**) se activó la cámara seleccionando la 5:

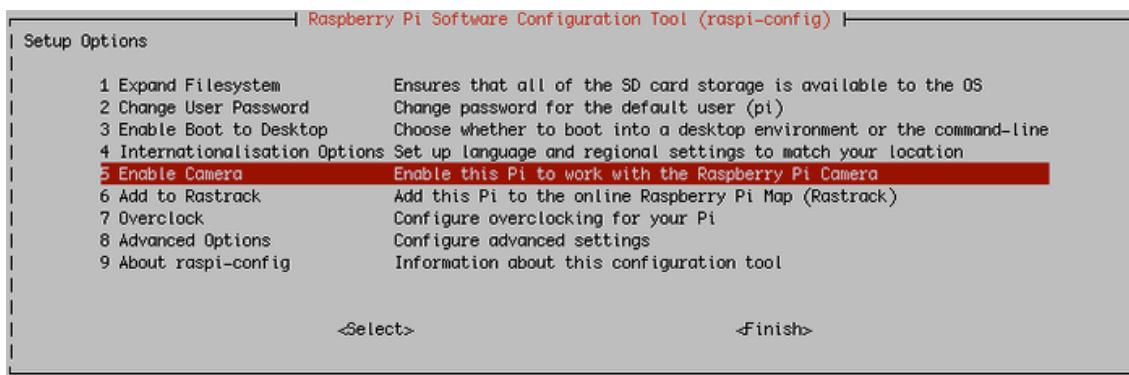


Ilustración 84 - Software de configuración de Raspberry

Seguidamente se instaló Motion de la siguiente manera:

```
sudo apt-get install -y motion
```

Este software automáticamente detecta la cámara de la Raspberry y captura las imágenes obtenidas por la misma. A su vez genera un servicio para poder visualizar en tiempo real la captura de video en el **IP** predeterminado de la SBC y un puerto por defecto.

Para configurar dichos parámetros, Motion cuenta con un archivo que permite realizar los distintos ajustes del software. Para acceder al mismo se ejecutó el siguiente comando:

```
sudo gedit /etc/motion/motion.conf
```

En el cual se modificaron los siguientes parámetros:

```
videodevice /dev/video0
width 640
height 480
threshold 1500
minimum_motion_frames 1
quality 75
webcam_port 9081
DAEMON = ON
Webcam_localhost = off
```

Luego, para aplicar los cambios se reinició Motion de la siguiente manera:

```
sudo service motion restart
```

Para poder visualizar la captura de la cámara, se debe acceder mediante un **Navegador web**, a la dirección <http://ip:puerto> configurada previamente para la Raspberry:

<http://192.168.2.1:9081>

Finalmente se modificó el siguiente archivo:

```
sudo gedit /etc/default/motion
```

Para que el servicio de Motion arranque al iniciar Raspbian. En el mismo se modificó la siguiente línea:

```
start_motion_daemon = yes
```

10.5.3 Instalación del gestor de procesos PM2

Node es administrado por PM2 (administrador de procesos para JavaScript) el cual inicia el servidor, automáticamente al arrancar Raspbian, controlando y monitoreándolo. En la siguiente captura (**Ilustración 85 - Monitor de PM2**) podemos apreciar como el gestor de procesos PM2, permite monitorear, controlar y ejecutar la aplicación definida (server.js) que realiza el despliegue del servidor del SAR. En caso de fallos, PM2 re-arranca la aplicación emitiendo mensajes. Además, permite monitorear cantidad de reinicios, generar un log, mostrar el tiempo de carga de la aplicación, como reiniciarla, pararla y listar todos los procesos que gestiona.

Por otro lado, permite generar el archivo de startup para iniciar con el sistema operativo de Raspbian.

The screenshot shows the PM2 monitor interface. On the left, there's a 'Process list' section with a single entry: '[0] server Mem: 43 MB CPU: 0 % online'. To the right of this is a 'Global Logs' pane displaying a log of errors and warnings from the application. Below these are two large sections: 'Custom metrics' and 'Metadata'. The 'Custom metrics' section shows values for Loop delay (1.66ms), Active requests (0), and Active handles (5). The 'Metadata' section provides detailed information about the application: App Name (server), Restarts (20), Uptime (3d), Script path (D:\sar\Software\Servidor\server.js), Script args (N/A), Interpreter (node), Interpreter args (N/A), Exec mode (fork), and Node.js version (8.9.3). At the bottom, there are navigation instructions ('left/right: switch boards | up/down/mouse: scroll | Ctrl-C: exit') and a link to 'To go further check out https://keymetrics.io/'.

Ilustración 85 - Monitor de PM2

Otra característica de PM2, es la integración con Keymetrics (**Ilustración 86 - Keymetrics**), donde si el servidor tiene acceso a **Internet**, es posible hacer el seguimiento desde **Internet** previa autenticación mediante token de seguridad.



Ilustración 86 - Keymetrics

Resumen

Este capítulo explicó la utilidad tanto del **Back-End** como del **Front-End**, necesarios para el funcionamiento de la aplicación web que despliega el SAR. A su vez se describieron los pasos realizados para configurar las Raspberry Pi como un punto de acceso (**AP**) inalámbrico, lo cual permite que diversos dispositivos se puedan conectar al robot móvil, mediante un **IP** y puerto previamente configurados.

Por otro lado, se describió la configuración del software utilizado para la captura de imagen y video, de la cámara del SAR, denominado Motion.

Finalmente se describió la utilidad de un programa de gestión de procesos (PM2), que sirve como administrador de Node y permite monitorear el funcionamiento de la aplicación web.

Capítulo 11 – Conclusión y trabajos futuros

En este capítulo se detallarán las conclusiones en base a los objetivos generales y específicos planteados para el desarrollo de la tesina. A su vez, se describirán trabajos futuros.

11.1 Conclusión final

Revisando el objetivo general y los diversos objetivos específicos, tratados al principio de esta tesina (**1.1 Objetivo general**), se llegó a las siguientes conclusiones:

Se completó el desarrollo de un prototipo de un Sistema Autónomo Robótico (SAR), gestionado por un software definido como agente inteligente (que responde al modelo basado en objetivos) para la exploración y análisis del medio ambiente.

Mediante la experiencia adquirida al trabajar en establecimientos educativos en áreas relacionadas con la robótica, y todas las investigaciones y ensayos realizados para producir este prototipo, podemos concluir que el mismo sirve de base o modelo para el desarrollo de proyectos afines.

11.1.1 Ensamblar un robot móvil integrando las plataformas Arduino y Raspberry Pi con diversos módulos y software.

Se investigaron variadas tecnologías, que permiten la integración de estas plataformas fundamentales para el ensamblado del robot móvil. En base a esta investigación, podemos concluir que la interacción entre la Raspberry Pi y Arduino es viable, dado los diferentes módulos y componentes compatibles a nivel hardware y los protocolos de comunicación a nivel software.

11.1.2 Desarrollar una aplicación web multiplataforma que mediante comunicación inalámbrica permita el control del Robot móvil.

Una de las principales características de las aplicaciones web, para el desarrollo de aplicaciones móviles, es la de permitir su utilización en distintos sistemas operativos y plataformas hardware. Esto permite que la aplicación cliente, ejecutada por un **Navegador web**, sea multiplataforma. Como consecuencia, existen diversos clientes, que se conectan de forma inalámbrica al SAR. Estos clientes pueden ejecutar acciones sobre el robot móvil.

El desarrollo del software del SAR basado en la arquitectura cliente/servidor y enfocado en el modelo **Front-End/Back-End**, generó múltiples beneficios en cuanto a las características del software, como la interoperabilidad, reutilización, portabilidad, flexibilidad, extensibilidad y escalabilidad.

Podemos concluir, que además de los beneficios mencionados, el tiempo de desarrollo de la aplicación fue menor a lo proyectado.

11.1.3 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.

A partir de la investigación de distintos protocolos de comunicación entre las plataformas utilizadas en el desarrollo de esta tesina, se identificó uno en particular, Firmata. Este protocolo resultó ser el más adecuado para realizar dicha comunicación.

La ventaja primordial del uso de Firmata, es que se encuentra ampliamente utilizado, razón por la cual, es compatible con múltiples librerías y lenguajes de programación. A su vez, queda claro que gracias a su existencia y lo expresado anteriormente no fue necesario el desarrollo de un nuevo protocolo que cumpla la misma función.

11.1.4 Ensamblar físicamente e integrar a nivel de software los distintos componentes (sensores y actuadores) al SAR.

Gracias a la utilización de la plataforma Arduino y su compatibilidad (mencionada en **Capítulo 3 – Arduino**) con sensores y actuadores, es que la integración física de los elementos que componen el SAR, no fue una actividad tediosa. Esto es así, porque, tanto la plataforma como los componentes, nacen con la idea de poder generar soluciones a implementaciones electrónicas o proyectos a fines, sin la necesidad de conocimientos técnicos previos.

11.1.5 Extender la aplicación para interactuar con la información que brinda el SAR de los sensores.

Para el almacenamiento de los valores obtenidos por los sensores, que componen al SAR, se utilizó un sistema de bases de datos (por medio de **Mongo 6.2.1 MongoDB**). Este sistema facilitó el procesamiento de datos.

Por otro lado, con la implementación de una **API REST**, se simplificó la generación y representación de estadísticas. A su vez, esta **API** permitió el flujo de datos en tiempo real, de los eventos y los valores sensados.

11.2 Trabajos futuros

Se proponen como líneas futuras de mejoras las detalladas a continuación:

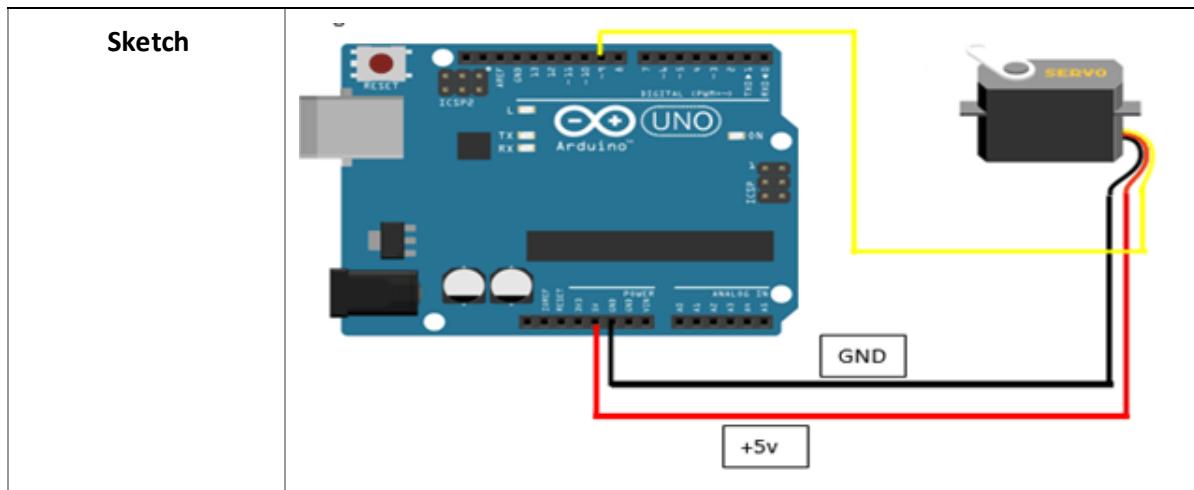
- Incorporar nuevos sensores y actuadores, como por ejemplo servo motores, sensor de humedad, sensor impacto, acelerómetro, sensor de llamas, etc., permitiendo así mejorar el reconocimiento del entorno en el que se encuentre el SAR.
- Extender la aplicación para generar mayor cantidad de estadísticas, que permitan comprender con mejor precisión el ambiente que lo rodee y así tomar decisiones que controlen su flujo de acciones.
- Adaptación de la estructura física a otros ambientes, es decir, agregar a la estructura variados componentes que permitan la movilidad del robot en distintos terrenos.

Anexo de casos de pruebas

En este anexo se detallan los distintos casos de pruebas realizados en los componentes compatibles con Arduino que se agregaron al SAR, como también de aquellos que por diversos motivos no fueron tenidos en cuenta para el armado final del mismo.

Servomotor SG90

Caso de prueba	Probar el funcionamiento del servomotor SG90
Identificador caso de prueba/s	SG90-01-funcionamiento
Función probar	Funcionamiento del servomotor SG90
Objetivo	Determinar el funcionamiento del servomotor
Descripción	Se desea conectar el servomotor SG90 a un Arduino UNO para determinar su correcto funcionamiento y ángulos de rotación con la precisión de 1° cada 20 ms (de fábrica)
Criterios de éxito	Funcionamiento correcto del servomotor en sus posibles ángulos de giro (90° a - 90°) con la precisión deseada
Criterios de falla	No alcanzar ángulos de giros correctos, fallas en conexiones
Precondiciones	Probar sin obstruir el servomotor con objetos
Necesidades para el caso de prueba	Módulo Arduino UNO SG90
Autor	Schlapp-Mansilla
Fecha de creación	25-04-2017
Resultados	[1] Se obtienen los ángulos de giros con la precisión correspondiente
Código fuente/s	[1] sg90-01-funcionamiento.ino
Imágenes	



Código sg90-01-funcionamiento

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
                // twelve servo objects can be created on most boards

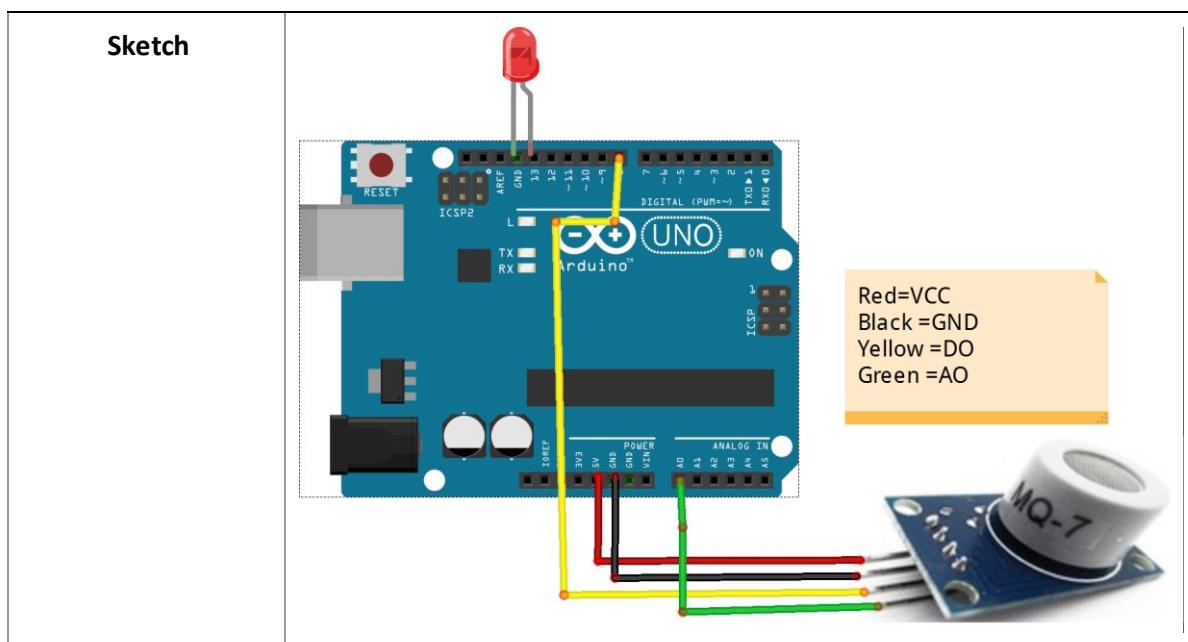
int pos = 0; // variable to store the servo position

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{
    for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180
degrees
    {
        myservo.write(pos); // tell servo to go to position in
variable 'pos'
        delay(15); // waits 15ms for the servo to reach
the position
    }
}
```

Pruebas en el sensor de Monóxido de Carbono

Caso de prueba	Probar la funcionalidad del sensor de monóxido de carbono MQ7
Identificador caso de prueba/s	MQ7-01-funcionamiento
Función probar	Fucionamiento del sensor MQ7
Objetivo	Determinar el funcionamiento correcto del sensor
Descripción	Se desea conectar el sensor de monóxido de carbono MQ7 con un Arduino UNO para verificar su correcta detección del gas CO
Criterios de éxito	Obtener la correcta existencia, o no, de gas CO en un ambiente determinado
Criterios de falla	No obtener la correcta existencia, o no, de gas CO
Precondiciones	Testear en entornos donde se esté seguro que los niveles de CO sean bajos o inexistentes Testear en entornos donde se esté seguro que existan al menos pocos niveles de CO
Necesidades para el caso de prueba	Módulo Arduino UNO MQ7 Cables Hembra-Macho (x3)
Autor	Schlapp-Mansilla
Fecha de creación	25-04-2017
Resultados	[1] Se obtuvieron niveles de CO esperados según los ambientes testeados.
Código fuente/s	[1]MQ7-01-funcionamientoi.ino
Imágenes	



Código MQ7-01-funcionamiento

```

void setup() {
    Serial.begin(9600);
}

void loop() {

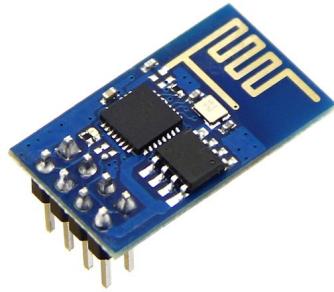
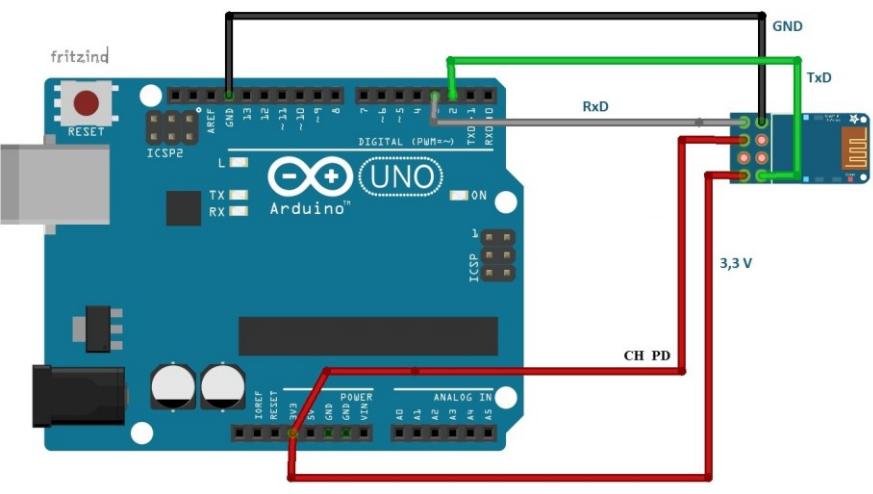
    int adc_MQ = analogRead(A0); //Lemos la salida analógica del MQ
    float voltaje = adc_MQ * (5.0 / 1023.0); //Convertimos la lectura en un
valor de voltaje

    Serial.print("adc:");
    Serial.print(adc_MQ);
    Serial.print("    voltaje:");
    Serial.println(voltaje);
    delay(100);
}

```

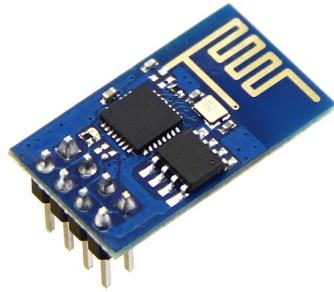
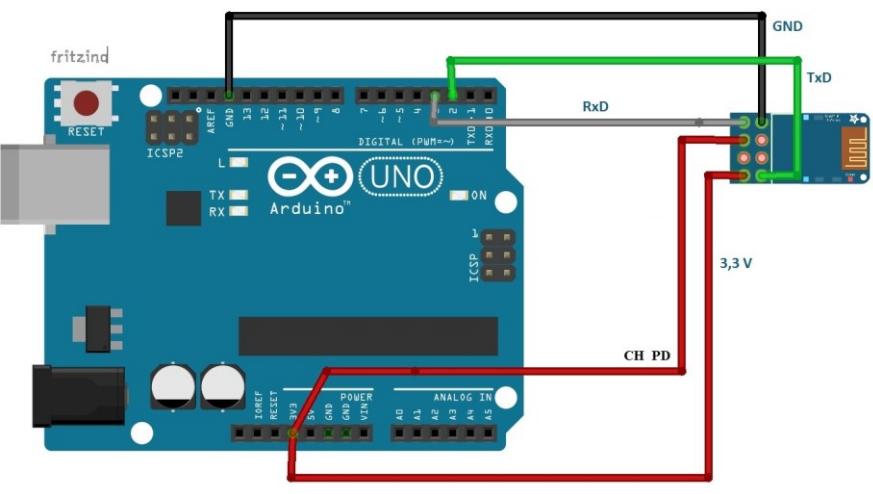
Caso de prueba N 1 Módulo WIFI ESP8266 Velocidad

Caso de prueba	Probar la velocidad del módulo Wifi
Identificador caso de prueba/s	WifiESP8266-01-pruebaVelocidad WifiESP8266-02-pruebaVelocidad WifiESP8266-03-pruebaVelocidad WifiESP8266-04-pruebaVelocidad WifiESP8266-05-pruebaVelocidad
Función probar	Comunicación por Wifi
Objetivo	Determinar la velocidad máxima de transferencia
Descripción	Se desea verificar la velocidad de conectividad que se puede alcanzar entre una computadora con Wifi y el Arduino conectado al ESP8266
Criterios de éxito	Alcanzar una velocidad que permita transmitir 10fps con un tamaño de 300kb por segundo, mínimamente
Criterios de falla	No alcanzar la velocidad requerida de fps
Precondiciones	Testear un entorno sin obstáculos y línea visual. Establecer la mayor velocidad posible de paquetes de transmisión [1] ESP8266 a 115200 baudios [2] ESP8266 a 921600 baudios [3] ESP8266 a 2500000 baudios [4] ESP8266 a 5000000 baudios [5] ESP8266 a 4500000 baudios
Necesidades para el caso de prueba	Módulo Arduino UNO ESP8266 Cables Hembra-Macho (x5)
Autor	Schlapp-Mansilla
Fecha de creación	28-3-2017
Resultados	[1]Se consigue una velocidad de 10kb/sg. Falla la prueba. [2]Se consigue una velocidad de 30kb/sg. Falla la prueba. [3]Se consigue una velocidad de 54kb/sg. Falla la prueba. [4] No se puede cumplir la prueba, dado que no es posible configurar la velocidad [5]Se consigue una velocidad de 56kb/sg. Falla la prueba.
Código fuente/s	[1]pruebaVelocidad-configuraciónWifi.ino [2]pruebaVelocidad2-configuraciónWifi.ino [3]pruebaVelocidad3-configuraciónWifi.ino [4]pruebaVelocidad4-configuraciónWifi.ino [5]pruebaVelocidad5-configuraciónWifi.ino

Imágenes	
Sketch	

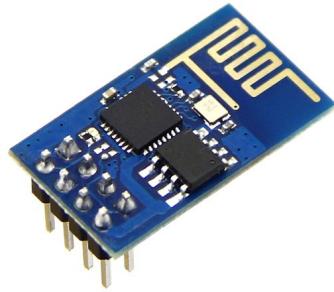
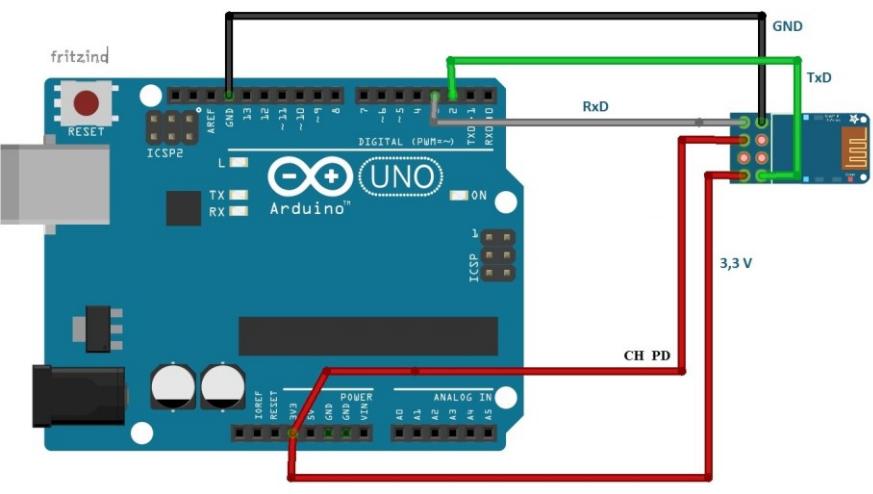
Caso de prueba N 2 Módulo WIFI ESP8266 Velocidad

Caso de prueba	Probar la velocidad del módulo Wifi
Identificador caso de prueba/s	WifiESP8266-01-pruebaVelocidad WifiESP8266-02-pruebaVelocidad WifiESP8266-03-pruebaVelocidad WifiESP8266-04-pruebaVelocidad WifiESP8266-05-pruebaVelocidad
Función probar	Comunicación por Wifi
Objetivo	Determinar la velocidad máxima de transferencia
Descripción	Se desea verificar la velocidad de conectividad que se puede alcanzar entre una computadora con Wifi y el Arduino conectado al ESP8266
Criterios de éxito	Alcanzar una velocidad que permita transmitir 10fps con un tamaño de 300kb por segundo, mínimamente
Criterios de falla	No alcanzar la velocidad requerida de fps
Precondiciones	Testear un entorno sin obstáculos y línea visual. Establecer la mayor velocidad posible de paquetes de transmisión [1] ESP8266 a 115200 baudios [2] ESP8266 a 921600 baudios [3] ESP8266 a 2500000 baudios [4] ESP8266 a 5000000 baudios [5] ESP8266 a 4500000 baudios
Necesidades para el caso de prueba	Módulo arduino UNO ESP8266 Cables Hembra-Macho (x5)
Autor	Schlapp-Mansilla
Fecha de creación	28-3-2017
Resultados	[1] Se consigue una velocidad de 10kb/sg. Falla la prueba. [2] Se consigue una velocidad de 30kb/sg. Falla la prueba. [3] Se consigue una velocidad de 54kb/sg. Falla la prueba. [4] No se puede cumplir la prueba, dado que no es posible configurar la velocidad [5] Se consigue una velocidad de 56kb/sg. Falla la prueba.
Código fuente/s	[1] pruebaVelocidad-configuraciónWifi.ino [2] pruebaVelocidad2-configuraciónWifi.ino [3] pruebaVelocidad3-configuraciónWifi.ino [4] pruebaVelocidad4-configuraciónWifi.ino [5] pruebaVelocidad5-configuraciónWifi.ino

Imágenes	
Sketch	

Caso de prueba Módulo WIFI ESP8266 Velocidad y configuración AP

Caso de prueba	Probar la velocidad del módulo Wifi
Identificador caso de prueba/s	WifiESP8266-01-ComandosAt-configuracionWifi
Función probar	Configurar módulo ESP8266 modo AP
Objetivo	Configurar el módulo ESP8266 para conocer la mayor velocidad alcanzable
Descripción	Se desea configurar el módulo como modo AP, con ssid:"SAR" sin contraseña y sin codificación. Aceptando 4 clientes simultáneos. Activando servidor DHCP. Habilitando el puerto 80 para el envío de caracteres entre PC<->Arduino a través de Putty. Comprobar los baudios, mínimos y máximos, posibles dentro del rango del Serial y Wifi
Criterios de éxito	Lograr configuración con los cambios solicitados en la descripción
Criterios de falla	No lograr la configuración deseada
Precondiciones	Testear un entorno sin obstáculos y línea visual. Actualizar el firmware del módulo a su última versión
Necesidades para el caso de prueba	Módulo Arduino UNO ESP8266 Cables Hembra-Macho (x5) Un dispositivo con terminal (Putty) para conectarse en modo RAW a la ip proporcionada por el ESP8266
Autor	Schlapp-Mansilla
Fecha de creación	28-3-2017
Resultados	La configuración es posible, pero con errores en los comandos AT. El rango en baudios permitido del Serial[9600 - 115200] el más efectivo es el 19200 El rango en baudios permitido del módulo para transmisión es [9600 - 921600] teórico. En la práctica fue posible llevarlo hasta 4.500.000
Código fuente/s	comandosAT-configuracionWifi.ino

Imágenes	
Sketch	

Código comandosAT-configuracionWIFI.ino

```
#include <SoftwareSerial.h>
SoftwareSerial ESP(9, 10); // RX | TX
/*
Enviar comando al esp8266 y verificar la respuesta del módulo, todo esto
dentro del tiempo timeout
*/
void sendData(String comando, const int timeout)
{
    long int time = millis(); // medir el tiempo actual para verificar
    timeout

    ESP.print(comando); // enviar el comando al ESP8266

    while( (time+timeout) > millis() ) //mientras no haya timeout
    {
        while(ESP.available()) //mientras haya datos por leer
        {
            // Leer los datos disponibles
            char c = ESP.read(); // leer el siguiente caracter
            Serial.print(c);
        }
    }
    return;
}
void setup()
{
    Serial.begin(9600);
    ESP.begin(19200);
    sendData("AT+CIPSTART='UDP','192.168.4.2',52485",1000);
    sendData("AT+CIPSTATUS",1000);

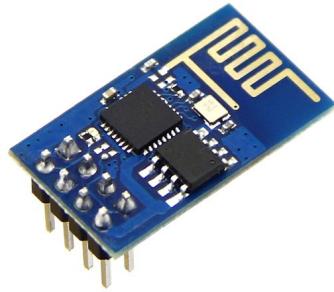
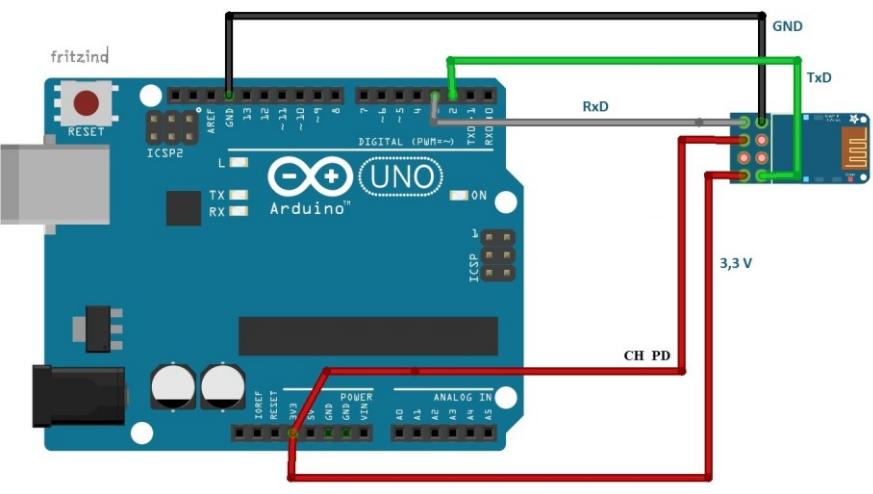
    { Serial.begin(19200);
        ESP.begin(19200);
        sendData("AT+CIPMUX=1\r\n",1000); // configurar para multiples
        conexiones
        sendData("AT+CIPSERVER=1,80\r\n",1000); // Configurar el servidor en
        el puerto 80
    }

void loop(){
    String B= ".";
    if (ESP.available())
    { char c = ESP.read();
        Serial.print(c);
    }
    if (Serial.available())
    { char c = Serial.read();
        ESP.print(c);
    }
}
```

}

Caso de prueba N 3 Módulo WIFI ESP8266 Velocidad

Caso de prueba	Probar la velocidad del módulo Wifi
Identificador caso de prueba/s	WifiESP8266-02-Pruebas-configuracionWifi
Función probar	Configurar módulo ESP8266 modo SOF AP
Objetivo	Configurar el módulo ESP8266 para conocer la mayor velocidad alcanzable
Descripción	Se desea configurar el módulo como modo AP, con ssid:"SAR" sin contraseña y sin codificación. Activando servidor DHCP. Habilitando el puerto para UDP y realizar el envío de caracteres entre PC<->Arduino a través de PacketSender. Comprobar los baudios, mínimos y máximos, posibles dentro del rango del Serial y Wifi y los distintos Buffers.
Criterios de éxito	Lograr configuración con los cambios solicitados en la descripción
Criterios de falla	No lograr la configuración deseada
Precondiciones	Testear un entorno sin obstáculos y línea visual.
Necesidades para el caso de prueba	Módulo Arduino UNO ESP8266 Cables Hembra-Macho (x5) Un dispositivo con PacketSender para generar un servidor UDP y recibir paquetes proporcionados por el ESP8266
Autor	Mansilla
Fecha de creación	30-3-2017
Resultados	
Código fuente/s	pruebaVelocidad6-configuracionWifi.ino

Imágenes	
Sketch	
Prueba1 - Buffer 64bytes PacketSender recibe 98 paquetes CIOBAUD=115200	<pre>-----El tiempo de transmisiÃ³n es:8KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg -----El tiempo de transmisiÃ³n es:8KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg -----El tiempo de transmisiÃ³n es:8KB/sg -----El tiempo de transmisiÃ³n es:8KB/sg -----El tiempo de transmisiÃ³n es:8KB/sg Rango de velocidad medido con Buffer=64KB Tiempo minimo entre paquetes:7ms<->Tiempo maximo entre paquetes:9ms Tiempo total de prueba:9 segundos Paquetes enviados197 Media:21.00paq/sg</pre>
Prueba 2 - Buffer 8 bytes PacketSender recibe 81 paquetes CIOBAUD=115200	<pre>-----El tiempo de transmisiÃ³n es:2KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg -----El tiempo de transmisiÃ³n es:2KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg -----El tiempo de transmisiÃ³n es:3KB/sg Rango de velocidad medido con Buffer=8KB Tiempo minimo entre paquetes:2ms<->Tiempo maximo entre paquetes:4ms Tiempo total de prueba:7 segundos Paquetes enviados164 Media:23.00paq/sg</pre>
Prueba 3 - Buffer 128 bytes PacketSender recibe 78 paquetes CIOBAUD=115200	<pre>-----El tiempo de transmisiÃ³n es:6KB/sg -----El tiempo de transmisiÃ³n es:6KB/sg -----El tiempo de transmisiÃ³n es:5KB/sg -----El tiempo de transmisiÃ³n es:6KB/sg -----El tiempo de transmisiÃ³n es:5KB/sg Rango de velocidad medido con Buffer=128KB Tiempo minimo entre paquetes:14ms<->Tiempo maximo entre paquetes:24ms Tiempo total de prueba:6 segundos Paquetes enviados144 Media:24.00paq/sg Transf:3.00KB/SG</pre>

Prueba 4 - Buffer 256 bytes PacketSender 110 paquetes}CIOBAUD =115200	<pre>-----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg -----El tiempo de transmisiÃ³n es:7KB/sg Rango de velocidad medido con Buffer=256KB Tiempo minimo entre paquetes:26ms<->Tiempo maximo entre paquetes:41ms Tiempo total de prueba:6 segundos Paquetes enviados40 Media:23.00paq/sg Transf:5.75KB/SG</pre>
Prueba 5 Buffer 512 bytes PacketSender 107 CIOBAUD=115200	<pre>-----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:10KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:10KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:50ms<->Tiempo maximo entre paquetes:77ms Tiempo total de prueba:5 segundos Paquetes enviados108 Media:21.00paq/sg Transf:10.50KB/SG</pre>
Prueba 6 Buffer 1024 bytes PacketSender 48 ERROR CIOBAUD=115200	<pre>-----El tiempo de transmisiÃ³n es:10KB/sg -----El tiempo de transmisiÃ³n es:10KB/sg 150ms Tiempo total de prueba:5 segundos Paquetes enviados50 Media:10.00paq/sg Transf:10.00KB/SG AT+CIPSEND=1024 Link type ERSOR</pre>
Prueba 7 Buffer 512 bytes PacketSender 6 CIOBAUD=9600	<pre>-----El tiempo de transmisiÃ³n es:0KB/sg -----El tiempo de transmisiÃ³n es:0KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:60ms<->Tiempo maximo entre paquetes:690ms Tiempo total de prueba:8 segundos Paquetes enviados13 Media:1.00paq/sg Transf:0.50KB/SG</pre>
Prueba 8 Buffer 512 bytes PacketSender 140 CIOBAUD=250000	<pre>-----El tiempo de transmisiÃ³n es:18KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:20KB/sg -----El tiempo de transmisiÃ³n es:20KB/sg -----El tiempo de transmisiÃ³n es:18KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg -----El tiempo de transmisiÃ³n es:9KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:25ms<->Tiempo maximo entre paquetes:42ms Tiempo total de prueba:6 segundos Paquetes enviados43 Media:23.00paq/sg Transf:11.50KB/SG</pre>
Prueba 9 Buffer 512 bytes CIOBAUD=500000	<pre>-----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:38KB/sg -----El tiempo de transmisiÃ³n es:33KB/sg -----El tiempo de transmisiÃ³n es:33KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:38KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:33KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:13ms<->Tiempo maximo entre paquetes:15ms Tiempo total de prueba:3 segundos Paquetes enviados65 Media:21.00paq/sg Transf:10.50KB/SG</pre>

Prueba 10 Buffer 512 CIOBAUD= 4000000	-----El tiempo de transmisiÃ³n es:50KB/sg -----El tiempo de transmisiÃ³n es:45KB/sg -----El tiempo de transmisiÃ³n es:45KB/sg -----El tiempo de transmisiÃ³n es:50KB/sg -----El tiempo de transmisiÃ³n es:41KB/sg -----El tiempo de transmisiÃ³n es:41KB/sg -----El tiempo de transmisiÃ³n es:50KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:10ms<->Tiempo maximo entre paquetes:12ms Tiempo total de prueba:5 segundos Paquetes enviados19 Media:23.00paq/sg Transf:11.50KB/SG
Prueba 11 4608000 baudios Buffer 512 bytes	-----El tiempo de transmisiÃ³n es:45KB/sg -----El tiempo de transmisiÃ³n es:45KB/sg -----El tiempo de transmisiÃ³n es:45KB/sg -----El tiempo de transmisiÃ³n es:41KB/sg -----El tiempo de transmisiÃ³n es:45KB/sg Rango de velocidad medido con Buffer=512KB Tiempo minimo entre paquetes:10ms<->Tiempo maximo entre paquetes:12ms Tiempo total de prueba:6 segundos Paquetes enviados127 Media:21.00paq/sg Transf:10.50KB/SG
Prueba 12 4608000 baudios y 256 bytes buffer	-----El tiempo de transmisiÃ³n es:41KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:41KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg -----El tiempo de transmisiÃ³n es:35KB/sg Rango de velocidad medido con Buffer=256KB Tiempo minimo entre paquetes:6ms<->Tiempo maximo entre paquetes:8ms Tiempo total de prueba:6 segundos Paquetes enviados125 Media:20.00paq/sg Transf:5.00KB/SG
Prueba 13 4608000 baudios y 768 bytes de buffer	-----El tiempo de transmisiÃ³n es:53KB/sg -----El tiempo de transmisiÃ³n es:46KB/sg -----El tiempo de transmisiÃ³n es:46KB/sg -----El tiempo de transmisiÃ³n es:49KB/sg -----El tiempo de transmisiÃ³n es:53KB/sg -----El tiempo de transmisiÃ³n es:49KB/sg -----El tiempo de transmisiÃ³n es:49KB/sg Rango de velocidad medido con Buffer=768KB Tiempo minimo entre paquetes:14ms<->Tiempo maximo entre paquetes:16ms Tiempo total de prueba:5 segundos Paquetes enviados10 Media:22.00paq/sg Transf:16.50KB/SG

Código pruebaVelocidad6-configurationWifi

```
#include <SoftwareSerial.h>
#define MAX 128
#define VELORIGINAL 115200
#define VELNUEVA 19200
//velocidad maxima 4608000
/**
 * Configuramos el BT a 4500000 y conseguimos una velocidad de 55 KB/sg
 */
SoftwareSerial ESP(3,2); // RX | TX
/*
Enviar comando al esp8266 y verificar la respuesta del módulo, todo esto
dentro del tiempo timeout
*/
void sendData(String comando, const int timeout)
{
    long int time = millis(); // medir el tiempo actual para verificar
timeout

    ESP.print(comando); // enviar el comando al ESP8266

    while( (time+timeout) > millis() ) //mientras no haya timeout
    {
        while(ESP.available()) //mientras haya datos por leer
        {
            // Leer los datos disponibles
            char c = ESP.read(); // leer el siguiente caracter
            Serial.print(c);
        }
    }
    return;
}

//Funcion para llenar un buffer con 1024 elementos
void armarBuffer(char buf[], int inicio, int fin){
    for(int i=inicio; i<=fin; i++){
        buf[i]='1';
    }
}

char frame[MAX]; //En 2048 se queda con problemas Arduino UNO, por
quedarse sin espacio
//Se configura el serial para imprimir las opciones
//con un buffer de 512, y misma velo va a 20kb/sg
//con un buffer de 256, misma velo va a 25-38kb/sg
//Con un buffer de 128, la misma velo va a 45-66kb/sg
//TODO hay que testear esto con el ESP, a otra velocidad
```

```

//Probando sin CIOMUX=0, CIPMODE=1, CIPSERVER=0, TCP rafagas de 20ms
buffer 2k
void setup()
{ Serial.begin(9600);
  Serial.println("Las opciones son: 1 para comenzar y 2 para
finalizar");
  ESP.begin(115200);

  sendData("AT+CIOBAUD="+String(VELNUEVA)+"\r\n", 3000);
  Serial.println("-----FIN CONFIG VELNUEVA-----");
  ESP.begin(VELNUEVA);

  sendData("AT+CIPSTART='UDP','192.168.4.2',56011",1000);
  Serial.println("-----conexion UDP-----");
  sendData("AT+CIPSTATUS",1000);
  Serial.println("-----STATUS-----");
  armarBuffer(frame,0,MAX-1);
}

//Este toma de la entrada estandar un 1 para comenzar la prueba de
transmitir un buffer de 1kb, en 1 sg, por medio del wifi.
//De esta forma, se puede terminar la máxima velocidad reduciendo el
timeout.
long minimo=0;
long maximo=0;
long paquetesEnviados;
long tiempoInicio;
long tiempoFinal;
bool primero=true;
void loop()
{
  if(Serial.available()){

    //Si esta disponible leo del buffer
    char opc = Serial.read();
    if(opc == '1'){
      bool detener=false;
      tiempoInicio=millis();
      paquetesEnviados=0;
      while(!detener){

        long tiempoAnterior=millis();
        sendData("AT+CIPSEND="+String(MAX)+"\r\n",0);
        ESP.print(frame);
        paquetesEnviados++;
        long tiempoActual = millis()-tiempoAnterior;
        Serial.println("-----El tiempo de transmisión
es:"+String(((1000/tiempoActual)*MAX)/1024)+" KB/sg");
      }
    }
  }
}

```

```

if (primero){
    primero=false;
    minimo=tiempoActual;
    maximo=tiempoActual;
}else{
    if (minimo>tiempoActual){
        minimo= tiempoActual;
    }
    if(maximo< tiempoActual){
        maximo= tiempoActual;
    }
}
opc = Serial.read();
if (opc=='2'){
    detener = true;
    tiempoFinal = millis();
    primero=true;
    Serial.println("Rango      de      velocidad     medido      con
Buffer="+String(MAX)+"KB           Tiempo             minimo      entre
paquetes:"+String(minimo)+"ms<->Tiempo           maximo      entre
paquetes:"+String(maximo)+"ms");
    Serial.println("Tiempo total de prueba:"+String((tiempoFinal-
tiempoInicio)/1000)+" segundos");
    Serial.println("Paquetes enviados"+String(paquetesEnviados));
    float paqTiem=paquetesEnviados/((tiempoFinal-
tiempoInicio)/1000);
    Serial.println("Media:"+String(paqTiem)+"paq/sg");
    Serial.println("Transf:"+String((paqTiem*MAX)/1024)+"KB/SG");

    sendData("AT+CIOBAUD="+String(VELORIGINAL)+"\r\n", 3000);

    ESP.begin(VELORIGINAL);
}
}
}
if(ESP.available()){//Hay algo en el buffer del wifi
    while(ESP.available()){
        char c= ESP.read();
        Serial.print(c);
    }
}
}
}

```

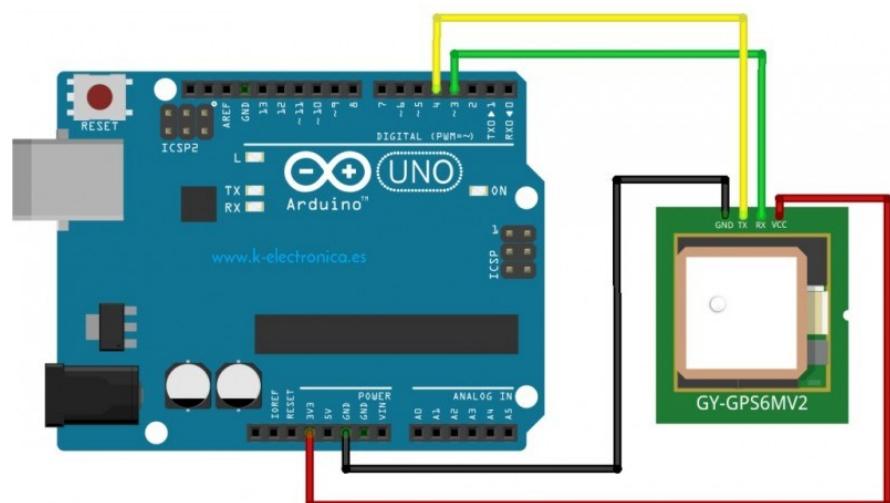
Caso de prueba Módulo GPS

Caso de prueba	Probar la conectividad del módulo GPS
Identificador caso de prueba/s	GPS-NEO6-01-conectividad
Función probar	Recepción por GPS
Objetivo	Determinar la recepción
Descripción	Se desea conectar el módulo NEO6 con el Arduino UNO, para probar la recepción de datos desde los satélites.
Criterios de éxito	Obtener una trama correctamente
Criterios de falla	No obtener una trama
Precondiciones	Testear un entorno sin obstáculos y campo abierto
Necesidades para el caso de prueba	Módulo arduino UNO NEO6-GPS Cables Hembra-Macho (x4)
Autor	Schlapp-Mansilla
Fecha de creación	25-3-2017
Resultados	[1]Se consigue la trama con el posicionamiento correspondiente en un tiempo prudente.
Código fuente/s	[1]GPS-NEO6-01-conectividad.ino

Imágenes



Sketch



Código GPS-NEO6-01Conectividad

```
#include <SoftwareSerial.h>

SoftwareSerial gps(4,3);

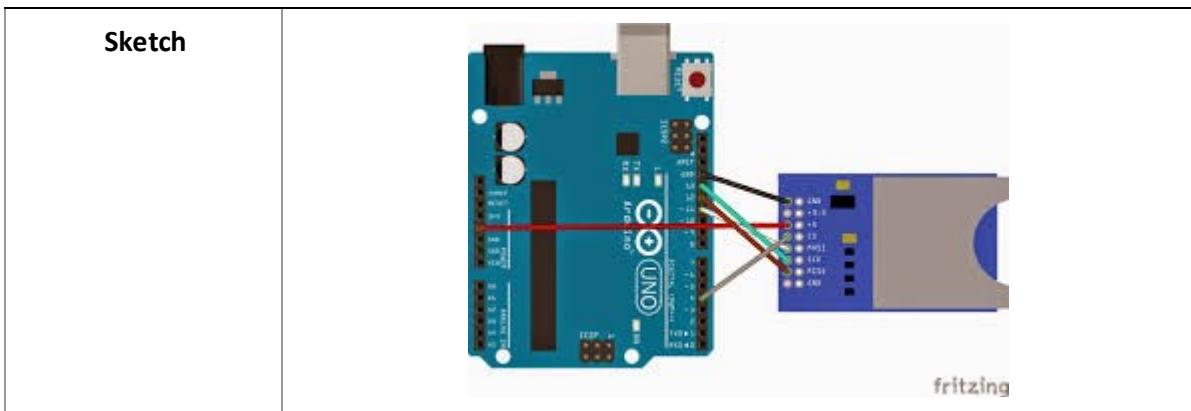
char dato=' ';

void setup()
{
    Serial.begin(9600);
    gps.begin(9600);
}

void loop()
{
    if(gps.available())
    {
        dato=gps.read();
        Serial.print(dato);
    }
}
```

Caso de prueba Módulo microSD Card Adapter

Caso de prueba	Probar el funcionamiento del microSD Card Adapter
Identificador caso de prueba/s	microSD-01-leerEscribir
Función probar	Almacenar y recuperar información en microsd de 16GB
Objetivo	Determinar velocidad de lectura y escritura desde Arduino
Descripción	Se desea almacenar y recuperar datos almacenados en una memoria microSD de 16GB conectada a un Arduino UNO. Además, tomar almacenar datos en ésta para visualizarlos en una PC
Criterios de éxito	Poder almacenar un/varios archivos/s y leerlos desde la PC
Criterios de falla	No poder almacenar y/o recuperar datos/archivos
Precondiciones	Se trabajará con el protocolo SPI
Necesidades para el caso de prueba	Módulo arduino UNO microSD Card Adapter Cables Hembra-Macho (x6)
Autor	Schlapp-Mansilla
Fecha de creación	25-4-2017
Resultados	[1]Se consigue almacenar y escribir a una microSD de 8GB
Código fuente/s	[1]microSD-01-leerEscribir.ino
Imágenes	



Código microSD-01-LeerEscribir

```
#include <SPI.h>
#include <SD.h>

File myFile;

void setup() {
    // Open serial communications and wait for port to open:
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }

    Serial.print("Initializing SD card...");

    if (!SD.begin(4)) {
        Serial.println("initialization failed!");
        return;
    }
    Serial.println("initialization done.");

    // open the file. note that only one file can be open at a time,
    // so you have to close this one before opening another.
    myFile = SD.open("test.txt", FILE_WRITE);

    // if the file opened okay, write to it:
    if (myFile) {
        Serial.print("Writing to test.txt...");
        myFile.println("testing 1, 2, 3.");
        // close the file:
        myFile.close();
        Serial.println("done.");
    } else {
        // if the file didn't open, print an error:
        Serial.println("error opening test.txt");
    }

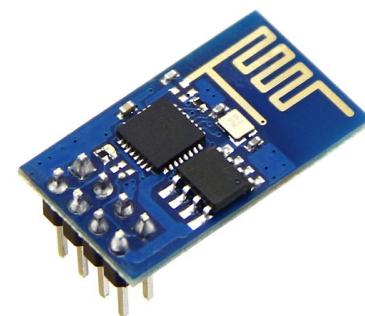
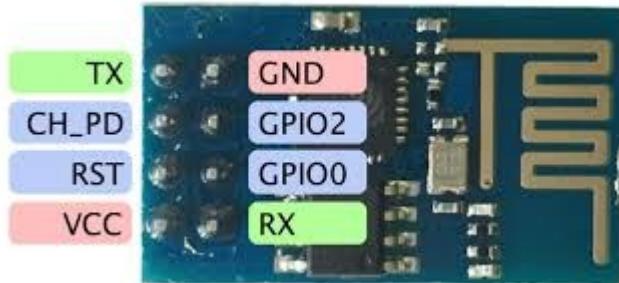
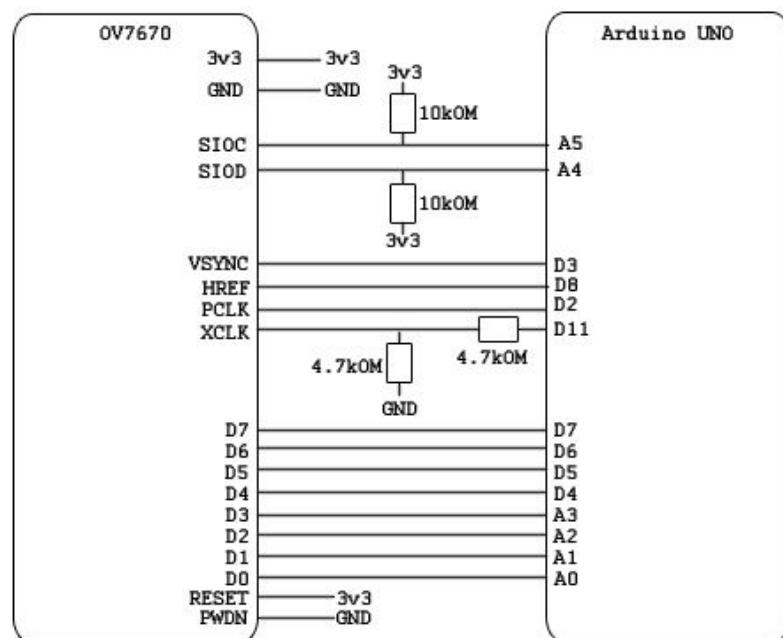
    // re-open the file for reading:
    myFile = SD.open("test.txt");
    if (myFile) {
        Serial.println("test.txt:");

        // read from the file until there's nothing else in it:
        while (myFile.available()) {
            Serial.write(myFile.read());
        }
    }
}
```

```
// close the file:  
myFile.close();  
} else {  
    // if the file didn't open, print an error:  
    Serial.println("error opening test.txt");  
}  
  
void loop() {  
    // nothing happens after setup  
}
```

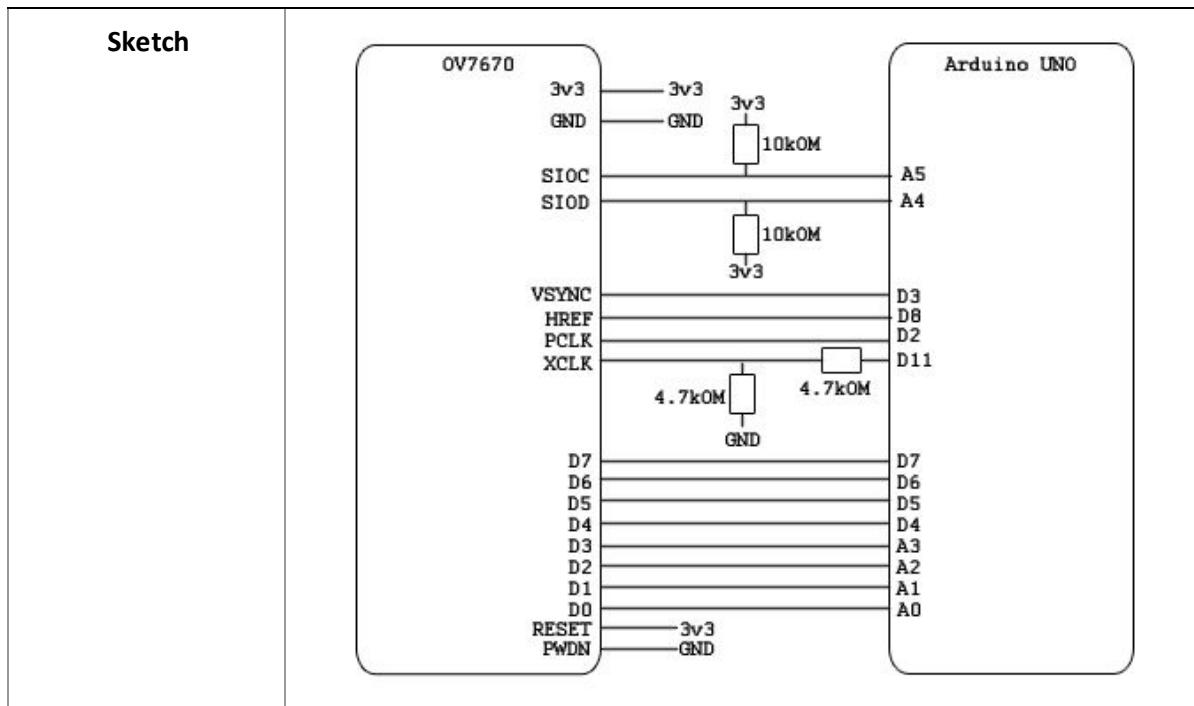
Caso de prueba Integración WIFI y Cámara

Caso de prueba	Fase 1 - Módulo WIFI ESP8266 y Cámara OV7670
Identificador caso de prueba/s	integración-fase1-transmisión
Función probar	Transmisión de imágenes a la PC
Objetivo	Determinar desempeño en la transmisión de imágenes y correcta comunicación entre el ESP8266 y OV7670 mediante un Arduino UNO.
Descripción	Se desea conectar el módulo ESP8266, y el OV7670 a un mismo Arduino UNO, para probar la transmisión, vía Wifi, de una imagen a la PC.
Criterios de éxito	Poder enviar al menos una imagen desde el Arduino UNO, a la PC.
Criterios de falla	Mala conexión o ensamblado, errores en transmisión
Precondiciones	Es necesario alimentar el ESP8266 por separado con 3.5V
Necesidades para el caso de prueba	Módulo Arduino UNO OV7670 ESP8266 Cables Hembra-Macho (18 pines) PC Portapilas 3 x AA Protoboard
Autor	Schlapp-Mansilla
Fecha de creación	14-04-2017
Resultados	[1]Falla, se supone que uno de los motivos es el alto procesamiento que efectúa el Arduino UNO, haciendo buffering de la OV7670, la cual no cuenta con chip propio. Además, para optimizar la velocidad se utilizan los registros a bajo nivel del Arduino UNO, lo que genera problemas en la transmisión al ESP8266, de esta forma éste no puede cumplir la entrega de los paquetes por WIFI.
Código fuente/s	[1]integración-fase1-transmisión.ino

Imágenes**Sketch**

Caso de prueba Cámara OV 7670

Caso de prueba	Probar la conectividad de la Cámara OV 7670 con un Arduino Uno
Identificador caso de prueba/s	CAMOV7670-01-conectividad
Función probar	Comunicación del Arduino UNO con la Cámara
Objetivo	Determinar el funcionamiento del módulo
Descripción	Se desea conectar el módulo OV7670 por medio de un módulo Arduino UNO a la PC.
Criterios de éxito	Poder enviar al menos una imagen desde el módulo, a través del Arduino UNO, a la PC.
Criterios de falla	Mala conexión o ensamblado, errores en transmisión
Precondiciones	Testear la transferencia de imágenes con una velocidad en el puerto serie de [1]CAM=OV7670 a 9200 baudios
Necesidades para el caso de prueba	Módulo Arduino UNO OV7670 Cables Hembra-Macho (18 pines) PC
Autor	Schlapp-Mansilla
Fecha de creación	8-3-2017
Resultados	[1]Se consigue transmitir una imagen con éxito.
Código fuente/s	[1]camaraOV7670.ino → En Arduino BMP.java, SimpleRead.java → En PC (Capturar y armar la imagen)
Imágenes	



Código OV7670

```
#include <stdint.h>
#include <avr/io.h>
#include <util/twi.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <SoftwareSerial.h>

#define MAX 8
#define VELORIGINAL 115200
#define VELNUEVA 19200

#define F_CPU 16000000UL
#define vga 0
#define qvga 1
#define qqvga 2
#define yuv422 0
#define rgb565 1
#define bayerRGB 2
#define camAddr_WR 0x42
#define camAddr_RD 0x43

/* Registers */
#define REG_GAIN 0x00 /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE 0x01 /* blue gain */
#define REG_RED 0x02 /* red gain */
#define REG_VREF 0x03 /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1 0x04 /* Control 1 */
#define COM1_CCIR656 0x40 /* CCIR656 enable */

#define REG_BAVE 0x05 /* U/B Average level */
#define REG_GbAVE 0x06 /* Y/Gb Average level */
#define REG_AECHH 0x07 /* AEC MS 5 bits */
#define REG_RAVE 0x08 /* V/R Average level */
#define REG_COM2 0x09 /* Control 2 */
#define COM2_SSLEEP 0x10 /* Soft sleep mode */
#define REG_PID 0xa /* Product ID MSB */
#define REG_VER 0xb /* Product ID LSB */
#define REG_COM3 0xc /* Control 3 */
#define COM3_SWAP 0x40 /* Byte swap */
#define COM3_SCALEEN 0x08 /* Enable scaling */
#define COM3_DCWEN 0x04 /* Enable downsamp/crop/window */
#define REG_COM4 0xd /* Control 4 */
#define REG_COM5 0xe /* All "reserved" */
#define REG_COM6 0xf /* Control 6 */
#define REG_AECH 0x10 /* More bits of AEC value */
```

```

#define REG_CLKRC 0x11 /* Clock control */
#define CLK_EXT 0x40 /* Use external clock directly */
#define CLK_SCALE 0x3f /* Mask for internal clock scale */
#define REG_COM7 0x12 /* Control 7 */ //REG mean address.
#define COM7_RESET 0x80 /* Register reset */
#define COM7_FMT_MASK 0x38
#define COM7_FMT_VGA 0x00
#define COM7_FMT_CIF 0x20 /* CIF format */
#define COM7_FMT_QVGA 0x10 /* QVGA format */
#define COM7_FMT_QCIF 0x08 /* QCIF format */
#define COM7_RGB 0x04 /* bits 0 and 2 - RGB format */
#define COM7_YUV 0x00 /* YUV */
#define COM7_BAYER 0x01 /* Bayer format */
#define COM7_PBAYER 0x05 /* "Processed bayer" */
#define REG_COM8 0x13 /* Control 8 */
#define COM8_FASTAEC 0x80 /* Enable fast AGC/AEC */
#define COM8_AECSTEP 0x40 /* Unlimited AEC step size */
#define COM8_BFILT 0x20 /* Band filter enable */
#define COM8_AGC 0x04 /* Auto gain enable */
#define COM8_AWB 0x02 /* White balance enable */
#define COM8_AEC 0x01 /* Auto exposure enable */
#define REG_COM9 0x14 /* Control 9- gain ceiling */
#define REG_COM10 0x15 /* Control 10 */
#define COM10_HSYNC 0x40 /* HSYNC instead of HREF */
#define COM10_PCLK_HB 0x20 /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV 0x08 /* Reverse HREF */
#define COM10_VS_LEAD 0x04 /* VSYNC on clock leading edge */
#define COM10_VS_NEG 0x02 /* VSYNC negative */
#define COM10_HS_NEG 0x01 /* HSYNC negative */
#define REG_HSTART 0x17 /* Horiz start high bits */
#define REG_HSTOP 0x18 /* Horiz stop high bits */
#define REG_VSTART 0x19 /* Vert start high bits */
#define REG_VSTOP 0x1a /* Vert stop high bits */
#define REG_PSHFT 0x1b /* Pixel delay after HREF */
#define REG_MIDH 0x1c /* Manuf. ID high */
#define REG_MIDL 0x1d /* Manuf. ID low */
#define REG_MVFP 0x1e /* Mirror / vflip */
#define MVFP_MIRROR 0x20 /* Mirror image */
#define MVFP_FLIP 0x10 /* Vertical flip */

#define REG_AEW 0x24 /* AGC upper limit */
#define REG_AEB 0x25 /* AGC lower limit */
#define REG_VPT 0x26 /* AGC/AEC fast mode op region */
#define REG_HSYST 0x30 /* HSYNC rising edge delay */
#define REG_HSYEN 0x31 /* HSYNC falling edge delay */
#define REG_HREF 0x32 /* HREF pieces */
#define REG_TSLB 0x3a /* lots of stuff */
#define TSLB_YLAST 0x04 /* UYVY or VYUY - see com13 */

```

```

#define REG_COM11 0x3b /* Control 11 */
#define COM11_NIGHT 0x80 /* NIght mode enable */
#define COM11_NMFR 0x60 /* Two bit NM frame rate */
#define COM11_HZAUTO 0x10 /* Auto detect 50/60 Hz */
#define COM11_50HZ 0x08 /* Manual 50Hz select */
#define COM11_EXP 0x02
#define REG_COM12 0x3c /* Control 12 */
#define COM12_HREF 0x80 /* HREF always */
#define REG_COM13 0x3d /* Control 13 */
#define COM13_GAMMA 0x80 /* Gamma enable */
#define COM13_UVSAT 0x40 /* UV saturation auto adjustment */
#define COM13_UVSWAP 0x01 /* V before U - w/TSLB */
#define REG_COM14 0x3e /* Control 14 */
#define COM14_DCWEN 0x10 /* DCW/PCLK-scale enable */
#define REG_EDGE 0x3f /* Edge enhancement factor */
#define REG_COM15 0x40 /* Control 15 */
#define COM15_R10F0 0x00 /* Data range 10 to F0 */
#define COM15_R01FE 0x80 /* 01 to FE */
#define COM15_R00FF 0xc0 /* 00 to FF */
#define COM15_RGB565 0x10 /* RGB565 output */
#define COM15_RGB555 0x30 /* RGB555 output */
#define REG_COM16 0x41 /* Control 16 */
#define COM16_AWBGAIN 0x08 /* AWB gain enable */
#define REG_COM17 0x42 /* Control 17 */
#define COM17_AECWIN 0xc0 /* AEC window - must match COM4 */
#define COM17_CBAR 0x08 /* DSP Color bar */
/*
 * This matrix defines how the colors are generated, must be
 * tweaked to adjust hue and saturation.
 *
 * Order: v-red, v-green, v-blue, u-red, u-green, u-blue
 * They are nine-bit signed quantities, with the sign bit
 * stored in 0x58.Sign for v-red is bit 0, and up from there.
 */
#define REG_CMATRIX_BASE 0x4f
#define CMATRIX_LEN 6
#define REG_CMATRIX_SIGN 0x58
#define REG_BRIGHT 0x55 /* Brightness */
#define REG_CONTRAS 0x56 /* Contrast control */
#define REG_GFIX 0x69 /* Fix gain control */
#define REG_REG76 0x76 /* OV's name */
#define R76_BLKPCOR 0x80 /* Black pixel correction enable */
#define R76_WHTPCOR 0x40 /* White pixel correction enable */
#define REG_RGB444 0x8c /* RGB 444 control */
#define R444_ENABLE 0x02 /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX 0x01 /* Empty nibble at end */
#define REG_HAECC1 0x9f /* Hist AEC/AGC control 1 */
#define REG_HAECC2 0xa0 /* Hist AEC/AGC control 2 */

```

```

#define REG_BD50MAX          0xa5 /* 50hz banding step limit */
#define REG_HAECC3           0xa6 /* Hist AEC/AGC control 3 */
#define REG_HAECC4           0xa7 /* Hist AEC/AGC control 4 */
#define REG_HAECC5           0xa8 /* Hist AEC/AGC control 5 */
#define REG_HAECC6           0xa9 /* Hist AEC/AGC control 6 */
#define REG_HAECC7           0xaa /* Hist AEC/AGC control 7 */
#define REG_BD60MAX          0xab /* 60hz banding step limit */
#define REG_GAIN              0x00 /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE              0x01 /* blue gain */
#define REG_RED               0x02 /* red gain */
#define REG_VREF              0x03 /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1              0x04 /* Control 1 */
#define COM1_CCIR656          0x40 /* CCIR656 enable */
#define REG_BAVE              0x05 /* U/B Average level */
#define REG_GbAVE             0x06 /* Y/Gb Average level */
#define REG_AECHH             0x07 /* AEC MS 5 bits */
#define REG_RAVE              0x08 /* V/R Average level */
#define REG_COM2              0x09 /* Control 2 */
#define COM2_SSLEEP           0x10 /* Soft sleep mode */
#define REG_PID               0xa /* Product ID MSB */
#define REG_VER               0xb /* Product ID LSB */
#define REG_COM3              0xc /* Control 3 */
#define COM3_SWAP              0x40 /* Byte swap */
#define COM3_SCALEEN          0x08 /* Enable scaling */
#define COM3_DCWEN            0x04 /* Enable downsamp/crop/window */
#define REG_COM4              0xd /* Control 4 */
#define REG_COM5              0xe /* All "reserved" */
#define REG_COM6              0xf /* Control 6 */
#define REG_AECH               0x10 /* More bits of AEC value */
#define REG_CLKRC              0x11 /* Clocl control */
#define CLK_EXT                0x40 /* Use external clock directly */
#define CLK_SCALE              0x3f /* Mask for internal clock scale */
#define REG_COM7              0x12 /* Control 7 */
#define COM7_RESET             0x80 /* Register reset */
#define COM7_FMT_MASK          0x38
#define COM7_FMT_VGA           0x00
#define COM7_FMT_CIF           0x20 /* CIF format */
#define COM7_FMT_QVGA          0x10 /* QVGA format */
#define COM7_FMT_QCIF          0x08 /* QCIF format */
#define COM7_RGB                0x04 /* bits 0 and 2 - RGB format */
#define COM7_YUV               0x00 /* YUV */
#define COM7_BAYER              0x01 /* Bayer format */
#define COM7_PBAYER             0x05 /* "Processed bayer" */
#define REG_COM8              0x13 /* Control 8 */
#define COM8_FASTAEC            0x80 /* Enable fast AGC/AEC */
#define COM8_AECSTEP            0x40 /* Unlimited AEC step size */
#define COM8_BFILT              0x20 /* Band filter enable */
#define COM8_AGC               0x04 /* Auto gain enable */

```

```

#define COM8_AWB      0x02 /* White balance enable */
#define COM8_AEC      0x01 /* Auto exposure enable */
#define REG_COM9      0x14 /* Control 9- gain ceiling */
#define REG_COM10     0x15 /* Control 10 */
#define COM10_HSYNC          0x40 /* HSYNC instead of HREF */
#define COM10_PCLK_HB        0x20 /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV        0x08 /* Reverse HREF */
#define COM10_VS_LEAD         0x04 /* VSYNC on clock leading edge */
#define COM10_VS_NEG          0x02 /* VSYNC negative */
#define COM10_HS_NEG          0x01 /* HSYNC negative */
#define REG_HSTART        0x17 /* Horiz start high bits */
#define REG_HSTOP         0x18 /* Horiz stop high bits */
#define REG_VSTART         0x19 /* Vert start high bits */
#define REG_VSTOP          0x1a /* Vert stop high bits */
#define REG_PSHFT          0x1b /* Pixel delay after HREF */
#define REG_MIDH           0x1c /* Manuf. ID high */
#define REG_MIDL           0x1d /* Manuf. ID low */
#define REG_MVFP           0x1e /* Mirror / vflip */
#define MVFP_MIRROR          0x20 /* Mirror image */
#define MVFP_FLIP            0x10 /* Vertical flip */
#define REG_AEW             0x24 /* AGC upper limit */
#define REG_AEB             0x25 /* AGC lower limit */
#define REG_VPT              0x26 /* AGC/AEC fast mode op region */
#define REG_HSYST          0x30 /* HSYNC rising edge delay */
#define REG_HSYEN          0x31 /* HSYNC falling edge delay */
#define REG_HREF            0x32 /* HREF pieces */
#define REG_TSLB            0x3a /* lots of stuff */
#define TSLB_YLAST          0x04 /* UYVY or VYUY - see com13 */
#define REG_COM11          0x3b /* Control 11 */
#define COM11_NIGHT          0x80 /* NIght mode enable */
#define COM11_NMFR           0x60 /* Two bit NM frame rate */
#define COM11_HZAUTO         0x10 /* Auto detect 50/60 Hz */
#define COM11_50HZ           0x08 /* Manual 50Hz select */
#define COM11_EXP            0x02
#define REG_COM12          0x3c /* Control 12 */
#define COM12_HREF           0x80 /* HREF always */
#define REG_COM13          0x3d /* Control 13 */
#define COM13_GAMMA          0x80 /* Gamma enable */
#define COM13_UVSAT          0x40 /* UV saturation auto adjustment */
#define COM13_UVSWAP         0x01 /* V before U - w/TSLB */
#define REG_COM14          0x3e /* Control 14 */
#define COM14_DCWEN          0x10 /* DCW/PCLK-scale enable */
#define REG_EDGE             0x3f /* Edge enhancement factor */
#define REG_COM15          0x40 /* Control 15 */
#define COM15_R10F0          0x00 /* Data range 10 to F0 */
#define COM15_R01FE          0x80 /* 01 to FE */
#define COM15_R00FF          0xc0 /* 00 to FF */
#define COM15_RGB565         0x10 /* RGB565 output */

```

```

#define COM15_RGB555          0x30 /* RGB555 output */
#define REG_COM16   0x41 /* Control 16 */
#define COM16_AWBGAIN         0x08 /* AWB gain enable */
#define REG_COM17   0x42 /* Control 17 */
#define COM17_AECWIN          0xc0 /* AEC window - must match COM4 */
#define COM17_CBAR            0x08 /* DSP Color bar */

#define CMATRIX_LEN           6
#define REG_BRIGHT             0x55 /* Brightness */
#define REG_REG76   0x76 /* OV's name */
#define R76_BLKPCOR           0x80 /* Black pixel correction enable */
#define R76_WHTPCOR           0x40 /* White pixel correction enable */
#define REG_RGB444             0x8c /* RGB 444 control */
#define R444_ENABLE            0x02 /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX              0x01 /* Empty nibble at end */
#define REG_HAECC1             0x9f /* Hist AEC/AGC control 1 */
#define REG_HAECC2             0xa0 /* Hist AEC/AGC control 2 */
#define REG_BD50MAX            0xa5 /* 50hz banding step limit */
#define REG_HAECC3             0xa6 /* Hist AEC/AGC control 3 */
#define REG_HAECC4             0xa7 /* Hist AEC/AGC control 4 */
#define REG_HAECC5             0xa8 /* Hist AEC/AGC control 5 */
#define REG_HAECC6             0xa9 /* Hist AEC/AGC control 6 */
#define REG_HAECC7             0xaa /* Hist AEC/AGC control 7 */
#define REG_BD60MAX            0xab /* 60hz banding step limit */
#define MTX1                   0x4f /* Matrix Coefficient 1 */
#define MTX2                   0x50 /* Matrix Coefficient 2 */
#define MTX3                   0x51 /* Matrix Coefficient 3 */
#define MTX4                   0x52 /* Matrix Coefficient 4 */
#define MTX5                   0x53 /* Matrix Coefficient 5 */
#define MTX6                   0x54 /* Matrix Coefficient 6 */
#define REG_CONTRAS            0x56 /* Contrast control */
#define MTXS                   0x58 /* Matrix Coefficient Sign */
#define AWBC7                  0x59 /* AWB Control 7 */
#define AWBC8                  0x5a /* AWB Control 8 */
#define AWBC9                  0x5b /* AWB Control 9 */
#define AWBC10                 0x5c /* AWB Control 10 */
#define AWBC11                 0x5d /* AWB Control 11 */
#define AWBC12                 0x5e /* AWB Control 12 */
#define REG_GFI                0x69 /* Fix gain control */
#define GGAIN                  0x6a /* G Channel AWB Gain */
#define DBLV                   0x6b
#define AWBCTR3                0x6c /* AWB Control 3 */
#define AWBCTR2                0x6d /* AWB Control 2 */
#define AWBCTR1                0x6e /* AWB Control 1 */
#define AWBCTR0                0x6f /* AWB Control 0 */

```

```
SoftwareSerial ESP(9,10); // RX | TX
```

```

/*
Enviar comando al esp8266 y verificar la respuesta del módulo, todo esto
dentro del tiempo timeout
*/
void sendData(String comando, const int timeout)
{
    long int time = millis(); // medir el tiempo actual para verificar
timeout

    ESP.print(comando); // enviar el comando al ESP8266

    while( (time+timeout) > millis() ) //mientras no haya timeout
    {
        while(ESP.available()) //mientras haya datos por leer
        {
            // Leer los datos disponibles
            char c = ESP.read(); // leer el siguiente caracter
            Serial.print(c);
        }
    }
    //Serial.println("Se ingreso el comando: " + comando);
    return;
}

struct regval_list{
    uint8_t reg_num;
    uint16_t value;
};

const struct regval_list qvga_ov7670[] PROGMEM = {
{ REG_COM14, 0x19 },
{ 0x72, 0x11 },
{ 0x73, 0xf1 },

{ REG_HSTART, 0x16 },
{ REG_HSTOP, 0x04 },
{ REG_HREF, 0xa4 },
{ REG_VSTART, 0x02 },
{ REG_VSTOP, 0x7a },
{ REG_VREF, 0x0a },

/* { REG_HSTART, 0x16 },
{ REG_HSTOP, 0x04 },
{ REG_HREF, 0x24 },
{ REG_VSTART, 0x02 },
{ REG_VSTOP, 0x7a },
```

```

{ REG_VREF, 0x0a }, */
{ 0xff, 0xff }, /* END MARKER */
};

const struct regval_list yuv422_ov7670[] PROGMEM = {
{ REG_COM7, 0x0 }, /* Selects YUV mode */
{ REG_RGB444, 0 }, /* No RGB444 please */
{ REG_COM1, 0 },
{ REG_COM15, COM15_R00FF },
{ REG_COM9, 0x6A }, /* 128x gain ceiling; 0x8 is reserved bit */
{ 0x4f, 0x80 }, /* "matrix coefficient 1" */
{ 0x50, 0x80 }, /* "matrix coefficient 2" */
{ 0x51, 0 }, /* vb */
{ 0x52, 0x22 }, /* "matrix coefficient 4" */
{ 0x53, 0x5e }, /* "matrix coefficient 5" */
{ 0x54, 0x80 }, /* "matrix coefficient 6" */
{ REG_COM13, COM13_UVSAT },
{ 0xff, 0xff }, /* END MARKER */
};

const struct regval_list ov7670_default_regs[] PROGMEM = { //from the
linux driver
{ REG_COM7, COM7_RESET },
{ REG_TSLB, 0x04 }, /* OV */
{ REG_COM7, 0 }, /* VGA */
/*
* Set the hardware window. These values from OV don't entirely
* make sense - hstop is less than hstart. But they work...
*/
{ REG_HSTART, 0x13 }, { REG_HSTOP, 0x01 },
{ REG_HREF, 0xb6 }, { REG_VSTART, 0x02 },
{ REG_VSTOP, 0x7a }, { REG_VREF, 0x0a },

{ REG_COM3, 0 }, { REG_COM14, 0 },
/* Mystery scaling numbers */
{ 0x70, 0x3a }, { 0x71, 0x35 },
{ 0x72, 0x11 }, { 0x73, 0xf0 },
{ 0xa2, /* 0x02 changed to 1 */1 }, { REG_COM10, 0x0 },
/* Gamma curve values */
{ 0x7a, 0x20 }, { 0x7b, 0x10 },
{ 0x7c, 0x1e }, { 0x7d, 0x35 },
{ 0x7e, 0x5a }, { 0x7f, 0x69 },
{ 0x80, 0x76 }, { 0x81, 0x80 },
{ 0x82, 0x88 }, { 0x83, 0x8f },
{ 0x84, 0x96 }, { 0x85, 0xa3 },
{ 0x86, 0xaf }, { 0x87, 0xc4 },
{ 0x88, 0xd7 }, { 0x89, 0xe8 },
/* AGC and AEC parameters. Note we start by disabling those features,

```

```

then turn them only after tweaking the values. */
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP },
{ REG_GAIN, 0 }, { REG_AECH, 0 },
{ REG_COM4, 0x40 }, /* magic reserved bit */
{ REG_COM9, 0x18 }, /* 4x gain + magic rsvd bit */
{ REG_BD50MAX, 0x05 }, { REG_BD60MAX, 0x07 },
{ REG_AEW, 0x95 }, { REG_AEB, 0x33 },
{ REG_VPT, 0xe3 }, { REG_HAECC1, 0x78 },
{ REG_HAECC2, 0x68 }, { 0xa1, 0x03 }, /* magic */
{ REG_HAECC3, 0xd8 }, { REG_HAECC4, 0xd8 },
{ REG_HAECC5, 0xf0 }, { REG_HAECC6, 0x90 },
{ REG_HAECC7, 0x94 },
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC },
{ 0x30, 0 }, { 0x31, 0 }, //disable some delays
/* Almost all of these are magic "reserved" values. */
{ REG_COM5, 0x61 }, { REG_COM6, 0x4b },
{ 0x16, 0x02 }, { REG_MVFP, 0x07 },
{ 0x21, 0x02 }, { 0x22, 0x91 },
{ 0x29, 0x07 }, { 0x33, 0x0b },
{ 0x35, 0x0b }, { 0x37, 0x1d },
{ 0x38, 0x71 }, { 0x39, 0x2a },
{ REG_COM12, 0x78 }, { 0x4d, 0x40 },
{ 0x4e, 0x20 }, { REG_GFIX, 0 },
/*{0x6b, 0x4a},*/{ 0x74, 0x10 },
{ 0x8d, 0x4f }, { 0x8e, 0 },
{ 0x8f, 0 }, { 0x90, 0 },
{ 0x91, 0 }, { 0x96, 0 },
{ 0x9a, 0 }, { 0xb0, 0x84 },
{ 0xb1, 0x0c }, { 0xb2, 0x0e },
{ 0xb3, 0x82 }, { 0xb8, 0xa },

/* More reserved magic, some of which tweaks white balance */
{ 0x43, 0xa }, { 0x44, 0xf0 },
{ 0x45, 0x34 }, { 0x46, 0x58 },
{ 0x47, 0x28 }, { 0x48, 0x3a },
{ 0x59, 0x88 }, { 0x5a, 0x88 },
{ 0x5b, 0x44 }, { 0x5c, 0x67 },
{ 0x5d, 0x49 }, { 0x5e, 0x0e },
{ 0x6c, 0xa }, { 0x6d, 0x55 },
{ 0x6e, 0x11 }, { 0x6f, 0x9e }, /* it was 0x9F "9e for advance AWB" */
{ 0x6a, 0x40 }, { REG_BLUE, 0x40 },
{ REG_RED, 0x60 },
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC |
COM8_AWB },

/* Matrix coefficients */
{ 0x4f, 0x80 }, { 0x50, 0x80 },
{ 0x51, 0 }, { 0x52, 0x22 },

```

```

{ 0x53, 0x5e }, { 0x54, 0x80 },
{ 0x58, 0x9e },

{ REG_COM16, COM16_AWBGAIN }, { REG_EDGE, 0 },
{ 0x75, 0x05 }, { REG_REG76, 0xe1 },
{ 0x4c, 0 }, { 0x77, 0x01 },
{ REG_COM13, /*0xc3*/0x48 }, { 0x4b, 0x09 },
{ 0xc9, 0x60 }, /*{REG_COM16, 0x38},*/
{ 0x56, 0x40 },

{ 0x34, 0x11 }, { REG_COM11, COM11_EXP | COM11_HZAUTO },
{ 0xa4, 0x82/*Was 0x88*/ }, { 0x96, 0 },
{ 0x97, 0x30 }, { 0x98, 0x20 },
{ 0x99, 0x30 }, { 0x9a, 0x84 },
{ 0x9b, 0x29 }, { 0x9c, 0x03 },
{ 0x9d, 0x4c }, { 0x9e, 0x3f },
{ 0x78, 0x04 },

/* Extra-weird stuff. Some sort of multiplexor register */
{ 0x79, 0x01 }, { 0xc8, 0xf0 },
{ 0x79, 0x0f }, { 0xc8, 0x00 },
{ 0x79, 0x10 }, { 0xc8, 0x7e },
{ 0x79, 0x0a }, { 0xc8, 0x80 },
{ 0x79, 0x0b }, { 0xc8, 0x01 },
{ 0x79, 0x0c }, { 0xc8, 0x0f },
{ 0x79, 0x0d }, { 0xc8, 0x20 },
{ 0x79, 0x09 }, { 0xc8, 0x80 },
{ 0x79, 0x02 }, { 0xc8, 0xc0 },
{ 0x79, 0x03 }, { 0xc8, 0x40 },
{ 0x79, 0x05 }, { 0xc8, 0x30 },
{ 0x79, 0x26 },
{ 0xff, 0xff }, /* END MARKER */
};

void error_led(void){
    DDRB |= 32; //make sure led is output
    while (1){ //wait for reset
        PORTB ^= 32; // toggle led
        _delay_ms(100);
    }
}

void twiStart(void){
    TWCR = _BV(TWINT) | _BV(TWSTA) | _BV(TWEN); //send start
    while (!(TWCR & (1 << TWINT))); //wait for start to be transmitted
    if ((TWSR & 0xF8) != TW_START)
        error_led();
}

```

```

}

void twiWriteByte(uint8_t DATA, uint8_t type){
    TWDR = DATA;
    TWCR = _BV(TWINT) | _BV(TWEN);
    while (!(TWCR & (1 << TWINT))) {}
    if ((TWSR & 0xF8) != type)
        error_led();
}

void twiAddr(uint8_t addr, uint8_t typeTWI){
    TWDR = addr;//send address
    TWCR = _BV(TWINT) | _BV(TWEN); /* clear interrupt to start
transmission */
    while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
    if ((TWSR & 0xF8) != typeTWI)
        error_led();
}

void wrReg(uint8_t reg, uint8_t dat){
    //send start condition
    twiStart();
    twiAddr(camAddr_WR, TW_MT_SLA_ACK);
    twiWriteByte(reg, TW_MT_DATA_ACK);
    twiWriteByte(dat, TW_MT_DATA_ACK);
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWST0);//send stop
    _delay_ms(1);
}

static uint8_t twiRd(uint8_t nack){
    if (nack){
        TWCR = _BV(TWINT) | _BV(TWEN);
        while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
        if ((TWSR & 0xF8) != TW_MR_DATA_NACK)
            error_led();
        return TWDR;
    }
    else{
        TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);
        while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
        if ((TWSR & 0xF8) != TW_MR_DATA_ACK)
            error_led();
        return TWDR;
    }
}

uint8_t rdReg(uint8_t reg){
    uint8_t dat;

```

```

twiStart();
twiAddr(camAddr_WR, TW_MT_SLA_ACK);
twiWriteByte(reg, TW_MT_DATA_ACK);
TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWST0); //send stop
_delay_ms(1);
twiStart();
twiAddr(camAddr_RD, TW_MR_SLA_ACK);
dat = twiRd(1);
TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWST0); //send stop
_delay_ms(1);
return dat;
}

void wrSensorRegs8_8(const struct regval_list reglist[]){
    uint8_t reg_addr, reg_val;
    const struct regval_list *next = reglist;
    while ((reg_addr != 0xff) | (reg_val != 0xff)){
        reg_addr = pgm_read_byte(&next->reg_num);
        reg_val = pgm_read_byte(&next->value);
        wrReg(reg_addr, reg_val);
        next++;
    }
}

void setColor(void){
    wrSensorRegs8_8(yuv422_ov7670);
}

void setRes(void){
    wrReg(REG_COM3, 4); // REG_COM3 enable scaling
    wrSensorRegs8_8(qvga_ov7670);
}

void camInit(void){
    wrReg(0x12, 0x80);
    _delay_ms(100);
    wrSensorRegs8_8.ov7670_default_regs);
    wrReg(REG_COM10, 32); //PCLK does not toggle on HBLANK.
}

void arduinoUnoInut(void) {
    cli(); //disable interrupts

    /* Setup the 8mhz PWM clock
     * This will be on pin 11*/
    DDRB |= (1 << 3); //pin 11
    ASSR &= ~(_BV(EXCLK) | _BV(AS2));
    TCCR2A = (1 << COM2A0) | (1 << WGM21) | (1 << WGM20);
}

```

```

TCCR2B = (1 << WGM22) | (1 << CS20);
OCR2A = 0; // (F_CPU)/(2*(X+1))
DDRC &= ~15; // low d0-d3 camera
DDRD &= ~252; // d7-d4 and interrupt pins
_delay_ms(3000);

// set up twi for 100khz
TWSR &= ~3; // disable prescaler for TWI (Two wire status register)
Informa el estado de las acciones TWI
TWBR = 72; // set to 100khz (Two wire bit rate) controla la frecuencia de reloj (SCL)

// enable serial
//UBRR0H = 0; // High ----- INVESTIGAR, buscar como UART o USART de Arduino
//UBRR0L = 1; // Low ----- 0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600 baud rate.

//UBRR0 = 103;
//UCSR0A |= 2; // double speed aysnc
//UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable receiver and transmitter
//UCSR0C = 6; //async 1 stop bit 8bit char no parity bits
}

void StringPgm(const char * str){
do{
    while (!(UCSR0A & (1 << UDRE0))); // wait for byte to transmit
    UDR0 = pgm_read_byte_near(str);
    while (!(UCSR0A & (1 << UDRE0))); // wait for byte to transmit
} while (pgm_read_byte_near(++str));
}

static void captureImg(uint16_t wg, uint16_t hg){
    uint16_t y, x;

    //StringPgm(PSTR("*RDY*"));

    while (!(PIND & 8)); // wait for high
    while ((PIND & 8)); // wait for low

    y = hg;
    while (y--){
        x = wg;
        //while (!(PIND & 256)); // wait for high
        while (x--){
            while ((PIND & 4)); // wait for low

```

```

        //UDR0 = (PINC & 15) | (PIND & 240);
        //UDR0 = 'A';
        int datos = (PINC & 15) | (PIND & 240);
        sendData("AT+CIPSEND=4\r\n",0);
        ESP.print(datos);
        //while (!(UCSR0A & (1 << UDRE0))); //wait for byte to transmit

        while (!(PIND & 4)); //wait for high
        while ((PIND & 4)); //wait for low
        while (!(PIND & 4)); //wait for high
    }
    // while ((PIND & 256)); //wait for low
}
_delay_ms(100);
}

void setup(){

    //sendData("AT+RST\r\n", 0); //Reseteo de modulo
    //sendData("AT+CIOBAUD=115200\r\n", 0); // Configuro la velocidad en
baudios
    //Serial.println("-----FIN CONFIG VELNUEVA-----");
    //ESP.begin(VELNUEVA);
    //sendData("AT+CIPSTART=\\"UDP\\",\\"192.168.4.2\\",50494",0);
//Establecer conexion con el IP/Puerto
    //Serial.println("-----conexion UDP-----");
    //sendData("AT+CIPSTATUS",1000);
    //armarBuffer(frame,0,MAX-1);

    // Configuración de camara y captura de imagenes

    Serial.begin(19200);
    //Configuracion de modulo Wifi
    ESP.begin(115200); //Setea la velocidad en la que va a trabajar el
modulo
    Serial.println("-----Camara configurada-----");
}

int comandosAT(int conf){
    Serial.println("-----Comandos AT-----");
    while (conf ==0){
        if (ESP.available())
        { char c = ESP.read() ;
            Serial.print(c);
        }
        if (Serial.available())
        { char c = Serial.read();
            ESP.print(c);
        }
    }
}

```

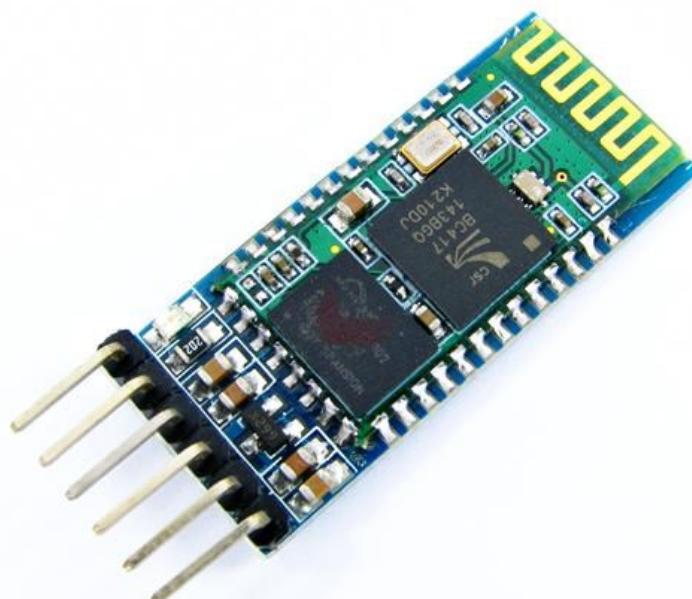
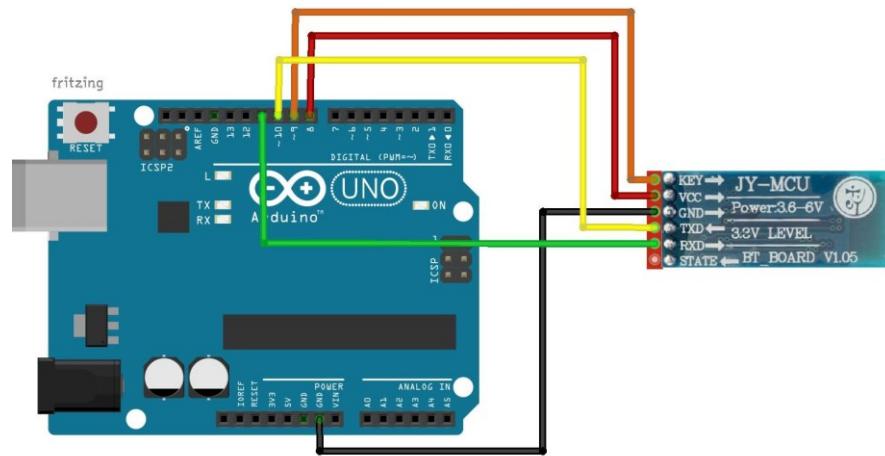
```
if (c=='X'){
    conf = 1;
}
}

arduinoUnoInut();
Serial.println("-----Paso arduinoUnoInut-----");
camInit();
Serial.println("-----Paso camInit-----");
setRes();
Serial.println("-----Paso setRes-----");
setColor();
Serial.println("-----Paso setColor-----");
wrReg(0x11, 11); //Earlier it had the value: wrReg(0x11, 12); New
version works better for me :) !!!!
return conf;
}
int conf = 0;

void loop(){
conf = comandosAT(conf);
if (conf == 1){
    captureImg(320, 240); // 320x240 formato por defecto
}
conf=0;
}
```

Caso de prueba Módulo Bluetooth HC05-01

Caso de prueba	Probar la velocidad del Bluetooth
Identificador caso de prueba	BluetoothHC05-01-pruebaVelocidad
Función probar	Comunicación por Bluetooth
Objetivo	Determinar la velocidad máxima de transferencia
Descripción	Se desea verificar la velocidad de conectividad que se puede alcanzar entre una computadora con Bluetooth y el Arduino conectado al HC05
Criterios de éxito	Alcanzar una velocidad que permita transmitir 10fps con un tamaño de 300kb por segundo, mínimamente
Criterios de falla	No alcanzar la velocidad requerida de fps
Precondiciones	Testear un entorno sin obstáculos y línea visual. Establecer la mayor velocidad posible de baudios de transmisión
Necesidades para el caso de prueba	Módulo arduino UNO BT HC05 Cables Hembra-Macho (x6)
Autor	Schlapp-Mansilla
Fecha de creación	8-3-2017
Resultados	Se estableció la comunicación entre los módulos Bluetooth, Arduino<->PC. La prueba no fue exitosa. Se alcanzó una velocidad de transmisión de 1,17kb/sg.
Código fuente	ComunicaciónBluetooth.ino

Imágenes**Sketch**

Comunicación Bluetooth.ino

```
#include <SoftwareSerial.h>

SoftwareSerial bluetooth(10, 11); // RX, TX
long cont;
long tiempoini;

void setup()
{
  Serial.begin(9600);
  bluetooth.begin(9600);
  cont = 0;
}

int paso = 0;
long segundos=0;
long tiempofin;
unsigned long antes=0;
unsigned long despues=0;

void loop()
{
  tiempoini = millis();
  while(segundos < 1.0){
    antes=millis();
    if ( bluetooth.available()){
      bluetooth.read();
      despues = millis() - antes;
      cont++;
      paso=1;
    }
    tiempofin = millis()- tiempoini;
    segundos = tiempofin/1000;
  }
  if (paso != 0){
    Serial.println("Bytes leidos:"+ String(cont)+" tiempo antesDespues
micros:"+String(despues));
    paso =0;
  }
  segundos =0;
  cont=0;
}
```

Anexo de códigos

En este anexo se adjuntaron los códigos más relevantes, tanto del lado del cliente como del servidor, que se desarrollaron para el funcionamiento del SAR.

Códigos del lado del servidor

Código StandarFirmata utilizado en el Arduino MEGA

```
/*
Firmata is a generic protocol for communicating with microcontrollers
from software on a host computer. It is intended to work with
any host computer software package.

To download a host software package, please click on the following link
to open the list of Firmata client libraries in your default browser.

https://github.com/firmata/arduino#firmata-client-libraries

Copyright (C) 2006-2008 Hans-Christoph Steiner. All rights reserved.
Copyright (C) 2010-2011 Paul Stoffregen. All rights reserved.
Copyright (C) 2009 Shigeru Kobayashi. All rights reserved.
Copyright (C) 2009-2016 Jeff Hoefs. All rights reserved.

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

See file LICENSE.txt for further informations on licensing terms.

Last updated October 16th, 2016
*/
#include <Servo.h>
#include <Wire.h>
#include <Firmata.h>

#define I2C_WRITE B00000000
#define I2C_READ B00001000
#define I2C_READ_CONTINUOUSLY B00010000
#define I2C_STOP_READING B00011000
#define I2C_READ_WRITE_MODE_MASK B00011000
#define I2C_10BIT_ADDRESS_MODE_MASK B00100000
#define I2C_END_TX_MASK B01000000
#define I2C_STOP_TX 1
#define I2C_RESTART_TX 0
#define I2C_MAX_QUERIES 8
#define I2C_REGISTER_NOT_SPECIFIED -1
```

```
// the minimum interval for sampling analog input
#define MINIMUM_SAMPLING_INTERVAL    1

/*=====
=====
 * GLOBAL VARIABLES
=====

=====*/
#ifndef FIRMATA_SERIAL_FEATURE
SerialFirmata serialFeature;
#endif

/* analog inputs */
int analogInputsToReport = 0; // bitwise array to store pin reporting

/* digital input ports */
byte reportPINS[TOTAL_PORTS];      // 1 = report this port, 0 = silence
byte previousPINS[TOTAL_PORTS];     // previous 8 bits sent

/* pins configuration */
byte portConfigInputs[TOTAL_PORTS]; // each bit: 1 = pin in INPUT, 0 =
anything else

/* timer variables */
unsigned long currentMillis;        // store the current value from
millis()
unsigned long previousMillis;       // for comparison with currentMillis
unsigned int samplingInterval = 19; // how often to run the main loop (in
ms)

/* i2c data */
struct i2c_device_info {
    byte addr;
    int reg;
    byte bytes;
    byte stopTX;
};

/* for i2c read continuous more */
i2c_device_info query[I2C_MAX_QUERIES];

byte i2cRxData[64];
boolean isI2CEnabled = false;
signed char queryIndex = -1;
```

```
// default delay time between i2c read request and Wire.requestFrom()
unsigned int i2cReadDelayTime = 0;

Servo servos[MAX_SERVOS];
byte servoPinMap[TOTAL_PINS];
byte detachedServos[MAX_SERVOS];
byte detachedServoCount = 0;
byte servoCount = 0;

boolean isResetting = false;

// Forward declare a few functions to avoid compiler errors with older
// versions
// of the Arduino IDE.
void setPinModeCallback(byte, int);
void reportAnalogCallback(byte analogPin, int value);
void sysexCallback(byte, byte*, byte*);

/* utility functions */
void wireWrite(byte data)
{
#if ARDUINO >= 100
    Wire.write((byte)data);
#else
    Wire.send(data);
#endif
}

byte wireRead(void)
{
#if ARDUINO >= 100
    return Wire.read();
#else
    return Wire.receive();
#endif
}

/*=====
=====
 * FUNCTIONS
=====
=====
=====*/
void attachServo(byte pin, int minPulse, int maxPulse)
{
    if (servoCount < MAX_SERVOS) {
        // reuse indexes of detached servos until all have been reallocated
```

```

if (detachedServoCount > 0) {
    servoPinMap[pin] = detachedServos[detachedServoCount - 1];
    if (detachedServoCount > 0) detachedServoCount--;
} else {
    servoPinMap[pin] = servoCount;
    servoCount++;
}
if (minPulse > 0 && maxPulse > 0) {
    servos[servoPinMap[pin]].attach(PIN_TO_DIGITAL(pin), minPulse,
maxPulse);
} else {
    servos[servoPinMap[pin]].attach(PIN_TO_DIGITAL(pin));
}
} else {
    Firmata.sendString("Max servos attached");
}
}

void detachServo(byte pin)
{
servos[servoPinMap[pin]].detach();
// if we're detaching the last servo, decrement the count
// otherwise store the index of the detached servo
if (servoPinMap[pin] == servoCount && servoCount > 0) {
    servoCount--;
} else if (servoCount > 0) {
    // keep track of detached servos because we want to reuse their
indexes
    // before incrementing the count of attached servos
    detachedServoCount++;
    detachedServos[detachedServoCount - 1] = servoPinMap[pin];
}
servoPinMap[pin] = 255;
}

void enableI2CPins()
{
byte i;
// is there a faster way to do this? would probaby require importing
// Arduino.h to get SCL and SDA pins
for (i = 0; i < TOTAL_PINS; i++) {
    if (IS_PIN_I2C(i)) {
        // mark pins as i2c so they are ignore in non i2c data requests
        setPinModeCallback(i, PIN_MODE_I2C);
    }
}
}

```

```

isI2CEnabled = true;

Wire.begin();
}

/* disable the i2c pins so they can be used for other functions */
void disableI2CPins() {
    isI2CEnabled = false;
    // disable read continuous mode for all devices
    queryIndex = -1;
}

void readAndReportData(byte address, int theRegister, byte numBytes, byte
stopTX) {
    // allow I2C requests that don't require a register read
    // for example, some devices using an interrupt pin to signify new data
available
    // do not always require the register read so upon interrupt you call
Wire.requestFrom()
    if (theRegister != I2C_REGISTER_NOT_SPECIFIED) {
        Wire.beginTransmission(address);
        wireWrite((byte)theRegister);
        Wire.endTransmission(stopTX); // default = true
        // do not set a value of 0
        if (i2cReadDelayTime > 0) {
            // delay is necessary for some devices such as WiiNunchuck
            delayMicroseconds(i2cReadDelayTime);
        }
    } else {
        theRegister = 0; // fill the register with a dummy value
    }

    Wire.requestFrom(address, numBytes); // all bytes are returned in
requestFrom

    // check to be sure correct number of bytes were returned by slave
    if (numBytes < Wire.available()) {
        Firmata.sendString("I2C: Too many bytes received");
    } else if (numBytes > Wire.available()) {
        Firmata.sendString("I2C: Too few bytes received");
    }

    i2cRxData[0] = address;
    i2cRxData[1] = theRegister;

    for (int i = 0; i < numBytes && Wire.available(); i++) {
        i2cRxData[2 + i] = wireRead();
    }
}

```

```

// send slave address, register and received bytes
Firmata.sendSysex(SYSEX_I2C_REPLY, numBytes + 2, i2cRxData);
}

void outputPort(byte portNumber, byte portValue, byte forceSend)
{
    // pins not configured as INPUT are cleared to zeros
    portValue = portValue & portConfigInputs[portNumber];
    // only send if the value is different than previously sent
    if (forceSend || previousPINs[portNumber] != portValue) {
        Firmata.sendDigitalPort(portNumber, portValue);
        previousPINs[portNumber] = portValue;
    }
}

/*
-----
-----
 * check all the active digital inputs for change of state, then add any
events
 * to the Serial output queue using Serial.print() */
void checkDigitalInputs(void)
{
    /* Using non-looping code allows constants to be given to readPort().
     * The compiler will apply substantial optimizations if the inputs
     * to readPort() are compile-time constants. */
    if (TOTAL_PORTS > 0 && reportPINs[0]) outputPort(0, readPort(0,
portConfigInputs[0]), false);
    if (TOTAL_PORTS > 1 && reportPINs[1]) outputPort(1, readPort(1,
portConfigInputs[1]), false);
    if (TOTAL_PORTS > 2 && reportPINs[2]) outputPort(2, readPort(2,
portConfigInputs[2]), false);
    if (TOTAL_PORTS > 3 && reportPINs[3]) outputPort(3, readPort(3,
portConfigInputs[3]), false);
    if (TOTAL_PORTS > 4 && reportPINs[4]) outputPort(4, readPort(4,
portConfigInputs[4]), false);
    if (TOTAL_PORTS > 5 && reportPINs[5]) outputPort(5, readPort(5,
portConfigInputs[5]), false);
    if (TOTAL_PORTS > 6 && reportPINs[6]) outputPort(6, readPort(6,
portConfigInputs[6]), false);
    if (TOTAL_PORTS > 7 && reportPINs[7]) outputPort(7, readPort(7,
portConfigInputs[7]), false);
    if (TOTAL_PORTS > 8 && reportPINs[8]) outputPort(8, readPort(8,
portConfigInputs[8]), false);
    if (TOTAL_PORTS > 9 && reportPINs[9]) outputPort(9, readPort(9,
portConfigInputs[9]), false);
    if (TOTAL_PORTS > 10 && reportPINs[10]) outputPort(10, readPort(10,
portConfigInputs[10]), false);
}

```

```

    if (TOTAL_PORTS > 11 && reportPINs[11]) outputPort(11, readPort(11,
portConfigInputs[11]), false);
    if (TOTAL_PORTS > 12 && reportPINs[12]) outputPort(12, readPort(12,
portConfigInputs[12]), false);
    if (TOTAL_PORTS > 13 && reportPINs[13]) outputPort(13, readPort(13,
portConfigInputs[13]), false);
    if (TOTAL_PORTS > 14 && reportPINs[14]) outputPort(14, readPort(14,
portConfigInputs[14]), false);
    if (TOTAL_PORTS > 15 && reportPINs[15]) outputPort(15, readPort(15,
portConfigInputs[15]), false);
}

// -----
-----
/* sets the pin mode to the correct state and sets the relevant bits in
the
 * two bit-arrays that track Digital I/O and PWM status
 */
void setPinModeCallback(byte pin, int mode)
{
    if (Firmata.getPinMode(pin) == PIN_MODE_IGNORE)
        return;

    if (Firmata.getPinMode(pin) == PIN_MODE_I2C && isI2CEnabled && mode != PIN_MODE_I2C) {
        // disable i2c so pins can be used for other functions
        // the following if statements should reconfigure the pins properly
        disableI2CPins();
    }
    if (IS_PIN_DIGITAL(pin) && mode != PIN_MODE_SERVO) {
        if (servoPinMap[pin] < MAX_SERVOS &&
servos[servoPinMap[pin]].attached()) {
            detachServo(pin);
        }
    }
    if (IS_PIN_ANALOG(pin)) {
        reportAnalogCallback(PIN_TO_ANALOG(pin), mode == PIN_MODE_ANALOG ? 1
: 0); // turn on/off reporting
    }
    if (IS_PIN_DIGITAL(pin)) {
        if (mode == INPUT || mode == PIN_MODE_PULLUP) {
            portConfigInputs[pin / 8] |= (1 << (pin & 7));
        } else {
            portConfigInputs[pin / 8] &= ~(1 << (pin & 7));
        }
    }
    Firmata.setPinState(pin, 0);
    switch (mode) {

```

```

case PIN_MODE_ANALOG:
    if (IS_PIN_ANALOG(pin)) {
        if (IS_PIN_DIGITAL(pin)) {
            pinMode(PIN_TO_DIGITAL(pin), INPUT);      // disable output
driver
#ifndef ARDUINO <= 100
            // deprecated since Arduino 1.0.1 - TODO: drop support in
Firmata 2.6
            digitalWrite(PIN_TO_DIGITAL(pin), LOW); // disable internal
pull-ups
#endif
        }
        Firmata.setPinMode(pin, PIN_MODE_ANALOG);
    }
    break;
case INPUT:
    if (IS_PIN_DIGITAL(pin)) {
        pinMode(PIN_TO_DIGITAL(pin), INPUT);      // disable output driver
#ifndef ARDUINO <= 100
        // deprecated since Arduino 1.0.1 - TODO: drop support in Firmata
2.6
        digitalWrite(PIN_TO_DIGITAL(pin), LOW); // disable internal pull-
ups
#endif
        Firmata.setPinMode(pin, INPUT);
    }
    break;
case PIN_MODE_PULLUP:
    if (IS_PIN_DIGITAL(pin)) {
        pinMode(PIN_TO_DIGITAL(pin), INPUT_PULLUP);
        Firmata.setPinMode(pin, PIN_MODE_PULLUP);
        Firmata.setPinState(pin, 1);
    }
    break;
case OUTPUT:
    if (IS_PIN_DIGITAL(pin)) {
        if (Firmata.getPinMode(pin) == PIN_MODE_PWM) {
            // Disable PWM if pin mode was previously set to PWM.
            digitalWrite(PIN_TO_DIGITAL(pin), LOW);
        }
        pinMode(PIN_TO_DIGITAL(pin), OUTPUT);
        Firmata.setPinMode(pin, OUTPUT);
    }
    break;
case PIN_MODE_PWM:
    if (IS_PIN_PWM(pin)) {
        pinMode(PIN_TO_PWM(pin), OUTPUT);
        analogWrite(PIN_TO_PWM(pin), 0);
    }
}

```

```

        Firmata.setPinMode(pin, PIN_MODE_PWM);
    }
    break;
case PIN_MODE_SERVO:
    if (IS_PIN_DIGITAL(pin)) {
        Firmata.setPinMode(pin, PIN_MODE_SERVO);
        if (servoPinMap[pin] == 255 ||
!servos[servoPinMap[pin]].attached()) {
            // pass -1 for min and max pulse values to use default values
set
            // by Servo library
            attachServo(pin, -1, -1);
        }
    }
    break;
case PIN_MODE_I2C:
    if (IS_PIN_I2C(pin)) {
        // mark the pin as i2c
        // the user must call I2C_CONFIG to enable I2C for a device
        Firmata.setPinMode(pin, PIN_MODE_I2C);
    }
    break;
case PIN_MODE_SERIAL:
#ifndef FIRMATA_SERIAL_FEATURE
    serialFeature.handlePinMode(pin, PIN_MODE_SERIAL);
#endif
default:
    Firmata.sendString("Unknown pin mode"); // TODO: put error msgs in
EEPROM
}
// TODO: save status to EEPROM here, if changed
}

/*
 * Sets the value of an individual pin. Useful if you want to set a pin
value but
 * are not tracking the digital port state.
 * Can only be used on pins configured as OUTPUT.
 * Cannot be used to enable pull-ups on Digital INPUT pins.
*/
void setPinValueCallback(byte pin, int value)
{
    if (pin < TOTAL_PINS && IS_PIN_DIGITAL(pin)) {
        if (Firmata.getPinMode(pin) == OUTPUT) {
            Firmata.setPinState(pin, value);
            digitalWrite(PIN_TO_DIGITAL(pin), value);
        }
    }
}

```

```

    }

}

void analogWriteCallback(byte pin, int value)
{
    if (pin < TOTAL_PINS) {
        switch (Firmata.getPinMode(pin)) {
            case PIN_MODE_SERVO:
                if (IS_PIN_DIGITAL(pin))
                    servos[servoPinMap[pin]].write(value);
                Firmata.setPinState(pin, value);
                break;
            case PIN_MODE_PWM:
                if (IS_PIN_PWM(pin))
                    analogWrite(PIN_TO_PWM(pin), value);
                Firmata.setPinState(pin, value);
                break;
        }
    }
}

void digitalWriteCallback(byte port, int value)
{
    byte pin, lastPin, pinValue, mask = 1, pinWriteMask = 0;

    if (port < TOTAL_PORTS) {
        // create a mask of the pins on this port that are writable.
        lastPin = port * 8 + 8;
        if (lastPin > TOTAL_PINS) lastPin = TOTAL_PINS;
        for (pin = port * 8; pin < lastPin; pin++) {
            // do not disturb non-digital pins (eg, Rx & Tx)
            if (IS_PIN_DIGITAL(pin)) {
                // do not touch pins in PWM, ANALOG, SERVO or other modes
                if (Firmata.getPinMode(pin) == OUTPUT || Firmata.getPinMode(pin)
== INPUT) {
                    pinValue = ((byte)value & mask) ? 1 : 0;
                    if (Firmata.getPinMode(pin) == OUTPUT) {
                        pinWriteMask |= mask;
                    } else if (Firmata.getPinMode(pin) == INPUT && pinValue == 1 &&
Firmata.getPinState(pin) != 1) {
                        // only handle INPUT here for backwards compatibility
#ifndef ARDUINO > 100
                        pinMode(pin, INPUT_PULLUP);
#endif
                    }
                    pinWriteMask |= mask;
                }
            }
        }
    }
}

```

```

        }
        Firmata.setPinState(pin, pinValue);
    }
}
mask = mask << 1;
}
writePort(port, (byte)value, pinWriteMask);
}

// -----
-----
/* sets bits in a bit array (int) to toggle the reporting of the
analogIns
*/
//void FirmataClass::setAnalogPinReporting(byte pin, byte state) {
//}
void reportAnalogCallback(byte analogPin, int value)
{
    if (analogPin < TOTAL_ANALOG_PINS) {
        if (value == 0) {
            analogInputsToReport = analogInputsToReport & ~ (1 << analogPin);
        } else {
            analogInputsToReport = analogInputsToReport | (1 << analogPin);
            // prevent during system reset or all analog pin values will be
reported
            // which may report noise for unconnected analog pins
        if (!isResetting) {
            // Send pin value immediately. This is helpful when connected via
            // ethernet, wi-fi or bluetooth so pin states can be known upon
            // reconnecting.
            Firmata.sendAnalog(analogPin, analogRead(analogPin));
        }
    }
}
// TODO: save status to EEPROM here, if changed
}

void reportDigitalCallback(byte port, int value)
{
    if (port < TOTAL_PORTS) {
        reportPINs[port] = (byte)value;
        // Send port value immediately. This is helpful when connected via
        // ethernet, wi-fi or bluetooth so pin states can be known upon
        // reconnecting.
        if (value) outputPort(port, readPort(port, portConfigInputs[port]),
true);
    }
}

```

```

    }

    // do not disable analog reporting on these 8 pins, to allow some
    // pins used for digital, others analog. Instead, allow both types
    // of reporting to be enabled, but check if the pin is configured
    // as analog when sampling the analog inputs. Likewise, while
    // scanning digital pins, portConfigInputs will mask off values from
    any
        // pins configured as analog
    }

/*=====
=====
 * SYSEX-BASED commands
=====
=====*/
void sysexCallback(byte command, byte argc, byte *argv)
{
    byte mode;
    byte stopTX;
    byte slaveAddress;
    byte data;
    int slaveRegister;
    unsigned int delayTime;

    switch (command) {
        case I2C_REQUEST:
            mode = argv[1] & I2C_READ_WRITE_MODE_MASK;
            if (argv[1] & I2C_10BIT_ADDRESS_MODE_MASK) {
                Firmata.sendString("10-bit addressing not supported");
                return;
            }
            else {
                slaveAddress = argv[0];
            }

            // need to invert the logic here since 0 will be default for client
            // libraries that have not updated to add support for restart tx
            if (argv[1] & I2C_END_TX_MASK) {
                stopTX = I2C_RESTART_TX;
            }
            else {
                stopTX = I2C_STOP_TX; // default
            }

            switch (mode) {
                case I2C_WRITE:

```

```

Wire.beginTransmission(slaveAddress);
for (byte i = 2; i < argc; i += 2) {
    data = argv[i] + (argv[i + 1] << 7);
    wireWrite(data);
}
Wire.endTransmission();
delayMicroseconds(70);
break;
case I2C_READ:
if (argc == 6) {
    // a slave register is specified
    slaveRegister = argv[2] + (argv[3] << 7);
    data = argv[4] + (argv[5] << 7); // bytes to read
}
else {
    // a slave register is NOT specified
    slaveRegister = I2C_REGISTER_NOT_SPECIFIED;
    data = argv[2] + (argv[3] << 7); // bytes to read
}
readAndReportData(slaveAddress, (int)slaveRegister, data,
stopTX);
break;
case I2C_READ_CONTINUOUSLY:
if ((queryIndex + 1) >= I2C_MAX_QUERIES) {
    // too many queries, just ignore
    Firmata.sendString("too many queries");
    break;
}
if (argc == 6) {
    // a slave register is specified
    slaveRegister = argv[2] + (argv[3] << 7);
    data = argv[4] + (argv[5] << 7); // bytes to read
}
else {
    // a slave register is NOT specified
    slaveRegister = (int)I2C_REGISTER_NOT_SPECIFIED;
    data = argv[2] + (argv[3] << 7); // bytes to read
}
queryIndex++;
query[queryIndex].addr = slaveAddress;
query[queryIndex].reg = slaveRegister;
query[queryIndex].bytes = data;
query[queryIndex].stopTX = stopTX;
break;
case I2C_STOP_READING:
byte queryIndexToSkip;
// if read continuous mode is enabled for only 1 i2c device,
disable

```

```

// read continuous reporting for that device
if (queryIndex <= 0) {
    queryIndex = -1;
} else {
    queryIndexToSkip = 0;
    // if read continuous mode is enabled for multiple devices,
    // determine which device to stop reading and remove it's
data from
    // the array, shifting other array data to fill the space
    for (byte i = 0; i < queryIndex + 1; i++) {
        if (query[i].addr == slaveAddress) {
            queryIndexToSkip = i;
            break;
        }
    }

    for (byte i = queryIndexToSkip; i < queryIndex + 1; i++) {
        if (i < I2C_MAX_QUERIES) {
            query[i].addr = query[i + 1].addr;
            query[i].reg = query[i + 1].reg;
            query[i].bytes = query[i + 1].bytes;
            query[i].stopTX = query[i + 1].stopTX;
        }
    }
    queryIndex--;
}
break;
default:
    break;
}
break;
case I2C_CONFIG:
    delayTime = (argv[0] + (argv[1] << 7));

    if (delayTime > 0) {
        i2cReadDelayTime = delayTime;
    }

    if (!isI2CEnabled) {
        enableI2CPins();
    }

    break;
case SERVO_CONFIG:
    if (argc > 4) {
        // these vars are here for clarity, they'll optimized away by the
compiler
        byte pin = argv[0];

```

```
int minPulse = argv[1] + (argv[2] << 7);
int maxPulse = argv[3] + (argv[4] << 7);

if (IS_PIN_DIGITAL(pin)) {
    if (servoPinMap[pin] < MAX_SERVOS &&
servos[servoPinMap[pin]].attached()) {
        detachServo(pin);
    }
    attachServo(pin, minPulse, maxPulse);
    setPinModeCallback(pin, PIN_MODE_SERVO);
}
break;

case SAMPLING_INTERVAL:
if (argc > 1) {
    samplingInterval = argv[0] + (argv[1] << 7);
    if (samplingInterval < MINIMUM_SAMPLING_INTERVAL) {
        samplingInterval = MINIMUM_SAMPLING_INTERVAL;
    }
} else {
    //Firmata.sendString("Not enough data");
}
break;

case EXTENDED_ANALOG:
if (argc > 1) {
    int val = argv[1];
    if (argc > 2) val |= (argv[2] << 7);
    if (argc > 3) val |= (argv[3] << 14);
    analogWriteCallback(argv[0], val);
}
break;

case CAPABILITY_QUERY:
Firmata.write(START_SYSEX);
Firmata.write(CAPABILITY_RESPONSE);
for (byte pin = 0; pin < TOTAL_PINS; pin++) {
    if (IS_PIN_DIGITAL(pin)) {
        Firmata.write((byte)INPUT);
        Firmata.write(1);
        Firmata.write((byte)PIN_MODE_PULLUP);
        Firmata.write(1);
        Firmata.write((byte)OUTPUT);
        Firmata.write(1);
    }
    if (IS_PIN_ANALOG(pin)) {
        Firmata.write(PIN_MODE_ANALOG);
        Firmata.write(10); // 10 = 10-bit resolution
    }
    if (IS_PIN_PWM(pin)) {
```

```

        Firmata.write(PIN_MODE_PWM);
        Firmata.write(DEFAULT_PWM_RESOLUTION);
    }
    if (IS_PIN_DIGITAL(pin)) {
        Firmata.write(PIN_MODE_SERVO);
        Firmata.write(14);
    }
    if (IS_PIN_I2C(pin)) {
        Firmata.write(PIN_MODE_I2C);
        Firmata.write(1); // TODO: could assign a number to map to SCL
or SDA
    }
#endif FIRMATA_SERIAL_FEATURE
    serialFeature.handleCapability(pin);
#endif
    Firmata.write(127);
}
Firmata.write(END_SYSEX);
break;
case PIN_STATE_QUERY:
    if (argc > 0) {
        byte pin = argv[0];
        Firmata.write(START_SYSEX);
        Firmata.write(PIN_STATE_RESPONSE);
        Firmata.write(pin);
        if (pin < TOTAL_PINS) {
            Firmata.write(Firmata.getPinMode(pin));
            Firmata.write((byte)Firmata.getPinState(pin) & 0x7F);
            if (Firmata.getPinState(pin) & 0xFF80)
                Firmata.write((byte)(Firmata.getPinState(pin) >> 7) & 0x7F);
            if (Firmata.getPinState(pin) & 0xC000)
                Firmata.write((byte)(Firmata.getPinState(pin) >> 14) & 0x7F);
        }
        Firmata.write(END_SYSEX);
    }
    break;
case ANALOG_MAPPING_QUERY:
    Firmata.write(START_SYSEX);
    Firmata.write(ANALOG_MAPPING_RESPONSE);
    for (byte pin = 0; pin < TOTAL_PINS; pin++) {
        Firmata.write(IS_PIN_ANALOG(pin) ? PIN_TO_ANALOG(pin) : 127);
    }
    Firmata.write(END_SYSEX);
    break;

    case SERIAL_MESSAGE:
#endif FIRMATA_SERIAL_FEATURE
    serialFeature.handleSysex(command, argc, argv);

```

```

#endif
        break;
    }
}

/*=====
=====
 * SETUP()

=====
=====*/
void systemResetCallback()
{
    isResetting = true;

    // initialize a defalt state
    // TODO: option to load config from EEPROM instead of default

#ifdef FIRMATA_SERIAL_FEATURE
    serialFeature.reset();
#endif

    if (isI2CEnabled) {
        disableI2CPins();
    }

    for (byte i = 0; i < TOTAL_PORTS; i++) {
        reportPINS[i] = false;      // by default, reporting off
        portConfigInputs[i] = 0;    // until activated
        previousPINS[i] = 0;
    }

    for (byte i = 0; i < TOTAL_PINS; i++) {
        // pins with analog capability default to analog input
        // otherwise, pins default to digital output
        if (IS_PIN_ANALOG(i)) {
            // turns off pullup, configures everything
            setPinModeCallback(i, PIN_MODE_ANALOG);
        } else if (IS_PIN_DIGITAL(i)) {
            // sets the output to 0, configures portConfigInputs
            setPinModeCallback(i, OUTPUT);
        }

        servoPinMap[i] = 255;
    }
    // by default, do not report any analog inputs
    analogInputsToReport = 0;
}

```

```
detachedServoCount = 0;
servoCount = 0;

/* send digital inputs to set the initial state on the host computer,
 * since once in the loop(), this firmware will only send on change */
/*
TODO: this can never execute, since no pins default to digital input
      but it will be needed when/if we support EEPROM stored config
for (byte i=0; i < TOTAL_PORTS; i++) {
    outputPort(i, readPort(i, portConfigInputs[i]), true);
}
*/
isResetting = false;
}

void setup()
{
    Firmata.setFirmwareVersion(FIRMATA_FIRMWARE_MAJOR_VERSION,
FIRMATA_FIRMWARE_MINOR_VERSION);

    Firmata.attach(ANALOG_MESSAGE, analogWriteCallback);
    Firmata.attach(DIGITAL_MESSAGE, digitalWriteCallback);
    Firmata.attach(REPORT_ANALOG, reportAnalogCallback);
    Firmata.attach(REPORT_DIGITAL, reportDigitalCallback);
    Firmata.attach(SET_PIN_MODE, setPinModeCallback);
    Firmata.attach(SET_DIGITAL_PIN_VALUE, setPinValueCallback);
    Firmata.attach(START_SYSEX, sysexCallback);
    Firmata.attach(SYSTEM_RESET, systemResetCallback);

    // to use a port other than Serial, such as Serial1 on an Arduino
Leonardo or Mega,
    // Call begin(baud) on the alternate serial port and pass it to Firmata
to begin like this:
    // Serial1.begin(57600);
    // Firmata.begin(Serial1);
    // However do not do this if you are using SERIAL_MESSAGE

    Firmata.begin(57600);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for ATmega32u4-based
boards and Arduino 101
    }

    systemResetCallback(); // reset to default config
}
```

```

/*=====
=====
 *  LOOP()
=====

=====
====*/
void loop()
{
    byte pin, analogPin;

    /* DIGITALREAD - as fast as possible, check for changes and output them
to the
     * FTDI buffer using Serial.print()  */
    checkDigitalInputs();

    /* STREAMREAD - processing incoming messagse as soon as possible, while
still
     * checking digital inputs.  */
    while (Firmata.available())
        Firmata.processInput();

    // TODO - ensure that Stream buffer doesn't go over 60 bytes

    currentMillis = millis();
    if (currentMillis - previousMillis > samplingInterval) {
        previousMillis += samplingInterval;
        /* ANALOGREAD - do all analogReads() at the configured sampling
interval */
        for (pin = 0; pin < TOTAL_PINS; pin++) {
            if (IS_PIN_ANALOG(pin) && Firmata.getPinMode(pin) ==
PIN_MODE_ANALOG) {
                analogPin = PIN_TO_ANALOG(pin);
                if (analogInputsToReport & (1 << analogPin)) {
                    Firmata.sendAnalog(analogPin, analogRead(analogPin));
                }
            }
        }
        // report i2c data for all device with read continuous mode enabled
        if (queryIndex > -1) {
            for (byte i = 0; i < queryIndex + 1; i++) {
                readAndReportData(query[i].addr, query[i].reg, query[i].bytes,
query[i].stopTX);
            }
        }
    }

#define FIRMATA_SERIAL_FEATURE
    serialFeature.update();

```

```
#endif
}
```

Código ConfigurableFirmata utilizado en el Arduino NANO

```
/*
Firmata is a generic protocol for communicating with microcontrollers
from software on a host computer. It is intended to work with
any host computer software package.

To download a host software package, please click on the following link
to open the download page in your default browser.

https://github.com/firmata/ConfigurableFirmata#firmata-client-libraries

Copyright (C) 2006-2008 Hans-Christoph Steiner. All rights reserved.
Copyright (C) 2010-2011 Paul Stoffregen. All rights reserved.
Copyright (C) 2009 Shigeru Kobayashi. All rights reserved.
Copyright (C) 2013 Norbert Truchs. All rights reserved.
Copyright (C) 2014 Nicolas Panel. All rights reserved.
Copyright (C) 2009-2017 Jeff Hoefs. All rights reserved.

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

See file LICENSE.txt for further informations on licensing terms.

Last updated: September 16th, 2017
*/
/*
README

This is an example use of ConfigurableFirmata. The easiest way to create
a configuration is to
use http://firmatabuilder.com and select the communication transport and
the firmata features
to include and an Arduino sketch (.ino) file will be generated and
downloaded automatically.

To manually configure a sketch, copy this file and follow the
instructions in the
ETHERNET CONFIGURATION OPTION (if you want to use Ethernet instead of
Serial/USB) and
```

```

FIRMATA FEATURE CONFIGURATION sections in this file.

*/
#include "ConfigurableFirmata.h"

/*=====
=====
 * ETHERNET CONFIGURATION OPTION
 *
 * By default Firmata uses the Serial-port (over USB) of the Arduino.
ConfigurableFirmata may also
 * communicate over ethernet using tcp/ip. To configure this sketch to use
Ethernet instead of
 * Serial, uncomment the appropriate includes for your particular
hardware. See STEPS 1 - 5 below.
 * If you want to use Serial (over USB) then skip ahead to the FIRMATA
FEATURE CONFIGURATION
 * section further down in this file.
*
 * If you enable Ethernet, you will need a Firmata client library with a
network transport that can
 * act as a server in order to establish a connection between
ConfigurableFirmataEthernet and the
 * Firmata host application (your application).
*
 * To use ConfigurableFirmata with Ethernet you will need to have one of
the following
 * boards or shields:
*
 * - Arduino Ethernet shield (or clone)
 * - Arduino Ethernet board (or clone)
 * - Arduino Yun
*
 * If you are using an Arduino Ethernet shield you cannot use the
following pins on
 * the following boards. Firmata will ignore any requests to use these
pins:
*
 * - Arduino Uno or other ATMega328 boards: (D4, D10, D11, D12, D13)
 * - Arduino Mega: (D4, D10, D50, D51, D52, D53)
 * - Arduino Leonardo: (D4, D10)
 * - Arduino Due: (D4, D10)
 * - Arduino Zero: (D4, D10)
*
 * If you are using an ArduinoEthernet board, the following pins cannot
be used (same as Uno):
 * - D4, D10, D11, D12, D13

```

```
*=====
====*/
// STEP 1 [REQUIRED]
// Uncomment / comment the appropriate set of includes for your hardware
//(OPTION A, B or C)

/*
 * OPTION A: Configure for Arduino Ethernet board or Arduino Ethernet
shield (or clone)
 *
 * To configure ConfigurableFirmata to use the an Arduino Ethernet Shield
or Arduino Ethernet
 * Board (both use the same WIZ5100-based Ethernet controller), uncomment
the SPI and Ethernet
 * includes below.
 */
//#include <SPI.h>
//#include <Ethernet.h>

/*
 * OPTION B: Configure for a board or shield using an ENC28J60 -based
Ethernet controller,
 * uncomment out the UIPEthernet include below.
 *
 * The UIPEthernet-library can be downloaded
 * from: https://github.com/ntruchsess/arduino_uip
 */
//#include <UIPEthernet.h>

/*
 * OPTION C: Configure for Arduino Yun
 *
 * The Ethernet port on the Arduino Yun board can be used with Firmata in
this configuration.
 * To execute StandardFirmataEthernet on Yun uncomment the Bridge and
YunClient includes below.
 *
 * NOTE: in order to compile for the Yun you will also need to comment
out some of the includes
 * and declarations in the FIRMATA FEATURE CONFIGURATION section later in
this file. Including all
 * features exceeds the RAM and Flash memory of the Yun. Comment out
anything you don't need.
 *
```

```

 * On Yun there's no need to configure local_ip and mac address as this
is automatically
 * configured on the linux-side of Yun.
 *
 * Establishing a connection with the Yun may take several seconds.
 */
//#include <Bridge.h>
//#include <YunClient.h>

#if defined ethernet_h || defined UIPETHERNET_H || defined _YUN_CLIENT_H_
#define NETWORK_FIRMATA

// STEP 2 [REQUIRED for all boards and shields]
// replace with IP of the server you want to connect to, comment out if
using 'remote_host'
#define remote_ip IPAddress(192, 168, 0, 1)
// OR replace with hostname of server you want to connect to, comment out
if using 'remote_ip'
// #define remote_host "server.local"

// STEP 3 [REQUIRED unless using Arduino Yun]
// Replace with the port that your server is listening on
#define remote_port 3030

// STEP 4 [REQUIRED unless using Arduino Yun OR if not using DHCP]
// Replace with your board or Ethernet shield's IP address
// Comment out if you want to use DHCP
#define local_ip IPAddress(192, 168, 0, 6)

// STEP 5 [REQUIRED unless using Arduino Yun]
// replace with Ethernet shield mac. Must be unique for your network
const byte mac[] = {0x90, 0xA2, 0xDA, 0x0D, 0x07, 0x02};
#endif

/*=====
=====
 * FIRMATA FEATURE CONFIGURATION
 *
 * Comment out the include and declaration for any features that you do
not need
 * below.
 *
 * WARNING: Including all of the following features (especially if also
using
 * Ethernet) may exceed the Flash and/or RAM of lower memory boards such
as the
 * Arduino Uno or Leonardo.

```

```
*=====
====*/
#include <DigitalInputFirmata.h>
DigitalInputFirmata digitalInput;

#include <DigitalOutputFirmata.h>
DigitalOutputFirmata digitalOutput;

#include <AnalogInputFirmata.h>
AnalogInputFirmata analogInput;

#include <AnalogOutputFirmata.h>
AnalogOutputFirmata analogOutput;

#include <Servo.h>
#include <ServoFirmata.h>
ServoFirmata servo;
// ServoFirmata depends on AnalogOutputFirmata
#ifndef ServoFirmata_h
#error AnalogOutputFirmata must be included to use ServoFirmata
#endif

#include <Wire.h>
#include <I2CFirmata.h>
I2CFirmata i2c;

#include <OneWireFirmata.h>
OneWireFirmata oneWire;

// StepperFirmata is deprecated as of ConfigurableFirmata v2.10.0. Please
// update your
// client implementation to use the new, more full featured and scalable
AccelStepperFirmata.
#include <StepperFirmata.h>
StepperFirmata stepper;

#include <AccelStepperFirmata.h>
AccelStepperFirmata accelStepper;

#include <SerialFirmata.h>
SerialFirmata serial;

#include <FirmataExt.h>
FirmataExt firmataExt;

#include <FirmataScheduler.h>
```

```

FirmataScheduler scheduler;

// To add Encoder support you must first install the FirmataEncoder and
Encoder libraries:
// https://github.com/firmata/FirmataEncoder
// https://www.pjrc.com/teensy/td_libs_Encoder.html
// #include <Encoder.h>
// #include <FirmataEncoder.h>
// FirmataEncoder encoder;

/*=====
=====
 * END FEATURE CONFIGURATION - you should not need to change anything
below this line

=====
=====*/
// dependencies. Do not comment out the following lines
#if defined AnalogOutputFirmata_h || defined ServoFirmata_h
#include <AnalogWrite.h>
#endif

#if defined AnalogInputFirmata_h || defined I2CFirmata_h || defined
FirmataEncoder_h
#include <FirmataReporting.h>
FirmataReporting reporting;
#endif

// dependencies for Network Firmata. Do not comment out.
#ifndef NETWORK_FIRMATA
#if defined remote_ip && defined remote_host
#error "cannot define both remote_ip and remote_host at the same time!"
#endif
#include <EthernetClientStream.h>
#ifdef _YUN_CLIENT_H_
YunClient client;
#else
EthernetClient client;
#endif
#if defined remote_ip && !defined remote_host
#ifdef local_ip
EthernetClientStream stream(client, local_ip, remote_ip, NULL,
remote_port);
#else
EthernetClientStream stream(client, IPAddress(0, 0, 0, 0), remote_ip,
NULL, remote_port);
#endif
#endif

```

```
#endif
#if !defined remote_ip && defined remote_host
#define local_ip
EthernetClientStream stream(client, local_ip, IPAddress(0, 0, 0, 0),
remote_host, remote_port);
#else
EthernetClientStream stream(client, IPAddress(0, 0, 0, 0), IPAddress(0,
0, 0, 0), remote_host, remote_port);
#endif
#endif
#endif

/*=====
=====
 * FUNCTIONS
=====
=====
====*/
void systemResetCallback()
{
    // initialize a default state

    // pins with analog capability default to analog input
    // otherwise, pins default to digital output
    for (byte i = 0; i < TOTAL_PINS; i++) {
        if (IS_PIN_ANALOG(i)) {
#ifndef AnalogInputFirmata_h
            // turns off pull-up, configures everything
            Firmata.setPinMode(i, PIN_MODE_ANALOG);
#endif
        } else if (IS_PIN_DIGITAL(i)) {
#ifndef DigitalOutputFirmata_h
            // sets the output to 0, configures portConfigInputs
            Firmata.setPinMode(i, OUTPUT);
#endif
        }
    }

#ifndef FirmataExt_h
    firmataExt.reset();
#endif
}

/*=====
=====
 * SETUP()
=====
```

```
*=====
====*/
void setup()
{
    /*
     * ETHERNET SETUP
     */
#ifndef NETWORK_FIRMATA
#ifndef _YUN_CLIENT_H_
    Bridge.begin();
#else
#ifndef local_ip
    Ethernet.begin((uint8_t *)mac, local_ip); //start Ethernet
#else
    Ethernet.begin((uint8_t *)mac); //start Ethernet using dhcp
#endif
#endif
    delay(1000);
#endif

    /*
     * FIRMATA SETUP
     */
    Firmata.setFirmwareVersion(FIRMATA_FIRMWARE_MAJOR_VERSION,
FIRMATA_FIRMWARE_MINOR_VERSION);

#ifndef FirmataExt_h
#ifndef DigitalInputFirmata_h
    firmataExt.addFeature(digitalInput);
#endif
#ifndef DigitalOutputFirmata_h
    firmataExt.addFeature(digitalOutput);
#endif
#ifndef AnalogInputFirmata_h
    firmataExt.addFeature(analogInput);
#endif
#ifndef AnalogOutputFirmata_h
    firmataExt.addFeature(analogOutput);
#endif
#ifndef ServoFirmata_h
    firmataExt.addFeature(servo);
#endif
#ifndef I2CFirmata_h
    firmataExt.addFeature(i2c);
#endif
#ifndef OneWireFirmata_h
```

```

        firmataExt.addFeature(oneWire);
#endif
#ifndef StepperFirmata_h
    firmataExt.addFeature(stepper);
#endif
#ifndef AccelStepperFirmata_h
    firmataExt.addFeature(accelStepper);
#endif
#ifndef SerialFirmata_h
    firmataExt.addFeature(serial);
#endif
#ifndef FirmataReporting_h
    firmataExt.addFeature(reporting);
#endif
#ifndef FirmataScheduler_h
    firmataExt.addFeature(scheduler);
#endif
#ifndef FirmataEncoder_h
    firmataExt.addFeature(encoder);
#endif
#endif
/* systemResetCallback is declared here (in ConfigurableFirmata.ino) */
Firmata.attach(SYSTEM_RESET, systemResetCallback);

// Network Firmata communicates with Ethernet-shields over SPI.
Therefor all
// SPI-pins must be set to PIN_MODE_IGNORE. Otherwise Firmata would
break SPI-communication.
// add Pin 10 and configure pin 53 as output if using a MEGA with
Ethernetshield.
// No need to ignore pin 10 on MEGA with ENC28J60, as here pin 53
should be connected to SS:
#endif NETWORK_FIRMATA

#ifndef _YUN_CLIENT_H_
// ignore SPI and pin 4 that is SS for SD-Card on Ethernet-shield
for (byte i = 0; i < TOTAL_PINS; i++) {
    if (IS_PIN_SPI(i)
        || 4 == i // SD Card on Ethernet shield uses pin 4 for SS
        || 10 == i // Ethernet-shield uses pin 10 for SS
    ) {
        Firmata.setPinMode(i, PIN_MODE_IGNORE);
    }
}
// pinMode(PIN_TO_DIGITAL(53), OUTPUT); configure hardware-SS as
output on MEGA
pinMode(PIN_TO_DIGITAL(4), OUTPUT); // switch off SD-card bypassing
Firmata

```

```

digitalWrite(PIN_TO_DIGITAL(4), HIGH); // SS is active low;
#endif

#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
  pinMode(PIN_TO_DIGITAL(53), OUTPUT); // configure hardware SS as
output on MEGA
#endif

// start up Network Firmata:
Firmata.begin(stream);
#else
  // Uncomment to save a couple of seconds by disabling the startup blink
sequence.
  // Firmata.disableBlinkVersion();

// start up the default Firmata using Serial interface:
Firmata.begin(57600);
#endif
  systemResetCallback(); // reset to default config
}

/*=====
=====
 * LOOP()

=====
=====
void loop()
{
#endif DigitalInputFirmata_h
  /* DIGITALREAD - as fast as possible, check for changes and output them
to the
   * stream buffer using Firmata.write() */
  digitalInput.report();
#endif

  /* STREAMREAD - processing incoming message as soon as possible, while
still
   * checking digital inputs. */
  while (Firmata.available()) {
    Firmata.processInput();
#endif FirmataScheduler_h
    if (!Firmata.isParsingMessage()) {
      goto runtasks;
    }
  }
  if (!Firmata.isParsingMessage()) {
runtasks: scheduler.runTasks();
}
}

```

```
#endif
}

/* SEND STREAM WRITE BUFFER - TO DO: make sure that the stream buffer
doesn't go over
 * 60 bytes. use a timer to sending an event character every 4 ms to
 * trigger the buffer to dump. */

#ifndef FirmataReporting_h
if (reporting.elapsed()) {
#endif
#ifndef AnalogInputFirmata_h
    /* ANALOGREAD - do all analogReads() at the configured sampling
interval */
    analogInput.report();
#endif
#ifndef I2CFirmata_h
    // report i2c data for all device with read continuous mode enabled
    i2c.report();
#endif
#ifndef FirmataEncoder_h
    // report encoders positions if reporting enabled.
    encoder.report();
#endif
#ifndef StepperFirmata_h
    stepper.update();
#endif
#ifndef AccelStepperFirmata_h
    accelStepper.update();
#endif
#ifndef SerialFirmata_h
    serial.update();
#endif

#if defined NETWORK_FIRMATA && !defined local_ip &&!defined
_YUN_CLIENT_H_
    // only necessary when using DHCP, ensures local IP is updated
appropriately if it changes
    if (Ethernet.maintain()) {
        stream.maintain(Ethernet.localIP());
    }
#endif
}
```

Código Servidor Node (server.js)

```

const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const http = require('http');
const app = express();

// API file for interacting with MongoDB
const api = require('./server/routes/api');

// Parsers
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false}));

// Angular DIST output folder
app.use(express.static(path.join(__dirname, 'dist')));

// CORS
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With,
Content-Type, Accept");
    next();
});

// API location
app.use('/api', api.rutas);
app.use('/api', api.placas);

// Enviar todo lo otro a Angular
app.get('*', (req, res) => {
    if (process.env.AMBIENTE == 'DESARROLLO'){
        //Estoy levantando angular en puerto 4200
        res.status(404).send('Estas en desarrollo');
    }else{
        res.sendFile(path.join(__dirname, 'dist/index.html'));
    }
});

//Set Port
const port = process.env.PORT || '3000';
app.set('port', port);

const server = http.createServer(app);

server.listen(port, () => console.log(`Running on localhost:${port}`));

```

Código API Express (api.js)

```

const express = require('express');
const router = express.Router();
const MongoClient = require('mongodb').MongoClient;
const ObjectId = require('mongodb').ObjectId;

const MINIMODISTANCIA = 20;
/*Esto es para apagar*/
const control = require('./apagar');
console.log(control.saludar());


if (process.env.AMBIENTE == 'DESARROLLO') {
    console.log('Iniciado desarrollo');
    var hola = require('./hola');
    module.exports = hola;
} else {
    console.log('Iniciado Test');
    var placas = require('./placas');
    module.exports.placas = placas;
}

var cont = 0;

// Connect
const connection = (closure) => {
    return MongoClient.connect('mongodb://localhost:27017/sar', (err, db)
=> {
        if (err) return console.log(err);
        closure(db);
    });
};

// Error handling
const sendError = (err, res) => {
    response.status = 501;
    response.message = typeof err == 'object' ? err.message : err;
    res.status(501).json(response);
};

// Response handling
let response = {
    status: 200,
    data: [],
    message: null
};

```

```
router.get('/temperaturas', (req, res) => {
    var lista = [];
    connection((db) => {
        db.collection('temperaturas')
            .distinct("fecha").then((fechas) => {

                fechas.forEach(f => {
                    db.collection("temperaturas")
                        .find({ "fecha": f }, { "valor": 1, "hora": 1,
                            "_id": 0 }).sort({ "fecha": 1, "hora": 1 }).batchSize(30000)
                        .toArray(function (err, result) {
                            if (err) throw err;
                            lista.push({ series: result, fecha: f });

                            if (lista.length === fechas.length) {

                                console.log('Mostrando ----*****');
                                console.log(lista[5].series.length);
                                console.log(lista[5].fecha);
                                res.json(lista);
                                db.close();
                            }
                        });
                });
            });
        })
        .catch((err) => {
            sendError(err, res);
        })
    });
});

router.get('/monoxidos', (req, res) => {
    connection((db) => {
        db.collection('mq7')
            .find()
            .toArray()
            .then((valores) => {
                response = valores;
                res.json(response);
                db.close();
            })
            .catch((err) => {
                sendError(err, res);
            })
    });
});
```

```
        });
    });
});

router.get('/monoxidosActual', (req, res) => {
    var ahora = new Date();
    var despues = new Date();
    despues.setSeconds(ahora.getSeconds() + 5);

    hora1 = ahora.getHours() + ':' + ahora.getMinutes() + ':' +
ahora.getSeconds();
    hora2 = despues.getHours() + ':' + despues.getMinutes() + ':' +
despues.getSeconds();
    // console.log('Buscando con '+hora1+ ' y '+hora2);

    connection((db) => {
        db.collection('monoxidos')
            .findOne({ "hora": { $gte: hora1, $lte: hora2 } }, { "valor": 1, "hora": 1, "_id": 0 }, {
                function (err, result) {
                    if (err) throw err;
                    // console.log(result);
                    res.json(result);
                    db.close();
                });
    });
}

/*
Se exportan las variables para que sean conocidas por Nodejs
*/
router.get('/apagar', (req, res) => {
    console.log('Llego solicitud de apagado...');
    control.apagar();
});

router.get('/reiniciar', (req, res) => {
    console.log('Llego solicitud de reinicio...');
    control.reiniciar();
});

module.exports.rutas = router;
```

Código Manejo de Arduino Mega y Arduino Nano (placas.js)

```

const express = require('express');
const router = express.Router();
const MongoClient = require('mongodb').MongoClient;
const ObjectId = require('mongodb').ObjectId;
const MINIMODISTANCIA = 20;

const connection = (closure) => {
    return MongoClient.connect('mongodb://localhost:27017/sar', (err, db)
=> {
        if (err) return console.log(err);
        closure(db);
    });
};

var ports = [{ id: "mega", port: "/dev/ttyACM0" },
{ id: "nano", port: "/dev/ttyUSB0" }
];

var five = require("johnny-five");

new five.Bords(ports).on("ready", function () {
    //Se enciende la placa
    var motor1;
    var motor2;
    var motor3;
    var motor4;

    // configuro el sensor de monoxido
    //Se enciende la placa
    var sensor = new five.Sensor({
        pin: "A0",
        board: this.byId("mega"),
        freq: 1000 //Frecuencia en msg

    });
    //Configuro el sensor de temperatura
    var thermometer = new five.Thermometer({
        controller: "DS18B20",
        pin: 2,
        board: this.byId("nano")
    });

    var gps = new five.GPS({
        pins: {
            rx: 15,
            tx: 14,
        }
    });
}
);

```

```

        board: this.byId("mega")
    }
});

// si latitud o longitud cambian
gps.on("ready", function () {
    console.log("position");
    console.log(" latitude : ", this.latitude);
    console.log(" longitude : ", this.longitude);
    console.log(" altitude : ", this.altitude);

    connection((db) => {
        db.collection('gps')
            .insert({ "latitude": this.latitude, "longitude": this.longitude, "altitude": this.altitude, "fecha": new Date() })
        .then((muestrasGPS) => {
            console.log('Insertando muestra GPS');
        })
        .catch((err) => {
            console.log('Error al insertar GPS');
        });
    });
});
// If speed, course change log it
gps.on("navigation", function () {
    console.log("navigation");
    console.log(" speed : ", this.speed);
    console.log(" course : ", this.course);
    connection((db) => {
        db.collection('navegacion')
            .insert({ "velocidad": this.speed, "curso": this.course, "fecha": new Date() })
        .then((muestrasGPS) => {
            console.log('Insertando muestra navegacion');
        })
        .catch((err) => {
            console.log('Error al insertar GPS');
        });
    });
});
//Configuro el sensor de proximidad en el pin 22
var proximityAdelante = new five.Proximity({
    controller: "HCSR04",
    pin: 22,
    board: this.byId("mega")
});

```

```
var proximityDerecho = new five.Proximity({
  controller: "HCSR04",
  pin: 24,
  board: this.byId("mega")
});

var proximityIzquierdo = new five.Proximity({
  controller: "HCSR04",
  pin: 26,
  board: this.byId("mega")
});

var distanciaAdelante = 0;

motor1 = new five.Motor({
  pins: {
    pwm: 10,
    dir: 7,
    cdir: 6,
    board: this.byId("mega")
  }
});

motor2 = new five.Motor({
  pins: {
    pwm: 11,
    dir: 8,
    cdir: 9,
    board: this.byId("mega")
  }
});

motor3 = new five.Motor({
  pins: {
    pwm: 13,
    dir: 5,
    cdir: 4,
    board: this.byId("mega")
  }
});

motor4 = new five.Motor({
  pins: {
    pwm: 12,
    dir: 2,
    cdir: 3,
    board: this.byId("mega")
  }
});
```

```

        }

    });

    function stop() {
        motor1.stop();
        motor2.stop();
        motor3.stop();
        motor4.stop();
    }

    //Muestro la temperatura
    thermometer.on("change", function () {
        //console.log(this.celsius + "°C");
        //Agregado para generar hora
        var ahora = new Date();
        var hora = ahora.getHours() + ':' + ahora.getMinutes() + ':' +
        ahora.getSeconds();
        var fechaAlmacenar = ahora.getFullYear() + '/' +
        (ahora.getMonth() + 1) + '/' + ahora.getDate();

        connection((db) => {
            db.collection('temperaturas')
                .insert({ "valor": this.celsius, "fecha": fechaAlmacenar,
"hora": hora, "fechaSys": ahora })
                .then((temperaturas) => {
                    console.log('Insertando temperatura: ' +
this.celsius);
                    db.close();
                })
                .catch((err) => {
                    console.log('Error al insertar');
                });
        });
    });

    sensor.on("change", function (value) {
        connection((db) => {
            var ahora = new Date();
            hora = ahora.getHours() + ':' + ahora.getMinutes() + ':' +
            ahora.getSeconds();
            db.collection('monoxidos')
                .insert({ "valor": sensor.scaleTo([20, 2000]), "fecha": new Date(), "hora": hora })
                .then((sensorMQ7) => {
                    console.log('Insertando MQ7');
                    console.log(sensor.scaleTo([20, 2000]) + 'ppm'); //
float
                    db.close();
                });
        });
    });
}

```

```
        })
        .catch((err) => {
            console.log('Error al insertar valores de mq7');
        });
    });
}

// Si se generan modificaciones en la distancia de objetos, paran o avanzan los motores
proximityAdelante.on("change", function () {

    if (proximityAdelante.cm <= MINIMODISTANCIA) {

        stop();
    }
});

router.get('/arriba', (req, res) => {
    console.log('Accionando arriba');
    if (proximityAdelante.cm > MINIMODISTANCIA) {
        motor1.forward(255);
        motor2.forward(255);
        motor3.forward(255);
        motor4.forward(255);

        res.json("ok");
    } else {
        res.json("{ 'error': 'objeto adelante' }");
    }
});

router.get('/izquierda', (req, res) => {
    console.log('Accionando izquierda');
    res.json("ok");
    motor1.forward(255);
    motor2.reverse(255);
    motor3.forward(255);
    motor4.reverse(255);

});
}

router.get('/derecha', (req, res) => {
    console.log('Accionando derecha');
    motor1.reverse(255);
    motor2.forward(255);
});
```

```
motor3.reverse(255);
motor4.forward(255);

res.json("ok");
});

router.get('/abajo', (req, res) => {
  console.log('Accionando abajo');

  motor1.reverse(255);
  motor2.reverse(255);
  motor3.reverse(255);
  motor4.reverse(255);

  res.json("ok");

});

router.get('/stop', (req, res) => {
  console.log('deteniendo');
  motor1.stop();
  motor2.stop();
  motor3.stop();
  motor4.stop();
  res.json("ok");
});

router.get('/ultrasonido', (req, res) => {
  res.json([
    { ultrasonidoAdelante: proximityAdelante.cm },
    { ultrasonidoDerecho: proximityDerecho.cm },
    { ultrasonidoIzquierdo: proximityIzquierdo.cm }
  ]);
});

router.get('/gps', (req, res) => {
  res.json({
    latitud: gps.latitude,
    longitud: gps.longitude,
    altitud: gps.altitude,
    velocidad: gps.speed,
    sat: gps.sat,
    curso: gps.course,
    tiempo: gps.time
  });
});

router.get('/temperatura', (req, res) => {
  res.json({
    temperatura: thermometer.celsius, unidad: "celsius"
  });
});
```

```
});  
});  
  
router.get('/monoxido', (req, res) => {  
    res.json({ monoxido: sensor.scaleTo([20, 2000]), unidad: "ppm"  
});  
});  
  
});  
  
module.exports = router;
```

Código de Apagar y reiniciar (apagar.js)

```
// Require child_process
var exec = require('child_process').exec;

// Create shutdown function
var os = require('os');

var apagar = function shutdown() {
    if (os.type()[0] === 'W' || os.type()[0] === 'w') {
        console.log('Es windows');
    } else {
        console.log('Es linux');
    }
    console.log(os.type());
    exec('shutdown -h now', function (error, stdout, stderr) {
        console.log(stdout);
    });
}

var reiniciar = function reiniciar() {
    exec('shutdown -r now', function (error, stdout, stderr) {
        console.log(stdout);
    });
}

var saludar = function saludar(callback) {
    console.log('Hola desde apagar.js');
}

module.exports.reiniciar = reiniciar;
module.exports.apagar = apagar;
module.exports.saludar = saludar;
```

Códigos del lado del cliente

Código de Servicio Angular (servicio.ts)

```

import { Gps } from './Gps';
import { Injectable } from '@angular/core';
import { Http, Headers, RequestOptions } from '@angular/http';
import {HttpClient} from '@angular/common/http';
import 'rxjs/add/operator/map';
import { Observable } from 'rxjs/Observable';
import { AppComponent } from './app.component';
import { Temperatura } from './Temperatura';
import { Sensores } from './Sensores';
import { GraficaTemperatura } from './graficaTemperatura';
import { Monoxido } from './Monoxido';

@Injectable()
export class ServicioAplicacion {

    public sensores: Sensores;

    constructor(public http: Http, public http2: HttpClient, public app: AppComponent) {
        this.sensores = new Sensores();
    }

    /**
     * Evento que sirve para enviar desde el componente
     * @param url Url a enviar el evento [arriba, abajo, izquierda,
     derecha]
     */
    enviarEvento(accion: String): Observable <any> {
        return this.http.get(this.app.rutaBasica + accion);
    }

    solicitarTemperaturaActual(): Observable <Temperatura> {
        return this.http.get(this.app.rutaBasica + 'temperatura')
            .map(res => res.json());
    }

    /**
     * Devuelve ([{ultrasonidoAdelante:proximityAdelante.cm},
     * {ultrasonidoDerecho: proximityDerecho.cm},
     * {ultrasonidoIzquierdo: proximityIzquierdo.cm}
     * ])
     */
    solicitarUltrasonidosActual(): Observable <any> {
        return this.http.get(this.app.rutaBasica + 'ultrasonido')
            .map(res => res.json());
    }
}

```

```
/**devuelve
 * {monoxido: sensor.scaleTo([20,2000]), unidad: "ppm"}
 */
solicitarMonoxidoActual(): Observable <any> {
  return this.http.get(this.app.rutaBasica + 'monoxido')
  .map(res => res.json());
}

solicitarGpsActual(): Observable <Gps> {
  return this.http.get(this.app.rutaBasica + 'gps')
  .map(res => res.json());
}

pedirImagen(): Observable<boolean> {
  return this.http.get(this.app.rutaWeb)
  .map(res => res.json());
}

actualizarValores(muestraSensores: Sensores) {
  this.sensores = muestraSensores;
}

apagar() {
  return this.http.get(this.app.rutaBasica + 'apagar');
}

reiniciar() {
  return this.http.get(this.app.rutaBasica + 'reiniciar');
}

solicitarTodasTemperaturas(): Observable <GraficaTemperatura[]> {
  return this.http2.get<GraficaTemperatura[]>(this.app.rutaBasica +
'temperaturas');
}

solicitarMonoxidoActualBD(): Observable <Monoxido> {
  return this.http2.get<Monoxido>(this.app.rutaBasica +
'monoxidosActual');
}

}
```

Código de appComponent.ts Angular

```
import { Sensores } from './Sensores';
import { Component } from '@angular/core';
import { TemperaturaService } from './temperatura.service';
import { Temperatura } from './Temperatura';

import { DatePipe } from '@angular/common';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [TemperaturaService],
})
export class AppComponent {
  titulo = 'Sistema Autónomo Robótico';
  public temperaturas: Temperatura[];
  public sensoresActuales: Sensores;
  public tiempoDelay;
  public mensaje = '';

  public rutaBasica = 'http://192.168.2.1:3000/api/';
  // public rutaBasica = 'http://localhost:3000/api/';
  // public rutaBasica = 'http://192.168.1.39:3000/api/';
  public rutaWeb = 'http://192.168.2.1:8081';

  constructor(public service: TemperaturaService) {
    this.sensoresActuales = new Sensores();
    this.tiempoDelay = 1000;
  }
}
```

Código de AppModule (rutas) [Extracto]

```
const routes: Routes = [
  { path: '', component: PizarraComponent },
  { path: 'estadisticas', component: EstadisticasComponent },
  { path: 'control', component: PanelInferiorComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    CapturaVideoComponent,
    TablaSensoresComponent,
    TablaInfoComponent,
    PanelInferiorComponent,
    PanelControlComponent,
    PizarraComponent,
    EstadisticasComponent,
    EncabezadoComponent,
  ],
  imports: [
    BrowserModule,
    HttpModule,
    RouterModule.forRoot(routes),
    NgxChartsModule,
    BrowserAnimationsModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [TemperaturaService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Glosario

Ampere

Unidad de medida de la intensidad de corriente eléctrica.

AP

Access Point o punto de acceso en castellano, se le denomina a un dispositivo de red utilizado para la conexión de dispositivos inalámbricos a una red, por lo general **WIFI**.

API

Application Programming Interface o interfaz de programación de aplicaciones, en informática, se le llama a un conjunto de subrutinas, funciones y procedimientos utilizados para ofrecer una biblioteca a otro software como una capa de abstracción.

Back-End

Es la parte que procesa la entrada desde el **Front-End**. Esta parte se aloja principalmente del lado del servidor, programado en lenguajes como Java, PHP, .Net, Python, etc. Se encarga principalmente de generar un medio para proporcionar datos a la vista (**Front-End**) a través de la manipulación de datos.

Daemon

Se le denomina así a un proceso informático que se ejecuta en segundo plano, y por lo general al iniciar el sistema operativo.

Datos raw

O datos “crudos”, también conocidos como datos atomizados, se les denomina a los datos de dispositivos informáticos que no han sido procesados para ser utilizados.

DHCP

Dynamic Host Configuration Protocol o protocolo de configuración dinámica de **Host** en castellano, es un protocolo de red (que trabaja en la capa de aplicación del modelo TCP/IP) utilizado como servidor de direcciones **IP** dinámicas, para los **Host** conectados a la respectiva red.

DOM

Document object Model o modelo de objetos del documento en castellano, es una **API** para la representación de documentos **HTML**, XML y XHTML, que

facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido.

Framework

O entorno de trabajo, es un conjunto de herramientas informáticas que facilitan el desarrollo de software.

Front-End

Es la parte del software que interactúa con los usuarios. Por otro lado, dentro de las tecnologías webs, el Front-end son todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del **Navegador web**, generalizándose más que nada en tres lenguajes, **HTML**, CSS Y JavaScript.

Host

Se le llama así, a todos los dispositivos de una red encargados de brindar y/o utilizar servicios en la misma.

HTML

HyperText Markup Language o lenguaje de marcas de hipertexto en castellano, es un lenguaje utilizado para la codificación de páginas web.

HTTP

Hypertext Transfer Protocol o protocolo de transferencia de hipertexto en castellano, es un protocolo de comunicación para transferencia de información en la *World Wide Web*.

IDE

Integrated Development Environment o entorno de desarrollo integrado en castellano, es una aplicación informática, que, mediante diversos servicios integrados, se utiliza para facilitar el desarrollo de software.

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.

Internet

Se le llama así, al conjunto descentralizado de redes mundiales interconectadas entre sí, las cuales utilizan la familia de protocolos TCP/IP para realizar sus comunicaciones.

IoT

Internet of Things o internet de las cosas en castellano, es el concepto que hace referencia a la conexión digital de objetos de uso cotidiano, para las personas, con **Internet**.

IP

Internet Protocol o protocolo de **Internet** en castellano, es el protocolo encargado del direccionamiento y enrutamiento de datos en una red. Para ello se utiliza una dirección IP, que no es más que un número que identifica a todos los dispositivos conectados en una red.

LAN

Local Area Network o red de área local en castellano, es una red de computadoras que abarca un área reducida como una casa, un departamento o un edificio.

Lenguaje de programación

Es como se les denomina a los lenguajes formales de informática, diseñados para realizar procesos que puedan llevar a cabo las computadoras y, por ende, se pueden utilizar para el desarrollo de software.

LESS

Es un lenguaje dinámico de hojas de estilo que puede ser compilado como Hojas de Estilo en Cascada (CSS) y ejecutarse del lado del cliente o servidor.

Linux

Se le llama así, al núcleo de sistema operativo basado en Unix y desarrollado por Linus Torvalds, el cual es de software libre y utilizado por un número considerable de sistemas operativos a los cuales se los denomina distribuciones Linux.

Marshaling

O serialización en castellano, es un proceso de codificación de un objeto, guardado en un medio de almacenamiento, para su transmisión a través de una conexión de red como una serie de bytes o en un formato legible por el humano (**HTML**, XML, entre otros).

Navegador web

O *browser*, es un software que permite el acceso a la Web. Funciona en la capa de aplicación del modelo de red TCP/IP.

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.

Open Source

O código abierto en castellano, es un modelo de desarrollo de software en el cual su base fundamental es permitir a sus usuarios el acceso al código fuente para la colaboración en la evolución de dicho software.

Query

En informática, se le llama así a una consulta realizada mediante un lenguaje de consultas para bases de datos.

Resolución de pantalla

Número de pixeles que pueden ser mostrados por la pantalla de un dispositivo electrónico.

Template

O plantilla en castellano, se le denomina a un medio utilizado como “guía” o modelo para diseñar, desarrollar o construir un esquema predefinido.

UART

Universally Asynchronous Receiver/Transmitter o receptor/transmisor asíncrono universal en castellano. Es una unidad que incorporan ciertos procesadores, encargada de realizar la conversión de los datos a una secuencia de bits y transmitirlos o recibirlas a una velocidad determinada.

WIFI

Tecnología inalámbrica que permite la interconexión de dispositivos electrónicos para conformar una red.

Bibliografía

- [1] Wikiepdia, «<https://es.wikipedia.org/wiki/Arduino>,» [En línea]. Available: <https://es.wikipedia.org/wiki/Arduino>. [Último acceso: Agosto 2017].
- [2] RIA. [En línea]. Available: <https://www.robotics.org/>. [Último acceso: 20 Septiembre 2017].
- [3] [En línea]. Available: <http://www.educaciontrespuntocero.com/noticias/raspberry-pi-educacion/34377.html>. [Último acceso: Septiembre 2017].
- [4] C. Angulo, «www.upc.edu,» 13 Enero 2017. [En línea]. Available: <http://www.upc.edu/latevaupc/usos-y-beneficios-robotica-las-aulas/>. [Último acceso: Septiembre 2017].
- [5] RobotGroup, «<http://robotgroup.com.ar/es/>,» [En línea]. Available: <http://robotgroup.com.ar/es/>. [Último acceso: Agosto 2017].
- [6] Wikipedia.org, «Wikipedia,» [En línea]. Available: <https://es.wikipedia.org/wiki/Arduino>. [Último acceso: 17 2 2018].
- [7] [En línea]. Available: <http://comoprogramarpic.blogspot.com.ar/2012/06/programando-un-atmel-mi-primer-programa.html>. [Último acceso: Septiembre 2017].
- [8] Arduino, «<https://www.arduino.cc/en/Main/Products>,» [En línea]. Available: <https://www.arduino.cc/en/Main/Products>. [Último acceso: Septiembre 2017].
- [9] Arduino, «<https://www.arduino.cc/en/aug/>,» [En línea]. Available: <https://www.arduino.cc/en/aug/>. [Último acceso: Septiembre 2017].
- [10] «<http://playground.arduino.cc/>,» [En línea]. Available: <http://playground.arduino.cc/>. [Último acceso: Septiembre 2017].
- [11] «<https://playground.arduino.cc/Es/Es>,» [En línea]. Available: <https://playground.arduino.cc/Es/Es>. [Último acceso: Septiembre 2017].
- [12] Arduino, «<https://www.arduino.cc/en/Reference/PortManipulation>,» [En línea]. Available: <https://www.arduino.cc/en/Reference/PortManipulation>. [Último acceso: Septiembre 2017].
- [13] Wikipedia, «https://es.wikipedia.org/wiki/Raspberry_Pi,» [En línea]. Available: https://es.wikipedia.org/wiki/Raspberry_Pi. [Último acceso: Septiembre 2017].
- [14] Raspberry Pi Foundation, «www.raspberrypi.org,» [En línea]. Available: <http://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/>. [Último acceso: Octubre 2017].
- [15] «www.developereconomics.com,» [En línea]. Available: <http://www.developereconomics.com/graphs/de11>. [Último acceso: Octubre 2017].
- [16] J. Pastor, «www.xatakamovil.com,» 12 Marzo 2014. [En línea]. Available: <http://www.xatakamovil.com/mercado/desarrollo-de-aplicaciones-moviles-i-asi-esta-el-mercado>. [Último acceso: Octubre 2017].
- [17] Wikipedia, «https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil,» [En línea]. Available: https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil. [Último acceso: Octubre 2017].
- [18] Wikipedia, «https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable,» [En línea]. Available: https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable. [Último acceso: Octubre 2017].

- [19] Wikipedia, «<https://es.wikipedia.org/wiki/PhoneGap>,» [En línea]. Available: <https://es.wikipedia.org/wiki/PhoneGap>. [Último acceso: Octubre 2017].
- [20] Wikipedia, «https://es.wikipedia.org/wiki/Apache_Cordova,» [En línea]. Available: https://es.wikipedia.org/wiki/Apache_Cordova. [Último acceso: Octubre 2017].
- [21] Wikipedia, «https://es.wikipedia.org/wiki/Android_Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Android_Studio. [Último acceso: Octubre 2017].
- [22] Google, «<http://appinventor.mit.edu/explore/ai2/windows.html>,» [En línea]. Available: <http://appinventor.mit.edu/explore/ai2/windows.html>. [Último acceso: Octubre 2017].
- [23] Apache Cordova, «<http://cordova.apache.org/>,» [En línea]. Available: <http://cordova.apache.org/>. [Último acceso: Octubre 2017].
- [24] «[www.campusmpv.es](https://www.campusmpv.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx),» [En línea]. Available: <https://www.campusmpv.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx>. [Último acceso: Noviembre 2017].
- [25] Wikipedia, «<https://en.wikipedia.org/wiki/Libuv>,» [En línea]. Available: <https://en.wikipedia.org/wiki/Libuv>. [Último acceso: Noviembre 2017].
- [26] github, «<https://github.com/firmata/arduino>,» [En línea]. Available: <https://github.com/firmata/arduino>. [Último acceso: Noviembre 2017].
- [27] «<https://programarfacil.com/>,» [En línea]. Available: <https://programarfacil.com/podcast/arduino-day-protocolo-de-comunicaciones-firmata/>. [Último acceso: Noviembre 2017].
- [28] Arduino, «<https://www.arduino.cc/en/Reference/Firmata>,» [En línea]. Available: <https://www.arduino.cc/en/Reference/Firmata>. [Último acceso: Noviembre 2017].
- [29] R. P. Foundation, «[www.raspberrypi.org](https://www.raspberrypi.org/downloads/raspbian/),» [En línea]. Available: <https://www.raspberrypi.org/downloads/raspbian/>. [Último acceso: Noviembre 2017].
- [30] «[geekytheory.com](https://geekytheory.com/tutorial-raspberry-pi-como-crear-un-punto-de-acceso-wifi),» [En línea]. Available: <https://geekytheory.com/tutorial-raspberry-pi-como-crear-un-punto-de-acceso-wifi>.
- [31] Wikipedia, «https://es.wikipedia.org/wiki/Apache_Cordova,» [En línea]. Available: https://es.wikipedia.org/wiki/Apache_Cordova. [Último acceso: Octubre 2017].