

**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 2017**

Índice

[Capítulo 1 - Introducción 6](#_Toc508729644)

[**1.1 Objetivo general** 6](#_Toc508729645)

[1.1.1 Objetivos específicos 6](#_Toc508729646)

[1.1.2 Metodología 6](#_Toc508729647)

[**1.2 Motivación** 7](#_Toc508729648)

[**1.3 Desarrollos Propuestos** 8](#_Toc508729649)

[**1.4 Resultados Esperados** 8](#_Toc508729650)

[Capítulo 2 - La robótica 9](#_Toc508729651)

[**2.1 ¿Qué es la robótica?** 9](#_Toc508729652)

[**2.2 Estructura física de los robots** 11](#_Toc508729653)

[2.2.1 Poliarticulados 11](#_Toc508729654)

[2.2.2 Móviles 11](#_Toc508729655)

[2.2.3 Androides 12](#_Toc508729656)

[2.2.4 Zoomórficos 12](#_Toc508729657)

[2.2.5 Híbridos 12](#_Toc508729658)

[**2.3 Distintas tecnologías para la robótica educativa** 13](#_Toc508729659)

[**2.4 Microcontroladores y computadora de placa reducida (SBC)** 13](#_Toc508729660)

[**2.5. Comunicación entre distintas arquitecturas de cómputo** 15](#_Toc508729661)

[2.5.1 Formas de comunicación 15](#_Toc508729662)

[2.5.2 Tipos de Medios de transmisión 15](#_Toc508729663)

[**2.6 ¿Qué es un SAR (Sistema Autónomo Robótico)?** 16](#_Toc508729664)

[**2.7 La robótica en la educación** 16](#_Toc508729665)

[**2.7 Diseño conceptual del SAR** 18](#_Toc508729666)

[**Resumen** 19](#_Toc508729667)

[Capítulo 3 – Arduino 20](#_Toc508729668)

[**3.1 Arduino** 20](#_Toc508729669)

[**3.2 Historia** 20](#_Toc508729670)

[3.2.1 Wiring 21](#_Toc508729671)

[3.2.2 Processing 23](#_Toc508729672)

[3.2.3 Fritzing 24](#_Toc508729673)

[**3.3 Características generales de la plataforma** 24](#_Toc508729674)

[3.4 Distintas plataformas para Arduino 26](#_Toc508729675)

[**3.5 Aplicaciones** 28](#_Toc508729676)

[**3.6 Motivaciones para su uso** 28](#_Toc508729677)

[3.6.1 La comunidad 28](#_Toc508729678)

[3.6.2 Sencillez de programación 29](#_Toc508729679)

[3.6.3 Hardware económico 29](#_Toc508729680)

[**3.7 Incorporación de Arduino en las escuelas** 30](#_Toc508729681)

[3.7.1 Las tres erres 30](#_Toc508729682)

[**3.8 Actuadores y sensores** 31](#_Toc508729683)

[**3.9 Actuadores en el SAR** 31](#_Toc508729684)

[**3.10 Sensores en el SAR** 32](#_Toc508729685)

[**3.11 Módulos o *shields* en el SAR** 33](#_Toc508729686)

[**Resumen** 34](#_Toc508729687)

[Capítulo 4 – Raspberry Pi 35](#_Toc508729688)

[**4.1 Raspberry Pi** 35](#_Toc508729689)

[**4.2 Especificaciones técnicas de las distintas versiones** 35](#_Toc508729690)

[En la siguiente tabla se puede observar la evolución de las diversas versiones de Raspberry Pi, más populares, a lo largo del tiempo. [12] 35](#_Toc508729691)

[**4.3 Entrada/Salida de propósito general (GPIO)** 36](#_Toc508729692)

[**4.4 Sistemas Operativos compatibles** 38](#_Toc508729693)

[**4.5 Accesorios para Raspberry Pi** 39](#_Toc508729694)

[**4.6 Ventajas del uso de Raspberry Pi** 40](#_Toc508729695)

[**Resumen** 42](#_Toc508729696)

[Anexo de casos de pruebas 43](#_Toc508729697)

[**Servomotor SG90** 43](#_Toc508729698)

[***Código sg90-01-funcionamiento*** 44](#_Toc508729699)

[**Pruebas en el sensor de Monóxido de Carbono** 45](#_Toc508729700)

[***Código MQ7-01-funcionamiento*** 46](#_Toc508729701)

[**Caso de prueba N 1 Módulo WIFI ESP8266 Velocidad** 47](#_Toc508729702)

[**Caso de prueba N 2 Módulo WIFI ESP8266 Velocidad** 49](#_Toc508729703)

[**Caso de prueba Módulo WIFI ESP8266 Velocidad y configuración AP** 51](#_Toc508729704)

[***Código comandosAT-configuracionWIfi.ino*** 53](#_Toc508729705)

[**Caso de prueba N 3 Módulo WIFI ESP8266 Velocidad** 54](#_Toc508729706)

[***Código pruebaVelocidad6-configuracionWifi*** 58](#_Toc508729707)

[**Caso de prueba Módulo GPS** 61](#_Toc508729708)

[***Código GPS-NEO6-01Conectividad*** 63](#_Toc508729709)

[**Caso de prueba Módulo microSD Card Adapter** 64](#_Toc508729710)

[***Código microSD-01-LeerEscribir*** 66](#_Toc508729711)

[**Caso de prueba Integración WIFI y Cámara** 68](#_Toc508729712)

[**Caso de prueba Cámara OV 7670** 70](#_Toc508729713)

[***Código OV7670*** 72](#_Toc508729714)

[**Caso de prueba Módulo Bluetooth HC05-01** 87](#_Toc508729715)

[***Comunicación Bluetooth.ino*** 89](#_Toc508729716)

[Glosario 90](#_Toc508729717)

[***Ampere*** 90](#_Toc508729718)

[***AP (Access Point)*** 90](#_Toc508729719)

[***API (Application Programming Interface)*** 90](#_Toc508729720)

[***Back-End*** 90](#_Toc508729721)

[***Open Source*** 90](#_Toc508729722)

[***Daemon*** 90](#_Toc508729723)

[***Datos raw*** 90](#_Toc508729724)

[***DHCP (Dynamic Host Configuration Protocol)*** 90](#_Toc508729725)

[***DOM (Document object Model)*** 91](#_Toc508729726)

[***Framework*** 91](#_Toc508729727)

[***Front-End*** 91](#_Toc508729728)

[***Host*** 91](#_Toc508729729)

[***IDE (Integrated Development Environment)*** 91](#_Toc508729730)

[***Inteligencia Artificial*** 91](#_Toc508729731)

[***Internet*** 91](#_Toc508729732)

[***Iot (Internet of Things)*** 91](#_Toc508729733)

[***IP (Internet Protocol)*** 92](#_Toc508729734)

[***LAN (Local Area Network)*** 92](#_Toc508729735)

[***Lenguaje de programación*** 92](#_Toc508729736)

[***Linux*** 92](#_Toc508729737)

[***Marshaling*** 92](#_Toc508729738)

[***Navegador web (browser)*** 92](#_Toc508729739)

[***Protoboard*** 92](#_Toc508729740)

[***Query*** 92](#_Toc508729741)

[***Resolución de pantalla*** 93](#_Toc508729742)

[***UART (universally asynchronous receiver/transmitter)*** 93](#_Toc508729743)

[***WIFI*** 93](#_Toc508729744)

[Bibliografía 94](#_Toc508729745)

[Ilustración 1 - Esquema básico de un robot 10](#_Toc508729885)

[Ilustración 2 - Ejemplo de robot poliarticulado 11](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729886)

[Ilustración 3 - Ejemplo de robot móvil 11](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729887)

[Ilustración 4 - Androide Asimo de Honda 12](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729888)

[Ilustración 5 - Robot Zoomórfico caminador 12](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729889)

[Ilustración 6 - Robot móvil-poliarticulado 12](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729890)

[Ilustración 7 - Arquitectura de un microcontrolador 14](#_Toc508729891)

[Ilustración 8 - Esquema conceptual orientado a servicios 18](#_Toc508729892)

[Ilustración 9 - Logo de Arduino 20](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729893)

[Ilustración 10 – Código de Blink en Wiring IDE 22](#_Toc508729894)

[Ilustración 11 - C++ Blink ejemplo 22](#_Toc508729895)

[Ilustración 12 - Logo de Processing 23](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729896)

[Ilustración 13 - Processing ejemplo 24](#_Toc508729897)

[Ilustración 14 - Entorno Fritzing 24](#_Toc508729898)

[Ilustración 15 - Ejemplo serie 25](#_Toc508729899)

[Ilustración 16 - Niveles de entrada a la plataforma Arduino 26](#_Toc508729900)

[Ilustración 17 - Arduino Uno 27](#_Toc508729901)

[Ilustración 18 - Logotipo comunidad open-source de Arduino 29](#_Toc508729902)

[Ilustración 19- Representación actuadores y sensores 31](#_Toc508729903)

[Ilustración 20 - Actuadores y sensores compatibles con Arduino 32](#_Toc508729904)

[Ilustración 21- Representación de sensores 33](#_Toc508729905)

[Ilustración 22 - Logo oficial de Raspberry Pi 35](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729906)

[Ilustración 23 - Raspberry Pi 2 y sus GPIOs 37](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729907)

[Ilustración 24 - Interfaces de Raspberry Pi 38](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729908)

[Ilustración 25 - Cámara Raspberry Pi V2 39](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729909)

[Ilustración 26 - Pantalla táctil de Raspberry Pi 39](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729910)

[Ilustración 27 - Adafruit Prototyping Pi 39](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729911)

[Ilustración 28 - Pidrive 40](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729912)

[Ilustración 29 - Pi TFT 40](file:///C:\Users\Agustin\Documents\Agustin\Universidad\Tesina\Proyecto%20de%20tesina\sar\Informe%20tesina\Borrador%20Final%20-%20Tesina%20Mansilla-Schlapp.docx.docx#_Toc508729913)

# Capítulo 1 - Introducción

## **1.1 Objetivo general**

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

### 1.1.1 Objetivos específicos

* 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 denominada cliente/servidor, donde el cliente es el dispositivo que ejecuta la aplicación y el servidor es el SAR.

## **1.2 Motivación**

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

Para esto se necesita incursionar en 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 (Phisical Computing).

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

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

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

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

Percepción:

* Sensores
* Tratamiento de información
* Procesamiento de información

Planificación:

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

Manipulación:

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

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

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

### 2.2.1 Poliarticulados

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

Ilustración - Ejemplo de robot poliarticulado

### 2.2.2 Móviles

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

Ilustración - Ejemplo de robot móvil

### 2.2.3 Androides



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

Ilustración - Androide Asimo de Honda

### 2.2.4 Zoomórficos

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

Ilustración - Robot Zoomórfico caminador

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

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

Ilustración - Robot móvil-poliarticulado

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

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

La gran ventaja de estas arquitecturas con respecto a las que se mencionan a continuación, es su gran soporte y compatibilidad, dada la amplia comunidad que las utiliza. [3]

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

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

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

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

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

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

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

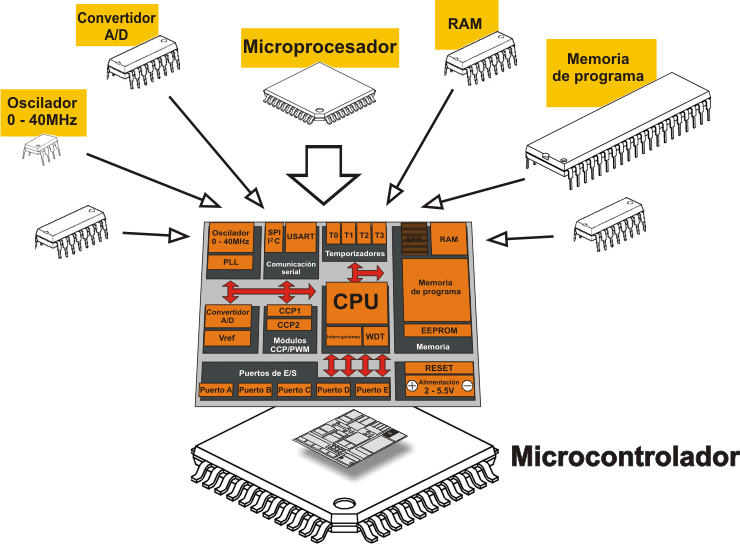


Ilustración - Arquitectura de un microcontrolador

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

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

## **2.5. Comunicación entre distintas 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 cumplan con los estándares IEEE 802.11 vigentes relacionados a redes inalámbricas de área local.

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

Se le considera SAR o sistema autónomo robótico a aquellos robots que presentan cierto grado de autonomía (**Inteligencia Artificial**) 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 Politécnica 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.

## **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 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.

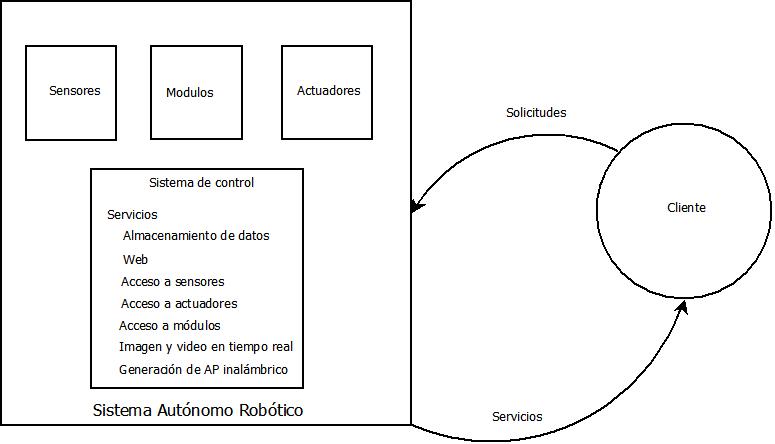


Ilustración - Esquema conceptual orientado a servicios

## **Resumen**

En este capítulo se abordó la definición de robot, definiéndose 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 robots generalmente cuentan 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 microcontroladores 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.

Ilustración - Logo de Arduino

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.

## **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 para los fines educativos 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. [5]

Un estudiante, Hernando Barragán, 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 (*Accesory 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.

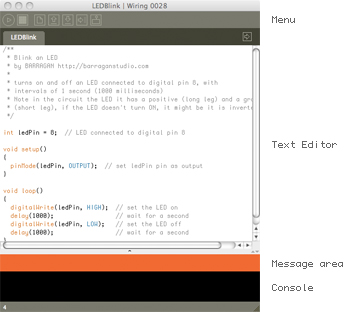


Ilustración – Código de Blink en Wiring IDE

Para ejemplificar 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**). [6]



Ilustración - 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.

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.

Ilustración - Logo de Processing

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);

}

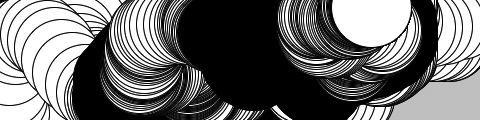


Ilustración - Processing ejemplo

### 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 a documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación

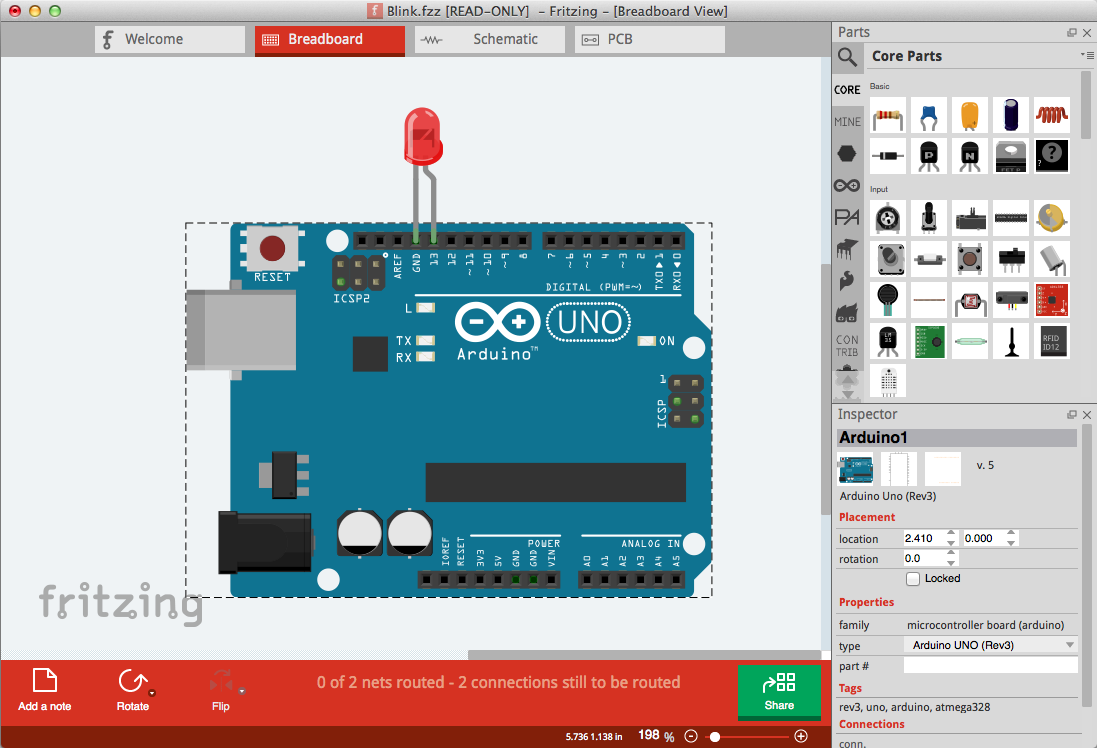


Ilustración - Entorno Fritzing

## **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 el 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 (universally asynchronous receiver/transmitter)***.

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) imprimiendo un contador. La zona marcada con rojo, es un botón que al presionarlo nos permite acceder a la terminal y ver el flujo serie seteando el clock correspondiente.

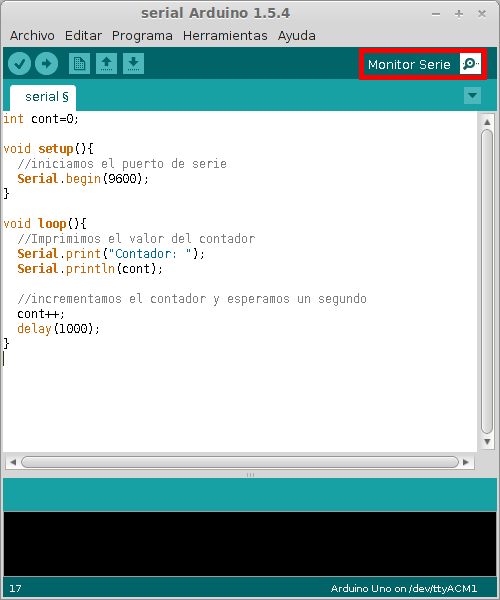


Ilustración - Ejemplo serie

3.4 Distintas plataformas para Arduino

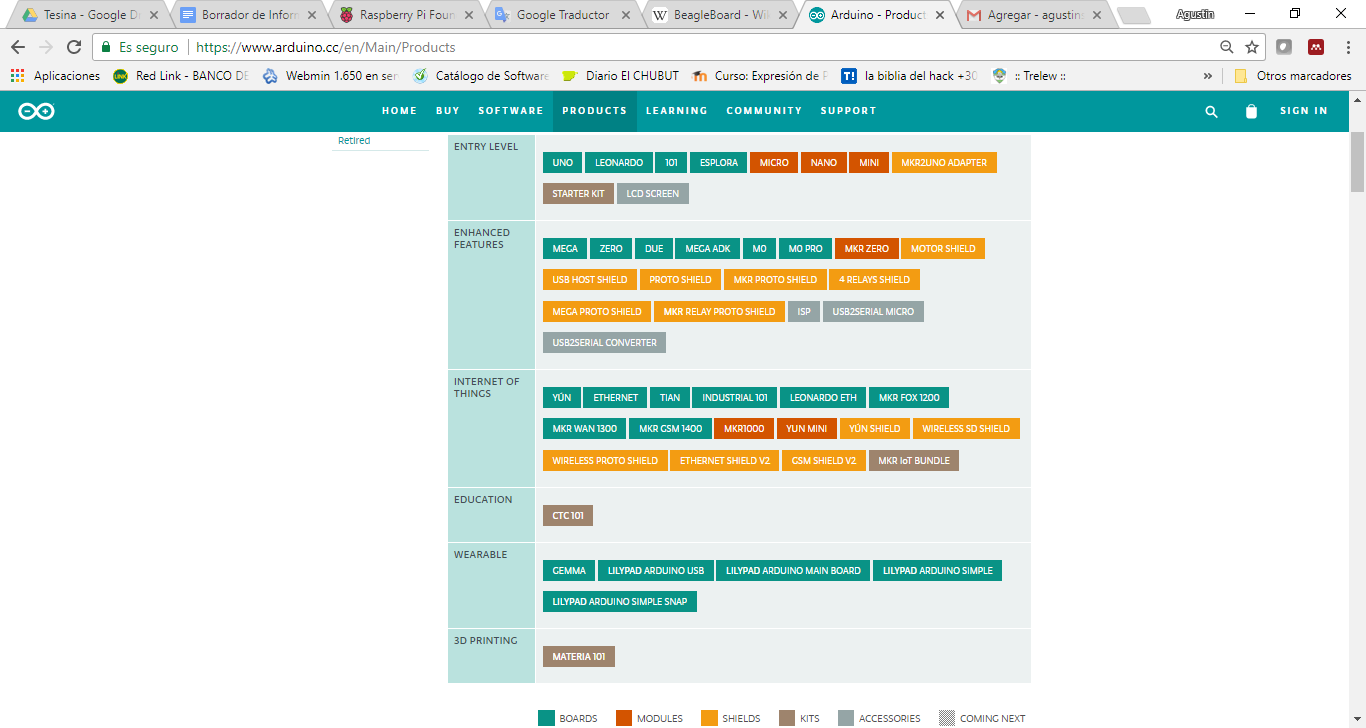


Ilustración - Niveles de entrada a la plataforma Arduino

Existe una gran variedad de productos Arduino, la compañía los cataloga, como se puede aprecias en (**Ilustración 16 - Niveles de entrada a la plataforma Arduino**), en distintos niveles según su utilidad [7]:

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

El hardware Arduino más sencillo consiste en una placa con un microcontrolador y una serie de puertos de entrada y salida. Los microcontroladores de 8 bits de AVR más utilizados en estas placas son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste, aunque también se dispone de microcontroladores ARM, cómo el caso del CortexM3 de 32 bits. A pesar de que ARM y AVR son plataformas diferentes, al utilizar la IDE de Arduino, los programas se compilan y luego se ejecutan sin cambios en cualquiera de las plataformas. En la imagen **(Ilustración 17 - Arduino Uno)** se visualiza la distribución física de puertos y componentes de la versión Arduino Uno R3.

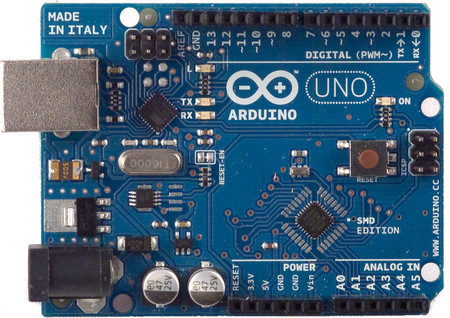


Ilustración - Arduino Uno

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 un ordenador y funcionamiento de forma independiente, recibiendo entradas de sensores y realizando acciones sobre actuadores en función de las entradas y el programa.
* Ídem anterior pero conectado a un ordenador (que también podría ser un 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” [8]

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. [9]

Este sitio a su vez tiene soporte de distintos lenguajes como el español. [10]

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 - 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; [11] // 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 debemos 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 encuentran la sencillez del lenguaje de programación que permite que alumnos y docentes no necesariamente del ámbito de la informática y la electrónica pueden utilizarlo, contribuyendo a la construcción colectiva del conocimiento, promoviendo la interdisciplinariedad escolar, permitiendo la colaboración de docentes de distintas áreas cooperar 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 desoldando 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 orden 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).

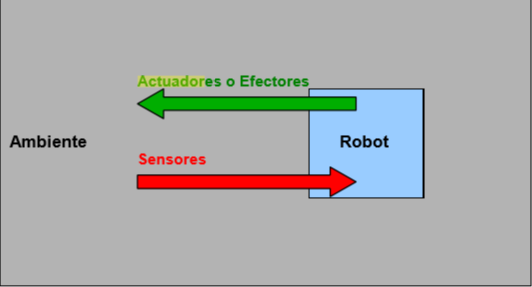


Ilustración - Representación actuadores y sensores

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 utilizarán:

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

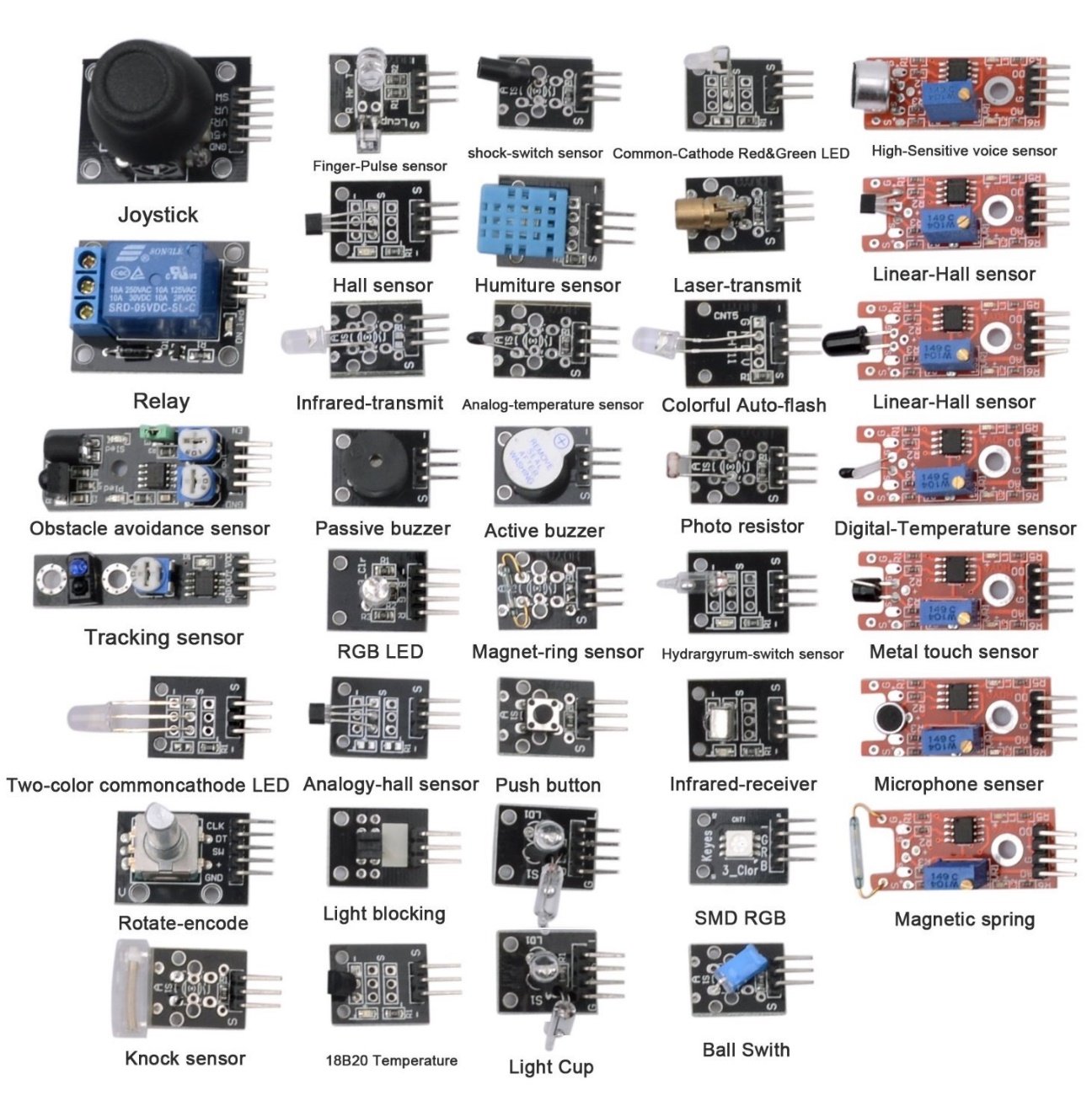


Ilustración - Actuadores y sensores compatibles con Arduino

## **3.10 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**).

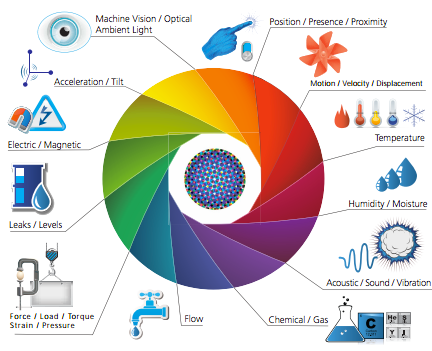


Ilustración - Representación de sensores

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 utiliza[[3]](#footnote-3):

* 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**)

## **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**

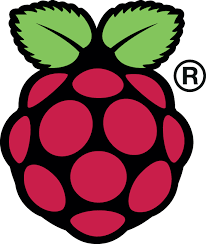
**Raspberry Pi** es un computador de placa reducida (SBC) desarrollado en Reino Unido por la Fundación Raspberry Pi. Su lanzamiento fue el 29 de febrero del 2012 con el *Raspberry Pi 1 Modelo A*. Su costo es relativamente bajo en relación a sus especificaciones técnicas (alrededor de U$D 25), dado que su 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.

Ilustración - Logo oficial de Raspberry Pi

## **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. [12]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Raspberry Pi 1 Modelo A** | | **Raspberry Pi 1 Modelo B** | | **Raspberry Pi 1 Modelo B+** | **Raspberry Pi 2 Modelo B** | **Raspberry Pi 3 Modelo B** |
| **SoC** | Broadcom BCM2835 ([CPU](https://es.wikipedia.org/wiki/CPU) + [GPU](https://es.wikipedia.org/wiki/GPU) + [DSP](https://es.wikipedia.org/wiki/Procesamiento_digital_de_se%C3%B1ales) + [SDRAM](https://es.wikipedia.org/wiki/SDRAM) + puerto USB) | | | | | Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + Puerto USB) | Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + Puerto USB |
| **CPU** | ARM 1176JZF-S a 700 MHz (familia ARM11) | | | | | 900 MHz quad-core ARM Cortex A7 | 1.2GHz 64-bit quad-core ARMv8 |
| **Juego de instrucciones** | RISC de 32 bits | | | | | | |
| **GPU** | Broadcom [VideoCore](https://es.wikipedia.org/w/index.php?title=VideoCore&action=edit&redlink=1) IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC | | | | | | |
| **Memoria SDRAM** | 256 MiB compartidos con la GPU | 512 MiB compartidos con la GPU, desde el 15 de octubre del 2012 | | | | 1 GB compartidos con la GPU | |
| **Puertos USB 2.0** | 1 | 2 | | 4 | | | |
| **Entradas de vídeo** | Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la Fundación Raspberry Pi | | | | | | |
| **Salidas de vídeo** | Conector RCA (PAL y NTSC), HDMI (rev 1.3 y 1.4), interfaz DSI para panel LCD | | | | | | |
| **Salidas de audio** | Conector de 3.5 mm, HDMI | | | | | | |
| **Almacenamiento integrado** | SD, MMC, ranura para SDIO | | | MicroSD | | | |
| **Conectividad de red** | Ninguna | 10/100 Ethernet (RJ45) via hub USB | | | | | 10/100 Ethernet (RJ45) vía hub USB, Wifi 802.11n, Bluetooth 4.1 |
| **Periféricos de bajo nivel** | 8 x GPIO, SPI, I2C, UART | | | | | 17 x GPIO y un bus HAT ID | |
| **Consumo energético** | 500 mA (2.5 W) | 700 mA (3.5 W) | | 600 mA (3.0 W) | | 800 mA (4.0 W) | |
| **Fuente de alimentación** | 5 V vía Micro USB o GPIO header | | | | | | |
| **Dimensiones** | 85.60mm × 53.98mm | | | | | | |
| **SO soportados** | GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, SUSE Linux Enterprise Server for ARM.  RISC OS | | | | | | |

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

Se le llama GPIO (En inglés, *General Purpose Input/Output*) a un conjunto de pines 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.

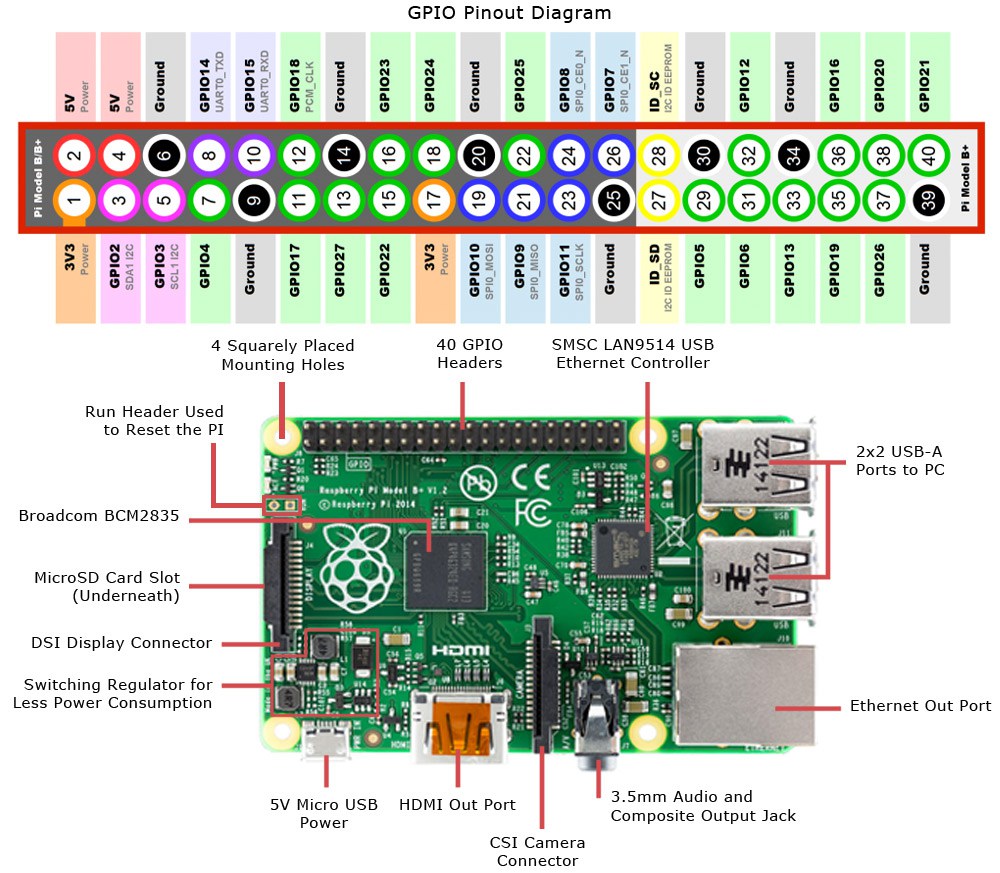
En la imagen (**Ilustración 23 - Raspberry Pi 2 y sus GPIOs**) se puede ver la Raspberry Pi 2 Modelo B de características bastante similares, en general, a la versión 3 de esta plataforma (utilizada en el desarrollo de esta tesina) y en detalle sus diversas interfaces. Un poco más arriba se pueden apreciar los distintos pines del tipo GPIO con los que cuenta esta plataforma (40 pines en total tanto la versión 2 como la 3). [13]

Ilustración - Raspberry Pi 2 y sus GPIOs

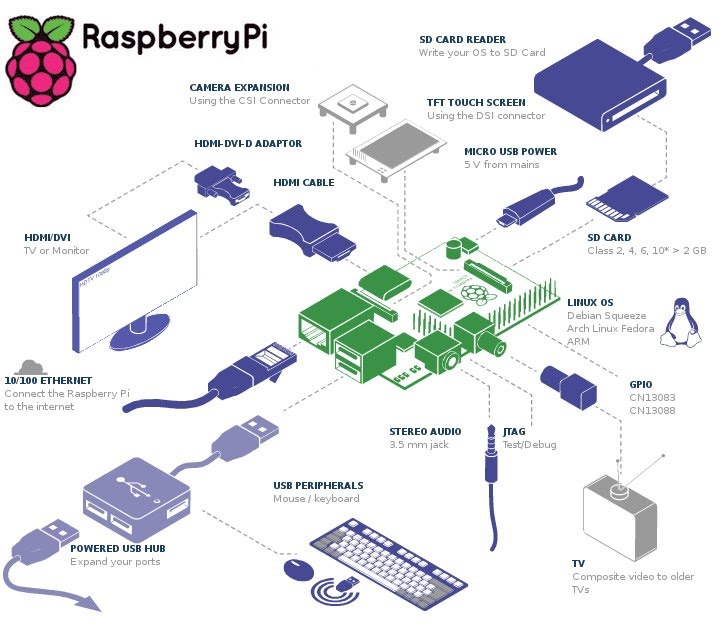
La siguiente imagen (**Ilustración 24 - Interfaces de Raspberry Pi**) ilustra los distintos periféricos que se pueden conectar a este computador.

Ilustración - Interfaces de Raspberry Pi

## **4.4 Sistemas Operativos compatibles**

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

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

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

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

## **4.5 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 carcasas para su resguardo.

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

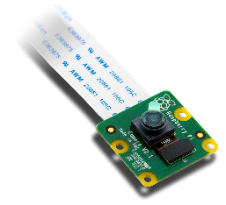
* *Cámara para Raspberry Pi V2*: Es una cámara de alta definición (HD) que se puede conectar a cualquier modelo de Raspberry para la captura de imágenes o videos en HD. Esta cámara posee un sensor de imagen IMX219PQ de Sony, el cual ofrece imágenes de video de alta velocidad y alta sensibilidad, además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

Ilustración - Cámara Raspberry Pi V2

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

Ilustración - Pantalla táctil de Raspberry Pi



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

Ilustración - Adafruit Prototyping Pi

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

Ilustración - Pidrive

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

Ilustración - Pi TFT

## **4.6 Ventajas del uso de Raspberry Pi**

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

* **Comunidad**: Existe una vasta comunidad en variadas partes del mundo que trabaja, da soporte y utiliza esta plataforma para diversos proyectos[[4]](#footnote-4), que dado esto, se expanden con él tiempo. A su vez, como se mostró en el apartado anterior, se cuenta con una serie de accesorios que facilitan su uso.
* **Bajo costo**: Como se mencionó con anterioridad, esta SBC se puede conseguir a un bajo costo teniendo en cuenta las prestaciones que posee.
* **Desarrollada con finalidad educativa**: Como ya se comentó anteriormente, según sus creadores, esta plataforma fue desarrollada con fines educativos y existe una comunidad que constantemente aporta lo necesario para trabajar con ella en el aula.
* **Interfaces y GPIO:** Cuenta con una variedad de interfaces para la conexión de distintos periféricos (HDMI, USB, Ethernet, Wifi, Bluetooth) y a su vez, los modelos más actuales (la versión 3), vienen con 40 pines del tipo GPIO, lo que lo convierte en un SBC muy versátil en cuanto a su utilidad.
* **Prestaciones**: Explicado todo lo anterior en este capítulo, podemos concluir que esta plataforma cumple con las prestaciones necesarias pretendidas en el desarrollo de esta tesina.

## **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 describen ventajas del uso de la Raspberry Pi, en relación con otras plataformas destinadas al mismo objetivo.

# Anexo de casos de pruebas

## **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 fabrica) |
| **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** |  |
| **Sketch** |  |

## ***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

{ // in steps of 1 degree

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** | Funcionamiento 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** |  |
| **Sketch** |  |

## ***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 10 fps 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 10 fps 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** |  |
| **Prueba 2 - Buffer 8 bytes**  **PacketSender recibe 81 paquetes**  **CIOBAUD=115200** |  |
| **Prueba 3 - Buffer 128 bytes**  **PacketSender**  **recibe 78 paquetes**  **CIOBAUD=115200** |  |
| **Prueba 4 - Buffer**  **256 bytes**  **PacketSender**  **110 paquetes}CIOBAUD=115200** |  |
| **Prueba 5**  **Buffer 512 bytes**  **PacketSender 107**  **CIOBAUD=115200** |  |
| **Prueba 6**  **Buffer 1024 bytes**  **PacketSender 48**  **ERROR**  **CIOBAUD=115200** |  |
| **Prueba 7**  **Buffer 512 bytes**  **PacketSender 6**  **CIOBAUD=9600** |  |
| **Prueba 8**  **Buffer 512 bytes**  **PacketSender 140**  **CIOBAUD=250000** |  |
| **Prueba 9**  **Buffer 512 bytes**  **CIOBAUD=500000** |  |
| **Prueba 10**  **Buffer 512**  **CIOBAUD= 4000000** |  |
| **Prueba 11**  **4608000 baudios Buffer 512 bytes** |  |
| **Prueba 12**  **4608000 baudios y 256 bytes buffer** |  |
| **Prueba 13**  **4608000 baudios y 768 bytes de buffer** |  |

## ***Código pruebaVelocidad6-configuracionWifi***

#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** |  |
| **Sketch** |  |

## ***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 imágen 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** |  |
| **Sketch** |  |

## ***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 0x0a /\* Product ID MSB \*/

#define REG\_VER 0x0b /\* Product ID LSB \*/

#define REG\_COM3 0x0c /\* 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 0x0d /\* Control 4 \*/

#define REG\_COM5 0x0e /\* All "reserved" \*/

#define REG\_COM6 0x0f /\* 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 \*/ //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 in0x58.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 0x0a /\* Product ID MSB \*/

#define REG\_VER 0x0b /\* Product ID LSB \*/

#define REG\_COM3 0x0c /\* 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 0x0d /\* Control 4 \*/

#define REG\_COM5 0x0e /\* All "reserved" \*/

#define REG\_COM6 0x0f /\* 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, 0x0a },

/\* More reserved magic, some of which tweaks white balance \*/

{ 0x43, 0x0a }, { 0x44, 0xf0 },

{ 0x45, 0x34 }, { 0x46, 0x58 },

{ 0x47, 0x28 }, { 0x48, 0x3a },

{ 0x59, 0x88 }, { 0x5a, 0x88 },

{ 0x5b, 0x44 }, { 0x5c, 0x67 },

{ 0x5d, 0x49 }, { 0x5e, 0x0e },

{ 0x6c, 0x0a }, { 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 << TWSTO);//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 << TWSTO);//send stop

\_delay\_ms(1);

twiStart();

twiAddr(camAddr\_RD, TW\_MR\_SLA\_ACK);

dat = twiRd(1);

TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//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 10 fps 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;

}

# 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.

## ***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.

## ***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 hosts conectados a la respectiva red.

## ***DOM (Document object Model)***

O modelo de objetos del documento, 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.

## ***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.

## ***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 (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.

## ***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.

## ***UART (universally asynchronous receiver/transmitter)***

O receptor/transmisor asíncrono universal. E*s una unidad que incorporan ciertos procesadores, encargada de realizar la conversión de los datos a una secuencia de bits y transmitirlos o recibirlos a una velocidad determinada.*

## ***WIFI***

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

# Bibliografía

|  |  |
| --- | --- |
| [1] | Wikiepdia, «es.wikipedia.org,» [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: https://www.upc.edu/latevaupc/usos-y-beneficios-robotica-las-aulas/. [Último acceso: Septiembre 2017]. |
| [5] | Wikipedia.org, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Arduino. [Último acceso: 17 2 2018]. |
| [6] | [En línea]. Available: http://comoprogramarpic.blogspot.com.ar/2012/06/programando-un-atmel-mi-primer-programa.html. [Último acceso: Septiembre 2017]. |
| [7] | Arduino, «https://www.arduino.cc/,» [En línea]. Available: https://www.arduino.cc/en/Main/Products. [Último acceso: Septiembre 2017]. |
| [8] | Arduino, «www.arduino.cc,» [En línea]. Available: https://www.arduino.cc/en/aug/ . [Último acceso: Septiembre 2017]. |
| [9] | «http://playground.arduino.cc/,» [En línea]. Available: http://playground.arduino.cc/. [Último acceso: Septiembre 2017]. |
| [10] | «https://playground.arduino.cc/Es/Es,» [En línea]. Available: https://playground.arduino.cc/Es/Es. [Último acceso: Septiembre 2017]. |
| [11] | Arduino, «https://www.arduino.cc/,» [En línea]. Available: https://www.arduino.cc/en/Reference/PortManipulation. [Último acceso: Septiembre 2017]. |
| [12] | «es.wikipedia.org,» [En línea]. Available: https://es.wikipedia.org/wiki/Raspberry\_Pi. [Último acceso: Septiembre 2017]. |
| [13] | Raspberry Pi Foundation, «www.raspberrypi.org,» [En línea]. Available: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/ . [Último acceso: Octubre 2017]. |

1. Agente basado en objetivos: “Almacena información del estado del mundo, así como del conjunto de objetivos que intenta alcanzar, y que es capaz de seleccionar la acción que eventualmente lo guiará hacia la consecución de sus objetivos” [Inteligencia Artificial un enfoque moderno. Person. Stuart Russell, Peter Norving 2da Ed. Pág. 57] [↑](#footnote-ref-1)
2. “Arduino nace como una solución para los diseñadores…”” Donde más se está potenciando es en la educación…” Matías Scovotti, director pedagógico y co-fundador de Educabot. <http://www.telam.com.ar/notas/201704/184406-robotica-arduino-day.html> [↑](#footnote-ref-2)
3. El uso de algunos de estos módulos queda en forma tentativa, dado que existen también en la Raspberry y su uso puede ser complementario. [↑](#footnote-ref-3)
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>). [↑](#footnote-ref-4)