CREACIÓN E IMPLEMENTACIÓN DE TALLERES ESCOLARES DE INICIACIÓN A LA ELECTRÓNICA, PROGRAMACIÓN EN SISTEMAS Y MICROCONTROLADORES CON ARDUINO EN LOS COLEGIOS DISTRITALES DEL CENTRO-HISTÓRICO DE BARRANQUILLA

Por:
CHERYL DALLÁN LAGOS GÓMEZ

Proyecto de Grado sometido en cumplimiento parcial de los requisitos para el grado de

INGENIERO DE SISTEMAS

Director: Ing. Paola Patricia Ariza Colpas

CORPORACIÓN UNIVERSITARIA DE LA COSTA CUC.
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
BARRANQUILLA- ATLÁNTICO-COLOMBIA.
2012

Nota de Aceptación:

_____

_____

_____

_____

_____

_____

_____

Fírma del Director

_____

Firma del Jurado

Barranquilla, 20 de Marzo de 2012

El autor dedica:

A Dios por estar conmigo en todo momento, porque día a día me ha dado fuerzas para seguir adelante.

A mis padres por enseñarme a soñar aun cuando el camino este lleno de piedras.

A mi hermana por ser el motor de mi vida.

A la memoria de Jafet Dix por haber sido mi gran amigo.

Cheryl Lagos.

AGRADECIMIENTOS

Primero y antes que nada, dar gracias a Dios, por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por haber puesto en nuestro camino a aquellas personas que han sido de soporte.

Agradezco hoy y siempre a mi familia por procurar mi bienestar.

Le doy gracias a la Ingeniera Paola Ariza Colpas, por haber sido mi guía durante este proceso, a quien debo su colaboración, paciencia y empeño, de igual forma mi agradecimiento al Ingeniero Andrés Sánchez, por la colaboración; y sobre todo agradecer por la gran amistad que ambos me brindaron.

En general les agradezco a mis amigos, y a todas aquellas personas que vivieron junto a mí la realización de este proyecto.

Sinceramente a todos les digo muchas gracias.

Cheryl Lagos.

TABLA DE CONTENIDO

pág.

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE FOTOGRAFÍAS

CUADROS DE CODIGO FUENTE

RESUMEN

La articulación con la educación media es uno de los aspectos más importantes, hoy por hoy, para diferentes instituciones de educación superior, sobre todo para realizar el proceso de captación del interés de los potenciales estudiantes en la vinculación de carreras profesionales de su afinidad.

Por lo tanto es meritorio que los estudiantes de los colegios distritales puedan tener contacto con la electrónica y los sistemas de información como ejes temáticos de vanguardia ocupacional. Este proyecto por lo tanto se encuentra enfocado a brindar la orientación necesaria para estos potenciales estudiantes en el proceso de elección de estos tópicos afines.

Se ha identificado la placa Arduino, como base fundamental para la enseñanza de la electrónica y los sistemas de información para los estudiantes de la media vocacional. Es, por lo tanto la principal finalidad de este proyecto de investigación la consecución de talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino en los colegios del Centro-Histórico de Barranquilla, como apoyo a la adquisición de estos conocimientos por parte de los estudiantes y acercamiento con estos en el proceso de elección de una carrera profesional a cursar.

ABSTRACT

The articulation with secondary education is one of the most important today, for various institutions of higher education, especially for the process of capturing the interest of potential students in linking the careers of their affinity.

Therefore it is commendable that students in district schools have contact with the electronics and information systems as leading occupational themes. This project is therefore focused on providing guidance to these potential students in the process of choosing these related topics.

It has identified the Arduino board, as the foundation for the teaching of electronics and information systems for vocational students in the average. It is therefore the main purpose of this research project the achievement of school workshops introduction to electronics, programming systems and microcontrollers with Arduino in schools of the Central Historical District of Barranquilla, to support the acquisition of this knowledge by students and approach these in the process of choosing a career to pursue.

1. INTRODUCCION

1.1 INTRODUCCIÓN

Arduino es una plataforma de desarrollo en computación física de código abierto, que consta de una placa con un sencillo microcontrolador y un entorno colaborativo para desarrollar software que controle esta placa. Arduino es un dispositivo que conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital[1].

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP)[2].

La placa arduino es un elemento que se ajusta a nuestro presupuesto, trabaja con software libre y por lo tanto se crean proyectos sin licencia, posee una alta documentación en el medio virtual para su uso, sus aplicaciones son muchas y es muy útil a la hora de aprender acerca de electrónica y programación rápida y sencilla.

[1] Tomado de http://www.fosforescente.cl/?p=590
[2] Tomado de http://www.arduino.cc/es/

David Cuartielles[3], ingeniero electrónico y docente de la Universidad de Mälmo, Suecia y Massimo Banzi[4], diseñador y desarrollador Web, son los creadores del proyecto Arduino.

Su idea surge de la necesidad de reducir el costo de implementar este tipo de plataformas, y hacerlo accesible tanto a estudiantes como a personas que quisieran experimentar en este tipo de ambientes[5].

"El hardware abierto significa tener la posibilidad de mirar lo que hay dentro de las cosas, que eso sea éticamente correcto, y que permita mejorar la educación. Educar en cómo funcionan las cosas…El hardware, aunque sea libre, no puede ser gratuito, es físico y cuesta dinero, lo que hicimos fue buscar el precio justo. Arduino no fabrica nada, diseña y mantiene un sitio web": **David Cuartielles**.

[3] Tomado de http://www.elpais.com/articulo/ocio/David/Cuartielles/creador/debe/convertirse/cientifico/elpportec/20060928elpciboci_1/Tes

[4] Tomado de http://www.massimobanzi.com/about/

[5] Tomado de http://www.fosforescente.cl/?p=590

1.2 ANTECEDENTES GENERALES

Software libre[6] es aquel software, producto o desarrollo a medida, que se distribuye bajo una licencia, según la cual el autor cede una serie de libertades básicas al usuario en el marco de un acuerdo de concesión. Estas libertades de los usuarios del software recogidas en la filosofía de la Fundación para el Software Libre (Free Software Fundation) son: La libertad de usar el programa con cualquier propósito; la libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades; la libertad de distribuir copias; y la libertad de mejorar el programa y hacer públicas las mejoras a los demás, de forma que toda la comunidad se beneficie.

En la época de los años 60 el software no era considerado un producto sino un añadido que los vendedores de los grandes computadores.
En los años 70 el software empezó a cobrar más relevancia y las compañías fabricantes de computadoras obligaron a los usuarios a aceptar software con licencias restrictivas, es decir, al que no se le podía realizar ninguna modificación.

Richard Stallman inicia un Proyecto GNU en 1983 con el objetivo de crear un sistema operativo completamente libre, y año más tarde, funda la Free Software Fundation.

Arduino nació en el Instituto Italiano de Diseño Interactivo Ivrea, una escuela donde los estudiantes centraban sus experimentos en la interacción con dispositivos, muchos de ellos basados en microcontroladores.

[6] Tomado de La producción de contenidos web -José Ángel Martínez Usero y Pablo Lara Navarra

Arduino surge de una necesidad, la de contar con un dispositivo para utilizar en clase, que fuera de bajo coste, que funcionase bajo cualquier sistema operativo y que contase con documentación adaptada a gente que quisiera empezar de cero.
En el año 2005, en Ivrea coincidieron todos los actores de esta historia. Así lo cuenta el profesor Massimo Banzi*: "Cuando estaba trabajando en esto conocí a* David Cuartelles *y comenzó a echarme una mano con el proyecto…Hicimos juntos el primer hardware de Arduino, luego vino* David Mellis*, un estudiante mío, que se unió para escribir el software, luego* Tom Igde *entró como consejero y Gianluca Martino que era el que producía las placas. Así se formó el equipo, añadiendo gente según sus habilidades".*

Arduino se implementó, no obstante, sobre los cimientos de Wiring. En Ivrea también daba clases Casey Reas, uno de los fundadores de la plataforma de programación Processing. Banzi pensó en cómo hacer un Processing para hardware. Comenzó, entonces, a trabajar con un estudiante suyo, que había hecho una tesis sobre el tema, Hernando Barragán. *"Después de que Hernando hiciera Wiring pensamos en cómo hacer toda la plataforma más simple, más barata y sencilla de usar. Se comenzó a reimplementar todo como un proyecto open source para que todo el mundo pudiera venir y ayudar, contribuir"*[7].

[7] Tomado de http://www.sorayapaniagua.com/2011/03/14/arduino-la-revolucion-silenciosa-del-hardware-libre/

1.3 DESCRIPCIÓN DEL PROBLEMA

En la actualidad las instituciones educativas no cuentan con un área de “computo” que posea herramientas y/o dispositivos tecnológicos actuales, entre ellos la placa Arduino; tampoco cuentan con docentes capacitados en estas nuevas plataformas tecnológicas para poder envolver a los estudiantes con los sistemas y la electrónica.

Según lo anteriormente dicho se ha notado que este proceso de comunicación entre docente - estudiante y tecnología debe ser apoyado en herramientas tecnológicas con el fin de ir a la vanguardia e involucrar a las personas en el mundo de la tecnología.

Por esta razón este trabajo de grado se muestra como una oportunidad de dar solución y de esta manera respaldar y apoyar a los docentes y estudiantes en temas de tecnología.

1.4 FORMULACIÓN DEL PROBLEMA

¿Cuál es la forma más fácil de poder capacitar a escolares en Electrónica, Sistemas y Microcontroladores?

1.5 OBJETIVOS

1.5.1. Objetivo General

Crear e implementar talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino en los colegios Distritales del Centro-Histórico de Barranquilla.

1.5.2. Objetivos Específicos

- ✓ Analizar el estado actual de la enseñanza de electrónica, programación en sistemas y microcontroladores en los colegios distritales del centro histórico de Barranquilla.
- ✓ Diseñar talleres amigables para la enseñanza de arduino en los colegios distritales del centro histórico de barranquilla.
- ✓ Diseñar cartillas de aprendizaje para la enseñanza de arduino en los colegios distritales del centro histórico de barranquilla.
- ✓ Implementar talleres de aprendizaje de arduino en los colegios distritales del centro histórico de barranquilla.

1.6 CONTRIBUCIONES DEL PROYECTO DE GRADO

Este proyecto busca forjar a estudiantes y aprendices a asumir una nueva perspectiva acerca de la electrónica, los sistemas y los microcontroladores con Arduino, a tal punto que ansíen su utilización, manejo y aplicación en la vida diaria.

1.7 ORGANIZACIÓN DEL PROYECTO DE GRADO

El siguiente proyecto se ha dividido en tres capítulos principales:

Capitulo 1: Este capítulo es la introducción y proporciona la información general del proyecto incluyendo los objetivos.

Capitulo 2: Este capítulo es el estado del arte.

Capitulo 3: Este capítulo es la solución y desarrollo de los talleres de Arduino.

2. ESTADO DEL ARTE

2.1 TECNOLOGIA EDUCATIVA

Cuando escuchamos la palabra tecnología enseguida nos llega a la mente todo lo que tenga que ver con un computador; no sabiendo que también se refiere a cualquier proceso a través del cual un ser humano diseña una herramienta o máquina para aumentar su control y su comprensión del entorno material. Las maquinas tecnológicas más sencillas que utilizamos a diario son: la televisión, la radio y el celular, pero hoy día también son avances tecnológicos el vídeo, la televisión interactiva, el Internet, videojuegos y el sistema de videoconferencias, entre otros.

El termino tecnología[8] proviene de las palabras griegas *tecné* que significa 'arte' u 'oficio', y *logos* que significa 'conocimiento' o 'ciencia'; por tanto, la tecnología es el conocimiento o ciencia de los oficios.

Cuando decimos que algo es educativo corresponde a tener la capacidad de transmitir conocimiento, ya sea la escuela o cualquier máquina.

Teniendo en cuenta lo anterior la tecnología educativa es una combinación de máquinas, electrónica, sistemas y métodos de enseñanza diseñados para satisfacer y/o entretener las necesidades de una sociedad, más que todo, la comunidad de jóvenes y niños que en este momento se considera cambiante.

Es muy importante que la tecnología este de la mano con la educación, para así poderla implementar con sensatez y se rescaten aportes positivos.

[8] Tomado de http://es.wikipedia.org/wiki/Tecnología

La tecnología educativa se debe constituir como una herramienta que busque fomentar las habilidades de los estudiantes, tanto así que la forma en que trabajen y piensen sea innovadora.

Algunas de las aplicaciones de la tecnología más utilizados por estudiante-docente son las video-conferencias, aulas virtuales, correos electrónicos, entre otras.

Uno de los sistemas creados por la tecnología que más impacto ha tenido sobre nuestra sociedad a nivel global es el Internet, este medio tecnológico permite la comunicación de personas entre diferentes lugares del mundo; entonces podemos decir que la tecnología educativa es importante porque avanza la comunicación.

Muchas personas están en “contra” de la tecnología cuando se trata de la educación de sus hijos, ya que está acabando con la búsqueda de un tema en un libro o porque la función del maestro no se ve cuando suben una tarea por el aula virtual o la envían por correo electrónico, por el contrario, sus funciones son aún más complejas, su destino ahora es enseñar a través de una herramienta tecnológica. Además, para lograr las metas educativas, siempre debe estar presente el toque humano.

Aunque también es cierto que los estudiantes no deben convertirse en esclavos de la tecnología, por el contrario ésta debe estar al servicio de ellos como una herramienta que pueda controlar para facilitar su estilo de vida.

Cada vez que hay una herramienta o máquina tecnológica nueva los docentes y estudiantes tendrán algo nuevo que aprender, no será fácil, pero no deben haber dudas de que lograran alcanzar el propósito.

Las posibilidades que brindan las nuevas tecnologías como herramienta didáctica, son de sin igual importancia y es necesario aprovechar todas sus potencialidades para formar seres humanos más justos, más capaces, más cooperativos.

Es determinante afirmar que lo importante no es la tecnología como tal, sino lo que las figuras formadoras, los docentes, y las figuras aprendices, los estudiantes, puedan hacer de la herramienta tecnológica, para transmitir a la humanidad algo positivo.

2.2 TRABAJOS REALIZADOS CON ARDUINO

Gracias al interés de algunas personas hacia placa arduino y el querer mostrar sus invenciones a la humanidad, en la web encontramos gran cantidad de trabajos que son interesantes y además son compartidos a tal punto que encontramos los tutoriales con las herramientas que utilizaron y cómo es el proceso para recrearlo.

Algunos de estos proyectos son:

Proyectos Realizados con Arduino	Autor del proyecto
Arpa laser	Stephen Hobley
Fotografías de alta velocidad	Sergio Vilches (España)
Mi robot, Makey	Kris Magri
Micrófono Alcoholímetro	OpenLab Eyebeam
Señales en una chaqueta de ciclismo	Leah Buechley

Tabla 1. Proyectos realizados con Arduino

2.2.1 Arpa Laser

Hay diversidad de arpas laser, la que se describe a continuación es llamada Arpa R simple[9] fue diseñada más recientemente a base de punteros laser.

[9] Tomado de http://makeprojects.com/Project/Laser-Harp/690/1

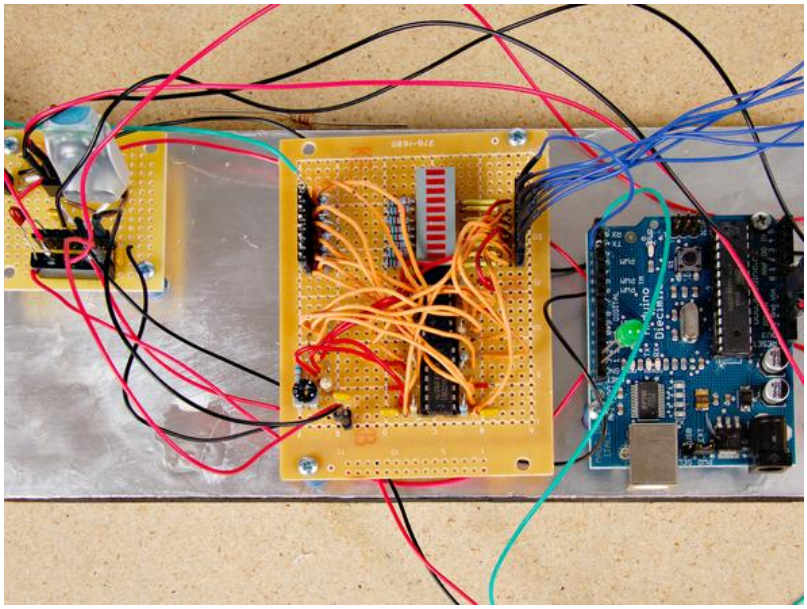


Figura 1. Partes del arpa laser[10]

El arpa trabaja para el controlador MIDI, por lo que no hace el sonido mismo, pero sí el flujo de datos MIDI para conducir un sintetizador de audio. Cada rayo golpea a una célula fotoeléctrica, y cuando interrumpe la mano del que toca, el sensor le pide un microcontrolador Arduino para enviar el MIDI "Note On" y dependiendo del rayo el sonido es diferente.

Figura 2. Arpa laser[11]

[10] Tomado de http://makeprojects.com/Project/Laser-Harp/690/1

[11] Tomado de http://makeprojects.com/Project/Laser-Harp/690/1

2.2.2 Fotografías de alta velocidad

Las fotografías de alta velocidad[12] es un proyecto que se logra usando un láser y un sensor para crear un cable de disparo electrónico, el cual utiliza el Arduino para capturar imágenes de alta velocidad. Como por ejemplo las salpicaduras de gotas de agua.

Figura 3. Cámara Antigua[13]

2.2.3 Mi robot, Makey

Makey[14] es un robot autónomo que ha sido programado para seguir objetos a su alrededor y evitar obstáculos, el control se hizo a partir del microcontrolador de arduino.

[12] Tomado de http://hacknmod.com/hack/high-speed-photography-how-to-trigger-using-arduino/
[13] Tomado de http://hacknmod.com/hack/high-speed-photography-how-to-trigger-using-arduino/
[14] http://makeprojects.com/Project/My-Robot-Makey/51/1

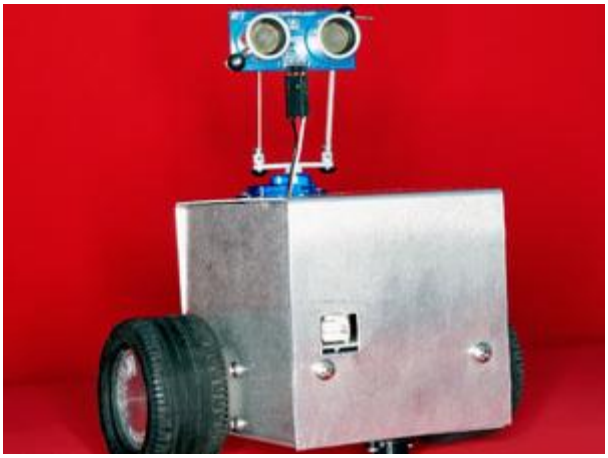


Figura 4. Makey el robot[15]

2.2.4 Micrófono Alcoholímetro

Este micrófono fue construido con el fin de conocer los niveles de alcohol en una persona.

Figura 5. Micrófono Alcoholímetro[16]

[15] Tomado de http://makeprojects.com/Project/My-Robot-Makey/51/1
[16] Tomado de http://www.instructables.com/id/Breathalyzer-Microphone/

Para realizar el micrófono alcoholímetro se necesitará: Un sensor de alcohol, un micrófono, una placa arduino, un lector y una tarjeta SD, un dispositivo de audio y herramientas de electrónica: resistencias, diodo, etc.

Figura 6. Materiales para realizar el micrófono[17]

2.2.5 Señales en una chaqueta de ciclismo

Las señales de esta chaqueta fueron construidas a base de arduino, funciona como direccionales, izquierda y derecha, para cuando una persona valla en su bicicleta las personas que vienen detrás sepan hacia donde se dirige.

Figura 7. Chaqueta direccional[18]

[17] Tomado de http://www.instructables.com/id/Breathalyzer-Microphone/
[18] Tomada de http://www.instructables.com/id/turn-signal-biking-jacket/

3. SOLUCION

3.1 DESARROLLO DE LOS TALLERES DE ARDUINO

Estos talleres se realizaron para aquellos que quieran iniciarse en arduino, son un instrumento de enseñanza de docentes para estudiantes, tomando en cuenta antes de todo los conceptos básicos de electrónica. Lo que se busca a través de ellos es innovar en diseños y desarrollos de proyectos.

Nota: Algunos de los talleres están basados en la fuente http://www.arduino.cc y la placa base con la que se trabajo Arduino UNO.

El orden en que se desarrollaron los talleres se muestra a continuación:

	Nombre del Taller
1	Hardware y software libre.
2	Símbolos y componentes básicos de electrónica.
3	Bit – Byte – Kilobyte.
4	Manejo de memoria.
5	Conociendo arduino (Introducción a microcontroladores).
6	Simulador arduino Virtual BreadBoard.
7	Instalación arduino.
8	Aprendiendo lenguaje de programación I: Estructura
9	Aprendiendo lenguaje de programación II: Funciones.
10	Aprendiendo lenguaje de programación III: Diccionario de palabras reservadas del IDE de Arduino.
11	Aprendiendo variables.
12	Apaga y prende un led N veces.
13	Semáforo
14	Manejando sensores
15	Sensor de inclinación - Alarma
16	Display de 7 segmentos
17	Toca tonos - Sensor fuerza
18	Practicas con arduino I parte
19	Practicas con arduino II parte
20	Practicas con arduino III parte

Tabla 2. Talleres de Arduino

3.1.1 Hardware y Software Libre

3.1.1.1 Hardware

Figura 8. Hardware[19]

El Hardware es denominado la parte física o tangible de un computador, incluyendo cualquier otro elemento físico involucrado.

3.1.1.2 Software

Figura 9. Software[20]

[19] Imagen tomada de pcexpertos.com
[20] Imagen tomada de bibliotecadeinvestigaciones.wordpress.com

El software se refiere a la parte intangible, aquella que no podemos tocar, como datos almacenados y programas que tal vez utilizamos a diario en un computador; Ejemplo: Utilizamos el programa Microsoft Office Word a la hora de redactar una carta, un memorando o una hoja de vida, etc.

3.1.1.3 Hardware libre

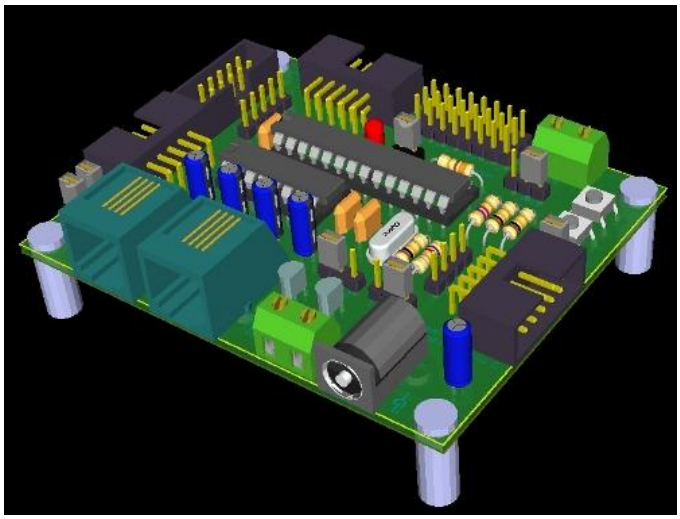


Figura 10. Hardware libre[21]

El Hardware libre es aquella parte física o dispositivo creado de forma abierta, con el objetivo principal de que todas las personas puedan acceder a su plano de construcción y a las partes con que fue construido, buscando así que en su utilización se puedan realizar cambios, corrección de errores y mejoras en su diseño. Un vivo ejemplo de Hardware libre es la placa *Arduino* de la cual hablaremos y conoceremos más adelante. Otro éxito ha sido *Ronja*(**R**easonable **O**ptical **N**ear **J**oint **A**ccess), Acceso Óptico Razonable de Nodo Cercano.

[21] Imagen tomada de iearobotics.com

Figura 11. Ronja[22]

Fue creado con el objetivo de crear enlaces de datos de transmisión óptica punto a punto inalámbrica (óptica de espacio libre) excelente para ser trabajado por inexpertos ya que las instrucciones de construcción están escritas por y para una persona sin experiencia, explicando las operaciones básicas de taladrado, soldadura, etc. Se muestran varias técnicas para reducir al mínimo los errores en lugares críticos y acelerar el trabajo - plantillas de taladrado, revisión de las soldaduras, procedimientos de prueba, etc. Es posible descargar los diseños de circuitos impresos listos para su construcción con todas las instrucciones.

3.1.1.4 Software libre

Figura 12. Software libre[23]

[22] Imagen tomada de blog.marques.cx
[23] Imagen tomada de tucamon.es

El software libre es aquel software al que se le conoce su código fuente, para que el usuario pueda emplearlo de la manera que más le parezca, además de poder modificarlo, distribuirlo y adaptarlo a sus necesidades.

Tal vez hemos escuchado el nombre de muchos de estos software´s tales como Linux, Mozilla Firefox, Ubuntu; pero sencillamente no sabíamos que eran libres.

Software´s libres que tienen mucha acogida han sido muchos como por ejemplo el navegador Mozilla Firefox, que es el segundo navegador de internet más utilizado por los usuarios.

Figura 13. Mozilla Firefox[24]

Este tiene características muy notables que hace que sea realmente exitoso en diversos aspectos, lo mejor de todo es que cada uno de nosotros podemos ser parte de estos proyectos como así también ser generadores y emprendedores de nuestros propios Proyectos de Software Libres.

[24] Imagen tomada de fondosypantallas.com

3.1.2 Símbolos y Componentes Básicos de Electrónica

Este taller se realizó con el fin de que cualquier persona se familiarice con el montaje de un circuito.

Interruptor

Figura 14. Interruptor[25]

Su símbolo es:

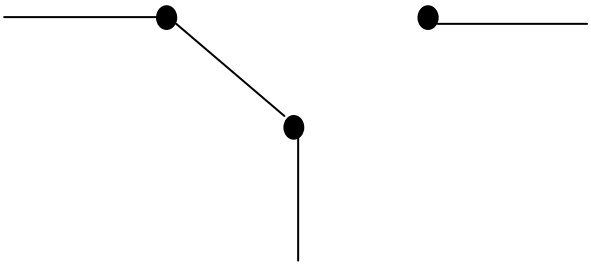


Figura 15. Símbolo del interruptor

Led

Figura 16. Leds

Su símbolo es:

Figura 17. Símbolo del led

[25] Imagen tomada de bricogeek.com

Diodo

Figura 18. Diodo[26]

Su símbolo es:

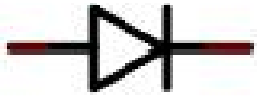


Figura 19. Símbolo del diodo

Resistencia

Figura 20. Resistencia

Su símbolo es:

Figura 21. Símbolo de la resistencia

Potenciómetro

Figura 22. Potenciómetro[27]

Su símbolo es:

Figura 23. Símbolo del potenciómetro

[26] Imagen de electronicamagnabit.com
[27] Imagen de es.wikipedia.org

Fotorresistencia

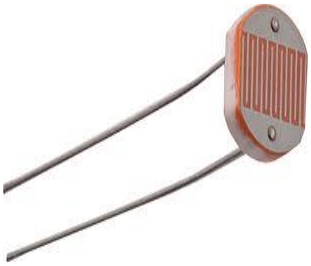


Figura 24 Fotorresistencia[28]

Su símbolo es:

Figura 25. Símbolo de la fotorresistencia

Capacitor de cerámica

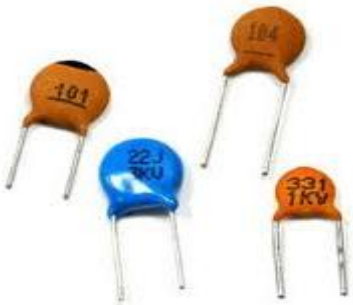


Figura 26 Capacitor de cerámica[29]

Su símbolo es:

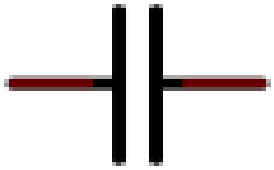


Figura 27 Símbolo del capacitor de cerámica

Condensador

Figura 28. Condensador[30]

Su símbolo es:

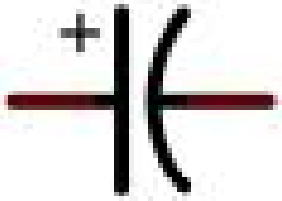


Figura 29. Símbolo del condensador

[28] Imagen de robotjuanydavid.blogspot.com
[29] Imagen tomada de donachip.gostorego.com
[30] Imagen tomada de tecnoblogmanu.blogspot.com

Transistor

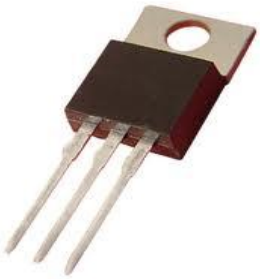


Figura 30. Transistor[31]

Su símbolo es:

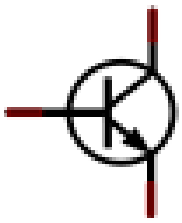


Figura 31. Símbolo del transistor

Relay

Figura 32. Relay[32]

Su símbolo es:

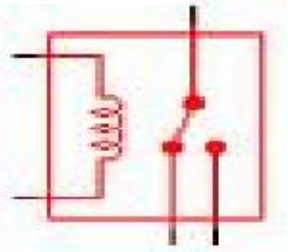


Figura 33. Símbolo del Relay[33]

Circuito integrado

Figura 34. Circuito integrado[34]

Su símbolo es:

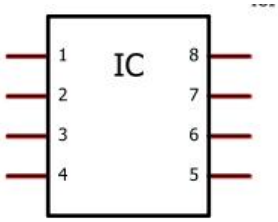


Figura 35. Símbolo del circuito integrado

[31] Imagen de compucanjes.com
[32] Imagen de foroselectronica.es
[33] Imagen de pablin.com.ar
[34] Imagen de eui.upm.es

Polo negativo o tierra

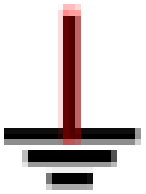


Figura 36. Polo negativo o tierra

Cruce de conductores

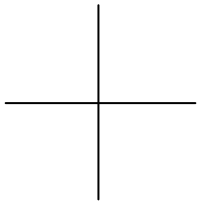


Figura 37. Cruce de conductores

Unión de conductores

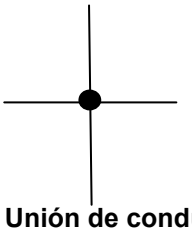


Figura 38. Unión de conductores

Batería

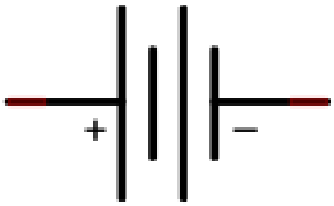


Figura 39. Bateria

Regulador de voltaje

Figura 40. Regulador de voltaje

3.1.3 Bit – Byte – Kilobyte

Un sistema de numeración[35] es un conjunto de símbolos y reglas que permiten representar datos numéricos. Los sistemas de numeración actuales son sistemas posicionales, que se caracterizan porque un símbolo tiene distinto valor según la posición que ocupa en la cifra.

3.1.3.1 Bit

Bit[36] es el acrónimo ***B**inary dig**it***. (Dígito binario). Un bit es un dígito del sistema de numeración binario.

Mientras que en el sistema de numeración decimal se usan diez dígitos, en el binario se usan sólo dos dígitos, el 0 y el 1. Un bit o dígito binario puede representar uno de esos dos valores, **0** ó **1**.

Se puede imaginar un bit como especie de un bombillo, esta encendido:

Figura 41. Bombillo encendido[37]

O apagado:

Figura 42. Bombillo apagado[38]

[35] Tomado de http://platea.pntic.mec.es/~lgonzale/tic/binarios/numeracion.html
[36] Tomado de http://es.wikipedia.org/wiki/Bit
[37] Tomado de elpolvorin.over-blog.es
[38] Imagen de fundaroraima.blogspot.org

El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información. Con él, podemos representar dos valores cuales quiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, masculino o femenino, rojo o azul, etc. Basta con asignar uno de esos valores al estado de "apagado" (0), y el otro al estado de "encendido" (1).

Figura 43. Bits[39]

Las combinaciones de bits, se utilizan para representar o codificar más información en un dispositivo digital, ejemplo si utilizamos dos bits obtendremos:

"Los bits se empiezan a contar de derecha a izquierda"

0	0	Los dos están apagados
0	1	El primero está encendido y el segundo apagado.
1	0	El primero está apagado y el segundo encendido
1	1	Los dos están encendidos

Tabla 3. Combinaciones de bits

Como se dijo anteriormente, los bits son dígitos del sistema de numeración binaria, eso quiere decir que se rige bajo la norma que dice que el valor de la combinación de bits es la suma de los productos de los digito por una base 2 elevada a un exponente desde 0, siempre de derecha a izquierda.

[39] Tomado de Otroblogmas.com

Por ejemplo:

101101

Esto equivale a:

$\mathbf{1}*2^0 + \mathbf{0}*2^1 + \mathbf{1}*2^2 + \mathbf{1}*2^3 + \mathbf{0}*2^4 + \mathbf{1}*2^5$

1 + 0 + 4 + 8 + 0 +32

101101 equivale a 45 en decimal.

3.1.3.2 Byte

Byte[40] es una palabra aceptada como equivalente a octeto(es decir a ocho bits), para fines correctos, un byte debe ser considerado como una secuencia de bits contiguos, cuyo tamaño depende del código de información o código de caracteres en que sea definido.

Se usa comúnmente como unidad básica de almacenamiento de datos en combinación con los prefijos de cantidad.

Figura 44. Byte[41]

[40] Tomado de http://es.wikipedia.org/wiki/Byte

[41] Imagen de byte2011.blogspot.com

El byte es equivalente a un único caracter, como puede ser una letra, un número o un signo de puntuación.

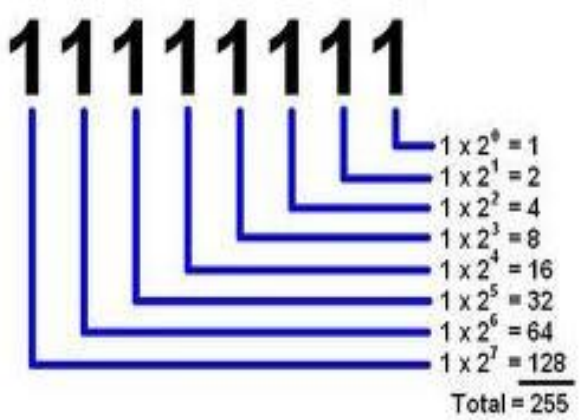


Figura 45. Equivalente de un Byte

Un byte en Arduino almacena un numero si signo de 8 bit desde 0 hasta 255.

Ejemplo:

Byte a = B10010

Donde B es binario y 10010 equivale a 18 en decimal.

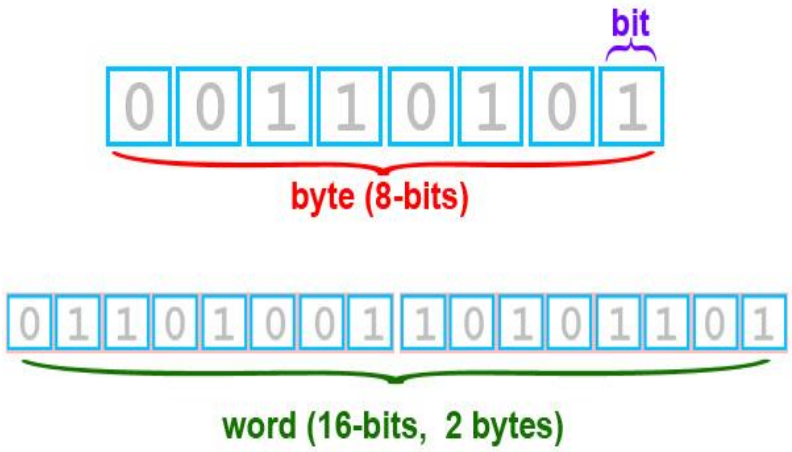


Figura 46. Equivalente de bits en un byte[42]

[42] Imagen de byte2011.blogspot.com

3.1.3.3 *Kilobyte*

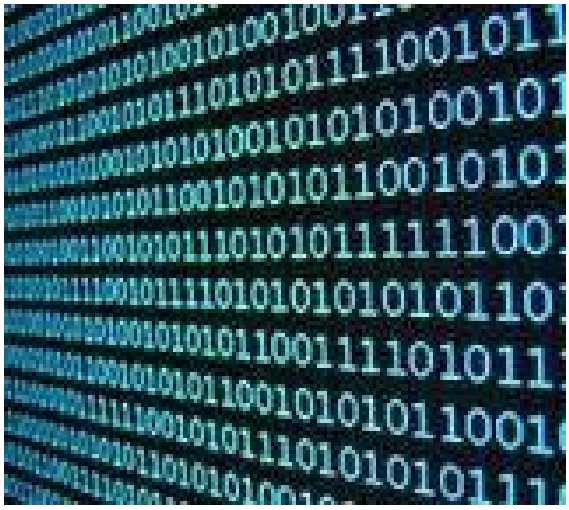


Figura 47. Kilobyte[43]

Un Kilobyte[44] (abreviado como KB o Kbyte) es una unidad de medida equivalente a mil bytes de memoria de ordenador o de capacidad de disco. Por ejemplo, un dispositivo que tiene 256K de memoria puede almacenar aproximadamente 256.000 bytes (o caracteres) de una vez.

En sistemas decimales, kilo significa 1.000, pero el mundo de los ordenadores se basa en un sistema binario de dos en vez de diez. Así pues, un kilobyte es realmente 1.024 (210) bytes. Para distinguir entre una K decimal (1.000) y una K binaria (1.024), el IEEE ha sugerido usar una k minúscula para un kilo decimal y una K mayúscula para un kilo binario.

Nota: Kb es el kilobit, KB es kilobyte.

[43] Imagen de definicion.de
[44] Tomado de http://www.masadelante.com/faqs/kilobyte

3.1.4 Manejo de Memoria

La unidad de manejo de memoria (MMU) es un dispositivo de hardware formado por un grupo de circuitos integrados, responsable del manejo de los accesos a la memoria por parte de la unidad de procesamiento central CPU.

Entre las funciones de este dispositivo se encuentran la traducción de las direcciones lógicas (o virtuales) a direcciones físicas (o reales), la protección de la memoria, el control de caché y, en arquitecturas de computadoras más simples (especialmente en sistemas de 8 bits), Bank switching.

Cuando la CPU intenta acceder a una dirección de memoria lógica, la MMU realiza una búsqueda en una memoria caché especial llamada Buffer de Traducción Adelantada (TLB, *Translation Lookaside Buffer*), que mantiene la parte de la tabla de páginas usada hace menos tiempo. En esta memoria se mantienen entradas de la tabla de páginas (llamadas PTE por sus siglas en inglés, Page Table Entry), donde se pueden rescatar las direcciones físicas correspondientes a algunas direcciones lógicas, de forma directa.

Cuando la dirección requerida por la CPU se encuentra en el TLB, su traducción a dirección real o física es entregada, en lo que se conoce como 'acierto en el TLB' ('TLB hit'). En otro caso, cuando la dirección buscada no se encuentra en el TLB (fallo en el TLB), el procesador busca en la tabla de páginas del proceso utilizando el número de página como entrada a la misma.

En la entrada de la tabla de páginas del proceso se encuentra un bit de presencia, que indica si la página buscada está en memoria principal. Si el bit de presencia

está activado, se carga esta PTE en el TLB y se devuelve la dirección física. En caso contrario, se informa al sistema operativo de la situación, mediante un fallo de página. Es el sistema operativo el encargado de realizar los ajustes necesarios (esto es, cargar la página en memoria física) usando uno de los Algoritmos de reemplazo de páginas, para continuar con la ejecución desde la instrucción que causó el fallo.

Un beneficio fundamental de la MMU es la posibilidad de implementar protección de memoria, evitando que los programas accedan a porciones de memoria prohibidas. Por ejemplo se puede evitar que un programa acceda o modifique sectores de memoria de otros programas.

3.1.4.1 Memoria Estática

La forma más fácil de almacenar el contenido de una variable en memoria en tiempo de ejecución es en memoria estática o permanente a lo largo de toda la ejecución del programa.

No todas las variables pueden ser almacenadas estáticamente.

Para que un objeto pueda ser almacenado en memoria estática su tamaño (número de bytes necesarios para su almacenamiento) ha de ser conocido en tiempo de compilación, como consecuencia de esta condición no podrán almacenarse en memoria estática:

- ✓ Los objetos correspondientes a procedimientos o funciones recursivas, ya que en tiempo de compilación no se sabe el número variables que serán necesarias.

- ✓ Las estructuras dinámicas de datos tales como listas, árboles, etc. ya que el número de elementos que las forman no es conocido hasta que el programa se ejecuta.

Las técnicas de asignación de memoria estática son sencillas.
A partir de una posición señalada por un puntero de referencia se aloja la variable X, y se avanza el puntero tantos bytes como sean necesarios para almacenar la variable X.

La asignación de memoria puede hacerse en tiempo de compilación y las variables están vigentes desde que comienza la ejecución del programa hasta que termina.

3.1.4.2 Memoria EEPROM

Figura 48. Memoria EEPROM[45]

EEPROM[46] son las siglas de Electrically-Erasable Programmable Read-Only Memory (Memoria de solo lectura programable y borrable eléctricamente).

[45] Imagen de lizykaryhermosas.blogspot.com
[46] Tomado de http://es.wikipedia.org/wiki/Memoria_EEPROM

Como su nombre lo indica es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, es una memoria no volátil.

Las memorias de tipo EEPROM tienen como principal cualidad el permitir el almacenamiento y la sobre-escritura de datos por medio de los voltajes de operación norma de los circuitos electrónicos, además sostienen la información por muchos años sin fuente de alimentación.

Podemos encontrar circuitos integrados de memorias EEPROM paralelas, compatibles pin a pin con circuitos con circuitos de memoria RAM o de memoria EPROM.

Este tipo de memorias precisamente por ser de interfaz paralela, tiene muchos pines externos por medio de los cuales recibe y entrega los datos y permite el direccionamiento de las distintas posiciones de almacenamiento. Debido a esto, los circuitos integrados son de gran tamaño físico, impidiendo ser utilizados en aplicaciones que requieran tamaño reducido.

Con las memorias EEPROM de interfaz serial, el control se ha reducido solamente a unos cuantos pines que son utilizados para entrada o salida de datos en forma serial (1 ó 2 pines), habilitación (1 pin), reloj de sincronismo (1 pin), direccionamiento de dispositivo (3 pines) que no existen en la interfaz paralela y por último los pines de alimentación del circuito (2 pines).

Los datos y la dirección de las posiciones de memoria utilizarán únicamente uno o dos pines, dependiendo del tipo de comunicación utilizada (dos o tres hilos). La velocidad de transferencia de datos puede variar desde lo 100 KHz hasta los 600 MHz, dependiendo del tipo de memoria y del sistema de comunicación utilizados.

Características principales de la EEPROM:

- ✓ Se pueden conectar fácilmente con microprocesadores o microcontroladores, algunas de estas memorias tienen pines para realizar esta labor.
- ✓ Transferencia de datos de manera serial, lo que permite ahorro del micro para dedicarlo a otras funciones.

- ✓ El consumo de corriente es mucho menor que en las memorias que trabajan en paralelo.

3.1.4.3 Flash

Figura 49. Flash[47]

La memoria flash es una tecnología de almacenamiento, derivada de la memoria EEPROM que permite la lecto-escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, la tecnología *flash*, siempre mediante impulsos eléctricos, permite velocidades de funcionamiento muy

[47] Imagen de jesustecman.blogspot.com

superiores frente a la tecnología EEPROM primigenia, que sólo permitía actuar sobre una única celda de memoria en cada operación de programación.

Este tipo de memoria suele ser usada en cámaras digitales, celulares, reproductores, PDA´s.

3.1.4.4 SRAM

Figura 50. SRAM[48]

SRAM son las siglas de Static Random Aleatory Memory, que traducido es Memoria estática de acceso aleatorio.

La SRAM es un tipo de memoria basada en semiconductores que a diferencia de la memoria DRAM, es capaz de mantener los datos, mientras esté alimentada, sin necesidad de un circuito de actualización. Sin embargo, sí son memorias volátiles, es decir que pierden la información si se les interrumpe la alimentación eléctrica.

Estas memorias son de Acceso Aleatorio, lo que significa que las posiciones en la memoria pueden ser escritas o leídas en cualquier orden, independientemente de cuál fuera la última posición de memoria accedida.

[48] Imagen de tweaktown.com

3.1.5 Conociendo arduino (Introducción a microcontroladores)

3.1.5.1 ¿Qué es un Microcontrolador?

Figura 51. Microcontrolador[49]

Un microcontrolador[50] es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de Entrada/Salida, es decir, se trata de un computador completo en un solo circuito integrado.

Los microcontroladores[51] son diseñados para disminuir el coste económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la CPU, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación.

[49] Imagen tomada de memito-trabajospics.blogspot.com
[50] Galeon Octubre 4 2011 http://microcontroladores-e.galeon.com/
[51] Galeon Octubre 4 2011 http://microcontroladores-e.galeon.com/

3.1.5.2 ¿Qué es arduino?

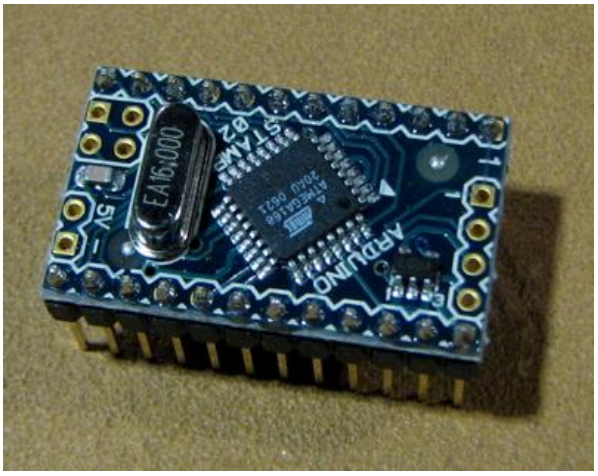


Figura 52. Placa Arduino[52]

Arduino[53] es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos.

Figura 53. Proyecto con Arduino[54]

[52] Imagen tomada de elecronicaestudio.com
[53] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >
[54] Imagen tomada de antoniotoriz.blogspot.com

Las ventajas de arduino son:

- ✓ Lo pueden utilizar artistas, diseñadores, aficionados o cualquier otro interesado.
- ✓ Facilidad de instalación tanto de sus componentes como de su software.
- ✓ Su programación es sencilla.
- ✓ Se encuentran gran cantidad de proyectos en la comunidad web.
- ✓ Es abierto, lo cual nos permite modificar con libertad.
- ✓ Sus componentes son reutilizables.

3.1.5.3 Modelos de Placas Arduino

3.1.5.3.1 Arduino Diecimila

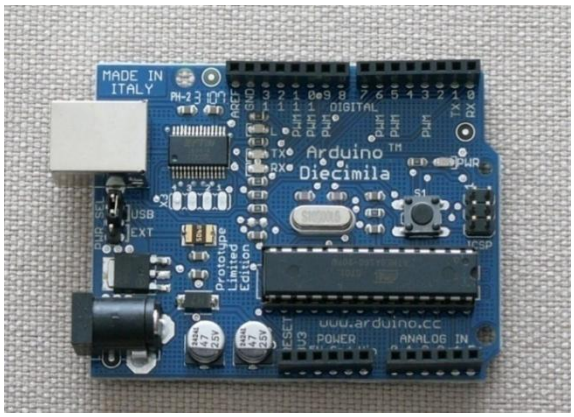


Figura 54. Arduino Diecimila[55]

La Arduino Diecimila[56] es una placa microcontroladora basada en el chip ATmega168. Tiene 14 Entradas/Salidas digitales, 6 entradas analógicas, un cristal de 16MHz, conexión USB y botón de reseteo. Contiene todo lo necesario para el soporte del microcontrolador.

[55] Imagen tomada de ingeniosolido.com
[56] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

3.1.5.3.2 Arduino Uno

Figura 55. Arduino Uno[57]

El Arduino Uno[58] es una placa electrónica basada en el ATmega328. Tiene 14 entradas / salidas digitales pines, 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reseteo.

3.1.5.3.3 Arduino Mega

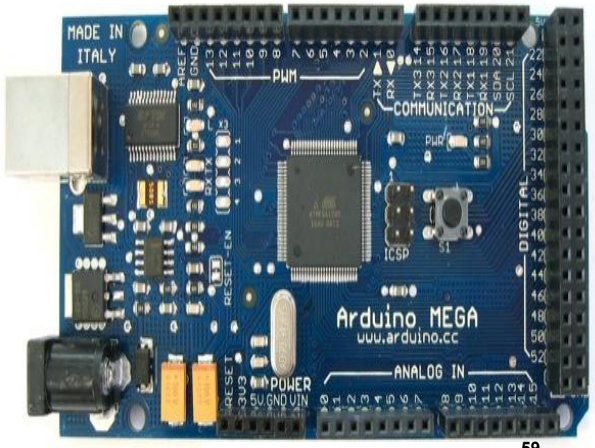


Figura 56. Arduino Mega[59]

[57] Imagen tomada de www.arduino.cc
[58] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >
[59] Imagen tomada de www.arduino.cc

El Arduino Mega[60] es una placa microcontrolador basada ATmeg1280. Tiene 54 entradas/salidas digitales, 16 entradas analógicas, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reseteo.

3.1.5.3.4 Arduino Mega2560

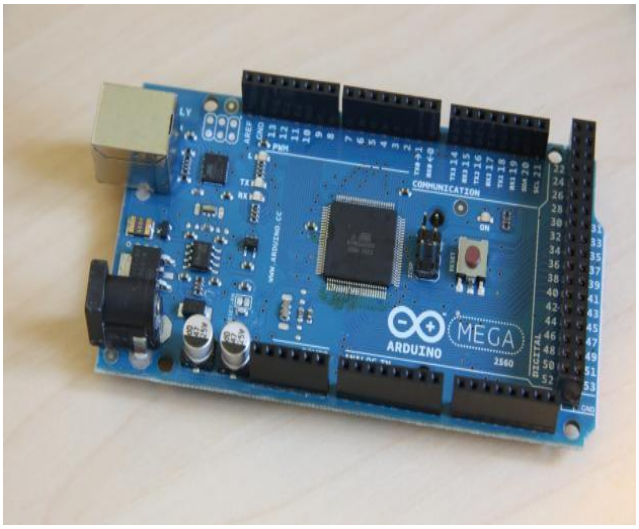


Figura 57. Arduino Mega2560[61]

El Arduino Mega2560[62] es una placa electrónica basada en el Atmega2560. Cuenta con 54 pines de entradas / salidas digitales, 16 entradas analógicas, 4 UART (puerto serie del hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reseteo.

[60] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >
[61] Imagen tomada de chemicaloliver.net
[62] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

3.1.5.3.5 Arduino Fio

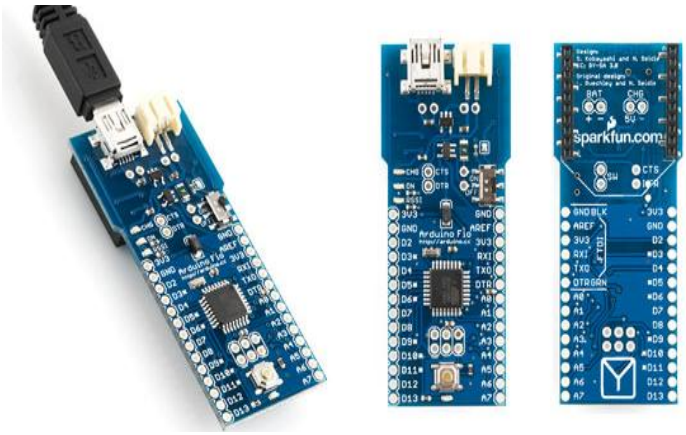


Figura 58. Arduino Fio[63]

El Arduino Fio[64] es una placa para microcontrolador basada en el ATmega328P. Funciona a 3.3V y 8MHz. Tiene 14 pines de Entrada/Salida digitales, 8 entradas analógicas, un resonador en placa, un botón de reinicio (*reseteo*), y agujeros para montar conectores de pines.

Tiene conexiones para una batería de polímero de Litio e incluye un circuito de carga a través de USB.

El reverso de la placa tiene disponible un zócalo para módulos XBee (Módulos de transmisión inalámbrica).

El Arduino FIO está diseñado para aplicaciones inalámbricas. El usuario puede subir sus *sketches* (bocetos) con un cable FTDI (Futura Tecnología de Dispositivos Internacionales) o una placa adicional adaptadora. Además, si utiliza un adaptador de USB a XBee modificado, como el USB Explorador de XBee, el usuario puede subir *sketches* de forma inalámbrica. La tarjeta viene sin conectores pre-montados, permitiendo el uso de diversos tipos de conectores o la soldadura directa de los cables.

[63] Imagen tomada de www.arduino.cc
[64] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

3.1.5.3.6 Arduino Nano

Figura 59. Arduino Nano[65]

El Arduino Nano[66] es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x) que se usa conectándola a una protoboard (Placa de prueba). Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B en vez del cable estándar.

3.1.5.3.7 Arduino Lilypad

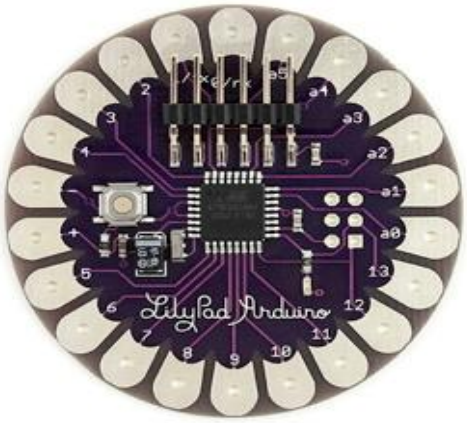


Figura 60. Arduino Lilypad[67]

[65] Imagen tomada de www.robotshop.com
[66] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >
[67] Imagen tomada de www.arduino.cc

El LilyPad[68] Arduino es una placa con microcontrolador diseñado para prendas y e-textiles. Puede utilizar con complementos similares como fuentes de alimentación, sensores actuadores unidos por hilo conductor. La placa está basada en el ARmega168V (la versión de baja consumo del ATmega168).

3.1.5.3.8 Arduino Duemilanove

Figura 61. Arduino Duemilanove[69]

El Arduino Duemilanove[70] ("2009") es una placa con microcontrolador basada en el ATmega168 o elATmega328, Tiene 14 pines con entradas/salidas digitales, 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP, y un botón de reseteo.

El Arduino Duemilanove puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

[68] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >
[69] Imagen tomada de olimex.cl
[70] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

3.1.6 Simulador arduino Virtual BreadBoard

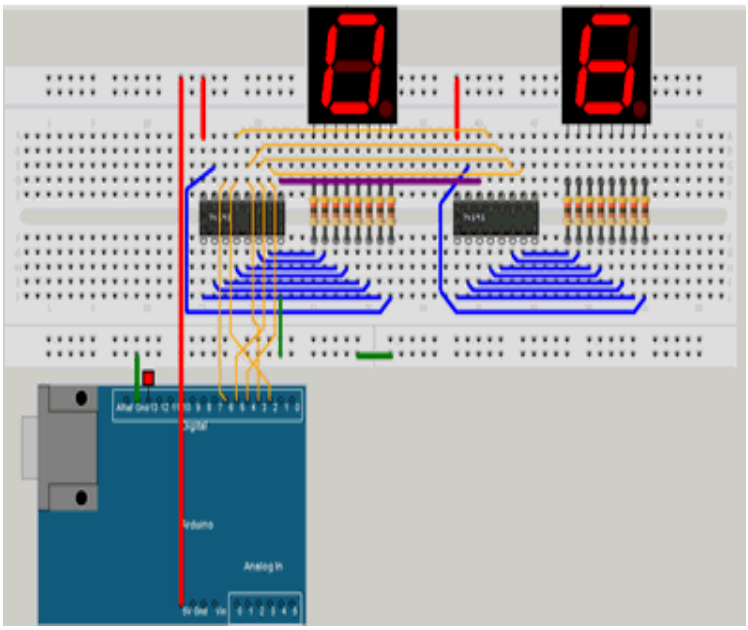


Figura 62. Entorno del Simulador Virtual BreadBoard[71]

Virtual Breadboard[72] es una simulación y entorno de desarrollo de aplicaciones integradas que utilizan los microcontroladores, es fácil de usar y puede sustituir una protoboard para experimentar con nuevos diseños.

Además, Virtual Breadboard nos ofrece:

- ✓ experimentar la electrónica con seguridad.
- ✓ Simular y visualizar de forma interactiva.
- ✓ Compartir las creaciones en vivo.
- ✓ Aprender con ejemplos.

[71] Imagen tomada de www.virtualbreadboard.com
[72] Tomado de www.virtualbreadboard.com

Para empezar a trabajar con este simulador antes debemos instalar Microsoft Visual J# 2.0, que podemos descargar su instalador fácilmente

Figura 63. Icono de J# 2.0

Lo ejecutamos y saldrá lo siguiente:

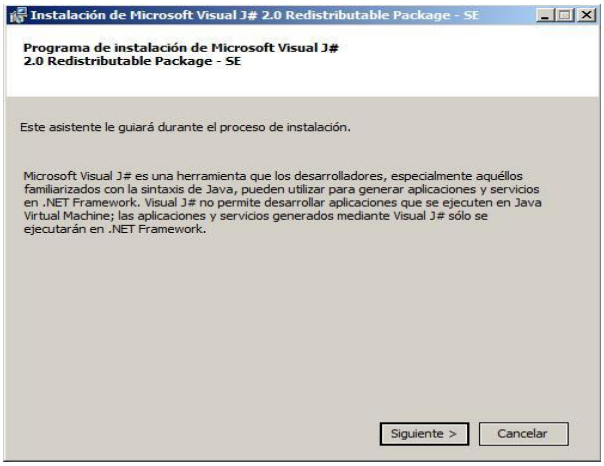


Figura 64. Instalación de J# 2.0 parte 1

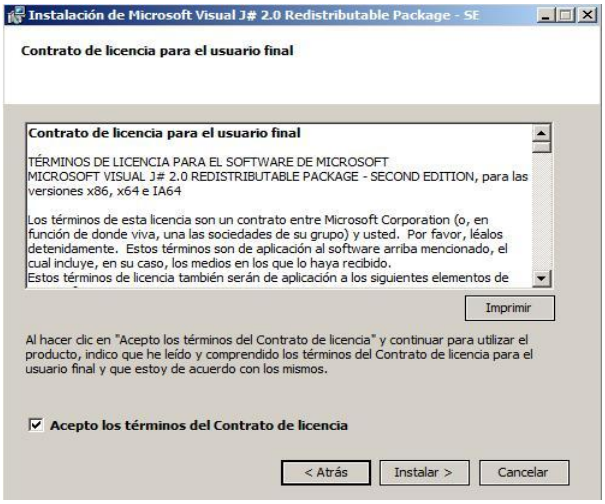


Figura 65. Instalación de J# 2.0 parte 2

Aceptamos los términos de licencia y le damos instalar

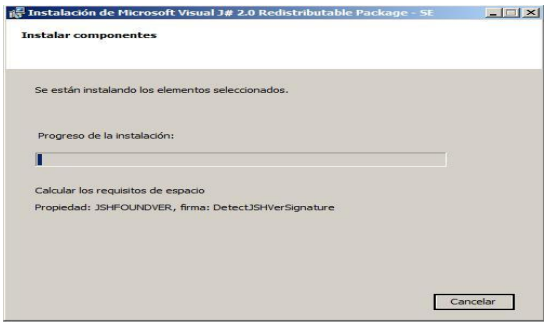


Figura 66. Instalación de J# 2.0 parte 3

Comienza la instalación

Figura 67. Instalación de J# 2.0 parte 4

Termina nuestra instalación, ahora instalemos Virtual Breadboard, ejecutemos Setup, y realicemos los siguientes pasos:

Figura 68. Icono de setup de Virtual BreadBoard

Figura 69. Instalación de Virtual BreadBoard

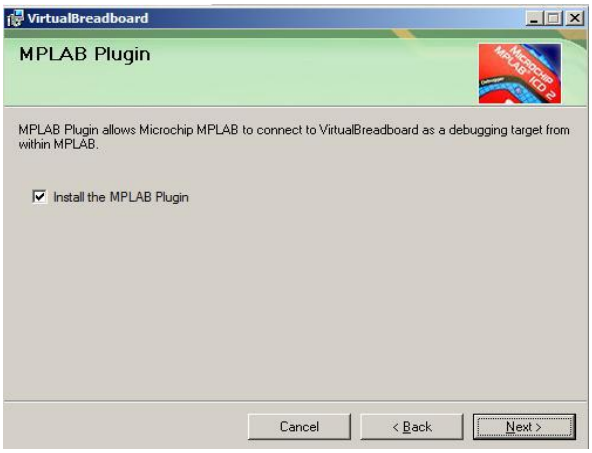


Figura 70. Instalación de Virtual BreadBoard parte 1

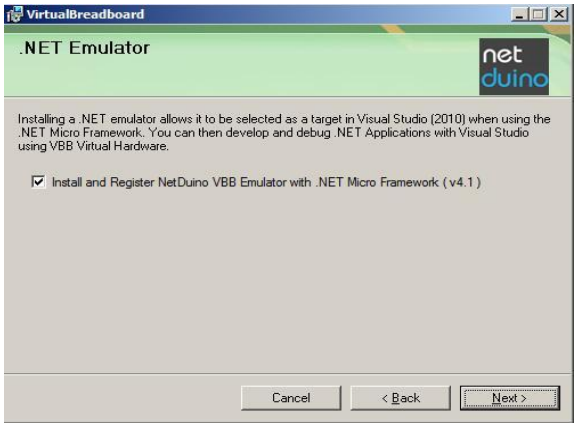


Figura 71. Instalación de Virtual BreadBoard parte 2

Elegimos la ubicación, y quien lo puede usar, solo yo (Just me) o si todos (Everyone).

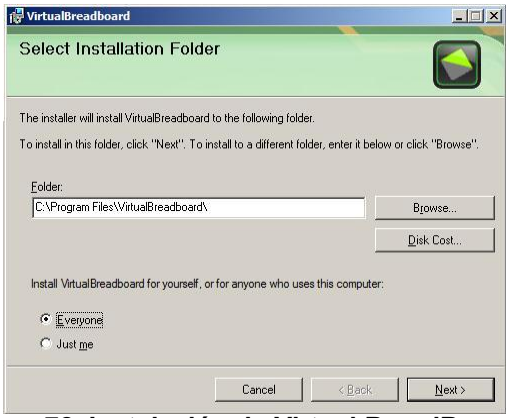


Figura 72. Instalación de Virtual BreadBoard parte 3

Confirmamos la instalación

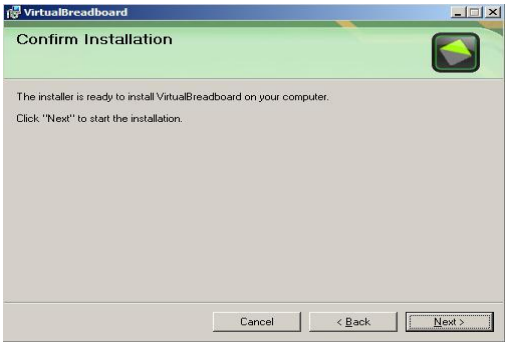


Figura 73. Instalación de Virtual BreadBoard parte 4

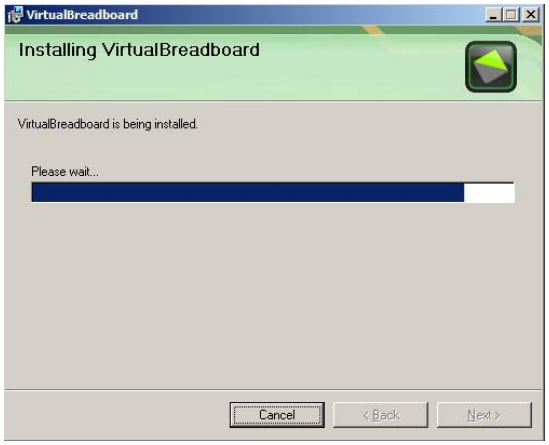


Figura 74. Instalación de Virtual BreadBoard parte 5

Instalación completada

Figura 75. Instalación de Virtual BreadBoard parte 6

En el menú de inicio nos aparecerá el acceso rápido a la aplicación:

Figura 76. Acceso rápido a Virtual BreadBoard

Y nos abrirá esta interface:

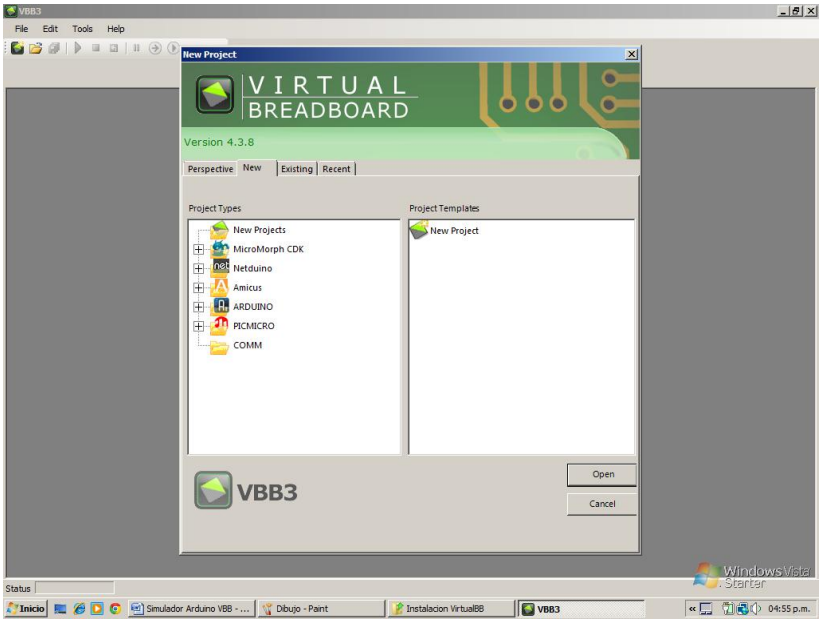


Figura 77. Interface de Virtual BreadBoard

Ahora si usemos Virtual Breadboard.

Para ver simulaciones ya creadas, hacemos lo siguiente, por ejemplo, como estamos trabajando la placa arduino: Ejecutamos Arduino, de nuevo Arduino, escogemos digital y de la serie de opciones seleccionaremos Blink, eso nos llevara a esta interface:

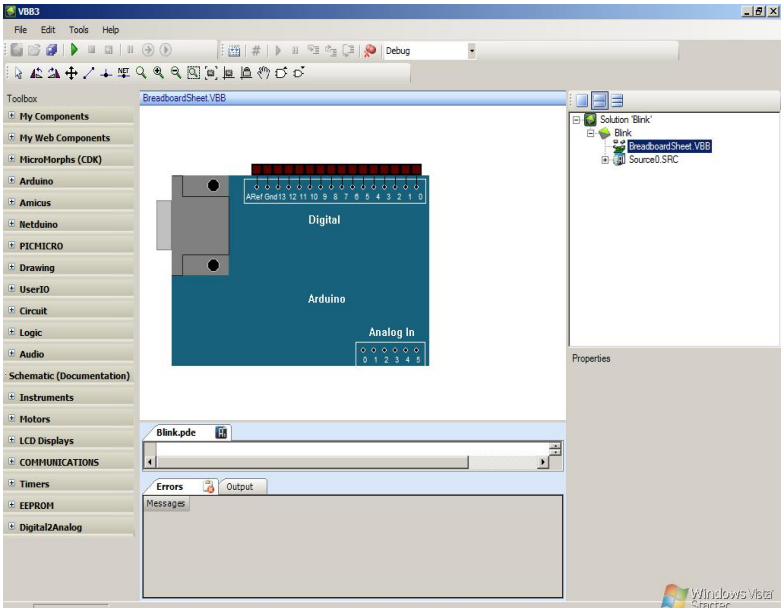


Figura 78. Funcionamiento de Virtual BreadBoard

Donde se encuentra un simulador de la placa, de led´s, la consola de programación y errores, además de todos los componentes que podemos utilizar.

Podemos darle play o run 

Figura 79. Icono de Play o Run

y después le damos nuevamente run 

Figura 80. Icono de Play o Run

y veremos el programa ejecutarse sobre la placa

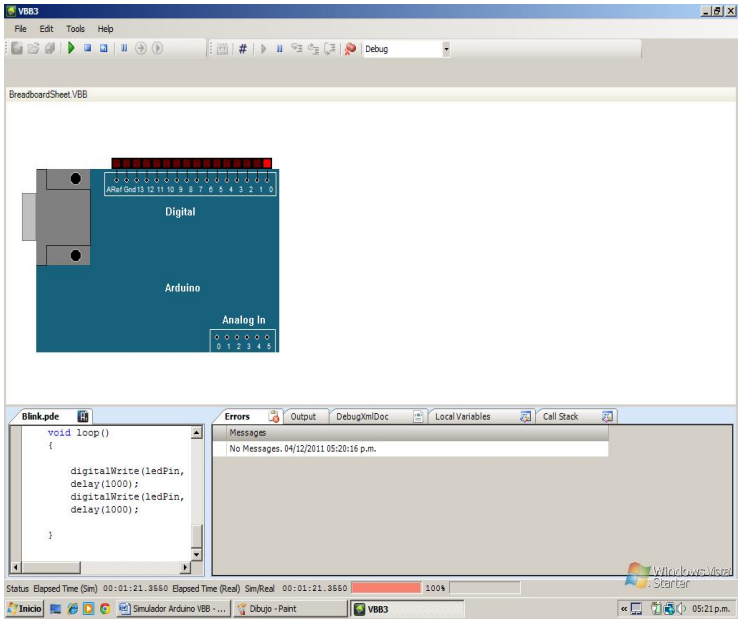


Figura 81. Funcionamiento de Virtual BreadBoard

Se ve como el led prende y apaga, como si estuviera sobre una placa real.

Le damos stop  y nos regresara a la ventana inicial.

Figura 82. Icono de Stop

Ahora entremos a File, y seleccionemos Close solución para empezar de nuevo; luego seleccionamos File nuevamente y escogemos New, y en la ventana de opciones escogemos new Project, para empezar a trabajar nuestro propio proyecto.

Guardamos el proyecto, le damos el nombre que creamos conveniente, por ejemplo, este proyecto se llamara bandera.

En el toolbox, encontraremos la opción Arduino, escogeremos ArduinoStandar y arrastraremos la placa hasta nuestro simulador, en el mismo toolbox entramos a UserIO y arrastramos 3 led´s, que serán los 3 colores de nuestra bandera colombiana.

Figura 83. Toolbox de Virtual BreadBoard

Después de puestos nuestros 3 led´s, le cambiamos el color a cada uno en la barra de propiedades

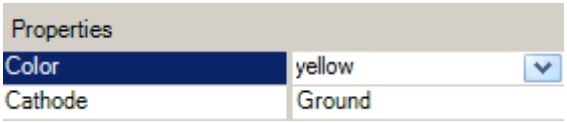


Figura 84. Propiedades de Virtual BreadBoard

Procedemos a programar la placa:

```
void setup(){

    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(8, OUTPUT);
}
void loop()  {

    digitalWrite(10, HIGH);
    delay(500);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(8, HIGH);

    delay(2000);

    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);

    delay(1000);
}
```

Cuadro de código fuente 1

Este código prende y apaga los led´s de manera paulatina, dependiendo de un retardo.

3.1.7 Instalación arduino

Para empezar debemos tener al alcance una placa Arduino UNO (opcional, el modelo de la placa es variable) y un cable USB como el que se usa para conectar una impresora.

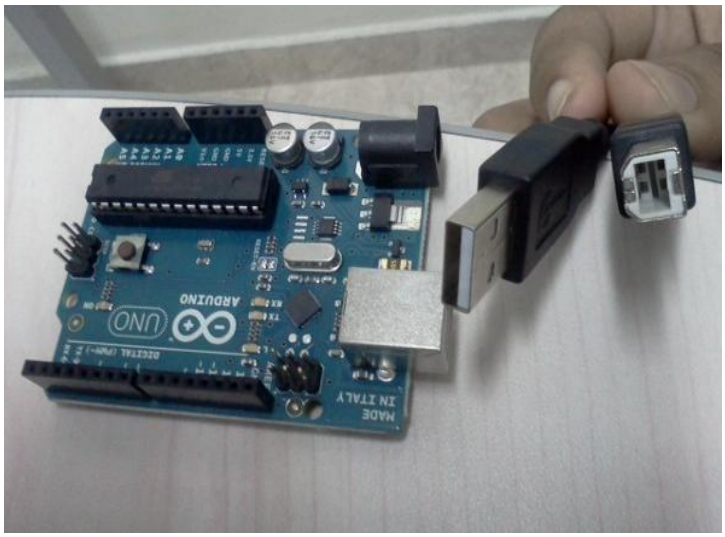


Figura 85. Funcionamiento de Virtual BreadBoard

Descargamos el IDE de Arduino dependiendo del modelo de la placa, en este caso para la placa Arduino UNO descargaremos Arduino-0022, pero puede ser otra versión Arduino-00XX que incluye los drivers de instalación de la placa.

Cuando la descarga finalice se descomprime el fichero, tratando siempre de mantener la estructura de los archivos, como muestra la imagen.

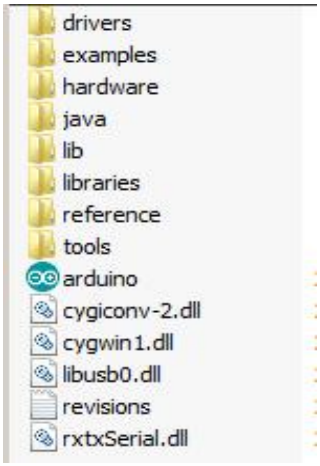


Figura 86. Fichero de Arduino

Conectamos la placa al ordenador, un led verde se encenderá indicando la alimentación, (mientras la placa se encuentre conectada el led deberá permanecer encendido).

Figura 87. Alimentación de la placa Arduino

El sistema operativo indicará inicializar la instalación de los drivers (Siempre y cuando nunca se haya trabajado con la placa) y debemos indicarle que no a todas las opciones.

Entrar a las propiedades del equipo (Mi PC) es lo siguiente, damos click derecho sobre él y listo.

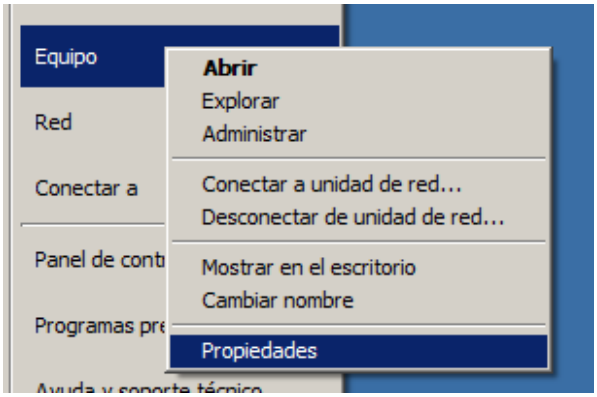


Figura 88. Propiedades del Equipo

En las propiedades se entrara al administrador de dispositivos.

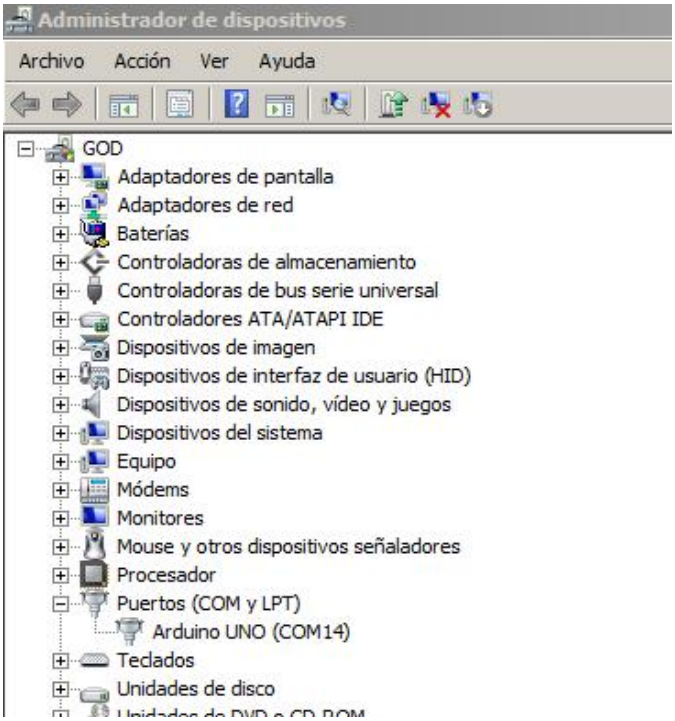


Figura 89. Administrador de dispositivos

En el administrador de dispositivos[73] se encuentran todos los dispositivos están instalados en el equipo, y mediante el se puede actualizar su software controlador, comprobar si algún dispositivo funciona correctamente y modificar sus opciones de configuración.

En la opción Puertos (COM y LPT) se encuentra ubicada nuestra placa, debemos tener en cuenta en que numero de puerto se encuentra, ya que puede variar su ubicación, (En la imagen vemos se encuentra en el puerto numero 14).

Le damos clik derecho donde se encuentra el nombre de nuestra placa y escogemos la opción actualizar software controlador nos aparecerá una pantalla como la siguiente:

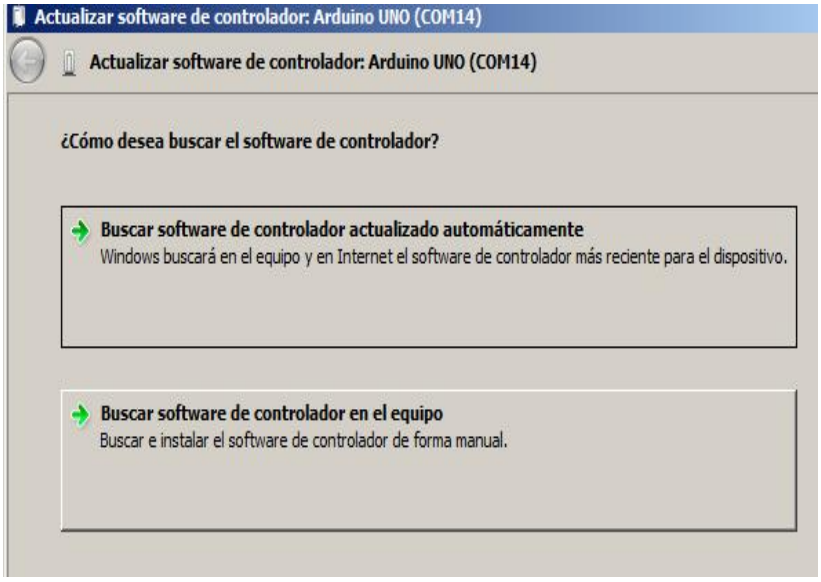


Figura 90. Búsqueda del software controlador

[73] Tomado de http://windows.microsoft.com/es-ES/

Y escogemos la opción “Buscar software de controlador en el equipo”.

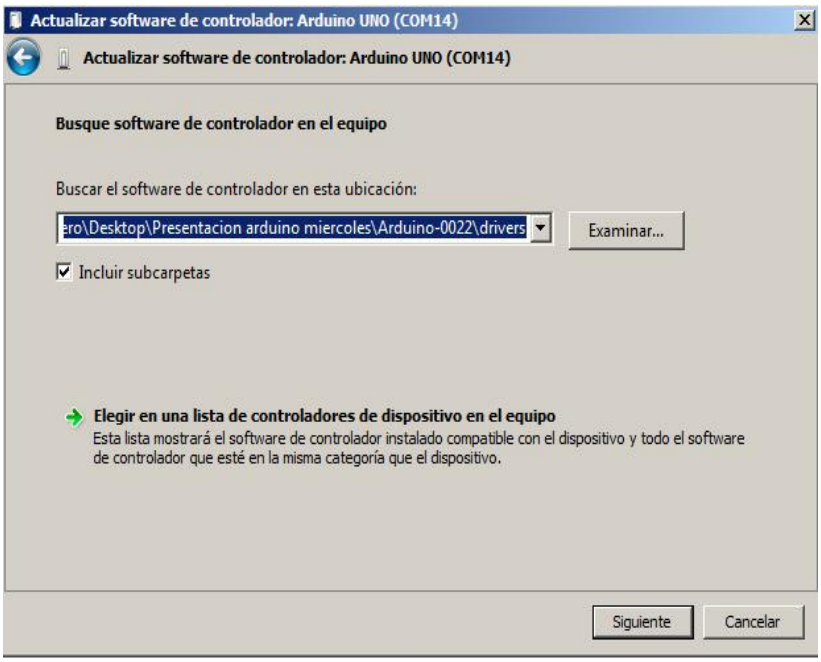


Figura 91. Software Controlador

Le damos examinar y buscamos la carpeta drivers que se encuentra en nuestra carpeta IDE Arduino-00XX que habíamos descargado antes e incluimos las subcarpetas y le damos siguiente, al finalizar no dirá que nuestro software esta actualizado y le damos cerrar.

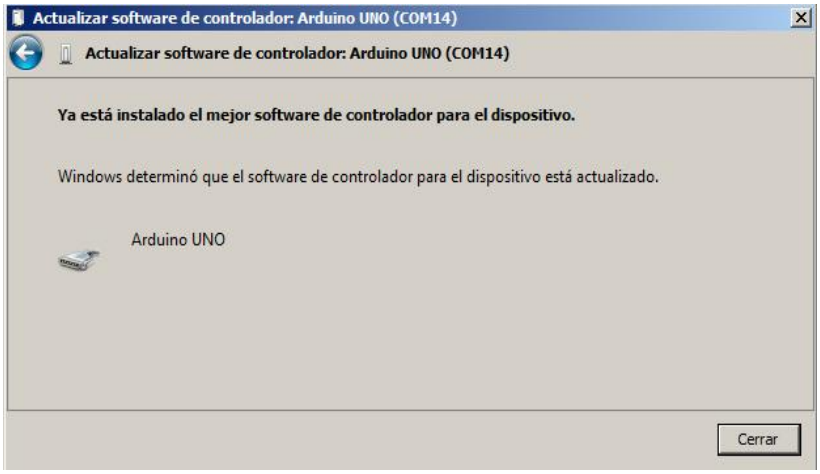


Figura 92. Software controlador actualizado

Nuestra placa está instalada, para comenzar a trabajar con ella debemos ejecutar nuestra aplicación Arduino que se encuentra en nuestra carpeta Arduino-00XX

Figura 93. Icono de la aplicación Arduino

Se abrirá una ventana como en la siguiente imagen:

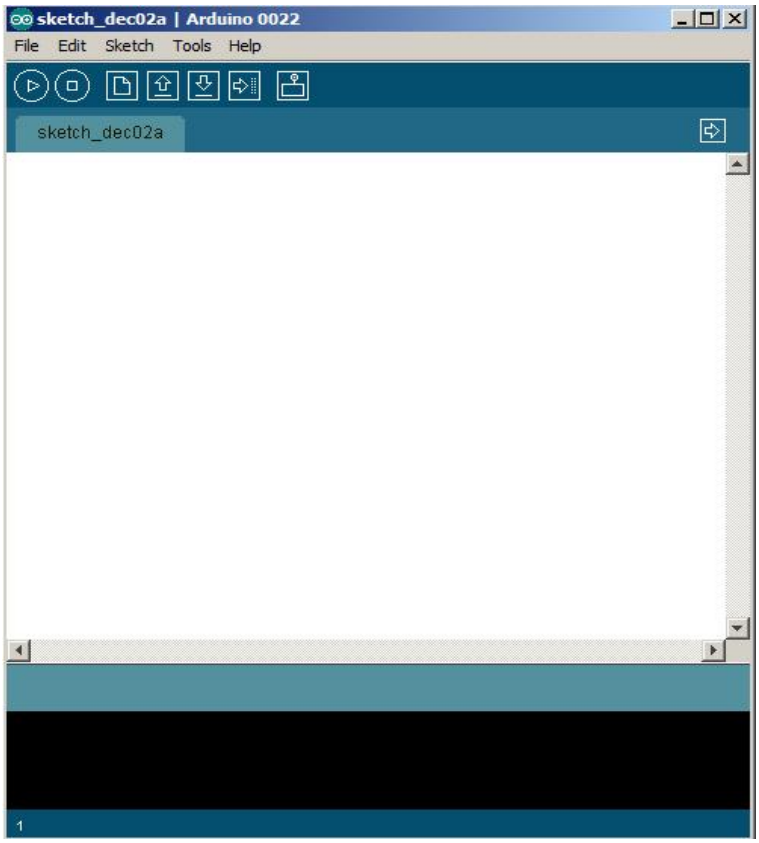


Figura 94. Entorno de la aplicación Arduino

En la cinta de opciones seleccionamos Tools, y entramos a la opción Board y escogemos la referencia de nuestra placa.

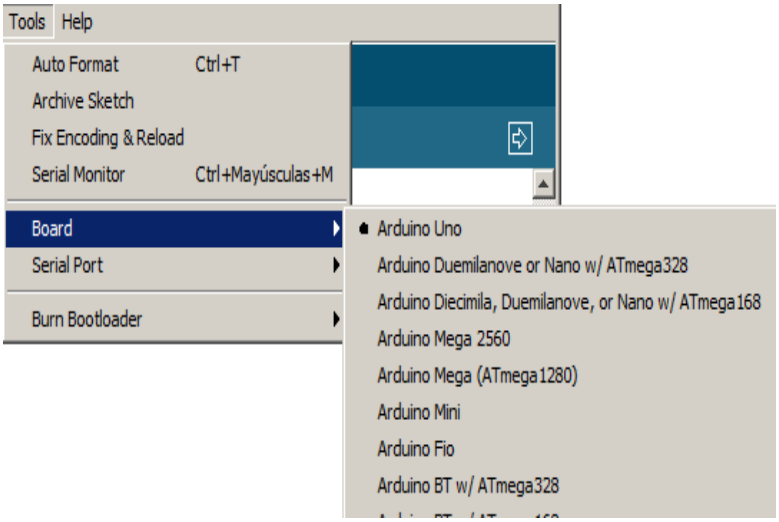


Figura 95. Cinta de herramientas de la aplicación Arduino parte 1

Luego ahí mismo en Tools seleccionamos Serial Port (puerto serial) y escogemos el puerto en el que se encuentra la placa, este puerto lo vimos en el administrador de dispositivos cuando actualizamos el software de controlador.

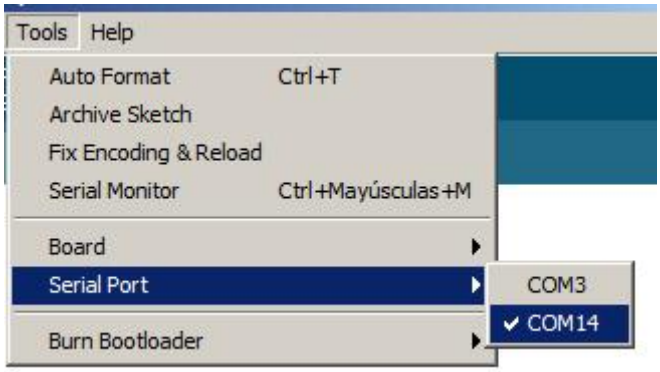


Figura 96. Cinta de herramientas de la aplicación Arduino parte 2

Nuestro IDE reconoce nuestra placa, vamos a probarlo con un ejemplo sencillo:

Nuestra placa tiene un led incorporado en el pin 13 haremos que este led encienda desde nuestro IDE.

En nuestra cinta de opciones entramos a File, luego Examples, después pasamos a Basics y le damos BLink.

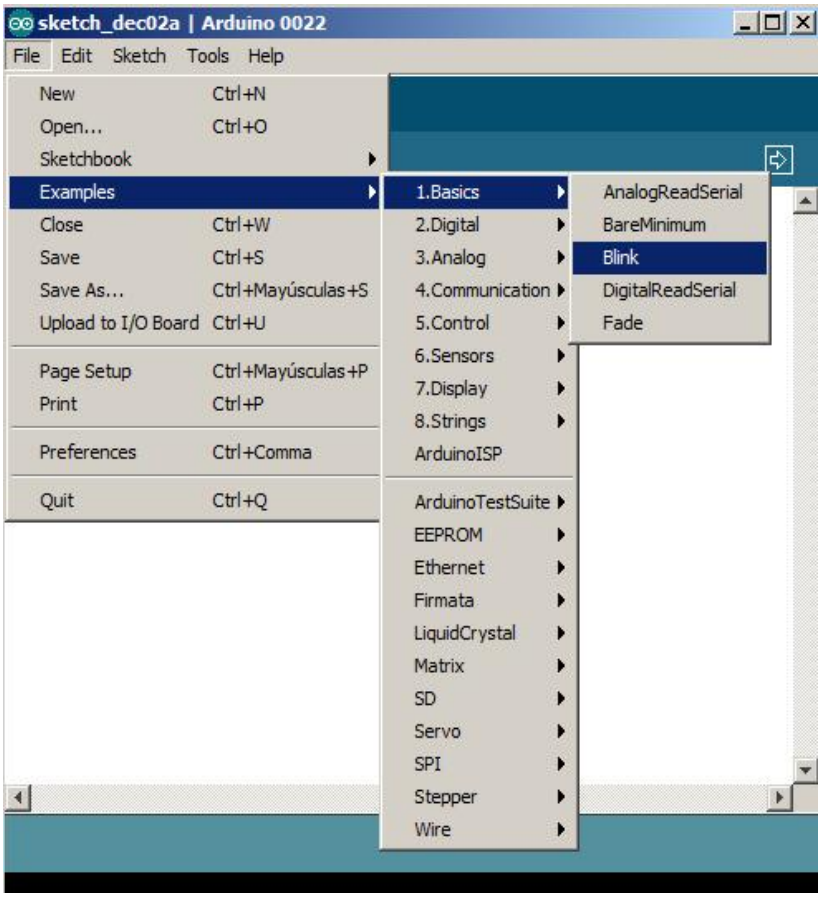


Figura 97. Cinta de opciones de la aplicación Arduino

Nos abrirá este sencillo programa, que hace que nuestro led ubicado en nuestro pin 13 parpadee

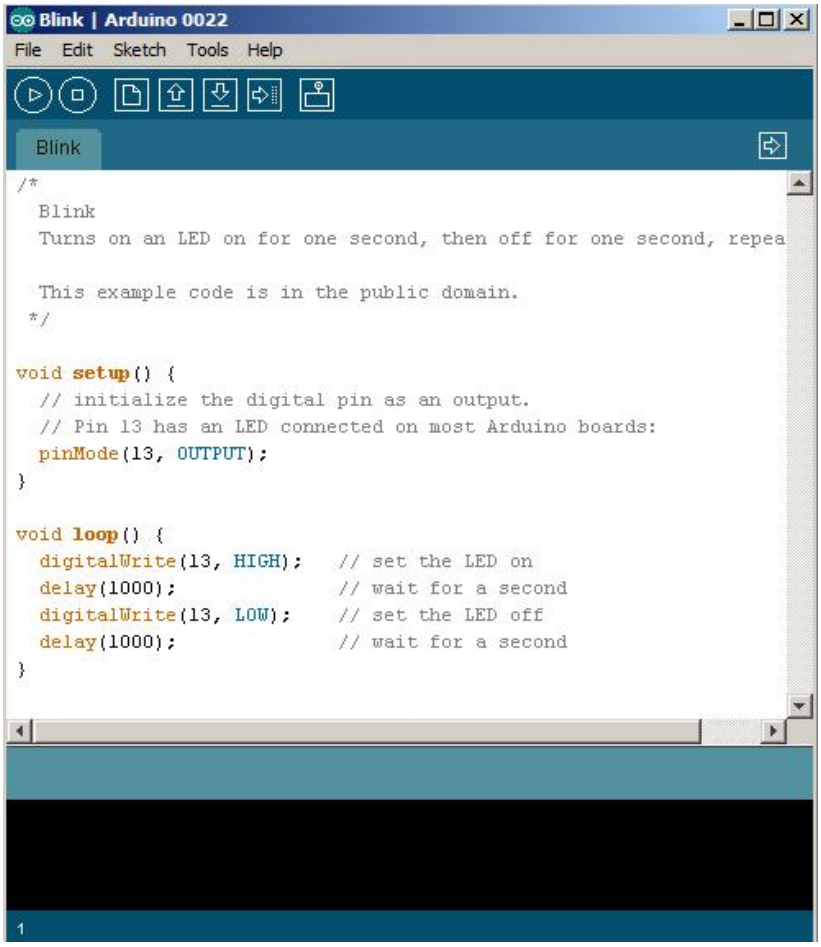


Figura 98. Probando la aplicación Arduino

En nuestra barra:

Figura 99. Barra de opciones de la aplicación Arduino

Presionamos 

Figura 100. Subiendo el programa a la placa

Que sube el programa a la placa, y en la consola debe salir:

Figura 101. Mensaje de la consola

Y al finalizar, si todo es correcto debe aparecer:

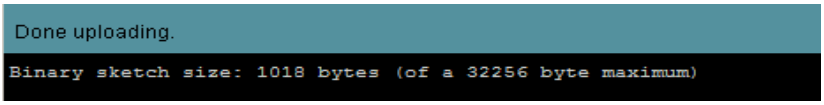


Figura 102. Placa programada

Indicando que no hay errores.

Miramos nuestra placa y nos damos cuenta que el led que se encuentra en el pin 13 parpadea, quiere decir prende y apaga cada segundo.

Nuestra placa esta lista y funcionando.

3.1.8 Aprendiendo Lenguaje de Programación I: Estructura

El lenguaje utilizado en la programación de la placa Arduino es llamado Processing, este es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización.

3.1.8.1 Estructura base

La estructura básica del lenguaje de programación de Arduino se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen declaraciones, estamentos o instrucciones.

```
void setup ()  {
        Instrucciones;
            }
void loop ()  {
        Instrucciones;
            }
```

Cuadro de código fuente 2

Void[74] que en español significa vacio, es una palabra reservada, que se utiliza solo para declarar funciones e indica que se espera que no devuelva información a la función donde fue llamada.

setup()

Es la parte encargada de recoger la configuración, se invoca una vez solo cuando el programa empieza, debe ser incluido en un programa aunque no haya declaración que ejecutar, se utiliza para inicializar y/o configurar los pin´s, modo de trabajo de entradas y salidas o puerto serie. Ejemplo:

```
Void setup()
    {
        PinMode (pin, OUTPUT); // Configura el pin como salida
    }
```

loop()

Es la función núcleo de cualquier programa, contiene el código que se ejecutará repetidamente, lee las entradas, activa las salidas, etc. Ejemplo:

[74] Tomado de http://arduino.cc/es/Reference/Void

```
void loop()
        {
digitalWrite(pin, HIGH); // Enciende el ´pin´
delay(3000); // espera tres segundos (3000 ms)
digitalWrite(pin, LOW); // Apaga el ´pin´
delay(4000); // espera cuatro segundos (4000 ms)
        }
```

Cuadro de código fuente 3

3.1.8.2 Sintaxis

Uso de llaves { }

Las llaves sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para las funciones de programación setup(), loop(), condicionales, etc. Una llave de apertura "{"siempre debe ir seguida de una llave de cierre "}", si no es así el entorno de programación de Arduino incluye una herramienta de gran utilidad que arrojará errores al compilar el programa.

Punto y coma ;

El punto y coma ";" es necesario para que el programa sepa cuándo termina cada instrucción y empieza otra, el olvidarse de un punto y coma es traducido en un error al momento de la compilación.

Bloque de comentarios /*… */

El bloque de comentarios es un área de texto que utilizamos para hacer referencia a la descripción de un código o del programa en general, éste no es reconocido por el programa empiezan con "/*" y terminan con "*/" y puede abarcar un número indeterminado de líneas. El no cerrar el bloque de comentarios desencadena errores en el programa.

Línea de comentario //

Una línea de comentario se utiliza con frecuencia al final de una instrucción, ya sea para recordar algo o para proporcionar alguna información de la instrucción, es precedida por "//" y termina con el inicio de la siguiente línea de código y al igual que los bloques de código son ignoradas por el programa.

3.1.8.3 Operadores Aritméticos

Asignación =

Guarda el valor en la derecha del símbolo igual "=" en la variable que se encuentra en la parte izquierda, ejemplo:

Int variable = 0;

Suma +, Resta - , Multiplicación *, División /

Los operadores aritméticos son suma, resta, multiplicación y división, estos devuelven la suma, diferencia, producto o cociente de dos operandos (Variables), teniendo en cuenta el tipo de dato con que se ha definido una variable ejemplo:

Int var1 = 4; **int** var2 = 3; y **int** var3 = 0;

Var3 = var1 / var2 El resultado (var3) será igual a 1 y no ha 1.33 ya que las variables var1, var2 y var3 son de tipo entero y no se reconocen decimales con este tipo de datos.

Modulo %

Este operador se utiliza para saber el residuo de una división, ejemplo:

Int var1 = 7; **int** var2 = 7; y **int** var3 = 0;

Var3 = var1 % var2 El resultado (var3) será igual a 0.

Si var2 = 5, el resultado (Var3) será igual a 2.

3.1.8.4 Operadores Comparativos

Los operadores comparativos son usualmente una forma de comparar dos variables, se utilizan con frecuencia en los condicionales del tipo if para saber si una condición es cierta. Los operadores utilizados son:

Igual a == *ejemplo: var1 == var2 // indicando que una variable es igual a otra*

Distinto de != *ejemplo: var1 != var2 // indicando que la variable 1 es diferente a la variable 2*

Menor que < *ejemplo: var1 < var2 // indicando que la variable 1 es menor que la variable 2*

Mayor que > *ejemplo: var1 > var2 // indicando que la variable 1 es mayor que la variable 2*

Menor o igual que <= *ejemplo: var1 <= var2 // indicando que la variable 1 puede ser menor o igual que la variable 2*

Mayor o igual que >= *ejemplo: var1 >= var2 // indicando que la variable 1 puede ser mayor o igual que la variable 2*

3.1.8.5 Operadores Booleanos o Lógicos

Estos operadores se utilizan para comparar dos expresiones y que al igual que los operadores comparativos se utilizan con frecuencia en los condicionales del tipo if, los operadores son:

&& (y), (AND) *ejemplo: if (var1==0 && Var2>0) // indicando que la instrucción contenida en if solo se ejecuta si se cumplen las dos condiciones de los operadores comparativos.*

|| (o), (OR) *ejemplo: if (var1==0 || Var2>0) // indicando que la instrucción contenida en if se ejecuta si se cumplen por lo menos una de las dos condiciones.*

¡ (negación), (NOT) *ejemplo: if (¡var1) // indicando que es verdadero si el valor de var1 es 0.*

3.1.8.6 Operadores de Composición

Los operadores de composición realizan operaciones matemáticas sin definir dos veces una misma variable, los operadores son:

++ Incremento *ejemplo: var1++ // incrementa en uno el valor de la variable.*

-- Disminución *ejemplo: var1-- // decremento en uno el valor de la variable.*

+= Composición suma *ejemplo: var1 += var2 // equivale a var1 = var1 + var2*

-= Composición Resta *ejemplo: var1 -= var2 // equivale a var1 = var1 - var2*

****= Composición Multiplicación*** *ejemplo: var1 *= var2 // equivale a var1 = var1 * var2*

/= Composición División *ejemplo: var1 /= var2 // equivale a var1 = var1 / var2*

3.1.8.7 Estructuras de Control

Las Estructuras de control[75] sirven para controlar la ejecución de instrucciones en el programa, todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Las estructuras que se utilizan en Arduino son:

If (Comparador Si)

Es una sentencia condicional, que se utiliza para saber si una condición se cumple, a través de los operadores comparativos, ejemplo:

```
If (var1 == 3) // Si la var1 es igual a 3 entonces  haga.
        {
                Haga
        }
```

If…else (Comparador si… Sino)

Esta sentencia nos da la opción de que si un condicional no se cumple se ejecutara otra instrucción, ejemplo:

```
If (var1 == 3) // Si la var1 es igual a 3 entonces  haga.
        {
                Haga A
        }  else //sino
        {
                Haga B
        }
```

[75] Tomado de http://es.wikipedia.org/wiki/Estructuras_de_control

For (Contador)

Es una declaración usada para repetir sentencias un determinado número de veces, comprende tres partes encerradas en paréntesis: inicialización de la variable (esta debe ser entera), condición, incremento o disminución separados por punto y coma (;) y encierra las sentencias entre llaves { }, ejemplo:

```
For (int i = 0; i< 10; i++) {
        Haga
        }
```

Es importante saber que 0 (Cero) es contado como una posición, eso quiere decir que el ciclo se repetirá 10 veces.

Switch case (Comparador múltiple)

Switch es una sentencia, encargado de encontrar el valor de una variable en el valor de un case (caso) y así ejecutar instrucciones. Ejemplo:

```
switch (var1) // donde var1 es una variable de tipo entero {
   case 1:
     //haga, cuando var1 sea igual a 1
     break;
   case 2:
     //haga, cuando var1 sea igual a 2
     break;
   default:
     // Si nada coincide, ejecuta el "default"
     // El "default" es opcional
 }
```

While (Mientras que)

Es un condicional que se ejecutará repetidamente mientras que la sentencia sea verdadera, ejemplo:

```
Var1 = 30
While (var1 > 10) {
        //Haga
        Var1 --;
}
```

Do…while (Condicional hacer mientras que)

Este condicional trabaja igual que el condicional while, con la diferencia de que la comparación se realiza al final por lo tanto esta sentencia se ejecutara por lo menos una vez, ejemplo:

```
Var1 = 30
Do {
        // haga
        Var1 --;
    } While (var1 > 10);
```

Break (Salida)

El termino break es utilizado para salir de una estructura de control sin importar la sentencia inicial.

Continue (Continuación)

Dentro de un condicional iterativo, la sentencia continue omite el resto de sentencias y continua con el resto de las iteraciones.

Return (Devolver)

Return, devuelve un valor a la función que lo llama, ejemplo:

```
Int devvalor () // método que recoge el valor
{
        // haga
        Return valor;
}
```

3.1.9 Aprendiendo lenguaje de programación II: Funciones

Las funciones son bloques de código y sentencias, que solo se ejecutan cuando la función es llamada. Las funciones son creadas por un usuario con el fin de reducir el tamaño de un programa, así también el de evitar realizar tareas repetitivas.

Las funciones cuando son invocadas necesitan parámetros de entrada que se encuentran dentro de paréntesis ejemplo: Función(Parametro1,..,etc), cabe agregar que pueden haber funciones que no necesitan parámetros como setup() o loop().

Las funciones incluidas en el IDE de Arduino que podemos utilizar son las siguientes:

3.1.9.1 Funciones de Entradas/Salidas Digitales[76]

pinMode ( )

Configura el pin indicado para que se comporte como entrada o salida, esta función necesita de dos parámetros, el pin y el modo que puede ser INPUT (entrada) – OUTPUT (Salida), ejemplo:

pinMode (pin1, OUTPUT); // El pin 1 es una salida.

[76] Tomado de http://Arduino.cc/es/Reference/

digitalWrite ( )

Configura el voltaje del pin, HIGH (Encendido) equivale a 5 v o LOW (Apagado) que equivale a 0v, utiliza como parámetros el pin y el valor, ejemplo:

Si el pin 1 es configurado como salida entonces,

digitalWrite (pin1, HIGH); // El pin 1 se encenderá

digitalRead ( )

Lee el valor digital de un pin especificado y lo devuelve a una variable, ejemplo:

Var1 = digitalRead (1); // Lee el valor digital del pin 1

// Devolverá HIGH o LOW y se lo asignara la variable var1

3.1.9.2 Funciones de Entradas/Salidas Analógicas[77]

analogReference ( )

Configura el voltaje de referencia usado por la entrada analógica, el tipo de referencia puede ser:

DEFAULT: Es el valor de referencia analógico que viene por defecto que de 5 voltios en placas Arduino de y de 3.3 voltios en placas Arduino que funcionen con 3.3 voltios.

INTERNAL: Es una referencia de tensión interna de 1.1 voltios en el ATmega168 o ATmega328 y de 2.56 voltios en el ATmega8.

[77] Tomado de http://Arduino.cc/es/Reference/

EXTERNAL: Se usará una tensión de referencia externa que tendrá que ser conectada al pin AREF.

analogRead ( )

Lee el valor de tensión de un pin definido como entrada analógica. Los pines analógicos a diferencia de los digitales su modo siempre es de entrada (INPUT).
Ejemplo:
Valor = analogRead (pin); // Asigna a valor lo que lee.

analogWrite ( )

Esta función lee un valor analógico PWM (Modulación por ancho de pulso (***pulse-width modulation***)) a uno de los pin´s de Arduino marcados como PWM. Necesita dos parámetros, el pin en el cual se quiere generar la señal PWM y el ciclo de trabajo que oscila entre 0 y 255, ejemplo:
Valor = 255;
analogWrite (pin1, valor);

3.1.9.3 Funciones de Entradas/Salidas Avanzadas[78]

Tone ( )

Esta función genera una onda cuadrada de la frecuencia especificada en un pin, sus parámetros son: el pin en el que se genera el tono, la frecuencia del tono en hercios y la duración del tono en milisegundos.
tone (pin1, frecuencia, duración); // La duración es opcional.

[78] Tomado de http://Arduino.cc/es/Reference/

noTone ( )

Esta función detiene la generación de la señal cuadrada que se activa al hacer uso de la función tone(). Esta función no tiene efecto si no se está generando ningún tono.
noTone (pin1); // Se detiene la señal del pin1

shiftOut ( )[79]

Desplaza un byte de datos bit a bit. Empieza desde el bit más significante (el de más a la izquierda) o el menos significante (el más a la derecha). Cada bit se escribe siguiendo su turno en un pin de datos, después de conmutar un pin de reloj (señal de reloj) para indicar que el bit está disponible, sus parámetros son:

pinDatos: el pin en el cual extraer cada bit (*int*)

pinReloj: el pin que hay que conmutar cada vez que a un pinDatos le ha sido enviado el valor correcto (int).

ordenBits: en qué orden desplazar los bits; si hacia el MSBFIRST (bit más significante primero) o hacia el LSBFIRST (bit menos significante primero).

valor: los datos que rotar. (*byte*)

Ejemplo: shiftOut(pinDatos, pinReloj, ordenBits, valor);

El pinDatos y pinReloj deben estar ya configurados como salida con una llamada previa a pinMode().

[79] Tomado de http://arduino.cc/es/Reference/ShiftOut

pulseIn ( )

Esta función lee el pulso en un pin, ya sea alto (HIGH) o bajo (LOW), utiliza como parámetros el pin del cual se lee la pulsación, el tipo de pulso y el tiempo en microsegundos es opcional, ejemplo:
pulseIn (pin, tipo de pulso, tiempo);

3.1.9.4 Tiempo[80]

Millis ( )

Esta función devuelve el numero de milisegundos transcurridos desde el inicio del programa en Arduino o la última vez que se pulso el botón reset hasta el momento actual, esta función no necesita parámetros.
Valor = millis ( );

Micros ( )

Esta función devuelve el numero de microsegundos transcurridos desde el inicio del programa en Arduino esta función no necesita parámetros.
Valor = micros ();

Delay ( )

Esta función detiene el programa la cantidad de milisegundos que se le indique en el parámetro, ejemplo:
Delay (3000); // Espera 3 segundos.

[80] Tomado de Arduino.cc/es/Reference/

delayMicroseconds ( )

Esta función detiene el programa la cantidad de microsegundos que se le indique en el parámetro, ejemplo:
DelayMicroseconds (300); //Espera 300 microsegundos.

3.1.9.5 Matemáticas[81]

min( )

Calcula el mínimo de dos números de cualquier tipo de dato, sus parámetros son dos números, ejemplo:

Valor = min (dato1, dato2); // Si dato1 es menor que dato2 a valor se le asignara dato1, y si dato1 es mayor que dato2 a valor se le asignara dato2.

max()

Calcula el máximo de dos números de cualquier tipo de dato, sus parámetros son dos números, ejemplo:

Valor = max (dato1, dato2); // Si dato1 es menor que dato2 a valor se le asignara dato2, y si dato1 es mayor que dato2 a valor se le asignara dato1.

abs( )

Esta función calcula el valor absoluto de un número, devuelve el número si este es mayor que cero, y devuelve el numero negativo si este es menor que cero.

Valor = abs (num);

[81] Tomado de Arduino.cc/es/Reference/

constrain( )

Esta función restringe un numero a un rango definido, sus parámetros son tres numero: el numero a restringir, y los dos números del rango, ejemplo:

Valor = Constrain (num, 20, 40); // Se limita a número entre 20 y 40, si numero el numero esta en el rango, es asignado a valor, si el numero es mayor que el rango a valor se le asigna 40 y si el número es menor que el rango a valor se le asignará 20.

Map ( ) [82]

Esta función Re-mapea un número desde un rango hacia otro. Esto significa que, un valor (*value*) con respecto al rango *fromLow-fromHight* será mapeado al rango *toLow-toHigh*, donde:

value: el número (valor) a mapear.
fromLow: el límite inferior del rango actual del valor.
fromHigh: el límite superior del rango actual del valor.
toLow: límite inferior del rango deseado.
toHigh: límite superior del rango deseado.
Ejemplo: val = map(val, 0, 1023, 0, 255);

pow( )

Esta función eleva un número al exponente deseado, sus parámetros son dos. Base y exponente, ejemplo:

valor = pow (2,3) // A valor se le asignara el resultado de 2 elevado a la 3, por lo tanto valor será 8.

sq( )

[82] Tomado de http://arduino.cc/es/Reference/Map

Esta función eleva un número al cuadrado, ejemplo:

Valor = sq (2); // Valor será 4.

sqrt()

Esta función determina la raíz cuadrada de un número, ejemplo:

Valor = sqrt (25); // Valor será igual a 5.

3.1.9.6 Trigonometría

Sin ( )

Calcula el seno de un ángulow en radianes, ejemplo:
Valor = sin (90); // A valor le será asignado 1.

Cos ( )

Calcula el coseno de un Angulo en radianes, ejemplo:
Valor = cos (180); // A valor le será asignado -1.

Tan ( )

Calcula la tangente de un Angulo en radianes, ejemplo:
Valor = tan (45); // A valor le será asignado 1.

3.1.10 Aprendiendo lenguaje de programación III: Diccionario de palabras reservadas del IDE de Arduino

Las palabras reservadas del IDE de Arduino[83] son constante, variables y funciones que se definen en el lenguaje de programación de Arduino. No se deben usar estas palabras clave para nombres de variables, algunas de estas palabras tienen la propiedad de cambiar de color cuando las escribimos en el IDE.

3.1.10.1 Constantes[84]

HIGH: Nivel alto
LOW: Nivel Bajo
INPUT: Entrada
OUTPUT: Salida
SERIAL: Puerto serie, comunicación entre la placa Arduino y el ordenador.
DISPLAY: Visualización
PI: Pi equivale a 3,14
HALF_PI: equivale a 1,570796…
TWO_PI: equivale a 6.28318…
QUARTER_PI: equivale a 0.7853982
LSBFIRST: Bit menos significante primero
MSBFIRST: Bit mas significante primero
CHANGE: Para disparar la interrupción en cualquier momento que el pin cambie de valor.
FALLING: Para cuando el pin cambie de valor alto a bajo.
RISING: Para disparar la interrupción cuando el pin cambie de valor alto a bajo.
False: falso
True: Verdadero
Null: Nulo

[83] Tomado de http://arduino.cc/playground/
[84] Tomado de http://arduino.cc/es/Reference

3.1.10.2 Variables de designación de puertos y constantes[85]

DDRB: dirección del registro de datos del puerto B

PINB: Registro del pin de entrada del puerto B

PORTB: Registro de datos del puerto B

PB0 - PB1 - PB2 - PB3 – PB4 - PB5 - PB6 - PB7

DDRC: dirección del registro de datos del puerto C

PINC: Registro del pin de entrada del puerto C

PORTC: Registro de datos del puerto C

PC0 - PC1 - PC2 - PC3 - PC4 - PC5 - PC6 - PC7

DDRD: dirección del registro de datos del puerto D

PIND: Registro del pin de entrada del puerto D

PORTD: Registro de datos del puerto D

PD0 - PD1 - PD2 - PD3 - PD4 - PD5 - PD6 - PD7

Private: Privado

Protected: Protegido

Public: Publico

Return: Devolver

Short: Corto (Corto circuito)

Signed: Signo

Static: Estático

Switch: Interruptor

Throw: Arrojar

Try: Tratar de

Unsigned: Sin signo

Void: Declara funciones

[85] Tomado de http://arduino.cc/es/Reference

3.1.10.3 Otras Variables[86]

Abs: Absoluto (Función, calcula el valor absoluto de un numero)

Acos : Arcocoseno

Asin: Arcoseno

Atan: Arcotangente

atan2: Arcotangente de dos parametros

case: Caso

ceil: Menor entero no menor que el parametro

constrain: Restringir a

cos: Coseno

delay: Espera (tiempo)

loop: Funcion nucleo

max: Maximo

millis: Milisegundos

min: minimo

new: nuevo

Setup: Recoge la configuración (Solo se invoca al iniciar el programa)

Sin: Seno

Sq: Cuadrado

Sqrt:Raiz cuadrada

Tan: Tangente

This: Este

While: Condicional Mientra que

Begin: Pne vvelocidad a la comunicación

Read: Leer

Print: Imprime

Write: Escribe

Peek: Funcion que devuelve el primer byte de entrada de datos

[86] Tomado de http://arduino.cc/es/Reference

Flush: Función que vacía el buffer de entrada de datos en serie

Println: Imprime

Available: Función que devuelve el numero de bytes disponibles para ser leídos por el puerto serie

digitalWrite: Escritura del pin digital

digitalRead: Lectura del pin digital

PinMode: Modo del pin

analogRead: Lectura del pin análogo

analogWrite: Escritura del pin análogo

attachInterrupts: Interrupción externa

detachInterrupts: Apaga la interrupción

tone: Tono de un pin

noTone: Detiene el tono del pin

pulseIn: Lee un pulso

shiftOut: Cambio

map: Función que remapea una rango de valores a otro

pow: Eleva un valor dado al exponente

max: Máximo

min: Mínimo

lowByte: Extrae un byte de derecha

highByte: Extrae un byte de izquierda

bitRead: Lee un bit

bitWrite: Escribe un bit

bitSet: Cambia un bit

bitClear: Borra un bit

bit: Calcula el valor de un bit

randomSeed: Inicializa el generador de números

random: Genera números

Class: Clase

Default

Do: Hacer

esp

delayMicroseconds

floor

log

3.1.10.4 Tipos de datos[87]

Boolean

Solo contiene valores que puede ser TRUE o FALSE (Verdadero o falso).

Byte

Numero entero de ocho bits.

Char

Almacena un valor de caracter (letra) entre comillas simples.

Double

Numero decimal.

Long

Numero entero de tamaño extendido.

Float

Numero decimal.

Int (integer)

Numero entero.

[87] Tomado de http://arduino.cc/es/Reference

String

Tipo de caracteres extendido entre comillas.

Array

Matriz de variables accedidas mediante un número de índice.

Unsigned Int

Solo almacena enteros positivos.

Unsigned Long

Solo almacena enteros positivos de tamaño extendido.

3.1.10.5 Funciones de Conversión

Char()

Convierte un valor de cualquier tipo a un tipo de dato char.

byte()

Convierte un valor de cualquier tipo a un tipo de dato byte.

int()

Convierte cualquier valor de tipo numérico a un tipo de dato entero.

long()

Convierte un valor de cualquier tipo a un tipo de dato long.

float()

Convierte un valor de cualquier tipo a un tipo de dato float.

3.1.11 Aprendiendo variables

Una variable es una posición en memoria en la cual se pueden guardar datos, debe estar inicializada y declarada como el tipo de dato que se almacenará y opcionalmente asignada a un determinado valor, puede nombrarse como nos guste siempre y cuando no sea una palabra reservada del IDE de Arduino.

Ejemplo: Declaremos una variable llamada Num y una de llamada letra

```
Int Num;

// Se declaro la variable de tipo entero

string letra;

// Se declaro la variable de tipo carácter
```

Ejemplo: Declaremos una variable llamada Num asignándole un determinado valor

```
Int Num = 0;

// Se declaro la variable de tipo entero
// La variable tiene un valor inicial de 0
```

Ejemplo: Declaremos una variable llamada letra asignándole un determinado valor

```
string letra = “a”;

// Se declaro la variable de tipo string
// La variable tiene un dato inicial “a”
```

Ejemplo: Después de declarada la variable Num se le asignara otro valor.

```
Num = 4;
// La variable de tipo entero se le pueden asignar valores, sin importar que no tenga valor inicial
```

Ejemplo: Después de declarada la variable Num y asignarle un valor de 0, se le asignara otro valor.

```
Num = 4;
// La variable de tipo entero se le pueden asignar valores, sin importar que dato tenga como valor inicial.
```

Las variables deben tomar nombres descriptivos, para hacer el código más legible. Nombres de variables pueden ser “contactoSensor” o “pulsador”, para ayudar al programador y a cualquier otra persona a leer el código y entender lo que representa la Variable.

Una vez que una variable ha sido asignada, o re-asignada, como vimos en los ejemplos anteriores podemos probar su valor para ver si cumple ciertas condiciones (instrucciones if...) o puede utilizar directamente su valor.

Vamos a realizar varios ejercicios en nuestro IDE de Arduino y compilamos para que no existan errores.

Prueba 1.

Si la variable “Num” es inferior a 100, si es cierto se le asigna el valor 100 a “Num” y, a continuación, establece un retardo (delay) utilizando como valor “Num” que ahora será como mínimo de valor 100:

Num = 90

Si Num < 100, entonces

haga

Num = 100

Retardo (Num)

fin

En nuestro IDE seria

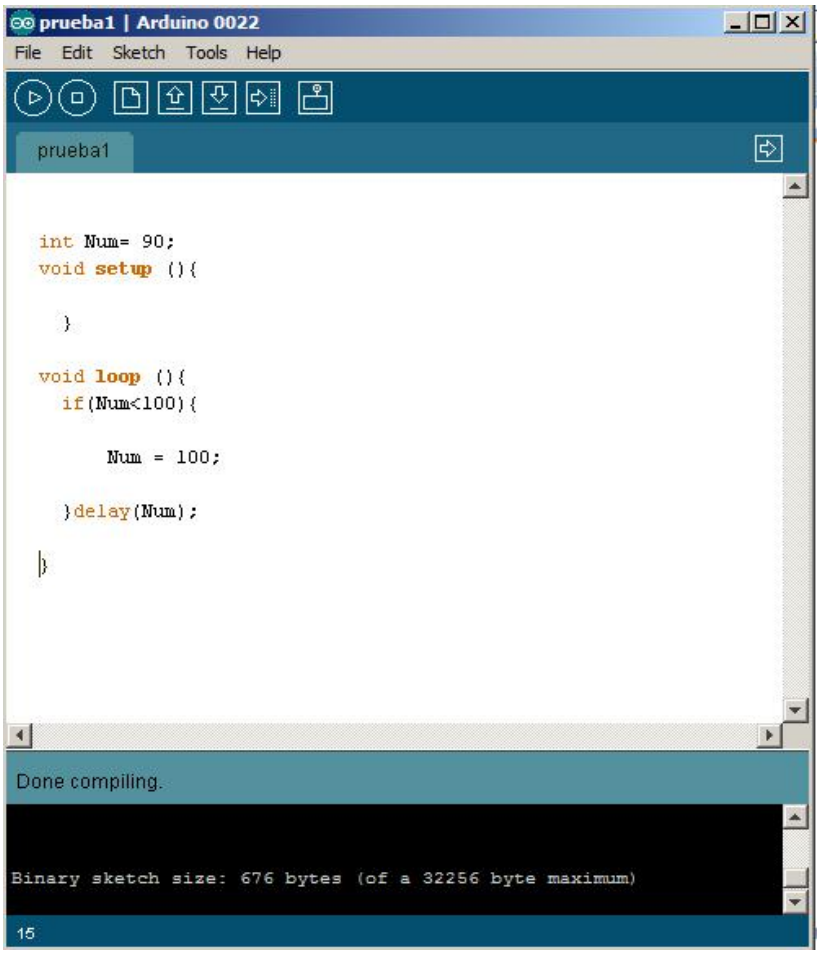


Figura 103. Programando ciclos en el IDE de Arduino parte 1

Done compiling, No hay errores.

Prueba 2.

Mientras que la variable “Num” sea mayor a 10 o menor a 50, se le asigna un incremento a la variable “Num” y, a continuación, a Num se le asignara un valor inicial de 12:

Num = 12
Mientras que Num sea mayor a 10 o menor de 50, entonces
Haga
Num = num + 1
Fin

En nuestro IDE seria

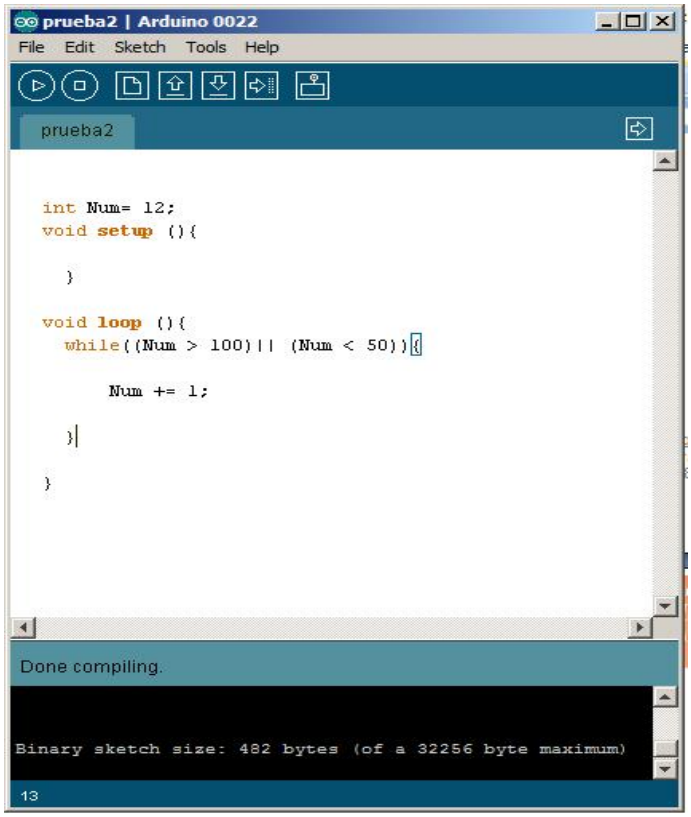


Figura 104. Programando ciclos en el IDE de Arduino parte 2

Prueba 3.

Con la placa Arduino, trabajaremos la variable pin con una asignación del pin 13 e insertaremos un led en el pin 12.

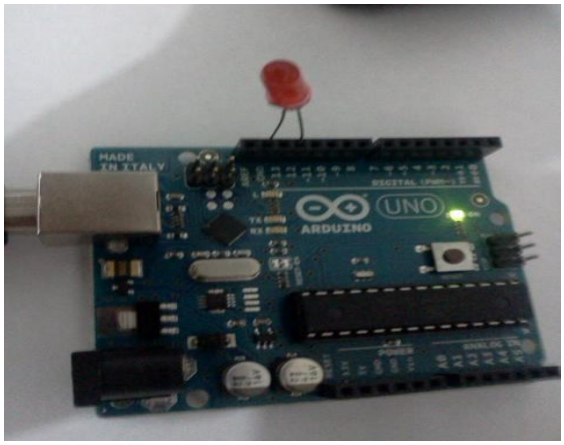


Figura 105. Configurando un led

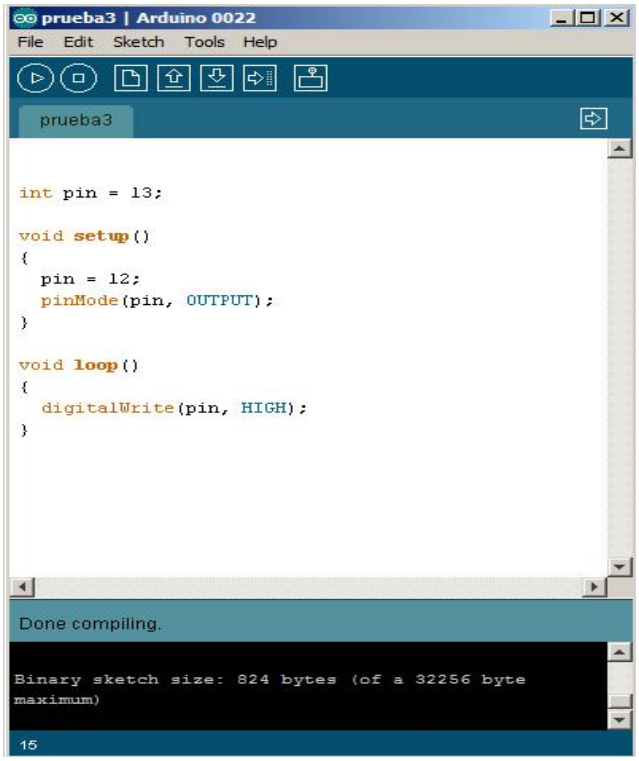


Figura 106. Configurando el led en el pin 12

Este programa se sube a la placa y debe encenderse como se muestra a continuación:

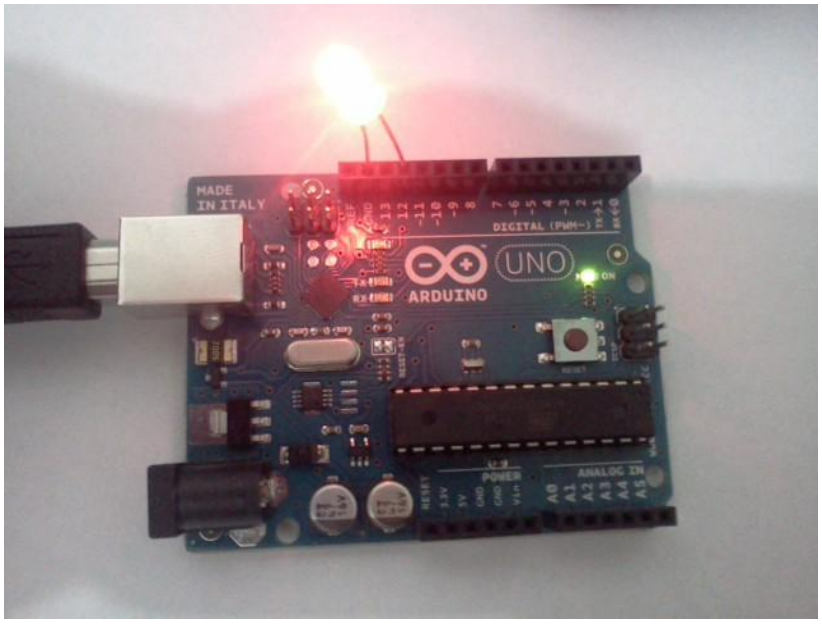


Figura 107. Ejecución del programa en la placa

3.1.12 Apaga y prende un led N veces

Observación: Esta práctica ha sido recogida de la página oficial de Arduino, http://www.arduino.cc

Como su nombre lo indica, esta práctica se trata de prender y apagar un número determinado de veces un led, un número que se indicará por medio del serial monitor que encontramos en nuestra IDE de Arduino.

Figura 108. Icono del monitor serial

Para el programa se utilizaran las funciones del puerto serie[88]:

- ✓ **Serial.begin**(rate): Abre un puerto serie y especifica la velocidad de transmisión. La velocidad típica para comunicación en el ordenador es de 9600, aunque se pueden soportar otras velocidades.

- ✓ **Serial.println**(data): Imprime los datos al puerto serie seguido por un retorno de línea automatico. Este comando tiene la misma forma que Serial.print( ), la diferencia es que Serial.Println tiene el salto de línea al final. Este comando puede ser utilizado para la depuración de programas. Para ello se pueden enviar mensajes de depuración y valores de variables por el puerto serie.

Para tener en cuenta: *El puerto serie y el Serial Monitor deben estar configurados a la misma velocidad.*

- ✓ Serial.read( ): Lee o captura un byte osea un carácter desde el puerto serie. Devuelve -1 si no hay ningún carácter en el puerto serie.

- ✓ Serial.available( ): Devuelve el número de caracteres disponibles para leer desde el puerto serie.

Esquema:

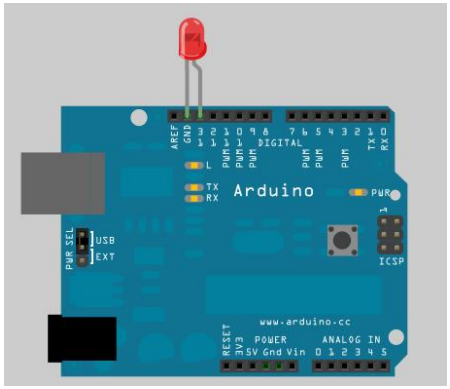


Figura 109. Esquema de configuración de un led

[88] Tomado de http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf

En nuestro IDE de Arduino:

```
int ledPin = 13; // selecciona el pin para el led

int val = 0; // variable que almacena el valor leído del puerto

void setup() {

pinMode(ledPin,OUTPUT); // declaración del modo del  led como salida
Serial.begin(19200); // conecta con el puerto serie a la velocidad de 19200
Serial.println("Bienvenido al monitor serial donde veras como se controla un
led");

}

void loop () {

val = Serial.read(); // lee el numero del puerto (una sola cifra)
//si el valor leído es un solo digito se ejecuta el programa

if (val > '0' && val <= '9' ) {
val = val - '0'; // convierte el carácter leído en un numero

for(int i=0; i<val; i++) {

Serial.println("Led Encendido");
digitalWrite(ledPin,HIGH); // enciende el LED
delay(150); // Espera 150 milisegundos
Serial.println(“Led Apagado”);
digitalWrite(ledPin, LOW); // apaga el LED

delay(150); // espera
}
}
}
```

Cuadro de Código fuente 4

Al subir nuestro programa a la Board obtendremos lo siguiente:

Insertamos un numero y le damos send (enviar).

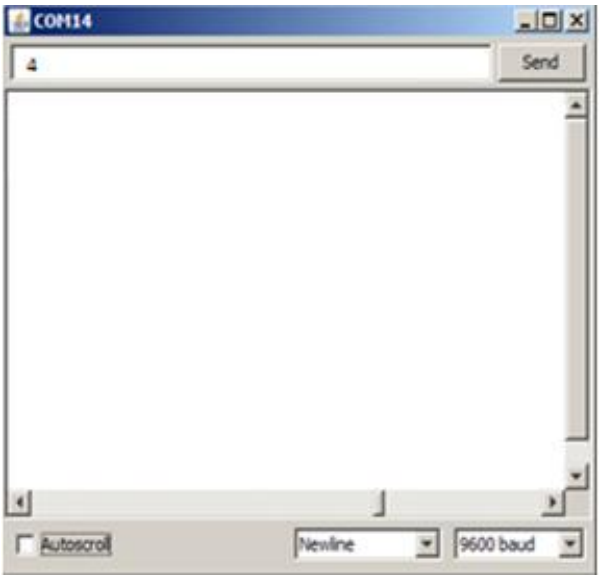


Figura 110. Serial monitor en ejecución

Así nos queda:

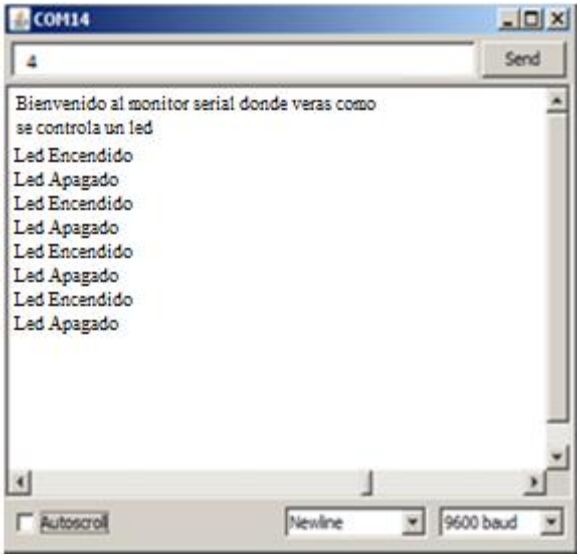


Figura 111. Respuesta del serial monitor

3.1.13 Semáforo

Figura 112. Semáforo [89]

Un semáforo es una señal de transito, utilizado para regular el tráfico; normalmente los vemos situados en las calles, pasos de peatones y en otros lugares.

Los semáforos[90] han ido evolucionando con el paso del tiempo y actualmente y debido a su rentabilidad, se están utilizando lámparas a LED para la señalización

Figura 113. Lámpara de leds[91]

[89] Imagen de educacionvialsvp.blogspot.com
[90] Tomado de http://es.wikipedia.org/wiki/Semaforo
[91] Imagen de solostocks.com

luminosa, puesto que las lámparas de LED utilizan sólo 10% de la energía consumida por las lámparas incandescentes, tienen una vida estimada 50 veces superior, y por tanto generan importantes ahorros de energía y de mantenimiento, satisfaciendo el objetivo de conseguir una mayor fiabilidad y seguridad pública.

Figura 114. Semáforo de leds[92]

Entre las mayores ventajas que tienen las señales luminosas con LED figuran:

- ✓ Muy bajo consumo y por tanto ahorran energía.
- ✓ Mayor vida útil de las lámparas.
- ✓ Mínimo mantenimiento.
- ✓ Respeto por el medio ambiente.
- ✓ Simple recambio.
- ✓ Unidad óptica a prueba de luz solar y Alto contraste con luz solar.
- ✓ Señalización luminosa uniforme.
- ✓ Evita el fundido de las luces, al estar formadas estas por una matriz de diodos, por lo que en ese caso solo lo harán unos cuantos diodos y no todo el conjunto, de forma que el semáforo nunca se apagará por un fallo de este tipo.
- ✓ Mayor seguridad vial.
- ✓ Su bajo consumo permite que funcionen automáticamente mediante una batería durante cierto tiempo.

[92] Imagen de infierno-verde.blogspot.com

✓ Precaución a los peatones.

Un semáforo para regular el tráfico de vehículos consta de tres lámparas de distinto color:

Rojo

Figura 115. Luz de leds rojos[93]

Significa que los vehículos deben detenerse.

Amarillo

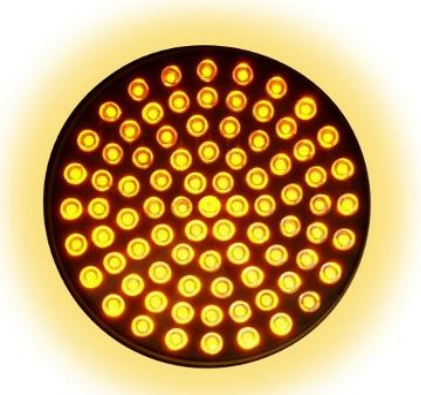


Figura 116. Luz de leds amarillos[94]

Significa precaución, y nos indica un cambio de luz: rojo – verde o verde – Rojo.

[93] Imagen de circulaseguro.com
[94] Imagen de http://www.trafictec.com/lampara_20.php

Verde

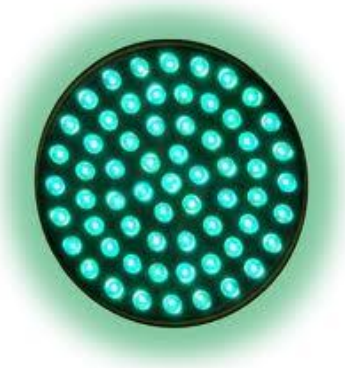


Figura 117. Luz de leds amarillos[95]

Significa que los vehículos pueden seguir.

Practica

Realizaremos un semáforo sencillo con nuestra placa Arduino, una protoboard y unos cuantos led´s.

Montaje

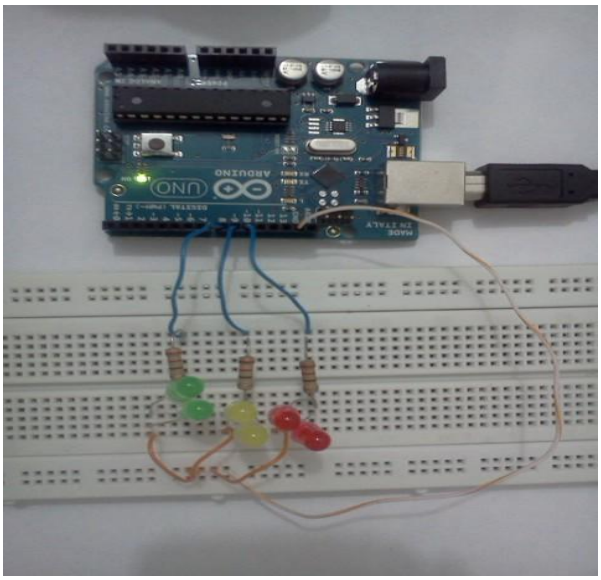


Figura 118. Montaje en protoboard

[95] Imagen de http://www.trafictec.com/lampara_20.php

En nuestro IDE de Arduino:

```
void setup(){

  pinMode(10,OUTPUT); //Color Rojo
  pinMode(9,OUTPUT); //Color Amarillo
  pinMode(8,OUTPUT); //Color Verde
}
 void loop(){

   digitalWrite(10,HIGH);
   delay(5000);
   digitalWrite(9,HIGH);
   delay (500);
   digitalWrite(10,LOW);

   delay(2000);
   digitalWrite(8,HIGH);
   delay(500);
   digitalWrite(9,LOW);

   delay(5000);
   digitalWrite(9,HIGH);
   delay(500);
   digitalWrite(8,LOW);

   delay(2000);
   digitalWrite(9,LOW);

 }
```

Cuadro de Código fuente 5

Cada pin digital tiene dos led´s del mismo color, el pin 10 tiene 2 led´s de color rojo, el pin 9 tiene 2 led´s de color amarillo y el pin 8 tiene dos led´s de color verde; todos los pines están a modo de salida.

En el void loop(), se le da la orden a cada pin para que los led´s enciendan y apaguen de acuerdo a un determinado retardo definido en la función delay().

Guardamos y subimos el programa a la placa y obtenemos lo siguiente:

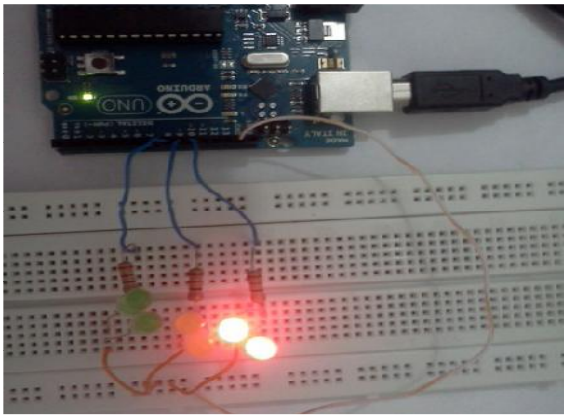


Figura 119. Secuencia semáforo con Arduino parte 1

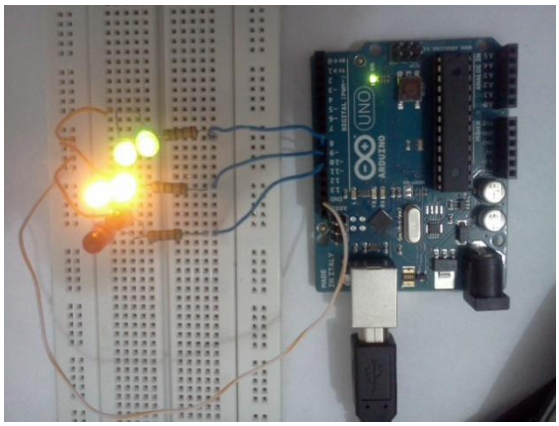


Figura 120. Secuencia semáforo con Arduino parte 2

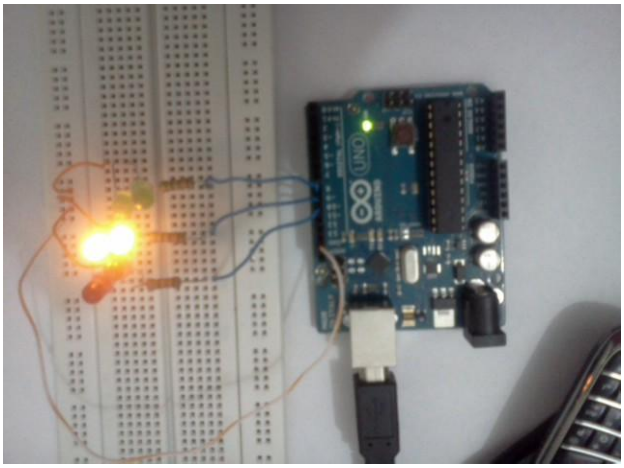


Figura 121. Secuencia semáforo con Arduino parte 3

3.1.14 Manejando sensores

3.1.14.1 Sensor de luz

Como sabemos un sensor de luz es más conocido como LDR que es una resistencia variable, esta varía su valor dependiendo de la cantidad de luz que incide sobre su superficie. Cuanta más intensidad de luz incide en la superficie de la LDR menor será su resistencia y cuanta menos luz incida mayor será la resistencia. Suelen ser utilizados como sensores de luz ambiental o como una fotocélula que activa un determinado proceso en ausencia o presencia de luz.

Practica:

Se encenderá y apagara un led dependiendo de la cantidad de luz que almacene el sensor.

A mayor cantidad de luz menor parpadeo del led, y a menor cantidad de luz mayor parpadeo del led.

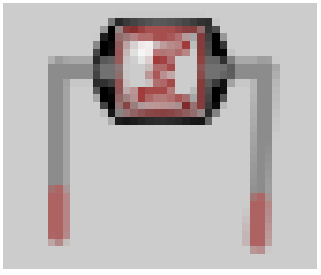


Figura 122. Sensor de luz

Esquema:

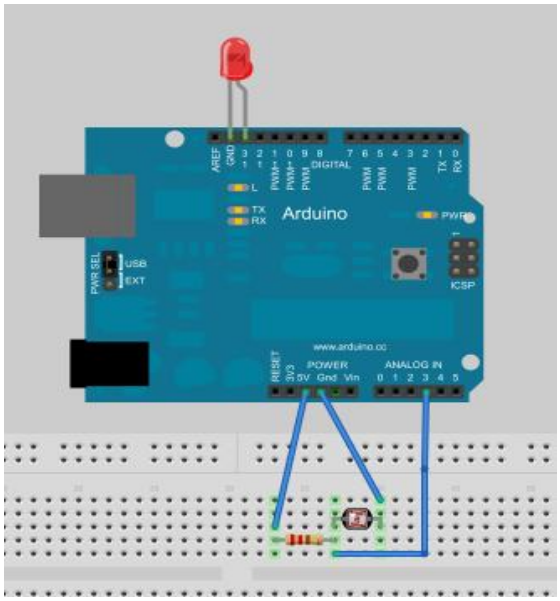


Figura 123. Montaje del sensor de luz con Arduino

En nuestro IDE de Arduino

```
int LightPin = 3; // selecciona el pin analógico de entrada para el sensor de luz
int ledPin = 13; // selecciona el pin digital para el led
int val = 0; // variable para almacenar el valor capturado desde el sensor
void setup() {
pinMode(ledPin, OUTPUT); // declara el ledPin en modo salida
}
void loop() {
val = analogRead(LightPin); //lee el valor del sensor
digitalWrite(ledPin, HIGH); // enciende el led
delay(val); // detiene el programa por un tiempo
digitalWrite(ledPin, LOW); // apaga el LED
delay(val); // detiene el programa por un tiempo
}
```

Cuadro de Código fuente 6

3.1.14.2 Sensor de temperatura

Un sensor de temperatura se compone de un termistor NTC, que es una resistencia variable, que varía su valor dependiendo de la temperatura ambiente. Cuanta más temperatura menor será su resistencia y cuanta menos temperatura mayor será la resistencia. Suelen ser utilizados en alarmas.

Practica:

Se medirá la temperatura con un NTC desde el PIN3 de entrada analógica y ver si este valor supera un valor dado de 500 (medida absoluta). Si supera este valor activará la salida digital PIN13 y si no la apagara. Además queremos que se muestre en el “Serial Monitor” del IDE Arduino el valor leído.

Figura 124. Sensor de Temperatura

Esquema:

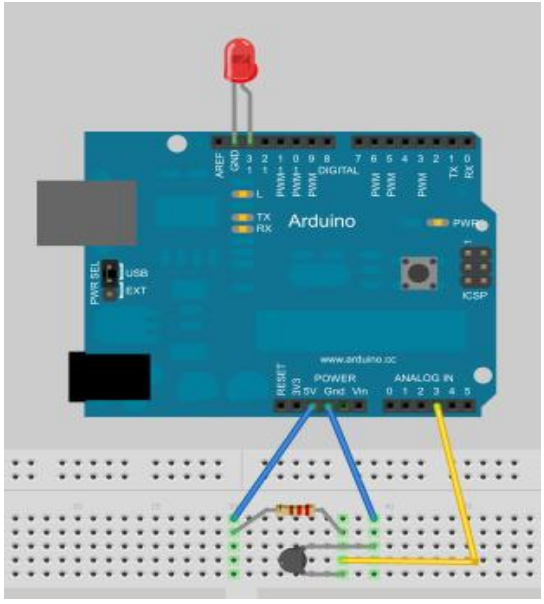


Figura 125. Montaje del sensor de temperatura con Arduino

En nuestro IDE de Arduino:

```
int led=13;
int ntc=3;
int medida=0;
int nivel=500; // Medida Absoluta

void setup(){

pinMode(led, OUTPUT);
pinMode(ntc, INPUT);
Serial.begin(9600);

}
void monitor(){
//procedimiento que envía al puerto serie, para ser leído en el monitor,
Integer.print(medida);
//el valor de la señal de la NTC en la entrada analógica
String.print(" ");
delay(100);
//para evitar saturar el puerto serie
}
void loop(){
medida = analogRead(ntc);
monitor();

if(medida>nivel){

//si la señal del sensor supera el nivel marcado:
digitalWrite(led, HIGH);
 //se enciende el led para avisar
}
else{
// si la señal está por debajo del nivel marcado

digitalWrite(led, LOW);
// El led estará apagado
}
}
```

Cuadro de Código fuente 7

3.1.15 Sensor de inclinación – Alarma

3.1.15.1 Sensor de Inclinación

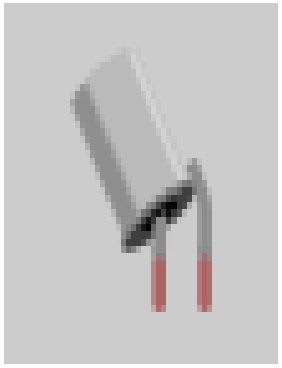


Figura 126. Sensor de inclinación

El sensor de inclinación es un componente que puede detectar la inclinación de un objeto. Sin embargo, no deja de ser un pulsador activado por un mecanismo físico diferente. Este tipo de sensor es la versión ecológica de un interruptor de mercurio.

Contiene una bola metálica en su interior que conmuta los dos pines del dispositivo de encendido a apagado, y viceversa, si el sensor llega a un cierto ángulo.

Practica

Se desea detectar si el sensor ha sido inclinado o no y enciende la luz en consecuencia.

Se debe tener en cuenta que al utilizar la "activación a nivel bajo" (mediante una resistencia de pulls-up) la entrada se encuentra a nivel bajo cuando el sensor se activa.

Esquema:

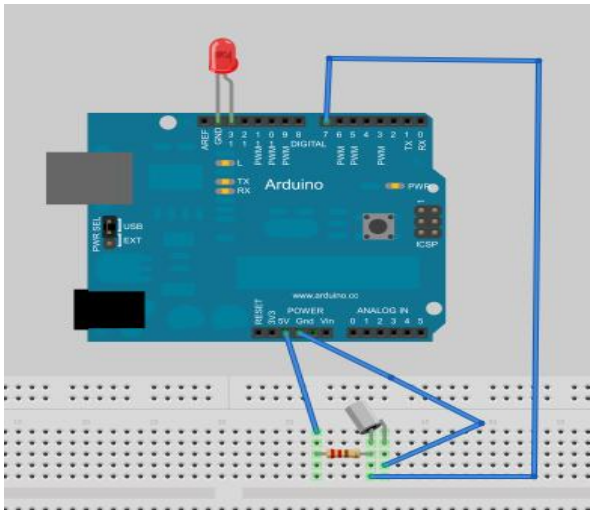


Figura 127. Montaje del sensor de inclinación con Arduino

En nuestro IDE de Arduino:

```
int ledPin = 13; // PIN del led
int inPin = 7; // PIN del pulsador
int value = 0; // Valor del pulsador
void setup() {
pinMode(ledPin, OUTPUT); // Inicializa el pin 13 como salida digital
pinMode(inPin, INPUT); // Inicializa el pin 7 como entrada digita
}
void loop() {
value = digitalRead(inPin);
 // Lee el valor de la entrada digital
digitalWrite(ledPin, value);
}
```

Cuadro de Código fuente 8

3.1.15.2 Alarma

Figura 128. Alarma

Practica

Cuando se “pulsa” el pulsador, se apaga y se enciende un led de forma intermitente en el pin 13.

Esquema:

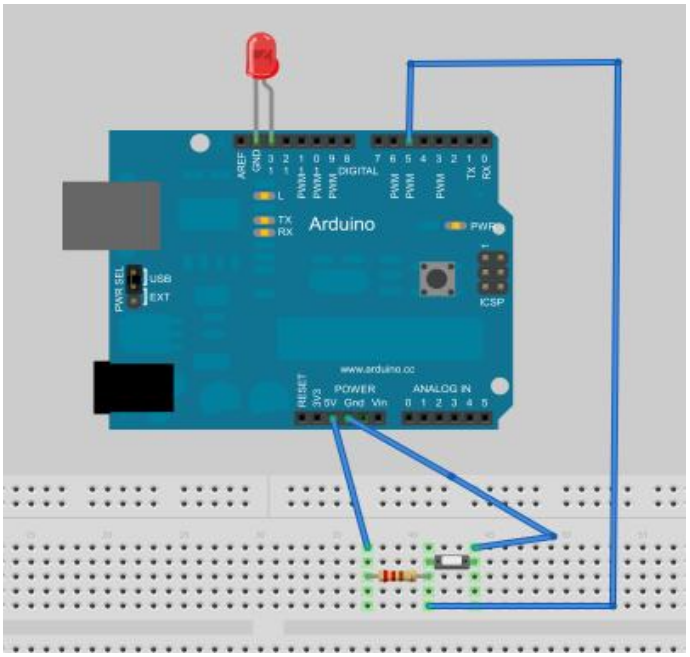


Figura 129. Montaje de la alarma con un pulsador

En nuestro IDE de Arduino:

```
int ledPin= 13; // Led en el pin 13
int inPin= 5; // Pin 5, pulsador
int val= 0;
void setup() {
pinMode(ledPin, OUTPUT);
// Se declara el led como salida
pinMode(inPin, INPUT);
// Se declara en pin5 como entrada, pulsador
}
void loop(){
val= digitalRead(inPin); // lee el  valor de entrada
if(val== HIGH) {
// Evalua si el botón está presionado
digitalWrite(ledPin, LOW);
// Apaga el led
}
else{    // parpadea el led
digitalWrite(ledPin, LOW);
delay(200);
digitalWrite(ledPin, HIGH);
delay(200);
}
}
```

Cuadro de Código fuente 9

3.1.16 Display de 7 segmentos

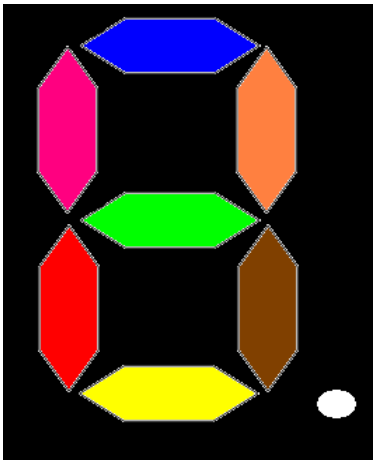


Figura 130. Display de 7 segmentos

El display de siete segmentos es un dispositivo bastante simple. En realidad, es una pantalla de 8 leds (el punto decimal es el octavo). Puede ser colocado de manera que las diferentes combinaciones se puedan utilizar para hacer dígitos numéricos, y algunas letras.

Los segmentos son denominados convencionalmente de la “a” a la “g” y el punto decimal como “PD”, como se muestra en la figura:

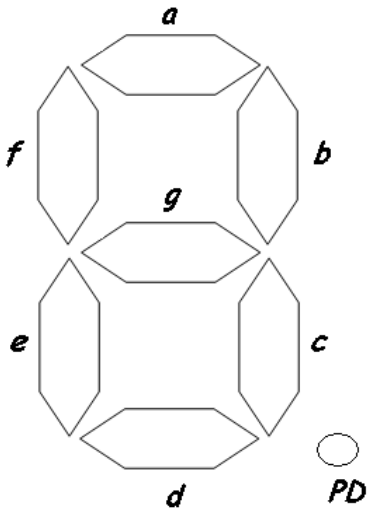


Figura 131. Segmentos del display

Los pines del display están distribuidos de la siguiente manera:

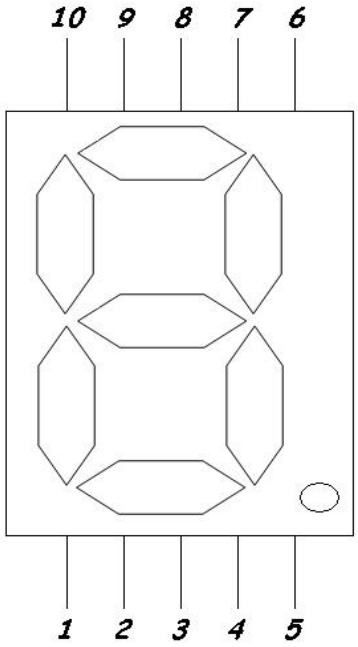


Figura 132. Pines del display

Donde, cada segmento enciende dependiendo del pin del display:

Pin del Display	Segmento
1	e
2	d
3	Tierra
4	c
5	PD
6	b
7	a
8	Tierra
9	f
10	g

Tabla 4. Segmentos de pines de un display

Practica:

Utilizando nuestro display de siete segmentos, realizaremos una cuenta regresiva del número nueve “9” al número cero “0”, tomando en cuenta lo siguiente:

Pin del Display	Segmento	Pin de la placa Arduino
1	e	9
2	d	8
4	c	7
5	PD	6
6	b	5
7	a	4
9	f	3
10	g	2

Tabla 5. Pines del display en la placa Arduino

Esquema:

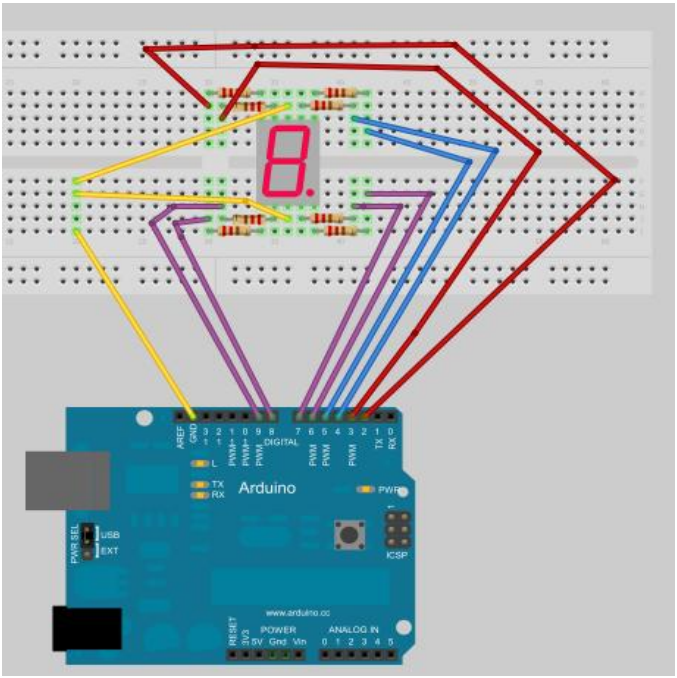


Figura 133. Montaje del display de 7 segmentos

Para tener una idea clara de lo que se va a realizar, tomaremos en cuenta lo siguiente:

Para realizar el número 9:

Segmentos encendidos: a, b, c, d, f, g.

Segmento apagado: e.

Para realizar el número 8:

Segmentos encendidos: a, b, c, d, e, f, g.

Segmento apagado: ninguno.

Para realizar el número 7:

Segmentos encendidos: a, b, c.

Segmento apagado: d, e, f, g.

Para realizar el número 6:

Segmentos encendidos: a, c, d, e, f, g.

Segmento apagado: b.

Para realizar el número 5:

Segmentos encendidos: a, c, d, f, g.

Segmento apagado: b, e.

Para realizar el número 4:

Segmentos encendidos: b, c, f, g.

Segmento apagado: a, d, e.

Para realizar el número 3:

Segmentos encendidos: a, b, c, d, g.

Segmento apagado: e, f.

Para realizar el número 2:

Segmentos encendidos: a, b, d, e, g.

Segmento apagado: c, f.

Para realizar el número 1:

Segmentos encendidos: b, c.

Segmento apagado: a, d, e, f, g.

Para realizar el número 0:

Segmentos encendidos: a, b, c, d, e, f.

Segmento apagado: g.

El PD se mantendrá encendido.

En nuestro IDE de Arduino:

```
void setup () {
// Todos nuestros pines son salidas digitales.
pinMode (2, OUTPUT);
pinMode (3, OUTPUT);
pinMode (4, OUTPUT);
pinMode (5, OUTPUT);
pinMode (6, OUTPUT);
pinMode (7, OUTPUT);
pinMode (8, OUTPUT);
pinMode (9, OUTPUT);
digitalWrite(6, HIGH); // Nuestro PD estará encendido
}
void loop () {
// Número 9
```

Cuadro de código fuente 10

```
digitalWrite (2, HIGH);
digitalWrite (3, HIGH);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, LOW);
delay (1000);  // Espera un segundo para cambiar al otro número.

// Número 8
digitalWrite (2, HIGH);
digitalWrite (3, HIGH);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, HIGH);
delay (1000);


// Número 7
digitalWrite (2, LOW);
digitalWrite (3, LOW);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, LOW);
digitalWrite (9, LOW);
delay (1000);


// Número 6
digitalWrite (2, HIGH);
digitalWrite (3, HIGH);
digitalWrite (4, HIGH);
digitalWrite (5, LOW);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, HIGH);
delay (1000);


// Número 5
digitalWrite (2, HIGH);
digitalWrite (3, HIGH);
digitalWrite (4, HIGH);
```

Cuadro 10 (Continuación)

```
digitalWrite (5, LOW);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, LOW);
delay (1000);

// Número 4
digitalWrite (2, HIGH);
digitalWrite (3, HIGH);
digitalWrite (4, LOW);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, LOW);
digitalWrite (9, LOW);
delay (1000);

// Número 3
 digitalWrite (2, HIGH);
digitalWrite (3, LOW);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, LOW);
delay (1000);
// Número 2
 digitalWrite (2, HIGH);
digitalWrite (3, LOW);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, LOW);
digitalWrite (8, HIGH);
digitalWrite (9, HIGH);
delay (1000);

// Número 1
 digitalWrite (2, LOW);
digitalWrite (3, LOW);
digitalWrite (4, LOW);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, LOW);
digitalWrite (9, LOW);
delay (1000);
```

Cuadro 10 (Continuación)

```
// Número 0
digitalWrite (2, LOW);
digitalWrite (3, HIGH);
digitalWrite (4, HIGH);
digitalWrite (5, HIGH);
digitalWrite (7, HIGH);
digitalWrite (8, HIGH);
digitalWrite (9, HIGH);
delay (3000); // Espera 3 segundos para iniciar nuevamente la
cuenta regresiva.
```

Cuadro 10 (Continuación)

3.1.17 Toca tonos - Sensor fuerza

3.1.17.1 Tocando tonos desde el puerto serial[96]

En esta práctica, se reproducirán tonos a través de un parlante. Se aprovechara el pin digital 11, por donde se pueden reproducir señales PWM.

Los tonos pueden ser generados a partir de cualquier programa capaz de enviar valores ASCII (Código estándar para el intercambio de información) a través del puerto serial. Los caracteres ASCII serán enviados a través del "Serial Monitor" de nuestro IDE de Arduino.

Esquema

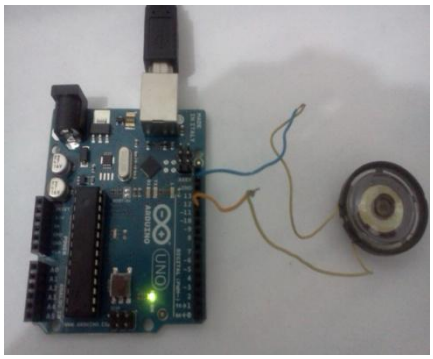


Figura 134. Tocando tonos con Arduino

[96] Este ejercicio fue tomado de la página principal de Arduino http://arduino.cc y fue modificado para esta cartilla.

Los tonos serán descritos de la siguiente manera:

Tecla	Frecuencia (Hz)	Periodo	Pulso Alto
a	523	1912	1915
b	493	2028	1700
c	440	2272	1519
d	392	2550	1432
e	349	2864	1275
f	329	3038	1136
g	294	3400	1014
h	261	3830	956

Tabla 6 Frecuencia, periodo y pulso alto de los tonos

Cualquier otra tecla producirá silencio.

En nuestro IDE de Arduino:

```
int ledPin = 13;

int parlante = 9;

byte names[] ={'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'}; // Definimos las letras que producirán cada tono

int tones[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; // Pulso alto de cada tono

byte val = 0;
int serByte = (-1);
int statePin = LOW;
int count = 0;

void setup() {
pinMode(ledPin, OUTPUT);

pinMode(parlante, OUTPUT); // Pin del parlante en modo de salida

Serial.begin(9600);

}
```

Cuadro de Código fuente 11

```
void loop() {


digitalWrite(parlante, LOW); // Parlante en silencio.
serByte = Serial.read();
if (serByte != (-1)) {
val = serByte;
Serial.print(val);
statePin = !statePin;
digitalWrite(ledPin, statePin);


}
for (count=0;count<=8;count++) {
if (names[count] == val) {
digitalWrite(parlante, HIGH);
delayMicroseconds(tones[count]);
digitalWrite(parlante, LOW);
delayMicroseconds(tones[count]);
}
else
digitalWrite(parlante, LOW);
}
}
```

Cuadro 11 (Continuación)

3.1.17.2 Sensor de fuerza[97]

Esta práctica trata de convertir un zumbador piezoeléctrico en un sensor de presión o fuerza, utilizando este como sensor de entrada en uno de los pines de entrada analógica de Arduino (pin 1), su salida se hará a través de un led ubicado en el pin 12 digital.

Esquema

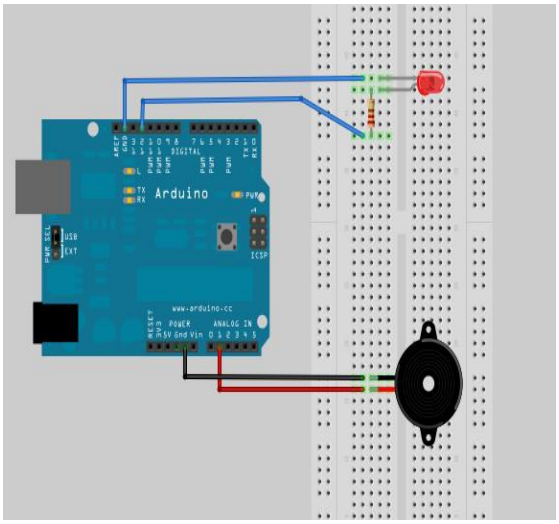


Figura 135. Montaje de zumbador piezoeléctrico

En nuestro IDE de Arduino:

```
int ledPin = 12;
int piezoPin = 1;
int valmin = 1; // Configura valor mínimo para que se encienda la salida en el pin 12
int val = 0; // variable que almacena el valor leído por el sensor
int t = 0; // valor del intervalo de medida
void setup() {
  pinMode(ledPin, OUTPUT);
Serial.begin(19200);
Serial.println("ready"); // indicador de espera
}
```

Cuadro de Código fuente 12

[97] Este ejercicio fue tomado de http://todbot.com y fue modificado para esta cartilla.

```
void loop() {
digitalWrite(ledPin,LOW); // indicador de reposo (esperando)
val = analogRead(piezoPin); // lectura de valor del piezoeléctrico
if( val > valmin ) {  // si el valor leído es superior al mínimo establecido
digitalWrite(ledPin, HIGH); // El led del pin 12 se enciende
t = 0;
while(analogRead(piezoPin) > valmin) {
t++;
}
if(t>100) {
Serial.print(" Intervalo de medida ");
Serial.println(t);

}
}
}
```

Cuadro 12 (Continuación)

3.1.18 Practicas con arduino I parte

Observación: *Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino* www.arduino.cc/

3.1.18.1 Luces Navideñas

Practica 1:

Esta práctica consta en hacer parpadear los led´s de los pines 2 al 7, en secuencia, para crear un efecto de luces navideñas.

Para aprender a programar secuencialmente se ha escogido esta práctica ya que solo se trabajara con la función digitalWrite, para cambiarle los estados a los pin´s, y así lograr un efecto parpadeante, y la función delay, para temporizador.

En el montaje se hace uso de 6 led´s y 6 resistencias, conectadas respectivamente una a una, además de tener puentes para hacer un solo polo a tierra GND.

Esquema:

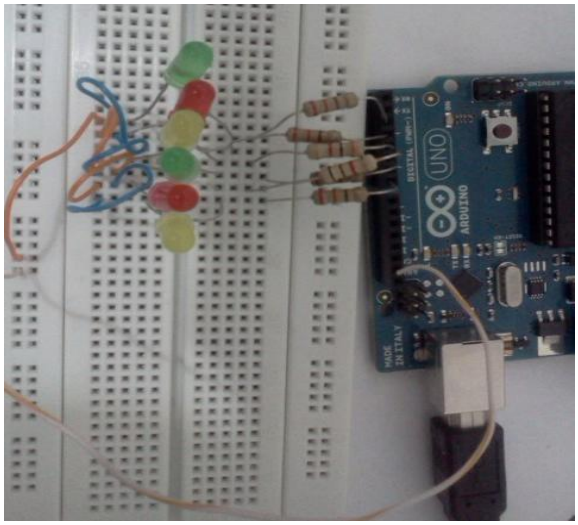


Figura 136. Montaje de las luces navideñas

En nuestro IDE de Arduino:

```
int pin2 = 2;
int pin3 = 3;
int pin4 = 4;
int pin5 = 5;
int pin6 = 6;
int pin7 = 7;
int timer = 100; // Temporizador

void setup(){
pinMode(pin2, OUTPUT); // Configuración de los pin´s como salida
pinMode(pin3, OUTPUT);
pinMode(pin4, OUTPUT);
pinMode(pin5, OUTPUT);
pinMode(pin6, OUTPUT);
pinMode(pin7, OUTPUT);
}
void loop() {
digitalWrite(pin2, HIGH); // Enciende y apaga secuencialmente LED-s
delay(timer);
digitalWrite(pin2, LOW);
delay(timer);
digitalWrite(pin3, HIGH);
delay(timer);
digitalWrite(pin3, LOW);
delay(timer);
digitalWrite(pin4, HIGH);
```

Cuadro de Código fuente 13

```
delay(timer);
digitalWrite(pin4, LOW);
delay(timer);
digitalWrite(pin5, HIGH);
delay(timer);
digitalWrite(pin5, LOW);
delay(timer);
digitalWrite(pin6, HIGH);
delay(timer);
digitalWrite(pin6, LOW);
delay(timer);
digitalWrite(pin7, HIGH);
delay(timer);
digitalWrite(pin7, LOW);
delay(timer);
digitalWrite(pin6, HIGH);
delay(timer);
digitalWrite(pin6, LOW);
delay(timer);
digitalWrite(pin5, HIGH);
delay(timer);
digitalWrite(pin5, LOW);
delay(timer);
digitalWrite(pin4, HIGH);
delay(timer);
digitalWrite(pin4, LOW);
delay(timer);
digitalWrite(pin3, HIGH);
delay(timer);
digitalWrite(pin3, LOW);
delay(timer);
}
```

Cuadro 13 (Continuación)

Practica 2

En esta práctica se hará lo mismo que la anterior, solo que la cantidad de código será reducida, porque se utilizara un vector, y el ciclo for.

En nuestro IDE de Arduino:

```
int pinArray[] = {2, 3, 4, 5, 6, 7}; // Define el array de pines
int count = 0; // Contador
int timer = 100; // Temporizador

void setup(){
for (count=0;count<6;count++){ // Configuramos todos los pin´s
pinMode(pinArray[count], OUTPUT);
}
}
void loop() { // Enciende y apaga secuencialmente los led´s

for (count=0;count<6;count++) { // utilizando la secuencia de control
digitalWrite(pinArray[count], HIGH); // Recorrido de ida
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer);
}
for (count=5;count>=0;count--) {
digitalWrite(pinArray[count], HIGH); // Recorrido de vuelta
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer);
}
}
```

Cuadro de Código fuente 14

Practica 3

En esta práctica se hará lo mismo que en las dos practicas anteriores, con la diferencia que tendrá otro efecto visual.

En nuestra IDE de Arduino

```
int pinArray[] = {2, 3, 4, 5, 6, 7}; // vector de pin´s
int count = 0; // Contador
int timer = 30; // Temporizador

void setup(){
for (count=0;count<6;count++) { // Configuramos todos los pin´s juntos
pinMode(pinArray[count], OUTPUT);
}
}
void loop() {
for (count=0;count<5;count++) { // Enciende los LED creando una estela visual
digitalWrite(pinArray[count], HIGH);
delay(timer);
digitalWrite(pinArray[count + 1], HIGH);
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer*2);
}
for (count=5;count>0;count--) {
digitalWrite(pinArray[count], HIGH);
delay(timer);
digitalWrite(pinArray[count - 1], HIGH);
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer*2);
}
}
```

Cuadro de Código fuente 15

3.1.19 Practicas con arduino II

Observación: *Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino www.arduino.cc/*

3.1.19.1 Practica 1: Generador de Notas Musicales

Generar 8 notas musicales por el pin 11 de Arduino, Se debe crear un array (vector) de datos compuesto por los valores correspondientes a las 8 notas que se pretende sacar:

```
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
```

Se deben definir también el tiempo de pausa entre nota y nota y el tiempo de pausa de fin de secuencia de notas:

```
int tnota=100; // Duracion de la nota
int pausa=100; // Tiempo de pausa entre una nota y otra
```

Las iteraciones para el recorrido de las 8 notas se realizan con un ciclo for:

```
for(n=0;n<8;n++)
```

El tiempo de activado y desactivado de la salida del parlante también se resuelve con un ciclo for:

```
for(m=0;m<=tnota;m++){
```

Esquema

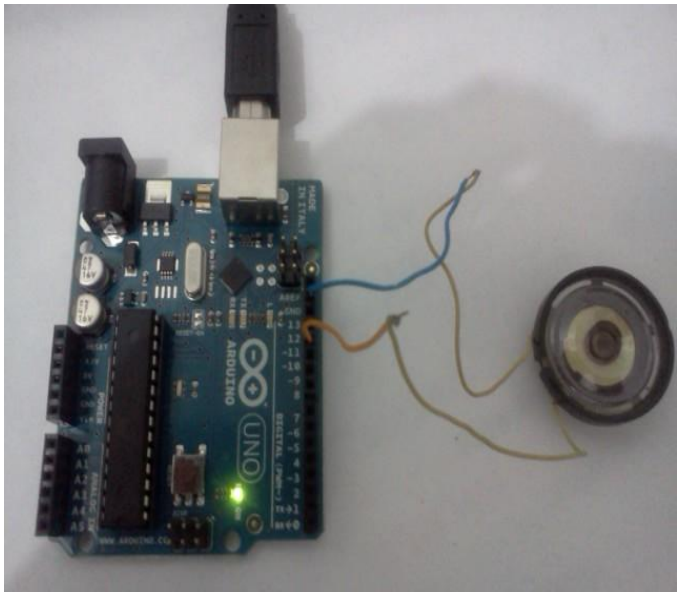


Figura 137. Montaje del generador de notas musicales

El parlante se conectará al pin 11 y a tierra GND

En nuestro IDE de Arduino:

```
int parlante=11; // Se declara la variable parlante como el pin 11.

int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; //vector con los tiempos que corresponden a las distintas notas

int n=0; // Variable  que se utiliza para el ciclo for del recorrido de las notas.

int m= 0; // Variable que se utiliza para el ciclo for del tiempo de activado y desactivado de la salida del parlante

int tnota=200; // Duración de la nota
int pausa=200; // Duración de la pausa
void setup() {

pinMode(parlante,OUTPUT);// pin 11 como modo de salida

}
```

Cuadro de Código fuente 16

```
void loop(){

for(n=0;n<8;n++){ // Ciclo que recorre el vector con las duraciones de los pulsos de cada nota

for(m=0;m<=tnota;m++){

digitalWrite(parlante,HIGH); // El parlante suena

delayMicroseconds(notas[n]); //Tiempo en microsegundos que está a 5V la salida del piezoeléctrico

digitalWrite(parlante,LOW);// El parlante deja de sonar

delayMicroseconds(notas[n]); //Tiempo en microsegundos que está a 0V la salida del piezoeléctrico

}

delay(pausa); //tiempo en silencio entre notas

}

}
```

Cuadro 16 (Continuación)

3.1.19.2 Práctica 2: Timbre con Pulsador

Se trata de realizar un timbre a través de un parlante que su salida será el pin 11, que emita dos tonos recogidos de una colección de ocho tonos, por ejemplo el tono 3 y el tono 7. El timbre se activa mediante un pulsador conectado en el pin 5 (entrada digital).

Utilizaremos las notas anteriores, y escogeremos el tono 3 equivalente a 1432 y el tono 7 equivalente a 956

Esquema

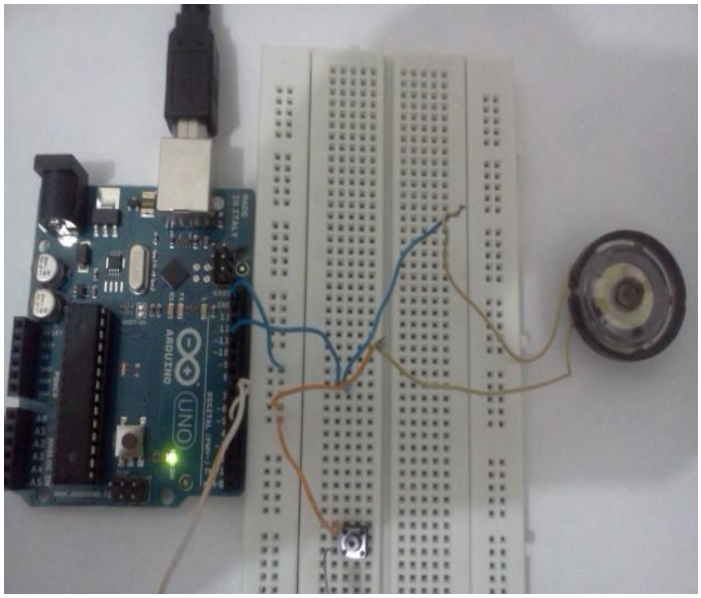


Figura 138. Montaje del timbre con pulsador

En nuestro IDE de Arduino:

```
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; //definición de vector de 8 notas

int puls=5; // designación del pulsador de llamada

int parlante=11; // designación de la salida hacia el parlante
int tnota=150;
int n=0;
int m=0;

void setup (){
for(n=0;n<4;n++){
pinMode(parlante,OUTPUT);
pinMode(puls,INPUT);
}
}
```

Cuadro de Código fuente 17

```
// Se crea la función nota
void nota() { // rutina que genera los tonos de llamada

for(m=0;m<=tnota;m++){
digitalWrite(parlante,HIGH);
delayMicroseconds(notas[n]);
digitalWrite(parlante,LOW);
delayMicroseconds(notas[n]);

}
}
void loop(){
if(digitalRead(puls)==HIGH){
n=3; //elegimos la primera nota del timbre
nota(); //que aquí es la tercera de la cadena
delay(200);
n=7; //elegimos la segunda nota del timbre
nota(); //que aquí es la séptima de la cadena
delay(200);
}
}
```

Cuadro 17 (Continuación)

3.1.19.3 Práctica 3: Encendido y Apagado de un led de manera Analógica

Se trata de que se envíe hacia el pin 4 un valor analógico descendente y ascendente, cíclicamente comprendido entre 0 y 255, con incrementos de 5.

Un pin analógico, su modo siempre es de salida OUTPUT, por lo tanto no es necesario que se le defina el modo.

Esquema

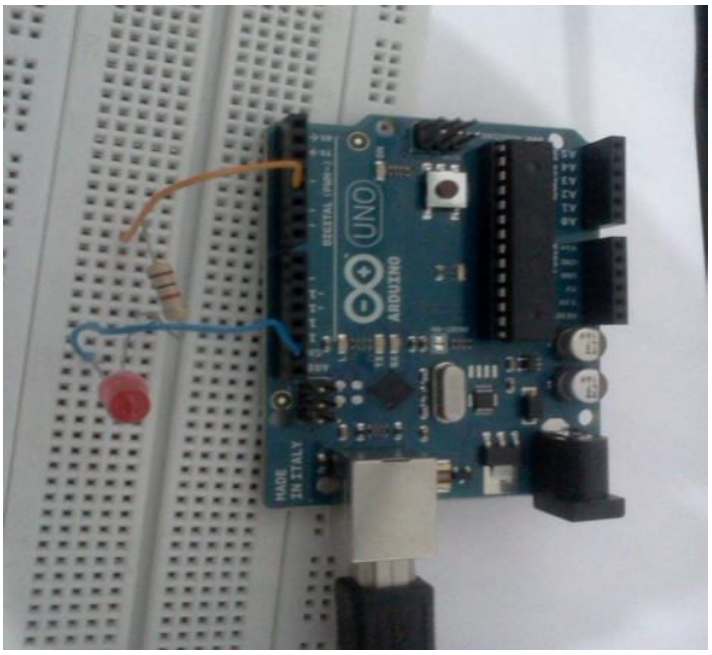


Figura 139. Esquema de configuración de un led de manera analógica

En nuestro IDE de Arduino:

```
int value = 0; // Valor a sacar por la salida analógica del pin  4
int ledpin = 4; // Salida analógica pin 4

void setup()  {
  // Analogico siempre es salida, por eso no se define el modo del pin
                    }
void loop() {
for(value = 0 ; value <= 255; value+=5) { // Variación de la variable se salida
entre el MIN y MAX
analogWrite(ledpin, value); // Enviar valor a la salida (entre 0 y 255)
delay(50); // Esperar 50 ms para ver el efecto de variación
}
for(value = 255; value >=0; value-=5) { // Variación de la variable de salida
entre MAX y MIN
analogWrite(ledpin, value);
delay(50);
}
}
```

Cuadro de Código fuente 18

3.1.20 Practicas con arduino III

Observación: *Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino* www.arduino.cc/

3.1.20.1 Práctica 1: Potenciómetro, lectura de señal analógica

Esta práctica consiste en encender y apagar un led por medio de un potenciómetro conectado a la parte analógica de la placa Arduino.

El tiempo que el led parpadeara dependerá de la lectura análoga por parte del potenciómetro.

Esquema:

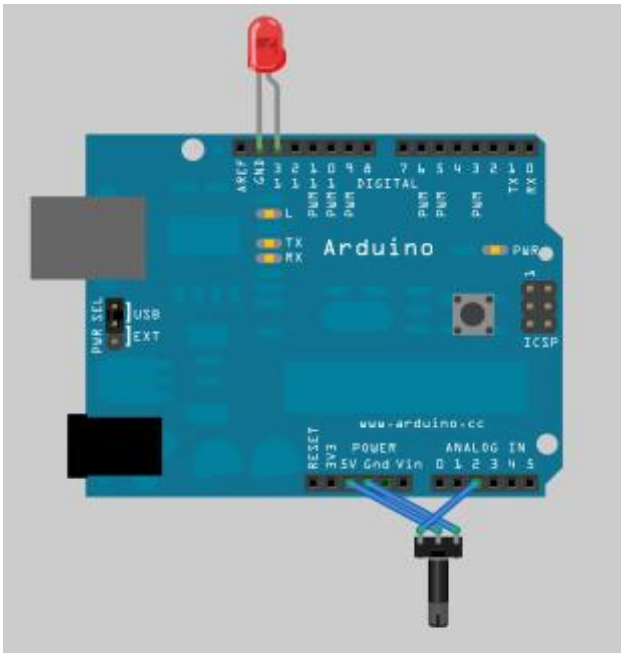


Figura 140. Configuración de un potenciómetro en Arduino

El led se encontrara ubicado en el pin 13 de la señal digital, el potenciómetro ira distribuido de la siguiente manera: la primera terminal ira conectada a tierra (GND), la segunda terminal ira conectada a la entrada analógica 2, y la tercera terminal estará conectada a la salida de 5 voltios

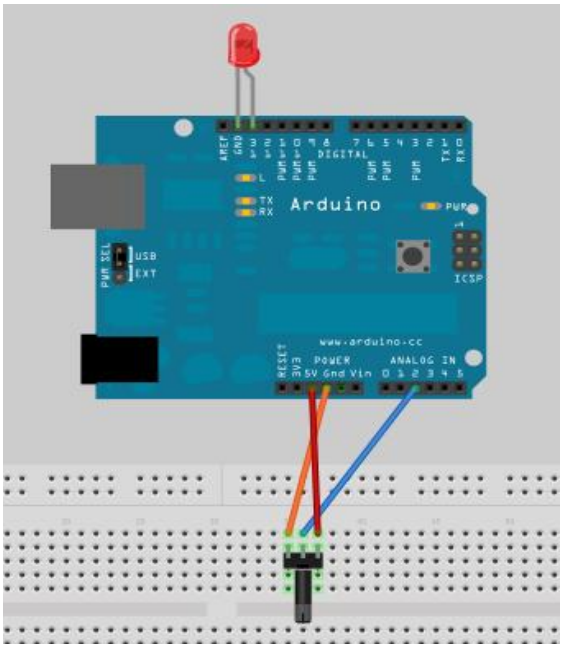


Figura 141. Configuración de un potenciómetro en una protoboard

En nuestro IDE de Arduino:

```
int potPin = 2; // seleccionar el pin de entrada analógico para el potenciómetro
int ledPin = 13; // seleccionar el pin de salida digital para el LED
int val = 0; // variable para almacenar el valor capturado desde el sensor
void setup() {
pinMode(ledPin, OUTPUT); // declara el ledPin en modo salida
}
void loop() {
val = analogRead(potPin); // lee el valor del sensor
digitalWrite(ledPin, HIGH); // enciende LED
delay(val); // detiene el programa por un tiempo “val”
digitalWrite(ledPin, LOW); // apaga el LED
delay(val); // detiene el programa por un tiempo “val”
}
```

Cuadro de Código fuente 19

3.1.20.2 Práctica 2: Rayo de luz

Este practica muestra como realizar un rayo de luz, moviéndose a través de una línea de led´s. Podremos configurar tanto la velocidad del rayo, así como la longitud.

Al final parecerá como si un rayo de luz solido atravesase los led´s.

Esquema:

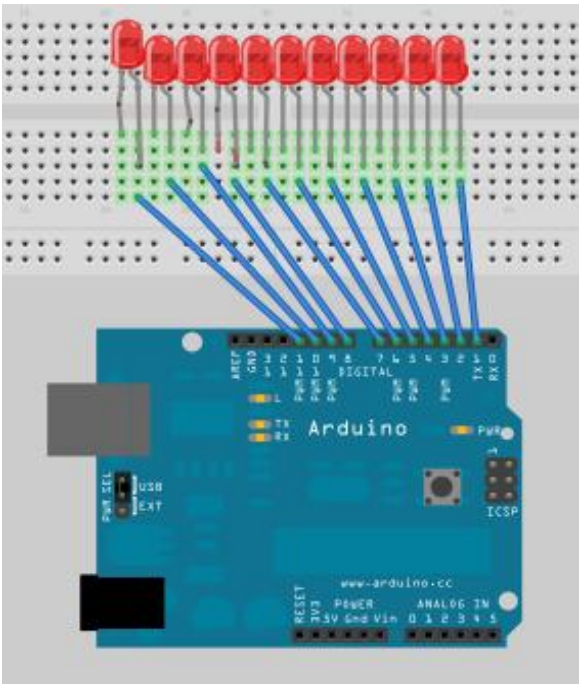


Figura 142. Configuración de la línea de leds

En nuestro IDE de Arduino:

```
int pinArray [] = { 1,2,3,4,5,6,7,8,9,10,11 };
int controlLed = 13; // LED de control
int waitNextLed = 100; // Tiempo antes de encender el siguiente LED
int tailLength = 4;
// Número de LED-s que permanecen encendidos antes de empezar a apagarlos para formar la cola.
int lineSize = 11;
// Número de LED-s conectados (que es también el tamaño del array)
```

Cuadro de Código fuente 20

```
void setup() // Configuración de los PIN-es como salida digital
{
int i;
pinMode (controlLed, OUTPUT);
for (i=0; i< lineSize; i++)
{
pinMode(pinArray[i], OUTPUT);
}
}
void loop()
{
int i;
int tailCounter = tailLength; // Se establece la longitud de la cola en un contador
digitalWrite(controlLed, HIGH); // Se enciende el LED de control para indicar el inicio del loop
for (i=0; i<lineSize; i++)
{
digitalWrite(pinArray[i],HIGH); // Se encienden consecutivamente los led´s
delay(waitNextLed); // Esta variable de tiempo controla la velocidad a la que se mueve el rayo de luz
if (tailCounter == 0)
{
digitalWrite(pinArray[i-tailLength],LOW); // Se apagan los led´s en función de la longitud de la cola.
}
else
if (tailCounter > 0)
tailCounter--;
}
for (i=(lineSize-tailLength); i<lineSize; i++)
{
digitalWrite(pinArray[i],LOW); // Se apagan los LED
delay(waitNextLed); // Esta variable de tiempo controla la velocidad a la que se mueve el rayo.

}
}
```

Cuadro 20 (Continuación)

3.1.20.3 Practica 3: Entrada Analógica y Escritura Serial

Esta práctica trata de configurar un canal de entrada analógica y enviar el valor leído al pc para visualizarlo.

Nota: Para visualizar los valores debemos entrar a serial monitor.

Figura 143. Icono del Serial Monitor

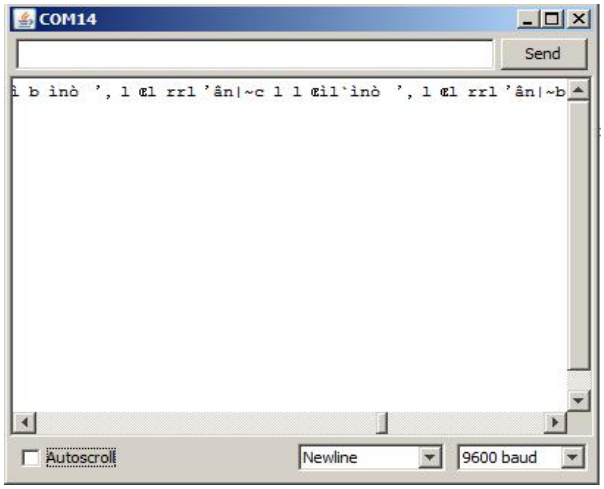


Figura 144. Interface del Serial Monitor

Esquema:

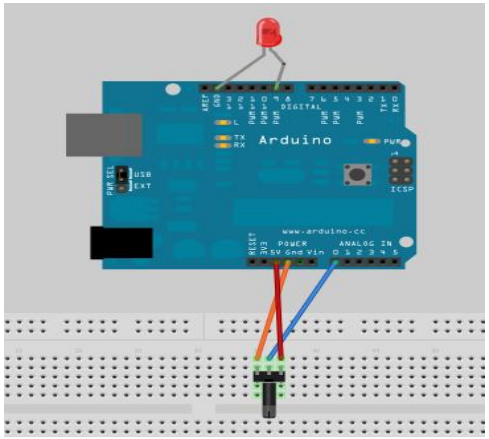


Figura 145. Esquema de configuración de un potenciómetro y un led

En nuestro IDE de Arduino:

```
const int analogInPin = A0;  // Pin analógico del potenciómetro.

const int analogOutPin = 9; // Salida analógica del led.

int sensorValue = 0;        // Lector del potenciometro

int outputValue = 0;

void setup() {

   Serial.begin(9600);  // Se inicializa la comunicación serial a 9600 bps.

}

void loop() {

sensorValue = analogRead(analogInPin);      // Lectura análoga del potenciómetro.

outputValue = map(sensorValue, 0, 1023, 0, 255);  // Mapeamos el rango de la salida analógica.

analogWrite(analogOutPin, outputValue);       // Cambiamos el valor de la salida análoga.

 Serial.print("sensor = " );

 Serial.print(sensorValue);

 Serial.print("\t output = ");

 Serial.println(outputValue);

// Se imprimen los valores en el monitor serial

  delay(10); // Se esperan 10 milisegundos para la siguiente ejecución.

}
```

Cuadro de Código fuente 21

3.2 IMPLEMENTACION DE LOS TALLERES DE ARDUINO

Para implementar los talleres de arduino, la Corporación Universitaria de la Costa CUC seleccionó los siguientes colegios distritales:

- ✓ Instituto Técnico Nacional de Comercio.
- ✓ Instituto Distrital de Experiencias Pedagógicas.

A su vez estos colegios distritales optaron por escoger docentes de diferentes áreas para que recibieran la capacitación con el fin de transmitirlo a sus estudiantes.

La semana en que los talleres fueron implementados transcurrió del día 10 al 14 de octubre de 2011 en un horario de 8:00 am a 12:00 pm, con un receso de 10:00 am a 10:30 am donde compartieron sus experiencias y disfrutaron de un refrigerio.

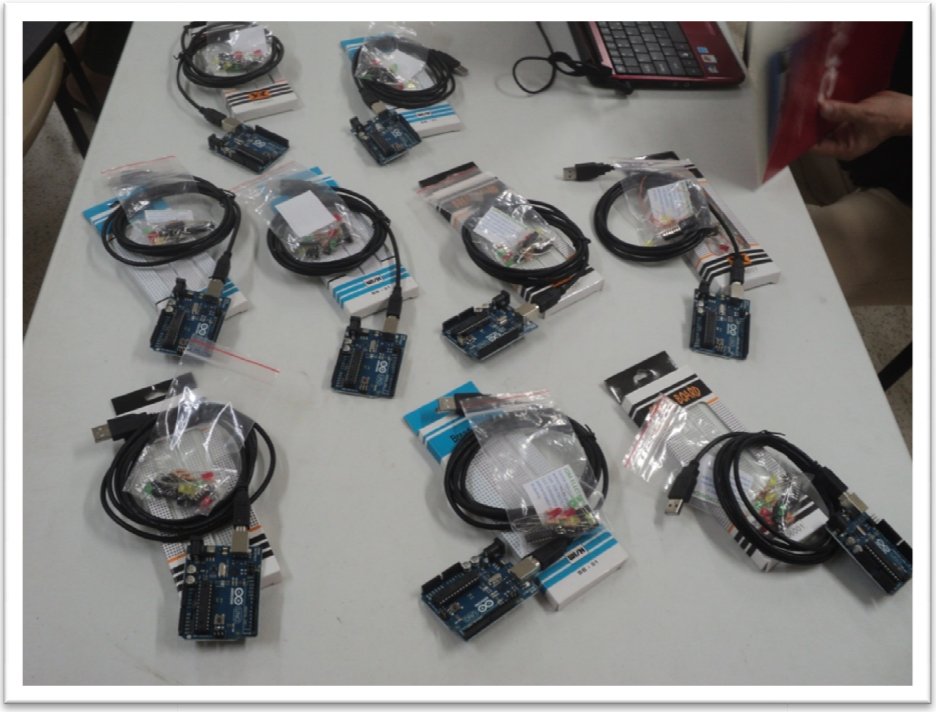
Seguidamente se presentan fotografías que demuestran el desarrollo de la capacitación en las instalaciones de la Corporación Universitaria de la Costa CUC.

Fotografía 1. Implementación de talleres Arduino parte 1

Fotografía 2. Implementación de talleres Arduino parte 2

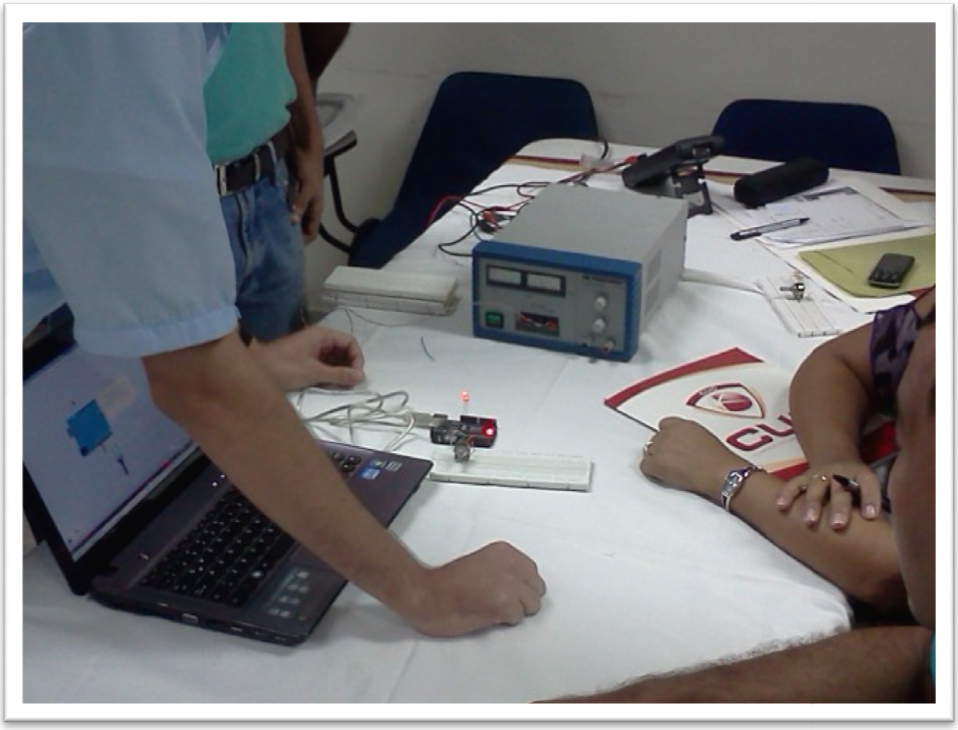
Fotografía 3. Implementación de talleres Arduino parte 3

Fotografía 4. Implementación de talleres Arduino parte 4



Fotografía 5. Kits de Arduino

Fotografía 6. Implementación de talleres Arduino parte 5



Fotografía 7. Implementación de talleres Arduino parte 6

Fotografía 8. Implementación de talleres Arduino parte 7

Fotografía 9. Implementación de talleres Arduino parte 8

Fotografía 10. Implementación de talleres Arduino parte 9

Fotografía 11. Final de la capacitación

CONCLUSIONES Y TRABAJOS FUTUROS

Para concluir podríamos decir que la tecnología es una de las principales fuentes de progreso e instrumento fundamental para el crecimiento de la sociedad; no solo en el campo educativo sino también en otros campos como el productivo y empresarial. Por consiguiente si logramos que la tecnología continúe su camino, sería un gran avance para la solución de otros problemas que presenta la ciudad.

Después de haber realizado la capacitación a docentes, resolvimos que la placa Arduino es una herramienta que ayuda a la realización de toda actividad obteniendo así múltiples beneficios.

Con el cambio tecnológico en las instituciones educativas se lograría alcanzar un nivel muy importante en la respuesta a los procesos que allí se realizan, por otra parte les permitiría a los estudiantes llegar a grados significativos de soporte e investigación. Con la capacitación que se les brindó a los educadores se logró proveer conocimiento acerca del manejo de los equipos tecnológicos y los datos contenidos en estos.

A nivel profesional esta capacitación fue una gran oportunidad de aplicar los conocimientos adquiridos a lo largo del tiempo. Donde esto tendría un espacio importante para el perfeccionamiento que nos garantizaría una mejor calidad vida.

Se espera que este proyecto solo sea una guía y posteriormente se extienda a tal nivel que se formen muchas instituciones educativas.

Como trabajo futuro se proyecta la realización de un proyecto único y creativo que involucre la investigación y la programación de la placa Arduino en conjunto con estudiantes de alguna institución educativa.

BIBLIOGRAFIA

MARTÍNEZ USERO, José Ángel – LARA NAVARRA, Pablo. La producción de contenidos web.

ARDUINO. Definición de Arduino [En línea].
http://www.fosforescente.cl/?p=590 [Citado el 9 de julio de 2011]

ARDUINO. Página oficial de Arduino en español [En línea].
http://www.arduino.cc/es/ [Citado el 9 de julio de 2011]

CUARTIELLES, David. El creador debe convertirse en científico. [En línea] En: <http://www.elpais.com/articulo/ocio/David/Cuartielles/creador/debe/convertirse/cientifico/elpportec/20060928elpciboci_1/Tes > [Citado el 9 de julio de 2011]

BANZI, Massimo. Página oficial de Massimo Banzi [En línea].
http://www.massimobanzi.com/about/ [Citado el 9 de julio de 2011]

PANIAGUA, Soraya. Arduino: La revolución silenciosa del hardware libre [En línea].

< http://www.sorayapaniagua.com/2011/03/14/arduino-la-revolucion-silenciosa-del-hardware-libre/> [Citado el 16 de julio de 2011]

GOOGLE, Imágenes. Imágenes de Google [En línea]. Disponibles en:
< bibliotecadeinvestigaciones.wordpress.com, pcexpertos.com, iearobotics.com, blog.marques.cx, tucamon.es, fondosypantallas.com, bricogeek.com, electronicamagnabit.com, robotjuanydavid.blogspot.com, donachip.gostorego.com, tecnoblogmanu.blogspot.com, compucanjes.com, foroselectronica.es, pablin.com.ar, eui.upm.es, elpolvorin.over-blog.es, fundaroraima.blogspot.org, definicion.de, byte2011.blogspot.com, lizykaryhermosas.blogspot.com, jesustecman.blogspot.com, tweaktown.com, memito-trabajospics.blogspot.com, elecronicaestudio.com, antoniotoriz.blogspot.com, ingeniosolido.com, chemicaloliver.net, www.robotshop.com, olimex.cl, educacionvialsvp.blogspot.com, solostocks.com, infierno-verde.blogspot.com, circulaseguro.com, http://www.trafictec.com/lampara_20.php, es.wikipedia.org, otroblogmas.com > [Citados del 9 de julio al 10 de octubre de 2011]

TECNOLOGÍA, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Tecnología [Citado el 10 de julio de 2011]

SEMAFORO, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Semaforo [Citado el 2 de octubre de 2011]

ESTRUCTURAS DE CONTROL, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Estructuras_de_control [Citado el 2 de octubre de 2011]

MEMORIA EEPROM, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Memoria_EEPROM [Citado el 2 de octubre de 2011]

SISTEMA DE NUMERACION, definición en binarios [En línea]
http://platea.pntic.mec.es/~lgonzale/tic/binarios/numeracion.html [Citado el 2 de octubre de 2011]

BYTE, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Byte [Citado el 2 de octubre de 2011]

BIT, Wikipedia la enciclopedia libre en español [En línea].
http://es.wikipedia.org/wiki/Bit [Citado el 2 de octubre de 2011]

FUNCIONES DE ARDUINO, Pagina oficial de Arduino [En línea].
http://Arduino.cc/es/Reference/ [Citados 3 de octubre de 2011]

PALABRAS RESERVADAS DE ARDUINO, Página oficial de Arduino [En línea].
http://arduino.cc/playground/ [Citado 3 de octubre de 2011]

MODELOS DE PLACAS ARDUINO, Página oficial de Arduino en español [En línea].
http://www.arduino.cc/es/ [citados 3 de octubre de 2011]

GALEON. Microcontroladores [En línea]
http://microcontroladores-e.galeon.com/ [Citado el 4 de octubre de 2011]

KILOBYTE, definición en masadelante [En línea]
http://www.masadelante.com/faqs/kilobyte [Citado el 3 de octubre de 2011]

VIRTUAL BREADBOARD, página oficial [En línea]
www.virtualbreadboard.com [Citado el 6 de octubre de 2011]

WINDOWS, MICROSOFT. Administrador de dispositivos [En línea]
http://windows.microsoft.com/es-ES/ [Citado el 6 de octubre de 2011]

FUNCIONES DEL PUERTO SERIE, definiciones [En línea].
http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf [Citado el 12 de octubre de 2011]

TOD, Ejercicio Sensor de fuerza [En línea].
http://todbot.com [Citado el 13 de octubre de 2011]

HOBLEY, Stephen. Arpa Laser, Makeprojects [En línea].
http://makeprojects.com/Project/Laser-Harp/690/1 [Citado 4 de diciembre de 2011]

MAGRI, Kris. My Robot Mikey, Makeprojects [En línea].
http://makeprojects.com/Project/My-Robot-Makey/51/1 [Citado el 4 de diciembre de 2011]

OPENLAB EYEBEAM. Micrófono alcoholímetro, instructables [En línea].
http://www.instructables.com/id/Breathalyzer-Microphone/ [Citado el 4 de diciembre de 2011]

VILCHES, Sergio. Fotografías de alta velocidad, hacknmond [En línea].
http://hacknmod.com/hack/high-speed-photography-how-to-trigger-using-arduino/ [Citado el 4 de diciembre de 2011]

BUECHLEY, Leah. Fotografías de alta velocidad, hacknmond [En línea].
http://www.instructables.com/id/turn-signal-biking-jacket/ [Citado el 4 de diciembre de 2011]

	NORMAS PARA LA ENTREGA DE TESIS Y TRABAJOS DE GRADO A LA UNIDAD DE INFORMACION	VERSION: 01
		FECHA: Febrero 2011
		CODIGO:DOC-VACRE-NETGUDI

ANEXO 1
CARTA DE ENTREGA Y AUTORIZACIÓN DE LOS AUTORES PARA LA CONSULTA, LA REPRODUCCIÓN PARCIAL O TOTAL, Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO DE TESIS Y TRABAJOS DE GRADO

Barranquilla, Fecha

Marque con una X
Tesis ☐ **Trabajo de Grado** ☐

Yo _____, identificado con C.C. No. _____, actuando en nombre propio y como autor de la tesis y/o trabajo de grado titulado_____ _____ presentado y aprobado en el año _____ como requisito para optar al título de _____;
hago entrega del ejemplar respectivo y de sus anexos de ser el caso, en formato digital o electrónico (DVD) y autorizo a la CORPORACIÓN UNIVERSITARIA DE LA COSTA, para que en los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia, utilice y use en todas sus formas, los derechos patrimoniales de reproducción, comunicación pública, transformación y distribución (alquiler, préstamo público e importación) que me corresponden como creador de la obra objeto del presente documento.

Y autorizo a la Unidad de información, para que con fines académicos, muestre al mundo la producción intelectual de la Corporación Universitaria de la Costa, a través de la visibilidad de su contenido de la siguiente manera:

Los usuarios puedan consultar el contenido de este trabajo de grado en la página Web de la Facultad, de la Unidad de información, en el repositorio institucional y en las redes de información del país y del exterior, con las cuales tenga convenio la institución y Permita la consulta, la reproducción, a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato DVD o digital desde Internet, Intranet, etc., y en general para cualquier formato conocido o por conocer.

El AUTOR - ESTUDIANTES, manifiesta que la obra objeto de la presente autorización es original y la realizó sin violar o usurpar derechos de autor de terceros, por lo tanto la obra es de su exclusiva autoría y detenta la titularidad ante la misma. PARÁGRAFO: En caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, EL ESTUDIANTE - AUTOR, asumirá toda la responsabilidad, y saldrá en defensa de los derechos aquí autorizados; para todos los efectos, la Universidad actúa como un tercero de buena fe.

Para constancia se firma el presente documento en dos (02) ejemplares del mismo valor y tenor, en Barranquilla D.E.I.P., a los ____días del mes de _____de Dos Mil _____200___

EL AUTOR - ESTUDIANTE:_____
FIRMA

ANEXO 2
F ORMULARIO DE LA DESCRIPCIÓN DE LA TESIS O DEL TRABAJO DE GRADO

TÍTULO COMPLETO DE LA TESIS O TRABAJO DE GRADO:

_____

_____

SUBTÍTULO, SI LO TIENE:

_____

_____

AUTOR AUTORES

Apellidos Completos	Nombres Completos

DIRECTOR (ES)

Apellidos Completos	Nombres Completos

JURADO (S)

Apellidos Completos	Nombres Completos

ASESOR (ES) O CODIRECTOR

Apellidos Completos	Nombres Completos

TRABAJO PARA OPTAR AL TÍTULO DE:_____

FACULTAD: _____

PROGRAMA: Pregrado ____ Especialización ____

NOMBRE DEL PROGRAMA_____

	NORMAS PARA LA ENTREGA DE TESIS Y TRABAJOS DE GRADO A LA UNIDAD DE INFORMACION	VERSION: 01 FECHA: Febrero 2011 CODIGO:DOC-VACRE-NETGUDI

CIUDAD: Barranquilla **AÑO DE PRESENTACIÓN DEL TRABAJO DE GRADO:**_____

NÚMERO DE PÁGINAS _____

TIPO DE ILUSTRACIONES:

- ☐ Ilustraciones
- ☐ Láminas
- ☐ Retratos
- ☐ Tablas, gráficos y diagramas
- ☐ Planos
- ☐ Mapas
- ☐ Fotografías

MATERIAL ANEXO (Vídeo, audio, multimedia o producción electrónica):

Duración del audiovisual: _____ minutos.

Número de casetes de vídeo: _____ Formato: VHS ___ Beta Max ___ ¾ ___ Beta Cam ____

Mini DV ____ DV Cam ____ DVC Pro ____ Vídeo 8 ____ Hi 8 ____

Otro. Cuál? _____

Sistema: Americano NTSC _____ Europeo PAL _____ SECAM _____

Número de casetes de audio: _____

Número de archivos dentro del DVD (En caso de incluirse un DVD diferente al trabajo de grado):

_____

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o tener una mención especial)*:

_____

DESCRIPTORES O PALABRAS CLAVES EN ESPAÑOL E INGLÉS: Son los términos que definen los temas que identifican el contenido. *(En caso de duda para designar estos descriptores, se recomienda consultar con la Unidad de Procesos Técnicos de la Unidad de información en el correo biblioteca@cuc.edu.co, donde se les orientará).*

ESPAÑOL	INGLÉS
_____	_____
_____	_____
_____	_____

RESUMEN DEL CONTENIDO EN ESPAÑOL E INGLÉS:(Máximo 250 palabras-1530 caracteres):

_____

_____

_____

_____

_____