

Curso: Programación de Arduino Profesor: Acosta Jacinto, Rubén Eusebio

# FUNDAMENTOS DE PROGRAMACIÓN ARDUINO UNO

# **PROPÓSITO**

El propósito de esta manual 1 es abordar el concepto de computación física que es la capacidad de interacción y comunicación de una máquina con los humanos, usando sensores y actuadores. Las decisiones de esto las va a tomar un *microcontrolador Atmega328P-PU* que se encuentra ubicado en la placa Arduino

# ARDUINO.

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

Arduino puede tomar información del entorno a través de sus pines de entrada, para esto toda una gama de sensores puede ser usada 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 conectarlo a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP).



# INDICE

1 ELECTRÓNICA	3
2 PROGRAMACIÓN	.11
3 ENTORNO DE PROGRAMACIÓN	.12
4 CARGANDO MI PRIMER PROGRAMA	.22
5 TUTORIALES	23
6 BOOTLOADER	47



# **1.- ELECTRÓNICA**

Desde el inicio de este capítulo te vas a enterar de los términos técnicos que tiene la electrónica y que al final de éste muy seguramente vas a manejar de manera experta.

# **1.1 CONCEPTO DE ELECTRÓNICA**

Estudia y emplea sistemas cuyo funcionamiento se basa en la conducción y el control del flujo de los electrones u otras partículas cargadas eléctricamente.

El diseño y la gran construcción de circuitos electrónicos para resolver problemas prácticos forman parte de la electrónica y de los campos de la ingeniería electrónica, electromecánica y la informática en el diseño de software para su control.



La electrónica desarrolla en la actualidad una gran variedad de tareas. Los principales usos de los circuitos electrónicos son el control, el procesado, la distribución de información, la conversión y la distribución de la energía eléctrica. Estos dos usos implican la creación o la detección de campos electromagnéticos y corrientes eléctricas.

Mira a tu alrededor radio, televisor, PC, teléfono móvil, lavadora todos ellos tienen electrónica.

# **1.2 VOLTAJE**

Una magnitud física que impulsa a los electrones a lo largo de un conductor (por ejemplo un cable) en un circuito eléctrico cerrado, provocando el flujo de una corriente eléctrica. Su unidad es el Voltio (V). El instrumento usado para medir el voltaje se conoce como voltímetro.

# **1.2.1VOLTAJE DC**

Es el flujo continuo de electrones a través de un conductor entre dos puntos de distinto potencial. En la corriente continua las cargas eléctricas circulan siempre en la misma dirección, es continua la corriente mantiene siempre la misma polaridad. En la norma sistemática europea el color negro corresponde al negativo y el rojo al positivo o sencillamente se simboliza para el positivo con VCC, +, VSS y para el negativo con 0V, -, GND





# **1.2.2 VOLTAJE AC**

Es la corriente eléctrica en la que la magnitud y dirección varían cíclicamente. La forma de onda de la corriente alterna más comúnmente utilizada es la de una onda seno.

El voltaje AC es el que llega a la tomas de electricidad de los hogares y a las empresas, es muy común encontrarla en las tomas de corriente donde se conectan nuestros electrodomésticos. Sin embargo, las señales de audio y de radio transmitidas por los cables eléctricos son también ejemplos de corriente alterna. En estos usos, el fin más importante suele ser la transmisión y recuperación de la información codificada (o modulada) sobre la señal de la AC.



# **1.3 CORRIENTE**

Es el flujo de electrones a través de un conductor o semiconductor en un sentido. La unidad de medida de ésta es el amperio (A). Una corriente eléctrica, puesto que se trata de un movimiento de cargas, produce un campo magnético, un fenómeno que puede aprovecharse en el electroimán, este es el principio de funciona-miento de un motor.

El instrumento usado para medir la intensidad de la corriente eléctrica es el galvanómetro que, calibrado en amperios, se llama amperímetro, colocado en serie con el conductor cuya intensidad se desea medir.



# **1.4 RESISTENCIA**

Es la propiedad física mediante la cual todos los materia-les tienden a oponerse al flujo de la corriente. La unidad de este parámetro es el Ohmio ( $\Omega$ ). Puedes encontrar resistencias en los calefactores eléctricos, tarjetas electrónicas, estufas son muy útiles para limitar el paso de la corriente.





Símbolo



# **1.4.1 CÓDIGO DE COLORES DE LOS RESISTORES.**

Color de la banda	Valor de la 1ºcifra significativa	Valor de la 2ºcifra significativa	Multiplicador	Tolerancia
Negro	-	0	1	-
Marrón	1	1	10	±1%
Rojo	2	2	100	±2%
Naranja	3	3	1 000	-
Amarillo	4	4	10 000	±4%
Verde	5	5	100 000	±0,5%
Azul	6	6	1 000 000	±0,25%
Violeta	7	7	1000000	±0,1%
Gris	8	8	100000000	±0.05%
Blanco	9	9	100000000	-
Dorado	-	-	0,1	±5%
Plateado	-	-	0,01	±10%
Ninguno	-	-	-	±20%

Tabla de código de colores





La caracterización de una resistencia de 2.700.000  $\Omega$  (2,7 M $\Omega$ ), con una tolerancia de ±10%, sería la representada en la figura :

- 1ª cifra: rojo (2)
- 2<sup>a</sup> cifra: violeta (7)
- Multiplicador: verde (100000)
- Tolerancia: plateado (±10%)

## 1.5 LEY DE OHM

La ley dice que la corriente (I) que circula por un conductor eléctrico es directamente proporcional al voltaje (V) e inversamente proporcional a la resistencia (R).

La pirámide de la derecha es muy útil para conocer la fórmula a la que es igual la variable que tapes con el dedo, por ejemplo: Tapa con tu dedo la V (voltaje), entonces voltaje va a ser igual a I (corriente) por R (resistencia), una más, tapa I (Corriente), I va ser igual a V divido R.



# **1.6 DIODO**

Un diodo es un componente electrónico de dos terminales que permite la circulación de la corriente eléctrica a través de él en un solo sentido. Tiene dos partes: el cáto-do y el ánodo.





# **1.7 TRANSISTOR**

El transistor es un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador. Tiene tres partes: la base (B), el emisor (E) y colector (C). Actualmente se encuentran prácticamente en todos los aparatos domésticos de uso diario: radios, televisores, grabadoras, reproductores de audio y video, hornos de microondas, lavadoras, automóviles,

equipos de refrigeración, alarmas, relojes de cuarzo, ordenadores, calculadoras, impresoras, lámparas fluorescentes, equipos de rayos X, tomógrafos, ecógrafos, reproductores mp3, teléfonos celulares, etc.



# **1.8 CONDENSADOR**

Un condensador o capacitor es un dispositivo utilizado en electrónica, capaz de almacenar energía sustentando un campo eléctrico. Está formado por un par de superficies conductoras, generalmente en forma de láminas o placas, en situación de influencia total separadas por un material dieléctrico o por el vacío. Las placas, sometidas a una diferencia de potencial, adquieren una deter-minada carga eléctrica, positiva en una de ellas y negativa en la otra.



# 1.9 LED

Un LED (Diodo emisor de luz, también "diodo luminoso") es un diodo semiconductor que emite luz. Se usan como indicadores en muchos dispositivos, y cada vez con mucha más frecuencia en iluminación. Los LEDs presentan muchas ventajas sobre las fuentes de luz incandescente como un consumo de energía mucho menor, mayor tiempo de vida, menor tamaño, gran durabilidad y fiabilidad.



El LED tiene una polaridad, un orden de conexión, y al conectarlo al revés se puede quemar, revisa los dibujos de la parte superior para conocer a que corresponde el positivo y el negativo.



# **1.10 EL PULSADOR**

Un botón o pulsador es utilizado para activar alguna función. Los botones son por lo general activados al ser pulsados, normalmente con un dedo. Un botón de un dispo-sitivo electrónico funciona por lo general como un interruptor eléctrico, es decir en su interior tiene dos contactos, si es un dispositivo NA (normalmente abier-to) o NC (normalmente cerrado), con lo que al pulsarlo se activará la función inversa de la que en ese momento este realizando.





# **1.11 POTENCIÓMETRO**

Un potenciómetro es una resistencia cuyo valor de resistencia es variable. De esta manera, indirectamente, se puede controlar la intensidad de corriente que fluye por un circuito si se conecta en paralelo, o controlar el voltaje al conectarlo en serie. Son adecuados para su uso como elemento de control en los aparatos electrónicos. El usuario acciona sobre ellos para variar los parámetros normales de funcionamiento. Por ejemplo, el volumen de un radio.



Símbolo



Componente

# **1.12 EL ZUMBADOR O BUZZER**

El zumbador, buzzer en inglés, es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono. Sirve como mecanismo de señalización o aviso, y son utilizados en múltiples sistemas como en automóviles o en electrodomésticos. Inicialmente este dispositivo estaba basado en un sistema electromecánico que era similar a una campana eléctrica pero sin el badajo metálico, el cual imitaba el sonido de una campana.

Su construcción consta de dos elementos, un electroimán y una lámina metálica de acero. El zumbador puede ser conectado a circuitos integrados especiales para así lograr distintos tonos. Cuando se acciona, la corriente pasa por la bobina del electroimán y produce un campo magnético variable que hace vibrar la lámina de acero sobre la armadura.







# **1.13 MOTOR DC**

El motor de corriente continua (DC) es una máquina que convierte la energía eléc-trica en mecánica, provocando un movimiento rotatorio. Esta máquina de corriente continua es una de las más versátiles. Su fácil control de posición, paro y velocidad la han convertido en una de las mejores opciones en aplicaciones de con-trol y automatización de procesos. Por ejemplo, los puedes encontrar en la tracción de los carros de juguetes de pilas o en las llantas de los robots



# **1.14 ENTRADAS**

Las entradas o Inputs: Son sensores (o transductores) electrónicos o mecánicos que toman las señales (en forma de tempera-tura, presión, humedad, contacto, luz, movimiento, pH etc.) del mundo físico y las convierten en señales de corriente o voltaje.

Por ejemplo un sensor de temperatura, un pulsador, una fotocelda, un potenciómetro, un sensor de movimiento entre muchos más.





# 1.15 SALIDAS

Las salidas o Outputs: Son actuadores u otros dispositivos (también transductores) que convierten las señales de corriente o voltaje en señales físicamente útiles como movimiento, luz, sonido, fuerza, rotación entre otros.

Por ejemplo: un motor que gire, un LED o sistema de luces que se encienda automáticamente cuando esté oscureciendo, un buzzer que genere diversos tonos.





# **1.16 SISTEMAS ELECTRÓNICOS**

Un sistema electrónico es un conjunto de circuitos que interactúan entre sí para obtener un resultado. Una forma de entender los sistemas electrónicos consiste en dividirlos en entradas, salidas y procesamiento de señal.



Como ejemplo supongamos un televisor. Su entrada es una señal análoga recibida por una antena o por un cable. Los circuitos integrados del interior del televisor extraen la información sobre el brillo, el color y el sonido de esta señal. Los dispositivos de salida son una pantalla LED que convierte las señales electrónicas en imágenes visibles en una pantalla y unos altavoces.

Otro ejemplo puede ser el de un circuito que controle la tempera-tura de un lugar, el sensor de temperatura y el circuito integrado se encarga de convertir la señal de entrada en un nivel de voltaje apropiado y si la temperatura registrada es muy alta el circuito integrado envía la información a un motor para que este encienda el ventilador y refrigere el lugar.

# **1.17 LAS SEÑALES ELECTRÓNICAS**

Las entradas y salidas de un sistema electrónico serán consideradas como las señales variables. En electrónica se trabaja con variables que se toman en forma de voltaje o corriente, éstas se pueden denominar comúnmente señales.

Las señales primordialmente pueden ser de dos tipos descritos a continuación: Digital o análoga



# **1.18 VARIABLE DIGITAL**



También llamadas variables discretas. Se caracterizan por tener dos estados diferenciados y por lo tanto se pueden llamar binarias. Siendo estas variables más fáciles de tratar (en lógica serían los valores Verdadero (V) y Falso (F) o podrían ser 1 ó 0 respectivamente).

Un ejemplo de una señal digital es el interruptor del timbre de tu casa, porque este interruptor tiene dos estados pulsado y sin pulsar.



# **1.19 VARIABLE ANÁLOGA**

Son aquellas que pueden tomar un número infinito de valores comprendidos entre dos límites. La mayoría de los fenómenos de la vida real son señales de este tipo. (sonido, temperatura, voz, video, etc.) Un ejemplo de sistema electrónico analógico es un parlante, que se emplea para amplificar el sonido de forma que éste sea oído por una gran audiencia. Las ondas de sonido que son analógicas en su origen, son capturadas por un micrófono y convertidas en una pequeña variación analógica de tensión denominada señal de audio.



Un ejemplo de sistema electrónico analógico es un parlante, que se emplea para amplificar el sonido de forma que éste sea oído por una gran audiencia. Las ondas de sonido que son analógicas en su origen, son capturadas por un micrófono y convertidas en una pequeña variación analógica de tensión denominada señal de audio.

# 2.- PROGRAMACIÓN

## 2.1 CONCEPTO DE PROGRAMACIÓN

Es el proceso de diseñar, escribir, probar, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación. El propósito de la programación es crear programas que exhiban un comportamiento deseado.

Para crear un programa que el computador interprete y ejecute las instrucciones escritas en él, debe usarse un **Lenguaje de programación**. En sus inicios los computadores interpretaban sólo instrucciones en un



lenguaje específico, del más bajo nivel conocido como código máquina, siendo éste excesivamente complicado para programar. De hecho sólo consiste en cadenas de números 1 y 0 (Sistema binario).

Para facilitar el trabajo de programación, los primeros científicos que trabajaban en el área decidieron reemplazar las instrucciones, secuencias de unos y ceros, por palabras o letras provenientes del inglés, codificándolas así y creando un lenguaje de mayor nivel, que se conoce como Assembly o lenguaje ensamblador.

Por ejemplo, para sumar se usa la letra A de la palabra inglesa add (sumar). En realidad escribir en lenguaje ensamblador es básicamente lo mismo que hacerlo en lenguaje máquina, pero las letras y palabras son bastante más fáciles de recordar y entender que secuencias de números binarios.

## **2.2 LENGUAJE DE PROGRAMACIÓN**

Un lenguaje de programación es un idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como los computadores. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

# **2.3 ALGORITMO**

Un algoritmo es un conjunto pre-escrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.



# **3.-** ENTORNO DE PROGRAMACIÓN ARDUINO

## **3.1 LA PLACA ARDUINO Y SUS PARTES**

Vista frontal





Descripción de las partes:

- 1. Conector USB para el cable tipo AB
- 2. Pulsador de reset
- 3. Pines de entrada y salida digitales y PWM
- 4. LED verde de placa encendida
- 5. LED naranja conectado al pin 13
- 6. ATMEGA 16U2 encargado de la comunicación con el PC
- 7. LED TX (Transmisor) y RX (Receptor) de la comunicación serial
- 8. Puerto ICSP para programación serial
- 9. Microcontrolador ATMEGA 328, cerebro del arduino
- 10. Cristal de cuarzo de 16MHz
- 11. Regulador de voltaje
- 12. Conector hembra 2.1mm con centro positivo
- 13. Pines de voltaje y tierra
- 14. Entradas análogas.

# **3.2 INSTALANDO DRIVERS**

- 3.2.1 Windows 7, XP o VISTA
  - Descargar el software de arduino.cc/en/Main/Software para Windows



- Descomprimir la carpeta de Arduino en una ubicación de fácil acceso
- Conectar la placa Arduino Uno al PC y ver este aviso. No nos debemos pre-ocupar

No se pudo instalar el softw	are de controlador de dispositivo
Póngase en contacto con el fabrican de este dispositivo.	te del dispositivo para obtener ayuda sobre la instalació
Dispositivo no identificado	Asistente cancelado por el usuario

Visitar Panel de control y luego Administrador de dispositivos, allí buscar la siguiente opción.

Otros dispositivos
 Dispositivo desconocido

Click derecho sobre **Dispositivo desconocido** y luego sobre la opción **Actualizar software del controlador.** 

🖃 🦣 Otros dispositivo	s	
⊡ <mark>∭a</mark> Dispositi ⊕ <b>∭</b> Procesador	Actualizar software de controlador	$\mathbf{X}$
🕀 🖓 Puertos (CO	Deshabilitar	63
	Desinstalar	
unidades de	Buscar cambios de hardware	
	Propiedades	

Ingresar a la opción Buscar software de controlador en el equipo

Actualizar software de controlador: Dispositivo desconocido	×
¿Cómo desea buscar el software de controlador?	
Buscar software de controlador actualizado automáticamente Windows buscará en el equipo y en Internet el software de controlador más reciente p dispositivo.	oara el
<ul> <li>Buscar software de controlador en el equipo Buscar e instalar el software de controlador de forma manual.</li> </ul>	
	Cancelar

Examinar y buscar la **carpeta de Arduino** previamente descomprimida en el paso 2. Dentro de esa carpeta acceder a la carpeta Drivers y dar **Aceptar** 



Buscar carpeta	x
Seleccione la carpeta que contiene los controladores pa su hardware.	ara
🛛 🚢 Disco local (C:)	<b>^</b>
SRECYCLE.BIN	
Archivos de programa	
Archivos de programa	
🔺 퉲 arduino-1.0	
Delta Macosx	
drivers	
A second seco	-
Para ver subcarpetas, haga clic en el símbolo junto a ur carpeta.	na
Acepta	ar

Una vez buscamos la carpeta de **Drivers** le damos **Siguiente** 

Busque software de controlador en el equipo Buscar el software de controlador en esta ubicación:	🕒 🚊 Actualizar sof	tware de controlador: Dispositi	vo desconocido	
Busque software de controlador en esta ubicación: Charduino-10/drivers Incluir subcarpetas Cancelar Elegir en una lista de controladores de dispositivo en el equipo Esta lista mostrará el software de controlador instalado compatible con el dispositivo y todo el software de controlador que esté en la misma categoria que el dispositivo. Cancelar Cancelar Cancelar Cancelar Cancelar Cancelar Cancelar Cancelar Cancelar Controlador de Vindows no puede comprobar el editor de este software de controlador Controlador Controlador Controlador Controlador Controlador del dispositivo. Cancelar Canc	Durana anti-			
Buscar el software de controlador en esta ubicación:	Busque softwar	e de controlador en el e	equipo	
Incluir subcarpetas Incluir subcarpetas Incluir subcarpetas Elegir en una lista de controladores de dispositivo en el equipo Esta lista mostrará el software de controlador instalado compatible con el dispositivo y todo el software de controlador que esté en la misma categoria que el dispositivo.   Siguie Cancelar   Cancelar Actualizar software de controlador. Dispositivo desconocido   Siguie Vindows   Instalar Seguridad de Windows   Instalar Seguridad de Windows no puede comprobar el editor de este software de controlador   Instalar No instalar este software de controlador   Debe comprobar el sitio web del fabricante para obtener software actualizado de (controlador del dispositivo.   Instalar Instalar este software de controlador obtenido en el sitio veb o di sco del fabricante. El software no firmado de otras fuentes puede do vel equipo robar información.	Buscar el software d	e controlador en esta ubicació	n: -	Evaminar
<ul> <li>Elegir en una lista de controladores de dispositivo en el equipo Esta lista mostrará el software de controlador instalado compatible con el dispositivo y todo el software de controlador que esté en la misma categoría que el dispositivo.</li> <li>Siguies Cancelar</li> <li>Cancelar</li> <li>Actualizar software de controlador. Dispositivo desconocido</li> <li>Seguridad de Windows</li> <li>Instalar</li> <li>Windows no puede comprobar el editor de este software de controlador</li> <li>No instalar este software de controlador Debe comprobar el sitio web del fabricante para obtener software actualizado del controlador el dispositivo.</li> <li>Instalar este software de controlador de todas formas Sólo instalar el software de controlador de todas formas Solo instalar el software no firmado de otras fuentes puede do cel equipo o robar información.</li> </ul>	Incluir subcarpet	as		Examinar
Siguie Cancelar  Cancelar  Cancelar  Controlador: Dispositivo desconocido  Controlador  Controlador  Controlador  No instalar este software de controlador  Debe comprobar el sitio web del fabricante para obtener software actualizado del controlador del dispositivo.  Instalar este software de controlador de todas formas Sólo instalar este software de controlador de todas formas Sólo instalar este software de controlador de todas formas Sólo instalar este software de controlador de todas formas Sólo instalar este software de controlador obtenido en el sitio web o disco del fabricante.  Lostfware no firmado de otras fuentes puede do Sel equipo o robar información.	<ul> <li>Elegir en u</li> <li>Esta lista mos</li> <li>software de co</li> </ul>	na lista de controladore trará el software de controlado ontrolador que esté en la mism	es de dispositivo e r instalado compatible la categoría que el dispo	n el equipo con el dispositivo y todo el ssitivo.
Actualizar software de controlador. Dispositivo desconocido Instala No instalar este software de controlador Debe comprobar el editor de este software de controlador No instalar este software de controlador Debe comprobar el sitio web del fabricante para obtener software actualizado del controlador del dispositivo. Instalar este software de controlador de todas formas Sólo instalar el software de controlador de todas formas Sólo instalar el software no firmado de otras fuentes puede de quipo o robar información.				Siguien Cancelar
✓ Ver detalles	Contraction of the second seco	ftware de controlador: Disposit ridad de Windows Windows no puede con controlador No instalar este so Debe comprobar el sitio del controlador del dispo functionadar del dispo functionadar del dispo functionadar del dispo functionadar del software ( fabricante. El software ( fabricante. El software ( fabricante. El software ( robar información. er detalles	ivo desconocido aprobar el editor d ftware de controla web del fabricante para sitivo. are de controlador de controlador obtenido e firmado de otras fuent	e este software de dor obtener software actualizado de todas formas en el sitio web o disco del es puede de cale equipo o

Recibimos la confirmación del **Puerto COM** asignado, este número de Puerto COM es muy importante tenerlo en cuenta a la hora de programar





# **3.2 CONOCIENDO EL SOFTWARE ARDUINO**

Para ejecutar el programa Arduino, ingresamos a la carpeta de Arduino y allí buscamos el icono ce Arduino y le damos doble click



### 1.- Puerto COM

Tools Help			
Auto Forma	t Ctrl+T		<u>o</u> ,
Archive Sket	ch		
Fix Encoding	y & Reload		
Serial Monit	or Ctrl+Mayúsc	ulas+M	
Board		•	
Serial Port		•	COM3
Programme	r	•	COM6
Burn Bootlo	' ader		COM7
Bull Bootio	auci	✓	COM30

2.- Seleccionar la placa a trabajar



Tools	Help				
A A F	Auto Format Archive Sketch Fix Encoding & Reload	Ctrl+T			
s	Serial Monitor	Ctrl+Mayúsculas+M		*	
В	Board	+	•	Arduino Uno	
S	Serial Port	•		Arduino Duemilanove w/ ATmega328	
P B	Programmer Burn Bootloader	÷		Arduino Diecimila or Duemilanove w/ AT Arduino Nano w/ ATmega328 Arduino Nano w/ ATmega168	mega168

## 3.- Consola serial

Тоо	ls Help		
	Auto Format	Ctrl+T	
	Archive Sketch		
	Fix Encoding & Reload		
	Serial Monitor	Ctrl+Mayúsculas+M	
	Board	•	
	Serial Port	+	
	Programmer	+	
	Burn Bootloader		

# **3.3 CUERPO DE UN PROGRAMA EN ARDUINO**



Arduino se programa en el lenguaje de alto nivel C/C++ y generalmente tiene los siguientes componentes para elaborar el algoritmo:

- Estructuras
- Variables
- Operadores matemáticos, lógicos y booleanos
- Estructuras de control (Condicionales y ciclos)
- Functiones



## **3.3.1 ESTRUCTURAS**

Son dos funciones principales que debe tener todo programa en Arduino: **SETUP()**{

## }

Código de configuración inicial, solo se ejecuta una vez.

## LOOP(){

}

Esta función se ejecuta luego del setup(), se mantiene ejecutándose hasta que se desenergice o desconecte el Arduino.

## **3.3.2 VARIABLES**

Es un dato o conjunto de datos que cambia su valor con la ejecución del programa.

Booleano	Entero	Carácter
true ó false	Valor entero	Almacena un ASCII
Boolean encendido=true;	int conta=5;	char letra='a';

## **3.3.3 OPERADORES BOOLEANOS**

Usados generalmente dentro del condicional if

- && (y)
  || (o)
  If (a || b)
- ! (negación)

## **3.3.4 OPERADORES DE COMPARACIÓN**

Ing. Rubén Eusebio Acosta Jacinto



### Usados generalmente dentro del condicional If y sobre el For y While

- == (igual a)
- != (diferente de)
- (menor que)

>= (mayor o igual)

• > (mayor que)

# lf (a == b)

### 3.3.5 OPERADORES MATEMÁTICOS

Se aplican al manejo de variables, condicionales y ciclos

- = (asignar)
  - % (módulo)
- + (suma)

#### / (división)

int valor = valor +5

(resta)

\* (multiplicación)



#### 3.3.6 ESTRUCTURAS DE CONTROL

Son instrucciones que nos permiten tomar decisiones y hacer diversas repeticiones de acuerdo a unos parámetros, dentro de las más importantes podemos destacar: lf, Switch/case, for y while 3.3.7 CONDICIONALES

Ideales para tomar decisiones luego de evaluar las condiciones lógicas:



## 3.3.8 CICLOS

Ideales para repetir lo que se encuentre dentro de ellos



#### 3.3.9 FUNCIONES

Una función es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Las funciones pueden tomar parámetros que modifiquen su funcionamiento. Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para imple-mentar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código.

Cuando una función es invocada se le pasa el control a la misma, una vez que ésta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.





### 3.3.10 FUNCIONES DIGITALES

Orientas a revisar el estado y la configuración de las entradas y salidas digitales



### 3.3.11 FUNCIONES ANÁLOGAS

Ideales para la lectura y escritura de valores análogos

analogRead()	analogWrite() -> PWM
Leer un valor análogo 0 a 1023	Escribir un valor análogo 0 a 255
analogRead(pin)	analogWrite(pin,valor de PWM)
int a = analogRead (A0);	analogWrite (9, 134);



#### 3.3.12 REFERENCIA RÁPIDA PARA PROGRAMAR

El programa de Arduino se puede dividir en tres partes principales: la estructura, las variables (valores y constantes) y funciones.

- **ESTRUCTURAS** 
  - setup()
  - loop()

#### +Estructuras de control

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

#### -Sintaxis

- -; (punto y coma)
- { } (llaves)
- / / (comentario de una sola línea)
- / \* \* / (comentario de varias líneas)
- # define
- # include

#### +Operadores matemáticos

- = (operador de asignación)
- + (suma)
- - (resta)
- \* (multiplicación)
- / (división)
- % (módulo)

#### +Operadores de comparación

- == (igual que)
- ! = (diferente de)
- < (menor que)
- > (mayor que)
- <= (menor o igual a)
- >= (mayor o igual a)

#### +Operadores booleanos

- -&& (y)
- | | (o)
- -! (no)

#### +Acceso con apuntadores

- \* eliminar la referencia del operador
  - & operador de referencia

#### +Operadores bit a bit

- & (bit a bit AND)
- | (bit a bit OR)
- ^ (bit a bit XOR)
- -~ (bit a bit NOT)
- << (a la izquierda BitShift)
- >> (a la derecha BitShift)

#### +Operadores compuestos

- + + (incremento)
- -- (decremento)
- + = (compuesto adición)
- - = (compuesto substracción)
- \* = (compuesto multiplicación)
- / = (compuesto división)
- -& = (compuesto bit a bit AND)
- | = (compuesto bit a bit OR)

### **VARIABLES**

#### +Constantes

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- Constantes enteras
- Constantes flotante
- +Tipos de datos
- void
- boolean
- char

- byte
- int
- word
- long
- unsigned long
- float
- double
- string arreglo char
- String objeto
- array

### +Utilidades

sizeof()

### +Conversión

- char()byte()
- int()
- word()
- -----
- long()
- float()

21



## **FUNCIONES**

## +Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

## +Analogo I/O

- analogReference()
- analogRead()
- analogWrite() PWM

## +Avanzadas I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- -pulseIn()

## +Tiempo

- millis()
- micros()
- delay()
- delayMicroseconds()

## +Matemáticas

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

## +Números aleatorios

- randomSeed()
- random()

### +Bits y Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

## +Interrupciones externas

- attachInterrupt()
- detachInterrupt()

### +Interrupciones

- interrupts()
- noInterrupts()

## +Comunicación

- Serial
  - begin()
  - end()
  - available()
  - read()



# 4.- CARGANDO MI PRIMER PROGRAMA

1 Para ello abre el software de Arduin	o y realiza l	o que se indica	en la siguiente imagen:
--	---------------	-----------------	-------------------------

🥯 sl	ketch_jan17a   Arduino 1.0				
File	Edit Sketch Tools Help				
	New	Ctrl+N			
	Open	Ctrl+O			
	Sketchbook				
	Examples	•	1.Basics	•	AnalogReadSerial
	Close	Ctrl+W	2.Digital	•	BareMinimum
	Save	Ctrl+S	3.Analog		Blink 🔆
	Save As	Ctrl+Mayúsculas+S	4.Communication	•	DigitalReadSerial
	Upload	Ctrl+U	5.Control	•	Fade
	Upload Using Programmer	Ctrl+Mayúsculas+U	6.Sensors	•	
			7.Display	•	

2.- Una vez cargado el programa Blink en el IDE de Arduino, verificar en la pestaña HERRAMIENTAS Que en: Tarjeta>Arduino uno

Que en: Puerto Serial> COM42 o el COM destinado al USB-SERIAL de Arduino.

Archivo Editar Sketch Herramientas Ayuda
Blink
*/
<pre>// Pin 13 has an LED connected on most Arduino boards. // give it a name: int led = 13;</pre>
<pre>// the setup routine runs once when you press reset: void setup() {     // initialize the digital pin as an output. pinMode(led, OUTPUT);</pre>
}
<pre>// the loop routine runs over and over again forever: void loop() {</pre>
<pre>digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level) delay(1000); // wait for a second</pre>
<pre>digitalWrite(led, LOW); // turn the LED off by making the voltage LOW     delay(1000); // wait for a second</pre>

3.- Verificar si el LED se enciende y apaga cada segundo





# **5.-** TUTORIALES

# **5.1LED INTERMITENTE**

- Conocimientos previos
  - ✓ Señal digital
    - ✓ Función digitalwrite()
    - ✓ Polaridad de un LED
    - ✓ Conexión de la placa arduino al ordenador.
- ¿Qué aprendo?
  - ✓ Activar una salida digital
  - ✓ Encender un LED en ON/OFF
  - ✓ Temporizar una señal de salida
  - ✓ Sintaxis de un programa en Arduino

# 5.1.1PLANO



# 5.1.2ESQUEMA





# 5.1.3CÓDIGO DE PROGRAMACIÓN

```
6
   /*
7
8
   Hola Mundo
9
10
  Enciende un LED por un segundo y lo apaga por el mismo tiempo
11
   */
12
   //-----
                          _____
13
  //Función principal
14 //-----
                       _____
15 void setup() // Se ejecuta cada vez que el Arduino se inicia
16 {
17 pinMode(13,OUTPUT); // Inicializa el pin 13 como una salida
18 }
19 //-----
20 //Función cíclica
21 //-----
22 void loop() // Esta función se mantiene ejecutando
23 { // cuando este energizado el Arduino
24 digitalWrite(13,HIGH); // Enciende el LED
25 delay(1000); // Temporiza un segundo (1s = 1000ms)
26 digitalWrite(13,LOW); // Apaga el LED
  delay(1000); // Temporiza un segundo (1s = 1000ms)
27
28
  }
29 // Fin del programa
```

# 5.1.4 EJERCICIOS

a) A partir del código de programación explicado anteriormente, queremos que un LED prenda de manera intermitente de la siguiente manera:

- **3** segundos prendido y 3 segundos apagado
- 200 milisegundos prendido y 500 milisegundos apagado



b) La empresa de automatización NRJ Inc. te contrata para hacer un montaje de un LED Intermitente en una placa Arduino UNO, con el único requisito de que el LED debe estar ubicado en el Pin 5, ellos muy amablemente te han facilita-do el esquema, tu tarea es:

A) Realizar el montaje y la respectiva programación de 2 segundos prendido y 1 segundo apagado



# 5.2 ENCENDER UN LED CON UN PULSADOR

- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalWrite() y digitalRead()
  - ✓ Divisor de voltaje
  - ✓ Condicional y operadores de comparación
- ¿Qué aprendo?
  - ✓ Cablear un circuito
  - ✓ Condicional If/else
  - ✓ Estado de un pulsador
  - ✓ Leer una entrada digital y escribir una salida digital
- 5.2.1 Plano





## 5.2.3 Código de programación

- 6 /\*
- 7 -----
- 8 Encender LED con un pulsador 9 ------
- 10 Oprimir un pulsador y mientras este se mantenga accionado
- 11 un LED se enciende
- 12 \*/
- 13 //-----
- 14 //Declara puertos de entradas y salidas
- 15 //-----
- 16 int pulsador=2; //Pin donde se encuentra el pulsador, entrada
- 17 int led=13; //Pin donde se encuentra el LED, salida
  18 //-----
- 18 //-----19 //Funcion principal
- 20 //-----



- 21 void setup() // Se ejecuta cada vez que el Arduino se inicia
- 22
- 23 pinMode(pulsador, INPUT); //Configurar el pulsador como una entrada
- pinMode(led,OUTPUT); //Configurar el LED como una salida 24
- 25 }
- 26 //--\_\_\_\_\_
- //Funcion ciclicla 27 //-----
- 28 29 void loop() // Esta funcion se mantiene ejecutando
- { // cuando este energizado el Arduino 30
- 31 //Condicional para saber estado del pulsador
- 32 if (digitalRead(pulsador)==HIGH)
- 33

{

- 34 //Pulsador oprimido
- digitalWrite(led,HIGH); //Enciende el LED 35
- 36 }
- 37 else
- 38
- { 39 //Pulsador NO oprimido
- digitalWrite(led,LOW); //Apaga el LED 40
- 41

}

}

- 42
- 43 //Fin programa

#### **Ejercicios**: 5.2.3

La multinacional francesa Flante experta en maquinaria industrial te ha contratado para que automatices una máquina cortadora de papel. La condición principal es que el operario de la máquina cuando vaya a realizar el corte siempre mantenga las dos manos ocupadas, esta es una regla de seguridad industrial para evitar accidentes. El operario debe oprimir los dos pulsadores uno con cada mano y la cuchilla cortadora debe bajar y hacer el corte. El siguiente montaje simula el control de la máquina, los dos pulsadores (S1 y S2) y el LED rojo simula la cuchilla cortadora.





Tu padre quiere que realices un sistema de iluminación LED para las escaleras de la casa. La condición es que si estás arriba y pulsas a S1 o si estás abajo y pulsas S2 el LED Rojo se enciende y al dejar de pulsar se apaga. Como guía de montaje toma la imagen anterior.



# 5.3 LECTURA SERIAL DE UNA ENTRADA DIGITAL

- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalRead() y Serial.println()
  - ✓ Opción de Consola serial,
- ¿Qué aprendo?
  - ✓ Manejar una entrada digital
  - ✓ Ver datos por la pantalla del computador
  - $\checkmark$  Consola serial
  - ✓ Leer una entrada digital y escribir por consola serial
- 5.3.1 Plano



5.3.2 Esquema





5.3.3 Código de programación \_\_\_\_\_ Lectura serial de una entrada digital \_\_\_\_\_ Leer una entrada digital y mostrar por la pantalla del computador (consola serial) el estado del pulsador cuando es oprimido. \*/ //-----//Declara puertos de entradas y salidas //\_\_\_\_\_ int boton=2; //Pin donde se encuentra el pulsador, entrada //\_\_\_ \_\_\_\_\_ //Funcion principal //----void setup() // Se ejecuta cada vez que el Arduino se inicia //Configuración pinMode(boton,INPUT); //Configurar el boton como una entrada Serial.begin(9600); //Inicia comunicación serial } //-----//Funcion ciclicla //----void loop() // Esta funcion se mantiene ejecutando { // cuando este energizado el Arduino //Guardar en una variable entera el valor del boton 0 ó 1 int estado = digitalRead(boton); //Condicional para saber estado del pulsador if (estado==1) // Pulsado Serial.println("Pulsado"); //Imprime en la consola serial } // "Pulsado" else // No esta pulsado Serial.println("NO Pulsado"); //Imprime en la consola serial } // "NO Pulsado" delay (100); //Retardo para la visualización de datos en la consola //Fin programa

#### 5.3.3 Ejercicios

Unas luces navideñas modernas son las siguientes: mientras se mantenga pulsado S1 una variable entera inicializada en 0 se comienza a incrementar de 20 en 20, al soltar S1 la variable para de incrementarse, cuando se pulse S2 el valor de la variable se debe cargar a un Hola Mundol del LED, esto quiere decir que el LED va estar intermitente en intervalos de tiempo iguales al valor de la variable. Por consola serial debes ir mostrando el valor de la variable. Para que puedas volver la variable a 0 y puedas hacer otra rutina de intermitencia, coloca la condición de que S1 y S2 se opriman al tiempo, con ello la variable entera debe volver a 0.



# 5.4 LECTURA SERIAL DE UNA ENTRADA ANÁLOGA

- Conocimientos previos
- ✓ Señal análoga
- ✓ Función analogRead() y Serial.println()
- ✓ Opción de Consola serial
- ¿Qué aprendo?
- ✓ Manejar una entrada análoga
- ✓ Ver datos por la pantalla del computador
- ✓ Múltiples estados de un potenciómetro
- ✓ Leer una entrada análoga
- 5.4.1 Plano



5.4.2 Esquema





- 5.4.3 Código de programación
- 6
- /\* 7 \_\_\_\_\_
- 8 Lectura serial de entrada análoga
- 9 \_\_\_\_\_
- 10 Leer una entrada análoga y mostrar por la pantalla del
- 11 computador (consola serial) el valor luego de girar
- 12 el potenciómetro
- 13 \*/
- 14 //---\_\_\_\_\_
- 15 //Funcion principal
- //-----16
- 17 void setup() // Se ejecuta cada vez que el Arduino se inicia
- 18 {
- 19 Serial.begin(9600); //Inicia comunicación serial
- 20 }
- 21 //--
- 22 //Función ciclicla
- 23 //-----
- void loop() // Esta funcion se mantiene ejecutando 24
- 25 { // cuando este energizado el Arduino
- 26 //Guardar en una variable entera el valor del potenciómetro 0 a 1024
- 27 int valor= analogRead(A0);
- 28 //Imprime en la consola serial el valor de la variable
- 29 Serial.println(valor);
- 30 //Retardo para la visualización de datos en la consola
- 31 delay(100);
- 32 }
- //Fin programa 33
- 5.4.4 Ejercicios



Piensa en un número secreto de 0 a 1023 y guárdalo en una variable entera, haz un programa para que tu amigo deba mover el potenciómetro, si el número que él va generando (mostrar por consola serial el valor de la lectura del potenciómetro) al girar el potenciómetro está 10 números por arriba o 10 números por abajo al número que tu pensaste y confirma con el pulsador S1 que ese es el número, el programa debe mostrar por consola "Adivinaste el número" de lo contrario "Intenta de nuevo".

Este caso podría ser un ejemplo para que lo apliques a una Caja de seguridad, para que guardes mucho dinero.



## **5.5 ESCRITURA SERIAL**

### Conocimientos previos

- ✓ Señal digital
  - ✓ Función digitalWrite() y Serial.read()
  - ✓ Configuración de una comunicación serial
  - ✓ Polaridad de un LED
- ¿Qué aprendo?
  - Entrada por consola (teclado)
  - ✓ Variables booleanas
  - ✓ Estado de un LED
  - $\checkmark$  Escritura serial digital
- 5.5.1 Plano



5.3.2 Esquema





- 5.3.3 Código de programación
- 6 /\*
- 7 -----
- 8 Escritura serial
- 9 -----
- 10 Consiste en escribir por la pantalla del computador (consola serial) una letra predeterminada, la primera vez que se escriba está un LED se enciende, si se vuelve a escribir por segunda vez el LED se apaga.
- 11 \*/
- 12 //-----
- 13 //Declara puertos de entradas y salidas y variables
- 14 //-----
- 15 int led = 13; //Pin donde se encuentra el LED, salida
- 16 char leer; //Variable donde se almacena la letra
- 17 boolean prendido=false; //Estado LED la primera vez, apagado
- 18 //-----
- 19 //Funcion principal
- 20 //-----
- 21 void setup() // Se ejecuta cada vez que el Arduino se inicia
- 22

{

- 23 Serial.begin(9600); //Inicia comunicación serial
- 24 pinMode(led, OUTPUT); //Configurar el LED como una salida
- 25 }
- 26 //-----
- 27 //Funcion ciclicla
- 28 //-----
- 29 void loop() // Esta funcion se mantiene ejecutando
- 30 { // cuando este energizado el Arduino
- 31 //Guardar en una variable el valor de la consola serial
- 32 leer=Serial.read();
- 33 // Si es la letra 'a' y además el LED está apagado
- 34 if ( (leer=='a') && (prendido==false) )
- 35 {
- 36 digitalWrite(led,HIGH); // Enciende el LED
- 37 prendido=true; // Actualiza el estado del LED
- 38

}

{

- 39 // Si es la letra 'a' y además el LED está encendido
- 40 else if ( (leer=='a') && (prendido==true) )
- 41
- 42 digitalWrite(led,LOW); // Apaga el LED



43 prendido=false; // Actualiza el estado del LED
44 }
45 }
46 //Fin programa
5.3.4 Ejercicios
Avisos luminosos S A te ha contratado para que

Avisos luminosos S.A. te ha contratado para que realices un programa que tiene la característica de recibir datos, porque la idea es que el aviso de luces se ubique en el piso 130 del Edificio Bulli y tu lo puedas controlar desde el piso 1, así que por cada letra que le escribas por teclado a la Consola serial el programa debe hacer determinada rutina con tres LEDs, si le escribes:

-Letra A: Prende los tres LED

**-Letra B:** Prende el LED1 por un segundo y lo apaga, luego prende el LED2 por un segun-do y lo apaga y finalmente prende el LED3 por un segundo y lo apaga, y vuelve a comenzar. Esta rutina genera una sensación de movimiento

-Letra C: Apaga los tres LED



# 5.6 ENCENDER UN LED POR PWM

- Conocimientos previos
  - ✓ PWM
  - ✓ Función analogWrite()
  - ✓ Polaridad de un LED
  - ✓ Incrementar
- ¿Qué aprendo?
  - $\checkmark$  Encender un LED de manera proporcional
  - ✓ Apropiar el concepto de PWM
  - ✓ Escribir una salida análoga
  - ✓ If/else con operadores lógicos

5.6.1 Plano





5.6.3código de programación

/\*

Enciende/Apaga un LED de forma proporcional

-----

Programa que enciende proporcionalmente un LED cuando llega a su máximo punto de brillo comienza a apagarse proporcionalmente. \*/

//-----//Declara puertos de entradas y salidas y variables

//-----

int brillo = 0; //Variable de brillo inicia en 0

int variacion = 5; //Variable de incremento configurada de 5 en 5

int led = 9; //Pin donde se encuentra el LED, salida

//-----//Funcion principal

//------

void setup () // Se ejecuta cada vez que el Arduino se inicia

{ pinMode(led, OUTPUT); //Configurar el LED como una salida

}

//-----//Funcion ciclicla

//-----

void loop () // Esta funcion se mantiene ejecutando



{ // cuando este energizado el Arduino // Escritura analoga (PWM) en el LED escribo el valor de brillo analogWrite(led, brillo); // Incremento la variable brillo de 5 en 5 brillo = brillo + variacion; // Nota: PWM ----> 0 - 255 // Si el brillo es 0 o 255 if (brillo == 0 || brillo == 255) variacion = -variacion; //La variación se vuelve negativa delay (30); //Tiempo de incremento en el brillo } //Fin programa

### 5.6.4 Ejercicios

Tu mamá quiere tener unas luces de intensidad variables en la sala de tu casa, quiere poca luz para los momentos en que la familia comparte para mantener la armonía pero requiere bastante luz cuando realiza sus tejidos, ella quiere tener el control de la intensidad de luz mediante un potenciómetro y que la bombilla LED se ilumine de acuerdo a lo graduado por ella. Tu tarea es realizar el programa que al leer una entrada análoga de un potenciómetro (0—1023) se ajuste el valor de PWM (0-255) de un LED, recuerda escalar el valor... map()



# 5.7 CONTROL ON/OFF POR POTENCIÓMETRO

- Conocimientos previos
  - ✓ If/else
  - Función digitalWrite() y analogRead()
  - ✓ Valor de una entrada análoga
  - ✓ Condicional y operadores
- ¿Qué aprendo?
  - ✓ Salida digital
  - ✓ Control ON/OFF
  - ✓ Comparación
  - ✓ Condicional a partir de un valor entero de una entrada análogo





- 5.7.3 Código de programación
- 6 /\*

- 7
  - --------
- 8 Control ON/OFF con potenciómetro
- 9 \_\_\_\_\_
- 10 Programa que enciende un LED cuando el valor de la entrada análoga comandada por el potenciómetro esta en cierto valor,
- cuando este valor cambia el LED se apaga, es un sistema con 11
- control ON/OFF 12
- 13 \*/
- 14 //-
- 15 //Función principal
- 16 //--
- 17 void setup() // Se ejecuta cada vez que el Arduino se inicia
- 18

{

}

- 19 Serial.begin(9600); //Inicia comunicación serial
- 20 pinMode(13,OUTPUT); //Configurar el pin 13 como una salida
- 21



- 22 //-----
- 23 //Funcion ciclicla
- 24 //-----
- 25 void loop() // Esta funcion se mantiene ejecutando
- 26 {// cuando este energizado el Arduino
- 27 //Guardar en una variable el valor de la lectura análoga
- 28 int valor = analogRead(A0);
- 29 Serial.println(valor); //Imprime el valor por la consola
- 30 //Si el valor es mayor o igual a 500
- 31 if (valor >= 500) 32 {
- 32
- 33 digitalWrite(13,HIGH); //Enciende el LED en el pin 13
- 34
- 35 //Si el valor es menor a 500
- 36 else

{

ł

- 37
- 38 digitalWrite(13,LOW); //Apaga el LED en el pin 13
- 39
- 40 delay(100); //Retardo de 100ms para ver los datos de la consola
- 41
- 42 //Fin programa

#### 5.7.4 Ejercicios

Un carro de tunnig tiene luces de neón en la parte de abajo, éste es la sensación de la feria. El piloto controla las luces (tres LEDs) por medio de un potenciómetro al lado de la palanca de cambios, él tiene tres opciones de luces de acuerdo al valor de lectura del potenciómetro que va de 0 a 1023:

- ✓ 0 350 Intermitencia de los 3 LED a 100ms
- ✓ 351 700 Intermitencia de los 3 LED a 500ms
- ✓ 701 1023 Intermitencia de los 3 LED a 1000ms



# 5.8 CONTADOR DE PULSOS

- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalWrite() y digitalRead()
  - ✓ Imprimir datos por consola
  - ✓ Declarar variables enteras
- ¿Qué aprendo?

- ✓ Incrementar una variables
- Condicional If/else anidado



✓ Anti-rebote de un pulsador

✓ Leer una entrada digital y escribir una salida digital a determinada condición

#### 5.8.1 Plano





pinMode(2,INPUT); //Configura el pin 2 como una entrada, pulsador pinMode(13,OUTPUT); //Configura el pin 13 como una salida, LED

//-----

```
//Funcion ciclicla
//_____
void loop() // Esta funcion se mantiene ejecutando
{ // cuando este energizado el Arduino
// Si el pulsador esta oprimido
if (digitalRead(2) == HIGH)
// Si el pulsador no esta oprimido, flanco de bajada
if (digitalRead(2) == LOW)
conta++; //Incrementa el contador
Serial.println(conta); //Imprime el valor por consola
delay (100); // Retardo
// Si el valor del contador es 5
if (conta==5)
digitalWrite(13,HIGH); //Enciende el LED
}
// Si el valor del contador es 8
if (conta==8)
{
digitalWrite(13,LOW); // Apaga el LED
//Fin programa
5.8.4 Ejercicios
```

El DJ Lui Lote del ejercicio 1 regreso recargado con nuevas ideas para su Show. Ahora cuenta con tres distintas zonas de iluminación para su pista de baile, en su consola de DJ tiene un botón para dar un cierto número de pulsos y luego por la Consola Serial él dice a qué zona de iluminación se le asignan esos pulsos a intervalos de 1s. Las palabras que recibe la Consola Serial son: **zona1 zona2 zona3** 



# 5.9 CONTROL ON/OFF DE UN MOTOR DC

- Conocimientos previos
  - ✓ Señal digital
  - Función digitalWrite() y digitalRead()
  - ✓ Divisor de voltaje
  - ✓ Reconocer un transistor y un motor
- ¿Qué aprendo?



- ✓ Control ON/OFF
- ✓ Condicional If/else
- ✓ Conectar un motor DC por transistor
- ✓ Condicionales a partir del estado del pulsador





5.9.3Código de programación

-----

Control ON/OFF de un motor

/\*Programa que hace uso de un motor y un pulsador, mientras se mantenga pulsado, el motor debe estar encendido (ON) de lo contrario debe estar apagado (OFF)

\*/

//-----

//Declara puertos de entradas y salidas y variables

//-----

int pulsador =7; //Declara Pin del pusaldor

int motor=3; //Declara Pin del motor //-----

//Funcion principal

//-----void setup() // Se ejecuta cada vez que el Arduino se inicia



{
pinMode(pulsador,INPUT); //El pulsador como una entrada
pinMode(motor,OUTPUT); //El motor como una salida
}
//-----//Funcion ciclicla
//-----void loop() // Esta funcion se mantiene ejecutando
{ // cuando este energizado el Arduino
// Si el pulsador se encuentra oprimido
if(digitalRead(pulsador) == HIGH){
digitalWrite(motor,HIGH); //Enciende el motor
}else{ //si el pulsador no esta oprimido
digitalWrite(motor,LOW); //Apaga el motor
}

// Fin programa

#### 5.9.4 Ejercicios

Al mercado ha salido un nuevo detergente para lavadoras, con una gran novedad, si cuando uses el deterge en tu lavadora esta bate hacia la derecha (**D**) a determinado tiempo es posible remover las manchas en la ropa mientras que si la lavadora bate a la izquierda (**I**) a determinado tiempo al final del ciclo de lavado la ropa tendrá una textura muy sedosa como si se le hubiera aplicado suavizante. La empresa **Lucky Luc** gran fabricante de lavadores quiere implementar este sistema, para ello te ha contratado para que desarrolles el programa que de acuerdo a lo que se reciba por consola se debe operar la lavadora, por ejemplo:

- **D,3** Lavadora bate a la derecha por 3 segundos
- **I,2** Lavadora bate a la izquierda por 2 segundos
- A Lavadora apagada



## **5.10** CONTROL POR PWM DE UN MOTOR

- Conocimientos previos
  - ✓ Señal análoga
     ✓ Función analog
    - Función analogWrite() y Serial.print()
    - PWM
    - Condicional y operadores de comparación
- ¿Qué aprendo?



- ~
- 1
- √
- Leer datos de la Consola Serial Manejo de la función map() Variar el PWM para producir 5 velocidades distintas Etapa de potencia para un motor a través de transistor ✓





5.10.2 Esquema



5.10.3 Código de programación /\* \_\_\_\_\_ Control por PWM de un motor -----



Programa que hace uso de un motor y la Consola serial de Arduino, tiene la posiblidad de configurar al motor 5 velocidades distintas, desde el teclado del PC puedes enviarle la velocidad deseada. Las 5 velocidades se configuran con 5 PWM distintos. \*/

//-----//Declara puertos de entradas y salidas y variables

//-----

int motor=3; //Declara Pin del motor

//-----//Funcion principal

//------

void setup() // Se ejecuta cada vez que el Arduino se inicia

Serial.begin(9600); //Inicia la comunicacion serial Arduino-PC

,

//Funcion ciclicla

//-----

void loop() // Esta funcion se mantiene ejecutando

{ // cuando este energizado el Arduino

// Si hay algun valor en la Consola Serial

if (Serial.available()){

//Variable donde se guarda el caracter enviado desde teclado

char a = Serial.read();

// Si el caracter ingresado esta entre 0 y 5  $\,$ 

if (a>='0' && a<='5'){

//Variable para escalar el valor ingresado a rango de PWM

int velocidad = map(a,'0','5',0,255); //Escritura de PWM al motor

analogWrite(motor,velocidad);

//Mensaje para el usuario

Serial.print("El motor esta girando a la velocidad ");

Serial.println(a):

}else{ // Si el caracter ingresado NO esta entre 0 y 5

//Mensaje para el usuario

Serial.print("Velocidad invalida");

Serial.println(a);

} }

//Fin programa

### 5.10.4 Ejercicios

La última novedad de la Feria de Electrodomésticos ha dejado con gran re-nombre una licuadora que incorpora dos pulsadores mediante los cuales es posible variar las 10 diferentes velocidades (0 a 9) al momento de licuar. Todo esto se apoya desde la Consola serial donde es posible ver la Velocidad escogida, los pulsadores se ajustan de la siguiente manera:

Pulsador 1 (S1) permite escoger una velocidad dentro de un menú (Velocidad 0 ... Velocidad 9), cada vez que se pulse S1 el texto por Consola Serial que dice la velocidad debe cambiar, debe tener antirebote para que por cada pulso dado solo cambie una velocidad. Un ejemplo del pulso dado y el valor mostrado por consola:

Pulso 1 -> Velocidad 0 Pulso 2 -> Velocidad 1

Pulso 3 -> Velocidad 2

Pulsador 2 (S2) al pulsar S2 por primera vez el motor debe comenzar a girar a la velocidad escogida mediante S1, si se pulsa por segunda vez S2 el motor debe parar. Por Consola serial se debe mostrar "Licuadora girando a la velocidad x" o "Licuadora detenida" según corresponda.





# 5.11 GENERAR TONOS POR UN BUZZER

- Conocimientos previos

  - ~ ~
  - Señal digital y análoga Función map() y analogRead() Enviar parámetros a las funciones Retardos a través de delay() ~
- ¿Qué aprendo?
  - 1 Manejo de variables de tipo entera
  - ~ Usar funciones especiales de Arduino
  - Generar diversos tonos
  - Producir salidas de frecuencia





### 5.12.2Esquema





5.12.3Código de programación

·

Generar tonos con un buzzer

\_\_\_\_\_

Programa que hace uso de un buzzer (chicharra) y un potenciómetro, la idea es generar diversos tonos en el buzzer a partir del estado análogo del potenciómetro. Además se hace uso de la función tone que es muy útil para generar diversas melodías

\*/ //---

//Declara puertos de entradas y salidas y variables

int buzzer = 9; //Declara Pin del buzzer

int tono = 0; //Declara Pin del potenciómetro

//-----

//Funcion principal

//-----

void setup() // Se ejecuta cada vez que el Arduino se inicia

{

 $/\!/$  No se configuran parámetros inciales, pero se debe

// colocar el encabezado de la función setup()

}

. //-----

//Funcion ciclicla

//-----

void loop() // Esta funcion se mantiene ejecutando

{ // cuando este energizado el Arduino

//Variable entera donde se almacena el valor del potenciómetro

int sensor = analogRead(tono);

//Variable donde se escala la frecuencia de 100 a 5000Hz

int frecuencia = map(sensor,0,1023,100,5000);

//Variable entera para guardar el tiempo deseado en ms

int duracion = 250;

//Funcion tone(), que recibe:

// 1ra posición: Pin del elemento sonoro

// 2<br/>da posición: Frecuencia deseada en Hz

// 3ra posición: Duración del tono

tone(buzzer, frecuencia, duracion);

//Retardo

delay(100);
}



### //Fin programa

### 5.12.4Ejercicios

Te proponemos hacer un sintetizador musical de dos tonos, usando dos pulsadores. La idea es que al pulsar S1 se produzca un tono a una frecuencia de 400Hz con una duración de 100ms, mientras si presionas S2 se produce un tono a frecuencia de 2600Hz con una duración de 200ms. Combina estos dos tonos hasta que desarrolles una bonita melodía.





# 6.-ARDUINO BOOTLOADER

El Bootloader es un pequeño programa que viene grabado de fábrica en la Flash y ocupa 512 bytes, aunque en otros modelos, como la Leonardo ocupa 4Kilobytes. Su función es gestionar el proceso de grabación de la memoria Flash del programa que queremos ejecutar, es decir recibe nuestro programa del IDE y lo almacena en la Flash, tras esto el bootloader termina su trabajo y el microcontrolador ejecuta de forma permanente el programa. En la Arduino Uno, el bootloader se ejecuta en el primer segundo de cada reinicio, periodo en el cual el bootloader espera respuesta del IDE para cargar algún programa, si esta respuesta no llega acaba su ejecución y se ejecuta lo que haya en la Flash. Si compramos un ATmega328P por separado, no tendrá bootloader, deberemos hacerlo a mano o no usarlo y cargar los programas a mano, para lo cual necesitaremos un programador ISP, el cual hay que conectar al ordenador y a la placa Arduino, haciendo de intermediario. En el instalador del IDE de Arduino hay copias de los diferentes bootloaders oficiales para según que placas en formato .hex.

#### **6.1 MATERIALES EMPLEADOS:**

- Arduino Uno R3 con chip Atmega 328P
- Chip Atmega 328P-PU
- Resistencia 10K
- Cristal de 16Mhz
- Condensador cerámico de 22pf (\*\*\*opcional)
- Arduino IDE 1.0.5-r2

#### Ref:

http://www.instructables.com/id/Burning-the-Bootloader-on-ATMega328-using-Arduino-/all/?lang=es

### **6.2 SOFTWARE**

Paso1: Conectar la tarjeta del Arduino UNO a la computadora y verificar que es detectado por el IDE.

💿 ArdunoISP   Arduno 1.0.5-	r7	x	tios
Archivo Editar Sketch Herra	mientas] Ayuca		
ArduinoISP vers     // ArduinoISP vers     // Copyright (c) 2	Formato Auto mático Ctrl + T Archiva : el Sketch Reperar Codificación y Fezargar Monitor Senal Ctrl + Mayúscu a: + M	•	<u>4+++</u>
// If you require	Taijsla 🕨		
// <u>http://www.</u>	Puerto Serial 🔸		COM1
// This sketch tur // using the follo	Programador Craba: Secuencia de Inicio		COM3 COM6
// nin name:	ma: memail281 and 2560)		COM7
// slave reset: IN:	53		COM10
77 MDST: 11:	.5 1		COM11
// MTS0: 12:	50		COM12
77 DUX: 101	22		COM13
// Put an LED (with re:	sistor) on the following pins:		COM14
// 9: Heartbeat - sn	ovs the programmer is running		COM15
// 8: Error - L1	ghts up it scmething goes wicng juse red i:		COM16
// /: Frogramming - In	communication with the Slave		COM17
			COM18
			COM19
			COM20
			COM21
			COM22
		<b>v</b> ]	COM34
9	Arduino Uno on COM3	4	



Archivo Editar Ski Nuevo Abrir Sketchbook Ejemplos	etch Herramientas Ayuda	-0		L
Nuevo Abrir Sketchbook Ejemplos				
Abrir Sketchbook Ejemplos	Ctrl+N	01.Basics	<b>,</b> •	
Sketchbook Ejemplos	Ctrl+0	02.Digital		
Ejemplos		• 03.Analog	,	
		04.Communi	cation 🕨 着	
Cerrar	Ctrl+W	05.Control	•	
Guardar	Ctrl+S	06.Sensors	•	
Guardar como	Ctrl+Mayúscu	las+S 07.Display	•	
Cargar	Ctrl+U	08.Strings	+	
Cargar usando	o Programador 🛛 Ctrl + Mayúscu	Ias+U 09.USB	+	
Configuración	n de Página Ctrl + Mavúscu	las + P 10.StarterKit	•	
Imprimir	Ctrl+P	ArduinoISP		
Defenseries	Ct   C	EEDDOM		
Preterencias	Ctri+Comma	Esplora	, 	
Salir	Ctrl+Q	Ethernet	•	
// 9: Heartbeat // 8: Error	<ul> <li>shows the programm</li> <li>Lights up if somet</li> </ul>	hing Firmata	•	
// 7: Programmi	ing - In communication w	with t GSM	•	
		LiquidCrystal	►	
<	III.	Robot_Contro	ol 🕨	
		Robot_Motor		
		SD	•	
		Servo	•	
		SoftwareSeria	d ►	
		SPI	• <b>•</b>	
1		Stepper	•	]
		TFT	•	
		WiFi	•	
	U 🔍 🔍 🛄	🦉 Wire	•	
				201 201
Ardumoise				
// ArduinoISP version	04m3			<u>^</u>
// Copyright (c) 2008-	2011 Randall B	ohn		
// If you require a li	.cense, see			
// http://www.open	source.org/lic	enses/bsd-lic	ense.ph	<u>up</u>
// // This sketch turns t	he Arduino int	o a AVRISP		
<pre>// using the following //</pre>	arduino pins:			
// pin name: not-me	:ga: me	ga(1280 and 2	2560)	
	53			
// slave reset: 10:	51			
// slave reset: 10: // MOSI: 11:	50			
// slave reset: 10: // MOSI: 11: // MISO: 12:	52			
// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13:	01			
// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: //				
// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re	sistor) on the	following pi	ins:	
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh</pre>	sistor) on the ows the program	following pi mmer is runni	ins: ing	
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li</pre>	sistor) on the ows the progra ghts up if som	following pi mmer is runni ething goes t	ins: ing Jrong (u	use red if
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In</pre>	sistor) on the ows the program ghts up if som communication	following pi mmer is runni ething goes v with the sla	ins: ing grong (u ave	use red if
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In //</pre>	sistor) on the lows the program ghts up if some communication	following pi mmer is runni ething goes w with the sla	ins: ing grong (u ave	use red if
<pre>// slave reset: 10: // MOSI: 11: // MIS0: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In //</pre>	sistor) on the ows the program ghts up if som communication	following pi mmer is runni ething goes w with the sla	ins: ing srong (u ave	use red if
<pre>// slave reset: 10: // MOSI: 11: // MIS0: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In //</pre>	sistor) on the lows the prograu ghts up if som communication	following pi mmer is runni ething goes w with the sla	ins: ing wrong (u ave	use red if ▼
<pre>// slave reset: 10: // MOSI: 11: // MIS0: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In //</pre>	sistor) on the lows the prograu ghts up if som a communication	following pi mmer is runni ething goes w with the slo	ins: ing vrong (u ave	use red if ▼
<pre>// slave reset: 10: // MOSI: 11: // MIS0: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In // // 20 Jule 2011 Dented</pre>	sistor) on the lows the prograu ghts up if som a communication	following pi mmer is runni ething goes w with the sla	ins: ing vrong (u ave	use red if ▼
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In //</pre>	sistor) on the lows the progra ghts up if som communication	following pi mmer is runni ething goes w with the sla	ins: ing grong (u ave	use red if ▼
<pre>// slave reset: 10: // MOSI: 11: // MISO: 12: // SCK: 13: // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - In // 22 Tuto 2011 Dendel</pre>	sistor) on the cows the progra ghts up if som communication	following pi mmer is runni ething goes w with the sla	ins: ing ørong (u ave	ise red if

Paso2: Cargar al Arduino UNO como ISP, lo podemos encontrar en: Archivo>ejemplo>ArduinoISP. Luego de terminado la carga cerramos el IDE.

Paso3: Preparando el software IDE del Arduino UNO

Al instalar el IDE del Arduino este se instaló en Archivo de programa/Arduino. Se debe copiar la carpeta Arduino completamente en otro disco o raíz. Ejemplo Disco D: Arduino\_1.5\_BURN\_BOOTS



^

😋 🕞 🗢 📙 « Disce local (C:) 🕨 Archi	vos de programa 🕨 Arduino 🕨 🛛 👻 🍫	Buscar Ardumo	Q				
<u>Archivo Edición Ver Herramientas Ayu</u> da							
Organizar 👻 Incluir en biblioteca 👻	Compartir con 🔻 Grabar Nueva carpeta		l= • 🔟 🔞				
_	Nombre	Fecha de modifica	Libo				
4 📳 Equipo	\mu drivers	28/05/2014 06:04	Carpeta de archivos				
4 🏭 Disco local (C:)	📄 🚇 examples	28/05/2014-06:04	Carpeta de archivos				
> 🏨 4nec2	hardware	28/05/2014 06:04	Carpeta de archivos				
4 🍈 Archivos de programa	📄 🌗 java	28/05/2014 06:06	Carpeta de archivos				
> 👘 Adobe	🕌 lib	28/05/2014 06:07	Carpeta de archivos				
▷ iii Aniitsu	🕌 libraries	28/05/2014 06:07	Carpeta de archivos				
	🔋 👔 reterence	28/05/2014 06:07	Carpeta de archivos				
Arduno	- 🐌 tools	28/05/2014-06:07	Carpeta de archivos				
P in Bda b D. Communitation Software (or T	🥺 arduino.exe	08/01/2014 02:46	Aplicación				
Camera Assistant Software for Te	s cygiconv 2.dll	08/01/2014 02:45	Estensión de la apl				
b Common Files	🚳 cygwin1.dll	08/01/2014 02:45	Estensión de la apl				
Demonstration	🛸 libusb0 dll	08/01/2014 02:45	Extensión de la apl				
Decision Analyst, Inc.	📄 revisions.txt	08/01/2014 02:45	Documento de tex				
Digi	🚳 rətəSerial.dll	08/01/2014 02:45	Estensión de la apl				
DVD Maker	👹 uninstall.exe	28/05/2014 06:07	Aplicación				
≥ in Elaborate Buter							
The elementer							
TS efementos							
15 elementos		🌉 Equip	U				

Esta carpeta esta en D: Arduino\_1.5\_BURN\_BOOTS Hacemos doble clic en arduino.exe para comprobar que se ejecuta, luego lo cerramos.

	· · · · · · · · · · · · · · · · · · ·	
() ▼ () ► Equipo ► Disco local (D	:) • Arduino_1.5_BURN_BOOTS • • • •	Buscar Arduino_1.5_BURIV_BOOTS
<u>A</u> rchivo <u>E</u> dición <u>V</u> er <u>H</u> eriamientas	Ay <u>u</u> da	
Organizar 👻 Incluir en biblioteca 👻	Compartir con 👻 Grabar Nueva carpeta	III 👻 🗍 🔞
Equipo  Disco local (C:) Disco local (D:)  Disco local (D:) Disco local (D:)  Disco local	<ul> <li>Nombre</li> <li>drivers</li> <li>examples</li> <li>hardware</li> <li>java</li> <li>lib</li> <li>libraries</li> <li>reference</li> <li>tocls</li> <li>arduino.exe</li> <li>cygiconv-2.dll</li> <li>cygiconv-2.dll</li> <li>cycycin1.dll</li> <li>hbust0.dll</li> <li>revisions.txt</li> <li>hotxSerial.dll</li> <li>uninstall.exe</li> </ul>	Fecha de modifica         Tipo           14/08/2014 10:55 a         Carpeta de archivos           14/08/2014 10:55 a         Carpeta de archivos           14/08/2014 10:56 a         Carpeta de archivos           14/08/2014 10:59 a         Carpeta de archivos           08/01/2014 02:45         Carpeta de archivos           08/01/2014 02:45         Extensión de la apl           08/01/2014 06:07
15 elementos		🌉 Едиіро



Paso4: Configurando el archivo de configuración de hardware **avrdude.conf**, lo ubicamos dentro del disco C: Arduino/hardware/tools/avr/etc/

				×
😋 🔵 🗢 🖳 « Arduina 🕨 hardware 🕨 to	ools 🕨 avr 🖡 etc 🔍 👻 🍫	Buscar etc		P
Archivo Edición Ver Herramientas Ayr	uda			
Organizar 🔻 📄 Abrir 🛛 Grabar 💦	Nueva carpeta		E 🔹 🔲	0
^	Nombre	Fecha de modifica	Τίρο	
👰 Equipo	avrdude.conf	02/09/2011 01:40	Archivo CONE	
Siscolocal (C:)				
● 4nec/ ■				
Adobe				
Anritsu				
Apoint2K				
Arduno				
🎽 bda				
🍌 Camera Assistant Software for To				
퉲 CCleaner				
🌡 Common Files				
🍌 Decision Analyst, Inc				
📕 Digi				
DVD Maker				
💼 Ekiga				-
avertude cont. Eache de mod	66ca 02/09/2011 01/0 p.m			
Archivo CONF Ta	maño: 537 KB			
Fecha de cre	eación: 02/09/2011 01:40 p.m.			
1 elemento seleccionado		j🌉 Equipo	o	đ

Al abrir el archivo avrdude.conf observamos en la fila 9245: **Signature = 0x1e \ 0x95 \ 0x0F**; Que corresponde al ATmega 328p



C\Program Files\Arduino\hardware\tools\avr\etc\avrdude.conf - Notepad++
<u>A</u> rchivo <u>E</u> ditar <u>B</u> uscar <u>V</u> ista Co <u>d</u> ificación <u>L</u> enguaje <u>C</u> onfiguración Macro Ejecutar Plugins
Ventana <u>?</u> X
C 🖶 🗄 🖻 🕞 🕞 🙏 🖌 🗈 💼   ⊃ ⊄   # 🍇   ≪ ≪   🖫 🕞 🚍 ¶ 戻 🖉 🔊
🔚 avrdude.conf 🗵
9231 # *
9232 # ATmega328P
9233 #
9234
9235 part
9236 id = "m328p";
9237 desc = "ATMEGA328P";
9238 has_debugwire = yes;
9239 flash_instr = 0xB6, 0x01, 0x11;
9240 eeprom_instr = 0xBD, 0xF2, 0xBD, 0xE1, 0xBB, 0xCF, 0xB4, 0x00,
9241 OxBE, OxO1, OxB6, OxO1, OxBC, OxOO, OxBB, OxBF,
9242 0x99, 0xF9, 0xBB, 0xAF;
9243 stk500_devcode = 0x86;
9244 # avisio_devode = 0x;
$\begin{array}{ccc} 3245 & \text{Signature} & -0x12 & 0x33 & 0x07, \\ 0246 & \text{page} & = 0xd7; \\ \end{array}$
9247 hs2 = 0xc2:
9248 chin crase delay = $9000$ :
9249 pom enable = "1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1",
9250 "x x x x x x x x x x x x x x x ***
9251
9252 chip_erase = "1 0 1 0 1 1 0 0 1 0 0 x x x x x",
9253 "x x x x x x x x x x x x x x x ;
9254
9255 <u>timeout</u> = 200;
9256 stabdelay = 100;
9257 <u>cmdexedelay</u> = 25;
length : 550198 Ln : 9245 Col : 38 Sel : 32   0 UNIX UTF-8 w/o BOM INS

En la carpeta de D: Arduino\_1.5\_BURN\_BOOTS abrimos el archivo avrdude.conf dentro de la carpeta D: Arduino\_1.5\_BURN\_BOOTS

						×
Solution → Tools → and the second	vr ▶ etc		<b>-</b> 4 <sub>7</sub>	Buscar etc		P
<u>A</u> rchivo <u>E</u> dición <u>V</u> er <u>H</u> erramientas	Ay <u>u</u> da					
Organizar 👻 Incluir en biblioteca 👻	Compartir con 🔻	Grabar	Nueva carpeta			(?)
Emino	<ul> <li>Nombre</li> </ul>	*		Fecha de modifica	Tipo	
🚢 Disco local (C:)	avidude.con	ſ		14/08/2014 12:30	Archivo CONF	
Disco local (D:)  Disco local						
🍌 Dropbox	₩ 4					÷
1 elemento						
1 elemento				j 🜉 Equip	a	

Al abrir el archivo avrdude.conf observamos en la fila 9245: Signature =  $0x1e \ 0x95 \ 0x0F$ ; lo reemplazamos por: Signature =  $0x1e \ 0x95 \ 0x14$ ; Guardamos y cerramos



```
D:\Arduino_1.5_BURN_BOOTS\hardware\tools\avr\etc\avrdude.conf - Notepad++
<u>Archivo Editar Buscar Vista Codificación L</u>enguaje <u>C</u>onfiguración Macro Ejecutar Plugins
Ventana ?
                                                                            Х
🕞 🕒 🗄 🖻 🕞 🎧 📥 🖌 🗅 🖿 🗢 🗢 🖕 🕷 🖕 🔍 🔍 🖉 🖾 🚛 🖉 🖉
                                                                            >>
🔚 avrdude.conf 🗵
9232
      # ATmega328P
                                                                            ۰
9233
      #-----
9234
9235
      part
                      = "m328p";
9236
        id
9237
                    = "ATMEGA328P";
          desc
9238
         has debugwire  = yes;
         flash_instr = 0xB6, 0x01, 0x11;
9239
          eeprom_instr
                          = OxBD, OxF2, OxBD, OxE1, OxBB, OxCF, OxB4, OxOO,
9240
9241
                    OxBE, OxO1, OxB6, OxO1, OxBC, OxOO, OxBB, OxBF,
9242
                    Ox99, OxF9, OxBB, OxAF;
          stk500_devcode = 0x86;
9243
          # avr910_devcode = 0x;
9244
9245 signature = 0x1e 0x95 0x14;
9246
          pagel = 0xd7;
9247
          bs2
                      = 0xc2;
9248
          chip_erase_delay = 9000;
          pgm_enable = "1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1",
9249
9250
               "x x x x x x x x x x x x x x x x x";
9251
92.52
          chip_erase = "1 0 1 0 1 1 0 0 1 0 0 x x x x x",
9253
               "x x x x x x x x x x x x x x x x ";
9254
9255
          timeout = 200;
9256
          stabdelay = 100;
9257
          cmdexedelay = 25;
92.58
          synchloops = 32;
٠ ......
                              Ш
length: 550198 Ln: 9245 Col: 38 Sel: 32 | 0
                                           UNIX
                                                         UTF-8 w/o BOM
                                                                        INS
```



## **6.3 HARDWARE**

Paso1: Implementamos el siguiente diagrama a partir del Arduino UNO R3.

Importante: para este caso no se colocó los 2 condensadores de 22pf. Si tiene algún problema podría probar colocando los condensadores de 22pf.



Para el caso del ATMEGA328P tenemos la distribución de pines el cual debe coincidir con el diagrama anterior.

Pin1: Reset Pin 7 y Pin 20: VCC (+5Vol DC) Pin 8 y Pin 22: GND Pin 9 y Pin 10: Cristal Pin 17: PB5 (Pin 11 Arduino) Pin 18: PB4 (Pin 12 Arduino) Pin 19: PB5 (Pin 13 Arduino)



	ATMEGA328P-PU C	hip to	Arduino Pin Mapp	ing
Arduino function				Arduino function
reset	(PCINT14/RESET) PC6	, U 28	PC5 (ADC5/SCL/PCINT13	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2 27	PC4 (ADC4/SDA/PCINT12	2) analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3 26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4 25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5 24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6 23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7 22	GND	GND
GND	GND	8 21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9 20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10 19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11 18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12 17	PB3 (MOSI/OC2A/PCINT3	) digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13 16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14 15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)
	Digital Pins 11, 12 & 13	are used by th	e ICSP header for MISO,	
	MOSI, SCK connection	s (Atmega168	pins 17, 18 & 19). Avoid low-	
	impedance loads on the	ese nins when	using the ICSP header	

## **6.4 GRABANDO EL BOOTLOADER**

Paso1: Iniciamos el programa del ejecutable arduino.exe El archivo ejecutable está ubicado dentro de la carpeta: D: Arduino\_1.5\_BURN\_BOOTS.

🛛 🌀 💭 🗢 🖳 🕨 Equipo 🔺 Discollocal (D)	) 🕨 Ardumo_15_BURN_BOOTS 🕨 🛛 👻 🍫	Buscar Ardumo_1.5_BURN_EOOTS 👂 🔎					
<u>A</u> rchivo <u>E</u> dición <u>V</u> er ∐erramientas A	Archivo Edición Ver Herramientas Ayuda						
Organizar 🔻 💼 Abrir 🛛 Grabar	Nueva carpeta	]== 🔻 🔟 🔞					
	* Nombre	Fecha de modifica Tipo					
🖳 Equipo	🕌 drivers	14/08/2014 10:55 a Carpeta de archivos					
🏭 Disco local (C:)	📙 examples	14/08/2014 10:55 a Carpeta de archivos					
👝 Disco local (D:)	🖟 hardware	14/08/2014 10:55 a Carpeta de archivos					
\mu(00_camara_didt	📕 🕌 java	14/08/2014 10:59 a Carpeta de archivos					
google_drive	🕌 lib	14/08/2014 10:59 a Carpeta de archivos					
UCarpeta_Personal	= 👪 libraries	14/08/2014 10:59 a Carpeta de archivos					
	🔒 reference	14/08/2014 L0:59 a Carpeta de archivos					
2 2bdccaca5a7a22288b1d271b	🔒 tools	14/08/2014 L0:59 a Carpeta de archivos					
Succession as 2000 Test ID	🥺 arduino.exe	08/01/2014 02:45 Aplicación					
	🚳 cygicony-2.dll	08/01/2014 02:45 Extensión de la apl					
Archine 15 PUEN BOOTS	🖄 cygwin t.dll	08/01/2014 02:45 Extensión de la apl					
	🚳 libush0.dll	08/01/2014 02:45 Extensión de la apl					
	revisions.txt	08/01/2014 02:45 Documento de tex					
	🚳 otxSenal.dll	08/01/2014 02:45 Extension de la apl					
Driver TU-S9 Win7	🗑 uninstall.exe	28/05/2014 06:07 Aplication					
Dropbox		+					
arcluino.exe Fecha de modifica 08/01/2014 02:46 p.m. Aplicación Tamaño: 840 KB Fecha de creación: 14/08/2014 10:55 a.m.							
1 elemento seleccionado		🜉 Equipo					

Verificar la conexión con el Arduino UNO R3 y la computadora.



📀 ArdunoISP   Arduno 1.0.5	-r7		×	t los		
Archivo Editar Sketch [Herramientas] Ayuda						
ArduinoISP vers	Formato Automático Archivar el Sketch Reperar Codificación y Recargar Monitor Serial	Ctrl +T Ctrl +Mayúscu as +M		1		
// If you require	Tauista	,				
// <u>http://www.</u>	Puarto Serial			COM1		
// // This sketch tur // using the follo	Programador Crabar Secuencia de Inicio	•		COM3 COM6		
// pin name:	yma: πega(128) and	2560)		COM7		
// slave reset: IO:	53			COM10		
// MOST: 11:	.51			COM11		
// MTS0: 12:	50			COM12		
// SCX: 13:	52			COM13		
// Put an LED (with re	sistor) on the following	pins:		COM14		
// 9: Heartbeat - s:	lovs the programmer is run	ning		COM15		
77 8: Error - Lights up if scmething goes wichg (use red i)				COM16		
<pre>// 7: Programming - 1: //</pre>	communication with the s	Lave		COM17		
- / / - / / · · · · · · · · · · · · · · · · ·				COM18		
•	11			COM19		
				COM20		
				COM21		
				COM22		
			$[\mathbf{v}]$	COM34		
9		Arduino Uno on COM3	4			

Seleccionar el tipo de tarjeta: Herramientas/Tarjeta/Arduino Uno



Cambiar el tipo de programador por defecto es **AVRISP mkII**. Seleccionar el programador Arduino as **ISP de: Herramientas/Programador/** 



		L	ituio 2 ituio
💿 ArduinoISP   Arduino 1.0.5	i-r2	x	itilos
Archivo Editar Sketch Herr			
ArduinoISP // ArduinoISP vers // Copyright (c) 2	Formato Automático Ctrl+T Archivar el Sketch Reparar Codificación y Recargar Monitor Serial Ctrl+Mayúsculas+M		4 · 1 · 4 · 1 · 16 · 1 · 17
// If you require	Tarjeta 🕨		
// <u>http://www.</u> //	Puerto Serial		
// This sketch tur	Programador •		AVR ISP
// using the follo	Grabar Secuencia de Inicio		AVRISP mkII
// pin name: not-me	ga: mega(1280 and 2560)		USBtinyISP
// slave reset: 10:	53		USBasp
// MOSI: 11:	51		Parallel Programmer
// MISO: 12: // SCK: 13:	50	•	Arduino as ISP
<pre>// // Put an LED (with re // 9: Heartbeat - sh // 8: Error - Li // 7: Programming - Ir // </pre>	esistor) on the following pins: nows the programmer is running ghts up if something goes wrong (use red if n communication with the slave		
1	Arduino Uno on COM3	4	

Finalmente ejecutar la carga del Bootloader: Herramientas/Grabar Secuencia de Inicio Si se ha realizado la grabación saldrá el mensaje " Secuencia de inicio grabado"

Importante: Al grabar este nuevo chip debemos considerar que vamos a colocarlo en la tarjeta del Arduino UNO R3.



			II SID 2504 III UIO I				
	💿 ArduinoISP   Arduino 1.0.5	-r2					
E	archivo Editar Sketch Herramientas Ayuda						
	ArduinoISP	Formato Automático Archivar el Sketch Reparar Codificación y Recargar	Ctrl+T				
	// Copyright (c) 2	Wonitor Serial					
	// If you require	Tariata					
	// http://www.	l arjeta 🕨					
	//	Puerto Serial	►				
	// This sketch tur	Programador •					
		Grabar Secuencia de Inicio					
	// nin name: not-me	// min more more/1280 and 2550)					
	// slave reset: 10:	53	2000)				
	// MOST: 11:	51					
	// MTSO: 12:	50					
	// SCK: 13:	52					
	//						
	// Put an LED (with re	sistor) on the following r	)ins:				
	// 9: Heartbeat - sh	ows the programmer is runn	ling				
	// 8: Error - Li	ghts up if something goes	wrong (use red if				
	// 7: Programming - In	communication with the sl	ave				
	11						
	// 00 Telle 0011 Dec 4-1	I D-1	· · ·				
	•		, , , , , , , , , , , , , , , , , , ,				
	4						
	1 Arduino Uno on CUM34						