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

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

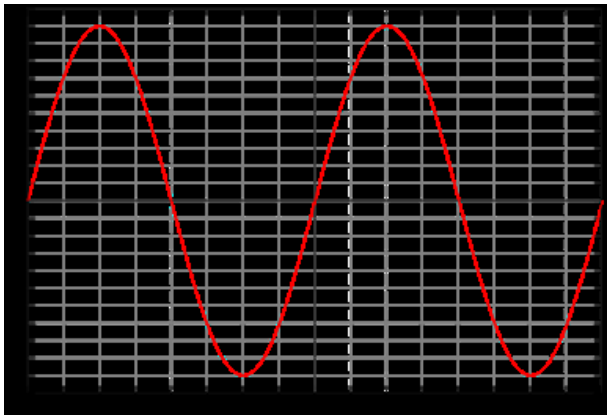
Una magnitud física que impulsa a los electrones a lo largo de un conductor (por ejemplo un cable) en un circuito eléctrico cerrado, provocando el flujo de una corriente eléctrica. Su unidad es el Voltio (V). El instrumento usado para medir el voltaje se conoce como voltímetro.

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.

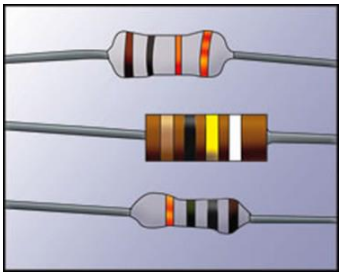


1.3 CORRIENTE

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 (Ω). Puedes encontrar resistencias en los calefactores eléctricos, tarjetas electrónicas, estufas son muy útiles para limitar el paso de la corriente.



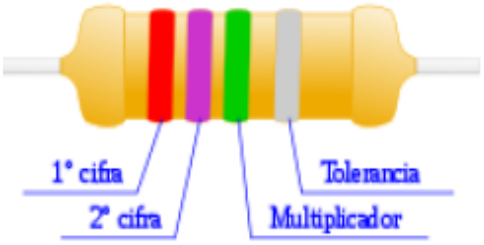
Símbolo

Componente

1.4.1 CÓDIGO DE COLORES DE LOS RESISTORES.

Color de la banda	Valor de la 1°cifra significativa	Valor de la 2°cifra 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	10000000	±0,1%
Gris	8	8	100000000	±0.05%
Blanco	9	9	1000000000	-
Dorado	-	-	0,1	±5%
Plateado	-	-	0,01	±10%
Ninguno	-	-	-	±20%

Tabla de código de colores

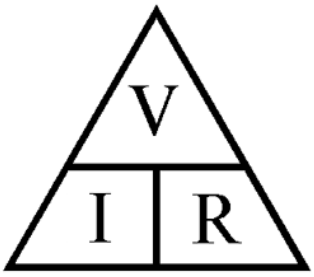


La caracterización de una resistencia de 2.700.000 Ω (2,7 MΩ), con una tolerancia de ±10%, sería la representada en la figura :

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

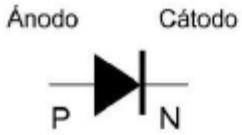
1.5 LEY DE OHM

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.

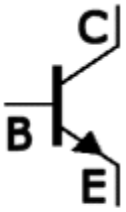


Símbolo

Componente

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.

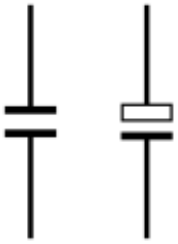


Símbolo

Componente

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.



Símbolo

Componente

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.

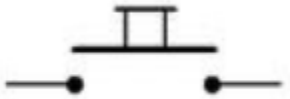
El LED tiene una polaridad, un orden de conexión, y al conectarlo al revés se puede quemar, revisa los dibujos de la parte superior para conocer a que corresponde el positivo y el negativo.

Símbolo

Componente

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 fun-ción inversa de la que en ese momento este realizando.



Símbolo

Componente

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

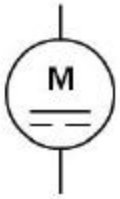
El zumbador, buzzer en inglés, es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono. Sirve como mecanismo de señalización o aviso, y son utilizados en múltiples sistemas como en automóviles o en electrodomésticos. Inicialmente este dispositivo estaba 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.

Símbolo

Componente

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



Símbolo

Componente

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

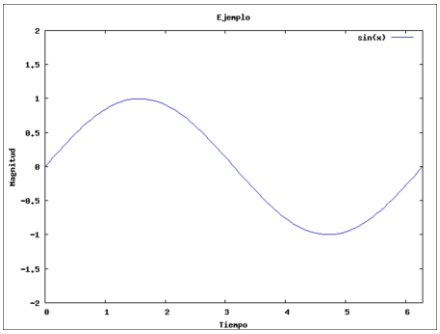
También llamadas variables discretas. Se caracterizan por tener dos estados diferenciados y por lo tanto se pueden llamar binarias. Siendo estas variables más fáciles de tratar (en lógica serían los valores Verdadero (V) y Falso (F) o podrían ser 1 ó 0 respectivamente).
Un ejemplo de una señal digital es el interruptor del timbre de tu casa, porque este interruptor tiene dos estados pulsado y sin pulsar.

Encendido
1
High

Apagado
0
Low

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

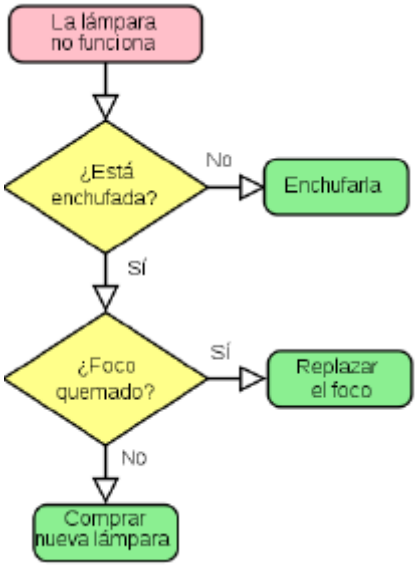
lenguaje específico, del más bajo nivel conocido como código máquina, siendo éste excesivamente complicado para programar. De hecho sólo consiste en cadenas de números 1 y 0 (Sistema binario).
Para facilitar el trabajo de programación, los primeros científicos que trabajaban en el área decidieron reemplazar 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

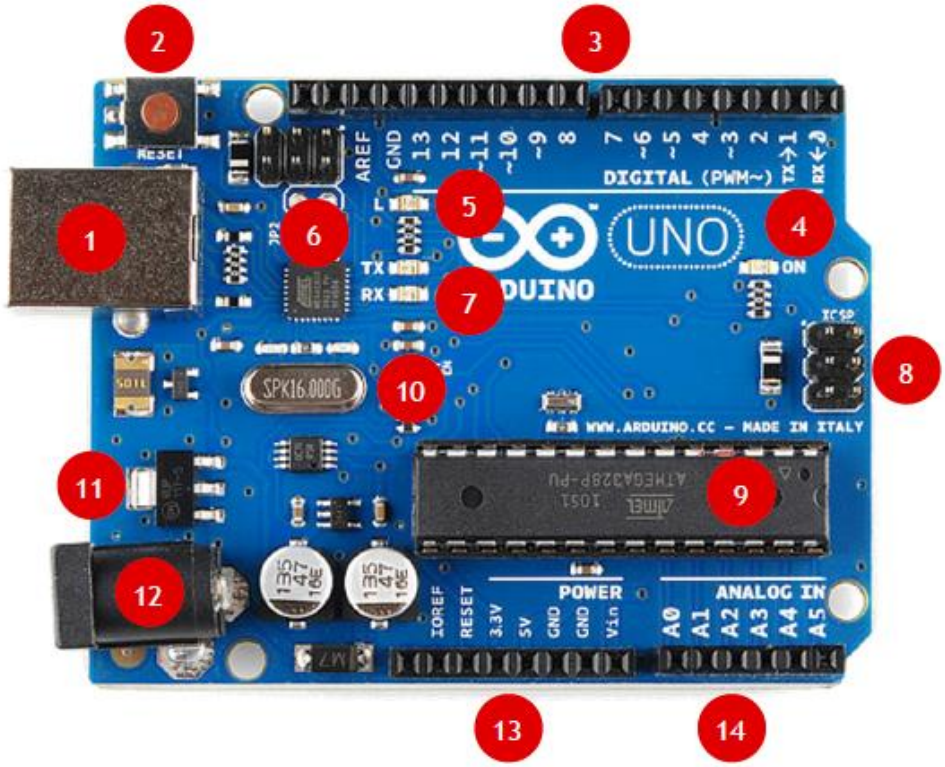
Un algoritmo es un conjunto pre-escrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a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

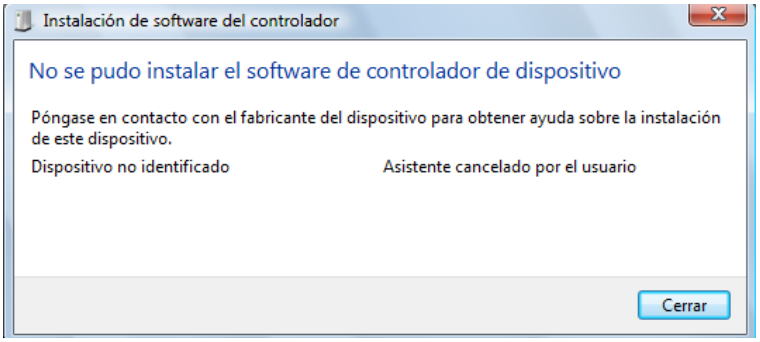
1. Conector USB para el cable tipo AB
2. Pulsador de reset
3. Pines de entrada y salida digitales y PWM
4. LED verde de placa encendida
5. LED naranja conectado al pin 13
6. ATMEGA 16U2 encargado de la comunicación con el PC
7. LED TX (Transmisor) y RX (Receptor) de la comunicación serial
8. Puerto ICSP 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

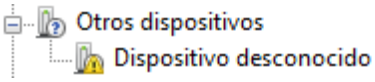
3.2.1 Windows 7, XP o VISTA

- Descargar el software de **arduino.cc/en/Main/Software** para Windows

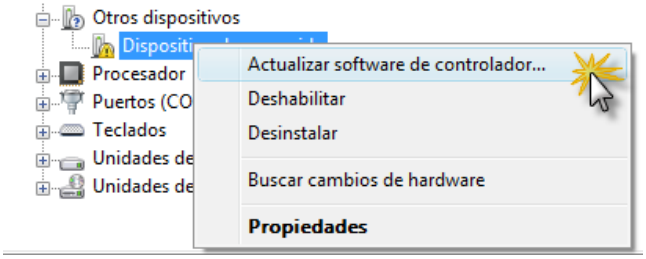
- Descomprimir la carpeta de Arduino en una ubicación de fácil acceso
- Conectar la placa Arduino Uno al PC y ver este aviso. No nos debemos pre-ocupar



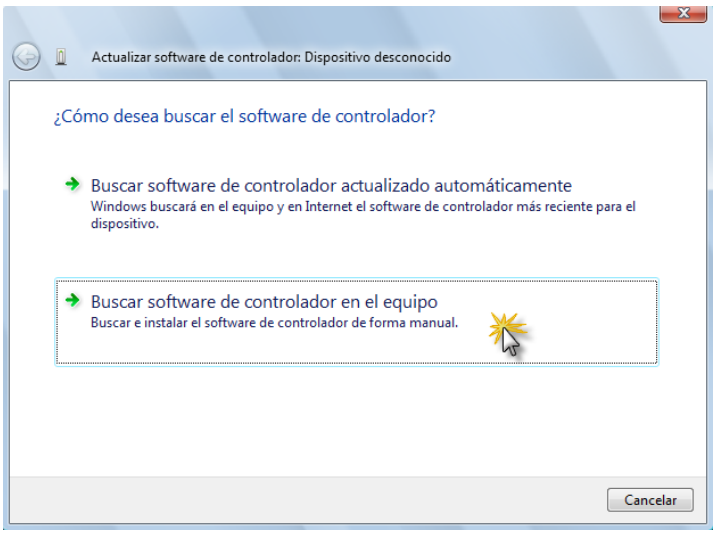
- Visitar **Panel de control** y luego **Administrador de dispositivos**, allí buscar la siguiente opción.



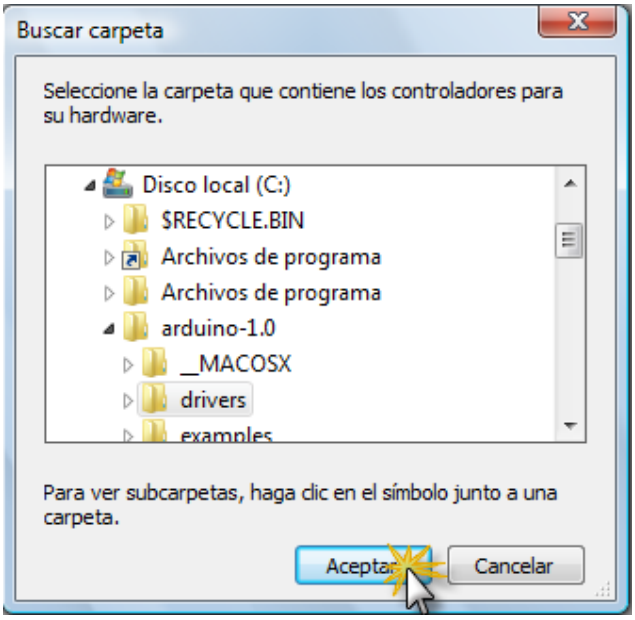
- Click derecho sobre **Dispositivo desconocido** y luego sobre la opción **Actualizar software del controlador.**



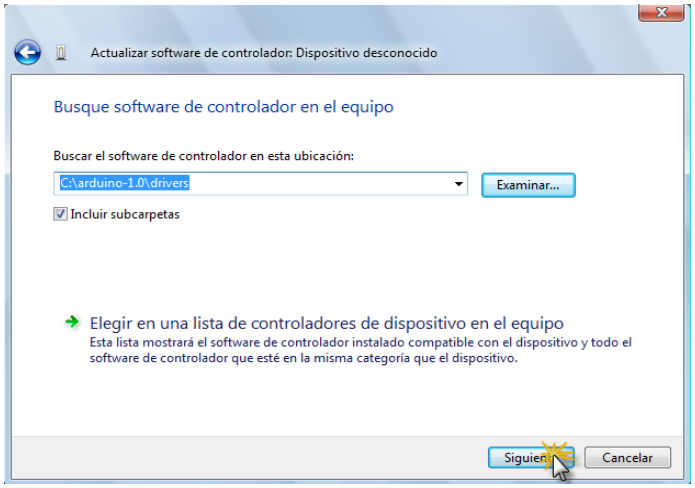
- Ingresar a la opción Buscar **software de controlador en el equipo**

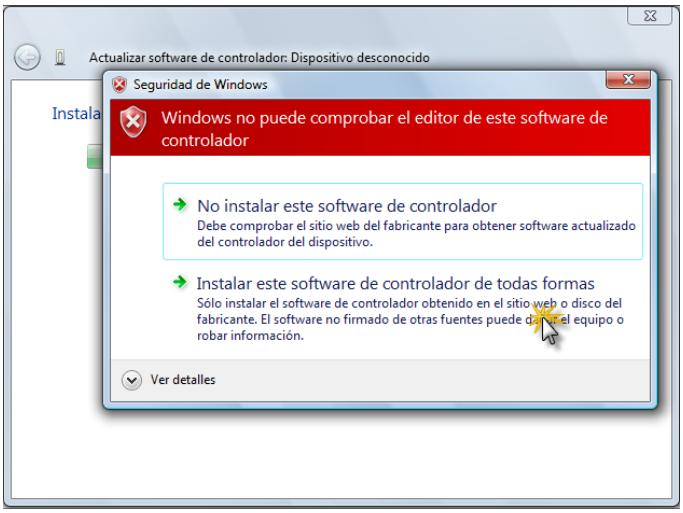


- Examinar y buscar la **carpeta de Arduino** previamente descomprimida en el paso 2. Dentro de esa carpeta acceder a la carpeta Drivers y dar **Aceptar**

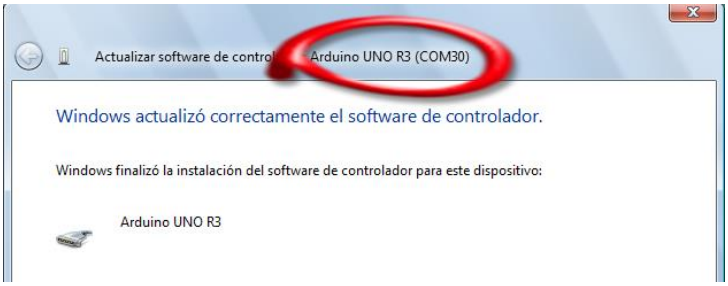


- Una vez buscamos la carpeta de **Drivers** le damos **Siguiente**



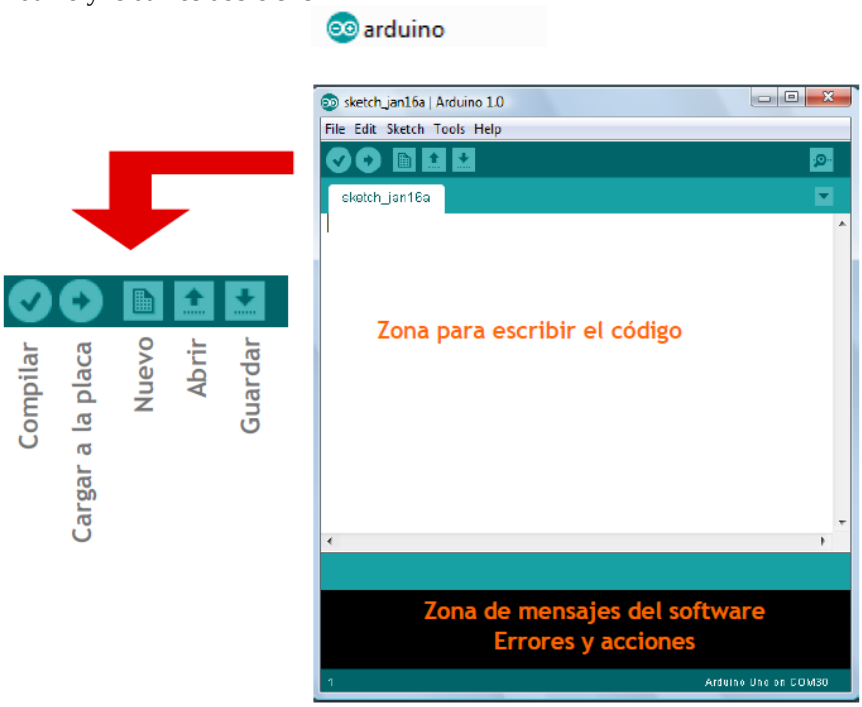


- Recibimos la confirmación del **Puerto COM** asignado, este número de Puerto COM es muy importante tenerlo en cuenta a la hora de programar

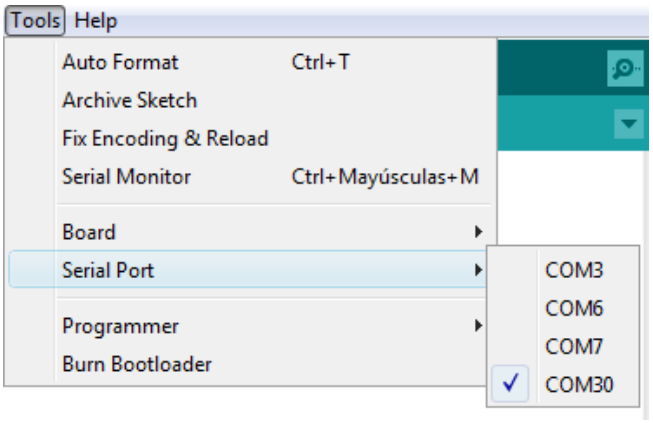


3.2 CONOCIENDO EL SOFTWARE ARDUINO

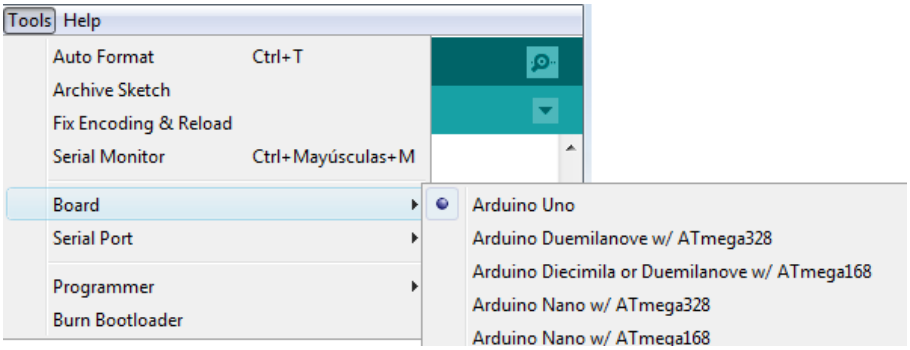
- Para ejecutar el programa Arduino, ingresamos a la carpeta de Arduino y allí buscamos el icono ce Arduino y le damos doble click



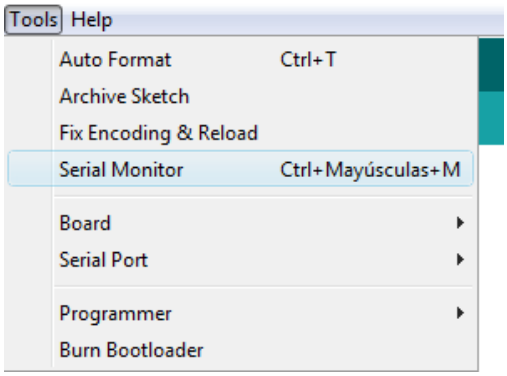
1.- Puerto COM



2.- Seleccionar la placa a trabajar



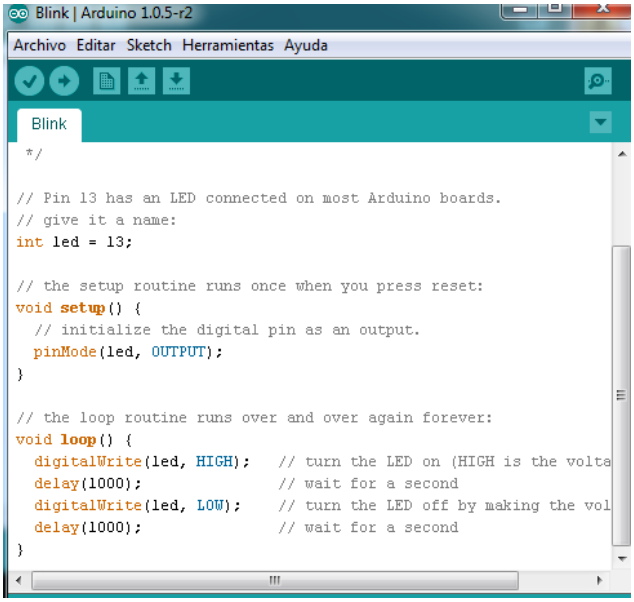
3.- Consola serial



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)
- Funciones



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)
- ! (negación)

If (a || b)

3.3.4 OPERADORES DE COMPARACIÓN

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

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

```
If (a == b)
```

3.3.5 OPERADORES MATEMÁTICOS

Se aplican al manejo de variables, condicionales y ciclos

- = (asignar)
- % (módulo)
- + (suma)
- - (resta)
- * (multiplicación)
- / (división)

```
int valor = valor +5
```

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: If, Switch/case, for y while

3.3.7 CONDICIONALES

Ideales para tomar decisiones luego de evaluar las condiciones lógicas:

If (Si)

```
if (entrada < 500)
{
      // acción A
} else
{
      // acción B
}
```

Switch/case (Casos)

```
switch (var) {
  case 1:
    // acción A
    break;
  case 2:
    // acción B
    break;
  default:
    // acción C
}
```

3.3.8 CICLOS

Ideales para repetir lo que se encuentre dentro de ellos

For (por)

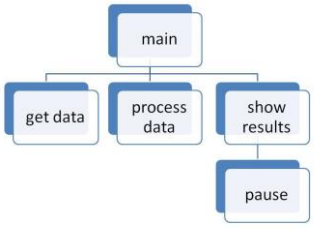
```
for( int a=0; a>10; a++ )
{
      // acción a repetir
}
```

While (mientras)

```
while ( var < 200) {
  // acción a repetir
  var++;
}
```

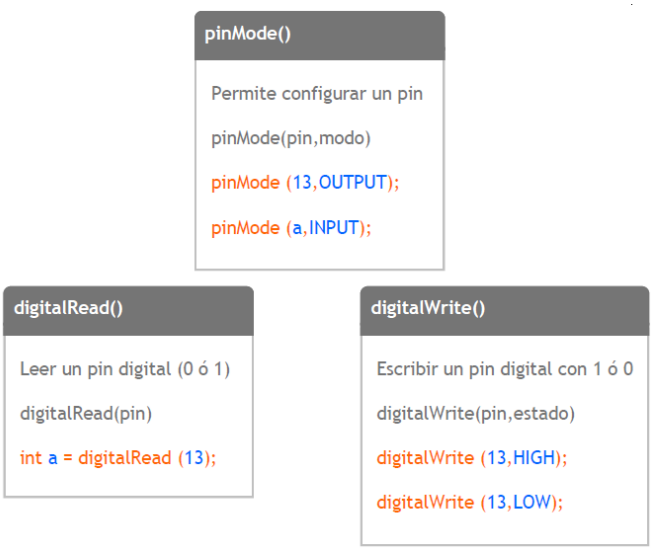
3.3.9 FUNCIONES

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

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()

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()

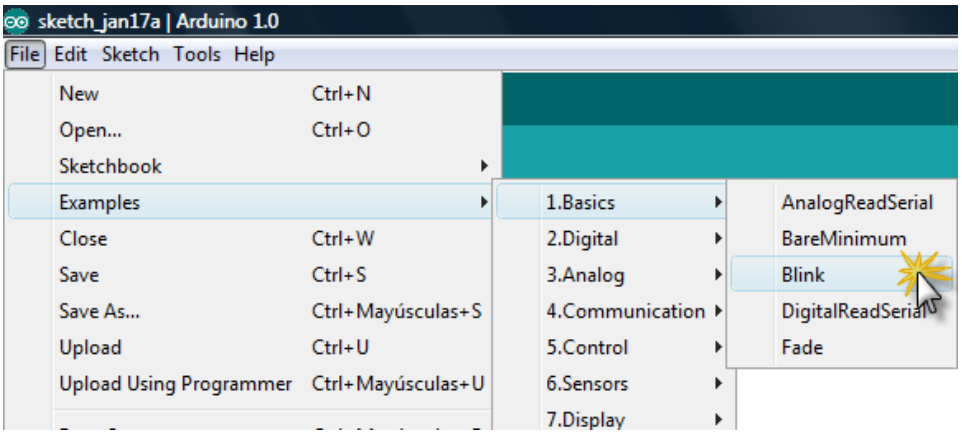
+Interrupciones externas
- attachInterrupt()
- detachInterrupt()

+Interrupciones
- interrupts()
- noInterrupts()

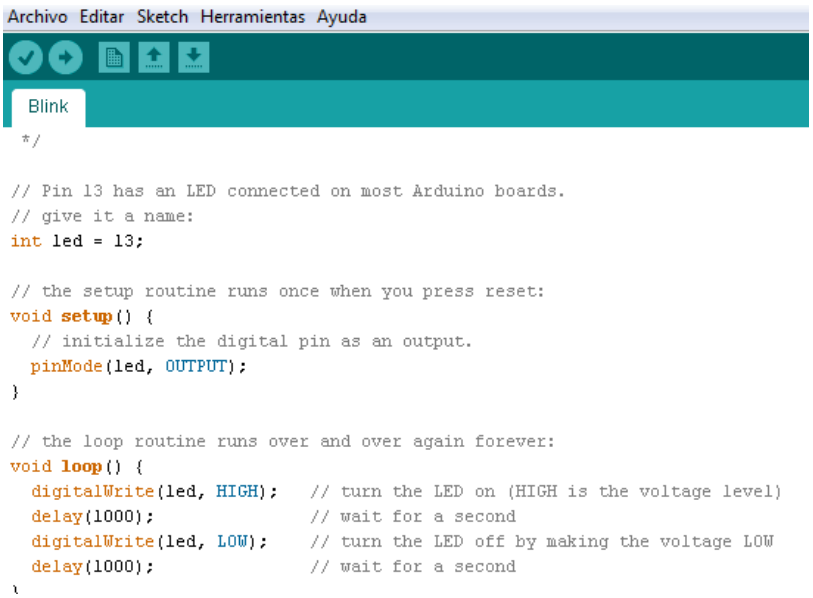
+Comunicación
- Serial
    - begin()
    - end()
    - available()
    - read()

4.- CARGANDO MI PRIMER PROGRAMA

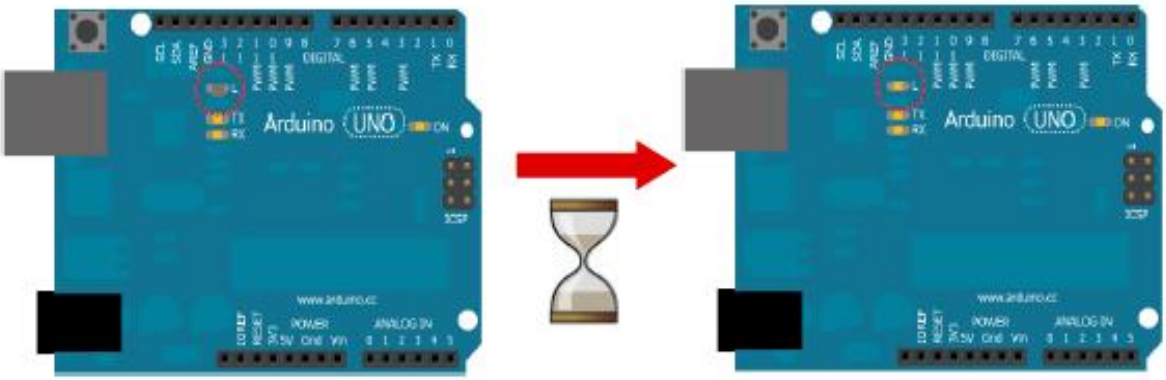
1.- Para ello abre el software de Arduino y realiza lo que se indica en la siguiente imagen:



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.



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

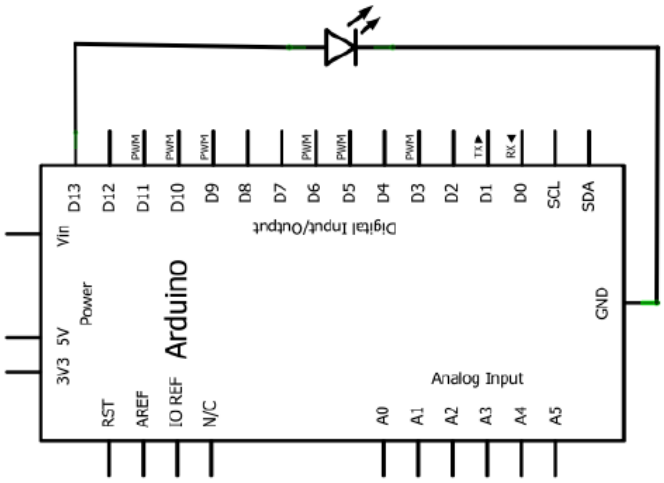


5.- TUTORIALES

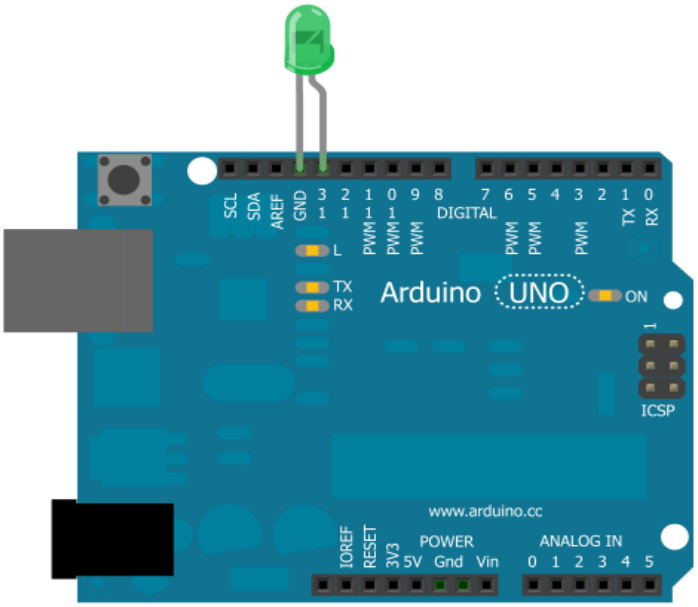
5.1LED INTERMITENTE

- 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



5.1.3 CÓDIGO DE PROGRAMACIÓN

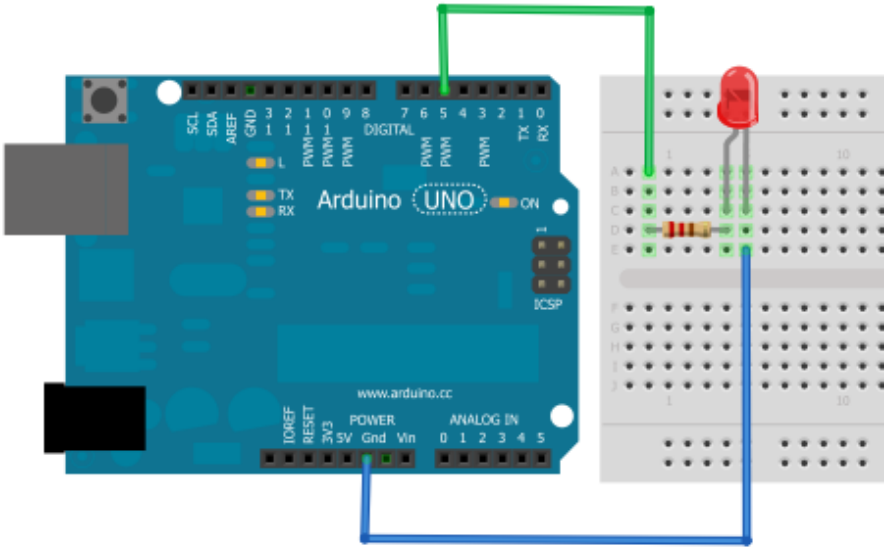
```
/*
---------------------------------
Hola Mundo
---------------------------------
Enciende un LED por un segundo y lo apaga por el mismo tiempo
*/
//---------------------------------
//Función principal
//---------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
pinMode(13,OUTPUT); // Inicializa el pin 13 como una salida
}
//---------------------------------
//Función cíclica
//---------------------------------
void loop() // Esta función se mantiene ejecutando
{ // cuando este energizado el Arduino
digitalWrite(13,HIGH); // Enciende el LED
delay(1000); // Temporiza un segundo (1s = 1000ms)
digitalWrite(13,LOW); // Apaga el LED
delay(1000); // Temporiza un segundo (1s = 1000ms)
}
// 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

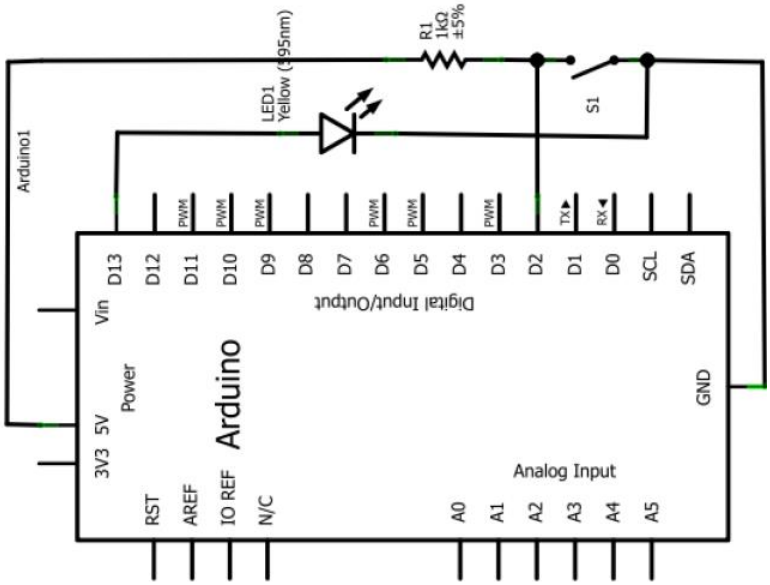


5.2 ENCENDER UN LED CON UN PULSADOR

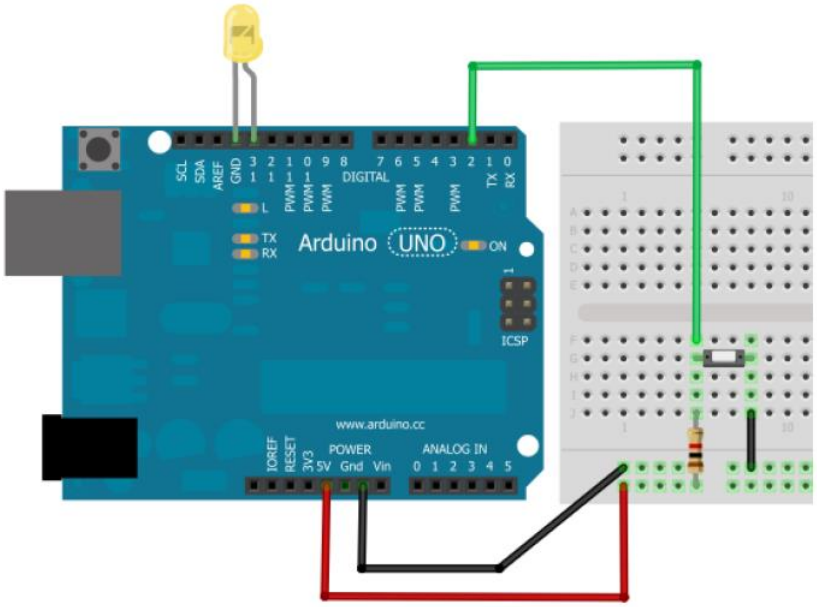
- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalWrite() y digitalRead()
  - ✓ Divisor de voltaje
  - ✓ Condicional y operadores de comparación

- ¿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



5.2.2 Esquema



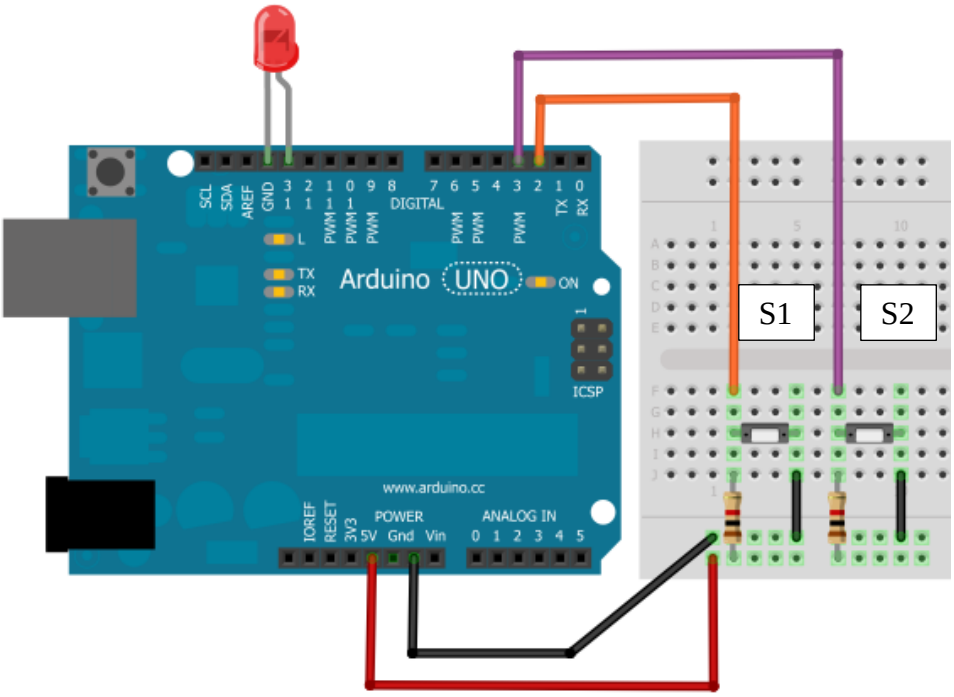
5.2.3 Código de programación

```
/*
---------------------------------
Encender LED con un pulsador
---------------------------------
Oprimir un pulsador y mientras este se mantenga accionado
un LED se enciende
*/
//---------------------------------
//Declara puertos de entradas y salidas
//---------------------------------
int pulsador=2; //Pin donde se encuentra el pulsador, entrada
int led=13; //Pin donde se encuentra el LED, salida
//---------------------------------
//Funcion principal
//---------------------------------
```

```
21  void setup() // Se ejecuta cada vez que el Arduino se inicia
22  {
23  pinMode(pulsador, INPUT); //Configurar el pulsador como una entrada
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  //Condicional para saber estado del pulsador
32  if (digitalRead(pulsador)==HIGH)
33  {
34  //Pulsador oprimido
35  digitalWrite(led,HIGH); //Enciende el LED
36  }
37  else
38  {
39  //Pulsador NO oprimido
40  digitalWrite(led,LOW); //Apaga el LED
41  }
42  }
43  //Fin programa
```

5.2.3 Ejercicios:

- 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.



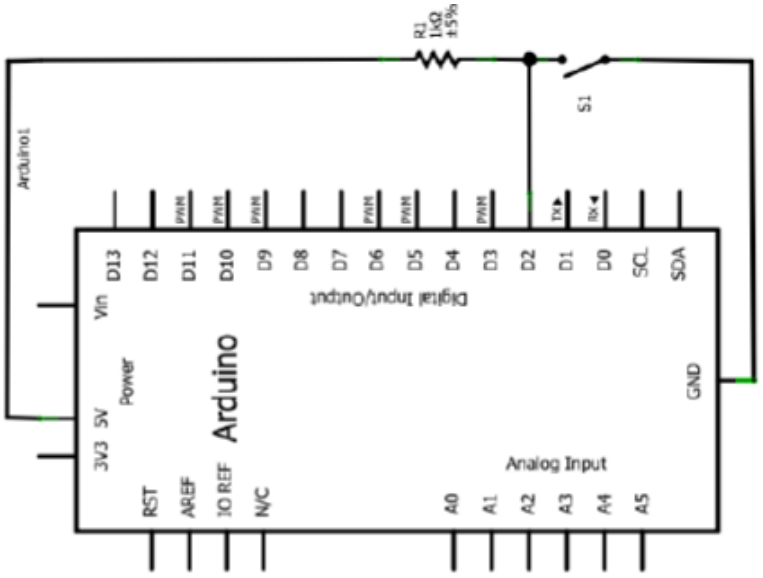
- Tu padre quiere que realices un sistema de iluminación LED para las escaleras de la casa. La condición es que si estás arriba y pulsas a S1 o si estás abajo y pulsas S2 el LED Rojo se enciende y al dejar de pulsar se apaga. Como guía de montaje toma la imagen anterior.

5.3 LECTURA SERIAL DE UNA ENTRADA DIGITAL

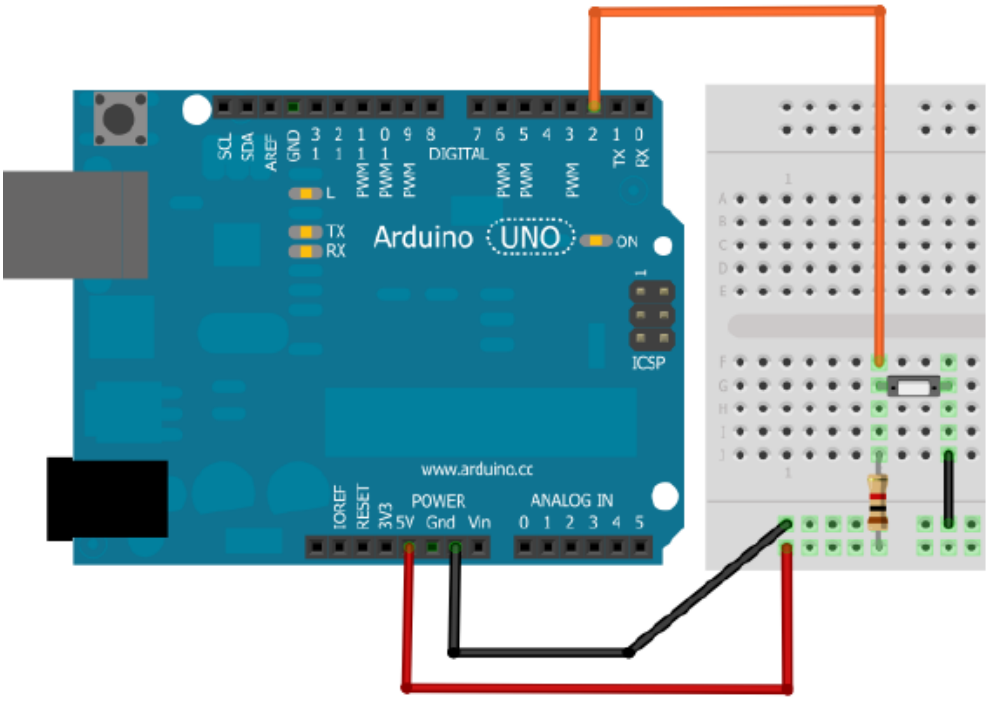
- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalRead() y Serial.println()
  - ✓ Opción de Consola serial,

- ¿Qué aprendo?
  - ✓ Manejar una entrada digital
  - ✓ Ver datos por la pantalla del computador
  - ✓ 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
```

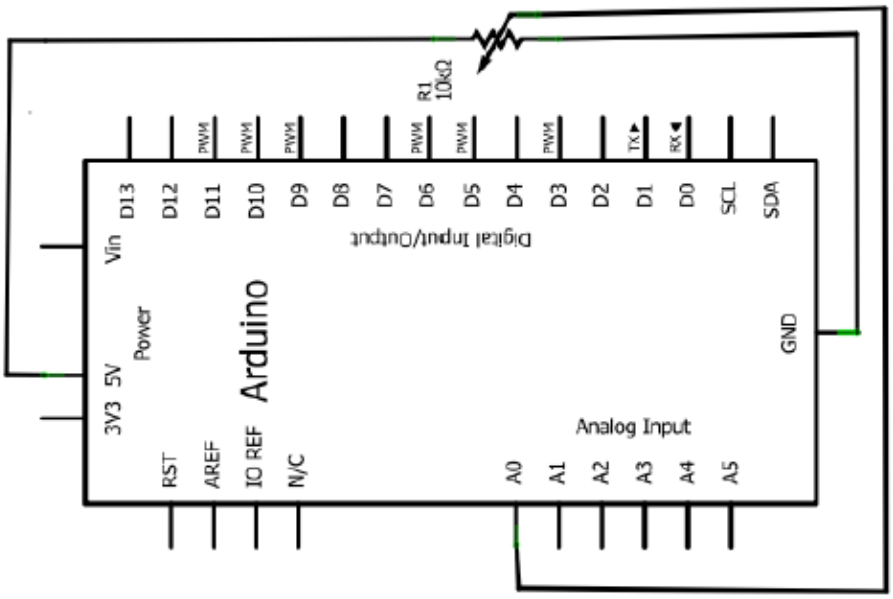
5.3.3 Ejercicios

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 Mundo‖ 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.

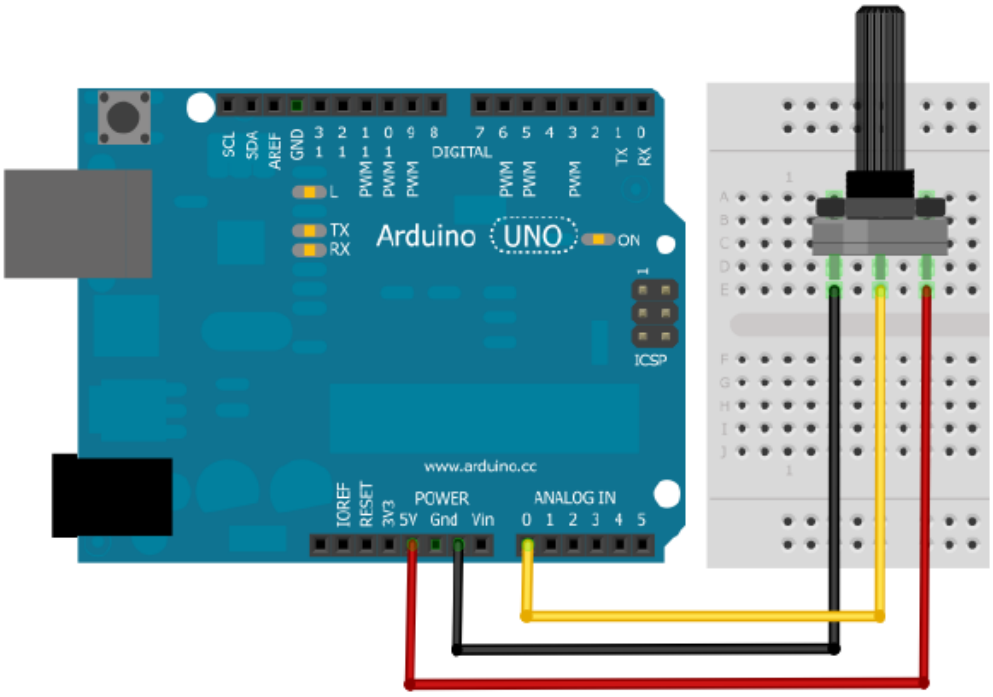
5.4 LECTURA SERIAL DE UNA ENTRADA ANÁLOGA

- 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



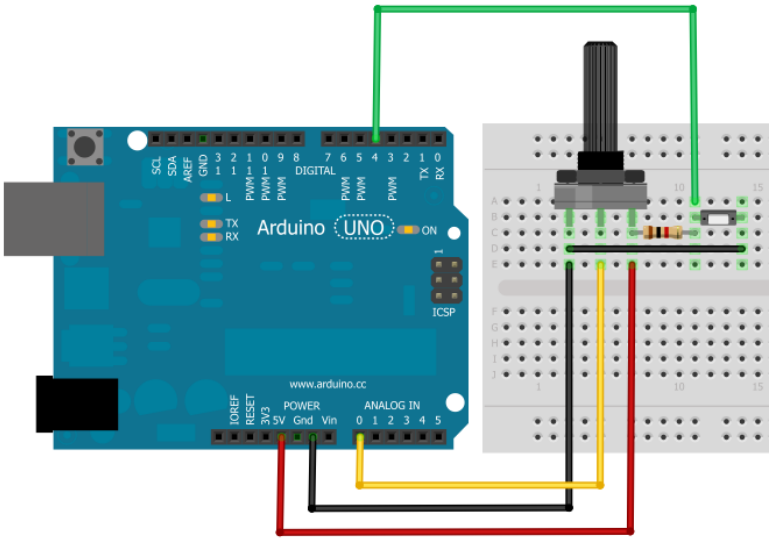
5.4.3 Código de programación

```
/*
---------------------------------------
Lectura serial de entrada análoga
---------------------------------------
Leer una entrada análoga y mostrar por la pantalla del
computador (consola serial) el valor luego de girar
el potenciómetro
*/
//-----------------------------------
//Funcion principal
//-----------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
Serial.begin(9600); //Inicia comunicación serial
}
//-----------------------------------
//Función ciclicla
//-----------------------------------
void loop() // Esta funcion se mantiene ejecutando
{ // cuando este energizado el Arduino
//Guardar en una variable entera el valor del potenciómetro 0 a 1024
int valor= analogRead(A0);
//Imprime en la consola serial el valor de la variable
Serial.println(valor);
//Retardo para la visualización de datos en la consola
delay(100);
}
//Fin programa
```

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".

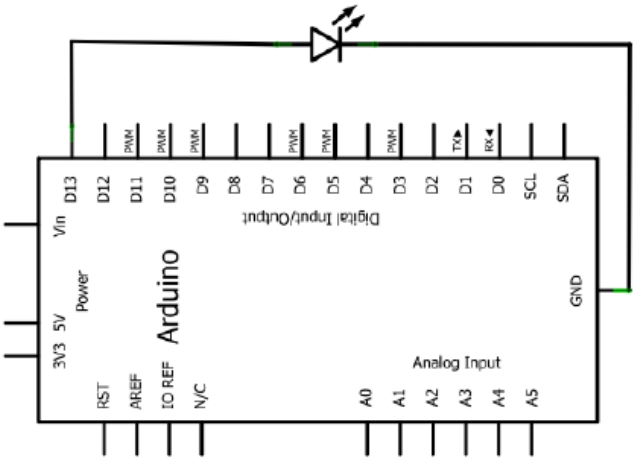
Este caso podría ser un ejemplo para que lo apliques a una Caja de seguridad, para que guardes mucho dinero.



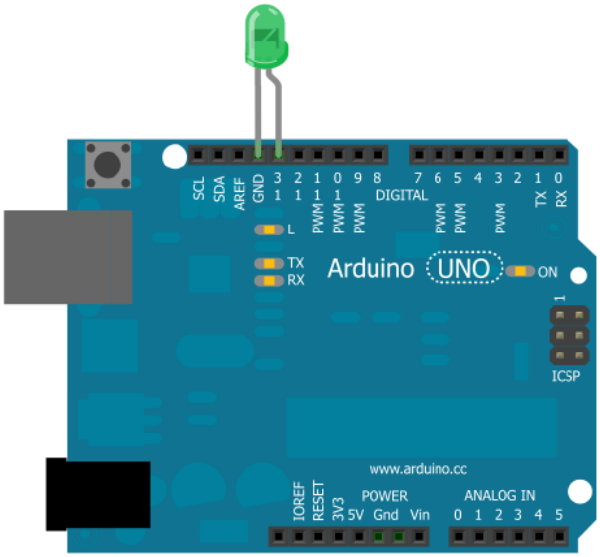
5.5 ESCRITURA SERIAL

- 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
  - ✓ Escritura serial digital

5.5.1 Plano



5.3.2 Esquema



5.3.3 Código de programación

```
/*
---------------------------------------
Escritura serial
---------------------------------------
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.
*/
//-----------------------------------------------------
//Declara puertos de entradas y salidas y variables
//-----------------------------------------------------
int led = 13; //Pin donde se encuentra el LED, salida
char leer; //Variable donde se almacena la letra
boolean prendido=false; //Estado LED la primera vez, apagado
//-------------------------------------
//Funcion principal
//-------------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
Serial.begin(9600); //Inicia comunicación serial
pinMode(led, OUTPUT); //Configurar el LED como una salida
}
//-------------------------------------
//Funcion ciclicla
//-------------------------------------
void loop() // Esta funcion se mantiene ejecutando
{ // cuando este energizado el Arduino
//Guardar en una variable el valor de la consola serial
leer=Serial.read();
// Si es la letra 'a' y además el LED está apagado
if ( (leer=='a') && (prendido==false) )
{
digitalWrite(led,HIGH); // Enciende el LED
prendido=true; // Actualiza el estado del LED
}
// Si es la letra 'a' y además el LED está encendido
else if ( (leer=='a') && (prendido==true) )
{
digitalWrite(led,LOW); // Apaga el LED
```

```
43    prendido=false; // Actualiza el estado del LED
44    }
45    }
46    //Fin programa
```

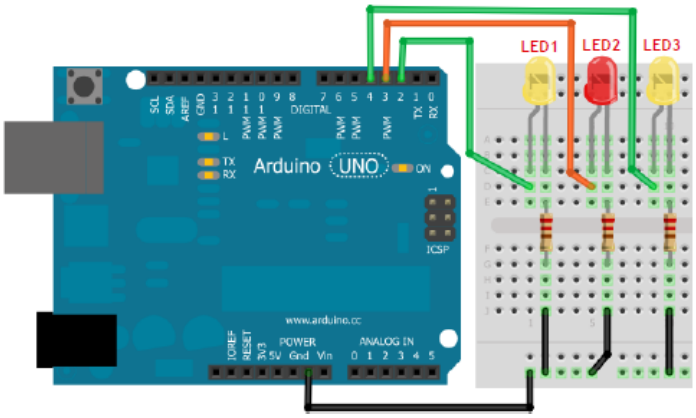
5.3.4 Ejercicios

Avisos luminosos S.A. te ha contratado para que 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

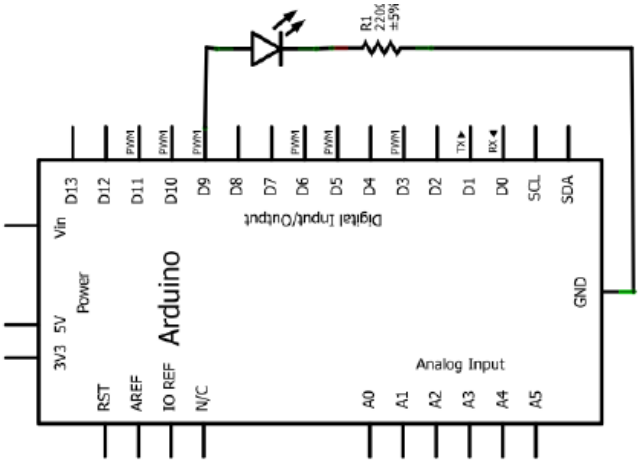
-Letra C: Apaga los tres LED



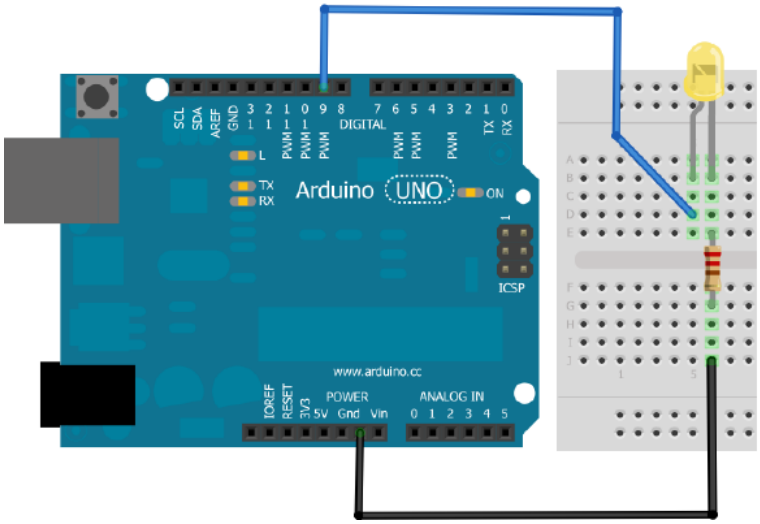
5.6 ENCENDER UN LED POR PWM

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

5.6.1 Plano



5.6.2 Esquema



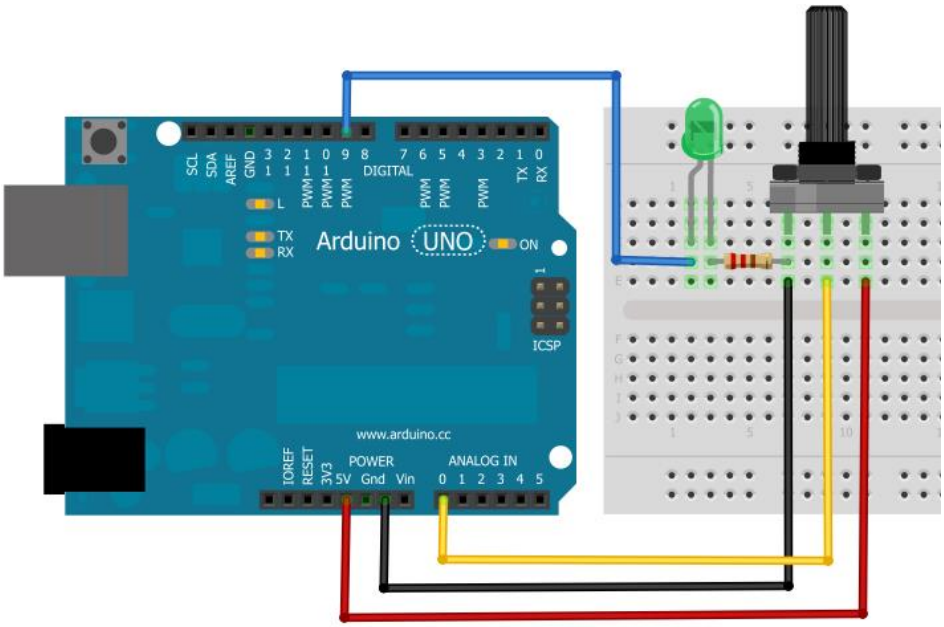
5.6.3código de programación

```
/*
-----------------------------------------
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

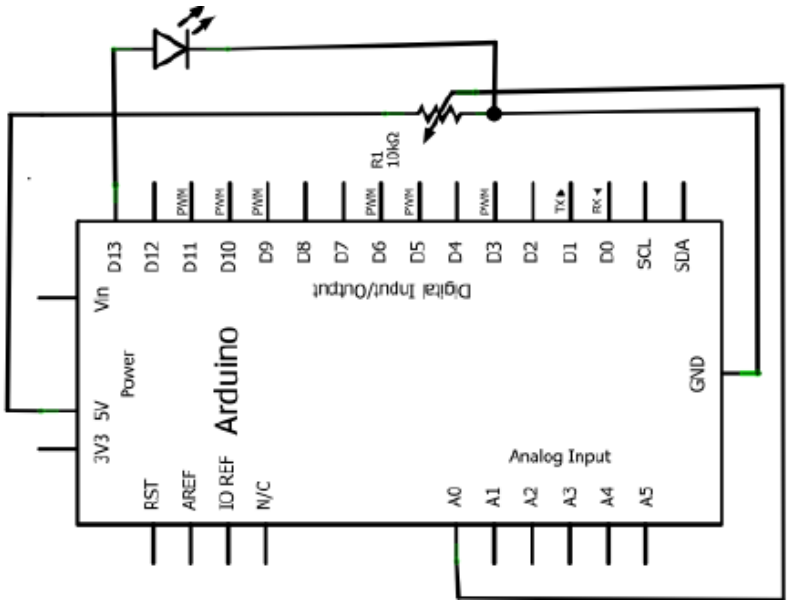
Tu mamá quiere tener unas luces de intensidad variables en la sala de tu casa, quiere poca luz para los momentos en que la familia comparte para mantener la armonía pero requiere bastante luz cuando realiza sus tejidos, ella quiere tener el control de la intensidad de luz mediante un potenciómetro y 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()



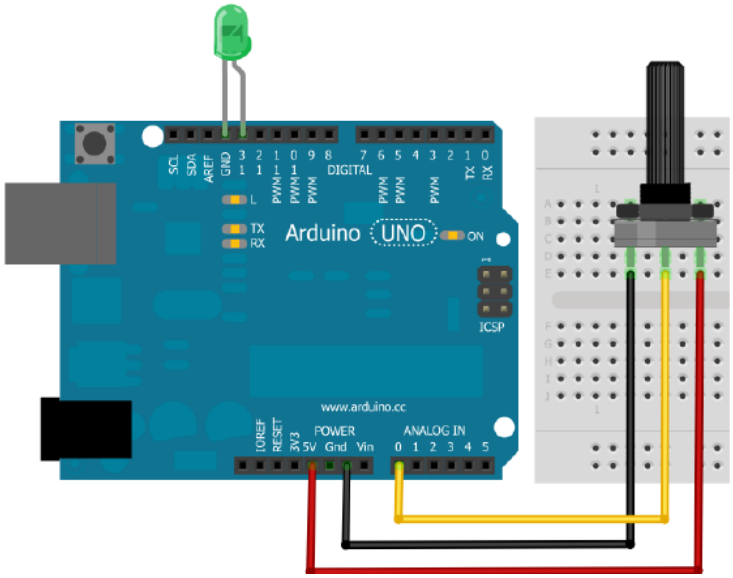
5.7 CONTROL ON/OFF POR POTENCIÓMETRO

- 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

5.7.1 Plano



5.7.2 Esquema



5.7.3 Código de programación

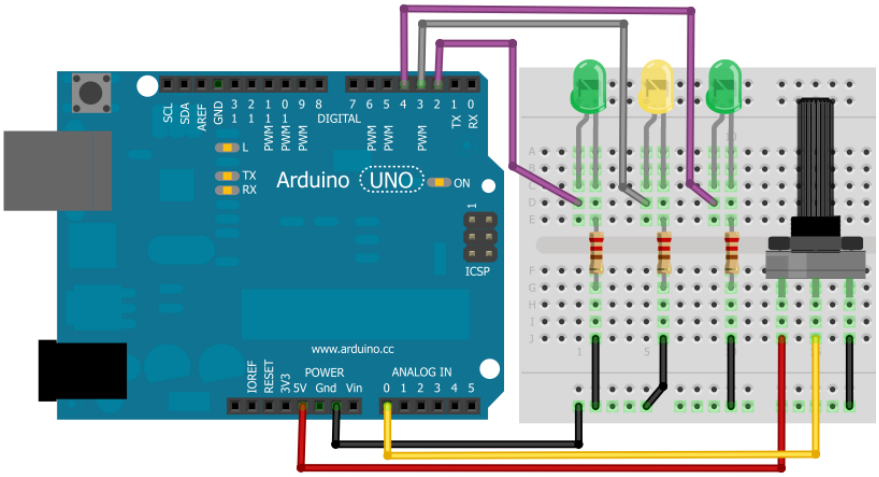
```
/*
----------------------------------------------
Control ON/OFF con potenciómetro
----------------------------------------------
Programa que enciende un LED cuando el valor de la entrada  análoga comandada por el potenciómetro esta en cierto valor,
cuando este valor cambia el LED se apaga, es un sistema con
control ON/OFF
*/
//-------------------------------------
//Función principal
//-------------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
Serial.begin(9600); //Inicia comunicación serial
pinMode(13,OUTPUT); //Configurar el pin 13 como una salida
}
```

```
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  {
33  digitalWrite(13,HIGH); //Enciende el LED en el pin 13
34  }
35  //Si el valor es menor a 500
36  else
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

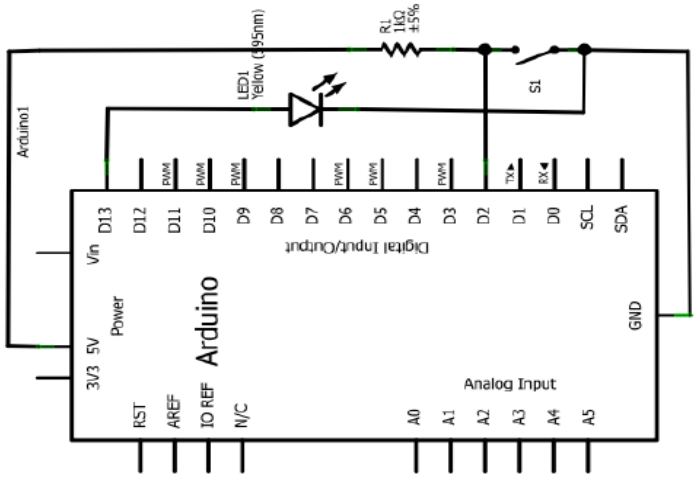


5.8 CONTADOR DE PULSOS

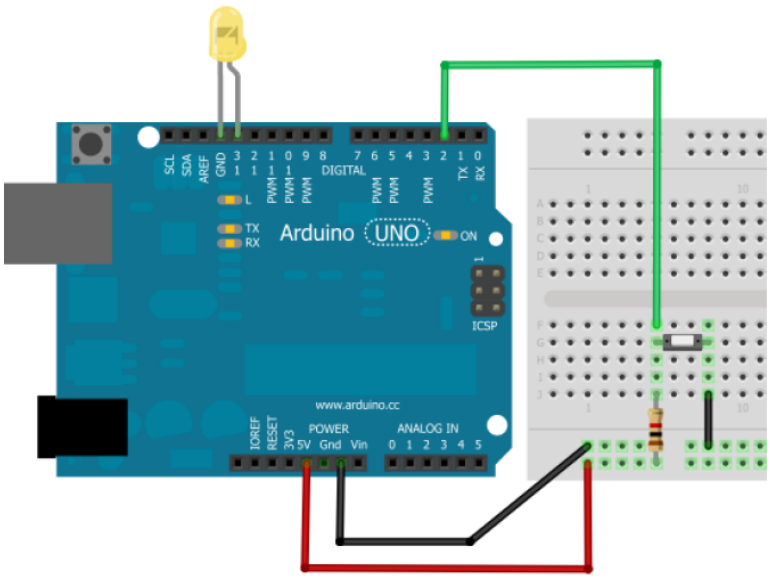
- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalWrite() y digitalRead()
  - ✓ Imprimir datos por consola
  - ✓ Declarar variables enteras
- ¿Qué aprendo?
  - ✓ Incrementar una variables
  - ✓ Condicional If/else anidado

- ✓ Anti-rebote de un pulsador
- ✓ Leer una entrada digital y escribir una salida digital a determinada condición

5.8.1 Plano



5.8.2 Esquema



5.8.3 Código de programación

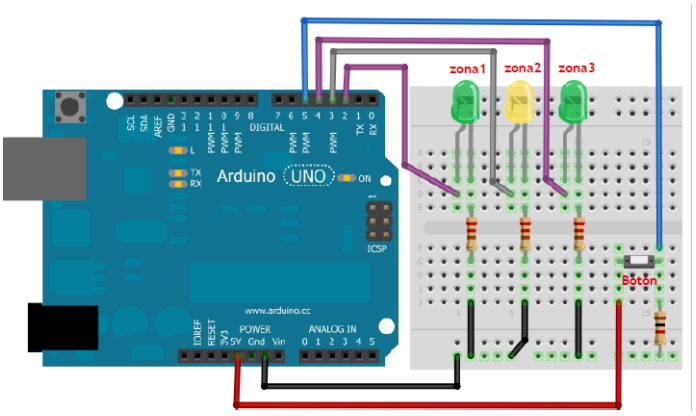
```
/*
-----------------------------------------------
Contador de pulsos
-----------------------------------------------
Programa que muestra por pantalla (consola serial) el número  de veces que el pulsador ha sido
presionado, se realiza un  proceso que de acuerdo al número de pulsaciones se enciende  un LED,
*/
//-------------------------------------------------
//Declara puertos de entradas y salidas y variables
//-------------------------------------------------

int conta = 0; //Variable para guardar el conteo de los pulsos
//-----------------------------------
//Funcion principal
//-----------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
Serial.begin(9600); //Inicia comunicación serial
```

```
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**

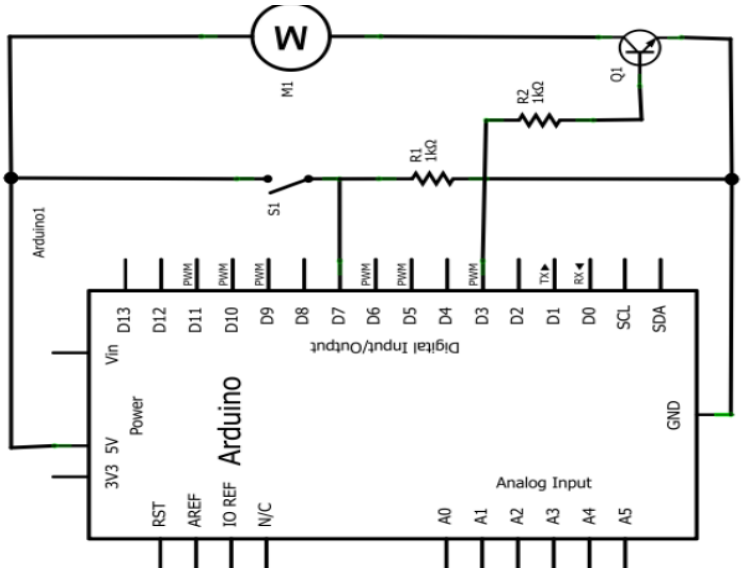


5.9 CONTROL ON/OFF DE UN MOTOR DC

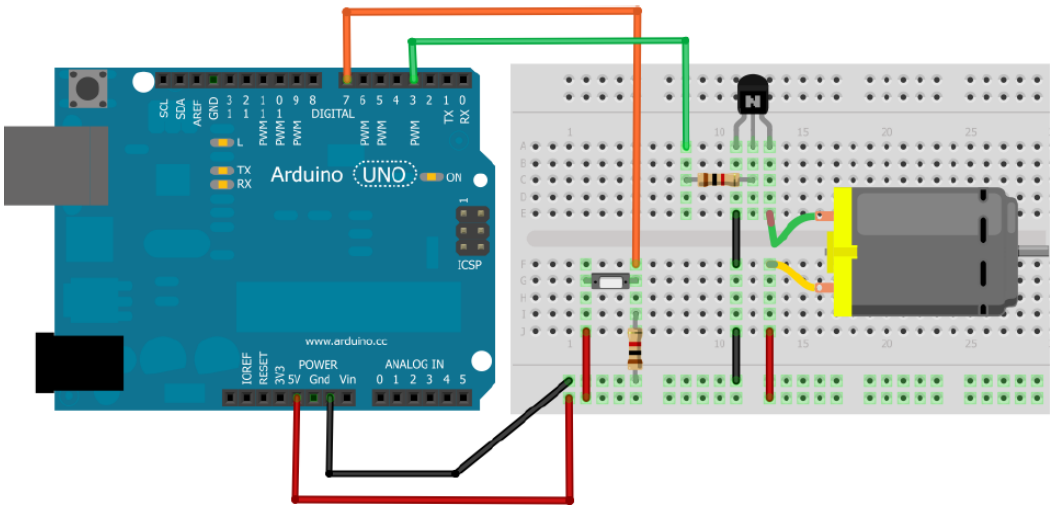
- Conocimientos previos
  - ✓ Señal digital
  - ✓ Función digitalWrite() y digitalRead()
  - ✓ Divisor de voltaje
  - ✓ Reconocer un transistor y un motor
- ¿Qué aprendo?

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

5.9.1 Plano



5.9.2 Esquema



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)*

/

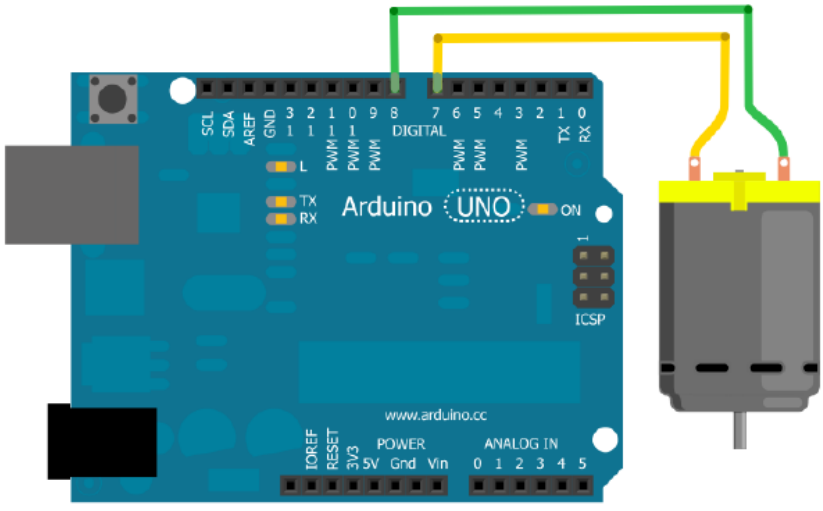
```
//-----------------------------------------------
//Declara puertos de entradas y salidas y variables
//-----------------------------------------------
int pulsador =7; //Declara Pin del pusaldor
int motor=3; //Declara Pin del motor
//-------------------------------------
//Funcion principal
//-------------------------------------
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

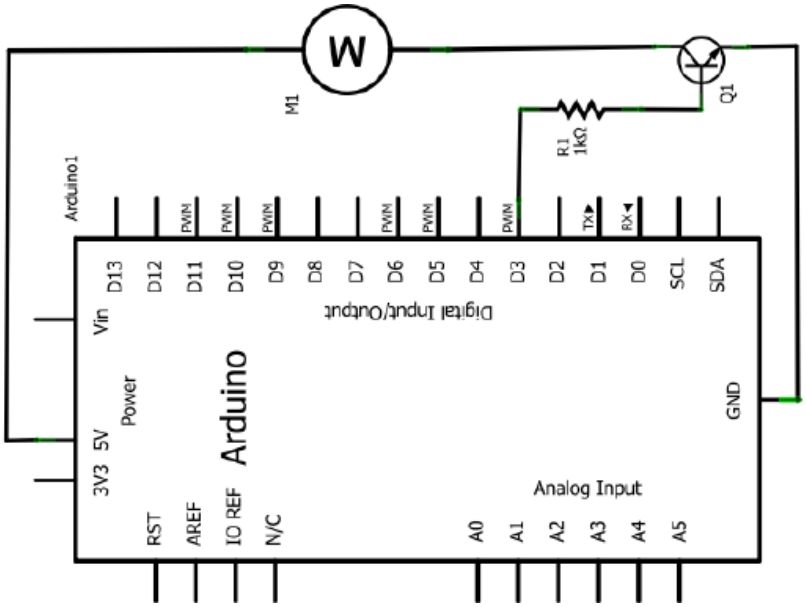


5.10 CONTROL POR PWM DE UN MOTOR

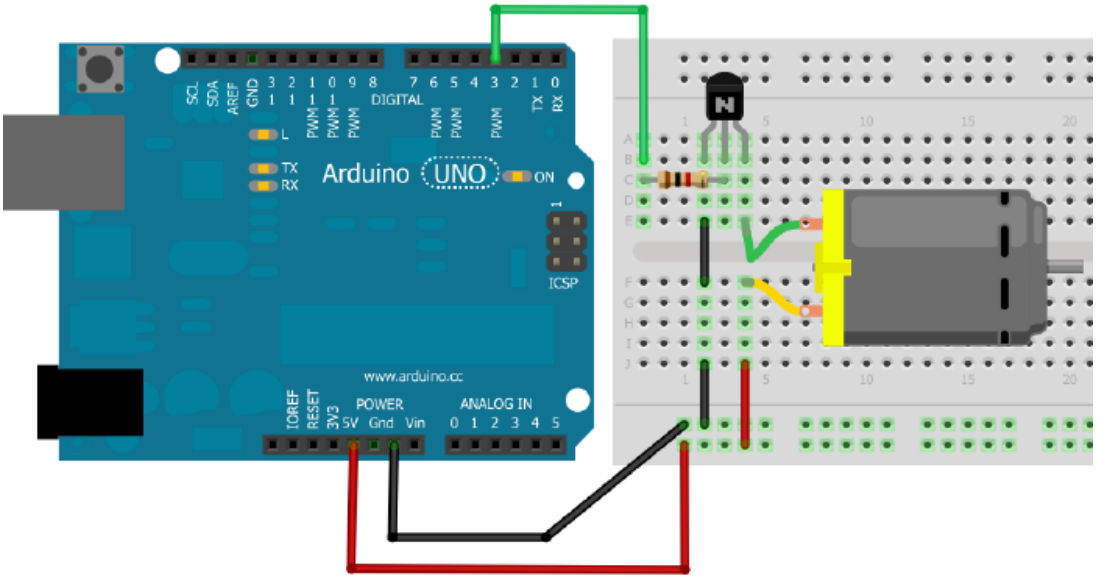
- Conocimientos previos
  - ✓ Señal análoga
  - ✓ Función analogWrite() y Serial.print()
  - ✓ PWM
  - ✓ Condicional y operadores de comparación
- ¿Qué aprendo?

- ✓ 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

5.10.1 Plano



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
```

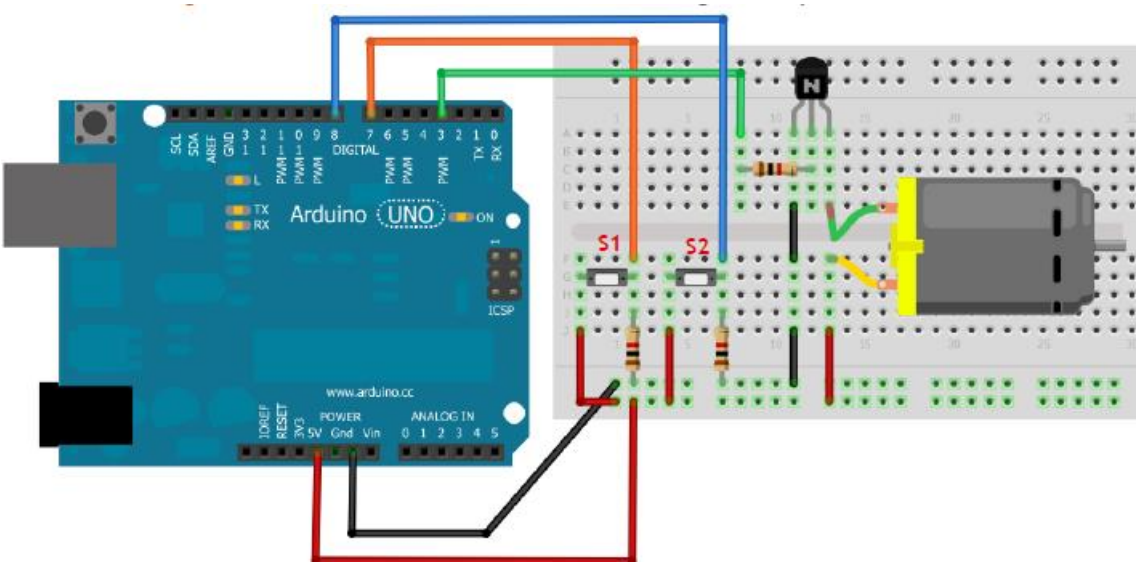
5.10.4 Ejercicios

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

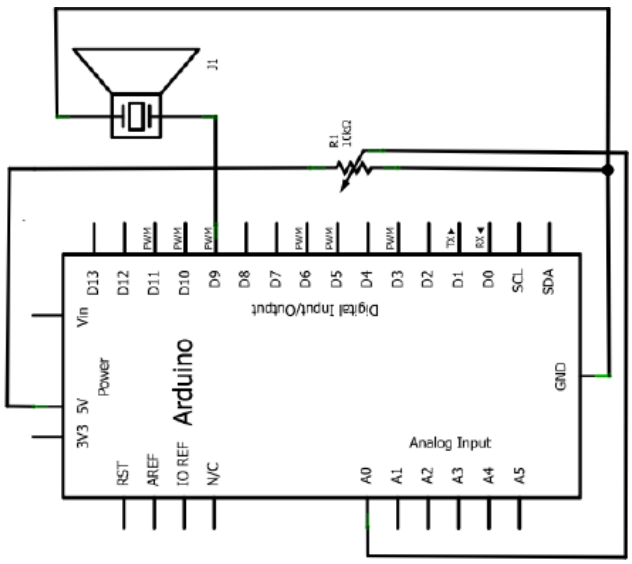
Pulsador 2 (S2) al pulsar S2 por primera vez el motor debe comenzar a girar a la velocidad escogida mediante S1, si se pulsa por segunda vez S2 el motor debe parar. Por Consola serial se debe mostrar "Licuadora girando a la velocidad x" o "Licuadora detenida" según corresponda.



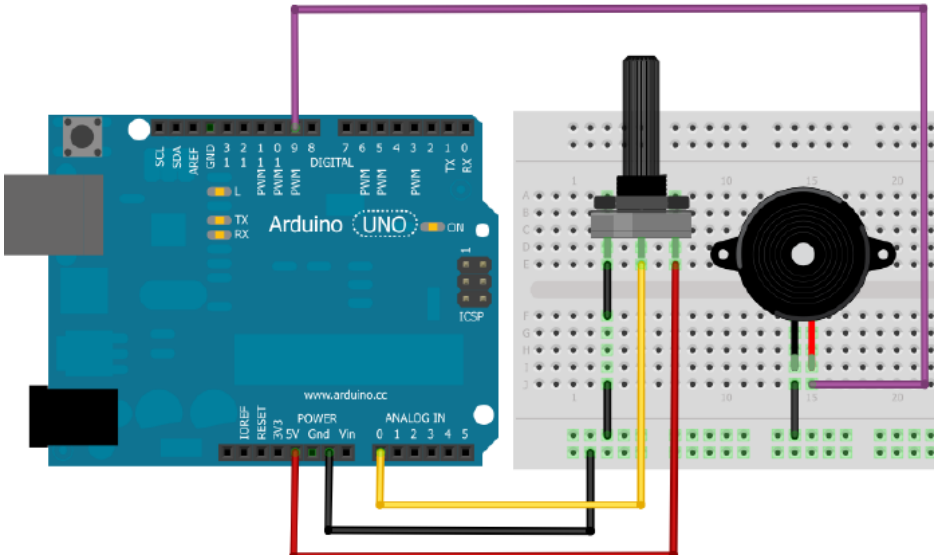
5.11 GENERAR TONOS POR UN BUZZER

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

5.11.1Plano



5.12.2Esquema



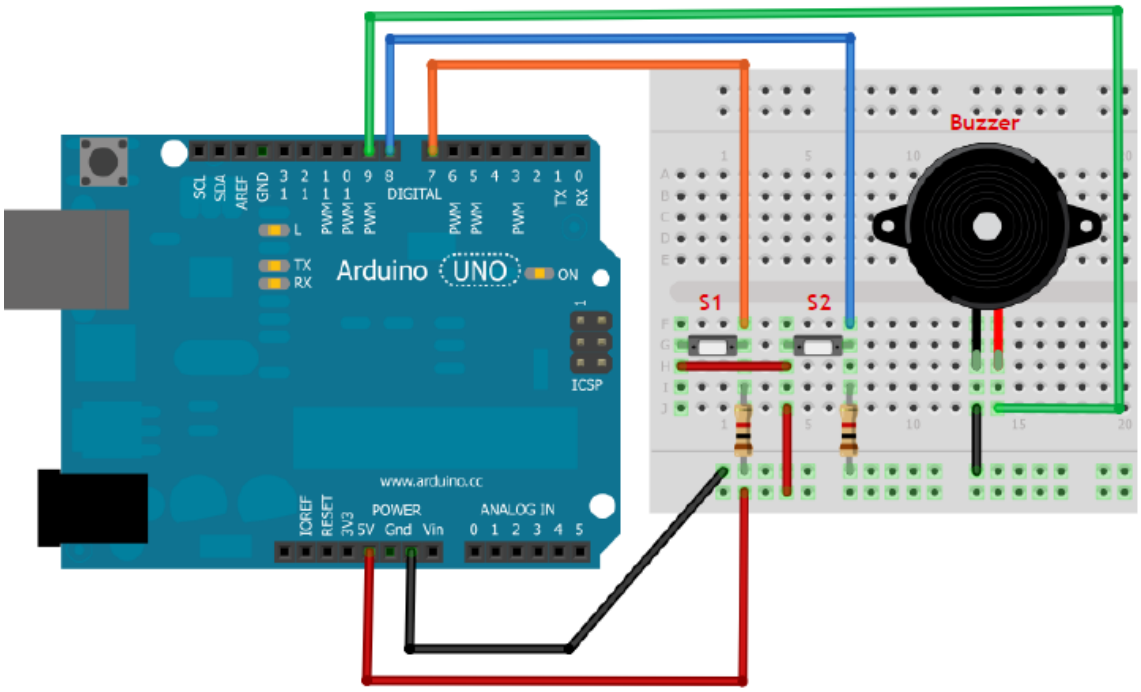
5.12.3Código de programación

```
/*
-----------------------------------------------
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
// 2da 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.

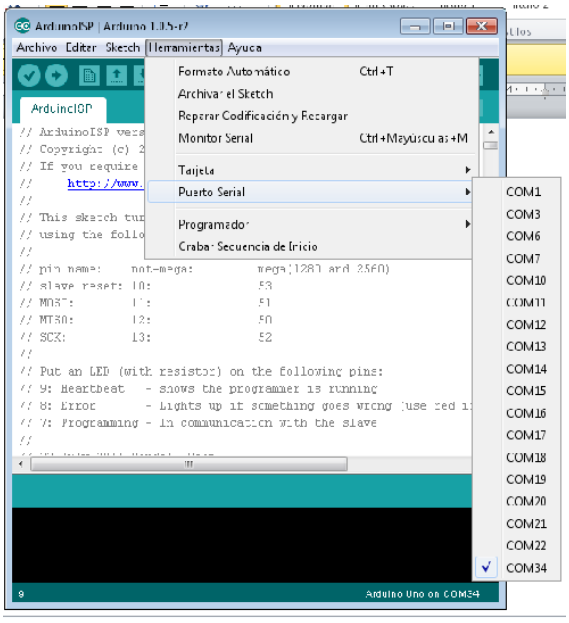
6.1 MATERIALES EMPLEADOS:

- 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

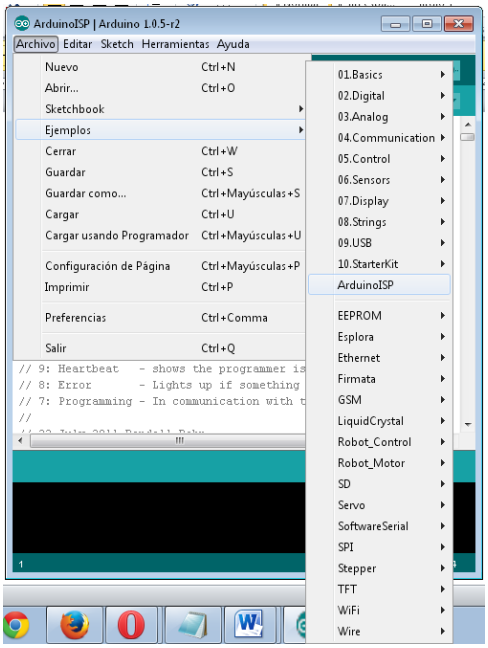
Ref:
http://www.instructables.com/id/Burning-the-Bootloader-on-ATMega328-using-Arduino-/all/?lang=es

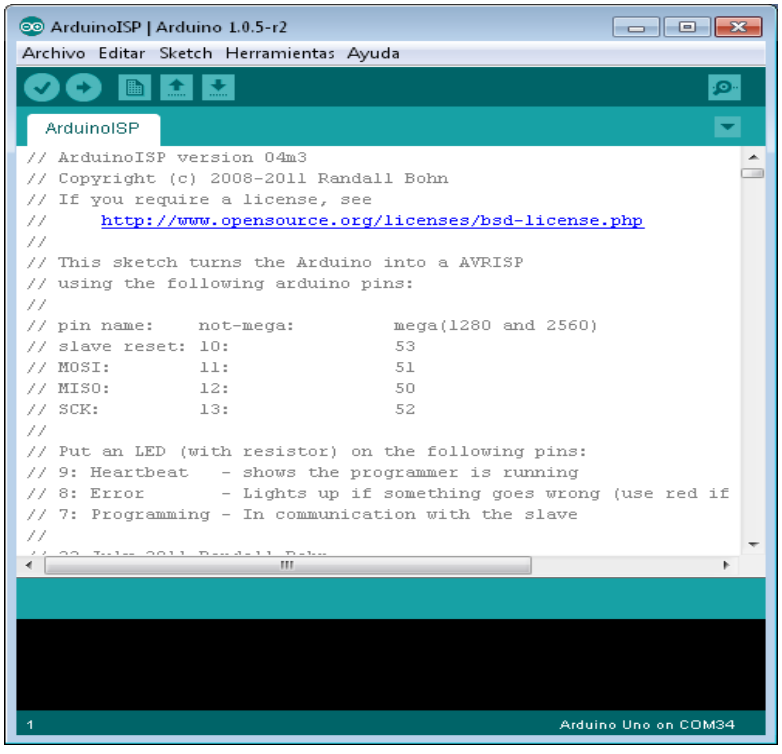
6.2 SOFTWARE

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



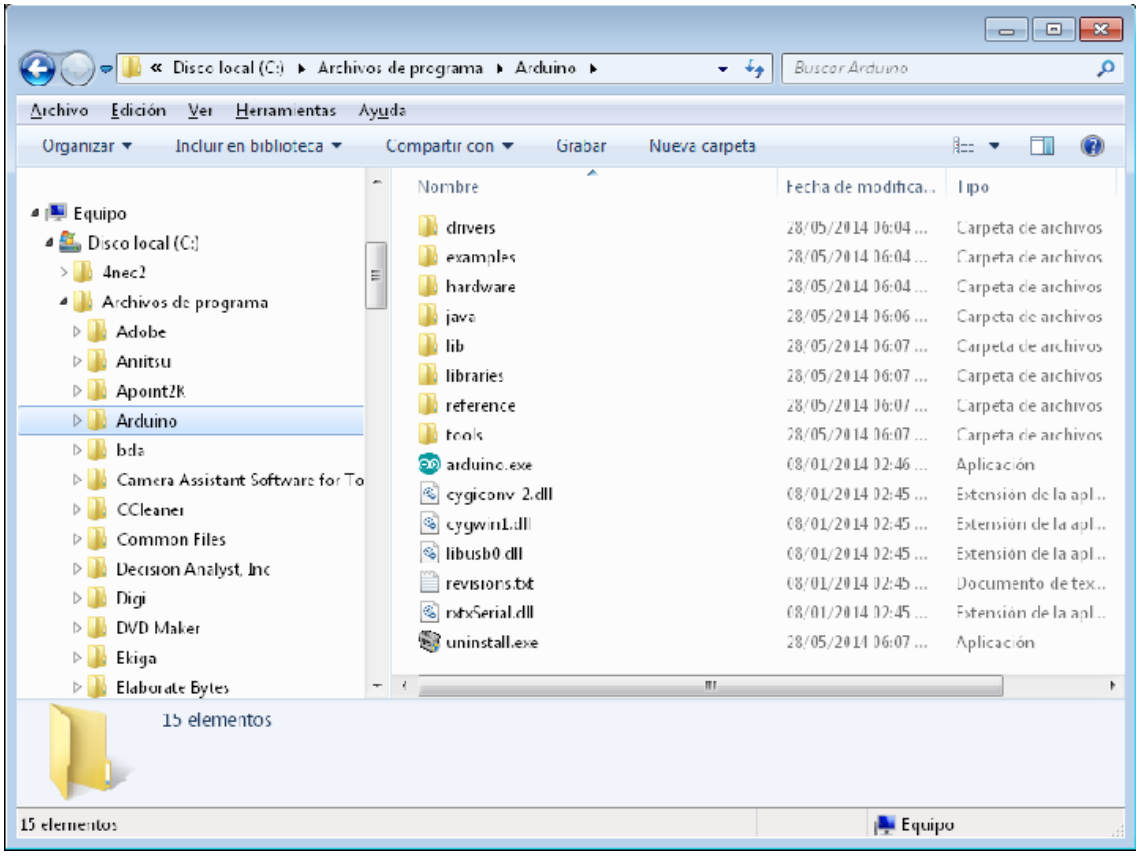
Paso2: Cargar al Arduino UNO como ISP, lo podemos encontrar en: Archivo>ejemplo>ArduinoISP. Luego de terminado la carga cerramos el IDE.



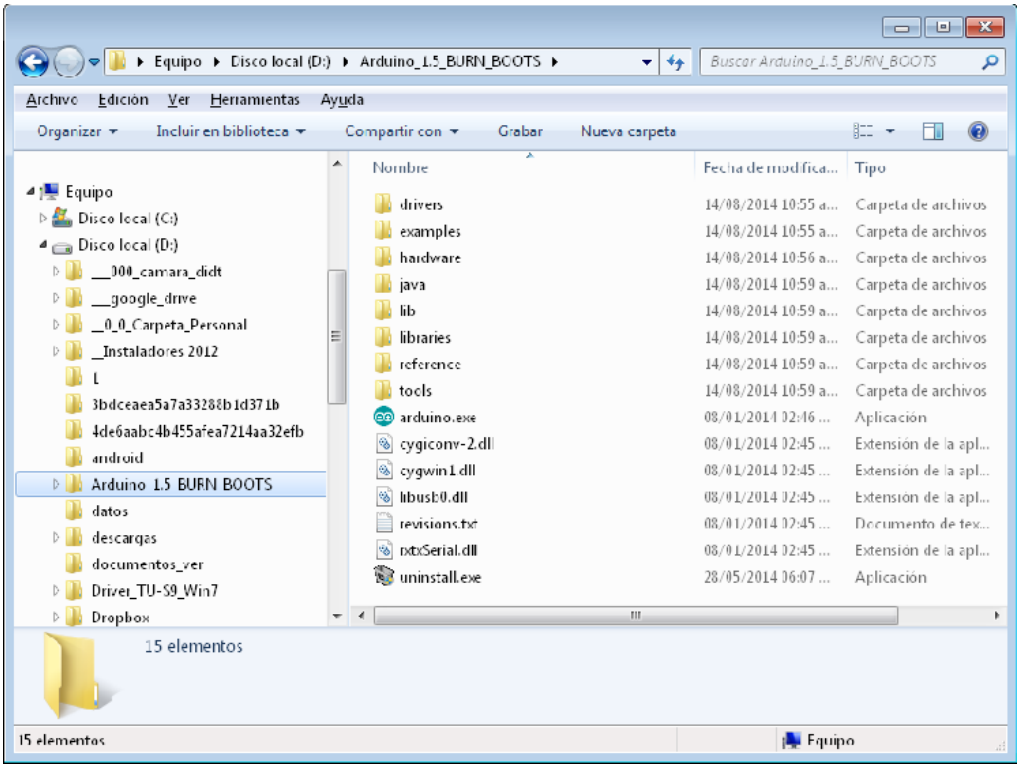


Paso3: Preparando el software IDE del Arduino UNO

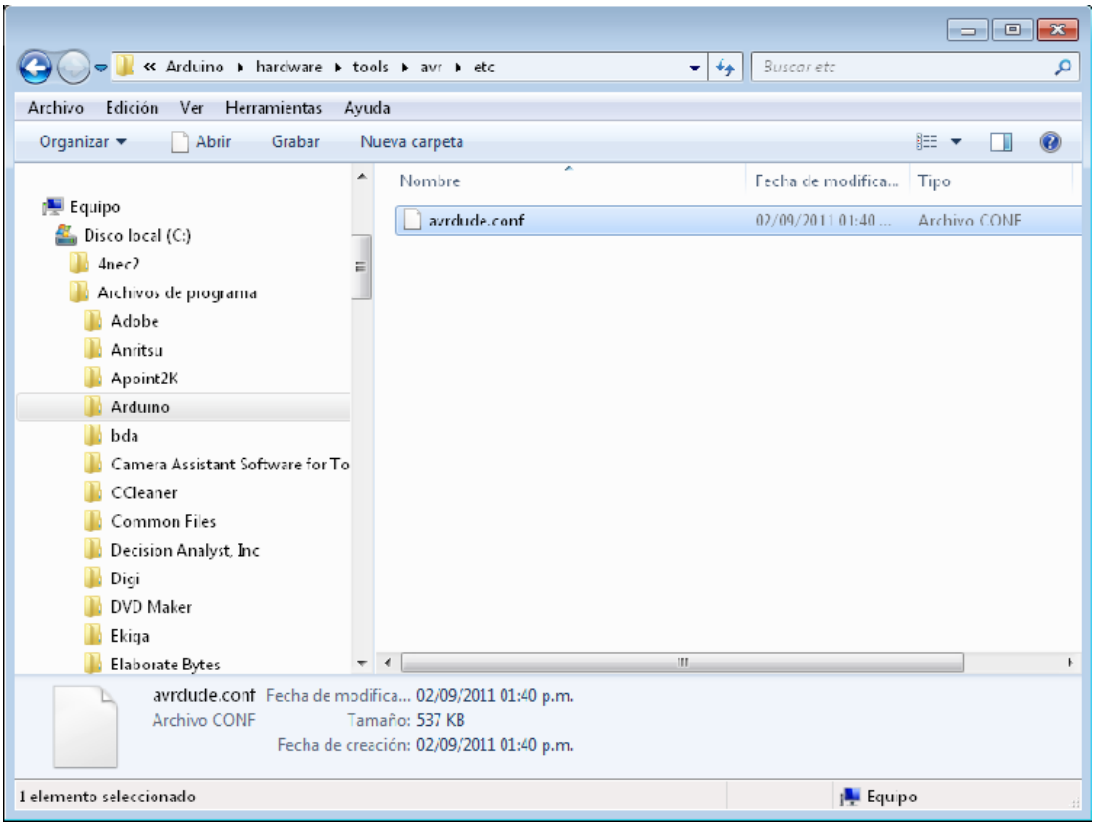
Al instalar el IDE del Arduino este se instaló en Archivo de programa/**Arduino**. Se debe copiar la carpeta **Arduino** completamente en otro disco o raíz. Ejemplo Disco D: **Arduino_1.5_BURN_BOOTS**



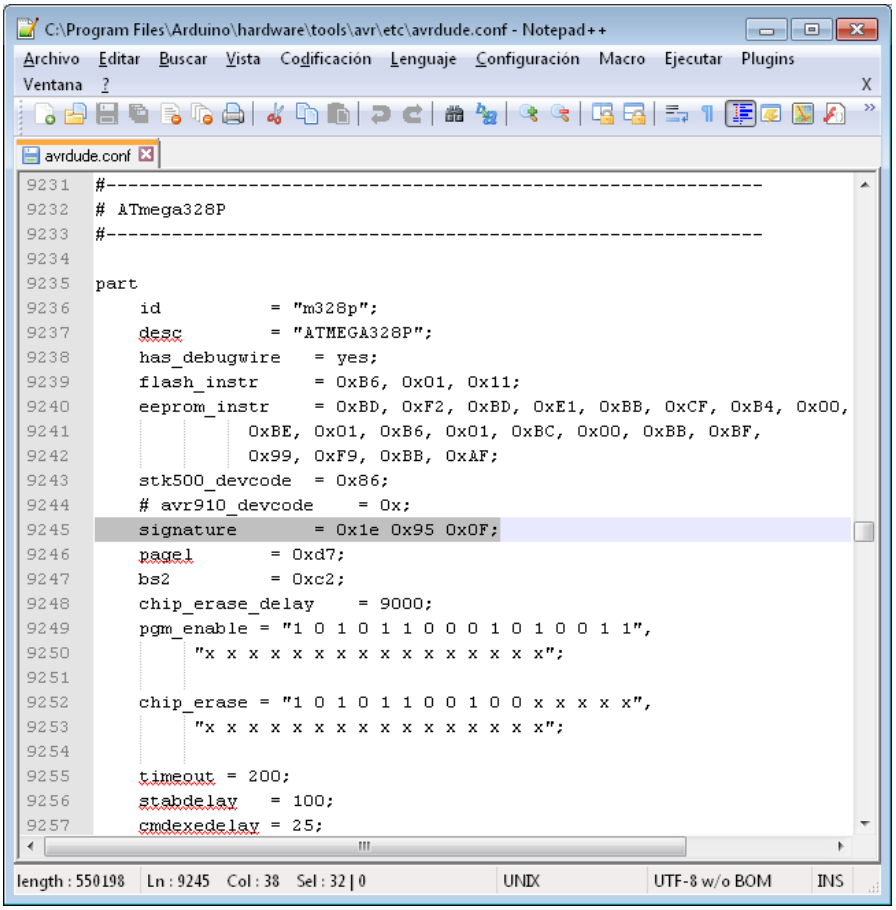
Esta carpeta esta en D: **Arduino_1.5_BURN_BOOTS** Hacemos doble clic en **arduino.exe** para comprobar que se ejecuta, luego lo cerramos.



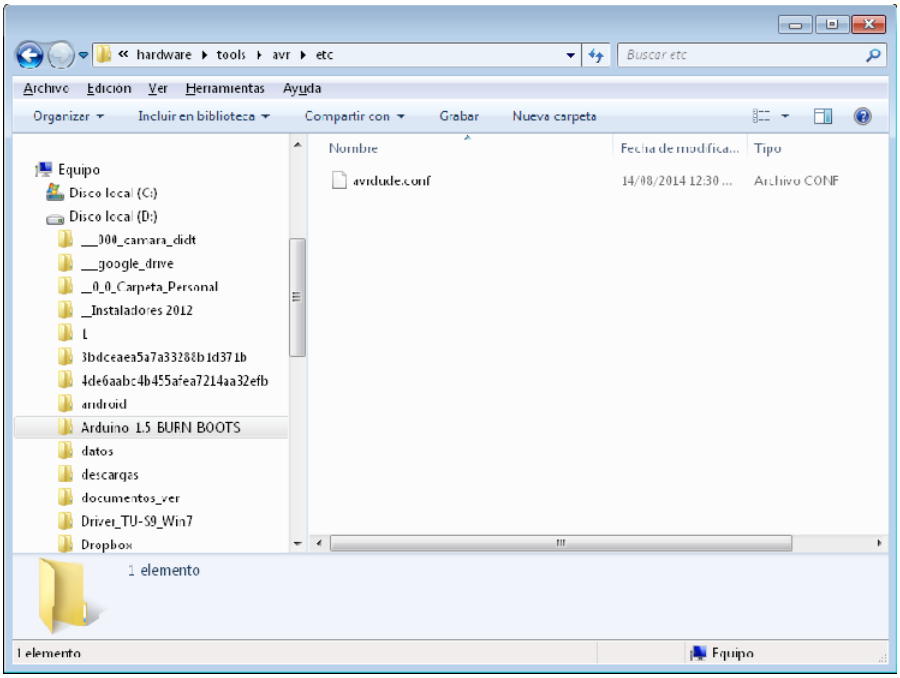
Paso4: Configurando el archivo de configuración de hardware **avrdude.conf**, lo ubicamos dentro del disco C: Arduino/hardware/tools/avr/etc/



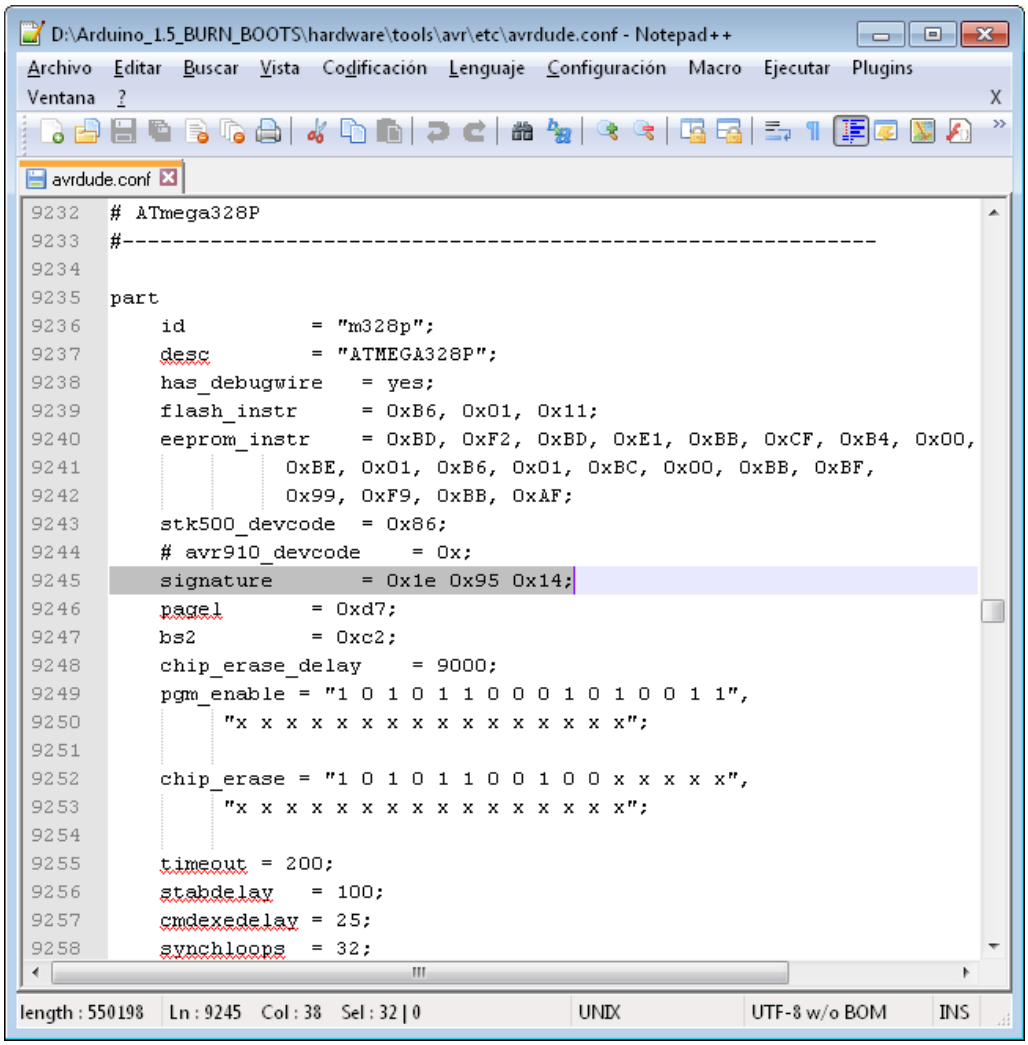
Al abrir el archivo avrdude.conf observamos en la fila 9245: **Signature = 0x1e 0x95 0x0F**; Que corresponde al ATmega 328p



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



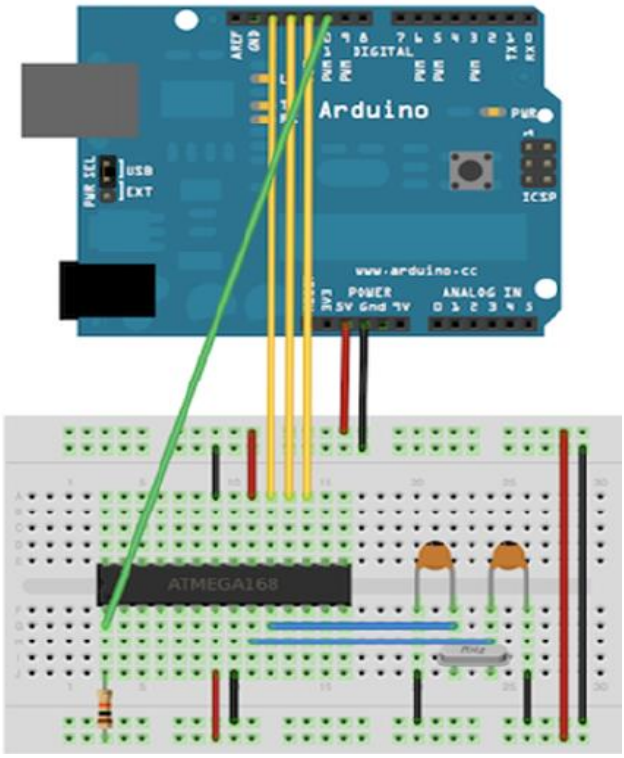
Al abrir el archivo avrdude.conf observamos en la fila 9245: **Signature = 0x1e 0x95 0x0F**; lo reemplazamos por: **Signature = 0x1e 0x95 0x14**; Guardamos y cerramos


D:\Arduino_1.5_BURN_BOOTS\hardware\tools\avr\etc\avrdude.conf - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Macro Ejecutar Plugins
Ventana ?
avrdude.conf
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 0xBE, 0x01, 0xB6, 0x01, 0xBC, 0x00, 0xBB, 0xBF,
9242 0x99, 0xF9, 0xBB, 0xAF;
9243 stk500_devcode = 0x86;
9244 # avr910_devcode = 0x;
9245 signature = 0x1e 0x95 0x14;
9246 pagel = 0xd7;
9247 bs2 = 0xc2;
9248 chip_erase_delay = 9000;
9249 pgm_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 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 x";
9254
9255 timeout = 200;
9256 stabdelay = 100;
9257 cmdexedelay = 25;
9258 synchloops = 32;
length : 550198 Ln : 9245 Col : 38 Sel : 32 | 0
UNIX
UTF-8 w/o BOM
INS

6.3 HARDWARE

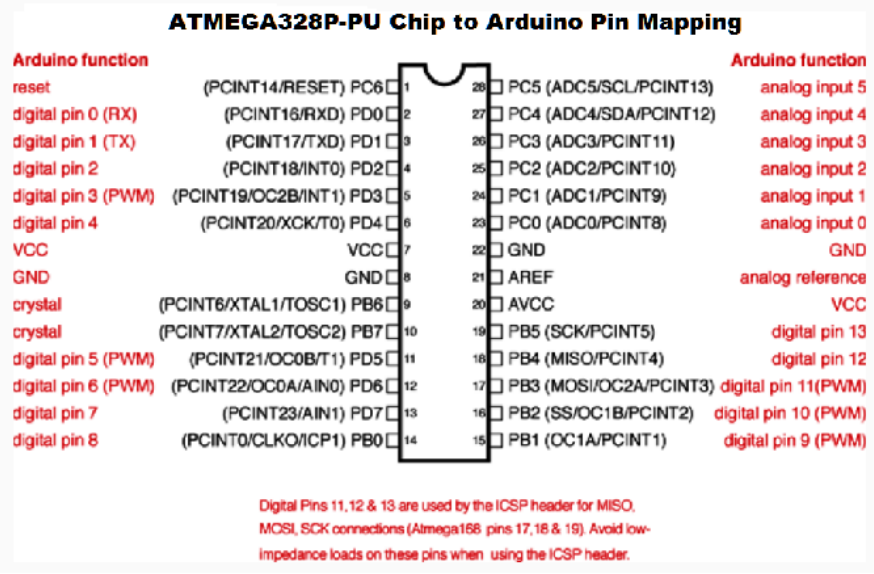
Paso1: Implementamos el siguiente diagrama a partir del Arduino UNO R3.

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



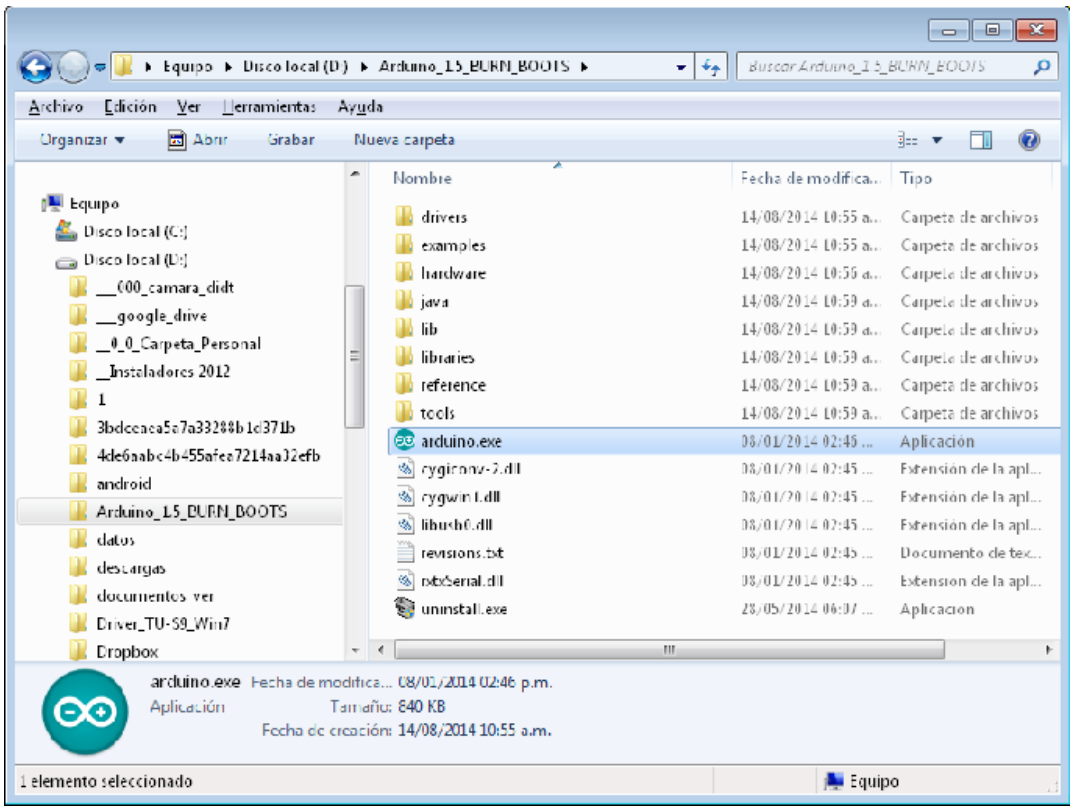
Para el caso del ATMEGA328P tenemos la distribución de pines el cual debe coincidir con el diagrama anterior.

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)

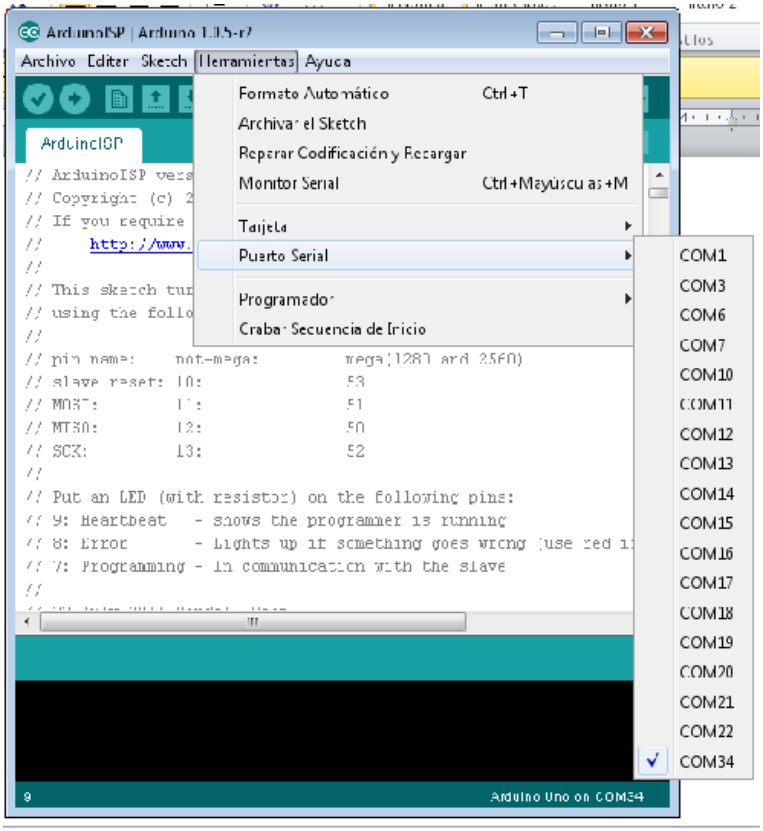


6.4 GRABANDO EL BOOTLOADER

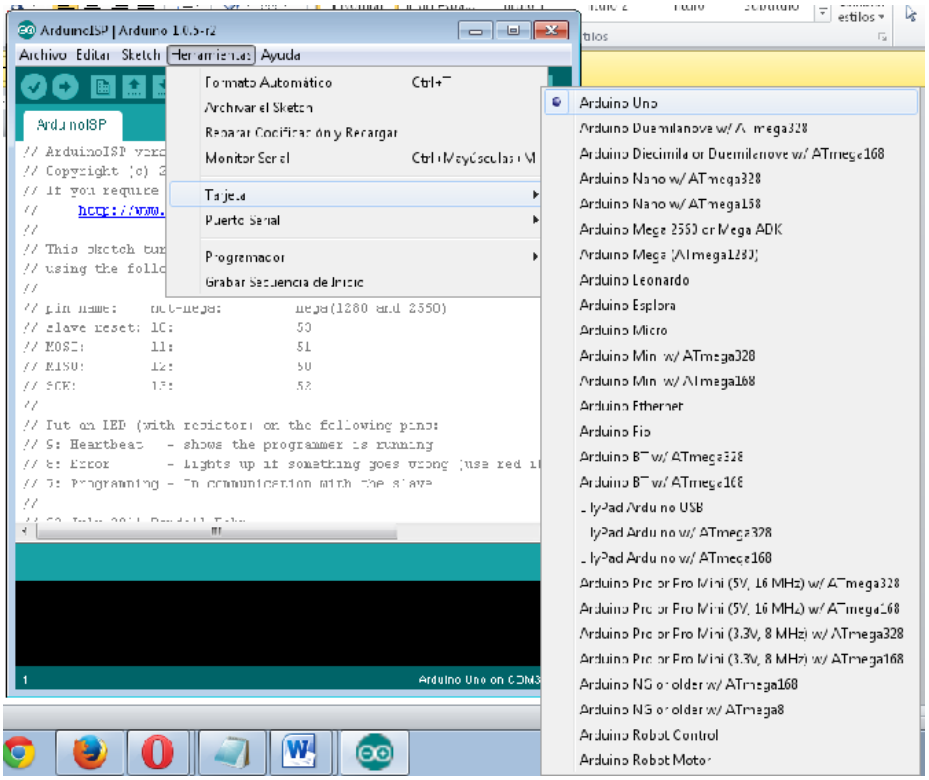
Paso1: Iniciamos el programa del ejecutable arduino.exe
El archivo ejecutable está ubicado dentro de la carpeta: D: Arduino_1.5_BURN_BOOTS.



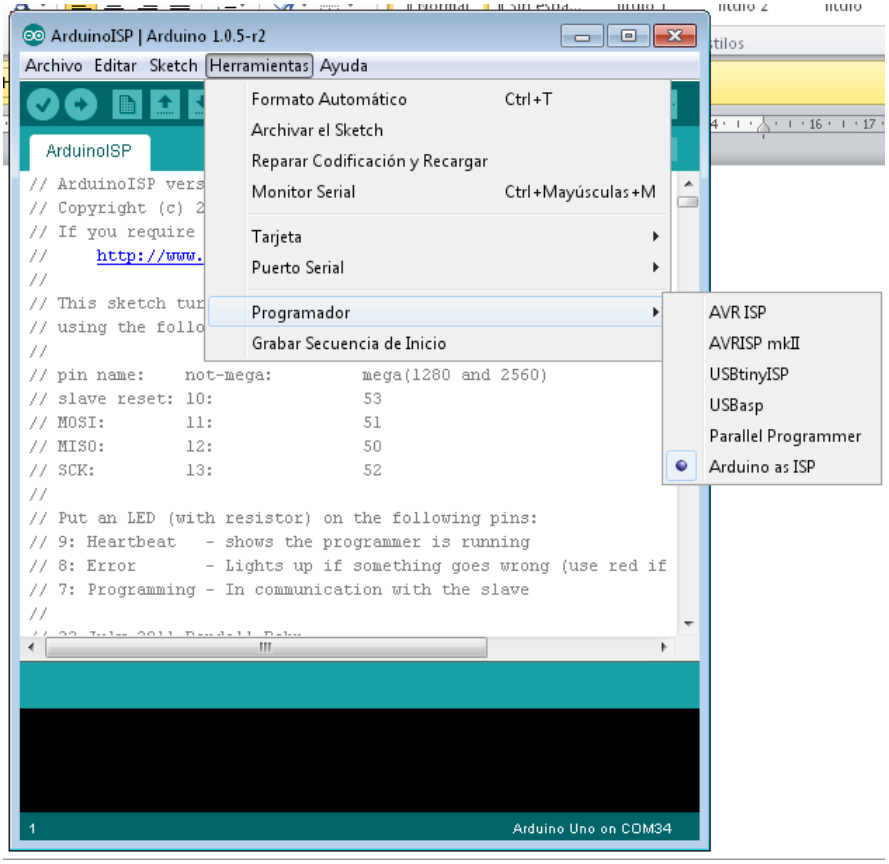
Verificar la conexión con el Arduino UNO R3 y la computadora.



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

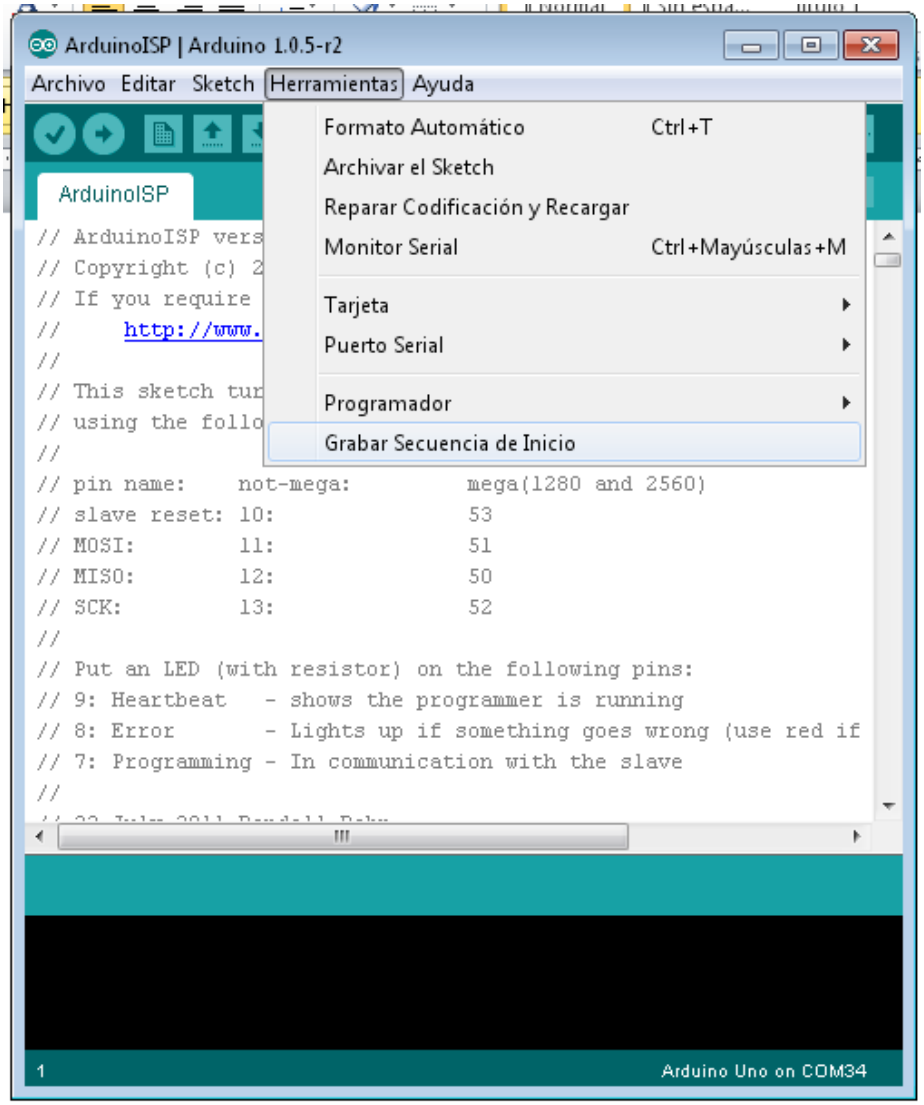


Cambiar el tipo de programador por defecto es **AVRISP mkII**. Seleccionar el programador Arduino as **ISP de: Herramientas/Programador/**



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.


ArduinoISP | Arduino 1.0.5-r2
Archivo Editar Sketch Herramientas Ayuda
Formato Automático Ctrl+T
Archivar el Sketch
Reparar Codificación y Recargar
Monitor Serial Ctrl+Mayúsculas+M
Tarjeta
Puerto Serial
Programador
Grabar Secuencia de Inicio
ArduinoISP
// ArduinoISP vers
// Copyright (c) 2
// If you require
// http://www.
// This sketch tur
// using the follo
// pin name: not-mega: mega(1280 and 2560)
// slave reset: 10: 53
// MOSI: 11: 51
// MISO: 12: 50
// SCK: 13: 52
// Put an LED (with resistor) on the following pins:
// 9: Heartbeat - shows the programmer is running
// 8: Error - Lights up if something goes wrong (use red if
// 7: Programming - In communication with the slave
1
Arduino Uno on COM34