**Arduino C 1**

**IoT**

**Sebastián Vargas Baena**

1. **¿Qué tamaño tiene el tipo de dato double? Compare el tipo double con el tipo float, ¿Qué puede concluir?**

* Doble precisión número de punto flotante. En la ONU y otras tarjetas ATMEGA basado, ocupa 4 bytes. Es decir, la doble ejecución es exactamente el mismo que el flotador, con ninguna ganancia en precisión. En el Arduino Due, dobles tienen precisión 8 bytes (64 bits).
* Tipo de datos de números de punto flotante, un número que tiene un punto decimal. Números de punto flotante se utilizan para aproximar valores analógicos y continuos porque tienen mayor resolución que enteros. Números de punto flotante pueden ser tan grande como 3.4028235E + 38 y tan bajas como - 3.4028235E + 38. Se almacenan como 32 bits (4 bytes) de información.

Los flotadores tienen sólo 6 a 7 dígitos decimales de precisión. Eso significa que el número total de dígitos, no el número a la derecha del punto decimal. A diferencia de otras plataformas, donde se obtiene mayor precisión utilizando un doble (por ejemplo hasta 15 dígitos), en el Arduino, doble es el mismo tamaño que flotan.

1. **Explore algunas de las bibliotecas estándar de arduino. Seleccione tres de ellas y explique para qué sirven y de un ejemplo cada una.**

**Wire** - Two Wire Interface (TWI/I2C) para enviar y recibir datos sobre una red de dispositivos o sensores.

Esta biblioteca le permite comunicarse con I2C / dispositivos de TWI. En las placas Arduino con el diseño de R3 (1,0 pinout), el SDA (línea de datos) y SCL (reloj de línea) están en las cabeceras de pin cerca el pin AREF. La debida Arduino tiene dos I2C / TWI interfaces SDA1 y SCL1 están cerca del pin AREF y el adicional sobre los pines 20 y 21.

Ejemplo: [Digital Potentiometer](http://arduino.cc/en/Tutorial/DigitalPotentiometer)

**GSM** - para conectar a un Red GSM/GRPS con el escudo GSM.

Esta biblioteca permite a una placa Arduino hacer la mayor parte de las operaciones que puede hacer con un teléfono GSM: lugar y recibir llamados de voz, enviar y recibir SMS y conectarse a internet en una red GPRS. El escudo GSM tiene un módem que transfiere datos desde un puerto serial a la red GSM. El módem realiza operaciones a través de una serie de comandos AT. La biblioteca de resúmenes de comunicaciones bajas niveles entre el módem y la tarjeta SIM. Se basa en la biblioteca de Software Serial de comunicación entre el moden y Arduino

Ejemplo: [Web Server](https://www.arduino.cc/en/Tutorial/GSMExamplesWebServer): crear un servidor web inalámbrico a través de GPRS

**WiFi** - Para conectarse a internet usando la Arduino WiFi shield

Esta biblioteca permite a una placa Arduino conectarse a internet. Puede servir como un servidor ya sea aceptar las conexiones entrantes o cliente hacer saliente. La biblioteca Soporta encriptación WEP y WPA2 Personal, pero no WPA2 Empresa. Nota, si el SSID no es transmitido, el escudo no se puede conectar también. Arduino se comunica con el escudo de WiFi mediante el bus SPI. Esto es en digitales pins 11, 12 y 13 en la ONU y pernos 50, 51 y 52 en la Mega. En ambas tarjetas, pin 10 se utiliza como SS.

En la Mega, no se utiliza el pin de hardware SS, 53, pero debe mantenerse como una salida o la interfaz SPI no funcionará. Pin digital 7 es utilizado como un pin de apretón de manos entre el protector de Wifi y el Arduino y no debe ser utilizado

Ejemplo: [UdpNTPClient](https://www.arduino.cc/en/Tutorial/UdpNTPClient) : Consulta un servidor Network Time Protocol (NTP) usando UDP

1. **¿Cuál es la estructura de un programa para un Arduino? ¿Cuál es la función de cada una de las partes de la estructura de un programa en Arduino?**

**Estructura**

* [setup](https://www.arduino.cc/en/Reference/Setup)()

La función setup() es llamada cuando se inicia un dibujo. Se usa para inicializar las variables, modos del perno, comenzar a usar las bibliotecas, etc.. La función de configuración sólo se ejecutará una vez, después de cada encendido o reset de la placa Arduino.

* [loop](https://www.arduino.cc/en/Reference/Loop)()

Después de crear una función setup(), que inicializa y establece los valores iniciales, la función loop() hace precisamente lo que su nombre sugiere y bucles consecutivos, permitiendo que su programa de cambio y responder.

1. **Localice el archivo main.cpp en el directorio donde descomprimió el archivo .zip descargado de la página del proyecto Arduino. No dude en revisar cada directorio. La función main en un programa en C es el punto de entrada del programa.**

**P5a. ¿Cuántas veces se ejecuta la función setup?**

* 1 ves

**P5b. ¿Cuántas veces se ejecuta la función loop?**

* Infinitas veces

**P5c. ¿Cuál es la ruta del archivo main.cpp en el sistema de archivos?**

* C:\Users\Administrador\Desktop\arduino-1.6.5-r2\hardware\arduino\avr\cores\arduino

1. **¿Qué tamaño en bytes y qué rango de valores tiene los siguientes tipos de datos: boolean, char, unsigned char, byte, int, unsigned int, word, long, unsigned long, float, double, string, String, array, void.**

* **Boolean:** Tiene uno de dos valores, verdadero o falso. (Cada variable boolean ocupa un byte de memoria).
* **Char:** Un tipo de datos que ocupa 1 byte de memoria que almacena un valor de carácter.
* **Unsigned char:** Un tipo de datos unsigned que ocupa 1 byte de memoria. Lo mismo que el tipo de datos byte. El tipo de datos unsigned char codifica números del 0 al 255.
* **Byte:** Un byte almacena un número sin signo de 8 bits, desde 0 hasta 255.
* **Int:** En el Arduino Uno (y otras tarjetas ATMega basado) un int almacena un valor de 16 bits (2 bytes). Esto produce una gama de-32,768 a 32.767 (valor mínimo de -2 ^ 15 y un valor máximo de (2 ^ 15) - 1). En el Arduino Due, un int almacena un valor de 32 bits (4 bytes). Esto produce una gama de

-2,147,483,648 a 2,147,483,647 (valor mínimo de -2 ^ 31 y un valor máximo de (2 ^ 31) - 1).

* **Unsigned int:** En la ONU y otras tarjetas ATMEGA basado, unsigned ints (enteros sin signo) son las mismas que ints que almacenan un valor de 2 bytes. En lugar de almacenar números negativos sin embargo solamente almacenan valores positivos, rindiendo una gama útil de 0 a

65.535 (2 ^ 16)-1) la debida almacena un valor de 4 bytes (32 bits), que van desde 0 hasta 4,294,967,295 (2 ^ 32-1).

* **Word:** Una palabra almacena un número sin signo de 16 bits, de 0 a 65535.
* **Long**: Largas variables son extendidas variables de tamaño para el almacenamiento de números y almacenan 32 bits (4 bytes), desde

-2,147,483,648 a 2,147,483,647.

* **Unsigned long:** Variables largas sin firmar son extendidas variables de tamaño para el almacenamiento de números y almacenan 32 bits (4 bytes). A diferencia de estándar longs largos sin firmar no almacenan números negativos, haciendo que su rango de 0 a 4,294,967,295 (2 ^ 32 - 1).
* **Float:** Números de punto flotante pueden ser tan grande como 3.4028235E + 38 y tan bajas como - 3.4028235E + 38. Se almacenan como 32 bits (4 bytes) de información
* **Doublé:** Doble precisión número de punto flotante. En la ONU y otras tarjetas ATMEGA basado, ocupa 4 bytes. Es decir, la doble ejecución es exactamente el mismo que el flotador, con ninguna ganancia en precisión. En el Arduino Due, dobles tienen precisión 8 bytes (64 bits).
* **String:**
* **Array:**
* **Void:** La palabra clave void se utiliza sólo en las declaraciones de función. Indica que se espera que la función no devuelva ninguna información a la función de la cual se llamaba

1. **¿Para qué sirve terminar un string en null (código ASCII 0)?**

Esto nos permite identificar donde es el final de la cadena, donde no se use este final se podría seguir leyendo bytes posteriores de la memoria que no son parte real de la cadena. De la única manera que no sería necesario usar el null, es cuando se ha especificado la duración, por ejemplo si se dice que la duración es Str2, aunque se puede comportar de forma extraña

1. **¿Cuál es el código ASCII para los número del 0 al 9?**

codigo ascii 00 = NULL ( Carácter nulo )

codigo ascii 01 = SOH ( Inicio de encabezado )

codigo ascii 02 = STX ( Inicio de texto )

codigo ascii 03 = ETX ( Fin de texto, palo corazón barajas inglesas de poker )

codigo ascii 04 = EOT ( Fin de transmisión, palo diamantes barajas de poker )

codigo ascii 05 = ENQ ( Consulta, palo tréboles barajas inglesas de poker )

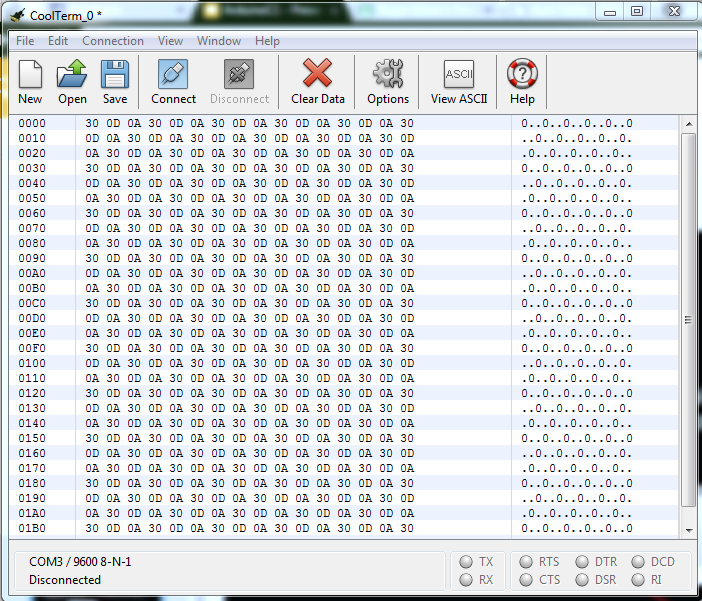
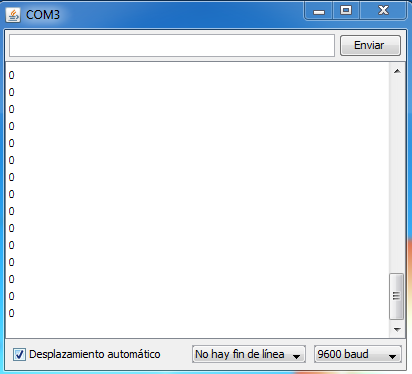
codigo ascii 06 = ACK ( Reconocimiento, palo picas cartas de poker )

codigo ascii 07 = BEL ( Timbre )

codigo ascii 08 = BS ( Retroceso )

codigo ascii 09 = HT ( Tabulador horizontal )

1. ¿¿??
2. **Explique la diferencia entre ambos. Utilice como referencia una tabla de códigos ASCII.**



1. **¿Cómo se crean un arreglo? muestre y explique un ejemplo.**

(Declaración) de crear una matriz

Todos de los siguientes métodos son formas válidas de crear (declarar) una matriz.

int myInts [6];

int myPins [] = {2, 4, 8, 3, 6};

int mySensVals [6] = {2, 4, -8, 3, 2};

mensaje de Char [6] = "Hola";

Puede declarar una matriz sin inicializar como en myInts. En myPins declaramos un array sin explícitamente elegir un tamaño. El compilador cuenta con los elementos y crea una matriz del tamaño adecuado. Finalmente usted puede inicializar y tamaño de su matriz, como en mySensVals. Tenga en cuenta que al declarar un array de tipo char, un elemento más de su inicialización es necesario, para mantener el necesario carácter nulo.

Ejemplo

Para crear un arreglo se puede hacer de dos maneras un ejemplo de ello es:

char name[]= "Sebastian";

y tiene sus variaciones:

char name[10]= "Sebastian";

La otra forma es:

char name[]= {'s','e','b','a','s','t','i','a','n','\0'};

char name[10]= {'s','e','b','a','s','t','i','a','n','\0'};

char name[]= {'s','e','b','a','s','t','i','a','n'};

1. **¿Cómo se accede y cómo se escribe un elemento de un arreglo? muestre y explique un ejemplo.**

Las matrices son cero indexado, es decir, refiriéndose a la inicialización de la matriz anterior, es el primer elemento de la matriz en el índice 0, por lo tanto  
mySensVals[0] == 2, mySensVals[1] == 4, and so forth.

También significa que en una matriz con diez elementos, índice nueve es el último elemento. Por lo tanto:

int myArray[10]={9,3,2,4,3,2,7,8,9,11};

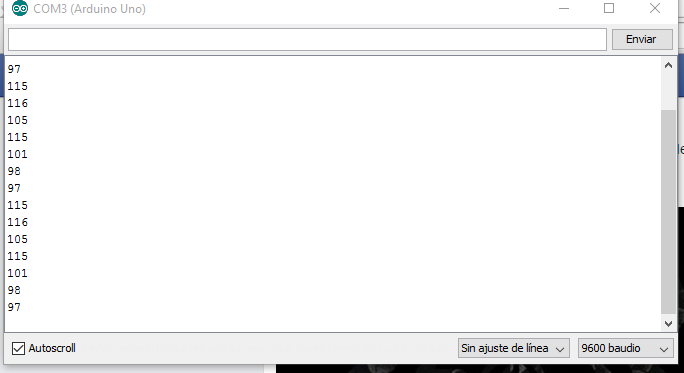
// myArray[9] contains 11

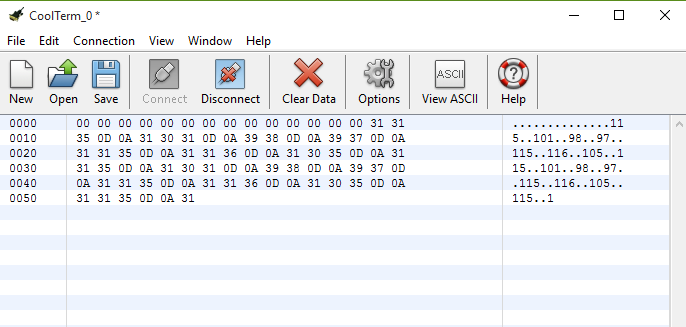
// myArray[10] is invalid and contains random information (other memory address)

Por esta razón debe tener cuidado en el acceso a arreglos de discos. Acceso a más allá del final de una matriz (con un número de índice mayor que el tamaño de la matriz declarado - 1) es la lectura de la memoria que está en uso para otros fines. Lectura desde estos lugares probablemente no va a hacer mucho excepto rendimiento datos no válidos. Escritura para localizaciones de memoria al azar es definitivamente una mala idea y a menudo puede conducir a resultados infelices, como accidentes o mal funcionamiento del programa. Esto también puede ser un error difícil de rastrear.

A diferencia de BASIC o JAVA, el compilador de C no hace ninguna comprobación para ver si acceso array está dentro de límites legales del tamaño de la matriz que se ha declarado.

1. **Cree un arreglo con su nombre y escriba un programa que envíe cada uno de los caracteres del nombre al puerto serial. Observe el resultado utilizando CoolTerm.**

****

****

1. **¿Cuáles son los operadores de comparación? de ejemplos de cada uno**

**Operadores de comparación**

* == (equal to)

Ejemplo x==y (x es igual a y)

* != (not equal to)

Ejemplo x[!=](https://www.arduino.cc/en/Reference/If)y (x no es igual a y)

* < (less than)

Ejemplo x<y (x es menor que y)

* > (greater than)

Ejemplo x>y (x es mayor que y)

* <= (less than or equal to)

Ejemplo x<=y (x es menor o igual a y)

* >= (greater than or equal to)

Ejemplo x>=y (x es mayor o igual a y)

1. **Construya programas ejemplo para las siguientes estructuras de control: if, if-else, if-else if- else, for, switch-case, while.**

• if

if (pulsado==0); {

digitalWrite(led1,HIGH);

delay(1000);

digitalWrite(led2,HIGH);

delay(1000);

digitalWrite(led3,HIGH);

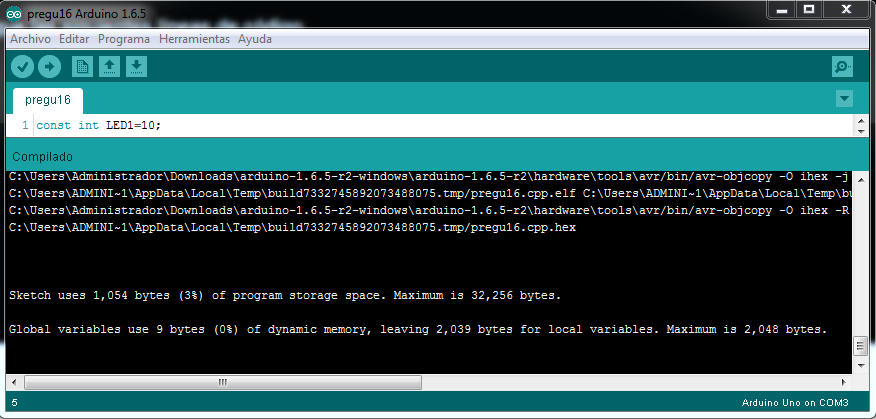
delay(1000);

}

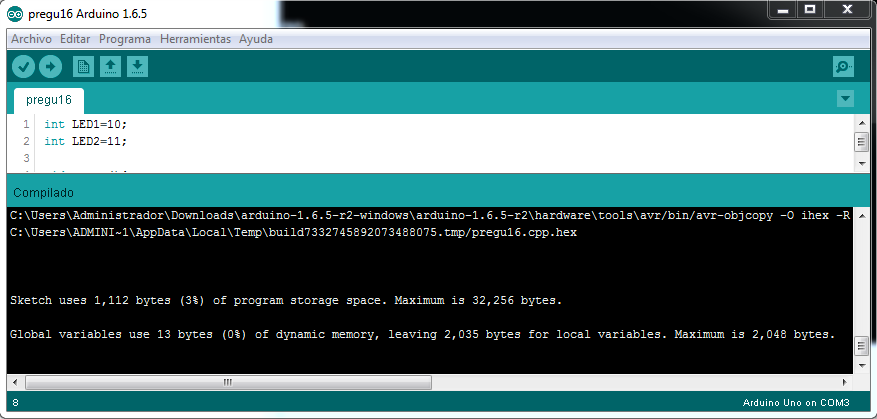
Este programa enciende 3 leds, uno despues de otro cada 1 segundo, aunque el pulsador se suelta igualmente van a estar prendidos los leds.

1. **Escriba e implemente el código que se muestra a la derecha. ¿Para qué sirve el modificador const?**

Para optimizar el funcionamiento del compilador, son constantes y no ocupan espacio en la memoria RAM.



1. **Compile de nuevo el programa. Compare el mensaje final del proceso de compilación con el mensaje que aparece en la figura de este slide. ¿Qué se puede concluir?**



Lo que se observa es que el tamaño que al implementar const era de 9 bytes y ahora solo aplicando int el tamaño es de 13 bytes, lo que pasa es que con const se le da un valor fijo a la variable y con el int seria solo un entero y este hace que ocupe más memoria RAM, la ventaja de usar const es que este es invisible a la memoria.

1. **¿Cómo funciona? explique cada línea de código**

Si los caracteres ingresados son minúsculas se convierten en mayúsculas, con esto comienzo a validar las opciones que se tienen de la A – F y de acuerdo a la opción que se elija se arroja uno de los mensajes establecidos en el menú de opciones del switch().

La explicación de cada línea de código se encuentra en el script del Arduino.

1. **¿Cómo funciona el programa que muestra la figura de la derecha?**

La condición cuando k=0 la ejecuta solo una vez porque cada que el for(;;) se ejecuta k aumenta y a medida que esto pasa sale el mensaje que aparece en código y se hace la multiplicación de k\*k.

1. **Explique el programa que muestra la figura.**

**P20a. ¿Para qué sirve memset?**

**P20b. ¿Para qué sirve sizeof?**

**P21c. ¿Qué valor devuelve sizeof(searchList)?**

**P20d. ¿Cuál es la función de la palabra reservada break en la línea 14?**