El uso de los Timer no es exclusivo de las salidas PWM, si no que es compartido con otras funciones.

Emplear funciones que requieren el uso de estos Timer supondrá que no podremos emplear de forma simultánea alguno de los pines PWM.

Incompatibilidades más frecuentes:

- *Servo: La librería servo hace un uso intensivo de temporizadores por lo que, mientras la estemos usando, no podremos usar algunas de las salidas PWM. En el caso de Arduino Uno, Mini y Nano, la librería servo usa el Timer 1, por lo que no podremos usar los pines 9 y 10 mientras usemos un servo.*
- *Comunicación SPI: En Arduino Uno, Mini y Nano, el pin 11 se emplea también para la función MOSI de la comunicación SPI. Por lo tanto,, no podremos usar ambas funciones de forma simultánea en este pin.*
- *Función Tone: La función Tone emplea el Timer 2, por lo que no podremos usar los pines 3 y 11.*

4.7 EL SISTEMA DE MEMORIA DE ARDUINO

Un microcontrolador usado en Arduino tiene varios tipos de memoria integradas en el mismo chip, en el caso de Arduino y los microcontroladores AVR de Atmel usan tres tipos de memorias.

Memoria RAM para el manejo de variables, igual que en una computadora.

La memoria FLASH o memoria de programa, algo como el disco duro de la computadora, y la memoria EEPROM, memoria para el manejo de variables pero que no se pierden cuando el controlador se apaga, su uso es similar al de una memoria SD.

Arduino	Processor	Flash	SRAM	EEPROM
UNO, Uno Ethernet, Menta, Boarduino	Atmega328	32K	2K	1K
Leonardo, Micro, Flora, 32U4 Breakout, Teensy, Esplora	Atmega 32U4	32K	2.5K	1K
Mega, MegaADK	Atmega2560	256K	8K	4K

- ***SRAM :*** *Variables locales, datos parciales. Es la zona de memoria donde el sketch crea y manipula las variables cuando se ejecuta. Es un recurso limitado y debemos supervisar su uso para evitar agotarlo, en un microcontrolador no podemos agregar más memoria RAM como en una computadora.*

- ***EEPROM:*** *Memoria no volátil para mantener datos después de un reset o cuando el microcontrolador se apaga. Se puede grabar desde el programa del microcontrolador, usualmente, constantes de programa. Las EEPROMs tienen un número limitado de lecturas/escrituras, tener en cuenta a la hora de usarla. Esta memoria solo puede leerse byte a byte y su uso puede se un poco incómodo. También es algo más lenta que la SRAM. La vida útil de la EEPROM es de unos 100.000 ciclos de escritura. En esta memoria se podría almacenar la clave de accesos de una alarma para que el propio usuario pudiera cambiarla sin necesidad de reprogramar todo el microcontrolador.*

- ***Flash:*** *Memoria de programa. Usualmente desde 1 Kb a 4 Mb (en controladores de familias grandes). Es donde se guarda el sketch o programa Arduino ya compilado. Sería el equivalente al disco duro de una computadora. En la memoria flash también se almacena del bootloader. Se puede ejecutar un programa desde la memoria flash, pero no es posible modificar los datos, sino que es necesario copiar los datos en la SRAM para modificarlos.*

  La memoria flash usa la misma tecnología que las tarjetas SD, los pen drives o algunos tipos de SSD, esta memoria tiene una vida útil de unos 100.000 ciclos de escritura, así que cargando 10 programas al día durante 27 años podríamos dañar la memoria flash de manera permanente.

Memoria de Arduino UNO:

- Flash 32k bytes (el bootloader usa 0.5k)
- SRAM 2k bytes
- EEPROM 1k byte

Memoria de Arduino MEGA:

- Flash 256k bytes (el bootloader 8k)
- SRAM 8k bytes
- EEPROM 4k byte

4.7.1 Algunos detalles de la memoria SRAM

Al ser el recurso más escaso en Arduino hay que entender bien cómo funciona. La memoria SRAM puede ser leída y escrita desde el programa en ejecución.

La memoria SRAM es usada para varios propósitos:

- ***Static Data:*** *Este bloque de memoria reservado en la SRAM para todas las variables globales y estáticas. Para variables con valores iniciales, el sistema copia el valor inicial desde la flash al iniciar el programa.*

- ***Heap:*** *Es usado para las variables o elementos que asignan memoria dinámicamente. Crece desde el final de la zona de Static Data a medida que la memoria es asignada. Usada por elementos como los objetos y los Strings.*

- ***Stack:*** *Es usado por las variables locales y para mantener un registro de las interrupciones y las llamadas a funciones. La pila crece desde la zona más alta de memoria hacia el Heap. Cada interrupción, llamada de una función o llamada de una variable local produce el crecimiento de la memoria.*

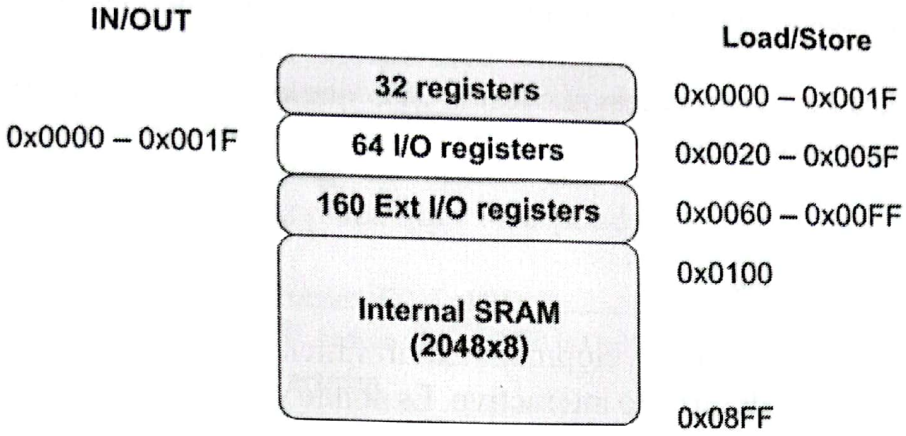
  La mayor parte de los problemas ocurren cuando la pila y el Heap colisionan.

  Cuando esto ocurre una o ambas zonas de memoria se corrompen con resultados impredecibles. En algunos casos se produce un "cuelgue" del programa y en otros casos la corrupción de memoria puede deriva en funcionamientos erráticos.

A partir de la versión 1.6 del IDE al compilar un sketch nos da el tamaño que va a ocupar en la flash el proyecto y el espacio que va a ocupar en la SRAM las variables globales, es decir la zona de static data.

Recuerde que en la memoria SRAM también se encuentran los registros que ocupan las primeras 256 direcciones de memoria. Por lo tanto, la memoria SRAM para el uso de variables del usuario empieza a partir de la dirección 0x0100.

Los registros al estar en la SRAM son volátiles y no conservan su valor después de un reset. Mirando la documentación del microcontrolador se puede ver las direcciones asignadas a las regiones de memoria.



A su vez la memoria SRAM tiene regiones para la gestión de datos variables, la administración de estas regiones se realiza en forma automática pero también depende de la forma en que declaramos una variable en un programa pues esta declaración define en que región de memoria se almacenará.

Data variables es la primera sección de RAM y se usa para almacenar datos estáticos del programa, como cadenas, estructuras inicializadas y variables globales.

Las variables .bss son la memoria asignada para variables globales y estáticas no inicializadas.

Heap es el área de memoria dinámica.

La pila es el área de memoria ubicada al final de la RAM y crece hacia el área de datos. El área de pila se usa para almacenar el estado del microcontrolador al saltar a una subrutina y recuperarlo luego, valores de retorno en interrupciones, etc. Como ocurre con todos los microcontroladores el manejo inadecuado del Stack o pila puede traer problemas serios ya que si el Stack crece de manera descontrolada destruirá las otras variables del programa y todo el sistema colapsa.

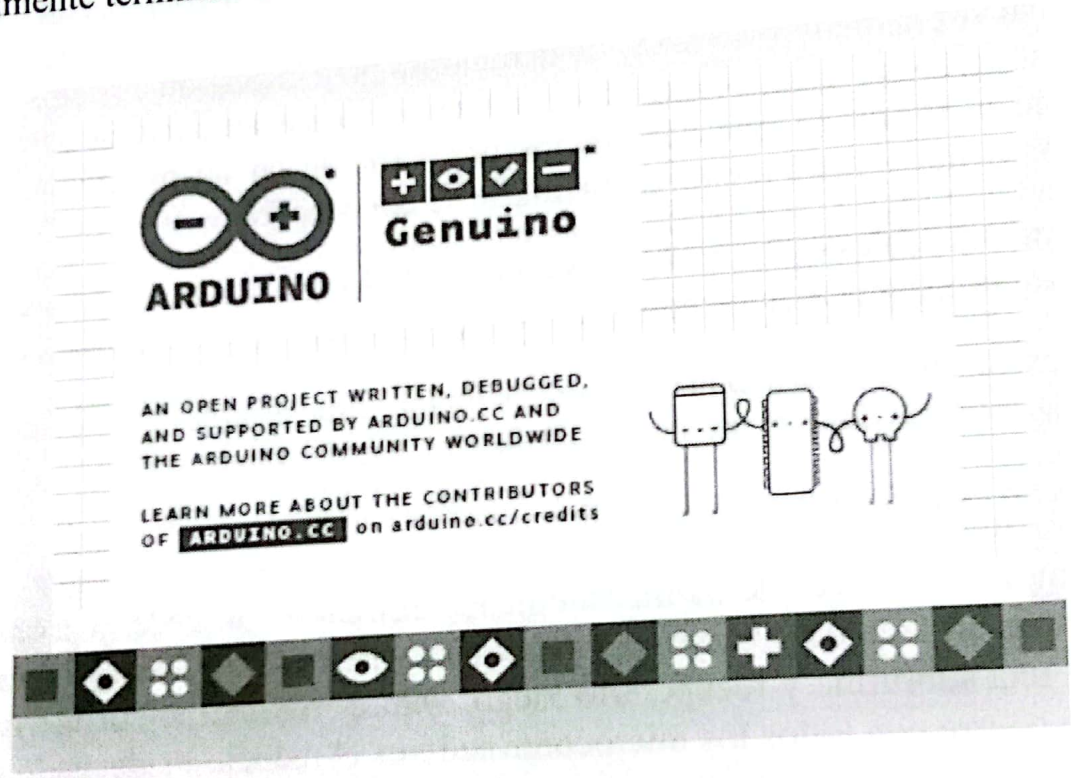
Es importante que siempre exista suficiente memoria libre entre el Stack y el bloque de memoria de datos del usuario. Cuando está área es demasiado pequeña para las tareas requeridas o falta espacio para el Stack, el microcontrolador comienza a extraviarse o reiniciarse.

Normalmente el programador no se preocupa por sus variables y donde se almacenan, esto lo resuelve el compilador, sin embargo como veremos más adelante, si importa conocer la naturaleza de la variable que estamos usando para ayudar al compilador a un manejo más eficiente de la memoria SRAM.

En algunos microcontroladores es posible agregar memoria RAM externa de una manera similar a como lo hacemos con una PC. Por lo general, esto es costoso (unos pocos KB de RAM externos cuestan más que el propio microcontrolador) y también requiere habilidades avanzadas de hardware y software.

4.8 EL IDE DE ARDUINO

*El IDE (*Integrated Development Environment*) es el* entorno de desarrollo integrado o entorno de desarrollo interactivo. Es donde vamos a escribir los programas que finalmente terminan en la memoria FLASH del microcontrolador.



El IDE de Arduino y todo lo necesario para trabajar se descarga desde su sitio en Internet, es libre y solo con descargarlo ya está listo para trabajar con el. Al momento de escribir estas líneas hay disponible para su descarga una versión que no necesita ser instalada, se descarga un archivo zip se descomprime y todas las herramientas quedan listas para su uso.

El entorno de trabajo de Arduino es bastante austero si lo comparamos con otros entornos para otras arquitecturas sin embargo es totalmente funcional. Se puede escribir un programa en él, depurarlo e incluso tiene todo lo necesario para programar la memoria del microcontrolador.