Memoria del Arduino (recopilado de Internet – Fuente al final del artículo)

Explicamos los diferentes tipos de memoria existentes en el microcontrolador usado por la placa Arduino. Indicamos qué uso tiene cada uno de ellos y mostramos un ejemplo de escritura y lectura de valores en la memoria no volátil EEPROM. Mostramos algunas alternativas para ampliar el almacenamiento en Arduino como EEPROM externa ó SD Card.

Tipos de memoria en Arduino: Flash, SRAM y EEPROM

Existen tres tipos de memoria en los microcontroladores utilizados por las placas Arduino (ATmega168, ATmega328, ATmega1280, etc.): memoria Flash, memoria SRAM y memoria EEPROM. A continuación indicamos las diferencias de cada una de ellas y el uso que Arduino puede darles.

Memoria Flash (espacio del programa) en Arduino

La **memoria Flash** (espacio del programa) es donde Arduino almacena el sketch. Un *sketch* es el nombre que usa Arduino para un programa. Es la unidad de código que se sube y ejecuta en la placa Arduino. Esta memoria es no volátil, si Arduino deja de ser alimentado eléctricamente los datos que haya en esta memoria permanecerán.

El tamaño de la memoria Flash de Arduino puede variar dependiendo del microcontrolador, aunque no es muy grande. Por ejemplo, para el chip ATmega168 el tamaño de la memoria Flash es de 16 kilobytes, de los cuales 2 kilobytes son utilizados por el bootloader. Para el caso del microcontrolador ATmega328 (como el que incorpora Arduino UNO) el tamaño de la memoria Flash es de 32KB, de los cuales el bootloader usa 0,5KB. Por lo que debemos desarrollar los programas de forma muy optimizada, usando los tipos de variables que menos memoria requieran, en la medida de lo posible. También debemos optimizar el código fuente de la aplicación para no repetir líneas de código.

Memoria SRAM (Static Random Access Memory ó memoria estática de acceso aleatorio) en Arduino

La memoria **SRAM** (Static Random Access Memory ó memoria estática de acceso aleatorio) es de tipo volátil, es el espacio donde los sketches (programas) almacenan y manipulan variables al ejecutarse. La información guardada en esta memoria será eliminada cuando Arduino pierda la alimentación. Esta memoria es de uso exclusivo para el programa en ejecución.

La memoria SRAM de Arduino es muy pequeña, por lo que debemos optimizar nuestros programas al máximo y no abusar de variables de tipo *char*muy grandes. Hay que tener en cuenta que cada carácter de una variable *char* utiliza un byte. En el microcontrolador ATmega 168 el tamaño de la memoria SRAM es de 1024 bytes, para el caso de un chip ATmega328 (como el que incorpora Arduino UNO) el tamaño es de 2KB (2048 bytes).

Si la SRAM se queda sin espacio, el programa de Arduino fallará de forma imprevista, aunque se compile y se suba a Arduino correctamente la aplicación no se ejecutará o se ejecutara de manera extraña.

A continuación mostramos algunos consejos para optimizar los programas y evitar que consuman toda la memoria SRAM disponible:

- Si el programa se comunica con una aplicación ejecutándose en un ordenador, se puede intentar trasladar los datos o cálculos al ordenador, reduciendo la carga en el Arduino.

- Si el programa usa tablas de referencia u otros arreglos de gran tamaño, es recomendable utilizar el tipo de datos más pequeño que se pueda para almacenar estos datos; por ejemplo, un *int* utliza 2 bytes, mientras que un *byte* utiliza solo 1 byte (pero puede almacenar un rango menor de datos).
- Si no se necesita modificar las cadenas o datos mientras el programa se ejecuta, se pueden almacenar en la memoria Flah (de programa) en vez de la SRAM; para esto, hay que utilizar el keyword *PROGMEM*.

Memoria EEPROM en Arduino

EEPROM es un espacio de memoria que puede ser utilizado por los programadores para almacenar información a largo plazo. Este tipo de memoria es no volátil, por lo que los datos guardados en ella permanecerán aunque Arduino pierda la alimentación. Esta memoria puede ser usada para guardar valores si es necesario. Más adelante explicaremos cómo guardar y leer valores de esta memoria.

El tamaño de la EEPROM para un chip ATmega128 es de 512 bytes, para un chip ATmega328 es de 1KB (1024 bytes). Hay que tener en cuenta que el tamaño de la memoria EEPROM interna de Arduino es "pequeño" pero Arduino admite añadir módulos de memoria EEPROM externa de mayor tamaño.

Escribir y leer en la memoria EEPROM de Arduino

A continuación mostramos un ejemplo de escritura y otro de lectura en la memoria EEPROM interna de Arduino UNO. Para el ejemplo necesitaremos conectar Arduino al PC y disponer del IDE de desarrollo de Arduino, todo ello lo explicamos en el siguiente artículo paso a paso:

Primer proyecto hardware con Arduino, encender un LED

Escribir en la memoria EEPROM de Arduino

A continuación mostramos un sencillo ejemplo para escribir 1024 valores en la memoria EEPROM de Arduino UNO:

```
#include

void setup()
{
  //usamos un bucle que se ejecutará 1024 veces
  //en la posición i de la memoria EEPROM
  //guardaremos el valor de i
  for (int i = 0; i < 1024; i++)
    if (i <= 255)
    {
      EEPROM.write(i, i);
    }
    else
    {
      EEPROM.write(i, i - 255);
    }
}

void loop()
{
}
```

El ejemplo anterior se ejecutará una sola vez (no hemos usado "loop"). El bucle *for* se ejecutará 1024 veces y guardará en cada posición de la memoria el valor actual de *i*, teniendo en cuenta que el valor máximo que se puede guardar en una posición de memoria es de 255, por ello cuando llegamos a 255 guardamos el valor de *i* menos 255.

Leer valores de la memoria EEPROM de Arduino

A continuación mostramos un sencillo ejemplo que lee y envía por el puerto serie de Arduino todos los valores guardados en la memoria EEPROM (de un Arduino UNO, que tiene 1024 valores):

```
#include

int posicionActual = 0;
int valorLeido;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  //obtenemos el valor de la posición "posicionActual" de la EEPROM
  valorLeido = EEPROM.read(posicionActual);

  //enviamos por el puerto serie la posición leída
  Serial.print(posicionActual);
  Serial.print("       ");
  //enviamos por el puerto serie el valor leído de la posición
  Serial.print(valorLeido);
  Serial.println();

  //incrementamos la posición actual, puesto que el programa se
  //ejecuta indefinidamente leeremos todas las posiciones de memoria EEPROM
  posicionActual = posicionActual + 1;

  //puesto que el programa se ejecuta indefinidamente
```

```
  //para evitar que dé error cuando nos excedamos de la última posición
  //de la EEPROM, cuando lleguemos al máximo 1024 empezaremos de nuevo
  if (posicionActual == 1024)
    posicionActual = 0;

  //esperamos un segundo
  delay(1000);
}
```

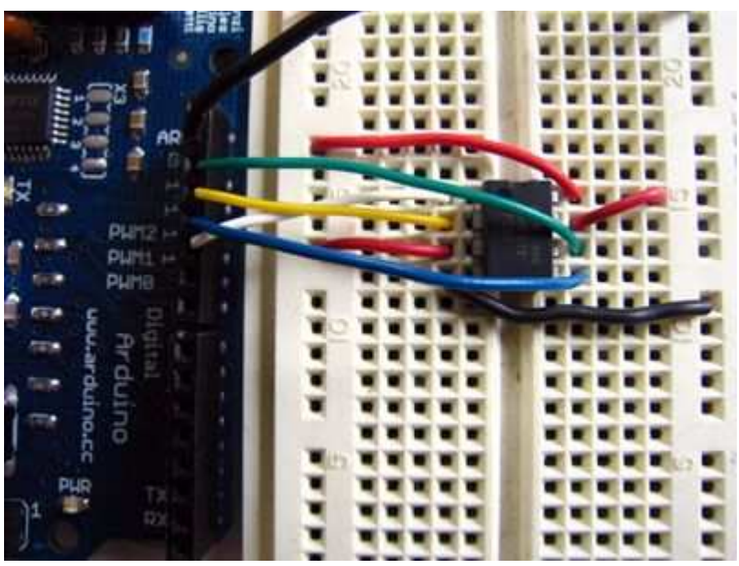
En ambos casos usamos la librería *EEPROM.h*, necesaria para acceso a la memoria EEPROM de Arduino.

Cómo ampliar las posibilidades de almacenamiento de Arduino

Cómo hemos indicado, las diferentes memorias de los chips de Arduino (Flash, SRAM y EEPROM) son demasiado limitadas para determinadas funcionalidades. Por ejemplo, si conectamos una cámara de fotos o de vídeo a Arduino y queremos guardar capturas de fotos o vídeos no nos servirán estas memorias.

Ampliar con memoria EEPROM externa

Una posibilidad para ampliar la capacidad de memoria de Arduino es usar memoria EEPROM externa, adquiriendo este tipo de memoria (que no es muy cara) podremos disponer de más memoria EEPROM para nuestro programa. Este tipo de memoria es no volátil. Por ejemplo, una EEPROM 24LC64 puede costar unos 4 euros y tiene 64K de tamaño:



Obviamente, este tipo de memorias EEPROM no tienen un gran tamaño, por ello, no serán útiles para proyectos con grandes requerimientos de espacio.

Por supuesto existen librerías para trabajar con estas tarjetas en Arduino.

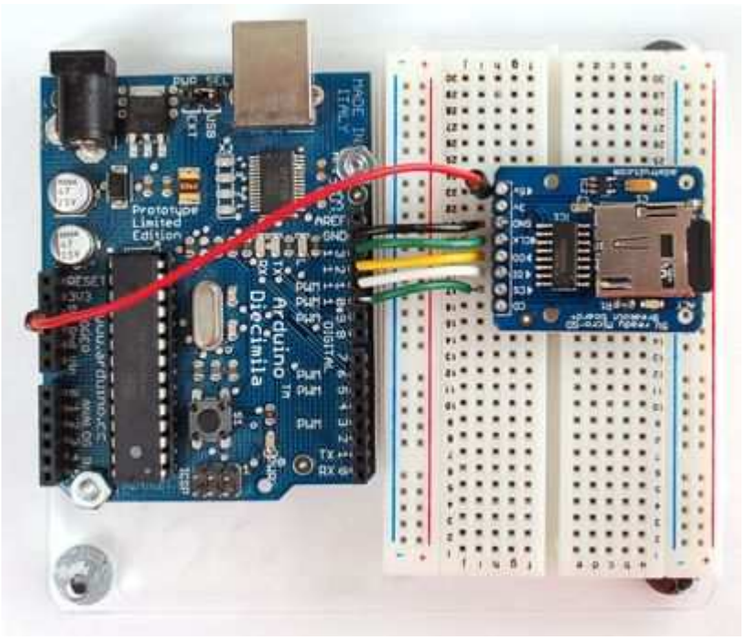
Ampliar memoria con SD Card

Si vamos a desarrollar un proyecto hardware con Arduino que tenga grandes requerimientos de espacio (Megas, Gigas), por ejemplo si queremos trabajar con audio, vídeo, fotos, almacenamiento de datos, etc., la mejor opción es utilizar algún tipo de medio de almacenamiento removible. La mayoría de los microcontroladores tienen una capacidad de memoria interna extremadamente limitada, como ya hemos comentado.

Para conseguir este almacenamiento extra utilizaremos las tarjetas flash (SD, microSD). Estás tarjetas permiten almacenar varios GBytes de datos en un espacio muy reducido.

En la imagen mostramos un módulo SD Card para Arduino, acoplable directamente:

En este otro ejemplo mostramos una SD Card para Arduino que tenemos que conectar manualmente, como se muestra en la imagen:



Por supuesto existen librerías para trabajar con estas tarjetas en Arduino.

Fuente:

http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=571