MEMORIA EEPROM DE ARDUINO

La memoria EEPROM tiene sus propias características y peculiaridades que la distinguen del resto de las memorias, pero lo más importante por ahora para nosotros es que: este tipo de memoria es **NO VOLÁTIL**, es decir, mantiene los valores almacenados cuando se pierde la alimentación de nuestro dispositivo (como un pequeño disco duro). Pero antes de comenzar estudiarla, debemos saber que Arduino tiene tres tipos de memoria:

- **FLASH**, memoria no volátil, donde grabamos el sketch (incluido el bootloader).
- **SRAM** (static random access memory), Memoria volátil, donde se almacenan las variables que usamos en nuestros programas.
- **EEPROM**, no volátil, que podemos usar para almacenar información entre reinicios.

Esta memoria será muy útil, cuando necesitamos que ciertos valores se conserven y podamos encontrarlos cada vez que inicie nuestro programa, y además, estos datos puedan ser actualizados por nuestro programa antes de apagarse. Por ejemplo, valores de calibración, mediciones, fechas y horas, que podemos usar en dataloggers (Archivos Logs), guardar un contador, o saber cuál era el estado de nuestros cálculos o del procesador cuando se perdió la alimentación, etc.

Características de la Memoria EEPROM

Veamos ahora las principales características de la memoria EEPROM.

- En primer lugar, como ya habíamos mencionado es no volátil, es decir, mantiene los valores almacenados cuando se pierde la alimentación.
- Por otro lado, la memoria EEPROM es un recurso bastante escaso en comparación del resto de las memorias. La mayoría de modelos de Arduino disponen de 1KB, mientras que el modelo Mega tiene 4KB.
- La memoria EEPROM es mucho más lenta que la memoria SRAM. El proceso de escritura de una celda (byte) puede llevar alrededor a 3.3 ms. Sin embrago el proceso de lectura es mucho más rápido (aunque sigue siendo más lento que la SRAM), Por ejemplo leer 1024 bytes demora alrededor de 0.3ms, es decir, 10.000 veces más rápida que la escritura.
- Una cosa muy importante que debemos tener en cuenta es que la memoria EEPROM tiene una vida útil limitada, se reduce con cada operación de escritura. Pero no existen límites para las operaciones de lectura. Las especificaciones garantizan que cada celda tiene una vida útil de al menos 100.000 grabaciones. Aunque en la práctica puede llegar a ser muy superior, hasta un millón de operaciones, pero por encima de 100.000 grabaciones el funcionamiento no está garantizado.

100.000 operaciones de Grabado parecen muchas, pero hay que pensar que:

Si, por error, grabamos un byte constantemente	Ese byte será útil por 5 minutos Aprox.
Si lo grabamos una vez por segundo	Será útil por un día aproximadamente.
Si Grabamos 10 veces al día	Alrededor de 27 Años

Es decir, la memoria EEPROM está pensada para ser utilizada muy pocas veces, es decir, realizar escrituras con tiempos largos entre ellas

Pero No te preocupes, ya que un Modulo adicional Memoria EEPROM para Arduino es muy Barato y es fácil de encontrar Arduino.

- Si bien puedes usar una lectora/Grabadora de Tarjetas SD, la memoria EEPROM es mucho más rápida

- Para usar la memoria EEPROM de nuestro Arduino, disponemos de la librería EEPROM.h que se encuentra en el IDE. No hace falta buscarla o actualizarla, ya que nuestro IDE se encarga de las actualizaciones periódicas.

Funciones de la Librería EEPROM.h

(Ver más)

El IDE Standard de Arduino ya incorpora esta librería, que contiene las funciones necesarias para manipular la memoria no volátil de Arduino.

FUNCIONES contenidas en la Librería	
length()	Informa la cantidad de memoria EEPROM disponible que tiene nuestro Arduino. Esto es independiente del modelo. Sabiendo esto, nuestro programa puede adaptarse a la placa en que este corriendo y saber en todo momento hasta donde puede usar esta memoria. Ver el siguiente ejemplo: `int Cantidad = EEPROM.length();` En la Variable "**Cantidad**" quedará almacenado el tamaño en Bytes (cantidad) de la memoria EEPROM de la placa Arduino en donde se haya ejecutado ese comando.
end()	Informa la cantidad de memoria EEPROM disponible que tiene nuestro Arduino. Esto es independiente del modelo (Igual que la función "**length()**"). Sabiendo esto, nuestro programa puede adaptarse a la placa en que este corriendo y saber en todo momento hasta donde puede usar esta memoria. Ver el siguiente ejemplo: `int Cantidad = EEPROM.end();` En la Variable "**Cantidad**" quedará almacenado el tamaño en Bytes (cantidad) de la memoria EEPROM de la placa Arduino en donde se haya ejecutado ese comando.
Las funciones más simples son: la función "**Write()**" y "**Read()**" que, respectivamente, escriben y leen un byte en una dirección de la memoria. Para cuando necesitemos guardar variables que tienen un tamaño superior a un byte, disponemos de las funciones Put, Get, que son las que usaremos con mayor frecuencia	Recordar que un byte = 8 bit, lo que nos permitirá contener valores hasta 2^8 es decir entre 0 y 255 (ambos incluidos).
write()	Escribe en dirección indicada un único byte. **EEPROM.write(Dirección , Dato)** **Dirección:** indica la posición de memoria, con valores entre 0 y N-1 (ambos incluidos), siendo N el número de bytes disponibles en la memoria EEPROM (ejemplo, 0 a 1023 en Arduino Uno y Nano, 0 a 4095 en Arduino Mega). **Dato:** Es el valor que grabaremos en la memoria, y este dato `EEPROM.write( a , "5" )` Para el caso del ejemplo, escribirá en la dirección indicada por "a" el caractér 5. Recordar que "a" debe tener un valor entre 0 a N-1, siendo N el número de bytes disponibles de memoria EEPROM. El complemente ideal de esta función es la función **update().**
read()	Lee un único byte de la dirección indicada. `int Valor = EEPROM.read( a )`

	Para el caso del ejemplo, leerá un byte de la dirección indicado por la variable "**a**". Recordar que "a" debe tener un valor entre 0 a N-1, siendo N el número de bytes disponibles de memoria EEPROM.	
update()	Escribe un byte en la EEPROM. El valor se escribe solo si es diferente del que ya se encuentra en la misma dirección. EEPROM.update(Dirección , Valor) **Dirección:** Debe tener un valor entre 0 a N-1, siendo N el número de bytes disponibles de memoria EEPROM. **Valor:** Tener en cuenta que el valor que se puede escribir, oscila entre 0 y 255 (byte). **Recordar que:** La memoria EEPROM tiene una vida especificada de 100.000 ciclos de escritura / borrado, por lo que el uso de esta función en lugar de "**write()**", prolongando significativamente la vida útil de la memoria, si los datos escritos no cambian con frecuencia. Esta función **NO SE ADAPTA** al tamaño de la variable como lo hacen las funciones **put** y **get**.	

EEPROM[]	Este operador permite utilizar el identificador 'EEPROM' como un arreglo (vector).
	
	Las posiciones o células de la memoria EEPROM se pueden leer, escribir y comparar directamente usando este método.
		#include <EEPROM.h>
		// Ver como se puede usar este operador
		void setup(){
		unsigned char val;
		//Guardamos o escribimos el primer byte.
		EEPROM[0] = val;
		//Leemos el primer byte .
		val = EEPROM[0];
		//Comparamos contenidos
		if(val == EEPROM[0]){ /* ACCIONES */ }
		}
		void loop(){ /* loop Vacío*/ }

Estas funciones graban o leen un único byte, es decir solo valores de **0** a **255**.		
Las funciones Put, Get tienen en cuenta el tamaño de la variable, y funcionan perfectamente con cualquier variable incluso del tipo de estructuras definidas por el usuario. Sin embargo, tendremos que tener en cuenta el tamaño de la variable para saber cuál es la siguiente dirección a escribir, y evitar que se "sobre escriban" las variables. Recomiendo repasar el uso de la función "**sizeof(Variable)**"		
get()	Lee cualquier tipo de datos incluso estructuras de datos en la memoria EEPROM, es decir, leerá el datos sin importar su tipo (int, float, char, struct, etc.) . EEPROM.get(Dirección, Dato) **Dirección:** Hace referencia a la posición (nro de byte) a partir del cual deberá grabarse el dato. **Dato:** Es el dato propiamente dicho o variable que lo contiene.	
put()	Escriba cualquier tipo de datos incluso estructuras de datos en la memoria EEPROM, es decir, escribirá el datos sin importar su tipo (int, float, char, struct, etc) . EEPROM.put(Dirección, Dato) **Dirección:** Hace referencia a la posición (nro de byte) a partir del cual deberá grabarse el dato. **Dato:** Es el dato propiamente dicho o variable que lo contiene.	

A Programar.!!

A continuación, comencemos con la programación. Primero ejemplos muy simples para comprender bien el funcionamiento de cada función, luego algunos ejemplos en los que verdaderamente le demos una aplicación a la memoria EEPROM.

1) Informar por el Puerto Serie El Tamaño en bytes de la Memoria EEPROM de la placa Arduino en la que este corriendo el programa.

Como primer ejemplo, informar por el monitor del puerto serie, el tamaño que tiene la memoria EEPROM de esta Placa. Sabiendo esto, nuestro programa puede adaptarse a la placa en que este corriendo y saber en todo momento hasta donde puede usar esta memoria.

(Programa "875_EEPROM_01_Ver_Tamanio")

```
1  #include <EEPROM.h>  // Esta Librería ya esta Incluida en el IDE de Arduino
2
3  void setup(){
4    int Tam;
5    Serial.begin(9600);
6    while (!Serial); // Esperamos que el puerto serie este abierto.
7
8    Serial.println("Iniciando Programa '875_EEPROM_01_Ver_Tamanio'...");
9
10   Tam = EEPROM.length();
11   Serial.print("\nEsta Placa Arduino Dispone de ");
12   Serial.print( Tam );
13   Serial.println(" bytes de Memoria EEPROM");
14 }
15
16 void loop(){
17   /* Vacío - Para que no repita */
18 }
```

Recordar que un byte = 8 bit, lo que nos permitirá contener valores hasta 2^8 es decir entre 0 y 255 (ambos incluidos).

Recuerda que solo estamos trabajando con la placa Arduino, no serán necesarias conexiones adicionales.

2) Escribir con un número las primeras 10 posiciones de la memoria EEPROM de nuestra placa Arduino.

En este ejercicio, grabaremos números correlativos en los primeros 10 bytes (posiciones) de nuestra placa Arduino, luego en el próximo ejercicio los leeremos y mostraremos por el monitor del puerto serie .

(Programa "875_EEPROM_02_Escribe_01_A")

```
1  // Esta Librería ya esta Incluida en el IDE de Arduino
2  #include <EEPROM.h>
3
4  void setup(){
5     Serial.begin(9600);
6  // Esperamos que el puerto serie este abierto.
7     while (!Serial);
8
9     Serial.println("Iniciando Programa\n");
10
11    for (int i = 0; i < 10; i++){
12 /* Formato  "EEPROM.write(Dirección , Dato)" */
13       EEPROM.write(i, i);
14
15       Serial.print("Escribio en el Byte ");
16       Serial.print( i );
17       Serial.print(" el dato ");
18       Serial.println( i );
```

(Programa "875_EEPROM_02_Escribe_01_B")

```
// Esta Librería ya esta Incluida en el IDE de Arduino
#include <EEPROM.h>

void setup(){
  Serial.begin(9600);
// Esperamos que el puerto serie este abierto.
  while (!Serial);

  Serial.println("Iniciando Programa\n");

  for (int x = 0; x < 15; x++){

    EEPROM[ x ] = x;

    Serial.print("Escribio en el Byte ");
    Serial.print( x );
    Serial.print(" el dato ");
    Serial.println( x );
```

```
18      }                                   }
20  }                                       }
21
22  void loop(){                            void loop(){
23    /* Vacío - Para que no repita */        /* Vacío - Para que no repita */
24  }                                       }
```

Acá se muestran dos formas de grabar cada una de las posiciones (bytes) de la memoria EEPROM. Usted podrá usar la que le resulte más simple.

-o-

Al ejecutar cualquiera de los dos programas, por el monitor del puerto serie podremos ver: ==========>

```
Iniciando Programa ' 875_EEPROM_02_Escribe_01 '...

Escribió en el Byte 0 el dato 0
Escribió en el Byte 1 el dato 1
Escribió en el Byte 2 el dato 2
Escribió en el Byte 3 el dato 3
Escribió en el Byte 4 el dato 4
Escribió en el Byte 5 el dato 5
Escribió en el Byte 6 el dato 6
Escribió en el Byte 7 el dato 7
Escribió en el Byte 8 el dato 8
Escribió en el Byte 9 el dato 9
```

Pero lo más importante es lo que realmente se graba en la EEPROM de Arduino.

Recordar que un byte = 8 bit, lo que nos permitirá contener valores hasta 2^8 es decir entre 0 y 255 (ambos incluidos).

Lo realmente Grabado en la Memoria EEPROM de Arduino lo podremos ver en el próximo programa.

3) Leer el contenido de los primeros 15 bytes (posiciones) de la memoria EEPROM de nuestra placa. Los valores que leeremos son los grabados en el ejercicio anterior.

En este ejercicio, leeremos y mostraremos por el monitor del puerto serie, los valores que grabamos en el ejercicio anterior (hasta posición 10), pero leeremos y mostraremos hasta la posición 15, para ver que hay en los lugares donde no había nada grabado.

```
    (Programa "875_EEPROM_03_Lee_01_A")                (Programa "875_EEPROM_03_Lee_01_B")
1   // Esta Librería ya esta Incluida en el IDE de Arduino  // Esta Librería ya esta Incluida en el IDE de Arduino
2   #include <EEPROM.h>                                     #include <EEPROM.h>
3
4   void setup(){                                          void setup(){
5     int Dato;                                              int Dato;
6     Serial.begin(9600);                                    Serial.begin(9600);
7   // Esperamos que el puerto serie este abierto.         // Esperamos que el puerto serie este abierto.
8     while (!Serial);                                       while (!Serial);
9
10    Serial.println("Iniciando Programa\n");                 Serial.println("Iniciando Programa\n");
11
12    for (int i = 0; i < 15; i++){                          for (int x = 0; x < 65; x++){
13      Dato = EEPROM.read( i );                               Dato = EEPROM[ x ];
14
15      Serial.print("Se lee en la posicion ");              Serial.print("Se lee en la posicion ");
16      Serial.print( i );                                     Serial.print( x );
17      Serial.print(" el dato ");                             Serial.print(" el dato ");
18      Serial.println( Dato );                                Serial.println( Dato );
19    }                                                      }
20  }                                                      }
21
22  void loop(){                                           void loop(){
23    /* Vacio - Para que no repita */                       /* Vacio - Para que no repita */
24  }                                                      }
```

Acá se muestran dos formas de Leer o acceder a cada una de las posiciones (bytes) de la memoria EEPROM. Usted podrá usar la que le resulte mas simple.

-o-

Al ejecutar cualquiera de los dos programas, por el monitor del puerto serie podremos ver: ==========>

```
Iniciando Programa '875_EEPROM_03_Lee_01'...

Se lee en la posición 0 el dato 0
Se lee en la posición 1 el dato 1
Se lee en la posición 2 el dato 2
```

Recordar que un byte = 8 bit, lo que nos permitirá contener

valores hasta 2^8 es decir entre 0 y 255 (ambos incluidos).

En el Ejercicio anterior grabamos números correlativos desde el 0 al 9, y ahora encontramos que las posiciones en donde no se había grabado nada, hay un 255. Este valor es con el que viene grabada la memoria EEPROM de nuestro Arduino de fábrica.

Por demás esta recordar que la memoria EEPROM se comporta como un mini Disco Duro, y mantendrá esos valores grabados en esas posiciones hasta que grabemos otro valor.

```
Se lee en la posición 3 el dato 3
Se lee en la posición 4 el dato 4
Se lee en la posición 5 el dato 5
Se lee en la posición 6 el dato 6
Se lee en la posición 7 el dato 7
Se lee en la posición 8 el dato 8
Se lee en la posición 9 el dato 9
Se lee en la posición 10 el dato 255
Se lee en la posición 11 el dato 255
Se lee en la posición 12 el dato 255
Se lee en la posición 13 el dato 255
Se lee en la posición 14 el dato 255
Se lee en la posición 15 el dato 255
```

4) Limpiar o borrar todo el contenido de la Memoria EEPROM.

A continuación se limpiará toda la memoria EEPROM de la placa Arduino que estemos usando. Este proceso será realizado por tres formas distintas, de las cuales se recomienda no usar los programas A y B, ya que al borrar la memoria, estarían **gastando** los lugares que no sea necesario borrar, usando una escritura. Por otro lado, el tercer programa, usa la función "**EEPROM.update()**" que solo borrara (escribiendo) las posiciones de memoria que no estén en cero.

(Programa "875_EEPROM_04_Borra_EEPROM_A")

```
1  /* Escribe todos los bytes de EEPROM en 0 */
2  #include <EEPROM.h>
3
4  void setup(){
5    int x;
6    Serial.begin(9600);
7  // Esperamos que el puerto serie este abierto.
8    while (!Serial);
9
10   Serial.println("Iniciando Programa A'...");
11
12 /* Recorre toda la EEPROM y reemplaza todo
13      valor existente por cero */
14   for (int x = 0; x < EEPROM.length(); x++){
15     EEPROM.write(x, 0);
16   }
17 }
18
19 void loop(){
20   /* Dacio - Para que no repita */
21 }
```

(Programa "875_EEPROM_04_Borra_EEPROM_B")

```
1  /* Escribe todos los bytes de EEPROM en 0 */
2  #include <EEPROM.h>
3
4  void setup(){
5    int x;
6    Serial.begin(9600);
7  // Esperamos que el puerto serie este abierto.
8    while (!Serial);
9
10   Serial.println("Iniciando Programa B'...");
11
12 /* Recorre toda la EEPROM y reemplaza todo
13      valor existente por cero */
14   for (int x = 0; x < EEPROM.length(); x++){
15     EEPROM[x]=0;
16   }
17 }
18
19 void loop(){
20   /* Vacío - Para que no repita */
21 }
```

(Programa "875_EEPROM_04_Borra_EEPROM_C")

```
1  /* EEPROM.update() - Solo Cambia Distintos de 0 */
2  #include <EEPROM.h>
3
4  void setup(){
5    int x;
6    Serial.begin(9600);
7  // Esperamos que el puerto serie este abierto.
8    while (!Serial);
9
10   Serial.println("Iniciando Programa C'...");
11
12 /* Recorre la EEPROM y reemplaza Solamente
13      Valores Distintos de cero por un cero */
14   for (int x = 0; x < EEPROM.length(); x++){
15     EEPROM.update(x, 0);
16   }
17 }
18
```

Se presentaron tres versiones de un mismo programa, y en los tres solo cambia la línea 15, en la que usamos las diferentes funciones que nos permitirán hacer la misma tarea.

Sin embargo, se recomienda no usar los programas A y B, ya que al borrar la memoria, estarían **gastando** los lugares que no es necesario borrar, usando una escritura.

Por otro lado, el tercer programa es el recomendado, ya que se usa la función "**EEPROM.update()**", que solo borrara

```
19  void loop(){
20    /* Vacío - Para que no repita */
21  }
```

(escribiendo) las posiciones de memoria que no estén en cero, respetando el resto.

5) Mostrar por el monitor del Puerto Serie el tamaño (en bytes) que ocupan los diversos tipos de variables en la memoria.

El propósito de este ejercicio es mostrar como se puede obtener el tamaño que ocupan en memoria (en bytes) los distintos tipos de datos, primer ejemplo. Sin embargo también funcionara con variabas y esto lo veremos en el ejemplo 2.

(Programa "'000_Tipos_de_Datos_01")

```
1   void setup() {
2     Serial.begin(9600);
3     while (!Serial) ; // Esperamos abra puerto serie.
4
5   }
6
7   void loop() {
8     Serial.print("sizeof(byte)= ");
9     Serial.println(sizeof(byte));
10    Serial.println();
11
12    Serial.print("sizeof(char)= ");
13    Serial.println(sizeof(char));
14    Serial.println();
15
16    Serial.print("sizeof(short)= ");
17    Serial.println(sizeof(short));
18    Serial.println();
19
20    Serial.print("sizeof(int)= ");
21    Serial.println(sizeof(int));
22    Serial.println();
23
24    Serial.print("sizeof(long)= ");
25    Serial.println(sizeof(long));
26    Serial.println();
27
28    Serial.print("sizeof(long long)= ");
29    Serial.println(sizeof(long long));
30    Serial.println();
31
32    Serial.print("sizeof(bool)= ");
33    Serial.println(sizeof(bool));
34    Serial.println();
35
36    Serial.print("sizeof(boolean)= ");
37    Serial.println(sizeof(boolean));
38    Serial.println();
39
40    Serial.print("sizeof(float)= ");
41    Serial.println(sizeof(float));
42    Serial.println();
43
44    Serial.print("sizeof(double)= ");
45    Serial.println(sizeof(double));
46    Serial.println();
47
48    Serial.print("sizeof(int8_t)= ");
49    Serial.println(sizeof(int8_t));
50    Serial.println();
51
55  // Continua ===>
56
```

```
    Serial.print("sizeof(int16_t)= ");
    Serial.println(sizeof(int16_t));
    Serial.println();

    Serial.print("sizeof(int32_t)= ");
    Serial.println(sizeof(int32_t));
    Serial.println();

    Serial.print("sizeof(int64_t)= ");
    Serial.println(sizeof(int64_t));
    Serial.println();

    Serial.print("sizeof(uint8_t)= ");
    Serial.println(sizeof(uint8_t));
    Serial.println();

    Serial.print("sizeof(uint16_t)= ");
    Serial.println(sizeof(uint16_t));
    Serial.println();

    Serial.print("sizeof(uint32_t)= ");
    Serial.println(sizeof(uint32_t));
    Serial.println();

    Serial.print("sizeof(uint64_t)= ");
    Serial.println(sizeof(uint64_t));
    Serial.println();

    Serial.print("sizeof(char*)= ");
    Serial.println(sizeof(char*));
    Serial.println();

    Serial.print("sizeof(int*)= ");
    Serial.println(sizeof(int*));
    Serial.println();

    Serial.print("sizeof(long*)= ");
    Serial.println(sizeof(long*));
    Serial.println();

    Serial.print("sizeof(float*)= ");
    Serial.println(sizeof(float*));
    Serial.println();

    Serial.print("sizeof(double*)= ");
    Serial.println(sizeof(double*));
    Serial.println();

    Serial.print("sizeof(void*)= ");
    Serial.println(sizeof(void*));

    while (1);
}
```

-o-

(Programa "'000_Tipos_de_Datos_02")

```
1  void setup() {
2    Serial.begin(9600);
3    while (!Serial) {
4      ; // Esperamos que el puerto serie este abierto.
5    }
6  }
7    byte      Var_byte;
8    char      Var_char;
9    short     Var_short;
10   int       Var_int;
11   long      Var_long;
12   long long Var_long_long;
13   bool      Var_bool;
14   boolean   Var_boolean;
15   float     Var_float;
16   double    Var_double;
17   int8_t    Var_int8_t;
18   int16_t   Var_int16_t;
19   int32_t   Var_int32_t;
20   int64_t   Var_int64_t;
21   uint8_t   Var_uint8_t;
22   uint16_t  Var_uint16_t;
23   uint32_t  Var_uint32_t;
24   uint64_t  Var_uint64_t;
25
26   char*     Var_P_char;
27   int*      Var_P_int;
28   long*     Var_P_long;
29   float*    Var_P_float;
30   double*   Var_P_double;
31
32 void loop() {
33   Serial.print("sizeof(Var_byte)= ");
34   Serial.println(sizeof(Var_byte));
35   Serial.println();
36
37   Serial.print("sizeof(Var_char)= ");
38   Serial.println(sizeof(Var_char));
39   Serial.println();
40
41   Serial.print("sizeof(Var_short)= ");
42   Serial.println(sizeof(Var_short));
43   Serial.println();
44
45   Serial.print("sizeof(Var_int)= ");
46   Serial.println(sizeof(Var_int));
47   Serial.println();
48
49   Serial.print("sizeof(Var_long)= ");
50   Serial.println(sizeof(Var_long));
51   Serial.println();
52
53   Serial.print("sizeof(Var_bool)= ");
54   Serial.println(sizeof(Var_bool));
55   Serial.println();
56
57   Serial.print("sizeof(Var_boolean)= ");
58   Serial.println(sizeof(Var_boolean));
59   Serial.println();
60
61
   Serial.print("sizeof(Var_float)= ");
   Serial.println(sizeof(Var_float));
   Serial.println();

   Serial.print("sizeof(Var_double)= ");
   Serial.println(sizeof(Var_double));
   Serial.println();

   Serial.print("sizeof(Var_int8_t)= ");
   Serial.println(sizeof(Var_int8_t));
   Serial.println();

   Serial.print("sizeof(Var_int16_t)= ");
   Serial.println(sizeof(Var_int16_t));
   Serial.println();

   Serial.print("sizeof(Var_int32_t)= ");
   Serial.println(sizeof(Var_int32_t));
   Serial.println();

   Serial.print("sizeof(Var_int64_t)= ");
   Serial.println(sizeof(Var_int64_t));
   Serial.println();

   Serial.print("sizeof(Var_uint8_t)= ");
   Serial.println(sizeof(Var_uint8_t));
   Serial.println();

   Serial.print("sizeof(Var_uint16_t)= ");
   Serial.println(sizeof(Var_uint16_t));
   Serial.println();

   Serial.print("sizeof(Var_uint32_t)= ");
   Serial.println(sizeof(Var_uint32_t));
   Serial.println();

   Serial.print("sizeof(Var_uint64_t)= ");
   Serial.println(sizeof(Var_uint64_t));
   Serial.println();

   Serial.print("sizeof(Var_P_char)= ");
   Serial.println(sizeof(Var_P_char));
   Serial.println();

   Serial.print("sizeof(Var_P_int)= ");
   Serial.println(sizeof(Var_P_int));
   Serial.println();

   Serial.print("sizeof(Var_P_long)= ");
   Serial.println(sizeof(Var_P_long));
   Serial.println();

   Serial.print("sizeof(Var_P_float)= ");
   Serial.println(sizeof(Var_P_float));
   Serial.println();

   Serial.print("sizeof(Var_P_double)= ");
   Serial.println(sizeof(Var_P_double));

   while (1);
}
```

6) Escribir en la memoria EEPROM Datos de distintos tipos y luego leerlos usando otras variables (distintas).

El propósito de este ejercicio es mostrar como las funciones **put** y **get** se adaptan al tamaño de las variables o datos de deben usar. Recuerde que esto mismo podría hacerse usando la función "**EEPROM.update()**" SOLO SI los valores que se escriben oscilan entre 0 y 255 y nunca con valores del tipo **struct**.

(Programa "'875_EEPROM_05_Put_Get_01")

```
1   #include <EEPROM.h>
2
3   struct Compuesto{
4     float Dato_float;
5     char  Dato_char[15];
6   };
7
8   int    Dato_int      = 7531;
9   float  Dato_float    = 4321.1234f;
10  char   Dato_char[20]  = {"Esta es una Cadena"};
11  String Dato_String    = {"Esto es una Cadena String"};
12  Compuesto Dato_struct = { 3.1415f, "Aprendiendo!" };
13
14  void setup(){
15    int Posicion;
16    Serial.begin(9600);
17    while (!Serial); // Esperamos que puerto serie este abierto.
18
19    Serial.println("Iniciando Programa '875_EEPROM_05_Put_Get_01'...");
20
21    Posicion = 0;
22    EEPROM.put( Posicion, Dato_int);
23
24    Posicion += sizeof(Dato_int);
25    EEPROM.put( Posicion, Dato_float);
26
27    Posicion += sizeof(Dato_float);
28    EEPROM.put( Posicion, Dato_char);
29
30    Posicion += sizeof(Dato_char);
31    EEPROM.put( Posicion, Dato_String);
32
33    Posicion += sizeof(Dato_String);
34    EEPROM.put( Posicion, Dato_struct);
35
36  /* Declaramos nuevas Variables, solo para que lo leído se guarde en
37     variables distintas aportando claridad en lo que se hace                */
38  int    L_Dato_int      = 0;
39  float  L_Dato_float    = 0.0f;
40  char   L_Dato_char[20]  = {""};
41  String L_Dato_String    = {""};
42  Compuesto L_Dato_struct = { 0.0f, "" };
43
44    Posicion = 0;  // Comienzo Grabando el primer dato en Posición cero
45    EEPROM.get( Posicion, L_Dato_int);
46    Serial.println( String("L_Dato_int= ") + L_Dato_int);
47
48    Posicion += sizeof(L_Dato_int); // Sumo el tamaño de la variable anterior
49    EEPROM.get( Posicion, L_Dato_float);
50    Serial.println( String("L_Dato_float= ") + L_Dato_float);
51
52    Posicion += sizeof(L_Dato_float); // Sumo el tamaño de la variable anterior
53    EEPROM.get( Posicion, L_Dato_char);
54    Serial.println( String("L_Dato_char= ") + L_Dato_char);
55
56    Posicion += sizeof(L_Dato_char); // Sumo el tamaño de la variable anterior
```

```
57    EEPROM.get( Posicion, L_Dato_String);
58    Serial.println( String("L_Dato_String= ") + L_Dato_String);
59
60    Posicion += sizeof(L_Dato_String); // Sumo el tamaño de la variable anterior
61    EEPROM.get( Posicion, L_Dato_struct);
62    Serial.println( String("L_Dato_struct.Dato_float= ") + L_Dato_struct.Dato_float);
63    Serial.println( String("L_Dato_struct.Dato_char= ") + L_Dato_struct.Dato_char);
64  }
65
66  void loop(){
67    /* Vacío - Para que no repita */
68  }
```

-o-

Luego, al ejecutar este programa, por el monitor del puerto serie veremos: ==========>

```
Iniciando Programa '875_EEPROM_05_Put_Get_01'...
L_Dato_int= 7531
L_Dato_float= 4321.12
L_Dato_char= Esta es una Cadena
L_Dato_String= Esto es una Cadena String
L_Dato_struct.Dato_float= 3.14
L_Dato_struct.Dato_char= Aprendiendo!
```

Note que se perdieron decimales en los números Float.
Note que este podría haber sido un archivo que se lee al iniciar el proceso y configura nuestro dispositivo.

01010110 01101001 01110110 01100101 00100000 01110100 01110101 00100000 01101101
01100101 01101101 01101111 01110010 01101001 01100001 00100000 01111001 00100000
01100001 01110011 11000011 10110011 01101101 01100010 01110010 01100001 01110100
01100101 00101100 00100000 01110000 01101111 01110010 01110001 01110101 01100101
00100000 01101100 01100001 00100000 01100010 01110101 01100101 01101110 01100001
00100000 01101101 01100101 01101101 01101111 01110010 01101001 01100001 00101100
00100000 01100101 01110011 00100000 01110000 01110010 01101001 01101110 01100011
01101001 01110000 01101001 01101111 00100000 01100100 01100101 00100000 01101100
01100001 00100000 01110011 01100001 01100010 01101001 01100100 01110101 01110010
11000011 10101101 01100001 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.