



**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**Facultad de Ingeniería en Mecánica y Ciencias de la  
Producción**

## **PRACTICA #5**

Fecha: 14/06/2021

Oscar Andres Guapi Mora  
oguapi@espol.edu.ec

## EEPROM en lenguaje Arduino

```

AVR Source Code - U1
main.ino

----- #include <EEPROM.h> // libreria EEPROM
----- float temperatura = 15.15; // asigna valor a variable de punto flotante
----- char cadena[] = "Hola esta es una prueba"; // crea cadena de caracteres con un texto
----- float temp2 = 50.50;
0284 void setup() {
028C   Serial.begin(9600); // inicializa monitor serie a 9600 bps
-----
029C   EEPROM.put(0x00, temperatura); // almacena en direccion cero el punto flotante
02AC   EEPROM.put(0x0a, cadena); // almacena en direccion diez la cadena
02BC   Serial.println("Valor de punto flotante en direccion 0:"); // imprime texto
02C8   Serial.println(EEPROM.get(0, temp2)); // obtiene valor de punto flotante
----- // en direccion cero y muestra
02D2   Serial.println(" "); // espacio en blanco
02FE   Serial.println("Valor de la cadena en direccion 10:"); // imprime texto
030A   Serial.println(EEPROM.get(10, cadena)); // obtiene cadena de caracteres en
0324 } // direccion diez y muestra
032A void loop() { // funcion loop() declarada pero sin contenido
----- // nada por aqui
0332 }
----- //Andres Guap1 P101

AVR EPROM Memory - U1
AVR Variables - U1

AVR EPROM Memory - U1
0000 66 66 72 41 FF FF FF FF FF FF 48 6F 6C 61 20 65 73 74 61 20 65 73 20 75 fffa.....Hola esta es u
0018 6E 61 20 72 75 65 62 61 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..na prueba.
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
0048 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
0078 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
0090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..
00A8
00C0 Virtual Terminal
00D8
00F0 Valor de punto flotante en direccion 0:
0108 15.15
0120
0138
0150 Valor de la cadena en direccion 10:
0168 Hola esta es una prueba
0180
0198
01B8
01C8
01E0
01F8
0210
0228
0240
0258
0278
0288
02A0
02B8
02D8
02E8

```

## Memoria FLASH

AVR Source Code - U1

```
main.c
----- #include <avr/io.h>
----- #include <avr/pgmspace.h> //libreria para almacenar variables en la FLASH
----- const char texto_flash[] PROGMEM = "Este texto se encuentra en la memoria FLASH";
----- //guardamos en la memoria flash con la variable PROGMEM
-----
----- char texto_sram[];
00BC int main(){
----- uint8_t i=0; //variable tipo entero
----- do{
00BE texto_sram[i] = pgm_read_byte(&texto_flash[i]); //leo lo almacenado y asigno a la memoria de datos
00DA }while(texto_sram[i++] !=0); //hasta que no tengamos nada
----- while(1){
00E0 asm("NOP"); //Para que no se elimine el paso a paso de Proteus
----- }
----- return 0;
----- }
----- //Andres Guapi Mora P101
```

AVR Variables - U1    AVR Data Memory - U1

AVR Data Memory - U1

0100	45	73	74	65	20	74	65	78	74	6F	20	73	65	20	65	6E	63	75	65	6E	74	72	61	20	65	6E	20	6C	Este texto se encuentra en l
011C	61	20	6F	61	20	46	4C	41	53	48	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	a memoria FLASH.....
0138	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0154	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
018C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
01A8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
01C4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
01E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
01FC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0218	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0234	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
026C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

AVR Program Memory - U1

0000	0C	94	4A	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	..3...\\...\\...\\...\\...\\...\\...\\
001C	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	..\\...\\...\\...\\...\\...\\...\\
0038	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	..\\...\\...\\...\\...\\...\\...\\
0054	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	0C	94	5C	00	45	73	74	65	20	74	65	78	..\\...\\...\\...\\...\\...\\...\\
0070	74	6F	20	73	65	20	65	6E	63	75	65	6E	74	72	61	20	65	6E	20	6C	61	20	6D	65	6D	6F	72	69	...Este tex
008C	61	20	46	4C	41	53	48	00	11	24	1F	BE	CF	EF	D8	E0	DE	BF	CD	BF	11	E0	A0	E0	B1	E0	01	C0	to se encue
00A8	1D	92	A1	30	B1	07	E1	F7	0E	94	5E	00	0C	94	72	00	0C	94	00	00	20	E0	48	E6	50	E0	60	E0	a FLASH. \$
00C4	71	E0	82	2F	90	E0	FA	01	E8	0F	F9	1F	E4	91	DB	01	A8	0F	B9	1F	EC	93	2F	5F	EE	23	99	F7	...H.P.
00E0	00	00	FE	CF	F8	94	FF	CF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	Q.../...^...r.../_...#...
00FC	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
0118	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
0134	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
0150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
016C	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
0188	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
01A4	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....
01C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	.....

4 Message(s)    PAUSED: 0.065791000s

## Memoria Estática

AVR Source Code - U1

```
main.c
----- /*CODIGO ASIGNACION ESTATICA*/
----- #include "config.h"
----- #include "funciones.h"
-----
----- int main(){
00A6   srandom(SEMILLA);
----- while (1){
00B2   iniciarArreglo();
00B6   cantidad=numeroAleatorio(); //guardamos la variables
00BE   llenarNumeros(cantidad);
----- }
----- return 0;
----- }
----- //Andres Guapi Mora P101
```

AVR Variables - U1

Name	Address	Value
<b>cantidad</b>	<b>00800104</b>	<b>30</b>
<b>i</b>	<b>00800105</b>	<b>0</b>
<b>estatico</b>	<b>00800106</b>	<b>byte[100]</b>
<b>cantidad</b>	<b>00800104</b>	<b>30</b>
<b>i</b>	<b>00800105</b>	<b>0</b>
<b>estatico</b>	<b>00800106</b>	<b>byte[100]</b>

Simulation Log

AVR Data Memory - U1

Address	Value
0100	42 CE 0C 1F 00 FF FF FF FF FF FF FF FF FF FF FF FF
0110	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0130	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0150	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0160	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0230	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

4 Message(s)

AVR Source Code - U1

```
funciones.c
----- #include "funciones.h"
----- //Andres Guapi P101
-----
----- uint8_t numeroAleatorio(){
00D8   return random()%MAX_DATOS; //valor aleatorio entre [0-100]
00EA   }
00EC   void iniciarArreglo(){ //Para ver de manera gráfica en el Proteus el tamaño del arreglo
00F4   uint8_t i=0;
00F8   for(j=0;j<MAX_DATOS;j++) estatico[j] = 0xFF; //inicialmente damos valor de todo FF para ver como cambian
0102   }
0104   void llenarNumeros(uint8_t cantidad){
0106   for(i=0;i<cantidad;i++){
0108   estatico[i] = 32 + random()%96; //valor aleatorio entre [32 - 128], damos otra semilla para que el valor
0110   }
0112   srandom(cantidad);
0114   }
0116   }
```

AVR Variables - U1

Name	Address	Value
<b>cantidad</b>	<b>00800104</b>	<b>30</b>
<b>i</b>	<b>00800105</b>	<b>30</b>
<b>estatico</b>	<b>00800106</b>	<b>byte[100]</b>
<b>cantidad</b>	<b>00800104</b>	<b>30</b>
<b>i</b>	<b>00800105</b>	<b>30</b>
<b>estatico</b>	<b>00800106</b>	<b>byte[100]</b>

Simulation Log

AVR Data Memory - U1

Address	Value
0100	84 0E 10 1E 1E 00 7F 00 00 00 00 00 00 00 00 00
0110	7C 2E 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F 2F
0120	60 70 00 00 FF FF FF FF FF FF FF FF FF FF FF FF
0130	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0150	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0160	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0230	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

4 Message(s)

PAUSED: 0.133430000s

## Asignación Dinámica

AVR Source Code - U1

funciones.c

```
----- #include "funciones.h"
----- #include "config.h"
----- //Andres Guapi P101

----- uint8_t numeroAleatorio(){
00C0     return random()%MAX_DATOS; //valor aleatorio entre [0-100]
00D2     }
00D4     void llenarNumeros(uint8_t cantidad){
00DC         (char * )malloc(cantidad*sizeof(char)); //adquiere la cantidad de espacio de memoria a utilizar
00EA         for(i=0;i<cantidad;i++){
00F4             ptr[i]=32 + random()%96; //valor aleatorio entre [32 - 128]
0124         }
0130         free(ptr); //liberar la memoria los espacios de memoria para usar por otra u la misma
013C         srandom(cantidad); //cambia la semilla de los numeros aleatorios
}
```

AVR Data Memory - U1

Address	Value
0100	20 00 12 01 00 00 00 00 00 00 00 00 00 00 00 00
011C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0138	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0154	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
018C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01C4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0218	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0234	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
026C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0288	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

AVR Variables - U1

Name	Address	Value
i	0080010D	1
ptr	0080010A	0x0114
cantidad	0080010C	30
i	0080010D	0x01
ptr	0080010A	0x0114
cantidad	0080010C	0x1E
i	0080010D	0x01
ptr	0080010A	0x0114
cantidad	0080010C	0x1E

main.c

```
----- /*CODIGO ASIGNACION DINAMICA*/
----- #include "funciones.h"
----- #include "config.h"
----- //conf():
----- //uint8_t cantidad,i;
----- //char *ptr; //usamos un puntero

----- int main()
00A6     {
-----         srandom(SEMILLA);
00B2         while (1){
00B8             cantidad=numeroAleatorio();
00BA             llenarNumeros(cantidad);
-----         }
-----         return 0;
-----     }
----- //Andres Guapi Mora P101
```

AVR Data Memory - U1

Address	Value
0100	20 00 12 01 00 00 00 00 00 00 00 00 00 00 00 00
011C	21 4E 5A 00 00 00 00 00 00 00 00 00 00 00 00 00
0138	32 50 32 64 48 53 77 60 34 3E 66 6C 2A 33 48 3A
0154	7E 78 40 24 29 47 35 58 49 71 37 28 6A 5E 49 28
0170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
018C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01A8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01C4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01FC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0218	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0234	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
026C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0288	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02A4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

AVR Variables - U1

Name	Address	Value
i	0080010D	46
ptr	0080010A	0x0134
cantidad	0080010C	70

## Memoria EEPROM

The image shows the AVR Studio IDE interface with three main windows open:

- main.c**: The source code for the program. It includes headers for EEPROM, delay, and IO. It defines variables for EEPROM data (dato0, dato1, dato2, texto, valor) and a main function that initializes hardware and writes data to EEPROM.
- AVR EPROM Memory - U1**: A window showing the memory layout of the AVR device. It displays a grid of memory addresses and their corresponding values. The text "...UHola Mundo..." is visible in the memory, indicating that the program has successfully written the string to the EEPROM.
- AVR Variables - U1**: A window showing the current values of the variables defined in the program. The variables are dato1, dato2, texto, valor, and dato0, with their respective addresses and values.

```
----- #include <avr/eeprom.h> //archivo de cabecera para la utilización de la EEPROM AVR
----- #include <util/delay.h>
----- #include <avr/io.h>
-----
----- uint8_t EEMEM dato0 = 0x0A; //creación de la variable dato de 8 bits en la EEPROM AVR
----- uint16_t EEMEM dato1 = 0xAABB;
----- uint8_t EEMEM dato2 = 0x55;
----- uint8_t EEMEM texto[] = "Hola Mundo";
----- uint8_t valor; // variable en la data memori
-----
----- int main(void)
----- {
0090     DDRD=0xFF; //como salidas digitales, para desafio
0094     PORTD=0x00; //se inicia a 0
0096     DDRB &= ~(1 << 0); //PIN 8 DE ARDUINO COMO ENTRADA
00A0     DDRB |= (1 << 3); //pin 11 del Arduino como salida
00A6     valor = eeprom_read_byte(&dato2); //en la variable dato de la EEPROM AVR
----- //leemos lo de dato2 y guarda en valor
00B2     eeprom_write_byte(0xF0,0xAB); //le digo la direccion y el valor
00BC     eeprom_write_byte(&dato0,0x5); //la variable y le asignamos el valor
-----
-----     while(1) //ciclo del programa
-----     {
00C6         asm("NOP");
-----     }
----- //Andres Guapi Mora P101
```

Name	Address	value
dato1	00810001	0
dato2	00810003	0
texto	00810004	byte[11]
valor	00800100	85
<b>dato0</b>	<b>00810000</b>	<b>0x02</b>

Como podemos apreciar en AVR Variables se asignó a una variable de memoria de datos (valor=85) que es el valor de dato2 mostrado en forma decimal.

## Conclusiones

- Fue posible familiarizarse con los distintos tipos de memorias presentes en los microcontroladores específicamente el ATmega328p, ya que conocimos las distintas librerías que necesita cada memoria para almacenar en ella.

- Conocimos la diferencia entre la programación en lenguaje Arduino y en C nativo para el almacenamiento en la memoria EEPROM en la cual vimos que las funciones get y set de Arduino son más cortas, pero estas consumen más memorias al procesarlas.
- Conocimos las distintas formas de asignar valores en la memoria de datos (SRAM) que son asignación estática que ocurre en tiempo de compilación donde establecemos una cantidad máxima de memoria, o asignación dinámica que ocurre en tiempo de ejecución donde hacemos uso de punteros.

## **Recomendaciones**

- En el archivo con las funciones asegurarse de llamar el archivo donde se encuentran creadas las variables del proyecto para evitar errores por no declaración de variables.
- Es recomendable que a la hora de simular no tan solo tener abierto la ventana de Data Memory si no también AVR Variables para poder apreciar de mejor manera como van cambiando los datos.