3. Arreglos de variables / Com. Serie

Tallerine Arduino/LED

Instituto de Ingenería Eléctrica

19 de agosto de 2019

Repaso

Arquitectura del sistema

Von Neumann

- Memoria (estado interno)
- Programa
- Entrada de datos
- Salida de datos

Programación estructurada en arduino

- Constantes
- Variables globales
- Tipos de variables
- Funciones y variables locales
- Dispositivos de E/S

Constantes

- Valores que no cambian durante la ejecución
- Por ejemplo el número π
- Se declaran con `const` al ppio.

Variables globales

- Cambian durante la ejecución
- Son visibles y modificables desde cualquier función
- Existen en un lugar `fijo` en memoria durante la ejecución

Funciones en Arduino

- Conjunto de sentencias en un bloque ({})
- Nombre único, conjunto de parámetros (opcional)
- Valor de retorno (opcional)

```
float promediar(float num1, float num2) {
  float p;
  p = (num1 + num2) / 2.0;
  return p;
}
```

Definición

```
// DEFINIMOS promediar:
// recibe 2 parametros float
// devuelve un float
float promediar(float num1, float num2) { //encabezado
  float p;  // variable local
  p = (num1 + num2) / 2.0; // sentencia
  return p; // termina (RETORNA) y devuelve resultado
}
```

Llamada

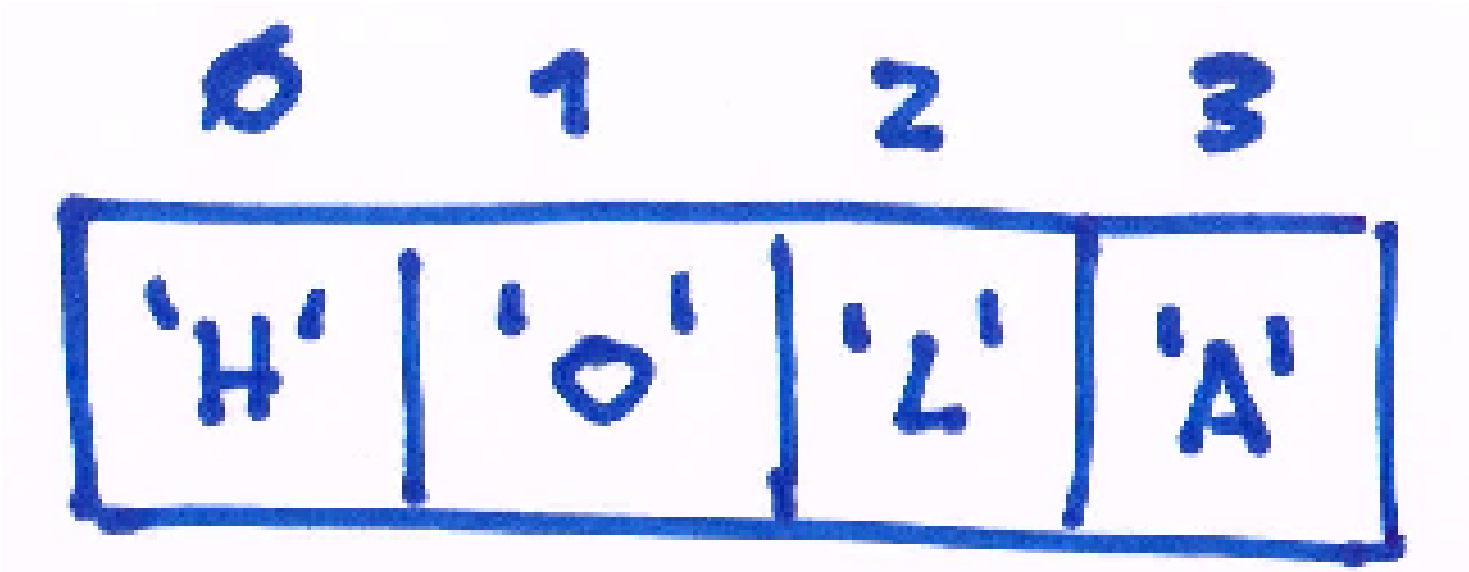
```
void loop() {
  int k;
  // LLAMAMOS a la funcion 'promediar' con num1=4, num2=5
  // al RETORNAR el resultado se guarda en variable k
  k = promediar (4, 5);
}
```

Variables locales (a una función)

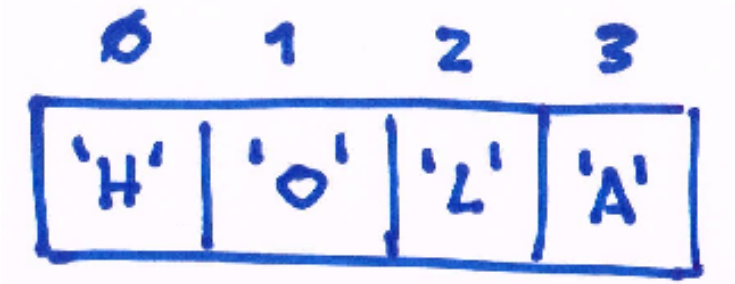
```
float promediar(float num1, float num2) {
  float p;
  p = (num1 + num2) / 2.0;
  return p;
}
```

- Variable `float p` en el ejemplo
- Sólo existen durante la ejecución de una función
- No tienen valor inicial!! (hay que darlo explícitamente)
- Pueden cambiar durante la ejecución de la función
- Su lugar en memoria se crea al entrar a la función, y se borra al terminar la función

Arreglos de variables



Qué es un arreglo?



- variable con múltiples (n) valores
- se accede al i-ésimo valor con un índice de 0 a $n-1$
- parecido a un vector, o una lista
- se puede crear un arreglo de cualquier tipo básico
  - `char`, `int`, `float`, `bool`, etc.

Ejemplo Definición de un arreglo de cuatro caracteres

```
char arr[4] = {'H','O','L','A'};
```

Declaración, definició y uso de arreglos

```
int i = 0;
int val[3]; // DECLARACION

void setup() {
  pinMode(0,OUTPUT);
  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  val[0] = HIGH; // DEFINICION de primer elemento
  val[1] = LOW;  // DEFINICION de segundo elemento
  val[2] = HIGH; // DEFINICION de tercer elemento
}

void loop() {
  digitalWrite(0,val[0]); // USO
  digitalWrite(1,val[1]); // USO
  digitalWrite(2,val[2]); // USO
}
```

Arreglos multidimensionales y matrices

- se puede definir arreglos de arreglos
- y arreglos, de arreglos, de arreglos de ...
- muy útil para manejar *matrices*

Ejemplo: declaración y acceso a matriz de 2×3 enteros:

```
int T[2][3];
void setup() {
  T[0][0] = 1; T[0][1] = 0; T[0][2] = 0;
  T[1][0] = -1; T[1][1] = 1; T[1][2] = 0;
}
```

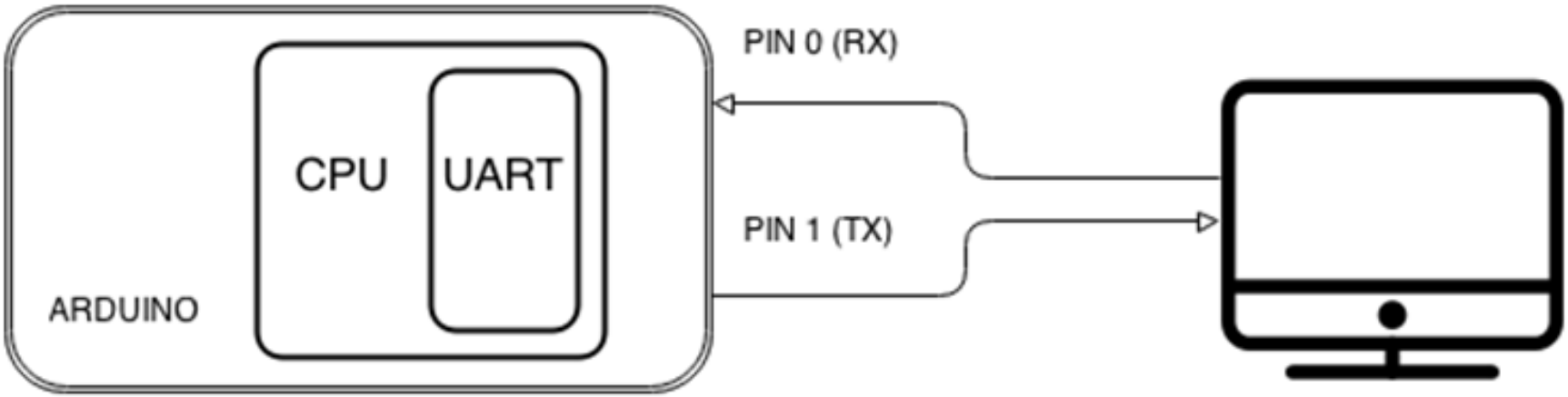
Comunicación Arduino-PC

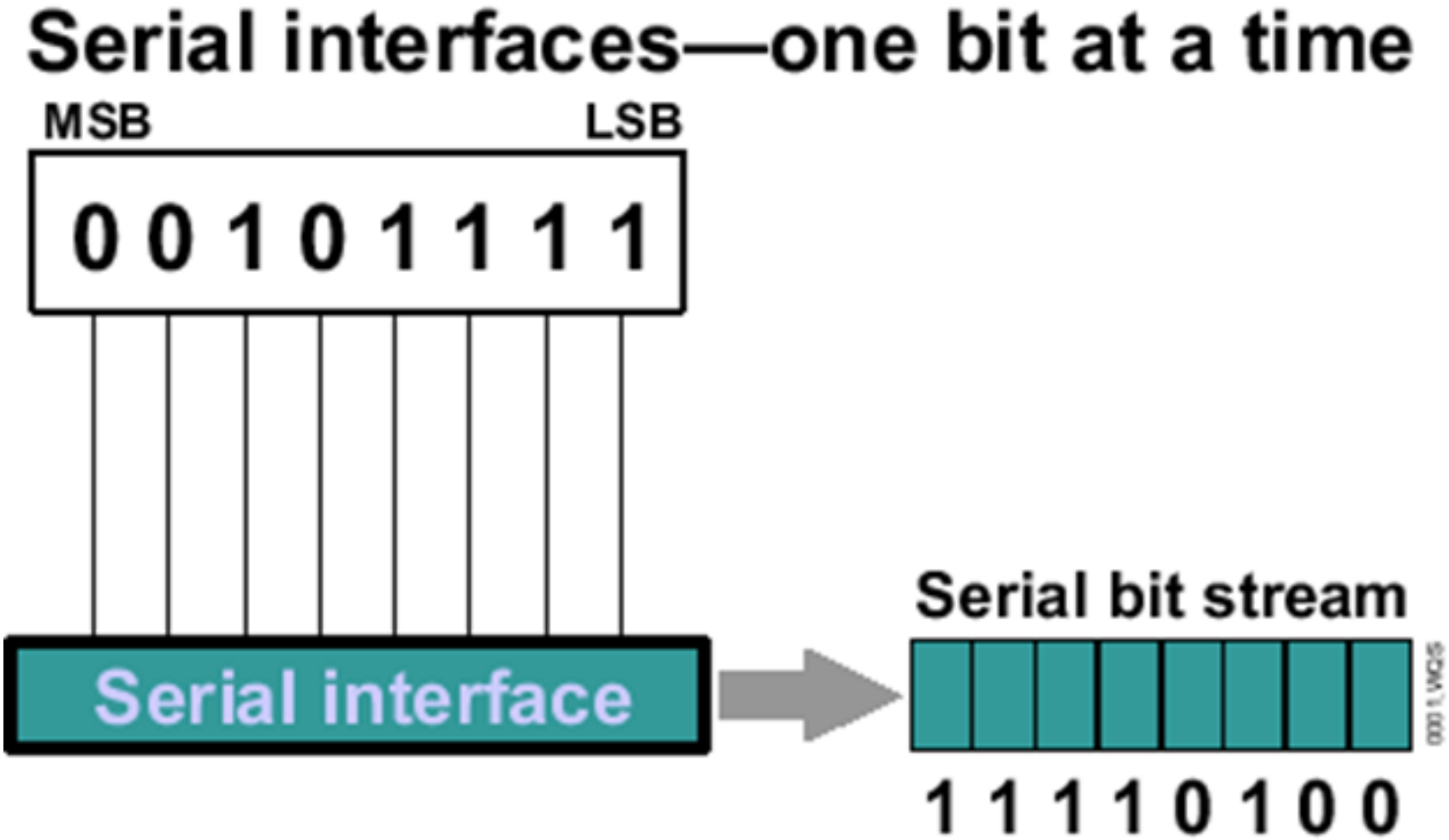
Para qué?

- Enviar comandos al Arduino durante su ejecución
- Recibir información del Arduino en tiempo real
- Depurar!

Comunicación serial Arduino

- Puerto serie incorporado
- Hacia/desde PC via USB
- Hacia/desde otro equipo via pines 0 y 1




Serial interfaces—one bit at a time
MSB
LSB
0 0 1 0 1 1 1 1
Serial interface
Serial bit stream
1 1 1 1 0 1 0 0

Comunicación serial básica

```
/*
 * Hello World!
 */

void setup()
{
  Serial.begin(9600);            // configuro puerto Serial a 9600 bps
}

void loop()
{
  Serial.print("Hello ");  // imprime el string en el puerto
  Serial.println("world!");  // imprime el string en el puerto y agrega salto de línea
  delay(1000);
}
```

Biblioteca Serial

Funciones predefinidas en el Arduino:

- `void Serial.begin(int velocidad)`: comenzar trans. a tal velocidad
- `bool Serial.available()`: hay nuevos datos para recibir?
- `int Serial.read()`: leer bytes recién llegados
- `void Serial.write(byte b)`: escribir bytes
- `void Serial.flush()`: enviar datos pendientes
- `void Serial.print(···)`: imprimir texto, números, etc.
- `void Serial.println(···)`: lo mismo con un ENTER al final

Tabla ASCII

- Lenguaje de máquina es con bits: PCs no manejan caracteres!
- Solución: codificar caracteres numéricamente

Tabla ASCII

- Lenguaje de máquina es con bits: PCs no manejan caracteres!
- Solución: codificar caracteres numéricamente

Tabla ASCII (cont)

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Otro ejemplo Serial: ASCIITable

```
byte byteSigno = 33;  // corresponde al caracter "!"


void setup() {
  Serial.begin(9600);           // configuro puerto Serial a 9600 bps
}

void loop() {

  Serial.write(byteSigno );     // Escribe el byte


  Serial.print(" dec: ");  Serial.print(byteSigno );     // En decimal


  Serial.print(" bin: ");  Serial.print(byteSigno, BIN );     // En binario


  Serial.print(" hex: ");  Serial.println(byteSigno, HEX );     // En hexadecimal

 byteSigno++;  // siguiente caracter

  if(byteSigno == 126) while(True){}; // una vez que termina se queda esperando
}
```

Ejercicio en clase

- Implementar un Echo: el Arduino lee un byte que se envía desde la PC y luego lo devuelve
- Cambiar el `System.write` por un `System.print`. Qué pasa? Cómo se explica?

Echo Serial

```
/*
 * Echo: El Arduino recibe el caracter enviado desde la PC, lo lee y lo devuelve
 */
byte byteLeido;  // variable auxiliar


void setup()
{
  Serial.begin(9600);           // configuro puerto Serial a 9600 bps
}

void loop()
{
  if(Serial.available()) {  // chequea si se recibe información
    byteLeido = Serial.read();  // leo el byte enviado y lo guardo en byteLeido

    Serial.write(byteLeido);  // mando el byte por el puerto serie
  }
}
```

Ejercicio: Enviar Comando PC-Arduino

En el siguiente ejercicio se enviará un comando desde el PC al arduino. Este comando será un número entre 0 y 255 que se utilizará para hacer de dimmer de un led.
Para que no se cometa un error de tipeo, se enviará el valor entre dos caracteres, uno que indica el inicio (@) y otro que indica el fin del mensaje(!). En medio van los bytes correspondientes al n"umero que se desea enviar.
En el código que se les entrega (en eva ejercicio 1) se hace de dimmer de un led. Se pide: entender los tres programas que se les entregan y se utilizan para variar la intensidad de un led y probarlo. En particular se deberá entender detalladamente las funciones Leocomando y comandoValido.

Ejercicio: Enviar Comando PC-Arduino

Modificar el ejercicio anterior para que el valor ingresado modifique la intensidad de tres leds que se encienden secuencialmente. Se quiere ademś, que desde el PC no solo se envíe la intensidad de los leds sino que ademá se envíe un 0 o un 1 que indique la dirección de secuenciamiento.

Ejercicios para próxima clase

Entregable: Animación LED

1. Montar un circuito que maneje cuatro LEDs en hilera
2. Definir un arreglo de tamaño 4×10 de tipo `int` con valores `LOW` o `HIGH` a gusto del consumidor
3. Programar el arduino para que cada 1/4s muestre una nueva fila de la matriz en la hilera de LEDs, donde cada elemento en la hilera de la matriz defina el valor del LED correspondiente en la hilera de LEDS
4. Las filas deben ser mostradas una por una comenzando desde la 0; al llegar a la fila 9 la próxima ser'e nuevo la 0.