

# Taller Introducción a la Ingeniería Eléctrica Robot y comunicaciones basados en Microcontrolador Arduino

Instituto de Ingeniería Eléctrica

Taller 7: Aspectos básicos de comunicación.

25 de Octubre de 2017

## ¿Qué es la comunicación?

- Es la transferencia de información de un lugar a otro.

## ¿Para qué se quiere comunicación?

- Necesidad de compartir información con otro/s dispositivo.

## Sistema de comunicación: SHANNON (1947)

## ¿Qué es la comunicación?

- Es la transferencia de información de un lugar a otro.

## ¿Para qué se quiere comunicación?

- Necesidad de compartir información con otro/s dispositivo.

## Sistema de comunicación: SHANNON (1947)

## ¿Qué es la comunicación?

- Es la transferencia de información de un lugar a otro.

## ¿Para qué se quiere comunicación?

- Necesidad de compartir información con otro/s dispositivo.

Sistema de comunicación: SHANNON (1947)

## ¿Qué es la comunicación?

- Es la transferencia de información de un lugar a otro.

## ¿Para qué se quiere comunicación?

- Necesidad de compartir información con otro/s dispositivo.

Sistema de comunicación: SHANNON (1947)

# Introducción

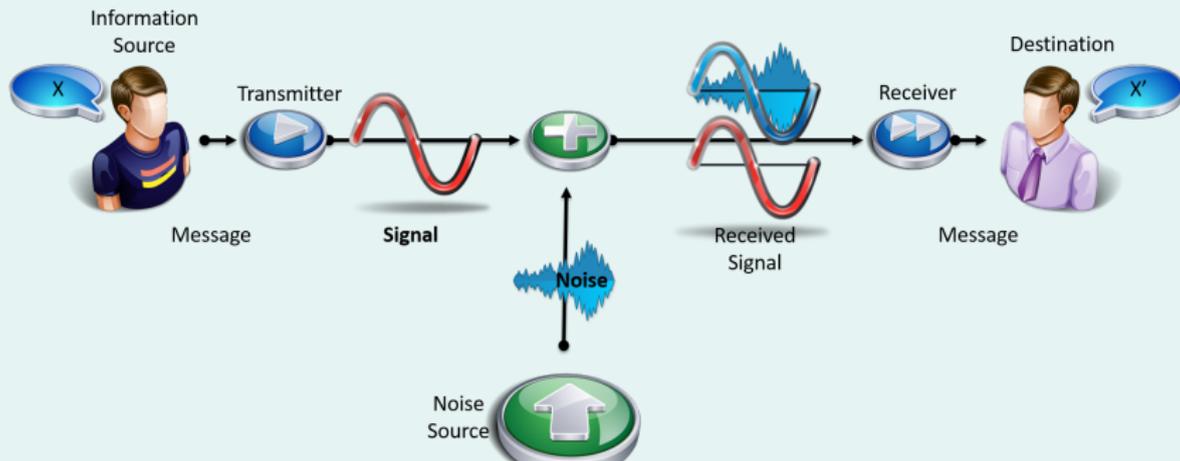
## ¿Qué es la comunicación?

- Es la transferencia de información de un lugar a otro.

## ¿Para qué se quiere comunicación?

- Necesidad de compartir información con otro/s dispositivo.

## Sistema de comunicación: SHANNON (1947)



## Comunicación en sistema tipo Arduino: ¿Para qué?

- Sensar eventos.
- Mostrar información al usuario, debug.
- Compartir información con otra parte del sistema.

Trasmisión de datos: SERIE - PARALELO

## Comunicación en sistema tipo Arduino: ¿Para qué?

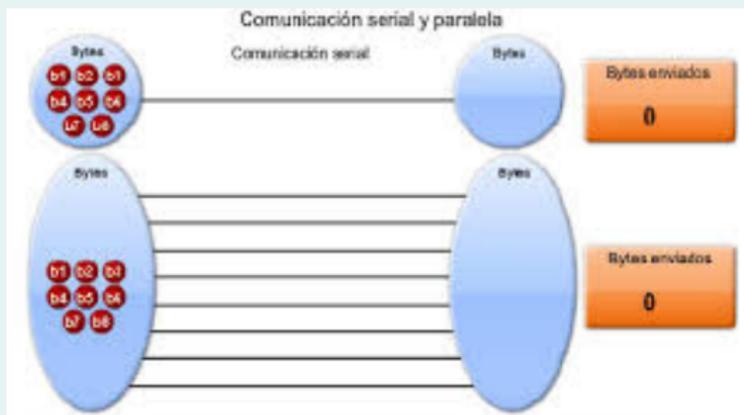
- Sensar eventos.
- Mostrar información al usuario, debug.
- Compartir información con otra parte del sistema.

Trasmisión de datos: SERIE - PARALELO

## Comunicación en sistema tipo Arduino: ¿Para qué?

- Sensar eventos.
- Mostrar información al usuario, debug.
- Compartir información con otra parte del sistema.

## Trasmisión de datos: SERIE - PARALELO

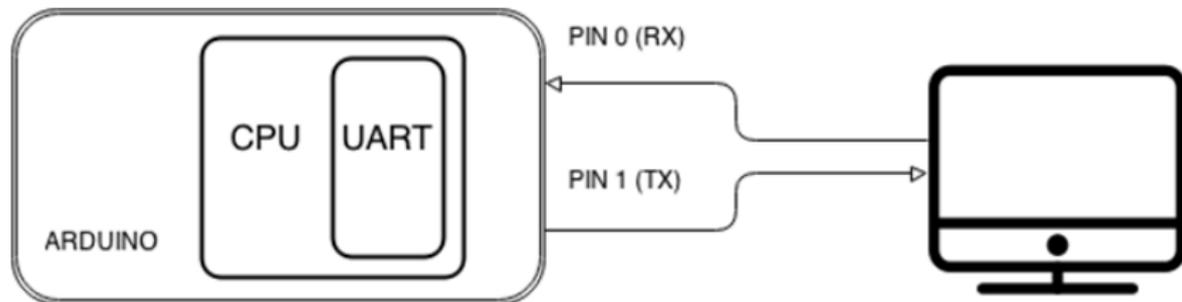


## Arduino: comunicación Serial

- Todos los Arduinos tienen por lo menos un puerto serie “inconsciente” (UART o USART).
- En pines 0 y 1 o también en USB.

## Arduino: comunicación Serial

- Todos los Arduinos tienen por lo menos un puerto serie “inconsciente” (UART o USART).
- En pines 0 y 1 o también en USB.

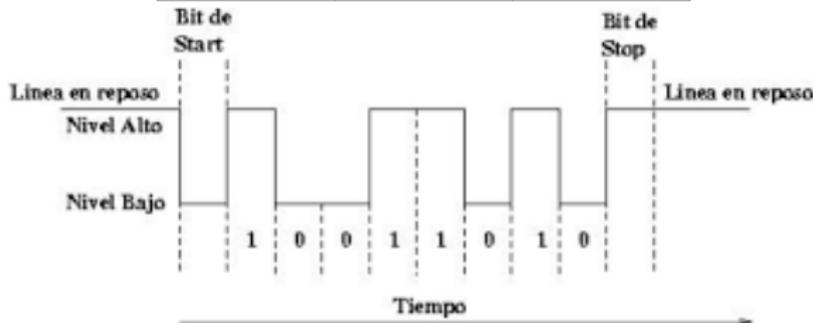


# Comunicación Serial Asíncrona

## Serial interfaces—one bit at a time



Pin Arduino	Pin	Modo Pin
0	Tx	INPUT
1	Rx	OUTPUT



## Funciones predefinidas para manejo de puerto serial:

- begin()
- available()
- read()
- write()
- print() / println()
- flush()

Uso: Serial.nombrefunción( );

# Ejemplo 1: librería Serial

```
/*  
 * 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);  
}
```

## Ejemplo 2: librería 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
  }
}
```

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

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

# Tabla ASCII

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

Source: [www.LookupTables.com](http://www.LookupTables.com)

## Ejemplo 3: 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
}
```



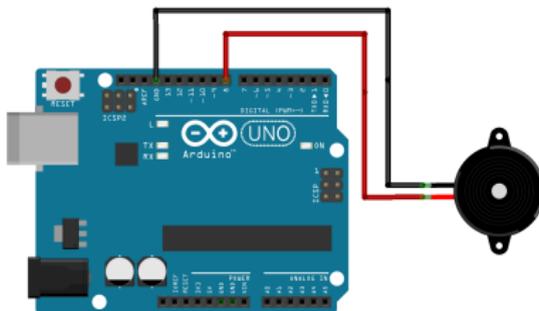
- ¿Qué otro tipo de alfabeto para comunicación serial conoce?

- ¿Qué otro tipo de alfabeto para comunicación serial conoce?

**SOS**  
• • — — — • •  
**Morse code**

# Ejercicio 1

- 1 Completar el alfabeto Morse escribiendo la codificación para las vocales.
- 2 Escribir la función *void MorseDash()* tomando como referencia la función *MorseDot()*.
- 3 Utilizando las funciones *tone( , )*, *delay()* y *noTone()*, que se pueden consultar en la web de Arduino, escribir una función *boozer()* que prenda y apague un piezo eléctrico. Se utilizará ésto para transmitir el mensaje Morse en forma sonora.
- 4 Recibir un caracter desde el puerto serial del Arduino y traducirlo a código Morse, mostrarlo en forma lumínica y/o sonora.



```
int elemPause = dotLen; // tiempo de la pausa entre elementos de un caracter
int Spaces = dotLen * 3; // tiempo de la pausa entre espacio de dos caracteres
int wordPause = dotLen * 7; // tiempo de pausa entre palabras
void setup() {
  // Inicializo los puentes del LED que enviara el mensaje luminico.
  // Inicializo los puentes del parlante que enviara el mensaje sonoro.
  // Inicializo el puerto Serial.
}
void loop()
{
  // espero recibir una letra desde el serial
  while (Serial.available() <= 0){
    Serial.print(".");
    delay(1000);
  }
  Serial.println("leo el mensaje");
  // leo la letra en el serial

  morseChar(letra) // traduzco el mensaje recibido una letra a la ves
  // Apago el Led y espero la proxima letra mensaje
  LightsOff(8000);
}

// Punto
void MorseDot()
{
  digitalWrite(led, HIGH); // prendo el Led
  digitalWrite(parlante, HIGH); // prendo el parlante
  delay(dotLen); // Espero tiempo de punto
}

// Linea
void MorseDash()
{
  // escribir MorseDash
}

// Apagar
void LightsOff(int espera)
```

Guardado Terminado.

- Hasta ahora conocemos *if*.

*if*

```
if (expresion){
    sentencias_if;
} else {
    sentencias_else;
}
```

- ¿Qué sucede si tenemos varias opciones?
  - Se usa la sentencia de selección *switch-case*.

- Hasta ahora conocemos *if*.

*if*

```
if (expresion){
    sentencias_if;
} else {
    sentencias_else;
}
```

- ¿Qué sucede si tenemos varias opciones?
  - Se usa la sentencia de selección *switch-case*.

- Hasta ahora conocemos *if*.

*if*

```
if (expresion){
    sentencias_if;
} else {
    sentencias_else;
}
```

- ¿Qué sucede si tenemos varias opciones?
  - Se usa la sentencia de selección *switch-case*.

## *switch-case*

```
switch (variable)    {
    case 1:
        sentencias1;
        break;
    case 2:
        sentencias2;
        break;
    case n:
        sentenciasN;
        break;
    default:
        sentencias_default;
        break;
}
```

- Es una estructura de datos que sirve para almacenar cadenas de texto, es fundamental a la hora de enviar texto por el puerto serie.
- ¿Cómo se inicializa un string?

## Recorro y muestro en pantalla *myArray*

```
char String1[15];
char St2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char St[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[ ] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
String str7 = String(str4 + " with more");
// concatena 2 string
```

Tener presente el caracter **null** al trabajar con “arreglos de caracteres”.

## Ejercicio 2

- 1 Estudiar y entender la función *void recibirMensaje()*.
- 2 Completar el código para que reciba un string por el serial del PC y recorriendo un array de tipo char pueda mostrarlo en Morse a través del Led y del parlante.

## Ejercicio 3: trabajamos colaborativamente entre equipos

- 1 Conectar dos Arduinos por su puerto serie.
- 2 Programar uno como transmisor el cual enviará un mensaje (tipo *string*) contenido en el programa al receptor, este último lo mostrará en Morse con el Led y el Parlante.
- 3 Escribir el código que recorra el *array* de caracteres y los muestre de a uno con la función *charMorse()*.