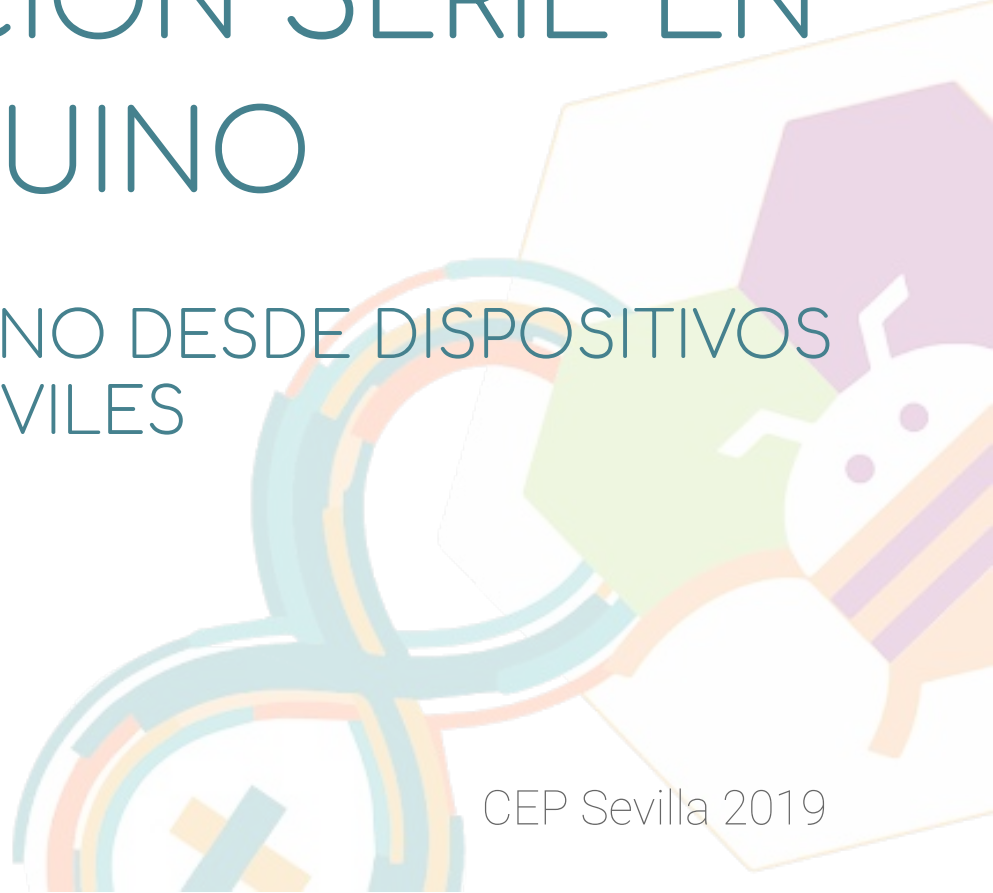


# COMUNICACIÓN SERIE EN ARDUINO

CONTROLANDO ARDUINO DESDE DISPOSITIVOS  
MÓVILES

Jose Luis Núñez  
José Pujol

CEP Sevilla 2019



# ÍNDICE



1. Comunicaciones en Arduino
2. Comunicación Serie en Arduino
3. Ejemplos Arduino → PC
4. Control encendido LED desde PC
5. Control brillo LED desde PC



# COMUNICACIONES EN ARDUINO



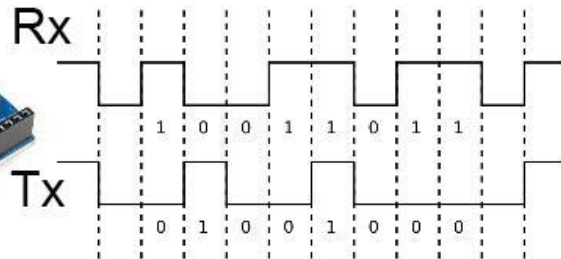
- Comunicación USB
- Comunicación SPI
- Comunicación I2C
- Comunicación OneWire



# COMUNICACIÓN USB ARDUINO



- Arduino usa comunicación serie asíncrona para comunicarse con el PC a través de Universal Serial Bus USB
- Usa los pines Rx y Tx
- Envía series de 8bits



# Aplicaciones de la comunicación serie

---

- Detección de errores
- Monitorización de sensores
  - Monitor serie
  - Serial plotter
- Control de Arduino desde PC
- Envío y recepción de datos vía BlueTooth

# Configuración de la comunicación Serie

---

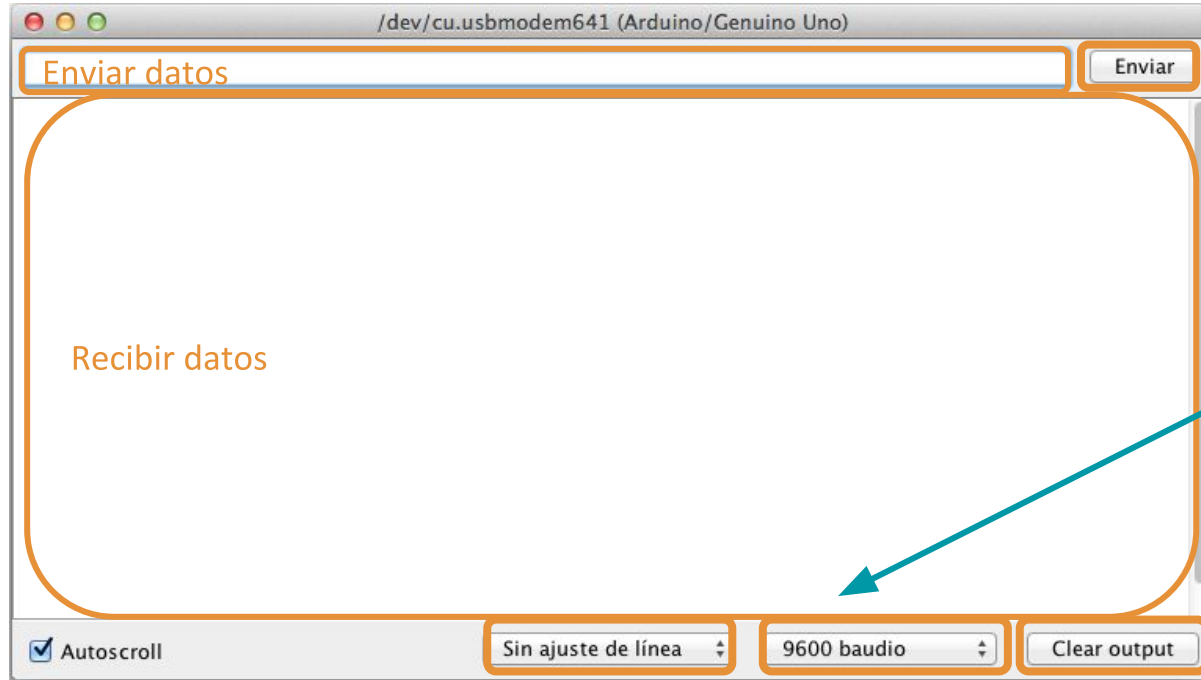
Necesitamos configurar la velocidad en el setup:

```
Serial.begin(baudrate);
```

**baudrate:** velocidad de comunicación en baudios por segundo. bits/s

La velocidad por defecto se suele configurar en 9600 baudios/s

# Monitor Serie



Velocidad comunicación  
`Serial.begin(9600);`  
debe coincidir con  
nuestra configuración

Configura la forma en que se  
envían los datos

Limpia la pantalla

# Impresión por puerto Serie

---

- `Serial.print ();`
- `Serial.print ("Texto");`
- `Serial.print (variable);`

Imprime los datos por  
el puerto serie

- `Serial.println ();`

Añade retorno  
de carro



# EJEMPLOS ARDUINO→ PC



- Realización de un contador de segundos
- Imprimir un sensor por monitor serie



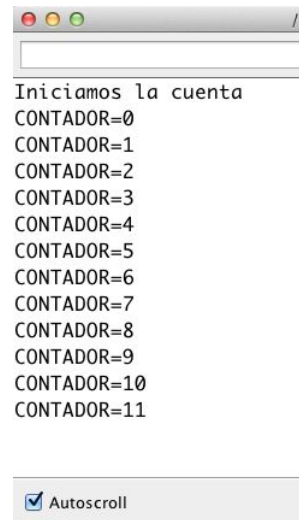
# Contador de segundos

```
int contador = 0; // variable para almacenar el valor del contador

void setup() {
  // abre el puerto serie
  // y establece la velocidad de conexion en baudios
  Serial.begin(9600);
  Serial.println("Iniciamos la cuenta"); // Imprime texto
}

void loop() {

  Serial.print("CONTADOR="); // Imprime texto
  Serial.println(contador); // Imprime el valor de la variable contador
  delay(1000); // tiempo de espera de 1s
  contador++; // se incrementa el valor del contador en 1
}
```



# Monitorización de sensores

```
2 const int sensorPin = A0;    // establece el pin del sensor
3 int sensorValue = 0;         // variable para almacenar el valor del sensor
4
5 void setup() {
6     // abre el puerto serie
7     // y establece la velocidad de conexión en baudios
8     Serial.begin(9600);
9 }
10
11 void loop() {
12     // lee el valor del sensor
13     sensorValue = analogRead(sensorPin);
14     // Imprime un texto
15     Serial.print("Valor Sensor=");
16     // Imprime el valor de la variable por el puerto serie
17     Serial.println(sensorValue);
18     // tiempo de espera para visibilidad
19     delay(1000);
20 }
```

# RECEPCIÓN DE DATOS EN ARDUINO



- Código ASCII
- Envío de datos desde PC
- Lectura de datos en Arduino
- Lectura de números decimales



# Código ASCII

**ASCII** | Estándar de representación de datos. Establece el significado de determinados códigos.

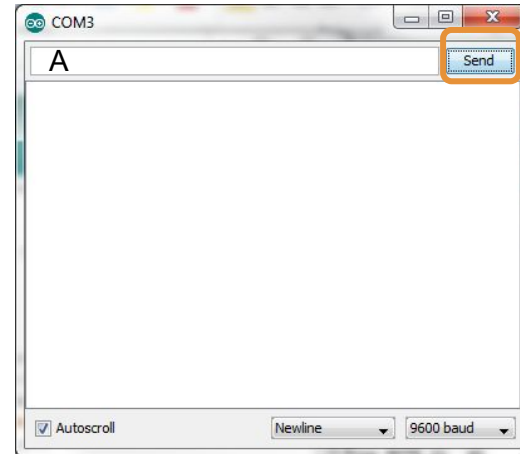
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Fuente

# Envío de datos

Carácter	Código ASCII
'0'	48
'1'	49
...	...
'g'	57
'A'	65
....	...
'Z'	90
'a'	97
...	...
'z'	122

Transmitimos una letra A  
Se envía 65  
01000001 (en binario)



# Lectura de datos en Arduino

## `Serial.available()`

Devuelve el número de bytes almacenados en el buffer de entrada (0 si no hay datos).  
Máximo 64 bytes

## `Serial.read()`

Lee el primer byte disponible en el buffer de entrada (devuelve -1 si no hay datos).

# CONTROL ENCENDIDO LED DESDE PC



**Finalidad** | Activar/desactivar un LED desde el monitor serie.

ENTRADAS

Puerto Serie:  
Enviar H encender  
Enviar L apagar

ARDUINO

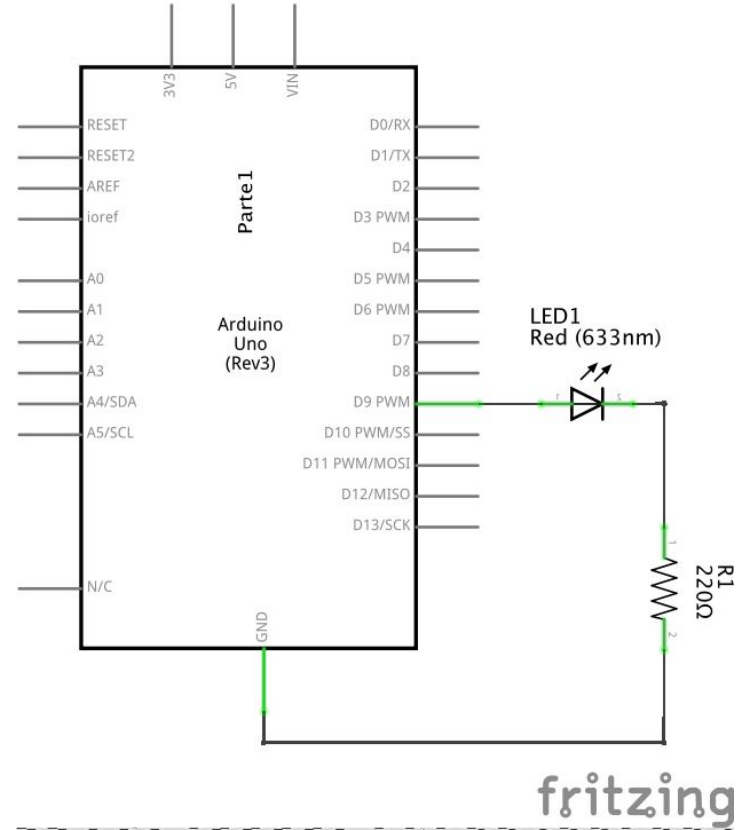
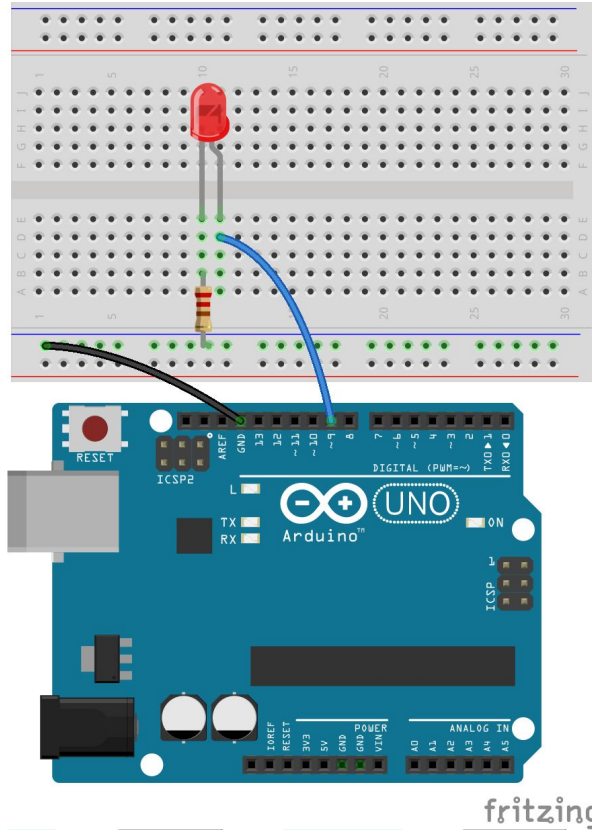
SALIDAS

LED (D)



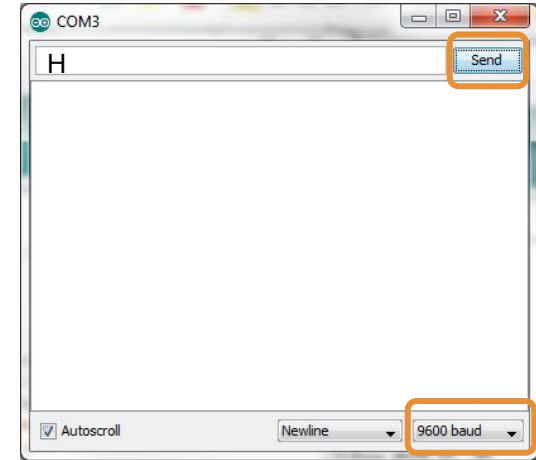


# Hardware



# Software

```
1 const int ledPin = 9; // pin al que el led esta conectado
2 int incomingByte;      // variable para leer los bytes de entrada
3
4 void setup() {
5     // inicializamos la comunicacion serie
6     Serial.begin(9600);
7     // inicializamos el led como pin digital salida
8     pinMode(ledPin, OUTPUT);
9 }
10
11 void loop() {
12     // comprobamos si hay datos de entrada
13     if (Serial.available() > 0) {
14         // lectura del byte mas antiguo del buffer serial
15         incomingByte = Serial.read();
16         // si el byte es H (ASCII 72) enciende el LED
17         if (incomingByte == 'H') {
18             digitalWrite(ledPin, HIGH);
19         }
20         // si el byte es L (ASCII 76) apaga el LED
21         if (incomingByte == 'L') {
22             digitalWrite(ledPin, LOW);
23         }
24     }
25 }
```



La velocidad de conexión debe ser la misma que estamos utilizando en nuestro programa.

Ejemplos → Communication → Physical Pixel

# Propuesta de actividades

---

- Probar el sistema
- Probar a que reconozca también las letras h y l

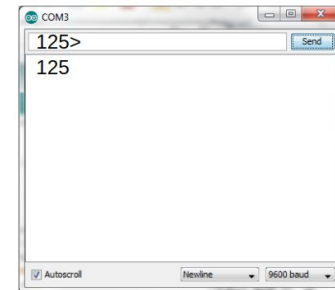
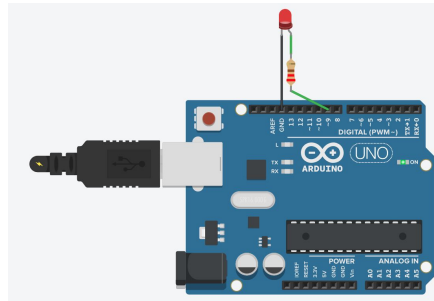
# CONTROL BRILLO LED DESDE PC



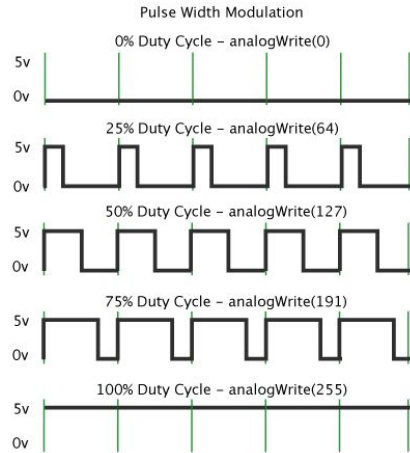
**Finalidad** | Controlar el valor de PWM de una salida vía Serie.



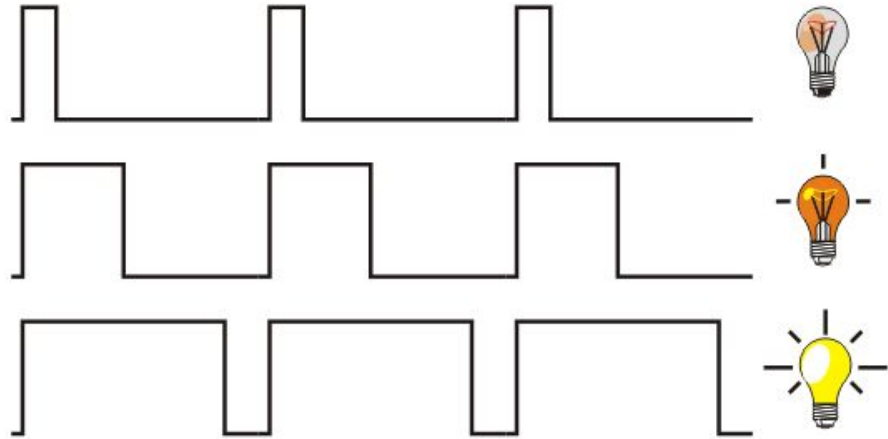
Escribimos en el puerto serie un valor que será leído por Arduino y el LED mostrará el nivel de brillo correspondiente.



# Salidas PWM

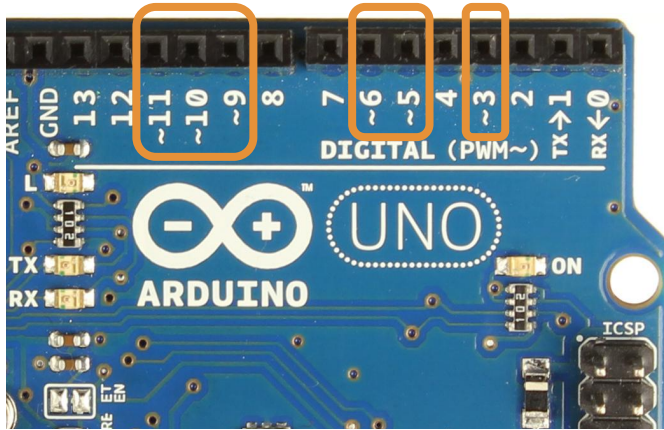


**8 bits =  $2^8$  = 256 niveles**



**Frecuencia en Arduino UNO: 490Hz**

# Salidas PWM



Salidas PWM (6)  
marcadas con ~

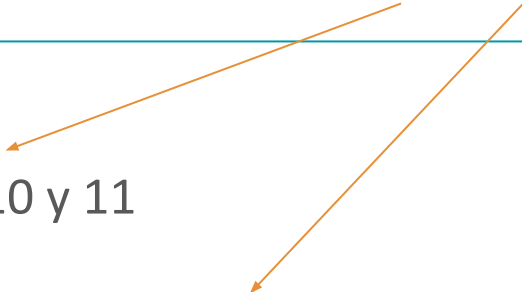
Las salidas PWM no necesitan configuración

# Salidas PWM

Estableceremos el % del ciclo ON (Duty cycle) con la siguiente función:

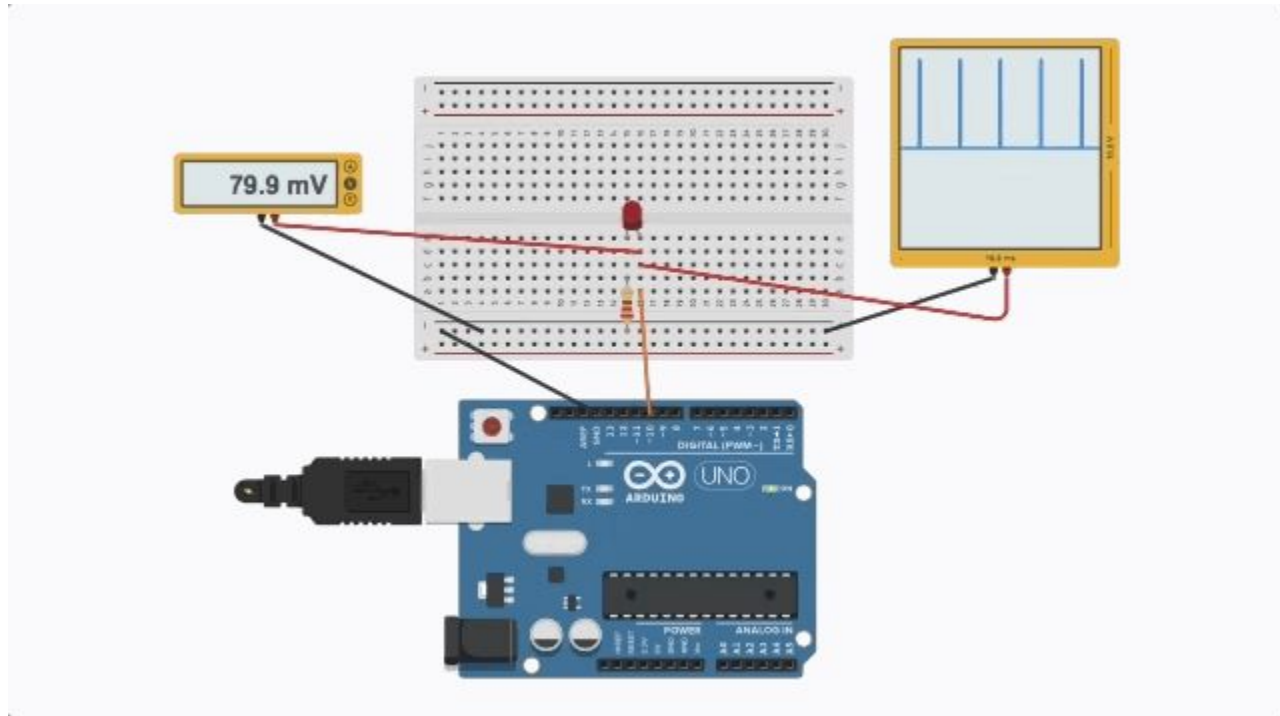
```
analogWrite(pin,valor);
```

3, 5, 6, 9, 10 y 11



Valor	Duty Cycle
0	0
255	100

# Simulación TinkerCAD





# Lectura de números decimales

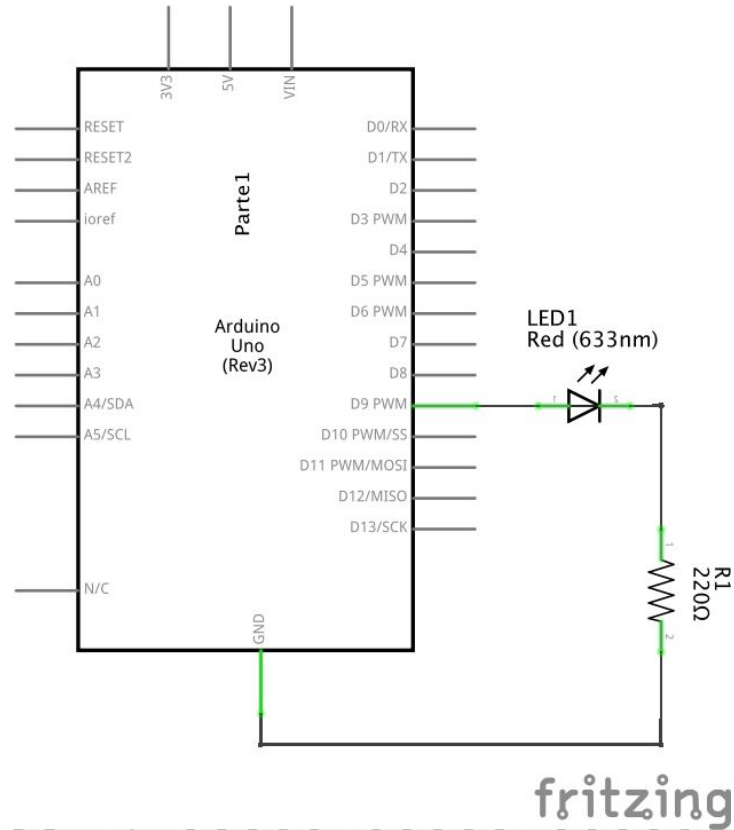
Carácter	Código ASCII
'0'	48
'1'	49
...	...
'9'	57
'A'	65
....	...
'Z'	90
'a'	97
...	...
'z'	122

Al enviar el número 125 (desde el terminal serie)  
Enviamos '1','2','5'

'1'	'2'	'5'
49	50	53
00110000	00110001	00110101

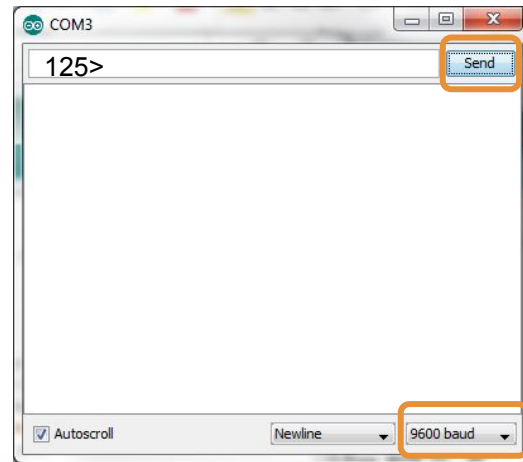
$$\begin{aligned} &1 \\ &1 \times 10 + 2 \\ &(1 \times 10 + 2) \times 10 + 5 \\ &1 \times 100 + 2 \times 10 + 5 = 125 \end{aligned}$$

# Hardware



# Software

```
1 const int ledPin = 9; // pin al que el LED esta conectado
2
3 int incomingByte; // variable para leer los bytes de entrada
4 int value = 0; // variable para almacenar el valor numerico
5 int pwmValue = 0; // variable para enviar el codigo pwm al led
6
7 void setup() {
8     // inicializamos la comunicacion serie
9     Serial.begin(9600);
10 }
11
12 void loop() {
13     // comprueba si hay datos entrantes en el puerto serie
14     if (Serial.available() > 0) {
15         // lectura del byte mas antiguo del buffer serial
16         incomingByte = Serial.read();
17         // si es un caracter ASCII entre 0 y 9
18         if (incomingByte >= '0' && incomingByte <= '9') {
19             //Acumula los datos numericos multiplicando por 10 el valor acumulado
20             value = (value * 10) + (incomingByte - '0'); // Resta 48 que es el valor decimal del 0 ASCII
21         }
22         else if (incomingByte == '>') // uso > como finalizador
23         {
24             pwmValue = value; // Guarda el valor en la variable pwmValue
25             Serial.println(pwmValue); // Lo imprime por monitor serie
26             value = 0; // Dejamos lista la variable para volver a escribir en ella
27         }
28     }
29     analogWrite(ledPin, pwmValue); // Escribimos el valor PWM en el LED
30 }
```



# Propuesta de actividades

---

- Limitar el rango a valores entre 0 y 255
- Añadir un mensaje de error si el valor PWM es mayor de 255 o menor de 0

# LICENCIA



Esta guía se distribuye bajo licencia Reconocimiento-CompartirIgual Creative commons 4.0

Las diapositivas son obra de Jose Pujol y Jose Luis Núñez creadas para el curso “Controlando Arduino desde el teléfono móvil” para el CEP de Sevilla y han sido creadas a partir del material elaborado para el curso “Tech Project: Arduino en el aula” que fue realizado por Jose Antonio Vacas y Jose Pujol en colaboración con Avante s.l.

