Obligatorio 2: ANEXO

Introducción

En este anexo se muestran algunas definiciones y conceptos útiles para el mejor desempeño en el curso.

Definiciones útiles

- **Bit**: Unidad mínima de información, solo 2 valores (cero y uno)
- **Byte**: 8 bits de información (256 valores distintos)
- **KByte**: 1024 bytes = 2^10 bytes
- **Mbyte**: 1024 kbytes = 2^10 kbytes = 2^20 bytes = 1048576 bytes

Números enteros

Se muestran números enteros de 3 y 8 bits. Es simple deducir para el caso de “n” bits.

Números enteros de 3 bits (rango de -4 a 3)	Números enteros de 8 bits (rango de -128 a 127)
3 ➔ 011 2 ➔ 010 1 ➔ 001 0 ➔ 000 -1 ➔ 111 -2 ➔ 110 -3 ➔ 101 -4 ➔ 100	127 ➔ 0111 1111 .... ➔ ..... 1 ➔ 0000 0001 0 ➔ 0000 0000 -1 ➔ 1111 1111 .... ➔ ..... -127 ➔ 1000 0001 -128 ➔ 1000 0000

Lo importante es comprender el rango representable.

Para 2 bytes (16 bits) el rango es -32768 a 32767.

Booleanos

Un dato de tipo booleano tiene 2 valores posibles: Verdadero y Falso

Booleano de 1 bit	Booleano de 8 bits
Verdadero ➔ 1 Falso ➔ 0	Verdadero ➔ 0000 0001 Falso ➔ 0000 0000

ASCII

Un dato ASCII codifica un carácter según la siguiente tabla:

Binario	Dec	Hex	Representación
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binario	Dec	Hex	Representación
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

Tipos de datos en Arduino IDE

Los siguientes son los tipos de datos que se utilizarán en el curso:

- **int** entero de 2 bytes -32 768 a 32 767
- **bool** booleano, 1 byte TRUE o FALSE
- **char** carácter, 1 byte tabla ASCII

Otros tipos de datos: byte, unsigned int, long, float, string, void, word, short, array, double, etc.

Variables de datos

Una variable es un identificador (nombre) que representa un lugar en memoria reservado para almacenar un dato y poder modificarlo cuando se están ejecutando los programas.
Para definir una variable (reservar un lugar en memoria) se debe indicar:

- Nombre de la variable
- Tipo de datos que maneja
- Valor inicial (es opcional, si no se indica su valor inicial es desconocido)

Ejemplo:

```
int val = 0;
```

Nombre: “val”
Tipo de Datos: int (entero -32 768 y 32 767)
Valor inicial: 0

Para modificar el valor de una variable se utiliza el signo “=”: `val= ang_ini;`

Constantes

Una constante es un identificador (nombre) que representa un dato y mantiene su valor en todo el programa. **No es modificable**.

Existen 2 formas para definir una constante:

- **const**: Es igual que una variable, pero no se puede escribir. Utiliza memoria.
  Esta declaración incluye el tipo de datos
  Ejemplo: `const int pin_servo = 9;`

- **#define**: Define un nombre que hace referencia a un valor. No utiliza memoria.
  El nombre es sustituido por el valor al compilar.
  Ejemplo: `#define TRIGGER_PIN 12`

Operadores en Arduino IDE

Los operadores que se utilizarán en el curso se pueden resumir en las siguientes tablas:

Aritméticos	
Suma	A + B
Resta	A – B
Producto	A * B
División	A / B
Resto de división entera	A % B

Comparación	
A igual a B	A == B
A distinto de B	A!=B
A menor que B	A<B
A menor o igual a B	A<=B
A mayor que B	A>B

Librerías (o Bibliotecas)

Una librería es código que ya trae resueltas de forma sencilla tareas como:

- trabajar con hardware (servos, sensores, bluetooth, etc)
- manipular datos (cálculos, audio, video, etc).

Existen miles de librerías que se pueden bajar de Internet para diferentes aplicaciones.

Para utilizar una librería, debe estar agregada a Arduino IDE. Para ello, seleccionar *Programa > Incluir Librería > Añadir biblioteca.ZIP...* Una vez realizado esto, reiniciar Arduino IDE y

verificar si se encuentra en el listado de librerías, *Programa* > *Incluir Librería.*

Para incluir una librería en el Sketch se agrega en la parte superior la sentencia lo siguiente:

```
#include <nombre_libreria.h>
```

Para utilizar las funciones de una librería se debe declara una variable asociada a la librería y entonces se pueden utilizar las funciones de la librería indicando como se indica a continuación:

- Se declara la variable: `nombre_objetos nombre_variable`
- Se utiliza una función: `nombre_variable.nombre_funcion(…)`

Esta variable “especial” se la conoce como variable objeto.

Ejemplo: Si se va a utilizar el controlador de servomotores, se declara la siguiente variable:

```
PCA9685ServoController servo_ctrl(ADDR);
```

En este caso se dice `servo_ctrl` es un objeto de tipo `PCA9685ServoController.` Al declarar el objeto puede ser necesario pasarles determinados parámetros, en este caso ADDR.

Estos objetos tienen funciones propias de él. Un ejemplo es la función `move_servo,` esta permite mover los servomotores. La utilizamos de la siguiente manera:

```
servo_ctrl.move_servo(PIN_SERVO, angulo);
```

Aclaración: este caso, la librería `udriver_pca9685_fing` debe ir acompañada de `using namespace UDriver_PCA9685;` debajo de `#include <udriver_pca9685_fing.h>`, **ver obligatorio 2**.