ARDUINO Y PROGRAMACION
EN EL DESARROLLO DE PROYECTOS DE CIENCIAS
Y
ELECTRONICA

SESION 2

PROTOCOLOS

DE COMUNICACIÓN SERIAL

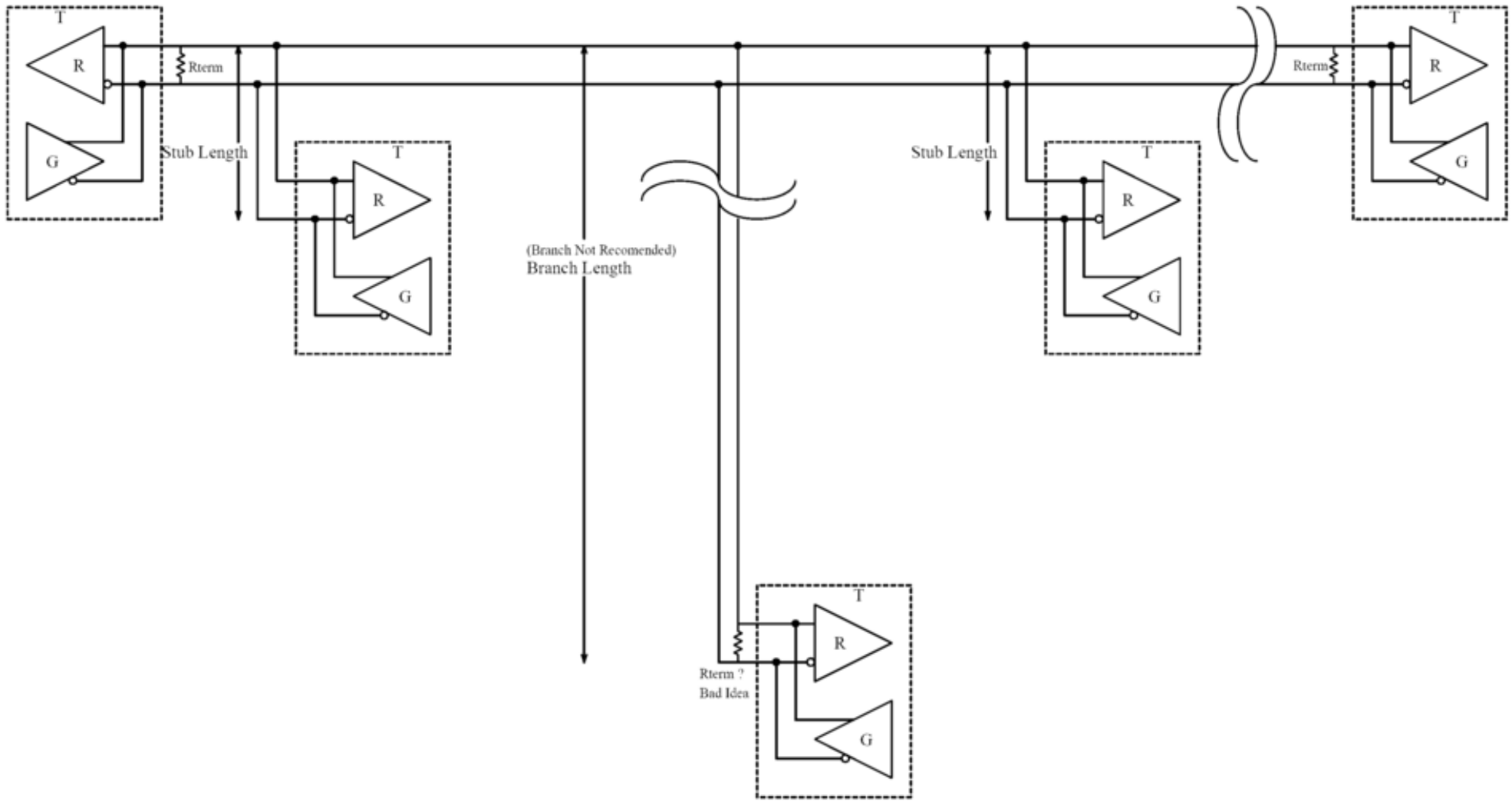
a

NIVEL DE HW

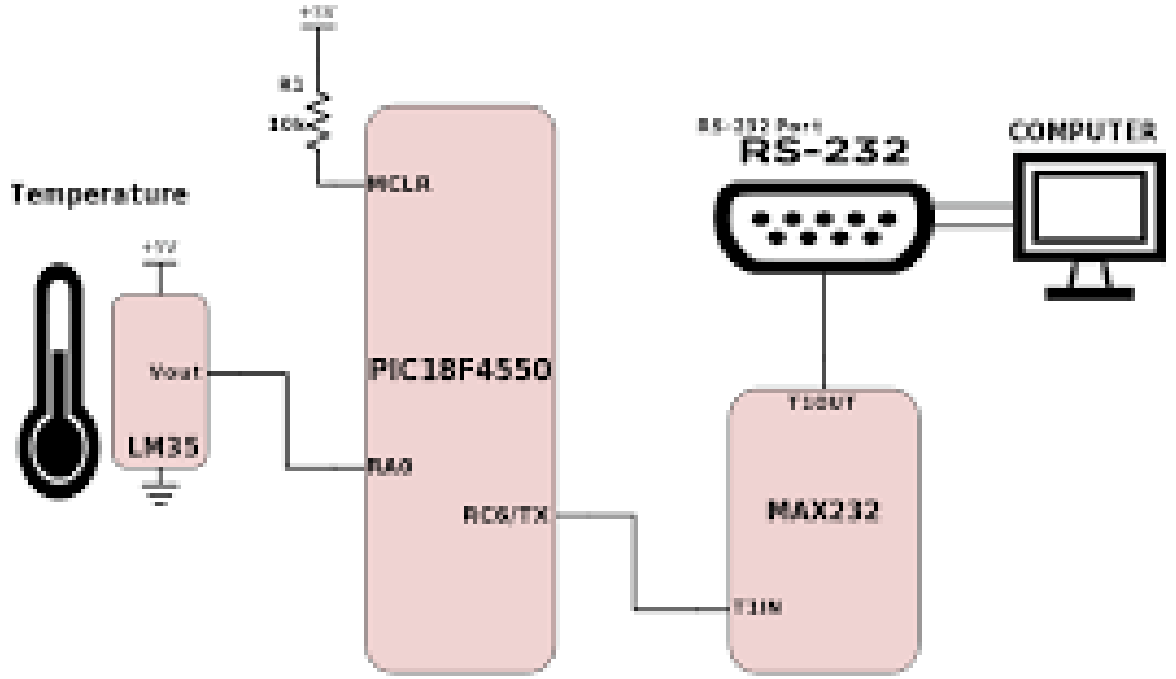
¿Qué protocolos existen?

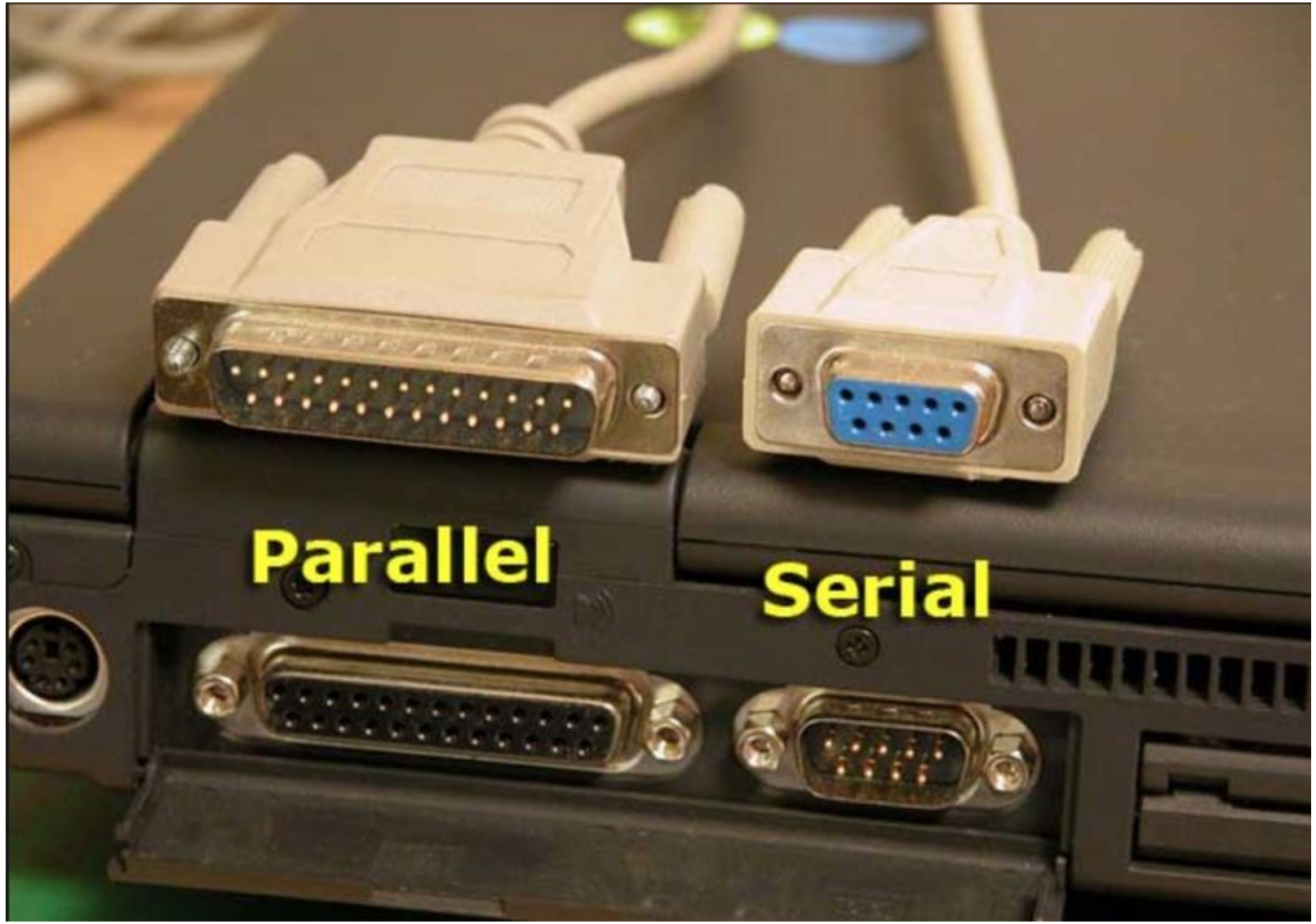
Ideas?

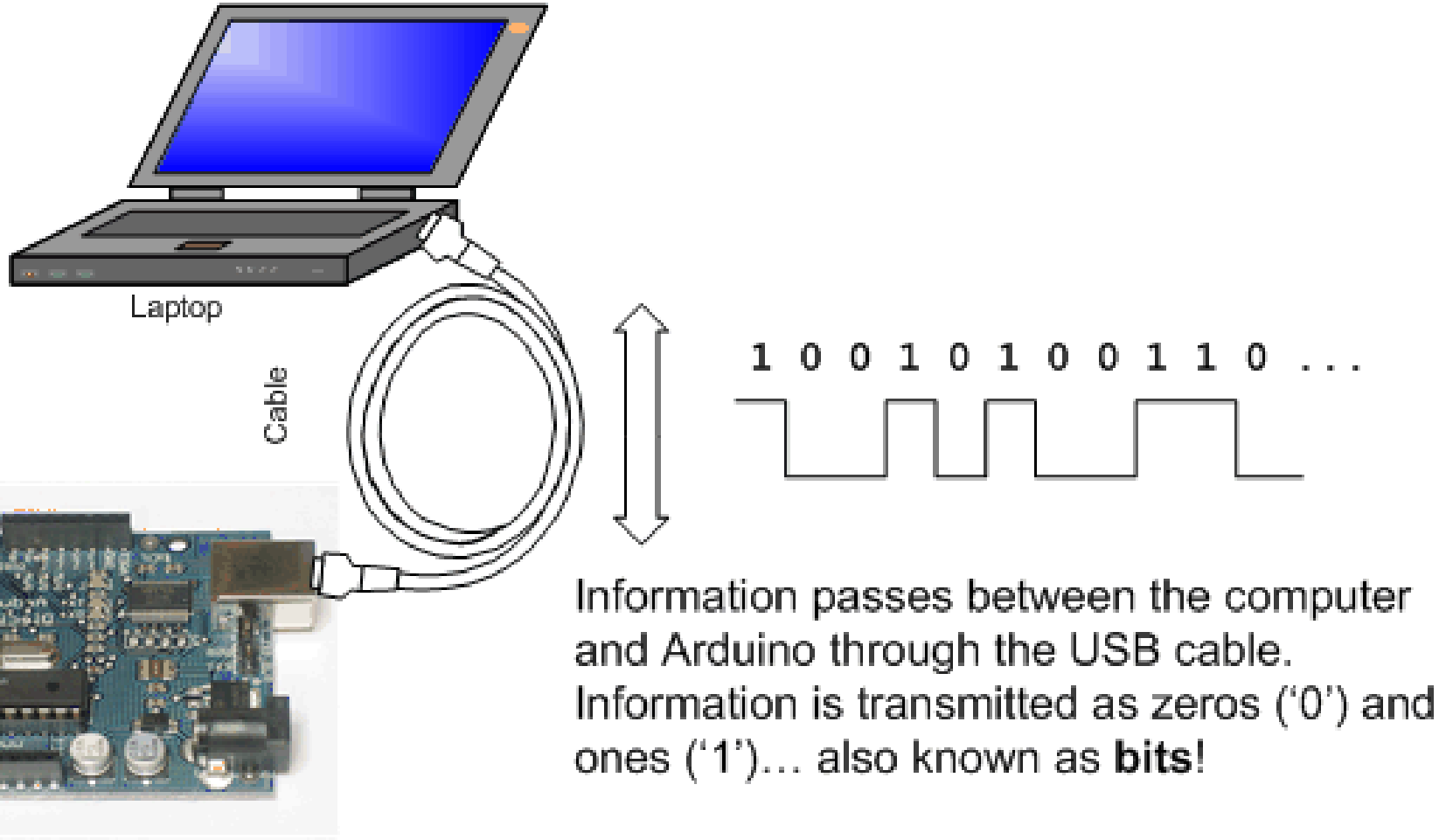
UART(RS232, RS222/RS485(medios industriales), I2C, SPI


T
R
G
Rterm
Stub Length
T
R
G
(Branch Not Recomended)
Branch Length
Stub Length
T
R
G
Rterm
T
R
G
Rterm ?
Bad Idea
T
R
G

PROTOCLO DE COMUNICACIÓN UART

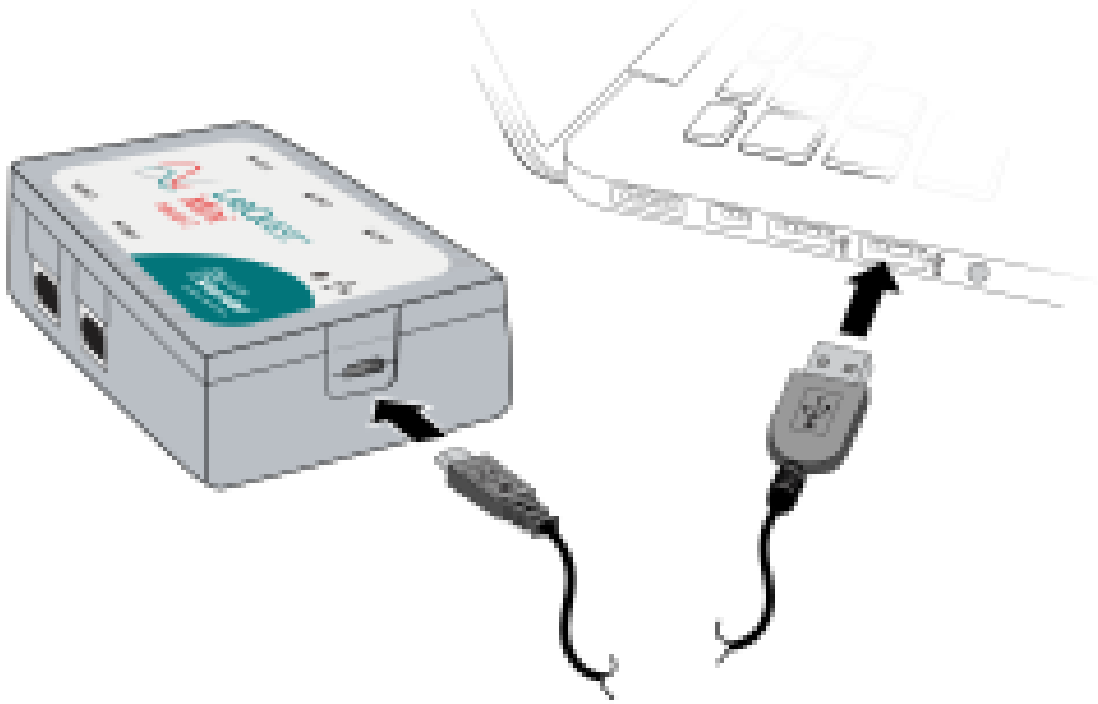



Parallel
Serial

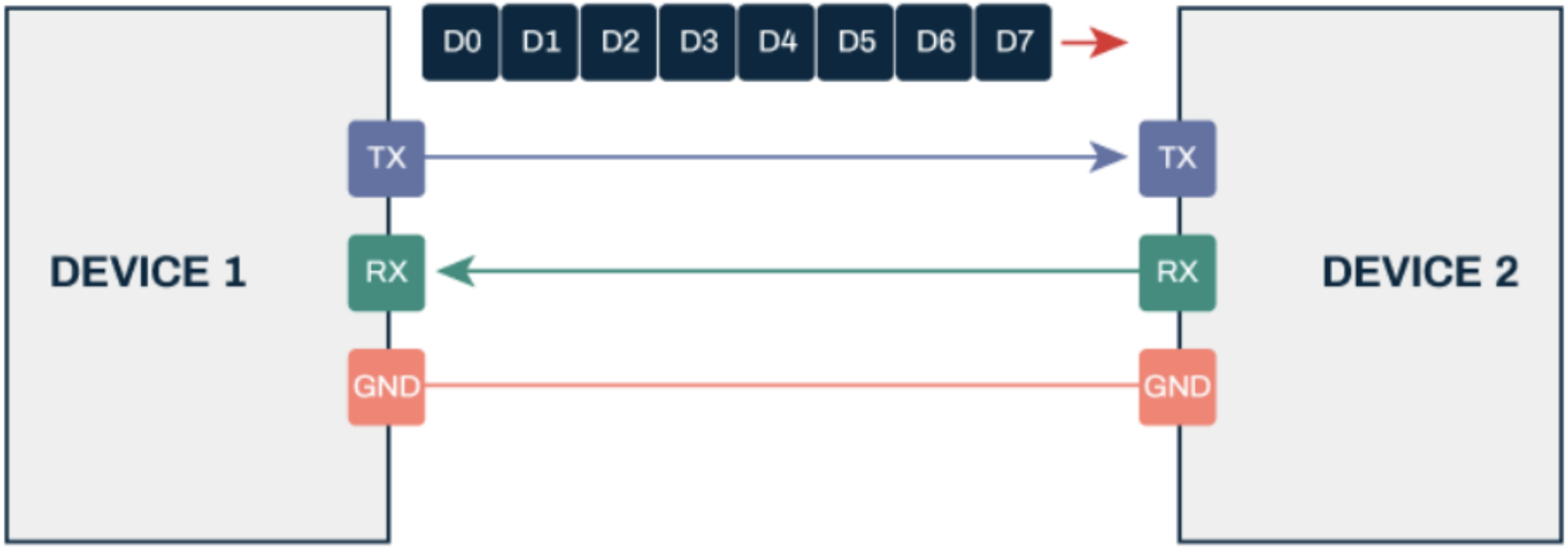


ESTE PROTOCLO PERMITE CONECTAR INTERFACES “Plusg Play” por USB, ya que el puerto serie ya esta descontinuado


Recolección de datos
con un Computador
Software
Sensor
interface para
recolección de datos

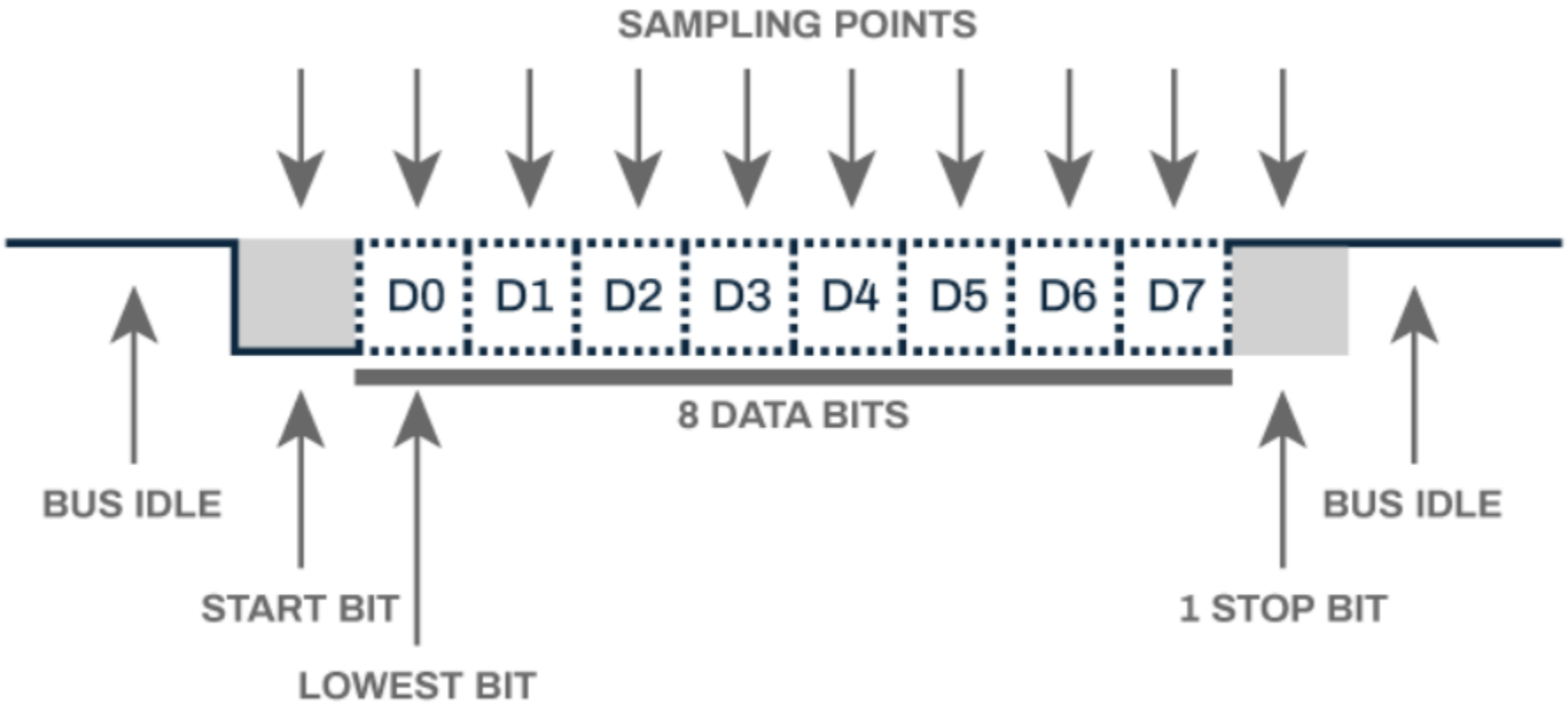


PROTOCLO DE COMUNICACIÓN UART



SE COMPARTEN 2 LINEAS DE TX, RX, y GND

ENVIO DE DATOS EN UART



Librería Serial Arduino

- Arduino tiene una **API** para manejar la comunicación serial, toda vez que se hace uso de dicho periférico en el uC para comunicarse con el puerto serial de la PC

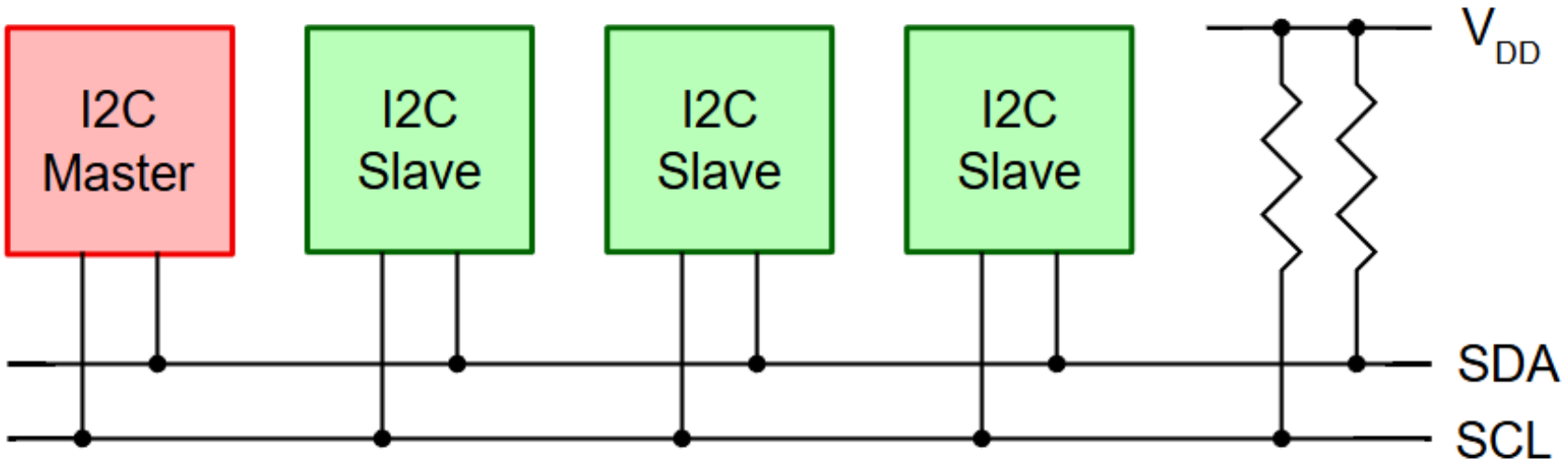
- Arduino utiliza la clase **“Serial” (Objeto→>Serial)**para establecer contactp cpn la PC y evaluar el estado del programa en ejecución (debugging).

- Objeto.método()

- Serial.print(“***Cadena de Texto***”)

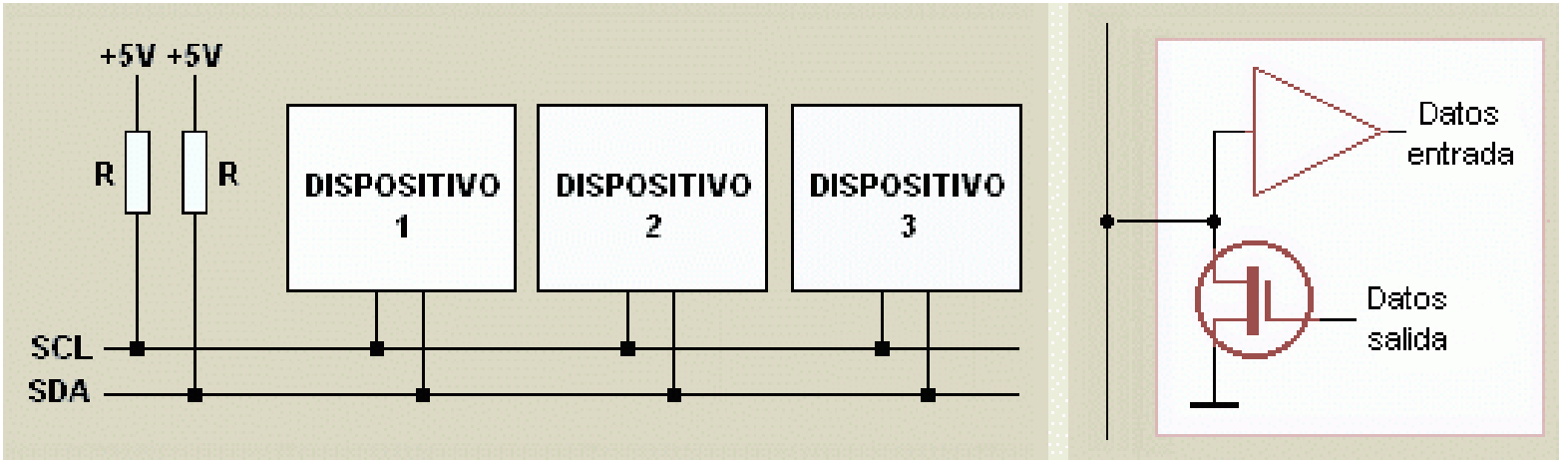
COMUNICACIÓN I2C

- Es un protocolo de serie síncrono que utiliza 2 cables o líneas de comunicación , siendo una de ellas el reloj (**SCL**) y la otra la de datos (**SDA)** conformando ambas un solo bus del que se pueden conectar dispositivos que arrojen datos (**ESCLAVOS**) y quienes soliciten datos(**MAESTRO**)



COMUNICACIÓN I2C

- Los 2 hilos o líneas del bus de comunicación I2C son de tipo drenaje o colector abierto , si se le ve como salida y de tipo buffer ***Zin***, si se le ve como entrada .

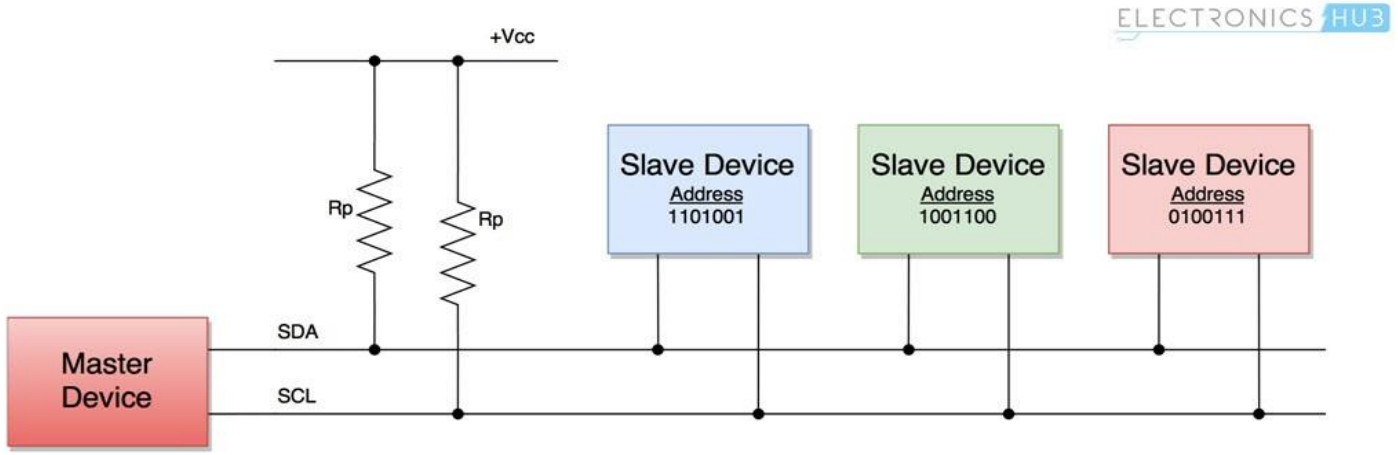


- Por lo que se requiere 2 resistencias de **PULLUP** para asegurar el estado lógico alto o HIGH.

Nota: Algunos dispositivos pueden traerlas incorporada

COMUNICACIÓN I2C

- Los dispositivos que se conecten al I2C (a excepción del Maestro) tienen una dirección **ADDR** (7 bits) única asignada a cada uno.



- Es el **MAESTRO** quien inicia la transferencia de información o datos(bytes-8bits), generando para ello la **señal de reloj o CLK (SCL)** en forma de un tren de ondas cuadradas, valiéndose de la **ADDR** de algún esclavo para poderlo direccionar(enviarle datos- recibir datos)

- Es el **MAESTRO** quien finaliza la comunicación.

COMUNICACIÓN I2C: VELOCIDADES

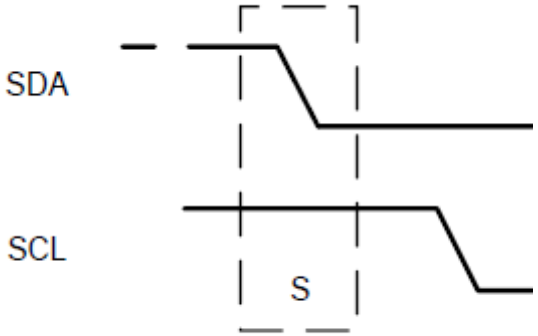
- La velocidad mas usada es la de 100kHz. Se tiene:

- standard mode: 100 kbit/s
- full speed: 400 kbit/s
- fast mode: 1 mbit/s
- ***high speed:*** 3,2 Mbit/s

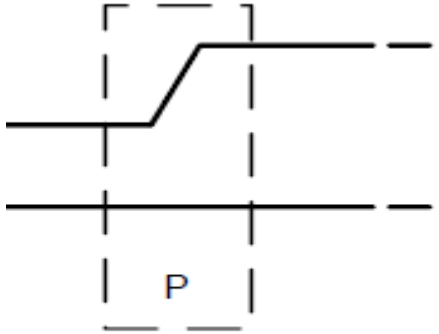
La mayoria de uC soportan las 2 primeras velocidades de reloj.

Funcionamiento comunicación I2C

- El **MAESTRO** inicia la transmisión enviando un bit de inicio **START (S)**, en donde la línea SDA pasa de un estado lógico ALTO a un estado lógico BAJO estando la línea SCL en nivel ALTO(aun no hay tren de pulsos)



START condition



STOP condition

- El **MAESTRO** finaliza la comunicación enviando un bit de **STOP (P)** en donde la línea SDA pasa de un estado lógico BAJO a un estado ALTO estando la línea SCL en nivel ALTO

Funcionamiento comunicación I2C

- Cuando el MAESTRO ya envió el bit de inicio **START (S)** , procede a enviar en sincronía con el pulso de reloj los **7 bits** de la dirección del esclavo con el que se quiere comunicar , seguido de un 1 bit adicional de **(R/W)** que indica **ESCRITURA** o **LECTURA**.

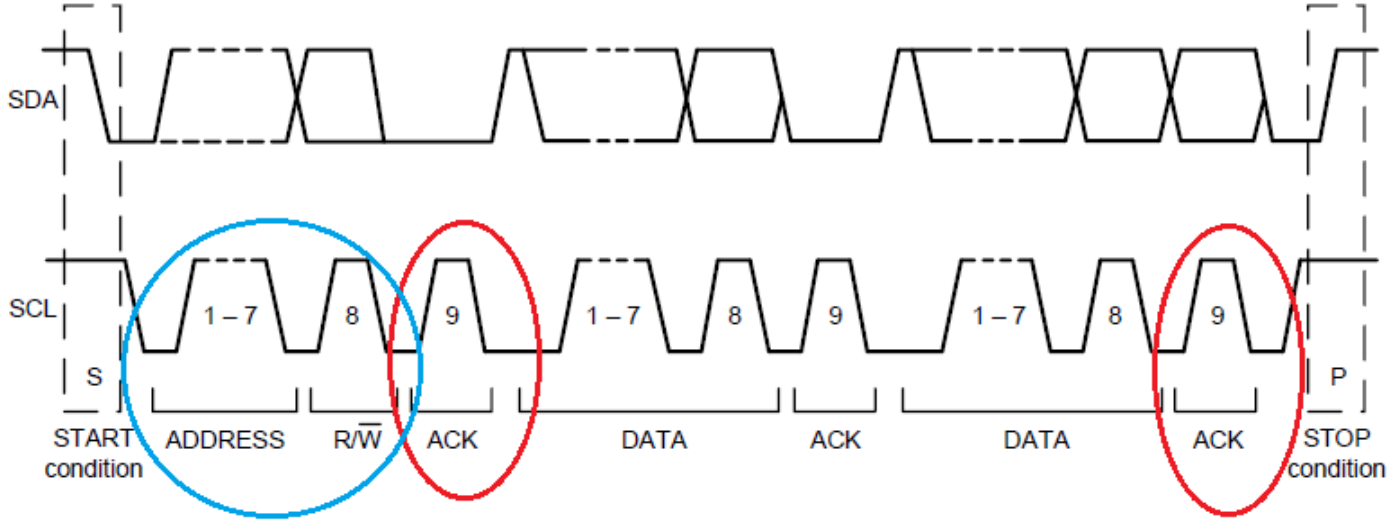
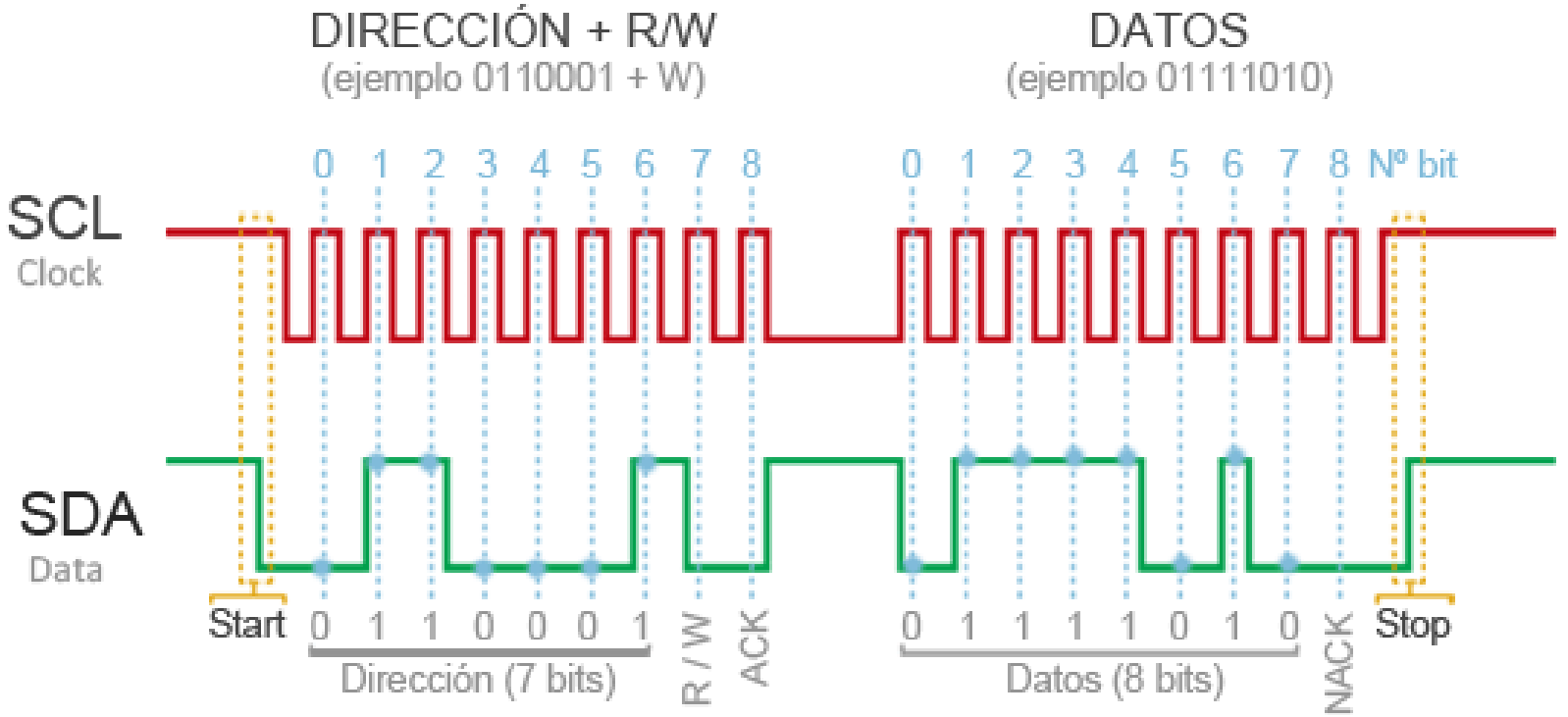


Figura.- Transferencia completa I2C

- Enseguida, el maestro suelta la línea del datos **SDA** (dejándola en ALTO por el **RPU**) y el dispositivo ESCLAVO que tenga grabado la dirección que señalo el **MAESTRO** debe responder con un bit de reconocimiento **(ACK)** haciendo que la línea **SDA** pase a estado lógico **BAJO** solo durante un periodo de ciclo de reloj presente en la línea **SCL**.
- Y los siguiente bytes que se transfieran finalizaran con su respectivo (**ACK)**

Funcionamiento comunicación I2C

- En resumen la transmisión I2C tiene el formato: **| start | A7 A6 A5 A4 A3 A2 A1 R/W | ACK | … DATA … | ACK | stop | idle |**



Funcionamiento comunicación I2C

- Si se ve en osciloscopio digital, veríamos algo como:

Funcionamiento comunicación I2C (Escritura)

- Si ya se transmitió la dirección del esclavo por la línea SDA y se recibió una confinación del **ESCLAVO (ACK)** , lo siguiente a transmitir seria la dirección del puntero del registro**(RPA)** (***Register Pointer Address***) donde se desea **escribir**, recibiendo nuevamente el **(ACK)** del esclavo. El subsiguiente byte que se envíe seria el contenido de dicho **(RPA)** o DATA.

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

Figura .- Escritura multiple en RPA(RA) del esclavo

- Si el **MAESTRO** no emite la condición de **STOP (P)** o detención y sigue enviando mas bytes, entonces el dispositivo **ESCLAVO** incrementa el valor de su (RPA) internamente , escribiendo dichos bytes de **DATA** que aparezcan en la línea de datos **SDA** en los siguientes **(RPA)** o registros

Funcionamiento comunicación I2C (Lectura)

- Para leer el contenido de algún **(RPA)** del **ESCLAVO** se envía la condición **(START)** seguido de la dirección del esclavo + bit de escritura **(W)** mas el **(RPA)** que se va desear leer, cada byte con su respectivo **ACK** por parte del esclavo.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

- Enseguida el **MAESTRO** debe enviar un inicio repetido **(RE-START)** , pero seguido de la dirección del esclavo + bit de lectura **(R)** con su **ACK** por parte de **ESCLAVO** y el byte de data contenido a partir del (RPA)

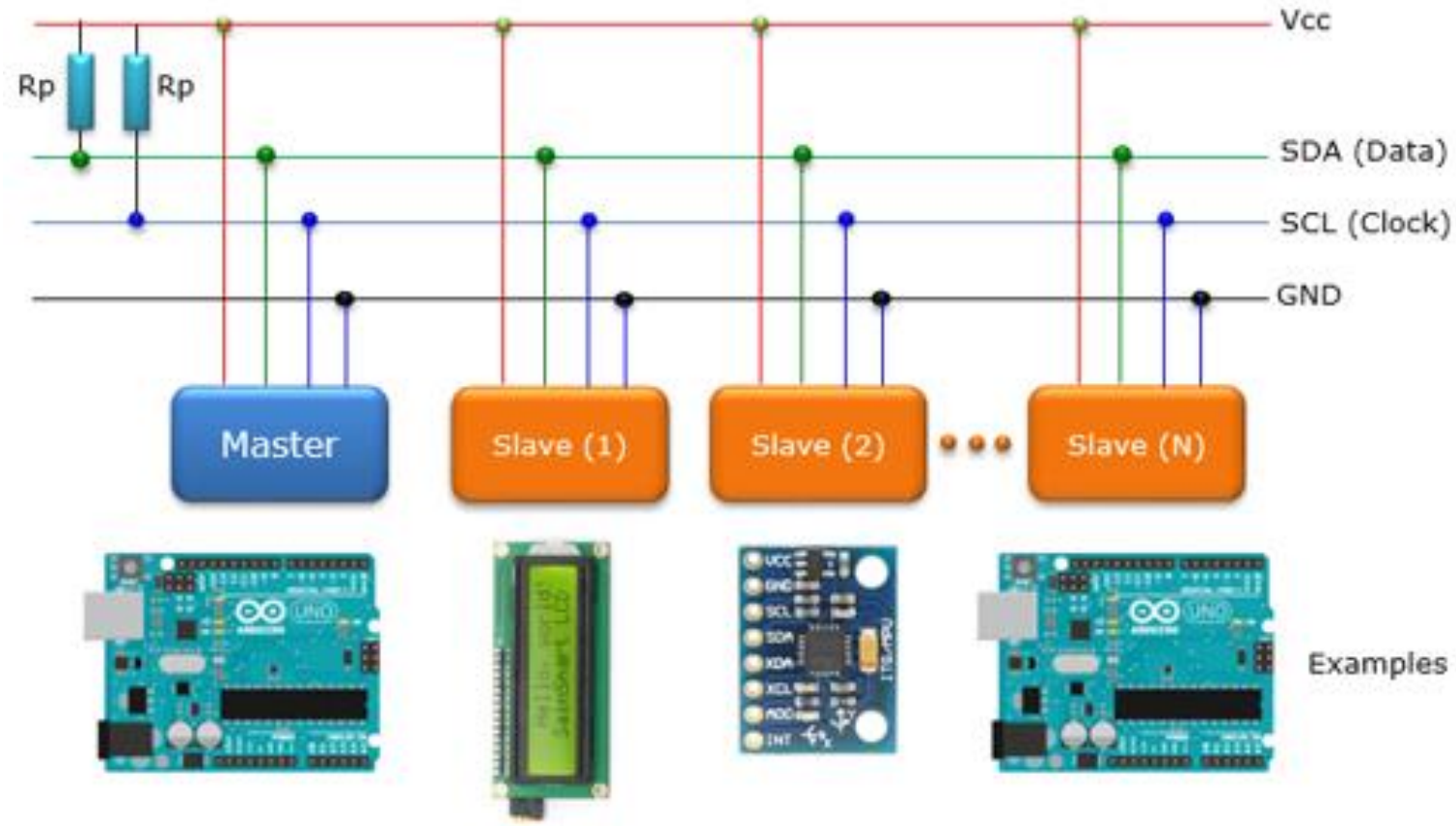
- Es ahora el MAESTRO quien debe enviar (**ACK)** para seguir leyendo los bytes que estén en los **(RPA + i)**.Caso contrario envía **NACK** y da por terminada la comunicación **STOP (P)**

Características I2C en *Atmega328P*

- Interfaz de comunicación sencilla, potente y flexible; solo requiere dos líneas de bus.
- Compatible con operaciones maestro y esclavo.
- El dispositivo puede funcionar como transmisor o receptor.
- Su espacio de direcciones de 7 bits permite hasta 128 direcciones de esclavo diferentes.
- Compatible con arbitraje multimaestro.
- Velocidad de transferencia de datos de hasta 400 kHz.
- Controladores de salida con velocidad de respuesta limitada.
- El circuito de supresión de ruido rechaza picos en las líneas de bus.
- Dirección de esclavo totalmente programable con compatibilidad con llamadas generales.
- El reconocimiento de dirección activa el AVR cuando está en modo de suspensión.
- Compatible con el protocolo I^2C de Philips.

¿Cómo utilizar I2C en Arduino?

- Se debe seguir el siguiente esquema de hw topologico



¿Cómo utilizar I2C en Arduino?

- La biblioteca para manejar la comunicación I2C desde Arduino se llama ***Wire***, la misma que es una clase que se puede importar directamente como

```
#include <Wire.h>
```

Funciones:

- **begin()** – Inicia la librería Wire y especifica si es master o slave
- **requestFrom()** – **Usado por el maestro para solicitar datos** del esclavo
- **beginTransmission(arg Slave-ADDR)** – Comenzar transmisión con esclavo.
- **endTransmission()** – Finaliza la transmisión que comenzó con un esclavo **y transmite (escribe) los bytes en cola**.
- **write()** – Escribe datos hacia/desde un esclavo como respuesta a una petición del maestro o pone en cola la transmisión de un maestro.
- **available()** – Devuelve el número de bytes para leer
- **read()** – Lee un byte transmitido desde un esclavo a un maestro o viceversa

01

EJEMPLO

Actividad de aprendizaje

Dado un dispositivo **ESCLAVO** del tipo sensor conectado al bus I2C de la plataforma Arduino mediante los **pines (SDA) y (SCL), determinar la dirección SLAVE ADDR** a la que este responde mediante un sketch:

Solución: Ver ***WokWI*** online

NOTA: La solución presentada es valida para determinar la dirección de cualquier dispositivo(s) I2C conectado

¿Qué sensores utilizan I2C?

Ideas?

¿Qué sensores utilizan I2C?

*Sensores **IMU**, ADC externos, Magnetometros, giróscopos, panatllas LCD*
Epansores IO de entrdas y salidas digitales

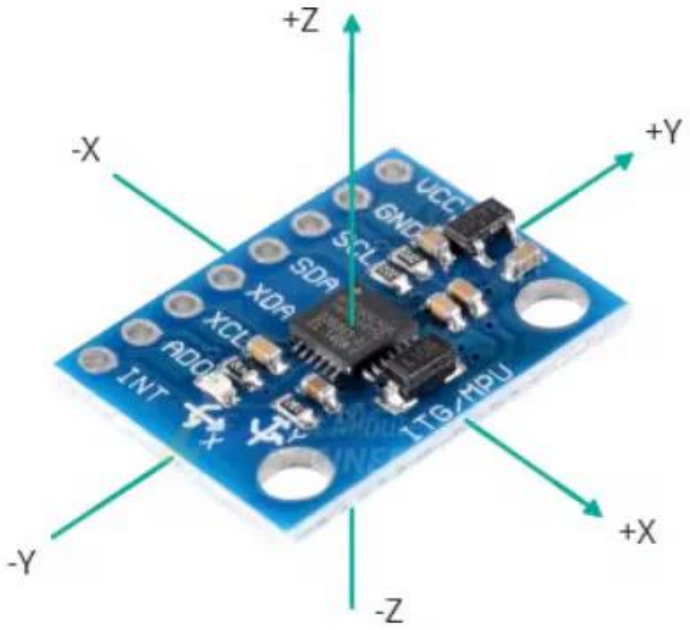
IMU (Inertial Measurement Unit)

- Una unidad de medición inercial es capaz de medir la aceleración lineal y velocidad angular por lo que puede servir para calcular desplazamientos lineales o curvos. Se caracterizan por su **(DOF)** o grados de libertad.

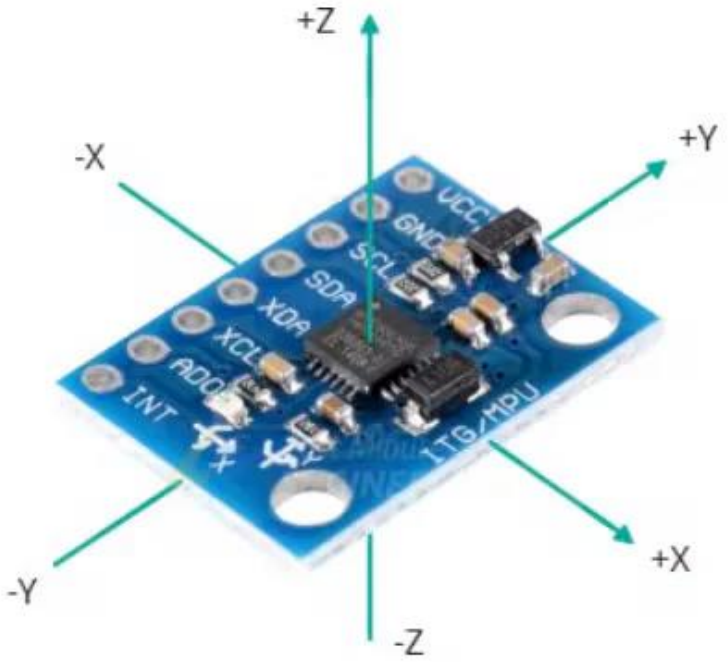
- Uno de ellos es el ***MPU6050*** presente en el modulo enchufable ***GY-521***, el mismo que consta de acelerómetro y giróscopo de 6 DOF o ejes, ***aplicable en la medición de movimientos***.

Sensor MPU6050

Medición de aceleración



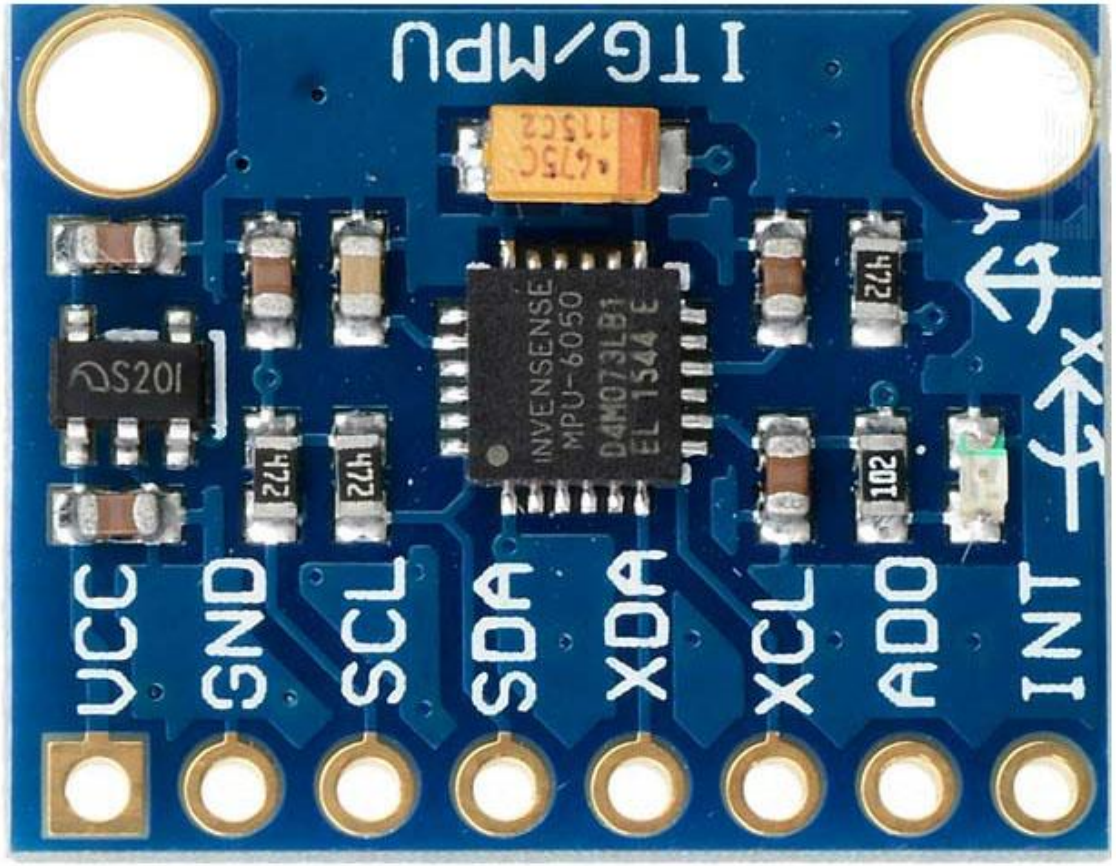
Medición de rotación



Sensor MPU6050: Características

- Fuente de alimentación: 3-5V
- Combinación de valores de acelerómetro MEMS de 3 ejes y giroscopio de 3 ejes
- Comunicación: protocolo I2C
- El **ADC** de 16 bits incorporado proporciona alta precisión
- El **DMP** incorporado proporciona una alta potencia computacional
- Se puede utilizar para **interactuar con otros dispositivos I2C** como un magnetómetro(externo) a través de su **bus auxiliar: XSDA, XSCL**
- Dirección I2C configurable
- Sensor de temperatura incorporado

Sensor MPU6050 (detalle)



Sensor MPU6050: Características

Giroscopo

- Sensores de velocidad angular (giroscopios) con salida digital en los ejes X, Y y Z, con un rango de escala completa programable por el usuario de ±250, ±500, ±1000 y ±2000°/s.

- Los convertidores analógico-digitales (ADC) de 16 bits integrados permiten el muestreo simultáneo de giroscopios.

- La estabilidad mejorada de la temperatura de polarización y sensibilidad reduce la necesidad de calibración por parte del usuario.
- Mejora del rendimiento frente al ruido de baja frecuencia.
- Filtro paso bajo **(suprimir ruidos)** programable digitalmente.
- Factor de escala de sensibilidad calibrado de fábrica.

Sensor MPU6050: Características

Acelerometro

- Acelerómetro de triple eje con salida digital y un rango programable de escala completa de **±2 g (+-19.4/ms^2), ±4 g, ±8 g y ±16 g.**

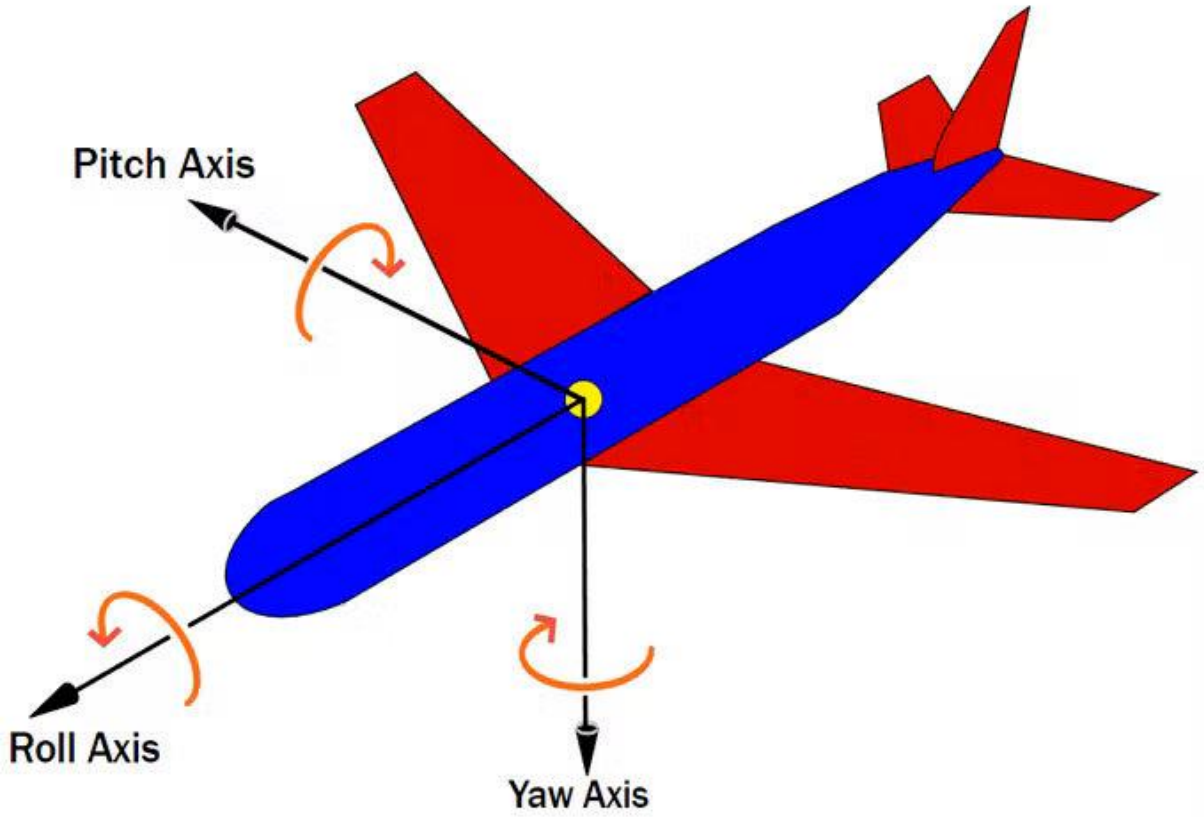
- Los convertidores analógico-digitales (ADC) de 16 bits integrados permiten el muestreo simultáneo de acelerómetros sin necesidad de un multiplexor externo.

- Detección de orientación y señalización.
- Detección de toques.
- Interrupciones programables por el usuario.

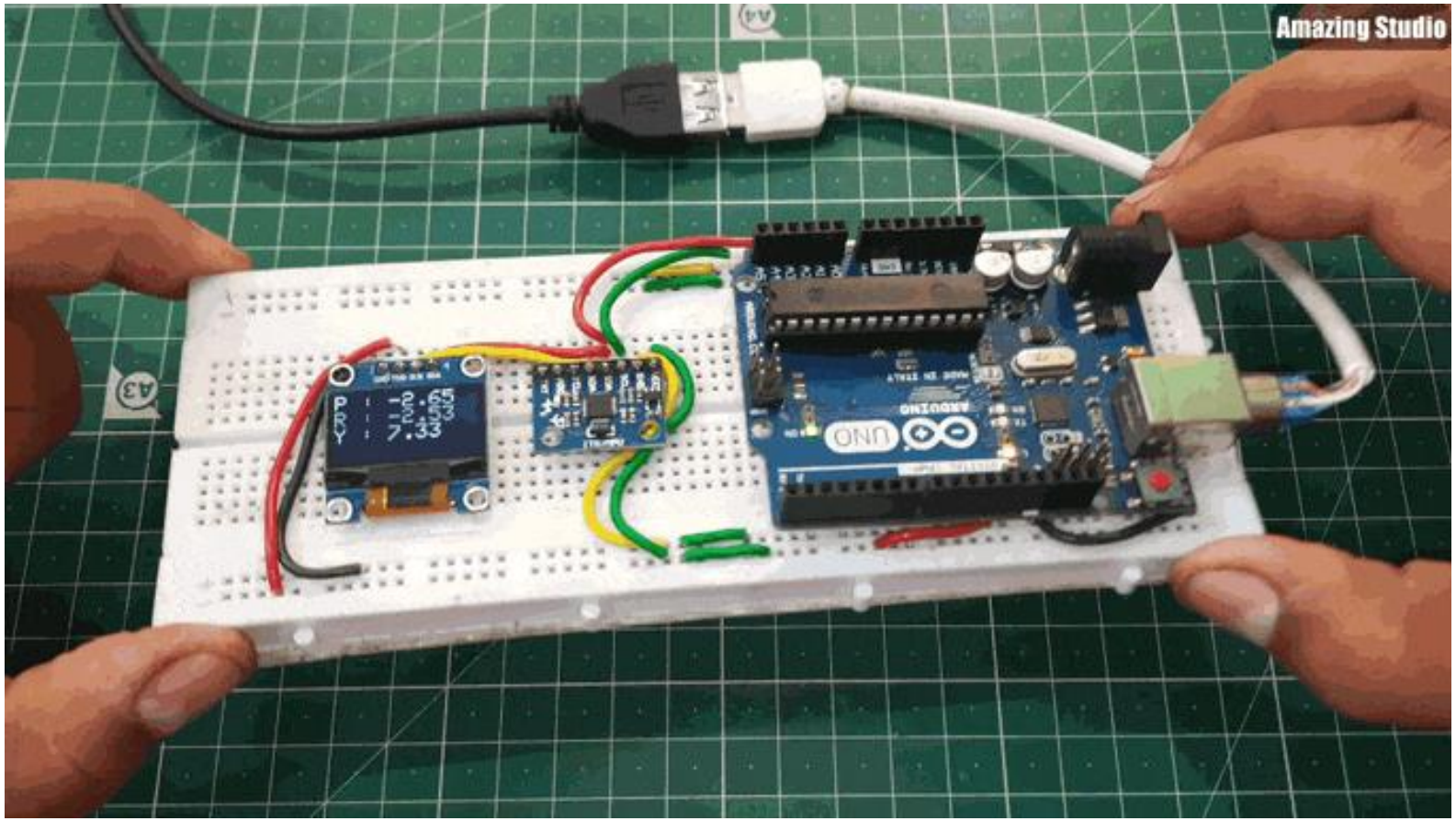
Sensor MPU6050: Aplicaciones

- Detección y control de movimiento robótico
- Tareas de localización y mapeo robótico
- **Tecnología MotionCommand™ (para atajos de gestos)**
- **Marco de juegos y aplicaciones basado en movimiento**
- **Reconocimiento de gestos InstantGesture™ iG™**
- Servicios basados en ubicación, puntos de interés y navegación a estima
- Juegos móviles y portátiles
- Controladores de juegos basados en movimiento
- Controles remotos 3D para televisores y decodificadores conectados a internet, ratones 3D, etc.
- Sensores portátiles para salud, fitness y deportes
- Juguetes

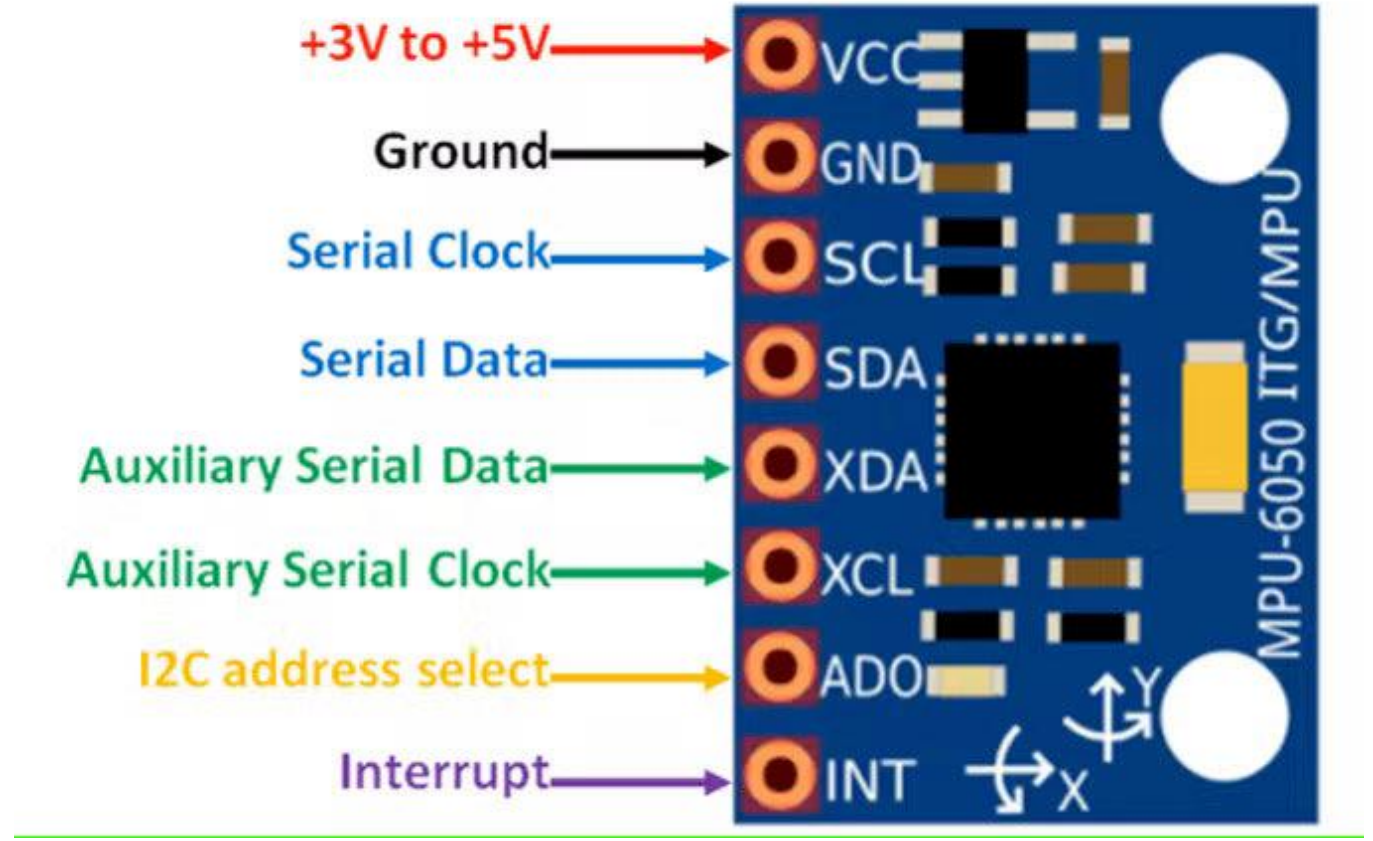
Sensor MPU6050: aplicacion



Sensor MPU6050: uso

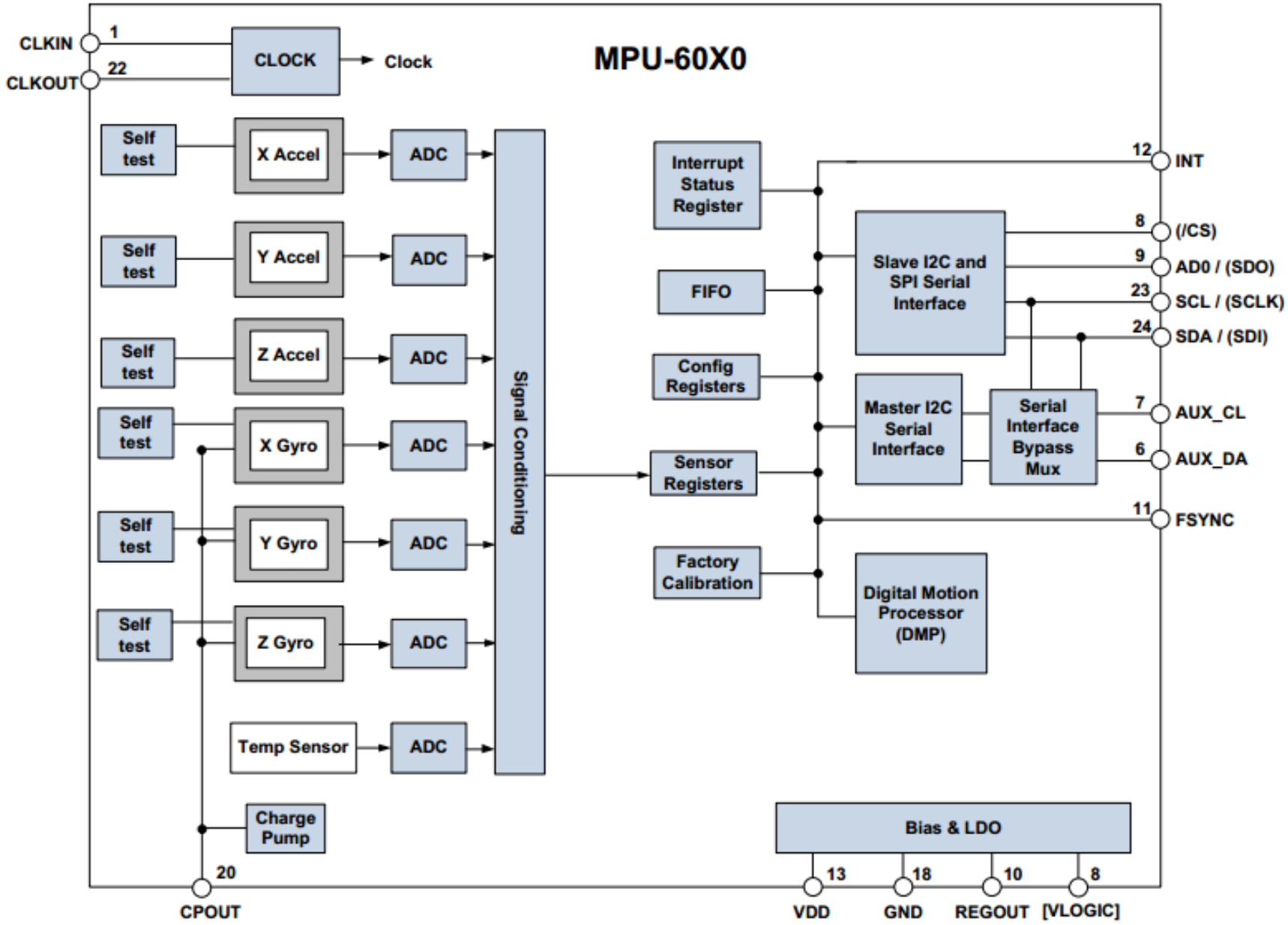


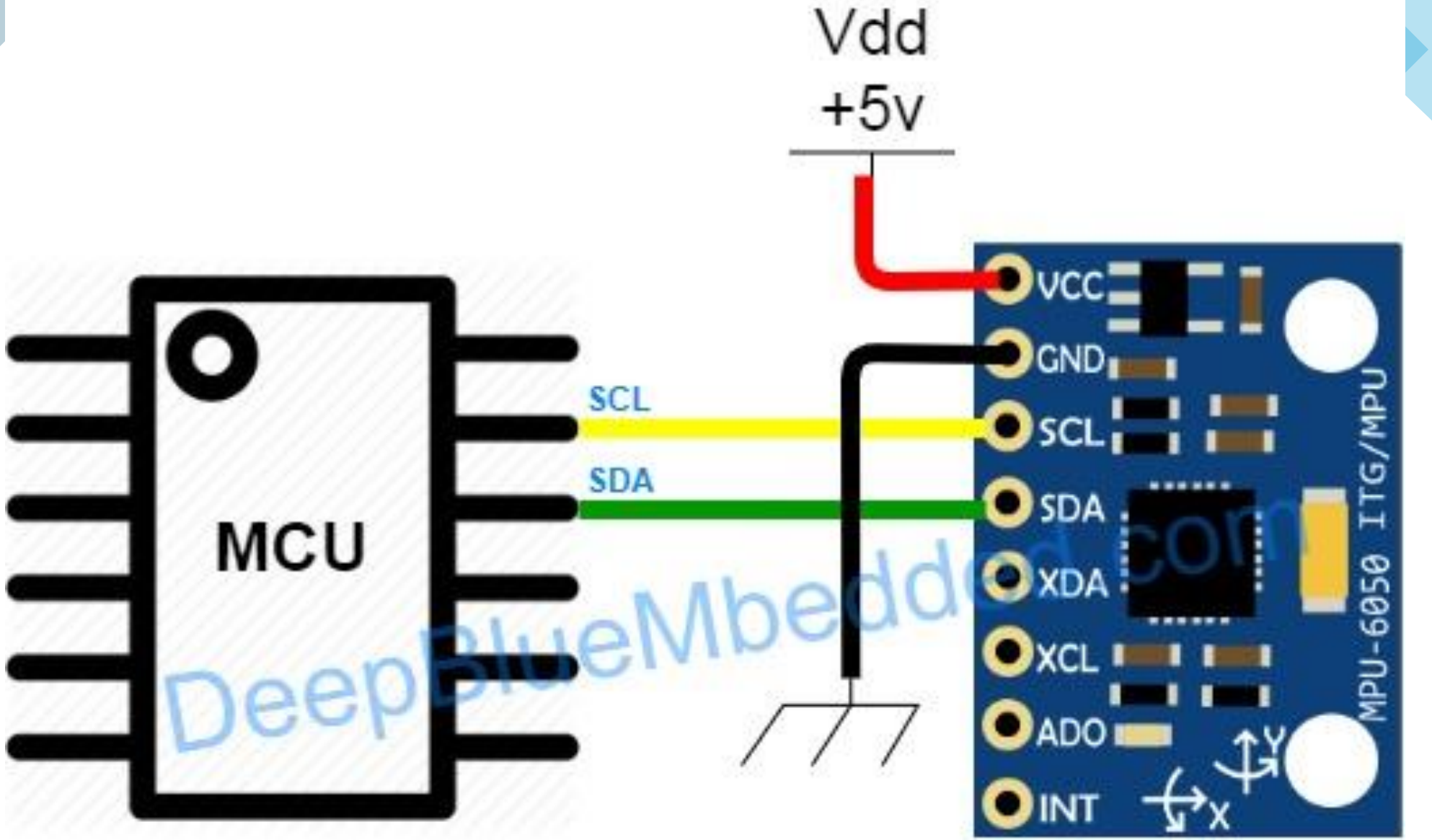
Sensor MPU6050(diagrama de pines)



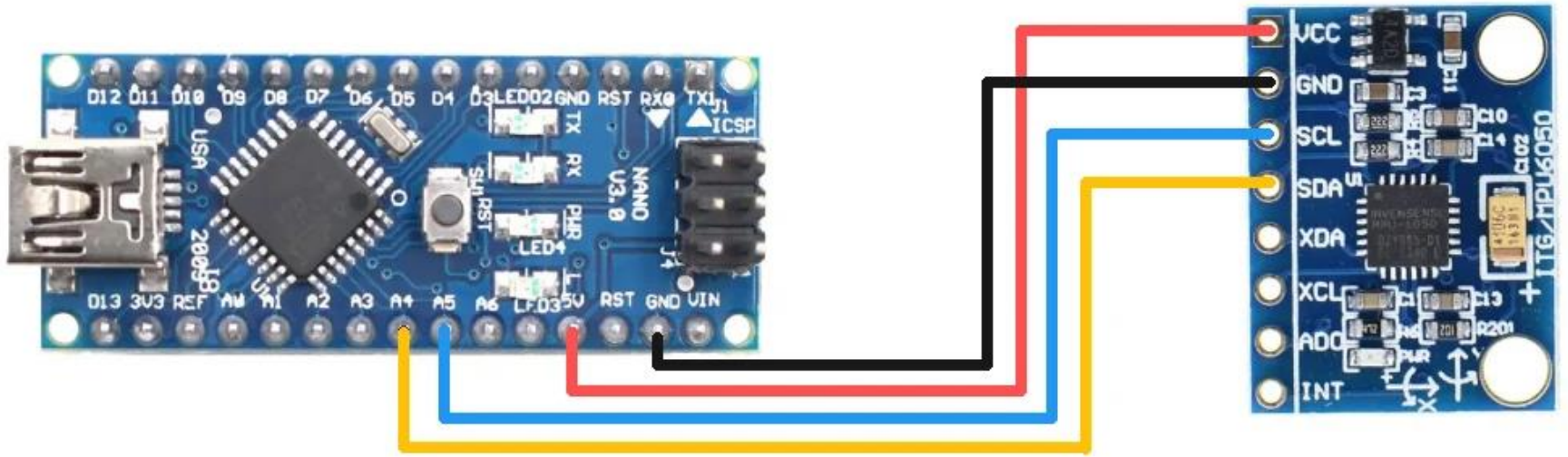
El pin AD0 puede modificar la dirección del esclavo ADDR.

Sensor MPU6050: Block Diagram





Conectar y Usar el Módulo GY-521 (MPU-6050) con Arduino



Pin GY-521	Pin Arduino UNO	Descripción
VCC	5V o 3.3V	Alimentación del módulo
GND	GND	Tierra
SDA	A4	Línea de datos I2C
SCL	A5	Línea de reloj I2C
INT	No conectado	Pin de interrupción (opcional)

Librería: Aqui

Sensor MPU6050: Mapa de Registros Ver "*Register-Map-Sheet*"

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			

Sensor MPU6050: Mapa de Registros

Ver "*Register-Map-Sheet*"

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Sensor MPU6050: Mapa de Registros

Ver "*Register-Map-Sheet*"

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

SENSOR: MPU6050

REGISTROS de Configuración: CFG

Register 25 – Sample Rate Divider
SMPRT_DIV

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

This register specifies the divider from the gyroscope output rate used to generate the Sample Rate for the MPU-60X0.

The sensor register output, FIFO output, and DMP sampling are all based on the Sample Rate.

The Sample Rate is generated by dividing the gyroscope output rate by *SMPLRT_DIV*:

Sample Rate = Gyroscope Output Rate / (1 + *SMPLRT_DIV*)

where Gyroscope Output Rate = 8kHz when the DLPF is disabled (*DLPF_CFG* = 0 or 7), and 1kHz when the DLPF is enabled (see Register 26).

Note: The accelerometer output rate is 1kHz. This means that for a Sample Rate greater than 1kHz, the same accelerometer sample may be output to the FIFO, DMP, and sensor registers more than once.

Register 107 – Power Management 1
PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Description:

This register allows the user to configure the power mode and clock source. It also provides a bit for resetting the entire device, and a bit for disabling the temperature sensor.

By setting *SLEEP* to 1, the MPU-60X0 can be put into low power sleep mode. When *CYCLE* is set to 1 while *SLEEP* is disabled, the MPU-60X0 will be put into Cycle Mode. In Cycle Mode, the device cycles between sleep mode and waking up to take a single sample of data from accelerometer at a rate determined by *LP_WAKE_CTRL* (register 108). To configure the wake frequency, use *LP_WAKE_CTRL* within the Power Management 2 register (Register 108).

An internal 8MHz oscillator, gyroscope based clock, or external sources can be selected as the MPU-60X0 clock source. When the internal 8 MHz oscillator or an external source is chosen as the clock source, the MPU-60X0 can operate in low power modes with the gyroscopes disabled.

Upon power up, the MPU-60X0 clock source defaults to the internal oscillator. However, it is highly recommended that the device be configured to use one of the gyroscopes (or an external clock source) as the clock reference for improved stability. The clock source can be selected according to the following table.

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference

Register 26 – Configuration

CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Description:

This register configures the external Frame Synchronization (FSYNC) pin sampling and the Digital Low Pass Filter (DLPF) setting for both the gyroscopes and accelerometers.

An external signal connected to the FSYNC pin can be sampled by configuring *EXT_SYNC_SET*.

Signal changes to the FSYNC pin are latched so that short strobes may be captured. The latched FSYNC signal will be sampled at the Sampling Rate, as defined in register 25. After sampling, the latch will reset to the current FSYNC signal state.

The sampled value will be reported in place of the least significant bit in a sensor data register determined by the value of *EXT_SYNC_SET* according to the following table.

EXT_SYNC_SET	FSYNC Bit Location
0	Input disabled
1	TEMP_OUT_L[0]
2	GYRO_XOUT_L[0]
3	GYRO_YOUT_L[0]
4	GYRO_ZOUT_L[0]
5	ACCEL_XOUT_L[0]
6	ACCEL_YOUT_L[0]
7	ACCEL_ZOUT_L[0]

Register 26 – Configuration

CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

The DLPF is configured by *DLPF_CFG*. The accelerometer and gyroscope are filtered according to the value of *DLPF_CFG* as shown in the table below.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Description:

This register is used to trigger accelerometer self test and configure the accelerometer full scale range. This register also configures the Digital High Pass Filter (DHPF).

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. The self-test for each accelerometer axis can be activated by controlling the *XA_ST*, *YA_ST*, and *ZA_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

Description:

This register is used to trigger accelerometer self test and configure the accelerometer full scale range. This register also configures the Digital High Pass Filter (DHPF).

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. The self-test for each accelerometer axis can be activated by controlling the *XA_ST*, *YA_ST*, and *ZA_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

FS_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

SENSOR: MPU6050

REGISTROS de Lectura de valores

4.17 Registers 59 to 64 – Accelerometer Measurements

ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, and ACCEL_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Description:

These registers store the most recent accelerometer measurements.

Accelerometer measurements are written to these registers at the Sample Rate as defined in Register 25.

The accelerometer measurement registers, along with the temperature measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the accelerometer sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit accelerometer measurement has a full scale defined in *ACCEL_FS* (Register 28). For each full scale setting, the accelerometers' sensitivity per LSB in *ACCEL_xOUT* is shown in the table below.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Parameters:

ACCEL_XOUT 16-bit 2's complement value.

Stores the most recent X axis accelerometer measurement.

ACCEL_YOUT 16-bit 2's complement value.

Stores the most recent Y axis accelerometer measurement.

ACCEL_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis accelerometer measurement.

4.18 Registers 65 and 66 – Temperature Measurement TEMP_OUT_H and TEMP_OUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
41	65	TEMP_OUT[15:8]							
42	66	TEMP_OUT[7:0]							

Description:

These registers store the most recent temperature sensor measurement.

Temperature measurements are written to these registers at the Sample Rate as defined in Register 25.

These temperature measurement registers, along with the accelerometer measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the temperature sensor's internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

The scale factor and offset for the temperature sensor are found in the Electrical Specifications table (Section 6.4 of the MPU-6000/MPU-6050 Product Specification document).

The temperature in degrees C for a given register value may be computed as:

Temperature in degrees C = (TEMP_OUT Register Value as a signed quantity)/340 + 36.53

4.19 Registers 67 to 72 – Gyroscope Measurements GYRO_XOUT_H, GYRO_XOUT_L, GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H, and GYRO_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

Description:

These registers store the most recent gyroscope measurements.

Gyroscope measurements are written to these registers at the Sample Rate as defined in Register 25.

These gyroscope measurement registers, along with the accelerometer measurement registers, temperature measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the gyroscope sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

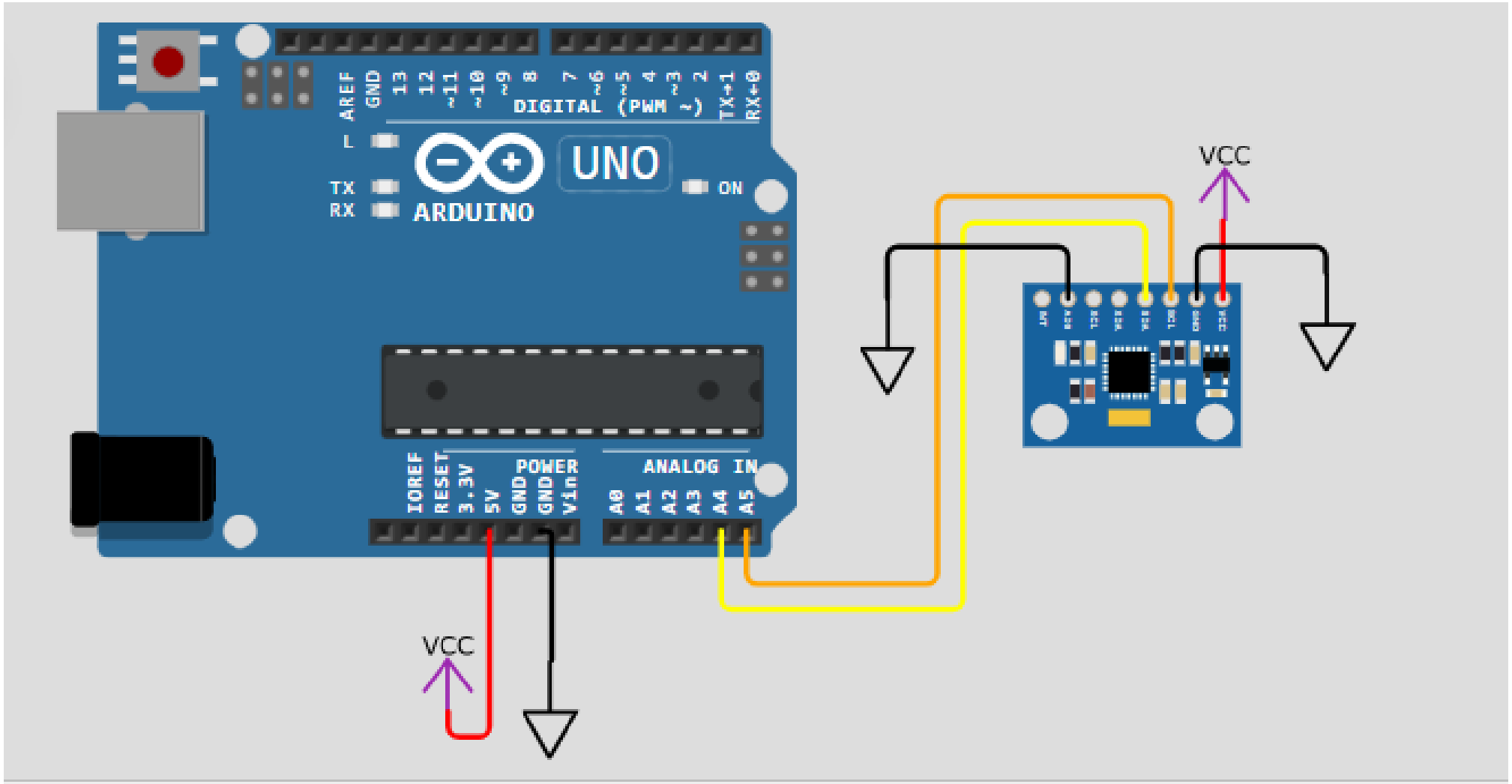
Each 16-bit gyroscope measurement has a full scale defined in *FS_SEL* (Register 27). For each full scale setting, the gyroscopes' sensitivity per LSB in *GYRO_xOUT* is shown in the table below:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

PROYECTOS

Leer y analizar la data de aceleración lineal (m/s^2) y Velocidad angular(°/s)

CONEXIONES CIRCUITO SENSOR IMU: MPU6050



Nota: Usar la librería WIRE de Arduino. No requiere descargar

Actividad de análisis

El diseño ahora es hacer hacer la lectura del sensor IMU MPU6050 presente en el modulo GY-521

- Considere un Sample Rate de 1kHz en el registro **SMPLRT_DIV**
- Considere el ancho de banda BW del Acelerometro de 260Hz, delay de 0ms y su muestro constante de 1kHz en el registro **CONFIG**
- Considere el ancho de banda BW del Giroscopo de 256Hz, delay de 0.98ms y su Fs=8kHz en el registro **CONFIG**

Actividad de análisis

El diseño ahora es hacer hacer la lectura del sensor IMU MPU6050 presente en el modulo GY-521

- Escoga algún rango de velocidad angular en el registro **GYRO_CONFIG**

- Escoga algún rango de aceleración en el registro **ACCEL_CONFIG**

OBS: Recuerde encender el modulo TURN ON configurando primero el registro: PWR_MGMT_1

Actividad de análisis

Construir **funciones(C)** y luego una librería que empaquete las siguientes funciones prototipo que ayudaran en la lectura del sensor MPU050

```
extern void MPU6050_Init(void){…
}

extern void MPU6050_ReadRaw(int16_t *Ax, int16_t *Ay, int16_t *Az, int16_t *Temp, int16_t *Gx, int16_t *Gy, int16_t *Gz){


.

extern void MPU6050_ReadScaled(float *AX, float *AY, float *AZ, float *TEMP, float *GX, float *GY, float *GZ){…
//Temperature in format °C…
}

extern bool MPU_6050_Test_If_Present(void){…
```