PROYECTO FINAL

**Arquitectura de Sistema de Elaboración de datos II**

**2019**

PROYECTO FINAL

Contenido

[OBJETIVO 4](#_Toc437551509)

[PRESENTACION 4](#_Toc437551510)

[MODO DE FUNCIONAMIENTO 4](#_Toc437551511)

[DESARROLLO TEORICO. 5](#_Toc437551512)

[Elección de Microcontrolador – Arduino Uno 5](#_Toc437551513)

[Conversor A/D 8](#_Toc437551514)

[Comunicación por Puerto Serial: 9](#_Toc437551515)

[Comunicación serial 9](#_Toc437551516)

[Salidas PWM 10](#_Toc437551517)

[Asociación de timers y PWM 11](#_Toc437551518)

[Frecuencia del PWM 11](#_Toc437551519)

[SECCIÓN 1: 13](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551520)

[Acelerómetro 13](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551522)

[ACELEROMETRO 14](#_Toc437551523)

[¿Qué es un acelerómetro? 14](#_Toc437551524)

[Interfaz de comunicaciones 18](#_Toc437551525)

[Potencia 19](#_Toc437551526)

[Otras características 19](#_Toc437551527)

[Conexión del módulo GY-80 20](#_Toc437551528)

[SECCIÓN 1: 22](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551529)

[Servomotores 22](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551531)

[Servomotor 23](#_Toc437551532)

[¿Qué es un servomotor? 23](#_Toc437551533)

[Partes de un servomotor 23](#_Toc437551534)

[Diagrama en bloques 24](#_Toc437551535)

[Funcionamiento 24](#_Toc437551536)

[SECCIÓN 3: 26](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551538)

[Módulo nRF24L01 26](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551539)

[Módulo nRF24L01 27](#_Toc437551540)

[Funcionamiento básico 27](#_Toc437551541)

[Conexión 28](#_Toc437551542)

[Acelerómetro + Servos + Transceptores 29](#_Toc437551543)

[Algoritmo 30](#_Toc437551544)

[33](#_Toc437551545)

[SECCIÓN 4: 33](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551547)

[Sensores Flexibles 33](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551548)

[33](#_Toc437551549)

[Análisis de Bloque 1 34](#_Toc437551550)

[Sensores Flexibles 35](#_Toc437551551)

[Creación de Resistencia Variable. 35](#_Toc437551552)

[Bibliografía 40](file:///C:\Users\sol\Desktop\ARQUITECTURASSSSSSSSSSSSSSSSSSS\Proyecto%20Final%20Arquitecturas%202015.docx#_Toc437551554)

# OBJETIVO

P

resentar a través de un proyecto basado en Arduino, las temáticas que se brindaron a lo largo de la cursada de Arquitectura de Sistema de Elaboración de datos II, logrando mediante la praxis que el alumnado ponga en práctica los conocimientos teóricos adquiridos durante el trayecto del cuatrimestre.

# PRESENTACION

El proyecto realizado por los alumnos se basa en un brazo robótico controlado esencialmente por:

* Sensores flexibles creados por los estudiantes
* Servos (Servo Tower Pro Sg90 9g)
* Acelerómetro
* Arduino

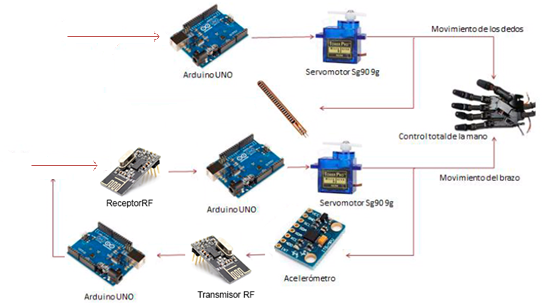
**Fig. 1** Proyecto Finalizado

# MODO DE FUNCIONAMIENTO

El modo de funcionamiento se basa en un guante que posee sensores flexibles, los cuales actúan como una resistencia variable. Por esto último, se puede controlar la tensión que tiene cada uno, leyendo el valor analógico a través de los pines de Arduino. Al contar con estos valores, se puede controlar el movimiento de los dedos, estableciendo parámetros, obtenidos mediante un promedio. Una vez establecidos los valores, se envían órdenes para el movimiento de los servos.

Además de controlar los dedos de la mano, se puede realizar un control de lo que es el antebrazo por medio del giroscopio y el acelerómetro.

**Diagrama en bloques general**



**Fig.2** Diagrama en Bloques general del Proyecto

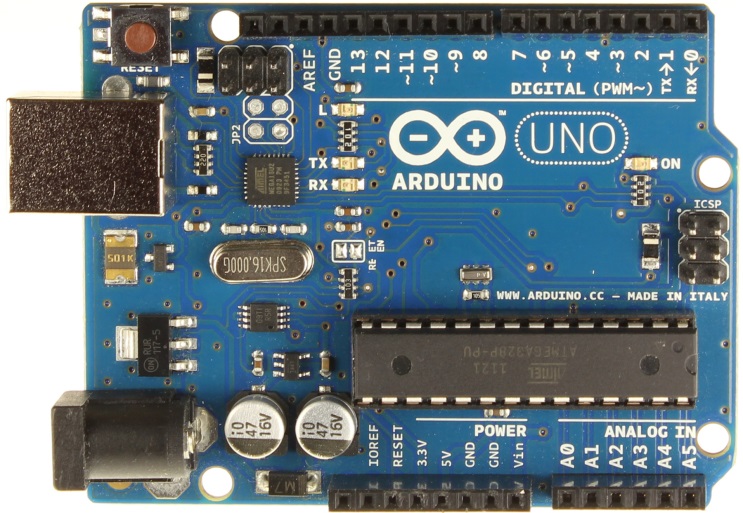
# 

# DESARROLLO TEORICO.

A través de la cursada, se estudiaron los distintos tipos de microcontrolador, pero se hizo énfasis en dos tipos: aquellos que pertenecen a la familia PIC (fabricados por *Microchip Technology*) y por otro lado, la familia de AVR (del fabricante estadounidense *Atmel*). Se debe tener en cuenta que sea analizaron las distintas ventajas de cada uno de estos microcontroladores, el proyecto era libre en cuanto a la elección de cualquiera de estos microcontroladores.

## Elección de Microcontrolador – Arduino Uno

Los estudiantes se basaron en la elección de una plataforma de hardware libre, conocida como ***Arduino,*** basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. El hardware consiste en una placa con un microcontrolador *Atmel AVR* y puertos de entrada/salida. Dentro de Arduino, existen varios productos, para este proyecto se utilizaron dos *Arduino Uno*, uno para el control de los servos (que manejan los dedos, a través del control de los sensores flexibles), y otro para el manejo del antebrazo (acelerómetro).



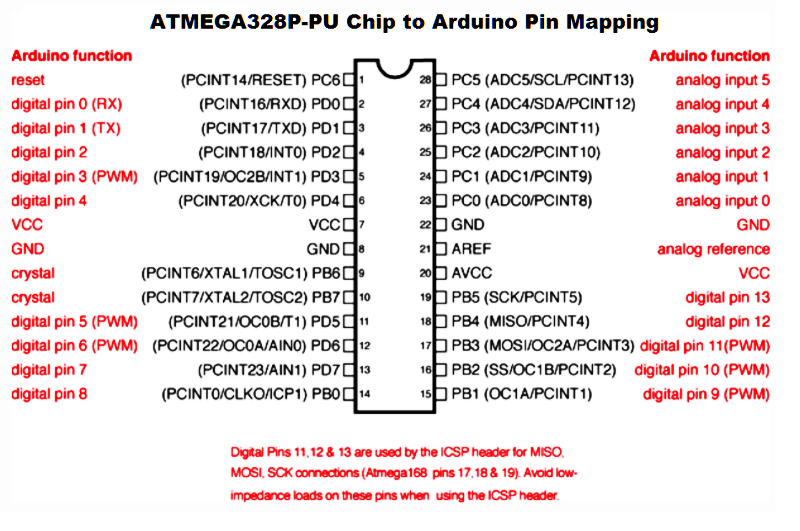
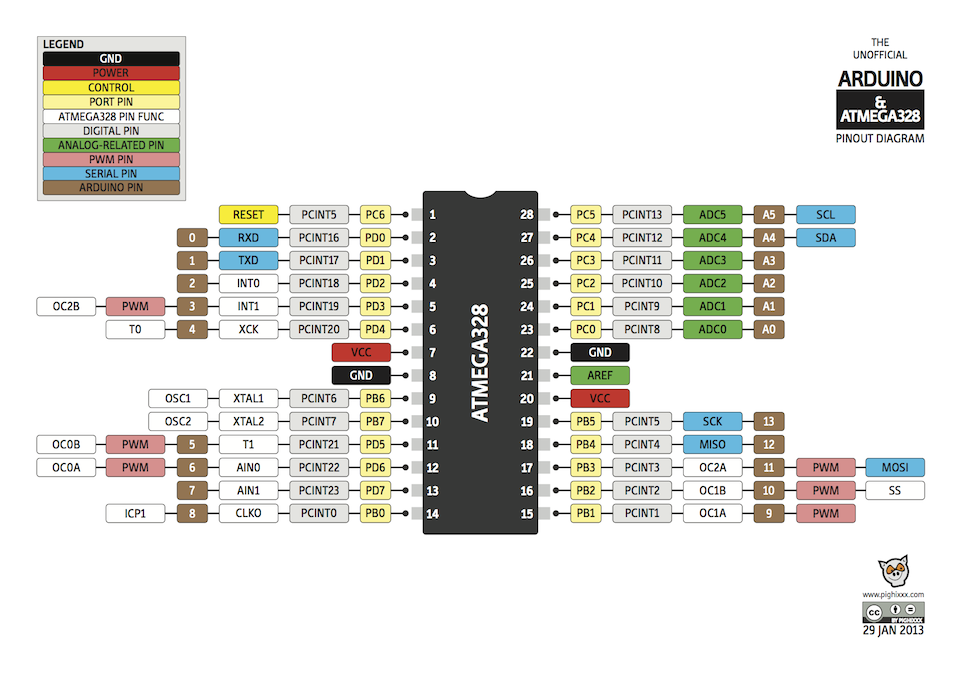
**Fig.3** Arduino Uno

***Arduino Uno*** es una placa electrónica basada en el microcontrolador ***ATmega328P***. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación

|  |  |
| --- | --- |
| Microcontrolador | [ATmega328P](http://www.atmel.com/Images/doc8161.pdf) |
| Tensión de funcionamiento | 5V |
| Tensión de entrada (recomendado) | 7-12V |
| Tensión de entrada (límite) | 6-20V |
| Pines Digitales de I / O | 14 (de las cuales 6 proporcionan salida PWM) |
| PWM pines Digitales I / O | 6 |
| Pines de entrada analógica | 6 |
| Corriente DC por E / S Pin | 20 mA |
| Corriente DC de 3.3V Pin | 50 mA |
| Memoria flash | 32 KB (ATmega328P)  de los cuales 0,5 KB utilizado por el gestor de arranque |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Velocidad de reloj | 16 MHz |
| Largo | 68,6 mm |
| Anchura | 53.4 mm |
| Peso | 25 g |

Los estudiantes evaluaron minuciosamente la elección de un microcontrolador para el desarrollo del proyecto, un factor determinante para diagnosticar la potencia de un microcontrolador es por medio de la medición de **MIPS** (Un Millón de Instrucciones por Segundo). En el caso de Arduino Uno, que como se aclaró anteriormente utiliza el microcontrolador ***Atmega328P alcanza una respuesta de 1 MIPS***, es decir, un millón de instrucciones por segundo, para el desarrollo de este trabajo es más que suficiente. La siguiente tabla contiene detalles esenciales de Arduino Uno:

Arduino Uno utiliza un microcontrolador Atmega328P, el mismo es un encapsulado DIP. La próxima figura muestra cómo están configurados los pines de salida:



**Fig.5** Comparación pines Atemega328 vs pines de Arduino Uno

**Fig.4** Pines Atemega328

A continuación se ofrecerá una lista de componentes que fueron usados de Arduino Uno para este proyecto:

1. Armado de dedos:
   1. *5 pines Analógicos*: para leer cada uno de los sensores flexibles. Los valores se obtendrán partir de un divisor de tensión y el propio sensor, ya que el sensor flexible actúa como una resistencia variable. Estos pines digitales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se miden desde 0 a 5 voltios, aunque es posible cambiar el extremo superior de su rango utilizando el *analogReference* () función.
   2. *5 pines PWM:* para cada uno de los servos que de acuerdo a la lectura de los sensores flexibles, enviaran la posición de un ángulo para efectuar el movimiento en la mano mecánica (dedos).
2. Armado de antebrazo
   1. 2 pines PWM: Cada uno conectado a un servo. Los servos se moverán según los datos recibidos del acelerómetro.
   2. 2 Transceptores: uno actuará como emisor, y el otro como receptor.
   3. Un módulo GY-80. Este módulo posee distintos elementos, del cual utilizaremos uno solo, el acelerómetro, el cual nos permitirá simular el movimiento real de un brazo.

Al utilizar pines analógicos, se desarrollara la conversión Analógica Digital (AD):

## Conversor A/D



**Fig. 6.** Conversión A/D

Un conversor analógico-digital es un dispositivo electrónico capaz de convertir una señal analógica en un valor binario, en otras palabras, éste se encarga de transformar señales analógicas a digitales (0 y 1). Justamente cuando realizamos un *AnalogicRead()* para obtener los valores del sensor flexible, necesitamos que esa señal analógica sea convertida a una digital para que pueda ser interpretada por el micro.

El dispositivo establece una relación entre su entrada (señal analógica) y su salida (digital) dependiendo de su resolución. La resolución determina la precisión con la que se reproduce la señal original. La misma se puede determinar mientras se conozca el valor máximo de la entrada a convertir y la cantidad máxima de la salida en dígitos binarios. Mediante la siguiente formula:

, donde n = bits.

Arduino utiliza un conversor A/D de 10-bits, así que: Resolución = Vref/1024 Mapeará los valores de tensión de entrada, entre 0 y Vref voltios, a valores enteros comprendidos entre 0 y 1023 (2^n-1), es decir, que nuestros sensores analógicos están caracterizados con un valor comprendido entre 0 y 1023.

Si Vref es igual a 5v, la resolución es aproximadamente de 5 mV. Por lo tanto el error en las medidas de tensión será siempre de sólo 5 mv.

## Comunicación por Puerto Serial:

Al enviar datos por el puerto serie, se debe tener en cuenta que la comunicación se realiza a través de valores con una longitud de 8-bits, mientras que como ya se ha indicado, el conversor A/D de Arduino tiene una resolución de 10-bits.

Por ejemplo, en este proyecto particular si se capturan los valores del sensor analógico (sensor flexible) y se envían por el puerto serie al PC (por ejemplo para corroborar los datos), una solución podría ser transformarlos en un rango entre 0 y 9 y en modo de codificación ASCII (carácter).

Lo cual se podría expresar como:

value1 = analogRead(analogPin1); //captura del valor de sensor analógico (0-1023)

serialWrite(treatValue(value1)); //escribir al puerto serie 8-bits

int treatValue(int data) {

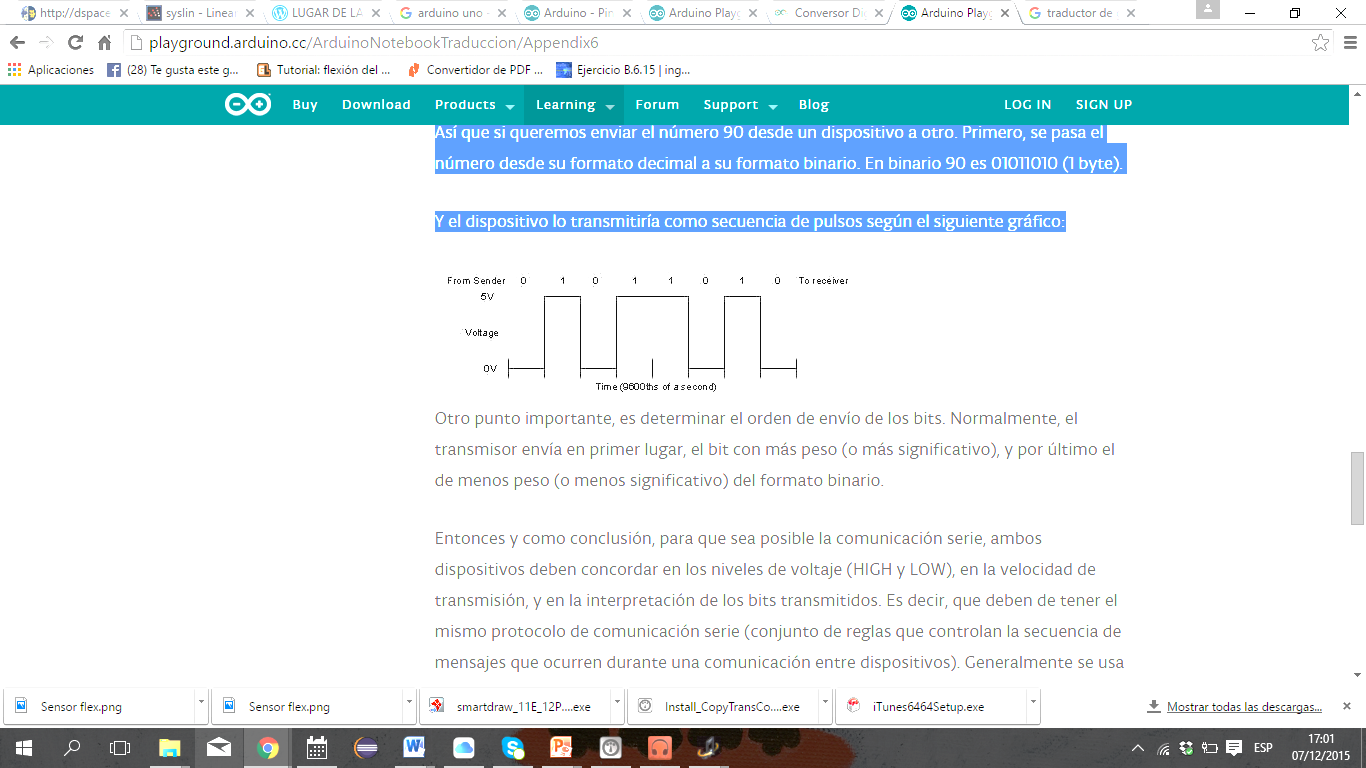
return (data \* 9 / 1024) + 48;// fórmula de transformación

}

## Comunicación serial

Para hacer que dos dispositivos se comuniquen se necesita un método de comunicación y un lenguaje o protocolo común entre ambos dispositivos. La forma más común de establecer dicha comunicación es utilizando la ***comunicación serial***, como vimos a lo largo de la cursada. La comunicación serial consiste en la transmisión y recepción de pulsos digitales, a una misma velocidad. El transmisor envía pulsos que representan el dato enviado a una velocidad determinada, y el receptor escucha dichos pulsos a esa misma velocidad. Esta técnica es conocida como ***comunicación serial asíncrona***.

Por ejemplo, si se tiene dos dispositivos conectados y que intercambian datos a una velocidad de 9600 bits por segundo (también llamados baudios), el receptor capturará la tensión que le está enviando el transmisor, y cada 1/9600 de un segundo, interpretará dicho tensión como un nuevo bit de datos (exactamente lo que sucede con los sensores flexibles). El dispositivo transmisor enviara una secuencia de pulsos según el siguiente gráfico:



**Fig.7** Pulsos de comunicación serial

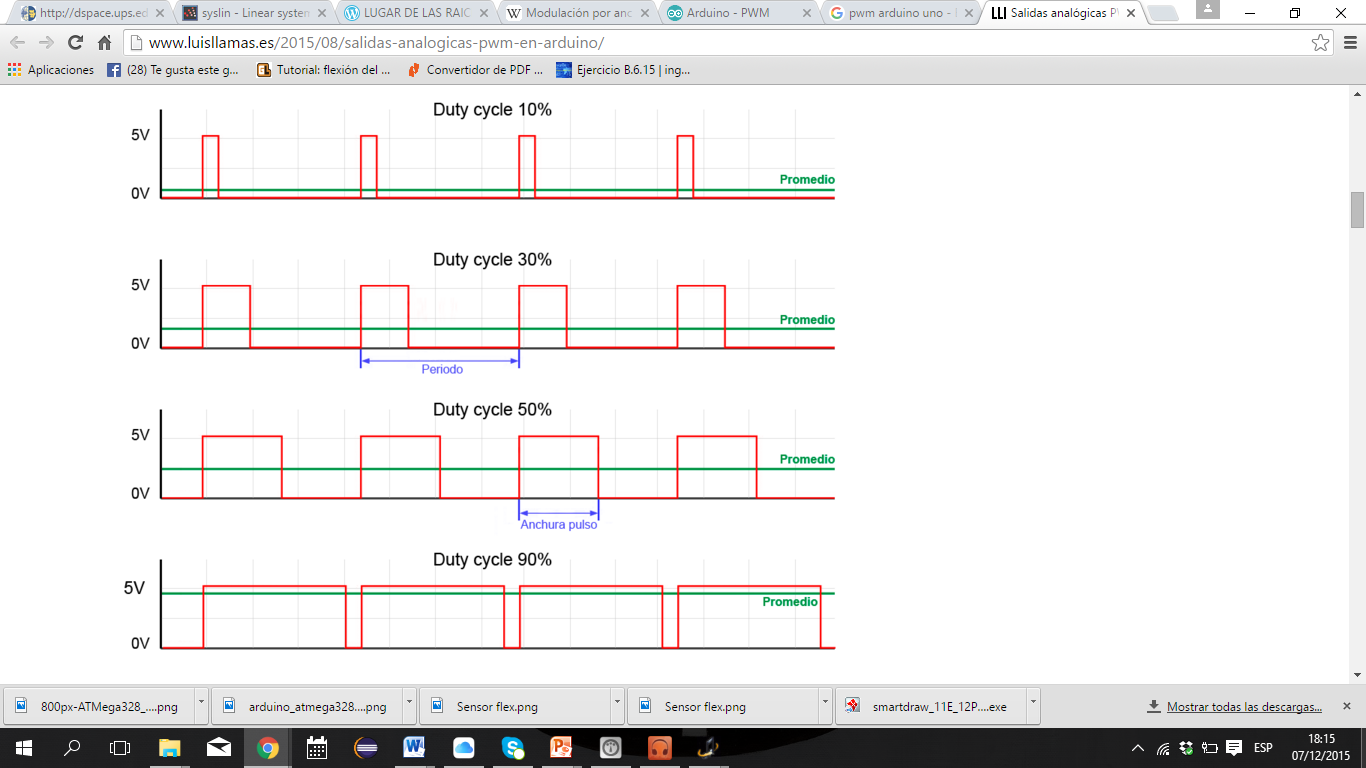
Otro punto importante, es determinar el orden de envío de los bits. Normalmente, el transmisor envía en primer lugar, el bit con más peso (o más significativo), y por último el de menos peso (o menos significativo) del formato binario.

Para que sea posible la comunicación serie, ambos dispositivos deben concordar en los niveles de voltaje (HIGH y LOW), en la velocidad de transmisión, y en la interpretación de los bits transmitidos.

Para conectar un dispositivo a un PC (o sistema operativo) necesitamos seleccionar un puerto serie y el cable apropiado para conectar al dispositivo serie.

## Salidas PWM

Se debe considerar que la modulación por ancho de pulso, es una aproximación a una salida analógica, debido a que lo único que puede proporcionar Arduino Uno es proporcionar una salida digital de -Vcc o Vcc. Para simular una señal digital durante un tiempo (particularmente se necesitaría para los servos), se utiliza PWM, la cual mantiene constante la frecuencia, mientras hace variar la anchura del pulso.



**Fig.8.** PWM – Modulación por ancho de pulso

Se debe aclarar que una salida PWM el valor de tensión realmente es Vcc. Por ejemplo, si se está alimentando un dispositivo que necesita 3V, en realidad se estará suministrando 5V durante un 60% del tiempo y 0V durante el 40%. Una señal pulsada es suficiente para emular una señal analógica en muchas aplicaciones. Las funciones PWM por hardware emplean los Timer para generar la onda de salida. Cada *Timer* da servicio a 2 o 3 salidas PWM. Para ello dispone de un registro de comparación por cada salida.

## Asociación de timers y PWM

En el caso de Arduino Uno

* El *Timer0* controla las salidas PWM 5 y 6.
* El *Timer1* controla las salidas PWM 9 y 10.
* El *Timer2* controla las salidas PWM 3 y 11.

## Frecuencia del PWM

La frecuencia de cada PWM depende de las características del temporizador al que está conectado, y de un registro de *preescalado*, que divide el tiempo por un número entero. La frecuencia de los PWM se puede modificar cambiando el *preescalado* de los Timer correspondientes.

* *Timer0*, con una frecuencia de 62500Hz, y preescalado de 1, 8, 64, 256 y 1024.
* *Timer1*, con una frecuencia de 31250Hz, y preescalado de 1, 8, 64, 256, y 1024.
* *Timer2,* con una frecuencia de 31250Hz, y preescalado de 1, 8, 32, 64, 128, 256, y 1024.

La frecuencia estándar para las salidas PWM en Arduino Uno, Mini y Nano es de 490 Hz para todos los pines, excepto para el 5 y 6 cuya frecuencia es de 980 Hz.

# SECCIÓN 1:

# SECCIÓN 1:

# Acelerómetro

# ACELEROMETRO

## ¿Qué es un acelerómetro?

Los acelerómetros son dispositivos que miden la aceleración, que es la tasa de cambio de la velocidad de un objeto. Esto se mide en metros por segundo al cuadrado (m/s²) o en las fuerzas gravitatorias (g). La sola fuerza de la gravedad para nosotros en el planeta Tierra es equivalente a 9,8 m/s², por lo que 1g= 9,8 m/s².  
Los acelerómetros detectan la aceleración, ya sea estática o dinámica. La fuerza estática incluye la gravedad. Es decir, cuando el acelerómetro este totalmente quieto, la única fuerza que actuará sobre el será la de la gravedad (1g) sobre el eje Z.  
Las fuerzas dinámicas incluyen vibraciones y movimiento. Los acelerómetros pueden medir la aceleración en uno, dos o tres ejes. Para este trabajo, utilizamos el módulo GY-80, el cual posee entre sus distintos dispositivos integrados, el acelerómetro ADXL345.

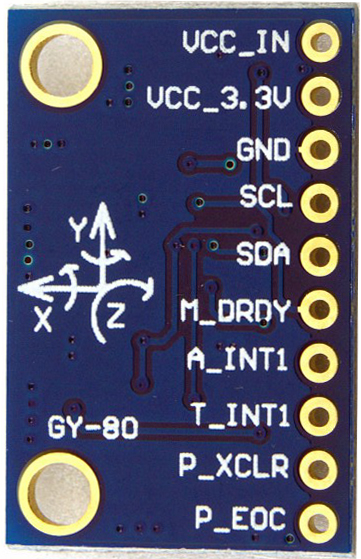
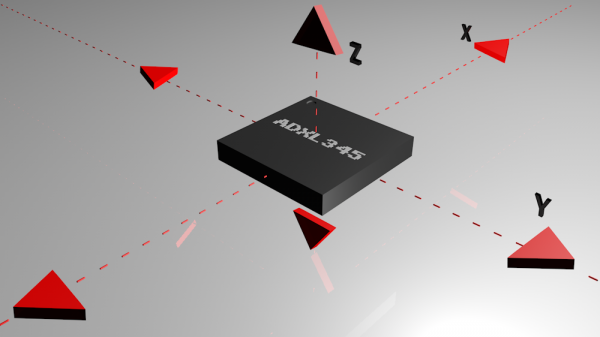
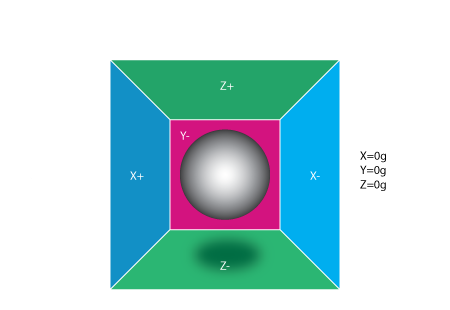


Figura 1. Módulo GY-80

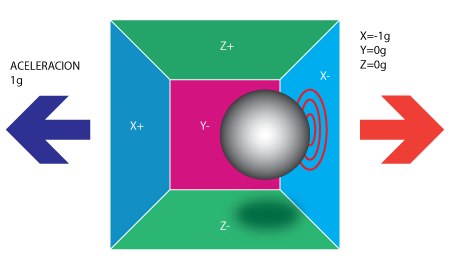
¿Cómo funciona un acelerómetro?

Los acelerómetros contienen placas capacitivas internamente. Algunas de estas son fijas, mientras que otras están unidas a resortes minúsculos que se mueven internamente conforme las fuerzas de aceleración actúan sobre el sensor. Como estas placas se mueven en relación la una a la otra, la capacitancia entre ellas cambia. Este cambio en la capacitancia en cada eje se convierte en una tensión de salida proporcional a la aceleración en ese eje.

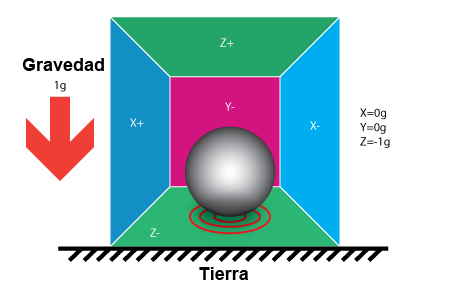
Si desea una explicación más práctica, continúe leyendo, caso contrario proceda a la sección de **interfaz de comunicaciones**. Para ver cómo funciona un acelerómetro, pensemos en una caja cuadrada con una esfera flotando en el medio.



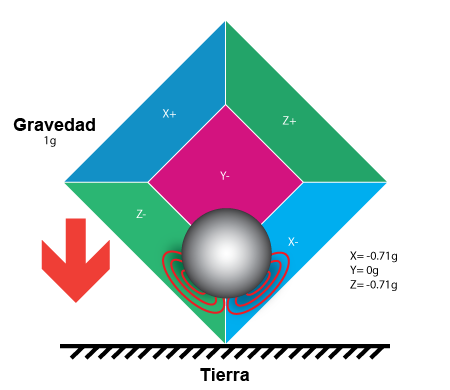
Si ponemos esta caja en un lugar sin gravedad y suponiendo que nada afecta la posición de la pelota - la pelota simplemente flotará en el medio de la caja. A partir de la imagen de arriba se puede ver que se le asigna a cada eje un par de paredes (he eliminado la pared Y+ para que podamos mirar dentro de la caja). Imagine que cada pared es sensible a la presión. Si movemos repentinamente la caja hacia la izquierda (aceleramos con 1g de aceleración = 9,8 m / s ^ 2), la pelota golpeará la pared X- .   
Si medimos la presión que causa la pelota en la pared X- obtenemos un valor de salida de -1g.



Hay que tener en cuenta que el acelerómetro detecta la fuerza que se dirige en dirección contraria al vector de aceleración. Como vemos en la imagen, la fuerza es hacia la izquierda, pero la pelota pega en la pared derecha, y es allí donde se detecta la fuerza (**fuerza inercial**). Yendo a la realidad, si ponemos nuestra caja en el planeta tierra, la pelota caerá en la pared Z- y aplicará una fuerza de 1g sobre la misma:

En este caso la caja no se mueve pero aun así tenemos una lectura de -1g en el eje Z.  
La presión que la pelota aplica en la pared es causada por la fuerza de la gravedad.  
En teoría, podría ser otra fuerza, por ejemplo, si la pelota es metálica, y ponemos un imán a un costado de la caja, la esfera se moverá e impactará con otra pared.  
Esto se aclara para demostrar que, en esencia, **un acelerómetro no mide aceleración, sino fuerza**. Tan solo sucede que la aceleración causa una fuerza inercial que es capturada por el mecanismo de detección de fuerzas del acelerómetro.

Hasta ahora solo consideramos un solo eje, pero lo interesante de nuestro acelerómetro, es que puede medir fuerzas en los 3 ejes. Girando la caja 45º obtenemos:



Los valores 0.71 no son arbitrarios, son una aproximación de .

Mediante simples cálculos matemáticos, es posible determinar las fuerzas, y con ellas la aceleración de cada eje. En este trabajo no se profundizará en la teoría matemática, solo se ha mostrado este ejemplo práctico para que el lector pueda comprender un poco más el funcionamiento de un acelerómetro.

# Interfaz de comunicaciones

Los acelerómetros pueden comunicarse a través de un convertidor analógico, digital o una interfaz de modulación de ancho de pulso (PWM).

Los acelerómetros con una interfaz analógica entregan una tensión proporcional a la aceleración en cada uno de sus que normalmente fluctúan entre tierra y el valor de alimentación Vcc.

Los acelerómetros con una interfaz de PWM producirán una onda cuadrada con una frecuencia fija, pero el ciclo de trabajo del pulso variará con la aceleración detectada.

Los acelerómetros con una interfaz digital pueden comunicarse a través de los protocolos de comunicación de SPI o I2C. Estos tienden a tener más funcionalidades y ser menos susceptibles al ruido que acelerómetros analógicos.

El ADXL345 es digital, y para la comunicación utilizaremos el protocolo I2C.

# Potencia

Los acelerómetros son generalmente dispositivos de baja potencia. La corriente requerida cae típicamente en la gama de micro (μ) o mili – amperios, con una tensión de alimentación de 5 V o menos. El consumo de corriente puede variar dependiendo de la configuración (por ejemplo, el modo de ahorro de energía en comparación con el modo de funcionamiento estándar). Estos modos diferentes pueden hacer a los acelerómetros muy adecuados para aplicaciones que funcionan con batería.  
En el caso del ADXL345, se alimenta con 3,3 V y su consumo de corriente es, en modo de medición, de 23 µA y en modo standby de 0,1 µA.

# Otras características

La gran mayoría de los sensores poseen 3 parámetros fundamentales:

* *El Offset (Zero rate level):* Cuando el acelerómetro esta "quieto" debería entregar en su salida 0 g (1g en el eje Z). Sin embargo no es así y con el acelerómetro en reposo siempre tendremos un valor distinto de 0. En la hoja de datos del ADXL, en la Tabla 1, podemos ver que el Zero g OFFSET es del orden de los +-150mg en los ejes X e Y y +-250mg en el eje Z.

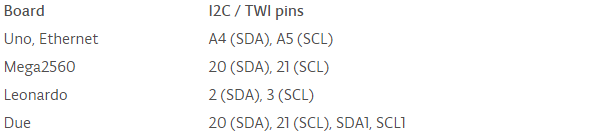
Es muy importante determinar el offset de nuestro sensor para obtener una medición correcta.

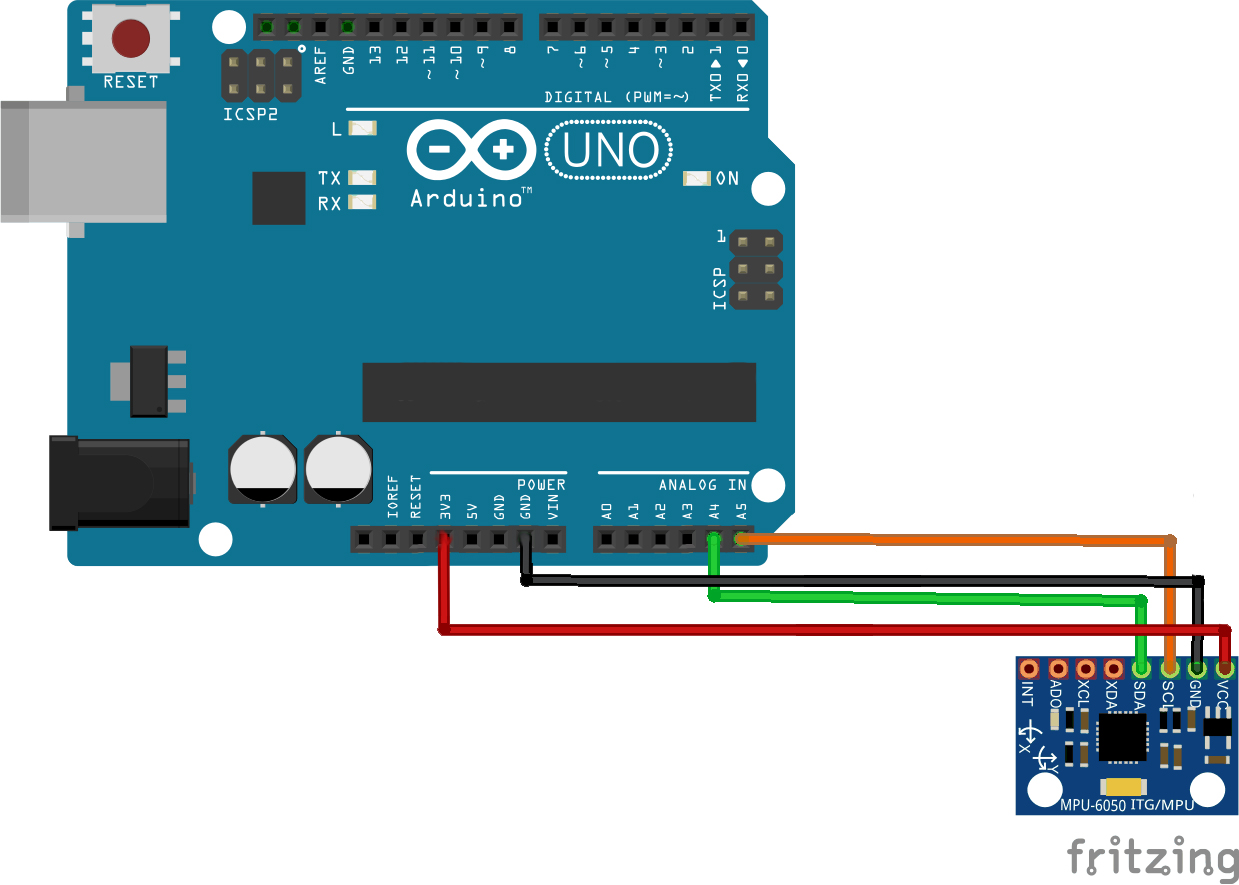
* *Sensibilidad*: Nos indica cuantos mg equivale cada bit del dato binario de salida. La sensibilidad es bastante fija y está muy cerca de lo que dice la hoja de datos. Se puede determinar prácticamente para mejorar la precisión de la medición.
* *Rango:* El rango es entre que valores el sensor puede medir. El ADXL345 posee 4 rangos configurables: +-16g, +-8g y +-4g y +-2g. A mayor rango, vamos a poder medir aceleraciones mayores pero será menos sensible a aceleraciones más pequeñas. Para nuestro proyecto, el movimiento de una mano provoca aceleraciones pequeñas, por lo que configuraremos el sensor en +-4g.

Otro dato importante es el ODR (Output Data Rate). En este sensor podemos elegir entre 0.1Hz hasta 3200Hz. A mayor ODR mayor nivel de ruido tendremos en la medición.

# Conexión del módulo GY-80

Como ya mencionamos, utilizaremos la interfaz I2C. Para esto, ya existe una librería que provee Arduino, llamada “Wire.h”, la cual posee el protocolo de I2C.  
Para la conexión I2C, existen pines específicos dependiendo de que placa utilicemos. Buscando en la página oficial de Arduino, podemos observar:



Como utilizamos la placa Arduino Uno, la conexión es la siguiente:  


* VCC -> 3,3V
* GND -> GND
* SCL (Serial Clock) -> A5
* SDA(Serial Data) -> A4

Hay que tener en cuenta que al estar usado una interfaz digital los valores entregados por el acelerómetro son paquetes de bytes que vamos a leer por los pines SDA y SCL localizados en el A4 y A5 del Arduino, pero se debe aclarar que estos pines serán usados como pines digitales no cómo analógicos.

# SECCIÓN 1:

# SECCIÓN 2:

# Servomotores

# Servomotor

## ¿Qué es un servomotor?

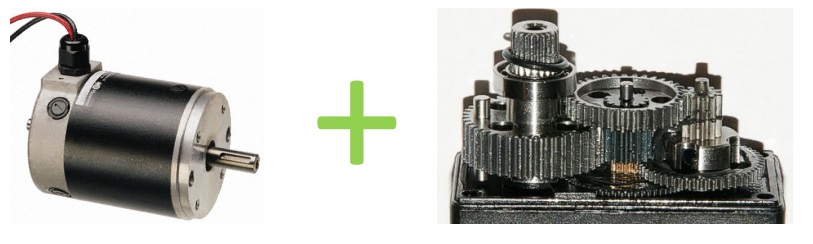
Es un dispositivo capaz de ubicarse en cualquier posición dentro de un rango de operación (generalmente de 180˚) y mantenerse estable en dicha posición.

## Partes de un servomotor

En general, los servos suelen estar compuestos por 4 elementos fundamentales:

• Motor DC: Es el elemento que le brinda movilidad al servo. Cuando se aplica un potencial a sus dos terminales, este motor gira en un sentido a su velocidad máxima. Si el voltaje aplicado sus dos terminales es inverso, el sentido de giro también se invierte.

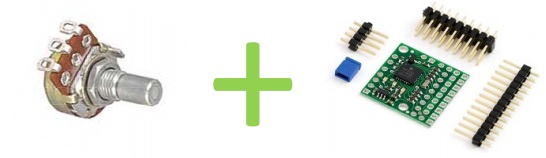
• Engranajes reductores: Tren de engranajes que se encarga de reducir la alta velocidad de giro del motor para aumentar su capacidad de torque.



**Fig.9.** Motor de CD y engranajes

• Sensor de desplazamiento: Suele ser un potenciómetro colocado en el eje de salida del servo que se utiliza para conocer la posición angular del motor.

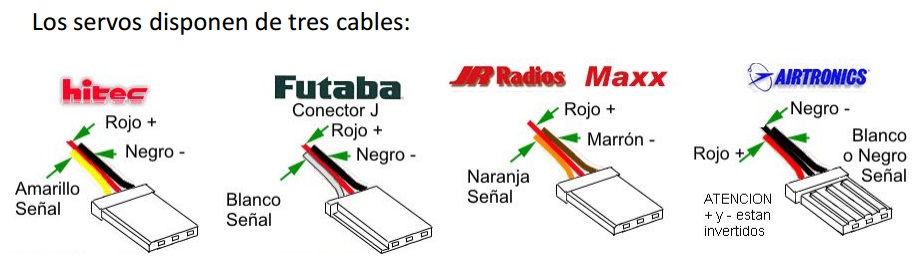
• Circuito de control: Es una placa electrónica que implementa una estrategia de control proporcional de la posición por realimentación. Para ello, este circuito compara la señal de entrada de referencia (posición deseada) con la posición actual medida por el potenciómetro. La diferencia entre la posición actual y la deseada es amplificada y utilizada para mover el motor en la dirección necesaria para reducir el error.



**Fig.10.** Circuito de control de servo motor

## E:\Facu\Arquitecturas II\Servo diagrama bloques.jpgDiagrama en bloques

**Fig.11.** Diagrama en bloques de Servomotor

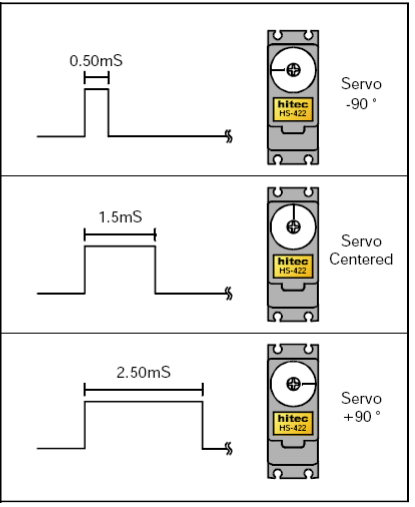


Dos cables de alimentación (positivo y negativo/masa) que suministran una tensión de 5V y un cable de control que indica la posición deseada al circuito de control mediante señales PWM (“Pulse Width Modulation”), las cuales, en este proyecto, se enviaran mediante Arduino.

## Funcionamiento

Las señales PWM utilizadas para controlar los servos están formadas por pulsos positivos cuya duración es proporcional a la posición deseada del servo y que se repiten cada 20ms (50Hz).

Todos los servos pueden funcionar correctamente en un rango de movimiento de 90˚, que se corresponde con pulsos PWM comprendidos entre 0.9 y 2.1 ms. Sin embargo, también existen servos(los que utilizaremos en este proyecto) que se pueden mover en un rango extendido de 180˚ y sus pulsos de control varían entre 0.5 y 2.5 ms.

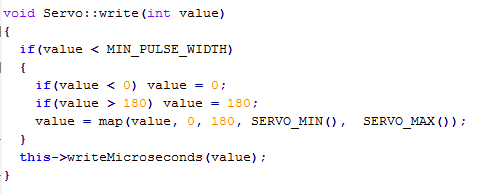


**Fig.9.** Análisis del PWM por medio de Servomotor

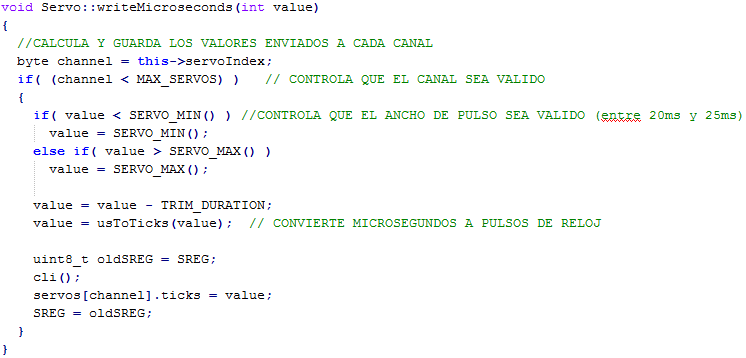
Para bloquear al servomotor en una posición es necesario enviarle continuamente la señal con la posición deseada. De esta forma, el sistema de control seguirá operando y el servo conservará su posición y se resistirá a las fuerzas externas que intenten cambiarlo de posición.

Entonces, para posicionar el servo en un ángulo determinado, se debe calcular el ancho del pulso. En Arduino, al utilizar la librería Servo.h, esta conversión es transparente al programador, ya que el mismo solo escribe el ángulo que desea( Servo.write (60º)), y la librería se encarga de la conversión.

Esta librería utiliza los timers 1 y 3 de Arduino, para generar pulsos que se repitan cada 20ms. Además, como se puede observar, la función write permite restringir los valores ingresados por el usuario en valores entre 0° y 180°, a su vez, mapear estos para asignarles valores correspondientes a los pulsos entre 0,5ms (SERVO\_MIN) y 2,5ms (SERVO\_MAX).

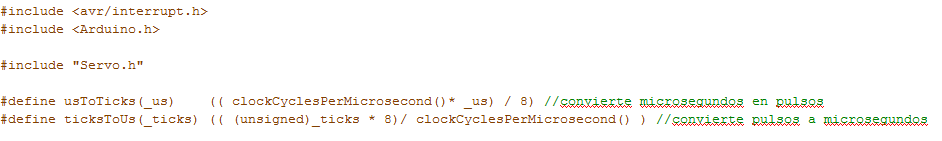


Luego la función writeMicroseconds se encarga de recoger este valor para comprobar su validez, eliminar el pequeño retardo que se dio debido a las operaciones anteriores y convertir ese valor a pulsos de reloj.



Esto se logra mediante una conversión realizada utilizando el preescaler. El prescaler toma la frecuencia básica del reloj temporizador (que puede ser la frecuencia de reloj de la CPU o puede ser cierta frecuencia mayor o menor) y lo divide por algún valor antes de alimentar al temporizador, de acuerdo a cómo se configuran los registros. Los valores de prescaler que se pueden configurar podría limitarse a unos valores fijos (potencias de 2), o pueden ser cualquier valor entero de 1 a 2P, donde P es el número de bits prescaler.

El propósito del prescaler es permitir que el temporizador tenga la tasa de velocidad de reloj que un usuario desea. Para temporizadores más pequeños (8 y 16 bits), a menudo habrá un compromiso entre la resolución (alta resolución requiere una alta tasa de reloj) y el rango (altas velocidades de reloj hacen que el temporizador se desborde más rápidamente).



# SECCIÓN 1:

# SECCIÓN 3:

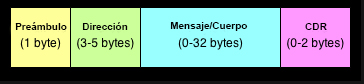
# Módulo nRF24L01

# Módulo nRF24L01

El módulo nRF24l01 es un transceptor avanzado que transmite en la banda de 2.4Ghz.  
Un transceptor es un dispositivo que posee un emisor y un receptor a la vez.  
  
Especificaciones

* Consumo de corriente de 13 mA aproximadamente.
* 26uA en modo Standby
* 900nA en modo Power Down
* Trabaja entre 1.9 V y 3.6 V

Funcionamiento básico

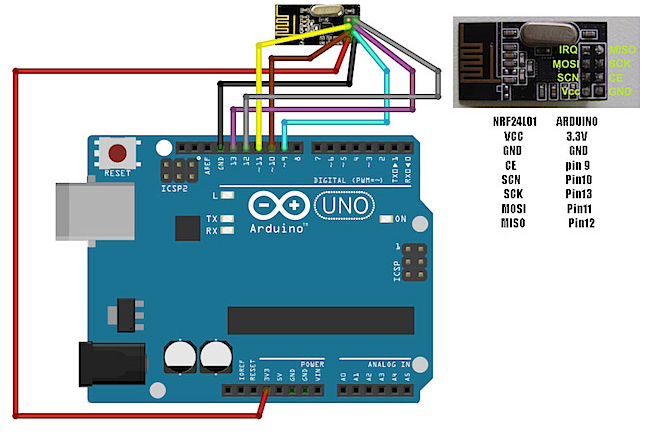
  
Como el nombre del módulo lo indica, es un transceptor que trabaja con ondas de radio (RF: Radio Frecuency). El emisor envía al aire un paquete que tiene la siguiente estructura:

**Fig.10.** Paquete de Emisor

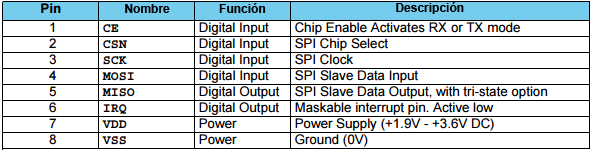
Cada mensaje comienza con un preámbulo, el cual se utiliza para identificar paquetes entrantes. Luego sigue la dirección del emisor. Después le siguen el mensaje que se quiere enviar, y el código de redundancia para verificar la integridad del mensaje una vez recibido.

El receptor estará esperando mensajes entrantes de una determinada dirección.  
Cuando un mensaje recibido concuerda con la dirección del emisor deseado, el receptor almacenará el mensaje en una memoria interna FIFO.

## Conexión



**Fig.11.** Conexión Arduino Modulo Inalámbrico



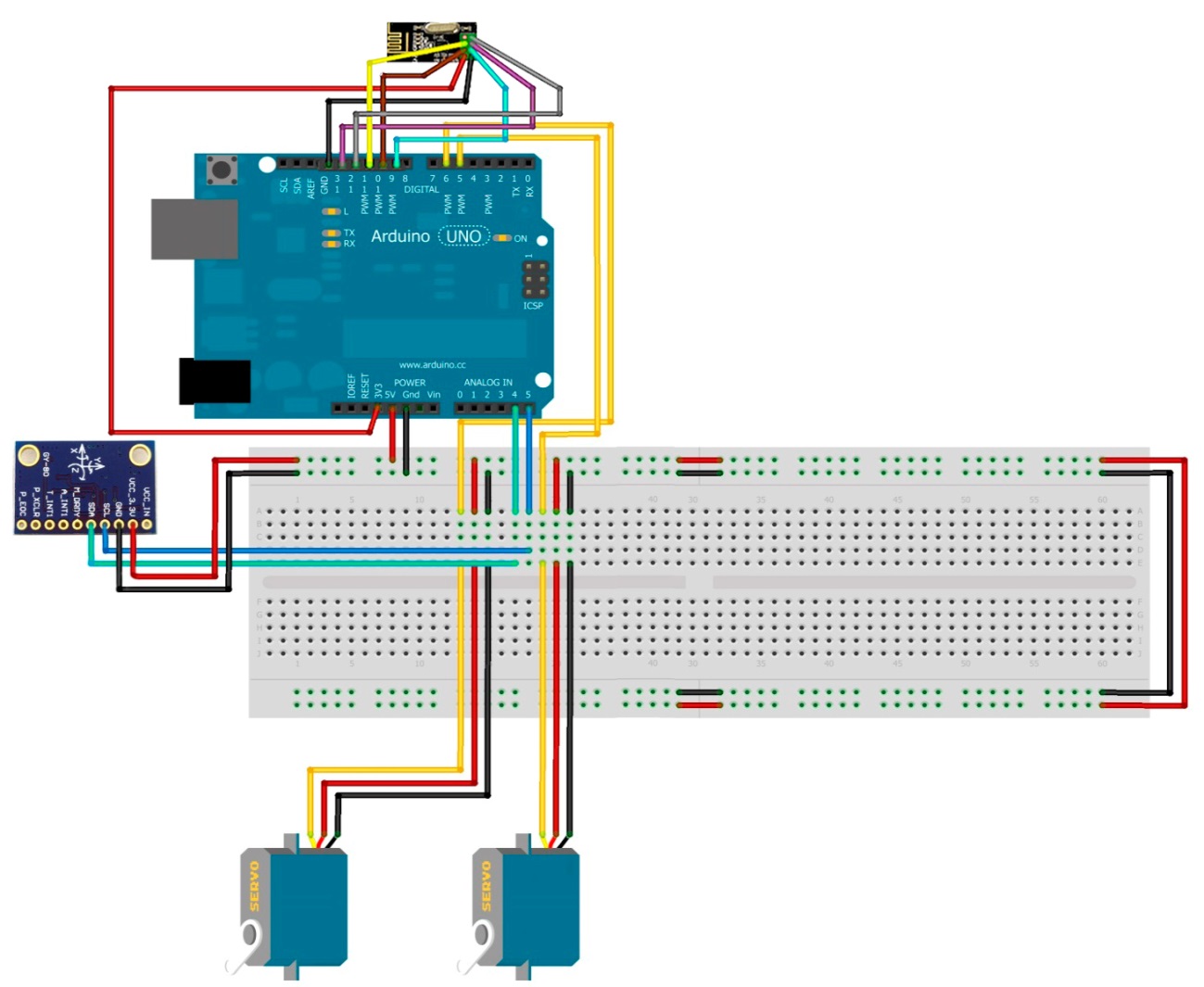
Como se puede observar en la tabla, el módulo utiliza un el protocolo de comunicación SPI, el cual es un esquema de maestro – esclavo. Por ende, los pines de MOSI, MISO, SCK y el CSN son necesarios. Cada uno realiza la siguiente tarea:

* **SCK** *(Clock o reloj)*: Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
* **MOSI** *(Master Output Slave Input)*: Salida de datos del Master y entrada de datos al Slave.
* **MISO** *(Master Input Slave Output)*: Salida de datos del Slave y entrada al Master.
* **CSN** *(Slave Select)*: Para seleccionar un Slave, o para que el Master le diga al Slave que se active.

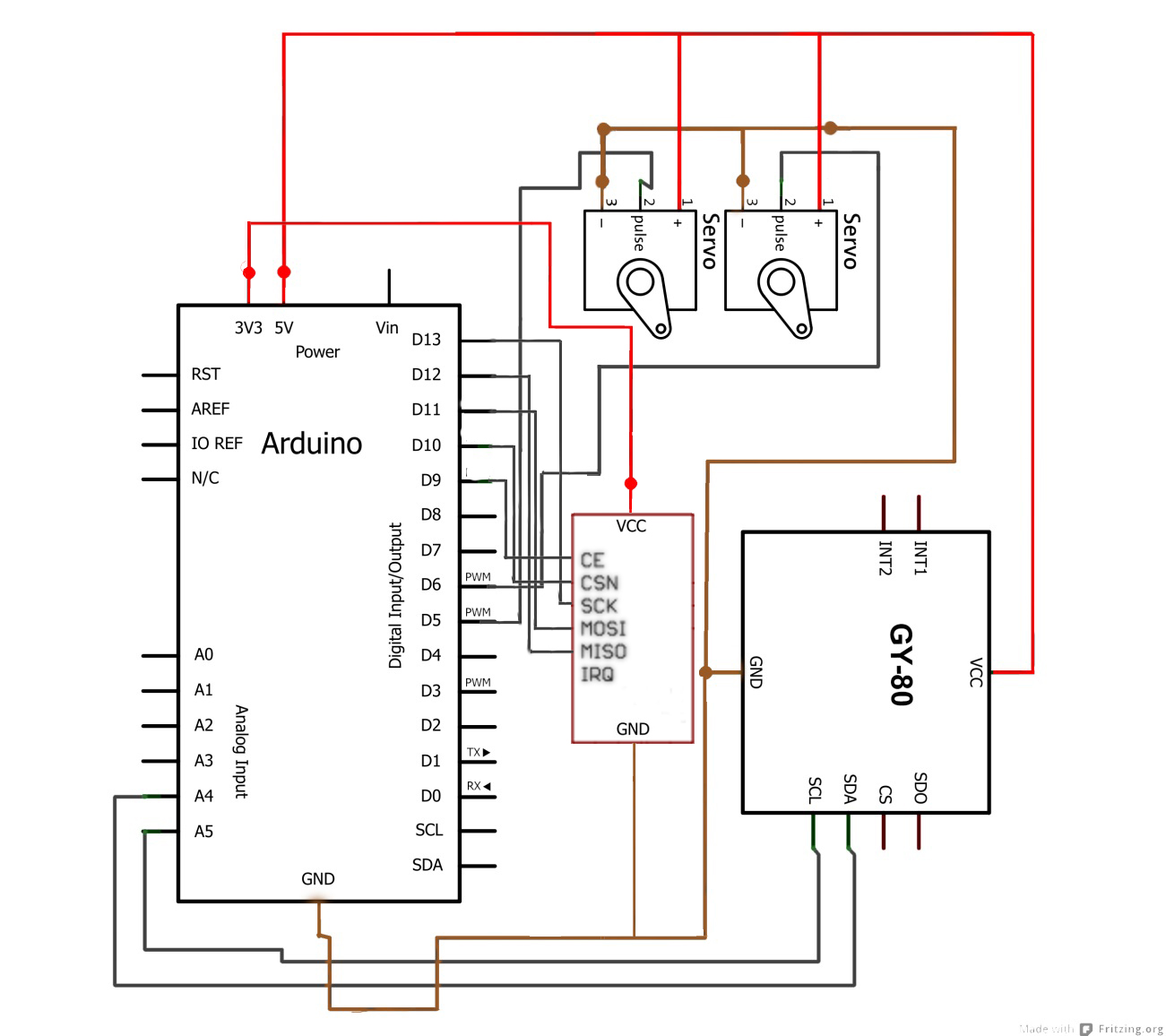
El pin CE define si el módulo trabajará como un emisor o como un transmisor.  
Finalmente, tenemos VDD (Tensión) y VSS (Tierra). El pin de interrupción IRQ no lo utilizaremos en este proyecto.

# Acelerómetro + Servos + Transceptores

Para este proyecto, se combinó los datos recibidos del acelerómetro para mover unos servomotores. Se utilizó dos servos, uno de los cuales se moverá dependiendo de los datos obtenidos del eje X, y el otro dependiendo de los datos obtenidos del eje Y.

A continuación se puede observar la conexión completa.   


**Fig.12.** Conexión de acelerómetro + Servos + Transceptores

El esquemático general es el siguiente:

**Fig.13.** Esquemático General de modulo inalámbrico, servos y acelerómetro

IMPORTANTE: Se debe aclarar que, por un lado tenemos una Arduino, a la cual se conectan los dos servos y un módulo **receptor.** Por el otro lado, tenemos otra Arduino a la cual se conecta el módulo GY-80 y un módulo **emisor.**

Algoritmo  
Al ser un algoritmo extenso, a continuación solo se explicará el funcionamiento en pseudocódigo. Para observar el programa completo, observe el anexo. El programa esta comentado de manera que sea fácil de entender y no queden dudas a la hora de implementarlo.  
Como se mencionó recientemente, poseemos dos Arduinos, cada una con un módulo emisor/receptor. Por ende, tendremos un código correspondiente para cada una.

**Pseudocódigo Arduino Emisor**

Primero incluimos las librerías. Wire.h permite manejar el protocolo I2C. El resto de las librerías se sobreentiende su utilidad.

#include <Wire.h>   
#include "ADXL345.h"  
#include <SPI.h>  
#include "nRF24L01.h"  
#include "RF24.h"

Int vector[2];

Primero obtenemos del acelerómetro las variables XY

Wire.IniciarTransmision (Dirección del acelerómetro);

Wire.Escribir (Registro X);

Vector[0] = Wire.Leer (Registro X);

Wire.Escribir (Registro Y);

Vector[1] = Wire.Leer (Registro Y);

Ahora, mediante la librería del acelerómetro, utilizaremos una función que permite saber si se ha tapeado el acelerómetro. Con tapeado nos referimos a una muy veloz variación en el eje Y. Cada vez que se tapea, se cambia de un estado a otro.  
El programa tiene 2 estados: en ambos estados siempre se enviará un vector que posee los datos X e Y. Lo que varía es que, en un estado 0 solo se actualizará la coordenada X, permaneciendo Y igual. Esto se realiza de manera que se mueva solo un servo a la vez. El estado 1 funciona al revez.

Si (Acelerometro.LeerActividad()== True ) { Cambiamos de un estado a otro si hay tap

Si (Estado ==1) Estado=0;

Sino Estado= 1;

Ahora enviamos los datos

Si (Estado == 0) radio.enviar(vector[0]);  
Si (Estado == 1) radio.enviar(vector[1]);

} Cerramos el loop

**Pseudocódigo Arduino Receptor**

#include <Servo.h>  
#include <SPI.h>  
#include "nRF24L01.h"  
#include "RF24.h"

Int Vector[2];

Loop(){

Si (Radio.Disponible ==True){

Msg= Radio.Leer(mensaje);  
Servo1.Escribir(Vector[0]);  
Servo2.Escribir(Vector[1]);

}}

# 

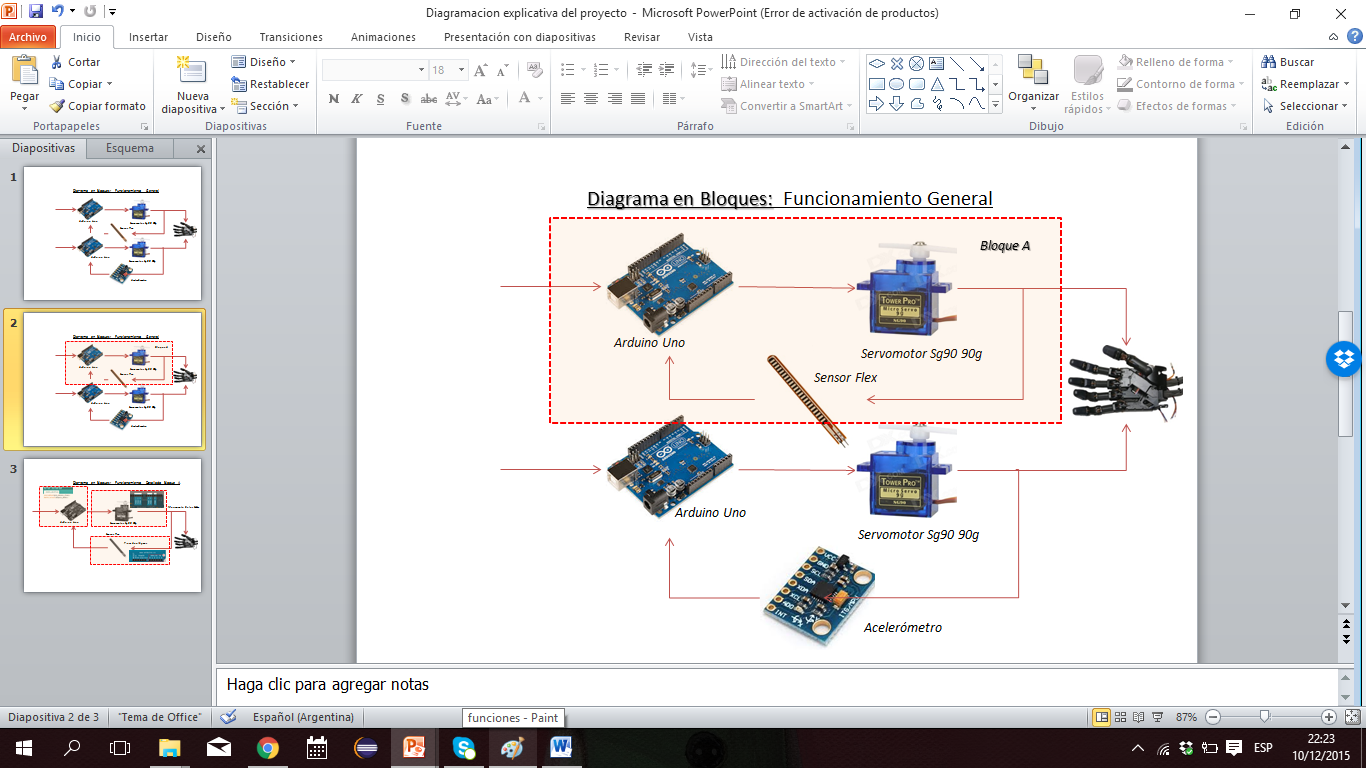
# 

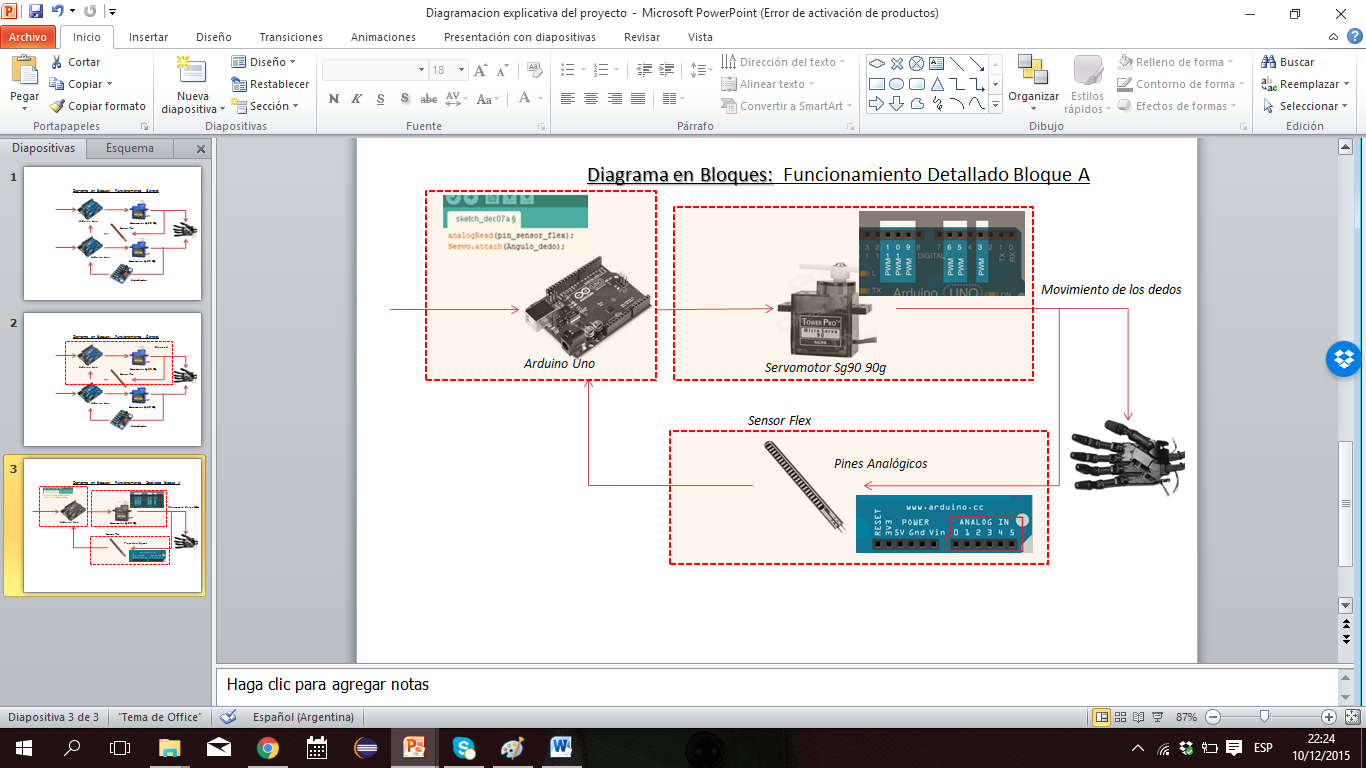
# SECCIÓN 1:

# SECCIÓN 4:

# Sensores Flexibles

# Análisis de Bloque 1





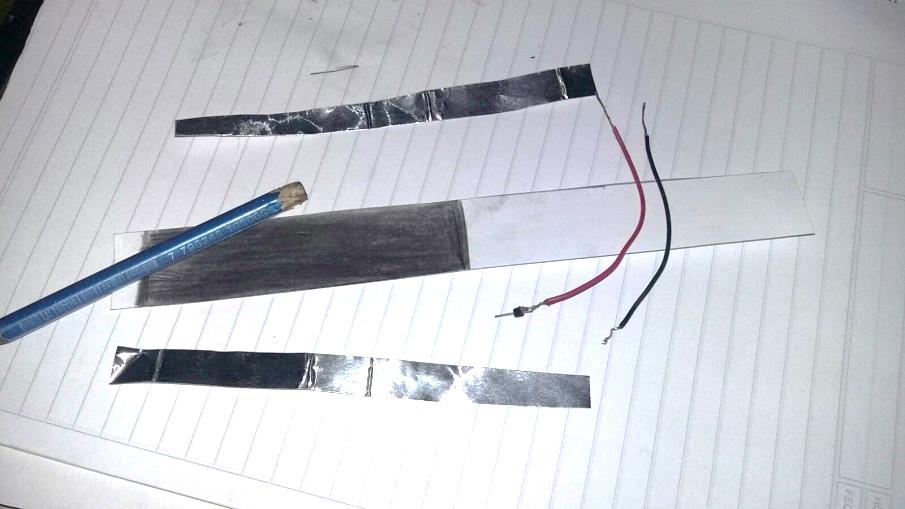
**Fig.14** Diagrama de Flujo de Bloque A Especificación

# Sensores Flexibles

Uno de los mayores desafíos era poder simular el movimiento de los dedos, la primera cuestión que surgió fue *¿Cómo podemos, reconocer el desplazamiento de cada una de las articulaciones de los dedos*?

La respuesta a esta pregunta fue por medio de los sensores flexibles. Sin embargo, se sumó otra dificultad ya que no se contaba con los módulos de *sensor flex* propio de Arduino. Ante tal desafío, los alumnos se ingeniaron para poder realizar, un sensor el cual cumpla la función de una resistencia variable.

### Creación de Resistencia Variable.

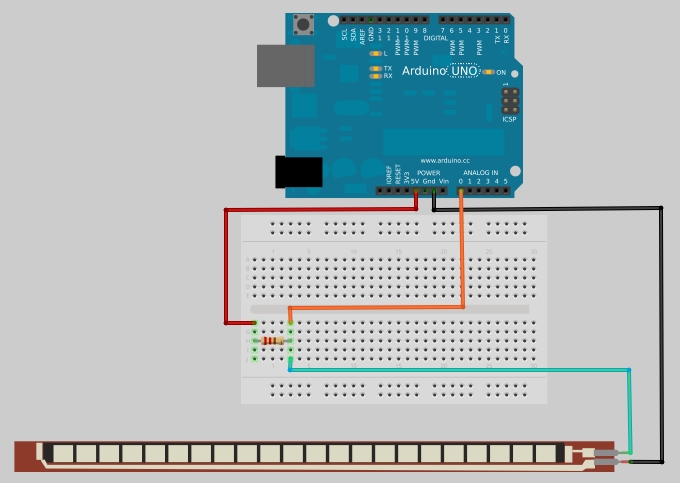


La resistencia conto especialmente de tres materiales:

* Papel
* Lápiz de grafito
* Cinta de aluminio

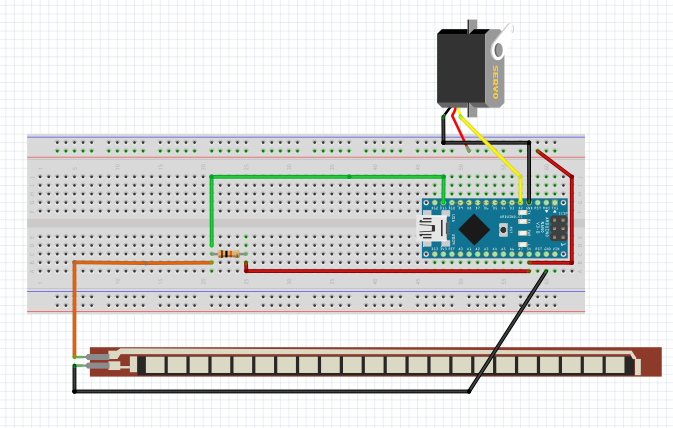
Pero la gran pregunta es *¿Cómo funciona?* La corriente fluye desde un terminal de la batería (en este caso los 5V de Arduino) hasta el otro terminal por medio de los cables, y la cinta de aluminio adherida al grafito que se encuentra en el papel. El grafito junto con la cinta de aluminio son materiales conductores, los cuales juntos poseen una cierta resistencia al paso de corriente, lo cual se conoce como Ohms (Ω). A medida que se dobla más el papel presenta mayor resistencia. Los valores pueden ser analizados por medio de los pines analógicos de la placa Arduino. A continuación se brinda el circuito básico para poder reconocer los valores analógicos, los cuales se pueden obtener a partir de un divisor de tensión:

**Fig.15** Componentes de resistencia variable

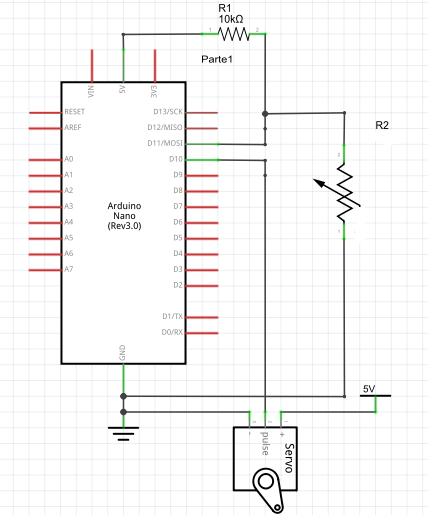


**Fig.16** Conexión Básica de Sensor Flex

A su vez, por lo anteriormente mencionado este sensor se puede esquematizar como una resistencia variable.



**Fig.17** Conexión Básica de sensor Flex y Servo



**Fig.18** Esquemático y diagrama de conexiones del bloque sensores flexibles + servos

Cabe aclarar, que los sensores realizados por los alumnos cumplían con determinados parámetros de resistividad cuando se colocaban en cierto ángulo. A continuación se brindara un gráfico característico del mismo:

**Fig.19** Diagrama característico de la resistencia en función al ángulo

# SECCIÓN 1:

# Bibliografía

BIBLIOGRAFIA:

<https://www.arduino.cc/en/Main/ArduinoBoardUno>

<http://www.mastec.com.mx/>

<http://www.instructables.com/id/Self-Balancing-Robot/>

<http://www.instructables.com/id/Wireless-Remote-Using-24-Ghz-NRF24L01-Simple-Tutor/>