**Guante del Tiempo**

**Wendy Montes de Oca**

**Mariana Álvarez**

**TallerInE Robótico 2020**

**Profesores**

**Guillermo Airaldi y Matías García**

# **Índice**

[**Índice**](#_k5r0j8k1wcg9) **1**

[**Resumen**](#_l845psdgv6rs) **3**

[Objetivo general](#_scz2ecdq5mx5) 3

[**Objetivos específicos**](#_h4xz83qyai4p) **3**

[Resultados](#_rzvuox5pyrix) 3

[**Introducción**](#_bifn4vx5a6m0) **4**

[**Materiales**](#_97mscl9u23p) **4**

[**Links de referencia**](#_709hj63qutvw) **5**

[**Marco Teórico**](#_bziop6tssqx2) **5**

[➪Glosario de programación](#_2yi9hrhqtbcp) 5

[➪Efecto Estroboscópico](#_lhu7lb66a9jn) 6

[➪Estroboscopio](#_53y1qi3i5r3) 7

[**Construcción del software**](#_k4oib71hkp56) **7**

[➪Potenciómetro](#_i1y4n2aw400m) 7

[➪IMU](#_4yb9hnha69l5) 8

[➪MPU 6050](#_dhqhj8xw4vcz) 8

[Acelerómetro](#_wqdeeurbgi66) 8

[Giroscopio](#_6jq52xod53ca) 9

[Errores en las medidas](#_78fo928gq60o) 10

[Filtro](#_htnns9nrf503) 10

[Calibración](#_8vzapapbw1e1) 11

[Código final](#_xsjfqrxnvomc) 12

[Declaración de las variables y librerías utilizadas](#_iz2z1fvz549m) 12

[Setup del código](#_vebb40y2tmwo) 12

[Función de control del parpadeo de la led a través del potenciómetro](#_z8puiq47wm2p) 12

[Función del MPU - 6050](#_84tqwsq5fdr0) 12

[Void loop](#_q1d8iya19acl) 13

[**Construcción del Hardware**](#_kd96klflqm7v) **13**

[➪Componentes y funciones en el circuito](#_jxlw5su3ycrx) 13

[**Análisis de los resultados**](#_clj58kq4g8oq) **14**

[➪Resultados del código y el armado](#_bqopsaib5u5c) 14

[**Conclusiones**](#_1q0i59fqeup0) **16**

[**Bibliografía**](#_9f2tbmw0csfd) **17**

[**Anexo**](#_5bvqnvtlej58) **18**

# **Resumen**

## **Objetivo general**

Construir un guante que sea capaz controlar la frecuencia de una led creando un efecto de “pausa temporal”.

## **Objetivos específicos**

1. Entender el manejo a través de un potenciómetro de la frecuencia de una led a nivel general y específico para nuestro proyecto, el objetivo es después poder igualarla a la frecuencia de giro de ciertos objetos. De esta forma se lograría el efecto esperado.
2. Estudiar el funcionamiento de un acelerómetro. Comprender cómo recibe y procesa los datos transformándolos en información.
3. Analizar la ventajas y desventajas de utilizar el filtro de Kalman. Luego se deberá discutir la posibilidad de utilizarlo en el proyecto.
4. Cuando se tenga una base sólida en relación al software, trabajar todo lo correspondiente a potencia y armado del hardware .

## **Resultados**

A grandes rasgos fuimos capaces (junto a la esencial tutoría de los profesores) de completar los objetivos específicos. Nuestro primer objetivo pudo ser llevado a cabo sin dificultad ya que el manejo del parpadeo de una led a través de un ***potenciómetro*** fue un tema abordado en el curso. En cambio ender el funcionamiento del ***MPU 6050***, como transmite y procesa los datos tuvo otro nivel de dificultad, de todas formas, luego de varios cambios en el código llegamos a poder utilizarlo y desarrollar un código final funcional.

Analizamos los diferente filtros que podíamos utilizar en el proyecto, el ***Filtro de Kalman*** es sorprendente, sin embargo además del costo en el procesamiento que tiene es muy complicado de entender, por esto concluimos que la mejor opción para nuestro proyecto sería utilizar el ***Filtro Complementario*** , es ideal para implementar con Arduino: fácil de utilizar, de bajo coste de procesamiento y con una precisión muy buena.

Por último, y no por ellos más sencillo pudimos trabajar la parte de potencia relacionada al Hardware, entendiendo cuál era el rol de cada componente.

# **Introducción**

Debido a la nueva modalidad del TallerIne causada por la situación actual de pandemia mundial, decidimos aprovechar la oportunidad para realizar un proyecto original y que nos motivara.

Tomamos como base para nuestro proyecto el efecto estroboscópico, cuya esencia radica en algo bastante sencillo que habíamos aprendido durante el curso, el control del parpadeo de una led a través de un potenciómetro. Asumiendo asimismo también, el desafío de buscar, estudiar y aprender sobre nuevos componentes que juntos conformarían la totalidad del proyecto. Todo para lograr una ilusión, la cual a simple vista parece que bajo el efecto del guante (que integra la led) un objeto que se encuentra girando es capaz de ralentizar su velocidad de giro, hasta incluso detenerse.

# **Materiales**

1. Guante
2. Arduino nano
3. Acelerómetro
4. Potenciómetro
5. Mosfet
6. 10 k and 100 Ohm resistores
7. Convertidor Ajustable Tensión 12v A 35
8. Placas de cobre
9. Cables
10. Led 100w
11. Batería lipo 12v - 1500mAh
12. Conector de batería
13. Multímetro
14. Equipo para soldar

# **Links de referencia**

1. <https://www.youtube.com/watch?v=rj9L1_-AzMc>
2. <https://www.instructables.com/id/DIY-Time-Control-Machine/>
3. <https://github.com/AlexGyver/EnglishProjects/tree/master/TimeGlove>

# **Marco Teórico**

## **➪Glosario de programación**

* ***“Serial.begin(9600)”***, este procedimiento se establece en el *setup()* del código y lo que hace es establecer la comunicación serial a 9600 bits por segundo.
* ***“Serial.println()”*** , esto significa que llama a un procedimiento llamado *println* que significa imprimir línea. Lo que hace es enviar datos desde el Arduino a la computadora.
* ***“Serial.available()”*** , se utiliza para saber si hay datos en el buffer[[1]](#footnote-0) que leer, si no hay datos envía un cero.
* ***“Serial.read()”***, se encarga de leer los datos del buffer.
* ***“Wire.h”***, es una librería que se utiliza para comunicar la placa arduino con dispositivos que trabajan mediante el protocolo I2C/TWI. Este sistema de comunicación utiliza dos líneas de transmisión: SDA (datos serie) y SCL (reloj serie) conectadas a dos resistencias tipo pull-up a 5 voltios.
* ***“float”***, es un tipo de dato para números de punto flotante, en sí es un número que tiene un punto decimal. Los números de coma flotante se utilizan a menudo para aproximar los valores analógicos y continuos porque tienen mayor resolución que los números enteros.
* ***“long”***, las variables long son variables de tamaño extendido para almacenamiento de números de 32 bits (4 bytes).
* ***“string”***, es un tipo de datos que es usado para guardar cadenas de caracteres. Los caracteres son datos de 8 bits representados en ascii.
* ***“boolean flag”***, tiene el tamaño de una celda. Una celda con todos los bits limpios representa la flag false y una flag con todos los bits establecidos representa la flag true. Las palabras que verifican una flag (por ejemplo IF) tratarán una celda que tenga cualquier bit establecido como true.
* ***“Wire.begin()”***, inicializa la biblioteca Wire y une al bus I2C como maestro o esclavo.
* **“(|)”**, es el operador binario OR en C ++ es el símbolo de barra vertical, |. Al igual que el operador &, | opera de manera independiente cada bit en sus dos expresiones enteras que lo rodean, pero lo que hace es diferente. El operador OR bit a bit es 1 si uno o ambos de los bits de entrada es 1, de lo contrario es 0.
* **“<<”**, el operador de desplazamiento a la izquierda hace que los bits del operando izquierdo se desplacen a la izquierda por el número de posiciones especificadas por el operando derecho.
* ***“millis()”***, devuelve el número de milisegundos transcurridos desde que la placa Arduino comenzó a ejecutar el programa actual.
* **“!”**, se utiliza para negar el valor de una variable o una función.

## **➪Efecto Estroboscópico**

El efecto visual en el cual se basa el proyecto es el estroboscópico, este es un efecto óptico que se produce cuando iluminamos con destellos un objeto que se mueve de forma rápida y periódica. Lo que genera se ve un objeto detenido, a pesar de que en realidad siga en funcionamiento, lo que se hace para generar este efecto es iluminar con una fuente de luz de rápida acción la cual debe apagarse a la misma frecuencia que la velocidad de giro del objeto “detenido”.

La causa por la cual vemos al objeto detenido radica en el sincronismo entre el momento en el cual el cuerpo en rotación se ilumina por un instante, este instante debe coincidir con la misma posición angular de la rotación, por ende siempre se verá la misma zona del cuerpo ubicada en el mismo lugar. El resto del movimiento angular no se percibe porque está a oscuras.

Cuando la sincronización entre el instante en que se enciende la luz y la velocidad de rotación del cuerpo no es exacta, lo que se aprecia es un lento giro del cuerpo en una u otra dirección en dependencia de si la luz se adelanta o atrasa en sincronismo.

A nivel anatómico lo que sucede es que en el caso de la visión, el detector fisiológico (el ojo) es incapaz de distinguir entre dos imágenes que llegan simultáneamente. Por esa razón, al recibir una sucesión rápida de estos estímulos los interpreta como un movimiento continuo ya que el ojo humano retiene una imagen aproximadamente 1/16 de un segundo.

## **➪Estroboscopio**

El proyecto en esencia es un estroboscopio con fines didácticos en vez de prácticos (es para lo que se lo utiliza generalmente). Este es un instrumento utilizado para hacer que lo objetos que se mueven circularmente parezca que se mueven a otra velocidad, o que ni siquiera se muevan. En referencia al apartado anterior, es la materialización del efecto estroboscópico.

Para desarrollar uno se necesita esencialmente una lámpara, en este caso una led, esta va a emitir destellos consecutivamente a una frecuencia regulable (a través de un potenciómetro en nuestro caso). Los estroboscopios que se encuentran en el mercado se utilizan para medir la frecuencia, estos trabajan a una de 50hz a 60hz.

# **Construcción del software**

## **➪Potenciómetro**

El potenciómetro nos proporciona una resistencia variable según vayamos modificando su posición. Si está totalmente cerrado obtendremos como salida el máximo voltaje (el de entrada), si lo tenemos totalmente abierto, obtendremos 0 voltios y si lo tenemos en una posición intermedia obtendremos una fracción del voltaje de entrada proporcional a la posición en la que se encuentre. Este comportamiento se llama divisor de tensión. A su vez este va conectado a una entrada analogica, esto nos permite que el valor que devuelva el potenciómetro se de entre 0 y 1023, en lugar de 0 a 255(en caso de que la entrada no fuera analogica). En nuestro proyecto este valor está directamente relacionado con el tiempo que se encuentra apagada la led.

## **➪IMU**

Se trata de un dispositivo capaz de medir la fuerza (aceleración) y la velocidad. Genéricamente consta de un Acelerómetro y un Giroscopio. El MPU-6050 es una IMU de 6DOF (se lee “6 Degrees Of Freedom“ o sea 6 grados de libertad). Esto significa que lleva un acelerómetro y un giroscopio, ambos de 3 ejes (3+3 = 6DOF)

## **➪MPU 6050**

Es un componente utilizado en la electrónica el cual consta de un acelerómetro y un giroscopio, este es un sensor de movimiento, que tiene un conversor ADC de 16 bits que convierte los datos a un valor digital, el módulo de giroscopio se comunica con el Arduino a través de la comunicación serie I2C a través del reloj serial (SCL) y datos (SDA) que puede conectarse a través de los pines A4 / A5 (SDA / SCL) en la placa Arduino Uno, el chip MPU 6050 necesita 3.3V.

### **Acelerómetro[[2]](#footnote-1)**

El acelerómetro mide (evidentemente) la aceleración, esta puede expresarse en 3 ejes: X, Y e Z, las tres dimensiones del espacio. Por ejemplo, si se mueve la MPU hacia arriba, el eje Z marcará un cierto valor. Si es hacia delante, marcará el eje X, etc.

Como ya sabemos la aceleración gravitacional es de aproximadamente 9.8 m/s² y es perpendicular al suelo. Así pues, la MPU se basa en detectar la aceleración de la gravitacional. Gracias a la gravedad terrestre se puede usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y.

Basándonos en el caso en el que la MPU esté perfectamente alineado con el suelo, en este ejemplo el eje Z marcará 9.8, y los otros dos ejes marcarán 0. Ahora supongamos que giramos la MPU 90 grados. Ahora es el eje X el que está perpendicular al suelo, por lo tanto marcará la aceleración gravitacional.[[3]](#footnote-2)

Si sabemos que la aceleración gravitacional es de 9.8 m/s², y sabemos qué medida dan los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la MPU.[[4]](#footnote-3)

Dado que el ángulo se calcula a partir de la gravedad, no es posible calcular el ángulo Z con esta fórmula ni con ninguna otra. Para hacerlo se necesita otro componente: el magnetómetro, que es un tipo de brújula digital. El MPU-6050 no lleva, y por tanto nunca podrá calcular con precisión el ángulo Z. Sin embargo, para la gran mayoría de aplicaciones sólo se necesitan los ejes X e Y.

### **Giroscopio**

El giroscopio mide la velocidad angular, esto es el número de grados que se gira en un segundo.

Si sabemos el ángulo inicial de la IMU, podemos sumarle el valor que marca el giroscopio para saber el nuevo ángulo a cada momento. Supongamos que iniciamos la IMU a 0º. Si el giroscopio realiza una medida cada cierto intervalo de tiempo, por ende marcara un cierto valor en el eje Y.[[5]](#footnote-4)

Lo mismo pasa con los ejes X, Z. Sólo que se suele ignorar el eje Z, puesto que al no poder calcular un ángulo Z con el Acelerómetro, no se puede aplicar un Filtro Complementario para el eje Z.

**Conexiones del MPU 6050[[6]](#footnote-5)**

Las conexiones son del modo I2C estándar

| ***MPU 6050*** | ***Arduino Uno, Nano, Mini*** |
| --- | --- |
| VCC | 5V |
| GND | GND |
| SCL | A5 |
| SDA | A4 |

### **Errores en las medidas**

Hay dos problemas muy importantes: el ruido y los errores. El ruido son todas aquellas interferencias que afectan a los dispositivos electrónicos. El acelerómetro es capaz de medir cualquier ángulo, sin embargo sus lecturas son ruidosas y tienen un cierto margen de error.[[7]](#footnote-6)

Por si esto fuera poco, el acelerómetro también detecta cualquier aceleración que no sea la de la gravedad. Por tanto, si se mueve el MPU sin girarlo, al aplicar una aceleración en otro eje, el MPU lo detectará como un cambio de rotación.

Por otra parte tenemos el giroscopio. A diferencia del acelerómetro, da las medidas con mucha precisión. Pero al realizar los cálculos del ángulo es inevitable que se produzca un pequeño error, que con el tiempo va acumulándose hasta que cualquier similitud con la realidad es pura coincidencia. Esto en inglés se llama drift.

Hay una (varias) forma(s) de combinar los datos del acelerómetro y el giroscopio para así obtener medidas adecuadas. Para esto necesitamos ver lo que es el filtro.

### **Filtro**

La idea es muy simple: hay que conseguir eliminar el ruido, el drift y conseguir que el acelerómetro no cambie de ángulo al detectar otra fuerza que no sea la gravedad.

Hay distintos algoritmos, llamados filtros, que hacen esta tarea. Uno de los mejores es el famoso ***Filtro de Kalman*** el cual se utiliza en los aviones, cohetes y satélites geoestacionarios.

El filtro de Kalman es sorprendente. Es considerado uno de los mayores hallazgos del siglo XX, capaz de calcular el error de cada medida a partir de las medidas anteriores, eliminarlo y dar el valor real del ángulo.

Sin embargo tiene dos problemas:

1. Tiene un coste de procesamiento algo elevado
2. Es complejo de entender

Así pues, tenemos otros filtros a nuestra disposición. El que vamos a utilizar es conocido como Filtro Complementario o Complementary Filter (en inglés), este es ideal para implementar con Arduino: fácil de utilizar, bajo coste de procesamiento y con una precisión muy buena.

¿En qué consiste exactamente? El Filtro Complementario es en realidad una unión de dos filtros diferentes: un High-pass Filter[[8]](#footnote-7) para el giroscopio y un Low-pass Filter[[9]](#footnote-8) para el Acelerómetro. El primero deja pasar únicamente los valores por encima de un cierto límite, al contrario que el Low-pass filter, que sólo permite a los que están por debajo.[[10]](#footnote-9)

### **Calibración[[11]](#footnote-10)**

La calibración es necesaria ya que el sensor MPU-6050 probablemente no se encuentre 100% en una posición horizontal, esto se da debido a que el sensor al ser soldado en el módulo puede estar desnivelado agregando un error en cada componente, además cabe destacar también que el módulo estará ubicado en nuestra mano, lo que no sería algo muy estable y difícil de asegurar que esté completamente horizontal.

Para solucionar este problema, se puede configurar en el módulo MPU6050 offsets y de esta forma compensar los errores que podamos tener.

Para calibrar las lecturas del sensor MPU6050, con el código de Calibración, se debe realizar la conﬁguración de escalas y unidades que se va a tomar de referencia, para ello, el sensor debe estar en una posición recta, que será tomada como nuestra posición de "set", además de no moverlo durante la ejecución del código de Calibración. Al ﬁnalizar está mostrarán los valores de lectura de cada eje en cada uno de los sensores, estos datos serán tomados de referencia para el código de Lectura. Al abrir el código de Lectura, se deﬁnen las escalas del giroscopio y del acelerómetro, también los valores obtenidos en el código de Calibración. En las escalas del giroscopio y del acelerómetro se deﬁnenlos valores máximos y mínimos de medición, de acuerdo con el datasheet del MPU 06050. Siendo el valor de 250 el más bajo para el giroscopio, y para el acelerómetro 2g.

### **Código final[[12]](#footnote-11)**

#### **Declaración de las variables y librerías utilizadas**

En esta parte del código declaramos las variables a utilizar, de la misma forma incorporamos la librería Wire.() la cual establece la comunicación I2C entre la placa Arduino con el MPU 6050. En esta parte también definimos ciertos datos de conversión (como de radianes a grados) y datos tabulados que son necesarios para la lectura apropiada por parte del MPU.

#### **Setup del código**

En el setup inicializamos la comunicación I2C, el MPU, la comunicación serial (a 57600 baudios) y establecemos el pin nueve como salida (es el pin de la led).

#### **Función de control del parpadeo de la led a través del potenciómetro**

En esta parte del código establecimos el parpadeo de la led a traves de un potenciómetro como una función específica de tipo void (ya que no devuelve datos).

#### **Función del MPU - 6050**

Esta es la primera parte de la función dedicada al MPU, o sea a la toma de datos, la transformación de los mismos (a grados) y el filtrado de los mismos (en este orden); primero se toman los datos del acelerómetro y luego del giroscopio. Luego de esta parte (la cual es la principal) al final se encuentran serial prints de los valores finales para visualizar en el monitor serial.

#### **Void loop**

En el loop establecemos un condicional (if) del cual depende el encendido de la led, esta se encenderá si en los valores raw del acelerómetro en eje z se detecta una aceleración notable (en este caso 23000, este número fue obtenido experimentalmente). Luego si se cumple esta condición el parpadeo será controlado a través del potenciómetro (declarando la función de la frecuencia en el loop) y la variación de la posición del MPU.

# **Construcción del Hardware**

## [➪](#_i1y4n2aw400m)**Componentes y funciones en el circuito[[13]](#footnote-12)**

* Arduino nano: el Arduino Nano es una pequeña y completa placa basada en el ATmega328. Tiene más o menos la misma funcionalidad que el Arduino Uno, pero con una presentación diferente. Este no posee conector para alimentación externa, y funciona con un cable USB Mini-B (incluido) en vez del cable estándar. Por lo cual cumple la misma función que un arduino uno, se utiliza este en vez del proporcionado por el curso ya que es más práctico a los efectos de este proyecto.
* Potenciómetro: el potenciómetro de variación lineal nos permite modificar el voltaje de una forma lineal. Encontraremos potenciómetros de diferentes resistencias, la más típica es la de 10 kΩ. Normalmente tienen 3 patillas, según el potenciómetro que utilicemos, deberemos identificar la funcionalidad de cada patilla. Una patilla irá conectada a la fuente de alimentación, otra a tierra o GND y por último la tercera patilla será la salida del potenciómetro.
* Mosfet: se le denomina así por ser Metal Óxido Semiconductor Field Effect Transistor. Es un transistor de efecto de campo con tres patillas. Conduce la corriente eléctrica entre dos de sus patillas (las exteriores) cuando aplicamos tensión en la patilla del centro. Funciona como un interruptor que se activa por tensión, por esto sería necesario para el proyecto ya que sería quien apague y prenda la led. Por esto utilizamos en particular utilizamos un mosfet of fast switching, debido a las altas frecuencias que debíamos hacer parpadear la led.
* 10 k and 100 Ohm resistores: Cómo utilizamos dichos mosfet fueron necesarias estas resistencias para el bajo o alto flujo de la corriente eléctrica respectivamente.
* Convertidor Ajustable Tensión: al utilizar una batería como alimentación para el proyecto fue necesario utilizar este regulador de tensión para controlar y estabilizar el voltaje de corriente continua, manteniendo el voltaje del circuito relativamente cerca de un valor deseado. La idea es que la corriente producida por la fuente no dañe ninguno de los otros componentes del sistema
* Led 50w: esta led ilumina lo suficiente para que el efecto estroboscópico se pudiera observar a simple vista sin dificultad.
* Batería lipo 12v - 1500mAh: es la fuente de alimentación de nuestro proyecto. Ideal para los requerimientos de todos los componentes.

# **Análisis de los resultados**

## [➪](#_i1y4n2aw400m)**Resultados del código y el armado**

La parte central de este proyecto depende firmemente de la precisión del código, por lo cual en el desarrollo del mismo hemos encontrado varios altercados, viéndonos en la situación de estar modificando varias partes del mismo. Luego al pasar todo lo pensado de manera lógica nos encontramos frente a diferentes problemas a la hora del armado también.

Estos son algunos de los problemas que hemos registrado y la solución encontrada a los mismos.

1. **Filtro:** como ya marcamos previamente el código original de este proyecto (el cual utiliza el desarrollador original) utiliza el filtro de Kalman para eliminar los errores generados en los valores de los grados, pero al ser un filtro que necesita de mucho más conocimiento y tiempo del que tenemos decidimos cambiarlo por otro. El que encontramos fue el filtro complementario, el cual no es tan refinado y no posee la sutileza del filtro de Kalman, pero que de todas formas nos es funcional y se adecua al nivel necesario. Es interesante remarcar que como alternativa también consideramos hacer un promedio, pero nos decidimos por el filtro complementario ya que deja lugar a menos errores.
2. **Error acumulado en el eje z:** luego de elegir el filtro complementario, investigando vimos que se generaba un error en el eje z (en los valores transformados) ya que el filtro no “funcionaba” en este. Este problema debía solucionarse ya que un tiempo en el loop dependía (a pesar de ser mínimo) de este valor, por lo cual decidimos cambiar el ángulo utilizado para este último tiempo del loop, por el valor filtrado del ángulo en x.
3. **NAN:** Este error es causado debido a la fórmula trigonométrica que utilizamos para calcular el ángulo que devuelve el acelerómetro. Utilizamos la función atan (arcotangente) y dentro un cociente que recibe los datos de los tres ejes, cuando los que están en el denominador son 0, aparece este error Not a Number, ya que la división entre 0 no es posible.
4. **Problema en el condicional del loop:** luego de desarrollado el código, cuando tratamos de probarlo físicamente la led no se encendía, por lo cual nos dimos cuenta de que el problema se encontraba en el condicional del loop. Luego de una reunión con el profesor asistente (Matías García) nos hizo notar que teníamos dos variables con el mismo nombre, estas se utilizaban para propósitos totalmente diferentes, una de ellas estaba en el condicional, por lo cual la solución fue cambiarle el nombre.
5. **Parpadeo final excesivamente largo:** habíamos definido una función global que mide el tiempo que transcurre desde que el programa se inicia, luego cuando entraba al loop habíamos definido la misma función (de forma local), al ya existir de forma global la función cuando el programa entraba al loop la función se redefinió pero tomaba como dato lo que ya había leído anteriormente, generando que dicho tiempo se duplicara. Lo solucionamos definiendo dos funciones locales, para lo que la necesitábamos en cada caso.
6. **Led con driver:** uno de los problemas a la hora del armado fue que adquirimos una led con drivers, la diferencia de de esta con otra es que los drivers estan para conectar a la misma a una corriente alterna, de esta forma, impidiendo conectarla a nuestra batería de corriente continua. Una solución momentánea fue encontrar las “trazas” de la led y conectarle los cables (uno para el borne positivo y otro al negativo), la corriente llegaba a la led pero esta no encendía por lo cual pensamos que podía ser un problema en el código. Para verificar esto nuestro profesor (Guillermo) utilizó un osciloscopio[[14]](#footnote-13)para verificar si había alguna irregularidad en el código analizando la señal[[15]](#footnote-14) producida por el mismo. Como el código funcionaba perfectamente concluimos que el problema debían de ser la led en sí por lo cual la solución final fue adquirir otra led que no viniera con drivers integrados.
7. **Mosfet:** los mosfet que se requerían para este proyecto no fueron encontrados en las casas de electrónica, por lo cual nuestro profesor tutor (Guillermo Airaldi) investigo y luego desarrolló una lista de modelos de mosfet que cumplian la misma funcion y se encontraban en estas casas.

# **Conclusiones**

Al igual que todos debimos adaptarnos a esta modalidad tan particular de este año, llevar a cabo un proyecto práctico a distancia fue todo un desafío. Nos sentimos satisfechas en lo que respecta al desarrollo lógico del proyecto tanto como el armado del mismo, a pesar de ciertos problemas, los cuales son totalmente naturales en los proyectos, llegamos a cumplir con los objetivos que nos propusimos al comienzo del semestre.

Tener este tipo de talleres al principio de la carrera nos parece muy motivador y un gran primer acercamiento a lo relativo con nuestra orientación. Además de esto nos da a conocer lo que es llevar a cabo un proyecto, más aún de una forma no presencial lo cual exige un verdadero interés por lo que se quiere hacer y mucha autonomía a la hora de estudiar sobre aquello que no conocemos y que será necesario para el proyecto.

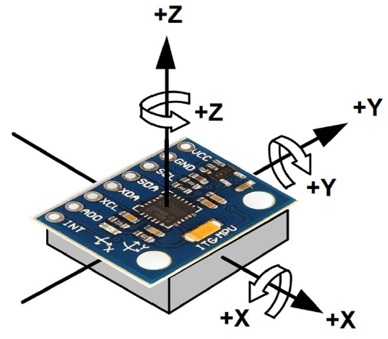
Fue una experiencia muy enriquecedora de todos los puntos de vista, desde el proyecto que elegimos a realizar, planteandonos un verdadero desafío pero despertando muchísimo entusiasmo. Creamos nuevas relaciones dentro del equipo tanto de compañerismo como de amistad, generando comprensión y compromiso de nuestra parte. Sobre todo queremos remarcar la excelente disposición y motivación constante de los tutores que estaban a cargo de nuestro equipo, gracias Guillermo y Matías por la contención y amabilidad ante todas las dudas y problemas que tuvimos a lo largo de todo el proceso

# **Bibliografía**

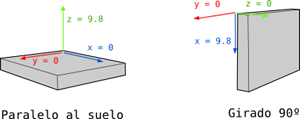
* Foros de discusión (son solo algunos a los cuales consultamos):
  + <https://forum.arduino.cc/index.php?topic=355427.0>
  + <https://forum.arduino.cc/index.php?topic=422392.0>
  + <https://forum.arduino.cc/index.php?topic=445636.0>
  + <https://forum.arduino.cc/index.php?topic=173842.0>
  + <https://forum.arduino.cc/index.php?topic=383006.0>
  + <https://forum.arduino.cc/index.php?topic=481774.0>
  + <https://forum.arduino.cc/index.php?topic=394659.0>
  + <https://forum.arduino.cc/index.php?topic=618940.0>
  + <https://forum.arduino.cc/index.php?topic=634495.0>
  + <https://forum.arduino.cc/index.php?topic=273086.0>
* Teórico
  + https: //[www.diarioelectronicohoy.com/blog/configurar-el-mpu6050](http://www.diarioelectronicohoy.com/blog/configurar-el-mpu6050)
  + <https://www.hobbytronics.co.uk/accelerometer-info>
  + <https://makersportal.com/blog/2019/8/17/arduino-mpu6050-high-frequency-accelerometer-and-gyroscope-data-saver>
  + <https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html>
  + <https://www.youtube.com/watch?v=SVI_NldMjlE>
  + <https://www.youtube.com/watch?v=awjXzXnRuwo>
  + <https://www.prometec.net/usando-el-mpu6050/>
  + <https://piensa3d.com/tutorial-programacion-arduino-acelerometro/>
  + <https://www.youtube.com/watch?v=bgCHOQUq734>

# **Anexo**

* Imagen (I)



* Imagen (II)



* Imagen (III)

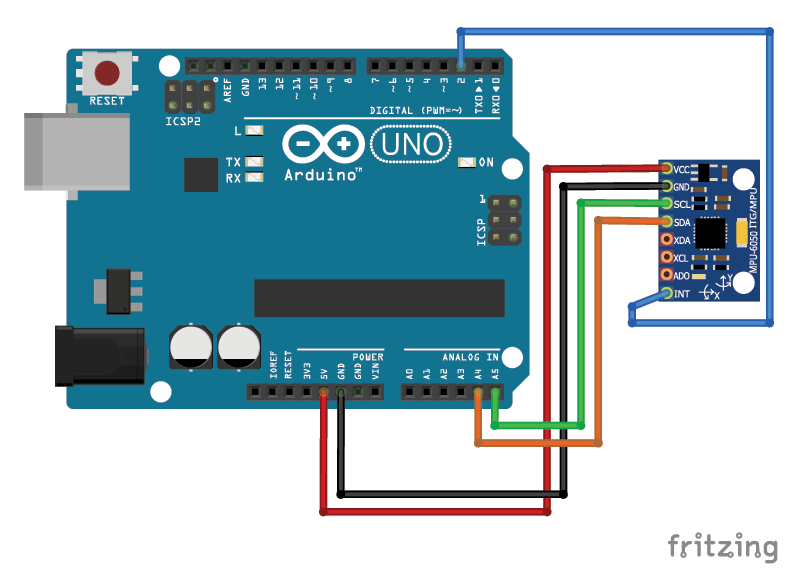


* Imagen (IV)

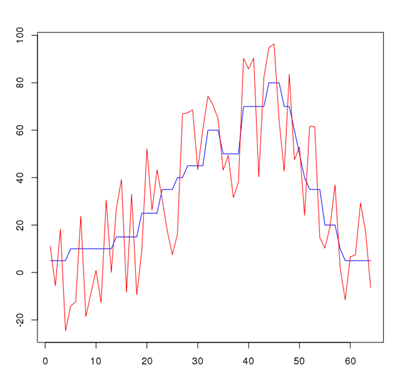


Comentario: Dónde Δt es el tiempo que transcurre cada vez que se calcula esta fórmula, AnguloYAnterior es el ángulo calculado la última vez que se llamó esta fórmula y GiroscopioY es la lectura actual del ángulo Y del giroscopio. *(Por ejemplo: si en el instante t=0 se inicia la IMU a 0º, y al cabo de 0’5 segundos el giroscopio marca 12º, el nuevo ángulo sería 6º)*

* Imagen (V)



* Imagen (VI)



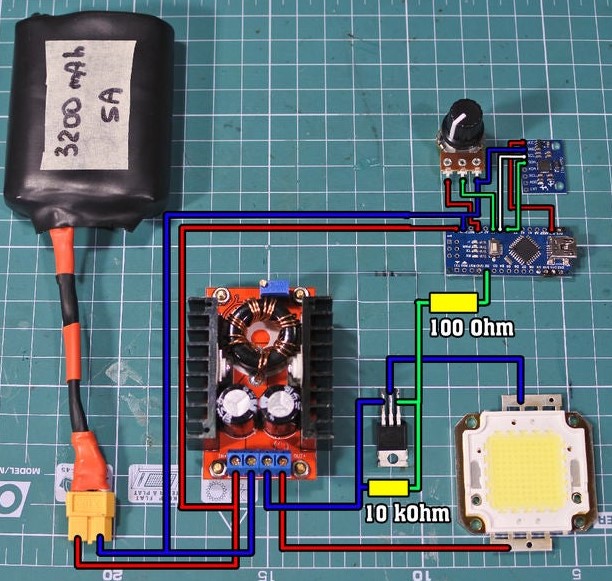
Comentario: este es un gráfico de las medidas de un acelerómetro en función del tiempo, el ángulo real (ideal) está marcado en azul, y las lecturas reales del sensor están en rojo.

* Imagen (VII)

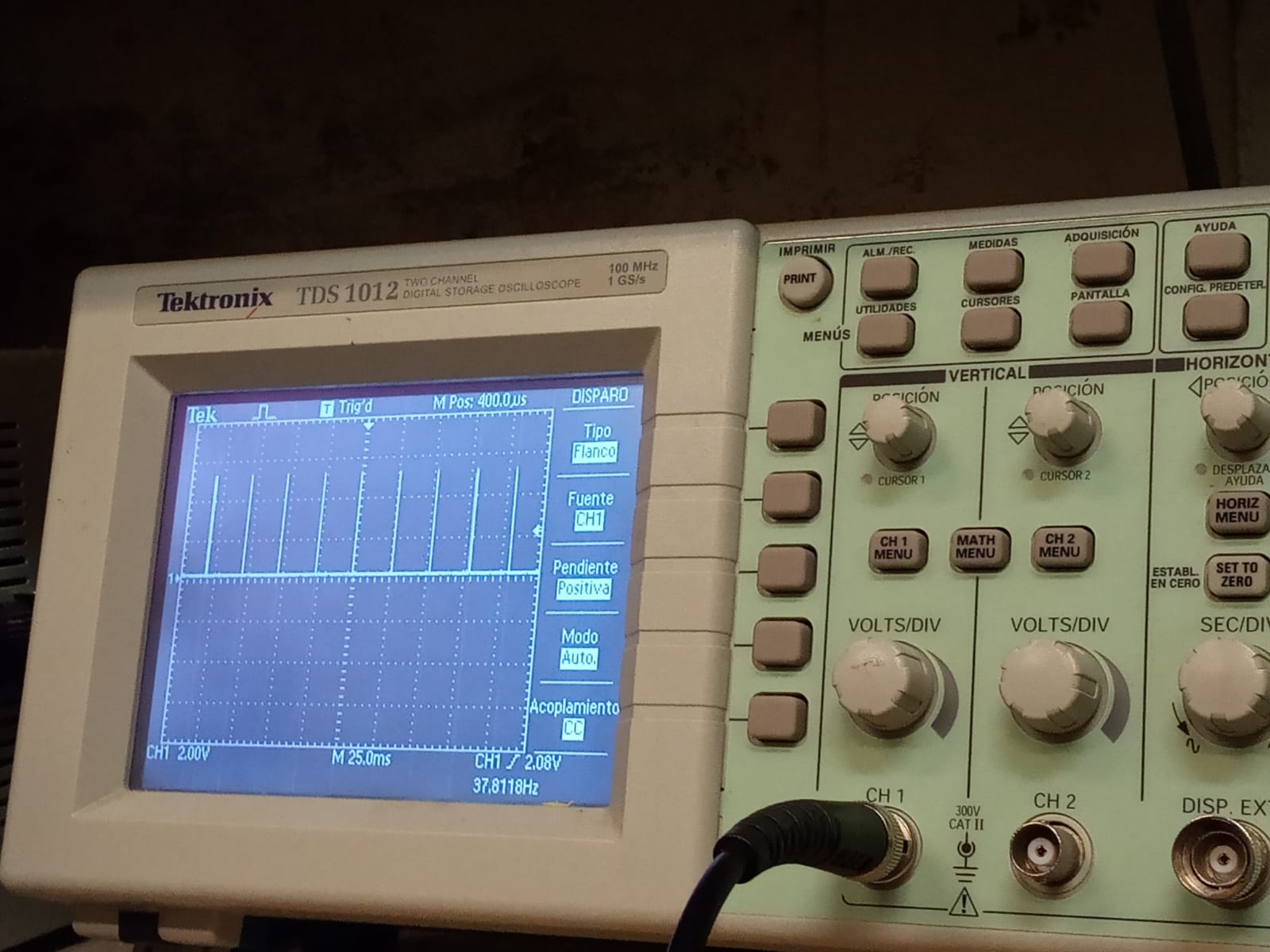


Comentario: en esta formula AnguloGyro es el ángulo del Giroscopio que hemos calculado previamente, y AnguloAccel con el ángulo del Acelerómetro calculado con la fórmula de la tangente. Δt es el tiempo que ha pasado (en segundos) desde la última vez que se ha calculado el filtro. Esta fórmula es la misma para el eje X, Y.[[16]](#footnote-15)

* Imagen (VIII)

2

* Imagen (IX)



**Código para la calibración de valores del MPU 6050[[17]](#footnote-16)**

#include <Wire.h>

#include <MPU6050.h>

#include <I2Cdev.h>

// Librerias I2C para controlar el mpu6050

// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo

// del estado de AD0. Si no se especifica, 0x68 estará implicito

MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z

int ax, ay, az;

int gx, gy, gz;

//Variables usadas por el filtro pasa bajos

long f\_ax,f\_ay, f\_az;

int p\_ax, p\_ay, p\_az;

long f\_gx,f\_gy, f\_gz;

int p\_gx, p\_gy, p\_gz;

int counter=0;

//Valor de los offsets

int ax\_o,ay\_o,az\_o;

int gx\_o,gy\_o,gz\_o;

void setup() {

Serial.begin(57600); //Iniciando puerto serial

Wire.begin(); //Iniciando I2C

sensor.initialize(); //Iniciando el sensor

if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");

// Leer los offset los offsets anteriores

ax\_o=sensor.getXAccelOffset();

ay\_o=sensor.getYAccelOffset();

az\_o=sensor.getZAccelOffset();

gx\_o=sensor.getXGyroOffset();

gy\_o=sensor.getYGyroOffset();

gz\_o=sensor.getZGyroOffset();

Serial.println("Offsets:");

Serial.print(ax\_o); Serial.print("\t");

Serial.print(ay\_o); Serial.print("\t");

Serial.print(az\_o); Serial.print("\t");

Serial.print(gx\_o); Serial.print("\t");

Serial.print(gy\_o); Serial.print("\t");

Serial.print(gz\_o); Serial.print("\t");

Serial.println("nnEnvie cualquier caracter para empezar la calibracionnn");

// Espera un caracter para empezar a calibrar

while (true){if (Serial.available()) break;}

Serial.println("Calibrando, no mover IMU");

}

void loop() {

// Leer las aceleraciones y velocidades angulares

sensor.getAcceleration(&ax, &ay, &az);

sensor.getRotation(&gx, &gy, &gz);

// Filtrar las lecturas

f\_ax = f\_ax-(f\_ax>>5)+ax;

p\_ax = f\_ax>>5;

f\_ay = f\_ay-(f\_ay>>5)+ay;

p\_ay = f\_ay>>5;

f\_az = f\_az-(f\_az>>5)+az;

p\_az = f\_az>>5;

f\_gx = f\_gx-(f\_gx>>3)+gx;

p\_gx = f\_gx>>3;

f\_gy = f\_gy-(f\_gy>>3)+gy;

p\_gy = f\_gy>>3;

f\_gz = f\_gz-(f\_gz>>3)+gz;

p\_gz = f\_gz>>3;

//Cada 100 lecturas corregir el offset

if (counter==100){

//Mostrar las lecturas separadas por un [tab]

Serial.print("promedio:"); Serial.print("t");

Serial.print(p\_ax); Serial.print("\t");

Serial.print(p\_ay); Serial.print("\t");

Serial.print(p\_az); Serial.print("\t");

Serial.print(p\_gx); Serial.print("\t");

Serial.print(p\_gy); Serial.print("\t");

Serial.println(p\_gz);

//Calibrar el acelerometro a 1g en el eje z (ajustar el offset)

if (p\_ax>0) ax\_o--;

else {ax\_o++;}

if (p\_ay>0) ay\_o--;

else {ay\_o++;}

if (p\_az-16384>0) az\_o--;

else {az\_o++;}

sensor.setXAccelOffset(ax\_o);

sensor.setYAccelOffset(ay\_o);

sensor.setZAccelOffset(az\_o);

//Calibrar el giroscopio a 0º/s en todos los ejes (ajustar el offset)

if (p\_gx>0) gx\_o--;

else {gx\_o++;}

if (p\_gy>0) gy\_o--;

else {gy\_o++;}

if (p\_gz>0) gz\_o--;

else {gz\_o++;}

sensor.setXGyroOffset(gx\_o);

sensor.setYGyroOffset(gy\_o);

sensor.setZGyroOffset(gz\_o);

counter=0;

}

counter++;

}

Comentario: el programa básicamente está modificando constantemente los offset intentando eliminar el error con la medida real que deseamos, en esta caso ax=0, ay=0, az=1g y gx=0, gy=0, gz=0.

Código final

int LED=9;

int POT=A0;

int valor\_pot;

int encendi2 = 200; //luces prendidas, o sea es un tiempo que se asigna de manera aleatoria

int apaga3; //luces apagadas delay

int Frec\_max=50;// max apagado delay

int Frec\_min=10;// min apagado delay

#include <Wire.h>

#define MPU 0x68

#define A\_R 16384.0

#define G\_R 131.072

#define RAD\_A\_DEG = 57.29577951

int16\_t AcX, AcY, AcZ, GyX, GyY, GyZ;

float Acc[2]; // angulo del acelerometro (X e Y)

float Gy[3];//giroscopio

float Angle[3];//resultado del filtro donde [0] corresponde a x, [1] corresponde a y.

String valores\_x;

String valores\_y;

String valores\_z;

int angulito;//angulo a calcular del cual va a depender el tiempo

long tiempo\_prev;

long timex;

boolean flag;//se va a usar para apagar o prender la led

float dt;//usado para el filtro

void setup (){

Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL

Wire.beginTransmission(MPU); //comunicacion I2C del acelerómetro con el arduino

Wire.write(0x6B); //primer registro

Wire.write(0); //inicializamos el MPU (primer pulso de 0)

Wire.endTransmission(true);//termina la transmisión al aparato esclavo que se comenzó

//en el paso anterior (beginTransmission), ademas transmite los bytes que se pidieron en el write.wire()

Serial.begin(57600); // comunicación serial del arduino con la compu

pinMode(9,OUTPUT);

}

void Frec\_dubidubi() {

//es para calcular el tiempo que esta apagada la led

valor\_pot=analogRead(POT);

apaga3= map (valor\_pot,0, 1023, Frec\_min, Frec\_max);

digitalWrite(LED,HIGH);//prendemos la led

delayMicroseconds(encendi2); //perate tantito

digitalWrite(LED,LOW); //se apaga la led

delay(apaga3); //perate tantito

}

void valoresMPU(){

//Leer los valores del Acelerometro de la IMU

Wire.beginTransmission(MPU); //comienza y manda la dirección del esclavo (previamente declarada)

Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX

Wire.endTransmission(false);//mantiene la transmision para luego

Wire.requestFrom(MPU,6,true);//pedir 6 registros (a partir del 0x3B)

AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros

AcY=Wire.read()<<8|Wire.read();

AcZ=Wire.read()<<8|Wire.read();

//A partir de los valores del acelerometro, se calculan los angulos Y, X

//respectivamente, con la formula de la tangente.

Acc[1] = atan(-1\*(AcX/A\_R)/sqrt(pow((AcY/A\_R),2) + pow((AcZ/A\_R),2)))\*RAD\_TO\_DEG;

Acc[0] = atan((AcY/A\_R)/sqrt(pow((AcX/A\_R),2) + pow((AcZ/A\_R),2)))\*RAD\_TO\_DEG;

// atan(arcotan), sqrt(raiz cuadrada), pow(base, exponente)

//Leer los valores del Giroscopio

Wire.beginTransmission(MPU);

Wire.write(0x43);

Wire.endTransmission(false);

Wire.requestFrom(MPU,6,true); //A partir del 0x43, se piden 6 registros

GyX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros

GyY=Wire.read()<<8|Wire.read();

GyZ=Wire.read()<<8|Wire.read();

//Calculo del angulo del Giroscopio

Gy[0] = GyX/G\_R;

Gy[1] = GyY/G\_R;

Gy[2] = GyZ/G\_R;

//dt = (millis() - tiempo\_prev) / 1000.0;

tiempo\_prev = millis();

//millis (milisegundos)

//Aplicar el Filtro Complementario o sea roll y pitch

Angle[0] = 0.98 \*(Angle[0]+Gy[0]\*(millis() - tiempo\_prev) / 1000.0) + 0.02\*Acc[0];

Angle[1] = 0.98 \*(Angle[1]+Gy[1]\*(millis() - tiempo\_prev) / 1000.0) + 0.02\*Acc[1];

// como no podemos calcular el filtro complementario para Z

//Integración respecto del tiempo para calcular el YAW

Angle[2] = (Angle[2]+Gy[2]\*(millis() - tiempo\_prev) / 1000.0);

//Mostrar los valores por consola

valores\_x = "Rotación en x: " +String(Angle[0]);

valores\_y = "Rotación en y: " +String(Angle[1]);

valores\_z = "Rotación en z: " +String(Angle[2]);

Serial.print(valores\_x);

Serial.print("|");

Serial.print(valores\_y);

Serial.print("|");

Serial.print(valores\_z);

Serial.print("|");

Serial.print(AcZ);

Serial.println("|");

delay(10);

}

void loop(){

valoresMPU();// obtiene los datos "procesados" por la funcion valoresMPU

if((AcZ>22000) && (millis () - timex > 300))// es laparte en la que la componente z detecta movimientos

{

flag = !flag; // alterna el LED , se usa para indicar el estado de la led

timex = millis(); //Devuelve el número de millisegundos desde que el programa comenzó a "correr"

}

if (flag == HIGH){

angulito = 250 - Angle[0];

Frec\_dubidubi();

delayMicroseconds(2000 + angulito\*10); //tiempo extra de espera

}

1. El buffer tiene una capacidad de almacenamiento de 128 bytes. [↑](#footnote-ref-0)
2. Imágen ilustrativa (I) en el anexo [↑](#footnote-ref-1)
3. Imagen ilustrativa (II) en el anexo [↑](#footnote-ref-2)
4. Imagen (III) de la fórmula en el anexo [↑](#footnote-ref-3)
5. Imagen (IV) de la fórmula en el anexo. [↑](#footnote-ref-4)
6. Imagen ilustrativa (V) en el anexo [↑](#footnote-ref-5)
7. Imagen ilustrativa (VI) en el anexo [↑](#footnote-ref-6)
8. <https://www.electronics-tutorials.ws/filter/filter_3.html> [↑](#footnote-ref-7)
9. <https://www.electronics-tutorials.ws/filter/filter_2.html> [↑](#footnote-ref-8)
10. Imagen (VII) ilustrativa de la combinación de los dos filtros [↑](#footnote-ref-9)
11. Se adjunta el código utilizado en el anexo. [↑](#footnote-ref-10)
12. Se adjunta el código final en el anexo [↑](#footnote-ref-11)
13. Imagen del circuito (VIII) en el anexo [↑](#footnote-ref-12)
14. *“(...) es un instrumento de medición para la electrónica. Representa una gráfica de amplitud en el eje vertical y tiempo en el eje horizontal. El osciloscopio presenta los valores de las señales eléctricas en forma de coordenadas en una pantalla, en la que normalmente el eje X (horizontal) representa tiempos y el eje Y (vertical) representa tensiones. La imagen así obtenida se denomina oscilograma.”*

    Extraido de: Final Test, <https://www.finaltest.com.mx/product-p/art-9.htm> [↑](#footnote-ref-13)
15. Imagen (IX) de la señal captada por el osciloscopio en el taller [↑](#footnote-ref-14)
16. Se puede cambiar el valor de 0.98 y 0.02 por un valor personalizado. Eso sí, ambos tienen que sumar 1. [↑](#footnote-ref-15)
17. Extraído de: https: //www.diarioelectronicohoy.com/blog/configurar-el-mpu6050 [↑](#footnote-ref-16)