



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Dispositivo para la evaluación del
proceso de sutura en alumnos de
segundo año**

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Octavio Anselmo Hernández Alvizo

ASESORA DE MATERIAL DIDÁCTICO

Dra. María del Pilar Corona Lira



Ciudad Universitaria, Cd. Mx., 2020

Índice

Índice de figuras	3
Agradecimientos	8
Introducción.	10
Manual para el profesor.	11
Práctica a realizar.	11
Descripción del sistema	11
Recomendaciones de sistema operativo y características.	12
Descripción y funcionamiento de la tarjeta Arduino®	14
Programando el Arduino UNO	17
Cómo funciona el programa Arduino	23
Pasos por seguir para poder usar Processing®, Arduino® y las MPU:	27
Manejo de los datos para la evaluación.	43
Uso de la plantilla	43
Manual para el estudiante.	46
Estandarización de la actividad	46
Primeros pasos	46
Materiales	48
Posición inicial para el registro	51
Como usar el dispositivo y los programas	54
Programas de ARDUINO®	55
Configuración de ARDUINOS ®	58
Ajustando el programa PROCESSING®	63
Interfaz Gráfica del Usuario PROCESSING®	64
Manejo de los datos para la evaluación	77

Validación del sistema.	81
Referencias bibliográficas	100
Anexos	101
Programa Arduino MPU6050	101
Programa en Processing	108

Índice de figuras

Figura 1 Vista del Sistema de captura y registro de movimientos para procedimientos quirúrgicos.	9
Figura 2 Vista frontal del Sistema de captura y registro de movimientos para procedimientos quirúrgicos.	10
Figura 3 Diseño de un Arduino.	12
Figura 4 Localización de cada entrada y salida, al igual que de cada componente.	15
Figura 5 Conexión de IMU MPU6050 con tarjeta Arduino UNO	16
Figura 6 Carpetas de los programas.	17
Figura 7 Líneas donde se introducen los valores obtenidos de cada MPU6050	17
Figura 8 Localización del número de la línea del programa en la que se encuentre.	18
Figura 9 Localización de los puertos COM en dispositivos	19
Figura 10 Configuración del puerto	20
Figura 11 Todas las variables COM# disponibles. Opciones avanzadas.	21
Figura 12 Carpetas de programas.	21
Figura 13 Ejemplo de líneas no comentadas las cuales se usan para la correcta realización del programa.	23
Figura 14 Ejemplo de líneas comentadas, donde las funciones no están siendo usada.	24
Figura 15 Carpetas de programas.	25
Figura 16 Localización de las librerías en Processing.	26
Figura 17 Interfaz de la biblioteca de las librerías y acciones que se pueden realizar.	27
Figura 18 Líneas del código fuente en donde se deben de indicar los puertos COM utilizados en cada computadora.	28
Figura 19 Interfaz gráfica en Processing. Ventana para registro.	28
Figura 20 Interfaz de botones (Botón Inicio).	29
Figura 21 Calibración	29
Figura 22 Gráfica en tiempo real del giroscopio de ambas IMUs.	30

Figura 23 Interfaz de botones (Botón Captura).	30
Figura 24 Estado del programa cuando deja de funcionar.	31
Figura 25 Administrador de tareas para poder cerrar los programas.	32
Figura 26 Interfaz cuando no registra los datos y entra en conflicto.	33
Figura 27 Error al no tener bien conectado los Arduinos, o alguna librería no está incluida en el programa, o alguno de los archivos necesarios no están incluidos en el sketch.	34
Figura 28 Archivos necesarios para la función correcta del programa.	34
Figura 29 Las líneas más comunes que pueden presentar algún conflicto en el programa.	35
Figura 30 Archivos necesarios para el uso de Matlab.	36
Figura 31 Líneas para identificar que modificar (COMX).	37
Figura 32 Interfaz de la aplicación en MATLAB®. Ventana para registro de participantes.	37
Figura 33 Gráfica en tiempo real del giroscopio de ambas IMUs.	38
Figura 34 Exportación de datos y su formato de entrega.	39
Figura 35 . Limpiar variables, funciones e interrupciones.	39
Figura 36 Error en los puertos COM.	40
Figura 37 Error mostrado al usar el mismo programa en diferentes computadoras y no configurarlas en la que se usa.	40
Figura 38 Inmovilización del sistema por la mala configuración.	41
Figura 39 Interfaz de la plantilla y los datos que brinda.	42
Figura 40 Manejo de datos en el archivo .txt.	43
Figura 41 Pestañas de trabajo para el manejo de datos.	43
Figura 42 Casilla inicial donde deben colocarse los datos del archivo con extensión .txt.	44
Figura 43 Plantilla con los datos analizados.	44
Figura 44 Técnica de higiene de manos según la OMS.	46
Figura 45 Colocación de la instrumentación al momento de realizar las pruebas.	49
Figura 46 Posición dentro del area delimitada para la prueba.	50
Figura 47 Colocación de los guantes médicos y guantes de los sensores.	51

Figura 48 Muestra de cómo deben de estar montados los sensores sobre la parte posterior del brazo, al igual que los cables siempre deberán de tener la dirección hacia los brazos.	52
Figura 49 Forma correcta de la sujeción de la pinza, esto se ilustra de manera correcta en la parte izquierda y en la parte derecha, la manera correcta de sujetar la piel, en este caso no se debe presionar la pinza, ya que en caso contrario se puede lastimar al paciente.	52
Figura 50 Ilustración de como sujetar correctamente la sutura Nylon 2-0, esta debe de estar sujeta a tres cuartos de la misma aguja, esto para poder atravesar todas las capas de la piel y en dado caso si se llegara a fracturar, poder extraer de manera sencilla el fragmento trozado.	52
Figura 51 Posición inicial para poder iniciar el registro. Así deberás de colocar la aguja y se ilustra la manera de como deberás manipular la cortada de la piel.	53
Figura 52 Conexión de IMU MPU6050 con tarjeta Arduino® UNO	54
Figura 53 Sketch de archivo de cada mano.	55
Figura 54 Líneas de código a introducir el OFFSET obtenido anteriormente con el profesor.	55
Figura 55 Puertos COM en el administrador de dispositivos.	56
Figura 56 Configuración del puerto	57
Figura 57 Opciones avanzadas	58
Figura 58 Archivos previamente configurados.	58
Figura 59 Localización de las librerías.	60
Figura 60 Interfaz de la ventana librerías.	60
Figura 61 Acciones que se pueden realizar en la ventana librerías.	61
Figura 62 Líneas a modificar del código en Processing.	61
Figura 63 Interfaz gráfica en Processing. Ventana para registro.	62
Figura 64 Interfaz entre el programa y usuario.	62
Figura 65 Línea de 10 segundos de calibración.	63
Figura 66 Gráfica en tiempo real del giroscopio de ambas IMUs.	63
Figura 67 Interfaz de selección en el objeto.	64

Figura 68 Error más frecuente en la interfaz, al no tener todas las características antes mencionadas o saturación en la memoria.	65
Figura 69 Solución rápida al estancamiento del programa.	66
Figura 70 Saturación de memoria en la interfaz de Processing.	67
Figura 71 Localización de la mala configuración del Processing.	67
Figura 72 Archivos del programa PROCESSING®.	68
Figura 73 Archivos básicos de MATLAB®.	68
Figura 74 Líneas del programa MATLAB® code a modificar.	69
Figura 75 Interfaz de la aplicación en MATLAB®. Ventana para registro de participantes.	70
Figura 76 Gráfica en tiempo real del giroscopio de ambas IMUs.	70
Figura 77 Exportación de datos y su formato de entrega.	71
Figura 78 Limpiar variables, funciones e interrupciones.	72
Figura 79 Error en los puertos COM.	72
Figura 80 Error mostrado al usar el mismo programa en diferentes computadoras y no configurarlas en la que se usa.	73
Figura 81 Inmovilización del sistema por la mala configuración.	73
Figura 82 Interfaz de la plantilla y los datos que brinda.	75
Figura 83 Azul cielo, datos sociodemográficos del participante.	75
Figura 84 Amarillo, variables Path Length (Distancia recorrida).	75
Figura 85 Verde, diferentes niveles evaluados.	76
Figura 86 Color oro se indica la tarea quirúrgica a realizar.	76
Figura 87 Color gris, resultado de las aceleraciones.	76
Figura 88 Manejo de datos en el archivo .txt.	77
Figura 89 Pestañas de trabajo para el manejo de datos.	77
Figura 90 Casilla inicial donde deben colocarse los datos del archivo .Txt.	78
Figura 91 Plantilla con los datos analizados.	78
Figura 92 Rangos del Shadow®.	79
Figura 93 Primer acercamiento del dispositivo de evaluación junto con el dispositivo Shadow (Correas sobre la persona).	80

Figura 94 Puntos de referencia que se usaron para establecer las mismas mediciones para tener una medida aproximada a la de un dispositivo comercial	81
Figura 95 Datos obtenidos de nuestro dispositivo al ser colocados en su posición inicial para su calibración.	82
Figura 96 Estas gráficas nos muestran los datos con nuestro dispositivo, como se puede observar se registra el movimiento, pero se puede ver un poco de ruido, lo que provoca que registre más datos de los necesarios.	83
Figura 97 Gráficas que nos muestran los datos con el dispositivo comercial.	83
Figura 98 Gráfica Fuerza vs Tiempo	84
Figura 99 Tabla de conversiones de valores	84
Figura 100 Ubicación del sensor para la estandarización	86
Figura 101 Visualización del dispositivo creado en conjunto con el comercial.	86
Figura 102 Primera prueba sin el ajuste los valores se entregaban en LSB.	87
Figura 103 Gráficas obtenidas de la segunda aplicación del dispositivo, con el ajuste aplicado.	87
Figura 104 Descripción de Unidades en las gráficas	88
Figura 105 Pruebas con el Shadow®.	88
Figura 106 Gráfica Fuerza vs Tiempo	89
Figura 107 Caracterización de la prueba con distancias.	90
Figura 108 Pruebas a 20 cm con el dispositivo creado sin la conversión de los datos sin la conversión de los LSB.	91
Figura 109 Muestra de una gráfica antes de la conversión de los LSB.	91
Figura 110 Gráficas del dispositivo con la conversión en G.	91
Figura 111 Muestra de las gráficas con las conversiones en G.	92
Figura 112 Pruebas a 20 cm con el dispositivo comercial	92
Figura 113 Muestra de las gráficas en el dispositivo comercial en G.	93

Agradecimientos

- A LA DRA. MARÍA DEL PILAR CORONA LIRA, QUE ME BRINDÓ SU APOYO A LO LARGO DE ESTE PROCESO DE TITULACIÓN, RESOLVIÉNDOME DUDAS, BUSCANDO ALTERNATIVAS QUE ME AYUDARAN A RESOLVER LOS PROBLEMAS, DANDO COMO RESULTADO UN ESCLARECIMIENTO ANTE LA INCERTIDUMBRE QUE SE TENÍAN A LO LARGO DE LA INVESTIGACIÓN DEL PROYECTO.
- AL DR. JOSÉ LUIS JIMÉNEZ CORONA, QUE GRACIAS A ESTOS AÑOS QUE TRABAJÉ CON ÉL, ME AYUDÓ MUCHO AL DESCUBRIR UNA NUEVA ÁREA EN LA QUE SE PUEDE INCURSIONAR PARA QUE MI CARRERA COMO INGENIERO APORTE UN PELDAÑO A LA MEDICINA, RESOLVIENDO LOS PROBLEMAS CONJUNTAMENTE QUE SE NOS PRESENTARON A LO LARGO DEL PROYECTO.
- A MIS PADRES QUE, AUNQUE NO LES EXPRESO TODOS MIS SENTIMIENTOS, LES AGRADEZCO DE TODO CORAZÓN QUE ME HAYAN BRINDADO SU APOYO TANTO EMOCIONA COMO ECONÓMICO PARA REALIZARME COMO INGENIERO.
- A MIS HERMANOS QUE, AUN CON LAS DIFERENCIAS QUE TENEMOS HEMOS SABIDO COMO APOYARNOS MUTUAMENTE PARA NO DEJAR DE PERSEGUIR LAS METAS QUE NOS PROPUSIMOS A LO LARGO DE ESTE TIEMPO.
- A LA UNAM, POR HABERME HECHO PARTICIPE DE ESTA GRAN UNIVERSIDAD QUE SIEMPRE SERÁ MI ALMA MATER, MI SEGUNDO HOGAR, QUE ADEMÁS DE HABERME BRINDADO GRANDES PROFESORES, AMIGOS Y AMISTADES, ME DIO LAS BASES QUE ME SEMBRARAN COMO UN HOMBRE DE BIEN ANTE LA SOCIEDAD, AL IGUAL QUE DARME LOS CONOCIMIENTO FUNDAMENTALES PARA CRECER TANTO EN LO PERSONAL, EMOCIONAL E INTELECTUALMENTE, A LO LARGO DE MI PASO POR LA UNIVERSIDAD.

- A LA DGAPA-UNAM QUE CON AYUDA DEL PROGRAMA PAPIME Y DEL PROYECTO PE202118 TITULADO “PE202217, SEGUNDA PARTE, ANÁLISIS DE VIDEO DE LA DESTREZA Y CARACTERIZACIÓN FÍSICO MECÁNICA DEL SIMULADOR”, ME APOYARON A LO LARGO DEL TRAYECTO PARA LA REALIZACIÓN DEL SIMULADOR Y LOGRAR RESULTADOS A TRAVÉS DE ESTE MATERIAL DIDÁCTICO PARA LA CONCLUSIÓN DE MIS ESTUDIOS.
- A MI AMIGOS, QUE GRACIAS A LA UNIVERSIDAD LOS CONOCÍ, BRINDÁNDOME SU APOYO Y AMISTAD, TENIENDO JUNTOS GRANDES EXPERIENCIAS QUE ME AYUDARON A SER MEJOR PERSONA Y EXPLORAR UN POCO MÁS SOBRE EL MUNDO EXTERIOR.

Introducción.

Con el avance tecnológico, se han generado diversos cambios en las propuestas de los planes de estudio; además, varias transformaciones han surgido en materias de carácter práctico en la carrera de Médico Cirujano de la Universidad Nacional Autónoma de México, donde se ha buscado una manera en la que el alumno adquiera y practique las destrezas manuales necesarias para ejercer su profesión y que se tradicionalmente han ido adquiriendo y transmitiendo por diferentes medios a lo largo de generaciones.

El desarrollo de dispositivo para la evaluación del proceso de sutura tiene como propósito evaluar la técnica de la sutura en alumnos que cursan la materia en el segundo año de la carrera. Se piensa que, con el presente dispositivo, al mismo tiempo que se evalúa el proceso de sutura en los alumnos, al profesor le servirá como apoyo para checar el desempeño de los estudiantes y su avance a lo largo del curso, esto con propósito de poder mejorar su desempeño en las destrezas manuales de los estudiantes, para asegurar que cuando se enfrenten a casos reales no incurran en errores médicos.

Manual para el profesor.

Practica a realizar.

Con ayuda de las siguientes imágenes se describirá el procedimiento con el que se realiza la preparación del sistema desarrollado para la evaluación del proceso de sutura en este caso se evaluar la actividad manual durante el llamado punto simple.

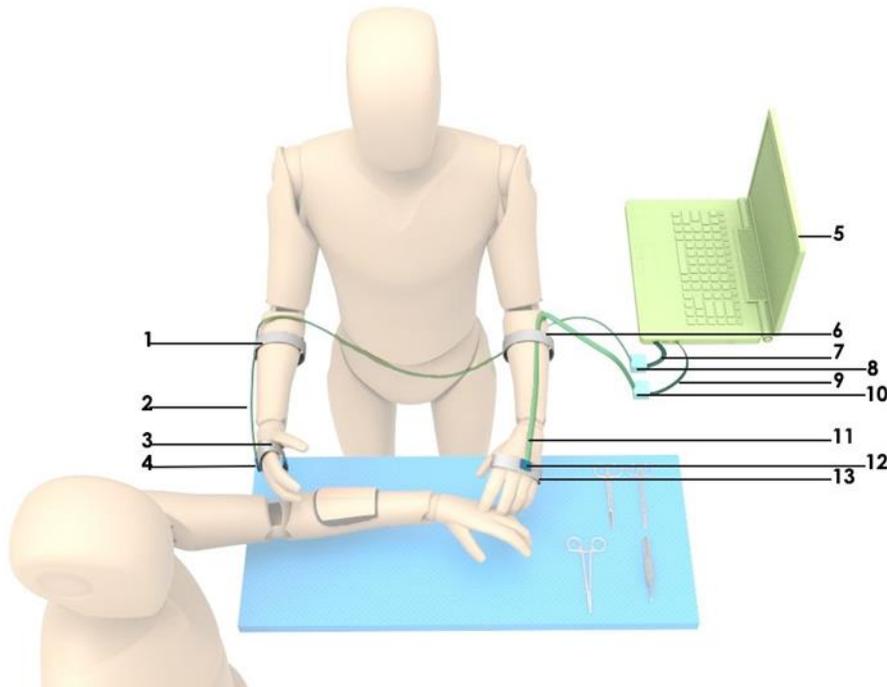


Figura 1 Vista del Sistema de captura y registro de movimientos para procedimientos quirúrgicos.

Descripción del sistema

El sistema consiste en dos Arduinos® UNO y dos Inertial Measurement Unit (IMU).

1. Elemento de sujeción ajustable brazo derecho, 2. Cable de conexión entre sensor de medición inercial y tarjeta de desarrollo, 3. Elemento de sujeción para mano derecha, 4. Sensor de medición inercial mano derecha, 5. Computadora, 6. Elemento de sujeción ajustable brazo izquierdo, 7. Cable de conexión entre sensor

de medición inercial y computadora, 8. Tarjeta de desarrollo, 9. Cable de conexión entre tarjeta de desarrollo y computadora, 10 Tarjeta de desarrollo, 11. Cable de conexión entre sensor de medición inercial y tarjeta de desarrollo, 12. Sensor de medición inercial mano izquierda y 13. Elemento de sujeción para mano izquierda

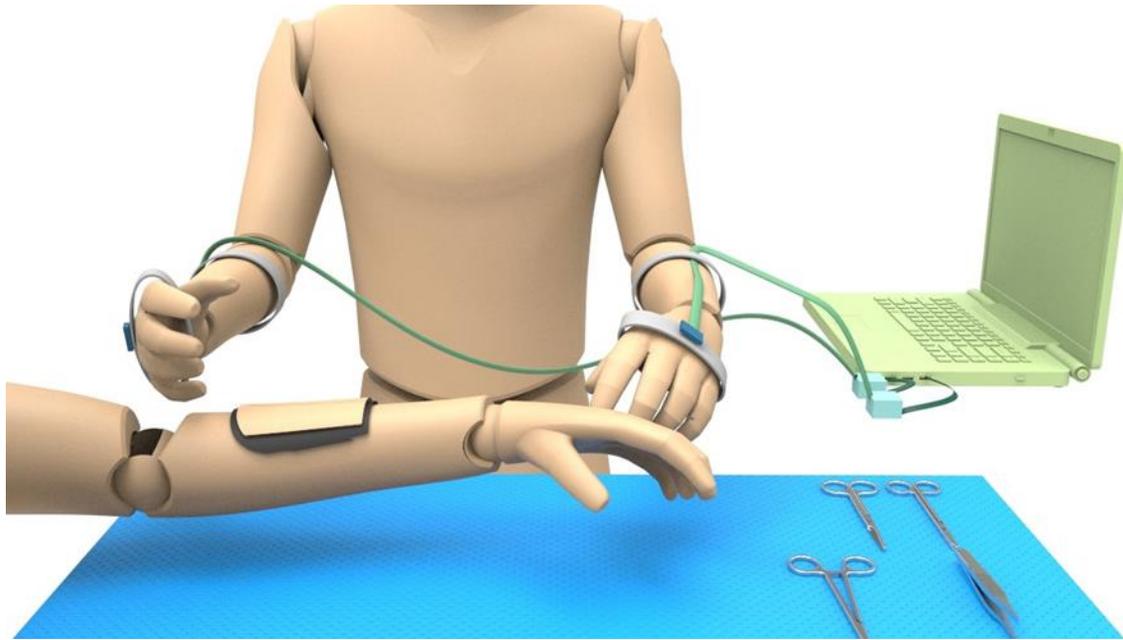


Figura 2 Vista frontal del Sistema de captura y registro de movimientos para procedimientos quirúrgicos.

La figura anterior nos muestra cómo debe de estar colocado el alumno frente a la instrumentación al igual que representa como deben colocarse los sensores sobre el alumno, también se ilustra la posición inicial del alumno como del paciente (La persona que estará usando el PAD).

Para saber que es el PAD, en este caso se ve reflejado en la figura 2 como un brazalete, este tiene un nombre de Dispositivo Entrenador para Sutura y es usado para el apoyo al estudiante, con el fin de ayudar en su desarrollo en habilidades de sutura.

Recomendaciones de sistema operativo y características.

Al saber que estamos trabajando con varios dispositivos a la vez y muchas veces estos ocupan muchos recursos que pueden ser tanto un recurso de hardware (dispositivos) como de software (programas), es necesario que en la computadora

que se vaya a utilizar se cuente con las siguientes especificaciones tanto de hardware como de software.

Software:

En caso de usar sistema operativo Windows®, se recomienda utilizar la versión de Windows 10 en cualquiera de sus versiones, si se llegara a recurrir a una versión anterior, se recomienda checar que los programas corran adecuadamente en esa versión del programa.

Si se está usando sistema operativo MAC OS® se recomienda tener una partición de disco, ya que alguno de estos programas que se corran puede que no funcionen, ya que como se sabe que son diferentes bases, cambia la programación, y esto puede afectar al correcto uso del software. La partición de disco consiste en agregar un sistema operativo diferente al que se maneja, en este caso para poder trabajar bajo la plataforma de Windows.

Programas utilizados:

Processing®. - Para este software, es necesario contar con la versión 3.3 en cualquiera de sus versiones, debido a que se trabajó con estas versiones, al cambiar de esta generación, provocaría un mal funcionamiento de los archivos y que no se encontraran las librerías actualizadas.

Matlab®. - Para el correcto funcionamiento del programa se recomienda usar la versión de Matlab 2016 en adelante, debido a que, si se usa versiones anteriores, los archivos no son compatibles a los recientes.

Arduino®. – Como es un programa de código abierto u open source, se recomienda trabajar con la versión más actual, es con la que se cuenta con la mayoría de las librerías actualizadas, esto para evitar cualquier fallo en la compatibilidad de los programas creados.

Especificaciones de hardware:

Debido a la enorme utilización de recursos que se piensan usar, es recomendable tener como mínimo 8gb de memoria RAM en un sistema operativo de la compañía Windows®, ya que los procesos que se realizan por la comunicación serial y la lectura continua de estos, si se llegara a utilizar una capacidad menor, estos programas podrían saturar a la memoria y provocar una falla en el sistema, provocando a la larga daño en los componentes. Si se usa el sistema operativo MAC OS®, se recomienda el uso de un mínimo de 16gb de memoria RAM, ya que como el mismo sistema operativo de Windows consume recursos, se divide esta memoria en 2 los dos sistemas operativos que se están ejecutando al mismo tiempo, y por lo mismo se requiere un mayor rendimiento y evitar fallas y retardos en el sistema.

Descripción y funcionamiento de la tarjeta Arduino®



Figura 3 Diseño de un Arduino.

El Arduino Uno es una placa de microcontrolador de código abierto basado en el microchip ATmega328P y desarrollado por Arduino.cc.

El Arduino Uno es una placa de microcontrolador basada en el ATmega328. Tiene 14 pines de entrada / salidas digitales (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, una resonador cerámico de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP y un botón de reinicio. Simplemente conéctelo a una computadora con un cable USB o enciéndalo con un adaptador de CA a CC o batería para comenzar.

Especificaciones

- Microcontrolador ATmega328
- Voltaje de Operación 5V
- Voltaje de entrada (recomendado) 7-12 V
- Voltaje de entrada (límites) 6-20V
- Pines de E / S digital 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica 6
- Corriente DC por I / O Pin 40 mA
- Corriente DC para 3.3V Pin 50 mA
- Memoria Flash 32 KB (ATmega328) de los cuales 0.5 KB utilizados por el gestor de arranque
- SRAM 2 KB (ATmega328)
- EEPROM 1 KB (ATmega328)
- Velocidad de reloj 16 MHz

Encendido y alimentación

El Arduino Uno se puede alimentar a través de la conexión USB o con una fuente de alimentación externa. El poder de la fuente se selecciona automáticamente.

La alimentación externa (no USB) puede provenir de un adaptador de CA a CC (pared) o de una batería. El adaptador se puede conectar enchufando un conector de 2.1 mm en el conector de alimentación de la placa del mismo diámetro.

Se puede conectar con una batería con ayuda de cables en los pines de Gnd y Vn del conector POWER. La placa puede funcionar con una fuente externa de 6 a 20 voltios. Sin embargo, si se suministra con menos de 7V, el pin de 5V puede suministrar menos de cinco voltios y la placa puede ser inestable. Si usa más de 12V, el regulador de voltaje puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

VIN. El voltaje de entrada a la placa Arduino cuando utiliza una fuente de alimentación externa (como opuesto a 5 voltios de la conexión USB u otra fuente de

alimentación regulada). Usted puede suministrar voltaje a través de este pin, o, si suministra voltaje a través del conector de alimentación, acceda a él a través de este pin.

5V. Este pin emite 5V regulados desde el regulador en la placa. La placa puede ser suministrada con alimentación desde el conector de alimentación de CC (7 - 12V), el conector USB (5V) o el pin VIN de la placa (7-12V). El suministro de voltaje es a través de los pines de 5V o 3.3V evita el regulador, y puede dañar su placa.

3V3. Un suministro de 3,3 voltios generado por el regulador de a bordo. El consumo de corriente máximo es de 50 mA.

GND. Pines de tierra.

Nota: En este dispositivo la corriente será suministrada a través de la computadora vía USB en donde estaremos realizando las pruebas.

Memoria

El ATmega328 tiene 32 KB (con 0,5 KB utilizados para el gestor de arranque). También tiene 2 KB de SRAM y 1 KB de EEPROM (que se puede leer y escribir con la biblioteca EEPROM).

Comunicación

El Arduino Uno tiene una serie de facilidades para comunicarse con una computadora, otro Arduino u otros microcontroladores. El ATmega328 proporciona comunicación serial UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX). Un ATmega16U2 en la placa canaliza esta serie comunicación a través de USB y aparece como un puerto virtual para el software en la computadora. El '16U2 El firmware utiliza los controladores USB COM estándar y no se necesita de un controlador externo. Sin embargo, en Windows, se requiere un archivo .inf, el que viene incluido en las mismas librerías.

El software Arduino incluye un monitor en serie que permite datos de texto simples para ser enviado hacia y desde la placa Arduino. Los LED RX y TX en la placa

parpadearán cuando se estén enviando datos transmitido a través del chip USB a serie y la conexión USB a la computadora (pero no para la conexión serial comunicación en los pines 0 y 1).

Una biblioteca de SoftwareSerial permite la comunicación en serie en cualquiera de los pines digitales de Uno. El ATmega328 también es compatible con la comunicación I2C (TWI) y SPI. El software Arduino incluye una biblioteca para simplificar el uso del bus I2C. Para la comunicación SPI, usa la biblioteca SPI.

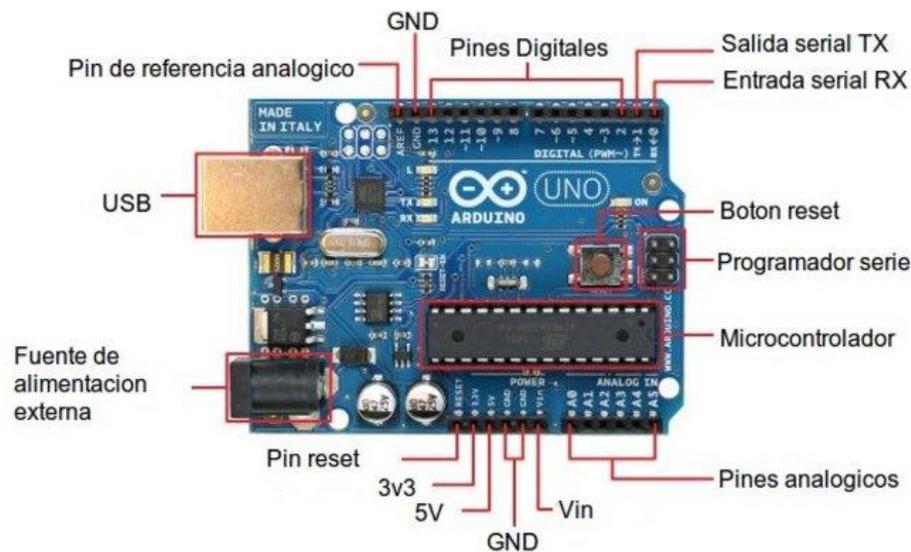


Figura 4 Localización de cada entrada y salida, al igual que de cada componente.

Nota: La comunicación a través del Arduino y la computadora es con el protocolo I2C, porque estamos usando dos placas, al momento de quererlas comunicar en una misma interfaz, es necesario el uso de este protocolo, al respecto la norma menciona, que, gracias a este protocolo, podemos comunicarnos entre varios chips, en este caso las tarjetas y los sensores inerciales.

Programando el Arduino UNO

La programación de la tarjeta de Arduino UNO para la adquisición de los datos de los acelerómetros se compone de dos fases: el cálculo de offset de cada sensor, y la programación de la rutina para adquirir los datos.

a) Cálculo de OFFSET

Cada IMU tiene determinado Offset, que son valores “únicos” para cada dispositivo. Para garantizar que las IMUs lean adecuadamente, lo primero que se debe hacer es obtener su OFFSET.

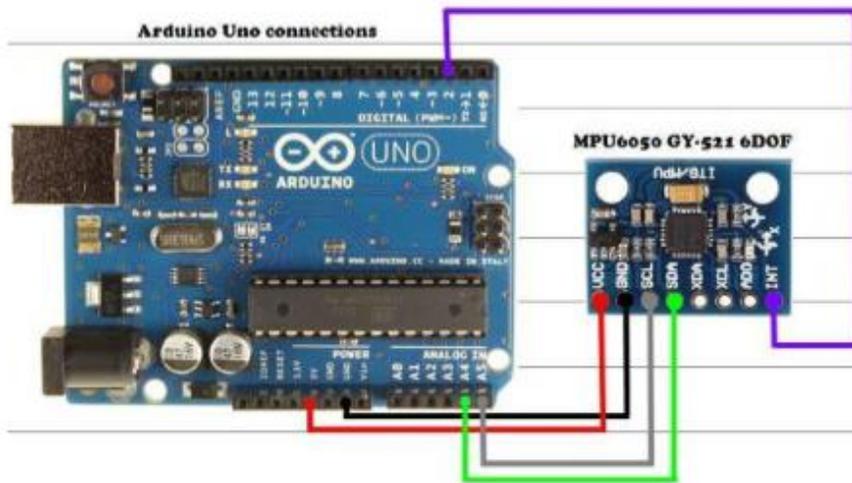


Figura 5 Conexión de IMU MPU6050 con tarjeta Arduino UNO

Una vez conectados, cargar al Arduino UNO el programa de calibración MPU6050_simple_calibration

Colocar la IMU en posición inicial, con las letras hacia arriba, y evitando cualquier movimiento o vibración.

Abrir el monitor serial de Arduino y seguir las instrucciones hasta obtener los OFFSETs de la IMU.

Al finalizar, se obtendrá el siguiente mensaje.

Ejemplo: Your offsets: 3029 -630 1185 21 -31 33

Nota: los valores serán distintos, serán los propios para la IMU en particular de cada mano.

Estos valores corresponden al Acelerómetro en X, Y y Z, y al Giroscopio en X, Y y Z respectivamente.

b) Programando Rutina

Una vez calculados los valores de OFFSET de cada IMU, abrir los siguientes sketches:



Figura 6 Carpetas de los programas.

Es importante decidir en este punto, cuál IMU será colocada en la mano derecha y cuál en la mano izquierda, a fin de tener repetitividad en los resultados. Así mismo, es importante recalcar que, una vez programados los Arduinos con los valores de OFFSET de la IMU, siempre se conecten de la misma forma, cada Arduino con su propia IMU.

A partir de la línea 168 del código, aparecen los siguientes comandos.

```
//Introduce aqui los datos obtenidos  
  
mpu.setXAccelOffset(2891);  
mpu.setYAccelOffset(-559);  
mpu.setZAccelOffset(1258);  
mpu.setXGyroOffset(3);  
mpu.setYGyroOffset(-30);  
mpu.setZGyroOffset(16);
```

Figura 7 Líneas donde se introducen los valores obtenidos de cada MPU6050

Los valores entre paréntesis corresponden a los valores del OFFSET de cada eje para el acelerómetro y el giroscopio, y deben ser cambiados por los valores calculados previamente.

Para su ubicación, es necesario colocar el cursor en cualquier línea del programa, después de esta acción, el número se nos mostrara en la parte inferior izquierda de la ventana del programa de Arduino, como se muestra en la siguiente imagen.

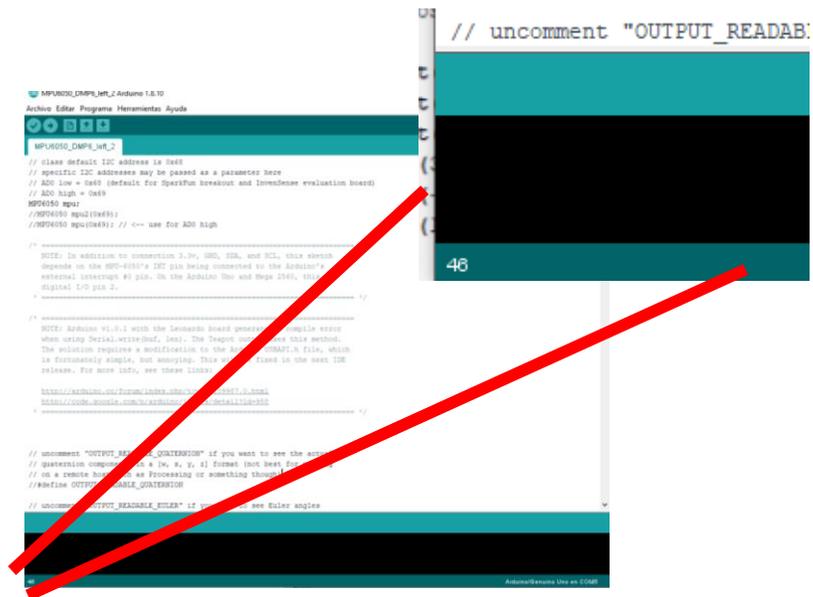


Figura 8 Localización del numero de la línea del programa en la que se encuentre.

Configuración Arduinos®

La interfaz de Processing® y Matlab® están configurada para que al conectar los Arduinos®, éstos sean reconocidos por la computadora con los puertos COM dependiendo de cada Arduino que se haya registrado

Para verificar la dirección de los puertos, es necesario abrir el administrador de dispositivos de Windows, el cual se puede abrir presionando las teclas Windows + X / Administrador de dispositivos o Equipo/ Propiedades / Administrador de dispositivos

Una vez ahí, con los Arduinos® conectados, buscar en Puertos COM.

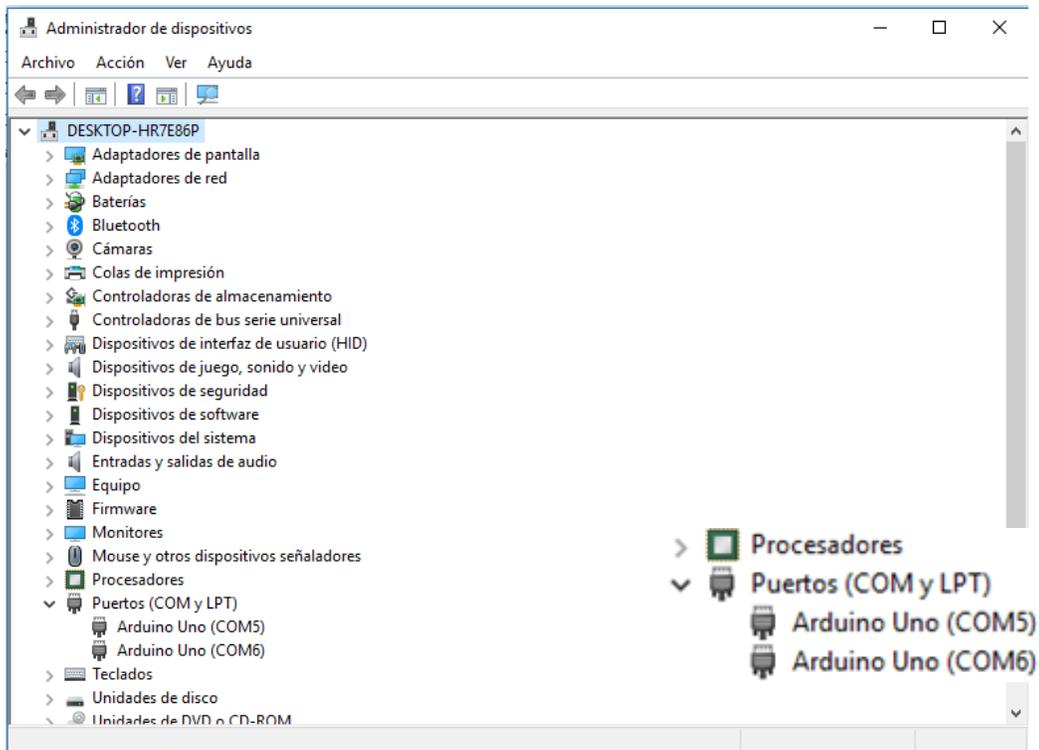


Figura 9 Localización de los puertos COM en dispositivos

Colocar el ratón sobre el Arduino UNO (COMX) y dar clic derecho / PROPIEDADES.

Aparecerá otra pantalla como se muestra en la figura siguiente. Pulsar en la pestaña Configuración del puerto

En la nueva ventana que aparece pulsar opciones avanzadas

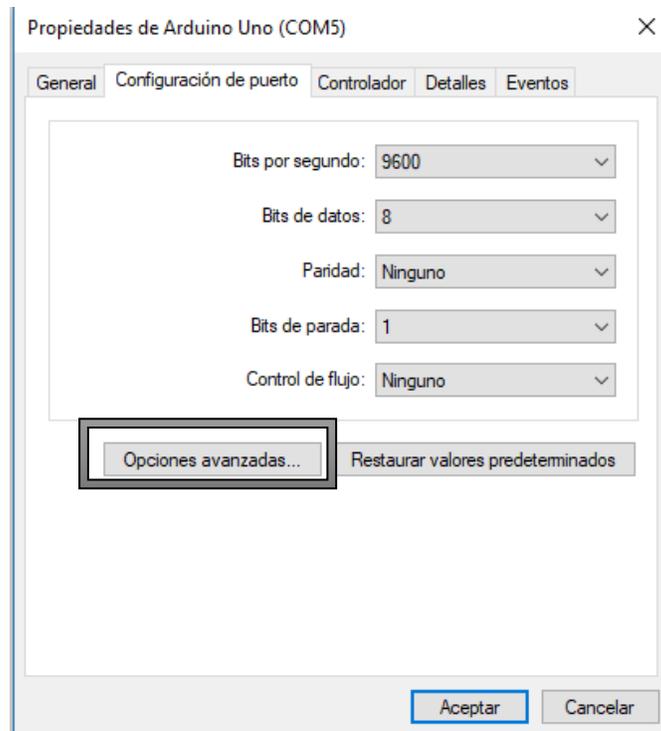


Figura 10 Configuración del puerto

Finalmente aparecerá la ventana mostrada en la Figura 4, esta se abrirá cuando presionemos el botón propiedades avanzadas como se muestra en la figura anterior, en esta se puede cambiar el número del puerto COM. Seleccionar la COM que deseemos dependiendo de la IMU a utilizar de cada mano, para poder establecerlas en los programas que se usaran a continuación.

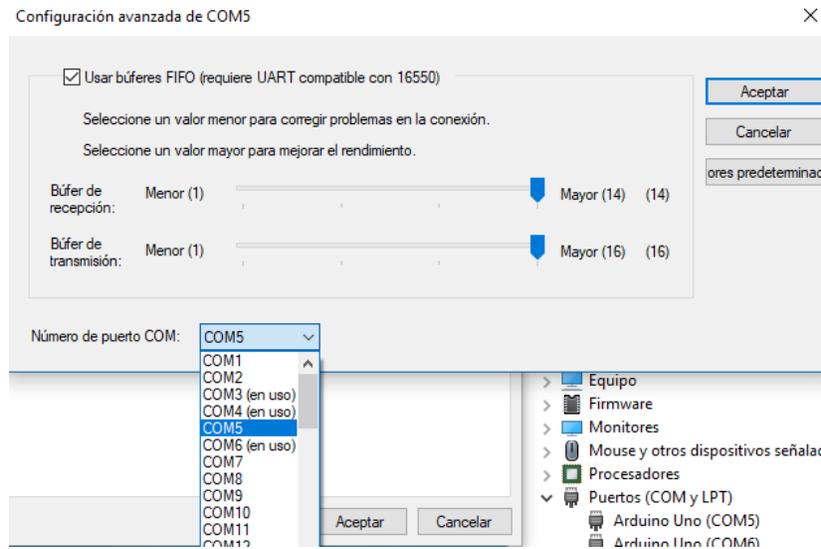


Figura 11 Todas las variables COM# disponibles. Opciones avanzadas.

Nos ayudará mucho definir como queremos nombrar a cada Arduino dependiendo el puerto que le deseemos otorgar, al realizar esta acción en los siguientes registros, será más rápido identificar cual es cada Arduino para cada mano, para que sea más fácil las correcciones de los programas o errores que nos vaya generando el sistema. Sería más fácil identificarlos, si se le pudiera personalizar el nombre a cada tarjeta, pero debido a como manejan su programación de los chips (ATmega328), estos vienen protegidos, lo que hace casi imposible mejorar la personalización visual o de la interfaz para una amigable entre el usuario y este tipo de dispositivos.

Como funciona el programa Arduino

Para saber cómo funciona el programa en Arduino es necesario saber para qué sirve cada programa, en este caso tenemos dos programas que son:

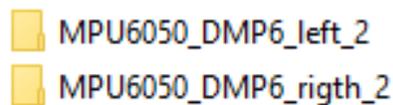


Figura 12 Carpetas de programas.

Estos programas contienen las características de cada IMU que hayamos seleccionado para cada mano, por eso se le clasifican como “left” y “right” que en español significan izquierda y derecha respectivamente. Dentro de estos

programas, podemos obtener, dependiendo las necesidades que se tengan en los análisis de las actividades realizadas, una variedad de diferentes datos.

Los dos programas son bastante parecidos, debido a que realizan la misma función, lo único que cambia son los datos que se ingresan los cuales se mencionan más adelante, debido a esto solo es necesario la explicación del código que se anexara en la parte final del trabajo, esto para que cualquier duda que se tenga y cómo funciona el programa, se pueda checar y realizar los ajustes deseados.

El programa es una plantilla que la misma librería del sensor en Arduino nos brinda, el uso es sencillo. Para poder usarlo de manera adecuada, debemos quitar las partes comentadas de las cuales pensemos que serán necesarias para el proyecto, en este caso, las funciones que serán usadas son las que no estarán comentadas, en este caso las que están comentadas se muestran con dos diagonales “\\”, se pueden distinguir de manera rápida y fácil.

En las figuras posteriores se mostrarán las funciones que no están siendo usadas en los renglones comentados y en los renglones no comentados, se muestran las funciones que estamos usando para obtener los datos que vamos a utilizar.

```

#ifdef OUTPUT_READABLE_REALACCEL
    // display real acceleration, adjusted to remove gravity
    mpu.dmpGetQuaternion(sq, fifoBuffer);
    mpu.dmpGetAccel(saa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, sq);
    mpu.dmpGetLinearAccel(saaReal, saa, &gravity);
    //Serial.print("areal\t");
    Serial.print(aaReal.x * (9.81/16384.0));
    Serial.print(",");
    Serial.print(aaReal.y * (9.81/16384.0));
    Serial.print(",");
    Serial.print(aaReal.z * (9.81/16384.0));
    Serial.print(",");
#endif

#ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(sq, fifoBuffer);
    mpu.dmpGetGravity(&gravity, sq);
    mpu.dmpGetYawPitchRoll(ypr, sq, &gravity);
    //Serial.print("ypr\t");
    Serial.print(ypr[0] * 180/M_PI);
    Serial.print(",");
    Serial.print(ypr[1] * 180/M_PI);
    Serial.print(",");
    Serial.print(ypr[2] * 180/M_PI);
    Serial.print(",");
#endif

```

Figura 13 Ejemplo de líneas no comentadas las cuales se usan para la correcta realización del programa.

```

//      #ifndef OUTPUT_READABLE_QUATERNION
//          // display quaternion values in easy matrix form: w x y z
//          mpu.dmpGetQuaternion(&q, fifoBuffer);
//          Serial.print("quat\t");
//          Serial.print(q.w);
//          Serial.print("\t");
//          Serial.print(q.x);
//          Serial.print("\t");
//          Serial.print(q.y);
//          Serial.print("\t");
//          Serial.println(q.z);
//      #endif
//
//      #ifndef OUTPUT_READABLE_EULER
//          // display Euler angles in degrees
//          mpu.dmpGetQuaternion(&q, fifoBuffer);
//          mpu.dmpGetEuler(euler, &q);
//          //Serial.print("euler\t");
//          Serial.print(euler[0] * 180/M_PI);
//          Serial.print("\t");
//          Serial.print(euler[1] * 180/M_PI);
//          Serial.print("\t");
//          Serial.print(euler[2] * 180/M_PI);
//          Serial.print("\t");
//      #endif

```

Figura 14 Ejemplo de líneas comentadas, donde las funciones no están siendo usada.

Este programa lo que nos da, son las diferentes funciones que deseamos tener de acuerdo a la necesidad del proyecto, en este caso como queríamos saber cuáles eran las aceleraciones, se necesitan las funciones que lean las velocidades en X, Y y Z, pero como también tenemos velocidades angulares, se usan en este caso dos funciones la de OUTPUT_READABLE_REALACCEL y la de OUTPUT_READABLE_YAWPITCHROLL, ya que cada uno nos muestra las aceleraciones unas que son directas en X, Y y Z, y las de yaw, pitch y roll, que son las aceleraciones angulares.

Función OUTPUT_READABLE_REALACCEL, esta función lo que nos da son los valores de las aceleraciones en X, Y y Z, estos valores nos los da sin ser convertidos, a que se refiere la conversión, cada IMU tiene

Por eso en este caso, se necesitaba tener estas dos funciones, ya que como establecimos las variables de acuerdo con nuestras necesidades. En general este

programa es muy sencillo de utilizar, solo se necesitan agregar los valores obtenidos en los sensores y de ahí quitar los comentarios de las funciones deseada.

Nota: Este programa solo funciona para los sensores inerciales de modelo MPU6050, debido a que las funciones que se usan no son tan demandantes y solo es un primer acercamiento, se usa un sensor económico, ya que la idea es que este dispositivo sea accesible.

Pasos para seguir para poder usar los programas

Subir cada programa en su respectivo Arduino (Mpu left and right, asignados con su respectiva COM).

Estos programas fueron los que anteriormente se usaron para programar las rutinas, los cuales se deben de incluir en la carpeta de los archivos brindados

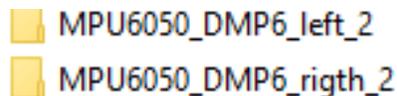


Figura 15 Carpetas de programas.

Después de haber subido los programas a cada Arduino asignado, ya solo será necesario abrir los programas y seguir las instrucciones que se darán a continuación.

Usando Processing®

Pasos por seguir para poder usar Processing®, Arduino® y las MPU:

Este programa esta adecuado a como deseemos observarlo en el ordenador, se pueden editar las propiedades básicas del programa, colores, objetos e interfaz.

En los anexos se muestra el programa de Processing® que se usa para la adquisición de datos, cada función que realiza el programa está dividida con una línea, para poder identificarlo de manera rápida y fácil cualquier fallo que se tenga o alguna nueva modificación que se le desee hacer.

Librerías

Al momento de usar el programa de Processing®, ver que se cuenta con las siguientes librerías:

- a) AP-Sync
- b) Arduino
- c) ControlP5
- d) G4p

para descargar cada una de estas librerías, se tiene que ir a la barra de tareas del programa Processing en la pestaña que dice sketch

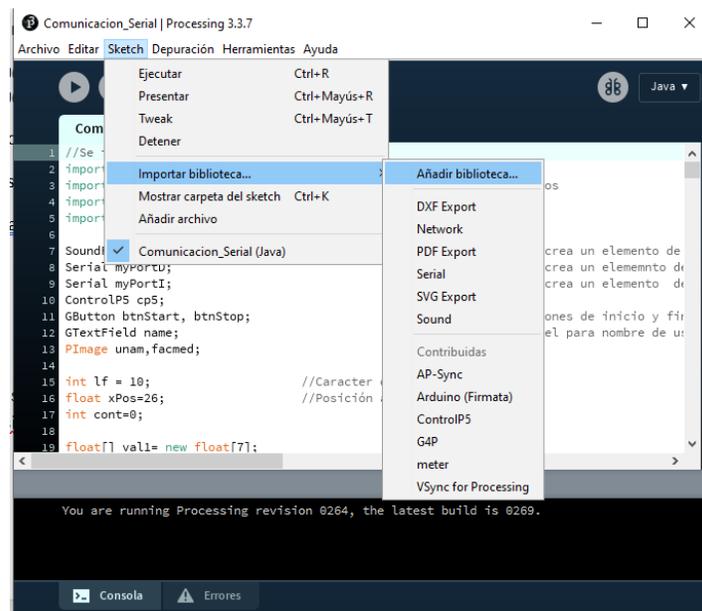


Figura 16 Localización de las librerías en Processing.

Haciéndole clic en la pestaña añadir biblioteca, saldrá la siguiente ventana:

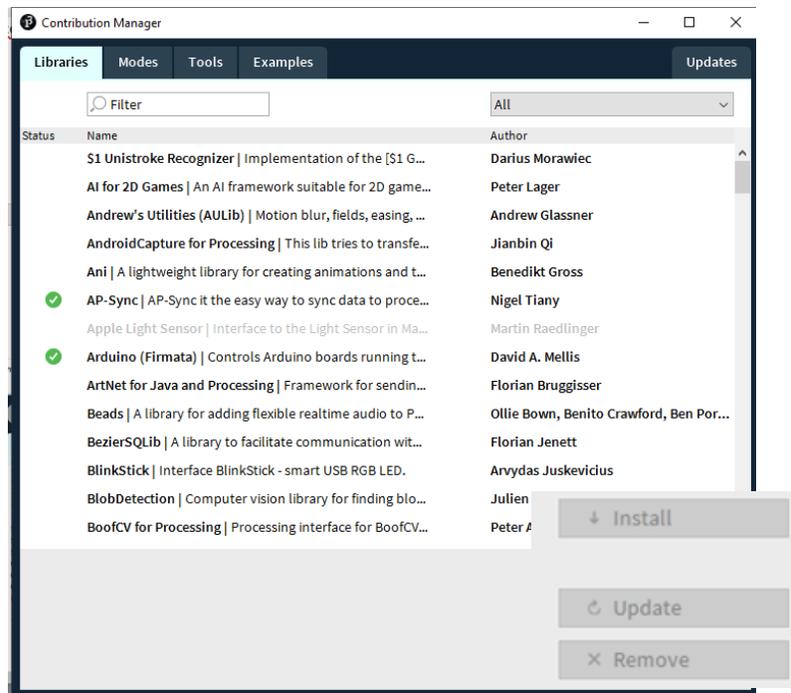


Figura 17 Interfaz de la biblioteca de las librerías y acciones que se pueden realizar.

A continuación, tendremos que escribir en el buscador las librerías antes mencionadas, después de seleccionar cada librería se tiene que instalar, solo hay que darle en la opción Install y se agregara automáticamente en el programa.

Estas librerías (**AP-Sync**, **Arduino**, **ControlP5**, **G4p**) son esenciales para que exista comunicación con los Arduino y el Processing, de esta manera la interfaz gráfica podrá desenvolverse de una manera correcta.

Nota: checar siempre las librerías, ya que, si se actualiza el Processing, puede que estas sean incompatibles con la versión descargada, en dado caso, se recomendaría regresar a la versión en la cual se esté trabajando o de la misma manera estas pueden ser borradas del mismo programa, debido a que, con cada actualización, este programa crea carpetas con direcciones distintas.

Ajustando el programa Processing®

En la parte de configuración de la comunicación serial que se encuentra a partir de la línea 62, checar en que puerto se encuentra cada Arduino para poder configurar dentro del programa de Processing los COM correspondientes de cada IMU.

```

62 ///////////////////////////////////////////////////////////////////INICIACIÓN DE LA COMUNICACIÓN SERIAL/////////////////////////////////////////////////////////////////
63   printArray(Serial.list());
64   myPortD=new Serial(this,"COM5",115200);
65   myPortD.clear();
66   myPortD.bufferUntil(lf);
67   myPortI=new Serial(this,"COM6",115200);
68   myPortI.clear();
69   myPortI.bufferUntil(lf);

```

Figura 18 Líneas del código fuente en donde se deben de indicar los puertos COM utilizados en cada computadora.

Correr el programa, si hay algún error este entrara en conflicto y este no podrá inicializar

Interfaz Gráfica del Usuario Processing®

La interfaz está diseñada para capturar el nombre del participante, o el texto que se desee. Ver figura siguiente.

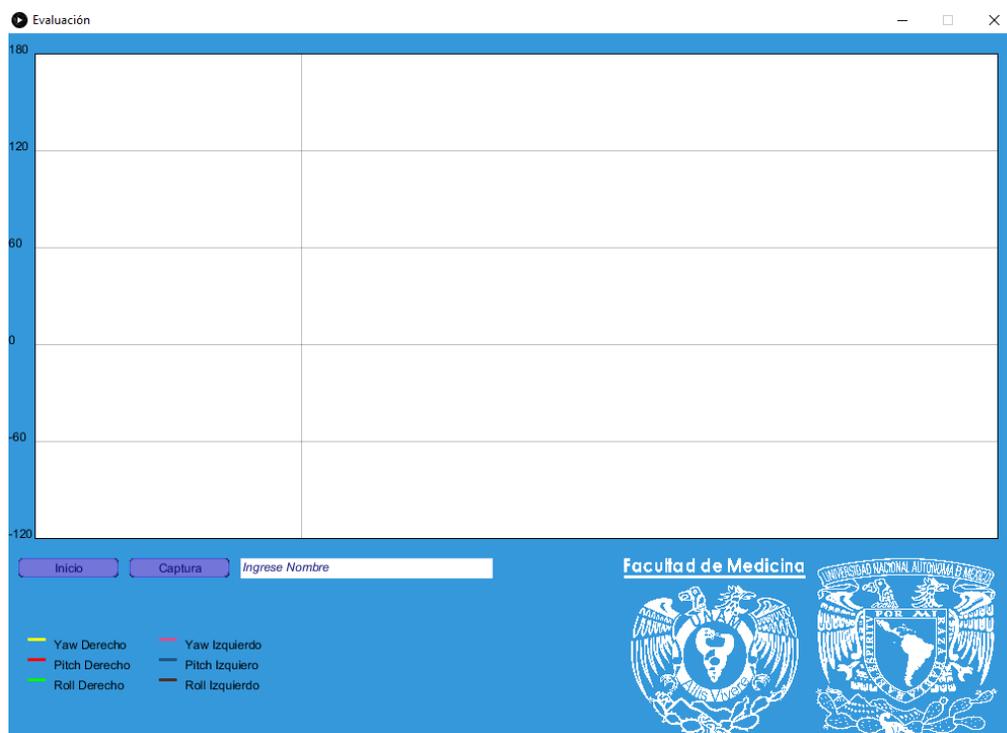


Figura 19 Interfaz gráfica en Processing. Ventana para registro.

Es importante mencionar que al iniciar la capturar de datos, nunca se deje en blanco el recuadro donde se ingresa el nombre, pues los datos no se podrán grabar, o no

se podrán identificar rápidamente, además de que este no iniciara correctamente o creará conflictos provocando un error en el sistema.

Cuando esté todo listo, dar clic en el botón Inicio



Figura 20 Interfaz de botones (Botón Inicio).

Para poder tener la prueba de manera correcta, el participante debe colocar sus manos en posición inicial y sin moverlas durante 10 segundos, para que las IMUs se calibren, estos 10 segundos se identificaran de una manera sencilla tan solo observando la línea que se muestra a continuación.

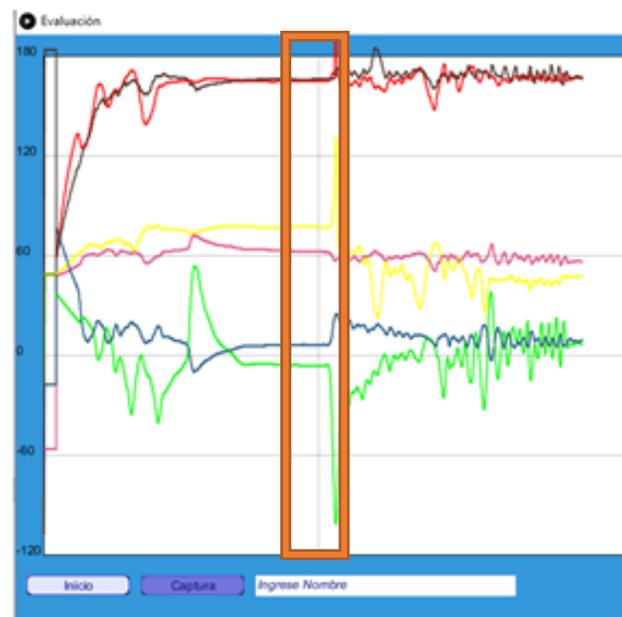


Figura 21 Calibración

Si todo está bien conectado, aparecerá una ventana con las gráficas del giroscopio en tiempo real (Ver figura 22).

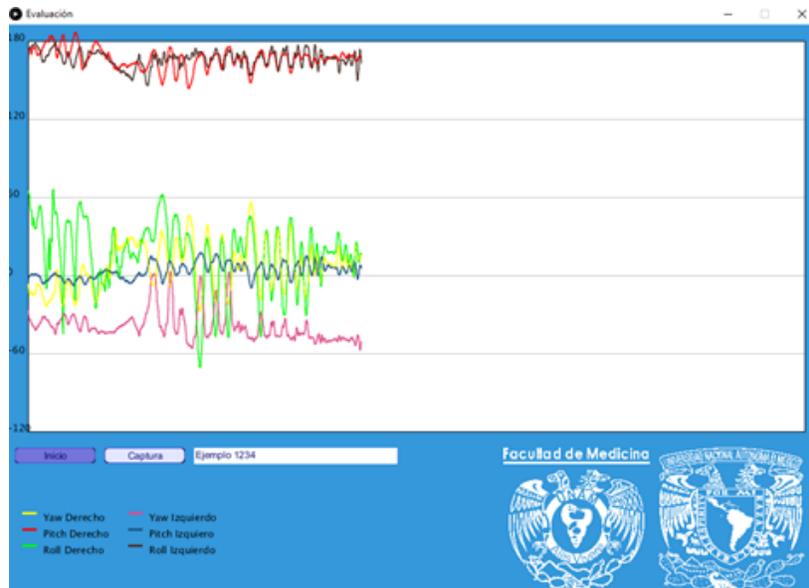


Figura 22 Gráfica en tiempo real del giroscopio de ambas IMUs.

Si dicha ventana no inicia, verificar en la sección de errores cual podría ser un posible problema. La mayoría de las veces es porque la conexión de las IMUs no es la correcta, o porque Processing no reconoce los puertos. Si este es el caso, verificar que los puertos COM sean los correctos, o volver a conectar los Arduinos®.

Al finalizar la prueba, la captura de datos se detiene automáticamente al presionar el botón Captura.



Figura 23 Interfaz de botones (Botón Captura).

El programa terminara y se regresara a la ventana del programa en PROCESSING®, los cuales se brindarán en un archivo con terminación .txt, el cual puede ser leído con el programa de bloc de notas. Lo restante es copiar los datos para poder analizarlos, este análisis se mostrará detalladamente en un apartado más adelante.

Sugerencias

Si se desea trabajar en diferentes computadoras, se necesitará realizar estos pasos nuevamente, por lo que siempre será necesario checar que se tengan instaladas

las librerías antes de correr el programa, de no ser así, podrán crearse conflictos con la computadora y se generaran errores, lo que provocaría que la computadora se llegara a detendrá la computadora y esta no reaccionara en un periodo de tiempo.

Otro punto importante es, que siempre contemos con los archivos esenciales para el correcto funcionamiento del programa, estos archivos son los que se han mencionado con anterioridad.

Errores frecuentes en PROCESSING®

Si aparece la siguiente pantalla, es debido a que el programa no reconoce los Arduinos®, o estos no se han configurado correctamente.

Una manera de solucionarlo es que se cierre automáticamente el programa con las indicaciones que te van apareciendo, en dado caso que no se pueda cerrar, la forma más efectiva será la que se mencionara a continuación.

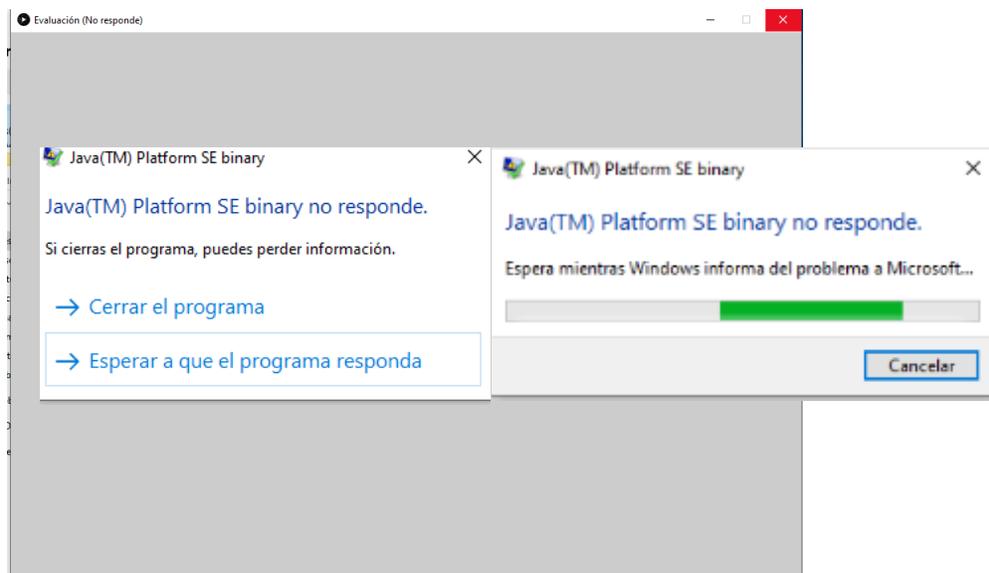


Figura 24 Estado del programa cuando deja de funcionar.

Tendremos que abrir el administrador de tareas, y para esto es necesario presionar las teclas, Ctrl+Alt+Supr, después de presionar estas teclas, nos aparecerá un menú, el cual nos aparecerá la opción de administrador de tareas, este podrá aparecer de las siguientes formas:

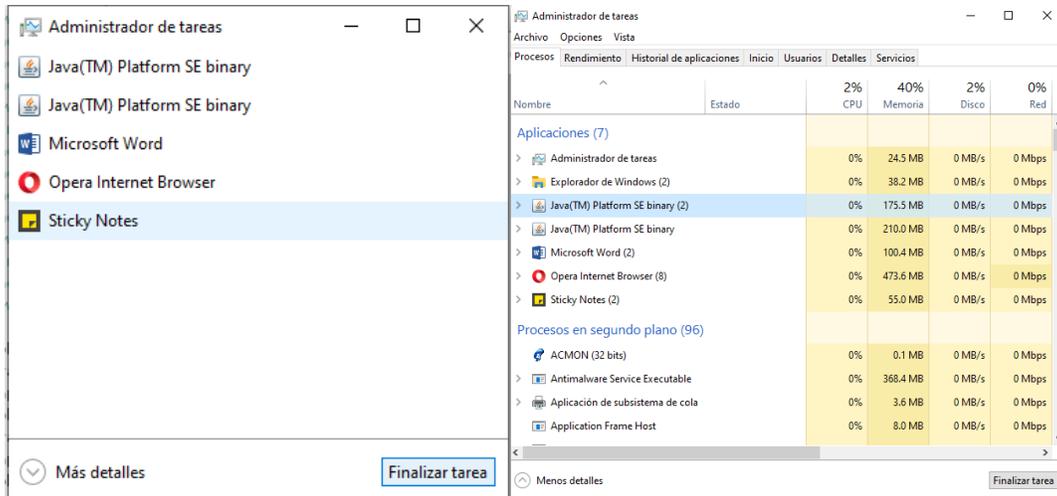


Figura 25 Administrador de tareas para poder cerrar los programas.

Y lo único que se tendrá que realizar, es seleccionar el programa Java(TM) Platform SE binary (2) y darle en finalizar tarea en la parte inferior derecha del administrador, se debe cerrar este programa, ya que es el que crea la interfaz, ya que si se cierra el java que no tiene la terminación 2, se cerraran todas las pestañas, incluyendo el programa donde se está utilizando el Processing.

Otro caso particular es el que jamás se inicialice la captura de datos, y esto se puede deber a diferentes factores.

Uno de estos es que, si notamos que solo se mueven 3 variables en vez de 6, es debido a que una de las IMUs está mal conectada, o se está creando algún falso, lo que provoca que se muestre una línea continua y solo 3 variables se muevan, esto se soluciona únicamente checando que la conexión de Arduino y las IMUs estén correctamente conectadas, o en dado caso si se detectara un falso, arreglarlo rápidamente para poder realizar las pruebas.

Que jamás se inicialice o se detenga el registro de las variables, esto es debido a que como no se limita el tiempo, se genera un registro demasiado grande, la manera de solucionar esto es reiniciando los Arduinos® desde su botón reset, para limpiar la memoria cache y evite conflictos entre programas.

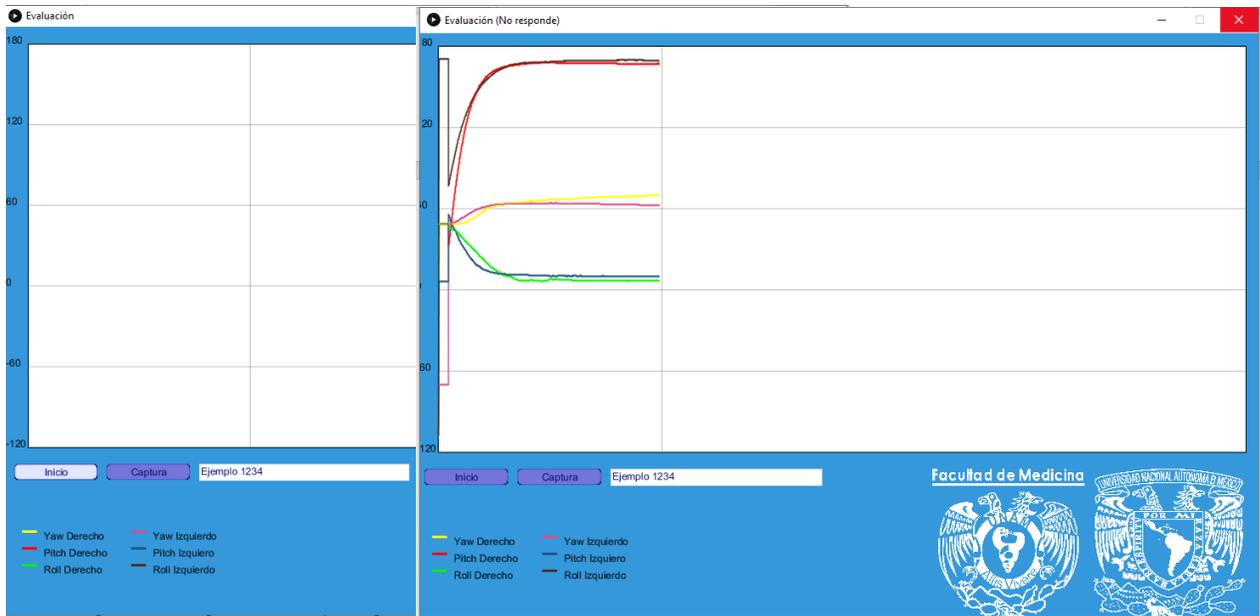


Figura 26 Interfaz cuando no registra los datos y entra en conflicto.

Otro error muy común es el que se muestra a continuación, en el que se ve la interfaz sin nada. Este error se genera debido a que la carpeta donde se tienen los archivos visuales que se muestran la interfaz, no se contengan dentro de la misma carpeta de donde se tenga el programa de Processing utilizado, cuyo nombre es Comunicación Serial.

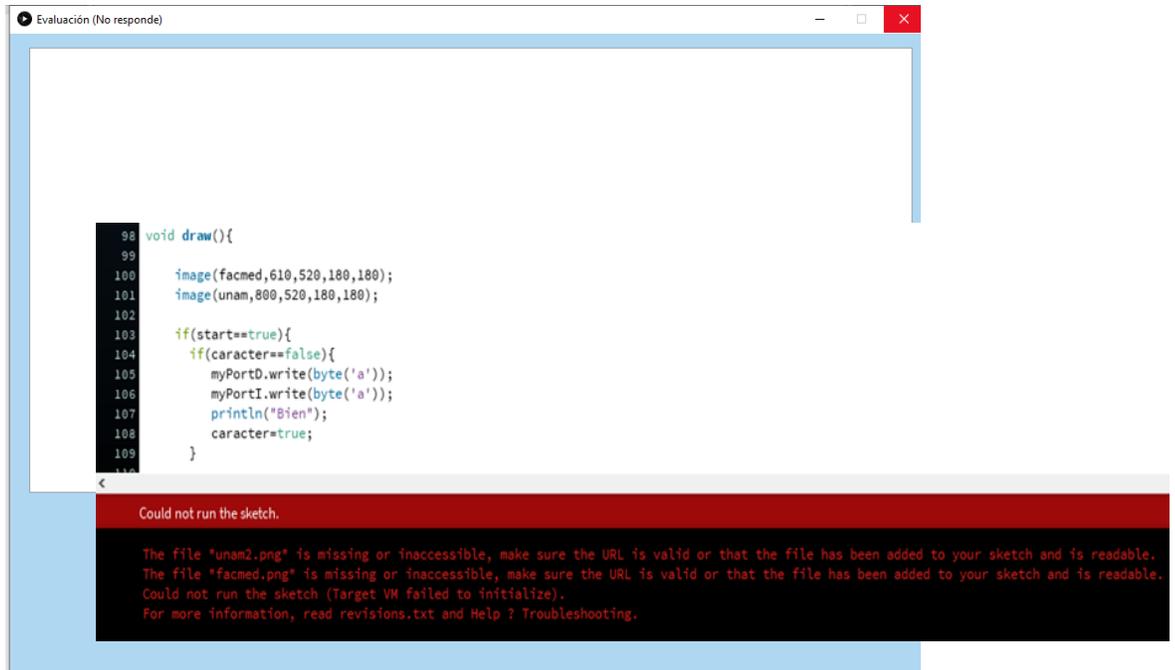


Figura 27 Error al no tener bien conectado los Arduinos, o alguna librería no está incluida en el programa, o alguno de los archivos necesarios no están incluidos en el sketch.

La forma de solucionarlo es teniendo la carpeta (data) en la misma carpeta donde se esté manejando el programa como se muestra en la siguiente imagen, esta carpeta contiene los archivos de imágenes necesarios para el correcto funcionamiento de la interfaz.

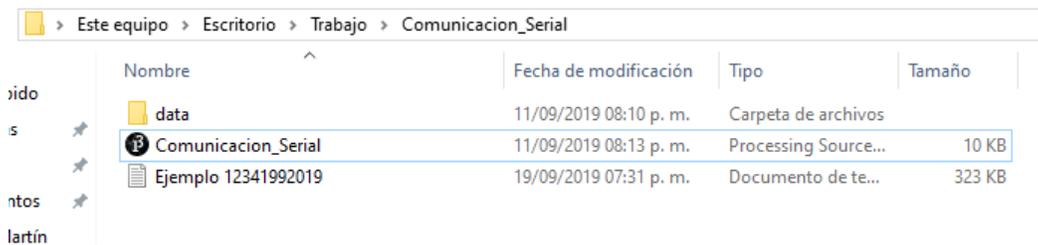


Figura 28 Archivos necesarios para la función correcta del programa.

En dado caso que no se llegara a contar con la carpeta, se puede solucionar comentando las siguientes líneas 216 y 217 del programa con los caracteres '//', lo que hacen estas barras en diagonal es poner el texto en gris, haciendo así que el programa no llame a estas rutinas, lo que ayudará a que no se tengan conflictos, de

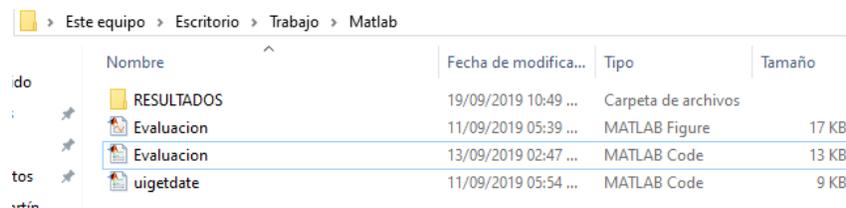
igual manera el programa se podrá usar sin los archivos de la interfaz, estos archivos se piden dentro del programa, a esta función se le conoce como mandar a llamar al objeto en este caso las imágenes.

```
213 ///////////////////////////////////////////////////RUTINA PARA ACTUALIZAR LA GRÁFICA//////////////////////////////////////
214
215 void window(){
216     unam=loadImage("unam2.png");
217     facmed=loadImage("facmed.png");
218
```

Figura 29 Las líneas más comunes que pueden presentar algún conflicto en el programa.

Usando MATLAB®

Tener los archivos correspondientes que se muestran a continuación, los cuales son los siguientes, siempre deberán estar en la misma carpeta en la que se corra el programa.



Nombre	Fecha de modifica...	Tipo	Tamaño
RESULTADOS	19/09/2019 10:49 ...	Carpeta de archivos	
Evaluacion	11/09/2019 05:39 ...	MATLAB Figure	17 KB
Evaluacion	13/09/2019 02:47 ...	MATLAB Code	13 KB
uigetdate	11/09/2019 05:54 ...	MATLAB Code	9 KB

Figura 30 Archivos necesarios para el uso de Matlab.

Los archivos esenciales son los dos archivos de Evaluación y el de uigetdate, para saber qué hace cada uno de estos archivos a continuación se detallará su función de cada uno. El nombre entre paréntesis es el tipo de archivo que se usa, puede referirse como a su extensión en otros archivos comúnmente utilizados:

- Evaluación (MATLAB Figure), este archivo es el que se encarga de mostrar la interfaz gráfica, esta se programa según las necesidades deseadas, en este caso se usa para poder ingresar los datos de la persona.
- Evaluación (MATLAB Code), este archivo es el encargado de hacer todo el procesamiento de la información, aquí mismo se programan los comandos del archivo de la interfaz, ya que estos mismos se muestran dentro del código.

- Uigetdate (MATLAB Code), este es un archivo que se obtiene a través de la página, funciona como una librería para obtener los datos de la fecha actualizada.

En el programa Evaluacion.m editar los siguientes datos referentes a las COM de cada IMU para que estos coincidan y así Matlab pueda funcionar correctamente

```

232
233 %borrar datos de comunicación serial previos
234 delete(instrfind({'Port'},{'COM5'}));
235 delete(instrfind({'Port'},{'COM6'}));
236
237 % delete(instrfind({'Port'},{'COM8'}));
238 % delete(instrfind({'Port'},{'COM7'}));
239
240 %crear objetos serie
241 a = serial('COM5', 'BaudRate', 115200, 'Terminator', 'CR/LF');
242 b = serial('COM6', 'BaudRate', 115200, 'Terminator', 'CR/LF');

```

Figura 31 Líneas para identificar que modificar (COMX).

Interfaz Gráfica de Usuario (GUIDE) MATLAB®

La interfaz está diseñada para capturar los datos del participante, la fecha y hora de la prueba, y para la adquisición y exportación de los datos de la IMU a los formatos de Excel (extensión xls), Matlab (extensión

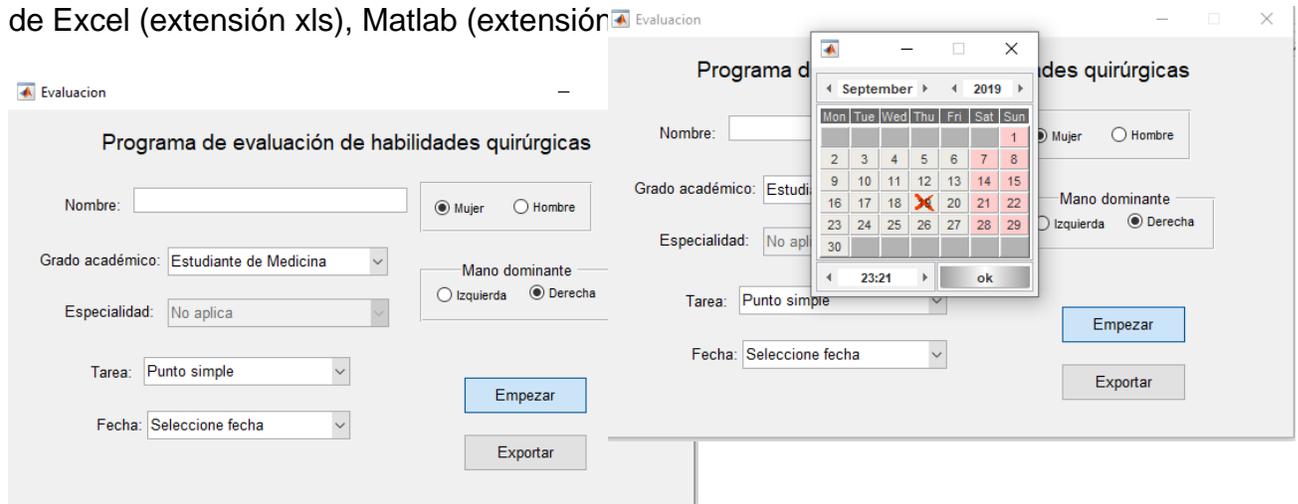


Figura 32 Interfaz de la aplicación en MATLAB®. Ventana para registro de participantes.

Es importante mencionar que, al capturar el nombre, nunca se deje un espacio en blanco al final, pues los datos no podrán ser exportados. Para comenzar con la prueba, el participante debe colocar sus manos en posición inicial y sin moverlas

durante 10 segundos, para que las IMUs se calibren. Cuando esté todo listo, dar clic en el botón Empezar. Si todo está bien conectado, aparecerá una ventana con las gráficas del giroscopio en tiempo real.

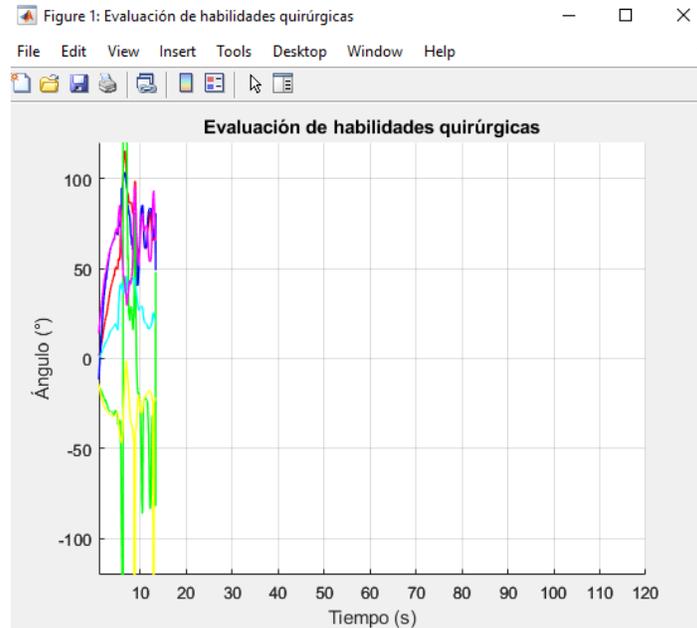


Figura 33 Gráfica en tiempo real del giroscopio de ambas IMUs.

Si dicha ventana no aparece, verificar en la ventana de comandos de Matlab (Command Window) el error. La mayoría de las veces es porque la conexión de las IMUs no es la correcta, o porque MATLAB® no reconoce los puertos. Si este es el caso, verificar que los puertos COM sean los correctos, o volver a conectar los Arduinos®.

Al finalizar la prueba, la captura de datos se detiene automáticamente al cerrar la ventana. Y volvemos a la ventana inicial.

Lo restante es obtener los datos dando en las diferentes extensiones haciendo clic en el botón exportar. Se creará una carpeta en resultados y ahí se agregarán los archivos.

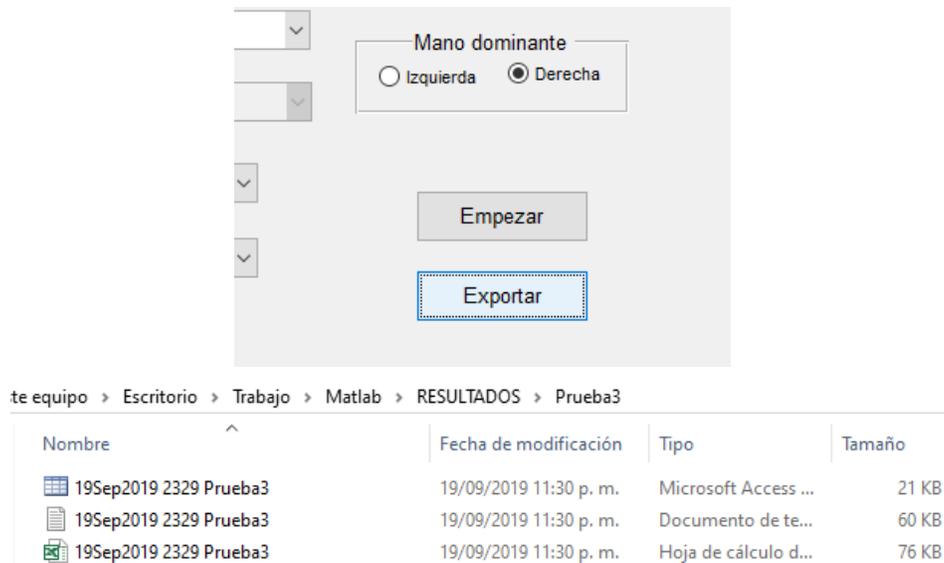


Figura 34 Exportación de datos y su formato de entrega.

Antes de iniciar la siguiente prueba, se deben borrar los datos y las variables almacenadas en la memoria de MATLAB®, si no se borran, los datos del nuevo participante se sobrescribirán en los del participante anterior.

Para ello, es necesario limpiar todas las variables, funciones e interrupciones que se hayan utilizado. También es conveniente limpiar la ventana de comandos, para que se puedan visualizar de mejor manera los mensajes de error, si es que llegara a existir un problema.

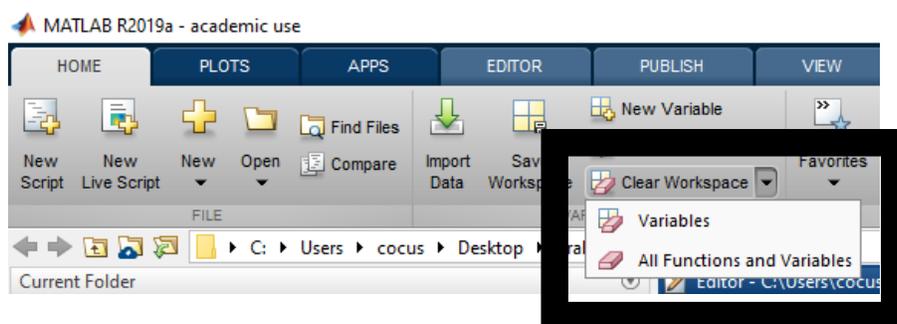
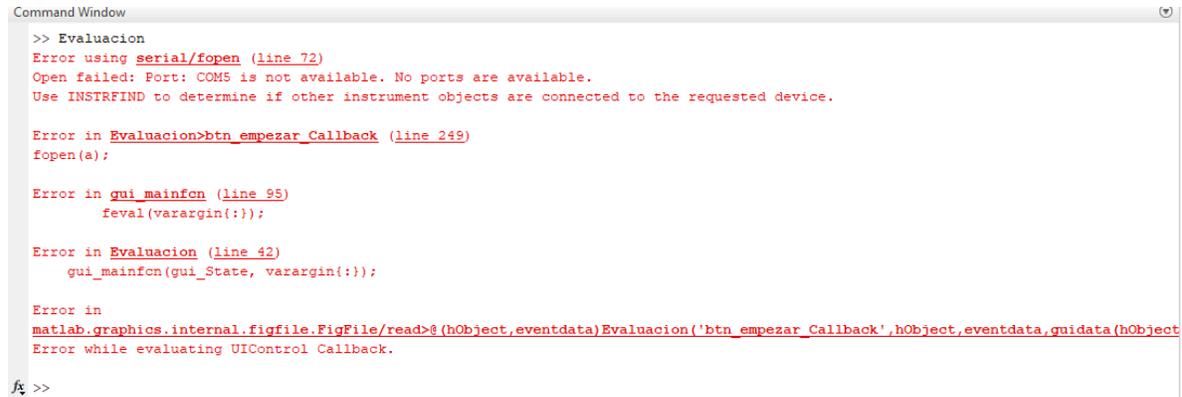


Figura 35 Limpiar variables, funciones e interrupciones.

Errores Frecuentes en MATLAB®

Errores en los puertos seriales, esto surge cuando no se configuraron bien los COM en el programa o estos no están siendo detectados por la computadora.



```
Command Window
>> Evaluacion
Error using serial/fopen (line 72)
Open failed: Port: COM5 is not available. No ports are available.
Use INSTRFIND to determine if other instrument objects are connected to the requested device.

Error in Evaluacion>btn_empezar_Callback (line 249)
fopen(a);

Error in gui_mainfcn (line 95)
feval(varargin{:});

Error in Evaluacion (line 42)
gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)Evaluacion('btn_empezar_Callback',hObject,eventdata,guidata(hObject)
Error while evaluating UIControl Callback.

fx >>
```

Figura 36 Error en los puertos COM.

Para solucionar este problema, solo hay que checar que en la programación si se configuraron bien los puertos COM, también checar que estén bien conectados y la computadora estén siendo reconocidos.

Errores por la figura de la interfaz (cambio de máquina, uso del mismo programa en otra computadora)



```
Command Window
Warning: Matching failure in format. The format specified for conversion, '%f,%f,%f,%f,%f', is incorrect.
Warning: Matching failure in format. The format specified for conversion, '%f,%f,%f,%f,%f', is incorrect.
Index exceeds the number of array elements (2).

Error in Evaluacion>btn_empezar_Callback (line 292)
c3(1)=s(3);

Error in gui_mainfcn (line 95)
feval(varargin{:});

Error in Evaluacion (line 42)
gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)Evaluacion('btn_empezar_Callback',hObject,eventdata,guidata(hObject)
Error while evaluating UIControl Callback.

fx >>
```

Figura 37 Error mostrado al usar el mismo programa en diferentes computadoras y no configurarlas en la que se usa.

Para solucionarlos, se guarda de nuevo en la interfaz gráfica en la computadora que se esté usando, esto para que la dirección quede guardada en la computadora y MATLAB® pueda leerla.

Se detiene automáticamente o no inicializa la ventana de las gráficas.

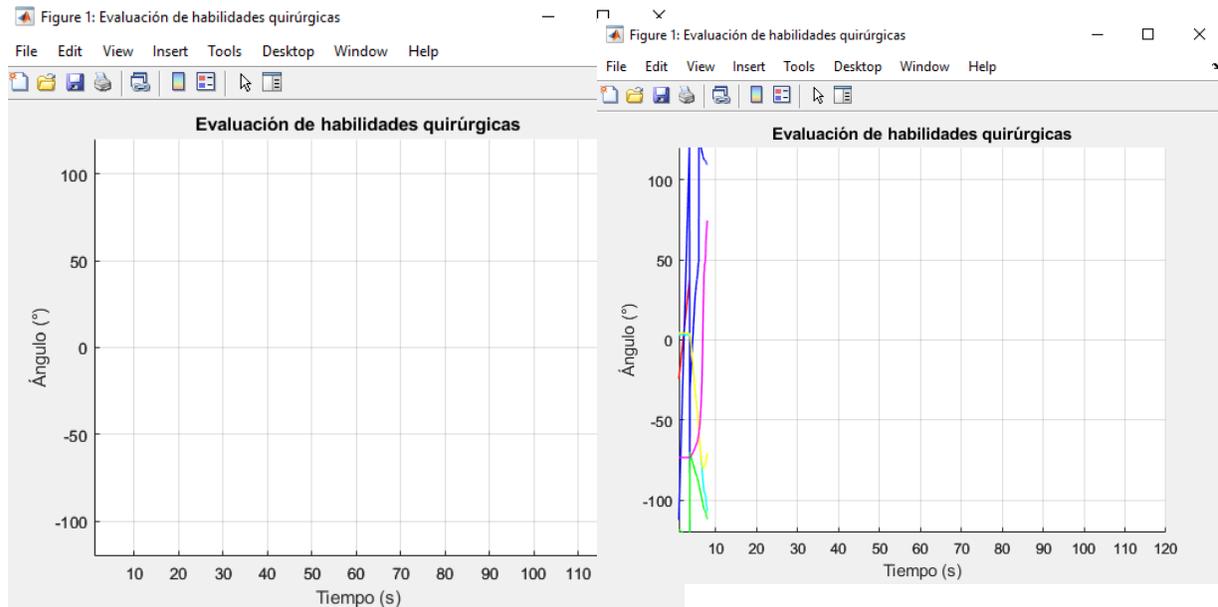


Figura 38 Inmovilización del sistema por la mala configuración.

Estos problemas suceden cuando se crea un almacenamiento de datos muy grande, provocando que las cadenas no puedan ser leídas por el MATLAB® antes de empezar el registro, otra razón es por no estar en el punto inicial al empezar la calibración para la prueba. Una manera de solucionarlo es limpiando todas las variables desde el menú de la consola de MATLAB®.

La razón más crítica que sucede es cuando se llena la memoria, la razón es porque al ser en tiempo real, Matlab entra en conflicto y al no tener un punto final, crea error en el programa, lo que causa un error en la computadora, y se tiene que reiniciar MATLAB®.

Manejo de los datos para la evaluación.

Uso de la plantilla

Nombre	Fecha	Género	Mano domin	Grado académico	Especialidad
Rango de valores de desempeño					Tarea: Ejemplo.
Niveles de desempeño					Sutura de herida superficial con punto simple
		Tiempo (S)	Distancia (cm)		
Path Length Derecha	Experto	20-30	150-300		
0	Intermedio	31-45	301-600		
	Principiante	46 - procedimiento no concluid 601-Procedimiento no concluido			
Path Length izqu	Entrega de resultados del participante				
0	Excelente	poco tiempo y menor distancia			
Path Length Total	Bueno	regular tiempo y regular distancia			
0	Suficiente	tiempo alto y mayor o menor distancia			
	Insuficiente				
Tiempo	Evaluacion de videos por el experto con el uso de rubrica				
0	Excelente				
	Bueno				
	Suficiente				
	Insuficiente				
Mano derecha					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración máxima en X	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	#NUM!	#NUM!
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Acleración en X	Desv Estandar Acleración en Y	Desv Estandar Acleración en Z	
#NUM!	#NUM!	#DIV/0!	#DIV/0!	#DIV/0!	
Mano izquierda					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración máxima en X	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	#NUM!	#NUM!
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Acleración en X	Desv Estandar Acleración en Y	Desv Estandar Acleración en Z	
#NUM!	#NUM!	#DIV/0!	#DIV/0!	#DIV/0!	

Figura 39 Interfaz de la plantilla y los datos que brinda.

Las características que tiene esta plantilla de reporte de resultados para el usuario son las siguientes variables: en color oro se indica la tarea quirúrgica a realizar, en color azul cielo se muestra los datos sociodemográficos del participante, en color amarillo se reporta las variables path length (Distancia recorrida) por mano y total y el tiempo en el que realizo el procedimiento y en color gris se muestra el resultado de las aceleraciones de la mano derecha y la mano izquierda, y en verde la evaluación obtenida de acuerdo a los diferentes niveles evaluados.

Para su uso correcto solo se tiene que hacer los siguiente:

- 1.- Abrir el archivo .txt que se generó a través de la interfaz gráfica de evaluación. Seleccionar todos los datos del archivo y copiarlos, para no hacerlo manualmente, solo es necesario presionar las teclas Ctrl+E.

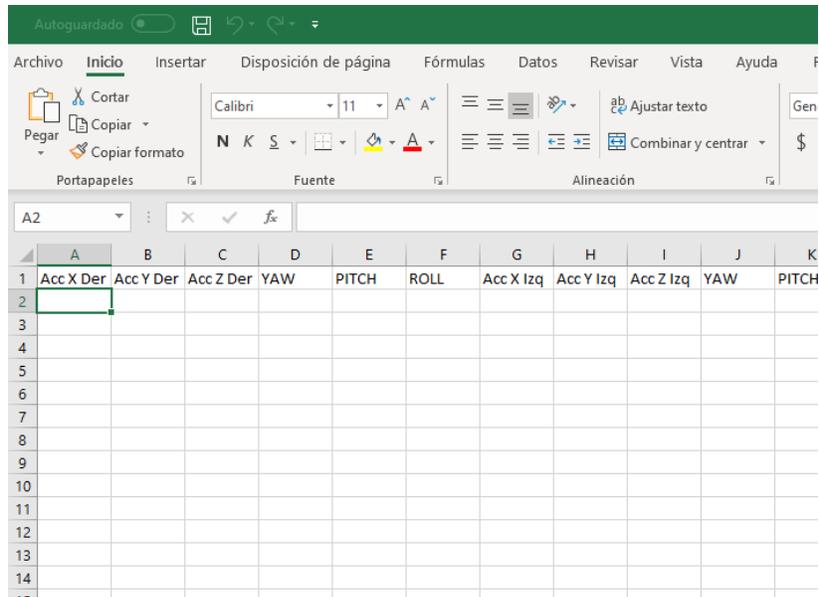


Figura 42 Casilla inicial donde deben colocarse los datos del archivo con extensión .txt.

4.- En automático se te entregaran los resultados previamente mencionados.

Nombre	Fecha	Género	Mano domin	Grato académico	Especialidad
		Rango de valores de desempeño			Tarea: Ejemplo.
		Niveles de desempeño			Sutura de herida superficial con punto simple
		Tempo (S)	Distancia (cm)		
Path Length Derecha		Experto 20-30	150-300		
1270.466772		Intermedio 31-45	301-600	pathlength	pathlength
Path Length Izqu		Principiante 46 - procedimiento no concluid	601-Procedimiento no concluido	11.2063775	9.37438122
8919382248		Entrega de resultados del participante			
Path Length Total		Excelente poco tiempo y menor distancia			
2162.402996		Bueno regular tiempo y regular distancia			
		Suficiente tiempo alto y mayor o menor distancia			
		Insuficiente			
Tiempo		Evaluación de videos por el experto con el uso de rubrica			
143.7		Excelente			
		Bueno			
		Suficiente			
		Insuficiente			
Mano derecha					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X	
6.889	2.173	3.193	1.683	74.059	
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	
-74	-0.008	0.243953892	0.328107928	0.328076711	
Mano izquierda					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X	
2.706	0.828	1.015	1.186	70.18	
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	
70	-0.136	0.032625151	0.059880314	0.048445751	-0.044

Figura 43 Plantilla con los datos analizados.

Manual para el estudiante.

Estandarización de la actividad

Primeros pasos

Se deberá realizar la actividad de manera como si se estuviera en un caso en la vida real, esto para tener una estandarización y recrear de manera practica una actividad común que se realizaría en la vida cotidiana al ejercer la profesión de médico.

Como parte de tu desarrollo en el rendimiento en la tarea de la sutura, deberás contar con el material necesario para realizar esta actividad, en dado caso que no contaras con todo el material se te brindará, todo este material brindado, se deberá reponer en las sesiones futuras.

También la idea de la actividad es acostumbrarte a que, como estudiante, sigas los lineamientos que se necesitan al entrar a cualquier quirófano cuando se realizan estas actividades, estarás generando una rutina para que desarrolles de manera correcta cualquier actividad relacionada con la sutura, y así al entrar a la práctica profesional no se te complique el desenvolvimiento de la actividad en el entorno, al igual que al estar con un paciente cuentas con un poco de experiencia para brindar confianza a las personas que atenderás.

Otra ventaja al usar el dispositivo consiste en reducir los errores médicos derivados de la falta de práctica.

Continuando con la actividad primero es necesario realizar la técnica de lavado de manos, esta técnica te servirá a lo largo de tu carrera profesional, a continuación, se te brindará un ejemplo de cómo debes realizar este ejercicio.

Técnica de higiene de las manos con soluciones alcoholadas.



Duración:
De 20 a 30 segundos



1. Deposite en la palma de la mano una dosis de producto suficiente para cubrir toda la superficie a tratar.

2. Frótese las palmas de las manos entre sí.

3. Frótese la palma de la mano derecha contra el dorso de la mano izquierda entrelazando los dedos, y viceversa

4. Frótese las palmas de las manos entre sí, con los dedos entrelazados.



5. Frótese el dorso de los dedos de una mano con la palma de la mano opuesta, agarrándose los dedos.

6. Frótese con un movimiento de rotación el pulgar izquierdo atrapándolo con la palma de la mano derecha y viceversa.

7. Frótese la punta de los dedos de la mano derecha contra la palma de la mano izquierda, haciendo un movimiento de rotación, y viceversa.

...una vez secas, sus manos son seguras

Basado en información de OMS

Técnica de lavado de las manos con agua y jabón.



Duración:
De 40 a 60 segundos



0. Mójese las manos con agua.

1. Deposite en la palma de la mano una cantidad de jabón suficiente para cubrir todas las superficies de las manos.

2. Frótese las palmas de las manos entre sí.

3. Frótese la palma de la mano derecha contra el dorso de la mano izquierda entrelazando los dedos, y viceversa.



4. Frótese las palmas de las manos entre sí, con los dedos entrelazados.

5. Frótese el dorso de los dedos de una mano con la palma de la mano opuesta, agarrándose los dedos.

6. Frótese con un movimiento de rotación el pulgar izquierdo atrapándolo con la palma de la mano derecha, y viceversa.

7. Frótese la punta de los dedos de la mano derecha contra la palma de la mano izquierda, haciendo un movimiento de rotación, y viceversa.



8. Enjuáguese las manos con agua.

9. Séquelas con una toalla de un solo uso.

10. Sírvese de la toalla para cerrar el grifo.

...una vez secas, sus manos son seguras

Basado en información de OMS

Figura 44 Técnica de higiene de manos según la OMS.

Materiales

El material que se usará para realizar el procedimiento de sutura es el que se muestra a continuación:

- Pinzas Adson
- Portagujas
- Tijera Mayo, recta o curva cual sea del agrado del usuario.
- Sutura Nylon 2 – 0
- Guantes médicos de la talla del usuario
- Dispositivo Entrenador para Sutura con Título de Modelo de Utilidad No. 4128 por el Instituto Mexicano de la Propiedad Intelectual. Dispositivo, trabajo realizado con el apoyo del programa UNAM, DGAPA, PAPIIME, Clave del proyecto. PE202217 para realizar la práctica.

➤

Material	Descripción
<p data-bbox="451 1142 659 1171">Pinzas Adson</p> 	<p data-bbox="898 1142 1385 1318">La pinza Adson es una pinza hecha de acero inoxidable, compuesta por dos varillas metálicas unidas por un extremo dentado o puntiagudo.</p> <p data-bbox="898 1360 1385 1497">Al ser una pinza de disección, se utiliza principalmente para atraer, comprimir, aproximar y sujetar tejidos tanto finos como duros.</p>
<p data-bbox="459 1509 651 1539">Porta agujas</p> 	<p data-bbox="898 1509 1385 1791">Los porta-agujas agarran la aguja entre garras especialmente diseñadas, incluyendo frecuentemente un cierre dentado para mantener la aguja agarrada con tensión. Son dirigidos mediante movimientos de pronosupinación del antebrazo.</p>

<p style="text-align: center;">Tijera Mayo</p> 	<p>Las tijeras mayo son un utensilio quirúrgico con múltiples usos, usado para seccionar, cortar y separar los tejidos. Puede haber pinzas rectas o curvas como se muestran en la imagen.</p>
<p style="text-align: center;">Sutura Nylon 2-0</p> 	<p>NYLON es una sutura de poliamida monofilamento. Gracias a su buen deslizamiento a través de los tejidos y su elevada resistencia a la tracción, resulta especialmente adecuada para la microcirugía donde los puntos deben ser muy delicados.</p>
<p style="text-align: center;">Guante Médicos</p> 	<p>Los guantes médicos son guantes desechables utilizados durante procedimientos médicos que impiden la contaminación cruzada entre el personal de la salud y los pacientes.</p>
<p style="text-align: center;">Dispositivo Entrenador para Sutura</p>	<p>Dispositivo creado para el apoyo al estudiante, con el fin de ayudar en su desarrollo en habilidades de sutura.</p>

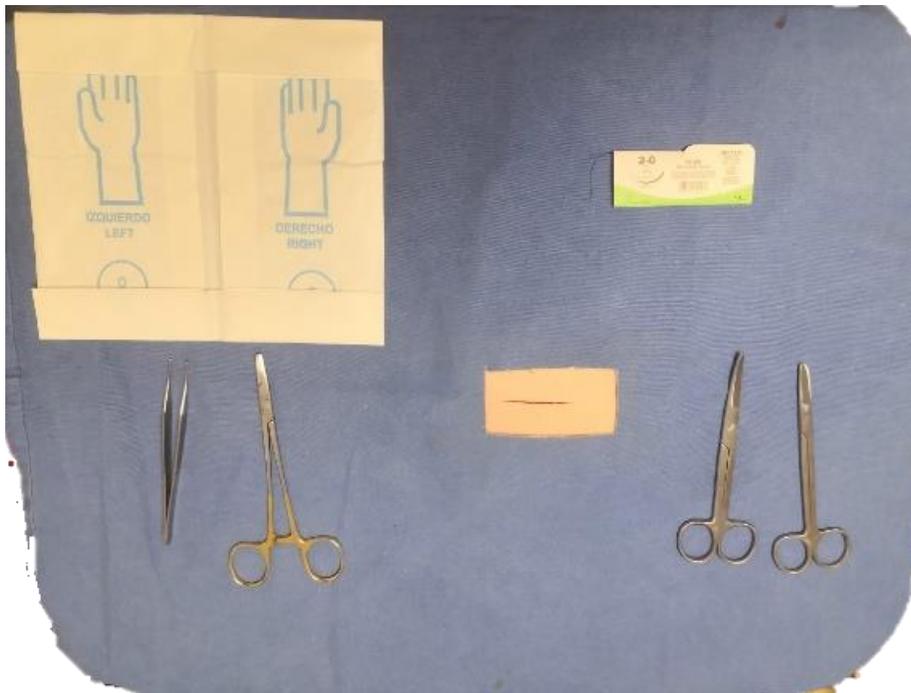
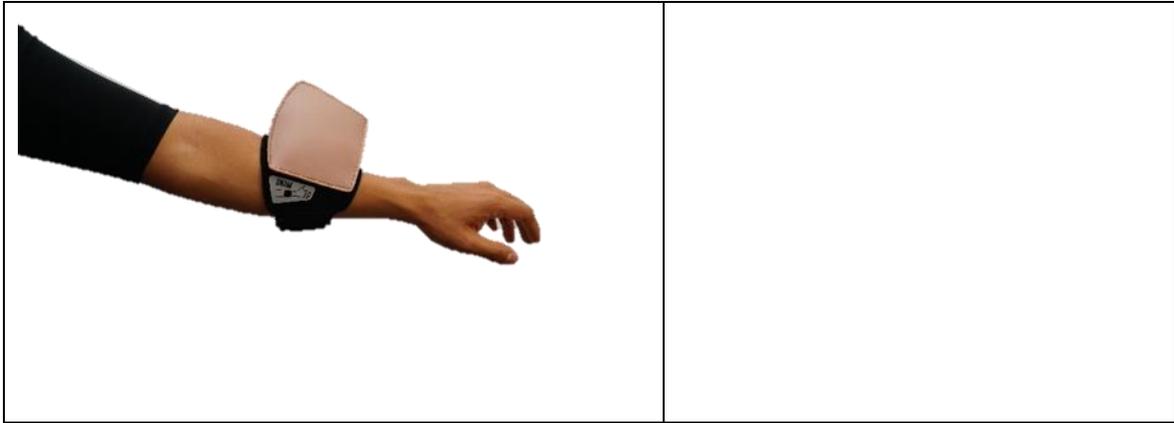


Figura 45 Colocación de la instrumentación al momento de realizar las pruebas. Siempre tener las pinzas al frente de la persona, la referencia se puede ver en la imagen anterior, esto para que sea más fácil la ubicación de los utensilios y tener una organización mayor de estos.

Posición inicial para el registro

Para poder realizar la correcta estandarización, es necesario seguir los siguientes pasos:

1.- Colocarse dentro de área que se marca en el piso, esto debido a que de igual manera se tomara capturas de video para poder ayudar a la evaluación del alumno y de esa manera, se te será complementado un comentario el cual te ayudara a cómo mejorar en el rendimiento de la prueba.



Figura 46 Posición dentro del área delimitada para la prueba.

2.- Colocar la instrumentación y los materiales como se muestra en la figura

3.- Después de colocarte en la posición marcada y tener todo tu instrumental y material en la mano, deberás de colocar los guantes médicos y los guantes de los sensores



Figura 47 Colocación de los guantes médicos y guantes de los sensores.

4.- Después de haber montado los sensores, te deberás de colocar en la posición inicial con los instrumentos y el material, esto para poder empezar a hacer el registro.

5.- Empezar con el registro usando el programa que sea de su agrado, ya sea el programa de Processing o el de MATLAB®. Este funcionamiento se detallará más adelante.

Nota: A continuación, se te mostraran una serie de fotografías en las cuales se muestran detalladamente como debes de seguir el proceso, al igual del uso correcto de los instrumentos y como deberás colocarte para iniciar el registro.



Figura 48 Muestra de cómo deben de estar montados los sensores sobre la parte posterior del brazo, al igual que los cables siempre deberán de tener la dirección hacia los brazos.

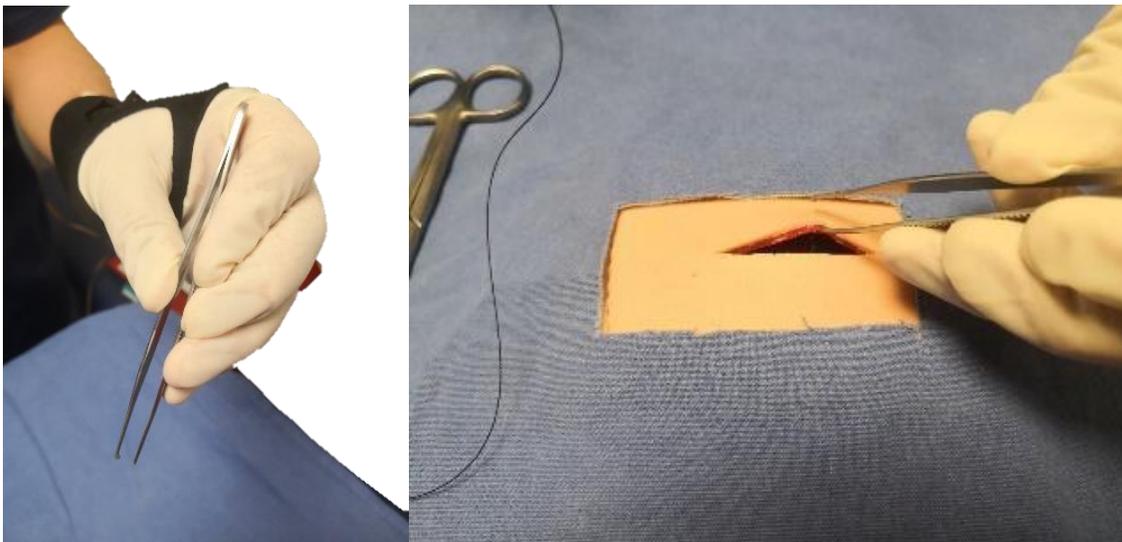


Figura 49 Forma correcta de la sujeción de la pinza, esto se ilustra de manera correcta en la parte izquierda y en la parte derecha, la manera correcta de sujetar la piel, en este caso no se debe presionar la pinza, ya que en caso contrario se puede lastimar al paciente.

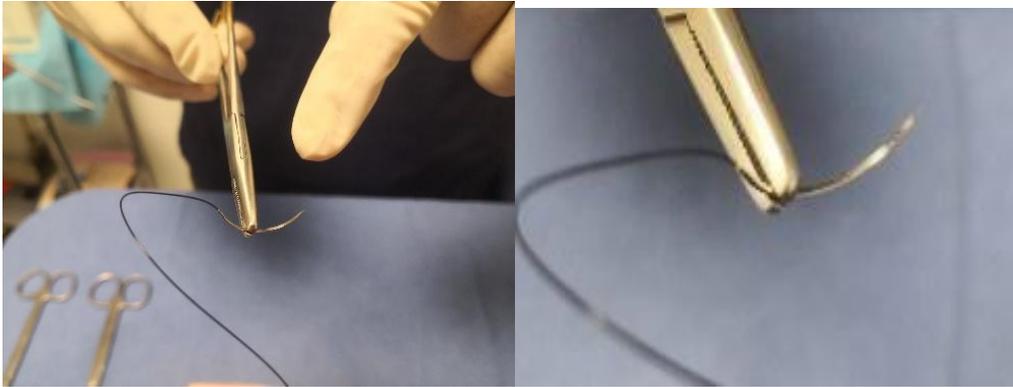


Figura 50 Ilustración de como sujetar correctamente la sutura Nylon 2-0, esta debe de estar sujeta a tres cuartos de la misma aguja, esto para poder atravesar todas las capas de la piel y en dado caso si se llegara a fracturar, poder extraer de manera sencilla el fragmento trozado.

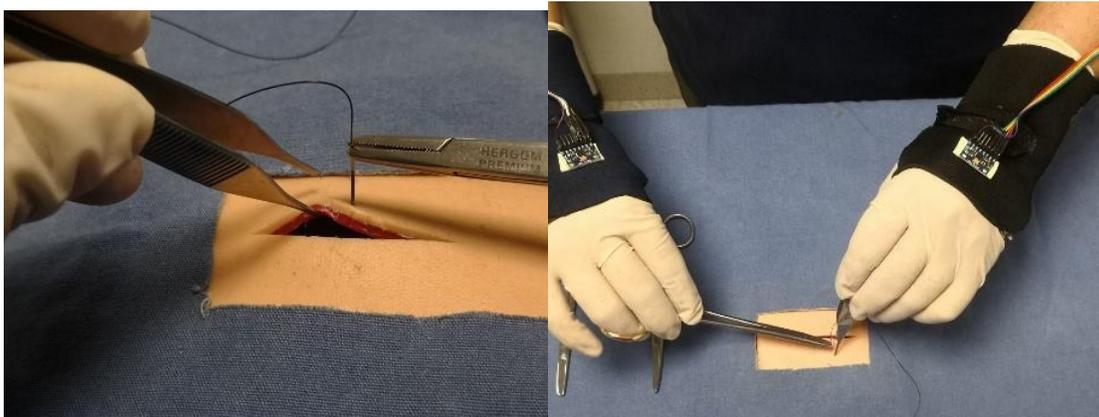


Figura 51 Posición inicial para poder iniciar el registro. Así deberás de colocar la aguja y se ilustra la manera de como deberás manipular la cortada de la piel.

Como usar el dispositivo y los programas

Se te brindara una carpeta con todos los archivos necesarios, en este caso se incluirán 3 archivos Arduinos® y uno en PROCESSING® o MATLAB®, de acuerdo con la necesidad o gusto que se tenga de cada persona, se recomienda usar el programa de Processing ya que, al ser una interfaz más amigable, su uso es más sencillo, y se evitan muchas complicaciones, pero por lo mismo se tienen más limitantes.

En caso de que las necesidades o destrezas de la persona sean superiores, se podrá brindar el programa de MATLAB®, este al estar enfocado a desarrolladores y modelos matemáticos más completos, nos brindara un entorno más completo, por esto mismo, se necesitara una comprensión de programas más a fondo, en este material se te brindara lo básico para poder comprender el programa que se usó y también algunos consejos que podrían brindar para mejorar el uso de este lenguaje.

Programas de ARDUINO®

Al ser un lenguaje casi universal, o que la mayoría de las personas han estado en contacto, se ha optado porque su uso sea de una manera sencilla, utilizando recursos que si llegara a surgir cualquier problema, se pueda solucionar de manera sencilla y rápida, toda esta información se podría profundizar si se desea explotar al máximo los recursos de los sensores, en este caso al ser de uso práctico, los programas no te serán difíciles de usar, ya que solo se necesita seguir la instrucciones que se te serán brindadas a continuación de una manera clara y sencilla.

La programación de la tarjeta de Arduino® UNO para la adquisición de los datos de los acelerómetros se compone de dos fases: el cálculo de offset de cada sensor, y la programación de la rutina para adquirir los datos.

a) Cálculo de OFFSET (Calibración)

Para empezar que es el offset, es el dato que te brindara el sensor cuando este en su posición inicial, esta posición inicial es necesaria para que la calibración sea haga de manera correcta y así evitar que los datos que se obtengan sean muy incoherentes o se genere un error problemático al momento del análisis de los datos.

Cada IMU tiene determinado Offset, que son valores “únicos” para cada dispositivo. Para garantizar que las IMUs leen adecuadamente, lo primero que se debe hacer es obtener su OFFSET.

En primer lugar, se debe conectar la IMU (siempre), tal como lo muestra la Figura siguiente.

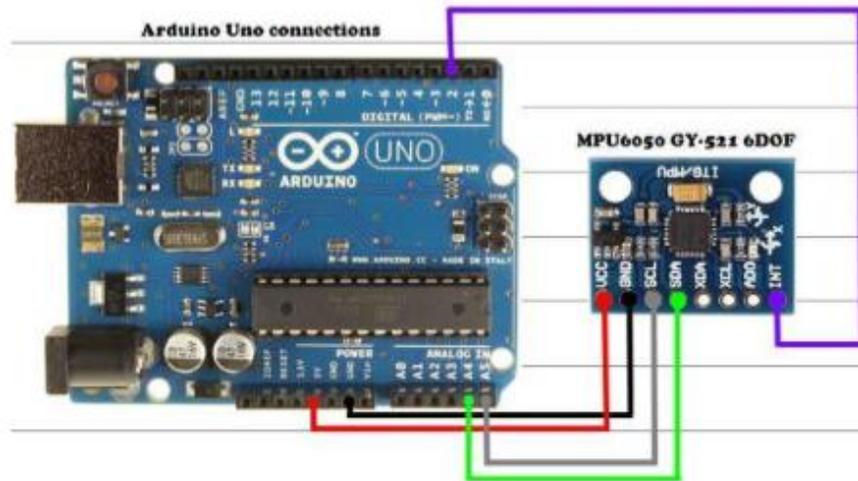


Figura 52 Conexión de IMU MPU6050 con tarjeta Arduino® UNO

Una vez conectados, cargar al Arduino® UNO el programa de calibración MPU6050_simple_calibration la cual será realizada en conjunto con el profesor en curso con el cual se vaya a elaborar las pruebas, esto para que cada alumno tenga el registro de sus datos de posición inicial, y evite en mayor medida cambios en los registros.

Abrir los siguientes sketches o programas de Arduino®:

- | | | |
|--|------------------------|---------------------|
|  MPU6050_DMP6_left_2 | 24/02/2017 08:36 p. m. | Carpeta de archivos |
|  MPU6050_DMP6_rigth_2 | 24/02/2017 08:36 p. m. | Carpeta de archivos |

Figura 53 Sketch de archivo de cada mano.

En estos programas se deberán introducir los valores de los offset que fueron calculados con el profesor.

Es importante decidir en este punto, cuál IMU será colocada en la mano derecha y cuál en la mano izquierda en conjunto con cada Arduino®, esto para poder agregar las carpetas antes mostradas. Así mismo, es importante recalcar que, una vez programados los Arduinos con los valores de OFFSET obtenidos de cada IMU, siempre se conecten de la misma forma, cada Arduino con su propia IMU deberán ser previamente definidos, para evitar cambios en el sistema.

A partir de la línea 168 del código en arduino, aparecen los siguientes comandos.

```
//Introduce aqui los datos obtenidos

mpu.setXAccelOffset(2891);
mpu.setYAccelOffset(-559);
mpu.setZAccelOffset(1258);
mpu.setXGyroOffset(3);
mpu.setYGyroOffset(-30);
mpu.setZGyroOffset(16);
```

Figura 54 Líneas de código a introducir el OFFSET obtenido anteriormente con el profesor.

Los valores entre paréntesis corresponden a los valores del OFFSET de cada eje para el acelerómetro y el giroscopio, y deben ser cambiados por los valores calculados previamente, esta actividad solo se realizará como primera vez cuando sea necesario cambiar de sensores o se necesite hacer una recalibración del sistema.

Configuración de ARDUINOS ®

La interfaz de PROCESSING® y MATLAB® está configurada para que al conectar los Arduinos®, éstos sean reconocidos por la computadora con los puertos COM dependiendo de cada Arduino que se haya registrado

Para verificar la dirección de los puertos, es necesario abrir el administrador de dispositivos de Windows®, el cual se puede abrir presionando las teclas Windows® + X / Administrador de dispositivos o Equipo/ Propiedades / Administrador de dispositivos

Una vez ahí, con los Arduinos® conectados, buscar Puertos COM.

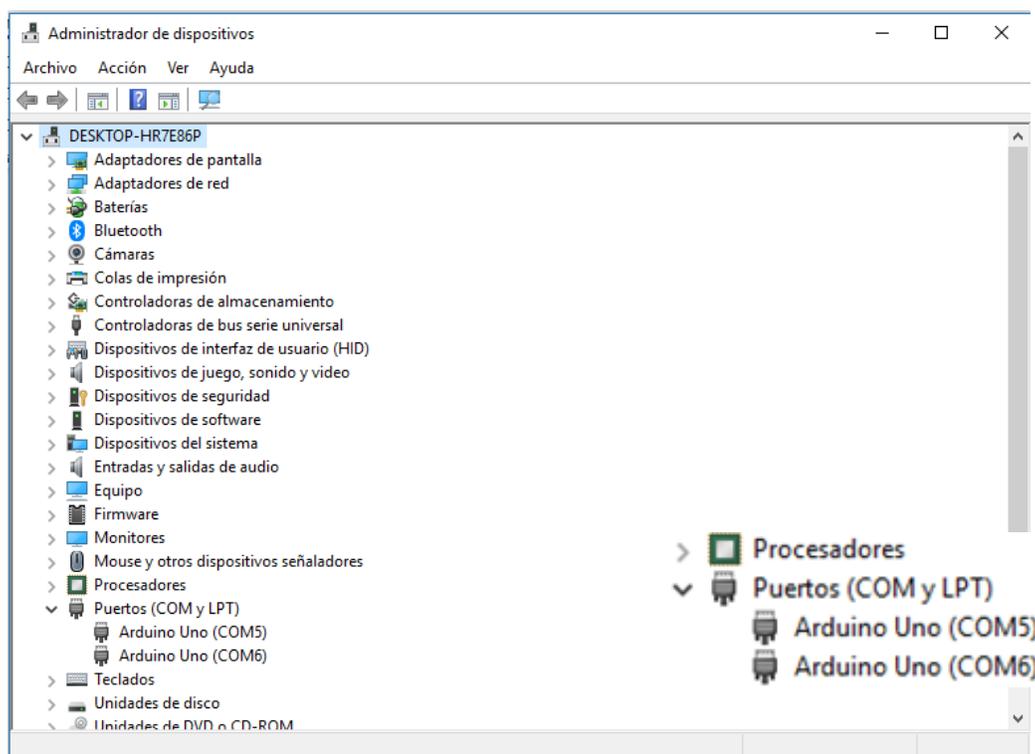


Figura 55 Puertos COM en el administrador de dispositivos.

Colocar el ratón sobre la pestaña de Puertos con el símbolo de una USB, enseguida colocarse en cada Arduino UNO (COMX) que se tenga conectado, para esto primero hay que conectar cada Arduino que ya hayamos establecidos como derecha o izquierda, se recomienda conectar Arduino por Arduino, para poder identificar el

nombre de cada uno y así designar el puerto COM de cada dispositivo de una manera más sencilla. Al final solo hay que dar clic derecho y seleccionar la pestaña de PROPIEDADES.

Aparecerá otra pantalla (Ver figura siguiente). Pulsar en la pestaña Configuración del puerto

En la nueva ventana que aparece pulsar opciones avanzadas

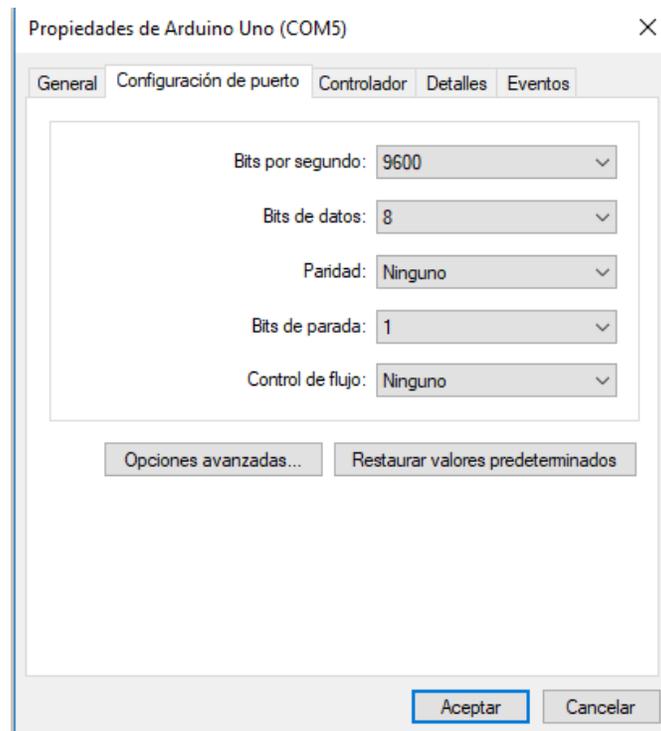


Figura 56 Configuración del puerto

Finalmente aparecerá la ventana mostrada en la Figura 4, en la que se puede cambiar el número del puerto COM. Seleccionar la COM que deseemos dependiendo de la IMU a utilizar de cada mano, para poder establecerlas en los programas que se usaran a continuación.

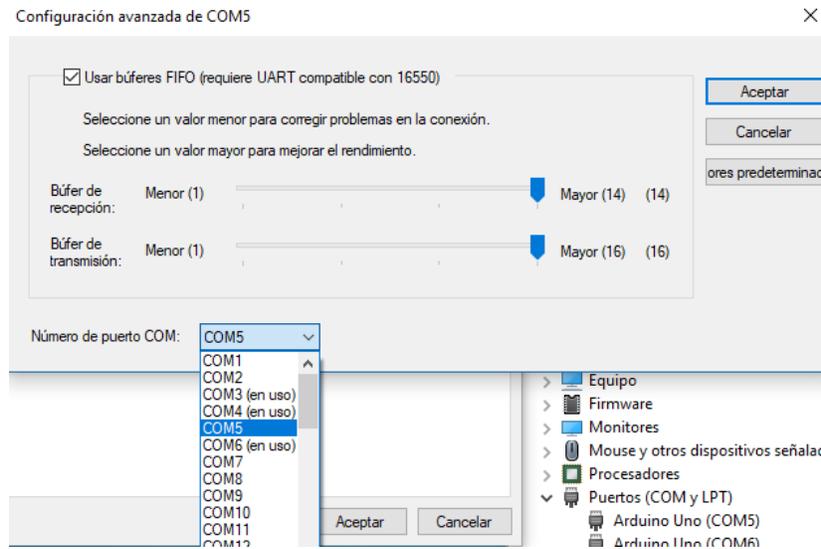


Figura 57 Opciones avanzadas

Pasos para seguir para poder usar los programas

Subir cada sketch de Arduino® que anteriormente fueron programado en su respectivo dispositivo de Arduino (Mpu left and right, asignados previamente con sus respectivas COM).

Estos programas se deben de incluir en la carpeta de los archivos brindados del material de apoyo para la configuración de los dispositivos.



Figura 58 Archivos previamente configurados.

Después de haber subido los programas a cada Arduino asignado, ya solo será necesario abrir los programas y seguir las instrucciones que se darán a continuación.

Usando PROCESSING®

Pasos por seguir para poder usar PROCESSING®, ARDUINO® y las MPU:

Librerías

Para poder usar el programa de PROCESSING® correctamente, es necesario conocer los siguientes pasos básicos:

1. Comprender como funciona el programa PROCESSING®. Al ser un programa de open source o código abierto, se puede hacer infinidad de trabajos en esta plataforma, por eso es necesario saber un poco de programación y su escritura, en este manual no se te dará una clase detallada del programa, debido a que previamente se te debe de capacitar si deseas usar este complemento de aprendizaje. (El propósito de saber lo básico en programación, es por si se desea modificar el programa o se haya cometido algún error de puntuación al momento de la edición de este mismo, esto con el fin de que sea más fácil la localización de algún error a futuro).
2. Tener instalado la versión más reciente de JAVA®, ya que este programa trabaja en base con esa plataforma de desarrollo, así que es indispensable contar con ese programa en la maquina con la que se vaya a trabajar.
3. Saber cómo instalar las siguientes librerías, las cuales se explicarán como instalarse a continuación.
 - a) AP-Sync (Comunicación ARDUINO® – PROCESSING®)
 - b) Arduino (Librería para el funcionamiento de las librerías Arduino dentro del programa).
 - c) ControlP5 (Interfaz de Processing usada para los objetos, en este caso el texto y los botones que se muestran en la interfaz).
 - d) G4p (Complementos para la interfaz)

Para descargar cada una de estas librerías, se tiene que ir a la barra de tareas del programa Processing en la pestaña que dice sketch

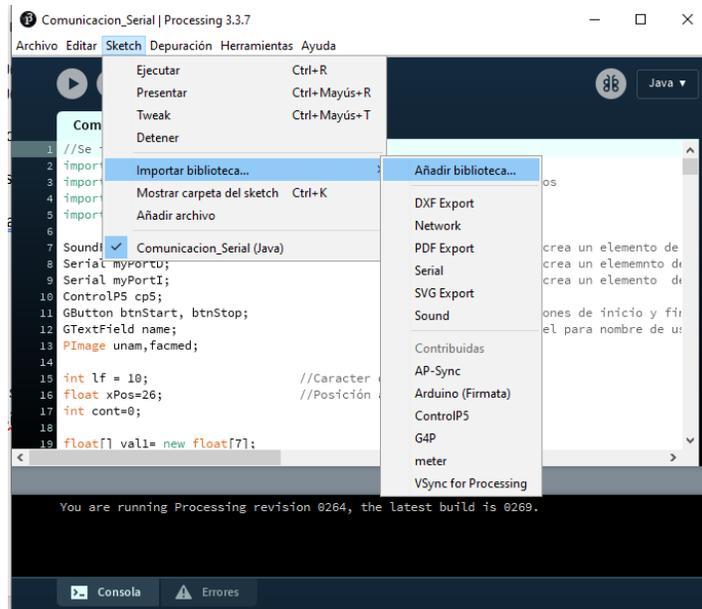


Figura 59 Localización de las librerías.

Haciéndole clic en la pestaña añadir biblioteca saldrá la siguiente ventana

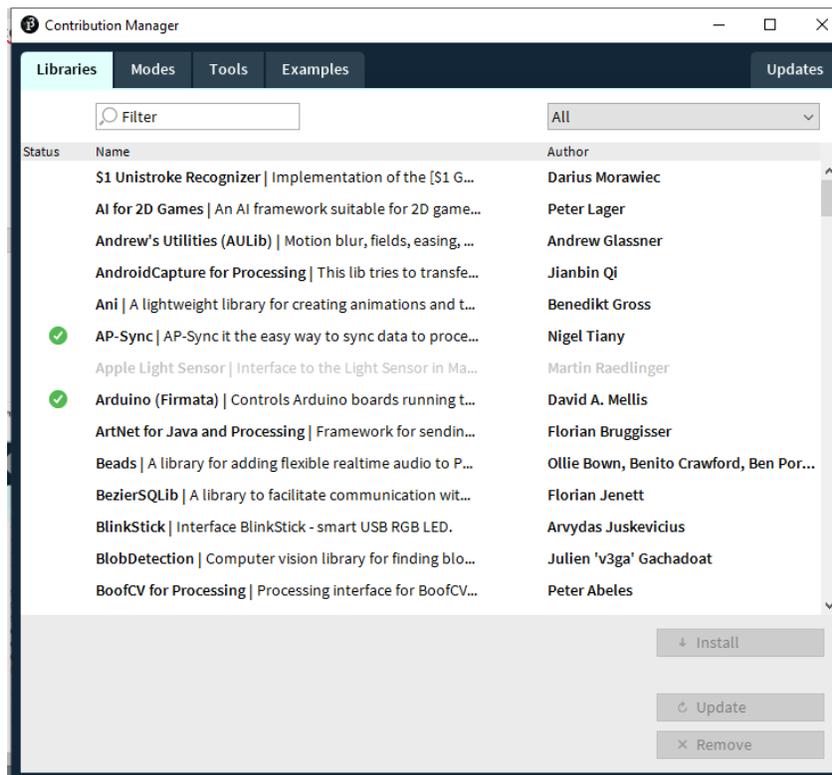


Figura 60 Interfaz de la ventana librerías.

Después solo tendremos que escribir en el buscador las librerías antes mencionadas, después de seleccionar cada librería se tiene que instalar, solo hay que darle en la opción Install y se agregara automáticamente en el programa.

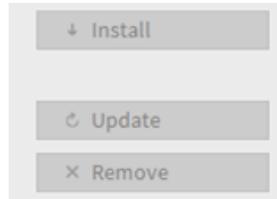


Figura 61 Acciones que se pueden realizar en la ventana librerías.

Estas librerías son esenciales para que haya una comunicación con los Arduino y que la interfaz gráfica pueda desenvolverse de una manera correcta.

Nota: checar siempre las librerías, si se actualiza el Processing, puede que estas sean incompatibles con la versión descargada, en dado caso, se recomendaría regresar a la versión en la cual se esté trabajando o se haya hecho el programa, esta versión que se uso es la 3.3.6 de PROCESSING®.

Ajustando el programa PROCESSING®

En la parte de configuración de la comunicación serial que se encuentra a partir de la línea 62, checar en que puerto se encuentra cada Arduino para poder configurar dentro del programa de Processing los COM correspondientes de cada IMU

```
62 ///////////////////////////////////////////////////////////////////INICIACIÓN DE LA COMUNICACIÓN SERIAL////////////////////////////////////.
63   printArray(Serial.list());
64   myPortD=new Serial(this,"COM5",115200);
65   myPortD.clear();
66   myPortD.bufferUntil(lf);
67   myPortI=new Serial(this,"COM6",115200);
68   myPortI.clear();
69   myPortI.bufferUntil(lf);
```

Figura 62 Líneas a modificar del código en Processing.

Correr el programa, si hay algún error este entrara en conflicto y este no podrá inicializar

Interfaz Gráfica del Usuario PROCESSING®

La interfaz está diseñada para capturar el nombre o el texto que se desee.

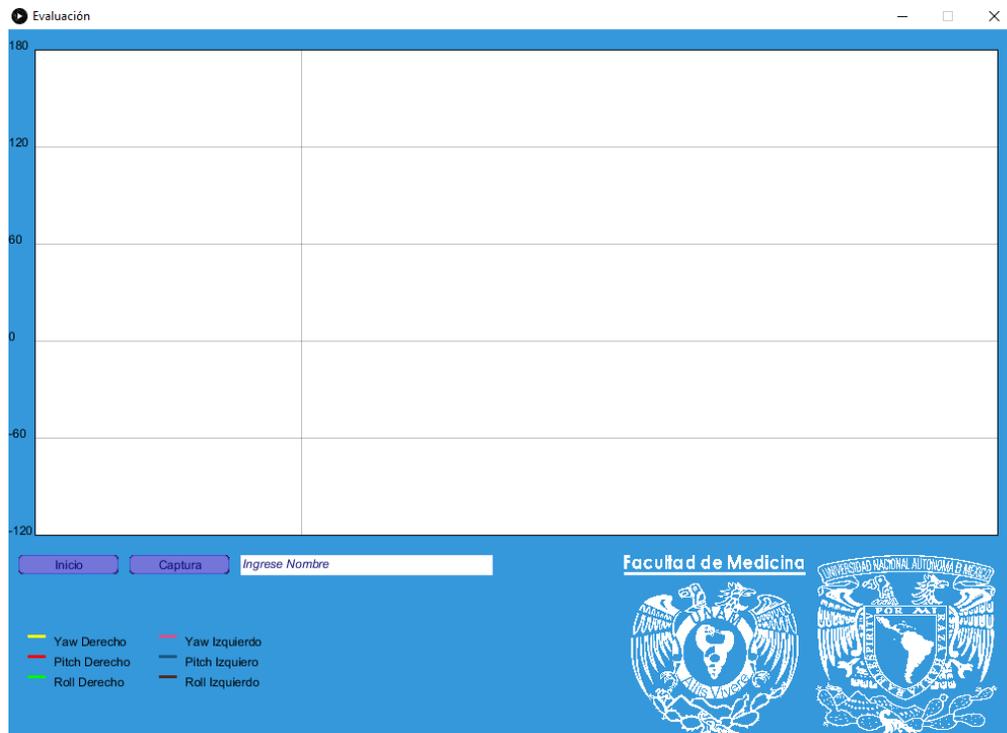


Figura 63 Interfaz gráfica en Processing. Ventana para registro.

Observación: Es importante mencionar que al iniciar la capturar de datos, nunca se deje en blanco el recuadro donde se ingresa el nombre, pues los datos no se podrán grabar correctamente.

Cuando esté todo listo, dar clic en el botón Inicio



Figura 64 Interfaz entre el programa y usuario.

Nota: Jamás cerrar el programa directo de la pestaña, esto propicia a que el registro no se guarde, es necesario siempre antes de cerrar el programa seleccionar la

opción de captura cuando acabe el registro, si no se realiza este paso, los datos no se guardan y no hay manera de recuperarlos y si por error, se le vuelve a presionar la opción de inicio, este entra en conflicto y te lanzara un error.

Para poder tener la prueba de manera correcta, el participante debe colocar sus manos en posición inicial y sin moverlas durante 10 segundos, para que las IMUs se calibren, estos 10 segundos se identificaran de una manera sencilla tan solo observando la línea que se muestra a continuación.

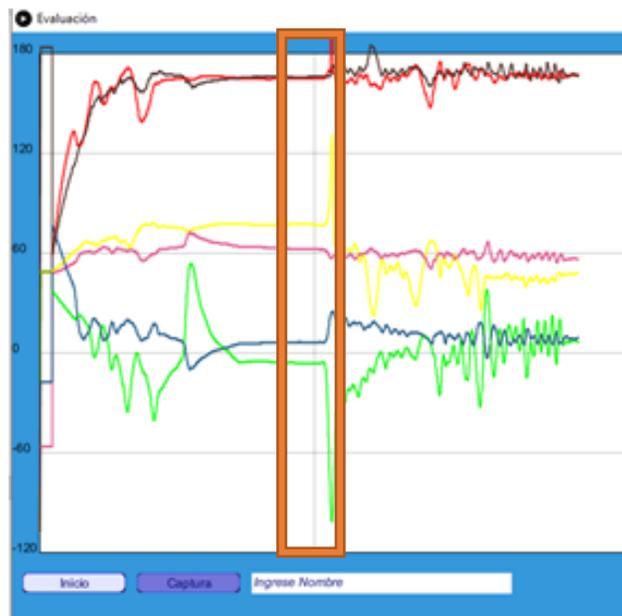


Figura 65 Línea de 10 segundos de calibración.

Si todo está bien conectado, aparecerá una ventana con las gráficas del giroscopio en tiempo real.

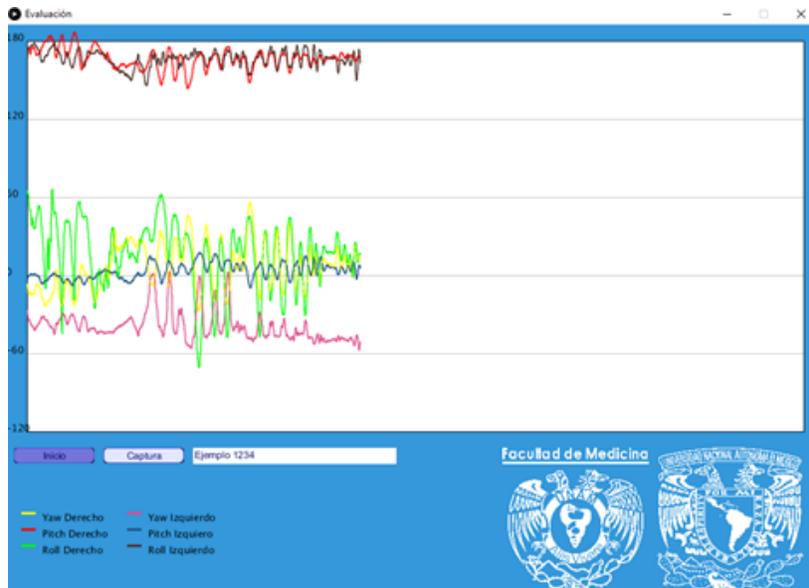


Figura 66 Gráfica en tiempo real del giroscopio de ambas IMUs.

Si dicha ventana no inicia, verificar en la sección de errores cual podría ser un posible problema. La mayoría de las veces es porque la conexión de las IMUs no es la correcta, o porque Processing no reconoce los puertos. Si este es el caso, verificar que los puertos COM sean los correctos, o volver a conectar los ARDUINOS®.

Al finalizar la prueba, la captura de datos se detiene automáticamente al presionar el botón Captura.



Figura 67 Interfaz de selección en el objeto.

El programa terminara y se regresara a la ventana del programa en PROCESSING®, los cuales se brindarán en un archivo con terminación .txt, el cual puede ser leído con el programa de bloc de notas. Lo restante es copiar los datos para poder analizarlos, este análisis se mostrará detalladamente en un apartado más adelante.

Sugerencias

Si se desea trabajar en diferentes computadoras, se necesitará realizar estos pasos nuevamente, por lo que siempre será necesario checar que se tengan instaladas las librerías antes de correr el programa, ya que, de no ser así, podrán crearse conflictos con la computadora y se generaran errores, lo que provocaría que la computadora se llegara a detendrá la computadora y esta no reaccionara en un cierto tiempo.

Otro punto importante es, que siempre contemos con los archivos esenciales para el correcto funcionamiento del programa, estos archivos son los que se han mencionado con anterioridad.

Errores frecuentes en PROCESSING®

Si aparece la siguiente pantalla, es debido a que el programa no reconoce los Arduinos, o estos no se han configurado correctamente.

Una manera de solucionarlo es que se cierre automáticamente el programa con las indicaciones que te van apareciendo, en dado caso que no se pueda cerrar, la forma más efectiva será la que se mencionara a continuación.

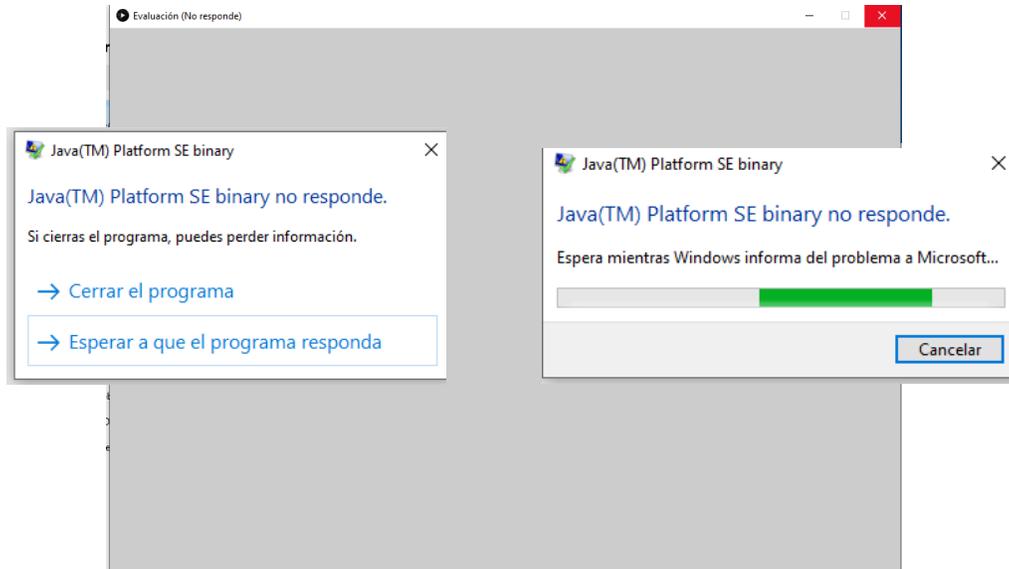


Figura 68 Error más frecuente en la interfaz, al no tener todas las características antes mencionadas o saturación en la memoria.

Tendremos que abrir el administrador de tareas, y para esto es necesario presionar las teclas, Ctrl+Alt+Supr, después de presionar estas teclas, nos aparecerá un menú, el cual nos mostrara las opciones de administrador de tareas, este podrá aparecer de las siguientes formas:

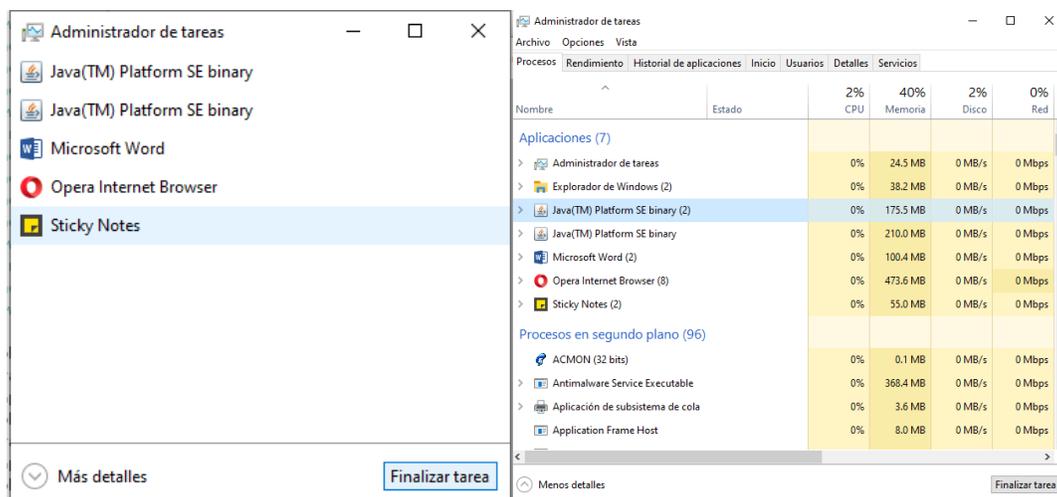


Figura 69 Solución rápida al estancamiento del programa.

Y lo único que se tendrá que realizar, es seleccionar el programa Java(TM) Platform SE binary (2) y darle en finalizar tarea en la parte inferior derecha del administrador,

se debe cerrar este programa, ya que es el que crea la interfaz, ya que si se cierra el java que no tiene la terminación 2, se cerraran todas las pestañas, incluyendo el programa donde se está utilizando el Processing.

Otro caso particular es el que jamás se inicialice la captura de datos, y esto se puede deber a diferentes factores.

Uno de estos es que, si notamos que solo se mueven 3 variables en vez de 6, es debido a que una de las IMUs está mal conectada, o se está creando algún falso, lo que provoca que se muestre una línea continua y solo 3 variables se muevan, esto se soluciona únicamente checando que la conexión de Arduino® y las IMUs estén correctamente conectadas, o en dado caso si se detectara un falso, arreglarlo rápidamente para poder realizar las pruebas.

Que jamás se inicialice o se detenga el registro de las variables, esto es debido a que como no se limita el tiempo, se genera un registro demasiado grande, la manera de solucionar esto es reiniciando los Arduinos® desde su botón reset, para limpiar la memoria cache y evite conflictos entre programas.

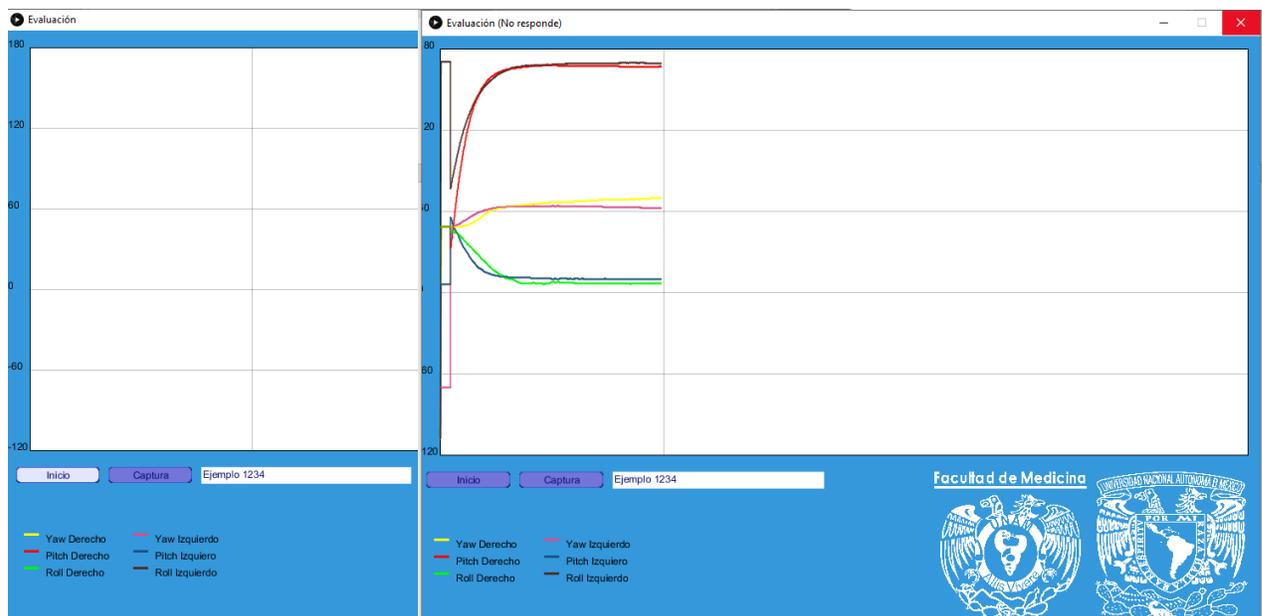


Figura 70 Saturación de memoria en la interfaz de Processing.

Otro error muy común es el que se muestra a continuación, en el que se ve la interfaz sin nada. Este error se genera debido a que la carpeta donde se tienen los archivos visuales que se muestran la interfaz, no se contengan dentro de la misma carpeta de donde se tenga el programa de PROCESSING®.

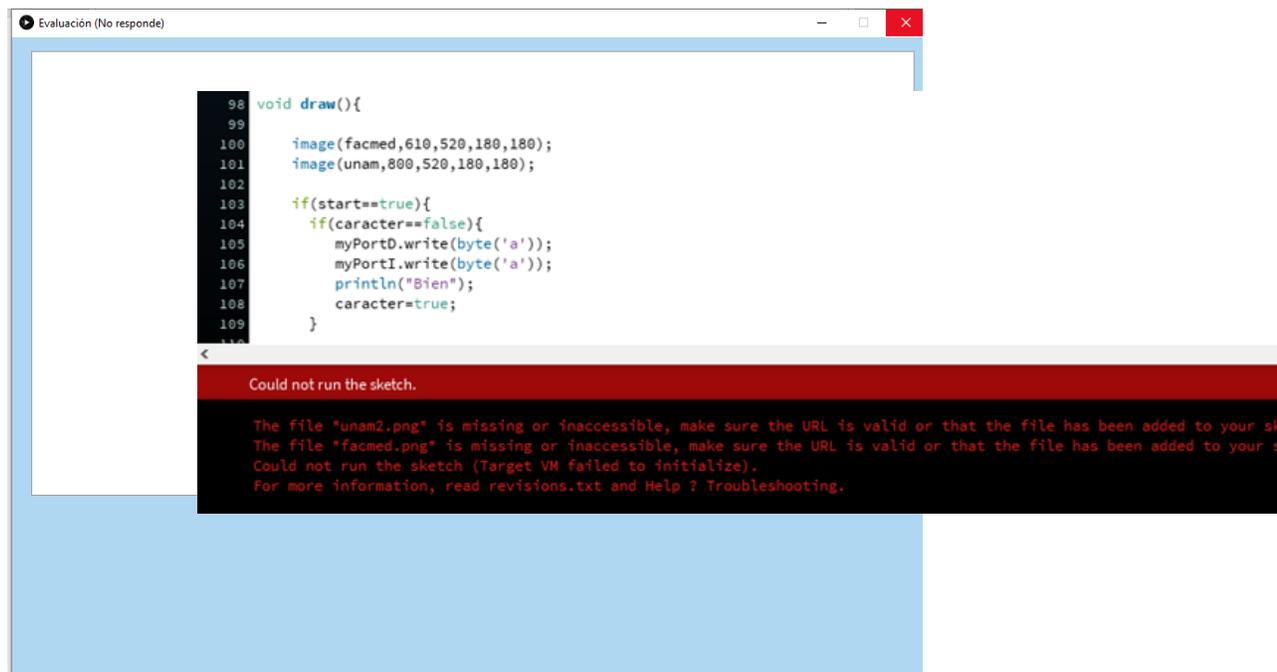


Figura 71 Localización de la mala configuración del Processing.

La forma de solucionarlo es teniendo la carpeta (**data**) en la misma carpeta donde se esté manejando el programa como se muestra en la siguiente imagen, esta carpeta contiene todos los recursos gráficos del programa, para evitar una edición más dentro del programa de PROCESSING®.

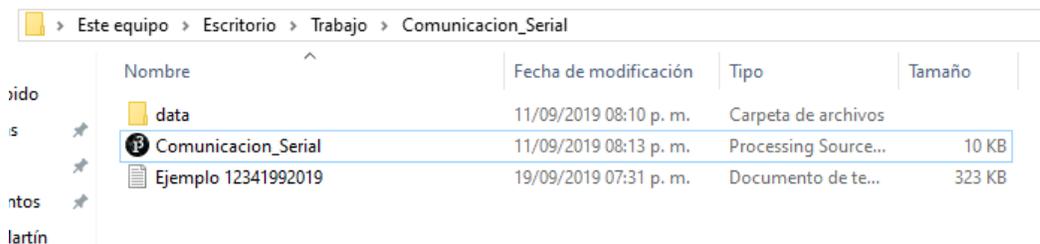


Figura 72 Archivos del programa PROCESSING®.

En dado caso que no se llegara a contar con la carpeta, se puede solucionar comentando las siguientes líneas 216 y 217 del programa con los caracteres '//', lo que hacen estas barras en diagonal es poner el texto en gris, haciendo así que el programa no llame a estas rutinas, lo que ayudara a que no se tengan conflictos, e igual el programa se podrá usar sin los archivos de la interfaz.

```
213 ///////////////////////////////////////////////////RUTINA PARA ACTUALIZAR LA GRÁFICA//////////////////////////////////////
214
215 void window(){
216     unam=LoadImage("unam2.png");
217     facmed=LoadImage("facmed.png");
218
```

Figura 72.1 Líneas de código para comentar.

Usando MATLAB®

Tener los archivos correspondientes para que haya un correcto funcionamiento, los cuales son los siguientes, siempre deberán estar en la misma carpeta en la que se corra el programa.

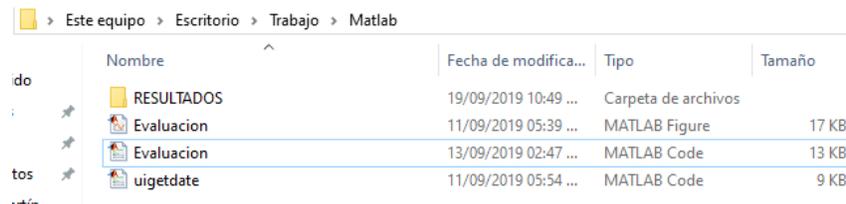


Figura 73 Archivos básicos de MATLAB®.

Los archivos esenciales son los dos archivos de Evaluación y el de uigetdate, que es lo que hace cada uno, a continuación, se detallara su función.

Evaluación (MATLAB Figure), este archivo es el que se encarga de mostrar la interfaz gráfica, esta se programa según las necesidades deseadas, en este caso se usa para poder ingresar los datos de la persona.

Evaluación (MATALAB Code), este archivo es el encargado de hacer todo el procesamiento de la información, aquí mismo se programan los comandos del archivo de la interfaz, ya que estos mismos se muestran dentro del código.

Uigetdate (MATLAB Code), este es un archivo que se obtiene a través de la página, funciona como una librería para obtener los datos de la fecha actualizada.

En el programa evaluacion.m editar los siguientes datos referentes a las COM de cada IMU para que estos coincidan y así Matlab pueda funcionar correctamente

```

232
233 %borrar datos de comunicación serial previos
234 delete(instrfind({'Port'},{'COM5'}));
235 delete(instrfind({'Port'},{'COM6'}));
236
237 % delete(instrfind({'Port'},{'COM8'}));
238 % delete(instrfind({'Port'},{'COM7'}));
239
240 %crear objetos serie
241 a = serial('COM5','BaudRate',115200,'Terminator','CR/LF');
242 b = serial('COM6','BaudRate',115200,'Terminator','CR/LF');

```

Figura 74 Líneas del programa MATLAB® code a modificar.

Interfaz Gráfica de Usuario (GUIDE) MATLAB®

La interfaz está diseñada para capturar los datos del participante, la fecha y hora de la prueba, y para la adquisición y exportación de los datos de la IMU a los formatos de Excel (extensión xls), Matlab (extensión

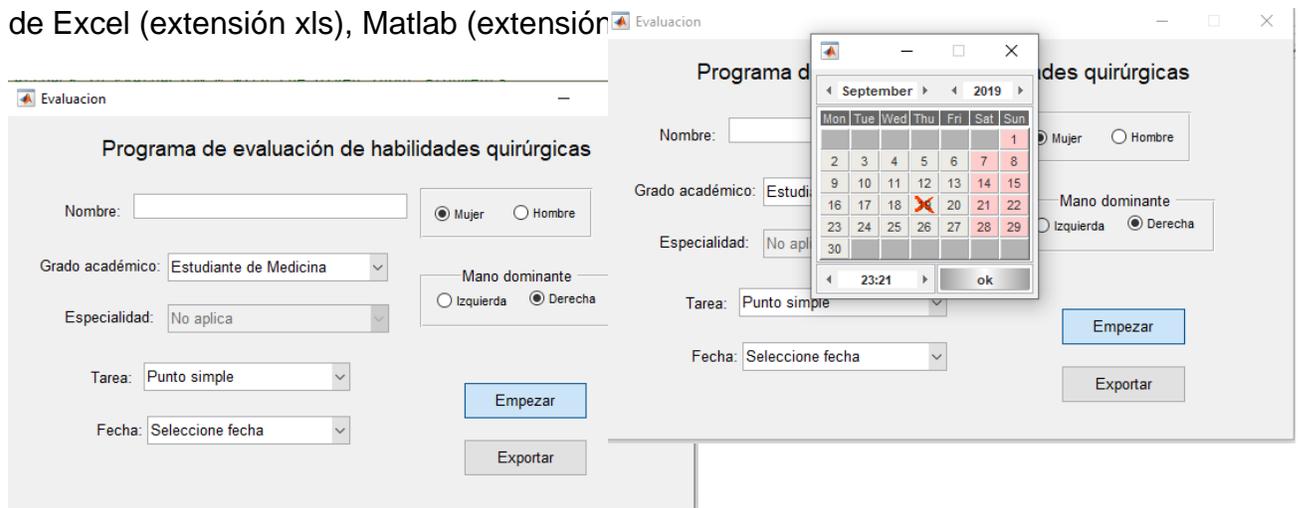


Figura 75 Interfaz de la aplicación en MATLAB®. Ventana para registro de participantes.

Es importante mencionar que, al capturar el nombre, nunca se deje un espacio en blanco al final, pues los datos no podrán ser exportados.

Para comenzar con la prueba, el participante debe colocar sus manos en posición inicial y sin moverlas durante 10 segundos, para que las IMUs se calibren.

Cuando esté todo listo, dar clic en el botón Empezar

Si todo está bien conectado, aparecerá una ventana con las gráficas del giroscopio en tiempo real, (ver figura 9).

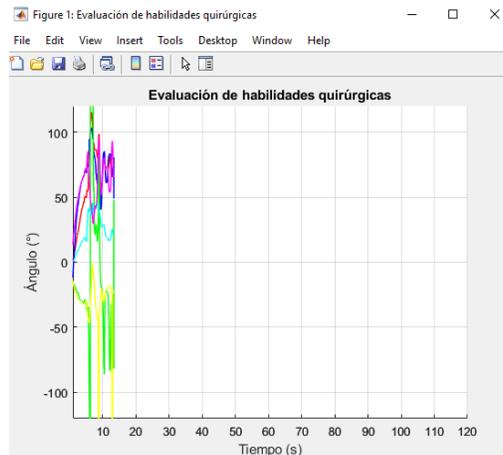


Figura 76 Gráfica en tiempo real del giroscopio de ambas IMUs.

Si dicha ventana no aparece, verificar en la ventana de comandos de MATLAB® (Command Window) el error. La mayoría de las veces es porque la conexión de las IMUs no es la correcta, o porque MATLAB® no reconoce los puertos. Si este es el caso, verificar que los puertos COM sean los correctos, o volver a conectar los Arduinos®.

Al finalizar la prueba, la captura de datos se detiene automáticamente al cerrar la ventana. Y volvemos a la ventana inicial.

Lo restante es obtener los datos dando en las diferentes extensiones haciendo clic en el botón exportar. Se creará una carpeta en resultados y ahí se agregarán los archivos.

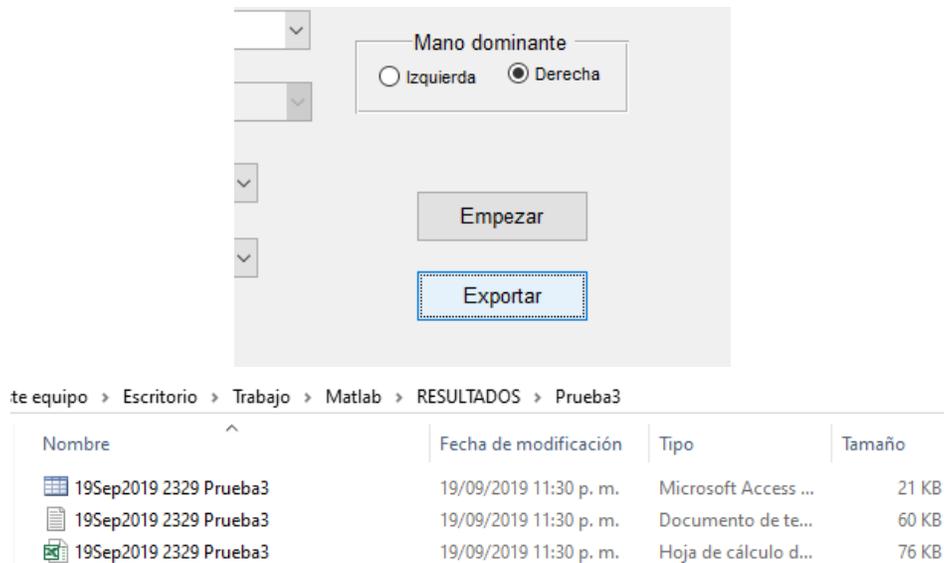


Figura 77 Exportación de datos y su formato de entrega.

Antes de iniciar la siguiente prueba, se deben borrar los datos y las variables almacenadas en la memoria de MATLAB®, ya que, si no se borran, los datos del nuevo participante se sobrescribirán en los del participante anterior.

Para ello, es necesario limpiar todas las variables, funciones e interrupciones que se hayan utilizado. También es conveniente limpiar la ventana de comandos, para que se puedan visualizar de mejor manera los mensajes de error, si es que llegara a existir un problema.

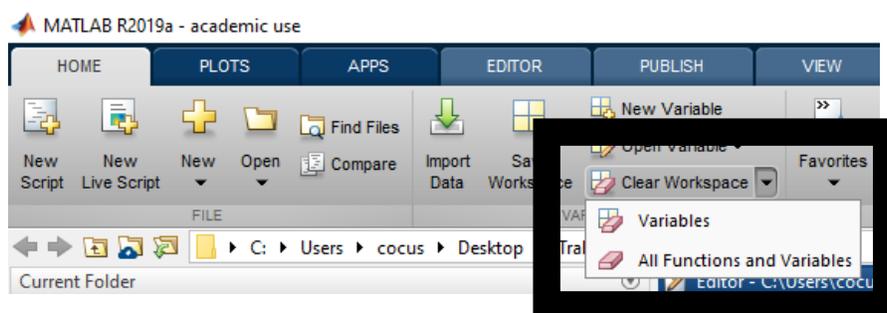


Figura 78 Limpiar variables, funciones e interrupciones.

Errores Frecuentes en MATLAB®

Errores en los puertos seriales, esto surge cuando no se configuraron bien los COM en el programa o estos no están siendo detectados por la computadora.

```
Command Window
>> Evaluacion
Error using serial/fopen (line 72)
Open failed: Port: COM5 is not available. No ports are available.
Use INSTRFIND to determine if other instrument objects are connected to the requested device.

Error in Evaluacion>btn_empezar_Callback (line 249)
fopen(a);

Error in gui_mainfcn (line 95)
feval(varargin{:});

Error in Evaluacion (line 42)
gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)Evaluacion('btn_empezar_Callback',hObject,eventdata,guidata(hObject))
Error while evaluating UIControl Callback.

fx >>
```

Figura 79 Error en los puertos COM.

Para solucionar este problema, solo hay que checar que en la programación si se configuraron bien los puertos COM, también checar que estén bien conectados y la computadora estén siendo reconocidos.

Errores por la figura de la interfaz (cambio de máquina, uso del mismo programa en otra computadora)

```
Command Window
To use this device with SAMBA, install SAMBA support package for Arduino hardware.

Warning: Matching failure in format. The format specified for conversion, '%f,%f,%f,%f,%f,%f', is incorrect.
Warning: Matching failure in format. The format specified for conversion, '%E,%f,%E,%f,%E,%f', is incorrect.
Index exceeds the number of array elements (2).

Error in Evaluacion>btn_empezar_Callback (line 292)
c3(i)=s(3);

Error in gui_mainfcn (line 95)
feval(varargin{:});

Error in Evaluacion (line 42)
gui_mainfcn(gui_State, varargin{:});

Error in
matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata)Evaluacion('btn_empezar_Callback',hObject,eventdata,guidata(hObject))
Error while evaluating UIControl Callback.

fx >>
```

Figura 80 Error mostrado al usar el mismo programa en diferentes computadoras y no configurarlas en la que se usa.

Para solucionarlos, se guarda de nuevo en la interfaz gráfica en la computadora que se esté usando, esto para que la dirección quede guardada en la computadora y Matlab pueda leerla.

Se detiene automáticamente o no inicializa la ventana de las gráficas.

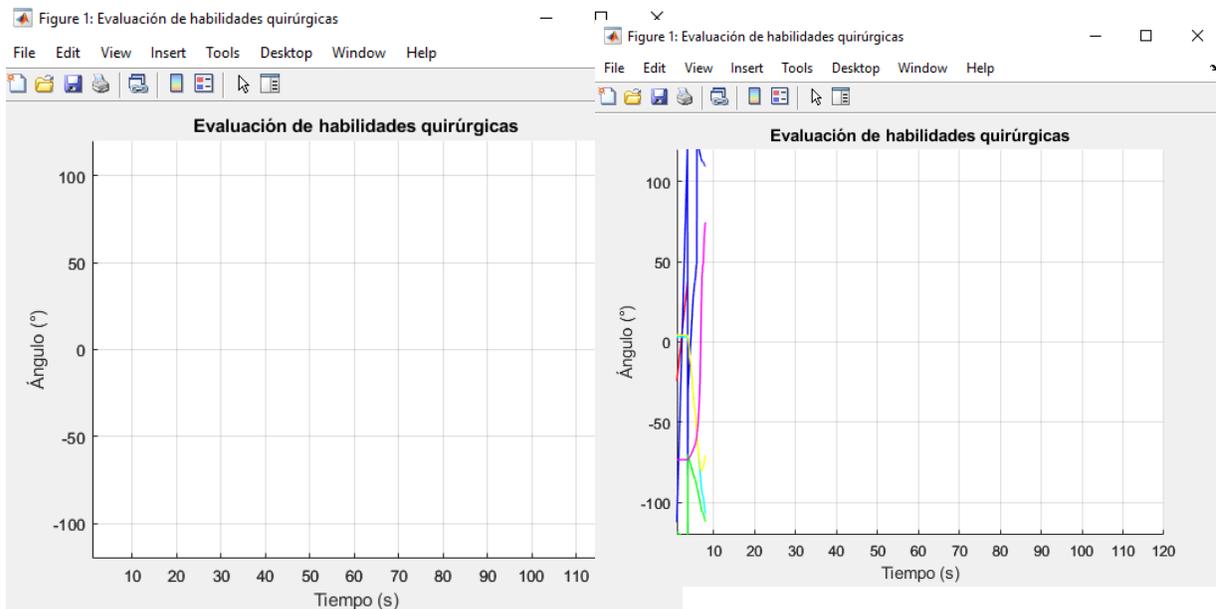


Figura 81 Inmovilización del sistema por la mala configuración.

Estos problemas suceden cuando se crea un almacenamiento de datos muy grande, provocando que las cadenas no puedan ser leídas por el Matlab antes de empezar el registro, otra razón es por no estar en el punto inicial al empezar la calibración para la prueba. Una manera de solucionarlo es limpiando todas las variables desde el menú de la consola de MATLAB®.

La razón más crítica que sucede es cuando se llena la memoria, la razón es porque al ser en tiempo real, MATLAB® entra en conflicto y al no tener un punto final, crea error en el programa, lo que causa un error en la computadora, y se tiene que reiniciar MATLAB®.

Manejo de los datos para la evaluación

Uso de la plantilla. – Esta plantilla te mostrara los resultados obtenidos a lo largo de tu evaluación o práctica, esto con el fin para que tengas seguimiento de la mejoría que has tenido, o dependiendo el caso, también te muestra las faltas en las habilidades que están careciendo mayor crecimiento, esto con el propósito de que tomes en cuenta tus puntos malos para mejorarlos y tener un mejor desenvolvimiento en el medio de la cirugía.

Nombre	Fecha	Género	Mano domin	Grado académico	Especialidad
Rango de valores de desempeño				Tarea: Ejemplo,	
Niveles de desempeño				Sutura de herida superficial con punto simple	
		<i>Tiempo (s)</i>	<i>Distancia (cm)</i>		
Path Length Derecha	Experto	20-30	150-300		
0	Intermedio	31-45	301-600		
Path Length izqu	Principiante 46 - procedimiento no concluid 601-Procedimiento no concluido				
0	Entrega de resultados del participante				
	Excelente poco tiempo y menor distancia				
Path Length Total	Bueno regular tiempo y regular distancia				
0	Suficiente tiempo alto y mayor o menor distancia				
Tiempo	Insuficiente				
0	Evaluación de videos por el experto con el uso de rubrica				
	Excelente				
	Bueno				
	Suficiente				
	Insuficiente				
Mano derecha					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en X	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	0	0
#NUM!	#NUM!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Mano izquierda					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en X	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	0	0
#NUM!	#NUM!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!

Figura 82 Interfaz de la plantilla y los datos que brinda.

Las características que tiene esta plantilla de reporte de resultados para el usuario son las siguientes variables: en color oro se indica la tarea quirúrgica a realizar, en color azul cielo se muestra los datos sociodemográficos del participante, en color amarillo se reporta las variables path length (Distancia recorrida) por mano y total y el tiempo en el que realizo el procedimiento y en color gris se muestra el resultado de las aceleraciones de la mano derecha y la mano izquierda, y en verde la evaluación obtenida de acuerdo a los diferentes niveles evaluados.

Figura 83 Azul cielo, datos sociodemográficos del participante.

Path Lenght Derecha	0
Path Lenght izqu	0
Path Lenght Total	0
Tiempo	0

Figura 84 Amarillo, variables Path Lenght (Distancia recorrida).

Rango de valores de desempeño		
Niveles de desempeño	Tiempo (S)	Distancia (cm)
Experto	20-30	150-300
Intermedio	31-45	301-600
Principiante	46 - procedimiento no concluid	601-Procedimiento no cor

Entrega de resultados del participante	
Excelente	poco tiempo y menor distancia
Bueno	regular tiempo y regular distancia
Suficiente	tiempo alto y mayor o menor distancia
Insuficiente	

Evaluacion de videos por el experto con el uso de rubrica	
Excelente	
Bueno	
Suficiente	
Insuficiente	

Figura 85 Verde, diferentes niveles evaluados.

Tarea; Ejemplo.
Sutura de herida superficial con punto simple

Figura 86 Color oro se indica la tarea quirúrgica a realizar.

Mano derecha						
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	0	#¡NUM!	#¡NUM!
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Z
#¡NUM!	#¡NUM!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!

Mano izquierda						
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X
0	0	0	0	0	#¡NUM!	#¡NUM!
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Z
#¡NUM!	#¡NUM!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!	#¡DIV/0!

Figura 87 Color gris, resultado de las aceleraciones.

Para su uso correcto solo se tiene que hacer lo siguiente:

1.- Abrir el archivo .txt que se generó a través de la evaluación. Seleccionar todos los datos del archivo y copiarlos, para no hacerlo copiar dato por dato, una manera más sencilla, debes colocarte al inicio del archivo, a continuación, solo debes presionar las teclas Ctrl+E y automáticamente se seleccionan todos los datos del archivo.

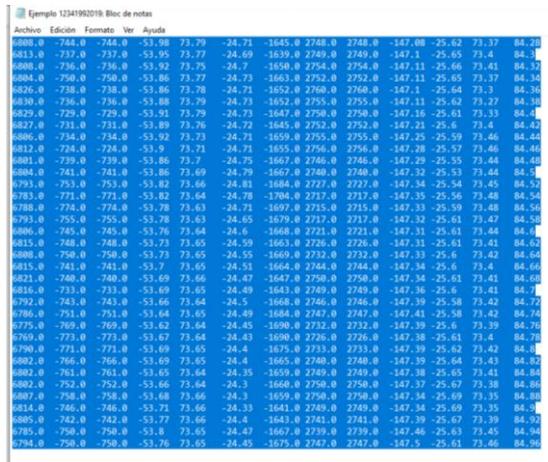


Figura 88 Manejo de datos en el archivo .txt.

Abrir en la pestaña del Excel la parte de datos que se encuentra en la parte inferior derecha.

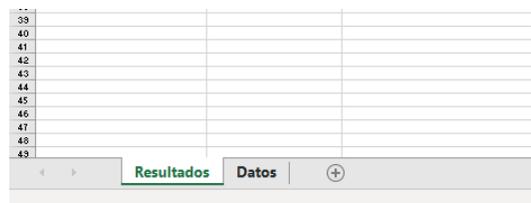


Figura 89 Pestañas de trabajo para el manejo de datos.

Basta con posicionarse en la primera casilla de la plantilla y pegarlos, ya sea manualmente o con el comando Ctrl+V.

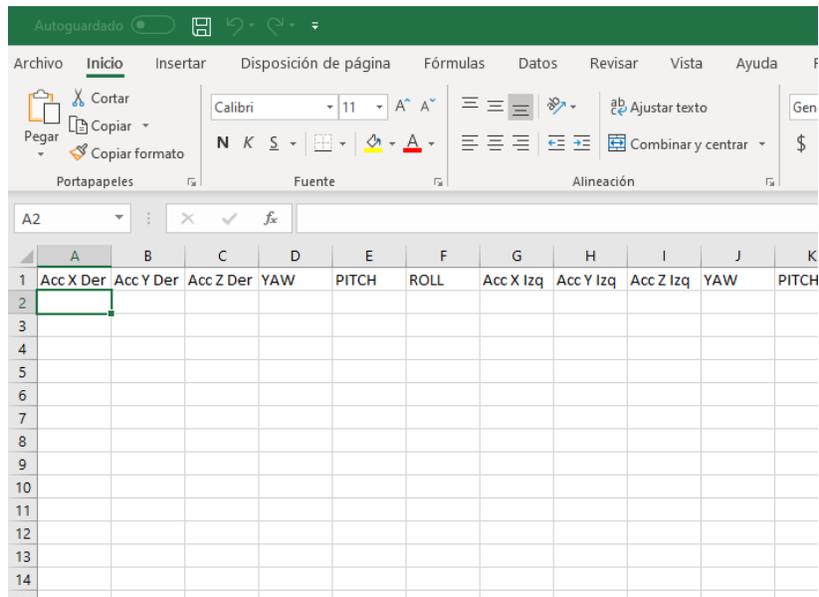


Figura 90 Casilla inicial donde deben colocarse los datos del archivo .Txt.

En automático los resultados se generarán automáticamente.

Nombre	Fecha	Género	Mano domin	Grado académico	Especialidad
Rango de valores de desempeño			Tarea: Ejemplo.		
Niveles de desempeño			Sutura de herida superficial con punto simple		
		Tiempo (S)	Distancia (cm)		
Path Length Derecha	Experto	20-30	150-300		
1270.466772	Intermedio	31-45	301-600		pathlength
Path Length Izq.	Principiante	46 - procedimiento no concluid 601-Procedimiento no concluido			11.2063726
0919302240	Entrega de resultados del participante				
Path Length Total	Excelente	poco tiempo y menor distancia			
282.402396	Buena	regular tiempo y regular distancia			
Tiempo	Suficiente	tiempo alto y mayor o menor distancia			
143.7	Insuficiente				
Evaluación de videos por el experto con el uso de rubrica					
	Excelente				
	Buena				
	Suficiente				
	Insuficiente				
Mano derecha					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración máxima en X	Aceleración Resultante Media	Aceleración Media en X
6.989	2.173	3.853	1.663	74.069	0.05
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	
-74	-0.006	0.243953932	0.326107326	0.328076711	
Mano izquierda					
Aceleración Total Resultante	Aceleración máxima en Y	Aceleración máxima en Z	Aceleración Resultante Media	Aceleración Media en X	
2.706	0.826	1.015	1.866	70.18	-0.044
Aceleración Media en Y	Aceleración Media en Z	Desv Estandar Aceleración en X	Desv Estandar Aceleración en Y	Desv Estandar Aceleración en Z	
70	-0.136	0.032625161	0.058980314	0.048466761	

Figura 91 Plantilla con los datos analizados.

Validación del sistema.

A lo largo que fuimos realizando pruebas con los estudiantes, fuimos observando cómo había una tendencia en el progreso de estos, por lo que nos dimos a la tarea de comparar nuestro sistema de captura con uno comercial, tenemos en cuenta que son valores que están entregando los acelerómetros.

Se realizaron varias pruebas con nuestro dispositivo y el comercial, que en este caso se usó el de la compañía Motion Capture System®, el cual se llama Shadow, realizamos varias pruebas para poder hacer una caracterización y tener un nivel más confiable con nuestro dispositivo, para que, al momento de realizar las pruebas, tengamos una certeza de que los registros se harán de manera correcta. Los datos de los rangos que nos entrega el dispositivo comercial se muestran en la siguiente tabla:

	Full Scale Range	Resolution
Accelerometer	±2, 4, 8, 16 g	0.18 mg
Gyroscope	±4000 °/s	0.12 °/s
Magnetometer	±8 gauss	0.25 mG

Figura 92 Rangos del Shadow®.

Las pruebas realizadas se enlistaran a continuación, estas pruebas fueron discriminatorias para saber cómo se comportaba nuestro sistema con las modificaciones iniciales que se habían realizado, estas fueron notorias a lo largo de las actividades, se mostraron los cambios notoriamente al llegar a un acercamiento al dispositivo comercial, esto para que el dispositivo tenga una confiabilidad mayor,

ya que al saber cómo se comporta un sistema ya comercial a comparación de uno propio, esto propicia a que el uso de un dispositivo económico sea más viable que los equipos costosos que se ofrecen en el mercado.

Prueba 1:

La primera prueba que se realizó fue para ver que acercamiento se tenía el dispositivo comercial con el creado, en esta evaluación se mostraron las diferencias y que se carecían con nuestro dispositivo en comparación con él comercial, después de este acercamiento, notamos que se tenían tendencias en los registros, aunque la adquisición de datos no era muy similar, mostrándonos así que se debían de realizar algunos ajustes en nuestro sistema.

Se montaron los sensores para poder hacer la comparación de los sistemas, tomamos en cuenta que se tuviera el mismo punto de referencia, en este caso se usaron los dos sensores del Shadow de la piernas o muslos, como se muestra en las siguientes imágenes:



Figura 93 Primer acercamiento del dispositivo de evaluación junto con el dispositivo Shadow (Correas sobre la persona).



Figura 94 Puntos de referencia que se usaron para establecer las mismas mediciones para tener una medida aproximada a la de un dispositivo comercial

Para poder calibrar las IMUS en la posición establecida, se usó el programa de calibración, el cual se incluye dentro de los manuales, los valores que este nos entregaba nos daban como referencia el valor en su posición inicial, esto para ingresarlos en los programas que se usan para la evaluación, esto para ver el comportamiento a lo largo de una prueba de los dos dispositivos. Los datos que se obtuvieron en la prueba se muestran en la figura siguiente.

```

Initializing I2C devices...
Testing device connections...
MPU6050 connection successful
PID tuning Each Dot = 100 readings
>.....>.....
at 600 Readings

//          X Accel  Y Accel  Z Accel  X Gyro  Y Gyro  Z Gyro
//OFFSETS  -1734,   2292,   2648,   111,   -75,    31

>.>.700 Total Readings

//          X Accel  Y Accel  Z Accel  X Gyro  Y Gyro  Z Gyro
//OFFSETS  -1760,   2294,   2670,    74,   -94,   -41

>.>.800 Total Readings

//          X Accel  Y Accel  Z Accel  X Gyro  Y Gyro  Z Gyro
//OFFSETS  -1746,   2294,   2692,    57,    19,    40

>.>.900 Total Readings

//          X Accel  Y Accel  Z Accel  X Gyro  Y Gyro  Z Gyro
//OFFSETS  -1746,   2294,   2694,    80,   -54,    47

>.>.1000 Total Readings

//          X Accel  Y Accel  Z Accel  X Gyro  Y Gyro  Z Gyro
//OFFSETS  -1758,   2294,   2694,    86,   -67,    -6

Any of the above offsets will work nice

Lets proof the PID tuning using another method:
averaging 1000 readings each time
expanding:
..

```

Figura 95 Datos obtenidos de nuestro dispositivo al ser colocados en su posición inicial para su calibración.

Estos son los valores que nos entregan las IMUs, de acuerdo con la orientación deseada, estos valores se introducen en el código que anteriormente fue mencionado estos son MPU6050_DMP6 ya sea en left y right respectivamente.

El primer experimento que se realizó arrojó resultados bastantes disparejos, esto pudo haberse debido a que los sensores estaban siendo usados de una manera incorrecta o mal configurados, se nota la diferencia cuando empiezan a usarse cada sensor, pero no recorren la misma trayectoria en las gráficas, tendrían casi el mismo comportamiento en los picos, pero no nos mostraban un patrón similar en todas las direcciones que deseábamos evaluar. Provocando así la diferencia en las aceleraciones, o que al momento de grabarlas estas no estaban adecuadas con las especificaciones necesarias o más comunes, al darnos cuenta de esto lo que se

hizo fue entrar al datasheet de los sensores y ver los rangos en los que trabaja el sensor, esto para saber en qué valores nos están entregando lo que se suponía eran aceleraciones en g/s.

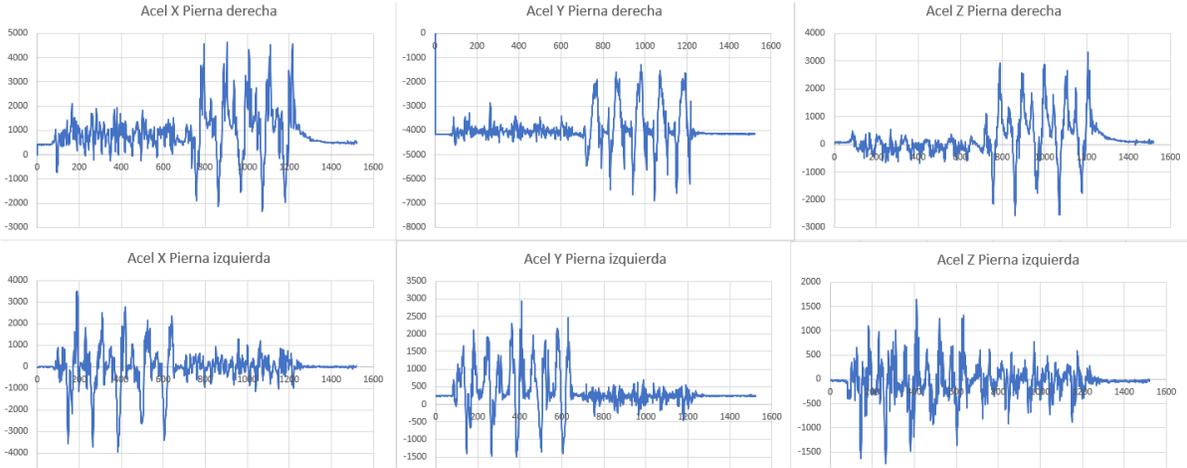


Figura 96 Estas graficas nos muestran los datos con nuestro dispositivo, como se puede observar se registra el movimiento, pero se puede ver un poco de ruido, lo que provoca que registre más datos de los necesarios.

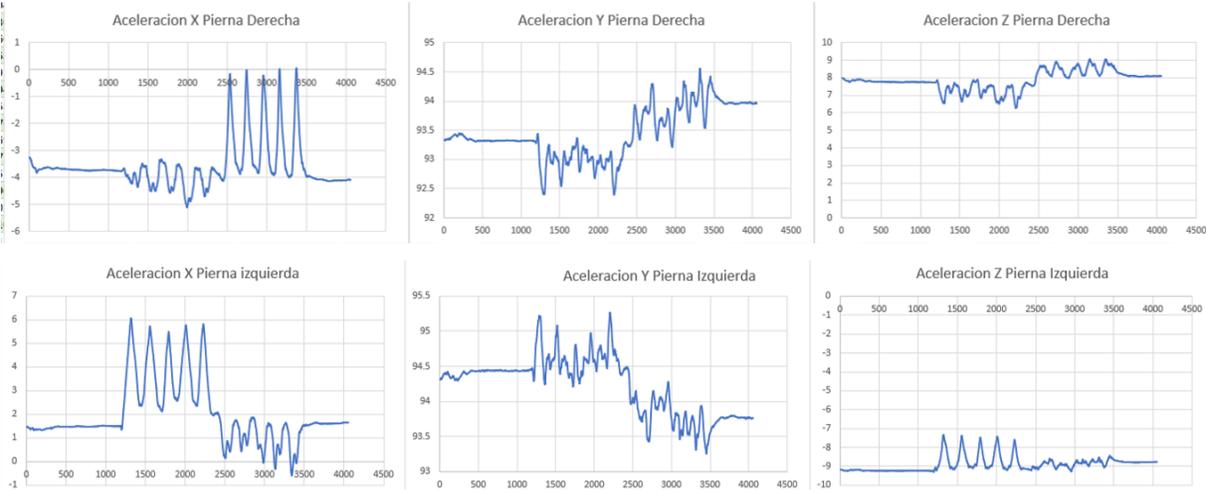


Figura 97 Graficas que nos muestran los datos con el dispositivo comercial.

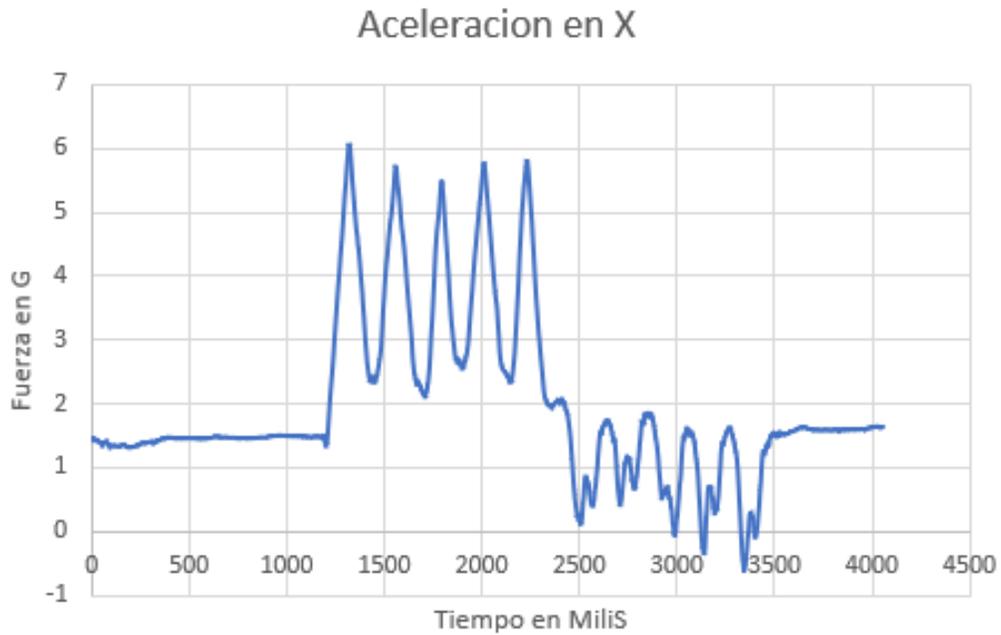


Figura 98 Grafica Fuerza vs Tiempo

Al ver las diferencias que se tenían, investigué acerca de que era los que estábamos midiendo y me di cuenta el ajuste de los valores que se entregan no eran los correctos ya que se estaban trabajando dentro de la IMU a LSB (Least Significant Bit), este es el valor que se nos entrega a través de las IMUS comerciales, por lo consiguiente con la tabla de los valores decidió hacer una segunda prueba aplicando los ajustes necesarios. La siguiente tabla muestra las conversiones necesarias para obtener los datos deseados:

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
ACCELEROMETER SENSITIVITY					
Full-Scale Range	AFS_SEL=0		±2		g
	AFS_SEL=1		±4		g
	AFS_SEL=2		±8		g
	AFS_SEL=3		±16		g
ADC Word Length	Output in two's complement format		16		bits
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/g
	AFS_SEL=1		8,192		LSB/g
	AFS_SEL=2		4,096		LSB/g
	AFS_SEL=3		2,048		LSB/g
Initial Calibration Tolerance					%
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C
Nonlinearity	Best Fit Straight Line		0.5		%
Cross-Axis Sensitivity			±2		%

Figura 99 Tabla de conversiones de valores

La tabla anterior nos muestra las conversiones necesarias para trabajar bajo los valores en G. Los valores en LSB al momento de transfórmalos se entregan en miliG que la equivalencia es 1g=1000 miliG o milig, este valor se puede trabajar directo con los valores entregados o convertirlos dentro del programa, lo que nos indica la tabla es que dependiendo los g en los que queramos trabajar es la conversión que hay que hacer.

Las conversiones de LSB a g se hace de la siguiente manera, con el valor que nos da en los recuadros marcados con el color azul, se divide entre 65,535 que es la equivalencia a los 16 bits ejemplo:

Si queremos trabajar a 2g lo que hay que hacer es meter los valores del 8 en bits dividirlos entre los 16 bits. $16,384 \text{ MilliGs} / 65,535 = 0.2500$ y así dependiendo de los g en los que se quiera trabajar.

Después de convertirlos con el valor deseado, como este está en milis, será necesario convertirlos a g, ya sea dividiendo entre 1000 o multiplicando por 0.001.

Prueba 2:

En la siguiente figura se ilustra la ubicación de los sensores, como se ilustra se usó la misma ubicación, esto para tener una estandarización en las pruebas y así poder validar el sensor de una manera más optima o en dado caso, hacer las modificaciones correctas para que su funcionamiento sea el óptimo.



Figura 100 Ubicación del sensor para la estandarización

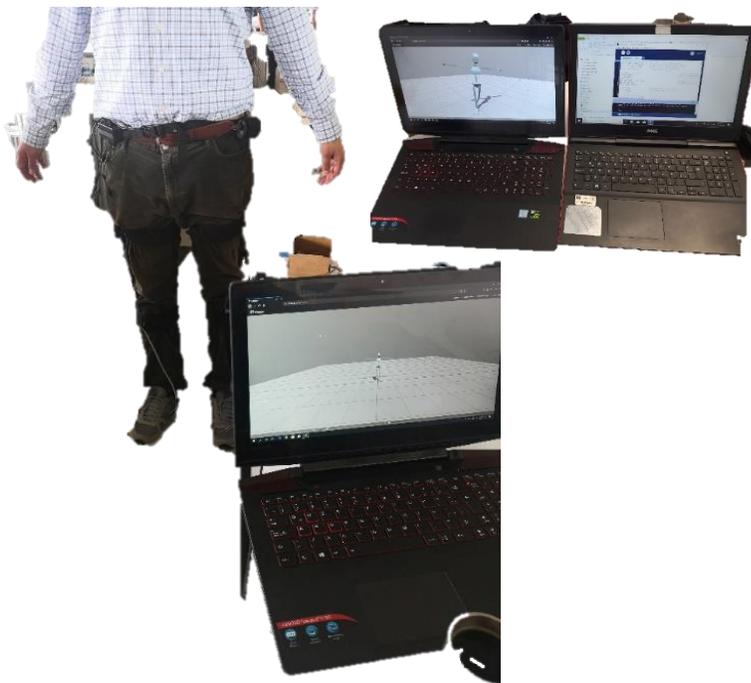


Figura 101 Visualización del dispositivo creado en conjunto con el comercial.

Al realizar la segunda prueba de la evaluación para ver como cambiaba el registro con las adecuaciones que se hicieron, uno sin ajuste y el otro con el nuevo ajuste para ver las diferencias que nos arrojaba el sistema. Se observo como el registro cambiaba de una manera significativa, aquí todavía no se aplicaba el cambio en las unidades, solo queríamos comprobar que los dispositivos actuaran de manera mas similar entre sí, ya que como sabemos, la conversión de unidades se puede realizar durante el proceso o después de este.

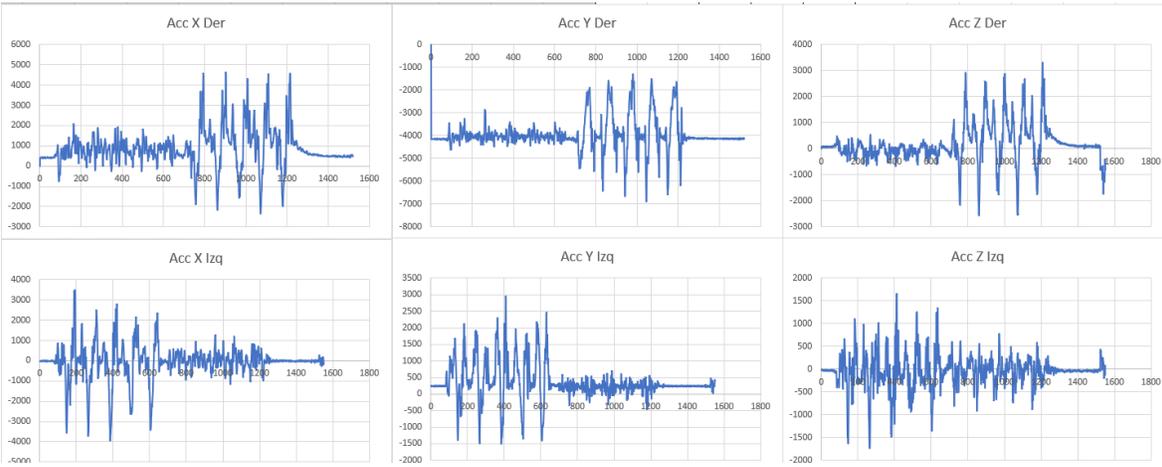


Figura 102 Primera prueba sin el ajuste los valores se entregaban en LSB.

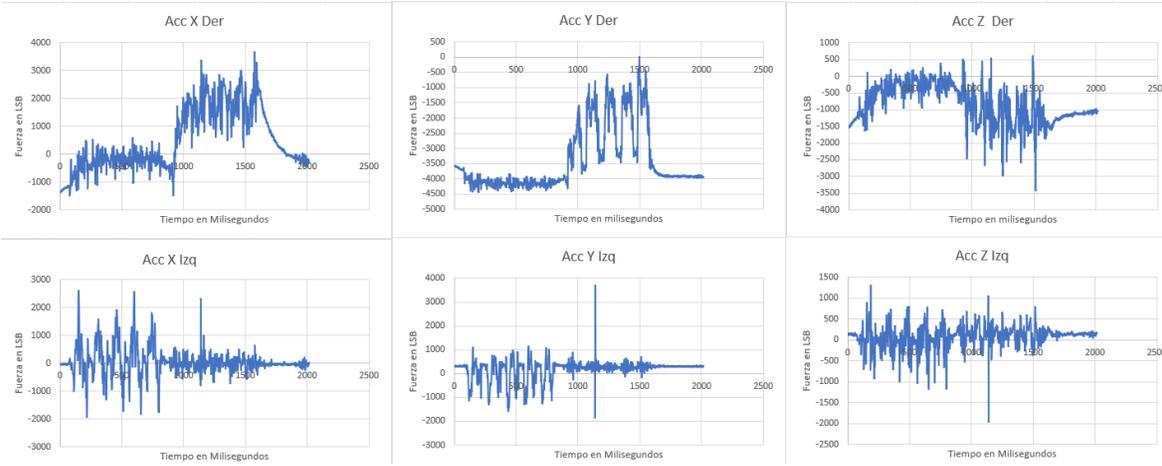


Figura 103 Graficas obtenidas de la segunda aplicación del dispositivo, con el ajuste aplicado.

Las unidas que se muestras en las gráficas a lo largo de las coordenadas X es el tiempo en milisegundos y en lo largo de las coordenadas Y es en LSB que esta

denominación nos expresa a la gravedad en bits, como se muestra en la siguiente imagen.

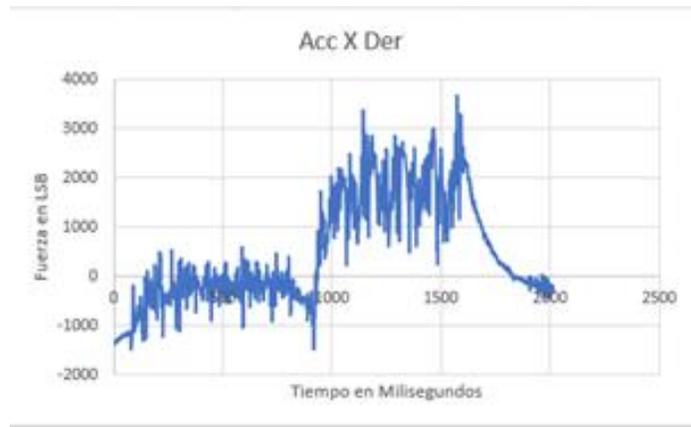


Figura 104 Descripción de Unidades en las graficas

Las primeras 12 graficas nos muestran el cambio que se tuvo entre el dispositivo creado las primeras 6 sin el ajuste realizado y las 6 consiguientes ya con el ajuste previamente hecho.

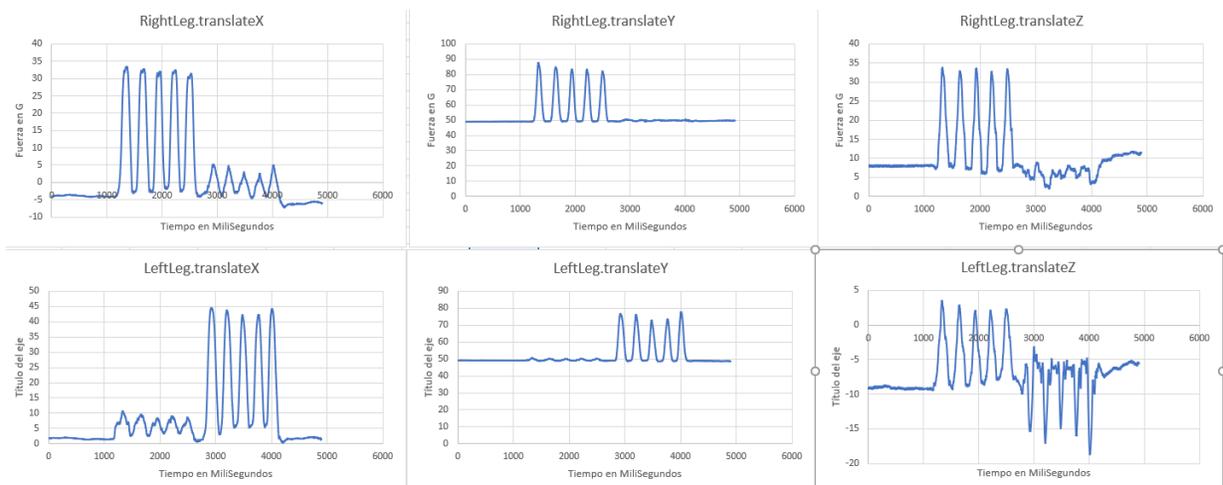


Figura 105 Pruebas con el Shadow®.



Figura 106 Grafica Fuerza vs Tiempo

En estas segundas pruebas que realizamos, se puede notar el cambio en la forma de la gráfica, aunque esta vez se nota un pequeño salto dentro de la gráfica y este salto pudo haber sido provocado por alguna perturbación del exterior, en este caso pudo haber sido por la interferencia entre los sensores, por lo que fue necesario hacer una tercera prueba.

Además de que aquí se empezó a notar el comportamiento ya un poco similar con el comercial, la diferencia es que en el dispositivo que se creó, no se tienen las mismas características de sensores, lo que pudiera provocar un cambio en los registros, como en este caso se genera un mayor ruido y además de que no son tan exactos estos sensores de uso comercial y económicos.

Decidimos hacer unas pruebas finales, ahora decidimos establecer metas con distancias, esto para ver cómo se comportaban cuando se limitaba la distancia y que era lo que sucedía, esto con el fin de poner limitantes y evitar que los ejercicios variaran por no limitar el área de trabajo.

Al decidir hacer la otra prueba en la que caracterizáramos los dos sistemas a algún limite antes establecido, esto para ver cuáles son las semejanzas y las diferencias de los dos dispositivos en un área de trabajo antes establecido.

En este caso se deberían ver 3 picos, esto como una suposición, porque al ser una prueba repetitiva en un límite antes establecido, ya que se le pidió al participante que realizara el movimiento 3 veces seguidas, esto para ver como reaccionaban los instrumentos en tareas continuas. Y con lo aprendido a lo largo de ingeniería se está realizando un trabajo continuo y al verse reflejado en las gráficas deberían de notarse ondas con características similares, esto podría verse reflejado ya sea en su frecuencia, tiempo, periodo, entre algunas otras características. Debido a estos principios, se supondría que su comportamiento debería verse similar en los dos dispositivos.



Figura 107 Caracterización de la prueba con distancias.

Se trató de caracterizar la prueba a la distancia marcada, esto para tener una referencia y ver como se comportaban los sensores cuando se hacían las pruebas a una distancia antes establecida para que podamos tener una validez más certera entre los diferentes dispositivos, ayudando a demostrar que tan cercana y correcta pueden ser los sensores entre sí y tratar de demostrar la fidelidad entre estos.

En la siguiente prueba observamos como se relacionan los movimientos conforme a las medidas que vamos incrementando, se nota la relación del incremento con las gráficas como se muestra en la siguiente figura.

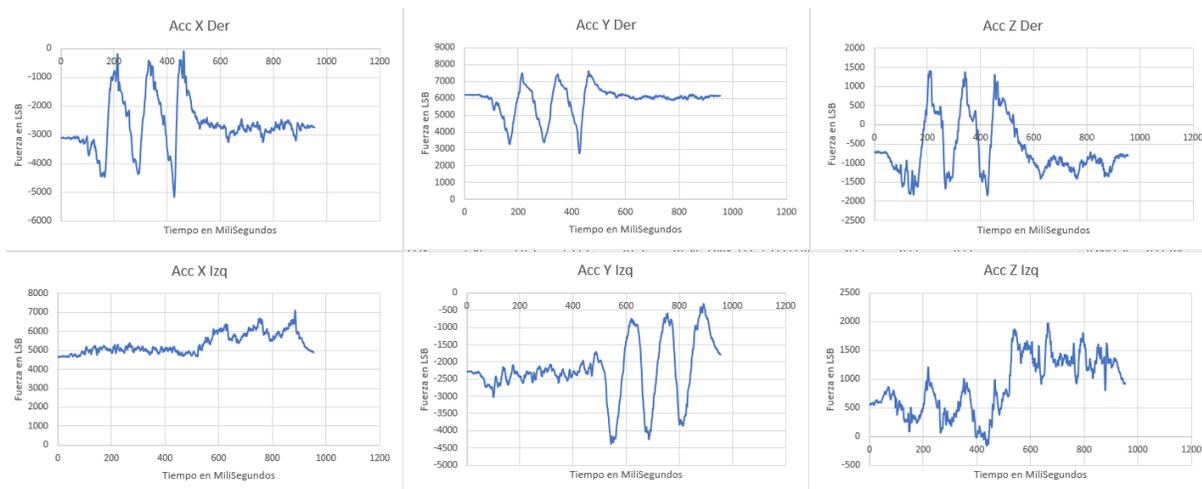


Figura 108 Pruebas a 20 cm con el dispositivo creado sin la conversión de los datos sin la conversión de los LSB.

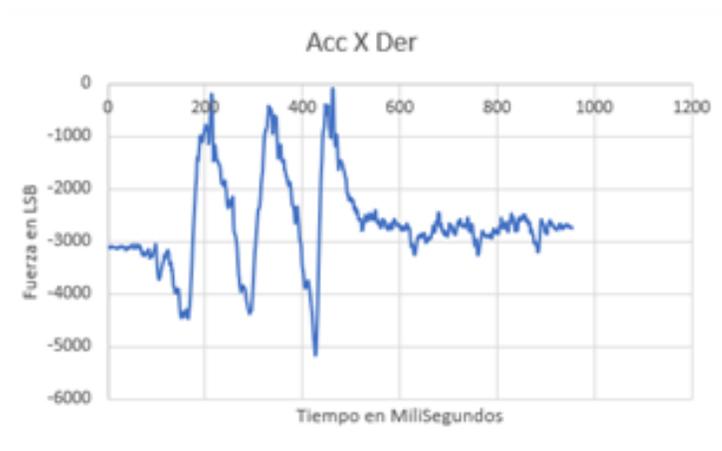


Figura 109 Muestra de una gráfica antes de la conversión de los LSB.

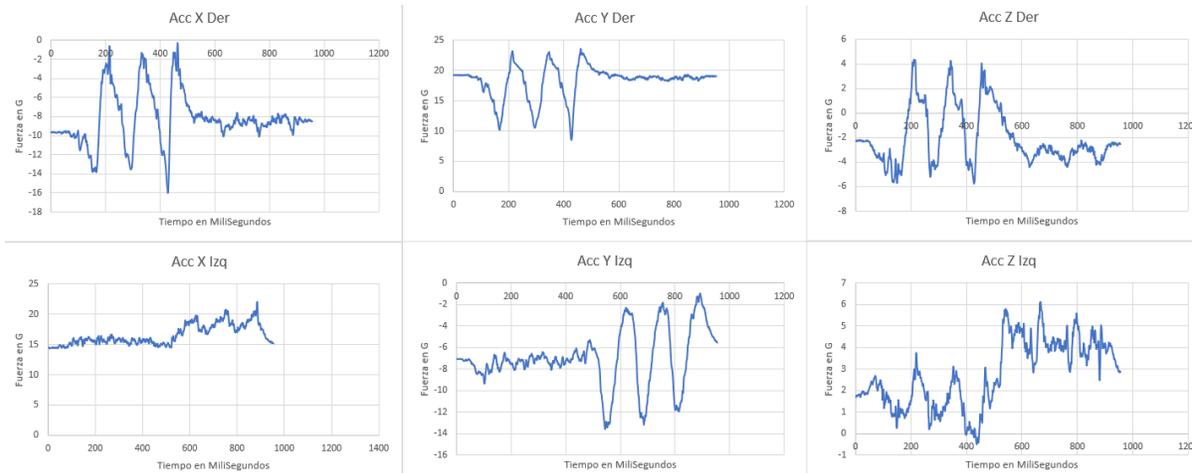


Figura 110 Gráficas del dispositivo con la conversión en G.



Figura 111 Muestra de las gráficas con las conversiones en G.

Como podemos ver la relación es muy parecida en los 2 casos, lo único que cambia es la amplitud de las dos gráficas, en el primero, la gráfica lo muestra la variación en LSB y el segundo caso ya con los datos convertidos en G's. El ruido se genera debido a que el sensor está pegado a los otros sensores, lo que puede provocar una magnetización, creando este ruido entre sensores.

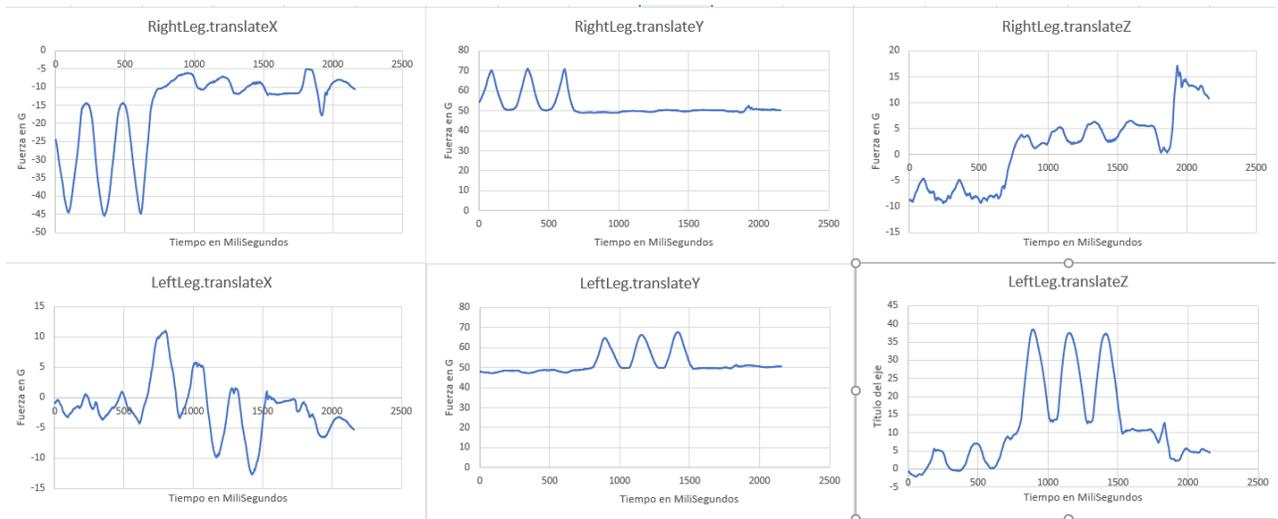


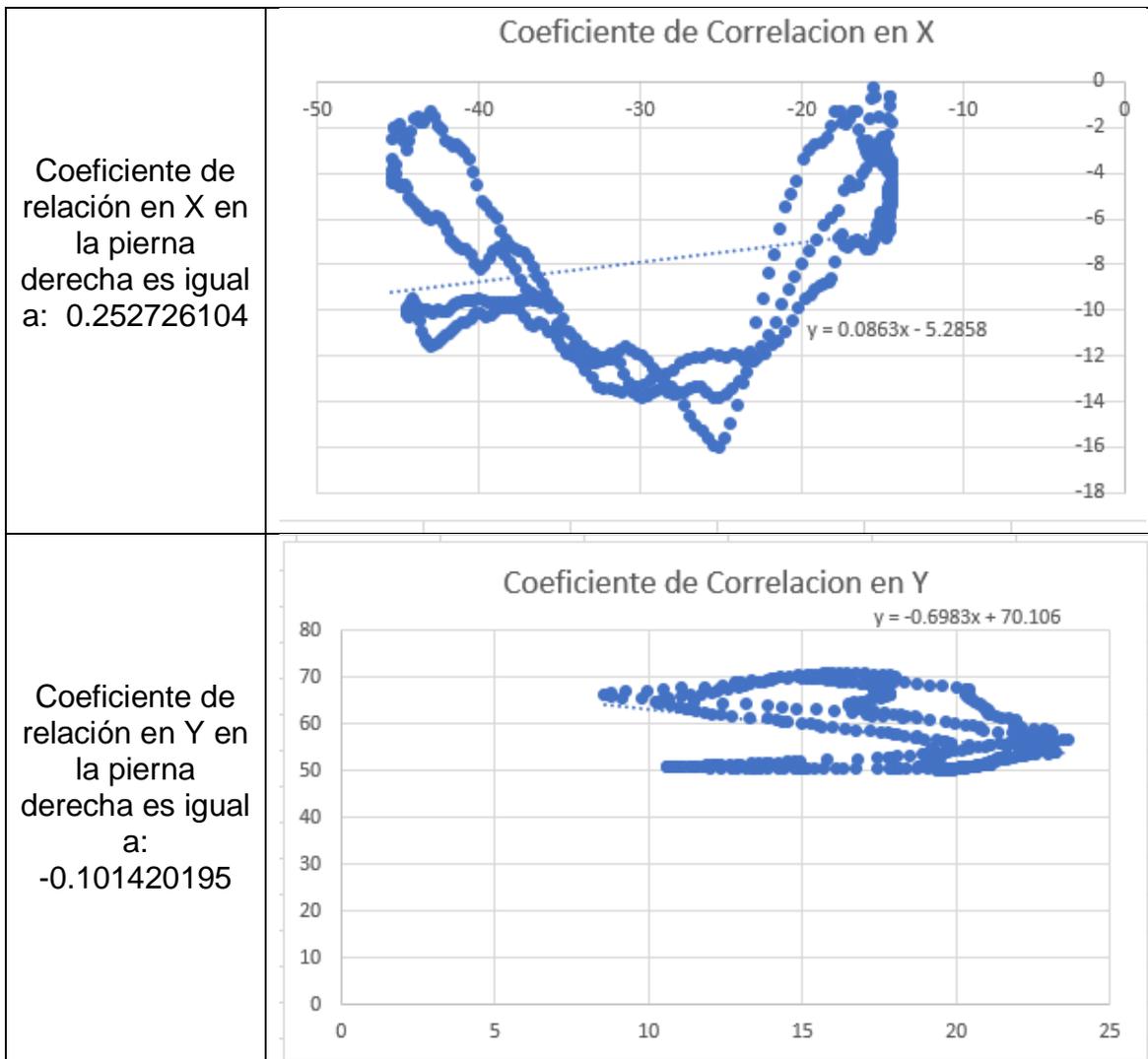
Figura 112 Pruebas a 20 cm con el dispositivo comercial

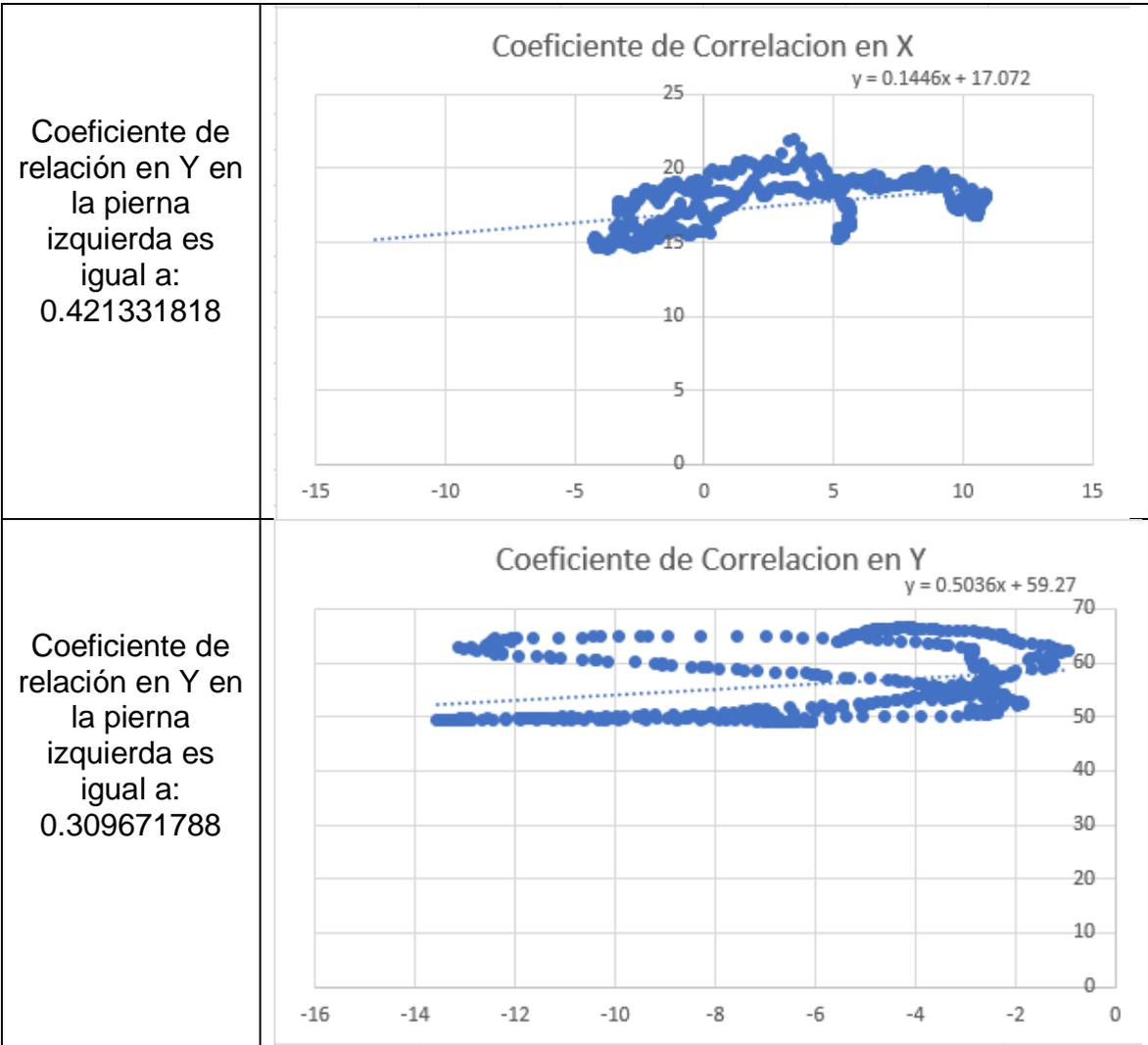


Figura 113 Muestra de las gráficas en el dispositivo comercial en G.

Como se puede observar el comportamiento es similar, los únicos cambios significantes son los del ruido y la colocación del punto inicial, todo esto depende de cómo se haya tomado en cuenta el punto inicial de cada dispositivo, al igual que estos varían en cada prueba, gracias a esto se puede ver que el comportamiento es similar en los dos dispositivos y como se había supuesto anteriormente, se muestran los 3 picos de las tres repeticiones que se habían previsto al inicio de la prueba y esto nos puede ayudar a tener una medición confiable para realizar las prueba continuas teniendo en cuenta que las mediciones que realicemos en un futuro pueden ser más confiables para la evaluación.

Esto se puede comprobar con una correlación en dos direcciones, en las coordenadas "X" y en las coordenadas "Y" entre los dispositivos, el comercial y el que se creó.





Lo que se puede observar en estas correlaciones es que se tiene una similitud, aunque no sea perfecta, esta correlación es comparada con las gráficas antes mostradas.

Existe una correlación entre los sensores, aunque en la pierna derecha sea menor que al de la izquierda en cada eje, debido a que el comportamiento en los sensores es diferente, puede depender ya sea del tipo del sensor, del ruido que se genere o de como se estén estableciendo cada sensor, en el caso de la pierna derecha en Y, se muestra una correlación negativa, esto no quiere decir que sea malo, es decir las dos variables se correlacionan en sentido inverso. Lo que significa que a valores altos de un sensor corresponde a un valor bajo del otro sensor y viceversa.

Trabajo a futuro.

Al tener una amplia área de trabajo en conjunto la ingeniería y la medicina, con ayuda de este proyecto, se pueden alcanzar nuevas metas, entre ellos nuevas formas de evaluar a los alumnos en diferentes actividades prácticas dentro de la formación de los alumnos de medicina, pero para poder complementar este trabajo, se puede desarrollar de manera más completa un sistema, incluyendo un análisis de imagen, ya sea de como el medico está realizando las posturas o movimientos adecuados a lo largo de cierto periodo de tiempo, en este interesa saber cuál es la postura correcta en la que el medico puede durar el mayor tiempo de pie durante una cirugía, esto para evitar complicaciones en un futuro y reducir la fatiga del médico, ayudando a un mejor rendimiento en la cirugía.

Un complemento más sencillo en vez de usar las cámaras, se pueden usar más sensores inerciales, esto para mantener la postura a través de los ángulos/segundos que el médico o practicante esto ayudaría para saber cada cuando un médico está en una posición incorrecta y como se está complementando con la imagen, podríamos saber cada cuanto tiempo una persona está cometiendo algún error o la razón del porque lo ha cometido, esto para tener un ambiente más controlado dentro del área de la cirugía, tanto para apoyar a los alumnos con un mejor rendimiento como a los profesores saber cómo están trabajando sus alumnos sin necesidad de estar todo el tiempo sobre el estudiante, y así solo atender casos que necesiten bastante ayuda.

Al realizar el análisis con las pruebas de correlación se observo que los sensores tenían una baja relación entre el comercial y el usado en nuestro dispositivo, esto es debido a que como son sensores económicos, se corre el riesgo de que se dañen más rápidamente y que sean inducidos a mayores alteraciones del sistema, por eso como trabajo a futuro, una recomendación es tener sensores similares a los usados por el dispositivo o uno de mejor calidad.

Conclusiones

Conforme se iba avanzando en los experimentos, se analizaron las necesidades de la cuales se había carecido a lo largo del proyecto, esto con el fin de saber que era lo que era de interés para evaluar, ya sea como un apoyo académico o para el progreso del estudiante dentro de la práctica. Clarificando así las dudas que se tuvieron a lo largo del proyecto y sobre que estábamos evaluando, al mismo tiempo descubrimos que dentro de este mismo proyecto podíamos incluir otras variables que ayudaban a discriminar la diferencia de los participantes y su categoría dentro del área del experto, intermedio y aprendiz.

El uso de sistemas como el desarrollado, ofrece una nueva forma de cuantificar el desempeño de los futuros alumnos en un aspecto práctico de la medicina que se evalúa tradicionalmente de una manera cualitativa y dependiendo de la percepción de quien realiza esta actividad. Siendo así un nuevo recurso que podría aplicarse, ya sea dentro de escuelas u hospitales.

El dispositivo ayudaría al área de la cirugía para saber quiénes cuentan con las destrezas básicas para realizar una actividad específica evitando errores médicos que surgen comúnmente debido a la baja inspección que se imparte desde las escuelas en estas tareas de cirugía. Asimismo, se apoya a los profesores para puedan enfocarse en las personas que tengan problemas con el desenvolvimiento del ejercicio o se les complique realizar la actividad de sutura, al igual que motivar al alumno a que mejore sus destrezas.

La correlación nos ayudaría a entender mejor como se comportan los sensores, aunque no fueran muy parecidas, no daban a entender que el comportamiento que se muestra puede ser similar, esto ayudando a que en un futuro sea más exacto, adaptando con nuevos sensores para lograr una mejor exactitud.

Referencias bibliográficas

- Inc. InvenSense. (2011). *MPU-6000 and MPU-6050 Register Map and Descriptions*. 1(408), 1–47.
- Luttmann, A., Jäger, M., & Sökeland, J. (2009). Ergonomic assessment of the posture of surgeons performing endoscopic transurethral resections in urology. *Journal of Occupational Medicine and Toxicology*, 4(1), 1–11. <https://doi.org/10.1186/1745-6673-4-26>
- Beard, J. D. (2008). Assessment of surgical skills of trainees in the UK. *Annals of the Royal College of Surgeons of England*, 90(4), 282–285. <https://doi.org/10.1308/003588408X286017>
- Yujin Jung, Donghoon Kang, Jinwook Kim, Upper Body Motion Tracking With Inertial Sensors, Proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics December 14-18, 2010, Tianjin, China.
- Qilong Yuan, I-Ming Chen, Human velocity and dynamic behavior tracking method for inertial capture system, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 597627, Singapore.

Anexos

Programa Arduino MPU6050

//Evaluación

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files

```

// for both classes must be in the include path of your project
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense
evaluation board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu2(0x69);
//MPU6050 mpu(0x69); // <-- use for AD0 high

/*
=====
NOTE: In addition to connection 3.3v, GND, SDA, and SCL, this sketch
depends on the MPU-6050's INT pin being connected to the Arduino's
external interrupt #0 pin. On the Arduino Uno and Mega 2560, this is
digital I/O pin 2.
*
=====
*/

/*
=====
NOTE: Arduino v1.0.1 with the Leonardo board generates a compile error
when using Serial.write(buf, len). The Teapot output uses this method.
The solution requires a modification to the Arduino USBAPI.h file,
which
is fortunately simple, but annoying. This will be fixed in the next
IDE
release. For more info, see these links:

http://arduino.cc/forum/index.php/topic,109987.0.html
http://code.google.com/p/arduino/issues/detail?id=958
*
=====
*/

// uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual
// quaternion components in a [w, x, y, z] format (not best for parsing
// on a remote host such as Processing or something though)
//#define OUTPUT_READABLE_QUATERNION

// uncomment "OUTPUT_READABLE_EULER" if you want to see Euler angles

```

```

// (in degrees) calculated from the quaternions coming from the FIFO.
// Note that Euler angles suffer from gimbal lock (for more info, see
// http://en.wikipedia.org/wiki/Gimbal\_lock)
// #define OUTPUT_READABLE_EULER

// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for
// more info, see: http://en.wikipedia.org/wiki/Gimbal\_lock)
#define OUTPUT_READABLE_YAWPITCHROLL

// uncomment "OUTPUT_READABLE_REALACCEL" if you want to see acceleration
// components with gravity removed. This acceleration reference frame is
// not compensated for orientation, so +X is always +X according to the
// sensor, just without the effects of gravity. If you want acceleration
// compensated for orientation, use OUTPUT_READABLE_WORLDACCEL instead.
#define OUTPUT_READABLE_REALACCEL

// uncomment "OUTPUT_READABLE_WORLDACCEL" if you want to see acceleration
// components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no
magnetometer
// is present in this case). Could be quite handy in some cases.
// #define OUTPUT_READABLE_WORLDACCEL

// uncomment "OUTPUT_TEAPOT" if you want output that matches the
// format used for the InvenSense teapot demo
// #define OUTPUT_TEAPOT

unsigned long previousMillis=0;
unsigned long tiempo = 0;
float t;

#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorInt16 aa; // [x, y, z] accel sensor measurements
VectorInt16 aaReal; // [x, y, z] gravity-free accel sensor
measurements
VectorInt16 aaWorld; // [x, y, z] world-frame accel sensor
measurements
VectorFloat gravity; // [x, y, z] gravity vector
float euler[3]; // [psi, theta, phi] Euler angle container

```

```

float ypr[3];          // [yaw, pitch, roll]   yaw/pitch/roll container
and gravity vector

// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00,
'\r', '\n' };

// =====
// ===                INTERRUPT DETECTION ROUTINE                ===
// =====

volatile bool mpuInterrupt = false;    // indicates whether MPU
interrupt pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}

// =====
// ===                INITIAL SETUP                ===
// =====

void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot Demo output, but
it's
    // really up to you depending on your project)
    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration, others continue
immediately

    // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3v or
Arduinio
    // Pro Mini running at 3.3v, cannot handle this baud rate reliably
due to
    // the baud timing being too misaligned with processor ticks. You
must use
    // 38400 or slower in these cases, or use some kind of external
separate
    // crystal solution for the UART timer.

    // initialize device
    //Serial.println(F("Initializing I2C devices..."));
    // mpu.initialize();
    //

```

```

// // verify connection
// //Serial.println(F("Testing device connections..."));
// //Serial.println(mpu.testConnection() ? F("MPU6050 connection
successful") : F("MPU6050 connection failed"));
//
// // wait for ready
// //Serial.println(F("\nSend any character to begin DMP programming
and demo: "));
// while (Serial.available() && Serial.read()); // empty buffer
// while (!Serial.available()); // wait for data
// while (Serial.available() && Serial.read()); // empty buffer again

// load and configure the DMP
//Serial.println(F("Initializing DMP..."));
devStatus = mpu.dmpInitialize();

// supply your own gyro offsets here, scaled for min
sensitivity
//Your offsets: 3029 -630 1185 21 -31 33

//Introduce aqui los datos obtenidos

mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76);
mpu.setZGyroOffset(-85);
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
mpu.setXAccelOffset(1588);
mpu.setYAccelOffset(1488);

// make sure it worked (returns 0 if so)
if (devStatus == 0) {
  // turn on the DMP, now that it's ready
  //Serial.println(F("Enabling DMP..."));
  mpu.setDMPEnabled(true);

  // enable Arduino interrupt detection
  //Serial.println(F("Enabling interrupt detection (Arduino
external interrupt 0)..."));
  attachInterrupt(0, dmpDataReady, RISING);
  mpuIntStatus = mpu.getIntStatus();

  // set our DMP Ready flag so the main loop() function knows it's
okay to use it
  //Serial.println(F("DMP ready! Waiting for first interrupt..."));
  dmpReady = true;

  // get expected DMP packet size for later comparison
  packetSize = mpu.dmpGetFIFOPacketSize();
} else {
  // ERROR!
  // 1 = initial memory load failed
  // 2 = DMP configuration updates failed
  // (if it's going to break, usually the code will be 1)
  Serial.print(F("DMP Initialization failed (code "));
  Serial.print(devStatus);

```

```

        Serial.println(F(""));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
    previousMillis = millis();
}

// =====
// ===                               MAIN PROGRAM LOOP                               ===
// =====

void loop() {

    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize) {
        // other program behavior stuff here
        // .
        // .
        // .
        // if you are really paranoid you can frequently test in between
other    // stuff to see if mpuInterrupt is true, and if so, "break;" from
the    // while() loop to immediately process the MPU data
        // .
        // .
        // .
    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is
too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

        // otherwise, check for DMP data ready interrupt (this should happen
frequently)
    } else if (mpuIntStatus & 0x02) {

        // wait for correct available data length, should be a VERY short
wait    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

```

```

// read a packet from FIFO
mpu.getFIFOBytes(fifoBuffer, packetSize);

// track FIFO count here in case there is > 1 packet available
// (this lets us immediately read more without waiting for an
interrupt)
fifoCount -= packetSize;

//
// #ifdef OUTPUT_READABLE_QUATERNION
// // display quaternion values in easy matrix form: w x y z
// mpu.dmpGetQuaternion(&q, fifoBuffer);
// Serial.print("quat\t");
// Serial.print(q.w);
// Serial.print("\t");
// Serial.print(q.x);
// Serial.print("\t");
// Serial.print(q.y);
// Serial.print("\t");
// Serial.println(q.z);
// #endif
//
// #ifdef OUTPUT_READABLE_EULER
// // display Euler angles in degrees
// mpu.dmpGetQuaternion(&q, fifoBuffer);
// mpu.dmpGetEuler(euler, &q);
// //Serial.print("euler\t");
// Serial.print(euler[0] * 180/M_PI);
// Serial.print("\t");
// Serial.print(euler[1] * 180/M_PI);
// Serial.print("\t");
// Serial.print(euler[2] * 180/M_PI);
// Serial.print("\t");
// #endif

#ifdef OUTPUT_READABLE_REALACCEL
// display real acceleration, adjusted to remove gravity
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetAccel(&aa, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
//Serial.print("areal\t");
Serial.print(aaReal.x * (9.81/16384.0));
Serial.print(",");
Serial.print(aaReal.y * (9.81/16384.0));
Serial.print(",");
Serial.print(aaReal.z * (9.81/16384.0));
Serial.print(",");
#endif

#ifdef OUTPUT_READABLE_YAWPITCHROLL
// display Euler angles in degrees
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
//Serial.print("ypr\t");

```

```

        Serial.print(ypr[0] * 180/M_PI);
        Serial.print(",");
        Serial.print(ypr[1] * 180/M_PI);
        Serial.print(",");
        Serial.print(ypr[2] * 180/M_PI);
        Serial.print(",");
    #endif

//      #ifdef OUTPUT_READABLE_WORLDACCEL
//          // display initial world-frame acceleration, adjusted to
remove gravity
//          // and rotated based on known orientation from quaternion
//          mpu.dmpGetQuaternion(&q, fifoBuffer);
//          mpu.dmpGetAccel(&aa, fifoBuffer);
//          mpu.dmpGetGravity(&gravity, &q);
//          mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
//          mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
//          Serial.print("aWorld\t");
//          Serial.print(aaWorld.x);
//          Serial.print("\t");
//          Serial.print(aaWorld.y);
//          Serial.print("\t");
//          Serial.println(aaWorld.z);
//      #endif
//
//      #ifdef OUTPUT_TEAPOT
//          // display quaternion values in InvenSense Teapot demo
format:
//          teapotPacket[2] = fifoBuffer[0];
//          teapotPacket[3] = fifoBuffer[1];
//          teapotPacket[4] = fifoBuffer[4];
//          teapotPacket[5] = fifoBuffer[5];
//          teapotPacket[6] = fifoBuffer[8];
//          teapotPacket[7] = fifoBuffer[9];
//          teapotPacket[8] = fifoBuffer[12];
//          teapotPacket[9] = fifoBuffer[13];
//          Serial.write(teapotPacket, 14);
//          teapotPacket[11]++; // packetCount, loops at 0xFF on
purpose
//          #endif

        Serial.println(t/1000);
        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);
        tiempo = millis();
        t = tiempo - previousMillis;
    }
}

```

Programa en Processing

```

//Se importan la librerías necesarias
import processing.serial.*; //Librería para comunicación Serial
import processing.sound.*; //Librería para reproducción de sonidos

```

```

import g4p_controls.*;           //Librería para interfaz gráfica
import controlP5.*;             //Librería para interfaz gráfica

SoundFile soundfile;           //Se crea
un elemento de la clase SoundFile

Serial myPortD;                 //Se crea
un elememnto de la clase Serial

Serial myPortI;                 //Se crea
un elemento de la clase Serial

ControlP5 cp5;

GButton btnStart, btnStop;     //Botones
de inicio y fin

GTextField name;               //Label
para nombre de usuario

PImage unam, facmed;

int lf = 10;                    //Caracter de salto de línea
float xPos=0.0001;              //Posición actual de la
gráfica////////////////////////////////////
/Checar
int cont=0;

float[] val1= new float[7];
float[] val2= new float[7];
float[] accxD= new float[0];
float[] accyD= new float[0];
float[] acczD= new float[0];
float[] yawD= new float[0];
float[] pitchD= new float[0];
float[] rollD= new float[0];
float[] accxI= new float[0];
float[] accyI= new float[0];
float[] acczI= new float[0];
float[] yawI= new float[0];
float[] pitchI= new float[0];
float[] rollI= new float[0];
float[] time= new float[0];

float[] yawDGraph = {0,0};
float[] pitchDGraph = {0,0};
float[] rollDGraph = {0,0};
float[] yawIGraph = {0,0};
float[] pitchIGraph = {0,0};
float[] rollIGraph = {0,0};
float[] timeGraph = new float[0];

boolean start=false;
boolean character=false;

float inByte1;
float inByte2;
float inByte3;
float inByte4;
float inByte5;
float inByte6;

```

```

void setup() {
  cont=0;
  window();
  //////////////////////////////////CONFIGURACIÓN DE LA PANTALLA////////////////////////////////////
  //////////////////////////////////
  size(1000,700);
  surface.setTitle("Evaluación");
  frameRate(50);
  //soundfile = new SoundFile(this, "Click_2.mp3");
  //////////////////////////////////INICIACIÓN DE LA COMUNICACIÓN SERIAL////////////////////////////////////
  //////////////////////////////////
  printArray(Serial.list());
  myPortD=new Serial(this,"COM3",115200);
  myPortD.clear();
  myPortD.bufferUntil(lf);
  myPortI=new Serial(this,"COM4",115200);
  myPortI.clear();
  myPortI.bufferUntil(lf);

  //////////////////////////////////CONFIGURACIÓN DE LA INTERFAZ GRÁFICA////////////////////////////////////
  //////////////////////////////////
  btnStart = new GButton(this, 10, 520, 100, 20, "Inicio");
  btnStop = new GButton(this, 120, 520, 100, 20, "Captura");
  name = new GTextField(this, 230, 520, 250, 20);
  name.setPromptText("Ingrese Nombre");

  strokeWeight(3);
  stroke(255,255,0);    line(20, 600, 35, 600);
  stroke(254,0,0);     line(20, 620, 35, 620);
  stroke(0,247,0);     line(20, 640, 35, 640);
  stroke(222,76,138);  line(150, 600, 165, 600);
  stroke(27,85,131);   line(150, 620,165, 620);
  stroke(76,47,39);    line(150, 640, 165, 640);

  fill(0, 0, 0);
  text("Yaw Derecho", 45, 610);
  text("Pitch Derecho",45, 630);
  text("Roll Derecho", 45, 650);
  text("Yaw Izquierdo", 175, 610);
  text("Pitch Izquiero", 175, 630);
  text("Roll Izquierdo", 175, 650);

  stroke(0,0,0);
  strokeWeight(0.3);
  line(290,20,290,500);
}

void draw() {

  //image(facmed,610,520,180,180);
  //image(unam,800,520,180,180);

  if(start==true) {
    if(caracter==false) {
      myPortD.write(byte('a'));
    }
  }
}

```

```

myPortI.write(byte('a'));
println("Bien");
character=true;
}

inByte1 = map(val1[3],-180, 180, 20, 480); //YAWD
inByte2 = map(val1[4],-90, 90, 20, 480); //PITCHD
inByte3 = map(val1[5],-90, 90, 20, 480); //ROLLD
inByte4 = map(val2[3],-180, 180, 20, 480); //YAWI
inByte5 = map(val2[4],-90, 90, 20, 480); //PITCHI
inByte6 = map(val2[5],-90, 90, 20, 480); //ROLLI

yawDGraph=append(yawDGraph,inByte1);
pitchDGraph=append(pitchDGraph,inByte2);
rollDGraph=append(rollDGraph,inByte3);
yawIGraph=append(yawIGraph,inByte4);
pitchIGraph=append(pitchIGraph,inByte5);
rollIGraph=append(rollIGraph,inByte6);
timeGraph =append(timeGraph,val2[6]);

float yDP=yawDGraph[cont];
float yD= yawDGraph[cont+1];
float pDP= pitchDGraph[cont];
float pD= pitchDGraph[cont+1];
float rDP= rollDGraph[cont];
float rD= rollDGraph[cont+1];

float yIP=yawIGraph[cont];
float yI= yawIGraph[cont+1];
float pIP= pitchIGraph[cont];
float pI= pitchIGraph[cont+1];
float rIP= rollIGraph[cont];
float rI= rollIGraph[cont+1];

float time1=timeGraph[cont];

strokeWeight(1.5);
strokeJoin(ROUND);
stroke(255,255,0);
line(xPos, -1*(yDP-480) , xPos+0.5, -1*((yD)-480));

stroke(254,0,0);
line(xPos, -1*(pDP-480) , xPos+0.5, -1*((pD)-480));

stroke(0,247,0);
line(xPos, -1*(rDP-480) , xPos+0.5, -1*((rD)-480));
////////////////////////////////////
////////////////////////////////////
stroke(222,76,138);
line(xPos, -1*(yIP-480) , xPos+0.5, -1*((yI)-480));

stroke(27,85,131);
line(xPos, -1*(pIP-480) , xPos+0.5, -1*((pI)-480));

stroke(76,47,39);

```

```

    line(xPos, -1*(rIP-480) , xPos+0.5, -1*((rI)-480));

    cont=cont+1;

    if(time1>9.99){
    accxD=append(accxD, val1[0]);
    accyD=append(accyD, val1[1]);
    acczD=append(accyD, val1[2]);
    yawD=append(yawD, val1[3]);
    pitchD=append(pitchD, val1[4]);
    rollD=append(rollD, val1[5]);
    accxI=append(accxI, val2[0]);
    accyI=append(accyI, val2[1]);
    acczI=append(accyI, val2[2]);
    yawI=append(yawI, val2[3]);
    pitchI=append(pitchI, val2[4]);
    rollI=append(rollI, val2[5]);
    time=append(time, val1[6]);
    }

    if (xPos >= 980) {
        window();
        xPos = 26;
        strokeWeight(3);
        stroke(255,255,0);    line(20, 600, 35, 600);
        stroke(254,0,0);     line(20, 620, 35, 620);
        stroke(0,247,0);     line(20, 640, 35, 640);
        stroke(222,76,138);  line(150, 600, 165, 600);
        stroke(27,85,131);   line(150, 620,165, 620);
        stroke(76,47,39);    line(150, 640, 165, 640);

        fill(0, 0, 0);
        text("Yaw Derecho", 45, 610);
        text("Pitch Derecho",45, 630);
        text("Roll Derecho", 45, 650);
        text("Yaw Izquierdo", 175, 610);
        text("Pitch Izquiero", 175, 630);
        text("Roll Izquierdo", 175, 650);
    }
    else {
        xPos=xPos+0.5;
    }

    if(time1==9.99); //soundfile.play();//

} //Fin de bucle while

} ///Fin de void Draw

//////////RUTINA PARA ACTUALIZAR LA GRÁFICA//////////
//////////

void window(){

```

```

unam=loadImage("unam2.png");
facmed=loadImage("facmed.png");

//////////////////////////////////DISPLAY PARA LA GRÁFICA//////////////////////////////////
//////////////////////////////////
background(52, 152, 219);
strokeWeight(1);
stroke(0,0,0);
fill(255,255,255);
rectMode(CORNERS);
rect(26,20,980,500);
//////////////////////////////////GRID PARA LA GRÁFICA//////////////////////////////////
//////////////////////////////////

for (int i = 0; i <= 5; i++) {
  stroke(0,0,0);
  strokeWeight(0.3);
  line(26, 20+(96*i), 980, 20+(96*i));
  fill(0,0,0);
  text(180-(60*i),0,20+(96*i));
}

//////////////////////////////////RUTINA PARA GUARDAR LOS DATOS EN UN ARCHIVO TXT//////////////////////////////////
//////////////////////////////////
void finished() {
  myPortD.stop();
  myPortI.stop();
  myPortD.clear();
  myPortI.clear();
  String[] data = new String[time.length];
  int y = year();
  int m = month();
  int d = day();
  String title = name.getText();
  String filename=title+d+m+y+".txt";

  //for (int j=0; j<time.length;j++){
  //  time[j]=time[j]-10;
  //}

  for (int i = 0; i < time.length; i++) {
    data[i] = accxD[i] + "\t" + accyD[i]+ "\t" +acczD[i]+ "\t"
+ yawD[i]+ "\t" +pitchD[i]+ "\t" +rollD[i]+ "\t" + accxI[i] + "\t" +
accyI[i]+ "\t" +acczI[i]+ "\t" +yawI[i]+ "\t" +pitchI[i]+ "\t"
+rollI[i]+ "\t"+time[i];
  }

  saveStrings(filename, data);
  exit();
}

//////////////////////////////////RUTINA PARA LOS BOTONES DE LA INTERFAZ//////////////////////////////////
//////////////////////////////////

```

```

void handleButtonEvents(GButton button, GEvent event) {
    if (button == btnStart && event == GEvent.CLICKED){
        start=true;
    }

    if (button == btnStop && event == GEvent.CLICKED){
        finished();
    }
}

//////////////////////////////////RUTINA PARA LA COMUNICACIÓN SERIAL//////////////////////////////////
//////////////////////////////////

void serialEvent (Serial myPort) {

    if(myPort==myPortD){
        String inStringD =myPortD.readStringUntil(lf);
        if (inStringD !=null){
            val1=float(split(inStringD,","));
        }
    }

    if(myPort==myPortI){
        String inStringI =myPortI.readStringUntil(lf);
        if (inStringI!=null){
            val2=float(split(inStringI,","));
        }
    }
}
}

```