Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

Tesis:

"TÉCNICAS DE COMPUTACIÓN GRANULAR APLICANDO SENSADO"

Que presenta el alumno:
Jose Isabel Garcia Rocha

Como opción de titulación en la carrera de:

Ingeniería en Computación

Noviembre de 2019 Tijuana, B.C.

Índice

Índice de tablas

Índice de figuras

1.Introducción

1.1 Problemática

Existe la necesidad en la sociedad de obtener mayor información sobre el movimiento corporal para fines de análisis médicos con mayor rapidez y menor costo, por lo se propone crear un dispositivo que proporcione dicha información, que cuente con un costo bajo y con elementos de fácil adquisición en tiendas comerciales.

1.2 Hipótesis

Es posible construir una red de sensores corporales de bajo costo para uso en la adquisición de datos de movimiento, pudiéndose usar estos en el análisis de padecimientos motores y hacer una recomendación temprana.

1.3 Objetivos

1. Diseñar una red de sensores corporales para la adquisición de datos en la parte baja del cuerpo.
2. Construir una red de sensores corporales para la adquisición de datos en la parte baja del cuerpo.

1.4 Metas

1. Investigar el estado del arte en sensores corporales para adquisición de datos del movimiento corporal.
2. Diseñar múltiples prototipos de redes de sensores corporales a partir de componentes de bajo costo
3. Seleccionar un prototipo y construirlo
4. Hacer pruebas y validar la calidad de la adquisición de datos
5. Redactar tesis

2. Marco teórico

2.1 Estado del arte

El desarrollo de un proyecto se inicia principalmente en base a ciertas necesidades de la sociedad que rodea al desarrollador, por lo que no es de extrañar que existan desarrollos ya terminados similares al que se desea crear, por ello se exponen aquí los datos recopilados de los proyectos que manejan planteamientos, necesidades de la comunidad o tecnologías implementadas similares al expuesto en esta tesis.

Por parte de la Universidad Politécnica Salesiana Sede Quito se encuentra una tesis la cual como título tiene “Sistema de captura y evaluación del movimiento humano”, la cual fue desarrollada por Guzmán Balcázar, Mayra Alejandra, Toscano Pozo y Washington Hernán, dicha tesis se toma en cuenta pues se utiliza un sensor para captar el movimiento de una persona y posteriormente determinar su condición física en un reporte médico, para lo cual se está utilizando un sensor el cual en este caso es el kinect de microsoft para determinar de acuerdo a ciertas variables que aporta el mismo software del dispositivo el estado del participante.

Por parte de la Universidad Politécnica de Catalunya se encuentra una tesis doctoral la cual como título tiene “Contribución al análisis del movimiento humano aplicado a la identificación de posturas y bloqueos de la marcha en pacientes con parkinson”, la cual fue desarrollada por Rodríguez Martín Daniel, dicha tesis doctoral se toma en cuenta pues se desarrolla un dispositivo capaz de sensar el movimiento de los pacientes con parkinson el cual cuenta con sensores inerciales, es de un tamaño reducido permitiendo la portabilidad y cuenta con una gran autonomía, siendo estos junto con su finalidad sumamente parecidos al tema que se aborda en esta tesis, aunque al ser esta una tesis doctoral se abarca más al desarrollar algoritmos capaces de detectar las posturas del paciente con las cuales al finalizar la prueba se puede saber si la persona tiene o puede llegar a tener parkinson en un corto tiempo.

Por parte de la Universidad Zaragoza se encuentra un trabajo fin de máster la cual como título tiene “Dispositivo de captura de movimiento basado en sensores inerciales con comunicación inalámbrica”, la cual fue desarrollada por , dicho trabajo fin de máster se toma en cuenta pues se desarrolla un dispositivo con la misma finalidad que en esta tesis, pues se busca capturar el movimiento de una persona de forma móvil, autónoma, con buena autonomía y en un dispositivo embebido, se puede encontrar una diferencia aquí pues se utiliza la comunicación inalámbrica para enviar la información y en el caso de este trabajo se realiza la comunicación con ayuda del protocolo I2C el cual es cableado. No tiene una finalidad final en ningún área sino que busca ser un dispositivo general que se

pueda usar en diversas áreas como deportes, medicina, desarrollo de videojuegos o cualquier área donde se requiera capturar el movimiento real de una persona. Se toma este trabajo pues es básicamente la misma elaboración pero con distintos dispositivos y con otra perspectiva.

Por parte de la Universidad Pedagogica y Tecnologia de Columbia se encuentra una la cual como título tiene “Arquitectura de un sistema de medición de bio parámetros integrando señales inerciales magnéticas y electromiográficas”, la cual fue desarrollada por Mauro Callejas Cuervo, Manuel A. Vélez Guerrero y Wilson Javier Pérez Holguín, dicho trabajo de investigación de un PhD en Energía y Control de Procesos, Magister de Ingeniería, PhD en Ingeniería se toma en cuenta pues se desarrolla un dispositivo en el cual se cuenta con sensores inerciales (IMU) y sensores electromiográficos (EMG)(sensores que capturan las señales eléctricas) el cual solamente se concentra en el brazo y el antebrazo, por lo que se especializa en esa área y obtiene la mayor cantidad de información sobre el cómo se mueven o comportan dichas secciones del cuerpo humano, en el trabajo de investigación no se le da ningún uso pero si se comenta que se desea su uso para la fabricación de un sistema de control electrónico para una plataforma de rehabilitación por medio de un exoesqueleto robótico.

Por parte del grupo de investigación DIGITI de la universidad Distrital se encuentra un trabajo de investigación la cual como título tiene “Sistema portátil de captura de movimiento para el análisis cinemático de la marcha humana”, la cual fue desarrollada por Camargo C. Esperanza, Garzón G. Yamid y Camacho P. Victor, dicho trabajo de investigación se toma en cuenta pues se desarrolla un dispositivo el cual cuenta con sensores de movimiento (en este caso acelerómetros) con los cuales captura el movimiento de una persona, la almacena en una memoria microSD y después se pasa a una computadora para modelar y poder representar visualmente el movimiento captado para facilitar su comprensión. Existe gran similitud entre este trabajo de investigación y la presente tesis, aunque algunas diferencias se pueden notar en el tipo de sensores utilizados pues en este trabajo se utiliza solamente el acelerómetro y en la presente tesis se utilizan tres, el acelerómetro, el giroscopio y el magnetómetro, mas se puede encontrar una gran similitud en el objetivo pues se busca capturar la información sobre el movimiento de la persona para ayudar en trabajos médicos donde se busca encontrar enfermedades asociadas al movimiento del cuerpo humano al caminar y/o moverse.

Por parte de la Universidad Politécnica Madrid se encuentra un trabajo de fin de grado la cual como título tiene “Desarrollo de un método de captación de movimiento humano para el control remoto de terapias de rehabilitación robóticas”, la cual fue desarrollada por Vivas Mateos Guillermo, dicho trabajo se toma en

cuenta pues se captura el movimiento de personas sanas para poder ser utilizado en terapias a personas con algun patogeno que les afecte y su tratamiento requiera terapia, entonces el dispositivo permitirá guardar estos ejercicios que el terapeuta le hace realizar al paciente para en un futuro poder crear un robot que pueda utilizar esta información capturada con exactitud y emularla para de esta manera poder dar tratamiento a los pacientes sin la necesidad del terapeuta. El dispositivo utilizado en este trabajo para capturar el movimiento es el Technaid el cual se vende por esta compañía para uso profesional y cuenta con su propio armazón, un microprocesador y hasta 16 sensores inerciales.

2.2 Productos comerciales

Anteriormente se mencionó y como es de esperarse existen dispositivos de hardware y software que realizan una tarea similar o exactamente la misma que en la presente tesis, en este caso hablaremos de DarkFish una compañía que tiene desde 1999 y hoy en día cuenta con sedes en diversos países abarcando gran parte del mundo tecnológico, cuentan con una gran variedad de productos y se enfocan solamente en la creación de software para analizar y etiquetar videos del movimiento humano.

Figura 1 Etiquetado de video por DarkFish.

Entre sus diversos productos podemos encontrar una versión móvil la cual nos provee de una aplicación para nuestro celular inteligente y nos permite analizar el rendimiento en nuestros videos, también cuentan con versiones para la computadora que realizan lo basicamente lo mismo pero con mayor variedad de opciones y mayor rapidez, y la última opción permite analizar el video de 4 cámaras, en sí es un software que analiza los videos que se le ingresan aportando una gran cantidad de información y permitiendo agregar etiquetas a los videos para aportar esa información de una manera más simple, y como es de esperar se puede utilizar para captar los movimientos de una persona.

Otra de estas compañías es Technaid la cual tiene como producto un dispositivo que es similar al de esta tesis aunque claro con mayor presupuesto y con un precio de venta sumamente elevado, este producto es el Tech MCS el cual es completamente móvil e inalámbrico utilizando la señal Bluetooth con la cual se permite la captura de datos hasta 200 metros de distancia, utiliza el protocolo de comunicación SPI con sus sensores por lo que esto le permite contar con mayor número de sensores (hasta 16 en este caso) y aun así contar con una gran velocidad de muestreo.

Figura 2. Unidad de control Tech MCS.

Por lo tanto este producto es básicamente lo mismo a lo que se realiza en esta tesis, pero utilizando otro protocolo de comunicación y agregando un módulo Bluetooth, el costo de este producto no se menciona pero en vista de su calidad esté ha de ser sumamente elevado.

2.3 Sensores inerciales

Con el desarrollo de los sistemas microelectromecánicos MEMS, fue posible crear dispositivos pequeños que se pudieran adherir a un cuerpo y medir la aceleración y la inclinación de los mismos, estos sistemas son llamados acelerómetros y giroscopios (fig. 3), se puede agregar magnetómetros para conocer la ubicación espacial.

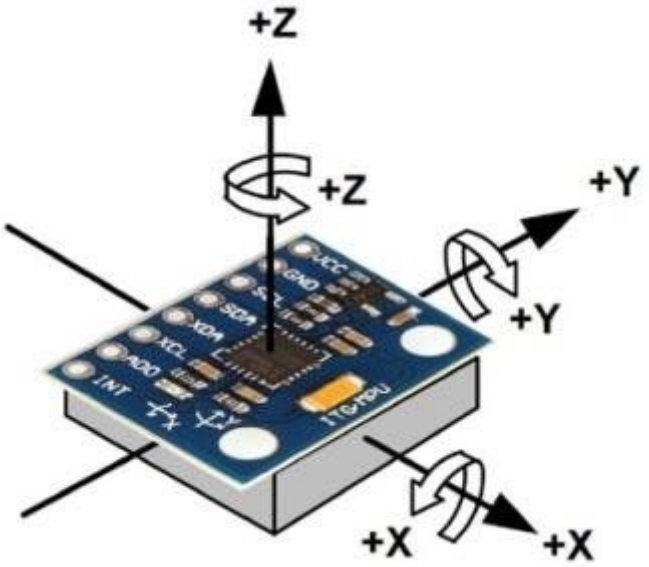

Figura 3. Giroscopio y acelerómetro con referencia de ejes y giro.

Los sensores de unidad de medición inercial (IMU) utilizan múltiples ejes de giroscopios de precisión, acelerómetros y magnetómetros los cuales al suceder una fuerza que les afecta registran una señal eléctrica.
La señal registrada se puede entender por parte de los giroscopios como velocidad angular y aceleración en caso de los acelerómetros, estos efectos se verán registrados en los aparatos de medición como una señal de voltaje la cual es proporcional y muy exacta.

El funcionamiento de un acelerómetro se basa en la segunda Ley de Newton (la fuerza es directamente proporcional a la masa y a la aceleración de un cuerpo). Un acelerómetro mide la aceleración indirectamente debido a la fuerza aplicada en alguno de sus ejes, de esta forma mide la aceleración inercial que resulta del movimiento o los movimientos repentinos y la aceleración estática, es decir, de la

gravedad. El giroscopio es un sensor que mide velocidad angular alrededor de un eje o de varios ejes y a diferencia del acelerómetro no se ve afectado por la aceleración de la gravedad, de modo quees posible complementar la información de ambos.

Existen trajes con sensores inerciales y magnetómetros (fig. 4), los cuales envían los datos que recopilan de los sensores instalados en el trajes hacía una computadora la cual procesa la información y con esta determina la posición, el eje de giro y la velocidad angular de cualquiera de los sensores instalados.

Figura 4. Traje de sensores inerciales MVN Biomech con 17 sensores inerciales.

2.4 Arduino

Para realizar este prototipo y poder controlar los tiempos para cada sensor y el almacenamiento se requiere de un microcontrolador.

Un arduino es una placa con un microcontrolador de la compañía ATMEL, la cual cuenta con los circuitos correspondientes para recibir senales digitales o analogicas por sus entradas o si se requiere generar salidas digitales o analogicas por sus salidas, posee memoria rom y flash, distintos puertos de comunicación para periféricos y diverso hardware especializado.

El arduino viene listo para conectarse a la computadora por su puerto serie y recibir el código para su aplicación, cuenta con puertos para conectar diversos periféricos o hardware especializado a sus puertos digitales o analogicos.

La variedad de tarjetas arduino es extensa pues existen para diferentes requerimientos, desde consumo de energía muy reducido hasta mayor poder de cómputo y más puertos de entrada y salida, así como salidas de voltaje para los periféricos.

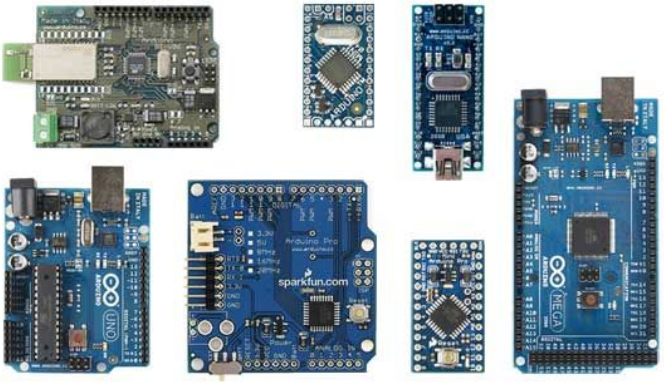

Figura 5. Diferentes tipos y modelos de tarjetas Arduino.

3. Desarrollo

3.1 Introducción y especificaciones de operación

En esta, la era de la información y los avances en la recolección de dicha información, podemos darnos cuenta que aún hay muchas áreas no cubiertas por los avances tecnológicos, una de ellas es la de medicina, podemos soñar con un dia simplemente utilizar un tipo de escáner con el cual poder detectar cualquier enfermedad antes de que nos afecte y poder tratarla de manera prematura, y aun cuando nos faltan años de investigación y trabajo para llegar a dicho escaner, hoy en día contamos con una gran variedad de sensores que nos puedes ayudar a recopilar la mayor cantidad de información, pero no es solo tener dicha información a la mano, sino también poder hacer algo con ella, y en este punto es donde entra la inteligencia artificial para poder darle sentido a los miles de datos obtenidos por los sensores, y poder encontrar un patrón que nos indique algo a nosotros los humanos, es aquí donde utilizando sensores que indican su posición en el espacio, después de implementar un software que es capaz de obtener la información de dichos sensores y poder almacenarla en una memoria para su posterior acceso, con dicho dispositivo facilitamos la obtención de patrones como por ejemplo podría ser el patrón de movimiento que identifica alguna enfermedad que afecte a los huesos o músculos los cuales utilizamos para desplazarnos, y de esta forma poder detectar posteriormente en una persona sana, si contiene o no dicho patrón y en caso de tenerlo, dicha persona debería tratarse prematuramente para evitar que se desarrolle dicha enfermedad, por lo tanto como mencionaba al inicio aún no tenemos un escáner que detecta cualquier enfermedad pero el estudio y desarrollo de proyectos como el que se llevó a cabo en esta investigación puede ayudar a acercarnos a ello.

Se debe desarrollar un dispositivo el cual contará con 5 sensores inerciales los cuales cuentan con 3 ejes para acelerómetro, otros 3 para giroscopio y 3 más para magnetómetro, los datos obtenidos de estos sensores se pasaran por medio del protocolo de comunicación I2C a la unidad de control la cual los iniciara y les solicitará la información por orden y cada cierto tiempo, estos datos se enviarán por medio del puerto serial al sistema de guardado el cual almacenará todo en una microSD, el sistema debe ser autónomo por lo que deberá contar con su propia batería de iones de litio y su módulo de carga, además debe poder visualizarse el número de la muestra en una pantalla la cual se comunicará por medio de serial, todo el equipo debe contar con una carcasa protectora y velcros para permitir su fácil instalación en la persona a prueba.

Se pide que el sistema sea autónomo, móvil, de tamaño reducido, de peso liviano, con una tasa de refresco de pocos ms, funcional, fácil de usar y robusto.

3.2 Selección y configuración de hardware

3.2.1 Microcontrolador

Para este proyecto se utilizó la tarjeta arduino pro mini de 3.3v por su reducido consumo de energía y su tamaño tan reducido (Fig 6). Se tomaron los siguientes criterios para su selección:

- Tamaño reducido para poder crear un módulo de control reducido
- Que contará con pines suficientes para la conexión de los periféricos
- Que contará con soporte a los protocolos de comunicación más comunes (I2C, SPI, Serial)
- Que contará con un bajo consumo de energía
- Que el costo no fuera muy elevado

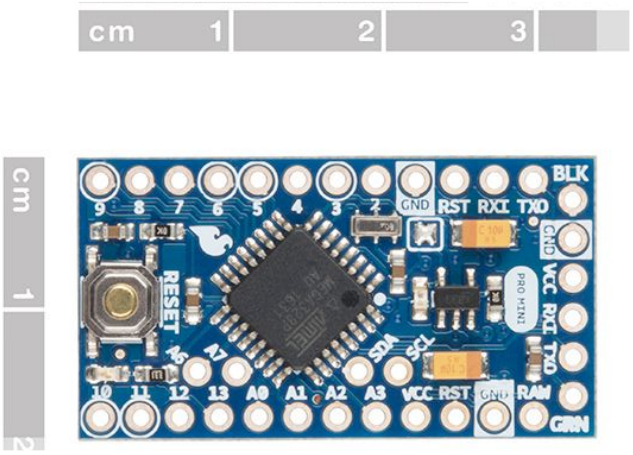

Figura 6. Tarjeta arduino pro mini 328.

Las características principales de la tarjeta arduino seleccionada las podemos ver en la figura 6.

Tabla 1. Especificaciones ATmega 328P.

Microcontrolador ATmega 328P	
Velocidad del CPU	8 MHz
Voltaje de operación	3.3 V
Voltaje de alimentación	3.3 V - 16 V (4 V - 12 V recomendado)
Máximo consumo de corriente al cargar	150 mA @ 3.3 V
Máximo consumo de corriente del chip	200 mA
Máxima entrega por pin	4 mA
Memoria Flash del programa	32 KB
Memoria EEPROM	1 KB
Memoria interna SRAM	2 KB
ADC	10 - bit
Largo	33 mm
Ancho	18 mm

3.2.2 Sensor inercial

Sensores inerciales existe una gran cantidad de ellos en el mercado, sus diferencias van desde el precio, dimensiones físicas, resolución de los datos, protocolos de comunicación soportados (velocidad de transferencia), consumo de energía y calidad de sus componentes.

El MPU-9250 utilizado en este proyecto (figura 3) es de la marca SparkFun, cuenta con acelerómetro, giroscopio y magnetómetro (solo accesible vía I2C) por eje (X,Y,Z). Las características principales del sensor son las siguientes:

Tabla 2. Características técnicas de MPU-9250.

Características técnicas de MPU-9250	
Rango de trabajo del giroscopio de 3 ejes	±250, ±500, ±1,000 and ±2,000°/sec e integra un convertidor analogico-digital (ADC) de 16-bit
Rango de trabajo del acelerómetro de 3 ejes	±2g, ±4g, ±8g and ±16g e integra un convertidor analogico-digital (ADC) de 16-bit
Consumo normal del giroscopio operando	3.2 mA
Consumo normal del acelerómetro operando	450 µA
Consumo normal del magnetómetro operando de 3 ejes	280 µA @ 8 Hz
Rango del voltaje de entrada	2.4V – 3.6 V
Dimensiones	17mm x 18 mm

Por todas ellas se seleccionó el sensor para este proyecto, sumandole su precio contenido, su calidad de materiales, su buena documentación y su disponibilidad.

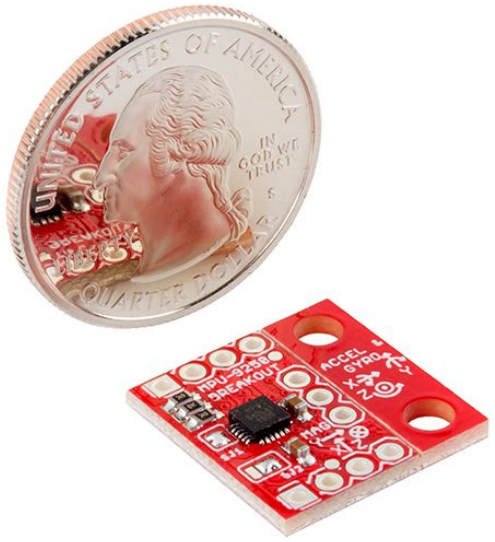

Figura 7. Sensor inercial MPU-9250 comparado con un cuarto de dolar.

3.2.3 Sistema de guardado en memoria

Los datos obtenidos de los sensores se deben guardar en un lugar por lo tanto se optó por la utilización de un sistema de guardado automatizado de la compañía SparkFun y este es el OpenLog el cual utiliza un microcontrolador similar al utilizado como unidad de control en este proyecto el ATmega328 aunque en este caso solamente se utiliza para guardar la información que recibe por el puerto serial.

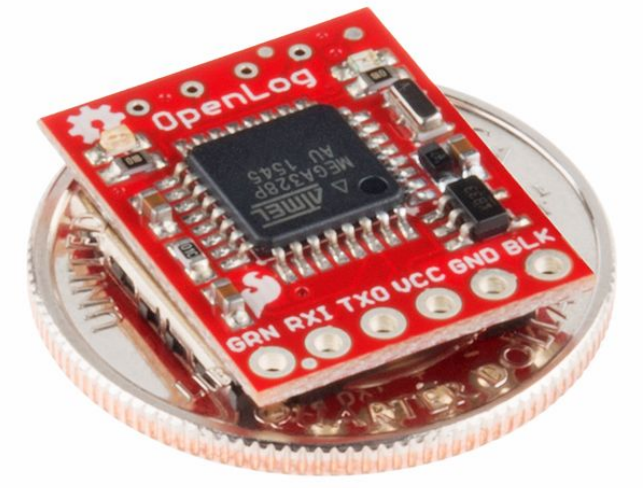

Figura 8. OpenLog de SparkFun.

Tabla 3. Especificaciones técnicas de OpenLog.

Especificaciones técnicas de OpenLog	
Entrada de voltaje	3.3 V - 12 V (3.3 V - 5 V recomendado)
Compatibilidad con memorias microSD	FAT16/32 y con tarjetas hasta 32 GB
Velocidad de baud	hasta 115200 bps
Atmega328 y bootloader pre programable	si
Pines de comunicación	Serial y SPI pogo
Consumo de energía en reposo	2 mA
Máximo consumo de energía	6 mA
Dimensiones	4 mm x 15 mm x 19 mm

3.2.4 Diseño electrónico

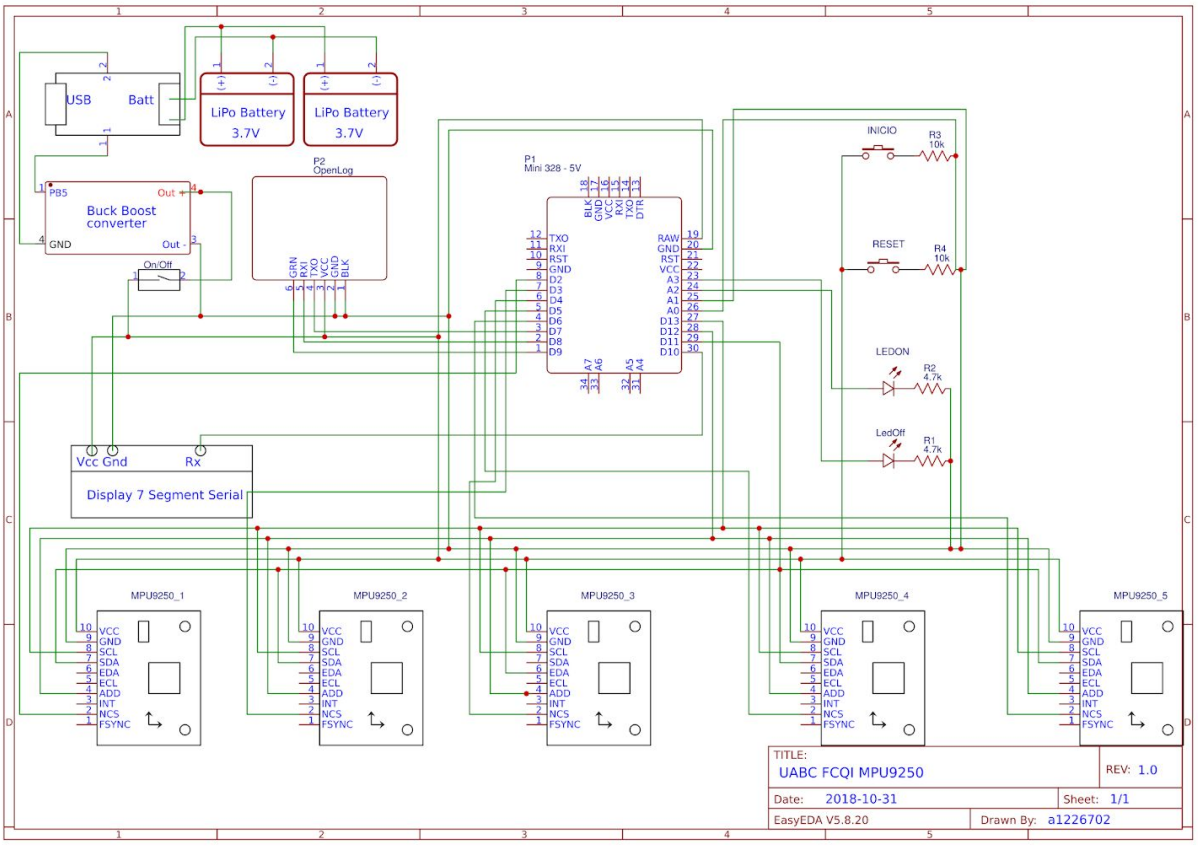

Figura 9. Diseño esquemático.

3.2.5 Diseño físico

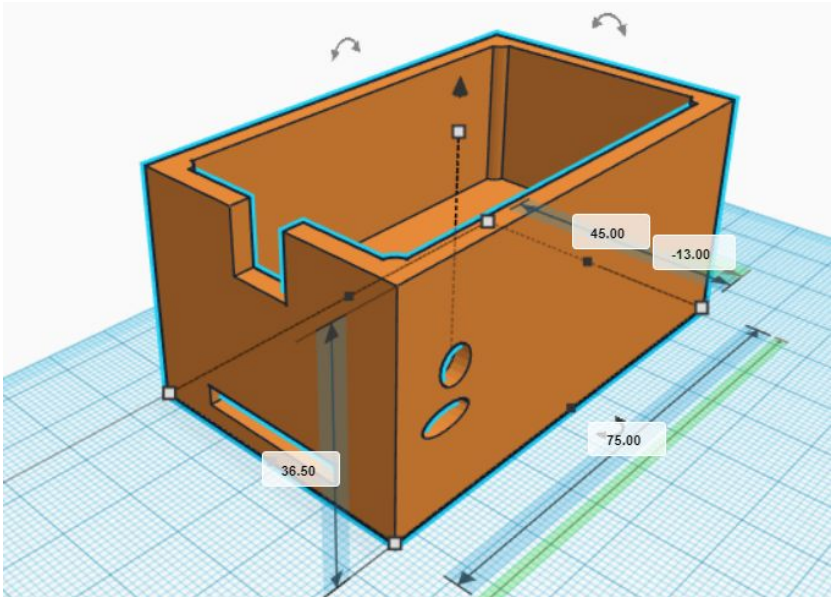

Figura 10. Contenedor de baterías.

Diseño de la caja creada para contener las baterías, la toma de carga, el convertidor de 3.3v a 5v y el interruptor de encendido.

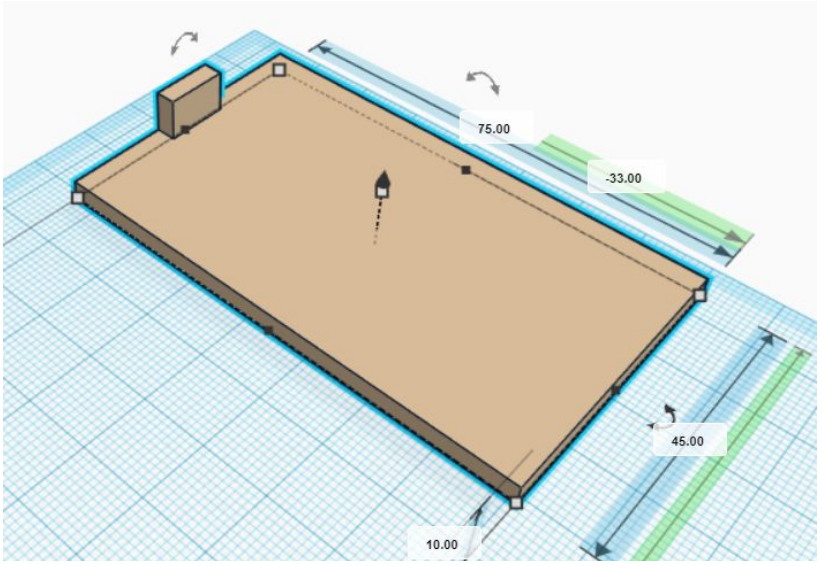

Figura 11. Tapa de baterías.

Diseño de la tapa de la caja de baterías.

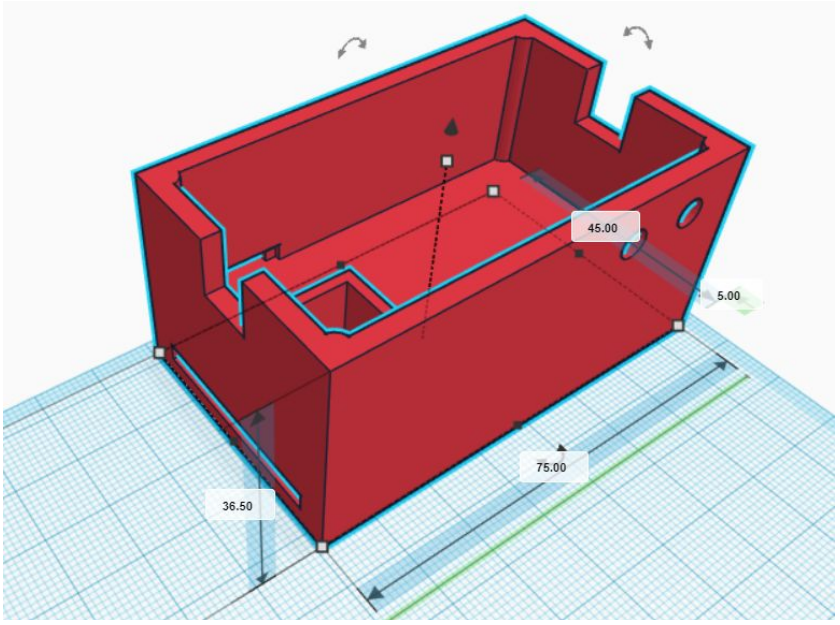

Figura 12. Contenedor de unidad de control.

Diseño de la caja creada para contener el arduino mini pro de 5v, el openlog, el botón de inicio de sensado, el interruptor de reseteo y los led de estado.

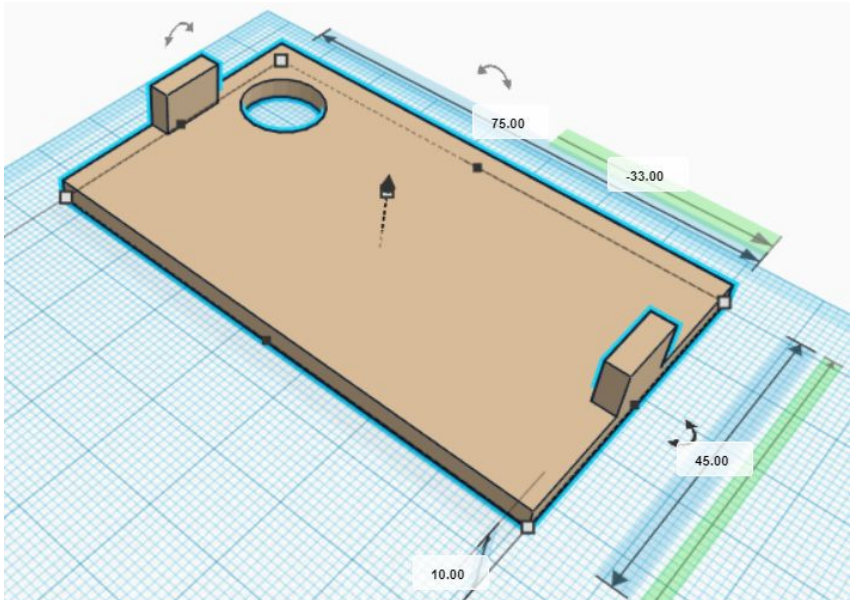

Figura 13. Tapa de la unidad de control

Diseño de la tapa de la caja de el centro de control.

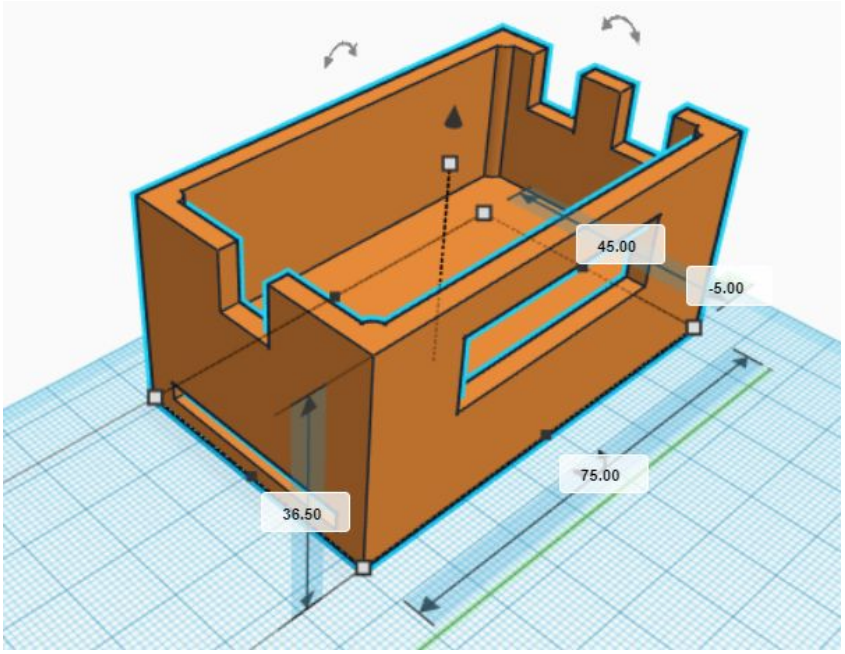

Figura 14. Contenedor del cableado.

Diseño de la caja creada para contener el cableado de tierra y corriente, la pantalla serial y el cableado necesario para el funcionamiento del sistema.

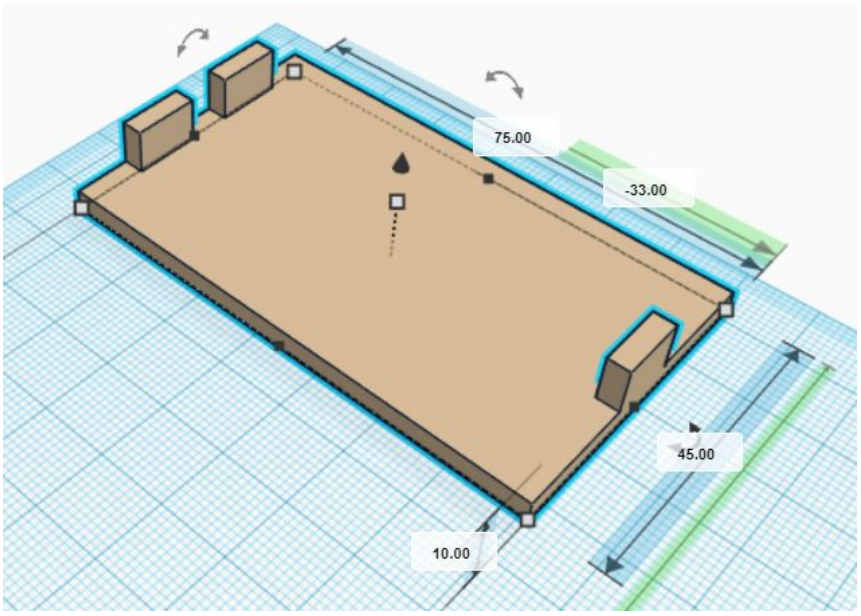

Figura 15. Tapa de cableado.

Diseño de la tapa de la caja del cableado.

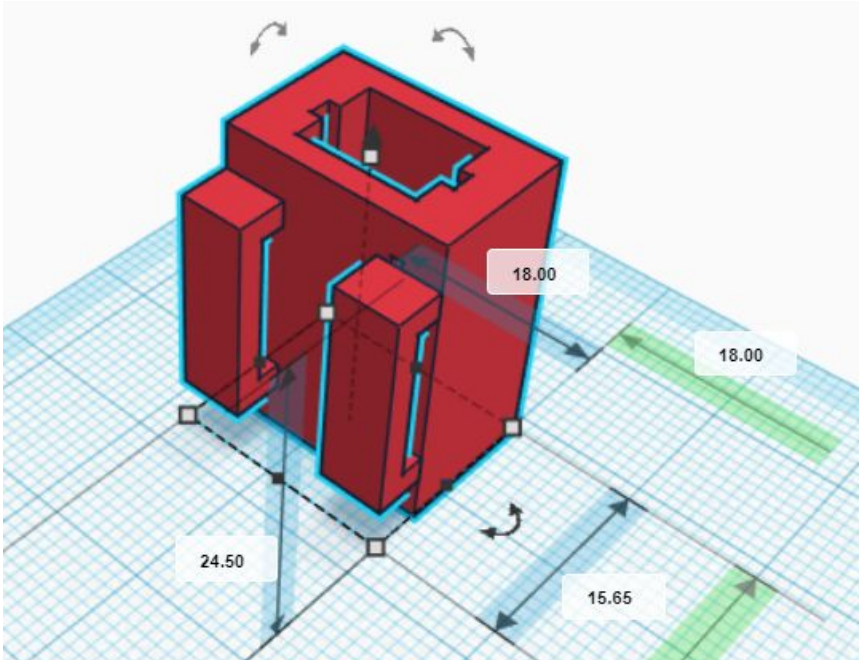

Figura 16. Contenedor de sensor.

Diseño de la caja creada para contener los sensores MPU 9250.

3.2.6 Construcción del equipo físico

Figura 17. Contenedor de baterías armado

Externamente la caja portadora del sistema de alimentación está protegida por la tapa y conectado a la unidad de control por medio de un tubo protector en el cual

internamente lleva los cables de corriente y tierra.

Figura 18. Contenedor de baterías visto de lado.

Cuenta con el interruptor de encendido y con la toma de corriente para cargar las baterías internas.

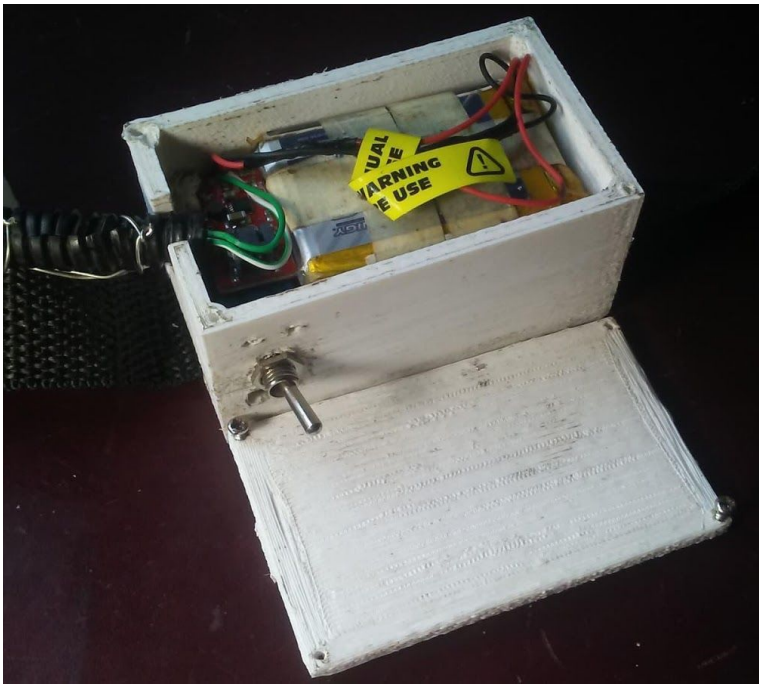

Figura 19. Contenedor de baterías expuesto.

Al remover los tornillos de la tapa podemos encontrarnos con las dos baterías, con el convertidor de voltaje que pasa de 3.3v a 5v y oculto se encuentra el sistema de carga.

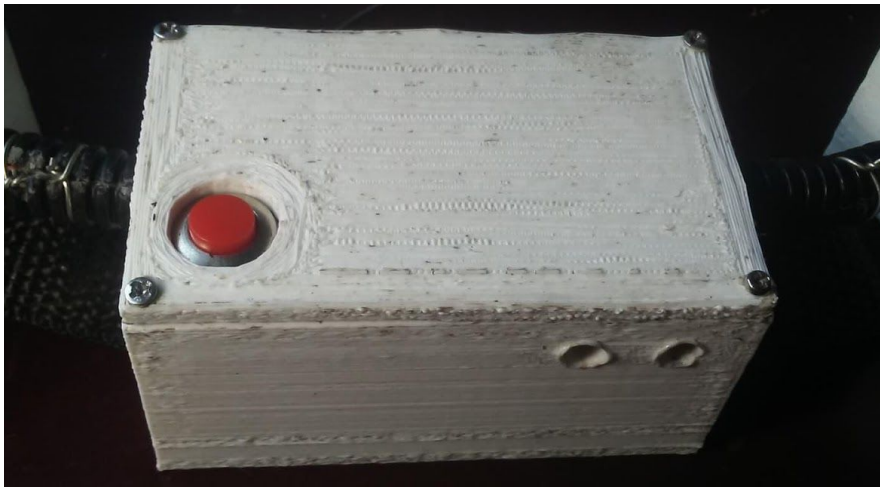

Figura 20. Contenedor de unidad de control armado.

La unidad de control en su parte superior podemos encontrar el botón con el cual se inicia la toma de muestras y en caso de ser presionado de nuevo se finaliza.

Figura 21. Unidad de control vista desde un lado.

Por un lado podemos encontrar la rendija para insertar la memoria microSD en la cual se guardarán los valores obtenidos de los sensores, y los dos leds que indican

el estado en el que se encuentra el dispositivo.

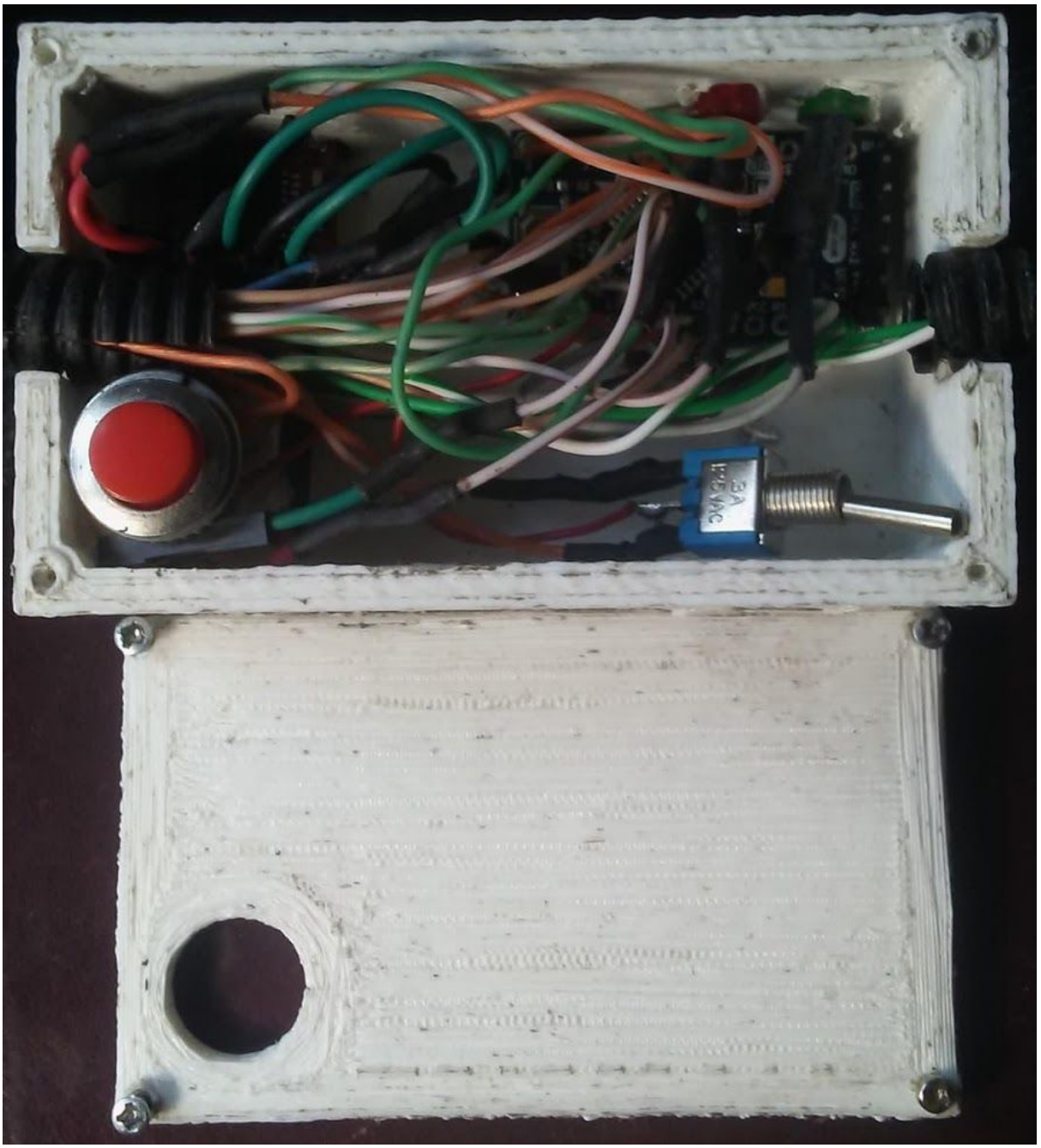

Figura 22. Unidad de control expuesta.

Al abrir la unidad de control, podemos encontrar en la esquina inferior derecha el interruptor de reinicio, en la esquina inferior izquierda el botón de inicio/parado, en la esquina superior derecha el arduino mini pro de 5v que controla el sistema y en la esquina superior izquierda el openlog encargado de guardar los datos obtenidos de

los sensores en la memoria microSD.

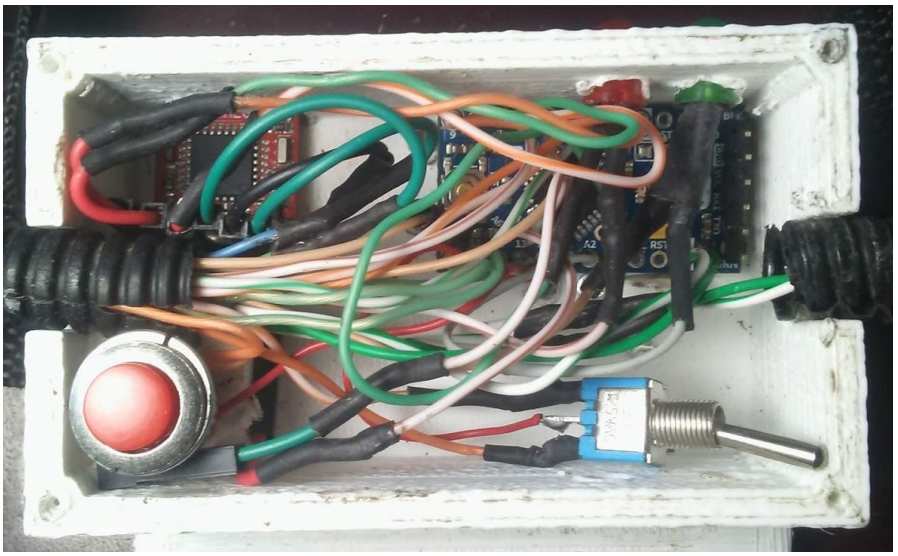

Figura 23. Unidad de control desde arriba.

Aquí visto desde otra perspectiva.

Figura 24. Unidad de cableado y pantalla encendido.

Por un lado la caja de los cables cuenta con la pantalla serial en la cual se mostraran algunos mensajes de estado o el número del archivo en el cual se trabaja.

Figura 25. Unidad de cableado y pantalla visto desde arriba.

Internamente la caja del cableado, contiene dos conectores de 15 pines cada uno de ellos, el de la parte superior es el de corriente y el de la parte inferior es el de tierra, además contiene la pantalla serial y es el puente entre los cables externos de los sensores con la unidad de control.

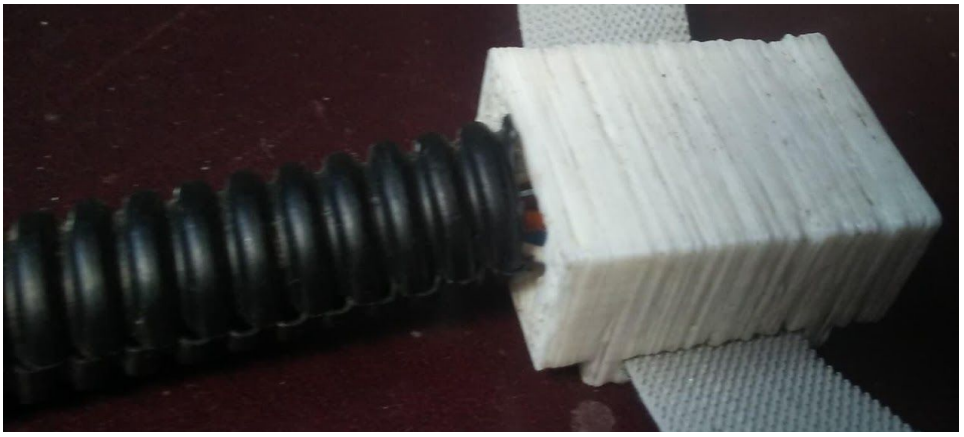

Figura 26. Contenedor de sensor armado.

La caja protectora de cada sensor está diseñada para dar cobijo al sensor y poder colocarlo ya sea con el eje X apuntando hacia fuera del cuerpo de la persona de prueba o hacia el interior de ella.

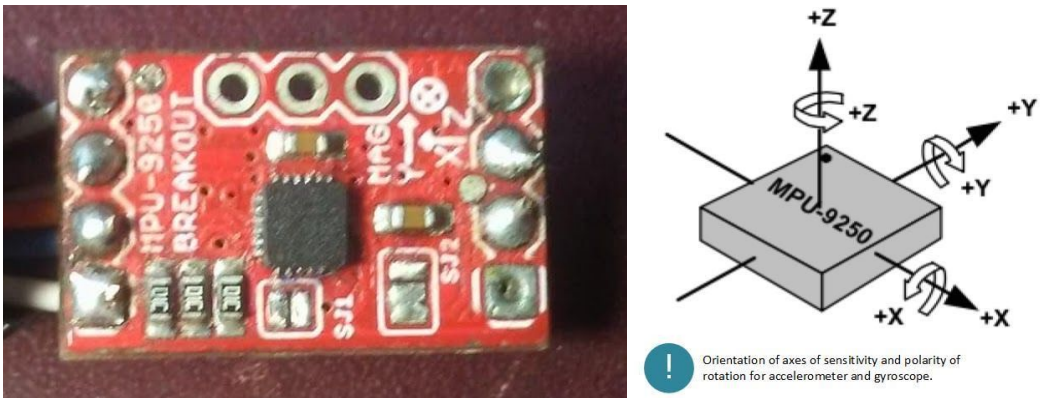

Figura 27. A la izquierda MPU-9250 y a la derecha ilustración de los ejes del MPU-9250.

El punto que indica la posición del sensor, aún cuando no se ve en la fotografía se encuentra en la parte superior izquierda de acuerdo a esta fotografía y en la imagen de un lado podemos ver hacia donde apunta cada eje.

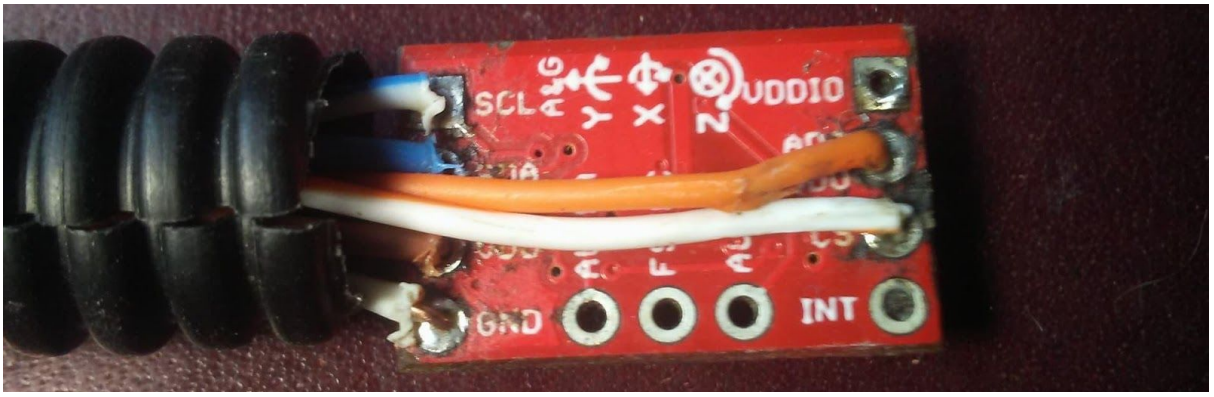

Figura 28. MPU-9250 visto desde abajo.

Por su parte trasera podemos encontrar el cableado y la indicación de los ejes que ya se muestra en la imagen expuesta arriba.

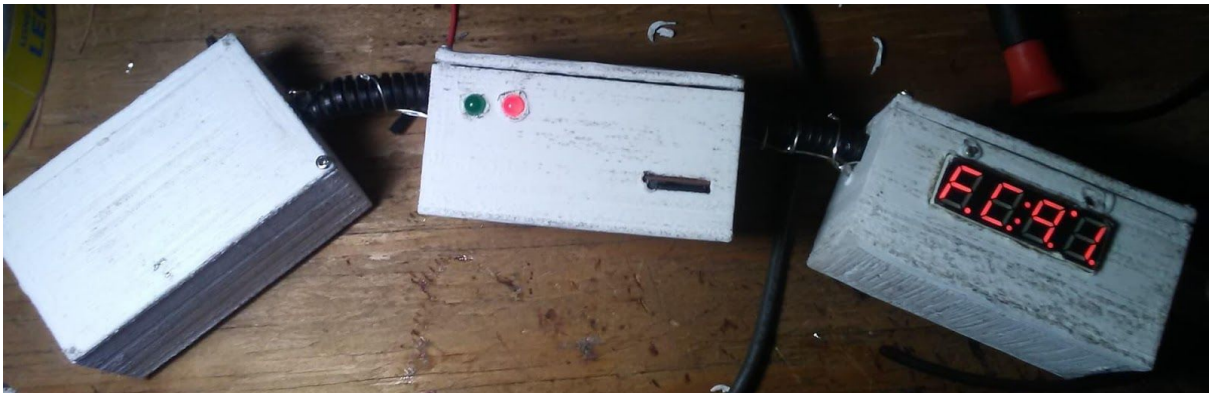

Figura 29. Equipo encendido.

Aquí una muestra de como se ve el dispositivo al ser encendido, muestra en pantalla el mensaje “FCQI” y enciende el led rojo.

Figura 30. Equipo encendido funcionando.

Aquí ya se presionó el botón de inicio/parado, y la pantalla pasa a mostrar el número de archivo en el cual se trabaja en este ciclo, además de haberse apagado el led rojo y encendido el led verde.

3.3 Programa

3.3.1 Instrucciones de uso

Pasos para iniciar el sensado

1. Activar el interruptor que enciende el dispositivo.
2. Esperar a que se deje de mostrar el mensaje “FCQI” en la pantalla.
3. Presionar el botón rojo por un tiempo mayor a 0.5 segundos.
4. Pasado el tiempo necesario, dejar de presionar el botón rojo.
5. La pantalla mostrará un número entre 0 y 9999 el cual representa el número de archivo en el cual se está trabajando en ese momento y en el cual se guardará la información obtenida, además se encenderá el led verde y se apagará el led rojo.
6. Iniciar la recolección de información.

Pasos para finalizar el sensado

1. Presionar el botón rojo por un tiempo mayor a 0.5 segundos.
2. Pasado el tiempo necesario, dejar de presionar el botón rojo.

3. El led rojo se encenderá y el led verde se apagará indicando que se finalizó la toma de datos.

Pasos para guardar el trabajo y resetear la memoria

1. Con el dispositivo apagado, remover la tapa que cubre la unidad de control.
2. Activar el interruptor que enciende el dispositivo.
3. Esperar a que se deje de mostrar el mensaje “FCQI” en la pantalla.
4. Activar el interruptor de reset por un tiempo mayor a 2.5 segundos.
5. Pasado el tiempo necesario, desactivar el interruptor de reset.
6. Al aparecer el mensaje en la pantalla de “Clen” apagar el dispositivo.
7. Sacar la memoria microSD del dispositivo.
8. Conectar la memoria microSD a la computadora.
9. Copiar los archivos que se encuentran en la memoria microSD menos el último archivo generado pues este se genera al resetear el dispositivo y queda vacío.
10. Eliminar los archivos de la memoria y solamente dejar el archivo nombrado “config.txt”
11. Sacar la memoria microSD de la computadora.
12. Insertar la memoria microSD en el dispositivo.
13. Colocar la tapa protectora del segmento de control.
14. Listo.

Significado de mensajes en pantalla serial

“FCQI”: Este mensaje se da al iniciar el dispositivo y mientras aparece en pantalla significa que el dispositivo está iniciando los sensores y por lo tanto no se puede accionar aún su funcionamiento, al desaparecer dicho mensaje de pantalla nos indica que ya podemos iniciar el sensado presionando el botón de inicio/parado.

“CLEN”: Este mensaje se da al accionar por más de dos segundos y medio el interruptor de reseteo, y no desaparecerá hasta apagar el dispositivo, nos indica que ya podemos retirar la memoria microSD para guardar los datos contenidos en ella y que ya se reseteo el contador interno en la Eeprom de arduino y del openlog.

“FAIL”: Este mensaje se da al intentar iniciar un dispositivo y este no dar correctamente su “nombre” el cual en realidad es un valor contenido en uno de sus registros que indica el correcto funcionamiento de todas las partes involucradas en

la comunicación del sensor y el arduino, por lo que si se da dicho error, se debe revisar el cableado pues puede estar roto en alguna de sus secciones.

“Número”: Este mensaje se da al inciar el sensado e indica el número del archivo en el cual se está trabajando en ese momento.

4.- Conclusión

La realización de este proyecto conllevo una gran cantidad de fallas pues fue el primer gran proyecto que lleve a cabo, con anterioridad había desarrollado software para sistemas embebidos pero en ningún momento algo con tanta complejidad y con la misma cantidad de piezas por unir y obtener un sistema funcional, rápido, eficaz, móvil y practico, surgieron problemas al trabajar con el guardado en el openlog pues este al disponer de un microcontrolador idéntico al utilizado en la unidad de control nos permite hacer más que solamente guardar pues se debía de conocer el número de archivos, resetear el contador de los archivos internos del cual se obtiene el número para cada archivo generado, cuidar de no saturar la reducida memoria caché que dispone el dispositivo para datos entrantes, por otra parte los sensores de movimiento seleccionados inicialmente disponían de un microcontrolador integrado, dichos sensores fueron seleccionados pues al contar con una unidad de control se podía relegar todo el trabajo del cálculo de los datos obtenidos a ellos y dejar la unidad de control principal solamente para recibir dichos datos y enviarlos al dispositivo de guardado, pero resultó ser una opción compleja de implementar pues la comunicación que se realizaba por el protocolo de comunicación I2C requería que tanto la unidad de control principal como las unidades de control de los sensores estuvieran habilitados como maestros lo cual indicaba que podían iniciar envíos de datos y en otras situaciones estar a la escucha de peticiones de la unidad de control, dicho modelo de comunicación no se podía realizar por limitaciones del hardware, se intentó implementar de una gran variedad de maneras desde intentar crear algo similar a una ramificación en la cual los sensores se iban pasando los datos unos a otros y al final llegan los datos a la unidad de control principal hasta la variación de maestro/esclavo de los dispositivos, finalmente se optó por utilizar los sensores sin microcontrolador y ello permitió una mayor simpleza en el código y en el modelo de comunicación que aún se realizaba con el mismo protocolo, del mismo modo me encontré con problemas a la hora de poder inicializar los sensores pues simplemente se inicializaba uno y los demás no se llegaban a activar, estudiando las bibliotecas encontré el fallo y cree un método que modifica una variable de dichas bibliotecas pues no estaban pensadas para inicializar más de un sensor a la vez de forma independiente a los demás, otro de los problemas encontrados fue el tiempo de lectura de los sensores y guardado en memoria pues por requerimientos se requiere que este fuera menor de 25ms lo cual en las primeras pruebas era prácticamente imposible de alcanzar pero al optar por simplificar la recolección de los datos y reducir el trabajo de la unidad de control principal se pudo llegar a dichos tiempos, esos fueron los problemas más notables por parte del software y por parte del hardware los principales se dieron con el cableado pues al tener la implementación final del programa y correrla en el equipo

de pruebas funcionaba perfectamente con el doble de los sensores utilizados en la versión final, pero al momento de cablear con las especificaciones requeridas en el dispositivo de control final, osea, el arduino mini pro me encontré con algunos fallos, el principal la comunicación ya no se daba correctamente con todos los sensores, al revisar el cableado después de varias pruebas se pudo comprobar y validar que se trataba por limitaciones de cableado y al voltaje en las líneas de comunicación del protocolo I2C, se optó por reducir el número de sensores de movimiento utilizados y conseguir mejor cableado lo cual soluciono la comunicación, realizar este proyecto me hizo darme cuenta de lo importante que puede ser una línea de código pues no solamente afecta al mundo virtual sino que puede llegar a causar estragos en el mundo real donde todos convivimos, hoy fue un sistema de sensado mañana quien sabe.

5.- Bibliografía

Guzmán Balcázar, G. B., Mayra Alejandra, M. A., Toscano Pozo, T. P., & Washington Hernán, W. H. (2013, 1 octubre). Repositorio Institucional de la Universidad Politécnica Salesiana: Sistema de captura y evaluación del movimiento del cuerpo humano.. Recuperado 29 noviembre, 2019, de https://dspace.ups.edu.ec/handle/123456789/5339

Rodríguez Martín, D. (2014, 21 mayo). Contribución al análisis del movimiento humano aplicado a la identificación de posturas y bloqueos de la marcha en pacientes con Parkinson. Recuperado 29 noviembre, 2019, de http://hdl.handle.net/10803/144622

Aparicio Pérez, M. A. (2015, 19 noviembre). Dispositivo de captura de movimiento basado en sensores inerciales con comunicación inalámbrica - Repositorio Institucional de Documentos. Recuperado 29 noviembre, 2019, de http://zaguan.unizar.es/record/36798

Callejas Cuervo, M., Vélez Guerrero, M. A., & Pérez Holguín, W. J. (2018, 8 diciembre). Arquitectura de un sistema de medición de bioparámetros integrando señales inerciales-magnéticas y electromiográficas | Callejas-Cuervo | REVISTA POLITÉCNICA. Recuperado 29 noviembre, 2019, de http://revistas.elpoli.edu.co/index.php/pol/article/view/1406/1088

Camargo C., E., Garzón G., Y, & Camacho P., V. A. (2012, 26 junio). Vista de Sistema portátil de captura de movimiento para el análisis cinemático de la marcha humana. Recuperado 29 noviembre, 2019, de https://revistas.udistrital.edu.co/index.php/Tecnura/article/view/6853/8437

Vivas Mateos, G. (2016, 1 septiembre). Desarrollo de un método de captación de movimiento humano para el control remoto de terapias de rehabilitación robóticas. Recuperado 29 noviembre, 2019, de http://oa.upm.es/43792/1/TFG_GUILLERMO_VIVAS_MATEOS.pdf

Dartfish. (s.f.). Use video and data analysis to detect vulnerabilities. Decide what to focus on and develop technical and tactical strategies to improve performance and reduce injuries.. Recuperado 29 noviembre, 2019, de https://www.dartfish.com/

Technaid. (s.f.). Sistema de Captura de Movimiento | Technaid - Leading Motion. Recuperado 29 noviembre, 2019, de https://www.technaid.com/es/productos/motion-capture-system/

A.- Apéndices

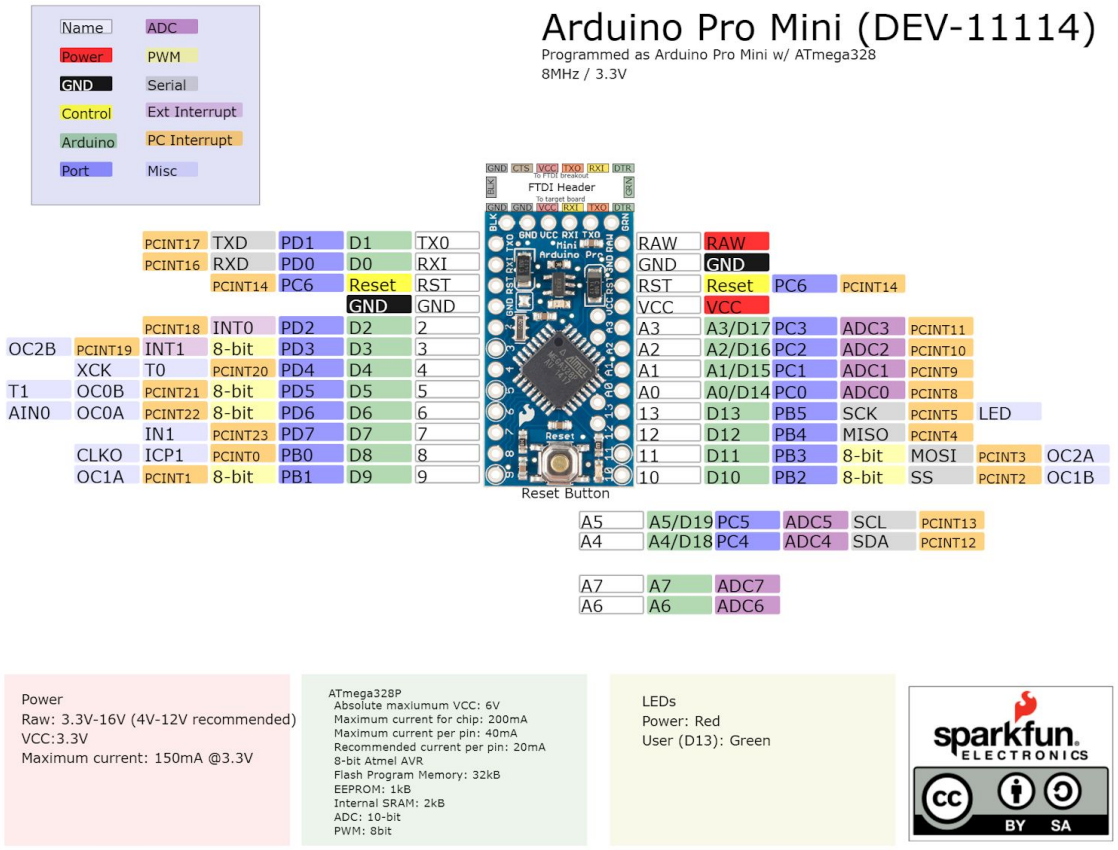

Figura 31. Especificaciones arduino mini pro.

Código

```
#include <SoftwareSerial.h>
// La biblioteca para los canales seriales virtuales
#include "MPU9250.h"
// La biblioteca para las funciones de los sensores
#include <TimeLib.h>
// La biblioteca necesaria para tomar los tiempos de los botones
#include <EEPROM.h>
// La biblioteca utilizada para guardar el número de archivo

// Definición de Pines
#define Disp1 2
```

```
// El pin 2 será el encargado de activar la comunicación del sensor 1
#define Disp2 3
// El pin 3 será el encargado de activar la comunicación del sensor 2
#define Disp3 4
// El pin 4 será el encargado de activar la comunicación del sensor 3
#define Disp4 5
// El pin 5 será el encargado de activar la comunicación del sensor 4
#define Disp5 6
// El pin 6 será el encargado de activar la comunicación del sensor 5
//#define Disp6 A6
// El pin A6 queda comentado para su posible utilización
//#define Disp7 A7
// El pin A7 queda comentado para su posible utilización
#define RxOpenlog 7
// El pin 7 es el Tx virtual para comunicarse con el OpenLog
#define TxOpenlog 8
// El pin 8 es el Rx virtual para recibir la información del OpenLog
#define PinRstOpenLog 9
// El pin 9 es el encargado de reiniciar el OpenLog
#define RxDisplay 10
// El pin 10 es el Tx virtual para comunicarse con el OpenLog
#define PinIniBoton A0
// El pin A0 es el encargado del botón para iniciar el sensado
#define PinRstBoton A1
// El pin A1 es el encargado del botón para reiniciar el número de archivo guardado en la EEPROM y además de reiniciar el conteo del OpenLog
#define LedOn A2
// El pin A2 activa el led color verde que indica el inicio del sensado
#define LedOff A3
// El pin A3 activa el led color rojo que indica el estado detenido
#define NumDisp 5
// Especifica el número de sensores conectados al arduino mini pro

// Se declaran e inicializan las variables que se utilizaran en el proyecto
int Direccion = 0, cambio = 0, paso = 0, numMedicionOpenlog = 0, seleccionador = 1, tmp = 0;
bool activado = false, inicioSensores = false, peticionApagar = false;
unsigned long inicio, pulsadoIn, pulsadoOut, rstIn, rstOut;
char letra, letraAnt;
char tempString[10];
```

```
// Se utiliza una union para pasar rápidamente y de manera más eficiente el número de archivo a la EEPROM o sacar de ella el valor almacenado y pasarlo a un entero
union dato_tag {
  byte stream[2];
  int val;
} dato_Union;

// Se utiliza una union por cada dato obtenido de los sensores para permitir rápidamente su conversión a byte y poder enviarla al OpenLog de manera más eficiente, obteniendo un ahorro de tiempo al enviar menos bytes
union dato {
  byte stream[4];
  float val;
} ax_Union, ay_Union, az_Union, gx_Union, gy_Union, gz_Union;

 // Se inicializa la variable que cargara todos los métodos referentes a los sensores, se le envía el pin del sensor 1 el cual posteriormente se cambia por los demás pines de acuerdo a como se requiera.
MPU9250 myIMU1 = MPU9250(Disp1);

// Se inicializan los seriales virtuales que se utilizaran para poder comunicar el arduino con el OpenLog y con la pantalla serial, se utiliza este método ya que arduino mini pro solo cuenta con un puerto serial, el cual se deja intacto pues se puede necesitar para programar el arduino
SoftwareSerial OpenLog(TxOpenlog, RxOpenlog); // (RX, TX)
SoftwareSerial Display(0, RxDisplay); // (RX, TX)

void setup() {
  Serial.begin(115200);
  // Se Inicia la comunicación serial
  OpenLog.begin(115200);
  // Se Inicia la comunicación serial virtual con el OpenLog
  pinMode(Disp1, OUTPUT);
  digitalWrite(Disp1, HIGH);
  pinMode(Disp2, OUTPUT);
  digitalWrite(Disp2, HIGH);
  pinMode(Disp3, OUTPUT);
  digitalWrite(Disp3, HIGH);
  pinMode(Disp4, OUTPUT);
  digitalWrite(Disp4, HIGH);
```

```
  pinMode(Disp5, OUTPUT);
  digitalWrite(Disp5, HIGH);
  pinMode(PinRstOpenLog, OUTPUT);
  digitalWrite(PinRstOpenLog, HIGH);
  pinMode(PinIniBoton, INPUT);
  pinMode(PinRstBoton, INPUT);
  pinMode(LedOn, OUTPUT);
  pinMode(LedOff, OUTPUT);
  digitalWrite(LedOn, LOW);
  digitalWrite(LedOff, HIGH);
  Display.begin(9600);
  setBrightness(200);
  clearDisplay();
  Display.print("FCQI");
  // Se envía un mensaje a mostrar en la pantalla
  setDecimals(0b111111);
  // Se encienden el acceso a todos los leds de la pantalla
  iniciarSensores();
  // Se inicializan los sensores
  clearDisplay();
  // Se limpia la pantalla dejándola lista para mostrar el número de archivo
}

void iniciarSensores() {
  for (int i = 1; i <= NumDisp; i++ ) {
// Para poder iniciar cada sensor se debe cambiar el estado del pin que lo
activa pero además se debe modificar el valor interno que maneja la biblioteca
del sensor para que ahora trabaje con dicho sensor y de este modo se evite
crear una variable de la biblioteca por cada sensor agregando tamaño extra
por cada variable
    switch (i) {
      case 1:
        myIMU1.cambiarCS(Disp1);
        break;
      case 2:
        myIMU1.cambiarCS(Disp2);
        break;
      case 3:
        myIMU1.cambiarCS(Disp3);
        break;
      case 4:
```

```
      myIMU1.cambiarCS(Disp4);
      break;
    case 5:
      myIMU1.cambiarCS(Disp5);
      break;
  }
```

// Se inicializa el sensor

```
  myIMU1.begin();
```

// Se procede a comprobar el estado de la conexión con el sensor, solicitando su "nombre" el cual nos responderá 0x71 si todo funciona correctamente

```
  Serial.println("Testing MPU9250::readBytes...");
  byte fc = 0;
   Serial.println( myIMU1.readBytes(MPU9250_ADDRESS, WHO_AM_I_MPU9250,
1, &fc) );
  Serial.println(fc, HEX);
  byte c = myIMU1.readByte(MPU9250_ADDRESS, WHO_AM_I_MPU9250);
  Serial.print(F("MPU9250 I AM 0x"));
  Serial.print(c, HEX);
  Serial.print(F(" I should be 0x"));
  Serial.println(0x71, HEX);
```

// Si el "nombre" obtenido del sensor es correcto se procede a terminar la configuración de dicho sensor

```
  if (c == 0x71) // WHO_AM_I should always be 0x71
  {
    Serial.println(F("MPU9250 is online el ..."));
```

// Se comprueba el estado de los ejes del acelerómetro y del giroscopio

```
    myIMU1.MPU9250SelfTest(myIMU1.selfTest);
    Serial.print(F("x-axis self test: acceleration trim within : "));
    Serial.print(myIMU1.selfTest[0], 1); Serial.println("% of factory value");
    Serial.print(F("y-axis self test: acceleration trim within : "));
    Serial.print(myIMU1.selfTest[1], 1); Serial.println("% of factory value");
    Serial.print(F("z-axis self test: acceleration trim within : "));
    Serial.print(myIMU1.selfTest[2], 1); Serial.println("% of factory value");
    Serial.print(F("x-axis self test: gyration trim within : "));
    Serial.print(myIMU1.selfTest[3], 1); Serial.println("% of factory value");
    Serial.print(F("y-axis self test: gyration trim within : "));
    Serial.print(myIMU1.selfTest[4], 1); Serial.println("% of factory value");
```

```
      Serial.print(F("z-axis self test: gyration trim within : "));
      Serial.print(myIMU1.selfTest[5], 1); Serial.println("% of factory value");

// Se calibran los sensores
      myIMU1.calibrateMPU9250(myIMU1.gyroBias, myIMU1.accelBias);

// Se termina el inicio del sensor, enviando los bytes necesarios a los registros para configurarlo correctamente
     myIMU1.initMPU9250();

// Se activa el acceso al giroscopio, acelerómetro y a la temperatura del sensor
      myIMU1.writeByte(0x68, 0x37, 0x22);
      Serial.println("MPU9250 initialized for active data mode....");

// Se habilita el acceso a los datos del acelerómetro y el giroscopio
      myIMU1.getAres();
      myIMU1.getGres();
    } // if (c == 0x71)
    else
    {

// En caso de no haber sido iniciado el sensor correctamente se envia por serial el error o en caso de no tener conectado el FTDI se muestra en la pantalla “FAIL” para denotar que existe un error
      Serial.print("Could not connect to MPU9250: 0x");
      Serial.println(c, HEX);
      Serial.println(F("Communication failed, abort!"));
      Serial.flush();
      Display.print("FAIL");  // Se envía el mensaje a la pantalla
      setDecimals(0b111111);  // Se activa el acceso a todos los leds de la pantalla
      abort();
    }
    delay(50);
  }
  Serial.println(F("Fin inicio sensores"));
}

void loop() {

// Se comprueba el valor del pin que representa el botón de inicio en caso de ser alto se toma una muestra de tiempo para evitar el accionamiento por error,
```

así que para su correcto funcionamiento el tiempo de presión en dicho botón debe ser superior a 500 milisegundos

```
  if ((digitalRead(PinIniBoton) == HIGH)) {
    pulsadoIn = millis();
    while (digitalRead(PinIniBoton) == HIGH);
    pulsadoOut = millis() - pulsadoIn;
```

// El proyecto puede presentar dos estados al presionar el botón de inicio, al presionar la primera vez desde su inicio, estando las variables con los valores declarados se pasará a iniciar la toma de muestras de los sensores, se comprobará el número de archivo almacenado en la memoria EEPROM y se reiniciara el OpenLog para finalizar el archivo anterior en caso de existir alguno y para iniciar uno nuevo en el cual guardar los datos recibidos

```
    if (pulsadoOut >= 500) {
      cambio++;
      if ((paso == 0) && (cambio == 1) && (!inicioSensores)) { // Prendido
        tmp = numMedicionOpenlog;
        comprobarEepromOpenlog();
        if (numMedicionOpenlog > 0 && tmp == numMedicionOpenlog)
          resetOpengLog();
        digitalWrite(LedOff, LOW);
        digitalWrite(LedOn, HIGH);
        inicioSensores = true;
        paso = 1;
      }
```

// El segundo estado se da al presionar por segunda vez el botón de inicio y en este estado se procede a apagar la recolección de los sensores, se aumenta el número de archivo para representar correctamente el número en el cual se trabajó y se pasa a guardarlo en la memoria EEPROM

```
      else if ((cambio == 2) && (inicioSensores)) { //vaciando buffer
        peticionApagar = true;
        cambio = 0;
        paso = 0;
        numMedicionOpenlog++;
        guardarEepromOpenlog(numMedicionOpenlog);
      }
    }
  }
```

```
// Al activar el interruptor de reset por más de 2500 milisegundos se accede al reinicio del OpenLog para comenzar de nuevo en el archivo 0 y además se cambia el valor guardado en la EEPROM por 0
  if ((!inicioSensores) && (digitalRead(PinRstBoton) == HIGH)) {
    rstIn = millis();
    while (digitalRead(PinRstBoton) == HIGH);
    rstOut = millis() - rstIn;

     // Se cambia Rx de alto a bajo para resetear el OpenLog con ello se reinicia el archivo config con el baud de 115200 y el valor de la eeprom tanto del OpenLog como del arduino mini pro del número de archivo a 0
    if (rstOut >= 2500)
    {
      digitalWrite(RxOpenlog, LOW);
      digitalWrite(PinRstOpenLog, LOW);
      delay(250);
      digitalWrite(PinRstOpenLog, HIGH);
      guardarEepromOpenlog(0);
      while (1) {
           Serial.println(F("Vaciar memoria MicroSD y solo dejar el archivo config.txt recordar el baud es 115200"));
        Display.print("CLEN");
        setDecimals(0b111111);
      }
    }
  }

  if (inicioSensores) {
       myIMU1.readAccelData(myIMU1.accelCount);   // Se leen los datos de los registros que corresponden a x/y/z del acelerómetro
       myIMU1.readGyroData(myIMU1.gyroCount);     // Se leen los datos de los registros que corresponden a x/y/z del giroscopio

// Se obtiene el valor en flotante de cada eje y se pasa a una unión que lo representa
    ax_Union.val = (float)myIMU1.accelCount[0] * myIMU1.aRes;
    ay_Union.val = (float)myIMU1.accelCount[1] * myIMU1.aRes;
    az_Union.val = (float)myIMU1.accelCount[2] * myIMU1.aRes;
    gx_Union.val = (float)myIMU1.gyroCount[0] * myIMU1.gRes;
    gy_Union.val = (float)myIMU1.gyroCount[1] * myIMU1.gRes;
    gz_Union.val = (float)myIMU1.gyroCount[2] * myIMU1.gRes;
```

```
// Se actualiza el tiempo interno del sensor
    myIMU1.updateTime();

// Se procede a guardar los datos obtenidos del sensor
    guardarDatos();

// Se cambia de sensor para acceder al siguiente
    cambiarDisp(seleccionador);
  }
}
```

// En esta función se accede a la memoria EEPROM y se obtiene el dato guardado en ella para comprobar si es mayor al que contiene numMedicionOpenlog, en caso de ser mayor significa que ya existen mediciones guardadas en el OpenLog con anterioridad y en ese caso el nuevo valor de numMedicionOpenlog es el guardado en la EEPROM, además de mostrar en la pantalla el número del archivo en el cual se trabajara en esa sesión de sensado

```
void comprobarEepromOpenlog() {
  clearDisplay();
  dato_Union.stream[1] = EEPROM.read(Direccion);
  dato_Union.stream[0] = EEPROM.read(Direccion + 1);
  if (dato_Union.val > numMedicionOpenlog)
    numMedicionOpenlog = dato_Union.val;
  sprintf(tempString, "%4d", numMedicionOpenlog);
  Display.print(tempString);
  setDecimals(0b00000000);
}
```

// En esta función se obtiene la parte alta y baja del número que representa el número de archivo que se guardará en la EEPROM con ayuda de una union y se pasa a guardarlo

```
void guardarEepromOpenlog(int num) {
  dato_Union.val = num;
  EEPROM.write(Direccion, dato_Union.stream[1]);
  EEPROM.write(Direccion + 1, dato_Union.stream[0]);
}
```

// Función demo de como convertir un grupo de 4 bytes en un flotante ejemplo de uso: Serial.println(byte_to_float(ax_Union.stream));

```
float byte_to_float(byte var[]) {
  byte b[4];
  b[3] = var[3];
  b[2] = var[2];
  b[1] = var[1];
  b[0] = var[0];
  return (*(float *)&b);
}
```

// La función que se encarga de guardar los datos obtenidos de los ejes de cada sensor, son enviadas al OpenLog en forma de byte de esta forma se realiza el envío de forma más eficiente, el grabado de un ciclo de lectura de los cinco sensores en la memoria microSd queda de la siguiente manera:

byte1Lowbyte1High … byte15Lowbyte15High … byte30Lowbyte30High

De los cuales:

byte1Low es ax parte baja y byte1High es ax parte alta del sensor 1
byte2Low es ay parte baja y byte2High es ay parte alta del sensor 1
byte3Low es az parte baja y byte3High es az parte alta del sensor 1
byte4Low es gx parte baja y byte4High es gx parte alta del sensor 1
byte5Low es gy parte baja y byte5High es gy parte alta del sensor 1
byte6Low es gz parte baja y byte6High es gz parte alta del sensor 1
byte7Low es ax parte baja y byte7High es ax parte alta del sensor 2
byte8Low es ay parte baja y byte8High es ay parte alta del sensor 2
byte9Low es az parte baja y byte9High es az parte alta del sensor 2
byte10Low es gx parte baja y byte10High es gx parte alta del sensor 2
byte11Low es gy parte baja y byte11High es gy parte alta del sensor 2
byte12Low es gz parte baja y byte12High es gz parte alta del sensor 2
byte13Low es ax parte baja y byte13High es ax parte alta del sensor 3
byte14Low es ay parte baja y byte14High es ay parte alta del sensor 3
byte15Low es az parte baja y byte15High es az parte alta del sensor 3
byte16Low es gx parte baja y byte16High es gx parte alta del sensor 3
byte17Low es gy parte baja y byte17High es gy parte alta del sensor 3
byte18Low es gz parte baja y byte18High es gz parte alta del sensor 3
byte19Low es ax parte baja y byte19High es ax parte alta del sensor 4
byte20Low es ay parte baja y byte20High es ay parte alta del sensor 4
byte21Low es az parte baja y byte21High es az parte alta del sensor 4
byte22Low es gx parte baja y byte22High es gx parte alta del sensor 4
byte23Low es gy parte baja y byte23High es gy parte alta del sensor 4
byte24Low es gz parte baja y byte24High es gz parte alta del sensor 4

byte25Low es ax parte baja y byte25High es ax parte alta del sensor 5
byte26Low es ay parte baja y byte26High es ay parte alta del sensor 5
byte27Low es az parte baja y byte27High es az parte alta del sensor 5
byte28Low es gx parte baja y byte28High es gx parte alta del sensor 5
byte29Low es gy parte baja y byte29High es gy parte alta del sensor 5
byte30Low es gz parte baja y byte30High es gz parte alta del sensor 5

```
void guardarDatos() {
  OpenLog.write(ax_Union.stream, 4);
  OpenLog.write(ay_Union.stream, 4);
  OpenLog.write(az_Union.stream, 4);
  OpenLog.write(gx_Union.stream, 4);
  OpenLog.write(gy_Union.stream, 4);
  OpenLog.write(gz_Union.stream, 4);
}

void cambiarDisp(int sel) {
```

// Si el número del dispositivo es el ultimo, osea si es el sensor 5 y se envió la orden de apagar el sensado (se presione por segunda vez el botón de inicio), entra en el if y procede a apagar el sensado por lo que se finaliza el envío de datos al OpenLog, si solamente se cumple la primer parte solo se reinicia el ciclo de recolección de datos del sensor 5 al sensor 1

```
  if (seleccionador == NumDisp + 1) {
    sel = seleccionador = 1;
    if (peticionApagar) {
      inicioSensores = false;
      peticionApagar = false;
      digitalWrite(LedOn, LOW);
      digitalWrite(LedOff, HIGH);
    }
  }
  switch (sel) {
    case 1:
      myIMU1.cambiarCS(Disp1);
      break;
    case 2:
      myIMU1.cambiarCS(Disp2);
      break;
    case 3:
      myIMU1.cambiarCS(Disp3);
```

```
      break;
    case 4:
      myIMU1.cambiarCS(Disp4);
      break;
    case 5:
      myIMU1.cambiarCS(Disp5);
      break;
  }
  seleccionador++;
}

// Funcion que reinicia el OpenLog para generar el nuevo archivo en el cual guardar los datos
void resetOpengLog() {
  digitalWrite(PinRstOpenLog, LOW);
  delay(250);
  digitalWrite(PinRstOpenLog, HIGH);
  while (OpenLog.available()) {
    letra = OpenLog.read();
    Serial.write(letra);
    if (letra == '<') break;
  }
}

void clearDisplay() {
  Display.write(0x76);  // Comando para limpiar la pantalla
}

void setBrightness(byte value) {
   Display.write(0x7A);  // Comando para indicar el envío del byte que cambiará el brillo
   Display.write(value);  // byte que representa el nivel de brillo que se le dará a la pantalla
}

void setDecimals(byte decimals) {
    Display.write(0x77); // Comando para indicar el envío del byte del mensaje a mostrar
  Display.write(decimals);  // byte que representa el mensaje a mostrar
}
```