INFORME 200 HORAS PRÁCTICA PROFESIONAL

Podómetro para sistema embebido

Practicante: Salvador Catalfamo
Tutor Docente: Gustavo Wolfmann

Contenido

Introducción 2
Objetivos 2
Marco teórico 3
Aceleración y acelerómetros 3
Velocidad Angular y giroscopio 3
IMU: Unidad de medición inercial 4
El acelerómetro 4
El Giroscopio 5
Error en las medidas 5
Comunicación 6
UART (recepción-transmisión asíncrona universal) 6
SPI (Serial Peripheral Interface) 7
I2C 7
Sistema embebido 8
Componentes del proyecto 8
MPU6050 9
ESP32 9
Modos de energía del ESP32: 11
Presentación del dispositivo 16
Software 17
Proceso de lectura de datos 17
Proceso de manipulación de los datos 18
El Conteo de pasos 19
Resultados 19
A futuro 19

Introducción

Un podómetro es un dispositivo, generalmente portátil y electrónico, que cuenta cada paso que realiza una persona al detectar el movimiento de esta. Debido a que la distancia de los pasos de cada persona varía, se requiere una calibración, realizada por el usuario, si se desea la presentación de la distancia recorrida en una unidad de longitud (como en kilómetros o millas), aunque ahora hay podómetros que utilizan dispositivos electrónicos y software para determinar automáticamente cómo varía el paso de una persona. La distancia recorrida (caminando o por cualquier otro medio) se puede medir directamente con un receptor GPS.

Usado originalmente por deportistas, los podómetros ahora se están volviendo populares como un contador y motivador de ejercicios diarios. Usado a menudo en el cinturón y durante todo el día, puede registrar cuántos pasos caminó el usuario ese día y, por lo tanto, los kilómetros o millas (distancia = número de pasos x longitud del paso). Los contadores de pasos pueden animar a competir con uno mismo para ponerse en forma y perder peso. Algunos recomiendan un total de 10,000 pasos por día, equivalente a 8 kilómetros (5,0 mi), como el punto de referencia para un estilo de vida activo, aunque este punto se debate entre los expertos. Treinta minutos de caminata moderada equivalen a 3,000-4,000 pasos según lo determinado por un podómetro. Los contadores de pasos se están integrando en un número creciente de dispositivos electrónicos de consumo portátiles, como reproductores de música y teléfonos inteligentes.

Objetivos

El objetivo principal de este trabajo es, diseñar un sistema el cual muestre a los usuarios, la cantidad de pasos que realiza una persona, para luego, poder tomar decisiones con estos datos.

Del objetivo principal, se derivan otros objetivos los cuales especifican la dirección que se tomará para la solución de la problemática a resolver:

Diseñar un dispositivo electrónico, el cual, contenga las siguientes características:

- Portable: Debe tener un tamaño reducido para que pueda ser colocado en alguna parte del cuerpo.
- Bajo consumo de corriente, de manera que requiera de pilas de bajo voltaje (3 V ~ 3.7 V) o pilas recargables. Se diseñará para que pueda contener pilas recargables, y un cargador a través de un panel solar, de manera que se contribuya al cuidado del medio ambiente
- Conectividad Inalámbrica. Debe ser capaz de comunicarse vía Wifi o Bluetooth con un servidor, para el envío de datos en el caso de que se quiera extender este proyecto.
- Sensor de pasos: Utilizando un sensor de aceleración debe ser capaz de detectar el momento en el cual se realiza un paso.

Marco teórico

Aceleración y acelerómetros

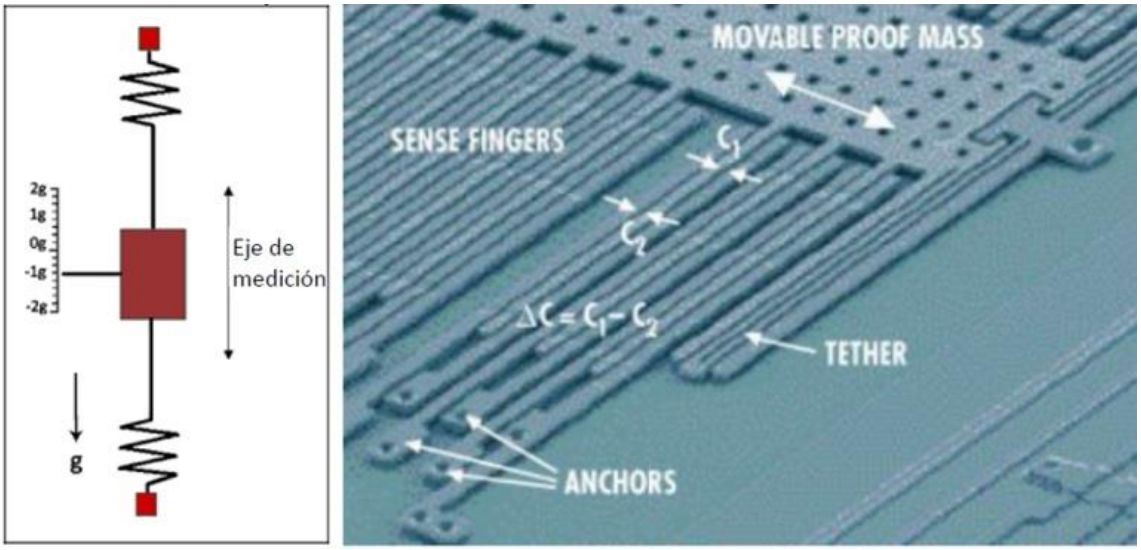
La aceleración es la variación de la velocidad por unidad de tiempo es decir razón de cambio en la velocidad respecto al tiempo:

$$a = \frac{dV}{dt}$$

Así mismo la segunda ley de Newton indica que en un cuerpo con masa constante, la aceleración del cuerpo es proporcional a la fuerza que actúa sobre él mismo:

$$a = \frac{F}{m}$$

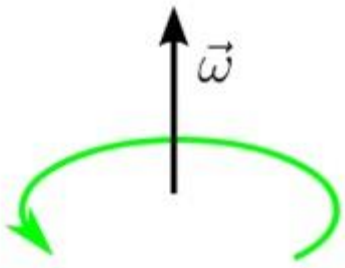
Este segundo concepto es utilizado por los acelerómetros para medir la aceleración. Los acelerómetros internamente tienen un MEMS(MicroElectroMechanical Systems) que de forma similar a un sistema masa resorte permite medir la aceleración.



Con un acelerómetro podemos medir esta aceleración, teniendo en cuenta que a pesar que no exista movimiento, siempre el acelerómetro estará censando la aceleración de la gravedad.

Velocidad Angular y giroscopio

La velocidad angular es la tasa de cambio del desplazamiento angular por unidad de tiempo, es decir que tan rápido gira un cuerpo alrededor de su eje:



$$\omega = \frac{d\theta}{dt}$$

Los giroscopios utilizan un MEMS (MicroElectroMechanical Systems) para medir la velocidad angular usando el efecto Coriolis

IMU: Unidad de medición inercial

Una unidad de medición inercial (IMU) es un dispositivo electrónico que mide e informa la fuerza específica, la velocidad angular y, a veces, la orientación del cuerpo mediante una combinación de acelerómetros, giroscopios y, a veces, magnetómetros.

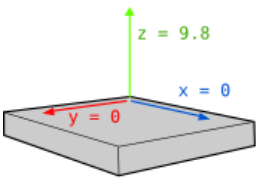
El acelerómetro

El acelerómetro mide la aceleración. La aceleración puede expresarse en 3 ejes: X, Y y Z, las tres dimensiones del espacio. Por ejemplo, si mueves la el módulo hacia arriba, el eje Z marcará un cierto valor. Si es hacia delante, marcará el eje X, etc.

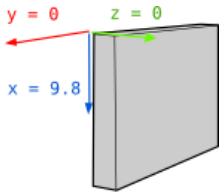
La gravedad de la Tierra tiene una aceleración de aprox. 9.8 m/s², perpendicular al suelo como es lógico. Así pues, la IMU también detecta la aceleración de la gravedad terrestre.

Gracias a la gravedad terrestre puedes usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y.

Supongamos que la IMU esté perfectamente alineada con el suelo. Entonces, como puedes ver en la imagen, el eje Z marcará 9.8, y los otros dos ejes marcarán 0. Ahora supongamos que giramos la IMU 90 grados**.** Ahora es el eje X el que está perpendicular al suelo, por lo tanto, marcará la aceleración de la gravedad.



Paralelo al suelo



Girado 90º

Si sabemos que la gravedad es 9.8 m/s², y sabemos que mesura dan los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la IMU. Una buena fórmula para calcular el ángulo es:

$$AnguloY = \text{atan}\left(\frac{x}{\sqrt{y^2 + z^2}}\right) \qquad AnguloX = \text{atan}\left(\frac{x}{\sqrt{x^2 + z^2}}\right)$$

El Giroscopio

En un principio, los giroscopios eléctricos eran unos voluminosos artefactos que valían la mayor parte del presupuesto militar de un estado. Más tarde, durante la segunda guerra mundial se emplearon para dirigir cohetes y torpedos. Por suerte, gracias la revolución digital y la miniaturización de circuitos, hoy en día cualquier persona puede acceder a uno de estos

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

Si sabemos el ángulo inicial de la IMU, podemos sumarle el valor que marca el giroscopio para saber el nuevo ángulo a cada momento. Supongamos que iniciamos la IMU a 0º. Si el giroscopio realiza una medida cada cierto intervalo de tiempo, y marca un cierto valor en el eje Y, tendremos el ángulo con esta sencilla fórmula:

$$AnguloY = AnguloY\ anterior * GiroscopioY * \Delta t$$

Dónde Δt es el tiempo que transcurre cada vez que se calcula esta fórmula, AnguloYAnterior es el ángulo calculado la última vez que se llamó esta fórmula y GiroscopioY es la lectura actual del ángulo Y del giroscopio.

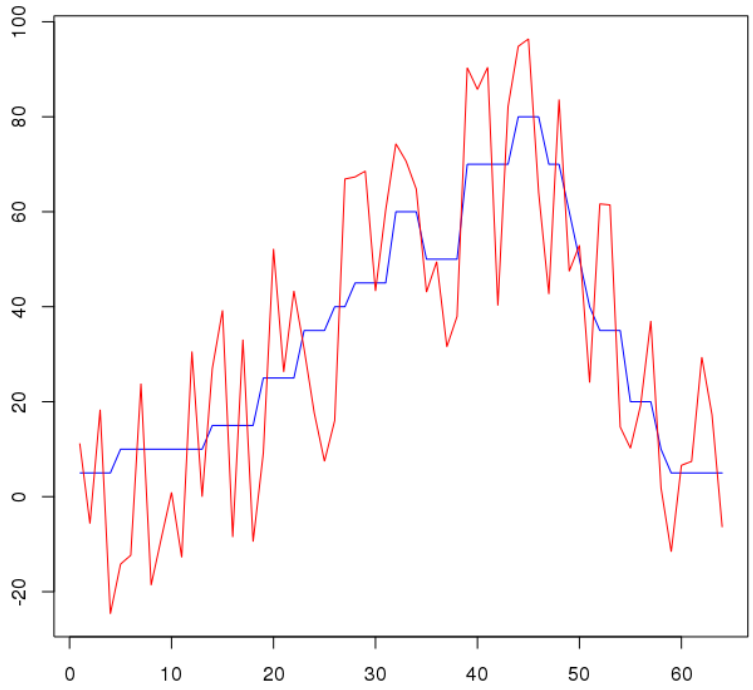
(Por ejemplo: si en el instante t=0 se inicia la IMU a 0º, y al cabo de 0'5 segundos el giroscopio marca 12º, el nuevo ángulo sería 6º)

Error en las medidas

En el mundo real, hay dos problemas muy importantes: el ruido y los errores.

El ruido son todas aquellas interferencias que afectan a los dispositivos electrónicos. El acelerómetro es capaz de medir cualquier ángulo, sin embargo, sus lecturas son ruidosas y tienen un cierto margen de error.

Este es un gráfico con las medidas de un acelerómetro en función del tiempo:



El ángulo real (ideal) está marcado en azul, y las lecturas reales del sensor están en rojo.

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

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

Comunicación

Dentro la comunicación serie integrada en los microcontroladores de Arduino tenemos:

UART (recepción-transmisión asíncrona universal)

Es uno de los protocolos serie más utilizados. La mayoría de los microcontroladores disponen de hardware UART. Usa una línea de datos simple para transmitir y otra para recibir datos. Comúnmente, 8 bits de datos son transmitidos de la siguiente forma: un bit de inicio, a nivel bajo, 8 bits de datos y un bit de parada a nivel alto. UART se diferencia de SPI y I2C en que es asíncrono y los otros están sincronizados con señal de reloj. La velocidad de datos UART está limitado a 2Mbps

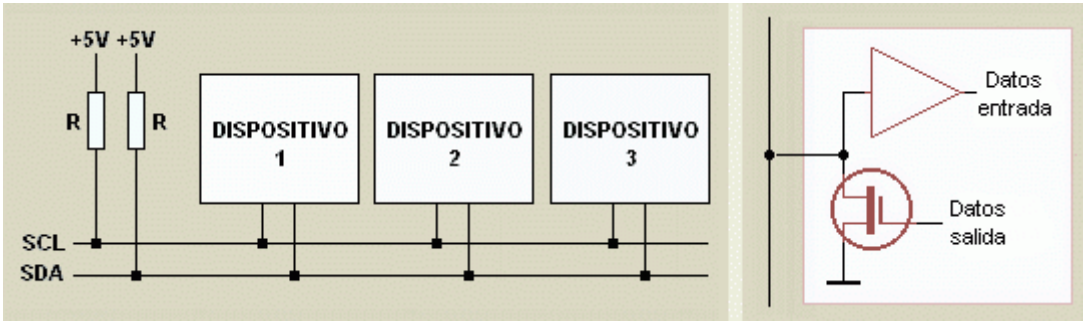
SPI (Serial Peripheral Interface)

Es otro protocolo serie muy simple. Un maestro envía la señal de reloj, y tras cada pulso de reloj envía un bit al esclavo y recibe un bit de éste. Los nombres de las señales son por tanto SCK para el reloj, MOSI para el Maestro Out Esclavo In, y MISO para Maestro In Esclavo Out. Para controlar más de un esclavo es preciso utilizar SS (selección de esclavo).

I2C

Es un protocolo síncrono. I2C usa solo 2 cables, uno para el reloj (SCL) y otro para el dato (SDA). Esto significa que el maestro y el esclavo envían datos por el mismo cable, el cual es controlado por el maestro, que crea la señal de reloj. I2C no utiliza selección de esclavo, sino direccionamiento. La principal característica de I^2C es que utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Como suelen comunicarse circuitos en una misma placa que comparten una misma masa esta tercera línea no suele ser necesaria.

Los dispositivos conectados al bus I2C tienen una dirección única para cada uno. También pueden ser maestros o esclavos. El dispositivo maestro inicia la transferencia de datos y además genera la señal de reloj, pero no es necesario que el maestro sea siempre el mismo dispositivo, esta característica se la pueden ir pasando los dispositivos que tengan esa capacidad. Esta característica hace que al bus I2C se le denomine bus multimaestro.



El proceso de comunicación en el bus I2C es:

- El maestro comienza la comunicación enviando un patrón llamado “start condition”. Esto alerta a los dispositivos esclavos, poniéndolos a la espera de una transacción.
- El maestro se dirige al dispositivo con el que quiere hablar, enviando un byte que contiene los siete bits (A7-A1) que componen la dirección del dispositivo esclavo con el que se quiere comunicar, y el octavo bit (A0) de menor peso se corresponde con la operación deseada (L/E), lectura=1 (recibir del esclavo) y escritura=0 (enviar al esclavo).

- La dirección enviada es comparada por cada esclavo del bus con su propia dirección, si ambas coinciden, el esclavo se considera direccionado como esclavo-transmisor o esclavo-receptor dependiendo del bit R/W.
- Cada byte leído/escrito por el maestro debe ser obligatoriamente reconocido por un bit de ACK por el dispositivo maestro/esclavo.
- Cuando la comunicación finaliza, el maestro transmite una “stop condition” para dejar libre el bus.

Sistema embebido

Un sistema integrado o embebido es un controlador con una función dedicada dentro de un sistema mecánico o eléctrico más grande, a menudo con restricciones de computación en tiempo real. Está integrado como parte de un dispositivo completo que a menudo incluye hardware y piezas mecánicas. Los sistemas integrados controlan muchos dispositivos de uso común hoy en día. El noventa y ocho por ciento de todos los microprocesadores fabricados se utilizan en sistemas integrados.

Los sistemas integrados modernos a menudo se basan en microcontroladores (es decir, CPU con memoria integrada o interfaces periféricas), pero también son comunes los microprocesadores comunes (que usan chips externos para circuitos de interfaz periféricos y de memoria), especialmente en sistemas más complejos. En cualquier caso, los procesadores utilizados pueden ser de varios tipos, desde propósitos generales hasta aquellos especializados en cierta clase de cómputos, o incluso diseñados a medida para la aplicación en cuestión. Una clase estándar común de procesadores dedicados es el procesador de señal digital (DSP).

Dado que el sistema integrado está dedicado a tareas específicas, los ingenieros de diseño pueden optimizarlo para reducir el tamaño y el costo del producto y aumentar la confiabilidad y el rendimiento. Algunos sistemas integrados son producidos en masa, beneficiándose de economías de escala.

Los sistemas integrados abarcan desde dispositivos portátiles como relojes digitales y reproductores de MP3, hasta grandes instalaciones fijas como semáforos, controladores de fábrica y sistemas en gran parte complejos como vehículos híbridos, MRI y aviónica. La complejidad varía de baja, con un solo chip de microcontrolador, a muy alta con varias unidades, periféricos y redes montadas dentro de un chasis o caja grande.

Componentes del proyecto

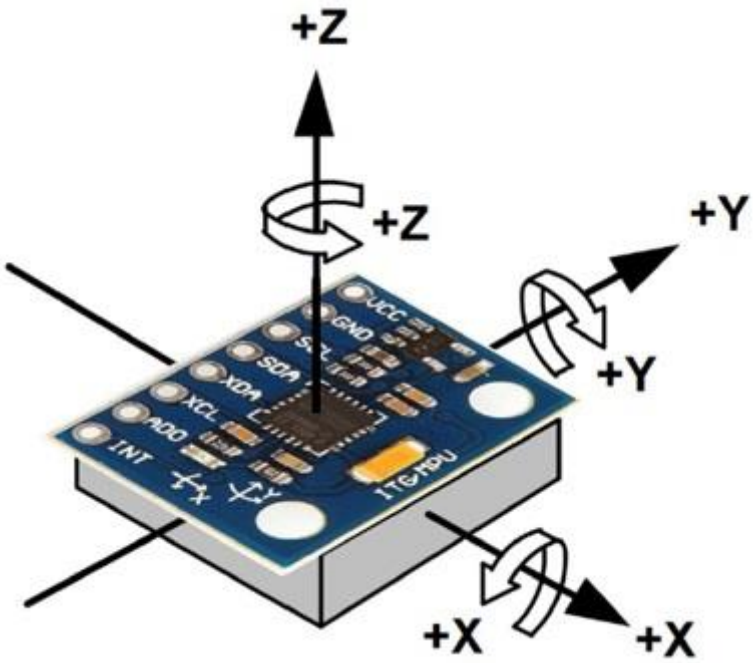
El dispositivo está compuesto por sistema embebido ESP32 y un acelerómetro MPU6050 descriptos a continuación

MPU6050

EL MPU6050 es una unidad de medición inercial o IMU de 6 grados de libertad (DoF) pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría, estabilización, etc.

EL módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y y Z de la aceleración.

La dirección de los ejes está indicada en el módulo el cual hay que tener en cuenta para no equivocarnos en el signo de las aceleraciones.



La comunicación del módulo es por I2C, esto le permite trabajar con la mayoría de microcontroladores. Los pines SCL y SDA tienen una resistencia pull-up en placa para una conexión directa al microcontrolador o Arduino.

ESP32

ESP32 es una serie de sistemas de bajo consumo y en microcontroladores de chip con Wi-Fi integrado y Bluetooth de modo dual. La serie ESP32 emplea un microprocesador Tensilica Xtensa LX6 con variaciones de doble núcleo y de núcleo único e incluye interruptores de antena incorporados, balun RF, amplificador de potencia, amplificador de recepción con bajo nivel de ruido, filtros y módulos de administración de energía. ESP32 es creado y desarrollado por Espressif Systems , una compañía china con sede en

Shanghai, y es fabricado por TSMC utilizando su proceso de 40 nm. [2] Es un sucesor del microcontrolador ESP8266 .

Las características del ESP32 incluyen lo siguiente:

- Procesadores
  - CPU: Xtensa microprocesador LX6 de doble núcleo (o de un solo núcleo) de 32 bits, que funciona a 160 o 240 MHz y tiene un rendimiento de hasta 600 DMIPS
  - Coprocesador de potencia ultra baja (ULP)
- Memoria: 520 KiB SRAM
- Conectividad inalámbrica:
  - Wi-Fi: 802.11 b / g / n
  - Bluetooth: v4.2 BR / EDR y BLE
- Interfaces periféricas:
  - ADC SAR de 12 bits hasta 18 canales
  - DAC de 2 × 8 bits
  - 10 sensores táctiles (GPIOs de detección capacitiva)
  - 4 × SPI
  - 2 × interfaces I^2S
  - 2 × interfaces I^2C
  - 3 × UART
  - Controlador de host SD / SDIO / CE-ATA / MMC / eMMC
  - Controlador esclavo SDIO / SPI
  - Interfaz MAC de Ethernet con DMA dedicado y compatibilidad con IEEE 1588 Precision Time Protocol
  - Bus CAN 2.0
  - Control remoto por infrarrojos (TX / RX, hasta 8 canales)
  - Motor PWM
  - LED PWM (hasta 16 canales)
  - Sensor de efecto hall
  - Preamplificador analógico de ultra bajo consumo.
- Seguridad:
  - Funciones de seguridad estándar IEEE 802.11 todas compatibles, incluyendo WFA, WPA / WPA2 y WAPI
  - Arranque seguro
  - Cifrado flash
  - OTP de 1024 bits, hasta 768 bits para clientes
  - Aceleración de hardware criptográfico: AES , SHA-2 , RSA , criptografía de curva elíptica (ECC), generador de números aleatorios (RNG)
- Gestión de energía:
  - Regulador interno de baja caída
  - Dominio de poder individual para RTC
  - 5µA corriente de sueño profundo
  - Despertar desde la interrupción GPIO, el temporizador, las mediciones de ADC, la interrupción del sensor táctil capacitivo

Modos de energía del ESP32:

Cuando un proyecto de esta clase es alimentado por un enchufe en la pared, uno tiende a no preocuparse demasiado por el consumo de energía. Pero si va a alimentar su proyecto con baterías, cada mA cuenta.

La solución aquí es reducir el uso de energía de ESP32 al aprovechar uno de sus modos de suspensión . Es realmente una excelente estrategia para extender dramáticamente la vida útil de la batería de un proyecto que no necesita estar activo todo el tiempo.

¿Qué es un modo de suspensión ESP32?

El modo de suspensión ESP32 es un estado de ahorro de energía que ESP32 puede ingresar cuando no está en uso. El estado del ESP32 se mantiene en la memoria RAM. Cuando ESP32 ingresa al modo de suspensión, la alimentación se corta a cualquier periférico digital innecesario, mientras que la RAM recibe la potencia suficiente para permitirle retener sus datos.

En este proyecto se utilizaron funciones de ahorro de energía, para esto, se describirán los distintos modos:

Para entender cómo el ESP32 logra un ahorro de energía, necesitamos saber qué hay dentro del chip. La siguiente ilustración muestra el diagrama de bloques de funciones del chip ESP32.

En el corazón del chip ESP32 se encuentra un microprocesador de doble núcleo de 32 bits, junto con 448 KB de ROM, 520 KB de SRAM y 4 MB de memoria Flash.

También contiene un módulo Wifi, un módulo Bluetooth, un acelerador criptográfico (un coprocesador diseñado específicamente para realizar operaciones criptográficas), el módulo RTC y muchos periféricos.

Modos de potencia ESP32

Gracias a la gestión avanzada de energía del ESP32, ofrece 5 modos de energía configurables. Según el requisito de potencia, el chip puede cambiar entre diferentes modos de potencia. Los modos son:

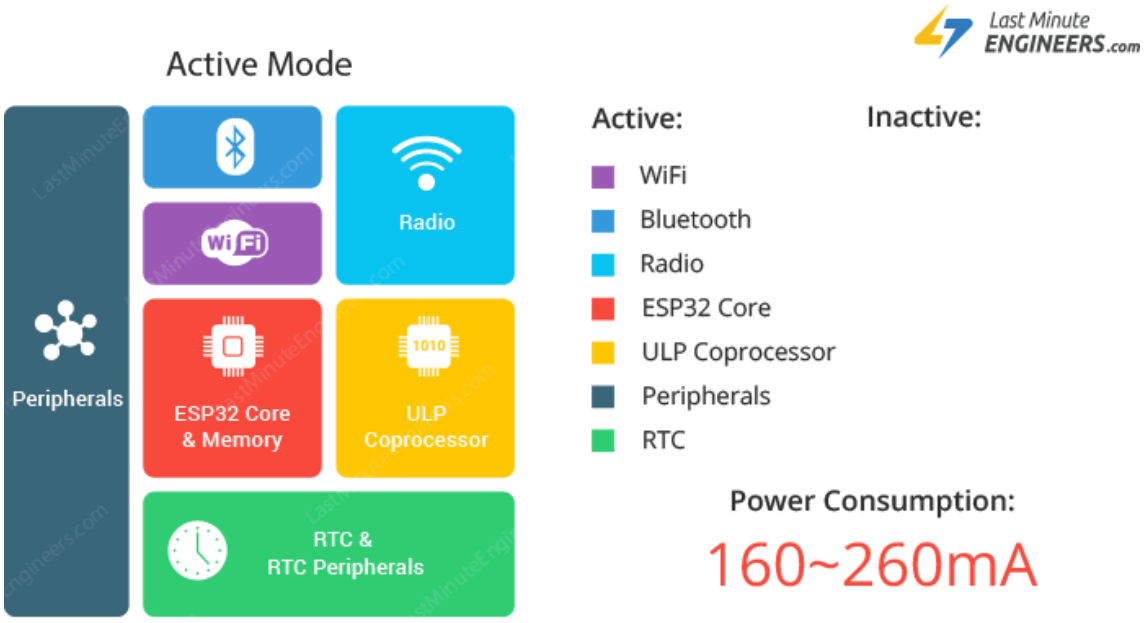
- Modo activo
- Modem Sleep Mode
- Modo de sueño ligero
- Modo de sueño profundo
- Modo de hibernación

Cada modo tiene sus propias características distintas y capacidades de ahorro de energía. Echemos un vistazo a ellos uno por uno.

Modo Activo ESP32

El modo normal también se conoce como modo activo. En este modo todas las características del chip están activas.

Como el modo activo mantiene todo activado (especialmente el módulo Wifi, los núcleos de procesamiento y el módulo Bluetooth) en todo momento, el chip requiere una corriente de más de 240 mA para funcionar. También observamos que, si se usan las funciones Wifi y Bluetooth juntas, a veces aparecen picos de alta potencia (el mayor fue 790mA).

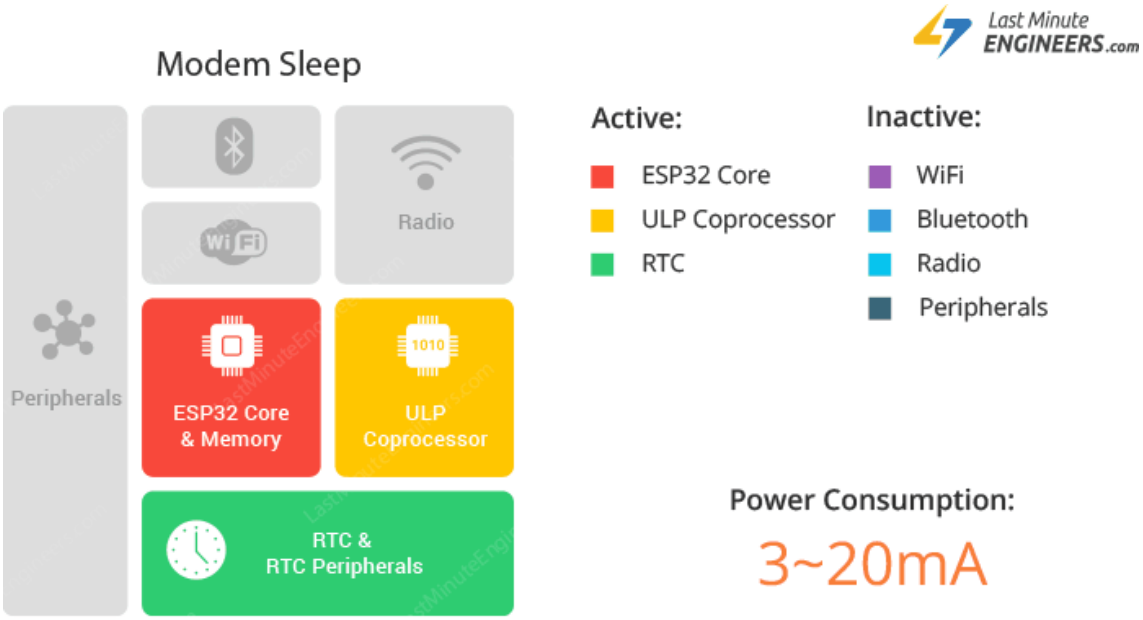


Obviamente, este es el modo más ineficiente y agotará la batería más rápidamente. Por lo tanto, si queremos ahorrar energía, tenemos que deshabilitarlos (aprovechando uno de los otros modos de energía) cuando no estamos en uso.

ESP32 Modem Sleep

En el modo de suspensión por módem, todo está activo mientras que solo Wifi, Bluetooth y los puertos periféricos están desactivados. La CPU también está operativa y el reloj es configurable.

En este modo, el chip consume alrededor de 3 mA a baja velocidad y 20 mA a alta velocidad.



Para mantener vivas las conexiones Wifi / Bluetooth, la CPU, Wi-Fi, Bluetooth y los periféricos se activan a intervalos predefinidos. Se le conoce como patrón de sueño de asociación **.**

Durante este patrón de reposo, el modo de energía cambia entre el modo activo y el modo de reposo del módem.

ESP32 Light Sleep

El modo de funcionamiento del sueño ligero es similar al del modem sleep. El chip también sigue el patrón de sueño de asociación.

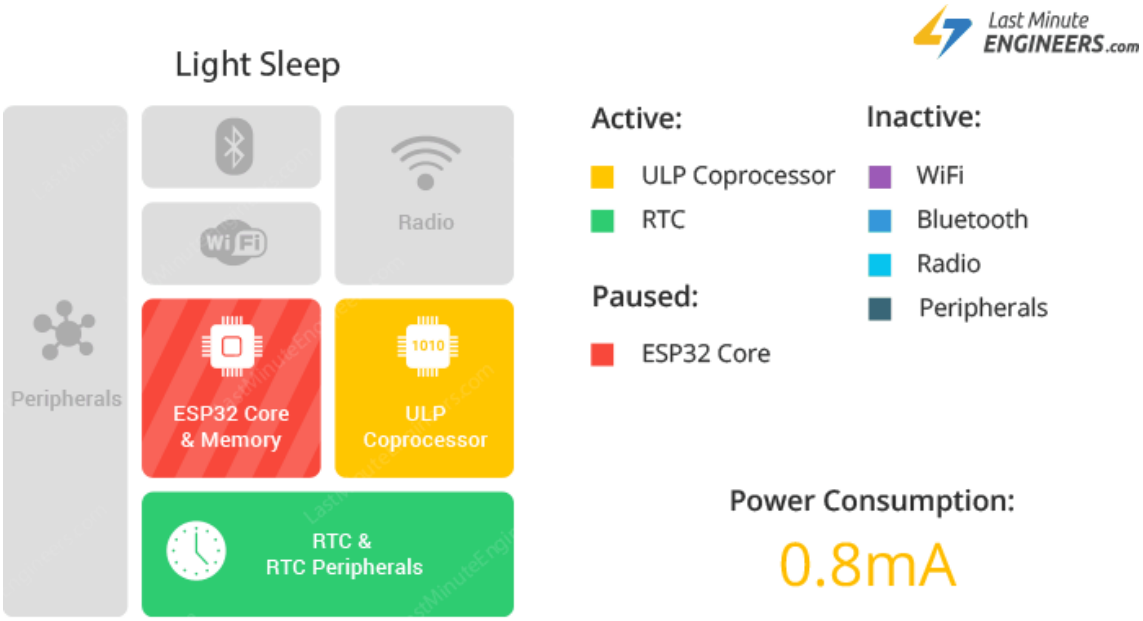
La diferencia es que, durante el modo Light Sleep, los periféricos digitales, la mayoría de la memoria RAM y la CPU están sincronizados.

¿Qué es Clock Gating?

La activación del reloj es una técnica para reducir el consumo dinámico de energía.

Desactiva partes del circuito apagando los pulsos del reloj, de modo que los flip-flops en ellos no tienen que cambiar de estado. A medida que los estados de conmutación consumen energía, cuando no se cambian, el consumo de energía se reduce a cero.

Durante el modo de Light Sleep, la CPU se detiene pausando sus pulsos de reloj, mientras que RTC y el coprocesador ULP se mantienen activos. Esto resulta en un menor consumo de energía que en el modo de suspensión de módem, que es de alrededor de 0,8 mA.

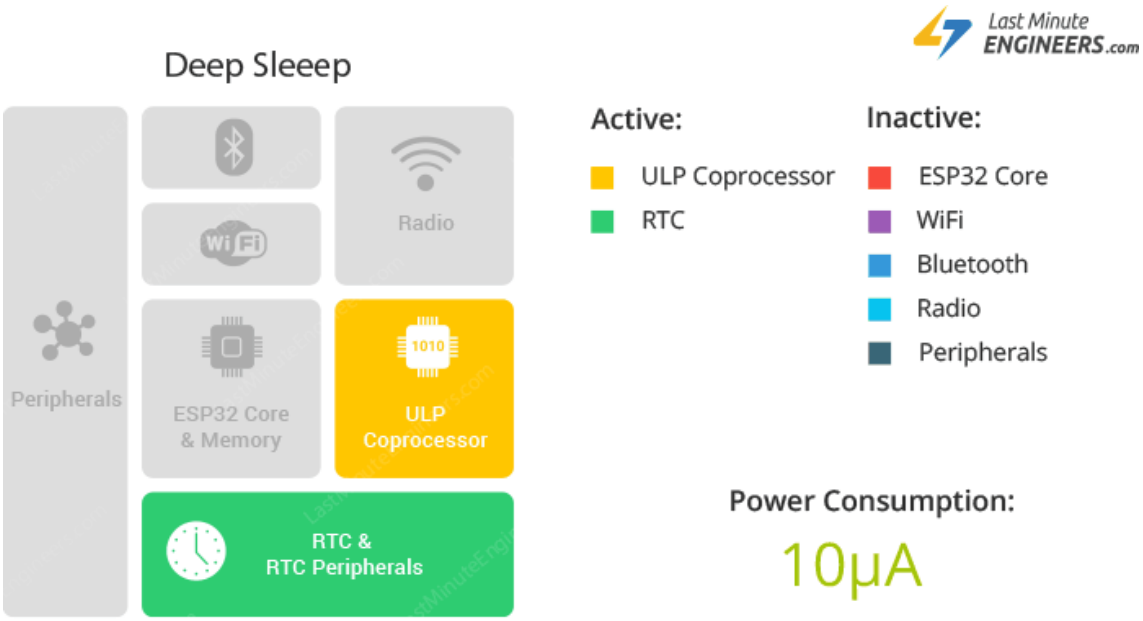


Antes de ingresar al modo de Light Sleep, el ESP32 conserva su estado interno y reanuda la operación al salir del modo de espera. Se conoce la **retención de RAM completa** .

ESP32 Deep Sleep (sueño profundo)

En el modo de suspensión profunda, la CPU, la mayoría de la RAM y todos los periféricos digitales están apagados. Las únicas partes del chip que permanecen encendidas son: controlador RTC, periféricos RTC (incluido el coprocesador ULP) y memorias RTC (lentas y rápidas).

El chip consume alrededor de 0.15 mA (si el coprocesador ULP está encendido) a 10µA.



Durante el modo de suspensión profunda, la CPU principal se apaga, mientras que el coprocesador ULP realiza mediciones del sensor y activa el sistema principal, según los datos medidos de los sensores. Este patrón de sueño se conoce como patrón controlado por sensor ULP **.**

Junto con la CPU, la memoria principal del chip también está deshabilitada. Entonces, todo lo almacenado en esa memoria se borra y no se puede acceder.

Sin embargo, la memoria RTC se mantiene encendida. Por lo tanto, su contenido se conserva durante el sueño profundo y se puede recuperar después de que activemos el chip. Esa es la razón, el chip almacena los datos de conexión de Wi-Fi y Bluetooth en la memoria RTC antes de deshabilitarlos.

Por lo tanto, si desea utilizar los datos durante el reinicio, guárdelos en la memoria RTC definiendo una variable global con el atributo RTC_DATA_ATTR . Por ejemplo, RTC_DATA_ATTR int bootCount = 0;

En el modo de suspensión profunda, la alimentación se desconecta de todo el chip, excepto el módulo RTC. Por lo tanto, cualquier dato que no se encuentre en la memoria de recuperación de RTC se pierde y el chip se reiniciará con un reinicio. Esto significa que la ejecución del programa comienza desde el principio una vez más.

Este último modo de energía es el que se utilizó para este proyecto, de manera que mientras no se estén realizando pasos, el módulo consuma el mínimo de energía

Estimación de la duración de la batería si estuviera alimentado por dos pilas AA (Capacidad típica: 2550-3000 mAh multiplicado por 2)

El tiempo de descarga viene dado por la expresión:

$$tiempo\ de\ descarga = \frac{carga\ electrica\ bateria}{consumo\ electrico\ dispositivo}$$

En este ejemplo se tomará a una persona que camina durante 3 horas por día. Además, se asumirá que el consumo del módulo giroscopio es despreciable.

Durante el estado activo el módulo consume entre 160 y 260 mah, tomaremos el promedio de estos valores: 210 mah

$$\frac{carga\ electrica\ bateria}{consumo\ electrico\ dispositivo} = \frac{5100\ mah}{210\ mah} = 26\ horas$$

Durante el estado Deep Sleep el módulo consume 0,15 mah

$$\frac{carga\ electrica\ bateria}{consumo\ electrico\ dispositivo} = \frac{5100\ mah}{0{,}15\ mah} = 34000\ \text{horas}$$

Con estos valores teóricos, el consumo está delimitado por el ESP en modo activo, por lo que, estando tres horas diarias en modo activo, dos pilas AA podrían durar 8 días aproximadamente. Próximamente, se trabajará en el consumo del módulo en modo activo, para poder mantener el dispositivo durante mas días encendido

Presentación del dispositivo

A continuación, se muestran las conexiones realizadas para este podómetro:

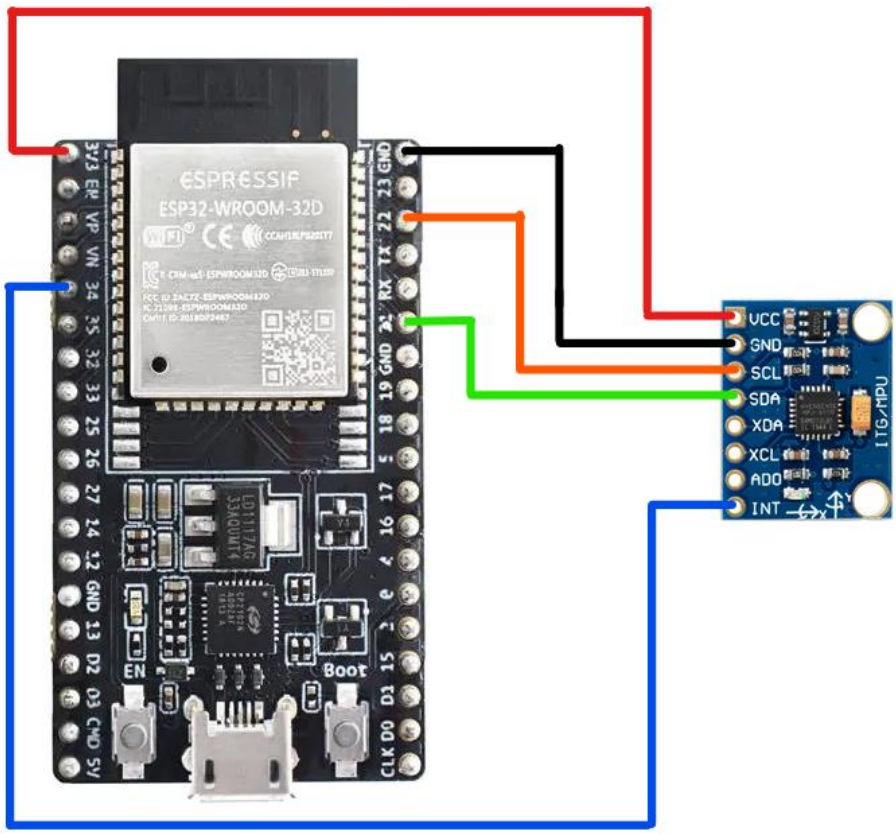
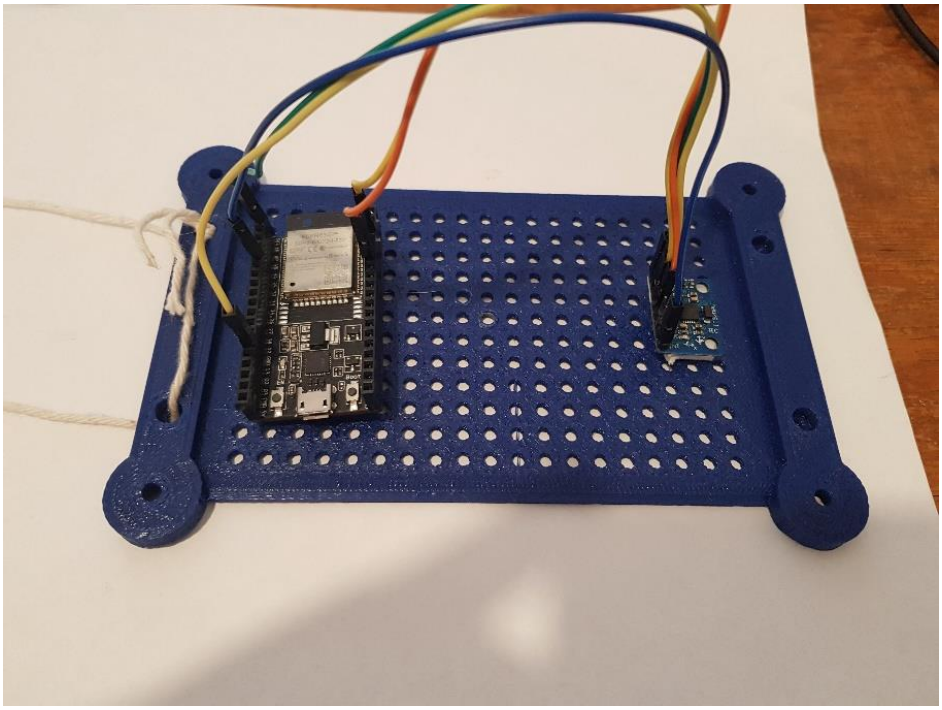


Imagen real del proyecto:



Software

Proceso de lectura de datos

Para la lectura de los datos del acelerómetro y giroscopio se realizan los siguientes pasos:

- Inicialmente como se indico es necesario incluir las librerías: En nuestro proyecto utilizamos:
    - Librería de comunicación mediante el protocolo I2C
    - Librería para la obtención de datos con el módulo MPU-6050: cabe aclarar que la librería utilizada es una composición de dos librerías para el módulo, ya que con una se realizan las mediciones, y con otra se generan los interrupciones para despertar al ESP32
- Se inicia la dirección I2C que se utilizara para la comunicación. En este proyecto se utilizó la dirección 0x68
- Posteriormente inicializamos tanto la comunicación I2C como el MPU6050 y en nuestro caso la comunicación serial (utilizada para realizar debugging)
- Al inicializar el sensor, los rangos por defecto serán:
    - acelerómetro -2g a +2g
    - Giroscopio: -250°/sec a +250°/sec

      La resolución de las lecturas es de 16 bits por lo que el rango de lectura es de -32768 a 32767. En este proyecto solo se utilizaron los datos del acelerómetro

- En el programa principal realizamos las lecturas y las guardamos en sus variables respectivas, para luego hacer el analisis y el conteo de los pasos

Proceso de manipulación de los datos

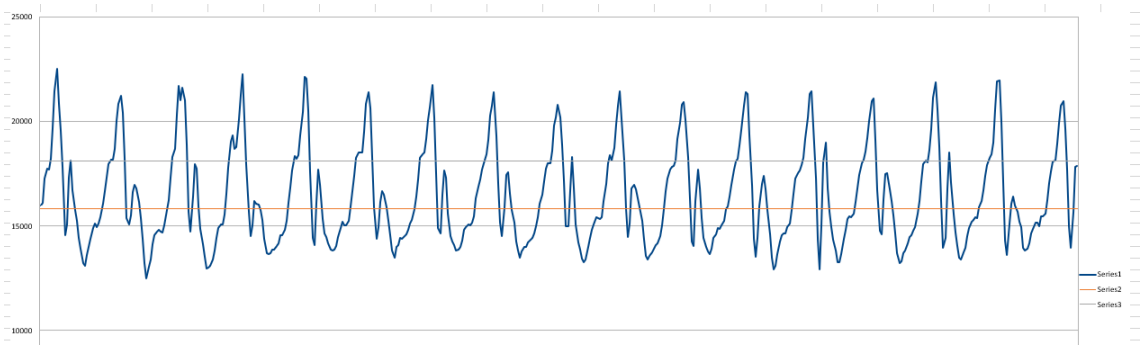
El acelerómetro entrega tres datos:

- La aceleración para el eje x (Ax)
- La aceleración para el eje y (Ay)
- La aceleración para el eje z (Az)

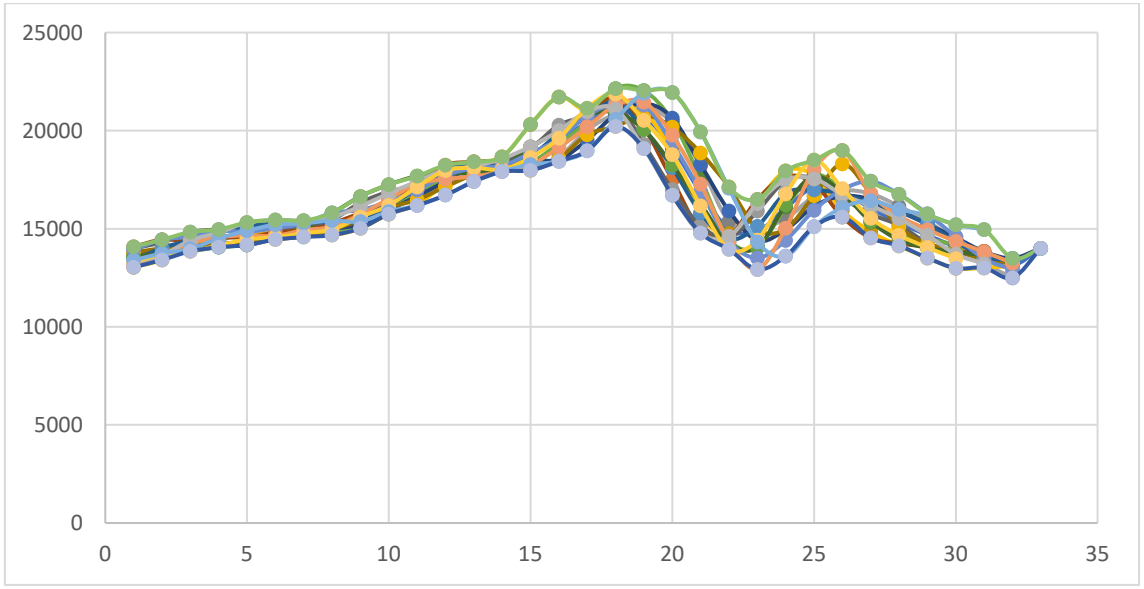
Si solo se trabajaran con estas componentes por separado, la detección de pasos se vería afectada por la orientación del módulo, por lo que se decidió trabajar con el módulo de estas componentes, esto es:

$$A = \sqrt{{A_x}^2 + {A_y}^2 + {A_z}^2}$$

Utilizando esta fórmula, se recolectaron datos durante una cierta cantidad de pasos para detectar el patrón de paso. Estos son los datos que se obtuvieron:



Este grafico da muestra que los pasos están bien determinados, y, haciendo un análisis completo de estos datos se logró establecer como está compuesto el “patrón paso”. A continuación, se muestra este patrón, a través de una cierta cantidad de pasos tomados



El Conteo de pasos

1- Inicialmente se hace una configuración inicial y se realizan lecturas del sensor, si no hay movimientos se ingresa en un estado “Deep sleep”
2- Cuando se detecta un movimiento, el sensor MPU6050 envía una interrupción para despertar al ESP32, ahí es cuando se inicia el conteo de pasos.
3- Inicialmente, el sistema se encuentra en un estado “reposo”: en este estado, si se descarta un paso, el ESP32 se duerme y se debe iniciar el proceso nuevamente
4- Cuando se cumplen 5 pasos, el sistema cambia a un estado “caminando”: en este estado, cuando se descarta un paso el ESP32 sigue despierto, esperando el inicio de pasos
5- Cuando se cumple un cierto tiempo sin realizar pasos, el módulo entra en suspensión, y se cambia al estado “reposo”

Resultados

Se obtuvieron los resultados esperados para este proyecto, el conteo de pasos se realiza de manera óptima en la mayoría de los casos, y con distintas posiciones del acelerómetro

Dado que el inicio del módulo ESP32 demora aproximadamente 1 segundo, es posible que se pierdan hasta 2 pasos iniciales, si es que la persona está parada, en el caso de que la persona este sentada, la perdida de pasos es menor

A futuro

Las ventajas de este proyecto, es que se puede extender a varios usos, además agregar nuevas funcionalidades. Unas posibles funcionalidades se enumeran a continuación:

- Alimentación mediante batería y utilización de paneles solares para la carga de estas
- Optimización del uso del ESP en estado “activo”
- Almacenamiento de datos en algún medio masivo, como por ejemplo una memoria SD
- Transmisión de datos a la nube para el análisis, conteo de calorías perdidas, predicciones, etc.

Personalmente, espero utilizar esta base para extenderlo a proyecto final de carrera, utilizando inteligencia artificial para la detección del patrón paso, y así poder aumentar la precisión y la cantidad de patrones posibles. Está claro que el módulo ESP no es apto para operar todo un sistema de inteligencia artificial, por lo que también está planeado el envío de datos a un servidor que pueda procesar toda esta información