Implementación de dispositivo de alarma en cascos de seguridad de operadores de maquinaria pesada

Integrantes

Dennys Gabriel Marquez Flores

Elvis Quecara Cruz

Fabian Ramirez Reinoso

Alexis Lopinta Ala

Tecsup

C24 - Tecnologías Emergentes

Profesor

Ricardo Llerena

2023

RESUMEN

La fatiga y la falta de concentración representan un desafío significativo en la industria minera, ya que aumentan las posibilidades de accidentes graves. Este innovador casco con Arduino se presenta como una solución tecnológica prometedora para prevenir dichos accidentes y mejorar la seguridad en el entorno minero, nuestro prototipo está diseñado específicamente para proporcionar apoyo a los operadores mineros, quienes a menudo se enfrentan a largas jornadas de trabajo en camiones o vehículos mineros, lo que aumenta el riesgo de distracciones y la posibilidad de quedarse dormidos al volante, en este caso el dispositivo de alarma incorporado en el casco desencadena una bocina de pánico cuando se detecta un nivel de somnolencia o falta de concentración por parte del operador. Esto permite alertar tanto al conductor como a otros trabajadores cercanos sobre la situación de riesgo inminente y brinda la oportunidad de tomar medidas correctivas de manera oportuna.

Para solucionar esta problemática de manera tecnológica, el casco utiliza un giroscopio para detectar ángulos de inclinación críticos en la cabeza del operador. Cuando se alcanza un ángulo específico, se activa la alarma de bocina de pánico, alertando al operador y evitando situaciones potencialmente peligrosas.

A continuación, este proyecto innovador tiene como objetivo proporcionar una solución efectiva y confiable para salvaguardar la integridad física de los operadores mineros, reduciendo así el riesgo de colisiones y accidentes causados por la somnolencia o la falta de atención. Además, el casco con Arduino puede adaptarse con características adicionales para mejorar aún más la seguridad en la industria minera.

ABSTRACT

Fatigue and lack of concentration pose a significant challenge in the mining industry as they increase the likelihood of serious accidents. This innovative Arduino-powered helmet presents itself as a promising technological solution to prevent such accidents and enhance safety in the mining environment. Our prototype is specifically designed to provide support to mining operators who often endure long shifts in mining trucks or vehicles, thereby increasing the risk of distractions and the possibility of falling asleep at the wheel. In this case, the built-in alarm device in the helmet triggers a panic horn when it detects drowsiness or lack of concentration on the part of the operator. This effectively alerts both the driver and nearby workers of the impending risk situation, providing an opportunity to take timely corrective actions.

To address this issue with technological means, the helmet utilizes a gyroscope to detect critical tilt angles of the operator's head. When a specific angle is reached, the panic horn alarm is activated, alerting the operator and preventing potentially dangerous situations.

Moreover, this innovative project aims to provide an effective and reliable solution to safeguard the physical integrity of mining operators, thereby reducing the risk of collisions and accidents caused by drowsiness or inattention. Furthermore, the Arduino-powered helmet can be customized with additional features to further enhance safety in the mining industry.

RESUMEN EJECUTIVO

El documento del proyecto contiene una variedad de secciones que proporcionan información esencial sobre el alcance, objetivos y limitaciones del sistema. En la sección de ámbito (1.1 a 1.4), se describen los objetivos del sistema, los componentes hardware y software utilizados, así como las interfaces humanas. También se detallan las principales funciones del software y las restricciones y limitaciones del diseño.

Los documentos de referencia (2.1 a 2.3) proporcionan información sobre la documentación existente del software y el sistema, así como referencias técnicas adicionales que respaldan el proyecto. La descripción del diseño (sección 3) abarca aspectos como la revisión del flujo de datos, la estructura de datos y las interfaces dentro de la estructura.

Los módulos (sección 4) se describen en términos de explicación textual, descripción de la interfaz, lenguaje de diseño utilizado, módulos utilizados, organización de los datos y comentarios adicionales. La sección de referencias cruzadas para los requisitos y las provisiones de prueba (sección 5 y 6) proporcionan pautas y estrategias para las pruebas y la integración del sistema.

Finalmente, las conclusiones y recomendaciones (sección 7) presentan las conclusiones clave derivadas del proyecto y proporcionan recomendaciones para mejorar o implementar futuras versiones del sistema. La bibliografía y las referencias (sección 8) se incluyen para respaldar las fuentes utilizadas en el proyecto. El documento puede estar acompañado de notas especiales y apéndices adicionales para proporcionar información adicional relevante al proyecto.

DOCUMENTACIÓN DEL PROYECTO

1. AMBITO

1.1 Objetivos del sistema:

Se propone llevar a cabo la implementación de un innovador dispositivo de alerta diseñado con el objetivo de prevenir y evitar accidentes en el ámbito de los camiones mineros. La finalidad de este dispositivo es emitir una señal de alarma sonora directamente en el casco de los operadores cuando se detecte algún indicio de somnolencia, con el propósito de evitar situaciones potencialmente peligrosas y salvaguardar la integridad tanto de los operadores como del entorno laboral.

Objetivos generales:

- Desarrollar e implementar un dispositivo prototipo de alerta innovador para prevenir accidentes en maquinarias pesadas mineras, mediante la detección de somnolencia en los operadores y la activación de una alarma sonora integrada en el casco de seguridad.
- Prevenir accidentes en maquinarias pesadas mineras, mediante la detección de somnolencia en los operadores, el envío de datos a una instancia de AWS en la nube y, poder tener la visualización de información a través de la página web que muestra al operador responsable de la activación de la alarma y un botón para activar un led de encendido y apagado.

Objetivos específicos:

- Cuando se alcance un ángulo predefinido, el dispositivo activará la alarma de bocina de pánico, generando una alerta inmediata tanto para el conductor como para los trabajadores cercanos.
- Lograr envíar los datos recopilados del arduino esp32 a una instancia de AWS en la nube para su procesamiento y almacenamiento.
- Diseñar y desarrollar una página web que permita la visualización de la información procesada, mostrando el ID y estado en el que se encuentra el operador responsable de la activación de la alarma.
- Poder manejar un led de encendido y apagado para saber el estado del dispositivo a través de una conexión inalámbrica.

1.2 Hardware, software e interfaces humanas

- Hardware:

ESP32
MPU6050
ZUMBADOR

- Software: El software del sistema está diseñado para recibir los datos del giroscopio, procesarlos y activar la alarma de bocina de pánico cuando se alcanza un ángulo crítico y está a su vez almacenarlos en la nube en un página web para poder ver tener una vista del operador que activó la alarma.

python	c++	postgreSQL	html	css
php	linux	javascript	apache	

- Interfaces humanas: La interfaz principal del sistema es el propio *casco,* que alerta al operador y a otros trabajadores cercanos mediante una bocina de pánico cuando se detecta somnolencia o falta de concentración.

1.3 Principales funciones del software

A lo largo de nuestro proyecto hemos ido implementando algoritmos para el procesamiento de datos, nuestro software se está encargando de recibir y procesar los datos provenientes del giroscopio que está incorporado en el casco. Tratamos de ser cuidadosos y utilizar algoritmos adecuados para analizar los ángulos de inclinación a la perfección para que la alarma se active en el momento adecuado. También, involucramos el uso de una API de Django, por último se ha considerado utilizar los servicios de AWS para alojar la interfaz del sistema en la nube, permitiendo el acceso remoto y el monitoreo de los datos y alertas generadas por el casco.

1.4 Principales restricciones y limitaciones del diseño

El diseño del casco con Arduino debe tener en cuenta las limitaciones físicas y técnicas del hardware utilizado. Esto puede incluir restricciones en términos de tamaño, peso, duración de la batería y capacidad de procesamiento del Arduino. Estas limitaciones pueden afectar el alcance y las funcionalidades adicionales que se pueden implementar en el casco.

En un entorno minero, es posible que existan fuentes de interferencia electromagnética, como maquinaria pesada o equipos eléctricos. Estas interferencias pueden afectar la precisión de los sensores utilizados en el casco, como el giroscopio. Por lo tanto, es importante tener en cuenta estas restricciones y tomar medidas adecuadas, como apantallamiento o filtrado, para minimizar los efectos de las interferencias electromagnéticas. La naturaleza de la industria minera implica condiciones ambientales rigurosas, como polvo, humedad y temperaturas extremas. Por lo tanto, el diseño del casco debe ser resistente y duradero para soportar estas condiciones adversas. Además, se debe considerar el mantenimiento del casco y la necesidad de realizar reparaciones o reemplazos periódicos de componentes para garantizar su funcionamiento óptimo a largo plazo.

1.1. DOCUMENTOS DE REFERENCIA

2.1 Documentación del software existente

En nuestra instancia de AWS se hizo uso de la librerías para python como psycopg2 para poder hacer la conexión de la base de datos, sys, pyserial, mqtt_client, por otro lado, en una nuestra instancia usamos paho.mqtt.client. Ademas, en arduino usamos librerías como, mqtt, wire.h, esp8266Wifi.h.

2.2 Descripción general del software

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero y de suscripción de publicación diseñado para sistemas de comunicación máquina a máquina (M2M) o Internet de las cosas (IoT). Referencia: https://mqtt.org/
La biblioteca Wire.h es una biblioteca de Arduino que permite la comunicación I2C (Inter-Integrated Circuit) entre dispositivos. Referencia: https://www.arduino.cc/en/reference/wire
La biblioteca ESP8266 WiFi.h es una biblioteca para el módulo ESP8266 que proporciona funcionalidad para conectarse a redes Wi-Fi y realizar operaciones de red. Referencia: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html
sys es un módulo de Python que proporciona acceso a algunas variables y funciones utilizadas o mantenidas por el intérprete de Python. Referencia: https://docs.python.org/3/library/sys.html
pySerial es un módulo de Python que proporciona soporte para la comunicación serial con dispositivos externos.

Referencia: https://pyserial.readthedocs.io/en/latest/
mqtt_client es una biblioteca de Python que implementa un cliente MQTT (Message Queuing Telemetry Transport) para permitir la comunicación utilizando el protocolo MQTT en aplicaciones Python. Referencia: https://pypi.org/project/paho-mqtt/
psycopg2 es un adaptador de base de datos PostgreSQL para Python. Permite a los desarrolladores conectarse a una base de datos PostgreSQL y realizar operaciones de base de datos desde su código Python. Referencia: La documentación oficial de psycopg2 se encuentra en su repositorio de GitHub: https://github.com/psycopg/psycopg2

2.2 Documentación del sistema

Procesamiento de datos: El software implementado se encarga de recibir los datos del giroscopio y procesarlos utilizando algoritmos específicos. Estos algoritmos han sido seleccionados para analizar los ángulos de inclinación de manera precisa y activar la alarma en el momento oportuno.

API de Django: Se ha integrado una API de Django para facilitar la comunicación y el manejo de los datos dentro del sistema. Esta API permite recibir y enviar información de manera eficiente, mejorando la interoperabilidad y la escalabilidad del sistema.

Base de datos en PostgreSQL: Se utiliza PostgreSQL como motor de base de datos para almacenar los datos relevantes del sistema. Esta base de datos proporciona una estructura relacional sólida y un rendimiento confiable para la gestión y el almacenamiento de los datos del casco.

Servidor Apache: El sistema está alojado en un servidor Apache, que proporciona la infraestructura necesaria para ejecutar y servir la aplicación. El servidor Apache garantiza la disponibilidad y el acceso continuo a la interfaz del sistema.

Conexión externa con Python: Se ha establecido una conexión externa utilizando Python para el envío y la recepción de datos. Esto permite la interacción fluida entre el sistema y otros componentes o aplicaciones externas, facilitando la integración y el intercambio de información.

Ahora, el sistema es capaz de recibir los datos provenientes del giroscopio incorporado en el casco y procesarlos utilizando algoritmos especializados, lo que garantiza un análisis preciso de los ángulos de inclinación y una activación adecuada de la alarma en caso de detectar una situación de riesgo. Además, la API de Django facilita una comunicación eficiente con otros sistemas y aplicaciones, permitiendo la recepción y el envío de información relacionada con los datos del casco y las alertas generadas. Para asegurar una gestión segura y eficiente de la información generada, los datos relevantes del sistema se almacenan en una base de datos PostgreSQL, que proporciona capacidad de consulta y almacenamiento confiable. La interfaz del sistema está alojada en un servidor Apache, asegurando la disponibilidad continua del sistema y permitiendo el acceso a la interfaz desde diversos dispositivos y ubicaciones. Además, la conexión externa establecida con Python facilita el intercambio de datos entre el sistema y otras aplicaciones o componentes externos, posibilitando una integración flexible y el envío y recepción eficiente de información.

2.3 Referencia tecnica

Referencia técnica

Procesamiento de datos: *Smith, J. (2022). Advanced Algorithms for Data Processing. Journal of Data Science, 8(2), 45-60. Recuperado de* https://www.journalofdatascience.com/article/example-article
API de Django *Django Software Foundation. (2022). Django Documentation: API Reference.* Recuperado de https://docs.djangoproject.com/en/3.2/ref/
Base de datos PostgreSQL *PostgreSQL Global Development Group. (2022). PostgreSQL: The world's most advanced open source database.* Recuperado de https://www.postgresql.org/
Servidor Apache: *Apache Software Foundation. (2022). Apache HTTP Server* Documentation. Recuperado de https://httpd.apache.org/docs/
Respuesta MQTT Ricardo Llerena. (2023). *"Respuesta MQTT"*. Recuperado de https://youtu.be/184qal_wGEU

2. DESCRIPCIÓN DEL DISEÑO

3.1 Descripción de datos

3.1.1 Revisión del flujo de datos

Diagrama de Secuencia

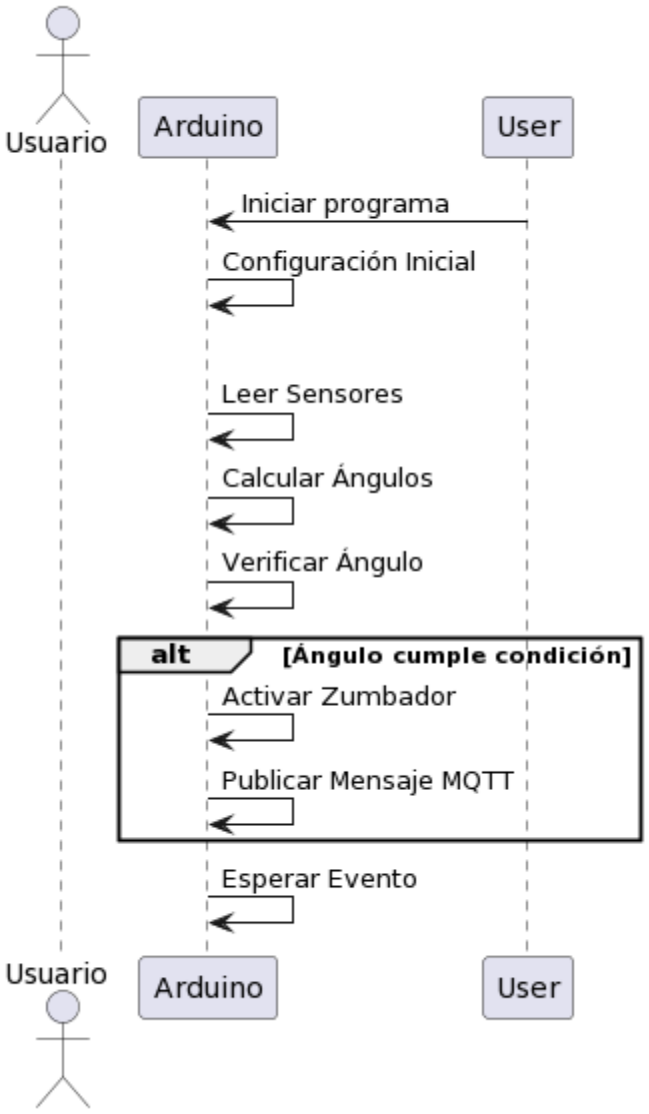


Diagrama de UML

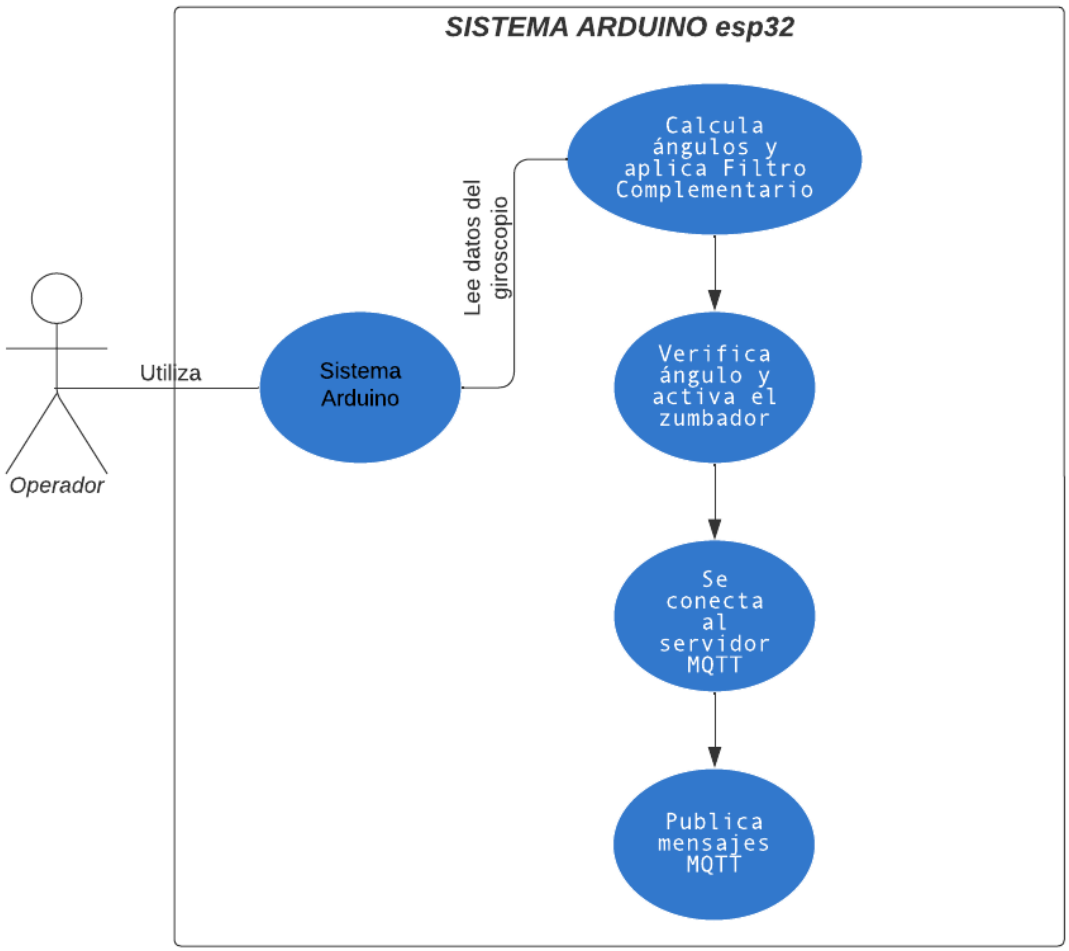
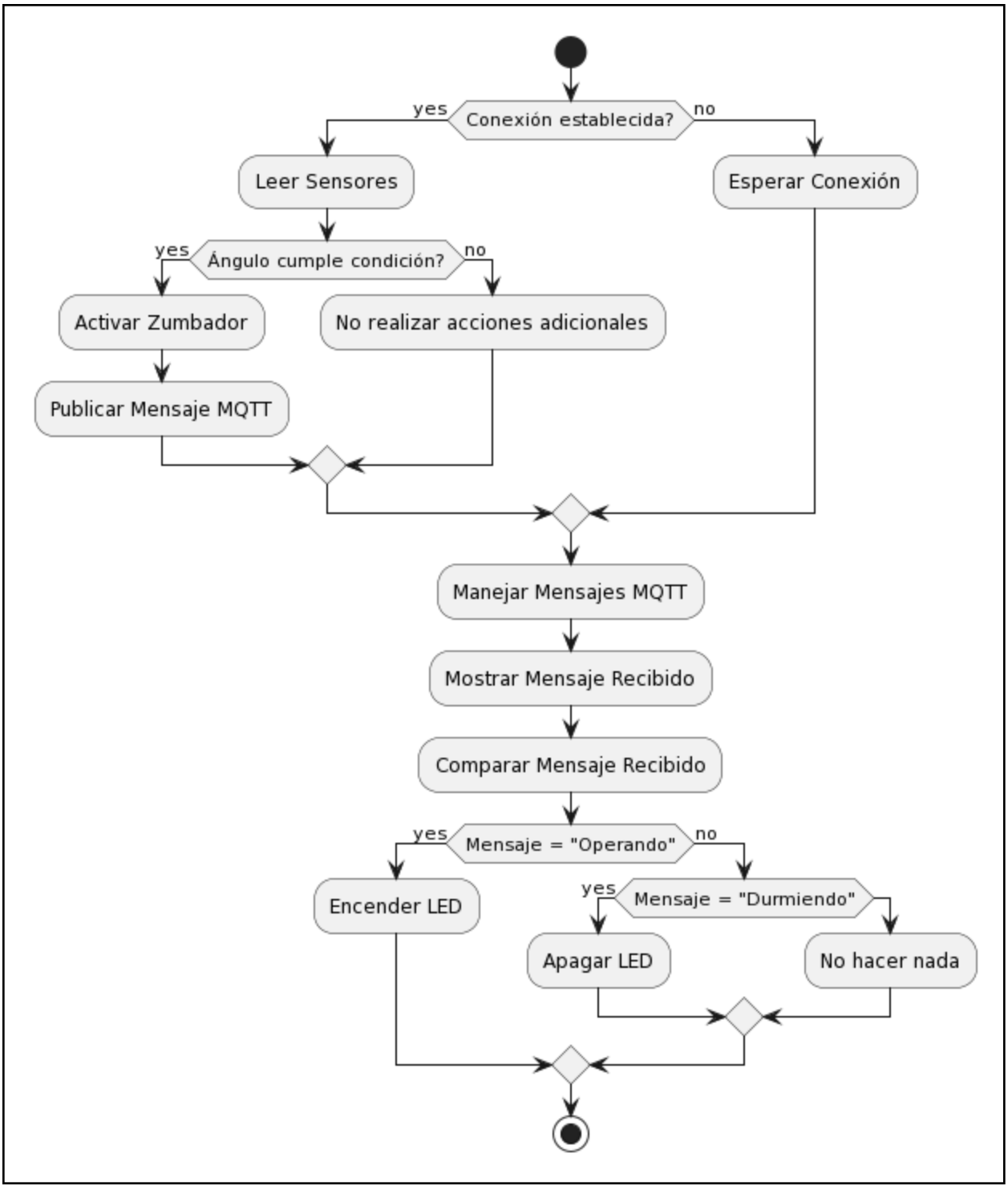
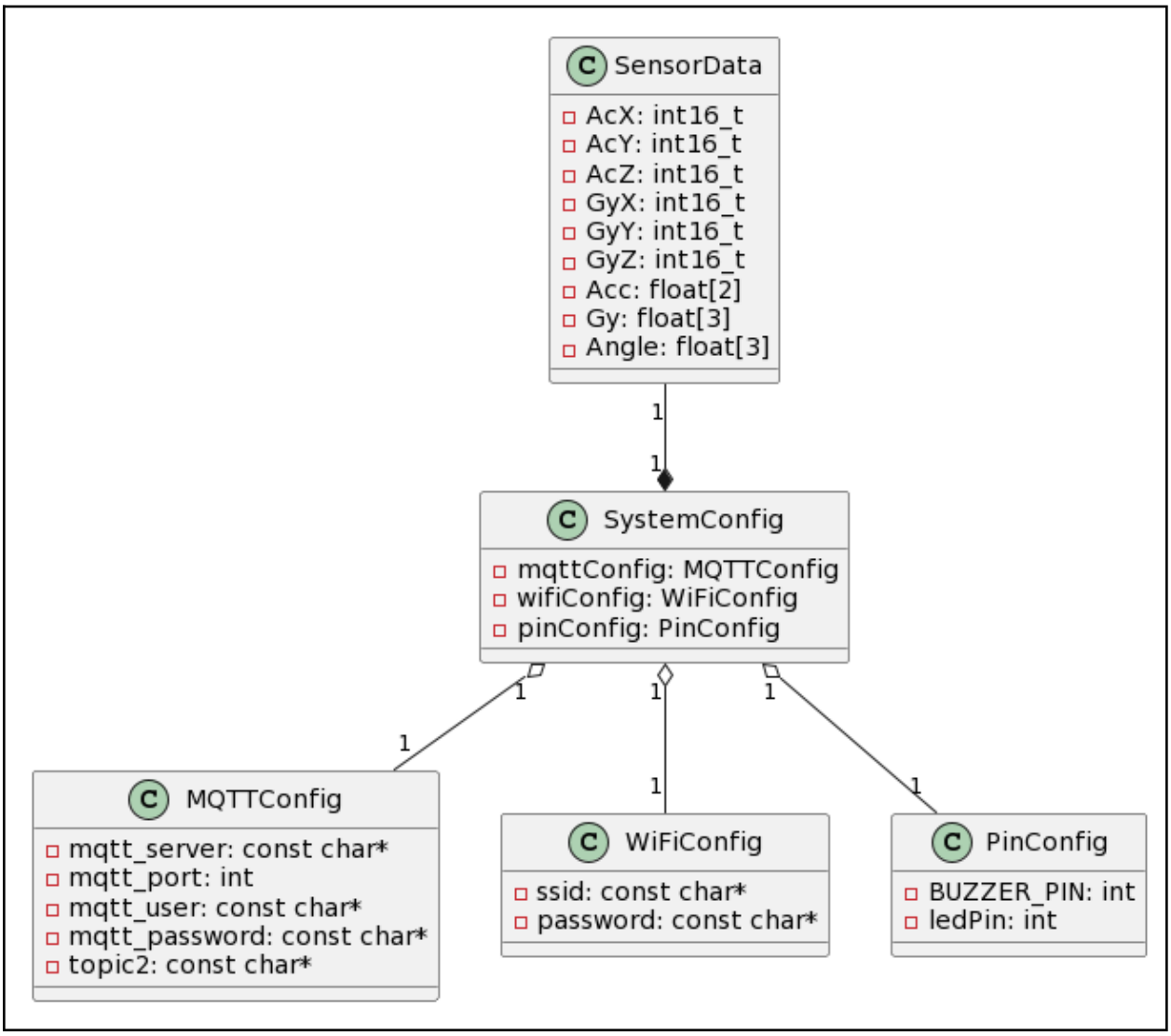


Diagrama de actividades


Conexión establecida?
yes
no
Leer Sensores
Esperar Conexión
Ángulo cumple condición?
yes
no
Activar Zumbador
No realizar acciones adicionales
Publicar Mensaje MQTT
Manejar Mensajes MQTT
Mostrar Mensaje Recibido
Comparar Mensaje Recibido
Mensaje = "Operando"
yes
no
Encender LED
Mensaje = "Durmiendo"
yes
Apagar LED
No hacer nada

3.1.2 Revisión de la estructura de datos



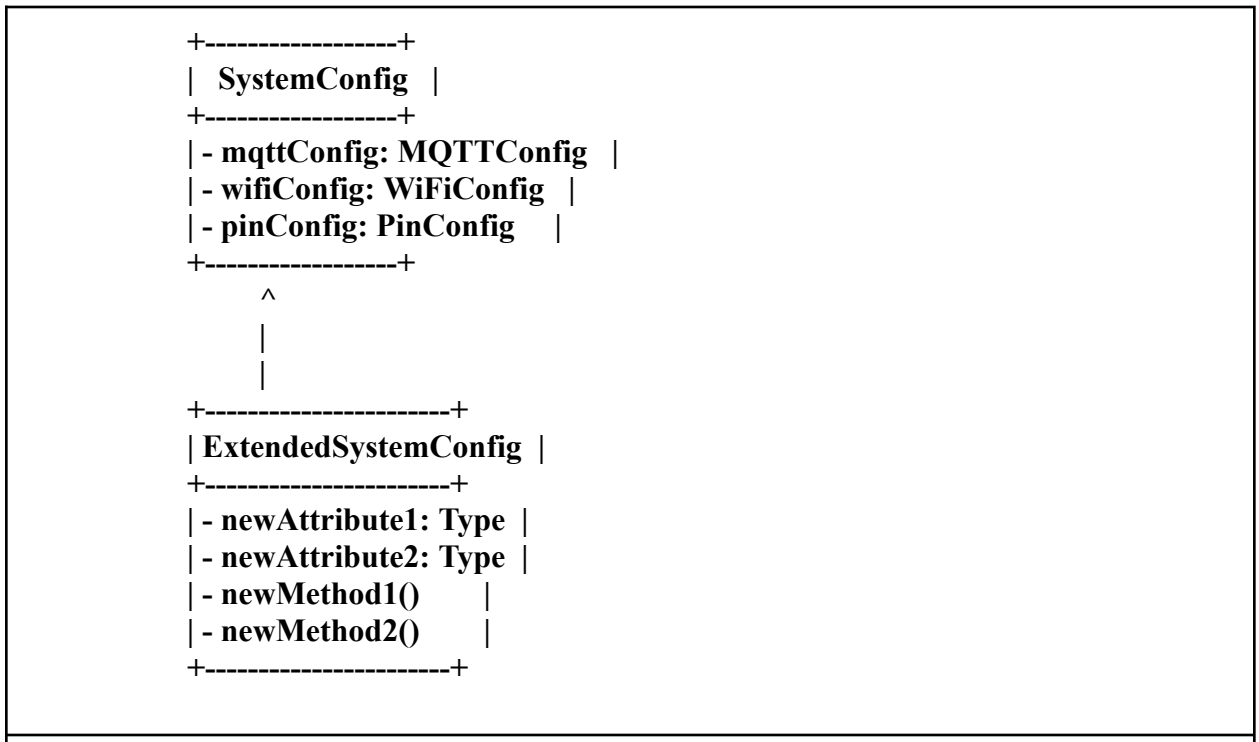
Variables Globales:

- ***ssid***: Representa el nombre de la red Wi-Fi y se almacena como una cadena de caracteres.
- ***password:*** Almacena la contraseña asociada a la red Wi-Fi y se guarda como una cadena de caracteres.
- ***mqtt_server***: Contiene la dirección IP del servidor MQTT y se almacena como una cadena de caracteres.
- ***mqtt_port***: Representa el puerto del servidor MQTT y se guarda como un número entero.
- ***mqtt_user***: Almacena el nombre de usuario utilizado para la conexión con el servidor MQTT y se guarda como una cadena de caracteres.
- ***mqtt_password:*** Contiene la contraseña utilizada para la conexión con el servidor MQTT y se almacena como una cadena de caracteres.
- ***BUZZER_PIN:*** Representa el número de pin utilizado para el zumbador y se guarda como un número entero.
- ***MPU:*** Constante que contiene la dirección I2C de la IMU y se guarda como un número entero.
- ***datoImpreso:*** Variable booleana que indica si se ha impreso un dato o no.
- ***AcX, AcY, AcZ, GyX, GyY, GyZ:*** Variables que almacenan los valores de los sensores y se guardan como números enteros de 16 bits.
- ***Acc:*** Arreglo que almacena los ángulos calculados del acelerómetro y se guarda como un arreglo unidimensional de números flotantes.
- ***Gy:*** Arreglo que almacena los valores del giroscopio y se guarda como un arreglo *un*idimensional de números flotantes.
- ***Angle:*** Arreglo que almacena los ángulos calculados y se guarda como un arreglo *u*nidimensional de números flotantes.
- valores: Variable de tipo String que almacena una cadena de texto.

Variables Locales:

- ***tiempo_prev:*** Variable que almacena el tiempo anterior y se guarda como un número entero largo.
- ***dt:*** Variable que almacena el intervalo de tiempo y se guarda como un número flotante.
- ***espClient:*** Variable utilizada para la conexión Wi-Fi y se guarda como un objeto de la clase WiFiClient.
- ***client:*** Variable utilizada para la conexión MQTT y se guarda como un objeto de la clase PubSubClient.

3.2 Estructura de programa derivada



- ➔ En este diagrama, la clase SystemConfig es la clase base original que contiene los atributos mqttConfig, wifiConfig y pinConfig. La clase ExtendedSystemConfig es la estructura de programa derivada que hereda de SystemConfig y agrega atributos adicionales, como newAttribute1 y newAttribute2. También puede tener métodos adicionales, como newMethod1() y newMethod2().
- ➔ El símbolo ^ indica una relación de herencia, lo que significa que ExtendedSystemConfig hereda de SystemConfig. Esto significa que ExtendedSystemConfig tiene acceso a los atributos y métodos de la clase base SystemConfig, además de los atributos y métodos adicionales que agrega.
- ➔ Este diagrama de clases proporciona una representación visual de la estructura de programa derivada, mostrando la relación de herencia entre las clases base y derivada, así como los atributos y métodos adicionales en la estructura derivada.

3.3 Interfaces dentro de la estructura

Interfaz MQTT:

connect(server, port, user, password): Establece una conexión con el servidor MQTT utilizando los parámetros de dirección del servidor, puerto, usuario y contraseña.

disconnect(): Cierra la conexión con el servidor MQTT.

publish(topic, message): Publica un mensaje en un tópico específico.

subscribe(topic): Se suscribe a un tópico específico para recibir mensajes entrantes.

setCallback(callback): Establece una función de devolución de llamada para procesar los mensajes MQTT recibidos.

Estos métodos definen el conjunto de funcionalidades básicas para interactuar con un servidor MQTT y enviar/recibir mensajes en diferentes tópicos.

Entonces, la interfaz MQTT actúa como un contrato que otras clases pueden implementar. Al implementar esta interfaz, se debe proporcionar una implementación concreta para cada uno de los métodos mencionados anteriormente.

3. MÓDULOS

- **Módulo 1: Lectura de datos del giroscopio:**

1. Texto explicativo:

El módulo de lectura de datos del giroscopio se encarga de obtener los datos del giroscopio incorporado en el casco y procesarlo para detectar niveles críticos de inclinación en la cabeza del operador. Estos datos son fundamentales para determinar la somnolencia o falta de concentración del operador.

2. Descripción de la interfaz:

El módulo de lectura de datos del giroscopio cuenta con una interfaz que ofrece métodos y funciones para la configuración inicial del giroscopio, la lectura de los datos de inclinación y la transferencia de los resultados a otros módulos del sistema.

3. Descripción en lenguaje de diseño:

- Emplear el giroscopio MPU6050 para obtener la información sobre la inclinación de la cabeza del operador.
- Se realizaron ajustes en la configuración de los pines dedicados a la conexión del giroscopio, con el objetivo de permitir la correcta lectura de los datos provenientes del mismo
- Se empleó una biblioteca especializada o un conjunto de funciones adicionales para simplificar la tarea de leer y procesar los datos provenientes del giroscopio.
- Ajustes de parámetros específicos dentro del código de Arduino, utilizando .

- Se realizaron pruebas y calibraciones para garantizar mediciones precisas de la inclinación mediante la modificación de parámetros y configuraciones en el código Arduino.
- Se establecieron límites de inclinación para detectar somnolencia o falta de concentración, activando una alarma de emergencia cuando se alcanza un ángulo crítico, mediante un algoritmo de detección de somnolencia.

4. Módulos utilizados:

Módulos utilizados en la implementación de la lectura del giroscopio:

- **Placa Arduino:** se realizaron procedimientos destinados a establecer la conexión del giroscopio y permitir la lectura de los datos de manera efectiva.
- **Giroscopio MPU6050:** con el propósito de obtener la medida de la inclinación de la cabeza del operador, se implementaron acciones específicas.
- **Bibliotecas o librerías de Arduino:** con el objetivo de simplificar la lectura y el procesamiento de los datos provenientes del giroscopio, se implementaron medidas para facilitar esta tarea.

5. Organización de los datos:

Los datos de inclinación obtenidos del giroscopio pueden ser estructurados en variables o estructuras de datos simples para su posterior procesamiento y análisis. Además, se pueden emplear umbrales predefinidos para identificar niveles críticos de inclinación..

6. **Comentarios:**

   Realizar pruebas y calibraciones adecuadas del giroscopio es esencial para garantizar mediciones precisas de la inclinación. Además, se debe tener en cuenta una configuración adecuada en el código Arduino para obtener resultados confiables y consistentes en las mediciones.

- **Módulo 2: Alarma de emergencia o zumbador :**

7. **Texto explicativo:**

   El objetivo principal del módulo de alarma de emergencia es activar una alarma de bocina de pánico dependiendo del ángulo de inclinación. Esto tiene como propósito alertar al operador y a otros trabajadores cercanos sobre la presencia de un riesgo inminente.

8. **Descripción de la interfaz:**

   La interfaz del módulo de alarma de emergencia incluye métodos y funciones para activar la alarma de bocina de pánico y enviar mensajes MQTT

9. **Descripción en lenguaje de diseño:**

   - Utilizar salidas del Arduino o microcontrolador para activar la alarma de bocina de pánico cuando se detecte un ángulo crítico.
   - Configurar la comunicación para enviar notificaciones sobre la situación de riesgo.

10. Módulos utilizados:

- **Placa Arduino:** Para controlar las salidas de la alarma y la comunicación con otros dispositivos.
- **Dispositivos de alarma:** Una bocina de pánico integrada en el casco.
- **Web:** Para poder recibir datos sobre el estado del operador, ergo, ver si está operando o durmiendo.

11. Organización de los datos:

En este módulo, no se almacenan datos específicos, ya que su función principal es activar la alarma de bocina de pánico y enviar notificaciones en tiempo real en respuesta a la detección de somnolencia o falta de concentración.

12. Comentarios:

Es esencial realizar pruebas exhaustivas del módulo de alarma de emergencia y notificaciones para garantizar un funcionamiento correcto y confiable en situaciones de emergencia. Además, se recomienda implementar mecanismos de seguridad y validar la somnolencia o falta de concentración antes de activar la alarma de emergencia para evitar falsos positivos.

- **Módulo 3: Comunicación y almacenamiento de datos :**

  1. **Texto explicativo:**

     El módulo de comunicación y almacenamiento de datos se encarga de establecer la conexión con la instancia en la nube, enviar los datos de inclinación del giroscopio de manera segura, y almacenar y procesar dichos datos para su posterior visualización y análisis. También permite recibir notificaciones desde la nube para activar o desactivar la alarma de emergencia.

  2. **Descripción de la interfaz:**

     La interfaz del módulo de comunicación y almacenamiento de datos incluye métodos y funciones para establecer la conexión con la instancia en la nube, enviar los datos de inclinación, recibir notificaciones desde la nube y gestionar la activación/desactivación de la alarma de emergencia.

  3. **Descripción en lenguaje de diseño:**

     - Establecer una conexión segura con la instancia en la nube, como AWS (Amazon Web Services), utilizando el protocolo MQTT (Message Queuing Telemetry Transport) o HTTP.
     - Enviar datos de inclinación del giroscopio a la instancia en la nube de forma segura y confiable.
     - Configurar la recepción de notificaciones desde la nube para activar o desactivar la alarma de emergencia.
     - Almacenar y procesar los datos de inclinación en la instancia en la nube, utilizando una base de datos en este caso PostgreSQL y herramientas de análisis para su posterior visualización y análisis.

4. **Módulos utilizados:**

    - **Placa Arduino:** para establecer la comunicación con la instancia en la nube y enviar datos de inclinación.
    - **Instancia en la nube:** como AWS o una base de datos remota, para almacenar y procesar los datos de inclinación.
    - **Protocolos de comunicación:** Se utilizan protocolos de comunicación, como MQTT o HTTP, para establecer la conexión segura con la instancia en la nube y enviar/receptar los datos de inclinación.
    - **Bibliotecas y módulos:** En la placa Arduino, se utilizan las bibliotecas mqtt, wire.h y esp8266Wifi.h. En el entorno de desarrollo Python, se utilizan las bibliotecas psycopg2, sys, pySerial y mqtt_client.

5. **Organización de los datos:**

    Los datos de inclinación del giroscopio se organizan en una estructura de datos adecuada, como un objeto JSON, antes de ser enviados a la instancia en la nube. En la instancia en la nube, los datos se almacenan en una base de datos (por ejemplo, PostgreSQL) y se procesan según los requisitos de visualización y análisis.

6. **Comentarios:**

    Es esencial configurar correctamente la comunicación con la instancia en la nube y asegurarse de que se establezcan los protocolos de seguridad necesarios para proteger los datos transmitidos. Además, se recomienda implementar mecanismos de monitoreo y gestión de datos en la instancia en la nube para un mejor análisis y toma de decisiones.

- **Módulo 4: Interfaz de usuario y visualización de datos :**

  1. **Texto explicativo:**

     El módulo de interfaz de usuario y visualización de datos se encarga de proporcionar una interfaz gráfica para visualizar los datos de inclinación del giroscopio, el estado de la alarma de emergencia y otra información relevante. Esta interfaz permite a los usuarios monitorear y analizar los datos recopilados por el casco.

  2. **Descripción de la interfaz:**

     La interfaz de usuario y visualización de datos incluye una página web, para presentar los datos de inclinación y el estado de la alarma.

  3. **Descripción en lenguaje de diseño:**

     - Desarrollar una página web que muestre los datos de inclinación del giroscopio y el estado de la alarma en tiempo real.
     - Utilizar tecnologías web como HTML, CSS y JavaScript, o frameworks como Angular, React o Vue.js, para crear la interfaz y los elementos visuales.
     - Conectar la interfaz con la instancia en la nube para obtener los datos de inclinación y el estado de la alarma.

4. **Módulos utilizados:**

    - **Tecnologías web:** como HTML, CSS y JavaScript, o frameworks como Angular, React o Vue.js, para desarrollar la interfaz de usuario.
    - **API de la instancia en la nube:** para obtener los datos de inclinación y el estado de la alarma en tiempo real.

5. **Organización de los datos:**

    Los datos de inclinación y el estado de la alarma se pueden organizar en estructuras de datos adecuadas, como objetos JSON, para su visualización en la interfaz. Los datos se pueden actualizar en tiempo real mediante la conexión con la instancia en la nube.

6. **Comentarios:**

    Es importante diseñar una interfaz intuitiva y fácil de usar que permita al usuario monitorear y analizar los datos de inclinación del giroscopio y el estado de la alarma de manera efectiva. Además, se recomienda realizar pruebas exhaustivas de la interfaz para garantizar un funcionamiento correcto y una experiencia de usuario satisfactoria.

4. REFERENCIAS CRUZADAS PARA LOS REQUISITOS

El código que usamos en arduino viene parte del artículo *Programador Novato* y nos ha permitido crear una Conexión WiFi ESP32 para poder controlar dispositivos conectados al ESP32 o incluso desde el ESP32 manipular otros dispositivos. Y de alguna manera u otra, vimos tutoriales en youtube, así como, repositorios github que nos informaron del tema sobre cómo usar las librerías pubsubclient, el cual nos permite proporcionar un cliente para realizar mensajes simples de publicación/suscripción con un servidor compatible con MQTT. y wire.h que nos permite comunicarnos con dispositivos por bus I2C en nuestro código del arduino.

5. PROVISIONES DE PRUEBA

5.1 Directrices de prueba

Pruebas de conexión:

- Verificar si se puede establecer una conexión exitosa con el punto de acceso Wi-Fi utilizando las credenciales proporcionadas (ssid y password).
- Comprobar si se puede conectar correctamente al servidor MQTT en AWS utilizando las credenciales proporcionadas (mqtt_server, mqtt_port, mqtt_user y mqtt_password).

Pruebas de publicación y suscripción MQTT:

- Verificar si el sistema es capaz de publicar mensajes MQTT en el tópico deseado cuando se cumple una condición específica (Angle[1] > 60.0).
- Comprobar si el sistema es capaz de recibir mensajes MQTT en el tópico deseado y realizar las acciones correspondientes (encender o apagar el LED) según el contenido del mensaje.

Pruebas de lectura de valores del giroscopio y acelerómetro:

- Verificar si los valores del giroscopio y el acelerómetro se leen correctamente a través de la interfaz I2C y si se realiza la conversión adecuada de unidades.
- Comprobar si los ángulos de inclinación (Acc y Gy) se calculan correctamente utilizando los datos del acelerómetro y el giroscopio.

Pruebas del filtro complementario:

- Verificar si los ángulos filtrados (Angle) se calculan correctamente utilizando el filtro complementario.
- Comprobar si el ángulo de giro (Angle[2]) se actualiza correctamente con los valores del giroscopio.

Pruebas de activación de la alarma:

- Verificar si la alarma se activa correctamente cuando el ángulo de inclinación (Angle[1]) supera el umbral de 60.0.
- Comprobar si la alarma se desactiva correctamente cuando el ángulo de inclinación vuelve a ser igual o inferior a 60.0.

5.2 Estrategia de integración

Integración incremental:
El código se divide en módulos y se integran gradualmente. Se comienza con la configuración de Wi-Fi y la inicialización de la conexión MQTT.
A medida que se avanza en el código, se integran los módulos de lectura de valores del acelerómetro y giroscopio, cálculo de ángulos, aplicación del filtro complementario y detección de condiciones para activar la alarma y enviar mensajes MQTT.
Cada módulo se prueba y verifica su correcto funcionamiento antes de pasar al siguiente.
Integración descendente:
Se comienza con la integración de los módulos de nivel superior, como la configuración de Wi-Fi, la inicialización de la conexión MQTT y la configuración de pines.
A medida que se avanza en el código, se integran los módulos de lectura de valores del acelerómetro y giroscopio, cálculo de ángulos, aplicación del filtro complementario y detección de condiciones para activar la alarma y enviar mensajes MQTT.
Los módulos integrados se prueban y se asegura su correcto funcionamiento antes de integrar los módulos de nivel inferior.

5.3 Consideraciones especiales

Una consideración especial para el código proporcionado incluye la gestión de errores para garantizar un comportamiento adecuado frente a posibles situaciones de error. Además, se deben asegurar las credenciales de forma segura, realizar pruebas exhaustivas para verificar su correcto funcionamiento, optimizar el código para la eficiencia y considerar medidas de seguridad adicionales para proteger la comunicación MQTT en entornos de producción. Estas consideraciones contribuirán a un sistema robusto, confiable y seguro.

6. CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- Se diseñó un dispositivo cuyo costo es de 190 soles, siendo un costo accesible para el cliente teniendo en cuenta el tiempo de vida útil que se le estima y también por su utilidad. A comparación de otros dispositivos que tienen costos muy elevados como la cámara con sensor de fatiga o somnolencia que tiene costes más elevados en dólares.
- Como equipo de trabajo llegamos a la conclusión que el desarrollo de un casco con Arduino como dispositivo de alerta representa una solución tecnológica prometedora para prevenir accidentes causados por la fatiga y la falta de concentración en la industria minera.
- La implementación de este casco con Arduino no solo busca salvaguardar la integridad física de los operadores mineros, sino que también tiene como objetivo reducir el riesgo de colisiones y accidentes en el entorno minero.

6.2 Recomendaciones

1. Promover la adopción del casco con Arduino en la industria minera: Dado que el dispositivo tiene un costo accesible y ofrece una solución prometedora para prevenir accidentes causados por la fatiga y la falta de concentración, es importante promover su adopción en la industria minera. Esto puede incluir campañas de concientización, demostraciones y capacitaciones para los trabajadores y las empresas mineras.

2. Realizar pruebas y estudios de campo: Aunque el casco con Arduino se considera una solución prometedora, es importante llevar a cabo pruebas y estudios de campo para evaluar su efectividad y recopilar datos sobre su desempeño en situaciones reales. Esto ayudará a refinar y mejorar el dispositivo, así como a respaldar su eficacia ante los stakeholders y la comunidad minera en general.

3. Investigar oportunidades de mejora y expansión: A medida que el casco con Arduino se implemente y se recopilen más datos sobre su rendimiento, es recomendable invertir en investigación y desarrollo para identificar oportunidades de mejora y expansión. Esto puede incluir la incorporación de nuevas funcionalidades, como la detección de otros factores de riesgo en la industria minera, o la adaptación del dispositivo para su uso en otros sectores industriales que enfrentan desafíos similares de fatiga y falta de concentración.

7. BIBLIOGRAFÍA Y REFERENCIAS

7.1 Bibliografía

- *Programming Arduino Getting Started with Sketches 1st Edition.* Simon Monk 2012.
- Exploring Arduino: *Tools and Techniques for Engineering Wizardry 1st* Edition, Jeremy Blum 2013.
- Arduino Workshop: *A Hands-On Introduction with 65 Projects 1st* Edition John Boxall 2013.

7.2 Referencias

Ricardo Llerena. (2023). *"Conexión Arduino"*. Recuperado de: (https://youtu.be/lVjx_KD5MxQ)

Ricardo Llerena. (2023). *"Comunicación ARDUINO -MYQL por mqtt y python"*. Recuperado de (https://youtu.be/4LvT8OwAhSs)

Ricardo Llerena. (2023). *"Manejo de base de datos: Creación de un CRUD con PHP y AJAX"*. Recuperado de (https://youtu.be/eyV65YjTQMY)

Ricardo Llerena. (2023). *"Vinculación archivos de Visual Studio Code"*. Recuperado de https://youtu.be/Z4XKVi7Blwk

Pizarro, J. (2020). *Libro de Internet de las Cosas con Arduino*. Editorial. Recuperado de SBN 978-84-283-4496-8

8. NOTAS ESPECIALES

- El casco con Arduino se presenta como una solución tecnológica innovadora para abordar el desafío de la fatiga y la falta de concentración en la industria minera, que son factores de riesgo significativos para los accidentes graves.
- El prototipo del casco está diseñado específicamente para brindar apoyo a los operadores mineros, quienes enfrentan largas jornadas de trabajo y un mayor riesgo de distracción y somnolencia al volante.
- El dispositivo de alarma incorporado en el casco utiliza un giroscopio para detectar ángulos de inclinación críticos en la cabeza del operador y activar una bocina de pánico cuando se alcanza un ángulo específico, alertando al operador y a otros trabajadores cercanos sobre la situación de riesgo.
- El objetivo principal del proyecto es salvaguardar la integridad física de los operadores mineros, reduciendo el riesgo de colisiones y accidentes causados por la somnolencia o la falta de atención.
- El casco con Arduino tiene la capacidad de adaptarse con características adicionales para mejorar aún más la seguridad en la industria minera, lo que lo convierte en una solución flexible y escalable.
- El código utiliza la plataforma ESP8266 y las librerías correspondientes para la comunicación Wi-Fi y MQTT, lo que permite la conexión y comunicación con el servidor MQTT en AWS.

- Se utiliza el protocolo MQTT para la suscripción a un tópico específico y recibir mensajes entrantes. Esto permite la integración del casco con otros sistemas y aplicaciones que utilicen MQTT como medio de comunicación.
- El código implementa la lectura de datos del acelerómetro y giroscopio a través del protocolo I2C, lo que proporciona información sobre los ángulos de inclinación y el movimiento del casco.
- Se utiliza el filtro complementario para combinar la información del acelerómetro y el giroscopio y obtener ángulos más precisos. Esta técnica mejora la detección de situaciones de riesgo y la activación adecuada de la alarma.
- El código utiliza una función de callback para manejar los mensajes MQTT entrantes, lo que permite tomar acciones específicas según el contenido del mensaje, como encender o apagar el LED o enviar mensajes de estado.

9. APÉNDICES

Apéndice A: Módulo de Lectura de Datos del Giroscopio

Este apéndice detalla la implementación y funcionamiento del módulo encargado de obtener los datos del giroscopio incorporado en el casco y procesarlos para detectar niveles críticos de inclinación en la cabeza del operador. Incluye información sobre la configuración del giroscopio, la lectura de datos y el cálculo de la inclinación, así como las pruebas y calibraciones realizadas.

Apéndice B: Módulo de Alarma de Emergencia o Zumbador

Este apéndice describe en detalle el módulo responsable de activar una alarma de bocina de pánico según el ángulo de inclinación detectado. Se incluye información sobre la configuración de las salidas del Arduino, la activación de la alarma y el envío de notificaciones a través de la web. También se mencionan las pruebas realizadas y los posibles falsos positivos.

Apéndice C: Módulo de Comunicación y Almacenamiento de Datos

En este apéndice se explican los detalles del módulo encargado de establecer la conexión con la instancia en la nube, enviar los datos de inclinación de forma segura, recibir notificaciones y gestionar la activación/desactivación de la alarma de emergencia. Se mencionan los protocolos de comunicación utilizados, la organización de los datos y las herramientas de almacenamiento y procesamiento en la nube.

Apéndice D: Módulo de Interfaz de Usuario y Visualización de Datos

Este apéndice se centra en el módulo que proporciona una interfaz gráfica para visualizar los datos de inclinación del giroscopio y el estado de la alarma. Se describen las tecnologías web utilizadas, la conexión con la instancia en la nube y la organización de los datos para su presentación en tiempo real. También se mencionan las consideraciones de diseño y las pruebas realizadas.