Propuestas Mini-Proyectos De Articulación

A continuación se presentan seis propuestas de miniproyectos, cada una con sus objetivos y una metodología de desarrollo que integra persistencia de datos, programación orientada a objetos (POO), bases de datos y el uso de sensores de bajo costo, junto con un breve estudio económico de los materiales

---

1. Monitorización de Signos Vitales (Bioingeniería) (asignación para 1 grupo)

Objetivos:
Desarrollar un sistema que recoja continuamente datos de signos vitales (como frecuencia cardíaca, temperatura corporal y niveles de oxígeno) mediante sensores biométricos de bajo costo. El sistema deberá almacenar la información en una base de datos para permitir análisis históricos, alertar ante valores anómalos y ofrecer una visualización sencilla de los datos.

Línea de investigación:
Sistemas Biomédicos y Biosensores

Puntos fuertes:

1. **Integración de sensores biométricos:** Permite la adquisición continua y no invasiva de datos vitales.
2. **Análisis en tiempo real y almacenamiento histórico:** Facilita la detección de anomalías y el seguimiento longitudinal de la salud.
3. **Interfaz de usuario intuitiva:** Posibilita la visualización y alerta inmediata, siendo escalable a aplicaciones clínicas.

Metodología de desarrollo:

- **Lenguajes y Frameworks:**
    - Lenguaje: Python
    - Framework (opcional): Flask para desarrollar una API o una interfaz web básica; Tkinter si se prefiere una interfaz de escritorio.
- **Bibliotecas y Persistencia:**
    - SQLite para la base de datos, que permitirá la persistencia de datos de forma local.
    - Librerías para manejo de datos (por ejemplo, Pandas para análisis y visualización) y para comunicación serial (pySerial) si se utiliza un microcontrolador.

- **Programación Orientada a Objetos (POO):**
    - Crear clases como SensorPulso, SensorTemperatura y MonitorBiometrico que encapsulen la lógica de adquisición de datos, validación, procesamiento y almacenamiento.
    - Creación y división de paquetes en el proyecto, junto a la creación de clases especializadas para la persistencia.
        - Controllers (La parte web)
        - Services (Lógica del negocio)
        - Models (Modelos/clases)
        - Databse (Manejo de persistencia)
        - GUI
        - Assets (Imagines[si es que es necesario], iconos, audios)
    - Crear diagramas UML para representar de mejor manera las clases y proponer mejoras en el sistema en caso de requerir actualizaciones posteriores.
        - Clases
        - Casos de uso
        - Secuencia
        - Actividades

        <básicamente planeación a nivel arquitectural, va para todos los proyecto

- **Materiales y Sensores:**
    - Sensor de pulso (Pulse Sensor) – costo aproximado: 15k.
    - Sensor de temperatura (LM35 o similar) – costo aproximado: 6k a 10k
    - Sensor de SpO_2 (si se desea medir saturación de oxígeno) – costo aproximado: 30k
    - Microcontrolador o placa de desarrollo (Arduino, Raspberry Pi o ESP8266, según el caso) – entre 30k a 60k
- **APIs:**
    - Se puede implementar una API REST con Flask para exponer los datos en una aplicación web o para integrarse con plataformas IoT.

Estudio económico:
El costo total de los sensores y la placa de desarrollo se estima en torno a 50K y 100k

---

2. Sistema de Detección de Intrusiones (Ciberseguridad) (asignación para 3 grupos)

Objetivos:
Crear un sistema de seguridad perimetral que utilice sensores de movimiento y sensores magnéticos para puertas. El sistema registrará eventos en una base de datos y enviará alertas cuando se detecten intrusiones o anomalías, facilitando un análisis histórico para investigaciones de seguridad.

Línea de investigación:
Sistemas de Seguridad Electrónica e IoT para Seguridad Física

Puntos fuertes:

1. **Monitoreo en tiempo real:** Utiliza sensores de movimiento y magnéticos para detectar accesos no autorizados de manera inmediata.
2. **Registro y análisis de eventos:** Permite llevar un historial de incidentes para análisis forense y mejora continua del sistema.
3. **Alertas automatizadas:** La integración de notificaciones (por ejemplo, vía Telegram) mejora la capacidad de respuesta ante amenazas.

    <se pueden integrar camaras, tambien podria implementarse machine learning para evitar falsos positivos,

    al monitoreo en tiempo real puede dársele un dashboard para control manual si se desea y el análisis de los logs

    Podría implementarse una respuesta automática en caso de intrusión como activar sirenas y bloquear accesos y dispositivos en la red>

Metodología de desarrollo:

- **Lenguajes y Frameworks:**
    - Lenguaje: Python
    - Framework: Flask para el desarrollo de una API que permita notificaciones y visualización de eventos; opcionalmente, se puede usar un framework de mensajería para enviar alertas (por ejemplo, integración con SMTP o servicios como Twilio).
- **Bibliotecas y Persistencia:**
    - SQLite para almacenar registros de eventos (fecha, hora, tipo de sensor activado). <para un proyecto que almacena logs de eventos físicos con demasiada “intensidad” puede que sqlite se quede corto y sea difícil escalar el proyecto además de los tipos de datos manejados son bastante limitados como para almacenar eventos >
    - Librerías para manejo de GPIO (por ejemplo, RPi.GPIO si se utiliza Raspberry Pi) y para el envío de correos o notificaciones.

- **Programación Orientada a Objetos (POO):**
    - Definir clases como SensorMovimiento, SensorPuerta y SistemaSeguridad que centralicen la lectura de datos, registro de eventos y gestión de alertas.

      <esas clases por ahora solo son entidades, para implementar la lógica se necesitan los manager para una serie amplia de sensores>

- **Materiales y Sensores:**
    - Sensor de movimiento PIR – costo aproximado: 7k a 10k.
    - Sensor magnético para puertas – costo aproximado: 4k a 8k.
    - Microcontrolador (por ejemplo, Raspberry Pi o Arduino) – costo aproximado: 30k a 60k

- **APIs:**
    - Una API REST en Flask para exponer los datos de eventos o integrarse con aplicaciones móviles de monitoreo.

Estudio económico:
Considerando los sensores y el microcontrolador, el costo total se sitúa aproximadamente entre 40k a 70k, lo que lo hace una opción viable para proyectos de ciberseguridad a nivel académico.

<También es importante la valoración económica del tiempo y proceso de desarrollo >

---

3. Sistema Inteligente de Riego Automatizado (AIoT)

Objetivos:
Desarrollar un sistema de riego que optimice el uso de agua en cultivos o jardines, utilizando sensores ambientales para medir la humedad del suelo, temperatura y luz. El sistema decidirá de forma automatizada cuándo activar la bomba de riego y almacenará los datos para análisis y monitoreo remoto.

Línea de investigación:
Sistemas Embebidos e Internet de las Cosas (IoT) para Agricultura Inteligente

Puntos fuertes:

1. **Optimización del consumo de agua:** Control automatizado basado en datos ambientales reduce el desperdicio de recursos hídricos.
2. **Integración de múltiples sensores ambientales:** Proporciona un monitoreo preciso de humedad, temperatura y luz para ajustar el riego y la iluminación.
3. **Conectividad y control remoto:** La aplicación móvil y la interfaz web permiten la supervisión y ajuste en tiempo real, favoreciendo la toma de decisiones.

Metodología de desarrollo:

- **Lenguajes y Frameworks:**
    - Lenguaje: Python o MicroPython (en caso de usar placas como ESP8266).
    - Framework: Flask para el desarrollo de una API y una interfaz web, o integración con plataformas IoT como Blynk o ThingSpeak para visualización remota.
- **Bibliotecas y Persistencia:**
    - SQLite para la persistencia local de datos ambientales.
    - Librerías para manejo de datos, comunicación WiFi y procesamiento en tiempo real.
- **Programación Orientada a Objetos (POO):**
    - Crear clases como SensorHumedad, SensorTemperatura, SensorLuz y ControlRiego que gestionen la adquisición de datos, la toma de decisiones (activar/desactivar la bomba) y la persistencia en la base de datos.

- **Materiales y Sensores:**
    - Sensor de humedad del suelo – costo aproximado: 5k a 10k.
    - Sensor de temperatura y humedad (DHT11 o DHT22) – costo aproximado: 15k a 20k.
    - Sensor de luz (fotorresistor) – costo aproximado: 1k a 3k.
    - Módulo de relé para controlar la bomba – costo aproximado:4k a 10k.
    - Placa de desarrollo (ESP8266 o similar) – costo aproximado: 20k a 30k.
- **APIs:**
    - Se puede desarrollar una API REST con Flask o utilizar servicios IoT existentes para enviar y visualizar datos de forma remota.

Estudio económico:
El conjunto de sensores y la placa de desarrollo tienen un costo total estimado entre 40k y 60k. Sumando módulos adicionales como relés y cables, el proyecto podría costar entre 50k y 70k, siendo una solución muy económica para aplicaciones AIoT.

---

4. Estación Meteorológica Inteligente con Análisis Predictivo (asignación doble)

Objetivos:
Se busca construir una estación meteorológica que mida parámetros ambientales como temperatura, humedad, presión, y, opcionalmente, radiación solar y velocidad del viento. La idea es almacenar estos datos en una base de datos de series temporales y utilizar algoritmos de aprendizaje automático para predecir el clima a corto plazo, además de disponer de una interfaz web interactiva para visualizar datos históricos y predicciones.

Línea de investigación:
Monitoreo Ambiental y Sistemas de Predicción mediante Aprendizaje Automático

Puntos fuertes:

1. **Recopilación integral de datos climáticos:** Mide múltiples parámetros (temperatura, humedad, presión, radiación y viento) para obtener un panorama ambiental completo.
2. **Modelado predictivo avanzado:** Emplea algoritmos de machine learning para predecir condiciones climáticas a corto plazo.
3. **Visualización interactiva:** La interfaz web con gráficos dinámicos facilita el análisis histórico y la toma de decisiones en la gestión ambiental.

Metodología de desarrollo:

- **Recopilación de datos:**
  - Conectar sensores a un microcontrolador (por ejemplo, ESP32 o Raspberry Pi).
  - Utilizar sensores como el DHT22 (temperatura y humedad), BMP280 (presión) y, opcionalmente, un LDR para radiación solar y un anemómetro para velocidad del viento.
  - Programar el microcontrolador en Python para leer y enviar datos de forma periódica.
- **Almacenamiento de datos:**
  - Diseñar e implementar una base de datos para series temporales. Se puede elegir entre SQLite para prototipos simples o InfluxDB/TimescaleDB para manejo de grandes volúmenes de datos.
  - Desarrollar scripts en Python para insertar y actualizar datos en la base de datos.
- **Análisis de datos y modelado predictivo:**
  - Extraer datos con Pandas y realizar un análisis exploratorio para identificar patrones.
  - Implementar modelos de aprendizaje automático (por ejemplo, regresión lineal o redes neuronales usando Scikit-learn o TensorFlow Lite) para predecir el clima a corto plazo.
- **Visualización e interfaz web:**
  - Desarrollar una interfaz web interactiva con frameworks como Flask o Django.
  - Utilizar bibliotecas de visualización como Plotly o Bokeh para crear gráficos dinámicos que muestren datos históricos y predicciones.
- **Lenguaje de programación:** Python
- **Framework Web:** Flask o Django
- **Bibliotecas:** Pandas, NumPy, Scikit-learn, TensorFlow Lite, Plotly/Bokeh

- **Base de datos:** SQLite, InfluxDB o TimescaleDB
- **Sensores:** DHT22 (15k), BMP280 (30k), LDR (1k) y anemómetro (20k)
- **Placa/Microcontrolador:** ESP32 o Raspberry Pi (40k -60k)
- **APIs:** Posible integración con servicios meteorológicos o APIs propias para exponer datos

Estudio económico:
El costo total estimado para los sensores y la placa oscila entre 80k y 150k , siendo de las opciones menos económicas si no se cuenta con una placa base, sin embargo, una opción más económica sería en los microcontroladores usar un Arduino con un módulo WiFi que reduciría costos a 50k cómo mínimo hasta unos 100k.

---

5. Sistema de Alarma de Seguridad Inteligente con Reconocimiento Facial (asignación doble)

Objetivos:
Desarrollar un sistema de seguridad que combine la detección de movimiento mediante sensores PIR con el reconocimiento facial para identificar posibles intrusos. El sistema registrará eventos en una base de datos y enviará notificaciones (por ejemplo, a través de Telegram), mientras que una interfaz web permitirá visualizar y gestionar los eventos, así como actualizar la base de datos de rostros conocidos.

Línea de investigación:
Visión por Computadora y Sistemas de Seguridad Inteligente

Puntos fuertes:

1. **Integración de detección de movimiento y reconocimiento facial:** Combina hardware y algoritmos avanzados para identificar intrusos de forma precisa.
2. **Registro y gestión de eventos en tiempo real:** Permite almacenar y analizar datos de seguridad para mejorar la respuesta ante incidentes.
3. **Notificaciones y control centralizado:** La interfaz web y las notificaciones automáticas (por ejemplo, vía Telegram) permiten una gestión efectiva del sistema de seguridad.

Metodología de desarrollo:

- **Detección de movimiento y activación:**
  - Conectar un sensor PIR a una Raspberry Pi para detectar movimientos en el área de vigilancia.
  - Programar la Raspberry Pi en Python para activar la cámara al detectar movimiento.
- **Reconocimiento facial:**
  - Utilizar la cámara de la Raspberry Pi para capturar imágenes.
  - Implementar algoritmos de reconocimiento facial utilizando OpenCV y TensorFlow Lite, comparando las imágenes capturadas con una base de datos de rostros conocidos.
- **Registro y notificaciones:**

    - Almacenar los eventos (detección, imágenes y resultados del reconocimiento) en una base de datos ligera como SQLite.
    - Integrar una API de Telegram para enviar notificaciones automáticas al detectar movimientos y rostros desconocidos.
- **Interfaz web:**
    - Desarrollar una interfaz web con Flask o Django que permita visualizar los eventos registrados, gestionar la base de datos de rostros y ajustar parámetros del sistema.
- **Lenguaje de programación:** Python
- **Framework Web:** Flask o Django
- **Bibliotecas:** OpenCV, TensorFlow Lite, Pandas (para análisis de datos), Requests (para comunicación con APIs)
- **Base de datos:** SQLite
- **Sensores:** Sensor PIR (8.5k)
- **Cámara:** Módulo de cámara compatible con Raspberry Pi (20k)
- **Placa:** Raspberry Pi (40k)
- **APIs:** API de Telegram para notificaciones

Estudio económico:
El conjunto de sensores, cámara y la Raspberry Pi se estima en un rango de 90k a 180k, debido a los costos de las Raspberry, pero, como en el proyecto anterior hay alternativas más económicas pero sin tan buen rendimiento, como arduino más un módulos de cámara, y la esp32. Sin embargo, cabe aclarar que el procesamiento de imagen con Arduino no es de tanto nivel como el proyecto lo exigiría, así que podría modificarse el objetivo.

Cada uno de estos mini proyectos se enmarca dentro de una línea de investigación específica en ingeniería electrónica, ofreciendo soluciones innovadoras que combinan hardware, software y técnicas de análisis de datos. Además, sus puntos fuertes destacan la integración de tecnologías modernas con un enfoque práctico, accesible y escalable para aplicaciones académicas y de prototipado.

RECURSOS

Frameworks de ciberseguridad y estándares

HIPAA – Ensures the security of healthcare providers, health plans, and clearinghouses.

CIS Controls – Best practices for securing organizations of all sizes and industries.

1. Monitorización de Signos Vitales (Bioingeniería)

- repositorio.udec.cl/bitstreams/d6863ce9-a52a-4011-94a3-1719950af0ed/download
- https://medlineplus.gov/spanish/ency/article/002341.htm#:~:text=Los%20signos%20vitales%20reflejan%20funciones,su%20nivel%20de%20funcionamiento%20físico.
- https://www-ncbi-nlm-nih-gov.translate.goog/books/NBK553213/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc
- http://hectorfutbool.mex.tl/images/32235/Control_de_signos_vitales.pdf

2. Sistema de Detección de Intrusiones (Ciberseguridad) (asignación para 3 grupos)

- https://blog.invgate.com/es/seguridad-perimetral-informatica
- https://kpmg.com/uk/en/blogs/home/posts/2023/03/cyber-security-and-physical-security.html
- https://www.linkedin.com/pulse/la-seguridad-física-como-primera-línea-de-eibic/
- https://guiasegura.com/seguridad-fisica/deteccion-intrusos/?utm_source=chatgpt.com
- https://www.avigilon.com/es/blog/physical-security-guide?utm_source=chatgpt.com

3. Sistema Inteligente de Riego Automatizado (AIoT)

- https://www.scielo.org.mx/scielo.php?pid=S0568-25172008000400009&script=sci_arttext
- www.plataformaextension.cl/archivos/2020/11/MANUAL-DE-RIEGO.pdf
- https://neiker.eus/wp-content/uploads/2023/05/optimizacion-riego.pdf?utm_source=chatgpt.com
- https://eos.com/es/blog/humedad-del-suelo/

4. Estación Meteorológica Inteligente con Análisis Predictivo (asignación doble)

- https://meteoescuela.aemet.es/cantabria/recursos
- https://blog.meteoclim.com/meteorologia-para-principiantes-i
- https://blog.meteoclim.com/meteorologia-para-principiantes-ii
- https://3ciencias.com/wp-content/uploads/2013/06/METEREOLOGIA.pdf

5. Sistema de Alarma de Seguridad Inteligente con Reconocimiento Facial (asignación doble)

- https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/157736/B_Valverde_FJC-SD.pdf?sequence=1&isAllowed=y
- https://arxiv.org/pdf/1804.06655
- https://arxiv.org/pdf/1503.03832
- https://www.dirzon.com/file/telegram/somali%20educational%20books/deeplearningforcomputervision.pdf
- https://www.researchgate.net/publication/328685305_Face_Recognition_From_Traditional_to_Deep_Learning_Methods
- https://hyperverge-co.translate.goog/blog/face-recognition-algorithm/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

VERSIONES SIMPLES:

1. Monitorización de Signos Vitales (Bioingeniería)

- **Sensores:** En lugar de múltiples sensores, usar solo uno (ejemplo: pulso o temperatura) para la primera versión u omitir solo uno de los 3 que se plantearon.
- **Base de Datos:** Guardar los datos en un archivo CSV en vez de SQLite para reducir la complejidad inicial.
- **Interfaz:** Usar un script simple en Python para graficar los datos en tiempo real, en lugar de hacer una API web.
- **Hardware:** Usar un Arduino que envíe datos al PC a través de puerto serie, sin necesidad de conectividad WiFi.

2. Sistema de Detección de Intrusiones (Ciberseguridad) (asignación para 3 grupos)

- **Sensores:** Usar solo sensores de movimiento PIR para la detección inicial.
- **Almacenamiento:** Guardar eventos en un log de texto en vez de una base de datos.
- **Alertas:** En vez de una API con notificaciones, hacer que el sistema simplemente muestre un mensaje en la pantalla o emita un sonido al detectar intrusos o incluso ambas.
- **Interfaz:** Solo mostrar eventos en consola o trabajar una interfaz web local.

3. Sistema Inteligente de Riego Automatizado (AIoT)

- **Sensores:** Utiliza un sensor de temperatura y humedad DHT11 o DHT22.
- **Almacenamiento:** Guarda los datos recopilados en un archivo CSV simple.
- **Interfaz:** Crea un script de Python que muestre los datos en la consola en tiempo real, o genera gráficos simples utilizando la biblioteca Matplotlib.
- **Hardware:** Utiliza un Arduino para leer los datos del sensor y enviarlos a la computadora a través del puerto serie.
- **Funcionalidad:** El sistema monitorea la temperatura y la humedad del entorno y muestra los datos en la consola o en un gráfico.

4. Estación Meteorológica Inteligente con Análisis Predictivo (asignación doble)

- **Predicción:** No implementar Machine Learning al inicio, solo mostrar tendencias simples con gráficos de datos históricos en matplotlib.
- **Almacenamiento:** Guardar los datos en un archivo CSV en vez de una base de datos.
- **Interfaz:** Mostrar los datos en una interfaz gráfica local en lugar de una web.

5. Sistema de Alarma de Seguridad Inteligente con Reconocimiento Facial (asignación doble)

- **Detección:** Usar solo sensor PIR sin reconocimiento facial en la primera versión.
- **Almacenamiento:** Registrar eventos en un archivo de texto en vez de una base de datos.
- **Alertas:** Solo emitir un sonido o prender un LED en caso de detección.
- **Reconocimiento Facial:** Usar OpenCV con detección de rostros en imágenes estáticas <estilo, en cuanto se detecte movimiento, tomar una imagen del evento y analizarla para tratar de captar el rostro para el registro de la intrusión> antes de pasar a video en tiempo real.

6. Sistema de Alerta de Nivel de Agua

- **Sensores:** Utiliza un sensor de nivel de agua ultrasónico o un sensor de flotador.
- **Almacenamiento:** Registra los eventos (nivel de agua alto/bajo) en un archivo de texto simple.
- **Alertas:** Cuando el nivel de agua alcance un umbral crítico, el sistema emite un sonido a través de un zumbador conectado al Arduino o muestra un mensaje de alerta en la consola.
- **Hardware:** Utiliza un Arduino para leer los datos del sensor y controlar el zumbador.
- **Funcionalidad:** El sistema monitorea el nivel de agua en un tanque o recipiente y alerta al usuario en caso de niveles peligrosos.
    - Este proyecto se puede usar para evitar inundaciones o para monitorear el nivel de agua en sistemas de riego.

7. Sistema de Alerta de Gas

- **Sensores**: Utiliza un sensor de gas MQ-2 o MQ-135 (dependiendo del tipo de gas que quieras detectar).
- **Almacenamiento**: Registra los eventos (niveles de gas alto/bajo) en un archivo de texto simple.
- **Alertas**: Cuando el nivel de gas alcance un umbral crítico, el sistema emite un sonido a través de un zumbador conectado al Arduino o muestra un mensaje de alerta en la consola.
- **Hardware**: Utiliza un Arduino para leer los datos del sensor y controlar el zumbador.
- **Funcionalidad**: El sistema monitorea la concentración de gas en el ambiente y alerta al usuario en caso de niveles peligrosos.