PROYECTO INTERGRADOR I

DOCENTE: GONZALO VERA

ALUMNO/A: KARINA JAZMIN BARBERO

CARRERA: TÉCNICO SUPERIOR EN TELECOMUNICACIONES

Trabajo Práctico 1: Introducción al IoT

Objetivos:

1. Comprender los Fundamentos del IoT: Familiarizarse con los conceptos básicos del Internet de las Cosas, incluyendo su importancia, aplicaciones y el impacto potencial en la sociedad y la industria.
2. Introducción a Git y GitHub: Aprender los fundamentos de Git y GitHub como herramientas esenciales para el desarrollo colaborativo de software, especialmente en proyectos de IoT.
3. Exploración de Herramientas de Desarrollo: Comenzar a utilizar herramientas de desarrollo clave como Visual Studio Code (VsCode), comprendiendo su importancia en la programación y desarrollo de proyectos.

Desarrollo

Parte 1: Investigación sobre IoT

1. Definición y Aplicaciones del IoT: Investigar y redactar un breve ensayo (250-300 palabras) sobre qué es el Internet de las Cosas (IoT), incluyendo al menos tres aplicaciones prácticas en diferentes industrias.

2. Impacto del IoT: Reflexionar sobre cómo el IoT puede transformar la vida cotidiana y los negocios en el futuro. Identificar un problema actual que podría ser resuelto mediante la implementación de una solución basada en IoT.

- **El Internet de las Cosas (IoT)** es una red de elementos físicos equipados con componentes electrónicos, sensores y actuadores, conectividad y software. Estos elementos capturan, filtran e intercambian datos sobre sí mismos y su entorno para diversos casos de uso y fines comerciales. Aquí te presento algunos conceptos básicos sobre el IoT:
- **Definición del IoT:** El IoT permite conectar objetos cotidianos al Internet, desde bombillas de luz hasta dispositivos médicos, prendas inteligentes y sistemas de ciudades inteligentes. Estos dispositivos pueden ser interruptores (que envían instrucciones) o sensores (que recopilan datos y los envían a otros lugares).
- **Funcionamiento del IoT:** Los sistemas de IoT consisten en dispositivos físicos que reciben y transfieren datos a través de redes inalámbricas, con poca intervención humana. Por ejemplo, un termostato inteligente puede ajustar la temperatura de tu casa basándose en datos de tu automóvil inteligente y la API de Google Maps. El análisis de datos puede realizarse casi de inmediato o a lo largo del tiempo, según la tecnología utilizada.

- **Impacto y aplicaciones:** El IoT tiene aplicaciones en diversos campos, como la salud (dispositivos médicos conectados), la industria (optimización de procesos), la agricultura (monitoreo de cultivos), la logística (seguimiento de envíos) y las ciudades inteligentes (gestión eficiente de recursos). Su impacto potencial incluye mayor automatización, recopilación de datos y eficiencia.

Un problema actual que podría ser abordado mediante una solución basada en IoT es la gestión de residuos sólidos urbanos. En muchas ciudades, la recolección de basura sigue un horario predeterminado que no siempre se ajusta a la cantidad real de desechos generados por los ciudadanos. Esto puede provocar desbordamientos de contenedores, acumulación de basura en las calles y problemas de higiene y salud pública.

Una solución basada en IoT podría consistir en la instalación de sensores en los contenedores de basura para monitorear su nivel de llenado en tiempo real. Estos sensores podrían enviar datos a una plataforma centralizada a través de conexión inalámbrica, permitiendo a los servicios de recolección de residuos programar rutas de recolección más eficientes y basadas en la demanda real.

Los beneficios de esta solución serían múltiples:

- Reducción de la acumulación de basura en las calles.
- Optimización de las rutas de recolección, lo que reduce los costos operativos y las emisiones de gases de efecto invernadero.
- Mejora de la salud pública al reducir la proliferación de plagas y enfermedades relacionadas con la basura acumulada.
- Mayor comodidad para los ciudadanos al garantizar una recolección de basura más oportuna y eficiente.

Implementar una solución basada en IoT para la gestión de residuos sólidos urbanos no solo aborda un problema actual, sino que también contribuye a la creación de ciudades más limpias, saludables y sostenibles.

Parte 2: Introducción a Git y GitHub

1. Creación de una Cuenta en GitHub: Si aún no tienes una, crea

una cuenta en GitHub.

2. Tutorial de Git/GitHub: Completar un tutorial básico de Git y

GitHub. https://classroom.github.com/a/iAxy1U65

3. Informe de Aprendizaje: Escribir un resumen de lo aprendido en

el tutorial.

- **Git** es un sistema de control de versiones distribuido que permite a los desarrolladores colaborar en proyectos de software de manera eficiente y controlar los cambios en el código fuente a lo largo del tiempo.
- **GitHub** es una plataforma de alojamiento de código basada en la web que utiliza Git para facilitar la colaboración en proyectos de software. Permite a los desarrolladores almacenar, compartir y colaborar en proyectos utilizando funciones como repositorios remotos, solicitudes de extracción y seguimiento de problemas.

- **El proceso básico para trabajar con Git y GitHub es el siguiente:**
  - Configuración inicial: Antes de comenzar a usar Git, es necesario configurar algunos ajustes, como el nombre de usuario y la dirección de correo electrónico que se utilizarán para identificar las contribuciones.
  - Creación de un repositorio: Un repositorio es un espacio donde se almacena el código fuente y los archivos de un proyecto. Puede ser local (en la máquina del usuario) o remoto (en un servidor, como GitHub). Para crear un repositorio, se utiliza el comando **git init** para inicializar un repositorio local o se crea uno en GitHub.
  - Agregar archivos y realizar commits: Una vez que se ha creado un repositorio, se pueden agregar archivos al mismo utilizando el comando **git add** y luego confirmar los cambios utilizando el comando **git commit**. Cada confirmación registra un conjunto de cambios en el historial del proyecto.
  - Trabajar con ramas: Git permite a los desarrolladores trabajar en paralelo en diferentes características o versiones del código utilizando ramas. Se pueden crear nuevas ramas con el comando **git branch** y cambiar entre ellas con el comando **git checkout**.
  - Sincronización con un repositorio remoto: Para colaborar con otros desarrolladores o respaldar el trabajo en un servidor remoto, es necesario sincronizar el repositorio local con un repositorio remoto. Esto se hace utilizando los comandos **git remote add** para agregar un repositorio remoto y **git push** para enviar cambios locales al repositorio remoto, así como **git pull** para traer cambios del repositorio remoto al local.

- Colaboración y revisión de código: GitHub facilita la colaboración en proyectos de software mediante funciones como las solicitudes de extracción (**pull requests**), que permiten a los desarrolladores proponer cambios en el código y solicitar su revisión por parte de otros colaboradores.

Git:

- **git init:** Inicializa un nuevo repositorio Git en el directorio actual.
- **git clone <URL>:** Clona un repositorio remoto en el directorio local.
- **git add <archivo>:** Agrega un archivo al área de preparación (staging).
- **git commit -m "Mensaje del commit":** Confirma los cambios y los registra en el historial del repositorio.
- **git status:** Muestra el estado actual del repositorio, incluyendo archivos modificados, agregados o eliminados.
- **git log:** Muestra el historial de commits del repositorio.
- **git branch:** Lista las ramas del repositorio.
- **git checkout <nombre_rama>:** Cambia a la rama especificada.
- **git merge <nombre_rama>:** Fusiona los cambios de una rama en la rama actual.
- **git push <repositorio_remoto> <rama_local>:** Envía commits locales a un repositorio remoto.
- **git pull <repositorio_remoto> <rama_remota>:** Obtiene cambios del repositorio remoto y los fusiona con la rama local.
- **git remote add <nombre> <URL>:** Agrega un nuevo repositorio remoto.
- **git remote -v:** Muestra los repositorios remotos configurados.

GitHub:

- **git remote add origin <URL_repositorio>**: Asocia el repositorio local con un repositorio remoto en GitHub.
- **git push -u origin <rama_local>**: Envía la rama local al repositorio remoto en GitHub.
- **git pull origin <rama_remota>**: Obtiene cambios del repositorio remoto en GitHub y los fusiona con la rama local.
- **git clone <URL_repositorio>**: Clona un repositorio remoto de GitHub en el directorio local.
- **git branch -d <nombre_rama>**: Elimina una rama local después de haber fusionado sus cambios.
- **git checkout -b <nombre_nueva_rama>**: Crea una nueva rama y cambia a ella en un solo paso.
- **git merge <rama_remota>/<rama_local>**: Fusiona cambios de una rama remota en la rama local.
- **git fetch**: Descarga los cambios del repositorio remoto en GitHub sin fusionarlos con la rama local.
- **git reset --hard**: Descarta todos los cambios locales y restaura el repositorio al estado de un commit específico.
- **git revert <identificador_commit>**: Deshace un commit específico creando un nuevo commit que revierte los cambios realizados en él.

Parte 3: Exploración de Herramientas de Desarrollo

1. Instalación de Visual Studio Code (VsCode): Descargar e instalar VsCode desde su sitio oficial.

2. Configuración Inicial: Configurar VsCode instalando las extensiones recomendadas para el desarrollo de IoT, como PlatformIO para desarrollo en Arduino y Python.

3. Primer Proyecto: Crear un archivo simple de texto plano en VsCode y usar Git para hacer commit y push del archivo a un repositorio nuevo en tu cuenta de GitHub.

Entrega:

• El trabajo debe ser presentado en un repositorio con estructura de monorepositorio y formato acorde.

• El ensayo sobre IoT.

• El resumen del tutorial de Git/GitHub.

• El primer proyecto.

• La fecha de entrega será el último día de la semana (14/04), antes de las 23:59 horas.