OBJETIVOS DEL PROYECTO

Las zonas rurales tienen una gran limitación en conectividad, y comunicación. Hay grandes áreas de terreno prácticamente despoblado, en muchas ocasiones muy distante y con baja cobertura de conexión a internet. Hay soluciones en el mercado pero en la mayoría de ocasiones estas requieren pagar una cuota mensual o una conexión a internet constante y cualquiera de estas dos condiciones no siempre se cumplen. Además esta el problema de la brecha digital es especialmente problemática en los entornos rurales.

Pensando en esta situación se han ideado los objetivos de realizar el diseño, implementación y prueba de un sistema capaz de las siguientes acciones:

- Centralizar la recogida de datos en distintos lugares de la geografía y tener estos datos centralizados
- Realizar acciones en esos lugares como forma de retroalimentación de los datos recibidos
- Permitir que se sigan recogiendo los datos aunque la red este caída de forma temporal
- Sea fácil de manejar por las personas que tengan pocas habilidades tecnológicas y pocos recursos para invertir en tecnología

Para cumplir con estos objetivos primarios sera necesario administrar una base de datos distribuida. Esta base de datos debe contar con una central que puede estar replicada todos los datos y además en cada lugar estén almacenados los datos recogidos en ese lugar para que en el caso de que la conexión caiga de forma temporal los datos recogidos en ese tiempo sean mandados a la central cuando la conexión sea restablecida.

También se debe cumplir que los componentes físicos del sistema deben de poderse conseguirse fácilmente, que sean fácilmente sustituibles por otros modelos similares por si llega el momento de no poder ser encontrados en el mercado en un momento dado. Lo cual no es nada raro en la situación actual y probablemente también en el futuro. Para conseguir esto cada parte debe estar aislada lo mas posible del resto y las tecnologías usadas para realizar el código fuente no deben de depender de otros códigos propietarios en la medida de lo posible. Además también se debe conseguir que la central donde se almacenen los datos pueda ser cualquier tipo de servidor u ordenador.

Para poder cumplir con el requisito de que el sistema pueda ser manejado por pocas habilidades tecnológicas y no demasiados recursos se debe conseguir que se pueda acceder al sistema desde casi cualquier dispositivo, consiguiendo de esta forma que se pueda acceder desde cualquier lugar que tenga acceso a internet y que se pueda ver en pantallas grandes. También se debe conseguir que cada usuario solo pueda ver sus correspondientes datos pero que los responsables con suficiente nivel de permisos podrán acceder a todos los datos de los usuarios. El login en el sistema debe poder hacerse de forma fácil, a ser posible que se puedan guardar las credenciales en el dispositivo del usuario pero sin perder seguridad. Además debe ser posible conectarse desde un gran número de sistemas operativos y a ser posible sin ningún tipo de instalación de programas.

Un aspecto muy importante del sistema será la visualización de los datos. Este tiene que mostrar rápidamente los datos mas actuales y además el historial de los datos. Si el proyecto va bien de tiempo se podrán crear mejores gráficas para mostrar los historiales de los datos a lo largo del tiempo. También se podrán crear mejores gráficas de las acciones sobre maquinaria realizadas durante el tiempo. Si hay suficiente tiempo también se podrán crear representaciones propias de las medidas de cada sensor. También se replanteará examinar mas detenidamente la seguridad del sistema descentralizado ya que no es una de las prioridades del proyecto y podría ser necesario revisar la seguridad en caso de poner el proyecto en producción. En el caso de tener el suficiente tiempo sería bastante útil hacer un diseño responsive de la aplicación para poder usarla correctamente en un móvil.

Dadas estas características se ha decidido que la visualización y manipulación de los datos y la maquinaria se realizara mediante una página web. De esta forma se garantiza la distribución de la aplicación en casi todo tipo de sistemas operativos, hardware y pantallas sin que sea necesaria instalación. Dada la cantidad de tecnologías que hay actualmente para desarrollar las páginas web en cada una de sus múltiples capas. Por este hecho hay muchas tecnologías donde escoger en cada capa que son suficientemente independientes de cualquier empresa que puede cambiar de estrategia en cualquier momento.

Lista de requisitos funcionales:

- El sistema controlará el acceso y lo permitirá solamente a los usuarios autorizados. Los usuarios deben ingresar al sistema con un nombre de usuario y contraseña.

- El software podrá ser utilizado en los sistemas operativos Windows y Linux

- La aplicación debe poder utilizarse sin necesidad de instalar ningún software adicional además de un navegador web.

- La aplicación debe poder utilizarse con los navegadores web Chrome, Firefox y Edge

- Cada usuario tendrá un rol perteneciente a un grupo de roles predefinido. Dependiendo del rol tendrá unos permisos y otros dentro de la aplicación. Los roles serán los siguientes: de usuario, de operario y de administrador.

- El rol de usuario tiene los permisos para visualizar los datos de sus propios datos generados por sus sensores en los lugares que tiene asignados.

- El rol de operario tiene los permisos para visualizar los datos de todos los usuarios. Esto tiene el fin de que los datos de todos los lugares estén vigilados y gestionados. Esto puede ser de vital importancia por que puede haber usuarios que no controlas debidamente sus lugares

- El rol de administrador además tiene permisos para gestionar los usuarios. De esta forma no tiene restricciones en las cosas que puede hacer la aplicación. Este es el principal responsable de la aplicación aparte de los desarrolladores.

ESTUDIO DE LAS POSIBLES TECNOLOGÍAS

TECNOLOGÍAS PARA LA WEB: Para elegir las tecnologías de este proyecto primero hay que diferenciar las diferentes partes que lo van a componer. En parte de la página web hay tres capas. El modelo, la vista y el controlador. El modo de diferenciarlos suele depender de la tecnología usada. Actualmente se usa la tecnología de los microservicios. Esta es actualmente la forma mas usada para estructurar las aplicaciones distribuidas. Este es un método usado principalmente para estandarizar las comunicaciones entre las partes de la aplicación que están en diferente lugar y ademas desacoplar unas partes de otras y además desacoplar las diferentes capas.

Otras estructuras anteriormente usadas en las páginas web, es la de generar el código html directamente desde el código ejecutado en el servidor como hacía PHP y otras tecnologías. Actualmente se usan frameworks que consumen código y generan código JavasCript para ser ejecutado en el navegador. Por lo tanto la página web se suele componer de alguno de estos frameworks para la capa de la vista que se ejecuta en el navegador en su mayor parte y la parte que se ejecuta enteramente en el servidor que usa microservicios para la comunicación. Además estos frameworks contienen una buena cantidad de componentes que permiten hacer cosas como simular ventanas emergentes. Hacer estas cosas desde html básico y además que ese código funcionase en todos los navegadores actuales sería terriblemente largo y tedioso.

Por lo tanto se escogerá este tipo de estructura: la de los microservicios. Una vez tomada esta decisión queda decidir que framework se usará para la parte frontal en el navegador y que tecnologías se usarán para generar los microservicios.

Para la parte frontal del navegador estas son algunas de los frameworks mas populares: Angular, React y Vue:

	Angular	React	Vue
Apoyo	Google	Facebook	Un equipo de colaboradores internacional
Filosofías	Tiene muchas características integradas en la base	Es minimalista. Tiene pocas características integradas de base. Por lo tanto suele ser necesario integrar dependencias de terceros	La cantidad de características es intermedia a las otras dos
Sintaxis	TypeScript con template	Mezcla HTML con JavaScript	JavaScript con template
Dificultad de aprendizaje	Alta. Tiene una estructura de proyectos compleja	Alta. Proyecto con estructura propia y HTML y JavaScript mezclado	Menos compleja al crear el proyecto

Por lo tanto la opción escogida es Angular por la mayor cantidad de características que hacen prescindir de la necesidad de la dependencia de terceros.

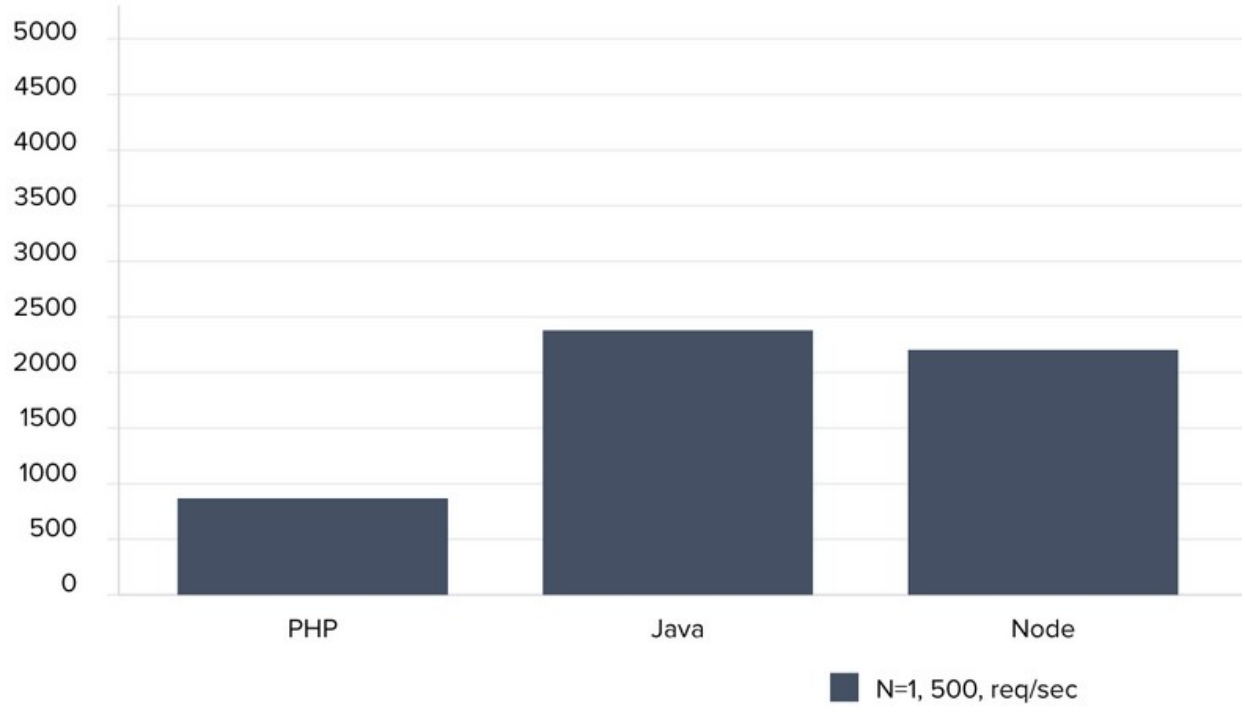
TECNOLOGÍAS PARA MICROSERVICIOS: En el caso de la generación de microservicios también hay que escoger tecnologías. En este caso unas buenas opciones de estudio son node.js, java y PHP. Esta comparación tiene que ser pensada exclusivamente en el tratamiento de microservicios y no en las otras capacidades que tiene cada uno de estos lenguajes.

En esta tabla se pueden ver algunas diferencias de estos lenguajes. Para tener esta tabla clara hay que diferenciar entre hilos y procesos. Un proceso es la instancia de un programa en ejecución. Es administrado por el sistema operativo, si un proceso se queda esperando a una operación de entrada/salida como un acceso a una base de

datos, fichero o a una red y en ese intervalo de tiempo esta ocupando recursos de memoria y de procesados. En cambio un solo proceso puede manejar muchos hilos. Estos hilos comparten parte de la memoria del proceso, por lo que comparten recursos y tiempo de ejecución de procesador. Si un hilo se queda esperando a un proceso de entrada/salida los demás usan el tiempo de procesador que queda libre. Cuando una llamada llega a Node.js o Java estos las responden mediante la ejecución de un hilo perteneciente al un único proceso de programa, mientras que en PHP se genera un nuevo proceso consumiendo mas recursos por cada llamada.

Lenguaje	Hilos vs Procesos	No-Bloqueos	Facilidad de uso
Node.js	hilos	Sí	Requiere llamadas
Java	hilos	Sí	Requiere llamadas
PHP	procesos	No	

Este es un gráfico de rendimiento con una prueba de estress con 500 llamadas a la vez. La gráfica representa el número de llamadas respondidas por segundo. Por lo tanto los mayores valores son los mejores:



Con estos datos y esta gráfica el caso de PHP queda descartado, ya que su rendimiento es menor con una gran alta carga de trabajo dado sus bloqueos de entrada/salida en los multiprocesos.

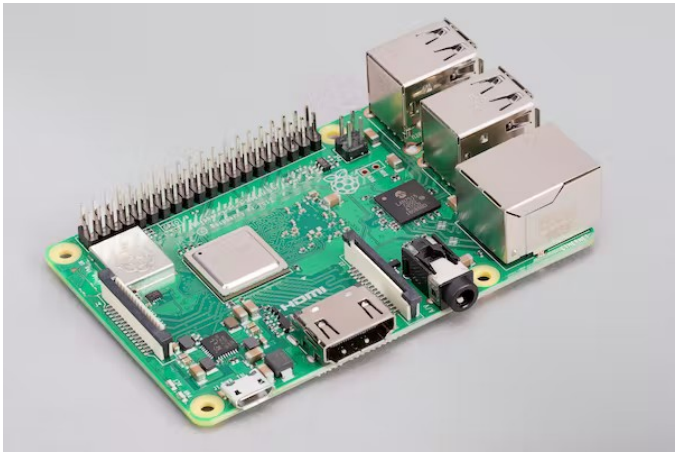
Llegamos a la conclusión de que con una gran carga de trabajo Java y Node son parecidos. Cada uno tiene sus ventajas sobre el otro: Java tiene mayor facilidad de uso ya que es un lenguaje fuertemente tipado mientras Node que usa código JavaScript no es tipado y eso es una desventaja en unas ocasiones como programas complejos pero también puede ser una ventaja para programas simples. Por el otro lado JavaScript y TypeScript son casi el mismo lenguaje y eso es una gran ventaja ya que usaremos lenguajes muy parecidos tanto en el servidor como en el navegador, y esto nos facilitará mucho las cosas. También hay que tener en cuenta que Node.js usa una licencia MIT que es mas libre que la usada por Java. Además JavaScript usa JSON de forma nativa y este es el formato en el que se manejan los datos la mayoría de las veces en los microservicios, por lo tanto esto nos facilitará mas las cosas. Por lo tanto el lenguaje elegido para realizar los microservicios sera JavaScript.

Dentro de JavaScript en el lado del servidor hay que elegir tencologías para implementar los microservicios y acceder a las bases de datos. Además hay que escoger alguna tecnología para manejar la base de datos que use la plataforma web para persistir los datos. La base de datos de la página será la misma que se use como central en el proyecto. Se ha decidido por el conocimiento previo de las tecnologías usar Express para levantar los microservicios y Sequelize como tecnología ORM para acceder a la base de datos. Estas usan la licencia MIT por lo que se pueden usar incluso de manera comercial.

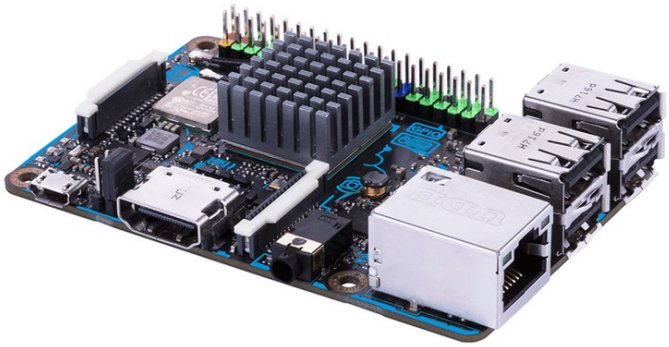
TECNOLOGÍAS PARA RECOPILAR LOS DATOS: como se ha comentado antes este va a ser un sistema distribuido y los datos se leerán en un lugar remoto y como en ese lugar la conexión puede fallar potencialmente los datos se guardarán en el lugar donde sean tomados. Por lo tanto necesitamos un hardware que sea capaz de mandar y recibir datos de la red, almacenar datos de forma relativamente fiable y de comunicarse con otros pequeños sistemas. Además es muy importante que sea barato, que consuma poca electricidad y que sea fácilmente reemplazable por si hay falta de stock de esa plataforma en el mercado, tanto actual como futura. Es conveniente que este hardware

sea capaz de ejecutar Node.js de forma eficiente para usar el mismo lenguaje en las comunicaciones con los microservicios. Estas son las alternativas:

Raspberry Pi: es el micro ordenador actual mas famoso y con mas soporte con mucha diferencia. Además el precio recomendado es uno de los mas competitivos. El resto de alternativas están en mayor o menor medida basadas en este. Desde casi todos los puntos de vista esto hace que sea la mas recomendable. En el aspecto negativo esta la actual disponibilidad: debido a la falta de chips su precio se ha multiplicado y además por la falta de stock es difícil conseguir una. Pero su larga trayectoria y la futura estabilización del mercado debería hacer que volviera a haber stock a precios aceptables. Y aunque esto no ocurriera sería relativamente sencillo reemplazarla por muchas de sus competidoras.



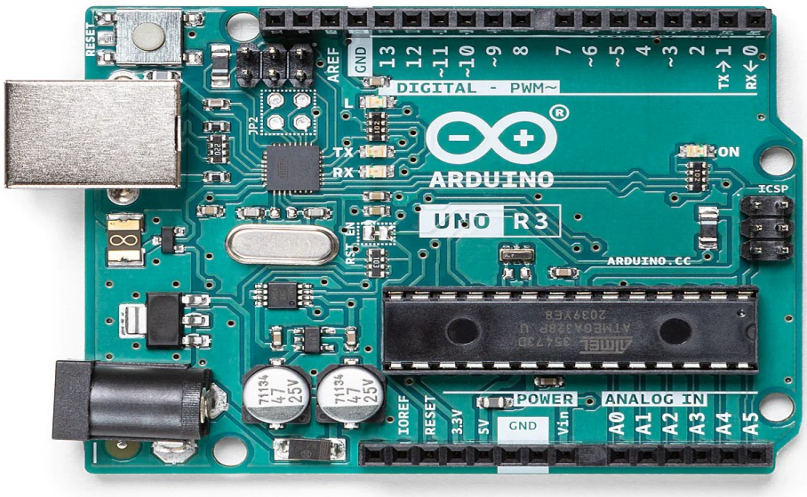
ASUS Tinker Board S: es una alternativa mas cara y potente. Para este proyecto no será necesaria esa potencia extra pero es una opción a tener en cuenta. Hay que tener en cuenta que no necesita una tarjeta SD para funcionar ya que tiene su propia memoria de 16GB por lo que se podría descontar la tarjeta SD del precio.



Orange Pi: una alternativa muy similar a Raspberry pi en características y a un precio muy competitivo. También se puede prescindir de tarjeta SSD y que sería su sucesora mas inmediata.

Como se puede ver en las imágenes los micro ordenadores tienen patillas que no suelen tener un ordenador normal. Estas patillas se pueden configurar como entradas o salidas digitales. Esto es una opción muy buena y para entradas o salidas digitales es posible usar fácilmente baratos convertidores digitales a analógicos y viceversa. Sin embargo se ha barajado y aceptado otra opción mas aconsejable para un proyecto grande: usar otro componente intermedio a los sensores para proteger al micro ordenador de sobre voltajes y no complicar el montaje con muchos componentes diferentes.

Arduino: es una pequeña placa barata y fácil de encontrar y si es necesario de reemplazar. Tiene múltiples formatos que se pueden escoger según las necesidades del proyecto. El modelo mas común y el que usaremos en el proyecto es el de la foto:

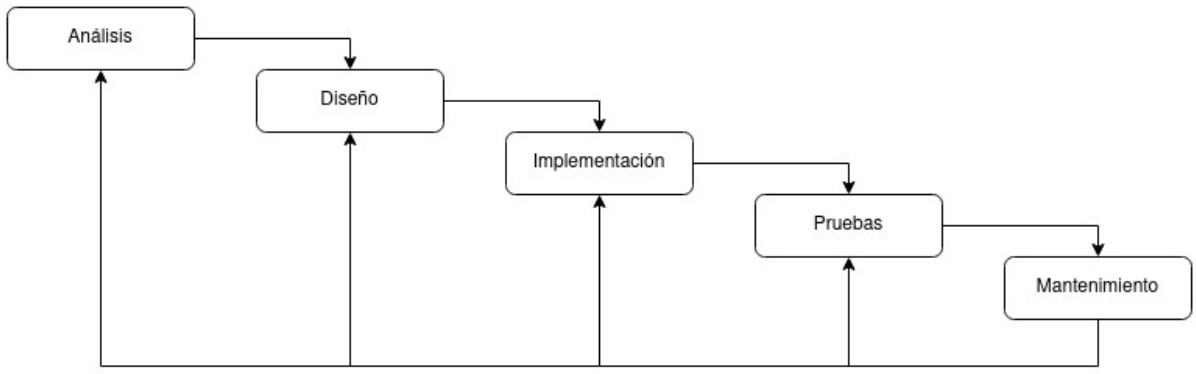


Hay varios motivos para usar esta placa como ampliación de un micro ordenador con entradas y salidas integradas. Además de contar directamente con entradas y salidas analógicas es mas barato y funciona a una frecuencia muchísimo mas baja que un micro ordenador. Estas dos características hacen que al sufrir sobre tensiones el reemplazo de hardware sea mas barato y como su frecuencia y consumo son mucho menores es posible estar leyendo las entradas constantemente y captar casi cualquier cambio en los valores aunque estos sean muy cortos en el tiempo sin sobre calentarse.

METODOLOGÍA PARA EL DESARROLLO

Existen varias metodologías para desarrollar software. Incluso varias clasificaciones. Estas son las metodologías mas comunes y sus ventajas e inconvenientes para este proyecto:

Metodología en cascada: en esta metodología no empieza cada fase del desarrollo hasta haber terminado la anterior. En cada fase se tratan todas las funcionalidades del programa. Para añadir un poco mas de flexibilidad se introdujo la opción de volver para atrás en las etapas de desarrollo. A pesar de esto la metodología sigue siendo muy poco flexible. Hay mucho tiempo transcurrido entre que la mayor parte de las funcionalidades son planificadas hasta que son realizadas. Por lo tanto cuando estas funcionalidades son implementadas ya pueden no tener interés o puede que no quede tiempo para realizarlas y su planteamiento haya sido un gasto de tiempo inútil. Esta metodología es útil en la actualidad para proyectos relativamente pequeños y cerrados que se quedarán casi concluidos al implantarlos. Este proyecto no es lo suficientemente pequeño para esta metodología así que no se usará esta. Se utilizará alguna metodología ágil que usará ciclos en espiral definidos al comienzo.



Metodologías ágiles: están basados principalmente para separar funcionalidades en orden de de desarrollo e importancia. De forma que cada una de estas partes se planea y se desarrolla dejando al resto para planificaciones posteriores. Estas son las metodologías mas comunes aunque siempre hay muchas variantes de cada una:

Extreme programming: normalmente abreviado a XP. Esta basado en varias elementos: usar varias cascadas seguidas, integrar al cliente durante muchas horas para interactuar con el cada cambio en el programa y adecuarse a cualquier cambio, en muchas ocasiones usar dos desarrolladores a la vez programando en un solo ordenador entre otras. Es aconsejado para desarrolladores con experiencia. Dadas las características de este proyecto no es la más aconsejable.

Kanban: esta basado en tarjetas repartidas en columnas. En cada tarjeta esta una funcionalidad o cosa por hacer. Estas tarjetas se reparten en columnas con títulos como los siguientes: Bloqueadas, Realizadas, Por hacer. Este es un método de desarrollo donde la comunicación con el cliente se basa en pequeñas tareas que pueden repartirse en tiempos muy diferentes y con distinto nivel de prioridad. No es el caso de este proyecto aunque usar estas tarjetas podría ser útil.

Scrum: probablemente el mas popular de los tres. Tiene un gran número de variantes. Esta compuestos por los denominados “sprints” que consisten en una iteración con todos los pasos del software para una cantidad de funcionalidades determinadas y que tienen una duración determinada. Al inicio de cada uno de estos sprints se acuerda con el cliente los elementos a desarrollar en ese sprint y cuando termina se le entrega el resultado. Hay convenciones de roles de cada persona del proyecto. Escogeremos esta metodología ya que es muy versátil y la dirección es muy clara y los resultados son rápidamente visibles por el cliente.

Diagrama de Gantt