Documentación de Pasantía Digitalización Industria 4.0

Por: Sebastian Ruiz

Introducción

Esto va a ser un documento que documenta mi experiencia, y mas específicamente el proceso de la pasantía de la cual participe durante junio y julio del 2025. Esta pasantía consistió de un proyecto dividido en 5 etapas en las cuales yo tenia que digitalizar un aparato industrial de la forma que se usaría en la industria. Para guiar este proyecto, se me presentó el siguiente documento con el siguiente objetivo general:

> Diseñar e implementar un sistema tipo Industria 4.0 basado en ESP32 para la digitalización de un activo industrial. El sistema deberá monitorear y controlar entradas y salidas, sensar variables del entorno y publicar información en tiempo real a través de un servidor web y/o protocolos IoT (MQTT, OPC UA).

Etapa 1 - Familiarización con el hardware y servidor web

Ya que yo nunca trabajé con el sistema de arduino, esta primer etapa era para familiarizarse con el sistema de programación de arduino junto con el esp32 que había comprado, y el plc que iba a estar usando para el proyecto.

Para hacer esto empece al descargar lo documentación para ambas plaquetas, y buscando paginas web que describían sus funcionamientos electrónicos. Yo utilice las siguientes paginas web y pdf: modulo relay datasheet, modulo de relay manual, devkit board manual, devkit board sitio web 1, devkit board pinout, y modulo de relay videos de configuración.

Con esto pude identificar el modelo de la plaqueta de industrial shields, y la plaqueta que vino con el kit que compré, familiarizarme con la terminología usada para ambas plaquetas (nombres de pines), las capacidades de plaquetas, y que adentro son plaquetas idénticas, y que hay pines que corresponden a cada uno con distintos nombres que identifica el arduino ide.

Para empezar a ver como es que se conecta a el plc, instale el software y las bibliotecas requeridas que se describieron en el manual del plc. Ademas, descubrí que el plc tiene 10 entradas o salidas que se pueden configurar individualmente para cumplir los objetivos de el dispositivo.

Para trabajar con la placa devkit, seguí el siguiente guía para configurar los drivers y use otro guía para aprender como era el sintaxis de arduino (una especie de C/C++). También compilé código de ejemplo para fijarse que todo lo que había hecho hasta ahora funcionó.

Después, para aprender como es que trabaja con un servidor web y web sockets (protocolo que automáticamente notifica todos los clientes conectados cuando hay algún cambio en estado) empece a

leer un guía que explicaba como funciona, y termine con un código que controla el led incluido en el esp32 por wifi, y mostraba su estado en ese mismo panel de control.

Ya que cumplimos el objetivo de un “microcontrolador funcional con interfaz web local” podemos seguir a la etapa 2

Etapa 2 - Sensado de variables analógicas o ambientales

Esta etapa se trata de leyendo variables y registrar/visualizarlas por arriba de el control de salidas que se hizo en la etapa 1.

Ya que el kit que compre vino con un sensor de temperatura y humedad, empece por testeando como era que funcionaba el sensor, y como una hace para interactuar con el sensor adentro del código. Ya que mi sensor era el dht11, lo hice con el siguiente código, pero existen códigos de prueba parecidos para otros tipos de sensores.

Después, empece a leer un guía que mostraba como mostrar datos en vivo en una pagina web con web sockets, lo cual en combinación con el código de ejemplo que venia con la documentación del kit de ejemplo me hizo entender como es que funciona el tema de la transmisión de datos de forma sincroniza, ya que tiene que saber cuan frecuentemente tomar mediciones de el sensor.

Ahora que estaba todo al día de como es que uno interactúa con las entradas y salidas por servidor web, estaba listo para ir a la etapa 3.

Etapa 3 – Comunicación con el exterior (IoT)

Como se podrá inferir por el titulo, esta etapa se trata de interactuar con el exterior por forma inalámbrica con protocolos de IoT por wifi como MQTT, y el uso de node-red en conexión con esto para hacer un dashboard y luego en la etapa 4 publicar los datos a algún sistema de registro largo plazo.

Yo empece este proceso al instalar el bróker de MQTT Mosquitto con el siguiente guía que encontré para Windows, y usando una pregunta de stack overflow, pude asegurarme que estaba funcionando bien el bróker.

Después instalé MQTT Explorer para monitorear lo que se escribe en los temas o “topics” del bróker. Después hice pruebas de cambiando el estado de los topics desde la linea de comando con:

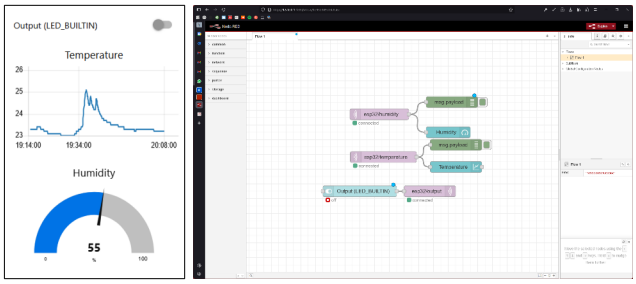
```
mosquitto_pub -h localhost -t prueba/chat -m "off"
mosquitto_pub -h localhost -t prueba/chat -m "on"
```

Y se podían ver los cambios desde el MQTT Explorer o desde otra ventana de linea de comando con:

```
mosquitto_sub -h localhost -t prueba/chat -v
```

A continuación, agarré el código de el siguiente tutorial y lo modifique para que funcione con mi sensor, y que haga lo que quería que haga (postear humedad y temperatura a topics, y controlar una luz

en la plaqueta al escuchar a otro topic), terminando con el siguiente código con el cual pude interactuar por MQTT Explorer.

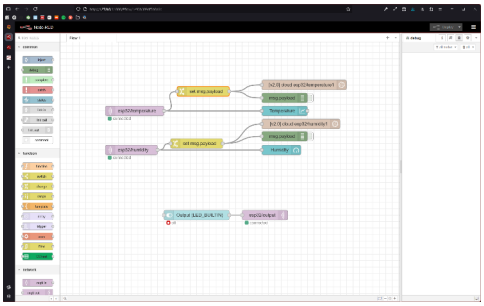


Después de haber hecho eso, empece el proceso de instalar node-red con el siguiente guía, y node-red dashboard para visualizar los datos que venían de forma sincrónica de el esp32, y seguí el guía que había usado para hacer código que escribía a los topics MQTT para también hacer un flow que también hostea un dashboard para mostrar los datos en tiempo real como se ve en las imágenes a la derecha.

Etapa 4 - Registro y análisis de datos

Como la ultima etapa en remoto, esta etapa se trata de tomar los datos que conseguimos de otros lados, y registrarlo en una base de datos en la nube que después se pueda usar para visualizar los datos y generar “gráficas, alarmas y tendencias”.

Como esquema general, yo decidí que quería un sistema donde los datos se guardan en InfluxDB Cloud, y se visualizan en Grafana. Para cumplir esto, el primer paso era conectar InfluxDB a Node-Red, lo cual hice con el siguiente tutorial que incluía generar llaves api.



Durante el proceso de generar gráficos con un query en InfluxDB, tenia un problema que no podía ver el grafico, pero podía ver los valores de temperatura y humedad sincronizados sin problema. Este problema resulto ser la culpa del mqtt, quien solo transmite valores en forma de un “string”, y para arreglar esto había que convertirlo de un “string” a un “float”, que pude hacer siguiendo un forum de Node-Red de este tema. El flujo Node-Red final es el siguiente, y esta a la derecha.

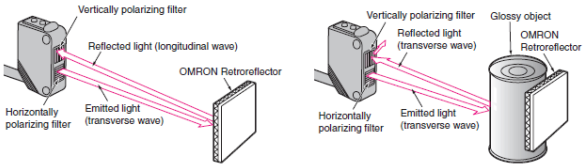
A continuación, empece el proceso de conectar la base de datos (InfluxDB) a el sistema de visualización de Grafana. Esto lo empece con otro tutorial de YouTube, y después leí la documentación para ver como es que uno tiene que hacer el query en Grafana para mostrar solo un tipo de data a la vez. Con esto completado, ahora era un simple proceso de ponerle unidades a todo, y seleccionar el tipo de gráficos que quería para visualizar mi data, que resulto en el siguiente dashboard también demostrado a la derecha.

Para mostrar las capacidades de el programa de Grafana, también arme una alarma de ejemplo que me mande un correo electrónico si la humedad excede 80%, que pude comprobar al respirar encima, y funcionó! También aprendí que uno puede cambiar una multitud de variables para configurar como es que funciona el sistema de alarmas, sea duración, mínimo o máximo, si es un valor sustenido, como mandar la alarma, etc.

Etapa 5 - Digitalización de un activo real

Ahora que estaba bien familiarizado con todas las herramientas que iba a necesitar, y presente de forma presencial en el laboratorio, podía elegir el equipo que iba a digitalizar (monitorear y controlar entradas y salidas, sensar variables del entorno y publicar información en tiempo real a través de un servidor web y protocolos IoT).

Como posiblemente ya sepas, yo decidí digitalizar una cinta corrediza de modelo que ya tenia armado un tablero de control, y sensores réflex polarizados (mostrados a la derecha) en el comienzo y el final como se encontraría en una escena industrial.



Para empezar, lo primero que hice era pensar que exactamente quería que haga el dispositivo cuando lo termine de armar, y hice la siguiente lista:

- Interactuar con el motor
  - Prendido/Apagado
  - Dirección
- Interactuar con sensores de interferencia
  - Detectar estado de interferencia para ambos sensores
  - Cambiar el estado del motor basado en el estado de los sensores
- Interactuar con el servidor web
  - Mostrar el estado de el motor y los sensores
  - Controlar el motor directamente, y como es que el motor interactúa con los sensores

Ya que el PLC solo tiene salidas en relay, había que cablear a el motor con lógica de control con dos relay, que esta explicado en un diagrama a la derecha. La idea es que uno puede controlar la dirección de un motor y si esta prendido o apagado con dos relay donde la corriente pasa en formas opuestas si un relay esta apagado y el otro prendido y de la forma opuesta. Esto se puede describir mejor con una tabla de condiciones donde un 0 lógico esta definido por ninguna diferencia en potencial entre las terminales de entrada, y un 1 lógico esta definido por 24v de diferencia de potencial entre las terminales de entrada:

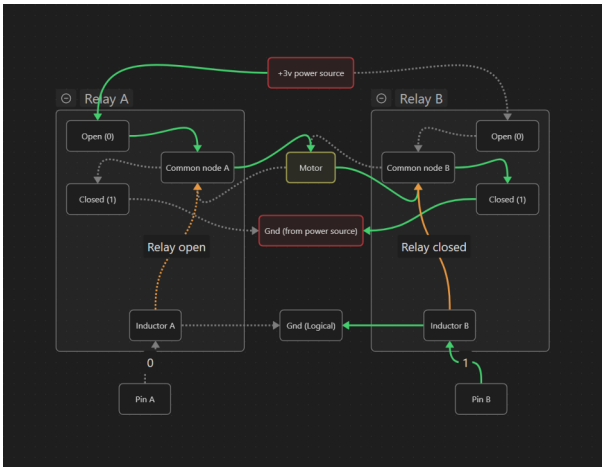
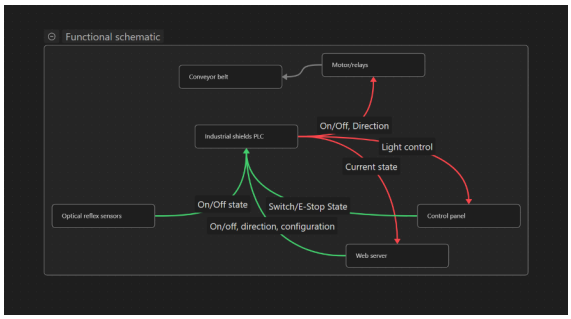


Figure 1: Cuando Pin A=0 y Pin B=1, motor se mueve para adelante

Pin A	Pin B	Estado
0	0	Apagado
0	1	Adelante
1	0	Reversa
1	1	Apagado

Después de hacer mis requisitos de que quería que haga el dispositivo, armé un diagrama funcional mostrado a la derecha para mostrar como es que iban a interactuar los componentes, y darme una idea de como voy a asignar las entradas y salidas cuando vaya a cablear todo el proyecto.



Sabiendo bien lo que quería hacer con el proyecto, me puse armar el diagrama mostrado a la derecha para mostrar como es que se iban a conectar físicamente todos los componentes y asignar los 10 puertos de entrada o salida que tenia, ya que son limitados. Esto también se hizo para simplificar el proceso de cablear y después programar, ya que yo se a donde esta conectado cada cable sin tener que seguirlos para ver donde esta conectado después.

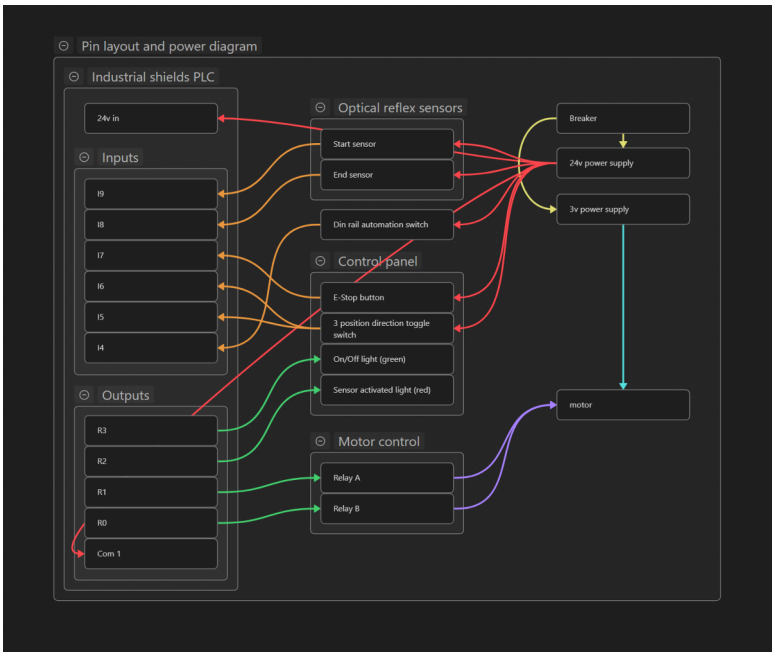


Figure 2: Todos los diagramas de este estilo fueron hechos en obsidian canvas con el plugin "advanced canvas"

Ya que toda la parte teórica estaba completada, había llegado la hora de físicamente cablear todo como fue planeado en los diagramas. Para hacer esto desarme todo lo que se había hecho hasta ese entonces con respecto las borneros redundantes en la tabla de la cinta corrediza, y después arme todo siguiendo el diagrama en figura 1 y 2.

Ahora para la etapa de programación. Empece prográmandolo para que interactué con todas las partes físicas del proyecto con lógica siguiendo los siguientes requisitos:

- Con el botón de emergencia: parar la cinta y parpadear la luz roja.
- Mover la cinta para adelante y para atrás con la cinta hasta el final de la correa en la dirección que esta apuntado el interruptor de 3 posiciones, y prender luz verde cuando este en movimiento la cinta.
- Mostrar la presencia de un objeto interrumpiendo el sensor en cualquiera punta de la cinta con una luz roja continua.

Esto lo hice basado en funciones para que sea mas rápido de escribir y fácil de leer, resultando en el siguiente código.

Después, empece el proceso iterativo de programar la parte de el proyecto que habilita el aspecto del servidor web con la inteligencia artificial (solo empece a usarlo ahora por el tema que uno no sabe lo que no sabe, entonces para la etapa de aprendizaje quiero tener que aprender lo mas posible, y ahora puedo entender hasta cierto punto lo que hace el código al leerlo).

Este proceso empezó con un pedido muy especifico para tirarme en la dirección que necesitaba, para después ir diciéndole lo que quiero que haga distinto o que cambie (esto es importante porque la inteligencia artificial te da la respuesta mas probable de acuerdo a tu mensaje, y no te puede leer la mente). Esto demostraba el uso de entradas y salidas, el access point, el servidor web, websockets, y el monitoreo de las entrabas, todo por wifi!

Esto después se fue elaborando con una serie de mensajes muy específicos que:

- Resolvió problemas con que no se actualizaban los estados de variables
- Agrego la funcionalidad de configurar la conexión a una red de wifi con un access point que genera el esp.
- Cambio la forma de refrescar la pagina a cuando haya un cambio en estado en vez de cada cantidad de segundos (lo hace mucho mas rápido sin sobrecargar el procesador, también mas eficiente.
- Agregar secciones a el código que dejarían que se agreguen otras funciones.
- Agregaron una visualización de el estado físico de la cinta (como el interruptor, luces, sensores, etc.)

La progresión de el código se puede ver a continuación:

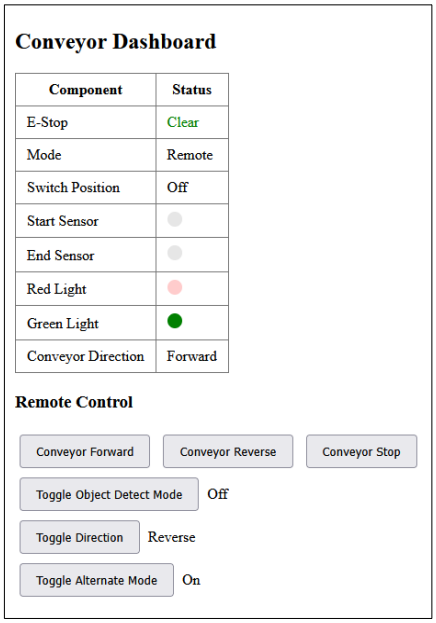
- Primera prueba de servidor web
- Versión mas nueva con “edge detection” de cambio de estado
- Versión con mejor interfaz visual

Después seguí trabajando con la IA para desarrollar y sacar problemas del código. La lista de requisitos que tenia para esta etapa del código (y que le iba dando a la IA de forma iterativa) eran las siguientes:

- Mostrar el estado de las entradas y las salidas en la forma de una tabla para cada variable.
- Al encenderse, hostear un access point que hostea un servidor web que permite que el usuario que se sube a ese servidor web pueda tipear las credenciales necesarias para entrar a la red wifi que se necesita, conectarse a el mismo tiempo, y decir en el servidor web local cual es el ip que tiene el segundo servidor web que esta en la otra red, para que después el usuario pueda tipear ese ip en la otra red, y interactuar con el servidor web.
- En el servidor web que esta en la red externa, conectar por web sockets para que cambios iniciados de el lado del servidor o el cliente resulten en un refresco del estado.
- Agregar un interruptor a el riel din que cambie entre el modo remoto y el modo local. El modo local solo deja que el servidor web monitoree lo que esta pasando con la cinta corrediza, y todos los periferos, mientras el modo remoto deshabilita la llave interruptora de dirección en la cinta corrediza, y agrega un panel de control a la pagina web. Este panel debería dejar que el usuario controle la dirección de la cinta explícitamente de los sensores, dejar que elige en que dirección este yendo el control de los sensores (en que punto parar/arrancar la cinta por resultado de los sensores), y agregar un modo que “rebota” el objeto de punta a punta.

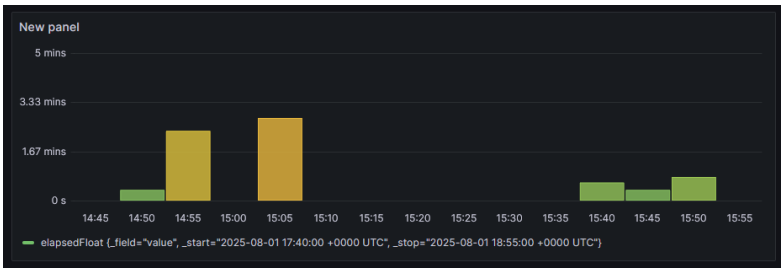
Para darles una idea de el nivel de detalle con el cual uno tiene que describirle las instrucciones a la inteligencia artificial, estas son las instrucciones que le di a Grok para que haga esta parte de la etapa.

Esto termino con la primera mitad del código que solo monitoreaba el estado, y la segunda mitad que cambiaba entre monitorear y controlar con el interruptor en el riel din. Esto se puede ver a la derecha con la parte de monitorea en la tabla de arriba, y la parte que controlaba la cinta de forma remota abajo.



La parte final de este proyecto es de conectar este proyecto a el aspecto de MQTT y el registro de data que después se puede usar para todo tipo de servicios como se ve en la “industria 4.0”. Para hacer esto, use lo que había aprendido durante la etapa 4 del proyecto para conectarlo a el Mosquitto, Node-Red, InfluxDB, y finalmente Grafana.

Para demonstrar el uso de la parte de la industria 4.0 que se enfoca en el registro de datos en la nube para proveerle servicio a el cliente, yo decidí que quería que la cinta mande el estado de movimiento en forma de “on”, y “off” cuando hay un cambio de estado (no constantemente), para después poder calcular cuanto tiempo estuvo prendida la cinta, y teóricamente poder proveer servicios después de cierta cantidad de tiempo prendido (parecido a cambio de aceite después de x cantidad de km de viaje).



Esto lo hice de vuelta con la IA, donde le pedí específicamente que solo mande la condición cuando haya un cambio de estado, resultando en el siguiente código final

```
from(bucket: "esp data test")
 |> range(start: v.timeRangeStart, stop:
v.timeRangeStop)
 |> filter(fn: (r) => r["_measurement"] ==
"conveyor/motor_state2")
 |> filter(fn: (r) => r["_field"] == "value")
 |> elapsed(unit: 1s)
 |> map(fn: (r) => ({ r with elapsedFloat:
int(v: r.elapsed)/1 }))
 |> filter(fn: (r) => r["_value"] == 0) // add
a filter statement after the elapsed function
to filter out only those whose _value = 0.
This is because the “active” time (when the
state 1 goes to state 0) is captured in those
records where _value = 0. If you wanted to
know the time when the phone is NOT active,
you would set the filter where _value = 1.
 |> aggregateWindow(every: 5m, fn: sum,
column:"elapsedFloat")
 |> yield(name: "sum")
```

Para que quede una solución mas permanente y posible de demonstrar en el futuro, decidí configurar el raspberry pi que había en el laboratorio para que se utilice como el bróker de mqtt, y que le mande los datos de prendido y apagado por Node-Red a InfluxDB.

De ahí tuve que conectar InfluxDB a Grafana, para el cual usé el query que conseguí en el siguiente posteo de forum para generar el query que se puede ver a la derecha para mostrar la data de la forma mas relevante y útil.

Conclusión

Gracias por leer!

Al fin, después de demasiadas horas adelante da la pantalla y muchas cosas nuevas aprendidas, esta terminado!

Si quieres ver un video de la demostración de la cinta en funcionamiento con el control remoto y local, esta enlazado.