Financiado por la Unión Europea
NextGenerationEU
MINISTERIO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y DEPORTES
Plan de Recuperación, Transformación y Resiliencia
SECRETARÍA GENERAL DE FORMACIÓN PROFESIONAL

ENERGIZANDO EL FUTURO: INNOVACIÓN EN BATERIAS PARA LA INDUSTRIA 4.0

Proyecto de Innovación Aplicada y Transferencia de Conocimiento en la Formación Profesional

TUTORIAL Nº5:

IOT proyecto Robot Colaborativo OMRON TM5-S

Antonio García

1 Alcance

El objetivo de este tutorial es implementar la tecnología IOT para extraer datos del robot colaborativo OMRON TM5-S y posteriormente poder acceder a ellos desde cualquier parte mediante router 5G.

2 Introducción

El presente tutorial tiene como finalidad servir de guía práctica y educativa dirigida al alumnado del Ciclo Formativo de Grado Superior en Automatización y Robótica Industrial. Su objetivo principal es facilitar el aprendizaje autónomo y guiado en el uso del software NODE-RED, herramienta de código abierto para desarrollar aplicaciones IOT.

A lo largo de este documento, los estudiantes adquirirán los conocimientos necesarios para desarrollar un proyecto completo desde cero, abarcando todas las fases fundamentales: desde la configuración inicial del entorno de trabajo hasta la implementación, programación y puesta en marcha de una aplicación robótica funcional. El enfoque del tutorial es eminentemente práctico, basado en una metodología paso a paso que permite consolidar los conceptos mediante la realización directa de tareas.

Además, este material busca fomentar competencias clave en el ámbito de la automatización industrial, tales como la resolución de problemas, la lógica de programación, la integración de sistemas y la comprensión del funcionamiento de robots colaborativos en entornos productivos reales. De este modo, se pretende acercar al alumnado a situaciones similares a las que encontrará en el ámbito profesional, mejorando su empleabilidad y capacidad de adaptación tecnológica.

El tutorial está diseñado para ser utilizado tanto en el aula como de forma autónoma, requiriendo unos conocimientos básicos previos en automatización y programación industrial, aunque se ha estructurado de forma progresiva para facilitar su seguimiento por parte de todos los estudiantes.

3 Descripción general del sistema

El sistema desarrollado en este tutorial consiste en la creación e implementación de una aplicación mediante el uso del software NODE-RED, para extraer los datos del robot OMRON TM5-S mediante el protocolo de comunicaciones Modbus-TCP y almacenarlos en una base de datos PostgreSQL. El objetivo principal es que el alumnado sea capaz de desarrollar un proyecto completo desde cero desde la programación de los nodos de Node-Red como la arquitectura general de la aplicación.

El proyecto se basa en un entorno de programación gráfica que permite definir secuencias de trabajo mediante bloques funcionales, integrando movimientos del robot, control de entradas y salidas, y estructuras de decisión. A través de este enfoque, se simulan o ejecutan tareas típicas del ámbito industrial, como operaciones de manipulación, posicionamiento o automatización de procesos sencillos.

El sistema está compuesto por los siguientes elementos principales:

- Robot colaborativo (cobot) OMRON TM5-S: encargado de ejecutar las acciones programadas.
- Software TMFlow: entorno de desarrollo, simulación y ejecución del programa.
- Software NODE-RED
- Elementos de entrada/salida: sensores y actuadores que permiten la interacción con el entorno. En este caso está compuesto por una pinza neumática SMC-MHZ2
- Entorno de trabajo: espacio donde se define la escena, incluyendo objetos, herramientas y referencias. En este caso está compuesto por la mesa de trabajo del robot, el alimentador de baterías a granel y el blíster de colocación de baterías.

4 Requisitos previos

Para el correcto seguimiento de este tutorial, se recomienda que el alumnado disponga de los siguientes conocimientos y recursos previos:

- Conocimientos básicos de automatización industrial, especialmente en lógica de procesos.

- Conceptos básicos de programación, como secuencias, condiciones y bucles.
- Nociones generales sobre robots industriales o colaborativos.
- Manejo básico de entornos informáticos en sistemas Windows.
- Haber realizado los pasos descritos en el "*Tutorial Nº1: Instalación y configuración Robot colaborativo Omron TM5-S"*.
- Haber realizado los pasos descritos en el "*Tutorial Nº2: Creación de un proyecto en software TMFlow. Primeros pasos"*.
- Haber realizado los pasos descritos en el *"Tutorial Nº3: Tareas de visión software TMFlow"*.
- Haber realizado los pasos descritos en el *"Tutorial Nº4: Integración proyecto Robot Colaborativo OMRON TM5-S"*.

5 Materiales BOM (Bill of Materials)

A continuación, se detallan los materiales necesarios para la realización del proyecto. En función de si se trabaja en modo simulación o con equipo real, algunos elementos pueden ser opcionales:

- Robot colaborativo OMNRON TM5-S
- Ordenador portátil o de sobremesa con capacidad para ejecutar TMFlow y NODE-RED.
- Cable de red Ethernet para conexión con el robot (en caso de uso real)
- Elementos de sujeción o herramientas del robot (pinza, ventosa, etc., si aplica)
- Objetos de trabajo (piezas para manipulación en ejercicios tipo pick & place)
- Fuente de alimentación y cableado básico, en caso de trabajar con periféricos

6 Desarrollo del proceso

En este apartado se guía paso a paso en la instalación del software NODE-RED, los conceptos básicos del mismo y en el desarrollo de la aplicación de IOT para extracción de datos

6.1 Instalación del software

Paso 1: Descarga

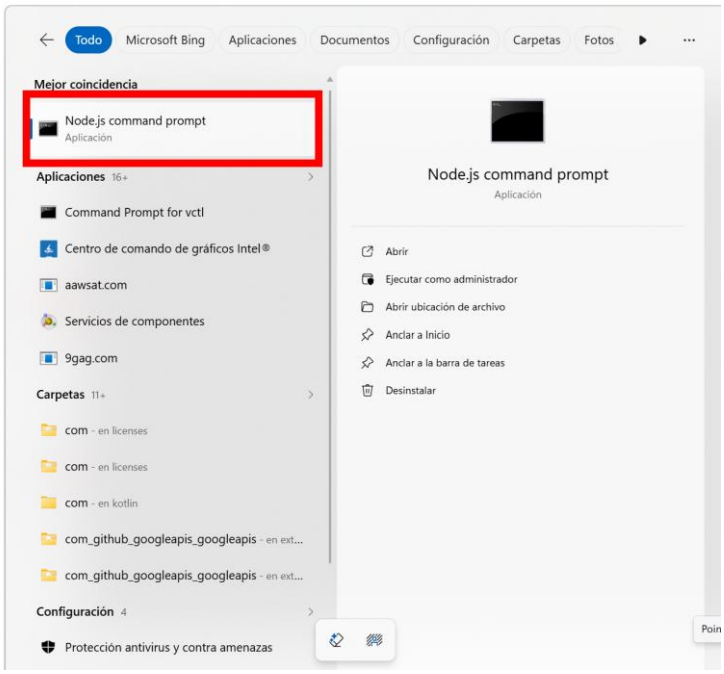
- Accede al portal oficial de NODE-RED o al repositorio autorizado proporcionado por el centro educativo.

https://nodered.org/docs/getting-started/windows

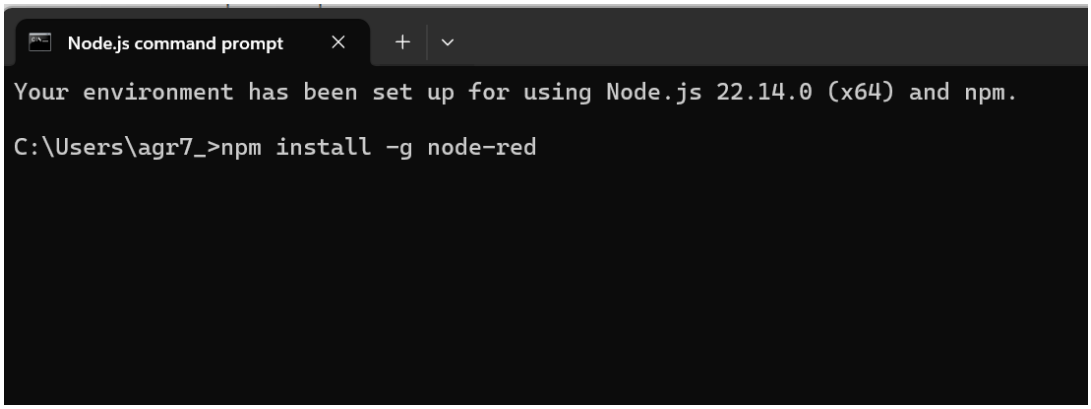
- Selecciona el sistema operativo de tu ordenador y sigue las instrucciones al respecto.
- Descargar e instalar el software Node.js que es el entorno de desarrollo donde se basa NODE-RED.

https://nodejs.org/en/

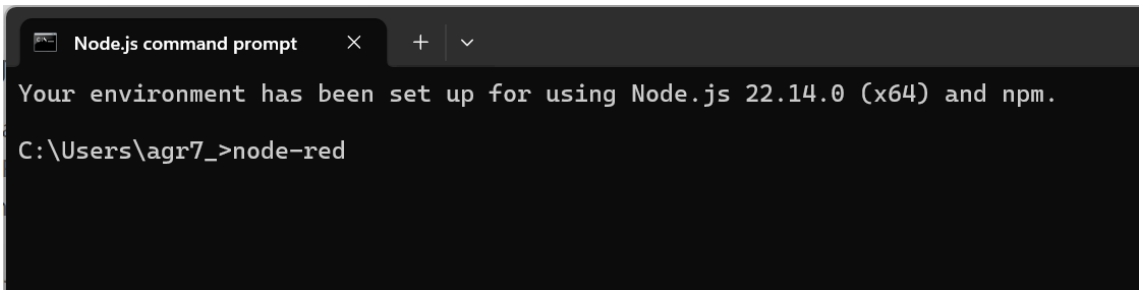
- Abrir una terminal Command Prompt.



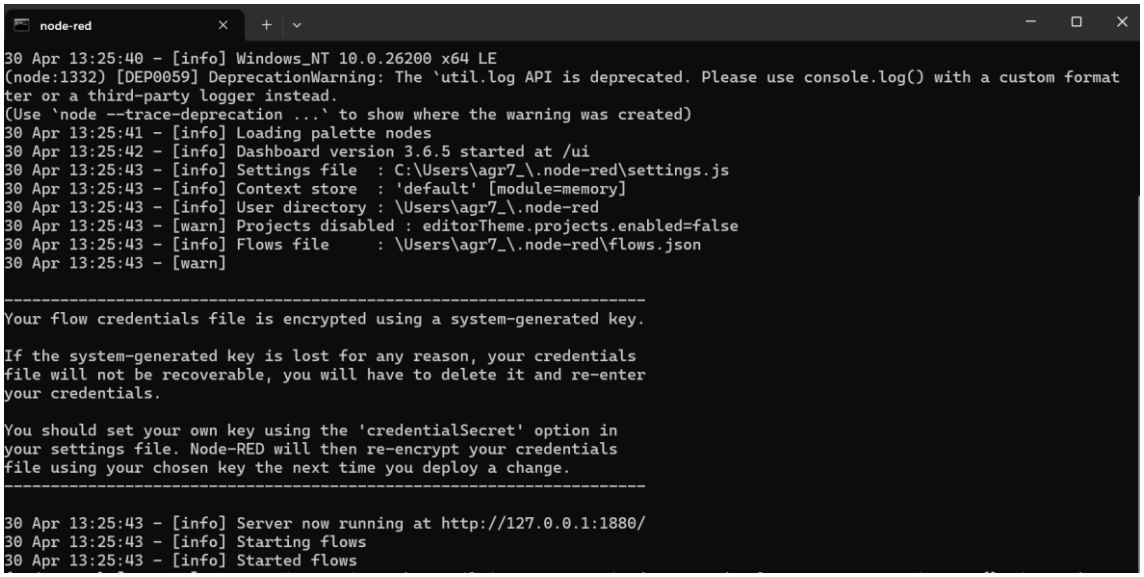
- Instalar NODE-RED empleando el siguiente comando en la terminal:



- Ejecutar NODE-RED empleando el siguiente comando en la terminal:



- Una vez ejecutado esperar a que inicie el programa y abrir en un navegador en la dirección indicada, en este caso el usuario localhost, 127.0.0.1:1800:

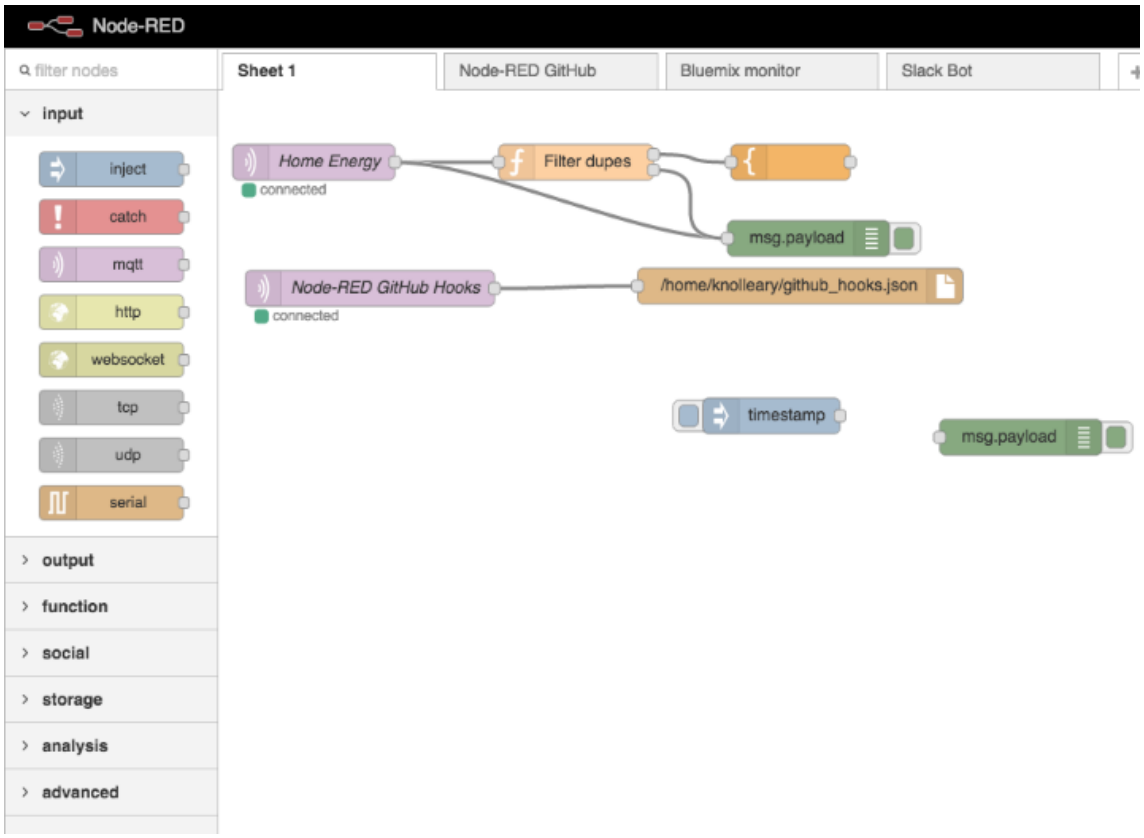


6.2 Principios básicos Node-Red

El programa Node-Red está compuesto de una serie de elementos que se describen a continuación:

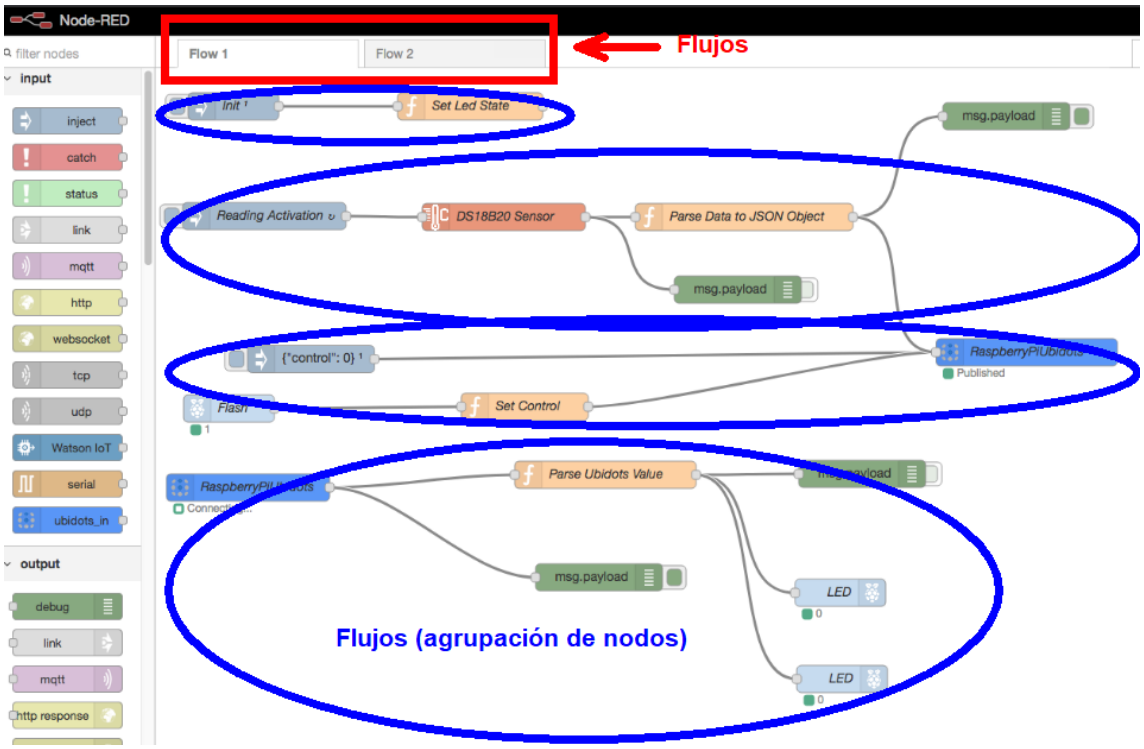
- Nodo:

En Node-Red la estructura mínima está compuesta por “nodos”. Estos nodos son como “cajitas” que realizan una función concreta (mandar un email, recibir un mensaje mqtt, leer una entrada del PLC, enviar datos por puerto serie, etc.) y que se arrastran a una interfaz gráfica.



- Flujo (Flow):

Todos estos nodos se agrupan en lo que se conoce como flujo o Flow. Este flujo está representado como una pestaña en la parte superior de la pantalla. Cuando hablamos de flujo también se refiere a de manera informal a un conjunto de nodos conectados, de tal manera que un flujo (pestaña) puede contener múltiples flujos (conjunto de nodos conectados).



- Mensajes:

En Node-Red la información pasa a través de los nodos en lo que se conoce como mensaje. Estos mensajes son objetos en JavaScript que pueden tener múltiples propiedades. Estos mensajes son llamados por el editor como “msg”.

Los mensajes contienen una carga útil (lo que se conoce como “payload”). Es una propiedad por defecto con la que la mayoría de nodos pueden trabajar. Este payload puede contener cuanta información necesitemos (booleanos, arrays, string, números, etc.) todo junto en el mismo mensaje.

Para hacernos una idea la estructura de los mensajes en Node-Red, estos pueden contener desde un simple 0, 1 de una variable binaria de un pulsador, hasta contener en un mismo mensaje todas las variables del PLC que quiero leer, ya sean booleanos, enteros, reales, todos ellos juntos en un mismo mensaje.

Para entender este concepto de forma fácil imaginemos un mensaje en el que hay mezclados los siguientes datos, es decir, el payload del mensaje tiene lo siguiente:

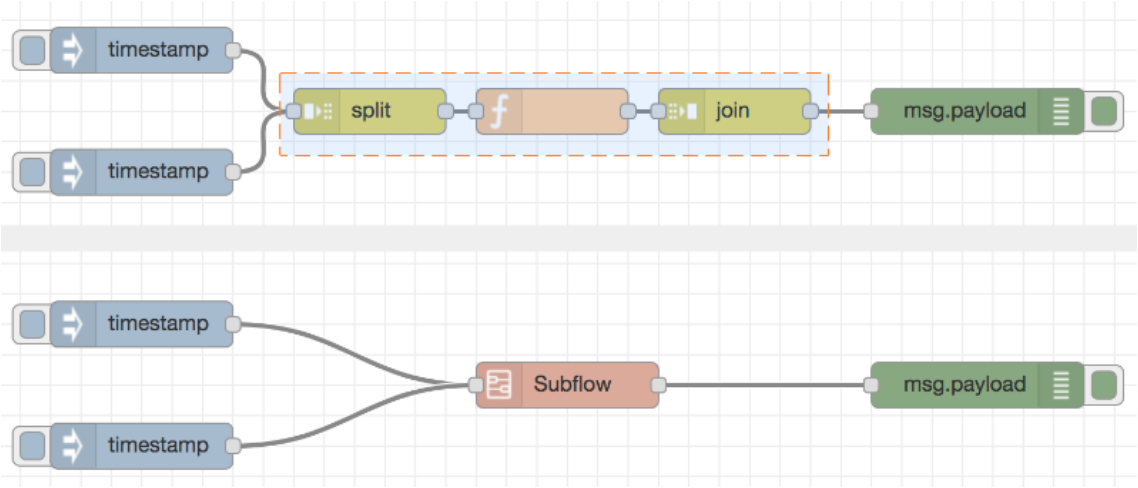
Si en un nodo posterior queremos coger solo parte de la información que contiene el mensaje debemos nombrar de la siguiente manera:

- Si quiero obtener el nombre, deberé acceder a la información como: "msg.payload.FirstName".
- Si quiero obtener el número de teléfono contenido en el elemento 2 dentro del array Phone, deberemos acceder como: "msg.payload.Phone[2].number".

- Subflujo (Subflow):

Un subflujo es una agrupación de nodos que se engloban en uno solo en el espacio de trabajo. Sirven para reducir el espacio de manera visual en un programa complejo con muchos nodos. Viene a ser como un bloque de función en el que esa agrupación de nodos que hace una tarea concreta además puede emplearse en diversas ocasiones sin tener que insertar todos los nodos.

Para crear un subflujo en la pestaña de opciones de menú, hacer clic en "Subflow" y "Create Subflow".

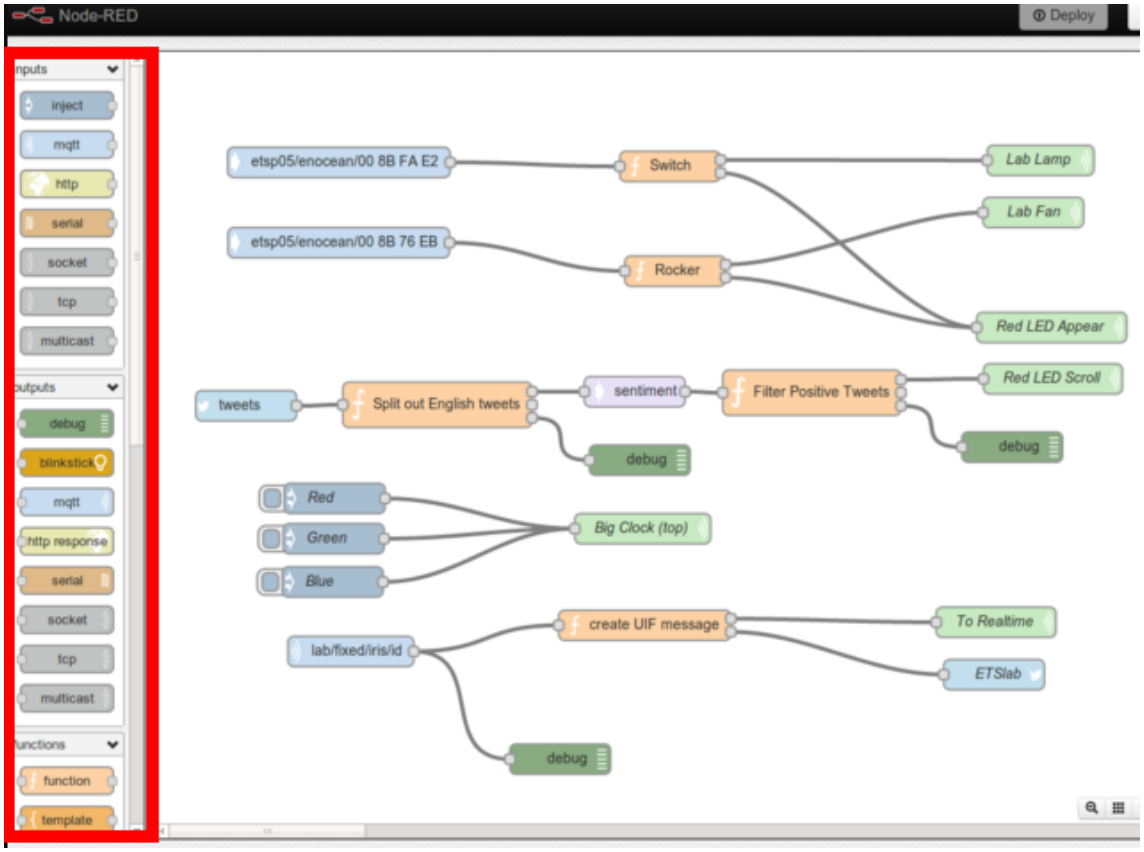


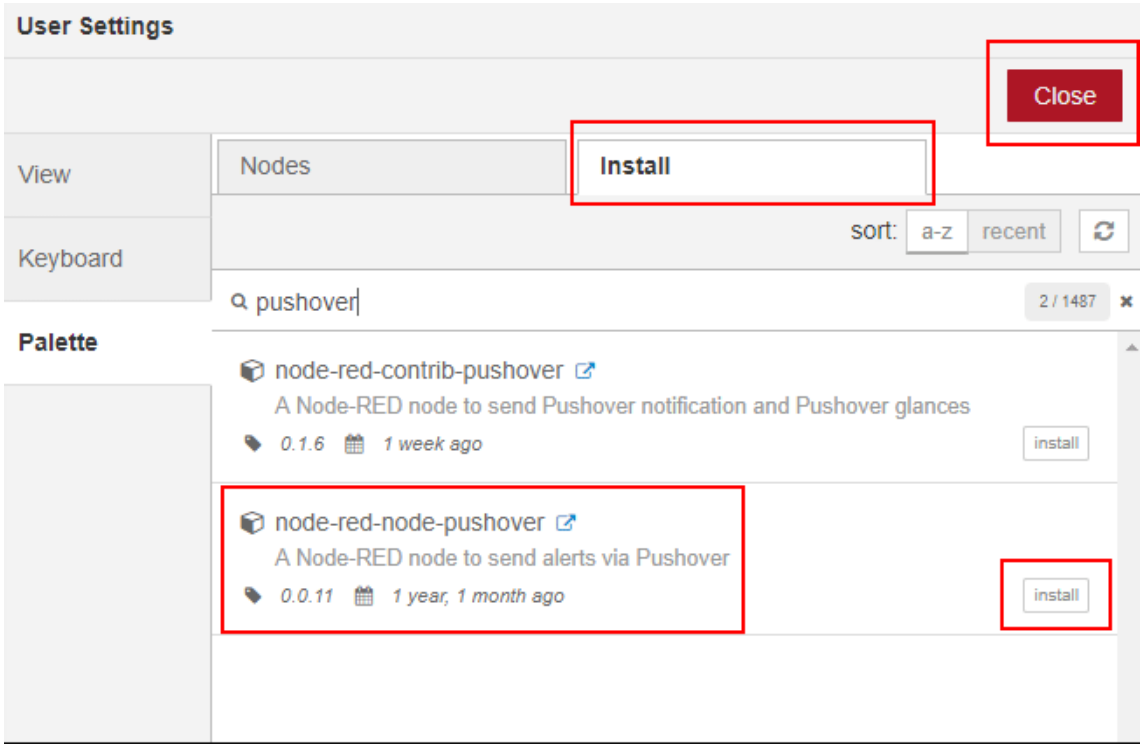
- Cables (Wires):

Los cables se emplean para conectar los nodos y a través de ellos pasan la información en forma de mensajes.

- Pallete:

Es la barra de herramientas donde se encuentran los nodos que tenemos disponibles instalados en nuestro sistema. Adicionalmente se pueden instalar nodos a través de una biblioteca.


Node-RED
Deploy
inputs
inject
mqtt
http
serial
socket
tcp
multicast
outputs
debug
blinkstick
mqtt
http response
serial
socket
tcp
multicast
functions
function
template
etsp05/enocean/00 8B FA E2
etsp05/enocean/00 8B 76 EB
Switch
Rocker
Lab Lamp
Lab Fan
Red LED Appear
tweets
Split out English tweets
sentiment
Filter Positive Tweets
Red LED Scroll
debug
debug
Red
Green
Blue
Big Clock (top)
lab/fixed/iris/id
create UIF message
To Realtime
ETSlab
debug


User Settings
Close
View
Keyboard
Palette
Nodes
Install
sort:
a-z
recent
pushover
2 / 1487
node-red-contrib-pushover
A Node-RED node to send Pushover notification and Pushover glances
0.1.6
1 week ago
install
node-red-node-pushover
A Node-RED node to send alerts via Pushover
0.0.11
1 year, 1 month ago
install

6.3 Implementación

Se ha desarrollado una aplicación IOT para almacenar los datos del robot en una base de datos y posteriormente poder acceder a ellos mediante un router 5G

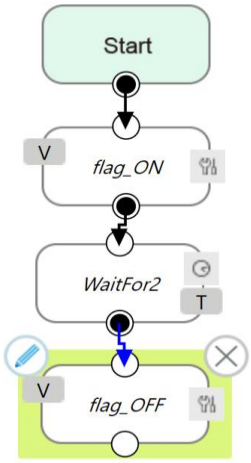
Para ello, se ha empleado el protocolo de comunicaciones Modbus, que es el protocolo de comunicaciones que acepta el robot colaborativo OMRON TM5-S.

Los nodos empleados para hacer comunicaciones Modbus han sido: *node-red-contrib-modbus*.

El robot dispone de una serie de registros donde almacena todos los datos relacionados con la cinemática del robot. En su manual establece las direcciones a las que hay que acceder para ver cada variable.

Robot Coordinate	FC	Address$_{10}$	Type	R/W	Note	S series	HW 3.2
X (Cartesian coordinate w.r.t. current Base without tool)	04	7001~7002	Float	R	mm		
Y (Cartesian coordinate w.r.t. current Base without tool)	04	7003~7004	Float	R	mm		
Z (Cartesian coordinate w.r.t. current Base without tool)	04	7005~7006	Float	R	mm		
Rx (Cartesian coordinate w.r.t. current Base without tool)	04	7007~7008	Float	R	degree		
Ry (Cartesian coordinate w.r.t. current Base without tool)	04	7009~7010	Float	R	degree		
Rz (Cartesian coordinate w.r.t. current Base without tool)	04	7011~7012	Float	R	degree		
Joint 1	04	7013~7014	Float	R	degree		
Joint 2	04	7015~7016	Float	R	degree		
Joint 3	04	7017~7018	Float	R	degree		
Joint 4	04	7019~7020	Float	R	degree		
Joint 5	04	7021~7022	Float	R	degree		
Joint 6	04	7023~7024	Float	R	degree		
X (Cartesian coordinate w.r.t. current Base with tool)	04	7025~7026	Float	R	mm		
Y(Cartesian coordinate w.r.t. current Base with tool)	04	7027~7028	Float	R	mm		
Z (Cartesian coordinate w.r.t. current Base with tool)	04	7029~7030	Float	R	mm		
Rx (Cartesian coordinate w.r.t. current Base with tool)	04	7031~7032	Float	R	degree		
Ry (Cartesian coordinate w.r.t. current Base with tool)	04	7033~7034	Float	R	degree		
Rz (Cartesian coordinate w.r.t. current Base with tool)	04	7035~7036	Float	R	degree		

En el flujo de programa del cobot se ha implementado un "subflow" para que cada vez que llegue a un punto el robot se haga una lectura de los puntos que se quieren controlar.



A continuación, se exponen las características de las variables que se van a extraer:

Las características de cada dato son:

- Columna "**id**"(int). Es un identificador único en la base de datos para cada uno de los datos introducidos:

- Columna "**timestamp**"(timestamp). Es un identificador de tiempo en el que se introduce cada fila/dato en la tabla:

- Columna "**coordinates**"(double precision[]). Es un array de 6 datos en los que se envían las coordenadas en mm del robot (x, y, z) y los ángulos de rotación de los ejes (Rx, Ry, Rz) en grados. De esta manera los datos corresponden:

[x, y, z, Rx, Ry, Rz]

- Columna "**joint_temperature**" (double precision[]). Es un array de 6 datos en los que se envían las temperaturas en ºC de los 6 ejes o articulaciones del robot. Las articulaciones (joints) se empiezan a numerar por el J1, siendo esta la base del robot hasta J6 que es la articulación de la punta del robot. De esta manera los datos son:

[Temp_J1, Temp_J2, Temp_J3, Temp_J4, Temp_J5, Temp_J6]

- Columna "**joint_torque**" (double precision[]). Es un array de 6 datos en los que se envía el par motor en mN.m de cada una de los ejes o articulaciones (joints). De esta manera los datos son:

[Torque_J1, Torque_J2, Torque_J3, Torque_J4, Torque_J5, Torque_J6]

- Columna "**joint_current**" (double precision[]). Es un array de 6 datos en los que se envía la corriente eléctrica de cada eje o articulación en amperios A. Los datos que se envían corresponden al valor máximo capturado durante el movimiento de un punto a otro. No se mandan cuando el robot está parado porque el consumo es cercano a 0. De esta manera los datos son:

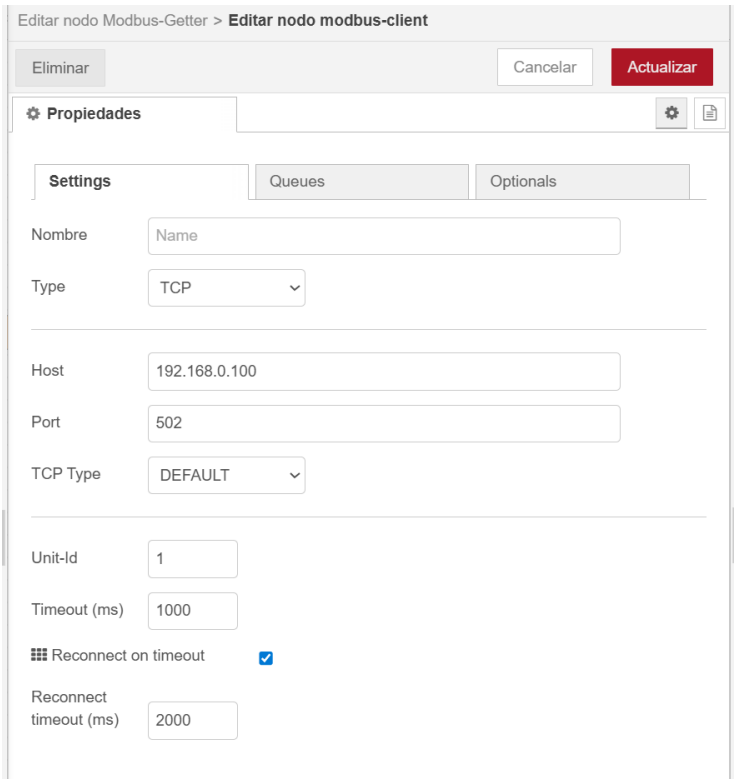
[Current_J1, Current_J2, Current_J3, Current_J4, Current_J5, Current_J6]

- Columna "**joint_speed**" (double precision[]). Es un array de 6 datos en los que se envía la velocidad de cada eje o articulación en degree/s. Los datos que se envían corresponden al valor máximo capturado durante el movimiento de un punto a otro. No se mandan cuando el robot está parado porque velocidad sería 0. De esta manera los datos son:

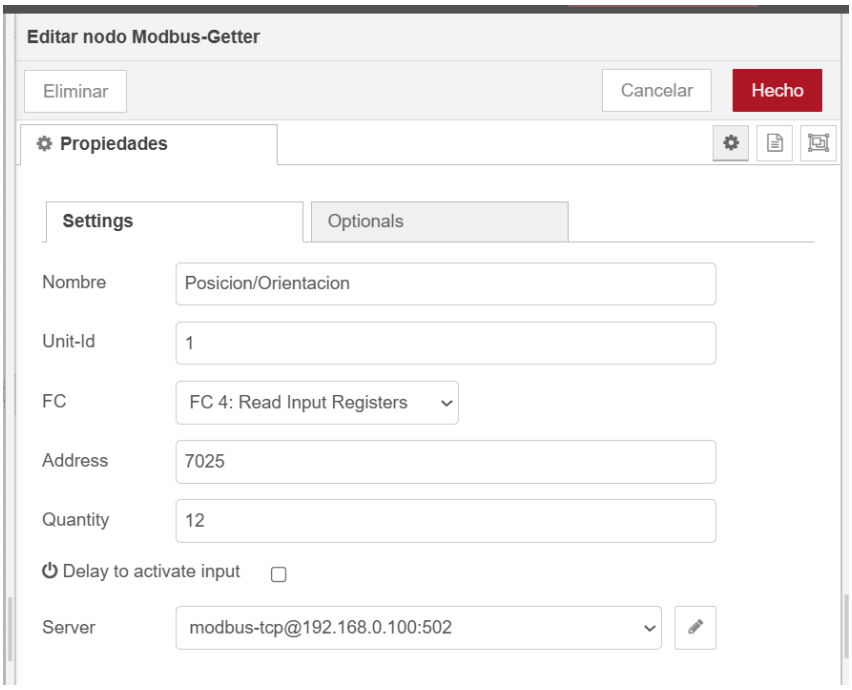
[Speed_J1, Speed_J2, Speed_J3, Speed_J4, Speed_J5, Speed_J6]

- Columna "**cycle_time**" (interval). Es el tiempo que dura el ciclo de ejecución del programa. En cada punto se manda el tiempo por el que va el ciclo. Cuando acaba un ciclo el tiempo se reinicia

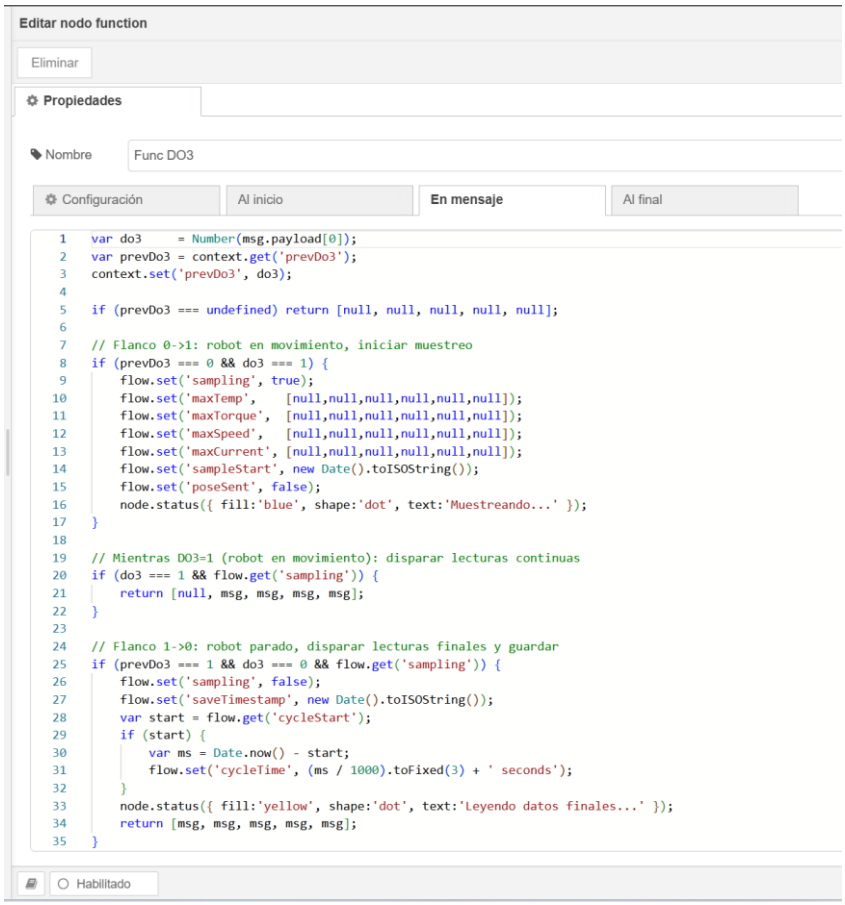
Para la parte de Node-Red se ha empleado el nodo "*Modbus-Getter*". Mediante este nodo se extrae información del robot. En primer lugar hay que definir un servidor, que este caso es el propio robot:



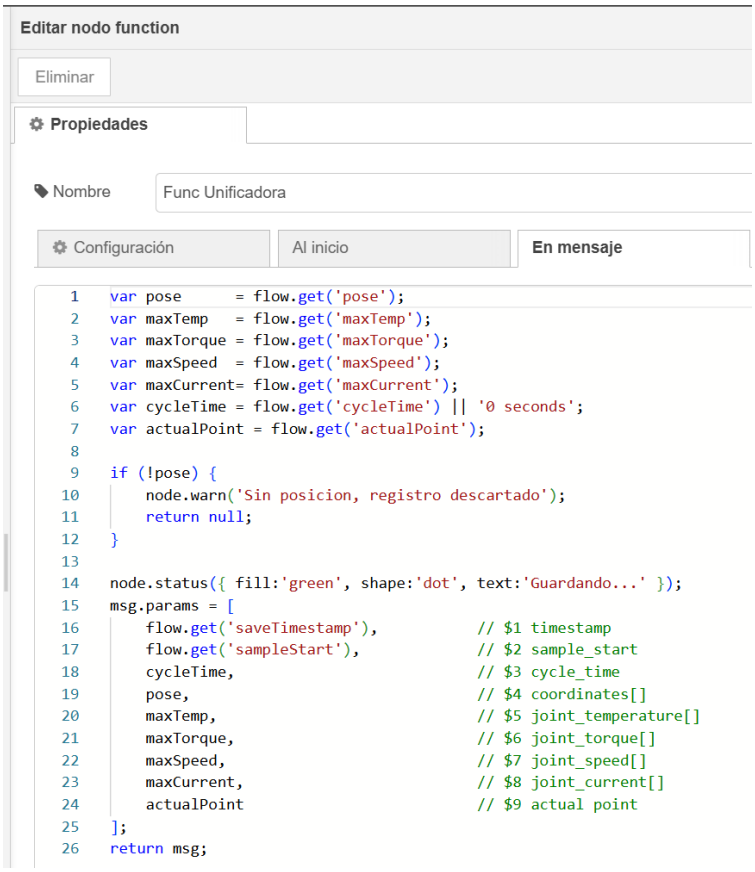
Posteriormente hay que indicarle el registro Modbus que se desea leer, la dirección y la cantidad de registros a partir del primero indicado



Siguiendo este principio, lo que se ha hecho es generar un nodo de este tipo por cada una de las variables que se van a almacenar. Mediante un nodo de función se juntan todas ellas para que vayan por un solo mensaje.

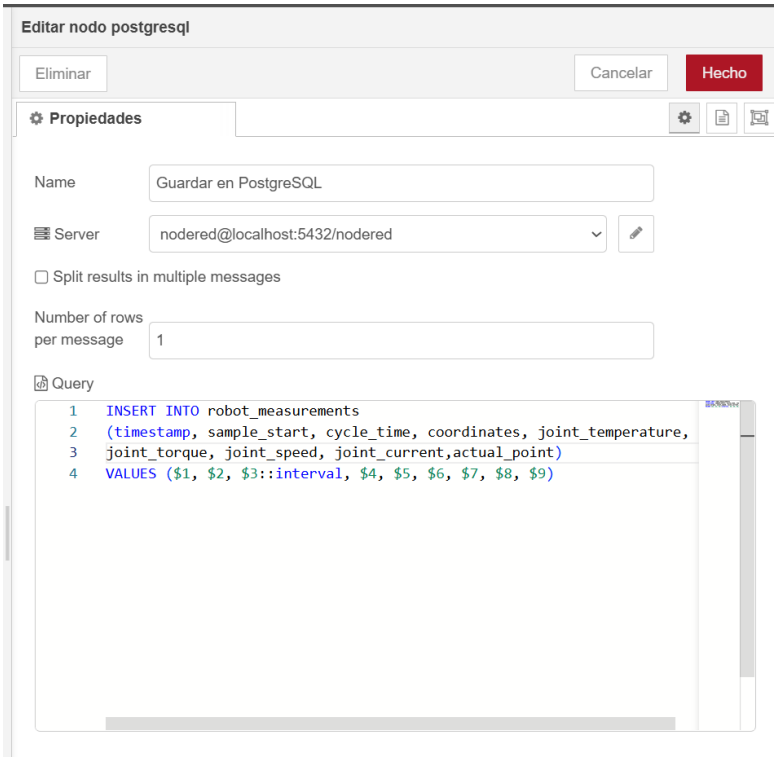



```
var do3     = Number(msg.payload[0]);
var prevDo3 = context.get('prevDo3');
context.set('prevDo3', do3);

if (prevDo3 === undefined) return [null, null, null, null, null];

// Flanco 0->1: robot en movimiento, iniciar muestreo
if (prevDo3 === 0 && do3 === 1) {
    flow.set('sampling', true);
    flow.set('maxTemp',    [null,null,null,null,null,null]);
    flow.set('maxTorque',  [null,null,null,null,null,null]);
    flow.set('maxSpeed',   [null,null,null,null,null,null]);
    flow.set('maxCurrent', [null,null,null,null,null,null]);
    flow.set('sampleStart', new Date().toISOString());
    flow.set('poseSent', false);
    node.status({ fill:'blue', shape:'dot', text:'Muestreando...' });
}

// Mientras DO3=1 (robot en movimiento): disparar lecturas continuas
if (do3 === 1 && flow.get('sampling')) {
    return [null, msg, msg, msg, msg];
}

// Flanco 1->0: robot parado, disparar lecturas finales y guardar
if (prevDo3 === 1 && do3 === 0 && flow.get('sampling')) {
    flow.set('sampling', false);
    flow.set('saveTimestamp', new Date().toISOString());
    var start = flow.get('cycleStart');
    if (start) {
        var ms = Date.now() - start;
        flow.set('cycleTime', (ms / 1000).toFixed(3) + ' seconds');
    }
    node.status({ fill:'yellow', shape:'dot', text:'Leyendo datos finales...' });
    return [msg, msg, msg, msg, msg];
}
```

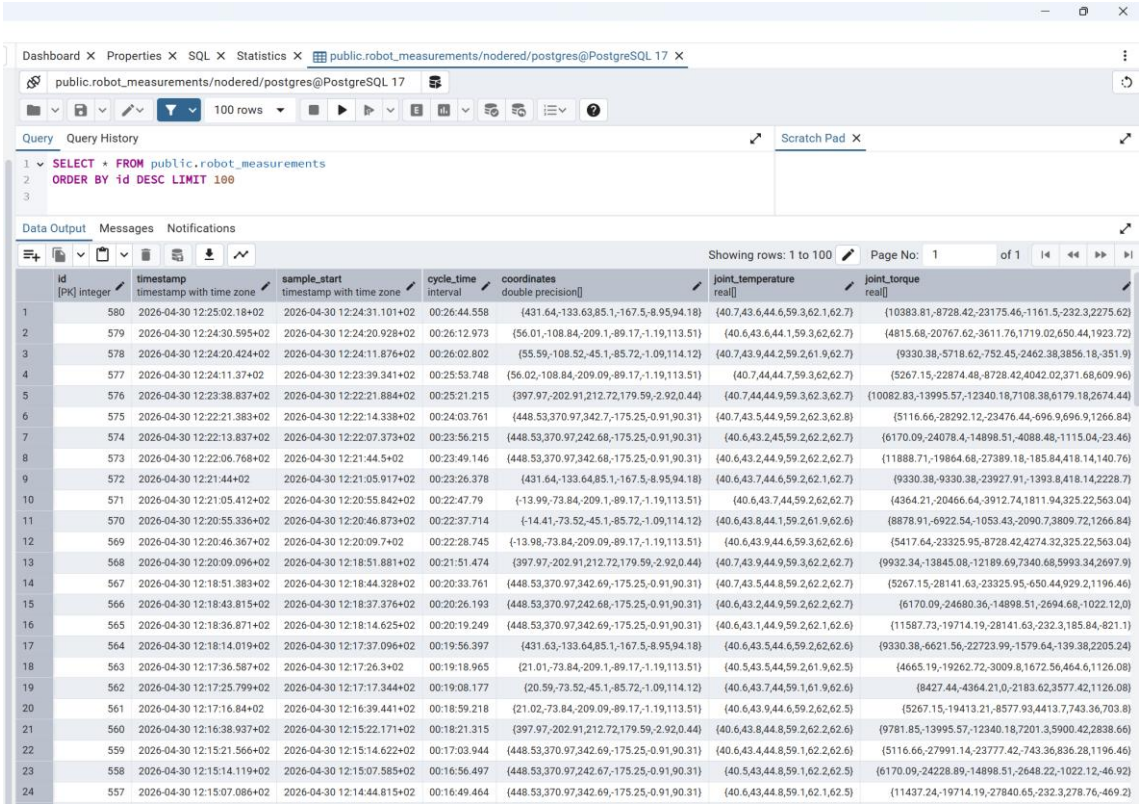



```
var pose      = flow.get('pose');
var maxTemp   = flow.get('maxTemp');
var maxTorque = flow.get('maxTorque');
var maxSpeed  = flow.get('maxSpeed');
var maxCurrent= flow.get('maxCurrent');
var cycleTime = flow.get('cycleTime') || '0 seconds';
var actualPoint = flow.get('actualPoint');

if (!pose) {
    node.warn('Sin posicion, registro descartado');
    return null;
}

node.status({ fill:'green', shape:'dot', text:'Guardando...' });
msg.params = [
    flow.get('saveTimestamp'),          // $1 timestamp
    flow.get('sampleStart'),            // $2 sample_start
    cycleTime,                          // $3 cycle_time
    pose,                               // $4 coordinates[]
    maxTemp,                            // $5 joint_temperature[]
    maxTorque,                          // $6 joint_torque[]
    maxSpeed,                           // $7 joint_speed[]
    maxCurrent,                         // $8 joint_current[]
    actualPoint                         // $9 actual point
];
return msg;
```

El último paso es insertar los datos en una base de datos PostgreSQL. Para ello, se ha empleado el paquete "*node-red-contrib-postgresql*".

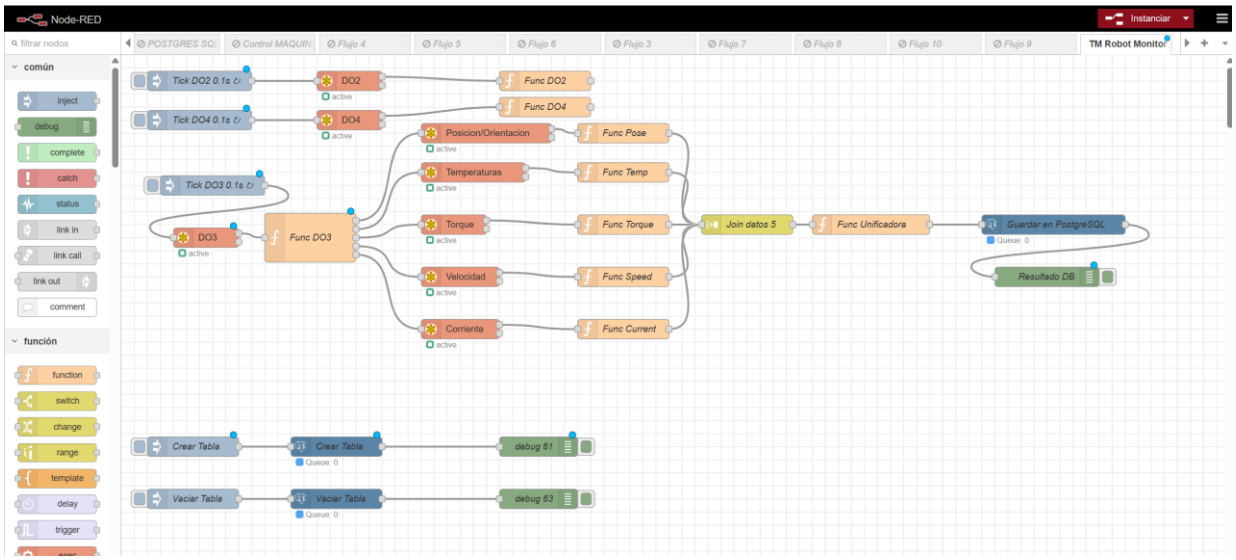


Los datos se pueden consultar en la base de datos, mediante el programa "pgadmin":



	id [PK] integer	timestamp timestamp with time zone	sample_start timestamp with time zone	cycle_time interval	coordinates double precision[]	joint_temperature real[]	joint_torque real[]
1	580	2026-04-30 12:25:02.18+02	2026-04-30 12:24:31.101+02	00:26:44.558	{431.64,-133.63,85.1,-167.5,-8.95,94.18}	{40.7,43.6,44.6,59.3,62.1,62.7}	{10383.81,-8728.42,-23175.46,-1161.5,-232.3,2275.62}
2	579	2026-04-30 12:24:30.595+02	2026-04-30 12:24:20.928+02	00:26:12.973	{56.01,-108.84,-209.1,-89.17,-1.19,113.51}	{40.6,43.6,44.1,59.3,62,62.7}	{4815.68,-20767.62,-3611.76,1719.02,650.44,1923.72}
3	578	2026-04-30 12:24:20.424+02	2026-04-30 12:24:11.876+02	00:26:02.802	{55.59,-108.52,-45.1,-85.72,-1.09,114.12}	{40.7,43.9,44.2,59.2,61.9,62.7}	{9330.38,-5718.62,-752.45,-2462.38,3856.18,-351.9}
4	577	2026-04-30 12:24:11.37+02	2026-04-30 12:23:39.341+02	00:25:53.748	{56.02,-108.84,-209.09,-89.17,-1.19,113.51}	{40.7,44,44.7,59.3,62,62.7}	{5267.15,-22874.48,-8728.42,4042.02,371.68,609.96}
5	576	2026-04-30 12:23:38.837+02	2026-04-30 12:22:21.884+02	00:25:21.215	{397.97,-202.91,212.72,179.59,-2.92,0.44}	{40.7,44,44.9,59.3,62.3,62.7}	{10082.83,-13995.57,-12340.18,7108.38,6179.18,2674.44}
6	575	2026-04-30 12:22:21.383+02	2026-04-30 12:22:14.338+02	00:24:03.761	{448.53,370.97,342.7,-175.25,-0.91,90.31}	{40.7,43.5,44.9,59.2,62.3,62.8}	{5116.66,-28292.12,-23476.44,-696.9,696.9,1266.84}
7	574	2026-04-30 12:22:13.837+02	2026-04-30 12:22:07.373+02	00:23:56.215	{448.53,370.97,242.68,-175.25,-0.91,90.31}	{40.6,43.2,45,59.2,62.2,62.7}	{6170.09,-24078.4,-14898.51,-4088.48,-1115.04,-23.46}
8	573	2026-04-30 12:22:06.768+02	2026-04-30 12:21:44.5+02	00:23:49.146	{448.53,370.97,342.68,-175.25,-0.91,90.31}	{40.6,43.2,44.9,59.2,62.2,62.7}	{11888.71,-19864.68,-27389.18,-185.84,418.14,140.76}
9	572	2026-04-30 12:21:44+02	2026-04-30 12:21:05.917+02	00:23:26.378	{431.64,-133.64,85.1,-167.5,-8.95,94.18}	{40.6,43.7,44.6,59.2,62.1,62.7}	{9330.38,-9330.38,-23927.91,-1393.8,418.14,2228.7}
10	571	2026-04-30 12:21:05.412+02	2026-04-30 12:20:55.842+02	00:22:47.79	{-13.99,-73.84,-209.1,-89.17,-1.19,113.51}	{40.6,43.7,44,59.2,62,62.7}	{4364.21,-20466.64,-3912.74,1811.94,325.22,563.04}
11	570	2026-04-30 12:20:55.336+02	2026-04-30 12:20:46.873+02	00:22:37.714	{-14.41,-73.52,-45.1,-85.72,-1.09,114.12}	{40.6,43.8,44.1,59.2,61.9,62.6}	{8878.91,-6922.54,-1053.43,-2090.7,3809.72,1266.84}
12	569	2026-04-30 12:20:46.367+02	2026-04-30 12:20:09.7+02	00:22:28.745	{-13.98,-73.84,-209.09,-89.17,-1.19,113.51}	{40.6,43.9,44.6,59.3,62,62.6}	{5417.64,-23325.95,-8728.42,4274.32,325.22,563.04}
13	568	2026-04-30 12:20:09.096+02	2026-04-30 12:18:51.881+02	00:21:51.474	{397.97,-202.91,212.72,179.59,-2.92,0.44}	{40.7,43.9,44.9,59.3,62.2,62.7}	{9932.34,-13845.08,-12189.69,7340.68,5993.34,2697.9}
14	567	2026-04-30 12:18:51.383+02	2026-04-30 12:18:44.328+02	00:20:33.761	{448.53,370.97,342.69,-175.25,-0.91,90.31}	{40.7,43.5,44.8,59.2,62.2,62.7}	{5267.15,-28141.63,-23325.95,-650.44,929.2,1196.46}
15	566	2026-04-30 12:18:43.815+02	2026-04-30 12:18:37.376+02	00:20:26.193	{448.53,370.97,242.68,-175.25,-0.91,90.31}	{40.6,43.2,44.9,59.2,62.2,62.7}	{6170.09,-24680.36,-14898.51,-2694.68,-1022.12,0}
16	565	2026-04-30 12:18:36.871+02	2026-04-30 12:18:14.625+02	00:20:19.249	{448.53,370.97,342.69,-175.25,-0.91,90.31}	{40.6,43.1,44.9,59.2,62.1,62.6}	{11587.73,-19714.19,-28141.63,-232.3,185.84,-821.1}
17	564	2026-04-30 12:18:14.019+02	2026-04-30 12:17:37.096+02	00:19:56.397	{431.63,-133.64,85.1,-167.5,-8.95,94.18}	{40.6,43.5,44.6,59.2,62,62.6}	{9330.38,-6621.56,-22723.99,-1579.64,-139.38,2205.24}
18	563	2026-04-30 12:17:36.587+02	2026-04-30 12:17:26.3+02	00:19:18.965	{21.01,-73.84,-209.1,-89.17,-1.19,113.51}	{40.5,43.5,44,59.2,61.9,62.5}	{4665.19,-19262.72,-3009.8,1672.56,464.6,1126.08}
19	562	2026-04-30 12:17:25.799+02	2026-04-30 12:17:17.344+02	00:19:08.177	{20.59,-73.52,-45.1,-85.72,-1.09,114.12}	{40.6,43.7,44,59.1,61.9,62.6}	{8427.44,-4364.21,0,-2183.62,3577.42,1126.08}
20	561	2026-04-30 12:17:16.84+02	2026-04-30 12:16:39.441+02	00:18:59.218	{21.02,-73.84,-209.09,-89.17,-1.19,113.51}	{40.6,43.9,44.6,59.2,62,62.5}	{5267.15,-19413.21,-8577.93,4413.7,743.36,703.8}
21	560	2026-04-30 12:16:38.937+02	2026-04-30 12:15:22.171+02	00:18:21.315	{397.97,-202.91,212.72,179.59,-2.92,0.44}	{40.6,43.8,44.8,59.2,62.2,62.6}	{9781.85,-13995.57,-12340.18,7201.3,5900.42,2838.66}
22	559	2026-04-30 12:15:21.566+02	2026-04-30 12:15:14.622+02	00:17:03.944	{448.53,370.97,342.69,-175.25,-0.91,90.31}	{40.6,43.4,44.8,59.1,62.2,62.6}	{5116.66,-27991.14,-23777.42,-743.36,836.28,1196.46}
23	558	2026-04-30 12:15:14.119+02	2026-04-30 12:15:07.585+02	00:16:56.497	{448.53,370.97,242.67,-175.25,-0.91,90.31}	{40.5,43,44.8,59.1,62.2,62.5}	{6170.09,-24228.89,-14898.51,-2648.22,-1022.12,-46.92}
24	557	2026-04-30 12:15:07.086+02	2026-04-30 12:14:44.815+02	00:16:49.464	{448.53,370.97,342.69,-175.25,-0.91,90.31}	{40.6,43,44.8,59.1,62.1,62.5}	{11437.24,-19714.19,-27840.65,-232.3,278.76,-469.2}

El flujo completo de Node-Red para este proyecto queda de la siguiente manera:



7 Conclusiones

Con este tutorial se ha desarrollado la aplicación IOT que extrae datos del robot colaborativo OMRON TM5-S mediante modbus y los almacena en una base de datos PostgreSQL para su consulta desde cualquier punto.