UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO

Torrecilla Matencio, Marc

ESTUDIO DE LAS ETAPAS DE AUTOMATIZACIÓN DE UN PROCESO INDUSTRIAL

ANEXO III

Director: Delgado Prieto, Miguel

Co-director: Fernández Sobrino, Ángel

Convocatoria: Julio, 2020

SUMARIO

ANEXO III:

SOFTWARE NODE-RED

1. FLUJOS

1.1. FLUJO GENERAL

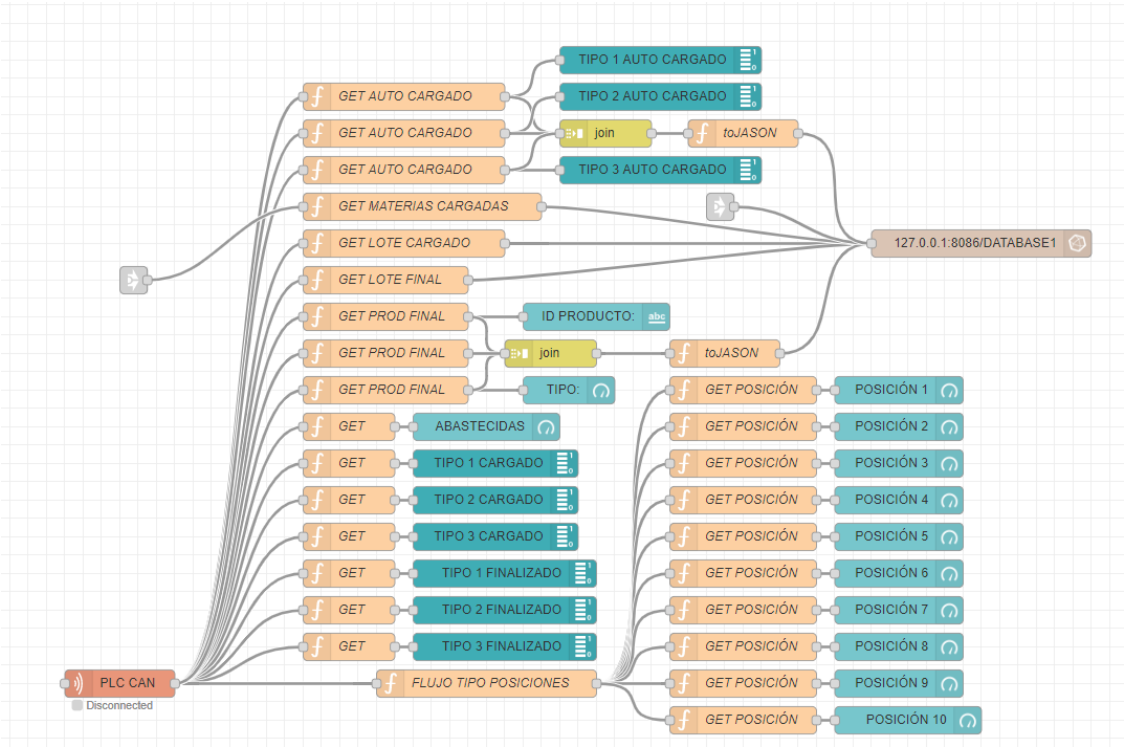

Figura 1. Flujo General [1].

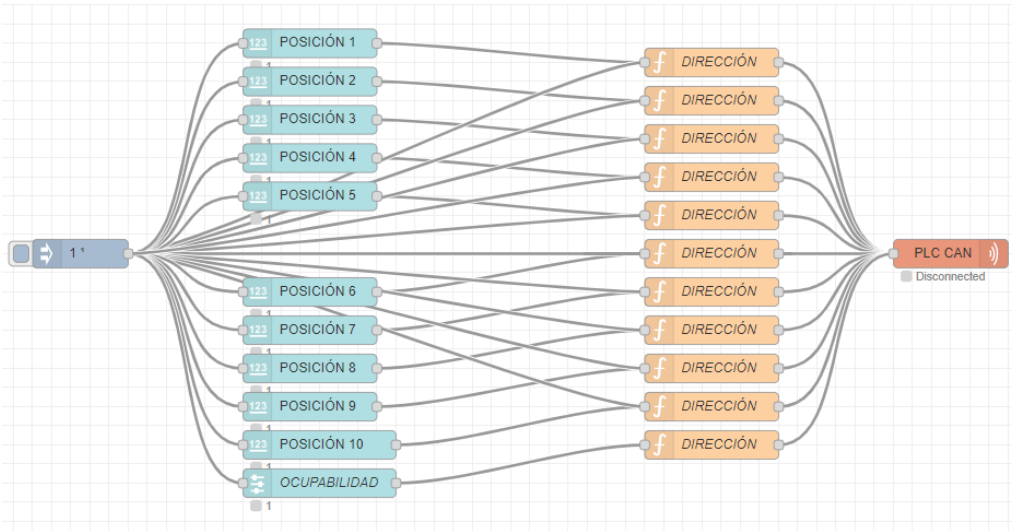

Figura 2. Flujo General [2].

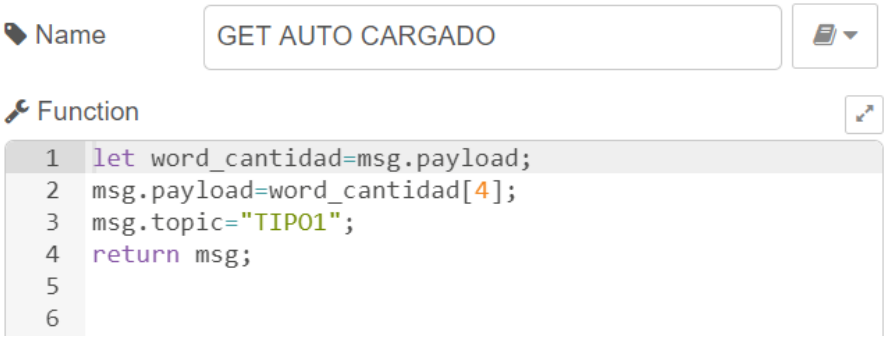

Figura 3. Nodo función GET AUTO CARGADO [1].

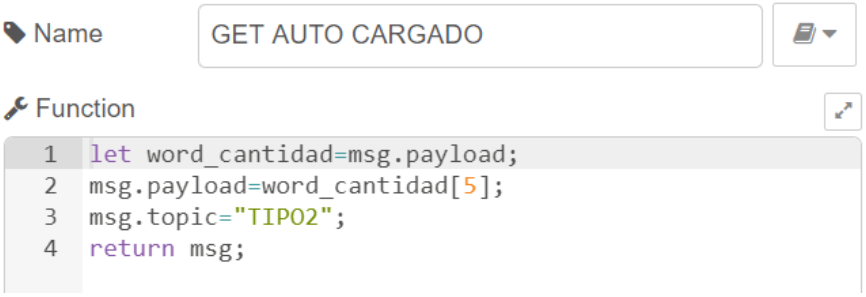

Figura 4. Nodo función GET AUTO CARGADO [2].

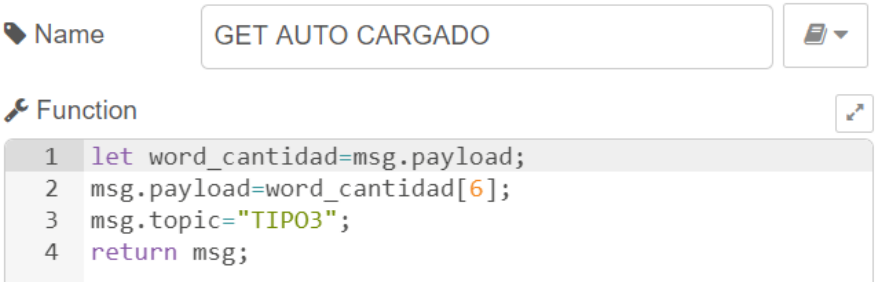

Figura 5. Nodo función GET AUTO CARGADO [3].

Name GET MATERIAS CARGADAS

Function

```
let cantidad= global.get("MATERIAS");
msg.payload=[
    {
        measurement: "MateriasCargadas",
        fields: {
            Materias: cantidad
        }
    }
    ]
return msg;
```

Figura 6. Nodo función GET MATERIAS CARGADAS.

Name GET LOTE CARGADO

Function

```
let lote, estado, tipo1, tipo2, tipo3;

estado=msg.payload[7];
tipo1=msg.payload[8];
tipo2=msg.payload[9];
tipo3=msg.payload[10];

if(estado===1){
    msg.payload=[
        {
        measurement: "LoteCargado",
        fields: {
            Tipo1:tipo1,
            Tipo2:tipo2,
            Tipo3:tipo3
            }
        }
    ];

    return msg;
}
```

Figura 7. Nodo Función GET LOTE CARGADO.

Name GET LOTE FINAL

Function

```
let lote, estado, tipo1, tipo2, tipo3;

estado=msg.payload[11];
tipo1=msg.payload[12];
tipo2=msg.payload[13];
tipo3=msg.payload[14];

if(estado===1){
    msg.payload=[
        {
        measurement: "LoteFinalizado",
        fields: {
            Tipo1:tipo1,
            Tipo2:tipo2,
            Tipo3:tipo3
            }
        }
    ];

    return msg;
}

```

Figura 8. Nodo función GET LOTE FINAL.

Name GET PROD FINAL

Function

```
let word_cantidad=msg.payload;
msg.payload=word_cantidad[0];
msg.topic= "ID";
return msg;
```

Figura 9. Nodo función GET PROD FINAL [1].

Name GET PROD FINAL

Function

```
let word_cantidad=msg.payload;
msg.payload=word_cantidad[1];
msg.topic= "TIPO";
return msg;
```

Figura 10. Nodo función GET PROD FINAL [2].

Figura 11. Nodo función GET PROD FINAL [3].

Figura 12. Nodo función GET (materias actuales abastecidas en la línea)

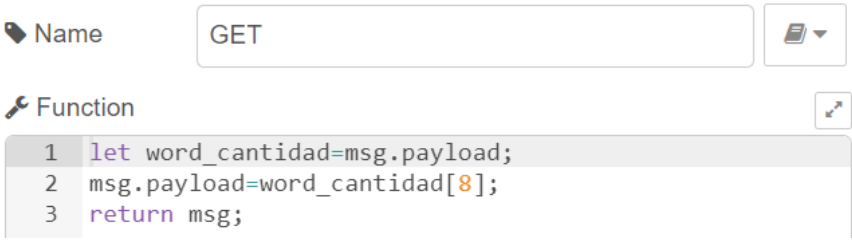

Figura 13. Nodo función GET (tipo cargado 1).

Figura 14. Node función GET (tipo cargado 2)

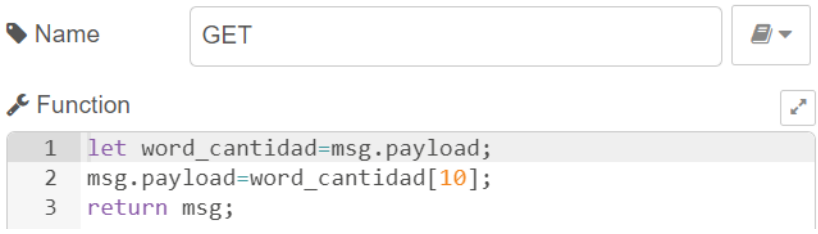

Figura 15. Nodo función GET (tipo 3 cargado).

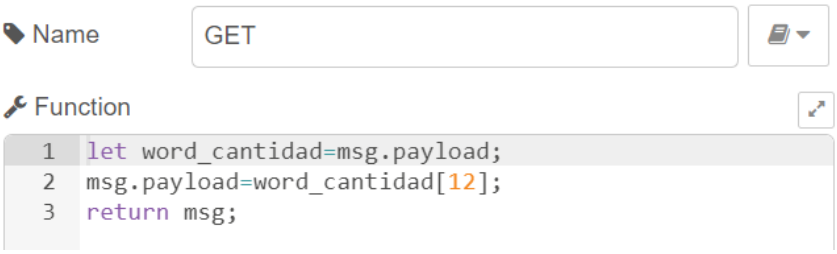

Figura 16. Nodo función GET (tipo 1 finalizado).

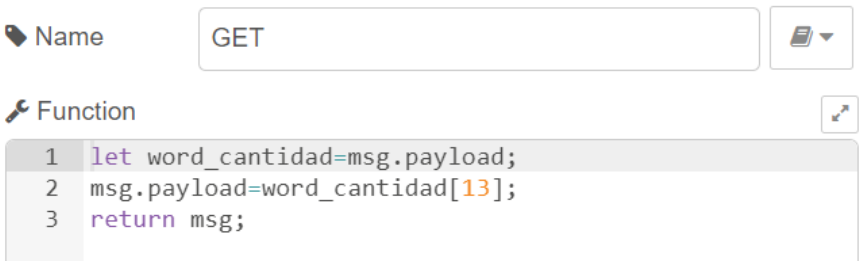

Figura 17. Nodo función GET (tipo 2 finalizado).

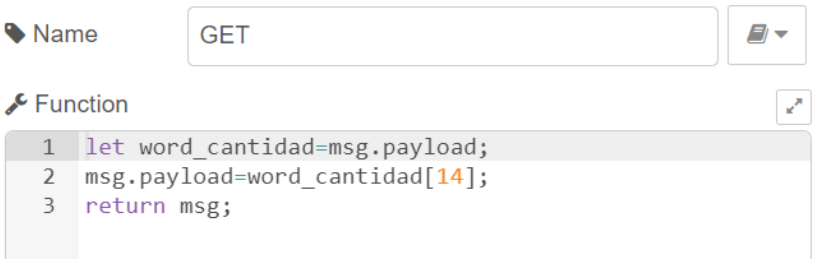

Figura 18. Nodo función GET (tipo 3 finalizado)

Name FLUJO TIPO POSICIONES

Function

```
let flujo= [];
for(let i=0; i<10; i++){
    flujo[i]=msg.payload[i+16];
}
global.set("FLUJO",flujo );
return msg;
```

Figura 19. Nodo función FLUJO TIPO POSICIONES.

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[19];
msg.payload=posicion;
return msg;
```

Figura 20. Nodo función GET POSICION (posición 1)

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[20];
msg.payload=posicion;
return msg;
```

Figura 21. Nodo función GET POSICION (posición 2)

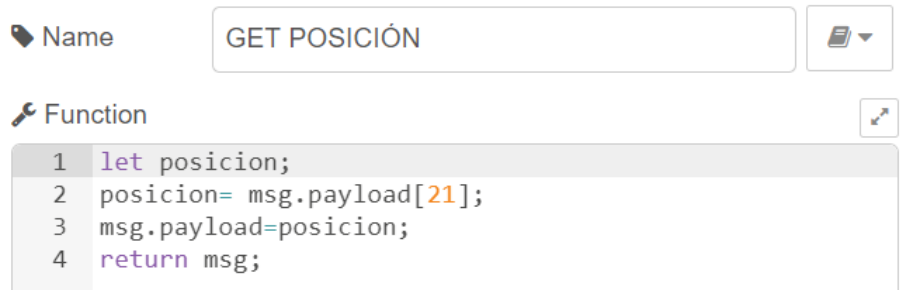

Figura 22. Nodo función GET POSICION (posición 3)

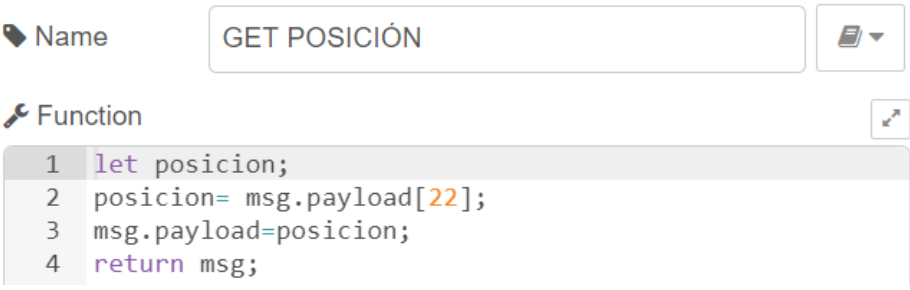

Figura 23. Nodo función GET POSICION (posición 4)

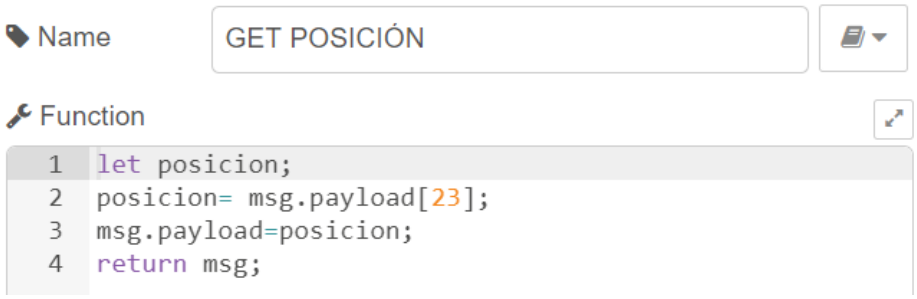

Figura 24. Nodo función GET POSICION (posición 5).

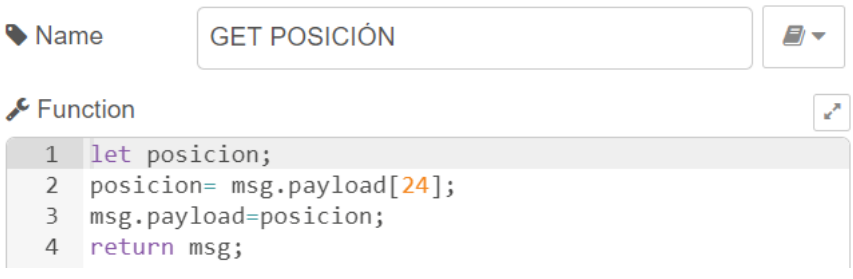

Figura 25. Nodo función GET POSICION (posición 6).

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[25];
msg.payload=posicion;
return msg;
```

Figura 26. Nodo función GET POSICION (posición 7)

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[26];
msg.payload=posicion;
return msg;
```

Figura 27. Nodo función GET POSICION (posición 8).

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[27];
msg.payload=posicion;
return msg;
```

Figura 28. Nodo función GET POSICION (posición 9).

Name GET POSICIÓN

Function

```
let posicion;
posicion= msg.payload[28];
msg.payload=posicion;
return msg;
```

Figura 29. Nodo función GET POSICION (posición 10).

Name toJASON

Function

```
let tipo1=msg.payload["TIPO1"];
let tipo2= msg.payload["TIPO2"];
let tipo3=msg.payload["TIPO3"];

msg.payload=[
    {
        measurement: "AutoCargado",
        fields: {
            TIPO1: tipo1,
            TIPO2: tipo2,
            TIPO3: tipo3
        }
    }]
return msg;
```

Figura 30. Nodo toJASON (auto cargado).

Name toJASON

Function

```
let id=msg.payload["ID"];
let tipo= msg.payload["TIPO"];
let bandeja=msg.payload["BANDEJA"];
let automatico= global.get("CICLO_AUTOMATICO");
let lote= global.get("CICLO_LOTE");

if(automatico || lote){
msg.payload=[
    {
        measurement: "UltimoProducto",
        fields: {
            ID: id,
            TIPO: tipo,
            BANDEJA: bandeja
        }
    }]
return msg;
}
```

Figura 31. Nodo toJASON (producto finalizado).

Figura 32. Nodo función DIRECCIÓN (posición 1)

Figura 33. Nodo función DIRECCIÓN (posición 2)

Figura 34. Nodo función DIRECCIÓN (posición 3).

Figura 35. Nodo función DIRECCIÓN (posición 4).

Figura 36. Nodo función DIRECCIÓN (posición 4).

Figura 37. Nodo función DIRECCIÓN (posición 5).

Figura 38. Nodo función DIRECCIÓN (posición 6).

Figura 39. Nodo función DIRECCIÓN (posición 7).

Figura 40. Nodo función DIRECCIÓN (posición 8).

Figura 41. Nodo función DIRECCIÓN (posición 9).

Figura 42. Nodo función DIRECCIÓN (posición 10).

1.2. FLUJO CICLOS

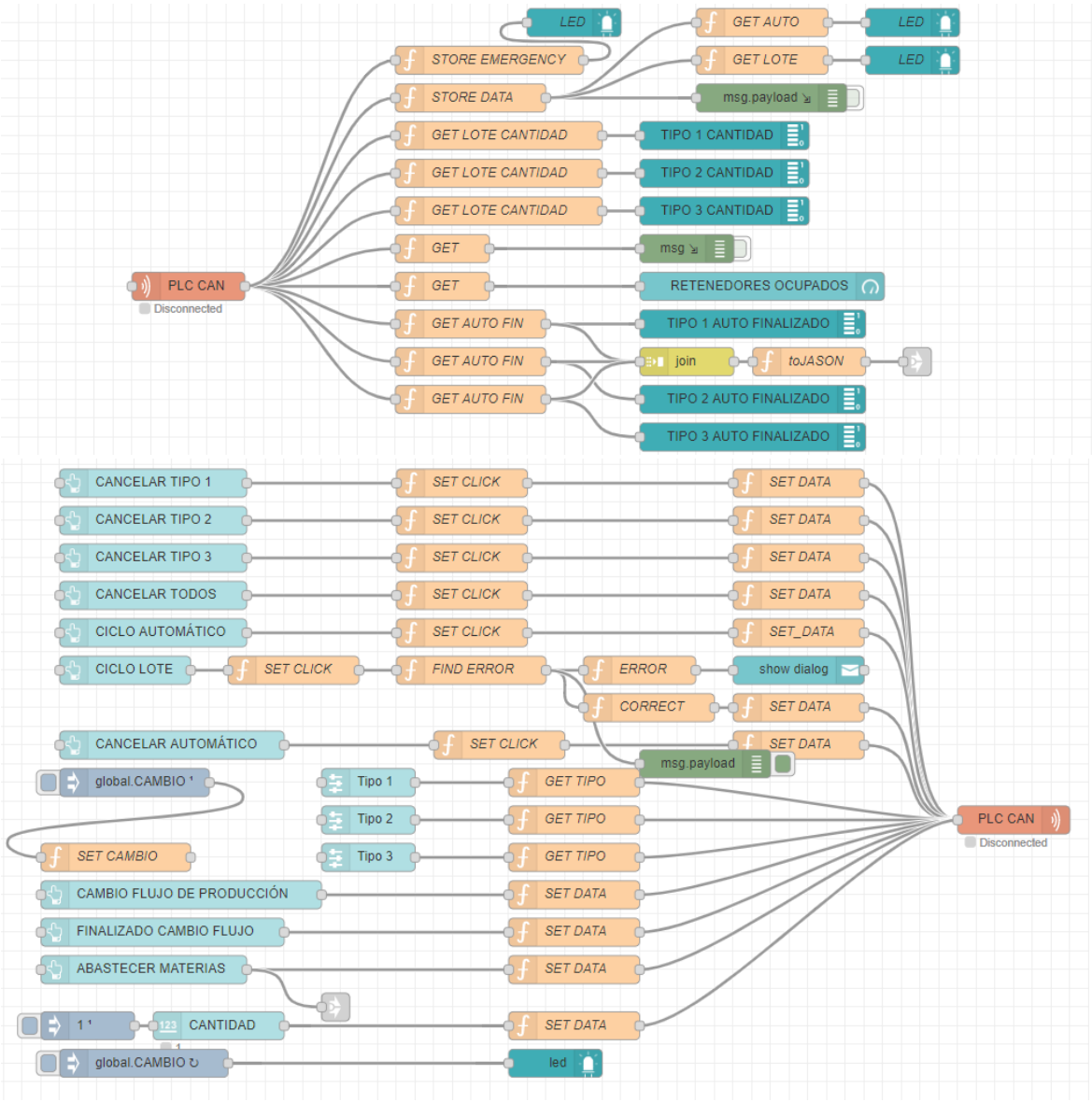

Figura 43. Flujo Ciclos.

Name STORE EMERGENCY

Function

```
let emergency = msg.payload[11];
global.set("EMERGENCY", emergency);
msg.payload=emergency;
return msg;
```

Figura 44. Nodo función STORE EMERGENCY.

Name STORE DATA

Function

```
let recived, word_MW1223=[];
recived=msg.payload[0];

for(let i = 0; i < 16; i++){
    if((recived & Math.pow(2,i)) >0){
        word_MW1223[i]=1;}
    else word_MW1223[i]=0;}

flow.set("MW1223", word_MW1223);
msg.payload = word_MW1223;
return msg;
```

Figura 45. Nodo función STORE DATA.

Name GET LOTE CANTIDAD

Function

```
let lote,word_cantidad=msg.payload;
msg.payload=word_cantidad[4];
return msg;
```

Figura 46. Nodo función GET LOTE CANTIDAD (tipo 1 lote cantidad)

Name GET LOTE CANTIDAD

Function

```
let lote, word_cantidad=msg.payload;
msg.payload=word_cantidad[5];
return msg;
```

Figura 47. Nodo función GET LOTE CANTIDAD (tipo 2 lote cantidad).

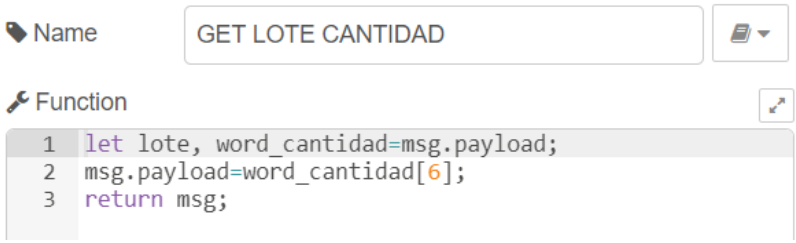

Figura 48. Nodo función GET LOTE CANTIDAD (tipo 3 lote cantidad).

Figura 49. Nodo función GET (guardado de lote cantidad)

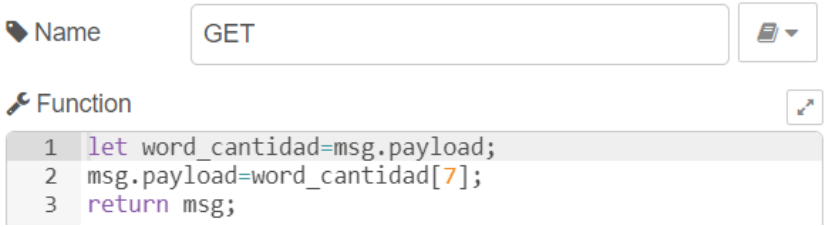

Figura 50. Nodo función GET (retenedores ocupados Profibus).

Figura 51. Nodo función GET AUTO FIN (tipo 1 auto finalizado)

Figura 52. Nodo función GET AUTO FIN (tipo 2 auto finalizado).

Figura 53. Nodo función GET AUTO FIN (tipo 3 auto finalizado)

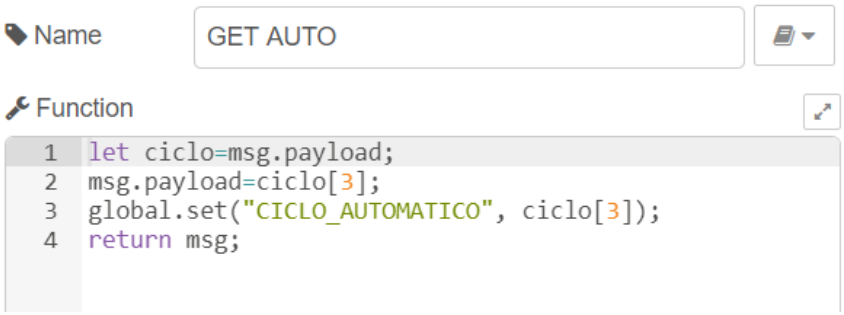

Figura 54. Nodo función GET AUTO (led)

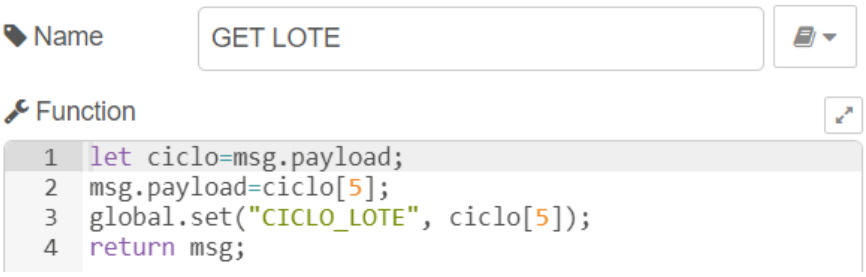

Figura 55. Nodo función GET LOTE (led).

Name toJASON

Function

```
let tipo1=msg.payload["TIPO1"];
let tipo2= msg.payload["TIPO2"];
let tipo3=msg.payload["TIPO3"];

let automatico= global.get("CICLO_AUTOMATICO");
let lote= global.get("CICLO_LOTE");

if(automatico){
msg.payload=[
    {
        measurement: "AutoFinalizado",
        fields: {
            TIPO1: tipo1,
            TIPO2: tipo2,
            TIPO3: tipo3
        }
    } ]
return msg;
}
```

Figura 56. Nodo función toJASON (auto finalizado).

Name SET CLICK

Function

```
let ciclo;
ciclo= flow.get("MW1223")[5];
if(ciclo===1 ){
    return msg;
}
```

Figura 57. Nodo función SET CLICK (cancelar tipo 1, tipo 2, tipo 3, todos).

Figura 58. Nodo función SET CLICK (ciclo automático).

Figura 59. Nodo función SET CLICK (ciclo lote).

Figura 60. Nodo función SET CLICK (cancelar automático).

Name SET DATA

Function

```
let word_MW1223, num_MW1223 = 0;

word_MW1223= flow.get("MW1223");
word_MW1223[6]=1;

flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+=Math.pow(2,i);}}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 61. Nodo función SET DATA (cancelar tipo 1).

Name SET DATA

Function

```
let word_MW1223, num_MW1223 = 0;

word_MW1223= flow.get("MW1223");

word_MW1223[7]=1;

flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+=Math.pow(2,i);
    }
}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 62. Nodo función SET DATA (cancelar tipo 2).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[8]=1;
flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+=Math.pow(2,i);
    }
}

msg.datatype=6;
msg.address=1223;
msg.payload=num_MW1223;

return msg;
```

Figura 63. Nodo función SET DATA (cancelar tipo 3).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[9]=1;
flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+=Math.pow(2,i);
    }
}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 64. . Nodo función SET DATA (cancelar todos).

Name SET_DATA

Function

```
let word_MW1223, num_MW1223 = 0;

word_MW1223= flow.get("MW1223");

word_MW1223[3]=1;

flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+=Math.pow(2,i);
    }
}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 65. Nodo función SET DATA (ciclo automático).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[5]=1;
flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]===1){
        num_MW1223+= Math.pow(2,i);
    }
}

msg.datatype=6;
msg.address=1223;
msg.payload=num_MW1223;

return msg;
```

Figura 66. Nodo función SET DATA (ciclo lote).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[4]=1;
flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]==1){
        num_MW1223+= Math.pow(2,i);
    }
}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 67. Nodo función SET DATA (cancelar automático).

Name FIND ERROR

Function

```
//Error 0: No hay ningún error.
//      1: Falta poner en el Flujo un tipo 1;
//      2: Falta poner en el Flujo un tipo 2;
//      3: Falta poner en el Flujo un tipo 3;
//      4: El lote está vacío;

let lote=[],flujo=[],error=[0,0,0,0];

lote= global.get("LOTE_CANTIDAD");
flujo=global.get("FLUJO");

if(lote[0]>0 & !flujo.includes(1)){
    error[1]=1; error[0]=1}
if(lote[1]>0 & !flujo.includes(2) ){
    error[2]=1;error[0]=1}
if(lote[2]>0 & !flujo.includes(3) ){
    error[3]=1;error[0]=1}

if(lote[0]===0 & lote[1]===0 & lote[2]===0 ){
    error[4]=1;
    error[0]=1;
}
msg.payload=error;
return msg;
```

Figura 68. Nodo función FIND ERROR.

Name ERROR

Function

```
let msg_error="";

if(msg.payload[1]===1){
    msg_error+="Falta Producto Tipo 1 en FLUJOS.<br>";
}
if(msg.payload[2]===1){
    msg_error+="Falta Producto Tipo 2 en FLUJOS.<br>";
}
if(msg.payload[3]===1){
    msg_error+="Falta Producto Tipo 3 en FLUJOS.<br>";
}
if(msg.payload[4]===1){
    msg_error+="El lote está vacío.<br>";
}

msg.payload=msg_error;
return msg;
```

Figura 69. Nodo función ERROR.

Name CORRECT

Function

```
if(msg.payload[0]===0){

msg.payload=true;
return msg;}
```

Figura 70. Nodo función CORRECT.

Name SET CAMBIO

Function

```
global.set("CAMBIO",false);
return msg;
```

Figura 71. Nodo función SET CAMBIO.

Name GET TIPO

Function

```
1  let ciclo_lote = flow.get("MW1223")[5];
2
3  msg.datatype = 6;
4  msg.address = 1227;
5
6  if(ciclo_lote === 0){
7      return msg;
8  }
```

Figura 72. Nodo función GET TIPO (tipo 1).

Name GET TIPO

Function

```
1  let ciclo_lote = flow.get("MW1223")[5];
2
3  msg.datatype = 6;
4  msg.address = 1228;
5
6  if(ciclo_lote === 0){
7      return msg;
8  }
```

Figura 73. Nodo función GET TIPO (tipo 2).

Name GET TIPO

Function

```
1  let ciclo_lote = flow.get("MW1223")[5];
2
3  msg.datatype = 6;
4  msg.address = 1229;
5
6  if(ciclo_lote === 0){
7      return msg;
8  }
```

Figura 74. Nodo función GET TIPO (tipo 3).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[0]=1;
flow.set("MW1223",word_MW1223);

for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]==1){
        num_MW1223+= Math.pow(2,i);
    }
}

global.set("CAMBIO",true);

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 75. Nodo función SET DATA (cambio flujo de producción).

Name SET DATA

Function

```
let word_MW1223, num_MW1223=0;

word_MW1223= flow.get("MW1223");
word_MW1223[1]=1;
flow.set("MW1223",word_MW1223);
global.set("CAMBIO",false);
for(let i=0; i<word_MW1223.length; i++){
    if (word_MW1223[i]==1){
        num_MW1223+= Math.pow(2,i);
    }
}

msg.datatype = 6;
msg.address = 1223;
msg.payload=num_MW1223;

return msg;
```

Figura 76. Nodo función SET DATA (finalizado cambio flujo).

Name SET DATA

Function

```
1  let word_MW1223, num_MW1223=0;
2
3  word_MW1223= flow.get("MW1223");
4  word_MW1223[2]=1;
5  flow.set("MW1223",word_MW1223);
6
7  for(let i=0; i<word_MW1223.length; i++){
8      if (word_MW1223[i]==1){
9          num_MW1223+= Math.pow(2,i);
10     }
11 }
12
13 msg.datatype = 6;
14 msg.address = 1223;
15 msg.payload=num_MW1223;
16
17 return msg;
```

Figura 77. Nodo función SET DATA (abastecer materias).

Name SET DATA

Function

```
1  msg.datatype = 6;
2  msg.address = 1224;
3  global.set("MATERIAS",msg.payload);
4  return msg;
```

Figura 78. Nodo función SET DATA (cantidad).

1.3. FLUJO PULMÓN

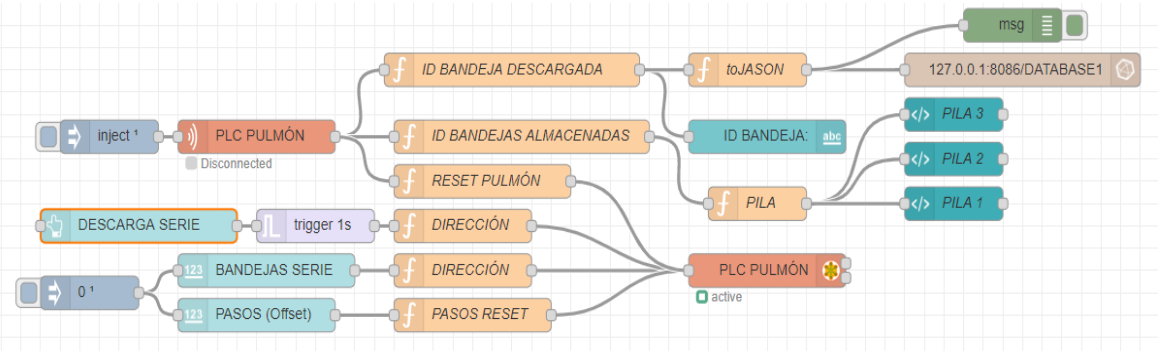

Figura 79. Flujo Pulmón.

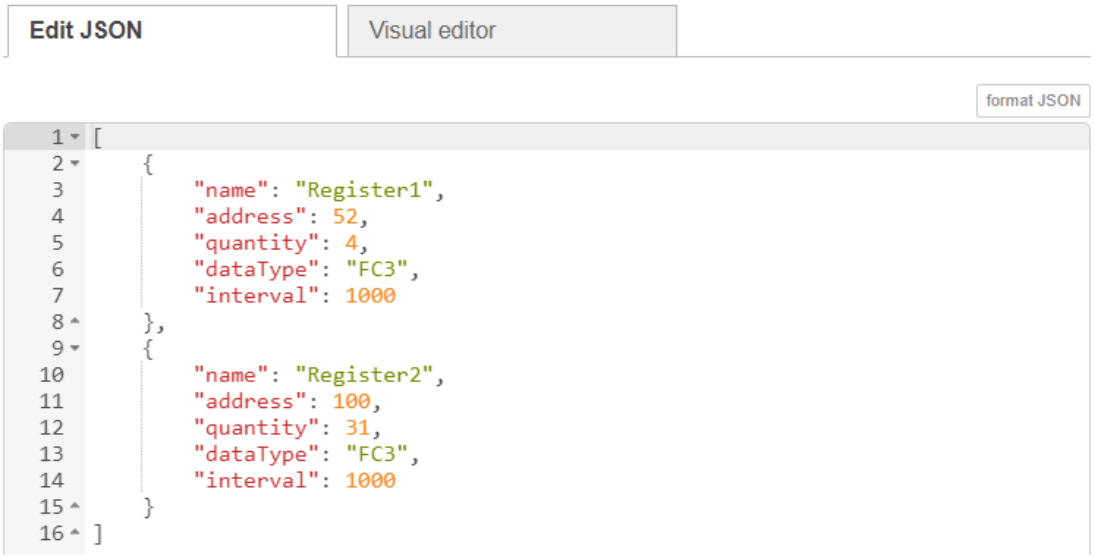

Figura 80. Nodo inject (json lectura PLC CAN).

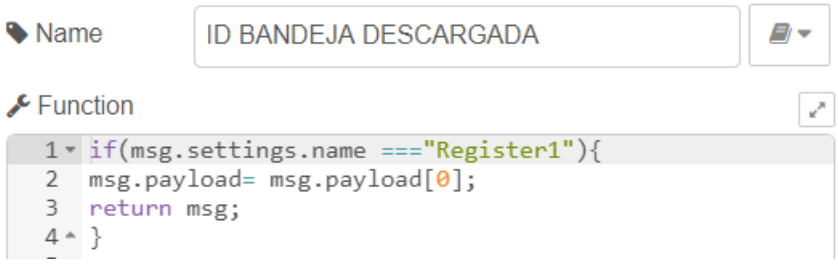

Figura 81. Nodo función ID BANDEJA DESCARGADA.

Name ID BANDEJAS ALMACENADAS

Function

```
if(msg.settings.name ==="Register2"){
return msg;
}
```

Figura 82. Nodo función ID BANDEJAS ALMACENADAS.

Name RESET PULMÓN

Function

```
if(msg.settings.name ==="Register1"){
    if(msg.payload[1]){
        let array=[];

        for(let i=0; i<30; i++){
            array[i]=0;
        }

       msg.payload = {
            value:array,
            'fc':16,
            'unitid': 1,
            'address':101,
            'quantity':30
        }

      //  msg.payload=array
        return msg;
    }
}
```

Figura 83. Nodo función RESET PULMÓN.

Name DIRECCIÓN

Function

```
msg.payload = {
            value:msg.payload,
            'fc':6,
            'unitid': 1,
            'address':99,
            'quantity':1
        }
return msg;
```

Figura 84. Nodo función DIRECCIÓN (descarga manual serie).

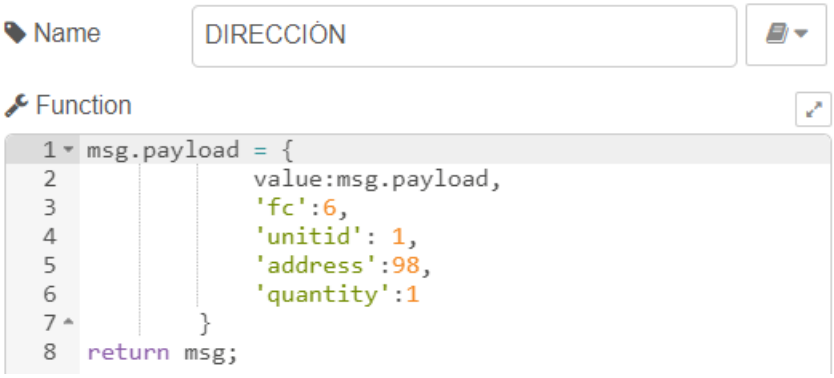

Figura 85. Nodo función DIRECCIÓN (bandejas serie).

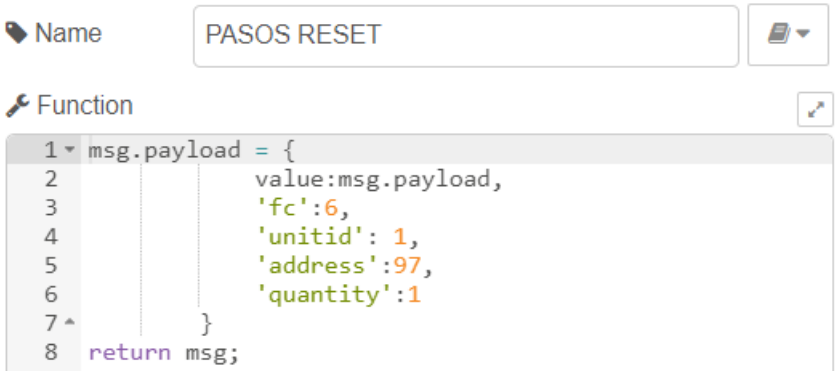

Figura 86 Nodo función PASOS RESET.

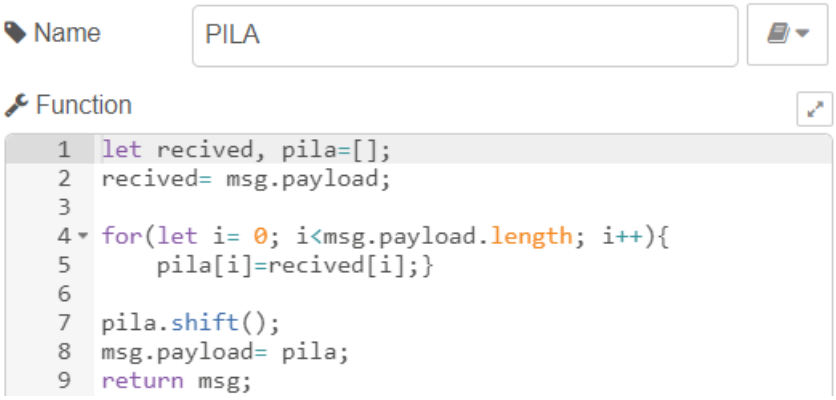

Figura 87. Nodo función PILA.

```
<style>
    .pos{
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items: center;


    }

    .box{
      display: flex;
      justify-content: center;
      align-items: center;
      border: 1px solid rgb(146, 144, 255);
      border-radius: 2rem;
      flex:1;
        }

     #text{
         flex:1;
         text-align: center;
     }

    p{  text-align:center;
        padding-right:1rem;
        padding-left:1rem;
        flex: 0.5;
    }
    #almacen{
        display: flex;
        flex-direction: column;
        align-items: stretch;
        justify-content: center;
    }
```

Figura 88. Nodo template PILA 3 [1].

```
    #head{

        padding-bottom:0.5rem ;
    }


</style>


<div id="almacen">

    <div id="head">
    <div class="pos">
        <p> <font color="#85b4ff"> POSICIÓN </font> </p>
         <div id=text> <font color="#85b4ff"> ID BANDEJAS </font> </div>
    </div>
    </div>

    <div  class="pos">
        <p>30: </p>
         <div class="box" style="{{(msg.payload[29] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[29]}} </div>
    </div>

    <div class="pos">
        <p>29: </p>
         <div class="box" style="{{(msg.payload[28] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[28]}} </div>
    </div>

    <div class="pos">
         <p>28: </p>
          <div class="box" style="{{(msg.payload[27] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[27]}} </div>
    </div>

    <div class="pos">
        <p>27: </p>
         <div class="box" style="{{(msg.payload[26] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[26]}} </div>
    </div>
```

Figura 89. Nodo template PILA 3 [2].

```
    <div class="pos">
        <p>26: </p>
         <div class="box" style="{{(msg.payload[25] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[25]}} </div>
    </div>

    <div class="pos">
        <p>25: </p>
         <div class="box" style="{{(msg.payload[24] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[24]}} </div>
    </div>

    <div class="pos">
        <p>24:  </p>
         <div class="box" style="{{(msg.payload[23] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[23]}} </div>
    </div>

    <div class="pos">
        <p>23: </p>
         <div class="box" style="{{(msg.payload[22] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[22]}} </div>
    </div>

    <div class="pos">
        <p>22:  </p>
         <div class="box" style="{{(msg.payload[21] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[21]}} </div>
    </div>

    <div class="pos">
        <p>21:  </p>
         <div class="box" style="{{(msg.payload[20] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[20]}} </div>
    </div>

</div>
```

Figura 90. Nodo template PILA 3 [3].

```
<style>
    .pos{
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items: center;


    }

    .box{
      display: flex;
      justify-content: center;
      align-items: center;
      border: 1px solid rgb(146, 144, 255);
      border-radius: 2rem;
      flex:1;
        }

     #text{
         flex:1;
         text-align: center;
     }

    p{  text-align:center;
        padding-right:1rem;
        padding-left:1rem;
        flex: 0.5;
    }
    #almacen{
        display: flex;
        flex-direction: column;
        align-items: stretch;
        justify-content: center;
    }

```

Figura 91. Nodo template PILA 2 [1].

```
    #head{

        padding-bottom:0.5rem ;
    }
</style>
<div id="almacen">

    <div id="head">
        <div class="pos">
            <p> <font color="#85b4ff"> POSICIÓN </font> </p>
            <div id=text> <font color="#85b4ff"> ID BANDEJAS </font> </div>
        </div>
    </div>

    <div class="pos">
        <p>20: </p>
        <div class="box" style="{{(msg.payload[19] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[19]}} </div>
    </div>


    <div  class="pos">
        <p>19: </p>
        <div class="box" style="{{(msg.payload[18] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[18]}} </div>
    </div>

    <div class="pos">
        <p>18: </p>
        <div class="box" style="{{(msg.payload[17] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[17]}} </div>
    </div>

    <div class="pos">
        <p>17: </p>
        <div class="box" style="{{(msg.payload[16] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[16]}} </div>
    </div>

```

Figura 92. Nodo template PILA 2 [2].

```

    <div class="pos">
        <p>15: </p>
        <div class="box" style="{{(msg.payload[14] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[14]}} </div>
    </div>

    <div class="pos">
        <p>14: </p>
        <div class="box" style="{{(msg.payload[13] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[13]}} </div>
    </div>

    <div class="pos">
        <p>13:  </p>
       <div class="box" style="{{(msg.payload[12] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[12]}} </div>
    </div>

    <div class="pos">
        <p>12: </p>
        <div class="box" style="{{(msg.payload[11] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[11]}} </div>
    </div>

    <div class="pos">
        <p>11:  </p>
        <div class="box" style="{{(msg.payload[10] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[10]}} </div>
    </div>
</div>

```

Figura 93. Nodo template PILA 2 [3].

```
<style>
    .pos{
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items: center;


    }

    .box{
      display: flex;
      justify-content: center;
      align-items: center;
      border: 1px solid rgb(146, 144, 255);
      border-radius: 2rem;
      flex:1;
        }

     #text{
         flex:1;
         text-align: center;
     }

    p{  text-align:center;
        padding-right:1rem;
        padding-left:1rem;
        flex: 0.5;
    }
    #almacen{
        display: flex;
        flex-direction: column;
        align-items: stretch;
        justify-content: center;
    }

```

Figura 94. Nodo template PILA 1 [1].

```
     #head{

        padding-bottom:0.5rem ;
    }


</style>


<div id="almacen">

    <div id="head">
    <div class="pos">
        <p> <font color="#85b4ff"> POSICIÓN </font> </p>
         <div id=text> <font color="#85b4ff"> ID BANDEJAS </font> </div>
    </div>
    </div>

    <div  class="pos">
        <p>10: </p>
        <div class="box" style="{{(msg.payload[9] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[9]}} </div>
    </div>

    <div class="pos">
        <p>9: </p>
        <div class="box" style="{{(msg.payload[8] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[8]}} </div>
    </div>

    <div class="pos">
        <p>8: </p>
        <div class="box" style="{{(msg.payload[7] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[7]}} </div>
    </div>

    <div class="pos">
        <p>7: </p>
        <div class="box" style="{{(msg.payload[6] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[6]}} </div>
    </div>
```

Figura 95. Nodo template PILA 1 [2].

```
74
75      <div class="pos">
76          <p>6: </p>
77          <div class="box" style="{{(msg.payload[5] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[5]}} </div>
78      </div>
79
80      <div class="pos">
81          <p>5: </p>
82          <div class="box" style="{{(msg.payload[4] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[4]}} </div>
83      </div>
84
85      <div class="pos">
86          <p>4:  </p>
87          <div class="box" style="{{(msg.payload[3] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[3]}} </div>
88      </div>
89
90      <div class="pos">
91          <p>3: </p>
92          <div class="box" style="{{(msg.payload[2] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[2]}} </div>
93      </div>
94
95      <div class="pos">
96          <p>2:  </p>
97          <div class="box" style="{{(msg.payload[1] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[1]}} </div>
98      </div>
99
100     <div class="pos">
101         <p>1:  </p>
102         <div class="box" style="{{(msg.payload[0] === 0) ? 'background-color: none;' : 'background-color:#FF007A;'}}"> {{msg.payload[0]}} </div>
103     </div>
104
105 </div>
```

Figura 96. Nodo template PILA 1 [3].

1.4. FLUJO RETENEDORES

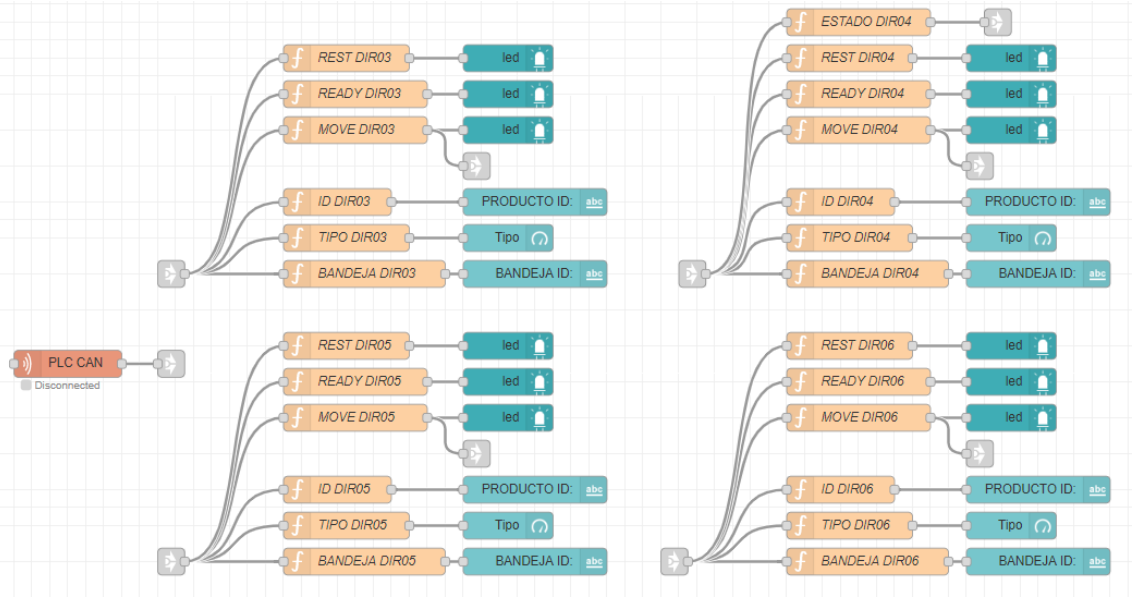

Figura 97. Flujo Retenedores [1].

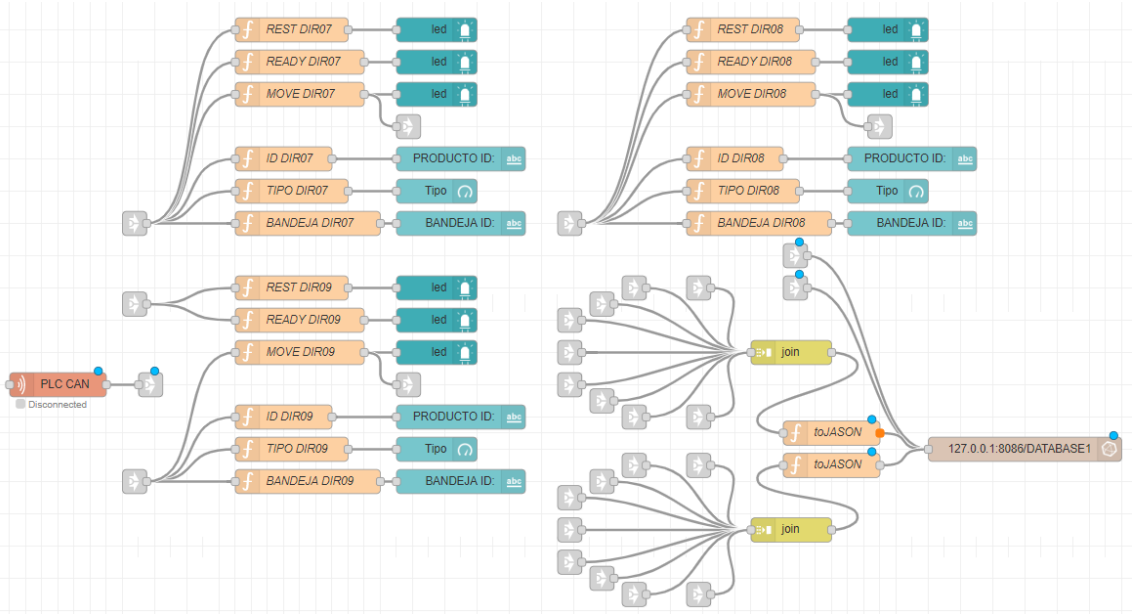

Figura 98. Flujo Retenedores [2].

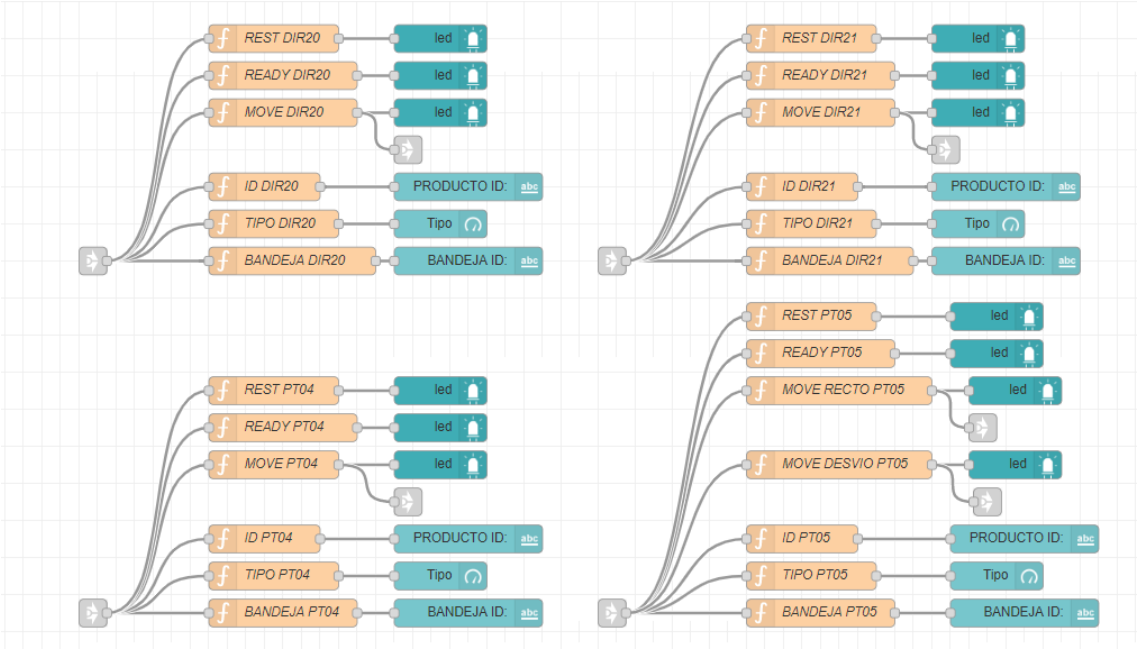

Figura 99. Flujo Retenedores [3].

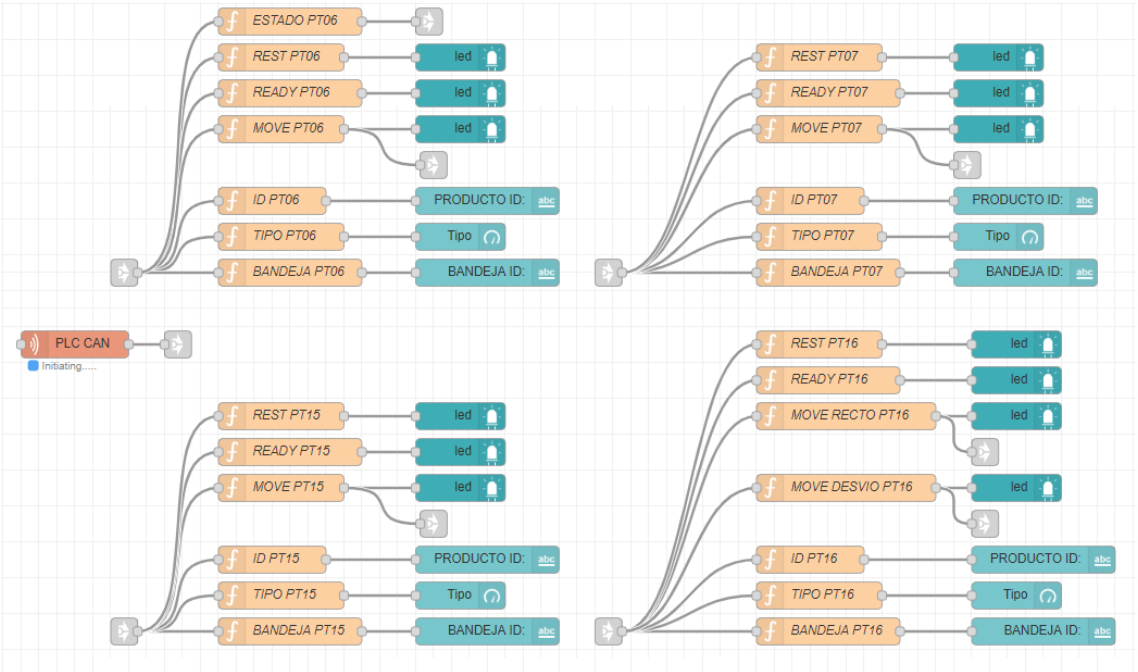

Figura 100. Flujo Retenedores [3].

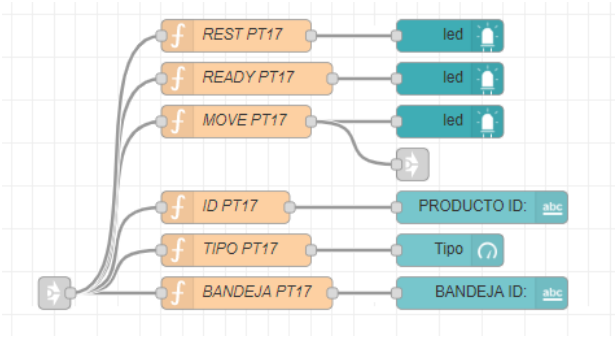

Figura 101. Flujo Retenedores [4].

Figura 102. Nodo función REST DIR03.

Figura 103. Nodo función REST DIR04.

Figura 104. Nodo función REST DIR05.

Name REST DIR06

Function

```
let rest=0;
rest= msg.payload[25];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 105. Nodo función REST DIR06.

Name REST DIR07

Function

```
let rest=0;
rest= msg.payload[33];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 106. Nodo función REST DIR07.

Name REST DIR08

Function

```
let rest=0;
rest= msg.payload[41];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 107. Nodo función REST DIR08.

Name REST DIR09

Function

```
let rest=0;
rest= msg.payload[49];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 108. Nodo función REST DIR09.

Name REST DIR20

Function

```
let rest=0;
rest= msg.payload[15];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 109. Nodo función REST DIR20.

Name REST DIR21

Function

```
let rest=0;
rest= msg.payload[23];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 110. Nodo función REST DIR21.

Name REST PT04

Function

```
let rest=0;
rest= msg.payload[31];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 111. Nodo función REST PT04.

Name REST PT05

Function

```
let rest=0;
rest= msg.payload[39];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 112. Nodo función REST PT05.

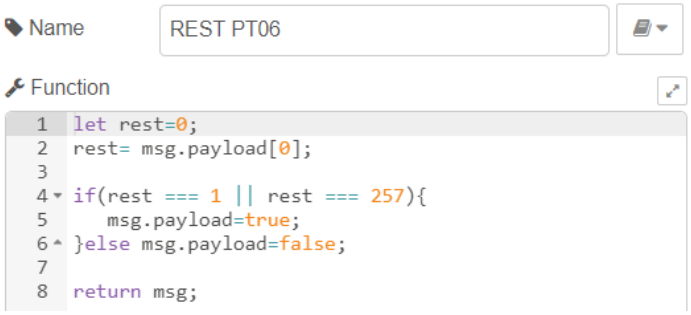

Figura 113. Nodo función REST PT06.

Figura 114. Nodo función REST PT07.

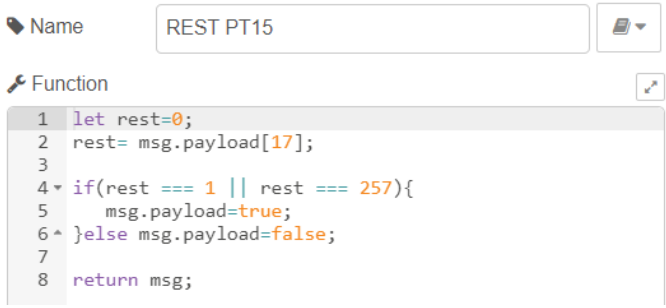

Figura 115. Nodo función REST PT15.

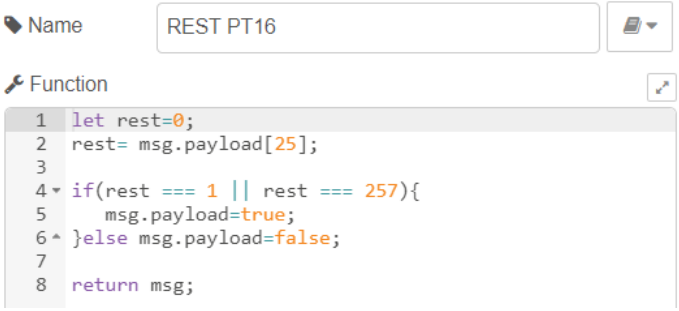

Figura 116. Nodo función REST PT16.

Name REST PT17

Function

```
let rest=0;
rest= msg.payload[36];

if(rest === 1 || rest === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 117. Nodo función REST PT17.

Name READY DIR03

Function

```
let ready=0;
ready= msg.payload[0];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 118. Nodo función READY DIR03.

Name READY DIR04

Function

```
let ready=0;
ready= msg.payload[8];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 119. Nodo función READY DIR04.

Name READY DIR05

Function

```
let ready=0;
ready= msg.payload[17];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 120. Nodo función READY DIR05.

Name READY DIR06

Function

```
let ready=0;
ready= msg.payload[25];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 121. Nodo función READY DIR06.

Name READY DIR07

Function

```
let ready=0;
ready= msg.payload[33];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 122. Nodo función READY DIR07.

Name READY DIR08

Function

```
let ready=0;
ready= msg.payload[41];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 123. Nodo función READY DIR08.

Name READY DIR09

Function

```
let ready=0;
ready= msg.payload[49];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 124. Nodo función READY DIR09.

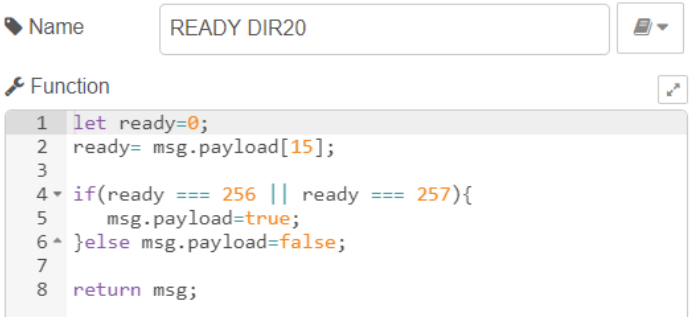

Figura 125. Nodo función READY DIR20.

Figura 126. Nodo función READY DIR21.

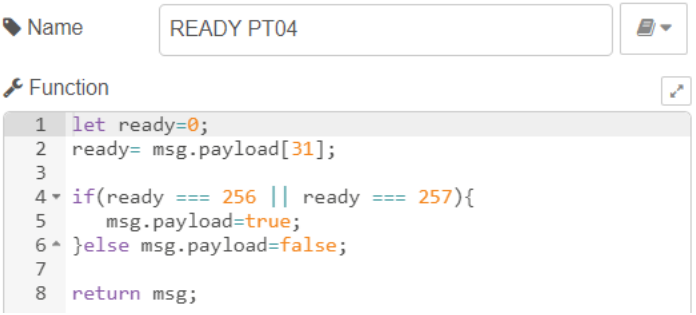

Figura 127. Nodo función READY PT04.

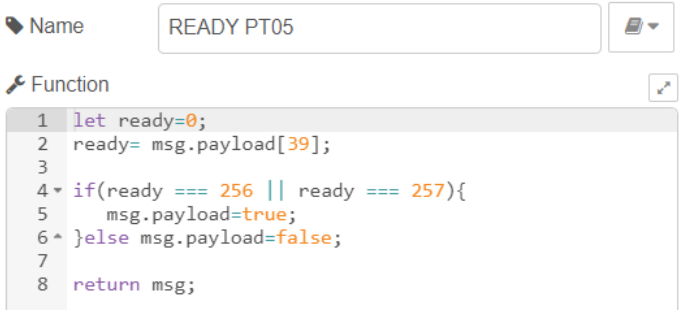

Figura 128. Nodo función READY PT05.

Name READY PT06

Function

```
let ready=0;
ready= msg.payload[0];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 129. Nodo función READY PT06.

Name READY PT07

Function

```
let ready=0;
ready= msg.payload[9];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 130. Nodo función READY PT07.

Name READY PT15

Function

```
let ready=0;
ready= msg.payload[17];

if(ready === 256 || ready === 257){
   msg.payload=true;
}else msg.payload=false;

return msg;
```

Figura 131. Nodo función READY PT15.

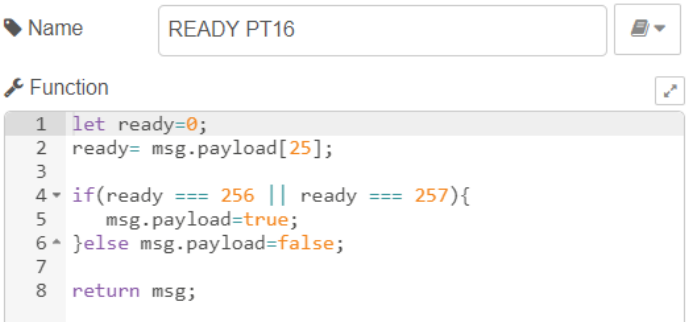

Figura 132. Nodo función READY PT16.

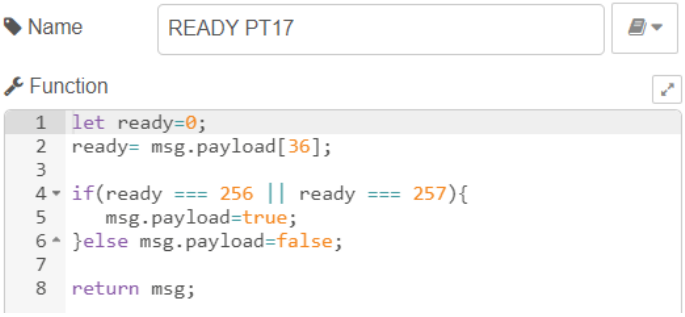

Figura 133. Nodo función READY PT17.

Figura 134. Nodo función MOVE DIR03.

Figura 135. Nodo función MOVE DIR04.

Figura 136. Nodo función MOVE DIR05.

Figura 137. Nodo función MOVE DIR06.

Figura 138. Nodo función MOVE DIR07.

Figura 139. Nodo función MOVE DIR08.

Name MOVE DIR09

Function

```
let ready=0;
ready= msg.payload[0];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="DIR09";
return msg;
```

Figura 140. Nodo función MOVE DIR09.

Name MOVE DIR20

Function

```
let ready=0;
ready= msg.payload[16];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="DIR20";
return msg;
```

Figura 141. Nodo función MOVE DIR20.

Name MOVE DIR21

Function

```
let ready=0;
ready= msg.payload[24];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="DIR21";
return msg;
```

Figura 142. Nodo función MOVE DIR21.

Name MOVE PT04

Function

```
let ready=0;
ready= msg.payload[32];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT04";
return msg;
```

Figura 143. Nodo función MOVE PT04.

Name MOVE RECTO PT05

Function

```
let move=0;
move= msg.payload[42];

if(move === 256 || move === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT05_RECTO";
return msg;
```

Figura 144. Nodo función MOVE RECTO PT05.

Name MOVE DESVIO PT05

Function

```
let move=0;
move = msg.payload[43];

if(move === 1 || move === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT05_DESVIO";
return msg;
```

Figura 145. Nodo función MOVE DESVIO PT05.

Name MOVE PT06

Function

```
let ready=0;
ready= msg.payload[1];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT06";
return msg;
```

Figura 146. Nodo función MOVE PT06.

Name MOVE PT07

Function

```
let ready=0;
ready= msg.payload[10];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT07";
return msg;
```

Figura 147. Nodo función MOVE PT07.

Name MOVE PT15

Function

```
let ready=0;
ready= msg.payload[18];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT15";
return msg;
```

Figura 148. Nodo función MOVE PT15.

Name MOVE RECTO PT16

Function

```
let move=0;
move= msg.payload[28];

if(move === 256 || move === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT16_RECTO";
return msg;
```

Figura 149. Nodo función MOVE RECTO PT16.

Name MOVE DESVIO PT16

Function

```
let move = 0;
move = msg.payload[29];

if(move === 1 || move === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT16_DESVIO";
return msg;
```

Figura 150. Nodo función MOVE DESVIO PT16.

Name MOVE PT17

Function

```
let ready=0;
ready= msg.payload[37];

if(ready === 1 || ready === 257){
   msg.payload=1;
}else msg.payload=0;
msg.topic="PT17";
return msg;
```

Figura 151. Nodo función MOVE PT17.

Name ID DIR03

Function

```
let id=0;
id= msg.payload[2];
msg.payload=id.toString();
return msg;
```

Figura 152. Nodo función ID DIR03.

Name ID DIR04

Function

```
let id=0;

id= msg.payload[11];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 153. Nodo función ID DIR04.

Name ID DIR05

Function

```
let id=0;

id= msg.payload[19];

msg.payload=id.toString();

return msg;
```

Figura 154. Nodo función ID DIR05.

Name ID DIR06

Function

```
let id=0;

id= msg.payload[27];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 155. Nodo función ID DIR06.

Name ID DIR07

Function

```
let id=0;

id= msg.payload[35];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 156. Nodo función ID DIR07.

Name ID DIR08

Function

```
let id=0;

id= msg.payload[43];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 157. Nodo función ID DIR08.

Name ID DIR09

Function

```
let id=0;

id= msg.payload[1];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 158. Nodo función ID DIR09.

Name ID DIR20

Function

```
let id=0;

id= msg.payload[17];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 159. Nodo función ID DIR20.

Name ID DIR21

Function

```
let id=0;

id= msg.payload[25];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 160. Nodo función ID DIR21.

Name ID PT04

Function

```
let id=0;

id= msg.payload[33];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 161. Nodo función ID PT04.

Name ID PT05

Function

```
let id=0;

id= msg.payload[44];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 162. Nodo función ID PT05.

Name ID PT06

Function

```
let id=0;

id= msg.payload[3];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 163. Nodo función ID PT06.

Name ID PT07

Function

```
let id=0;

id= msg.payload[11];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 164. Nodo función ID PT07.

Name ID PT15

Function

```
let id=0;

id= msg.payload[19];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 165. Nodo función ID PT15.

Name ID PT16

Function

```
let id=0;

id= msg.payload[30];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 166. Nodo función ID PT16.

Name ID PT17

Function

```
let id=0;

id= msg.payload[38];

msg.payload=id.toString();
console.log(id.toString());
return msg;
```

Figura 167. Nodo función ID PT17.

Name TIPO DIR03

Function

```
let tipo=0;
tipo= msg.payload[3];
msg.payload=tipo;
return msg;
```

Figura 168. Nodo función TIPO DIR03.

Figura 169. Nodo función TIPO DIR04.

Figura 170. Nodo función TIPO DIR05.

Figura 171. Nodo función TIPO DIR06.

Figura 172. Nodo función TIPO DIR07.

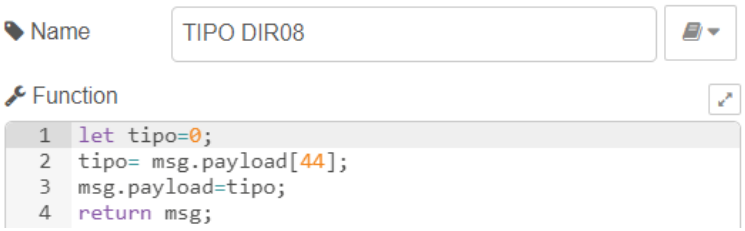

Figura 173. Nodo función TIPO DIR08.

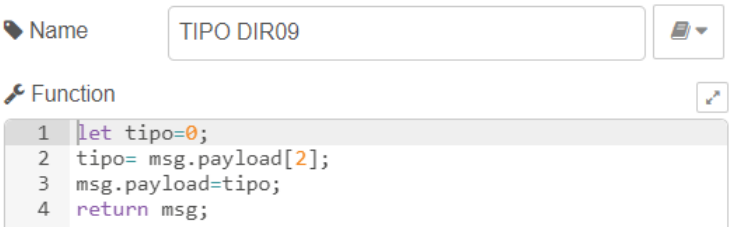

Figura 174. Nodo función TIPO DIR09.

Figura 175. Nodo función TIPO DIR20.

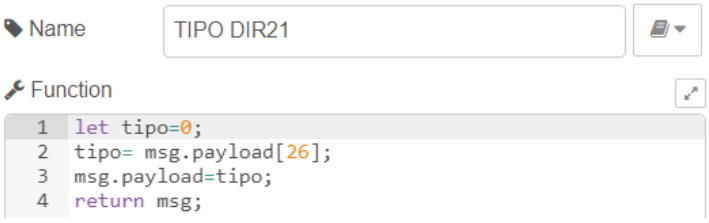

Figura 176. Nodo función TIPO DIR21.

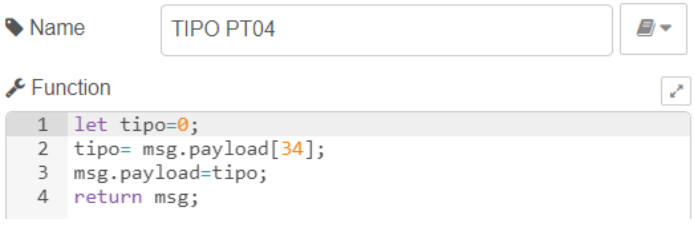

Figura 177. Nodo función TIPO PT04.

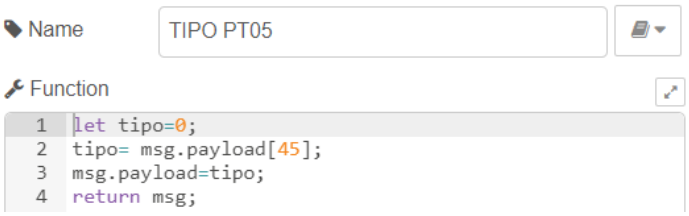

Figura 178. Nodo función TIPO PT05.

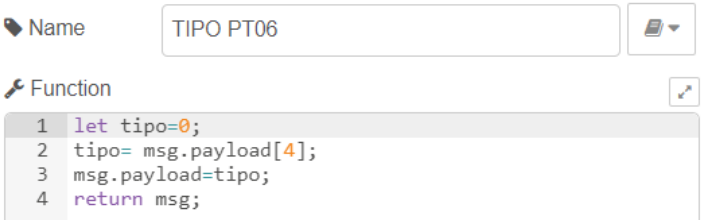

Figura 179. Nodo función TIPO PT06.

Figura 180. Nodo función TIPO PT07.

Figura 181. Nodo función TIPO PT15.

Figura 182. Nodo función TIPO PT16.

Figura 183. Nodo función TIPO PT17.

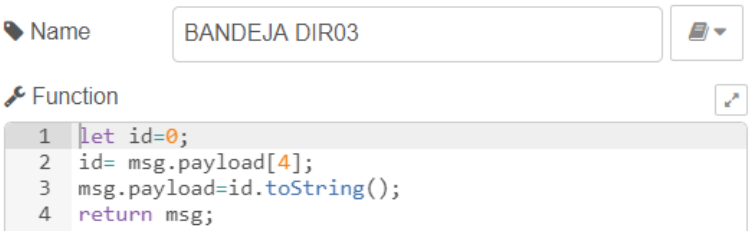

Figura 184. Nodo función BANDEJA DIR03.

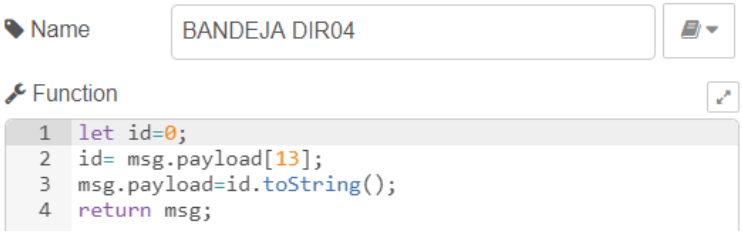

Figura 185. Nodo función BANDEJA DIR04.

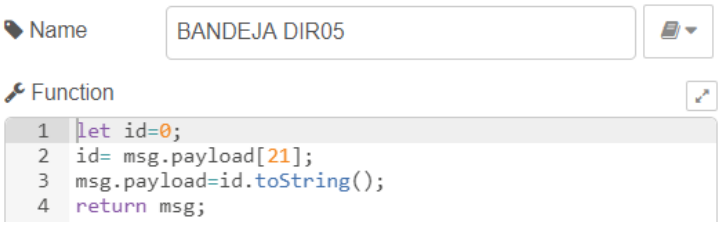

Figura 186. Nodo función BANDEJA DIR05.

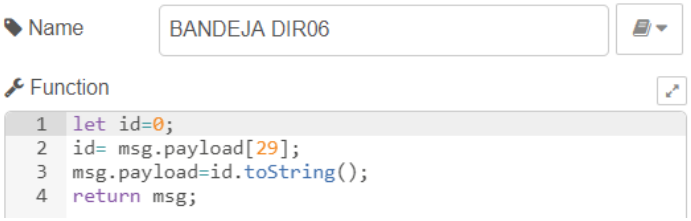

Figura 187. Nodo función BANDEJA DIR06.

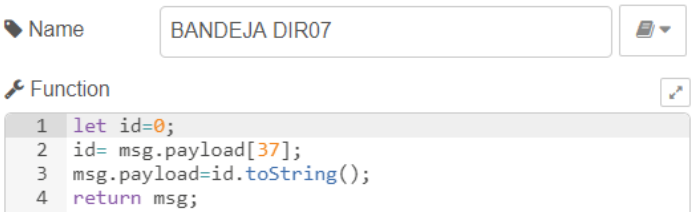

Figura 188. Nodo función BANDEJA DIR07.

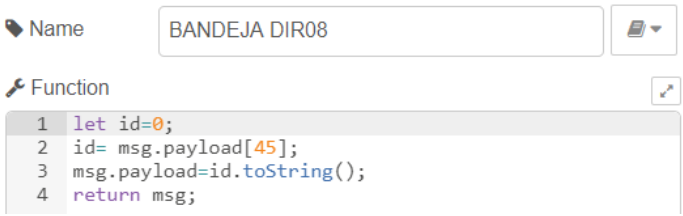

Figura 189. Nodo función BANDEJA DIR08.

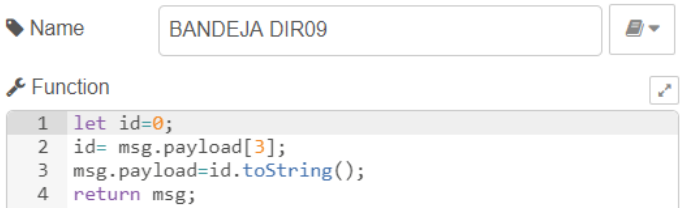

Figura 190. Nodo función BANDEJA DIR09.

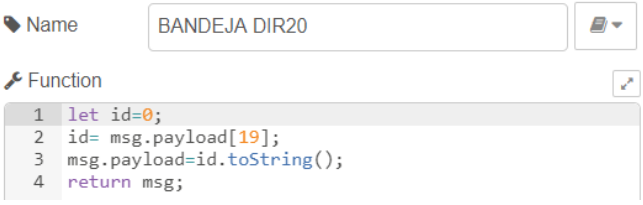

Figura 191. Nodo función BANDEJA DIR20.

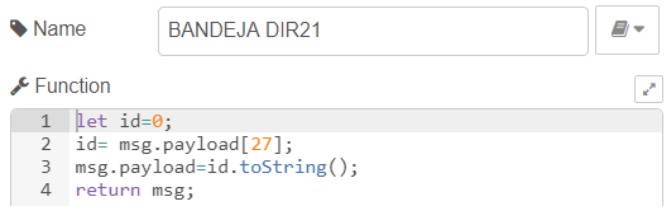

Figura 192. Nodo función BANDEJA DIR21.

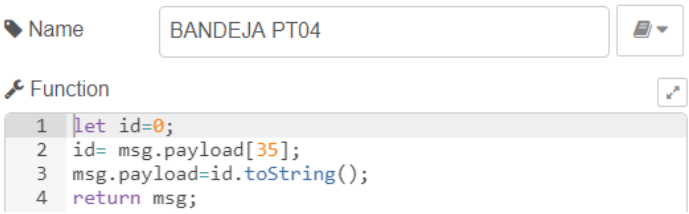

Figura 193. Nodo función BANDEJA PT04.

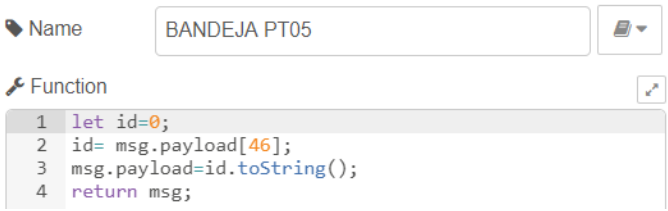

Figura 194. Nodo función BANDEJA PT05.

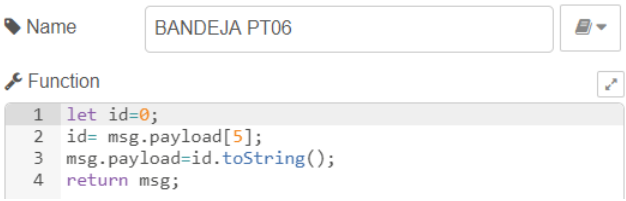

Figura 195. Nodo función BANDEJA PT06.

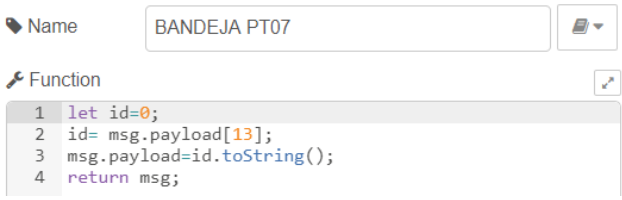

Figura 196. Nodo función BANDEJA PT07.

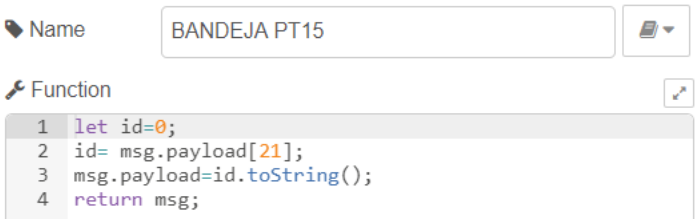

Figura 197. Nodo función BANDEJA PT15.

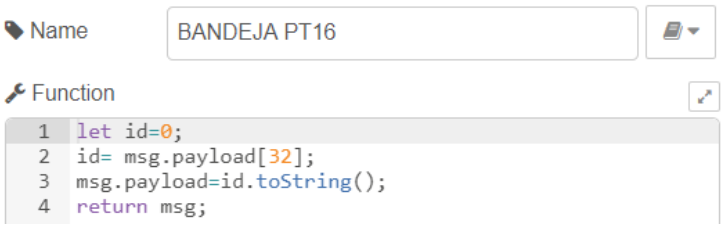

Figura 198. Nodo función BANDEJA PT16.

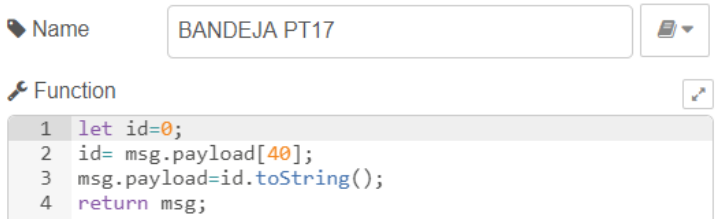

Figura 199. Nodo función BANDEJA PT17.

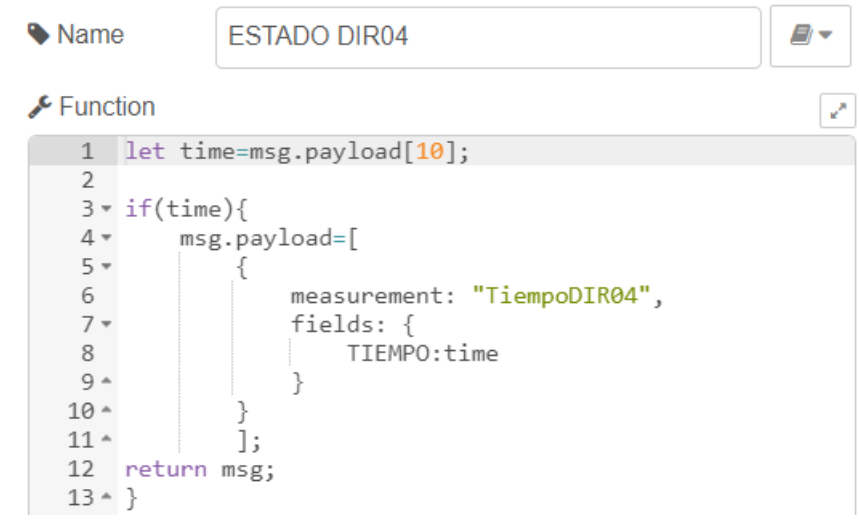

Figura 200. Nodo función ESTADO DIR04.

Name ESTADO PT06

Function

```
let time=msg.payload[2];

if(time){
    msg.payload=[
        {
            measurement: "TiempoPT06",
            fields: {
                TIEMPO:time
            }
        }
        ];
return msg;
}
```

Figura 201. Nodo función ESTADO PT06.

```
let emergency= global.get("EMERGENCY");

let dir03=msg.payload["DIR03"];
let dir04=msg.payload["DIR04"];
let dir05=msg.payload["DIR05"];
let dir06=msg.payload["DIR06"];
let dir07=msg.payload["DIR07"];
let dir08=msg.payload["DIR08"];
let dir09=msg.payload["DIR09"];
let dir20=msg.payload["DIR20"];
let dir21=msg.payload["DIR21"];

let automatico= global.get("CICLO_AUTOMATICO");
let lote= global.get("CICLO_LOTE");

if(emergency & (automatico || lote)){

msg.payload=[
    {
        measurement: "MoveRetenedores",
        fields: {
            MOVE_DIR03: dir03,
            MOVE_DIR04: dir04,
            MOVE_DIR05: dir05,
            MOVE_DIR06: dir06,
            MOVE_DIR07: dir07,
            MOVE_DIR08: dir08,
            MOVE_DIR09: dir09,
            MOVE_DIR20: dir20,
            MOVE_DIR21: dir21
        }
    } ];

return msg;
}
```

Figura 202. Nodo función toJASON (estado movimiento retenedores.)

```
let emergency= global.get("EMERGENCY");

let pt04=msg.payload["PT04"];
let pt05_recto=msg.payload["PT05_RECTO"];
let pt05_desvio=msg.payload["PT05_DESVIO"];
let pt06=msg.payload["PT06"];
let pt07=msg.payload["PT07"];
let pt15=msg.payload["PT15"];
let pt16_recto=msg.payload["PT16_RECTO"];
let pt16_desvio=msg.payload["PT16_DESVIO"];
let pt17=msg.payload["PT17"];

let automatico= global.get("CICLO_AUTOMATICO");
let lote= global.get("CICLO_LOTE");

if(emergency & (automatico || lote)){

msg.payload=[
    {
        measurement: "MovePlataformas",
        fields: {
            MOVE_PT04: pt04,
            MOVE_PT05_RECTO: pt05_recto,
            MOVE_PT05_DESVIO: pt05_desvio,
            MOVE_PT06: pt06,
            MOVE_PT07: pt07,
            MOVE_PT15: pt15,
            MOVE_PT16_RECTO: pt16_recto,
            MOVE_PT16_DESVIO: pt16_desvio,
            MOVE_PT17: pt17
        }
    }
    ];

return msg;
}

```

Figura 203. Nodo función toJASON (estado movimiento plataformas).