Memoria de prácticas STF

Sistemas Tolerantes a Fallos – 4º Grado de Ingeniería Informática

Universidad de Córdoba, EPSC

2021/2022

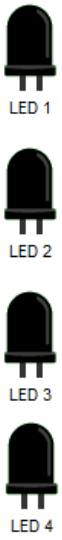
Autores:

Antonio Gómez Giménez (i72gogia@uco.es)

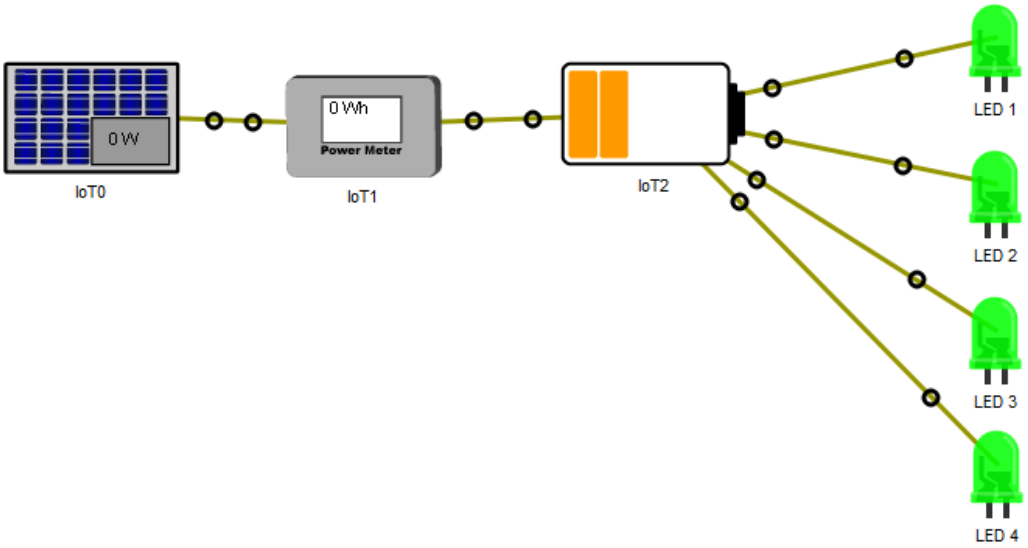
Índice:

1. Práctica 1: Simulando Sistemas Empotrados usando Packet Tracer

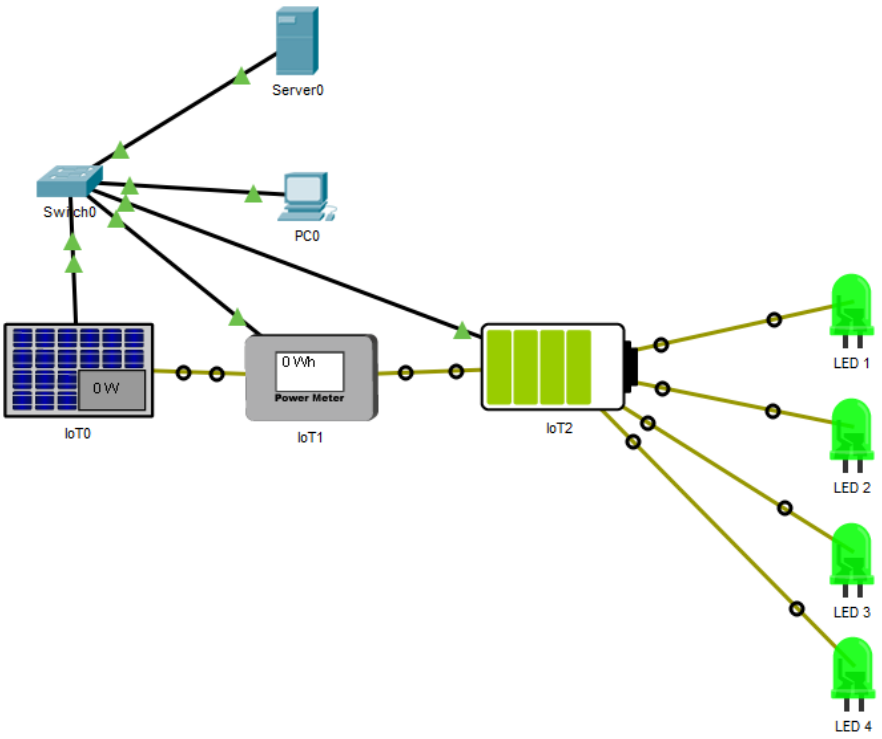
En esta práctica primeramente se nos da el siguiente circuito:



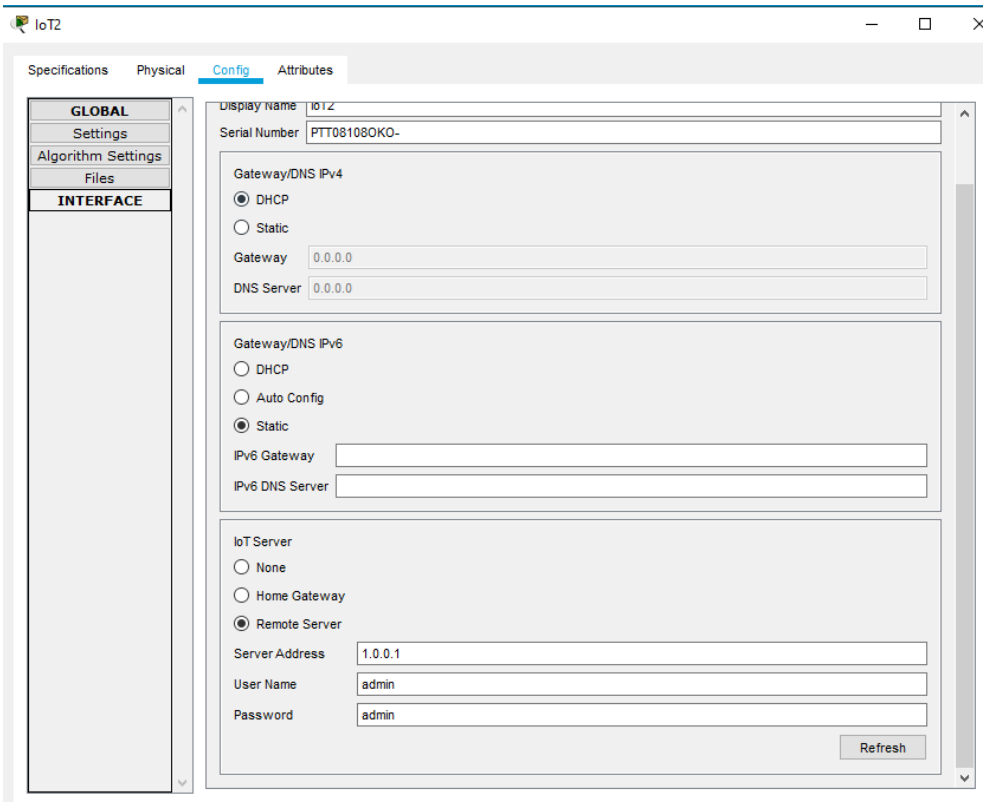
Usando este circuito, complementaremos con los componentes pedidos y realizaremos las conexiones entre estos componentes. El circuito quedaría de la siguiente forma:



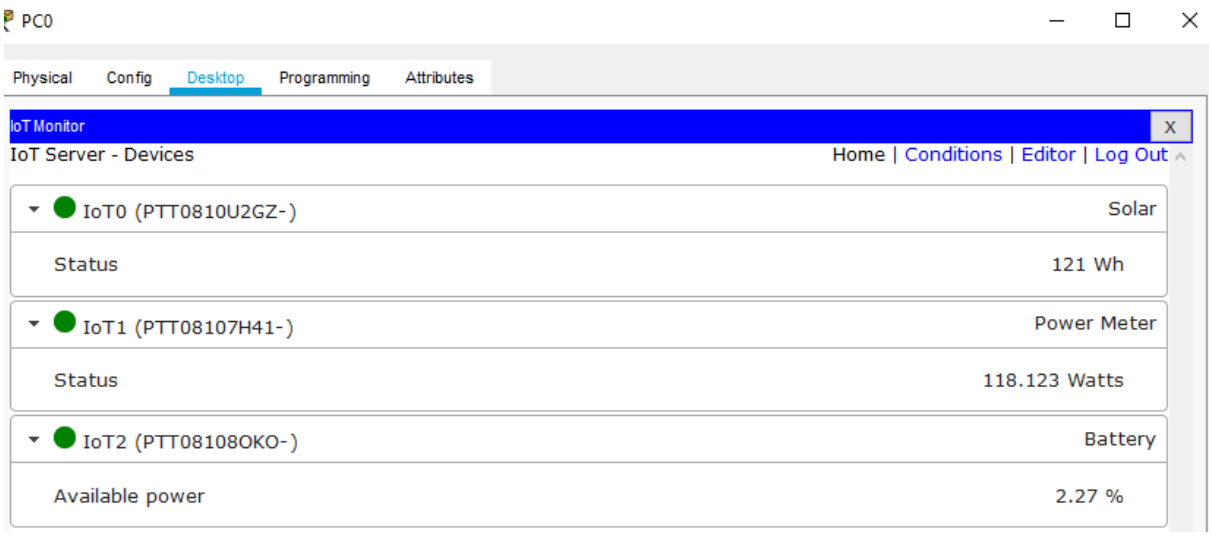
Dependiendo de la hora del día, el panel solar dará cierta potencia que se almacenará en la pila y alimentará los LED. El valor de hora del día se puede modificar en el simulador de cisco e incluso la velocidad del paso del tiempo en la simulación.
Una vez realizado el paso anterior se añadirá conectividad al circuito, para ello usaremos un switch, un servidor donde alojaremos el DHCP y dará ip a todos los componentes del circuito y un pc donde se podrán observar todos los valores de información útiles que extraemos del circuito.



Una vez tenemos el circuito, configuramos cada componente (la ip del servidor que da el servicio DHCP la ponemos nosotros) de tal forma que recibirá su ip con DHCP. Aparte, en cada componente activaremos el remote server en IoT server que nos permitirá ver toda la información en una página web que nos hemos registrado previamente (hay que poner la dirección ip, el usuario y contraseña).

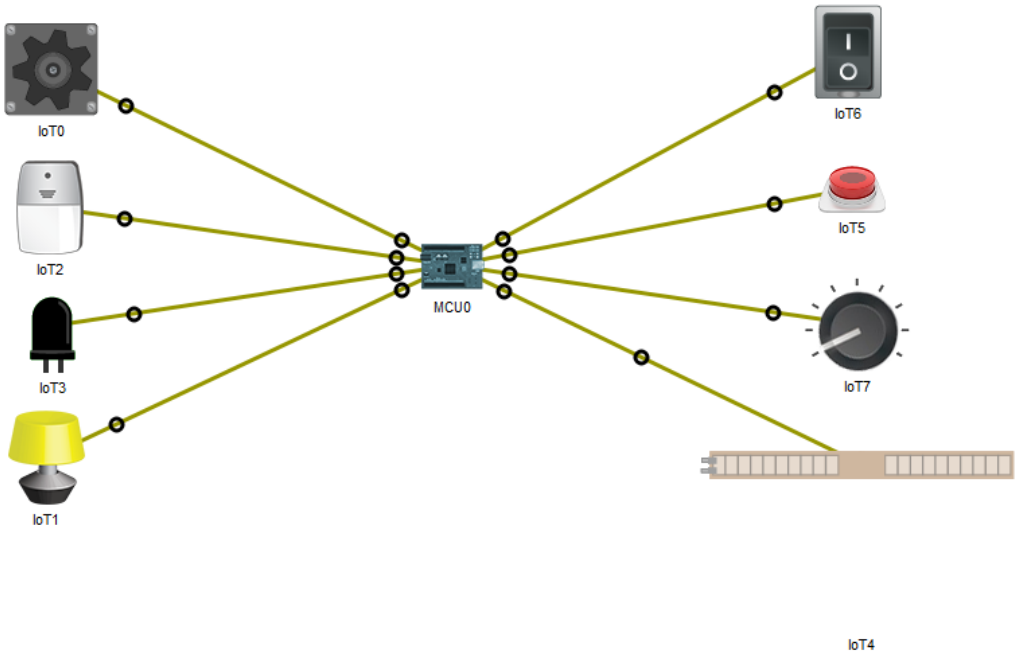


El resultado del circuito con sus configuraciones en la configuración se puede apreciar que funciona correctamente con la siguiente imagen donde se ve toda la información reunida en la página web:



2. Práctica 2: Programando un MCU

Primeramente se va a crear el circuito pedido con las conexiones necesarias en los pines necesarios, el circuito sería el siguiente:



Tras realizar el circuito, diseñar el código de programación del MCU. Primeramente vamos a crear el código para:

• Si se pulsa el interruptor, encender la luz.
• Si se pulsa el pulsador, encender el LED.
• Si se gira el potenciómetro hasta 100 o más, activar la alarma de la sirena.
• Activar la velocidad del motor acorde al ángulo del sensor de flexibilidad, cuánto más se doble, mayor será la velocidad del motor.

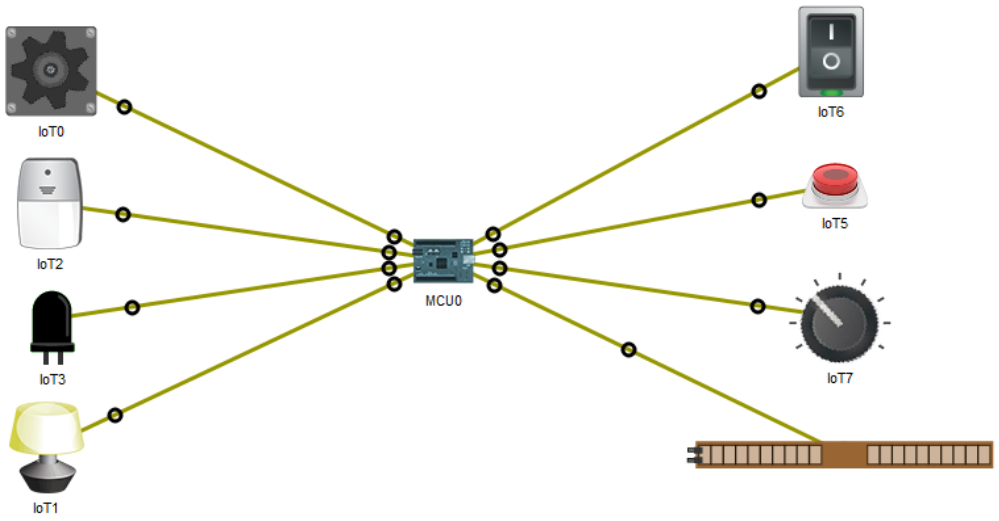
El código inicial para el MCU es el siguiente:

```
1  var lampara, led, sirena, motor, flexor_ant;
2
3  function setup() {//los analogicos a0 (potenciometro) y a1 son entradas
4      pinMode(0, INPUT);//interruptor
5      pinMode(1, INPUT);//pulsador
6      pinMode(2, OUTPUT);//lampara
7      pinMode(3, OUTPUT);//led
8      pinMode(4, OUTPUT);//sirena
9      pinMode(5, OUTPUT);//motor
10
11     lampara = 0;
12     customWrite(2, 0);
13     led = 0;
14     customWrite(3, 0);
15     sirena = 0;
16     customWrite(4, 0);
17     motor = 0;
18     analogWrite(5, 0);
19     flexor_ant = 0;
20 }
```

Con el código anterior nos aseguramos que todo se encuentre apagado al comienzo de la ejecución.

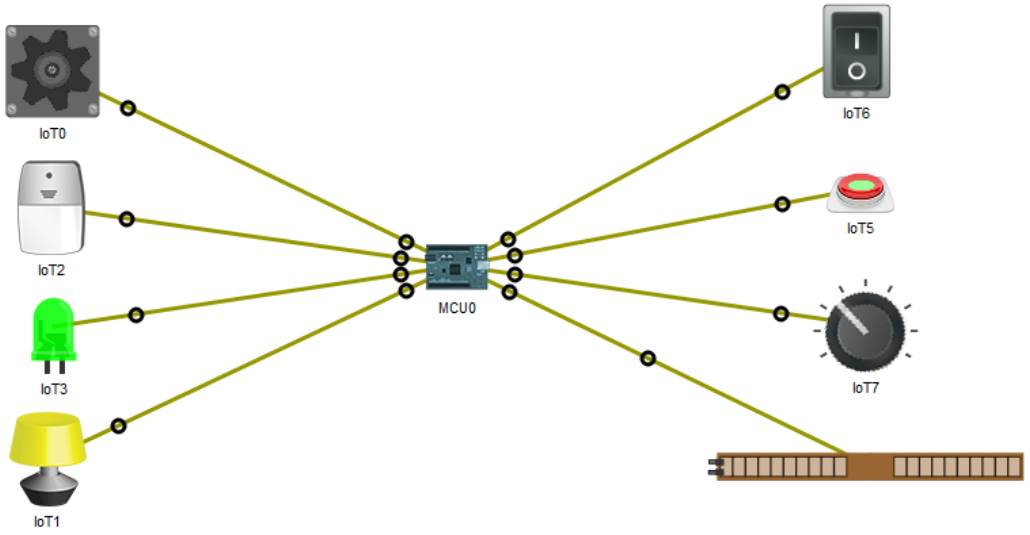
El código para el funcionamiento de la lámpara y su simulación es el siguiente:

```
21
22  function loop() {
23      var interruptor, pulsador, potenciometro, flexor;
24      interruptor = digitalRead(0);
25      pulsador = digitalRead(1);
26      potenciometro = analogRead(A0);
27      flexor = analogRead(A1);
28      //codigo para la lampara
29      if ((interruptor == HIGH)&&(lampara === 0)){
30          lampara = 1;
31          customWrite(2, 2);
32          Serial.println("LAMPARA ENCENDIDA");
33      }else if ((interruptor == LOW)&&(lampara == 1)){
34          lampara = 0;
35          customWrite(2, 0);
36          Serial.println("LAMPARA APAGADA");
37      }
38
```



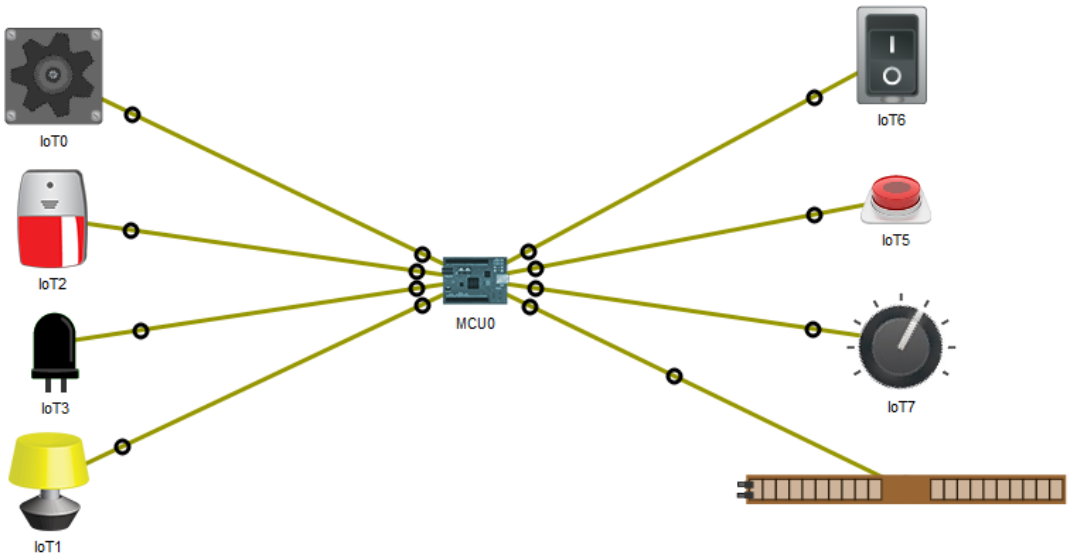
El código para el funcionamiento del LED y su simulación es el siguiente:

```
39      //codigo para el led
40      if ((pulsador == HIGH)&&(led === 0)){
41          led = 1;
42          analogWrite(3, 1023);
43          Serial.println("LED ENCENDIDA");
44      }else if ((pulsador == LOW)&&(led == 1)){
45          led = 0;
46          analogWrite(3, 0);
47          Serial.println("LED APAGADO");
48      }
```



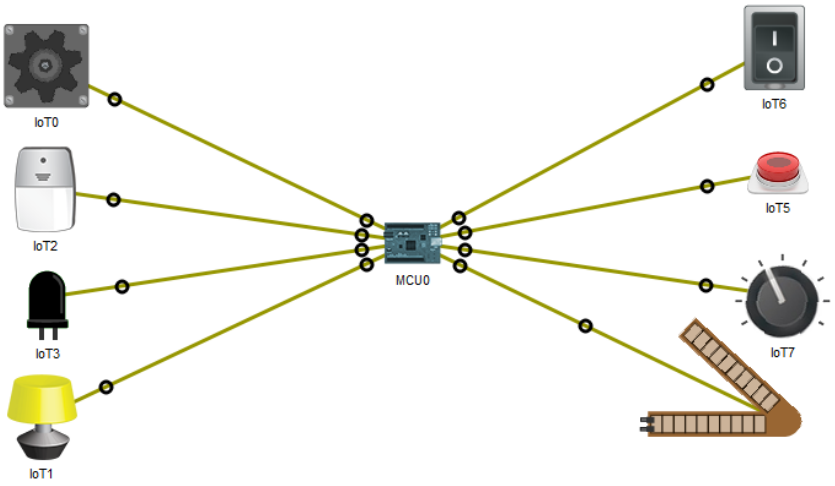
El código para el funcionamiento de la alarma y su simulación es el siguiente:

```
50      //codigo para la sirena
51      if ((potenciometro >= 500)&&(sirena === 0)){
52          sirena = 1;
53          customWrite(4, 1);
54          Serial.println("SIRENA ENCENDIDA");
55      }else if ((potenciometro < 500)&&(sirena == 1)){
56          sirena = 0;
57          customWrite(4, 0);
58          Serial.println("SIRENA APAGADO");
59      }
```



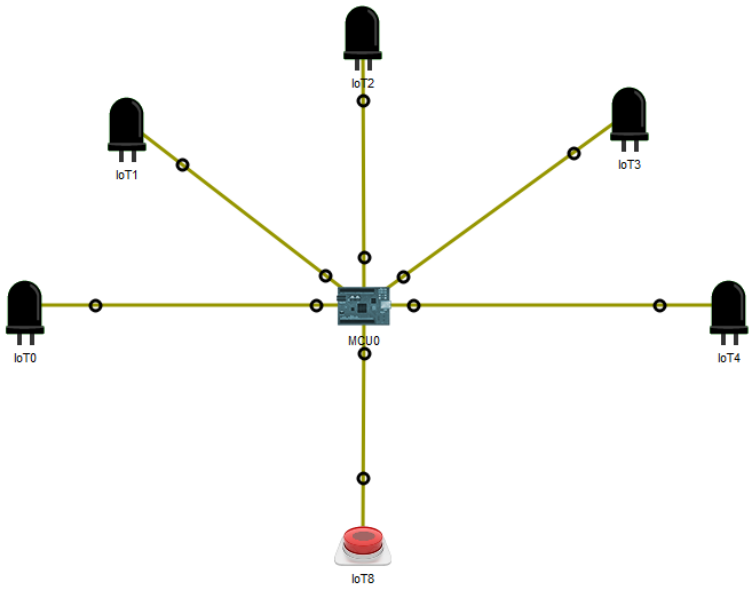
Y por último, el código para el funcionamiento del motor y su simulación es el siguiente (aunque no se va a poder apreciar en una imagen):

```
60      //Serial.println(flexor);
61      //codigo para el motro
62      if (flexor != flexor_ant){
63          flexor_ant = flexor;
64          motor = ((1023 - 0)/(401 - 0))*(flexor-0)+0;//formula para normalizar
65          //Serial.println(motor);
66          analogWrite(5, motor);
67          Serial.println("MOTOR VARIANDO VELOCIDAD");
68      }
69  }
```



En la imagen anterior, dependiendo de cuanto flexiones el sensor, más rápido o más lento girará el motor.

Posteriormente se pide realizar un nuevo circuito donde inicialmente estarán todos los LED apagados y cada vez que se pulse el pulsador se apagará el actualmente iluminado y se encenderá el siguiente LED. Solo habrá un LED encendido cada vez. Cuando esté iluminado el 8º LED, al pulsar el pulsador quedarán los 8 LED apagados, comenzando la iteración de nuevo desde el principio.
El circuito creado se ha modificado usando solo 5 LED ya que para usar 8 leds necesitaríamos más de 6 pines digitales. El circuito es el siguiente:



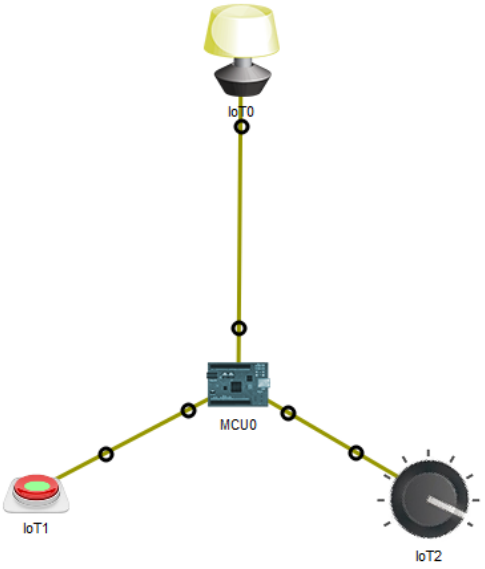
El código realizado para que funcione correctamente es el siguiente:

```
var led, posicion;
var v_led = [1,2,3,4,5];

function setup() {//los analogicos a0 (potenciometro) y al son entradas
    pinMode(0, INPUT);//interruptor
    pinMode(1, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);

    led = 0;
    posicion = 0;

    for(var i = 0; i< 5; i++){//limpiamos los led
        analogWrite(v_led[i], 0);
    }
}

function loop() {
    var pulsador;
    pulsador = digitalRead(0);


    //codigo para el led
    if ((pulsador == HIGH)&&(led === 0)){
        led = 1;
        if (posicion != 5){
            if(posicion !== 0){
                analogWrite(v_led[posicion-1], 0);
            }
            analogWrite(v_led[posicion], 1023);
            posicion++;
        }else{
            analogWrite(v_led[posicion-1], 0);
            posicion = 0;
        }

    }else if ((pulsador == LOW)&&(led == 1)){
        led = 0;
    }
}
```

Simplemente se almacena el orden de salida en un vector donde posteriormente se incrementará el valor de la posición para mostrar en orden los leds, eliminando el led que estaba anteriormente encendido.

Finalmente se pide diseñar un sistema con una lámpara, un pulsador y un potenciómetro. donde la lámpara tiene 3 posibles valores de iluminación (0, apagado; 1, luz suave; 2, luz intensa). Seleccionaremos mediante el potenciómetro el valor de iluminación y mientras se mantenga presionado el pulsador, la luz de la lámpara se iluminará con el valor indicado en el potenciómetro.

El circuito realizado es el siguiente:



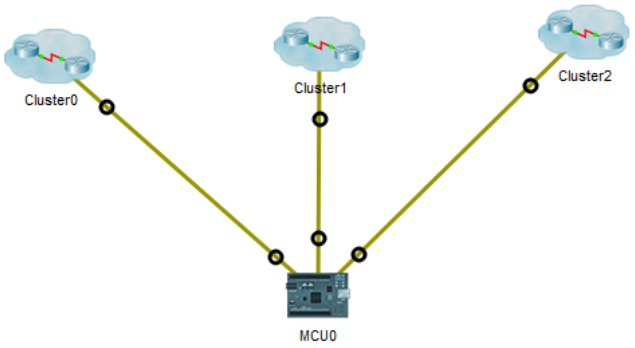
El código creado para que funcione según lo pedido es el siguiente:

```
var lampara, potenciometro_ant;

function setup() {//los analogicos a0 (potenciometro) y al son entradas
    pinMode(0, INPUT);//interruptor
    pinMode(1, OUTPUT);//pulsador

    lampara = 0;
    potenciometro_ant = 0;
    customWrite(1, 0);

}

function loop() {
    var pulsador, potenciometro;
    pulsador = digitalRead(0);
    potenciometro = analogRead(A0);

    //codigo para el lampara
    if ((pulsador == HIGH)&&(potenciometro != potenciometro_ant)){
        lampara = 1;
        potenciometro_ant = potenciometro;
        if (potenciometro === 0){
            customWrite(1, 0);
            Serial.println("LAMPARA APAGADA");
        }else if(potenciometro === 1023){
            customWrite(1, 2);
            Serial.println("LAMPARA FULL POTENCIA");

        }else{
            customWrite(1, 1);
            Serial.println("LAMPARA MEDIA POTENCIA");
            Serial.println(potenciometro);
        }
    }else if ((pulsador == LOW)&&(lampara == 1)){
        lampara = 0;
        potenciometro_ant = 0;
        customWrite(1, 0);
        Serial.println("LAMPARA APAGADO");
    }
}
```

Es un código donde hemos tenido que añadir una pequeña lógica para, dependiendo del potenciómetro y si está activado el pulsador, la lámpara se ilumine, se ilumine un poco o esté a máxima potencia.

3. Práctica 3: Redundancia Triple Modular (TMR)

Primeramente se va a crear el circuito pedido con las conexiones necesarias en los pines necesarios, el circuito sería el siguiente:



En cada cluster hay una ciudad donde hemos ajustado unos parámetros con las temperaturas de cada una.

El código realizado para poder realizar este ejercicio es el siguiente:

```
function setup() {//los sensores de temperatura de cada ciudad
    pinMode(0, INPUT);//temp0
    pinMode(1, INPUT);//temp1
    pinMode(2, INPUT);//temp2
}
function conversor(dato){//convierto de 0 a 1023 a -100 100
    dato=((dato - 0)/(1023 - 0))*(100+100)-100;//formula para normalizar
    return dato;
}

function loop() {
    var temp0, temp1, temp2;
    temp0 = conversor(digitalRead(0));
    temp1 = conversor(digitalRead(1));
    temp2 = conversor(digitalRead(2));
    Serial.println(digitalRead(0));
    Serial.println(digitalRead(1));
    Serial.println(digitalRead(2));
    Serial.println(temp0);
    Serial.println(temp1);
    Serial.println(temp2);
    //codigo del tmr
    if (temp0 == temp1 || temp0 == temp2 || temp1 == temp2){
        if(temp0 == temp1){
            Serial.print("1. El resultado es:");
            Serial.println(temp0);
        }else if (temp0 == temp2){
            Serial.print("2. El resultado es:");
            Serial.println(temp0);
        }else{
            Serial.print("3. El resultado es:");
            Serial.println(temp2);
        }
    }else{
        Serial.println("No hay solucion");
    }
}
```

El código en resumen, se basa en coger los datos de 0 a 1023 de cada ciudad, los normaliza entre los valores de -100 y 100 grados celsius y se comparan entre sí, de tal forma que con que dos o más sean iguales dan solución. En el caso contrario no encontraría solución. Algunos ejemplos de soluciones dadas son las siguientes:

Si dos son iguales:

```
594
566
594
16.129032258064527
10.654936461388061
16.129032258064527
2. El resultado es:16.129032258064527
```

O también:

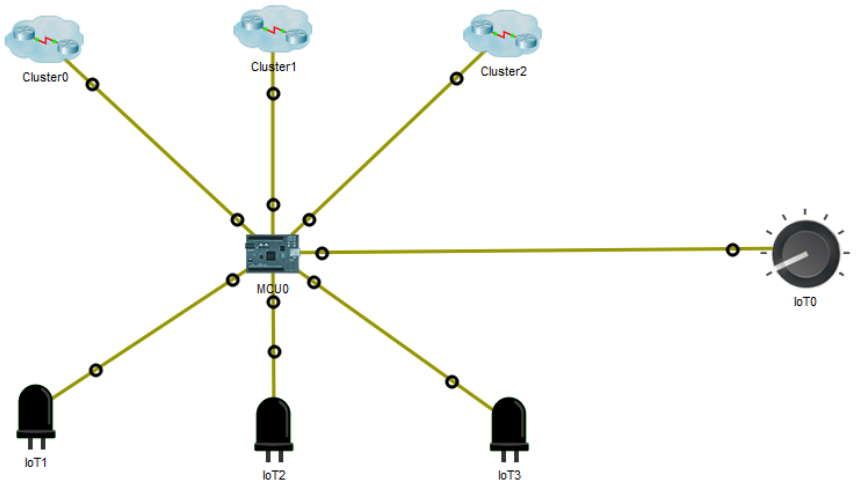
```
534
526
526
4.398826979472133
2.8347996089931513
2.8347996089931513
3. El resultado es:2.8347996089931513
```

Si todos son diferentes:

```
582
574
578
13.782991202346054
12.218963831867043
13.000977517106563
No hay solucion
```

También puede haber casos donde los tres coinciden, pero es complicado que ocurra.

En el apartado b, se nos pide que apliquemos una máscara sobre los bits de entrada de los sensores, se va a añadir en el circuito un potenciómetro que permite enmascarar de forma dinámica entre 0 y 8 bits ya que para 1023 se necesitan 10 bits. Se va a complementar en con el apartado c donde tenemos que añadir un identificador para ver los sensores que dan mal el resultado. El circuito es el siguiente:



El código realizado para solucionar este problema es el siguiente:

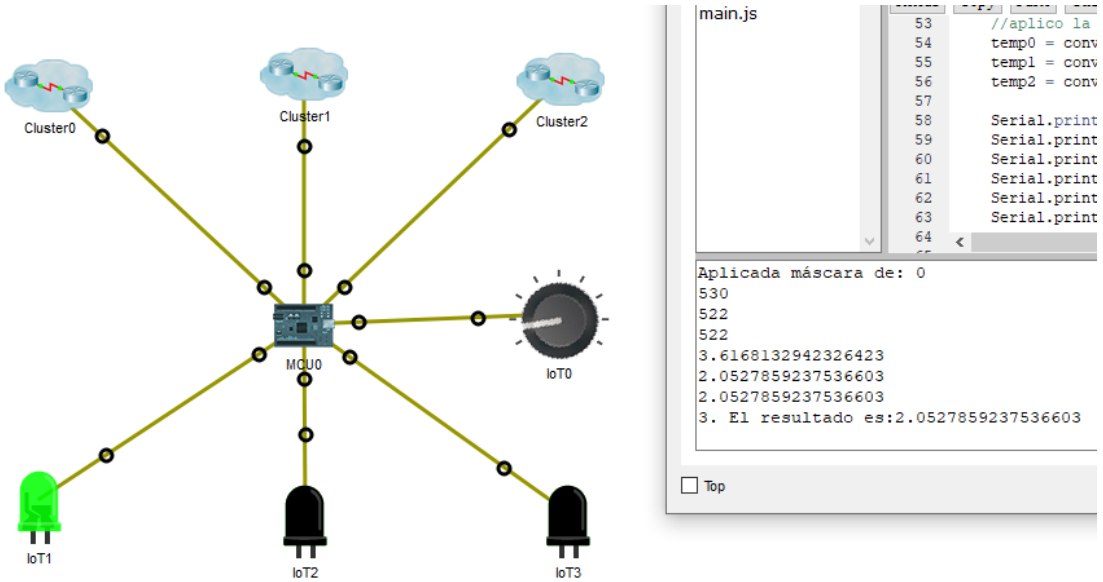
```
var v_led = [3,4,5];
var potenciometro_ant, bits_mascara;

function setup() {//los sensores de temperatura de cada ciudad
    pinMode(0, INPUT);//temp0
    pinMode(1, INPUT);//temp1
    pinMode(2, INPUT);//temp2
    pinMode(3, INPUT);//led1
    pinMode(4, INPUT);//led2
    pinMode(5, INPUT);//led3

    for(var i = 0; i< 3; i++){//limpiamos los led
        analogWrite(v_led[i], 0);
    }
    potenciometro_ant = 0;
    bits_mascara = 0;
}
function conversor(dato){//convierto de 0 a 1023 a -100 100
    dato=((dato - 0)/(1023 - 0))*(100+100)-100;//formula para normalizar
    return dato;
}
```

```
function loop() {
    var temp0, temp1, temp2, potenciometro;

    potenciometro = analogRead(A0);

    //pongo la mascara dependiendo del valor del potenciometro
    if (potenciometro != potenciometro_ant){
        potenciometro_ant = potenciometro;

        if(potenciometro < 127){
            bits_mascara = 0;
        }else if (potenciometro > 126 && potenciometro < 254){
            bits_mascara = 1;
        }else if (potenciometro > 253 && potenciometro < 381){
            bits_mascara = 2;
        }else if (potenciometro > 126 && potenciometro < 508){
            bits_mascara = 3;
        }else if (potenciometro > 126 && potenciometro < 635){
            bits_mascara = 4;
        }else if (potenciometro > 126 && potenciometro < 762){
            bits_mascara = 5;
        }else if (potenciometro > 126 && potenciometro < 889){
            bits_mascara = 6;
        }else if (potenciometro > 126 && potenciometro < 1016){
            bits_mascara = 7;
        }else{
            bits_mascara = 8;
        }
    }

    //aplico la mascara antes de transformar el dato
    temp0 = conversor(digitalRead(0) & (~(Math.pow(2,bits_mascara)-1)));
    temp1 = conversor(digitalRead(1) & (~(Math.pow(2,bits_mascara)-1)));
    temp2 = conversor(digitalRead(2) & (~(Math.pow(2,bits_mascara)-1)));

    Serial.print("Aplicada máscara de: ");
    Serial.println(bits_mascara);
    Serial.println(digitalRead(0) & (~(Math.pow(2,bits_mascara)-1)));
    Serial.println(digitalRead(1) & (~(Math.pow(2,bits_mascara)-1)));
    Serial.println(digitalRead(2) & (~(Math.pow(2,bits_mascara)-1)));
    Serial.println(temp0);
    Serial.println(temp1);
    Serial.println(temp2);
```

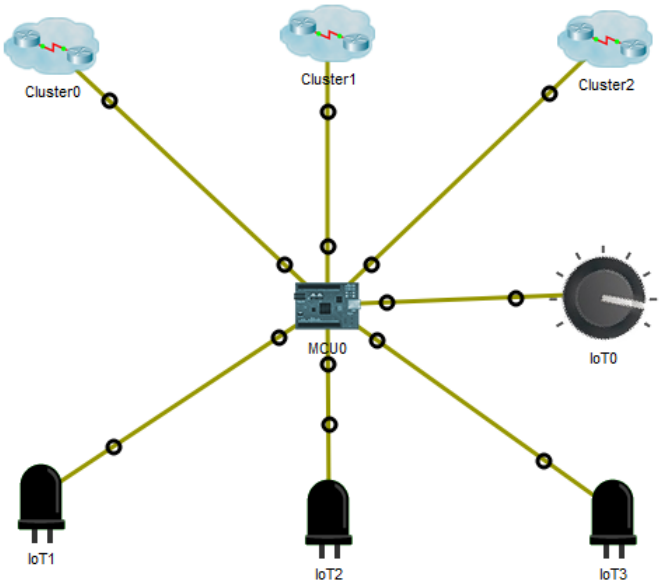
```
67      //codigo del tmr
68      if (temp0 == temp1 || temp0 == temp2 || temp1 == temp2){
69          if(temp0 == temp1){
70              Serial.print("1. El resultado es:");
71              Serial.println(temp0);
72              if(temp0 != temp2){
73                  for(var i = 0; i< 3; i++){//limpiamos los led
74                  analogWrite(v_led[i], 0);
75                  }
76                  analogWrite(v_led[2], 1023);
77              }
78          }else if (temp0 == temp2){
79              Serial.print("2. El resultado es:");
80              Serial.println(temp0);
81              for(var z = 0; z< 3; z++){//limpiamos los led
82                  analogWrite(v_led[z], 0);
83              }
84              analogWrite(v_led[1], 1023);
85          }else{
86              Serial.print("3. El resultado es:");
87              Serial.println(temp2);
88              for(var w = 0; w< 3; w++){//limpiamos los led
89                  analogWrite(v_led[w], 0);
90              }
91              analogWrite(v_led[0], 1023);
92          }
93      }else{
94          Serial.println("No hay solucion");
95          for(var x = 0; x< 3; x++){//limpiamos los led
96              analogWrite(v_led[x], 1023);
97          }
98      }
99  }
```

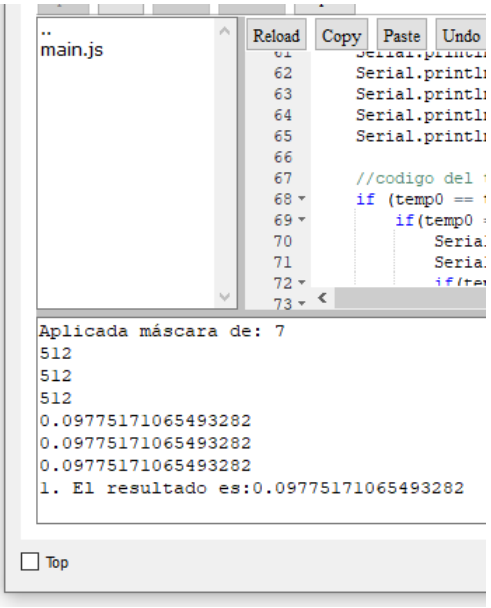
En este código se ha añadido la funcionalidad de las led, para ver que sensor falla y el código necesario con la lógica para transformar el valor del potenciómetro en los valores de 0 a 8 siendo la máscara a utilizar.

Un ejemplo sin usar máscara sería el siguiente:



Usando máscara sería el siguiente:

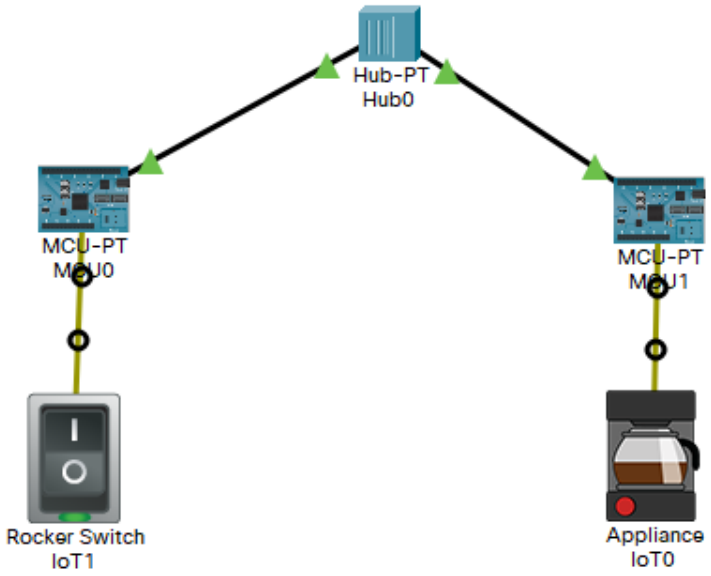




Al usar máscara es mucho más fácil que den el mismo resultado ya que hay menos bits a comparar.

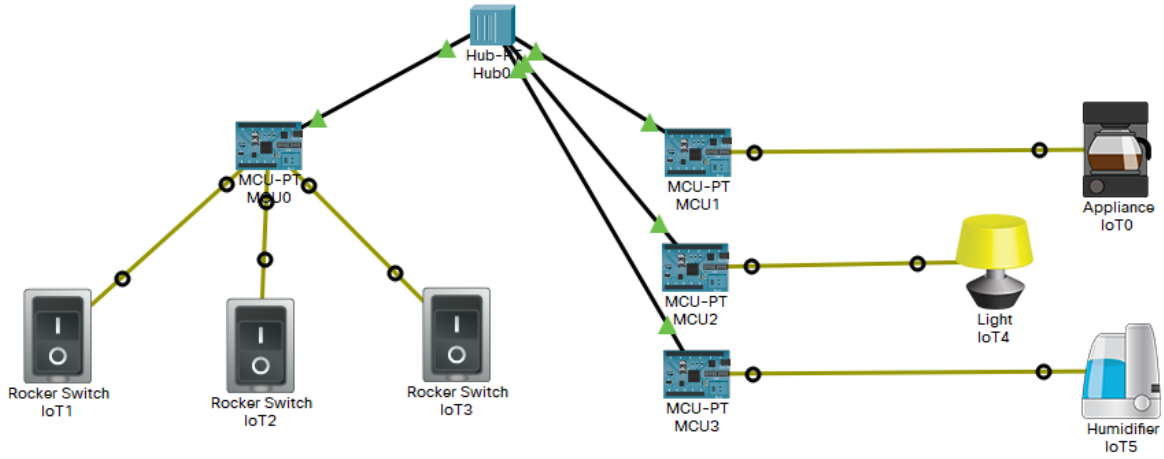
4. Práctica 4: Interconectando dos MCU

Primeramente en esta práctica se crea el circuito con el código proporcionado para realizar una comunicación sencilla UDP básica:



Cuando el interruptor se enciende, se enciende la cafetera.

En el siguiente apartado, se pide crear una comunicación UDP con múltiples destinos, el circuito creado es el siguiente:



Donde se ha creado un interruptor para cada sensor. El código del MCU de los interruptores es el siguiente:

```
var port = 1234;
var dstIPCAFETERA = "192.168.1.1";
var dstIPLAMPARA = "192.168.1.3";
var dstIPHUMIDIFICADOR = "192.168.1.4";

var socket;
var stateCAFETERA;
var stateLAMPARA;
var stateHUMIDIFICADOR;

function setup() {
 pinMode(0,INPUT);
 pinMode(1,INPUT);
 pinMode(2,INPUT);
 stateCAFETERA = 0;
 stateLAMPARA = 0;
 stateHUMIDIFICADOR = 0;

 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
//CAFETERA
 if (digitalRead(0)) {
  if (stateCAFETERA === 0) {
   stateCAFETERA = 1;
   socket.send(dstIPCAFETERA, port, "1");
   Serial.println("ON CAFETERA");
  }
 }
 else{
  if (stateCAFETERA === 1) {
   stateCAFETERA = 0;
   socket.send(dstIPCAFETERA, port, "0");
   Serial.println("OFF CAFETERA");
  }
 }
//LAMPARA
if (digitalRead(1)) {
 if (stateLAMPARA === 0) {
  stateLAMPARA = 1;
  socket.send(dstIPLAMPARA, port, "2");
  Serial.println("ON LAMPARA");
 }
}
else{
 if (stateLAMPARA === 1) {
  stateLAMPARA = 0;
  socket.send(dstIPLAMPARA, port, "0");
  Serial.println("OFF LAMPARA");
 }
}
//HUMIDIFICADOR
if (digitalRead(2)) {
 if (stateHUMIDIFICADOR === 0) {
  stateHUMIDIFICADOR = 1;
  socket.send(dstIPHUMIDIFICADOR, port, "1");
  Serial.println("ON HUMIDIFICADOR");
 }
}
else{
 if (stateHUMIDIFICADOR === 1) {
  stateHUMIDIFICADOR = 0;
  socket.send(dstIPHUMIDIFICADOR, port, "0");
  Serial.println("OFF HUMIDIFICADOR");
 }
}
delay(1000);
}
```

En este código, dependiendo del interruptor pulsado, enviará los datos hacia ip correspondiente.

El código para un MCU de recepción es el siguiente:

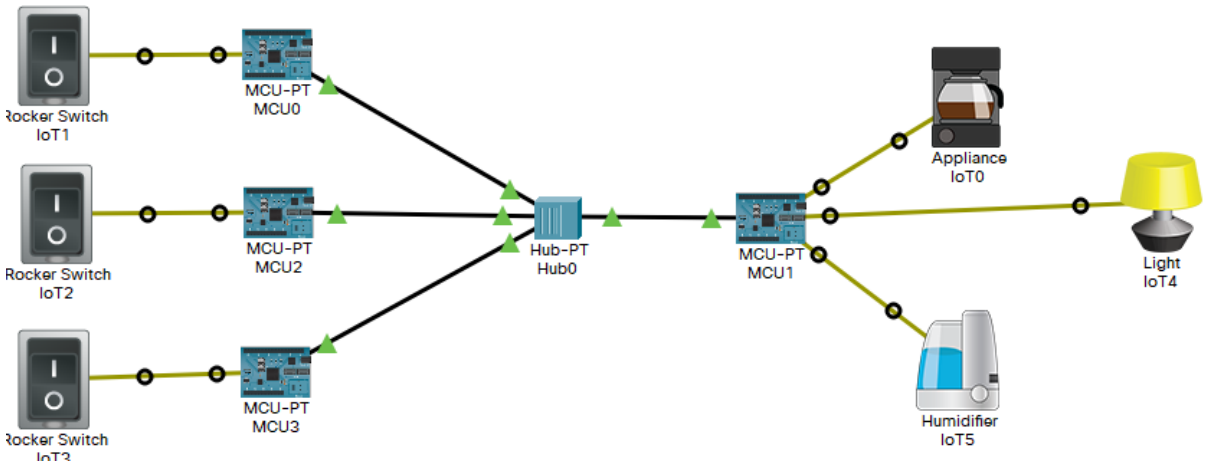
```
var port = 1234;
var dstIP = "192.168.1.2";
var socket;

function setup() {
 socket = new UDPSocket();
 customWrite(0,"0");

 // Recepcion
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   if(data=="1"){
    Serial.println("CAFETERA ENCENDIDA");
    customWrite(0,"1");
   }
   else {
    Serial.println("CAFETERA APAGADA");
    customWrite(0,"0");
   }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
 // Nada
}
```

Este código se repite 2 veces más con ligeras modificaciones (en el caso de la lámpara escribir un 2 en vez de un 1).

Para el último apartado, se pedía realizar la comunicación UDP con múltiples fuentes a un mismo destino. Para ello, se pedía que se realizará de dos formas diferentes:

- Diferentes sockets asociados a diferentes puertos en el destino.
- Protocolo de selección incorporado en el propio flujo de datos.

Para ambos casos el circuito es el siguiente:



Para el primer caso el código es el siguiente:
Para los MCU asociados a un interruptor:

```
var port = 1111;
var dstIP = "192.168.1.1";

var socket;
var stateCAFETERA;

function setup() {
 pinMode(0,INPUT);
 stateCAFETERA = 0;

 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
//CAFETERA
 if (digitalRead(0)) {
  if (stateCAFETERA === 0) {
   stateCAFETERA = 1;
   socket.send(dstIP, port, "1");
   Serial.println("ON CAFETERA");
  }
 }
 else{
  if (stateCAFETERA === 1) {
   stateCAFETERA = 0;
   socket.send(dstIP, port, "0");
   Serial.println("OFF CAFETERA");
  }
 }
 delay(1000);
}
```

Para el MCU que está asociado a los aparatos:

```
var portCAFETERA = 1111;
var portLAMPARA = 2222;
var portHUMIDIFICADOR = 3333;
var port;
var socketCAFETERA, socketLAMPARA, socketHUMIDIFICADOR;

function setup() {
 socketCAFETERA = new UDPSocket();
 socketLAMPARA = new UDPSocket();
 socketHUMIDIFICADOR = new UDPSocket();
 customWrite(0,"0");
 customWrite(1,"0");
 customWrite(2,"0");

 //CAFETERA
 socketCAFETERA.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   //compruebo si es cafetera
   if(port == 1111){
    if(data=="1"){
     Serial.println("CAFETERA ENCENDIDA");
     customWrite(0,"1");
    }
    else {
     Serial.println("CAFETERA APAGADA");
     customWrite(0,"0");
    }
   }
 };
```

```
//LAMPARA
socketLAMPARA.onReceive = function(ip, port, data) {
 Serial.println("Recibido de "
  + ip + ":" + port + ": " + data);
  //compruebo si es LAMPARA
  if(port == 2222){//compruebo si es lampara
   if(data=="2"){
    Serial.println("LAMPARA ENCENDIDA");
    customWrite(1,"2");
   }
   else {
    Serial.println("LAMPARA APAGADA");
    customWrite(1,"0");
   }
  }
};

//HUMIDIFICADOR
socketHUMIDIFICADOR.onReceive = function(ip, port, data) {
 Serial.println("Recibido de "
  + ip + ":" + port + ": " + data);
  //compruebo si es HUMIDIFICADOR
 if(port == 3333){//compruebo si es humidificador
  if(data=="1"){
   Serial.println("HUMIDIFICADOR ENCENDIDA");
   customWrite(2,"1");
  }
  else {
   Serial.println("HUMIDIFICADOR APAGADA");
   customWrite(2,"0");
  }
 }
};

// Activa el socket UDP en el puerto
Serial.println(socketCAFETERA.begin(portCAFETERA));
Serial.println(socketLAMPARA.begin(portLAMPARA));
Serial.println(socketHUMIDIFICADOR.begin(portHUMIDIFICADOR));
}

function loop() {
 // Nada
}
```

Asignamos un puerto para cada socket, de tal forma que cada electrodoméstico tiene su propio puerto.

Para el segundo caso el código es el siguiente:
Para los MCU asociados a un interruptor:

```
var port = 1234;
var dstIP = "192.168.1.1";

var socket;
var stateCAFETERA;

function setup() {
 pinMode(0,INPUT);
 stateCAFETERA = 0;
 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
//CAFETERA
 if (digitalRead(0)) {
  if (stateCAFETERA === 0) {
   stateCAFETERA = 1;
   socket.send(dstIP, port, "C,1");
   Serial.println("ON CAFETERA");
  }
 }
 else{
  if (stateCAFETERA === 1) {
   stateCAFETERA = 0;
   socket.send(dstIP, port, "C,0");
   Serial.println("OFF CAFETERA");
  }
 }
 delay(1000);
}
```

En este caso, aparte de el dato, le enviamos una letra para que sepa a que aparato pertenece ese dato.

Para el MCU que está asociado a los aparatos:

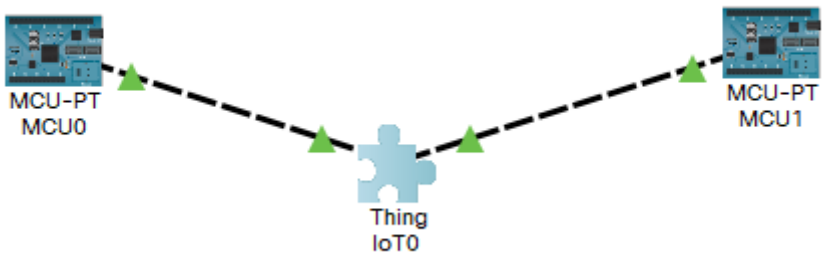
```
var port = 1234;
var socket;

function setup() {
 var coma = ",";
 socket = new UDPSocket();
 customWrite(0,"0");
 customWrite(1,"0");
 customWrite(2,"0");

```

```
socket.onReceive = function(ip, port, data) {
 Serial.println("Recibido de "
  + ip + ":" + port + ": " + data);
  //compruebo si es cafetera
  if(port == 1234){
   var arrayCadenas = data.split(coma);

   if(arrayCadenas[0]== "C"){//cafetera
    if(arrayCadenas[1]== "1"){
     Serial.println("CAFETERA ENCENDIDA");
     customWrite(0,"1");
    }
    else {
     Serial.println("CAFETERA APAGADA");
     customWrite(0,"0");
    }

   }else if(arrayCadenas[0]== "L"){//lampara
    if(arrayCadenas[1]== "2"){
     Serial.println("LAMPARA ENCENDIDA");
     customWrite(1,"2");
    }
    else {
     Serial.println("LAMPARA APAGADA");
     customWrite(1,"0");
    }
   }else if(arrayCadenas[0]== "H"){//humidificador
    if(arrayCadenas[1]== "1"){
     Serial.println("HUMIDIFICADOR ENCENDIDO");
     customWrite(2,"1");
    }
    else {
     Serial.println("HUMIDIFICADOR APAGADO");
     customWrite(2,"0");
    }
   }
  }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

```

En este caso, hay un solo socket y puerto, se diferencia los datos de todas los aparatos con el código que lleva el propio dato, siendo H de humidificador, L de lámpara o C de cafetera.

5. Práctica 5: Tolerancia a fallos en comunicación con datos

En esta práctica, primeramente se nos pide usar tres MCU de tal forma que el MCU central sirva de comunicación entre ambos MCU (Receptor y emisor). Por tanto el circuito sería el siguiente:



El MCU emisor se encarga de enviar el dato al intermediario y recibe el dato del intermediario siendo la respuesta de si el dato ha llegado correctamente al receptor. El código es el siguiente:

```
var port = 1111;
var dstIP = "192.168.1.2";
var datoant = "valorbasura0132";
var socket;

function setup() {
 var coma = ",";
 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
  if(port == 1111){
   var arrayCadenas = data.split(coma);
   if(arrayCadenas[1]=="1"){
    Serial.println("ENVIADO CORRECTAMENTE");
   }
   else {
    Serial.println("ERROR EN EL ENVIO, NECESARIO REENVIO");
   }
  }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
 //EMISOR

 var dato = "HOLA"
 if(dato != datoant){
  socket.send(dstIP, port, dato);
  Serial.println("ENVIO EL DATO->");
  Serial.println(dato);
 }
 datoant = dato;
 delay(1000);
}
```

El MCU que hace de intermediario es el más complejo, ya que recibe el dato del emisor, lo procesa (genera un error de forma aleatoria), y lo envía al receptor. Posteriormente, recibe el mensaje del receptor de si le ha llegado el dato bien y lo envía al emisor. El código es el siguiente:

```
var portEMISOR = 1111;
var portRECEPTOR = 2222;
var socketEMISOR, socketRECEPTOR;
var dstIPEMISOR = "192.168.1.1";
var dstIPRECEPTOR = "192.168.2.1";
var temp, aleatorio;

function setup() {
 var coma = ",";
 socketEMISOR = new UDPSocket();
 socketRECEPTOR = new UDPSocket();

 //EMISOR
 socketEMISOR.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   //compruebo si es EMISOR
   if(port == 1111){
     Serial.println("RECIBO EL DATO ->");
     Serial.println(data);
     aleatorio = Math.random();
     Serial.println(aleatorio);
     if(aleatorio < 0.3){
      temp = data + ",0";
     }else{
      temp = data + ",1";
     }
     socketRECEPTOR.send(dstIPRECEPTOR, portRECEPTOR, temp); //envio el dato al receptor
   }
 };

 //RECEPTOR
 socketRECEPTOR.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   //compruebo si es RECEPTOR
   if(port == 2222){
     var arrayCadenas = data.split(coma);
     if(arrayCadenas[1]=="1"){
      Serial.println("RECIBIDO CORRECTAMENTE EL MENSAJE ->");
      Serial.println(arrayCadenas[0]);
      socketEMISOR.send(dstIPEMISOR, portEMISOR, data); //envio sin error
     }
    else {
     Serial.println("ERROR EN EL RECIBO, PIDO REENVIO");
     socketEMISOR.send(dstIPEMISOR, portEMISOR, data); //envio con error
    }
   }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socketEMISOR.begin(portEMISOR));
 Serial.println(socketRECEPTOR.begin(portRECEPTOR));
}

function loop() {
 // Nada
}
```

Finalmente, el módulo receptor recibe el mensaje, y dependiendo de si es erróneo o no, pide reenvío o es correcto. El código es el siguiente:

```
var port = 2222;
var dstIP = "192.168.2.2";
var datoant = "valorbasura0132";
var socket;

function setup() {
 var coma = ",";
 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   if(port == 2222){
    var arrayCadenas = data.split(coma);
    if(arrayCadenas[1]=="1"){
     Serial.println("RECIBIDO CORRECTAMENTE EL MENSAJE ->");
     Serial.println(arrayCadenas[0]);
     socket.send(dstIP, port, data); //envio sin error
    }
    else {
     Serial.println("ERROR EN EL RECIBO, PIDO REENVIO");
     socket.send(dstIP, port, data); //envio con error
    }
   }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
 //NADA
}
```

En el siguiente apartado se pide que usando la misma metodología, añadamos control de codificación. Para ello el emisor debe añadir la codificación al dato y el receptor decodificarlo. El receptor debe avisar si es correcto, y en el caso de que no sea así, solicitar el dato de nuevo. Para este caso, el MCU intermediario se limita a reenviar datos, no interfiere.

El MCU va a tener dos datos originales, y genera un dato erróneo, con las características mencionadas en la práctica. Se aplica paridad y checksum. El código realizado es el siguiente:

```
var port = 1111;
var dstIP = "192.168.1.2";
var dato_original1 = '10010110';
var dato_original2 = '00110000';
var socket;
var stop = 0;

//funciones que no son mias para calcular la suma de dos numeros en binario
//https://www.it-swarm-es.com/es/javascript/javascript-agregue-dos-numeros-binarios-devuelve-binario/827551236/
var addBinary = function(a, b) {
    var i = a.length - 1;
    var j = b.length - 1;
    var carry = 0;
    var result = "";
    while(i >= 0 || j >= 0) {
        var m = i < 0 ? 0 : a[i] | 0;
        var n = j < 0 ? 0 : b[j] | 0;
        carry += m + n; // sum of two digits
        result = carry % 2 + result; // string concat
        carry = carry / 2 | 0; // remove decimals,  1 / 2 = 0.5, only get 0
        i--;
        j--;
    }
    if(carry !== 0) {
        result = carry + result;
    }
    return result;
};
```

```
function setup() {
 var exito = 0;
 socket = new UDPSocket();

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
  if(port == 1111){
   if(data == "00"){
       Serial.println("HA FALLADO LA PARIDAD, SE NECESITA REENVIO");
       Serial.println("-------------------------------------------");
   }else if (data == "01"){
       Serial.println("NO HA FALLADO LA PARIDAD");
       Serial.println("-------------------------------------------");
       exito++;
   }else if (data == "10"){
       Serial.println("HA FALLADO EL CHECKSUM, SE NECESITA REENVIO");
       Serial.println("-------------------------------------------");
   }else{
       Serial.println("NO HA FALLADO EL CHECKSUM");
       Serial.println("-------------------------------------------");
       exito++;
   }
  }
 };
 if(exito == 2){
    Serial.println("DATO ENVIADO CORRECTAMENTE");
 }

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}
```

```
function loop() {
 //EMISOR

 if(stop === 0){
  var dato = dato_original1;
  var contador;
  var aux;
  var datoaux;
  //modificamos el dato cambiando un bit e intercambiando dos bits entre sí (solo el original 1)
  dato = '11011010';
  datoaux = dato_original1 + dato_original2;

  //añadimos el protocolo de control
  //paridad
  contador = 0;
  for(var i = 0; i<datoaux.length ; i++){
    if(datoaux[i] == 1){
        contador++;
    }
  }
  if (contador%2 === 0){//comprobamos si tiene paridad par
    aux = '1';
  }else{
    aux = '0';
  }
  dato = dato + dato_original2 +aux;//creo el dato con el error y el código

  socket.send(dstIP, port, dato);
  Serial.println("-------------------------------------------");
  Serial.println("ENVIO EL DATO->");
  Serial.println(dato);
  Serial.println("TIENE UN TAMAÑO DE->");
  Serial.println(dato.length);

  Serial.println("-------------------------------------------");

  //checksum de simple precisión sumamos los dos datos
  aux = addBinary(dato_original1, dato_original2);

  dato = '11011010';//dato con el error
  dato = dato + dato_original2 + aux; //creo el dato con el error y el código

  socket.send(dstIP, port, dato);
  Serial.println("ENVIO EL DATO->");
  Serial.println(dato);
  Serial.println("TIENE UN TAMAÑO DE->");
  Serial.println(dato.length);

  Serial.println("-------------------------------------------");

  stop = 1;
 }


 delay(1000);
}
```

Este código tiene dos partes principales, una es creación de código a enviar y recepción de datos de comprobación.

El MCU destino realiza los mismos cálculos para decodificar el código recibido y comprobar si los datos son correctos. El código es el siguiente:

```
var port = 2222;
var dstIP = "192.168.2.2";
var socket;

//funciones que no son mias para calcular la suma de dos numeros en binario
//https://www.it-swarm-es.com/es/javascript/javascript-agregue-dos-numeros-binarios-devuelve-binario/827551236/
var addBinary = function(a, b) {
    var i = a.length - 1;
    var j = b.length - 1;
    var carry = 0;
    var result = "";
    while(i >= 0 || j >= 0) {
        var m = i < 0 ? 0 : a[i] | 0;
        var n = j < 0 ? 0 : b[j] | 0;
        carry += m + n; // sum of two digits
        result = carry % 2 + result; // string concat
        carry = carry / 2 | 0; // remove decimals,  1 / 2 = 0.5, only get 0
        i--;
        j--;
    }
    if(carry !== 0) {
        result = carry + result;
    }
    return result;
};

function setup() {
 socket = new UDPSocket();
 var dato1, dato2, codigo, aux, contador, datoaux;

 // Recepcion UDP
 socket.onReceive = function(ip, port, data) {
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   if(port == 2222){

    if (data.length == 17){
        dato1 = data.slice(0, 7);
        dato2 = data.slice(8, 15);
        codigo = data [16];

        //comprobamos paridad
        //paridad
        datoaux = dato1 + dato2;
        contador = 0;
        for(var i = 0; i<datoaux.length ; i++){
            if(datoaux[i] == 1){
                contador++;
            }
        }
        if (contador%2 === 0){//comprobamos si tiene paridad par
            aux = '1';
        }else{
            aux = '0';
        }
        if(codigo == aux){
            socket.send(dstIP, port, "01"); //envio sin error
        }else{
            socket.send(dstIP, port, "00"); //envio con error
        }
    }else if(data.length == 24){
        dato1 = data.slice(0, 7);
        dato2 = data.slice(8, 15);
        codigo = data.slice(6, 23);

        //checksum de simple precisión sumamos los dos datos
        aux = addBinary(dato1, dato2);
        if(codigo == aux){
            socket.send(dstIP, port, "11"); //envio sin error
        }else{
            socket.send(dstIP, port, "10"); //envio con error
        }
    }
   }
 };

 // Activa el socket UDP en el puerto
 Serial.println(socket.begin(port));
}

function loop() {
 //NADA
}
```

Para comprobar que todo se encuentra correctamente podemos ver que para los datos:

- 10010110
- 00110000

Y transformando el dato un a 11011010. Obtenemos la siguiente salida:

```
-------------------------------------------
ENVIO EL DATO->
11011010001100001
TIENE UN TAMAÑO DE->
17
-------------------------------------------
Recibido de 192.168.1.2:1111: 00
HA FALLADO LA PARIDAD, SE NECESITA REENVIO
-------------------------------------------
ENVIO EL DATO->
110110100011000011000110
TIENE UN TAMAÑO DE->
24
-------------------------------------------
Recibido de 192.168.1.2:1111: 10
HA FALLADO EL CHECKSUM, SE NECESITA REENVIO
-------------------------------------------
```

Por tanto, podemos decir que tanto paridad como checksum detectan que hay un error mínimo. Pero si cambiaramos el dato 2 por 00110001 la salida sería la siguiente:

```
-------------------------------------------
ENVIO EL DATO->
11011010001100010
TIENE UN TAMAÑO DE->
17
-------------------------------------------
Recibido de 192.168.1.2:1111: 01
NO HA FALLADO LA PARIDAD
-------------------------------------------
ENVIO EL DATO->
110110100011000111000111
TIENE UN TAMAÑO DE->
24
-------------------------------------------
Recibido de 192.168.1.2:1111: 10
HA FALLADO EL CHECKSUM, SE NECESITA REENVIO
-------------------------------------------
```

Como podemos ver la paridad no puede detectar ninguno de los dos errores, esto se debe a que hemos cambiado dos bits aleatoriamente de 0 a 1, entonces ese fallo para paridad es indetectable y el otro fallo consiste en intercambiar dos bits de lugar, por tanto no influye en el número de bits par o impar, si no en la información. Por ello, podemos decir que el checksum es más fiable. Si el fallo que hubiéramos impuesto es poner un 1 o un cero en bits consecutivos, si se hubiese detectado el cambio, a no ser que se realizará junto con un cambio de bit aleatorio.

Finalmente, se pide que el MCU intermedio inyecte errores de manera aleatoria. Para este caso, no se inyectarán los errores en el MCU de emisión.

El código del MCU receptor y emisor es similar, por consiguiente, se va a incidir más en el código del MCU intermediario. El código es el siguiente:

```
var portEMISOR = 1111;
var portRECEPTOR = 2222;
var socketEMISOR, socketRECEPTOR;
var dstIPEMISOR = "192.168.1.1";
var dstIPRECEPTOR = "192.168.2.1";



function setup() {
 socketEMISOR = new UDPSocket();
 socketRECEPTOR = new UDPSocket();
 var aleatorio, aleatorioaux;
 var dataaux = '';

 //EMISOR
 socketEMISOR.onReceive = function(ip, port, data) {
  Serial.println("----------------------------------------");
  Serial.println("Recibido de "
   + ip + ":" + port + ": " + data);
   //compruebo si es EMISOR
   if(port == 1111){

    //inyecto errores
    aleatorio = Math.random();
    if(aleatorio < 0.5){//50% de inyectar error en alguno de los dos casos
      Serial.println("INYECTO ERRORES");
      Serial.println("DATOS SIN ERRORES->");
      Serial.println(data);
      dataaux = '';
      if (data.length == 17){
        //inyecto error tipo 1 modifico un bit aleatorio
        aleatorio = Math.floor(Math.random() * 17);
        Serial.print("SE MODIFICA EL BIT->");
        Serial.println(aleatorio);
        for(var x = 0; x < data.length ; x++){//copio el dato con la modificacion
         if(x == aleatorio){
          if(data[x] === '0'){
           dataaux = dataaux + '1';
          }else{
           dataaux = dataaux + '0';
          }
         }else{
          dataaux = dataaux + data[x];
         }
        }
        data = dataaux;
        dataaux = '';

        //inyecto error de tipo 2, intercambio valores
        aleatorio = Math.floor(Math.random() * 17);
        aleatorioaux = Math.floor(Math.random() * 17);
        Serial.print("SE INTERCAMBIA EL BIT->");
        Serial.print(aleatorio);
        Serial.print(" CON EL BIT->");
        Serial.println(aleatorioaux);
        for(var w = 0; w < data.length ; w++){//copio el dato con la modificacion
         if(w == aleatorio){
          dataaux = dataaux + data[aleatorioaux];
         }else if(w == aleatorioaux){
          dataaux = dataaux + data[aleatorio];
         }else{
          dataaux = dataaux + data[w];
         }
        }
        data = dataaux;
        Serial.println("DATOS CON ERRORES->");
        Serial.println(data);
```

```
      }else if(data.length == 24){
        //inyecto error tipo 1 modifico un bit aleatorio
        aleatorio = Math.floor(Math.random() * 17);
        Serial.print("SE MODIFICA EL BIT->");
        Serial.println(aleatorio);
        dataaux = '';
        for(var y = 0; y < data.length ; y++){//copio el dato con la modificacion
         if(y == aleatorio){
          if(data[y] === '0'){
           dataaux = dataaux + '1';
          }else{
           dataaux = dataaux + '0';
          }
         }else{
          dataaux = dataaux + data[y];
         }
        }
        data = dataaux;
        dataaux = '';
        //inyecto error de tipo 2, intercambio valores
        aleatorio = Math.floor(Math.random() * 17);
        aleatorioaux = Math.floor(Math.random() * 17);
        Serial.print("SE INTERCAMBIA EL BIT->");
        Serial.print(aleatorio);
        Serial.print(" CON EL BIT->");
        Serial.println(aleatorioaux);
        for(var z = 0; z < data.length ; z++){//copio el dato con la modificacion
         if(z == aleatorio){
          dataaux = dataaux + data[aleatorioaux];
         }else if(z == aleatorioaux){
          dataaux = dataaux + data[aleatorio];
         }else{
          dataaux = dataaux + data[z];
         }
        }
        data = dataaux;
        Serial.println("DATOS CON ERRORES->");
        Serial.println(data);
      }
     }
     socketRECEPTOR.send(dstIPRECEPTOR, portRECEPTOR, data); //envio el dato al receptor
    }
  };

  //RECEPTOR
  socketRECEPTOR.onReceive = function(ip, port, data) {
   Serial.println("----------------------------------------");
   Serial.println("Recibido de "
    + ip + ":" + port + ": " + data);
    //compruebo si es RECEPTOR
    if(port == 2222){
     socketEMISOR.send(dstIPEMISOR, portEMISOR, data); //envio el dato al emisor
    }
  };

  // Activa el socket UDP en el puerto
  Serial.println(socketEMISOR.begin(portEMISOR));
  Serial.println(socketRECEPTOR.begin(portRECEPTOR));
}

function loop() {
 // Nada
}
```

Para entender mejor todo este código, voy a explicar que se ha realizado. Primeramente comprobamos si va a haber fallo o no, esto se calcula de manera aleatoria con una probabilidad del 50%. En el caso de no fallar, actúa con normalidad. En el caso de que si se inyecte un fallo, se comprueba el tamaño del dato para ver si es de checksum o paridad. Dando igual el caso en el que estemos, se añadirán dos fallos. El primero es un bit al azar transformarlo en el bit contrario y posteriormente se intercambiarán entre dos posiciones aleatorias dos bits entre sí. De esta forma, tendremos una modificación del paquete de datos inicial que se enviará al receptor. Puede ocurrir que falle enviando el código de

checksum y no el de paridad, y viceversa. También puede no fallar ninguno o fallar los dos. Algunos resultados obtenidos son los siguientes:

```
----------------------------------------
Recibido de 192.168.1.1:1111: 10010110001100010
----------------------------------------
Recibido de 192.168.2.1:2222: 01
----------------------------------------
Recibido de 192.168.1.1:1111: 100101100011000111000111
INYECTO ERRORES
DATOS SIN ERRORES->
100101100011000111000111
SE MODIFICA EL BIT->0
SE INTERCAMBIA EL BIT->9 CON EL BIT->12
DATOS CON ERRORES->
000101100011000111000111
----------------------------------------
Recibido de 192.168.2.1:2222: 10
```

En esta imagen podemos ver que con la paridad no ha habido ningún error ya que no se ha generado de forma aleatoria. Ya que se envía el dato al receptor y envía la contestación del mismo al emisor.
En el caso del checksum si ha aparecido un error modificando el bit 0 y realizando un intercambio entre el bit 9 y el 12. Se pueden apreciar mejor los resultados en la siguiente imagen:

```
-----------------------------------------------
ENVIO EL DATO->
10010110001100010
-----------------------------------------------
Recibido de 192.168.1.2:1111: 01
NO HA FALLADO LA PARIDAD
-----------------------------------------------
ENVIO EL DATO->
100101100011000111000111
-----------------------------------------------
Recibido de 192.168.1.2:1111: 10
HA FALLADO EL CHECKSUM, SE NECESITA REENVIO
-----------------------------------------------
```

Esta imagen es la salida del emisor mientras que la anterior es la salida del intermediario (este muestra más específicamente los errores). Como podemos ver en la imagen coincide con lo explicado anteriormente.

En las siguientes imágenes se pueden ver el resto de casos:

```
------------------------------------------
ENVIO EL DATO->
10010110001100010
------------------------------------------
ENVIO EL DATO->
100101100011000111000111
------------------------------------------
Recibido de 192.168.1.2:1111: 00
HA FALLADO LA PARIDAD, SE NECESITA REENVIO
------------------------------------------
Recibido de 192.168.1.2:1111: 11
NO HA FALLADO EL CHECKSUM
------------------------------------------


------------------------------------------
ENVIO EL DATO->
10010110001100010
------------------------------------------
ENVIO EL DATO->
100101100011000111000111
------------------------------------------
Recibido de 192.168.1.2:1111: 10
HA FALLADO EL CHECKSUM, SE NECESITA REENVIO
------------------------------------------
Recibido de 192.168.1.2:1111: 00
HA FALLADO LA PARIDAD, SE NECESITA REENVIO
------------------------------------------


------------------------------------------
ENVIO EL DATO->
10010110001100010
------------------------------------------
Recibido de 192.168.1.2:1111: 01
NO HA FALLADO LA PARIDAD
------------------------------------------
ENVIO EL DATO->
100101100011000111000111
------------------------------------------
Recibido de 192.168.1.2:1111: 11
NO HA FALLADO EL CHECKSUM
------------------------------------------
DATOS ENVIADOS CORRECTAMENTE, TODAS LAS COMPROBACIONES SON CORRECTAS
```

Respecto a los algoritmos en sí, va a suceder exactamente lo mismo que ocurría en el caso anterior donde checksum era más fiable ya que en la paridad puede ocurrir que se enmascare un error al cambiar dos bits de posición. En el caso de cambiar un solo bit de valor si que funcionaría correctamente.