


MEMORIA BLOQUE 2

Proyecto Mysensors



IOT + DIY = 
MySensors

José Gallardo Harillo

10/11/2022

Indice:

Objetivos.....	3
Proyecto basado en Mysensors	3
Configuración previa	5
Configuración del Gateway	5
Instalación y configuración de Domoticz	6
NODOS	9
Nodo 1:.....	9
Conexión Hardware.....	9
Código.....	9
Resultado.....	12
Nodo 2:.....	12
Conexión Hardware.....	12
Código.....	13
Resultado.....	16
Consumo de potencia:	16
Conexión Hardware de medición.....	17
Medida 1: con regulador	17
Medida 2: sin regulador	18
Signing:	18
Configuración en Gateway	19
Configuración en Arduino Pro Mini (Nodo 2).....	21
Conclusión.....	22

Objetivos

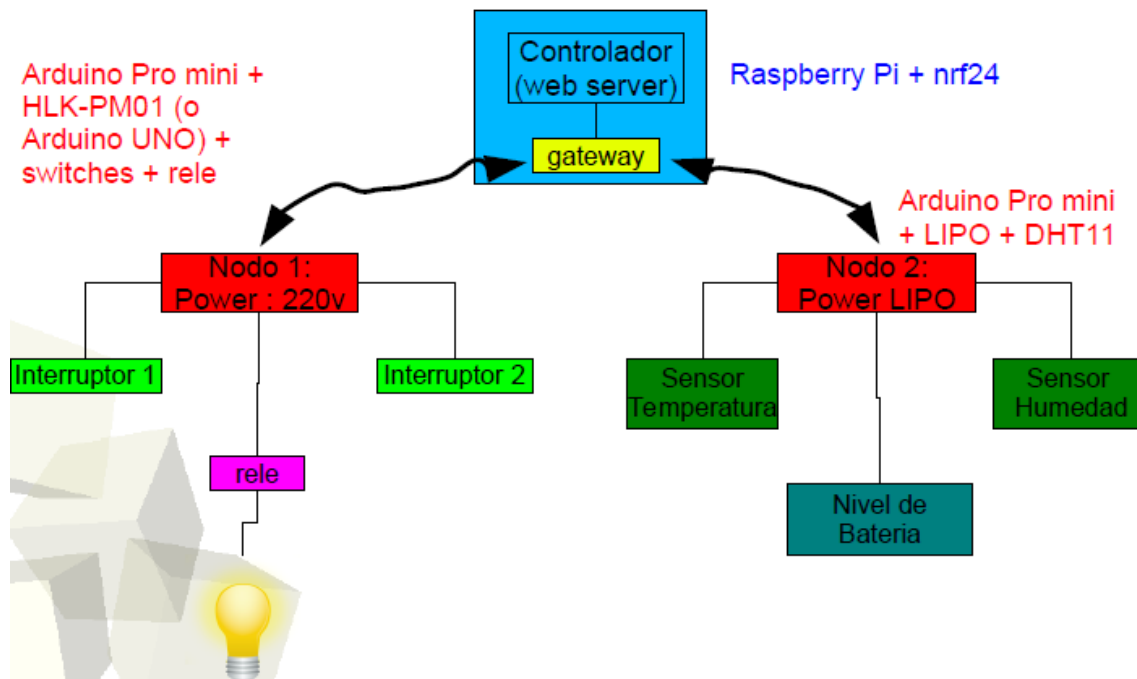
1. Construir un sistema basado en MySensors
2. Saber el papel que tiene Arduino y Raspberry en un sistema MySensor.
3. Toma de contacto con Domoticz y poder controlar el sistema a distancia con este.
4. Comprender la importancia del regulador de tensión en placas como Arduino.

Proyecto basado en Mysensors

MySensors es una fuente abierta de software y hardware para dispositivos inalámbricos IoT (Internet de las cosas) que permite que los dispositivos se comuniquen mediante transmisores de radio .



Nuestro sistema se basará en los aportes que nos da, teniendo que quedar de la siguiente forma contruido:



Donde:

- Los nodos serán dos placas Arduino (Nodo 1 será un Arduino Nano y el nodo 2 un Arduino Pro Mini en mi caso), que serán aquellos que transmitan la información de los sensores y reciban las órdenes para los actuadores.
- El Gateway (puerta de enlace) es la parte del sistema que recibirá y enviará la información para los sensores/actuadores de los nodos “conectados” a este, siendo configurado en la Raspberry pi, y este será controlado por el controlador/servidor web Domoticz, donde nosotros podremos visualizar e interactuar con la información proporcionada.
- Las comunicaciones entre los nodos y el Gateway se realizarán mediante módulos nrf24 conectados en cada uno de ellos (en el caso de Arduino Nano ya lo lleva incluido).
- El nodo 1 se encargará de enviar la información de dos switches que cambien el estado de una bombilla/relé (en mi caso solo el relé),

pudiendo cambiar de estado también en la propia página de Domoticz.

- El nodo 2 se encargará de enviar la información de un sensor DHT11 (temperatura y humedad) y del voltaje actual de la batería del Arduino Pro Mini para que sean ilustrados en la página de Domoticz.

Configuración previa

Configuración del Gateway

En primer lugar, vamos a conectar a la Raspberry pi un módulo nrf24 para que se pueda comunicar con los nodos.



Para el caso de la Raspberry pi podemos hacer uso de un adaptador y conectarlo de la siguiente forma (IMPORTANTE QUE SE CONECTE DE ESA FORMA EN CONCRETO).

Hecho el montaje, ahora vamos a clonar el repositorio git de MySensors:

```
"$ git clone https://github.com/mysensors/MySensors.git"
```

A continuación nos vamos a la carpeta "MySensors" mediante el comando `cd ($ cd MySensors)`, y realizamos la siguiente configuración:

```
"$ ./configure --my-transport=rf24 --my-rf24-channel=36 --my-gateway=ethernet --my-port=5003"
```

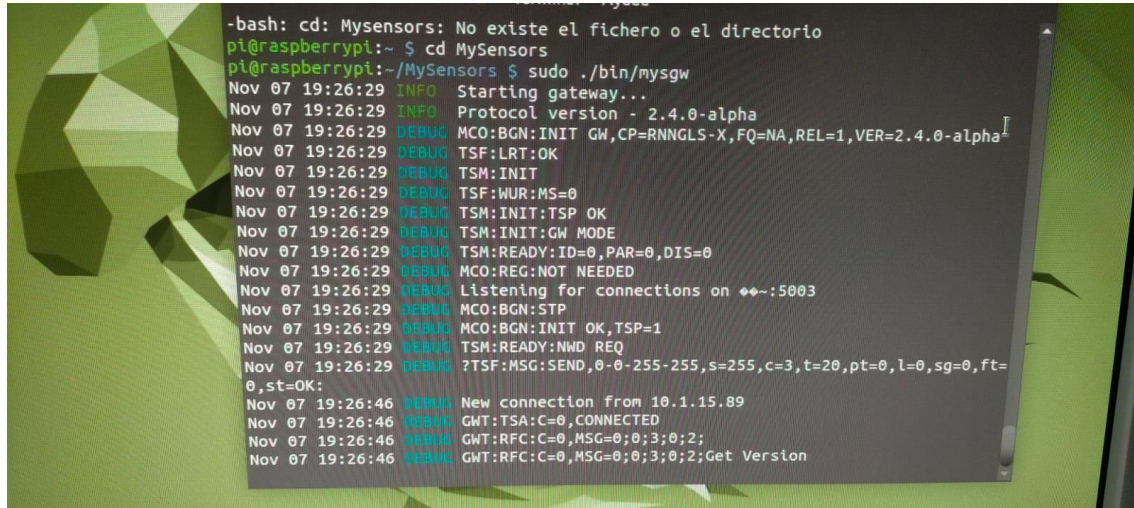
(En mi caso el canal rf24 es el 36, en tu caso pon el canal que se te asigne)

Ejecutado el comando, usamos el comando `make` para que se ejecuten los cambios.

Una carpeta llamada “bin” se nos habrá creado en la carpeta MySensors, donde tendremos un ejecutable llamado “mysgw” el cuál tendremos que ejecutar mediante:

“\$ sudo ./bin/mysgw”

Si la comunicación se está realizando saldrá lo siguiente:



```
-bash: cd: Mysensors: No existe el fichero o el directorio
pi@raspberrypi:~$ cd MySensors
pi@raspberrypi:~/MySensors$ sudo ./bin/mysgw
Nov 07 19:26:29 INFO Starting gateway...
Nov 07 19:26:29 INFO Protocol version - 2.4.0-alpha
Nov 07 19:26:29 DEBUG MCO:BGN:INIT GW,CP=RNNGLS-X,FQ=NA,REL=1,VER=2.4.0-alpha
Nov 07 19:26:29 DEBUG TSF:LRT:OK
Nov 07 19:26:29 DEBUG TSM:INIT
Nov 07 19:26:29 DEBUG TSF:WUR:MS=0
Nov 07 19:26:29 DEBUG TSM:INIT:TSP OK
Nov 07 19:26:29 DEBUG TSM:INIT:GW MODE
Nov 07 19:26:29 DEBUG TSM:READY:ID=0,PAR=0,DIS=0
Nov 07 19:26:29 DEBUG MCO:REG:NOT NEEDED
Nov 07 19:26:29 DEBUG Listening for connections on 5003
Nov 07 19:26:29 DEBUG MCO:BGN:STP
Nov 07 19:26:29 DEBUG MCO:BGN:INIT OK,TSP=1
Nov 07 19:26:29 DEBUG TSM:READY:NMD REQ
Nov 07 19:26:29 DEBUG ?TSF:MSG:SEND,0-0-255-255,s=255,c=3,t=20,pt=0,l=0,sg=0,ft=
0,st=OK:
Nov 07 19:26:46 DEBUG New connection from 10.1.15.89
Nov 07 19:26:46 DEBUG GMT:TSA:C=0,CONNECTED
Nov 07 19:26:46 DEBUG GWT:RFC:C=0,MSG=0;0;3;0;2;
Nov 07 19:26:46 DEBUG GWT:RFC:C=0,MSG=0;0;3;0;2;Get Version
```

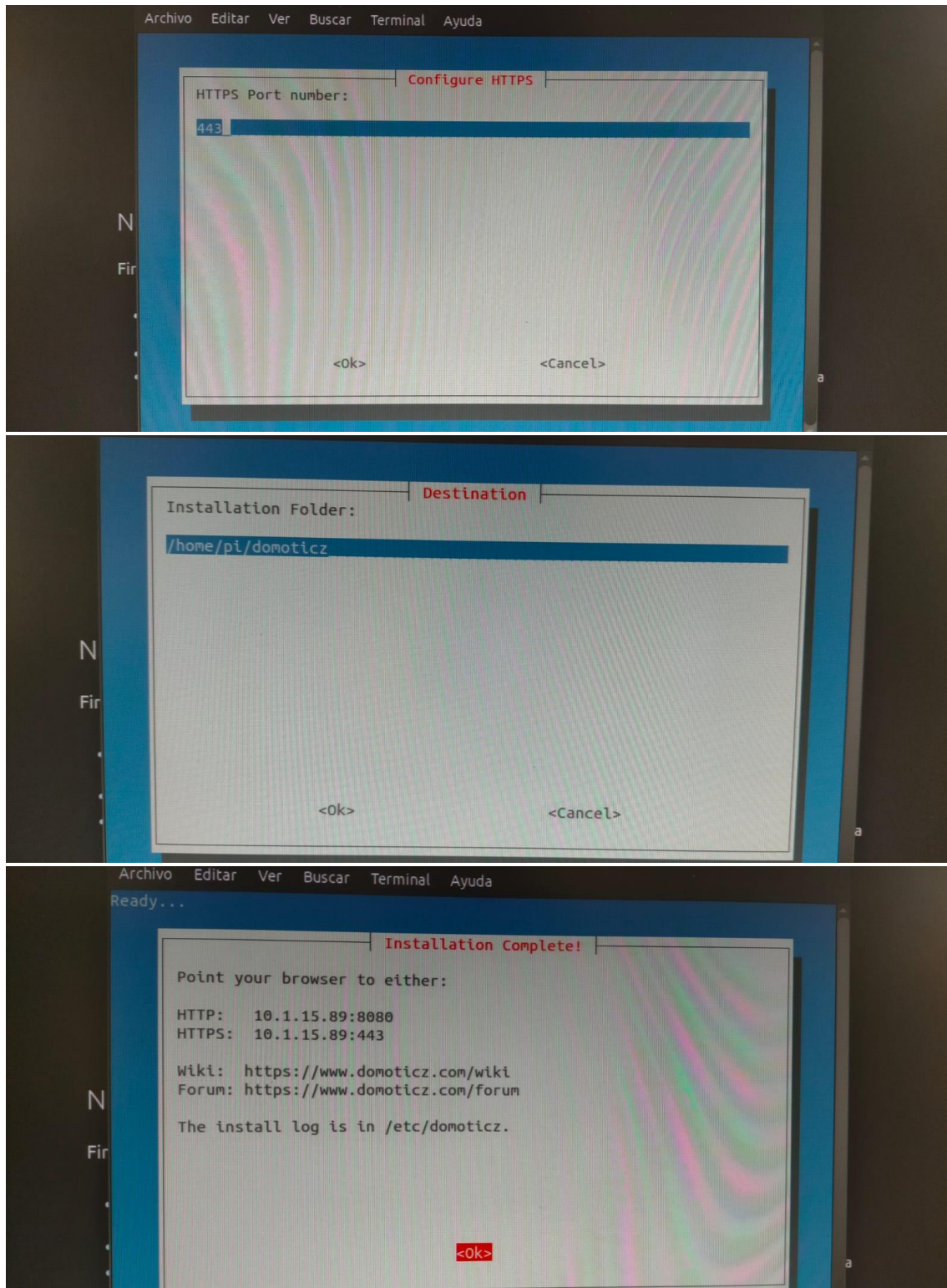
Instalación y configuración de Domoticz

Para instalar Domoticz debemos ejecutar el siguiente comando:

“curl -L <https://install.domoticz.com> | bash”

Yo previamente para que me funcionase el comando anterior tuve que instalar libudev-dev mediante “sudo apt install” .

Cuando ejecutes el comando para instalar Domoticz, si se ha instalado correctamente, te saldrá las siguientes pantallas de configuración:

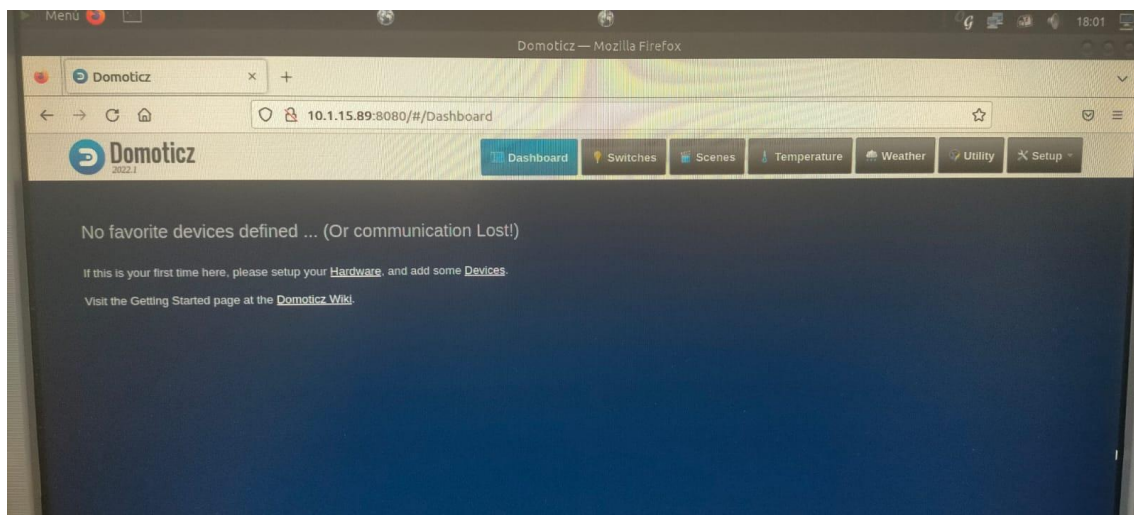


Tras ello, si nos vamos al buscador y ponemos en el buscados:

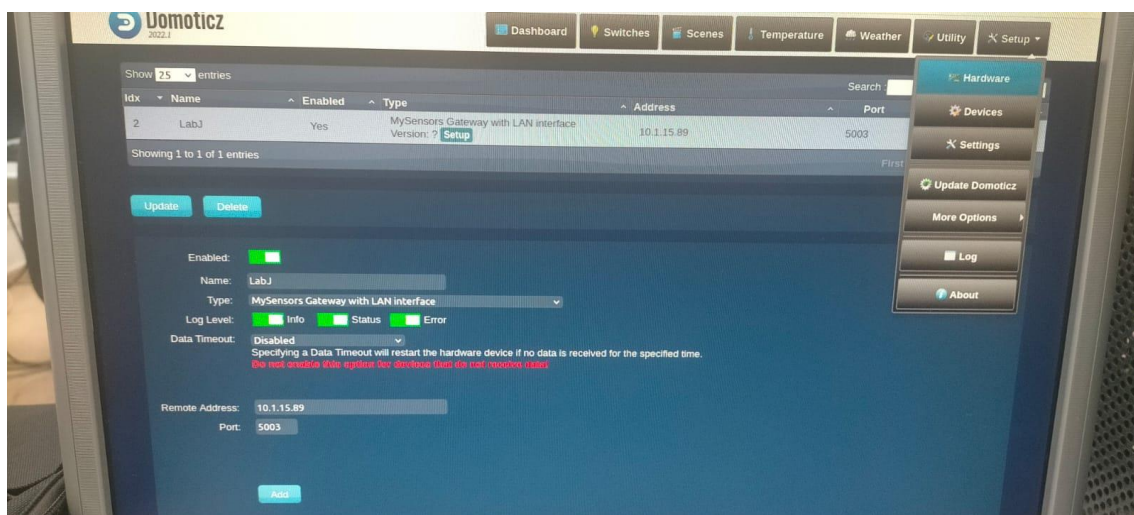
10.1.15.89:8080

(En tu caso pones la dirección IP que tengas asignada)

De esta forma ya estaremos en la página de Domoticz.



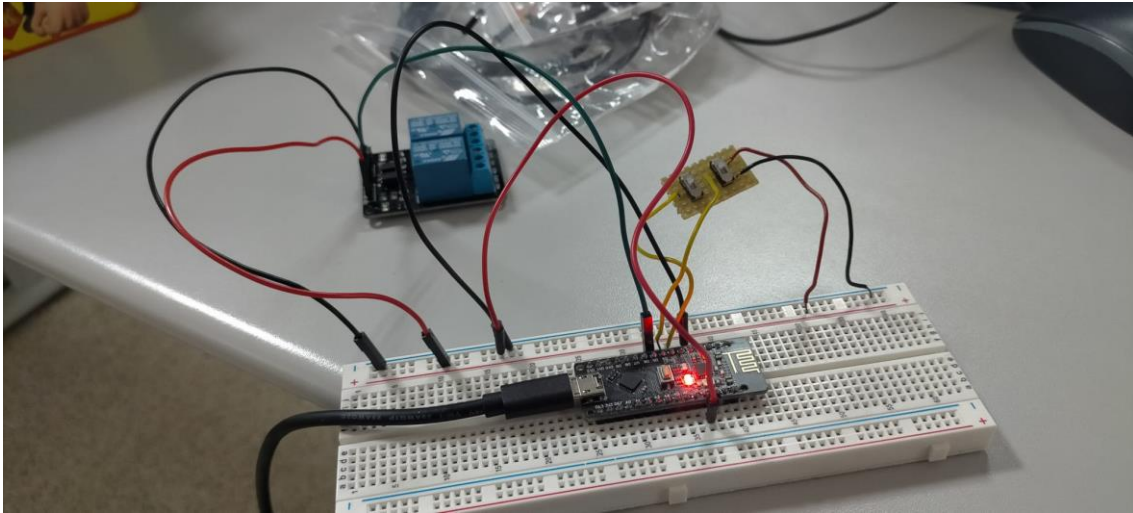
Como configuración final, en la página actual nos vamos a “Configuración”, y seleccionamos “Hardware”, donde tendemos que añadir la configuración para que pueda conectar al Gateway de nuestra Raspberry pi.



NODOS

Nodo 1:

Conexión Hardware



Código

```
//Solo cuando usemos Arduino Nano
#define MY_RF24_CE_PIN 10
#define MY_RF24_CS_PIN 9

#define MY_RF24_CHANNEL 36 //Canal
#define MY_NODE_ID 67 //Mi rango de ids esta entre 66 y 70, sirve para
//identificar a los nodos que incorporemos a nuestro sistema.
#define MY_DEBUG //Para imprimir por puerto serie
#define MY_RADIO_RF24 //Usamos un módulo nrf24

#include <MySensors.h>
#include <Bounce2.h>

//id's
#define child_id1 1 //id del switch 1
#define child_id2 2 //id del switch 2
#define child_id3 3 //id del relé

//pines
#define switch1 3
#define switch2 4
#define rele 5
```

```

//Variables para conservar el estado del relé para los switches
int antiguoValor1 = -1;
int antiguoValor2 = -1;
bool estado = false;

//Dos debouncers para impedir rebote en los switches/botones
Bounce debouncer1 = Bounce();
Bounce debouncer2 = Bounce();

//canal de comunicación entre x id y la raspberry pi
MyMessage mensaje1(child_id1, V_STATUS);
MyMessage mensaje2(child_id2, V_STATUS);
MyMessage mensaje3(child_id3, V_STATUS);

//Presentación de las ids para que se muestren por Domoticz
void presentation(){
  present(child_id1, S_BINARY); //S_BINARY sirve para comportamientos binarios,
  //ideal para botones y relés
  present(child_id2, S_BINARY);
  present(child_id3, S_BINARY);
}

void setup() {
  // put your setup code here, to run once:

  //Configuración de pines
  pinMode(switch1, INPUT);
  pinMode(switch2, INPUT);
  pinMode(rele, OUTPUT);

  //Inicialización
  digitalWrite(switch1,HIGH);
  digitalWrite(switch2,HIGH);
  digitalWrite(rele, 0);

  debouncer1.attach(switch1);
  debouncer1.interval(10);
  debouncer2.attach(switch2);
  debouncer2.interval(10);
}

```

```

void loop() {
  // put your main code here, to run repeatedly:

  estado = false;
  debouncer1.update();
  debouncer2.update();

  //Valores de switches
  int valor1 = debouncer1.read();
  int valor2 = debouncer2.read();

  //Usando switches

  //Comprobación switch 1
  if(valor1 != antiguoValor1){ //Si el estado del switch 1 cambia
    //entonces se informa a Domoticz
    send(mensaje1.set(valor1 == HIGH? 1:0)); //Encendido/Apagado (Esto solo sirve
    //para que se encienda o se apague la bombillita del estado del switch en Domoticz)
    antiguoValor1 = valor1;
    estado = true;
  }

  //Comprobación switch 2
  if(valor2 != antiguoValor2){ //Si el estado del switch 2 cambia entonces se informa a Domoticz
    send(mensaje2.set(valor2 == HIGH? 1:0)); //Encendido/Apagado (Esto solo sirve
    //para que se encienda o se apague la bombillita del estado del switch 2 en Domoticz)
    antiguoValor2 = valor2;
    estado = true;
  }

  if(estado){ //Si el estado ha cambiado, se envía un mensaje para cambiar el relé
    int led = digitalRead(rele);
    send(mensaje3.set(led == HIGH? 1:0)); //Encendido/Apagado en Domoticz
    digitalWrite(rele, led == HIGH? 0:1); //Para cambiar el estado del relé con los switches,
    //cuando se interactua con el relé directamente el valor de encendido/apagado es al revés
  }

}

//Usando domoticz
void receive (const MyMessage &message){
  //identificar a qué sensor se refieren

  if(message.getSensor() == child_id3 && message.type == V_STATUS){
    //Si el mensaje va relacionado con el relé

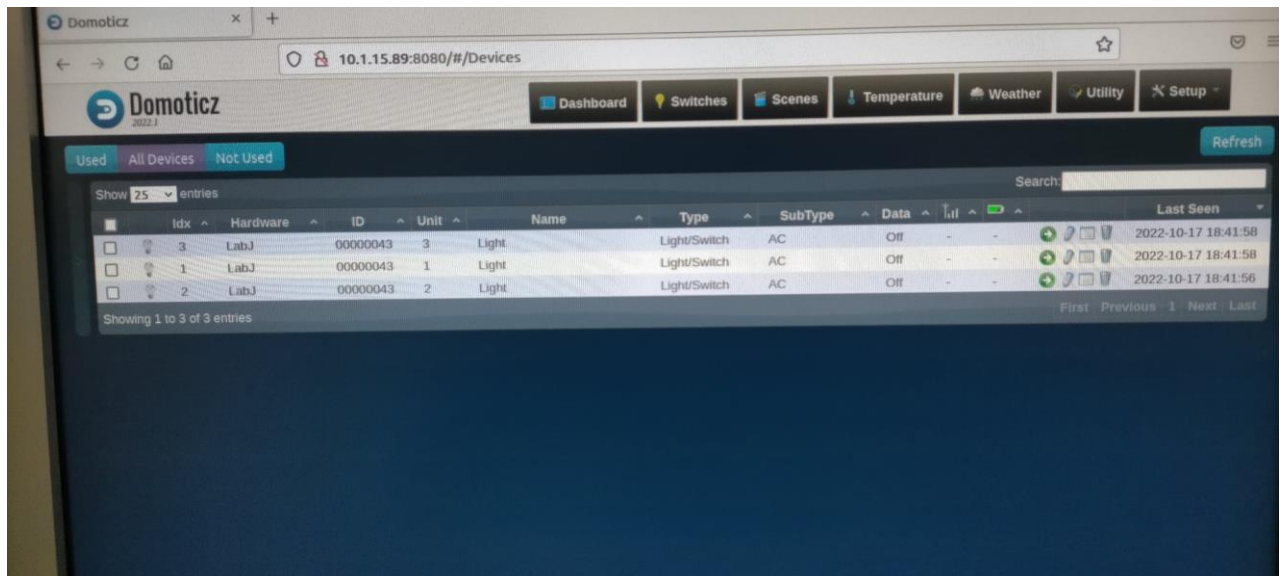
    estado = message.getBool();
    digitalWrite(rele, estado? 0:1); //Encendido/apagado de relé desde domoticz
    //cuando se interactua con el relé directamente el valor de encendido/apagado es al revés

    //Impresión puerto serie cuando se activa desde domoticz si está debug activo
    #ifdef MY_DEBUG
    Serial.print("Cambios detectados en el sensor:");
    Serial.print(message.getSensor());
    Serial.print(", Nuevo estado: ");
    Serial.println(message.getBool());
    #endif

  }
}

```

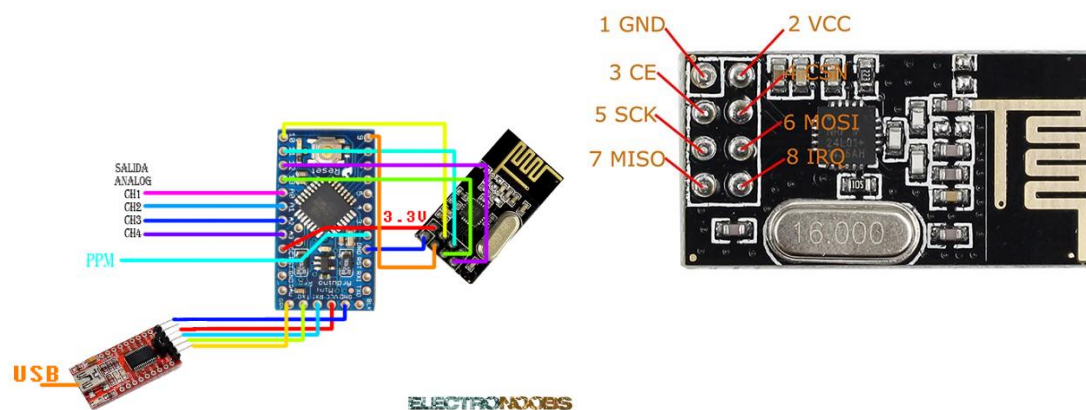
Resultado

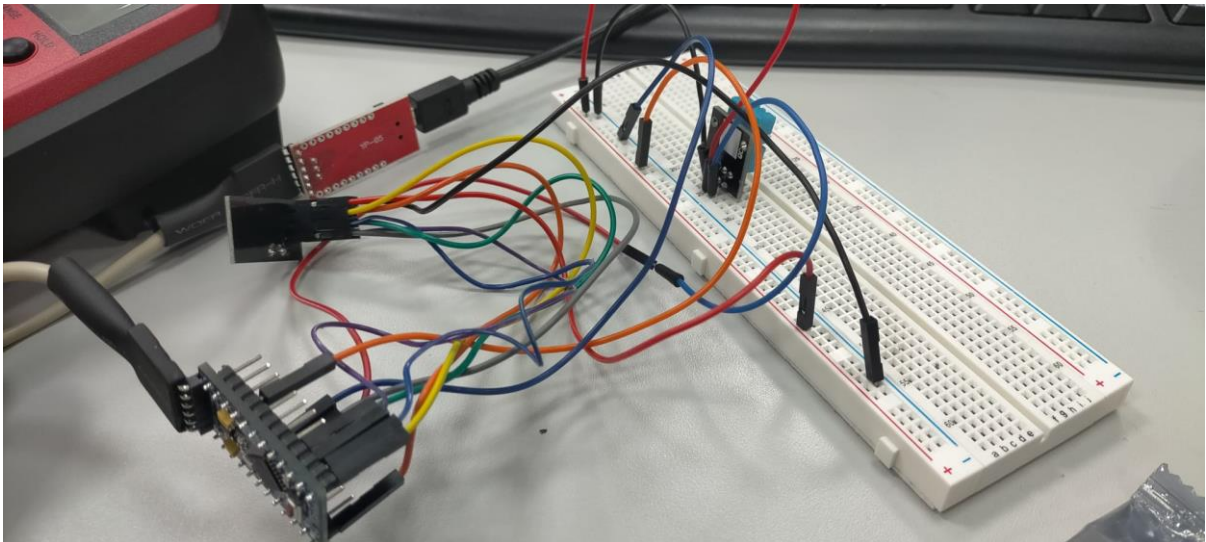


Nodo 2:

Conexión Hardware

El Arduino Pro Mini no lleva incluido un módulo nrf24 como en el caso del Arduino Nano, por tanto habrá que conectarlo de la siguiente forma:





Código

```
#define MY_RF24_CHANNEL 36 //Canal
#define MY_NODE_ID 67 //Mi rango de ids esta entre 66 y 70, sirve para
//identificar a los nodos que incorporemos a nuestro sistema.
#define MY_DEBUG //Para imprimir por puerto serie
#define MY_RADIO_RF24 //Usamos un módulo nrf24

#include <MySensors.h>
#include <DHT.h>
#include <Vcc.h>

#define DHT_DATA_PIN 3 //Pin DHT

// Se establece esta compensación si el sensor tiene una pequeña compensación
// permanente con respecto a las temperaturas reales.
//En grados celcius (como temperatura por defecto)
#define SENSOR_TEMP_OFFSET 0

// Tiempo en el que se mantiene dormido entre actualización (en milisegundos)
// Debe ser >1000ms para DHT22 y >2000ms para DHT11
static const uint64_t UPDATE_INTERVAL = 60000;

//Objeto DHT
DHT dht;

//Batería

const float VccExpected = 3.0;
const float VccCorrection = 2.860/2.92;
Vcc vcc(VccCorrection);
float v;
int oldBatteryPcnt = 0;
int BatteryPcnt;

// Fuerza enviar una actualización de la temperatura despues de 10 lecturas de sensor
static const uint8_t FORCE_UPDATE_N_READS = 10;

//id's
#define CHILD_ID_HUM 1
#define CHILD_ID_TEMP 2
#define CHILD_ID_BATTERY 3

//Tempertura y humedad
float lastTemp;
float lastHum;

uint8_t nNoUpdatesTemp;
uint8_t nNoUpdatesHum;

bool metric = true; //Para indicar si se va a usar la medida por defecto (Celcius) o en algún otro (p.e Fahrenheit)
```



```

MyMessage msgHum(CHILD_ID_HUM, V_HUM);
MyMessage msgTemp(CHILD_ID_TEMP, V_TEMP);
MyMessage msgBat(CHILD_ID_BATTERY, V_VOLTAGE);

//Presentación de las ids para que se muestren por Domoticz
void presentation(){

    sendSketchInfo("TemperatureAndHumidity", "1.1");
    present(CHILD_ID_HUM, S_HUM);
    present(CHILD_ID_TEMP, S_TEMP);
    present(CHILD_ID_BATTERY, S_MULTIMETER);

    metric = getControllerConfig().isMetric; //Configurado para Celcius
}

void setup() {
    // put your setup code here, to run once:

    dht.setup(DHT_DATA_PIN); // Inicializa el DHT para que el pin de dato dea el indicado
    if (UPDATE_INTERVAL <= dht.getMinimumSamplingPeriod()) { //Si no se respeta el intervalo
        Serial.println("Warning: UPDATE_INTERVAL is smaller than supported by the sensor!");
    }
    //Duerme el tiempo del mínimo periodo de muestreo para darle tiempo a activarse
    //(de otra manera, podría ocurrir un error de timeout en la primera lectura)
    sleep(dht.getMinimumSamplingPeriod());
}

void loop() {
    // put your main code here, to run repeatedly:

    //ZONA BATERIA

    //Para el porcentaje(Sólo se imprime en DEBUG)
    if (oldBatteryPcnt != BatteryPcnt)
    {
        BatteryPcnt = (int)vcc.Read_Perc(VccExpected);
        oldBatteryPcnt = BatteryPcnt;
    }

    float v = vcc.Read_Volts(); //Lectura del voltaje

    //Impresión puerto serie de voltaje si está debug activo
#ifdef MY_DEBUG
    Serial.print("VCC: ");
    Serial.print(v);
    Serial.println(" V");
    Serial.print("Pcnt: ");
    Serial.print(BatteryPcnt);
    Serial.println("%");
#endif

    send(msgBat.set(v,1)); //Envío de voltaje a Domoticz
}

```

```

//ZONA TEMPERATURA

// Lectura del DHT
dht.readSensor(true);

//Obtención de la temperatura
float temperature = dht.getTemperature();

if (isnan(temperature)) {
    Serial.println("Failed reading temperature from DHT!");
} else if (temperature != lastTemp || nNoUpdatesTemp == FORCE_UPDATE_N_READS) {
    // Sólo envía temperatura si cambia desde la última medida o si no enviamos una actualización 10 veces
    lastTemp = temperature;

    //Aplica el compensador antes de convertirlo en algo diferente que grados Celcius
    temperature += SENSOR_TEMP_OFFSET;

    if (!metric) {
        temperature = dht.toFahrenheit(temperature);
    }

    //Resetea el contador de actualizaciones
    nNoUpdatesTemp = 0;
    send(msgTemp.set(temperature, 1)); //Envío de temperatura a Domoticz

    //Impresión puerto serie de temperatura si está debug activo
    #ifdef MY_DEBUG
        Serial.print("T: ");
        Serial.println(temperature);
    #endif
} else {
    // Incrementa el contador de iteraciones sin actualización si la temperatura se mantiene igual
    nNoUpdatesTemp++;
}

//ZONA HUMEDAD

// Obtención de humedad
float humidity = dht.getHumidity();

if (isnan(humidity)) {
    Serial.println("Failed reading humidity from DHT!");
} else if (humidity != lastHum || nNoUpdatesHum == FORCE_UPDATE_N_READS) {
    // Sólo envía humedad si cambia desde la última medida o si no enviamos una actualización 10 veces
    lastHum = humidity;
    //Resetea el contador de actualizaciones
    nNoUpdatesHum = 0;
    send(msgHum.set(humidity, 1)); //Envío de humedad a Domoticz

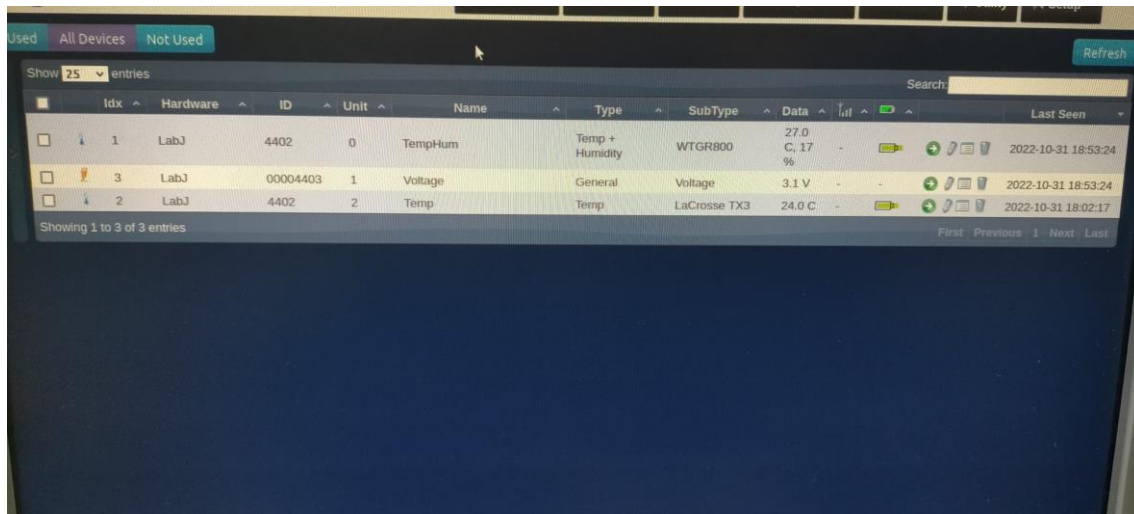
    //Impresión puerto serie de humedad si está debug activo
    #ifdef MY_DEBUG
        Serial.print("H: ");
        Serial.println(humidity);
    #endif
} else {
    // Incrementa el contador de iteraciones sin actualización si la temperatura se mantiene igual
    nNoUpdatesHum++;
}

// Se duerme durante el intervalo establecido
sleep(UPDATE_INTERVAL);
}

```

Resultado

Configurándolo en Arduino a 3.3 V en un procesador ATmega328p:



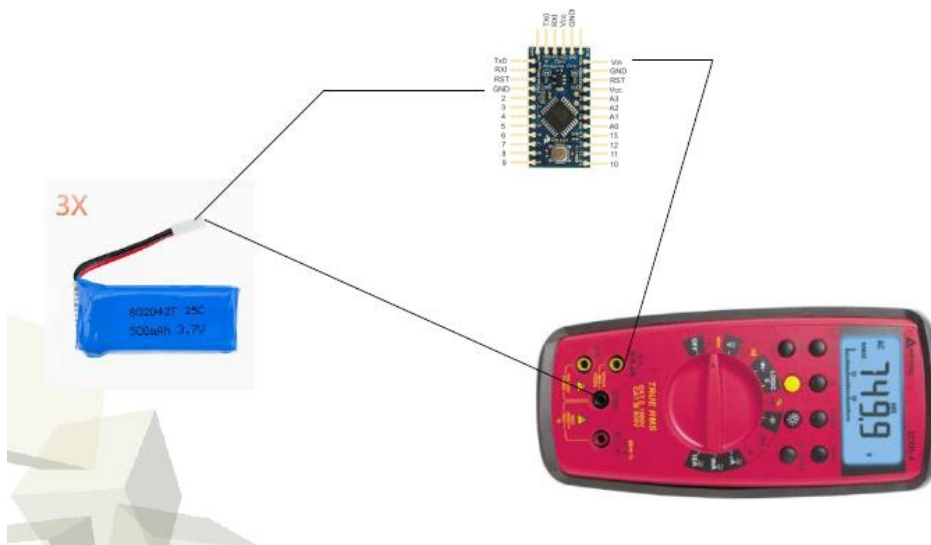
Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Last Seen
1	LabJ	4402	0	TempHum	Temp + Humidity	WTGR800	27.0 C, 17 %	2022-10-31 18:53:24
3	LabJ	00004403	1	Voltage	General	Voltage	3.1 V	2022-10-31 18:53:24
2	LabJ	4402	2	Temp	Temp	LaCrosse TX3	24.0 C	2022-10-31 18:02:17

Consumo de potencia:

Usando el nodo 2 como código de prueba, en este apartado vamos a medir la Potencia en dos configuraciones de alimentación diferentes:

1. Alimentando a través de la entrada RAW del Arduino Pro-mini (en esta configuración se emplea el regulador de tensión del Pro-mini).
2. Alimentando a través de algunas de las entradas Vcc de arduino pro-mini (en esta configuración no se emplea el regulador, por lo que la tensión de alimentación tanto de microcontrolador como del módulo NRF24 será diferente de 3.3v (en función de la batería LIPO podrá variar desde 4,2 -100% carga hasta 3v -0% de carga-)

Conexión Hardware de medición



Medida 1: con regulador

Configurado como ATmega328p a 3.3V, la medición da lo siguiente:



Medida 2: sin regulador

Configurado como ATmega168 a 3.3V, sin regulador, y con escasez de memoria. Previamente debemos de comentar la línea que define el DEBUG, para que la impresión no se realice y no consuma memoria de la que no se puede permitir.



Como podemos ver, el consumo del ATmega168 sin regulador es considerablemente inferior al del ATmega328p con regulador.

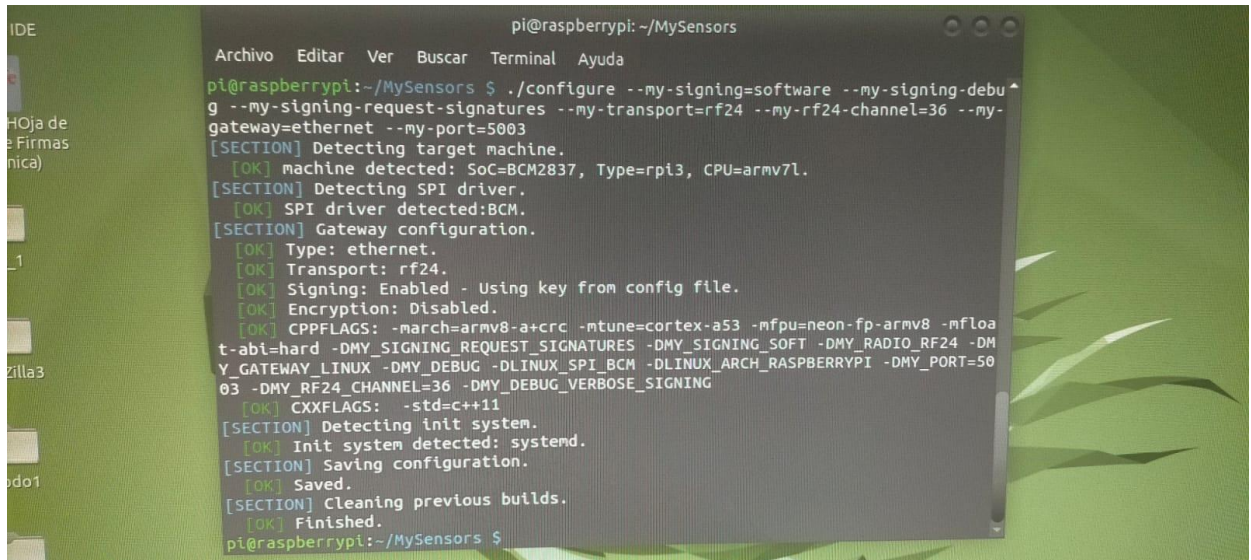
Signing:

En esta última fase configuraremos el sistema MySensors para que aplique una capa de seguridad para impedir casos como el “man in the middle”.

Configuración en Gateway

Tendremos para que el Gateway permita la capa de seguridad y lo depure por puerto serie que reconfigurarlo y añadirle los siguientes comandos:

“\$ --my-signing=software --my-signing-debug --my-signing-request-signatures”



```

pi@raspberrypi: ~/MySensors
Archivo Editar Ver Buscar Terminal Ayuda
pi@raspberrypi:~/MySensors $ ./configure --my-signing=software --my-signing-debug --my-signing-request-signatures --my-transport=rf24 --my-rf24-channel=36 --my-gateway=ethernet --my-port=5003
[SECTION] Detecting target machine.
[OK] machine detected: SoC=BCM2837, Type=rpi3, CPU=armv7L.
[SECTION] Detecting SPI driver.
[OK] SPI driver detected:BCM.
[SECTION] Gateway configuration.
[OK] Type: ethernet.
[OK] Transport: rf24.
[OK] Signing: Enabled - Using key from config file.
[OK] Encryption: Disabled.
[OK] CPPFLAGS: -march=armv8-a+crc -mtune=cortex-a53 -mfpu=neon-fp-armv8 -mfloa
t-abi=hard -DMY_SIGNING_REQUEST_SIGNATURES -DMY_SIGNING_SOFT -DMY_RADIO_RF24 -DM
Y_GATEWAY_LINUX -DMY_DEBUG -DLINUX_SPI_BCM -DLINUX_ARCH_RASPBERRYPI -DMY_PORT=50
03 -DMY_RF24_CHANNEL=36 -DMY_DEBUG_VERBOSE_SIGNING
[OK] CXXFLAGS: -std=c++11
[SECTION] Detecting init system.
[OK] Init system detected: systemd.
[SECTION] Saving configuration.
[OK] Saved.
[SECTION] Cleaning previous builds.
[OK] Finished.
pi@raspberrypi:~/MySensors $
  
```

Hacemos make y generamos las claves Serial y HMAC mediante los siguientes comandos:

“\$ sudo ./bin/mysgw --gen-soft-serial-key”

“\$ sudo ./bin/mysgw --gen-soft-hmac-key”

```

core      examples_linux  library.properties  MyConfig.h
pi@raspberrypi:~/MySensors $ sudo ./bin/mysgw --gen-soft-hmac-key
Generating key... done.
To use the new key, update the value in /etc/mysensors.conf with:
soft_hmac_key=F0A11060D849670164CF639C8494277AEFA237288D8B86396E19E55068FC9641

The next line is intended to be used in SecurityPersonalizer.ino:
#define MY_HMAC_KEY 0XF0,0XA1,0X10,0X60,0XD8,0X49,0X67,0X1,0X64,0XCF,0X63,0X9C,0
X84,0X94,0X27,0X7A,0XEF,0XA2,0X37,0X28,0X8D,0X8B,0X86,0X39,0X6E,0X19,0XE5,0X50,0
X68,0XFC,0X96,0X41

pi@raspberrypi:~/MySensors $ sudo ./bin/mysgw --gen-soft-serial-key
Generating key... done.
To use the new key, update the value in /etc/mysensors.conf with:
soft_serial_key=9D72E57EC437F0409D

The next line is intended to be used in SecurityPersonalizer.ino:
#define MY_SOFT_SERIAL 0X9D,0X72,0XE5,0X7E,0XC4,0X37,0XF0,0X40,0X9D

pi@raspberrypi:~/MySensors $

```

Las claves deberán ser escritas en el archivo /etc/mysensors.conf

```

# EEPROM settings
eeeprom_file=/etc/mysensors.eeprom
eeeprom_size=1024

# Software signing settings
# Note: The gateway must have been built with signing
# support to use the options below.
#
# To generate a HMAC key run mysgw with: --gen-soft-hmac-key
# copy the new key in the line below and uncomment it.
soft_hmac_key=F0A11060D849670164CF639C8494277AEFA237288D8B86396E19E55068FC9641
# To generate a serial key run mysgw with: --gen-soft-serial-key
# copy the new key in the line below and uncomment it.
soft_serial_key=9D72E57EC437F0409D

# Encryption settings
# Note: The gateway must have been built with encryption
# support to use the options below.
#

```

Finalmente comprobamos que el Gateway funciona correctamente ejecutando “\$ sudo ./bin/mysgw”

```

pi@raspberrypi:~ $ sudo nano /etc/mysensors.conf
pi@raspberrypi:~ $ sudo nano /etc/mysensors.conf
pi@raspberrypi:~ $ cd MySensors
pi@raspberrypi:~/MySensors $ sudo ./bin/mysgw
Nov 17 17:47:18 INFO Starting gateway...
Nov 17 17:47:18 INFO Protocol version - 2.4.0-alpha
Nov 17 17:47:18 DEBUG MCO:BGN:INIT GW,CP=RNGLS--,FQ=NA,REL=1,VER=2.4.0-alpha
Nov 17 17:47:18 DEBUG SGN:PER:OK
Nov 17 17:47:18 DEBUG SGN:INI:BND OK
Nov 17 17:47:18 DEBUG TSF:LRT:OK
Nov 17 17:47:18 DEBUG TSM:INIT
Nov 17 17:47:18 DEBUG TSF:WUR:MS=0
Nov 17 17:47:18 DEBUG TSM:INIT:TSP OK
Nov 17 17:47:18 DEBUG TSM:INIT:GW MODE
Nov 17 17:47:18 DEBUG TSM:READY:ID=0,PAR=0,DIS=0
Nov 17 17:47:18 DEBUG MCO:REG:NOT NEEDED
Nov 17 17:47:18 DEBUG Listening for connections on *-:5003
Nov 17 17:47:18 DEBUG MCO:BGN:STP
Nov 17 17:47:18 DEBUG MCO:BGN:INIT OK,TSP=1
Nov 17 17:47:18 DEBUG TSM:READY:NWD REQ
Nov 17 17:47:18 DEBUG SGN:SGN:NREQ=255
Nov 17 17:47:18 DEBUG ?TSF:MSG:SEND,0-0-255-255,s=255,c=3,t=20,pt=0,l=0,sg=0,ft=
0,st=OK:

```

Configuración en Arduino Pro Mini (Nodo 2)

Para configurar nuestro Arduino Pro Mini para la capa de seguridad, antes tendremos que cargar en el un fichero .ino llamado “SecurityPersonalizer”, que podremos encontrar su código en:

https://www.mysensors.org/apidocs/SecurityPersonalizer_8ino-example.html

En el fichero SecurityPersonalizer debemos realizar unas pocas modificaciones:

1. Comentamos `#include "sha204_library.h"`
2. Comentamos `#include "sha204_lib_return_codes.h"`
3. Indicamos las claves HMAC y Serial
4. Descomentamos `#define PERSONALIZE_SOFT`

```
//#include "sha204_library.h"
//#include "sha204_lib_return_codes.h"
#define MY_CORE_ONLY
#include <MySensors.h>

/***** User defined key data *****/

#define MY_HMAC_KEY 0xF0, 0xA1, 0x10, 0x60, 0xD8, 0x49, 0xE7, 0x01, 0x64, 0xCF, 0x63, 0x9C, 0x84, 0x94, 0x27, 0x7A, 0xEF, 0xA2, 0x37, 0x28, 0x8D, 0x8B, 0x86, 0x39, 0xE5, 0x19, 0xE5, 0x50, 0x68, 0xFC, 0x96, 0x41

#define MY_AES_KEY 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

#define MY_SOFT_SERIAL 0x9D, 0x72, 0xE5, 0x7E, 0xC4, 0x37, 0xF0, 0x40, 0x9D

/***** Flags for guided personalization flow *****/

//#define GENERATE_KEYS_ATSHA204A
//#define GENERATE_KEYS_SOFT
//#define PERSONALIZE_ATSHA204A
#define PERSONALIZE_SOFT
```

Cargamos el código al Arduino, la configuración queda en la placa ya.

A continuación en el segundo código a cargar, ósea, el del nodo 2, vamos a añadir las siguientes cabeceras:

```
#define MY_SIGNING_SOFT
#define MY_SIGNING_SOFT_RANDOMSEED_PIN 7
#define MY_SIGNING_REQUEST_SIGNATURES
#define MY_DEBUG_VERBOSE_SIGNING
```



```

#define MY_RF24_CHANNEL 36 //Canal
#define MY_NODE_ID 67 //Mi rango de ids esta entre 66 y 70, sirve para
//identificar a los nodos que incorporemos a nuestro sistema.
#define MY_DEBUG //Para imprimir por puerto serie
#define MY_RADIO_RF24 //Usamos un módulo nrf24

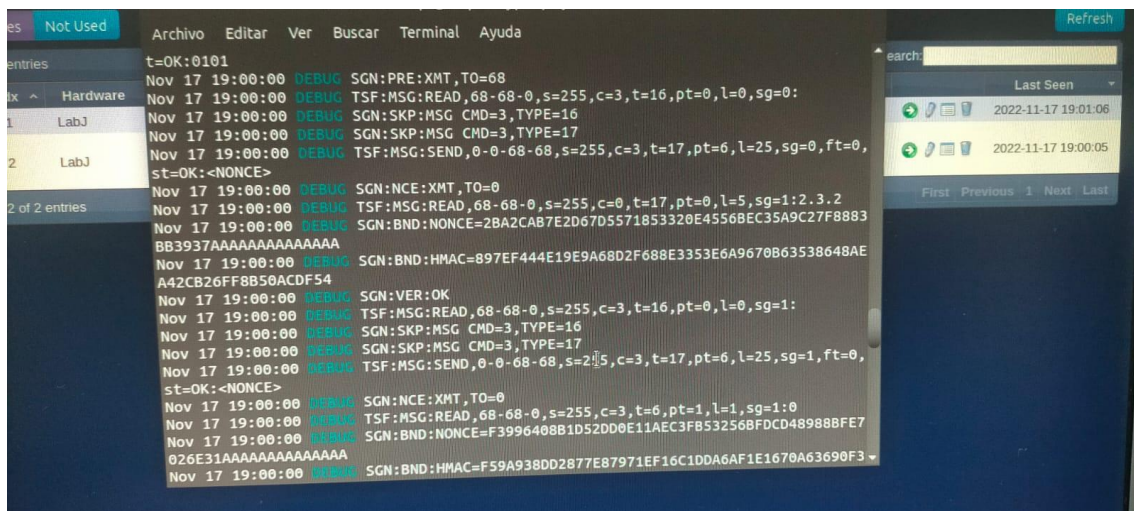
//SIGNING
#define MY_SIGNING_SOFT
#define MY_SIGNING_SOFT_RANDOMSEED_PIN 7
#define MY_SIGNING_REQUEST_SIGNATURES
#define MY_DEBUG_VERBOSE_SIGNING

#include <MySensors.h>
#include <DHT.h>
#include <Vcc.h>

#define DHT_DATA_PIN 3 //Pin DHT

```

Cargado el código, ahora por puerto serie veríamos líneas de código haciendo alusión a una previa comprobación de claves de signing (SGN), y por tanto, daríamos por configurada la capa de seguridad.



Conclusión

He podido cumplir los objetivos, y además comprender mejor la infraestructura iot. Considero los ejercicios muy interesantes, ya que en algún momento de la carrera alguno de nosotros habremos pensado en automatizar nuestra casa, y pudiendo ser capaz mediante lo que hemos aprendido. La documentación en las dos últimas fases me ha sido más complicada de abordarlas por lo poco que hay en las transparencias de ello, pero pese a ello se ha podido completar perfectamente.