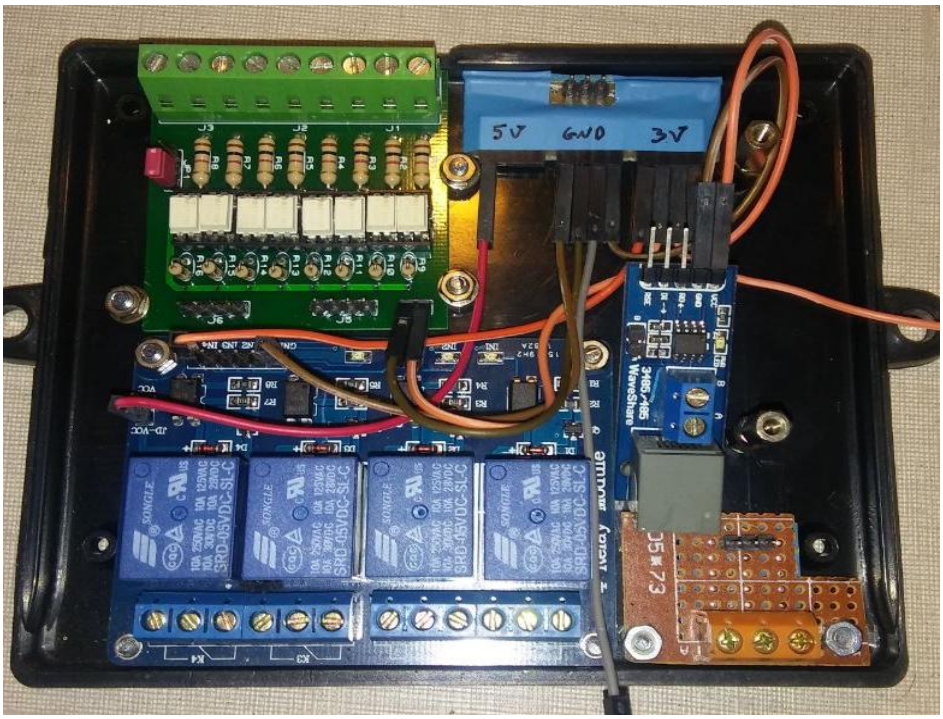
PROYECTO GERYON



20/09/2018

DOMÓTICA DIY

Sistema domótico integrado por un módulo central, varios esclavos distribuidos y una pasarela MQTT.

Bus físico RS485 con protocolo Modbus RTU.

Pasarela RS485 – MQTT.

Versión 0.1

Autor: JR Medusa

PROYECTO GERYON

DOMÓTICA DIY

INDICE

INDICE 1

TABLA DE ILUSTRACIONES 3

INTRODUCCIÓN 4
- OBJETIVOS 4

PRESENTACION 5
- ARQUITECTURAS 7
  - SISTEMA 7
  - ESCLAVO 7
  - MAESTRO 8
  - PASARELA MQTT 8

UNIDAD CENTRAL [MF-UC] 9

ENTRADAS [MF-EO] 10
- CALCULOS 12
- ENTRADA 240VAC [MF-EAC] 13

SALIDAS [MF-SXR] 16

COMUNICACIÓN [MF-COM] 17

TEMPERATURA [MF-TX] 20

ALIMENTACIÓN [MF-DA] 22

IMPLEMENTACION MAESTRO 23

PROGRAMACION 24
- TEMPORIZADORES [crono.h] 24
- TEMPERATURA 24
  - DHTxx [dhtxx.h] 24
  - NTC [ntc.h] 24
- REGISTROS 25
  - REGISTROS DE UN ESCLAVO 25
  - REGISTROS DEL MAESTRO 26
  - REGISTROS DE LA PASARELA MQTT 27
- ESCLAVO 28
  - MODBUS 28
  - CODIGO 29
- MAESTRO 31
  - MODBUS 31
  - CÓDIGO 31
- PASARELA MQTT 32

MODBUS 32
CODIGO MQTT 32
CASO PRACTICO 34

HOME-ASSISTANT 35

DISEÑO MANTENIMIENTO 36

EVOLUCIONES 37

HERRAMIENTAS 38
SOFTWARE 38
DISEÑO Y SIMULACION 38
PROGRAMACION 38
HARDWARE 38
MATERIALES 38

TABLA DE ILUSTRACIONES

Ilustración 1. Cableado convencional 5
Ilustración 2. Cableado domótico 5
Ilustración 3. Telerruptores 6
Ilustración 4. Arquitectura de red 7
Ilustración 5. Arquitectura de un esclavo 8
Ilustración 6. Pasarela MQTT 8
Ilustración 7. Esquemático MF-EO 10
Ilustración 8. Diseño PCB MF-EO 11
Ilustración 9. Prototipo, PCB y placa terminada 12
Ilustración 10. Esquema de conexión PIR 13
Ilustración 11. Convertidor 240vac a 5vdc 14
Ilustración 12. Prototipo y pruebas 15
Ilustración 13. Detalle montaje condensador 15
Ilustración 14. Detalle montaje módulos 2/4 relés 16
Ilustración 15. Módulos RS485 17
Ilustración 16. Convertidor RS485-USB 18
Ilustración 17. Sinóptico de cableado de red 18
Ilustración 18. Bornero MF-COM 18
Ilustración 19. Diagrama de conexionado de red 19
Ilustración 20. Módulos DHTxx 20
Ilustración 21. Diseño placa NTC 21
Ilustración 22. Implementación placas NTC 21
Ilustración 23. Distribuidor pasivo 22
Ilustración 24. Mapping esclavo 25
Ilustración 25. Mapping maestro 0-380 26
Ilustración 26. Mapping pasarela 0-34 27
Ilustración 27. Estructura funcional pasarela 32

INTRODUCCIÓN

Según la mitología antigua, el décimo trabajo de Hércules consistió en robar el ganado de Geryon. Es descrito como un ser antropomorfo formado por tres cuerpos, con sus respectivas cabezas y extremidades.

En este proyecto crearemos un sistema domótico completo. Para ello diseñaremos e implementaremos distintos módulos que interconectados … formarán un monstruo como Geryon.

OBJETIVOS

El objetivo principal es el diseño, fabricación, programación e instalación de un sistema domótico completo para iluminación, control de persianas, alarmas técnicas y presencias.

PRESENTACION

Supongamos un caso práctico en el que vamos a encender una luz en función del estado de un interruptor. Una representación esquemática de este ejemplo con cableado convencional sería la Ilustración 1 y con cableado domótico sería la Ilustración 2.

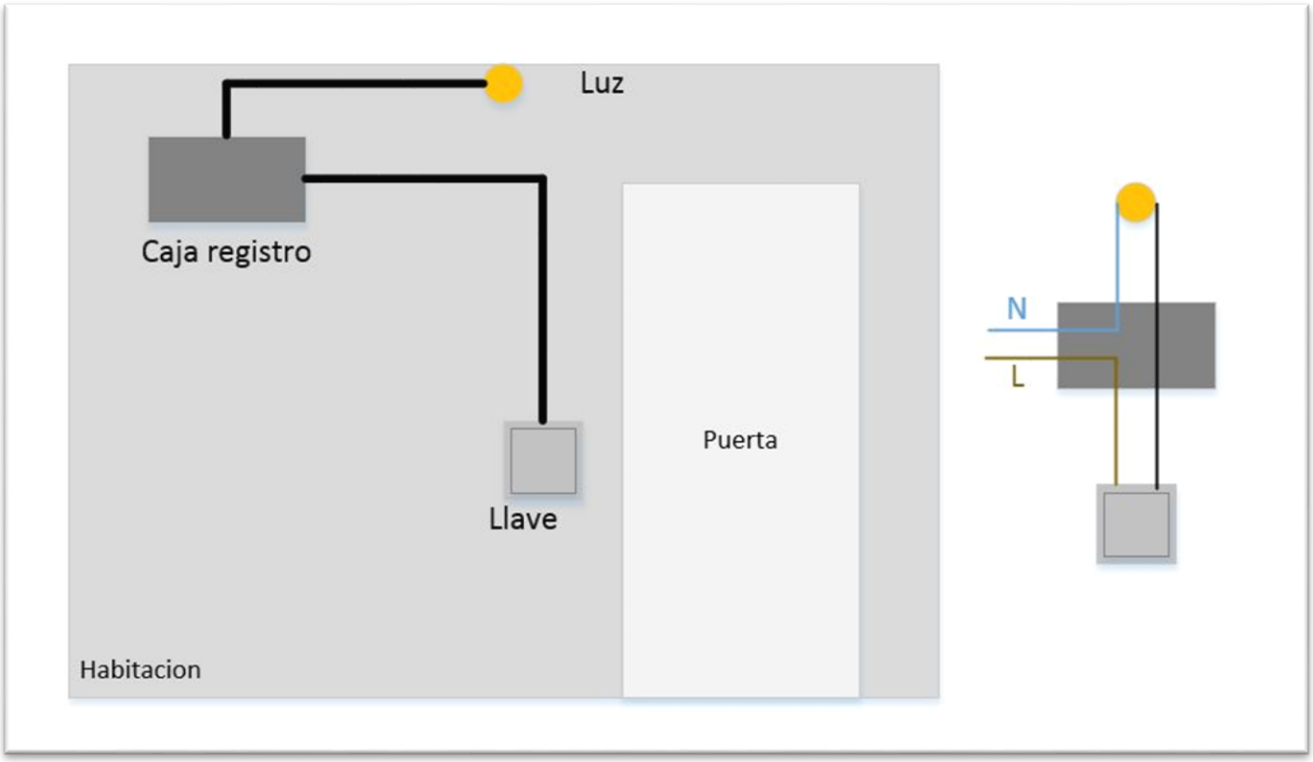


ILUSTRACIÓN 1. CABLEADO CONVENCIONAL

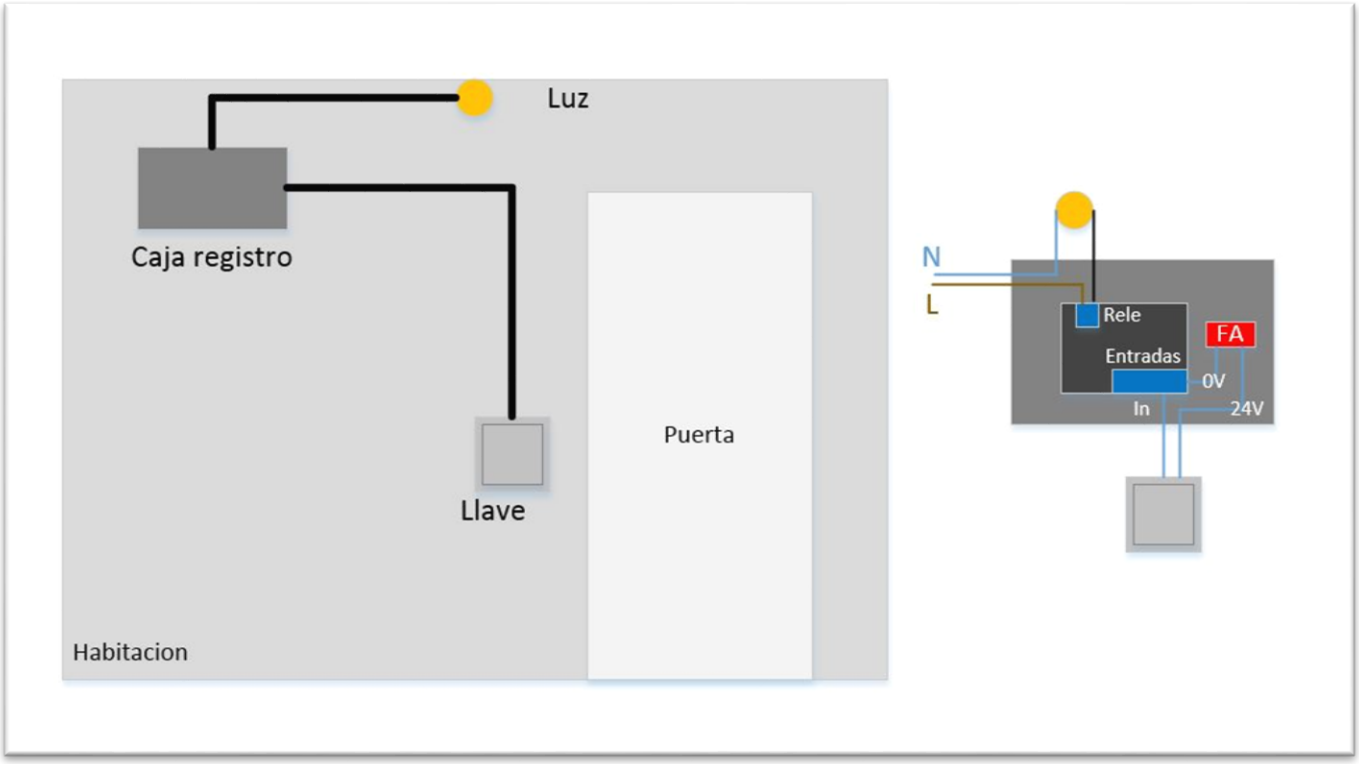


ILUSTRACIÓN 2. CABLEADO DOMÓTICO

Los distintos tipos de alimentaciones, organización de los módulos, topologías, buses, direccionamiento, etc, dan lugar a los distintos sistemas domóticos comerciales ya sea KNX, Lonworks, EIB, X10, …

Para algo tan simple, hasta un cableado convencional se complica dependiendo de si la llave es un interruptor o un pulsador, de si la luz debe ser con cruzamiento o conmutada o incluso temporizada, …

Un sistema domótico hace una separación entre cableado y funcionalidad, siendo el cableado el que se presenta en la Ilustración 2, la funcionalidad depende de la programación del sistema.

En mi caso, sirva como ejemplo, casi todas las llaves de mi casa son pulsadores. Antes de poner el sistema domótico no me quedo otra que montar telerruptores (relé biestable) para los puntos de luz. A medida que fui montando la instalación, podía ir quitando los telerruptores pues el sistema domótico se encargaba de realizar dicha función. En la ilustración 3 podéis ver como me fui deshaciendo de los telerruptores.

ILUSTRACIÓN 3. TELERRUPTORES

Son varios los motivo por el que cuento el tema de los telerruptores:

1.- No todo el mundo los conoce

2.- No todos los automatistas (mi profesión) son capaces de programarlos a la primera

3.- Son muy útiles

4.- Son fácilmente programables en un Arduino

5.- Aparecerán varias veces en el código más adelante

ARQUITECTURAS

SISTEMA

El sistema consta de (Ilustración 4):

- Módulo maestro, 8 entradas optoacopladas, 2 salidas a relé, sensor NTC, comunicación RS485 Modbus RTU, reloj RTC. Implementado en un PIC32-Pinguino.
- Varios módulos esclavos, 8 entradas optoacopladas, 2 o 4 salidas a relé, sensor NTC o DHTxx, comunicación RS485 Modbus RTU. Implementado en un PIC32-Pinguino y/o en un Arduino Uno.
- Un esclavo implementado en un Siemens S7-200, 14 entradas, 10 salidas relé.
- Pasarela RS485 Modbus RTU a Wifi MQTT implementado en un Nodemcu.
- Nivel N+1 con plataforma Home-Assistant implementado en una Raspberry-Pi.

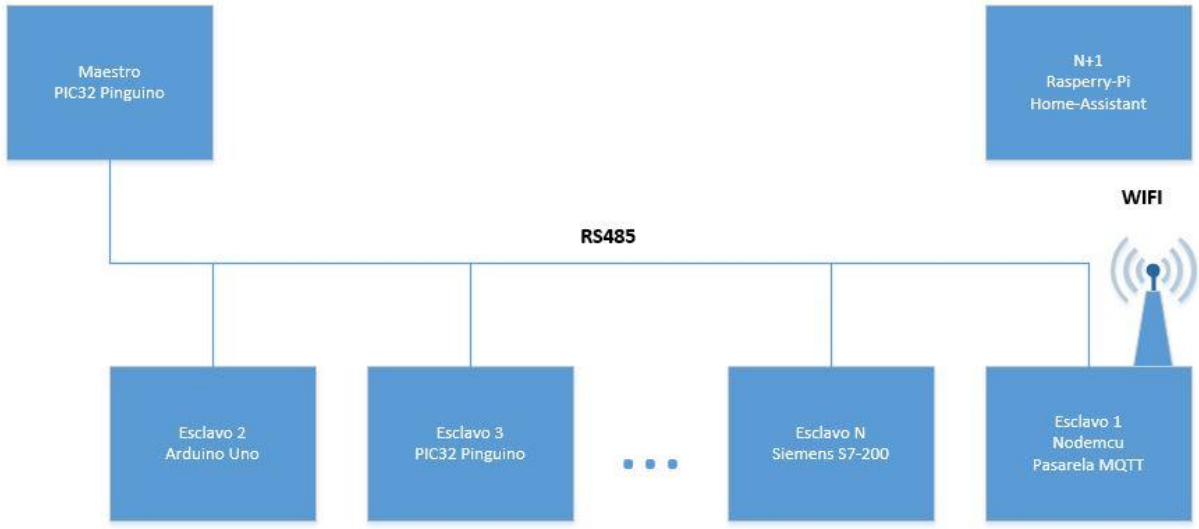


ILUSTRACIÓN 4. ARQUITECTURA DE RED

ESCLAVO

En la Ilustración 5, podemos ver cada uno de los módulos funcionales (MF-xx) que componen un esclavo.

- MF-UC: Módulo funcional – Unidad Central, puede ser un PIC32 Pingüino, Nodemcu, Arduino Uno, Arduino Nano, …
- MF-EO: Módulo funcional – Entradas optoacopladas, 8 canales
- MF-SxR: Módulo funcional – Salidas a relé de 2 o de 4 canales
- MF-COM: Módulo funcional – Comunicaciones
- MF-DA: Módulo funcional – Distribuidor de alimentación
- MF-Tx: Módulo funcional – Temperatura (según el sensor también humedad), no aparece en la ilustración pero a todas los esclavos se le ha implementado al menos sensor NTC para vigilar la temperatura del mismo.
- MF-EAC: Módulo funcional – de entradas AC, depende de si el esclavo tiene conectado detectores de presencia.

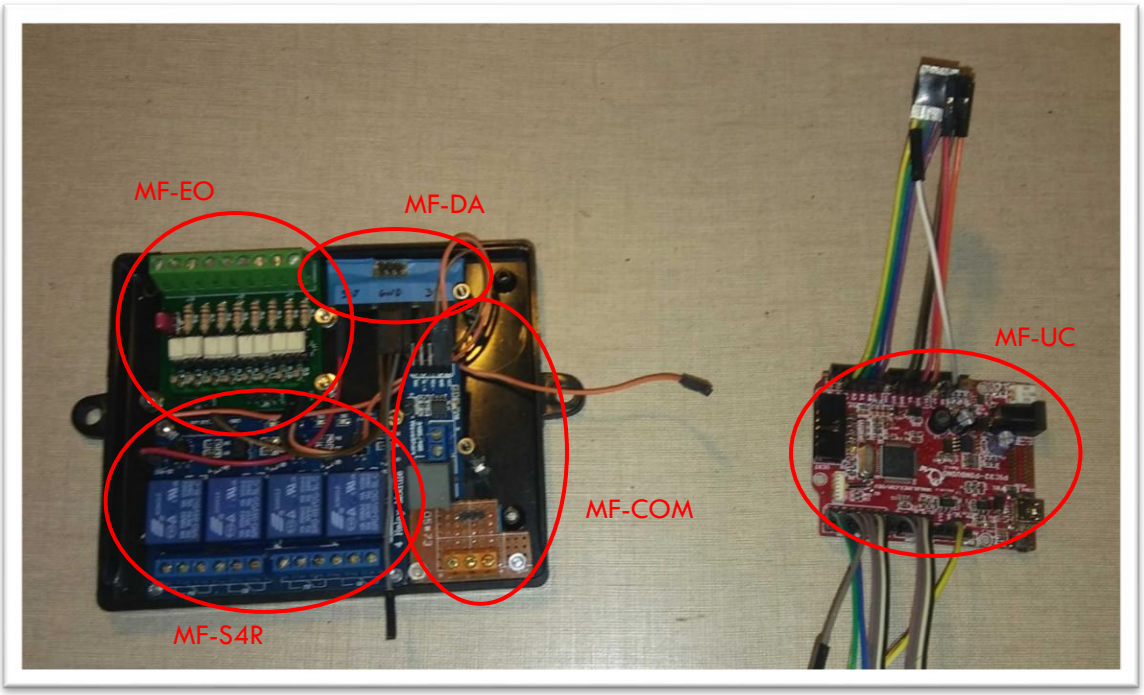


ILUSTRACIÓN 5. ARQUITECTURA DE UN ESCLAVO

MAESTRO

Idéntica a la de un esclavo.

PASARELA MQTT

Ver Ilustración 5.

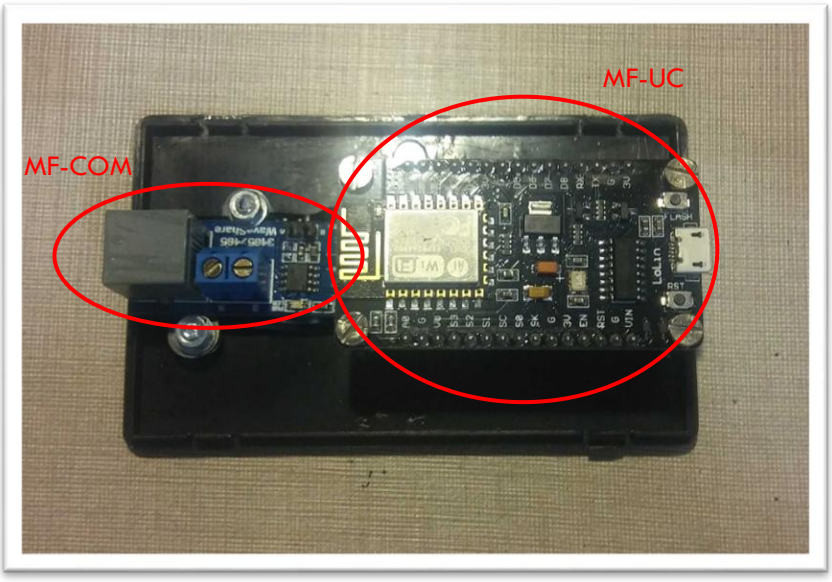


ILUSTRACIÓN 6. PASARELA MQTT

UNIDAD CENTRAL [MF-UC]

Encargada del procesamiento de las entradas, salidas y de las comunicaciones. Se realizaron pruebas tanto con un Pingüino como con un Arduino. La elección se realizó en base a motivos económicos, corría el 2015, por aquel entonces conseguí las placas Pinguinos a 7€ casi el mismo precio que costaba un Arduino Uno y las capacidades de procesamiento eran mayores.

A medida que el proyecto fue evolucionando y el número de esclavos creciendo me vi en la necesidad de implementar en un Arduino, resultando este a día de hoy igual de efectivo, más fácil de programar y más económico.

A continuación presento la asignación de pines:

Función	MF-xx	Pingüino	Arduino Uno	Nodemcu	Arduino Nano
COM – RX	MF-COM	D0	D0	RX-GPIO3	D1
COM – TX	MF-COM	D1	D1	TX-GPIO1	D0
COM – TX enabled	MF-COM	D13	D2	D8-GPIO15	D2
IN – IN0	MF-EO	D3	D3	D0-GPIO16	D3
IN – IN1	MF-EO	D4	D4	D1-GPIO5	D4
IN – IN2	MF-EO	D5	D5	D2-GPIO4	D5
IN – IN3	MF-EO	D6	D6	D3-GPIO0	D6
IN – IN4	MF-EO	D7	D7	D4-GPIO2	D7
IN – IN5	MF-EO	D8	D8	D5-GPIO14	D8
IN – IN6	MF-EO	D9	D9	D6-GPIO12	D9
IN – IN7	MF-EO	D10	D10	D7-GPIO13	D10
AIN – Ntc/DHTxx	MF-Tx	A6	A0	A0-ADC0	A0
OUT – OUT0	MF-SxR	A0	A1	SD2-GPIO9	A1
OUT – OUT1	MF-SxR	A1	A2	SD3-GPIO10	A2
OUT – OUT2	MF-S4R	A2	A3	NC	A3
OUT – OUT3	MF-S4R	A3	A4	NC	A4

ENTRADAS [MF-EO]

Después de buscar por todas las tiendas online un módulo "económico" que se adaptara a nuestras necesidades y no encontrarlo me decido a diseñar uno desde cero.

Para el diseño hemos utilizado optoacopladores en configuración no-inversora, con resistencia de pull-down. No vamos a explicar aquí el funcionamiento de un optoacoplador (hay un montón de información al respecto en la red). Si comentar que hemos realizado el diseño pensando que la placa sea reutilizable para otras aplicaciones, es por ello por lo que no hemos puestos led señalizadores de estados. Además la placa también es fácilmente adaptable a 4 canales.

Utilizando la herramienta Proteus se realizó la captura de esquemático (ver Ilustración 1) así como el ruteado de la placa (ver Ilustración 2). Se montó un prototipo en placa perforada y tras comprobar su correcto funcionamiento se mandó a fabricar las PCB a la empresa Itead. El servicio fue estupendo y la calidad totalmente profesional. Posteriormente se terminaron las placas soldándole los componentes. En la Ilustración 3 podemos ver el prototipo, la PCB y la placa terminada.

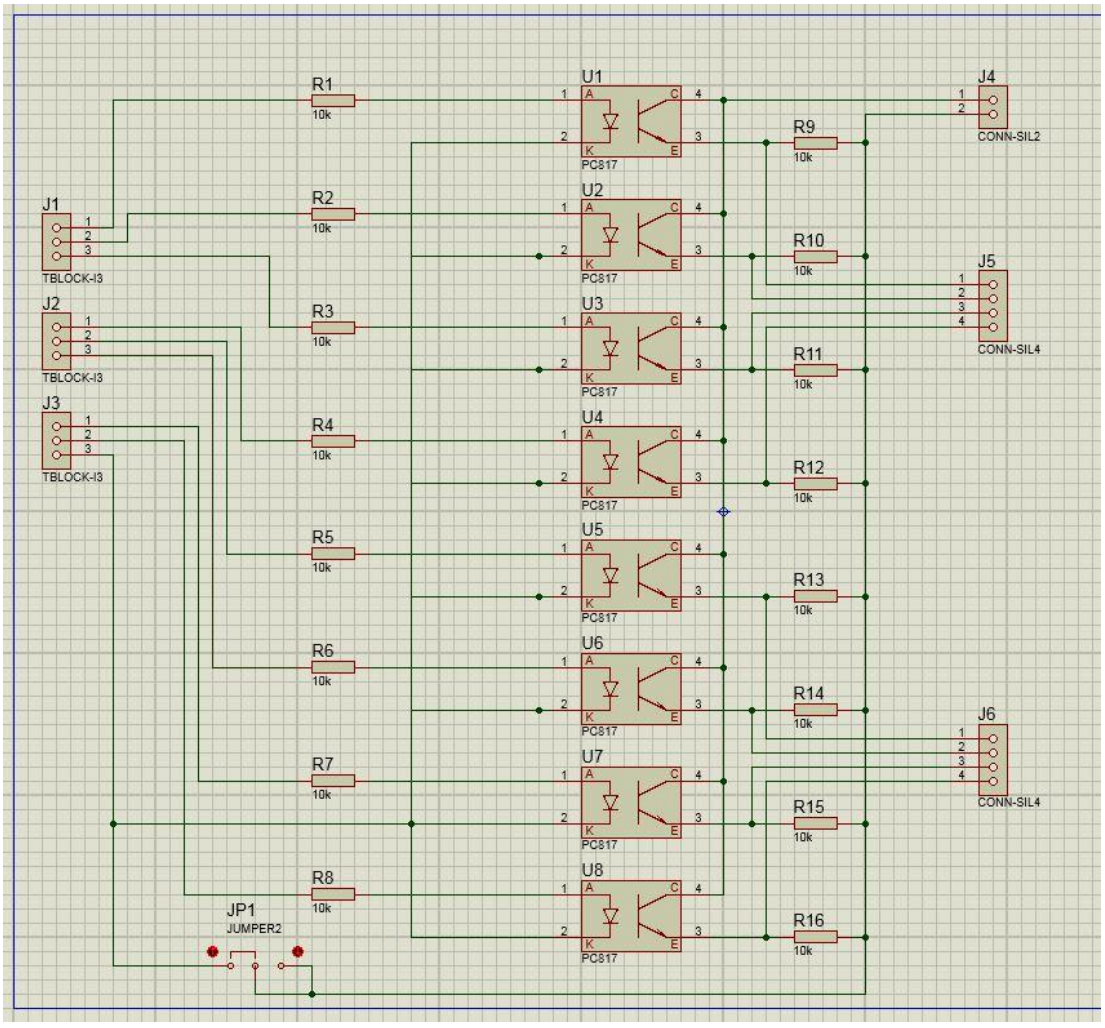


ILUSTRACIÓN 7. ESQUEMÁTICO MF-EO

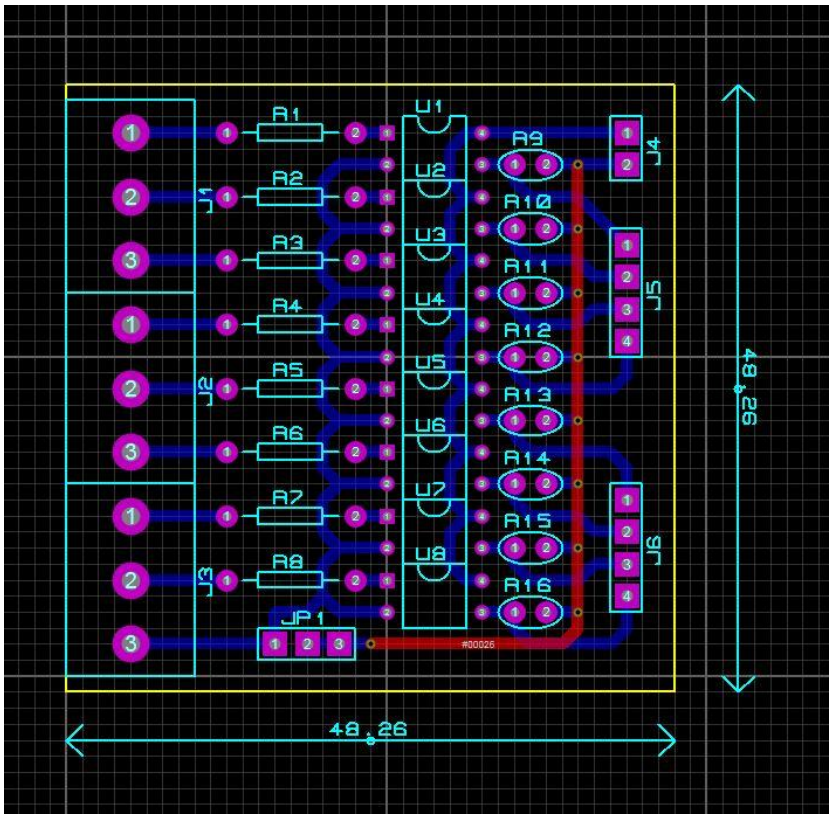


ILUSTRACIÓN 8. DISEÑO PCB MF-EO

Descripción:

J1 – Entradas 1-3

J2 – Entradas 4-6

J3 – Entradas 7-8 y COMUN

JP1 – Pos 1-2 para tierras unificadas, pos 2-3 para tierras aisladas

J4 – Alimentación, 1 Vcc, 2 Gnd

J5 – Entradas 1-4

J6 – Entradas 5-8

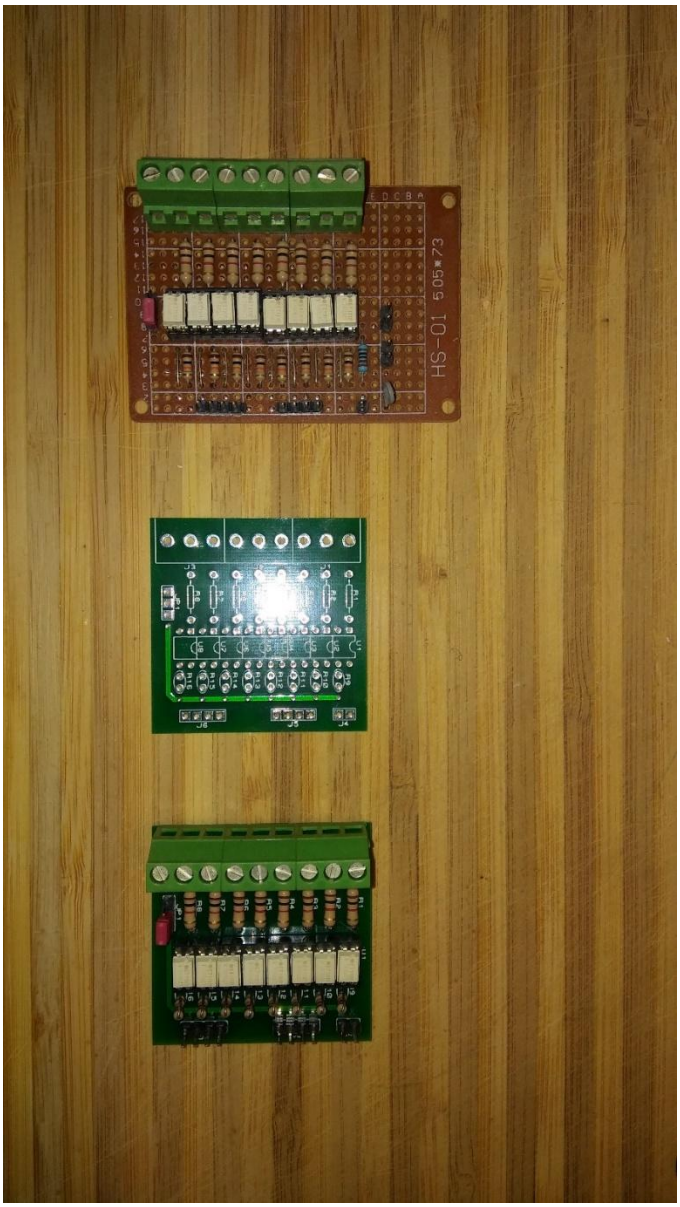


ILUSTRACIÓN 9. PROTOTIPO, PCB Y PLACA TERMINADA

CALCULOS

1. Elección del optoacoplador: Primero de todo, quería un ci que fuese conocido, con footprint estándar y fácilmente reemplazable. Por ello me decante por uno del tipo PC817 de 4 pines, aunque el que monte por motivos económicos fue el TCET1100 (Vishay).
2. Calculo de las resistencias de pull-down, valor de facto 10k
3. Calculo de la resistencia limitadora para TCET1100:
   Ve = I·R + Vled, Vled=1,2V y la I=20mA (máx 60mA)
   Para 24 vdc → R = (24-1,2) / 0,02 = 1140
   Para 12 vdc → R = (12-1,2) / 0,02 = 540
   Elegimos una **R=1K** y tenemos:
   para 24 vdc --> If = (24-1,2) / 1000 = 22,8mA
   para 12 vdc --> If = (12-1,2) / 1000 = 10,8mA

ENTRADA 240VAC [MF-EAC]

Para poder procesar una entrada de un detector de presencia comercial (PIR) hemos realizado un adaptador de 240VAC a 24VDC también con un opto-acoplador. Dicho circuito, ver Ilustración 4, también se puede usar para vigilar la presencia de tensión de red.

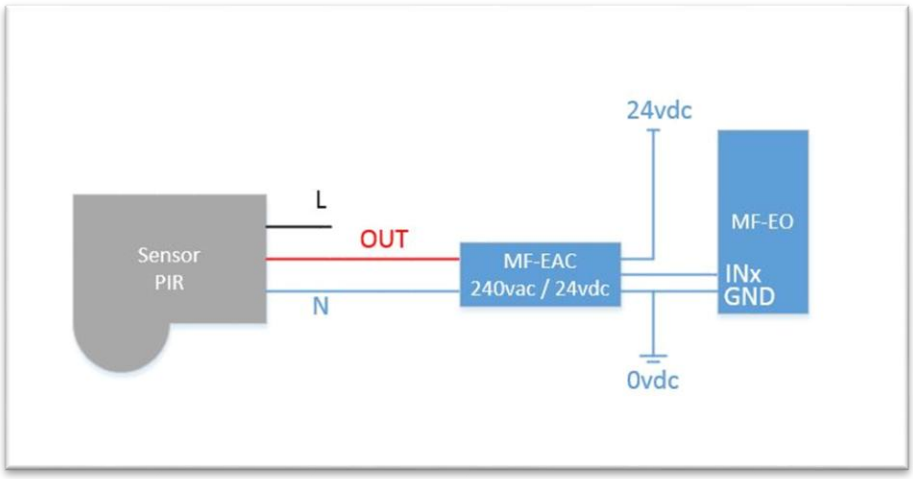


ILUSTRACIÓN 10. ESQUEMA DE CONEXIÓN PIR

A continuación el esquemático y los cálculos, ver Ilustración 5, está basado en uno de los muchos ejemplos que hay en la red para encender un led con tensión de 240VAC solo que en vez de encender un led vamos a polarizar el led de un opto-acoplador. Básicamente vamos a calcular una reactancia capacitiva para que por el led circulen aproximadamente 20 mA. El diodo rectificador D1 es necesario para evitar que en el semiciclo correspondiente el led del opto-acoplador quede en inversa. R1 sirve de protección para los picos de tensión.

Xc = 1 / (2 π f C). Fórmula de la reactancia capacitiva, donde f=50hz

I = V / Xc. Ley de Ohm para la reactancia capacitiva, despreciamos Vled

Para un valor comercial de C1 de 220nF 400V obtenemos una corriente de 16mA por el led.

Xc = 1 / (2 * 3.1415 * 50 * 220e-9) = 14.46 → I = 240 / 14460 = 16.59 mA

Potencia en R1 → $P = I^2 * R$ = 275 mW, la de 1/4W se queda corta así que ponemos de 1/2W

A la salida del opto-acoplador obtendríamos una onda cuadrada. Para el tratamiento de esta señal en Arduino tenemos 2 posibilidades:

1.- Software: Activamos un temporizador, durante 100ms contamos los flancos positivos de la señal, si el contador >=2 (debería valer 5) entonces hay presencia de tensión AC. Si contador <2 entonces no hay presencia de señal AC. Reiniciamos temporizador y volvemos a contar.

2.- Hardware: Añadimos un condensador a la salida para estabilizar la onda cuadrada. Para evitar los cálculos de carga/descarga del condensador realizo una simulación con el software NI Multisim.

2.1 Pruebo con 0.1uF y veo que apenas estabiliza, prueba y error y doy con un valor normalizado estándar de 10uF dejando una onda bastante lineal, como vamos a trabajar en 24VDC, aseguro un voltaje de trabajo del condensador adecuando y pongo un condensador electrolítico de 10uF 63V.

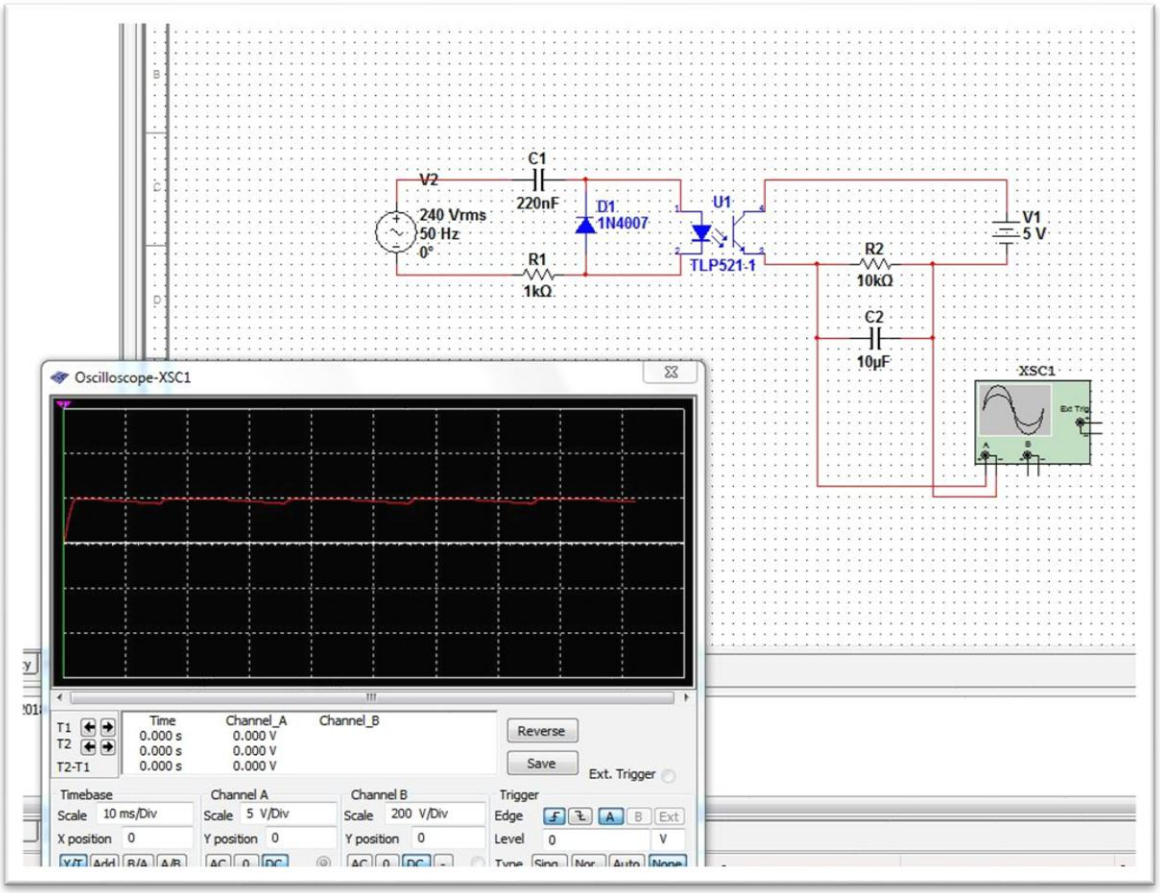


ILUSTRACIÓN 11. CONVERTIDOR 240VAC A 5VDC

2.2 Diseño del prototipo con el software Lochmaster, montaje antes de poner el condensador electrolítico y prueba con el osciloscopio DSO138, ver Ilustración 6. En la Ilustración 7 podemos ver el detalle del montaje del condensador.

De las 2 soluciones posible me decanto por la segunda pues la complejidad es sólo un condensador electrolítico a mayores y así simplifico el programa en el Arduino, además tenía 20 condensadores muertos de risa en mi stock.

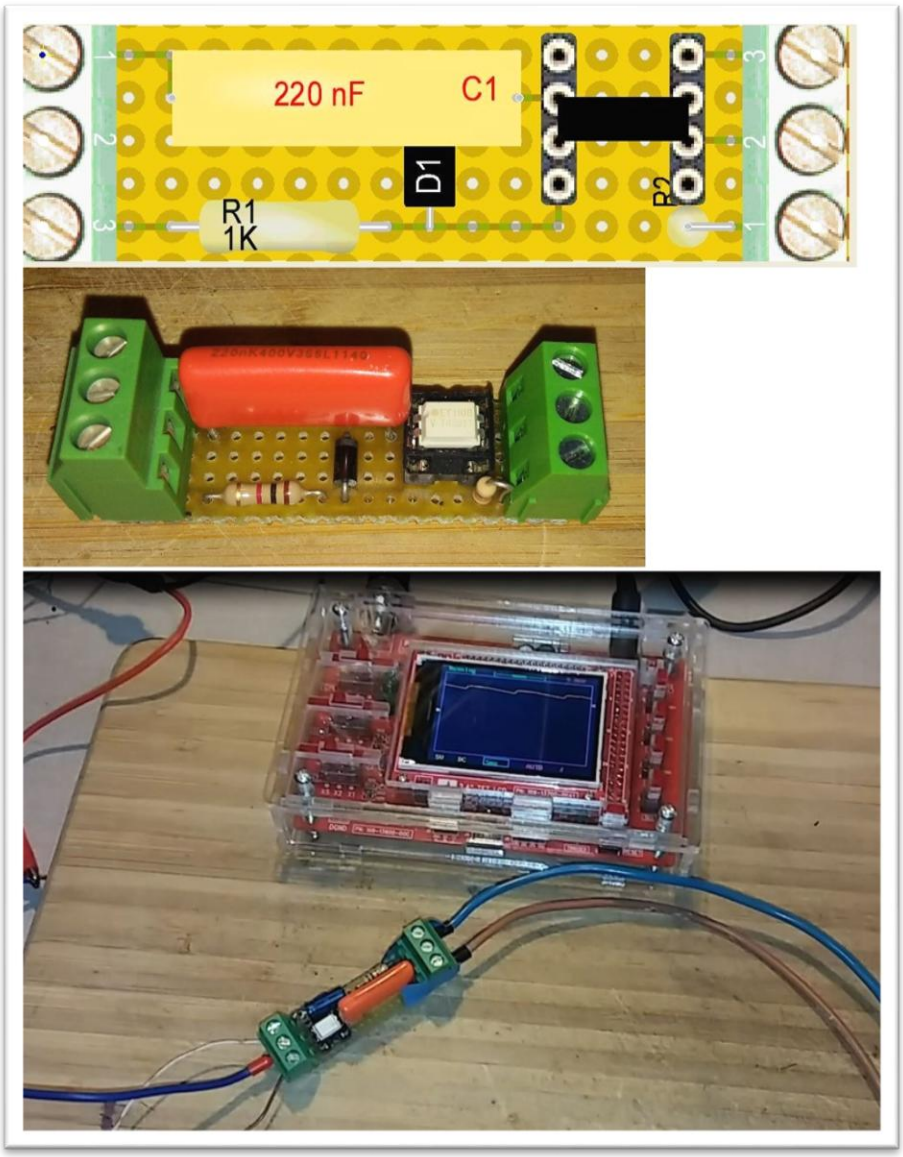


ILUSTRACIÓN 12. PROTOTIPO Y PRUEBAS

ILUSTRACIÓN 13. DETALLE MONTAJE CONDENSADOR

SALIDAS [MF-SxR]

Para la implementación de las salidas a relés hemos usados los módulos de 2 o 4 relés. Para la conexión con el pingüino hay que tener en cuenta que tenemos que eliminar el puente JD-VCC ya que el pingüino es una placa de 3V (ídem para los Arduinos de 3 voltios y el Nodemcu).

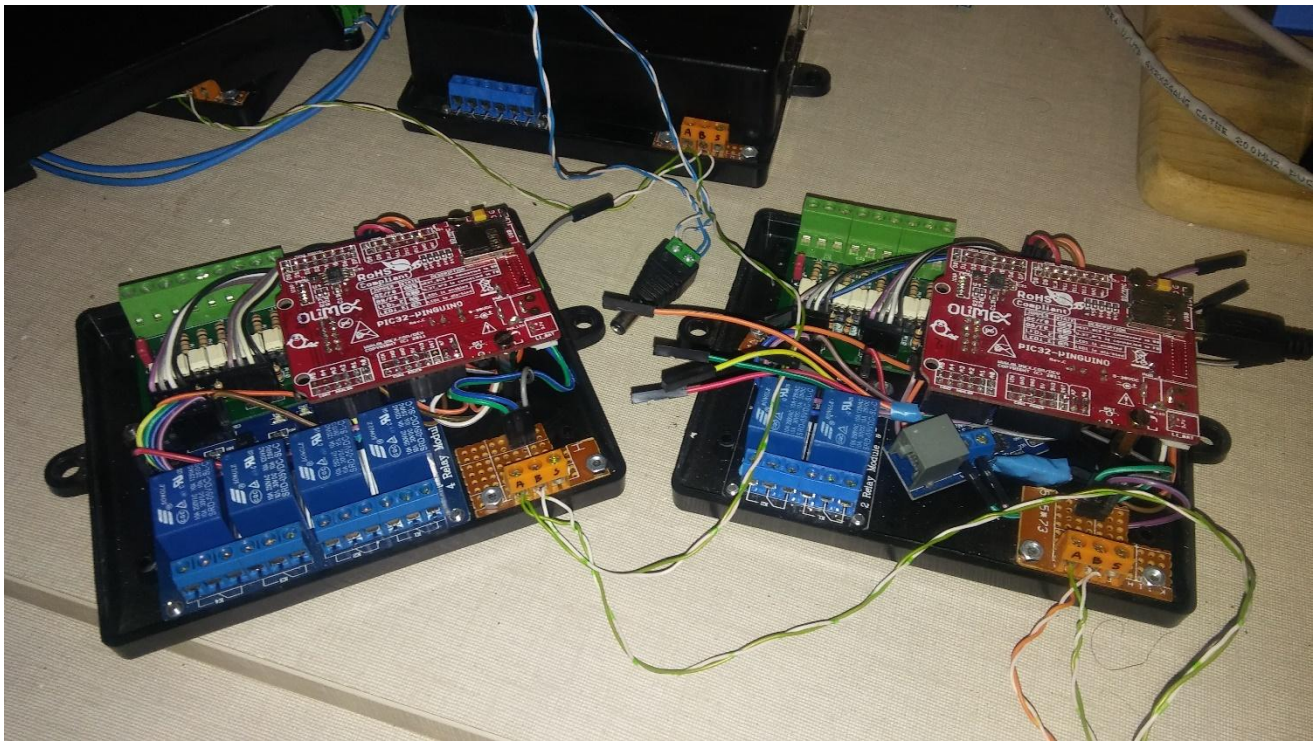


ILUSTRACIÓN 14. DETALLE MONTAJE MÓDULOS 2/4 RELÉS

COMUNICACIÓN [MF-COM]

El bus físico elegido para nuestro sistema domótico es el RS485. Hemos usado 2 módulos distintos, ver Ilustración 9, el superior tiene una alimentación de 5 vdc y el inferior de 3.3 vdc. Hemos usado el que correspondiese a la alimentación del MF-UC al que fuera conectado. Pero para un esclavo concreto PIC32-Pingüino tuve que conectarle un módulo de 5V porque se me agotaron los de 3.3V y he de decir que funciona sin problemas.

Como bien sabemos, el bus RS485 debe de llevar dos resistencias terminadoras en cada extremo del bus de 120R. Estos módulos vienen con la resistencia terminadora conectada, así que si no la removéis, no podréis hacer que la red funcione correctamente. Con un golpe seco de un destornillador de punta plana se retiran sin problemas. He resaltado las resistencias terminadoras, para el módulo superior es la R7, en el módulo inferior detalle de la resistencia removida. Para no complicarme, en mi caso, he removido la resistencia de todos los módulos y la añado de forma externa según me convenga.

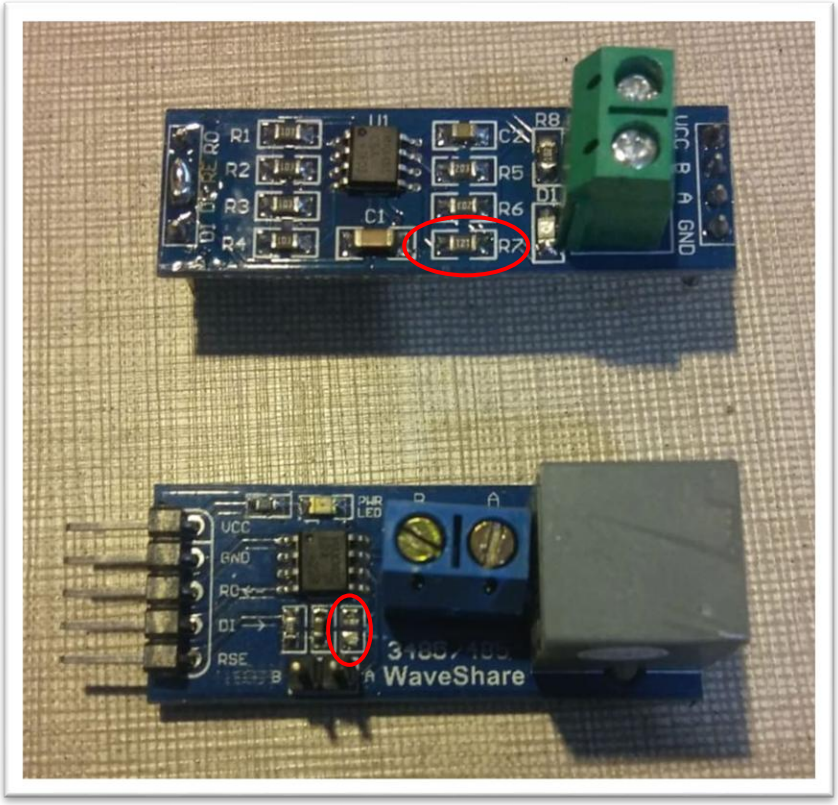


ILUSTRACIÓN 15. MÓDULOS RS485

Para la depuración de las tramas desde el PC he usado un convertidor RS485-USB, ver Ilustración 10.

ILUSTRACIÓN 16. CONVERTIDOR RS485-USB

La red se ha realizado con cable Ethernet UTP-Cat5, hemos usado el par Naranja(A) y Blanco-Naranja(B) y VERDE(A) y Blanco-Verde(B), los otros 2 pares los usamos para mandar tensión 24VDC. En la Ilustración 17 podemos ver un sinóptico de cableado. Usamos el par verde para el retorno así las resistencias terminadoras siempre las tenemos localizadas en el maestro. Sólo en los extremos el par verde está conectado al bornero del módulo RS485, en el resto es un paso a través.

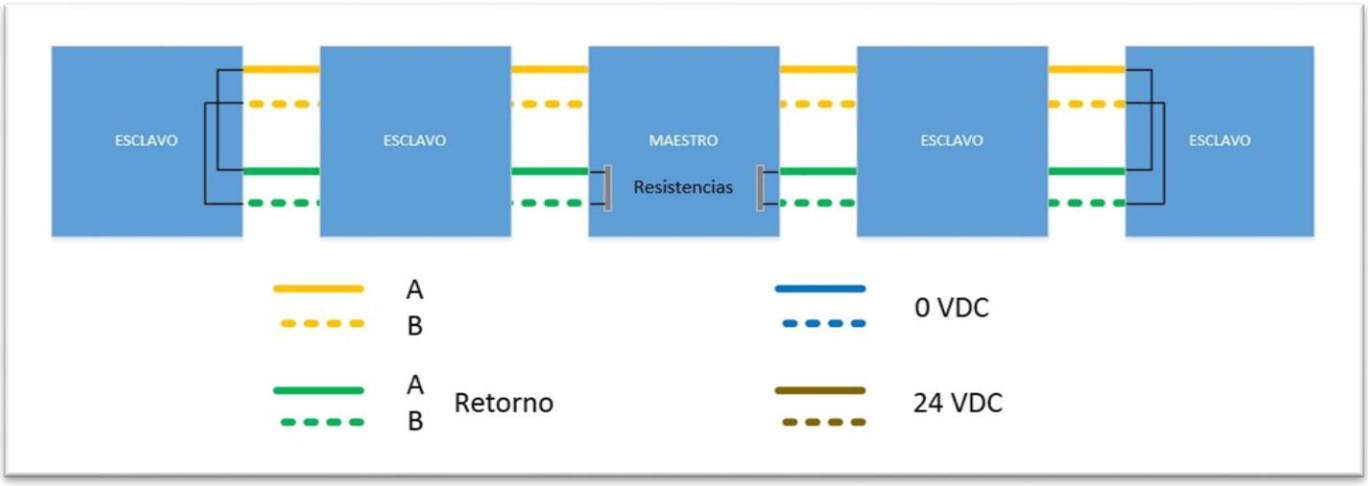


ILUSTRACIÓN 17. SINÓPTICO DE CABLEADO DE RED

Para facilitar la conexión del cable de red en los esclavos y debido a la geometría de los módulos MF-COM, decidimos dejar el módulo completamente dentro de la caja y exponer una pequeña plaquita con un bloque de terminales, ver Ilustración 18 y el detalle de la instalación en la Ilustración 8.

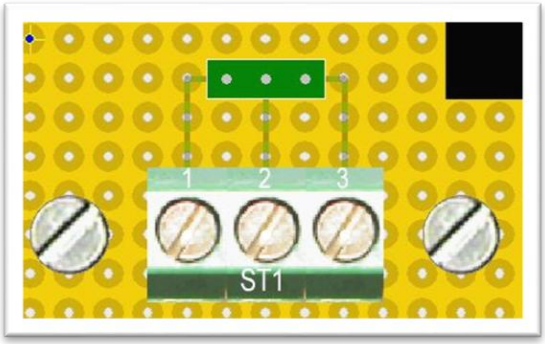


ILUSTRACIÓN 18. BORNERO MF-COM

En la Ilustración 19 podemos ver el diagrama de conexión de la red para mi caso particular. En la red hay 12 esclavos más el maestro. El maestro, el esclavo 1 y el esclavo 11 se encuentran ubicados en la localización ARM, un armario metálico de pared que tengo en la cochera, el resto de esclavos se encuentran cada uno en la caja de registro de la habitación que van a controlar.

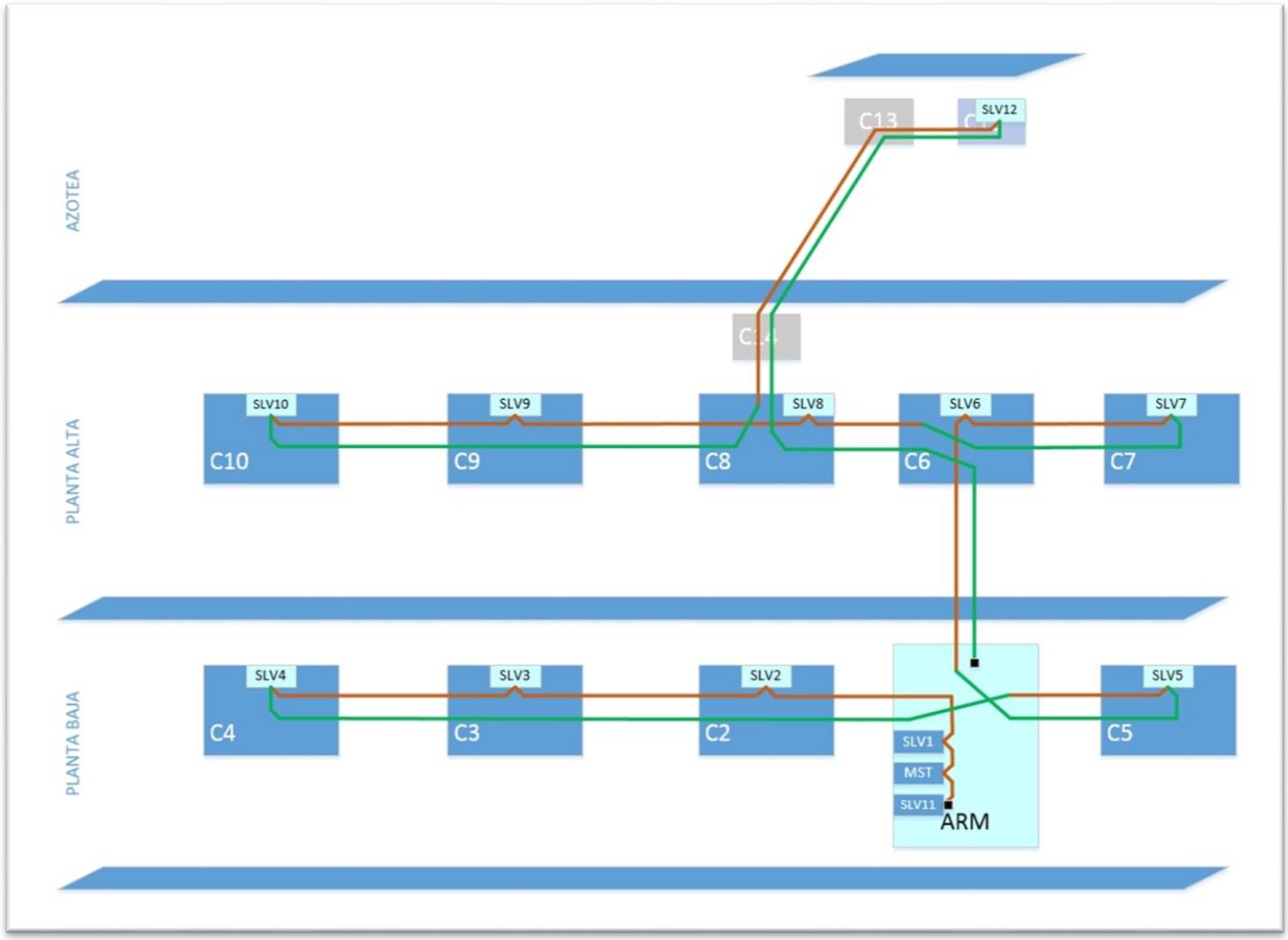


ILUSTRACIÓN 19. DIAGRAMA DE CONEXIONADO DE RED

La distancia desde la caja C4(C10) a la C5(C7) es de 15m y la altura entre plantas de 3m, así que la distancia total del bus extremo a extremo es de aproximadamente de 75m.

TEMPERATURA [MF-Tx]

Por un momento pensé en implementar en todos los esclavos un sensor de tipo DHTxx para poder leer tanto la temperatura como la humedad. Mi primera prueba con una NTC, puede verse dicha prueba en la placa prototipo de las entradas (ver Ilustración 3), fue un verdadero fracaso. Una vez avanzado el proyecto con el elevado número de esclavos vi absurdo tener tantos sensores DHTxx que además me producían una pérdida de ciclo de unos **5ms** por lectura influyéndome en las comunicaciones.

Por tanto decidí volver a la NTC e implementarla paso a paso. Se obtuvo un resultado funcional y correcto. Lo que más costo fue buscar las constantes correctas del sensor. Como conclusión, los esclavos pueden montar una NTC y medir la temperatura interna de la caja, o un sensor de tipo DHTxx y medir temperatura y humedad o no montar ninguno según convenga.

La conexión de un módulo DHT11, DHT22 o DHT21 no tiene ningún misterio. Conectamos positivo, negativo y señal, soportan 5 y 3.3 voltios y su programación está ampliamente extendida. En la Ilustración 19 podemos ver estos módulos listos para conectar.

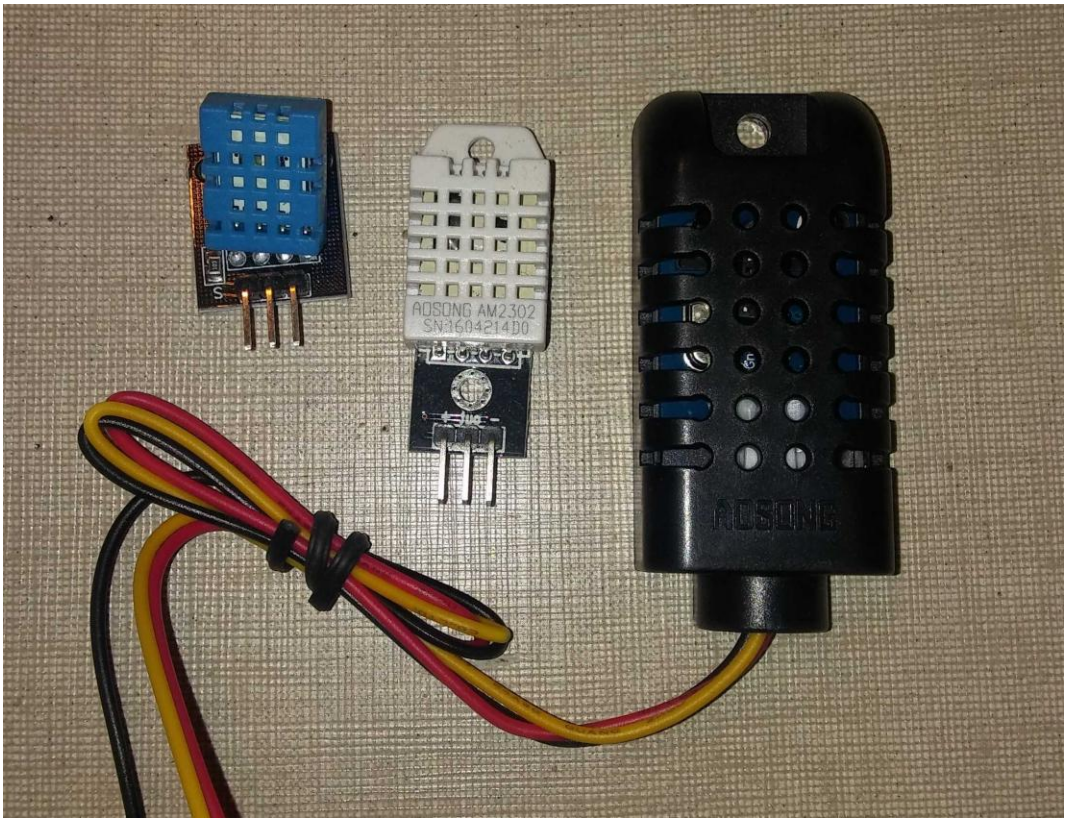


ILUSTRACIÓN 20. MÓDULOS DHTXX

La conexión de un sensor NTC si bien es muy simple, no lo es tanto su programación, pues para ello necesitamos una serie de constantes que no siempre son fáciles de encontrar y si no ponemos sus valores correctos obtenemos unos resultados que no son medidas válidas. Para la conexión de los sensores NTC decidimos crear una plaquita para que fuese reutilizable en otros proyectos y simulase la conexión de un sensor DHTxx de esa forma sería fácilmente intercambiable en los esclavos.

Podemos ver muchos ejemplos en la red, todos ellos basados en un divisor resistivo cuyo punto central conectamos a una entrada analógica. En la tabla de pin-out podemos ver a que entrada analógica se conectan

dependiendo del MF-UC elegido. La NTC elegida es TDK B57164K0103K000, de 10K, así que nuestro divisor resistivo tendrá otra resistencia fija de 10K de precisión.

A continuación presentamos la plaquita diseñada con Lochmaster (Ilustración 20) así como su implementación (Ilustración 21).

ILUSTRACIÓN 21. DISEÑO PLACA NTC

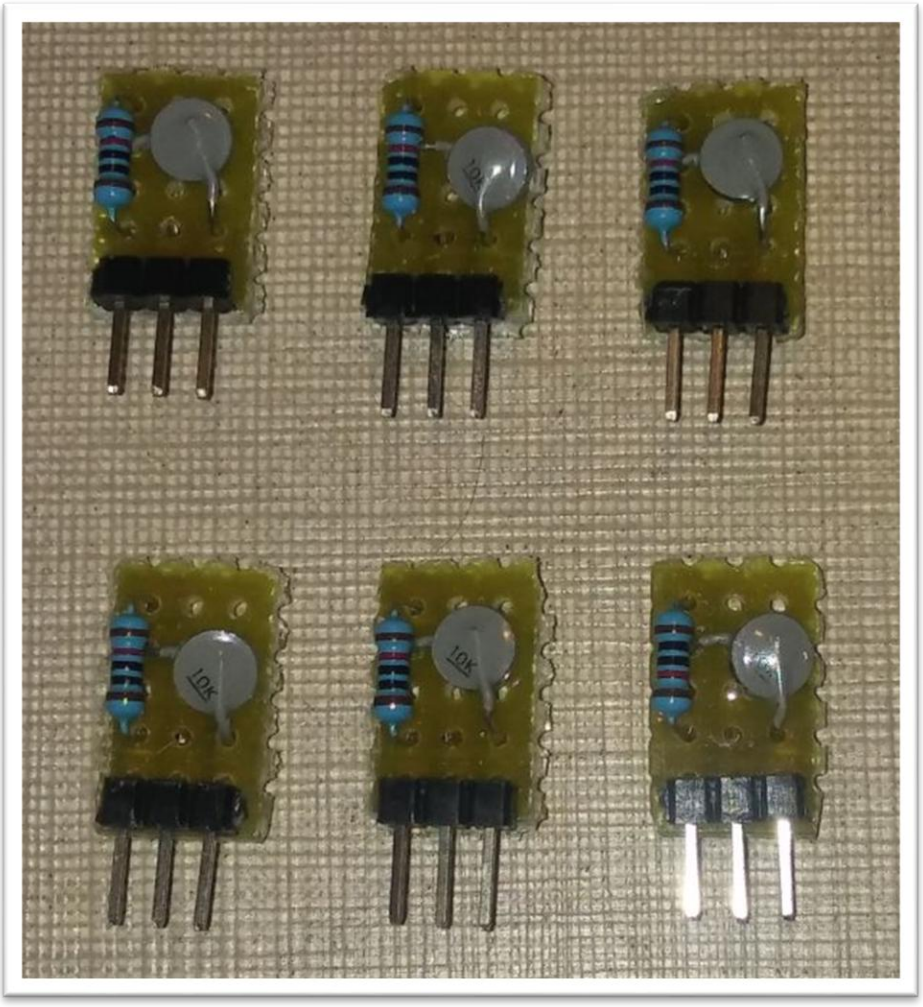


ILUSTRACIÓN 22. IMPLEMENTACIÓN PLACAS NTC

ALIMENTACIÓN [MF-DA]

Para los esclavos Pinguinos y el modulo maestro (también es pinguino), aprovechando que su regular integrado soporta alimentación de hasta 24vdc, hemos alimentado directamente a 24vdc por el Jack

Para los esclavos Arduinos y Nodemcu (pasarela MQTT) nos hemos decantado por una fuente externa del tipo cargador de teléfono móvil.

En el interior de las cajas hemos realizado un repartidor pasivo de 5V – GND – 3V para la alimentación de los distintos MF. En la Ilustración 22 podemos ver este distribuidor pasivo y su implementación en la Ilustración de la portada en la esquina superior derecha. En la Ilustración 8 en medio de las 2 cajas vemos el Jack de alimentación.

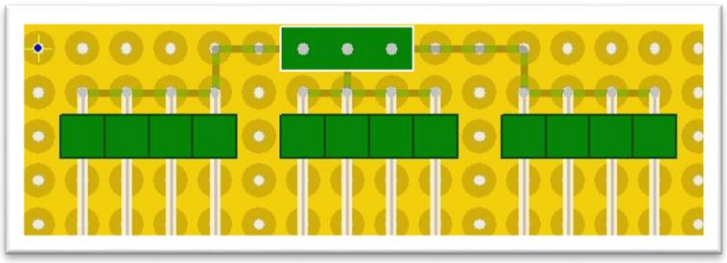


ILUSTRACIÓN 23. DISTRIBUIDOR PASIVO

IMPLEMENTACION MAESTRO

El maestro se ha implementado en una caja como cualquier esclavo más, en la Ilustración 8, la caja de la derecha es el maestro. Se le ha añadido 3 led 3mmm, buzzer y botones externos para Bootloader y Reset.

Función	MF-xx	Pingüino
COM – RX	MF-COM	D0
COM – TX	MF-COM	D1
BUT - Bootloader	MF-COM	D2
IN – IN0	MF-EO	D3
IN – IN1	MF-EO	D4
IN – IN2	MF-EO	D5
IN – IN3	MF-EO	D6
IN – IN4	MF-EO	D7
IN – IN5	MF-EO	D8
IN – IN6	MF-EO	D9
IN – IN7	MF-EO	D10
Reserva		D11
Reserva		D12
COM – TX enabled	MF-COM	D13
OUT – OUT0	MF-S2R	A0
OUT – OUT1	MF-S2R	A1
Led 3mm		A2
Led 3mm		A3
Led 3mm		A4
Buzzer		A5
AIN – Ntc/DHTxx	MF-Tx	A6
Reserva		A7

PROGRAMACION

Mapa de ficheros fuentes:

Fuente	Esclavo Pinguino	Esclavo Arduino	Mestro Pinguino	Pasarela Nodemcu
crono.h	X	X	X	X
dhtxx.h	X	X	X	
ntc.h	X	X	X	
mod_v08.h	X	X	X	X
slave_v08	slave_v08.pde	slave_v08.ino		slave_v08.ino
maestro_v08			Master_v08_cp.pde	

TEMPORIZADORES [crono.h]

En el fichero fuente “crono.h” tenemos una forma común de utilizar esperas no bloqueantes. Existen ligeras diferencias entre el código para el Pingüino y para el Arduino. Dependiendo a las funciones que utilicemos podemos usarlos con base de tiempo de ms o us. Podemos utilizar cuantos nos hagan falta en el código. Se ha integrado el control de desbordamiento del registro contador.

TEMPERATURA

DHTxx [dhtxx.h]

En el fichero fuente “dhtxx.h” tenemos agrupado el código para los sensores DHTxx.

NTC [ntc.h]

En el fichero fuente “ntc.h” tenemos agrupado el código para un sensor NTC. Especial atención a las constantes y variables globales que deben de ser personalizadas para cada tipo de NTC y MF-UC.

REGISTROS

Para entender fácilmente el código fuente de los archivos "mod_v08.h" antes de nada debemos presentar la estructura de memoria tanto de los esclavos como del maestro. Los datos aquí expuestos son un caso práctico real pero deben de ser personalizados para cada sistema domótico concreto.

REGISTROS DE UN ESCLAVO

El mapa de memoria de un esclavo se compone de 40 registros organizados en el mapping de la Ilustración 24.

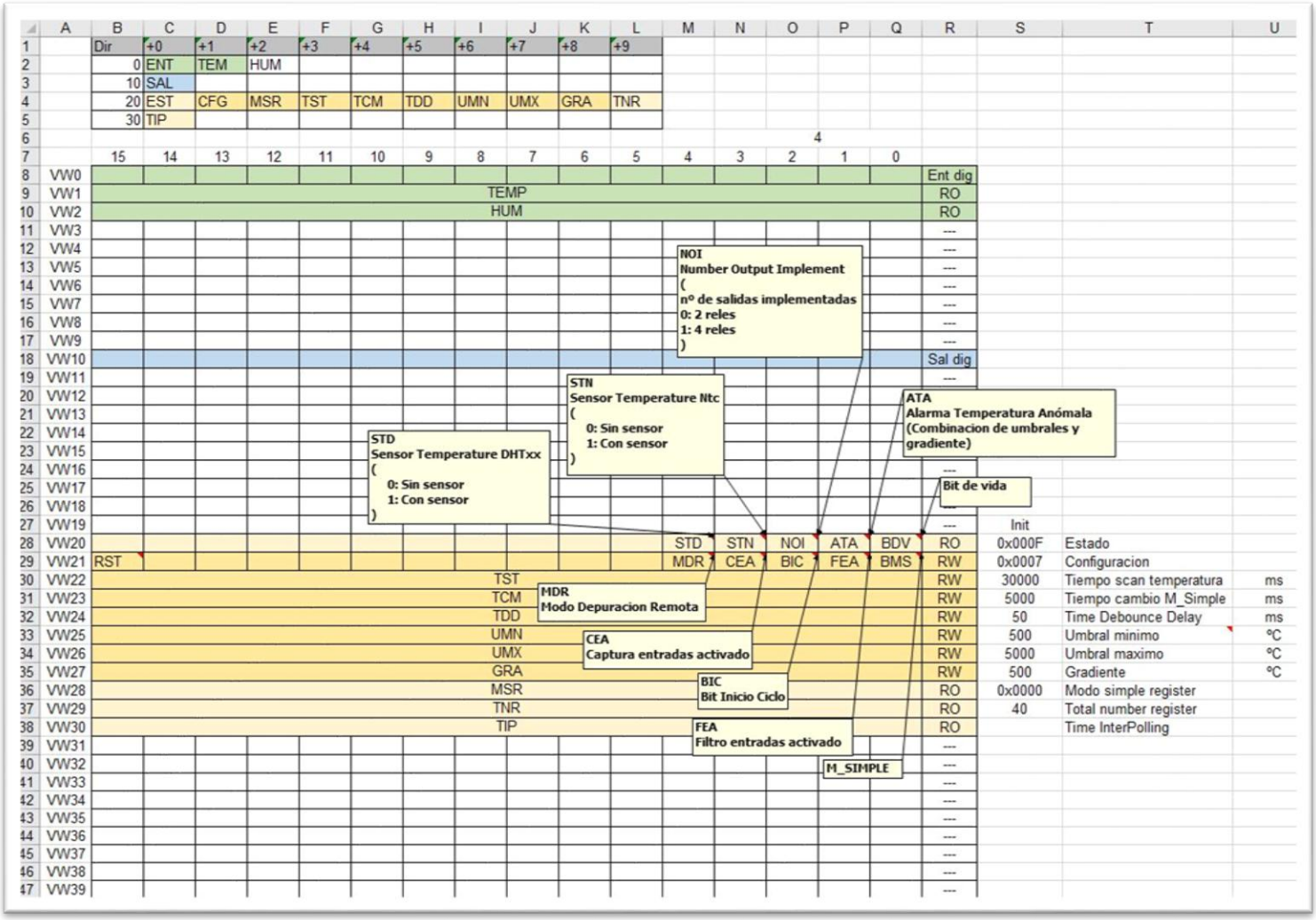


ILUSTRACIÓN 24. MAPPING ESCLAVO

Para cada bit de la palabra de entradas (VW0) y de la palabra de salida (VW10) podemos crear alias para simplificar el código. En el código del archivo "mod_v08.h" podemos ver todas estas definiciones así como unas funciones que nos ayudan a tratar con los bits de palabras de forma individual aplicando máscaras. Las definiciones de todos los esclavos están contenidas en el mismo fichero.

REGISTROS DEL MAESTRO

El mapa de memoria del maestro se compone de 4096 registros organizados en el mapping de la Ilustración 25. Mapping maestro 0-380

Dir	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	
0	CMD	VAL1	VAL2								
10	TOT	POL	RRC	TRE	RDT	TRC	TRS				
20	RDYM	RDDW	RTHM	RTSD	RCAL						
30	TcyMin	TcyMax	TcyLast	comTcyMin	comTcyMax	comTcyLast	TNP			MCY	
40	STA0	CFG0	TST0		TDD0	UMN0	UMX0	GRA0	MSR0	TNR0	
50											
60											
70											
80	ENT0	T0	H0								
90	SAL0										
100	ENT1_1	ENT1_2	ENT1_3								1
110	SAL1_1	SAL1_2	SAL1_3	Ii	Ti	Hi					
120	ENT2	T2									2
130	SAL2										
140	ENT3	T3	H3								3
150	SAL3										
160	ENT4	T4									4
170	SAL4										
180	ENT5	T5									5
190	SAL5										
200	ENT6	T6	H6								6
210	SAL6										
220	ENT7	T7									7
230	SAL7										
240	ENT8	T8									8
250	SAL8										
260	ENT9	T9	H9								9
270	SAL9										
280	ENT10	T10									10
290	SAL10										
300	ENT11										11
310	SAL11										
320	ENT12	T12	H12								12
330	SAL12										
340											13
350											
360											14
370											
380											15
390											

ILUSTRACIÓN 25. MAPPING MAESTRO 0-380

Al igual que con el esclavo, en el archivo “mod_v08.h” del maestro podemos ver todas estas definiciones.

REGISTROS DE LA PASARELA MQTT

El mapa de memoria de la pasarela (esclavo 1) se compone de 100 registros organizados en el mapping de la Ilustración 26.

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		Master
0		VKL16	VKL15	VKL14	VKL13	VKL12	VKL11	VKL10	VKL9	VKL8	VKL7	VKL6	VKL5	VKL4	VKL3	VKL2	VKL1	Ent dig	VW100
1									VKL25	VKL24	VKL23	VKL22	VKL21	VKL20	VKL19	VKL18	VKL17	Ent dig	VW101
2														VKM4	VKM3	VKM2	VKM1	Ent dig	VW102
3																		---	VW103
4																		---	VW104
5																		---	VW105
6																		---	VW106
7																		---	VW107
8																		---	VW108
9																		---	VW109
10		VSL16	VSL15	VSL14	VSL13	VSL12	VSL11	VSL10	VSL9	VSL8	VSL7	VSL6	VSL5	VSL4	VSL3	VSL2	VSL1	Sal dig	VW110
11									VSL25	VSL24	VSL23	VSL22	VSL21	VSL20	VSL19	VSL18	VSL17	Sal dig	VW111
12		ARI								VSP8	VSP7	VSP6	VSP5	VSP4	VSP3	VSP2	VSP1	Sal dig	VW112
13		Indice i																	VW113
14		Ti																	VW114
15		Hi																	VW115
16																			VW116
17																			VW117
18																			VW118
19																			VW119
20																			VW120
21																			
22																			
23																			
24																			
25																			
26																			
27																			
28																			
29																			
30																			
31																			
32																			
33																			
34																			

ILUSTRACIÓN 26. MAPPING PASARELA 0-34

Al igual que en los casos anteriores, en el archivo “mod_v08.h” de la pasarela podemos ver todas estas definiciones.

ESCLAVO

El esclavo dispone de 2 modos de funcionamiento, cuando tiene una comunicación sostenida con el maestro se comporta simplemente como un módulo RIO (Remote Input Output). Pero si la comunicación con el maestro se pierde durante un tiempo (configurable en los registros del esclavo) entonces el esclavo pasa a un modo especial (M_SIMPLE) en donde ejecuta un programa de control básico. En este programa básico sólo puede actuar sobre sus propias entradas/salidas como es lógico.

Supongamos que es de noche y estamos en el baño dándonos una ducha y se produce un fallo de comunicaciones o avería en el maestro que deja al módulo esclavo sin comunicación, entonces no va a funcionar la luz del baño, pero si nuestro esclavo implementa el M_SIMPLE, pasados unos segundos cambia a este modo y aunque puede ser que no disponga de toda la funcionalidad por lo menos puede hacer que la luz funcione.

Otro caso aún más común es que te dispongas a realizar alguna modificación en el maestro y mientras lo tengas desconectado, por lo menos se mantenga una funcionalidad básica en la vivienda.

MODBUS

La elección del protocolo Modbus se debe a dos principales razones, la primera es un protocolo abierto y la segunda es un estándar de facto, ver Wikipedia "https://es.wikipedia.org/wiki/Modbus".

La implementación del protocolo Modbus está basada en la librería "SimpleModbusSlave", https://github.com/angeloc/simplemodbusng con licencia GPL-3.0. Compila sin problemas también en Nodemcu.

Para la implementación en Pingüino también usamos la misma librería que fue reescrita he integrada en el entorno Pingüino Ide v11.

Modificaciones:

1.- Se han añadido 2 variables globales con sus correspondientes funciones observadores/modificadoras para poder saber desde el loop() cuando la librería a procesado una función 3 (lectura de registros) o una función 16 (escritura de registros) en el caso de Pingüino las variables se han añadido como campos en la estructura.

2.- Rutina de cálculo del CRC, pendiente de modificar en Arduino, modificada para el Pingüino. Para la rutina de cálculo del CRC disponemos de dos versiones.

> 2.1 La primera de ellas y más ampliamente utilizada es una rutina que se basa en hacer rotaciones y potencias aplicando el polinomio generador hasta dar con el CRC. Tiene como ventaja lo compacto del código (de agradecer para implementar en microcontroladores) y como desventaja su lentitud.
>
> 2.2 La segunda opción hace el cálculo del CRC con un algoritmo guiado por tablas. Es mucho más rápido que el anterior pero consume mucha más memoria. Su funcionamiento se describe al final del documento "MODBUS over serial line specification and implementation guide V1.02"

CODIGO

Se ha buscado un mismo código para todos los esclavos que compartan el mismo MF-UC. Al principio del código principal (.ino o .pde) existen unos #define que nos permiten personalizar la compilación tal y como se muestra en el siguiente fragmento de código:

```
//--------------------------------- CAMBIAR PARA CADA ESCLAVO -----------------
//Reles
#define x2RELE
//#define x4RELE
//Direccion del esclavo
#define SA 2
//Incluye Temperatura
#define DHTxx 0
#define NTC 1
//Recuerda actualizar en nth.h el SensorPIN = A0 y Vcc = 5.0
```

Para este ejemplo concreto, especificamos que el esclavo montará una placa de 2 Relés, que tendrá la dirección de esclavo 2 y que llevará un sensor de temperatura NTC. El último comentario nos recuerda actualizar las variables del fichero fuente ntc.h.

Para el tratamiento de las entradas digitales, se ha implementado un filtro para evitar rebotes y o ruido, por defecto está configurado a 50ms pero este ajuste se puede cambiar desde el maestro a través de los registros de configuración. Los siguientes fragmentos de código detallan este filtro.

```
//Filtro entradas
bool DIR[16]; //Digital Inputs Reading
bool DIS[16]; //Digital Inputs State (Trabajar siempre con estas)
bool DILS[16]; //Digital Inputs Last State
long LDT[16]; //Last Debounce Time

// …

void filterIn()
{
    int i;
    unsigned long m = millis();
    for (i = 0; i < 16; i++)
    {
        if (DIR[i] != DILS[i]) LDT[i] = m;
        if ((m - LDT[i]) > TDD) DIS[i] = DIR[i];
        DILS[i] = DIR[i];
    }
}
```

En el setup() configuramos los valores de los registros de configuración a valores por defecto, configuramos los pines, inicializamos la máquina de estados para el Modbus, iniciamos los cronómetros y activamos el Watchdog (para el Pingüino el watchdog no funciona, esta comentado). En el loop() realizamos las llamadas a las funciones:

```
void loop()
{
    leer_entradas();  //Tratamiento de entradas, filtrado, posibilidad de latch, temperatura, …
    comunicaciones(); //Actualizamos Modbus
    ssoo();           //Cambio a M_Simple, alarmas ATA, actualizamos cronometros, limpiamos WD
```

```
6.      m_simple(SA);     //Funcionalidad básica, para cada salida una función de activación
7.      escribir_salidas(); //Activación de las salidas
8.  }
```

Como podemos ver nos queda un código relativamente simple, compacto, y fácilmente adaptable a otros esclavos.

MAESTRO

MODBUS

La implementación del protocolo Modbus está basada en la misma librería que usa el esclavo, código correspondiente al maestro SimpleModbusMaster. Para la implementación en Pingüino esta librería fue reescrita para compilar en el entorno Pingüino Ide v11 porque no estaba portada.

Modificaciones:

1.- Se han añadido 2 variables globales con sus correspondientes funciones observadores/modificadoras para poder saber desde el loop() cuando la librería a procesado una función 3 (lectura de registros) o una función 16 (escritura de registros) en el caso de Pingüino las variables se han añadido como campos en la estructura.

2.- Rutina de cálculo del CRC guiada por tablas

CÓDIGO

PASARELA MQTT

En la Ilustración 27 podemos ver una estructura funcional de la pasarela:

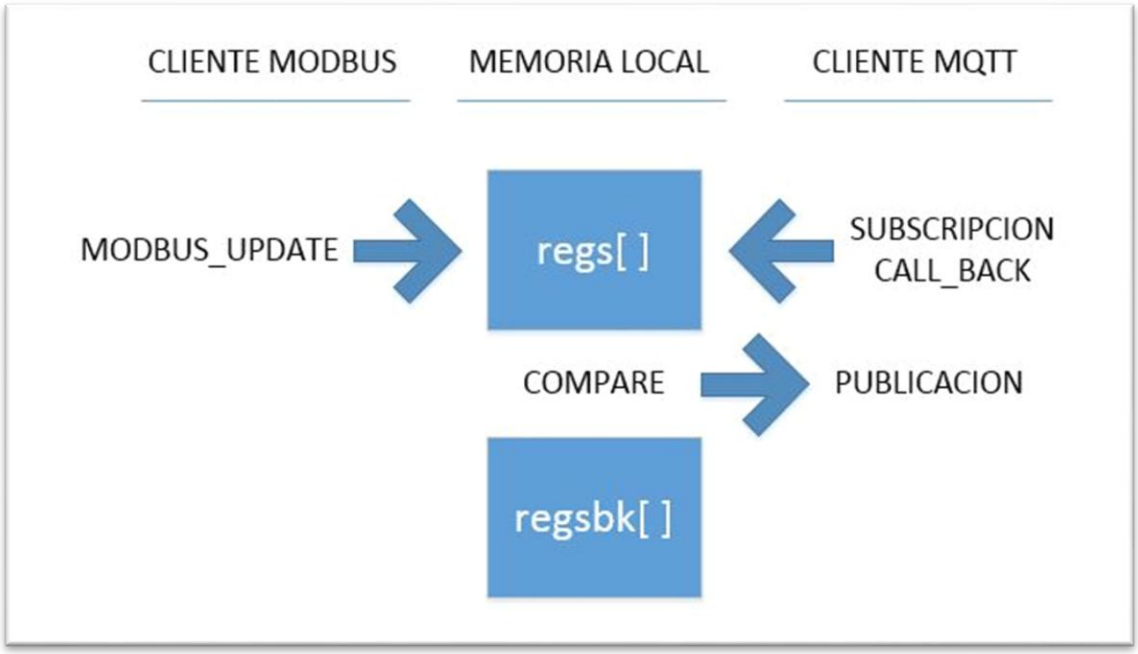


ILUSTRACIÓN 27. ESTRUCTURA FUNCIONAL PASARELA

MODBUS

Para la implementación Modbus usamos la misma librería que en el esclavo, de hecho es exactamente la misma que la de cualquier MF-UC implementado con Arduino pues la pasarela no deja de ser un esclavo más con dirección 1 y el entorno de compilación, IDE de Arduino, sería el mismo.

CODIGO MQTT

Al principio del código (.ino) existen unos #define que nos permiten personalizar los índices de los registros que vamos a utilizar para la publicación y la subscripción, tal y como se muestra en el siguiente fragmento de código:

```
24. // Define indices
25. #define IPI 10 //Indice de publicacion inicial
26. #define IPF 12 //Indice de publicacion final
27. #define ISI 0 //Indice de subscripcion inicial
28. #define ISF 2 //Indice de subscripcion final
29. //Define los mapas de memoria
30. unsigned int regs[TOTAL_NO_OF_REGISTERS];
31. unsigned int regsbk[TOTAL_NO_OF_REGISTERS];
```

Estos #define establecen que subconjunto de los registros locales usaremos para la publicación y la subscripción.

Las funciones principales son:

1.- void subscribe(); // Se ejecuta cada vez que tiene lugar una reconexión al bróker Mqtt

2.- void callback(char* topic, byte* payload, unsigned int length); // Se ejecuta cada vez que el broker nos informa de actualizaciones de los valores de las variables a las que estamos subscritos. Necesita de la función void desindexa(char* topic, unsigned int *a, unsigned char *b); // Calcula el índice de la variable de subscripción.

3.- void publica(); // Se ejecuta cada vez que se produce una actualización de las variables que nosotros como cliente publicamos en el broker. Para detectar el cambio de valor en las variables cuya actualización realiza la librería Modbus (función modbus_update()) realizamos una comparación entre el valor actual de las variables (regs[]) y el valor anterior que almacenamos en la tabla de registros regsbk[].

Para automatizar tanto la función de publicación como la función de subscripción es necesario que el nombre de las variables este normalizado de tal forma que podamos establecer una biyección entre el nombre de una variable y su índice.

Esta relación se establece en las tablas de tópicos definidas en el archivo “mod_08.h”. En estas tablas cada índice referencia al tópico:

char* mptt_sub[]={ … }	→	Tabla de tópicos subscritos
char* mptt_pub[]={ … }	→	Tabla de tópicos publicados
char* mptt_ptem[]={ … }	→	Tabla de tópicos de temperaturas publicados
char* mptt_phum[]={ … }	→	Tabla de tópicos de humedades publicados

CASO PRACTICO

HOME-ASSISTANT

DISEÑO MANTENIMIENTO

EVOLUCIONES

HERRAMIENTAS

SOFTWARE

DISEÑO Y SIMULACION

- PROTEUS 8.3 SP2
- NI MULTISIM 10.1
- LOCHMASTER 4.0

PROGRAMACION

- ARDUINO IDE 1.8.7
- PINGÜINO IDE v11

HARDWARE

- OSCILOSCOPIO DSO138
- FUENTE ALIMENTACION VARIABLE DIY
- MULTIMETRO
- SOLDADOR 30W
- DREMEL

MATERIALES

Caja negra

Caja estanca

Caja pasarela []

Ficha repartidora 4

Ficha repartidora de 8

Placa perforada [MF-COM, MF-Tx, MF-DA]

Puntera hueca

Jack [MF-UC]

Separadores hexagonales, tuerca y tornillos

Cables dupont

Bloque terminales [MF-EO, MF-COM]

Placa rele 2x [MF-SxR]

Placa Rele 4x [MF-SxR]

Modulo RS485 3V [MF-COM]

Modulo RS485 5V [MF-COM]

Ntc [MF-Tx]

DHTxx [MF-Tx]

Pines hembra x4 [MF-DA]

Jumpers [MF-EO]

Resistencias 1K 1/2W 5% [MF-EO, MF-EAC]

Resistencias 10K 1/4W 1% [MF-Tx]

Resistencias 10K 1/4W 5% [MF-EO, MF-EAC]

Condensador electrolítico 10Uf 63V [MF-EAC]

Condensador poliéster 220Nf 400V [MF-EAC]

Diodo 1N4007 [MF-EAC]

Zocalo 8 pines [MF-EO, MF-EAC]

Tira pines macho [MF-EO, MF-COM, MF-Tx, MF-DA]

Buzzer

Led 3mm

Resistencia xxx, 5%

PIC32 Pingüino USB OTG [MF-UC]

Arduino Uno R3 [MF-UC]

Nodemcu Lolin v3 [MF-UC]

…