



Máster Universitario en Software Libre

Automatización de instalaciones domóticas mediante PanStamp

*Proyecto Final de Máster
Desarrollo de aplicaciones de Software Libre*

Autor: Alberto Pelarda Royo

Consultor: Gregorio Robles Martínez

Tutor de Prácticas Externas: Oriol Palenzuela i Rosés

Fecha: Junio de 2015

Licencia



Este documento está sujeto a la licencia CC BY-SA 4.0 (Atribución-CompartirIgual 4.0 Internacional). El contenido completo de la licencia puede consultarse en:

<http://creativecommons.org/licenses/by-sa/4.0/legalcode>

Usted es libre para:

- *Compartir - copiar y redistribuir el material en cualquier medio o formato*
- *Adaptar - remezclar, transformar y crear a partir del material*

Para cualquier propósito, incluso comercialmente

El licenciante no puede revocar estas libertades en tanto usted siga los términos de la licencia

Bajo los siguientes términos:

	<i>Atribución - Usted debe darle crédito a esta obra de manera adecuada, proporcionando un enlace a la licencia, e indicando si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo del licenciante.</i>
	<i>CompartirIgual - Si usted mezcla, transforma o crea nuevo material a partir de esta obra, usted podrá distribuir su contribución siempre que utilice la misma licencia que la obra original.</i>

No hay restricciones adicionales — Usted no puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otros hacer cualquier uso permitido por la licencia.

Resumen

El presente proyecto se enmarca dentro de los estudios del Máster Universitario de Software Libre de la Universitat Oberta de Catalunya y se presentará como proyecto final del máster. El objetivo del proyecto será la creación de una serie de herramientas que permitirán la automatización de instalaciones domóticas mediante la tecnología panStamp, unos módulos inalámbricos de bajo consumo diseñados para control y telemetría.

El proyecto panStamp está basado en en las plataformas de software y hardware libres, disponiendo de una comunidad de usuarios detrás que se encargan de discutir los aspectos técnicos del proyecto, proponiendo y desarrollando mejoras y nuevas funcionalidades. Detrás del proyecto panStamp está una empresa cuya negocio consiste en comercializar tanto los módulos panStamp, como diversas tarjetas controladas mediante estos módulos.

El presente proyecto abarcará el estudio del funcionamiento actual del sistema y las redes de dispositivos implementadas mediante la tecnología panStamp, incluyendo los protocolos de comunicaciones utilizados y los requerimientos para la programación de nuevos dispositivos. A continuación se realizará un análisis para la aplicación de dicha tecnología a la automatización de viviendas, estudiando las desventajas del sistema actual e implementando las herramientas necesarias para creación de redes inalámbricas domesticas de forma rápida y fiable. Y finalmente se documentará todo el proceso de forma exhaustiva, publicando el proyecto a través de alguna comunidad virtual con el objetivo de compartir la información con la comunidad de usuarios de panStamp y que el proyecto pueda tener continuidad.

Abstract

This project is part of the master degree studies “Máster Universitario de Software Libre” in “Universitat Oberta de Catalunya” and will be presented as final degree project. The purpose of this project is the production of a procedure and the tools needed to automate the designing of domotic installations based on panStamp technology, autonomous low-power wireless modules made for telemetry and control projects

Everything around panStamp is open source, including hardware designs, libraries and software tools, with an important community of users who design new devices based in this technology, suggest upgrades and create new functionalities . PanStamp is also a company which sells the panStamp modules and different hardware controlled by panStamps.

This project will study how panStamp works and how to create wireless nets, including the communication protocols and requirements to create new devices. Next we will analyze the pros and cons of using panStamp devices in home automation and we will create the needed tools to design domotic installations in a quickly, reliably and simple way. Finally we will document the process and publish the project in a virtual community.

Tabla de contenidos:

Índice

Licencia.....	2
Resumen.....	3
Abstract.....	4
Tabla de contenidos:.....	5
Introducción.....	7
El proyecto panStamp.....	7
Automatización de viviendas.....	9
Motivaciones.....	10
Descripción del problema.....	11
Planificación.....	15
Tecnologías relacionadas.....	16
panStamp.....	16
Equipos prediseñados:.....	19
Programación de dispositivos.....	20
Protocolo de comunicaciones SWAP.....	20
Lagarto.....	23
Lagarto SWAP.....	24
Lagarto MAX.....	24
Node-red.....	26
Configuración de dispositivos en Lagarto.....	27
Automatización de instalaciones.....	28
Objetivos.....	28
Planificación de la instalación.....	30
Sketch Arduino.....	31
Definiciones preliminares.....	31
Hardware.....	32
Programación Arduino.....	35
Herramienta de generación de nodos.....	38
Ejemplo de proyecto de instalación domótica:.....	44
Definición de entradas y salidas.....	44
Definición de eventos.....	48
Automatización mediante node-red.....	55
Conclusiones.....	61
Objetivos concluidos.....	61
Mejoras.....	62
Evolución.....	63
Valoración personal del proyecto.....	65
Valoración personal del máster.....	66
Anexos.....	67
Esquema placa.....	67
Gerber.....	68

Introducción

El proyecto panStamp

Los dispositivos panStamp son circuitos de pequeñas dimensiones con un sistema de radiofrecuencia incorporado que trabaja en las frecuencias libres de 868-900-915 Mhz. Dichos circuitos están basados en un microcontrolador (Atmega328p o CC430F5137 dependiendo de la versión), integrando todo en un pequeño módulo compacto con formato DIP-24. Dispone de diversas entradas configurables como puertos de entrada/salida, puertos analógicos o funciones de PWM y, opcionalmente, también se pueden equipar con acelerómetros o sensores de temperatura, humedad o presión. Una de las principales ventajas de estos módulos es el bajo consumo, lo cual permite que, dependiendo de los periodos de transmisión utilizados, puedan durar varios meses alimentados únicamente a través de pilas convencionales.



Ilustración 1: Logo de panStamp

La programación de dichos dispositivos se realiza a partir de la plataforma de hardware libre Arduino y su entorno de desarrollo. En dicha plataforma se pueden incluir las librerías proporcionadas por el fabricante para el desarrollo de aplicaciones y permite la programación directa de los dispositivos. Además esta disponible una gran cantidad de información en Internet y una comunidad usuarios amplia que facilitan la creación de nuevos proyectos.

Las capacidades de comunicación están cubiertas mediante la implementación de un protocolo libre denominado SWAP que permite la creación de redes inalámbricas de bajo consumo. La idea principal del proyecto panStamp es reunir en un pequeño módulo las capacidades de computación y comunicaciones, de forma que el usuario solo deba preocuparse de la configuración de los dispositivos y de la electrónica necesaria para gestionar entradas, medir sensores externos o activar cargas. De esta forma los módulos panStamp pueden interactuar con cualquier otro módulo panStamp independientemente del desarrollador que haya creado la aplicación.



Ilustración 2: Modulo panStamp

El control de las redes inalámbricas implementadas mediante panStamp recae en dos aplicaciones implementadas en Python, Lagarto SWAP y Lagarto Max. Lagarto SWAP es una aplicación que permite la comunicación mediante el protocolo de comunicaciones SWAP entre los diferentes dispositivos de panStamp y el nodo central de la red. Lagarto Max es el encargado de administrar los eventos dentro de la propia red, recibiendo y procesando los datos recibidos, y enviando información a cualquier nodo de la red. Ambas aplicaciones cuentan con una interficie web que permite la configuración de las comunicaciones y gestionar las tareas a automatizar dentro de nuestra red inalámbrica.

No obstante, durante la realización del presente proyecto han habido cambios importantes en el diseño de Lagarto con la publicación de una nueva versión de la herramienta, con un nuevo diseño visual, y cambios en la gestión interna de los mensajes para la automatización de eventos. La aplicación Lagarto Max para automatización ha dejado de ser mantenida, optando por herramientas de terceros visualmente mas intuitivas y sencillas de implementar como node-red.

Automatización de viviendas

Existen diferentes proyectos panStamp en el campo de la automatización, y en el presente proyecto tratará de aplicar la tecnología actual al campo de la automatización de viviendas. La domótica es un conjunto de sistemas que permiten automatizar las instalaciones de una vivienda, como pueden ser la iluminación, los sistemas de climatización, alarmas o controles de acceso. Entre los beneficios que podemos encontrar en las instalaciones domóticas se pueden destacar los siguientes:

- Eficiencia energética: El hecho de tener un equipo controlando los diferentes parámetros energéticos de un vivienda permite aprovechar los recursos de forma mas eficiente. Por ejemplo, controlando la apertura y cierre de persianas para un mejor aprovechamiento de la luz solar o gestionando de forma inteligente las instalaciones de informatización.
- Gestión remota: Mediante el uso de la tecnologías de la información es posible gestionar una instalación sin la necesidad de estar presente en la vivienda. Por ejemplo, deshabilitando servicios desde un teléfono móvil si tenemos previsto regresar a la vivienda en un horario diferente al habitual.
- Seguridad: Tener un hogar domotizado aumenta la seguridad de las personas que habitan en él así como sus pertenencias. Por ejemplo lanzando avisos en caso de emergencias (incendios, fugas de agua, etc...), o permitiendo simular la presencia de personas en el hogar durante un viaje.
- Confort: La automatización de ciertos sistemas permiten facilitar la vida de las persona e implica que los usuarios deben de preocuparse menos por temas como la temperatura del hogar o el gasto en electricidad.

Dadas las características de los productos panStamp, se puede considerar que son una buena alternativa para la implementación de instalaciones domóticas inalámbricas. Por un lado, las capacidades de computación de los módulos son suficientes para las aplicaciones domóticas, y el bajo consumo con el que trabajan nos permite despreocuparnos de mantener un dispositivo conectado indefinidamente.

Ademas tiene la ventaja de ser software libre. Cada vivienda que se quiera domotizar tiene unas características y condicionamientos diferentes, con lo cual es posible que las aplicaciones comerciales tradicionales no cubran todas las necesidades, mientras que con panStamp, al estar basado en software libre, podremos estudiar y modificar el software para adaptarlo a las características concretas de nuestro proyecto. Y por ultimo, el hecho de ser inalámbrico nos proporciona una gran facilidad a la hora de implementar nuevas instalaciones y modificar o redimensionar instalaciones existentes sin la necesidad de modificar la instalación eléctrica o instalar nuevo cableado.

No obstante actualmente no existe una tecnología para panStamp con la que podamos realizar esta tarea de forma rápida y optimizando de los recursos disponibles por lo cual este proyecto intentará dar una solución de forma que se puedan implementar nuevas instalaciones o modificar las existentes de forma sencilla, intuitiva y económica.

Motivaciones

A nivel personal existen diversas motivaciones que me impulsaron a realizar los estudios del Máster Universitario de Software Libre y concluirlo con la realización de este proyecto, entre las que se podrían destacar las siguientes:

1. La filosofía tras los proyectos de software libre. El desarrollo del software libre promueve una serie de valores como la libertad de los usuarios a la hora de adaptar las herramientas informáticas a sus necesidades, el esfuerzo de estudiar un software con el único retorno del conocimiento adquirido, la constatación de como el trabajo en equipo puede sacar adelante proyectos enormes, y finalmente comprobar que otro tipo de desarrollos tecnológicos son posibles sin que el único objetivo sea el económico.
2. Las emergentes plataformas de desarrollo de hardware libre. En los últimos años, proyectos como Arduino y Raspberry Pi han ido incrementando el interés de los usuarios en los proyectos de hardware libre, cuyo desarrollo estaba algo retrasado en comparación con el software. Una gran plataforma de usuarios se encarga de realizar tutoriales y participando en foros para que cualquier persona pueda realizar sus propios proyectos. Además, la disponibilidad de empresas capaces de realizar placas de prototipos de buena calidad a precios reducidos y la mayor accesibilidad a la compra de componentes hace que el factor económico no sea tan importante como lo era hace pocos años.
3. El interés en la domótica y las diferentes tecnologías aplicadas a la automatización de viviendas que permiten dotar a los hogares de un mayor bienestar, seguridad, economía y conectividad.
4. El desafío de poner en práctica los conocimientos adquiridos durante la realización de los estudios del Máster Universitario de Software Libre realizando un proyecto que puede resultar de utilidad a una comunidad de usuarios.

A nivel empresarial, este proyecto comenzó como una colaboración entre la empresa OpenDomo y la Universitat Oberta de Catalunya, y consistía en la realización de un Plug-in que permitiera la interconexión entre los sistemas de panStamp y OpenDomo. No obstante, a raíz de cambios en la organización de OpenDomo se decidió modificar el proyecto para crear una nueva aplicación contando únicamente con la parte de panStamp, pero contando con la estimable ayuda de los tutores del proyecto.

Durante el proceso de selección del proyecto de final de máster escogí realizar el proyecto relacionado con panStamp ya que la estructura que propone la empresa me parece que puede convertirla en un proyecto de éxito dentro del software libre, con un producto base muy bueno y una comunidad de usuarios ya consolidada y bastante activa. Además se trata de una empresa local y que apuesta por productos innovadores demostrando que con trabajo y conocimiento se puede crear buenos productos.

Descripción del problema

El presente proyecto tratara de automatizar la tarea de instalación de sistemas domóticos. Obviamente cada vivienda o instalación a realizar será diferente en función de las características de la misma, así que para ilustrar el proceso de creación de la instalación nos basaremos en un caso de estudio practico. Para el desarrollo del proyecto tomaremos como ejemplo un pequeño apartamento ficticio sobre el que crearemos la instalación. Se trata de un pequeño apartamento vacacional al cual aplicaremos los cambios necesarios para convertirlo en una vivienda mas económica, segura y cómoda. El plano de la vivienda a gestionar sería el siguiente:

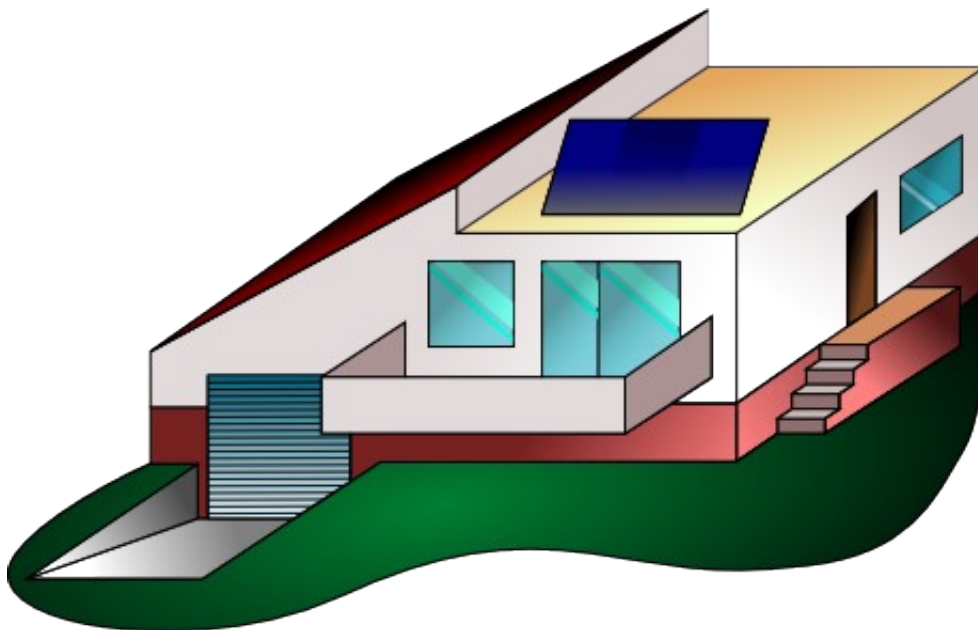


Ilustración 3: Caso de estudio

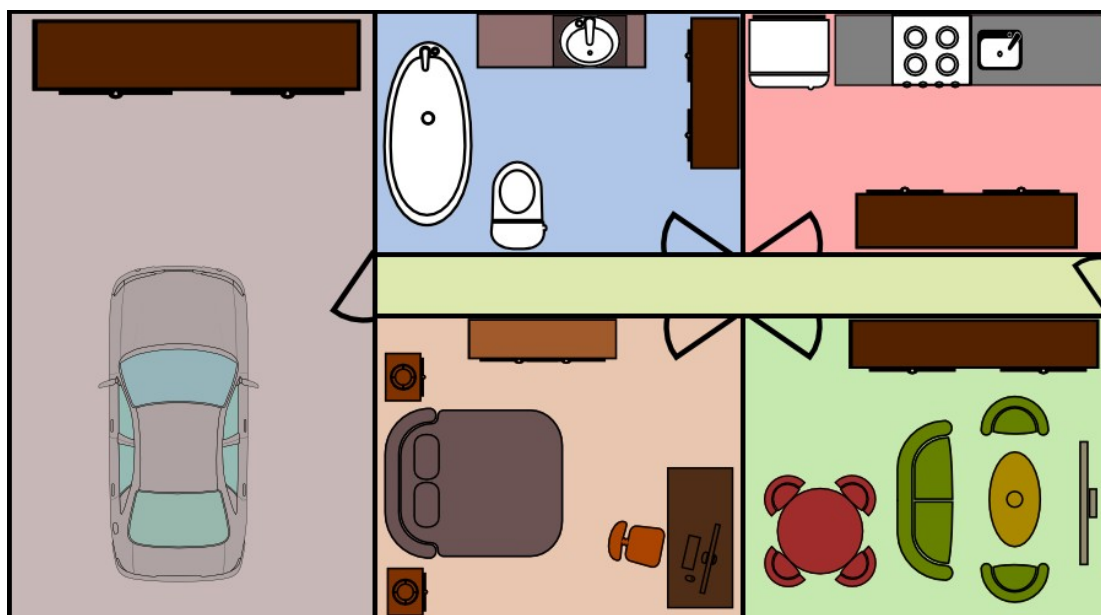


Ilustración 4: Plano de la vivienda

La vivienda consta de las siguientes estancias: una cocina, un comedor, un cuarto de baño, una habitación principal, un parking y un pasillo distribuidor. Dentro de cada estancia podemos encontrar diversos sistemas con los que podemos interaccionar, y que serán los elementos de nuestra instalación domótica.


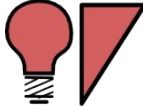
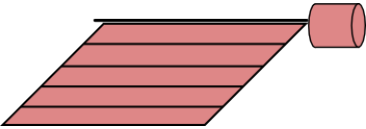
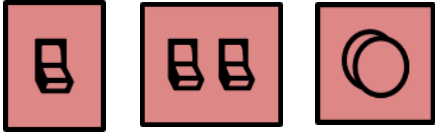




Elemento	Descripción
	<p>Lamparas: Como en cualquier instalación eléctrica contaremos con diferentes sistemas de iluminación en las diferentes estancias.</p>
	<p>Lamparas con control de iluminación regulables: Para un mejor confort en el comedor y la habitación principal la iluminación en estas estancias sera regulable.</p>
	<p>Persianas motorizadas: La motorización de las persianas nos permitirá una mayor comodidad ademas de poder programar el cierre y apertura de las mismas dependiendo del horario o de las condiciones del día.</p>
	<p>Interruptores: El control de los diferentes elementos de la vivienda se realizará a partir de interruptores convencionales que pueden ser de diferentes tipos según su función: interruptores para las lamparas, pulsadores dobles (up/down) de las persianas, reguladores de intensidad, etc...</p>
	<p>Tomas de corriente: Las tomas de corriente también estarán gestionadas por nuestro sistema, permitiendo que en casos en que la vivienda este desocupada podamos apagar algunos de los sistemas permitiendo ahorrar el consumo de los diferentes productos en stand-by. Obviamente existirán algunas tomas de corriente que no estarán controladas por nuestra red al tener que estar continuamente activadas (por ejemplo el frigorífico)</p>
	<p>Detectores de presencia: Los detectores de presencia nos permiten automatizar el encendido de lamparas en ciertas estancias y actuar como alarma anti-robo en caso de que no haya nadie en la vivienda.</p>
	<p>Alarma de humo: Permitirán detectar posibles incendios y avisar remotamente al propietario de la vivienda en caso de emergencia.</p>
	<p>Sensores de luz solar: mediante este sensor podremos determinar si es necesario abrir o cerrar persianas automáticamente en función de la luz solar.</p>

Tabla 1: Tabla de elementos a controlar

Sobre el plano de la vivienda vemos que los diferentes elementos de nuestra instalación quedarán distribuidos de la siguiente forma:

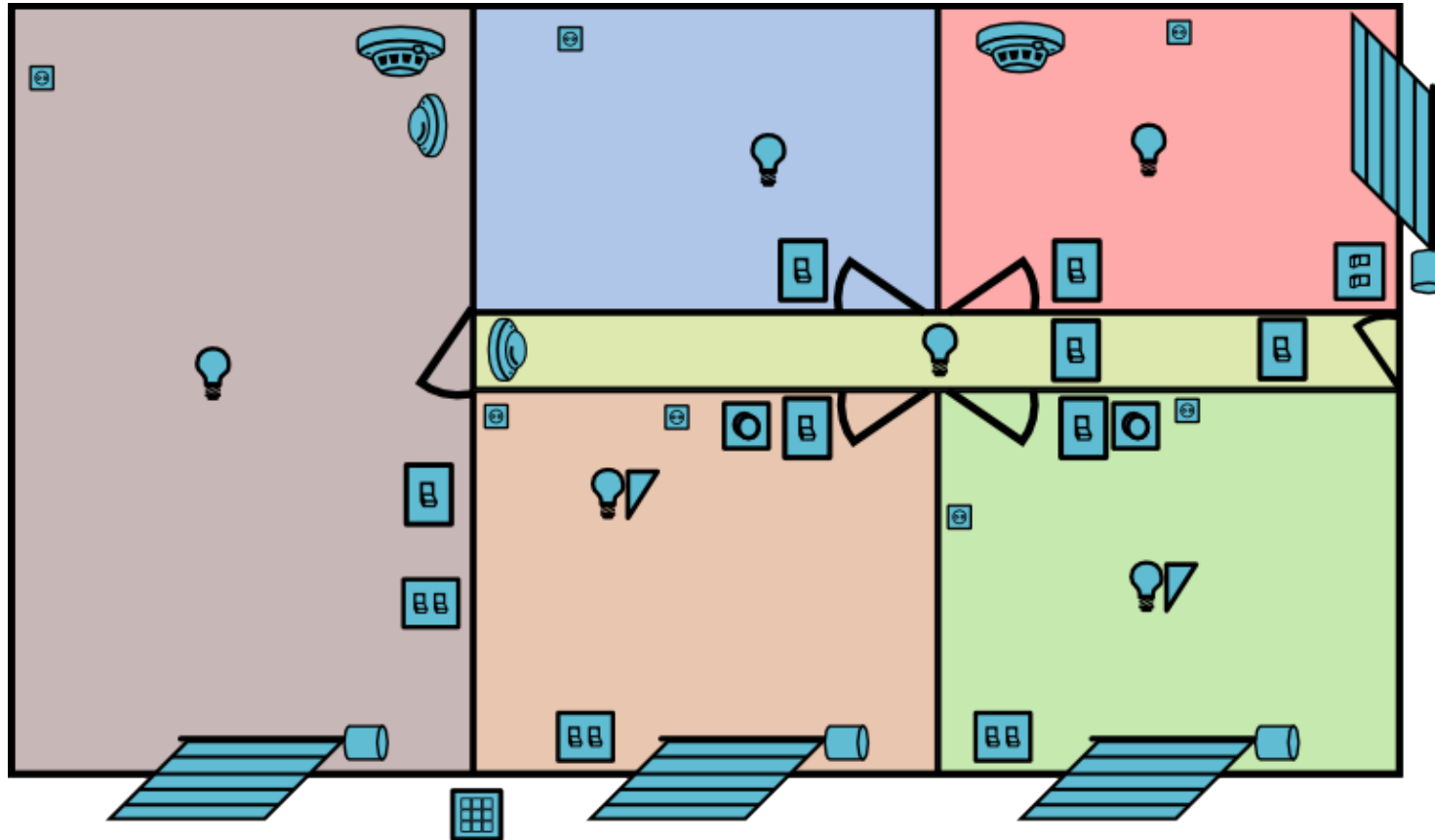


Ilustración 5: Plano con elementos de la instalación

Estancia	Elementos
Cocina	- 1 Lámpara - 1 Interruptor - 1 Detector de humo - 1 Toma de corriente - 1 Persiana + Interruptor doble
Comedor	- 1 Lámpara con control de intensidad + 1 Interruptor + 1 Regulador de luz - 1 Toma de corriente - 1 Persiana + Interruptor doble
Cuarto de baño	- 1 Lámpara - 1 Interruptor - 1 Toma de corriente
Habitación principal	- 1 Lámpara con control de intensidad + 1 Interruptor + 1 Regulador de luz - 1 Toma de corriente - 1 Persiana + Interruptor doble - 1 Sensores de luz solar
Parking	- 1 Lámpara - 1 Interruptor - 1 Detector de humo - 1 Toma de corriente - 1 Puerta de parking + Interruptor doble - 1 Detectores de Presencia
Pasillo	- 1 Lámpara - 2 Interruptores - 1 Detectores de Presencia

Tabla 2: Listado de elementos en la vivienda

Como podemos ver en la lista de elementos nos podemos encontrar con diferentes tipos de señales electrónicas a gestionar según su función (entradas o salidas) y el rango de la señal (digitales o analógicas). Además el número de señales necesarias para controlar cada elemento será diferente. Por ejemplo, para controlar una persiana motorizada necesitaremos dos pulsadores (función up/down), dos líneas de control del motor (para girar en un sentido u otro), y dos finales de carrera para controlar cuando la persiana está completamente abierta o cerrada. Por tanto, el objetivo de este proyecto será un sistema que nos permita definir las diferentes señales de control de cada elemento y la puesta en marcha del sistema que controlará cada elemento. Como ya hemos comentado, este sería un caso práctico de ejemplo, así que para la realización del proyecto debemos tener en cuenta que nuestro sistema debe ser fácilmente adaptable a otras instalaciones y escalable para controlar instalaciones más complejas.

Como veremos más adelante, el principal inconveniente a la hora de crear una red domótica mediante panStamp es la poca adaptabilidad que tenemos a la hora de crear nuestro dispositivo. Dependiendo del número y tipo de entradas que se quieran utilizar se debe realizar un nuevo proyecto con las especificaciones de cada nodo que queramos implementar. Esto provoca que realizar una instalación sería sumamente costoso en cuanto a tiempo, lo cual intentaremos remediar llevando a cabo la creación de un proceso automatizado que nos permita realizar la instalación de forma rápida y segura.

Planificación

El proyecto se desarrollara principalmente en tres fases diferenciadas, que finalizaran coincidiendo con las entregas a realizar durante el proyecto:

- Primera fase: Desarrollo de hardware y software de los módulos de entradas y salidas de panStamp.
- Segunda fase: Desarrollo de herramientas para crear los archivos de configuración para gestionar los dispositivos de entradas y salidas creados.
- Tercera fase: Fase de pruebas de funcionamiento y documentación de la memoria final del proyecto.

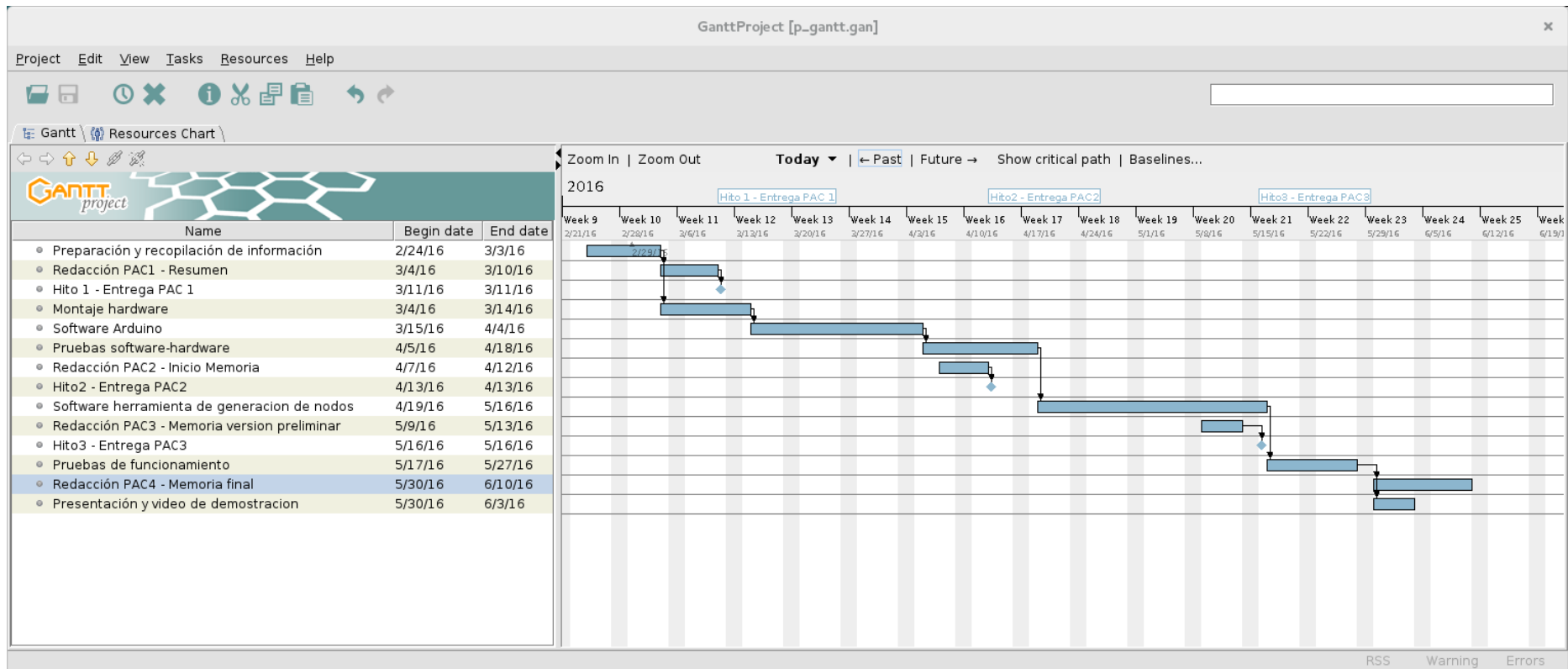


Ilustración 6: Planificación

Tecnologías relacionadas

panStamp

Como hemos comentado, la base del proyecto estará formada por una red de dispositivos panStamp. Dichos dispositivos están disponibles en dos versiones panStamp AVR y panStamp NGR. La diferencia entre ambos modelos es la CPU que incorporan, mientras que los AVR trabajan con una CPU Atmega328p (idéntica a la que incorpora la plataforma Arduino Uno), los NGR incorporan el MSP430 diseñado específicamente para aplicaciones de bajo consumo. Así mismo, los módulos panStamp han ido evolucionando en diferentes versiones, variando principalmente el formato de encapsulado de los componentes disminuyendo progresivamente las dimensiones. La versión mas actual es la 2.0 (Septiembre 2015), aunque para nuestro proyecto se utilizará la anterior versión 1.1:

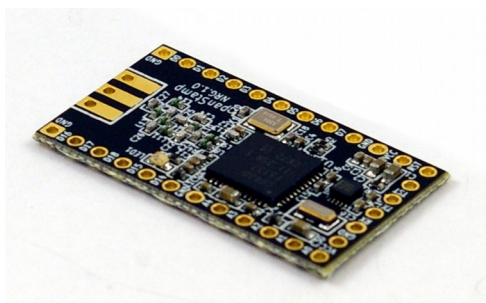


Ilustración 7: panStamp v1.1

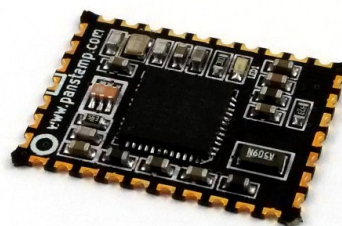


Ilustración 8: panStamp v2.0

Las especificaciones técnicas mas relevantes de los panStamp NRG 1.1 son las siguientes:

- Dimensiones: 17.7 x 30.5 mm)
- Microcontrolador CC430F5137 (MSP430 core + CC11XX radio SOC)
- Memoria Flash: 32KB
- Memoria RAM: 4KB
- Tensión de alimentación: de 2VDC a 3.6VDC
- Consumo en recepción: 18 mA max
- Consumo en transmisión: 36 mA max
- Consumo en modo "sleep": 1-2 uA
- Potencia máxima de transmisión: +12 dBm
- Radio máximo de comunicación: 200m en espacio abierto a 0dBm
- Seguridad de comunicaciones: 128-bit AES Security Encryption
- Accelerometro MMA8652FFC 3-axis y thermistor incorporados
- Programable mediante SBW, serial BSL y inalambrica (SWAP)

Para interactuar con los módulos panStamp también necesitaremos un PC y un módem que nos permita comunicar y programar los dispositivos. En el proyecto panStamp este requerimiento está cubierto mediante el dispositivo panStick.

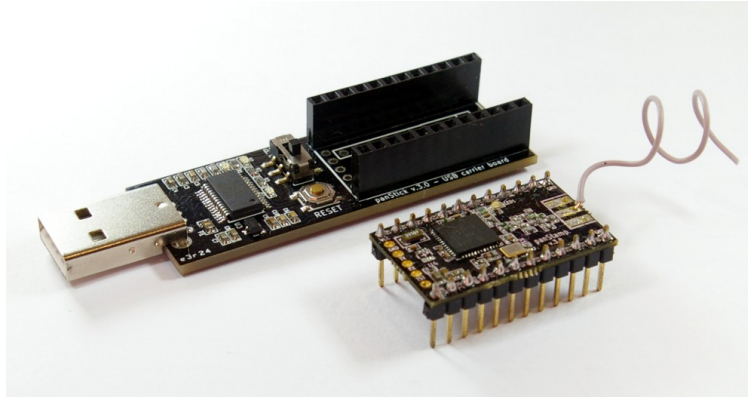


Ilustración 9: Módulo de programación panStick

La principal función de este dispositivo en nuestro proyecto será la de programación de los dispositivos panStamp, ya que la parte de comunicaciones para nuestra red se realizará mediante una Raspberry Pi. Existe una carta controladora (shield) que nos permite hacer uso de los pines de salida de la raspberry para controlar un módulo panStamp. Para este módulo panStamp cargaremos el sketch de Arduino para funcionar como módem proporcionado por el fabricante, lo cual convertirá la Raspberry Pi en un sistema GNU/Linux con acceso a la red inalámbrica panStamp. En nuestro caso este será el punto central de nuestra red inalámbrica, y nos permitirá, mediante el software necesario, gestionar los eventos de cada uno de los nodos panStamp.



Ilustración 10: Raspberry pi con shield para panStamp

El pinout para la versión 1.1 es el siguiente, en el cual podemos distinguir entre tres tipos de pins:

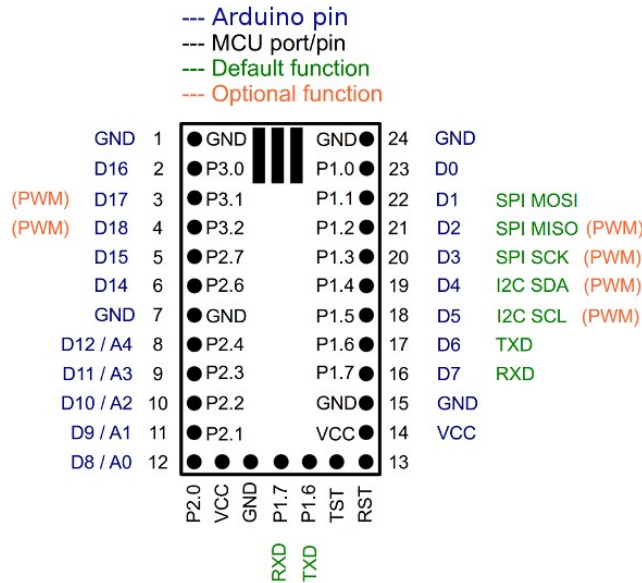


Ilustración 11: pinout de panStamp NGR 1.1

- Los pines marcados como Arduino pin, en color azul, son los terminales que podemos configurar desde el entorno Arduino como entradas o salidas digitales y son los que normalmente usaremos en nuestro proyecto.
- Los pines marcados en negro (MCU) son los pines utilizados para la grabación del dispositivo.
- Los pines marcados en verde son pines con alguna funciones genéricas por defecto que intentaremos no utilizar en nuestro proyecto. Por ejemplo, para utilizar la terminal de depuración de Arduino se deberían liberar los pines 1.4 y 1.5

Podemos comprobar que existen diferentes funciones para los pines utilizados para entrada/salida. Los pines marcados como Dxx son puertos digitales y nos permitirán leer o escribir datos binarios, como por ejemplo examinar el estado de interruptores, o activar cargas para encendido de lamparas. Los pines P2.0, P2.1, P2.2, P2.3 y P2.4, marcados como Ax, son pines de lectura analógicos y nos permitirán leer datos de sensores externos como sensores de luz solar o de temperatura. Y por ultimo los pines D2, D3, D4, D5, D17 y D18, marcados como PWM (*pulse-width modulation*), son pines en los la salida es una señal periódica con valores alto y bajo en los que podemos regular el tiempo en que estará la señal en cada estado (*duty-cycle*). Esto es equivalente a una señal analógica, ya que con una sencilla etapa de salida (*resistencia+condensador*) y regulando el ancho de los pulsos, podemos crear una señal continua entre los valores alto y bajo.

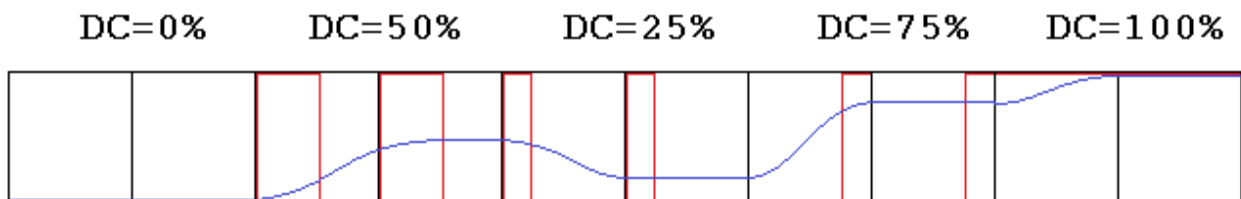


Ilustración 12: PWM(rojo) y señal analógica obtenida(azul)

Equipos prediseñados:

Como comentábamos anteriormente existen diversos ejemplos implementados que podemos consultar en el proyecto panStamp e incluso están disponibles comercialmente diversas placas de control para estos dispositivos. En la propia pagina web de panStamp podemos destacar las siguientes opciones:

- Modulo de control de salidas: Carta que permite el control de diversas 8 salidas digitales y 4 salidas PWM. Podemos encontrar una versión comercial de la placa en:

<http://www.panstamp.com/es/product/output-board/>

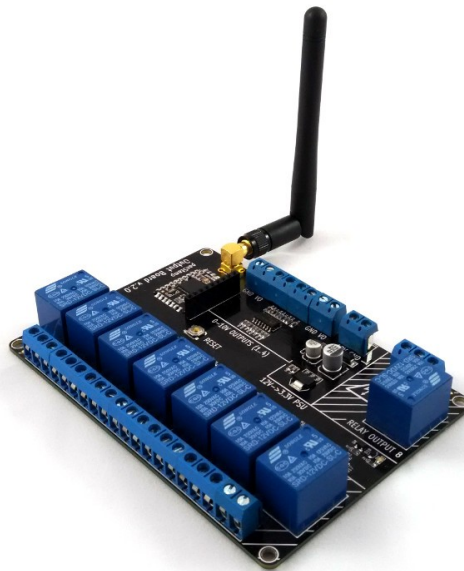


Ilustración 13: Placa de control de entradas/salidas

- Modulo RGB: Placa que permite controlar iluminación LED RGB. También podemos encontrar la versión comercial en <http://www.panstamp.com/es/product/rgb-driver/>
- Pulse-counter: Implementación de un contador de pulsos (p.ej. De un contador eléctrico o de agua) que permite la obtención de gráficas de consumos de suministros
- Temphumpress: Implementación de un modulo de control de temperatura, humedad y presión.
- Respira (CO2 y N2O meter): Implementación de un modulo de sensores de medida de gases.

Aparte de la documentación del proyecto panStamp podemos encontrar aportaciones de diversos desarrolladores que han compartido sus propios proyectos. No obstante, debido al carácter académico y por presupuesto, en nuestro proyecto no se utilizaran estos dispositivos sino que se realizará un prototipo de pruebas emulando entradas y salidas mediante indicadores led, pulsadores o lecturas directas de voltajes.

Programación de dispositivos

Para la programación de los dispositivos panStamp el proyecto cuenta con 3 interfaces diferenciadas con diferentes objetivos y capacidades de control:

- Arduino API: Es la API genérica de Arduino, la cual nos permite tener control sobre entradas/ salidas, tanto digitales como analógicas y funciones especiales como PWM, comunicaciones UART, SPI, etc...
- panStamp API: Es la API de control del núcleo de panStamp y nos proporciona las funciones mas básicas de funcionamiento y comunicaciones de los módulos. Sin embargo, no proporciona ningún tipo de control sobre el protocolo de comunicaciones SWAP. Esta API incluye funciones tales como inicialización de los módulos y comunicaciones, envío y recepción directos de datos, funciones de modo reposo y tratamiento de interrupciones.
- SWAP API: Es la API de control del protocolo SWAP, la cual nos permite hacer interaccionar los módulos panStamp sin tener que preocuparnos con las tareas de bajo nivel de comunicaciones. Es la API que usaremos mas habitualmente para configurar las especificaciones que queramos implementar en nuestro dispositivo.

Protocolo de comunicaciones SWAP

El protocolo SWAP es la base de la sencillez de panStamp. Consiste en una serie de funciones, variables y registros que permiten que la comunicación entre dispositivos sea automática. El protocolo implementado es el responsable de escuchar las transmisiones dentro de la red, filtrar los paquetes dirigidos a cada nodo, transmitir paquetes o enviar respuestas automáticas. A la hora de programar los dispositivos cada nodo cuenta con una serie de registros, por ejemplo estado de una entrada/salida, valores de sensores, etc... Cada registro se define independientemente y puede contar con una variable para almacenar el estado, una función de lectura del valor (getter function) y escritura del valor (setter function). La definición de cada registro tiene el siguiente formato:

```
// Allocating the register field
static byte value1[2];
// Definition of the register itself
REGISTER regValue1(value1, sizeof(value1), &getValue1, &setValue1 );
```

donde value1 es el valor de nuestro registro y &getValue1 y &setValue1 son punteros a las funciones de lectura y escritura. Obviamente dependiendo del tipo de registro que queramos implementar las funciones de escritura y lectura no serán necesarias (p.ej podemos leer el valor de un sensor, pero no escribirlo). Una vez tenemos definidos los registros de nuestro modulo panStamp se declaran en la tabla de registros:

```
//Initialize table of registers (regTable)
DECLARE_REGISTERS_START()
    &regValue1,
    &regValue2,
    [...]
    &regValueN,
DECLARE_REGISTERS_END()
```

Una vez finalizadas las declaraciones de registros y programadas las funciones necesarias para cada registro el sistema se vuelve automático y ya no nos tendremos que volver a preocupar de como funciona las comunicaciones. Esta fase del proyecto puede ser la mas complicada a la hora de trabajar con los panStamp, ya que conceptualmente es difícil de comprender su funcionamiento a primera vista, pero consultando los diversos ejemplos implementados se puede comprobar que una vez entendido el funcionamiento, el desarrollo de nuevas aplicaciones es mucho mas sencillo y rápido de lo que se podría pensar. Mediante las herramientas de comunicaciones proporcionadas por panStamp podemos ver y analizar las tramas de información del protocolo SWAP. Dichas tramas tienen la siguiente estructura:

SWAP status packet								
dest addr	scr addr	hop	secu	nonce	function	reg addr	reg id	valor
1 byte	1byte	4bit	4bit	1byte	1byte	1byte	1byte	x bytes

Donde:

- dest addr es la dirección del nodo de destino del mensaje dentro de la red panStamp
- scr addr es la dirección del nodo de origen del mensaje dentro de la red panStamp
- hop, secu, nonce son valores para la encriptación y el modo repetidor
- function es la función del mensaje
- reg addr es la dirección del nodo con el registro usado
- reg id es el identificador del registro usado
- valor es la información del registro

En el siguiente ejemplo vemos el mensaje provocado por la activación y desactivación de un puerto de entrada:

```
Rved: (352E)000A00DC000A0B01
Register addr= 10 id=11 changed to 01
alarmaParking_10 in address 10 changed to on
Rved: (352E)000A00DD000A0B00
Register addr= 10 id=11 changed to 00
alarmaParking_10 in address 10 changed to off
```

Ilustración 14: Trama tras la activación de un pulsador

Podemos analizar la primera de las tramas y ver que el mensaje es el siguiente:

Rved: (352E)000A00DC000A0B01

dest addr	scr addr	hop	secu	nonce	function	reg addr	reg id	valor
00	0A (10)	0	0	DC	00	0A(10)	0B(11)	01(on)

El nodo con dirección 10 (0x0A) envía al nodo central un mensaje de escritura del registro 11 (0x0B) con el valor 01 (on).

En el siguiente ejemplo vemos la activación de una salida desde la interficie web:

```
Sent: 0A0100DD020A0E80
Rved: (3930)000A00DE000A0E80
Register addr= 10 id=14 changed to 80
LampHabitacion_10 in address 10 changed to on
Sent: 0A0100DE020A0E00
Rved: (3C2F)000A00DF000A0E00
Register addr= 10 id=14 changed to 00
LampHabitacion_10 in address 10 changed to off
```

Ilustración 15: Trama tras la activación de una salida

Sent: 0A0100DD020A0E80

dest addr	scr addr	hop	secu	nonce	function	reg addr	reg id	valor
0A(10)	01	0	0	DD	02	0A(10)	0E(14)	80(on)

El nodo central envía al nodo con dirección 10 (0x0A) un mensaje de escritura del registro 14 (0x0E) con el valor 80 (on).

Ademas de datos binarios (on-off) también se pueden leer diferentes tipos de sensores de forma que se enviarán los mensajes según vaya cambiando el valor del registro:

```
Rved: (3730)000A0003000A1601
Register addr= 10 id=22 changed to 01
RegHabitacion_10 in address 10 changed to 1
Rved: (3730)000A0004000A1602
Register addr= 10 id=22 changed to 02
RegHabitacion_10 in address 10 changed to 2
Rved: (3830)000A0005000A1603
Register addr= 10 id=22 changed to 03
RegHabitacion_10 in address 10 changed to 3
Rved: (3830)000A0006000A1604
Register addr= 10 id=22 changed to 04
RegHabitacion_10 in address 10 changed to 4
Rved: (3830)000A0007000A1605
Register addr= 10 id=22 changed to 05
RegHabitacion_10 in address 10 changed to 5
```

Ilustración 16: Tramas de lectura de sensor

Lagarto

Una vez tengamos los dispositivos panStamp en funcionamiento el siguiente paso es configurar las interacciones entre los diferentes módulos. En este punto es donde entran en escena el servidor Lagarto. Lagarto es una plataforma implementada en Python diseñada para automatizar tareas y que proporciona la infraestructura necesaria para construir redes de aplicaciones complejas y capaz de integrar diferentes tecnologías. Se compone de dos procesos diferenciados:

Servidor Lagarto: Es la parte que se encarga de la comunicación con redes externas. En nuestro caso se usará Lagarto-SWAP que nos permitirá interactuar con nuestra red de dispositivos panStamp.

Cientes Lagarto: Los clientes son los elementos que trabajan con los datos de la red. Por una parte reciben eventos desde los servidores y por otra envían comandos hacia ellos. En nuestro caso trabajaremos principalmente con la interficies web Lagarto-MAX que nos permitirá interactuar con los módulos panStamp.

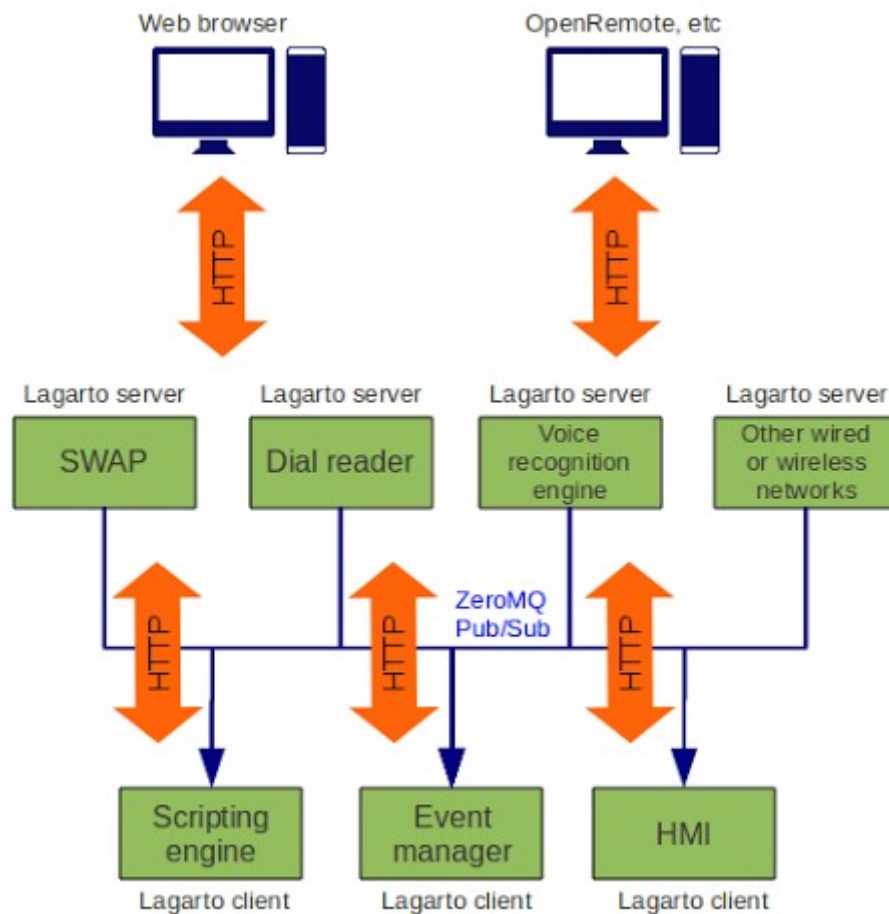


Ilustración 17: Esquema cliente-servidor de Lagarto

Lagarto SWAP

Lagarto SWAP es el software que nos permitirá la comunicación con los panStamp. Una vez instalado el software el manejo es sumamente intuitivo accediendo a una interficie web accesible a través de la dirección `http://ip_address:8001`. Dentro de la interficie web podemos configurar los parámetros de nuestra red inalámbrica, configuraciones del módem, seguridad de transmisiones, etc...

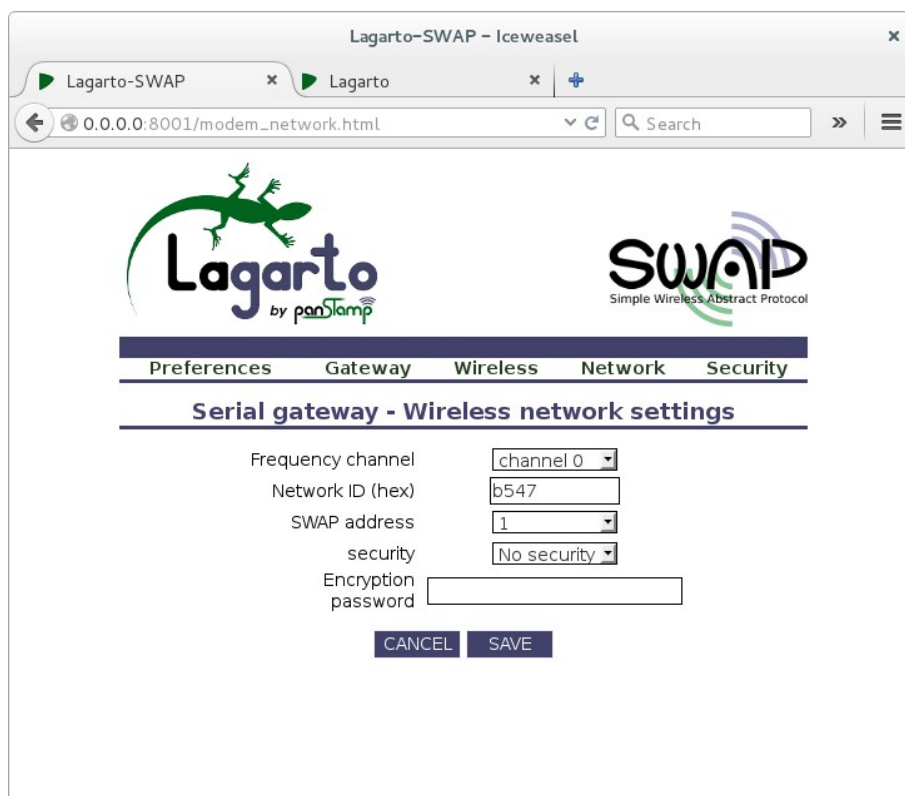


Ilustración 18: Configuraciones generales de la red inalámbrica

Lagarto MAX

Lagarto-MAX (Modulo de automatización extendida) es el proceso que nos permitirá automatizar eventos dentro de nuestra red inalámbrica. Este proceso es el encargado de recibir y procesar eventos desde Lagarto-SWAP, y enviar comandos sobre los diferentes elementos de nuestra red. El control de este proceso se realiza igualmente a través de una interficie web, accediendo a través de la dirección `http://ip_address:8002`.

La principal funcionalidad que podemos encontrar en este proceso es la capacidad de programar eventos. Dichos eventos se activan a través de un disparador (trigger) que provocará que se realicen ciertas acciones. En el siguiente ejemplo vemos como la activación de un interruptor en un modulo panStamp provocará la activación de una salida en otro dispositivo dentro de la red.

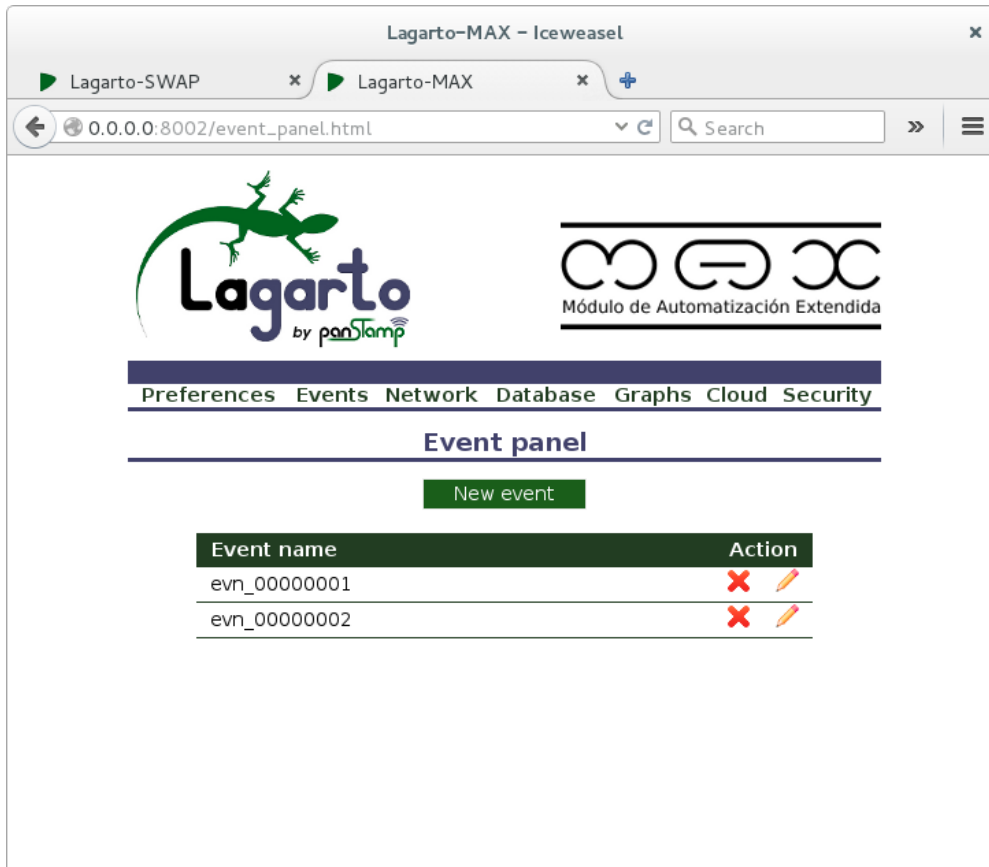


Ilustración 19: Eventos de LagartoMax

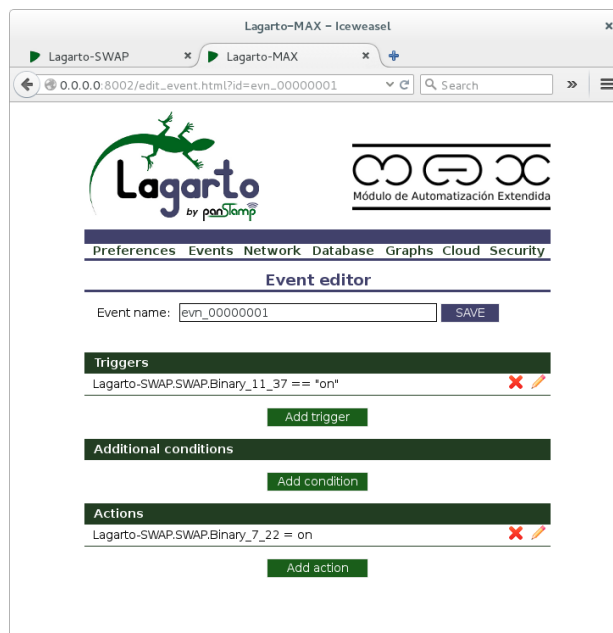


Ilustración 20: Ejemplo de configuración de un evento consultando la entrada de un modulo y activando la salida en otro

Node-red

PanStamp es un proyecto aún en desarrollo y por esto se publican periódicamente nuevas mejoras a los diferentes módulos que componen el sistema. Durante la realización de nuestro proyecto se han actualizado las herramientas de comunicación de panStamp para añadir nuevas funcionalidades y una interficie renovada.

El cambio mas importante ha sido a nivel interno, dejando de utilizar las librerías de gestión de mensajes ZeroMQ por las librerías MQTT, un protocolo de mensajería estándar diseñado para aplicaciones con ancho de banda limitado. El principal motivo para este cambio es el uso mas extendido de las librerías MQTT en aplicaciones de automatización y la evolución de este protocolo respecto a su predecesor desde la creación de panStamp en 2011. MQTT es un protocolo de mensajería ligero diseñado específicamente para control de sensores. Se basa en el principio de publicación de mensajes y suscripción a topics, y esta compuesto por un broker, que se encarga de gestionar la comunicación, y los clientes, que serán los que publicarán mensajes y recibirán información de los topics a los estén suscritos.

La versión de Lagarto SWAP 3.0 por tanto ha supuesto que el modulo Lagarto MAX, el modulo de automatización del nuestro sistema, deja de ser mantenido para dar paso a nuevos procedimientos capaces de utilizar sistemas mas potentes para la gestionar la lógica de nuestra automatización. En la nueva versión Lagarto SWAP se conecta a un broker MQTT, en nuestro caso la herramienta Mosquitto, y este se conecta a alguno de los diferentes programas de automatización que pueden comunicarse con MQTT como pueden ser node-red, OpenHAB, Domoticz, Home Assistant o HomeSeer.

En el caso de panStamp, el software recomendado para automatización es node-red, una herramienta gráfica visualmente muy intuitiva y fácil de gestionar. Esta herramienta cuenta con acceso a los mensajes MQTT del nuevo sistema, permite implementar funciones lógicas adicionales, y ademas muestra la información a través de un navegador web. La implementación de un proyecto con node-red es puramente gráfica, gestionando los diferentes elementos en forma de bloques y uniendo los diferentes bloques a través de líneas o flows. Los bloque pueden ser bloques de entrada o salida de información MQTT, bloques aritméticos, bloque de función, bloques de publicación de mensajes (mail, twitter, etc...), bloques de trazado de gráficas, etc...

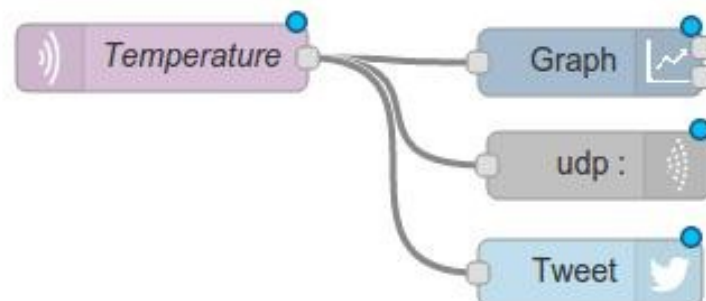


Ilustración 21: Ejemplo de conexión de bloques con node-red

Configuración de dispositivos en Lagarto

Como comentábamos anteriormente, cuando programamos un nuevo dispositivo panStamp tenemos que definir una serie de registros (estado de una entrada/salida, valores de sensores, etc...) que serán los datos enviados en las transmisiones SWAP. Para que nuestro servidor Lagarto sea capaz de reconocer cada dispositivo y trate los datos recibidos de forma correcta también es necesario que cada dispositivo este debidamente definido en el servidor. Cada tipo de dispositivo de la red SWAP debe de disponer su correspondiente archivo de configuración. Estos archivos son ficheros xml en los cuales se especifican el nombre de la aplicación y el nombre de cada registro y su función. Un ejemplo de implementación sería el siguiente:

```
<?xml version="1.0"?>
<device>
  <developer>NombreDesarrollador</developer>
  <product>NombreProducto</product>
  <regular>
    <reg name="NombreRegistro1" id="11">
      <endpoint name="DescripciónRegistro1" type="num" dir="inp">
        <size>2</size>
        <units>
          <unit name="V" factor="0.001" offset="0"/>
        </units>
      </endpoint>
    </reg>

    <reg name="NombreRegistro2" id="12">
      <endpoint name="DescripciónRegistro1" type="bin" dir="out">
        <position>0.0</position>
        <size>0.1</size>
      </endpoint>
    </reg>
  </regular>
</device>
```

donde:

- NombreDesarrollador: Es el nombre del desarrollador de la aplicación
- NombreProducto: Es el nombre de la aplicación tal como queremos que aparezca en la interficie web.
- NombreRegistroX y DescripciónRegistroX: Son el nombre y la descripción de cada registro, que pueden estar definidos como valores numéricos (type=num) o binarios (type=bin) y tanto de entrada (dir=inp) como de salida (dir=out)

Aparte de cada dispositivo, existe también una tabla general de todos los dispositivos panStamp creados en los que también deberemos añadir nuestros nuevos diseños

```
[...]
<developer id="IdDesarrollador" name="NombreDesarrollador">
  <dev id="IdDispositivo" name="NombreProducto" label="DescripciónProducto"/>
</developer>
[...]
```

Automatización de instalaciones

Objetivos

Una vez conocida las características y funcionalidades de panStamp y Lagarto retomamos nuestra tarea de crear una instalación para nuestra vivienda. En una primera aproximación, podríamos intentar crear la instalación a partir de los equipos comercializados, lo cual no sería muy poco práctico ya que únicamente existen comercialmente módulos de control de salidas (módulo con 8 relees + 4 PWMs), mientras que los módulos de control de entradas aún no se comercializan y tendríamos que implementarlos nosotros mismos.

Además, el hecho de tener fijadas el número de salidas y/o entradas en los equipos comercializados nos podría complicar el proyecto, tanto en distribución de los elementos fijos como en el número de módulos panStamp necesarios para la instalación. Por ejemplo, supongamos que necesitamos controlar en un punto de la vivienda 1 persianas (2 salidas), 3 lámparas (3 salidas), 5 interruptores (5 entradas) y un control de presencia (1 entrada). Con los módulos comerciales necesitaríamos un módulo de control de entradas y un módulo de control de salidas, los cuales además estarían infrutilizados al no usar todos los puertos disponibles.

La mejor solución, por tanto, es configurar las entradas y salidas específicas para nuestro panStamp para adaptarlo a la instalación real del proyecto domótico. No obstante, la creación de proyecto específico en panStamp es un proceso lento y laborioso, puesto que como hemos visto, para cada módulo panStamp que utilicemos hemos de configurar los puertos, crear registros para cada señal, funciones para lectura y escritura de registros, etc... además de los archivos de configuración para cada módulo en el servidor Lagarto. Hay que tener en cuenta también que en el caso de realizar proyectos panStamp "a medida", hemos de tener en cuenta que será bastante costoso realizar cambios en la instalación o ampliarla. Con la tecnología actual de panStamp esto implicaría que se deberían diseñar varios proyectos panStamp, lo cual se traduciría en las siguientes tareas:

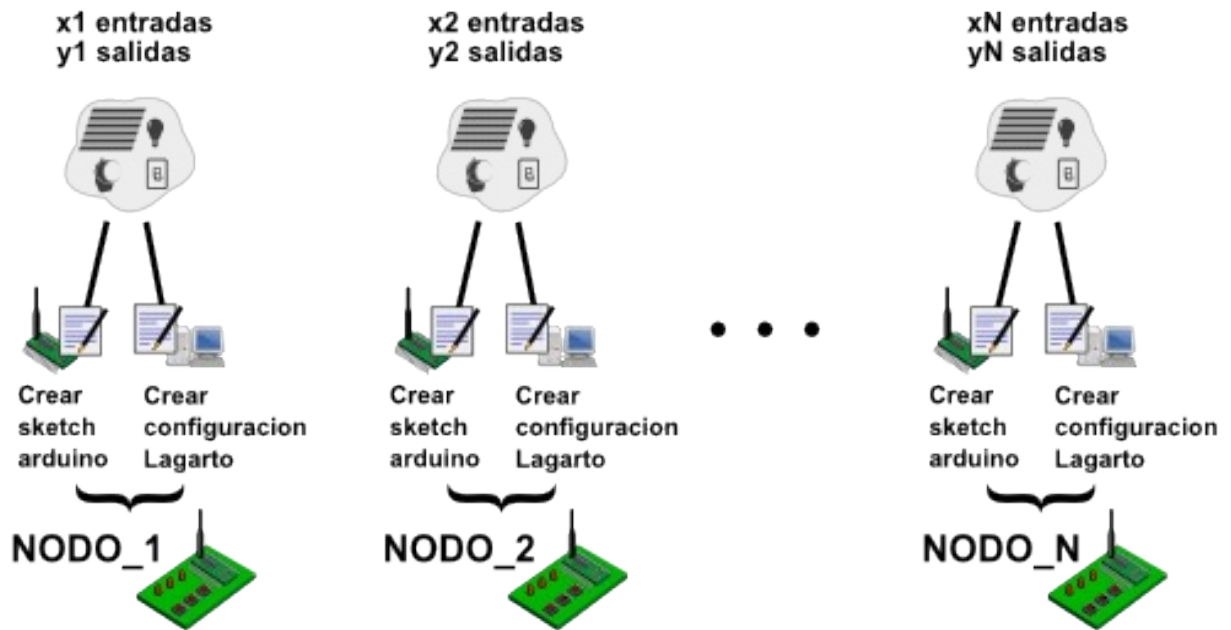
- Crear un sketch específico de Arduino
- Definir el número de entradas y salidas y asignar los puertos a cada actuador.
- Definir los registros y variables necesarias para cada actuador
- Programar el control de cada puerto según su función
- Compilación del sketch específico de Arduino
- Crear la configuración en función de los registros utilizados para el servidor Lagarto

Si en una instalación domótica tenemos diversos nodos panStamp con diferentes entradas y salidas cada uno, significa que estas tareas se deben de realizar para cada uno de los nodos involucrados, incrementando el tiempo de desarrollo y con una complejidad alta, lo cual se traduciría en un gran esfuerzo a la hora de buscar y depurar errores en la red.

Nuestro propósito es automatizar este proceso, de forma que el usuario tenga que definir únicamente el número de entradas y salidas que quiere gestionar en cada panStamp, olvidándose de la programación del sketch de Arduino y la configuración del servidor Lagarto. Por lo tanto las tareas:

- Definición de entradas y salidas y generación de archivos de configuración
- Compilación de un sketch de Arduino genérico con los archivos de configuración generados

Proceso habitual



Proceso automatizado

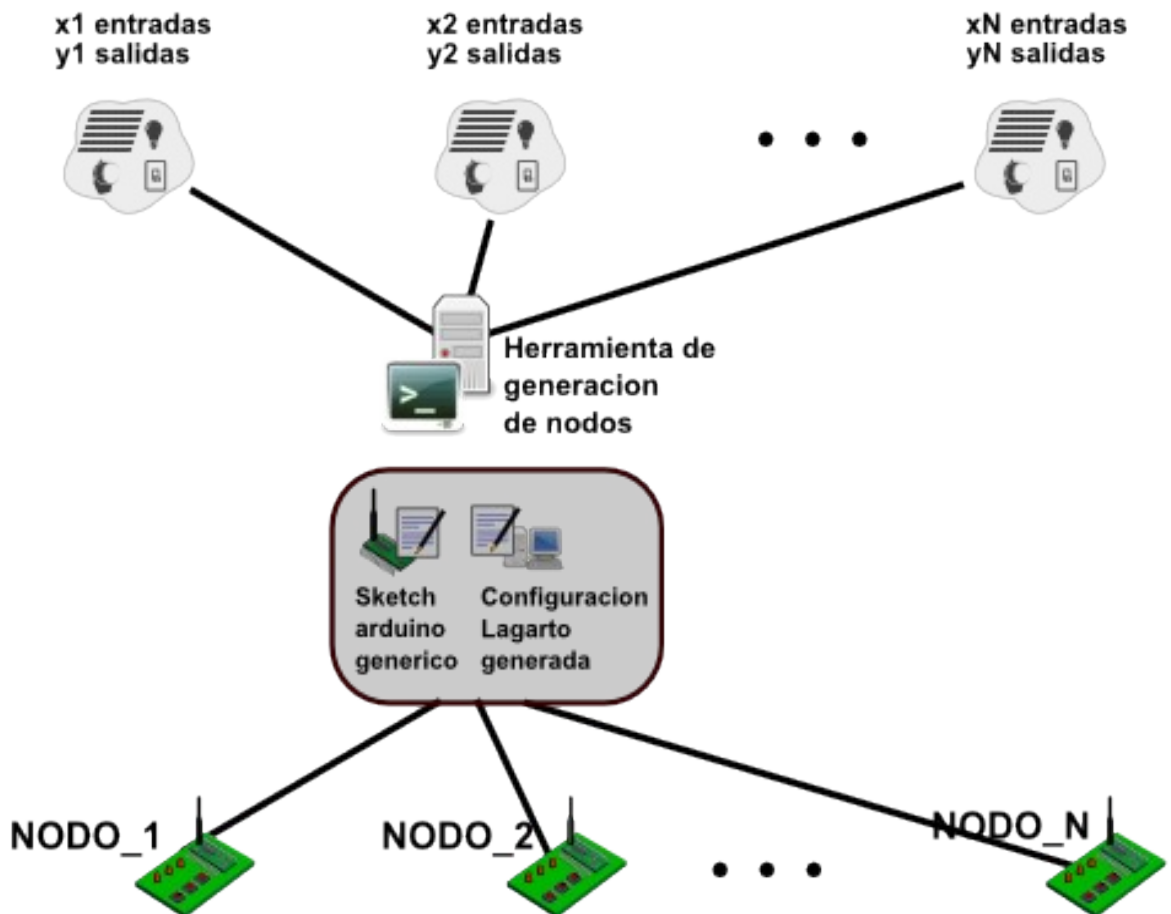


Ilustración 22: Diferencias en el proceso de creación de nuevos dispositivos

Planificación de la instalación

El primer paso del proyecto por lo tanto será un sketch que nos permitirá controlar diferentes entradas o salidas configurables por el usuario únicamente mediante un archivo de definición. Una vez definida la aplicación de cada uno de los pins, el sketch de Arduino se debería compilar correctamente con cada registro y funcionalidad asociada a cada pin correctamente. En nuestro ejemplo de instalación domótica podemos ver que necesitaremos 3 panStamp para controlar un total de 47 entradas/salidas distribuidas de la siguientes forma:

Estancia	Elemento	# Entradas Digitales	# Salidas Digitales	# Entradas Analógicas	# Salidas Analógicas
Cocina	1 Lampara		1		
Cocina	1 Interruptor	1			
Cocina	1 Detector de humo	1			
Cocina	1 Persiana	2	2		
Cocina	Interruptor doble	2			
Comedor	1 Lampara regulable		1		1
Comedor	1 Interruptor	1			
Comedor	1 Regulador de luz			1	
Comedor	1 Persiana	2	2		
Comedor	Interruptor doble	2			
Baño	1 Lampara		1		
Baño	1 Interruptor	1			
Habitacion	1 Lampara regulable		1		1
Habitacion	1 Interruptor	1			
Habitacion	1 Regulador de luz			1	
Habitacion	1 Persiana	2	2		
Habitacion	Interruptor doble	2			
Habitacion	1 Sensores de luz solar			1	
Parking	1 Lampara		1		
Parking	1 Interruptor	1			
Parking	1 Detector de humo	1			
Parking	1 Puerta de parking	2	2		
Parking	1 Interruptor doble	2			
Parking	1 Detector de presencia	1			
Pasillo	1 Lampara		1		
Pasillo	2 Interruptor	2			
Pasillo	1 Detector de presencia	1			
Global	1 Toma de corriente	1			

Tabla 3: Planificación de elementos en la vivienda

Sketch Arduino

Definiciones preliminares

Durante las primeras fases del proyecto se definieron los pines a utilizar estudiando las diferentes alternativas y con previsión de las circunstancias que podrían afectar al diseño final, y como resultado de este estudio se definieron los siguientes parámetros para la creación del sketch genérico:

- Se usarían 16 pines de los 18 configurables, dejando dos pines libre por si fuesen necesarios para nuevas funciones (por ejemplo, resetear el dispositivo, enviar funciones de echo, etc...). En concreto se usarían los pines D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D14, D15 y D16.
- Se usarían 4 de los 6 pines disponibles para la función PWM.
- Se usarían 4 de los 5 pines disponibles para la función de lectura analógica.

El archivo de configuración para el sketch de Arduino sería, por lo tanto, una lista de defines con los diferentes valores que podrían tener los diferentes puertos, con el siguiente formato:

```
#define IO_PORT00  BINARY_OUTPUT
#define IO_PORT01  BINARY_INPUT
#define IO_PORT02  PWM_OUTPUT
[...]
#define IO_PORT16  BINARY_OUTPUT
```

Donde IO_PORTxx es el define enumerado de los 16 puertos que vamos a controlar y puede tomar los siguientes valores:

- BINARY_OUTPUT: Salida digital (ON/OFF)
- BINARY_INPUT: Entrada digital (ON/OFF)
- ANALOG_INPUT: Entrada analógica
- PWM_OUTPUT: Salida analógica/PWM

Así, la base de nuestro trabajo será la creación de un sketch Arduino genérico en el que estarán previamente definidos 16 registros que pueden tomar valores analógico o digitales, con las funciones de lectura y escritura necesarias. A la hora de definir entradas y salidas el usuario únicamente deberá definir que tipo de función quiere darle a cada uno de los puertos mediante un archivo header de definición. El archivo de definición (define_device_io.h) será el único que se deberá modificar en el proyecto para la configuración específica de un panStamp y podrá compilar y funcionar con cualquiera de las posible configuración de puertos.

Hardware

Para testear el funcionamiento de las configuraciones de los equipos Arduino se han desarrollado placas de prototipos en los que se podrá emular los componentes reales (lamparas LED para las salidas, pulsadores para las entradas y entradas adicionales para señales analógicas) configurando los componentes necesarios mediante jumpers.

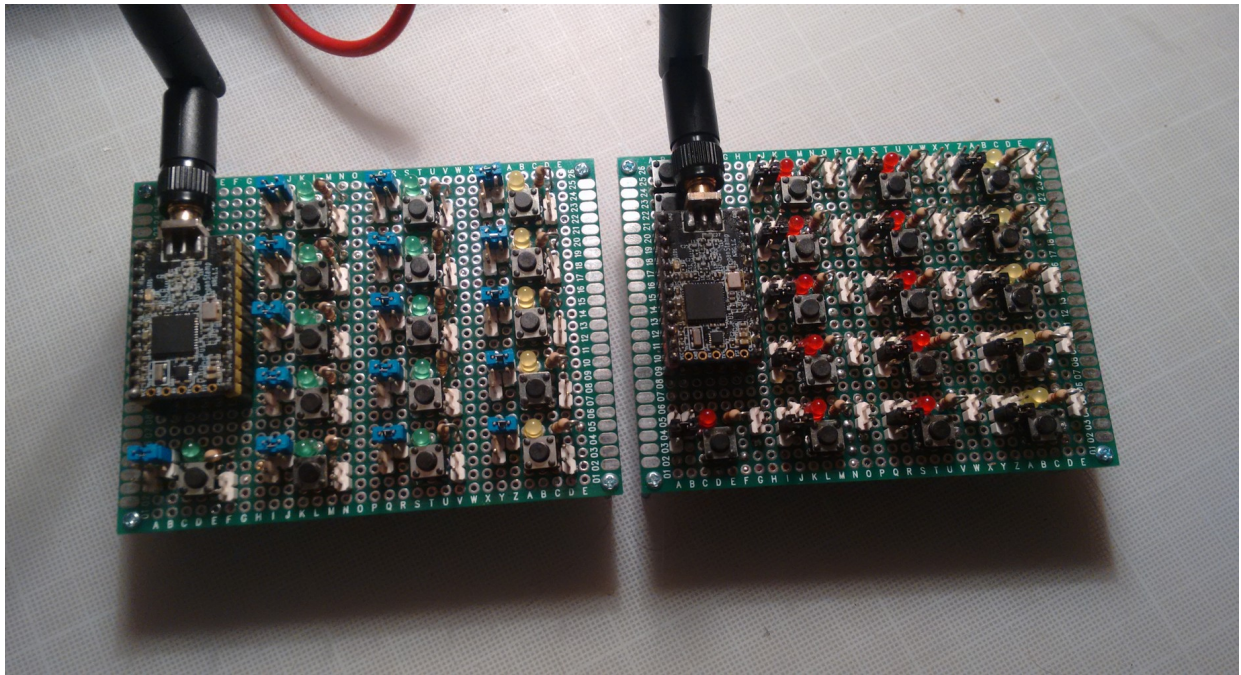


Ilustración 23: Primer prototipo de test de entradas y salidas

El circuito electrónico para testear entrada y salidas en un circuito simple en el que la mediante un jumper podemos seleccionar si un terminal del circuito se conecta a un led con una resistencia en serie para el caso de definirlo como salida, o a un interruptor con una resistencia de pull-up para el caso de definirlo como entrada.

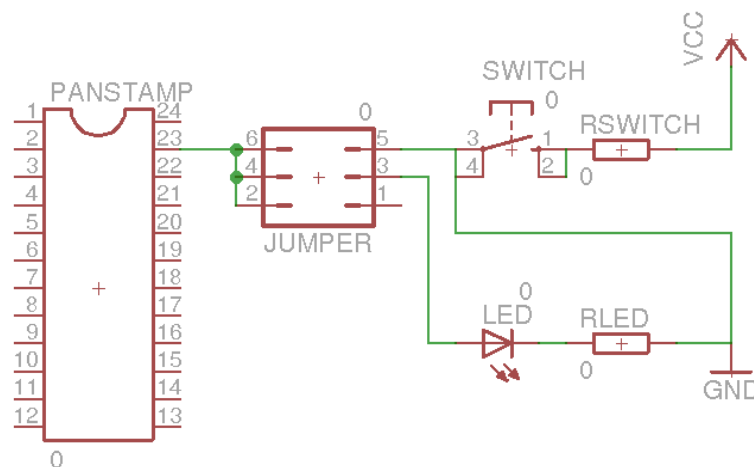


Ilustración 24: Circuito entradas/salidas

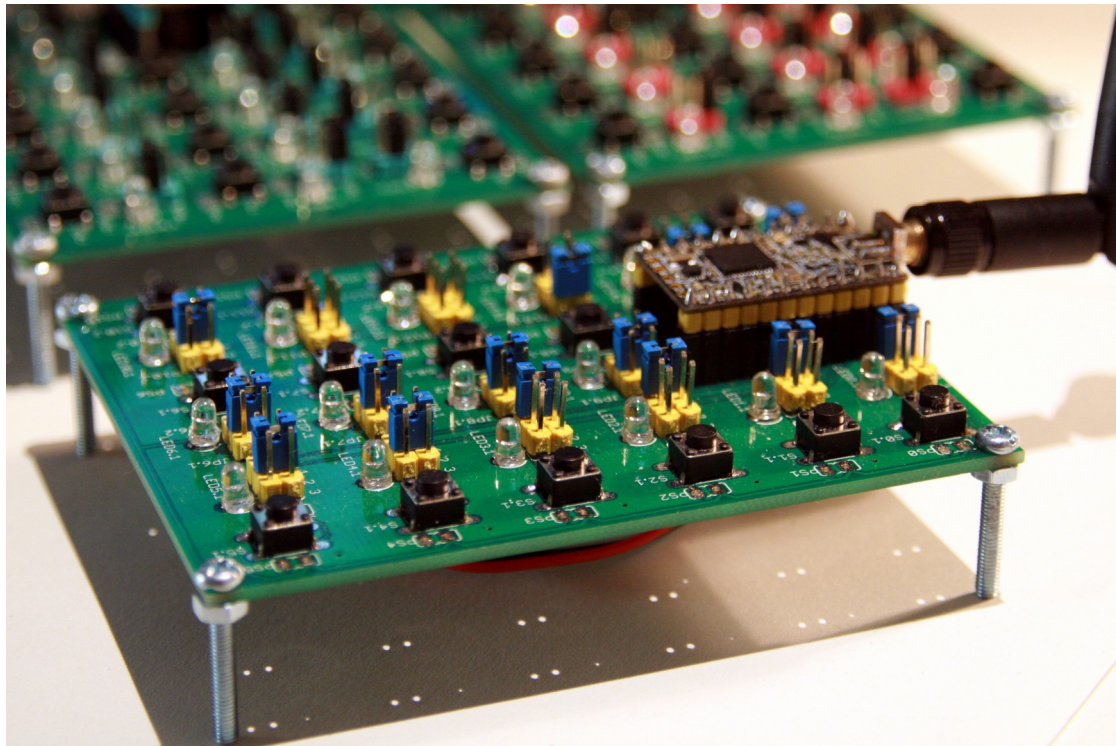


Ilustración 25: Segundo prototipo de test de entradas y salidas

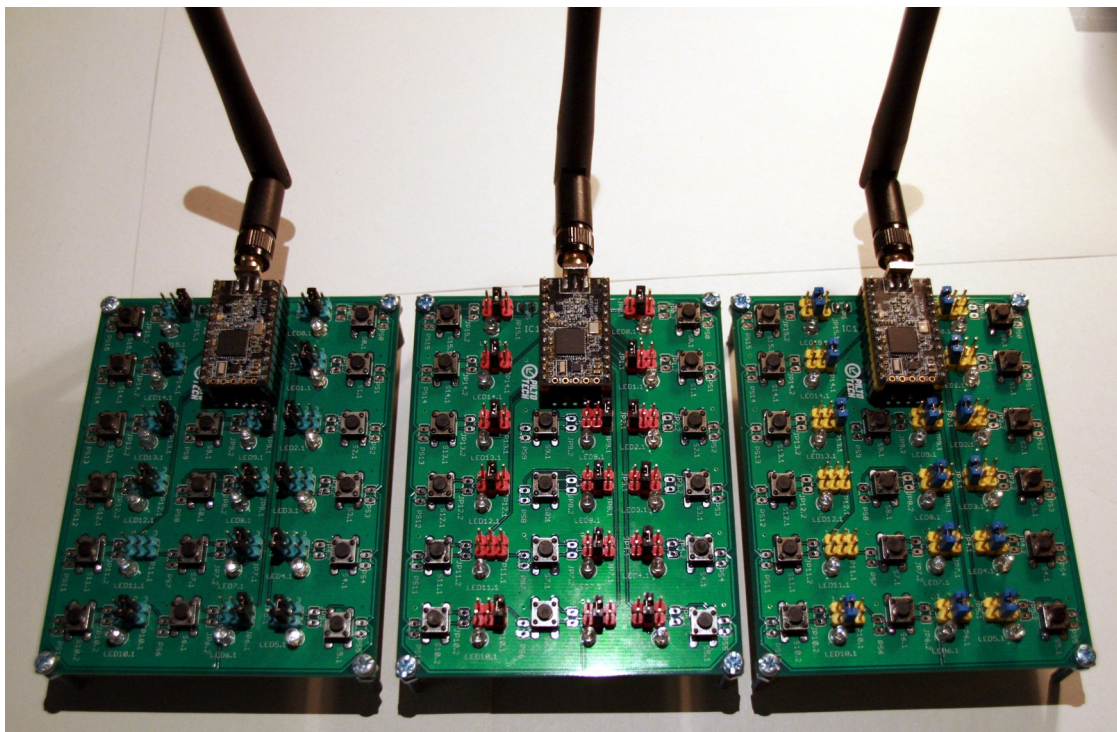


Ilustración 26: Grupo de placas para test de entradas y salidas

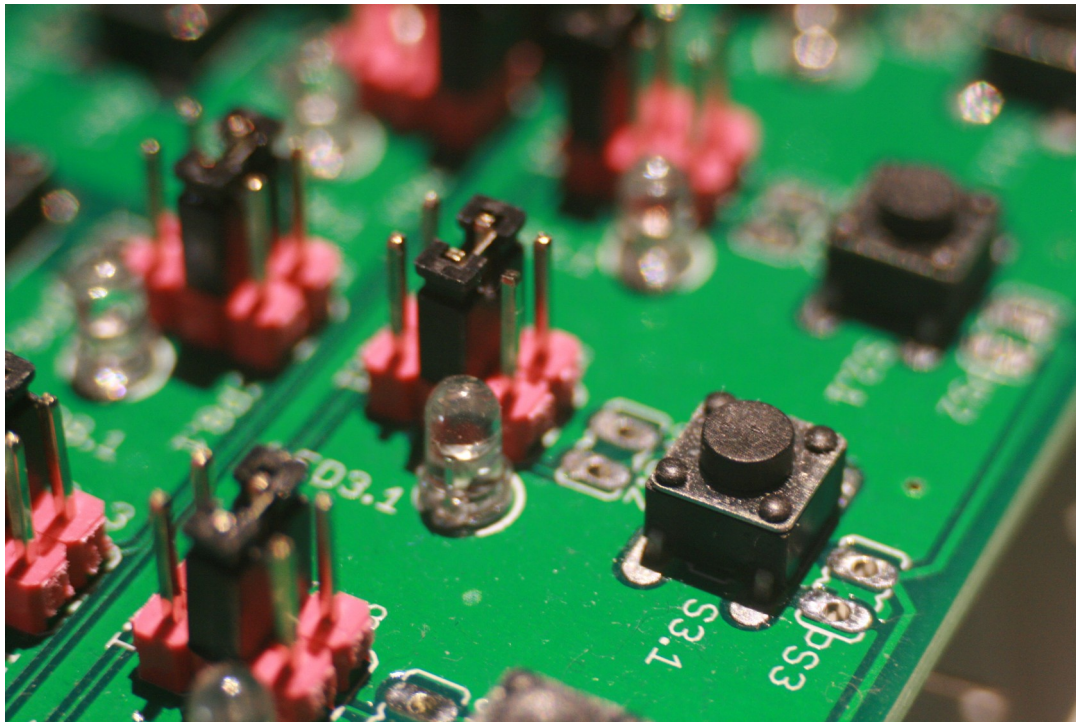


Ilustración 27: Detalle de pulsadores, leds y jumpers de configuración

Programación Arduino

El sketch genérico es la base de nuestro proyecto y se trata de un proyecto Arduino con el que controlaremos las diferentes entrada y salidas. El proyecto tiene un programa principal compuesto por la función de inicialización y el bucle del programa principal. Y por otra parte utilizará las librerías las librerías de panStamp para comunicar con los diferentes dispositivos. En el siguiente esquema vemos como se gestionan la comunicación:

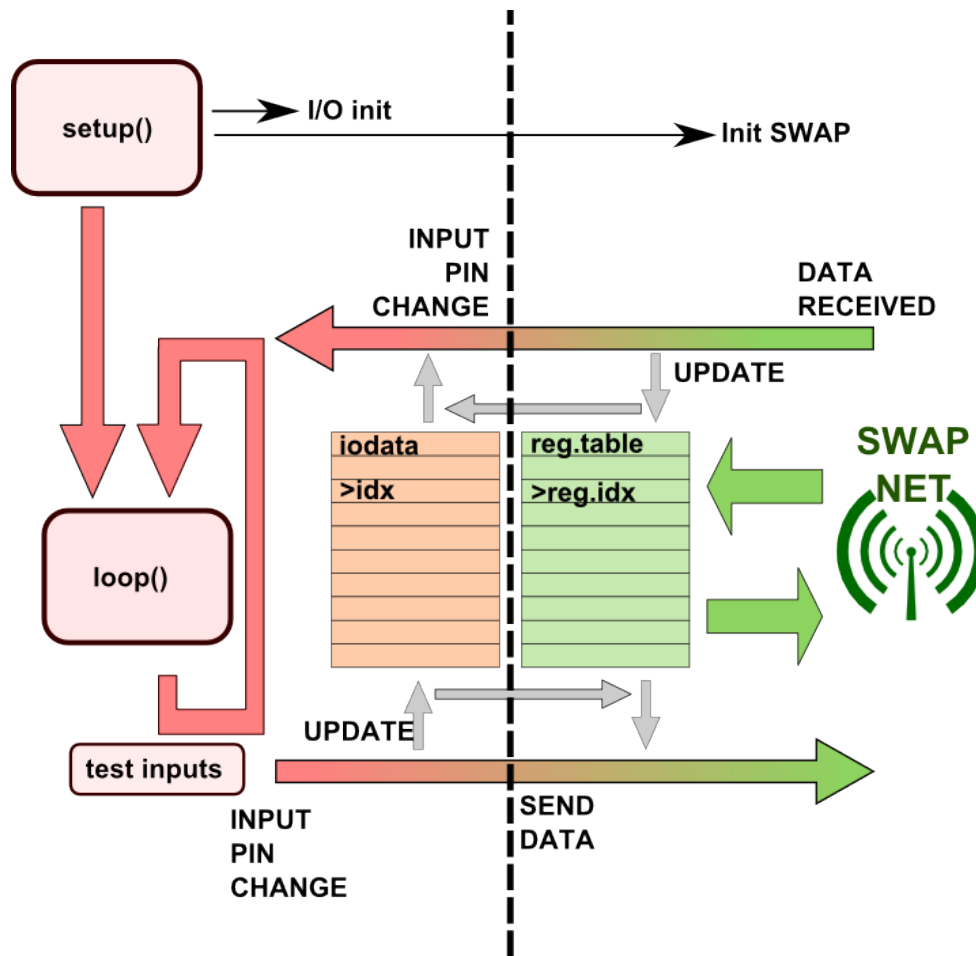


Ilustración 28: Esquema de comunicaciones del sketch de Arduino

Por una parte el programa cuenta con una función de inicialización `setup()`, que será la encargada de configurar los pines mediante la función `pinMode(pin, mode)`, función genérica de Arduino ; de acuerdo con la configuración deseada según el archivo `define_device_io.h` generado mediante la herramienta de generación de proyectos. También será la encargada de iniciar el protocolo de comunicaciones SWAP y de poner valores por defecto a las variables usadas por el sistema. El tratamiento del estado de las diferentes señales se coordina mediante dos tablas que contienen el estado de los puertos y de los registros. Por una parte tenemos la tabla de variables que consta de 16 variables tipo byte que contiene la información relativa a los puertos, usando todos los bits en caso de puertos analógicos, y el bit 0 para los puertos digitales. Y en la parte de comunicaciones tenemos la tabla de registros que consta de los 16 registros creados para cada entrada/salida, además del resto de registros propios de las librerías de panStamp. La coordinación entre las dos tablas es la clave para conseguir que la comunicación sea efectiva entre los puertos de Arduino y la red SWAP.

Después de la función de inicialización comienza a ejecutarse el bucle principal de programa loop(). En dicho bucle se estarán evaluando constantemente los pins configurados como entrada (tanto analógicos como digitales). Cuando el programa detecta algún cambio en uno de los pins, por ejemplo la activación de un pulsador, se actualiza la tabla con las variables que contienen la información asociada a cada pin, seguidamente se actualiza la tabla de registros del protocolo SWAP y a partir de aquí son las librerías de panStamp las que se encargan de gestionar el envío de la información a través de la red.

De forma inversa, cuando se recibe un mensaje desde la red inalámbrica, las librerías de panStamp gestionan el mensaje actualizando los valores en la tabla de registros del protocolo SWAP. A continuación se ejecuta la función setter implementada en el registro del mensaje recibido, que será la encargada de actualizar la tabla de variables iodata y de activar las salidas, tanto analógicas como digitales, asociadas al registro, como por ejemplo activar una lámpara.

Los datos tratados comparten las tablas de datos y de registros sin ninguna separación en función de si son datos de lectura o escritura, por lo tanto se hace necesario contar con rutinas de error para evitar incoherencias durante las operaciones realizadas. En concreto se lanzarán errores si tratamos de escribir un registro de lectura, o si tratamos de leer un registro de escritura. En estos casos se detendrá la ejecución del programa, mostrando un código de error a través del LED instalado en el módulo panStamp.

El proyecto consta de los siguientes archivos:

- define_device_io.h: Es el archivo de definición de entradas y salidas generado mediante la herramienta de creación de nodos descrita anteriormente. Será el único archivo que variará en los diferentes nodos de panStamp.
- define_type.h: Es el archivo de definiciones generales del proyecto. En él podemos encontrar los diferentes tipos de configuración para cada puerto BINARY_INPUT, BINARY_OUTPUT, ANALOG_INPUT, PWM_OUTPUT, el número y la lista de pines utilizados, etc...
- iosketch.ino: Es el programa principal del proyecto Arduino, que contiene las funciones de configuración setup(), y el bucle principal de programa loop(). El bucle principal de programa será el encargado de leer las diferentes entradas mediante polling y actualizar los registros en caso de detectar cambios. También implementa las funciones de lectura de pines de entrada, y funciones de lectura "suave" de valores analógicos para evitar que el ruido de la señal provoque que se lancen mensajes a la red continuamente.
- regtable.h: Es el archivo de definición de registros de panStamp. En este archivo encontramos la definición de registros y la tabla de variables utilizadas para el control de los diferentes puertos.
- regtable.ino: Definición de funciones utilizadas por los registros de panStamp. Aquí podemos encontrar las definiciones de los registros con las funciones de escritura y lectura de puertos necesarias.
- errores.h y errores.ino: Son archivos para el control de errores para detectar incoherencias durante el desarrollo del producto. Por ejemplo, al intentar escribir un dato en un registro de lectura.
- product.h: Es el archivo de definición de producto panStamp y contiene las definiciones de versión, desarrollador, id del dispositivo, etc...

```

/**
 * io_v07
 *
 * Author: Alberto Pelarda
 * Creation date: 01/04/2016
 */

#include "swap.h"
#include "regtable.h"
#include "define_device_io.h"
#include "errores.h"

/**

```

Ilustración 29: Sketch Arduino

Herramienta de generación de nodos

El siguiente paso en nuestro proyecto será la creación de la herramienta de generación de nodos panStamp. El servidor Lagarto está desarrollado en Python así que por coherencia con el resto de herramientas de panStamp se ha decidido que realizar esta herramienta en el mismo lenguaje de programación. Asimismo, el proyecto panStamp es multiplataforma, pudiéndose ejecutar tanto en Linux, como Windows como en OSX, por lo tanto se ha buscado una solución en la que la herramienta funcione en cualquier plataforma. Y por último, existe la posibilidad de ejecutar el servidor en una Raspberry Pi, con lo cual debemos tener en cuenta que esta herramienta que se podría ejecutar desde el mismo servidor, por lo tanto sería necesario poder ejecutarlo también en este entorno. Todo esto implica que para desarrollar la aplicación se ha usado únicamente la instalación por defecto de Python, sin hacer uso de librerías externas ni ningún tipo de interfaz visual con ventanas.

Por los motivos anteriormente expuestos se ha desarrollado la herramienta con un menú textual, en la cual el usuario escogerá las diferentes opciones mediante el teclado. No obstante, con la idea de mejorar esta herramienta con interfaces más intuitivas, se ha separado el proyecto en dos partes:

- ioboard.py: Archivo con el modelo de puertos y carta controladora. Este archivo contiene los modelos de puerto y carta, con las diferentes funciones necesarias para la creación de una carta de entradas y salidas, así como las funciones necesarias para la creación de los archivos header para el sketch de Arduino y los archivos de configuración de Lagarto.
- iocreator.py: En este archivo encontramos el programa principal de la herramienta que se encarga de pintar en pantalla la configuración actual del proyecto y las diferentes opciones del usuario.

La herramienta consta de los siguientes apartados:

1. Información de la placa con los siguientes datos:
 - Nombre de la placa
 - Breve descripción para poder identificar su funcionalidad
 - Código de producto, formado a partir de la función de cada entrada
2. Tabla con la configuración actual con los siguientes campos:
 - Número de puerto
 - Función actual
 - Opciones posibles en cada puerto
 - Breve descripción de la función
3. Entradas de menú:
 - Configurar información de la placa
 - Configurar puerto
 - Generar archivos de configuración

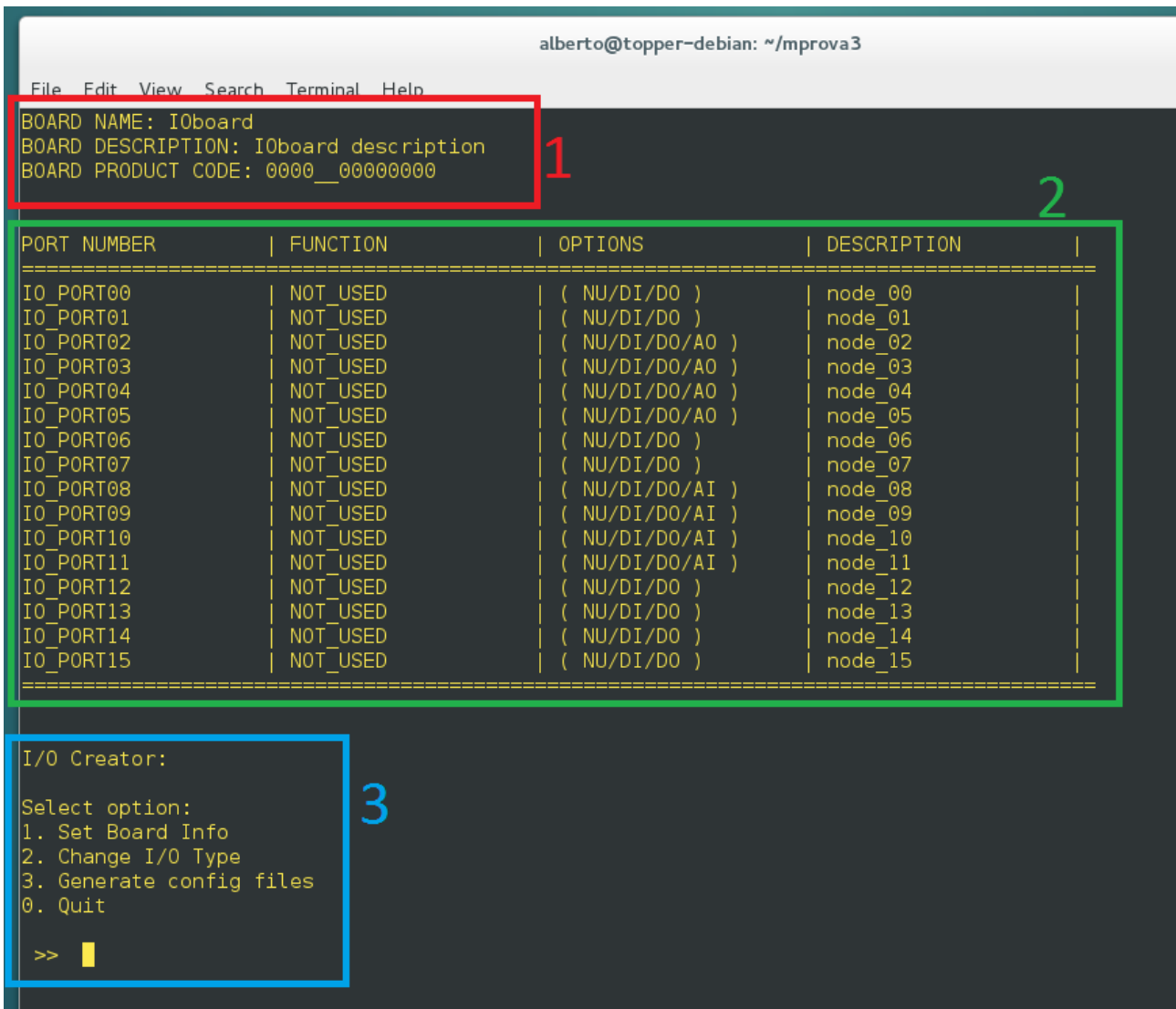


Ilustración 30: Herramienta de generación de nodos

El menú de la herramienta consistirá en los siguientes apartados:

Cambiar información de la placa:

En este apartado podremos configurar tanto el nombre de la placa como una pequeña descripción que nos ayudará a identificar la placa. Dichos campos estarán condicionados a un número máximo de caracteres y al tipo de caracteres que podemos utilizar, generando un error en caso de no cumplir las condiciones:

```
Select option:
1. Set Board Info
2. Change I/O Type
3. Generate config files
0. Quit

>> 1
Define board name (max:15 character - Only letters, numbers, and underscore)
>> GreenBoard
Define additional description (max:15 character - Only letters, numbers, and underscore)
>> 4out_4in_2pwm_2an
```

Ilustración 31: Edición de la información de placa

Cambiar tipo de entrada/salida:

En este apartado podremos seleccionar para cada puerto el tipo de entrada que queremos utilizar. Una vez seleccionado el numero de puerto se mostrarán las opciones disponibles para que el usuario puedas escoger la funcionalidad. Y por ultimo se deberá introducir una pequeña descripción o nombre del puerto para que podamos identificar la entrada fácilmente en el servidor Lagarto. También en este caso los campos estarán condicionados según el paso en el que estemos, generando error en caso de que los datos introducido no sean coherentes.

```
Select option:
1. Set Board Info
2. Change I/O Type
3. Generate config files
0. Quit

>> 2
Select I/O port ( 0-15 )
>> 15
Select I/O type
( NU/DI/DO )
>> DI
Define port name (max:15 character - Only letters, numbers, and underscore)
>> in4
```

Ilustración 32: Edición de puerto

A medida que se van rellenando estos datos se va actualizando el código de producto que podemos encontrar en la parte superior de la pantalla. Este código será el identificador del producto en el servidor lagarto y en el estará codificada la información relativa a las entradas en dos números hexadecimales. El primer numero, de cuatro números hexadecimales, contendrá la información de los puertos usados. El segundo numero, de ocho números hexadecimales, contendrá la información de los tipos de entrada/salida de cada puerto.

Generar archivos de configuración:

Una vez introducido la información de la placa y de los puertos utilizados, la tercera opción del menú nos permitirá generar los archivos de configuración para el sketch de Arduino y el servidor lagarto:


```
I/O Creator:
Select option:
1. Set Board Info
2. Change I/O Type
3. Generate config files
0. Quit

>> 3
Config files generated.

Press ENTER to continue...|
```

Ilustración 33: Generación de archivos de configuración

Al seleccionar esta opción se generarán una carpeta con el siguiente formato NombreDePlaca_CodigoIO_CodigoPuertosUsados. Dentro de la carpeta podremos encontrar 3 archivos:

- define_device_io.h: El archivo necesario para la compilación del sketch de Arduino
- NombreDePlaca_CodigoIO_CodigoPuertosUsados.xml: archivo de configuración del servidor.
- IO_data.pkl: Información del proyecto generado. Gracias a este archivo podremos recuperar la información de un proyecto generado al introducirlo como entrada del programa Python y realizar las modificaciones necesarias.

Ejemplo de archivo de definición generado – define_device_io.h:

```
#ifndef _DEFINE_IO_DEVICE_H
#define _DEFINE_IO_DEVICE_H
#include "define_type.h"
#define NUM_IO_PORTS 16
#define IO_PORT00 BINARY_INPUT
#define IO_PORT01 BINARY_INPUT
#define IO_PORT02 PWM_OUTPUT
#define IO_PORT03 BINARY_OUTPUT
#define IO_PORT04 BINARY_OUTPUT
#define IO_PORT05 BINARY_INPUT
#define IO_PORT06 BINARY_OUTPUT
#define IO_PORT07 BINARY_INPUT
#define IO_PORT08 BINARY_INPUT
#define IO_PORT09 BINARY_OUTPUT
#define IO_PORT10 BINARY_INPUT
#define IO_PORT11 ANALOG_INPUT
#define IO_PORT12 BINARY_INPUT
#define IO_PORT13 BINARY_OUTPUT
#define IO_PORT14 BINARY_INPUT
#define IO_PORT15 BINARY_OUTPUT
#define PROD_ID 0x44841170L
#endif
```

Ejemplo de archivo de configuración generado – GreenBoard_44841170_FFFF.xml:

```
<?xml version='1.0' encoding='ASCII'?>
<device>
  <developer>apelarda</developer>
  <product>GreenBoard - 8DI_6DO_1AI_1AO</product>
  <pwrdownmode>>false</pwrdownmode>
  <regular>
    <reg id="11" name="IO_PORT00">
      <endpoint dir="inp" name="alarmaParking" type="bin">
        <position>0.0</position>
        <size>1</size>
      </endpoint>
    </reg>
    <reg id="12" name="IO_PORT01">
      <endpoint dir="inp" name="DetPresParking" type="bin">
        <position>0.0</position>
        <size>1</size>
      </endpoint>
    </reg>
    <reg id="13" name="IO_PORT02">
      <endpoint dir="out" name="LRegHabitacion" type="num">
        <position>0</position>
        <size>1</size>
      </endpoint>
    </reg>
    <reg id="14" name="IO_PORT03">
      <endpoint dir="out" name="LampHabitacion" type="bin">
        <position>0.0</position>
        <size>0.1</size>
      </endpoint>
    </reg>
    <reg id="15" name="IO_PORT04">
      <endpoint dir="out" name="LampParking" type="bin">
        <position>0.0</position>
        <size>0.1</size>
      </endpoint>
    </reg>
    <reg id="16" name="IO_PORT05">
      <endpoint dir="inp" name="IntParking" type="bin">
        <position>0.0</position>
        <size>1</size>
      </endpoint>
    </reg>
    <reg id="17" name="IO_PORT06">
      <endpoint dir="out" name="LampPasillo" type="bin">
        <position>0.0</position>
        <size>0.1</size>
      </endpoint>
    </reg>
    <reg id="18" name="IO_PORT07">
      <endpoint dir="inp" name="DetPresPasillo" type="bin">
        <position>0.0</position>
        <size>1</size>
      </reg>
  </regular>
</device>
```

```

    </endpoint>
</reg>
<reg id="19" name="IO_PORT08">
  <endpoint dir="inp" name="IntBano" type="bin">
    <position>0.0</position>
    <size>1</size>
  </endpoint>
</reg>
<reg id="20" name="IO_PORT09">
  <endpoint dir="out" name="LampBano" type="bin">
    <position>0.0</position>
    <size>0.1</size>
  </endpoint>
</reg>
<reg id="21" name="IO_PORT10">
  <endpoint dir="inp" name="IntHabitacion" type="bin">
    <position>0.0</position>
    <size>1</size>
  </endpoint>
</reg>
<reg id="22" name="IO_PORT11">
  <endpoint dir="inp" name="RegHabitacion" type="num">
    <position>0</position>
    <size>1</size>
  </endpoint>
</reg>
<reg id="23" name="IO_PORT12">
  <endpoint dir="inp" name="AlarmaCocina" type="bin">
    <position>0.0</position>
    <size>1</size>
  </endpoint>
</reg>
<reg id="24" name="IO_PORT13">
  <endpoint dir="out" name="LampCocina" type="bin">
    <position>0.0</position>
    <size>0.1</size>
  </endpoint>
</reg>
<reg id="25" name="IO_PORT14">
  <endpoint dir="inp" name="IntCocina" type="bin">
    <position>0.0</position>
    <size>1</size>
  </endpoint>
</reg>
<reg id="26" name="IO_PORT15">
  <endpoint dir="out" name="TomasCorriente" type="bin">
    <position>0.0</position>
    <size>0.1</size>
  </endpoint>
</reg>
</regular>
</device>

```

Ejemplo de proyecto de instalación domótica:

Definición de entradas y salidas

Retomando nuestra instalación de ejemplo con el plano de la vivienda planteada podemos agrupar los diferentes elementos de la instalación en 3 dispositivos panStamp:

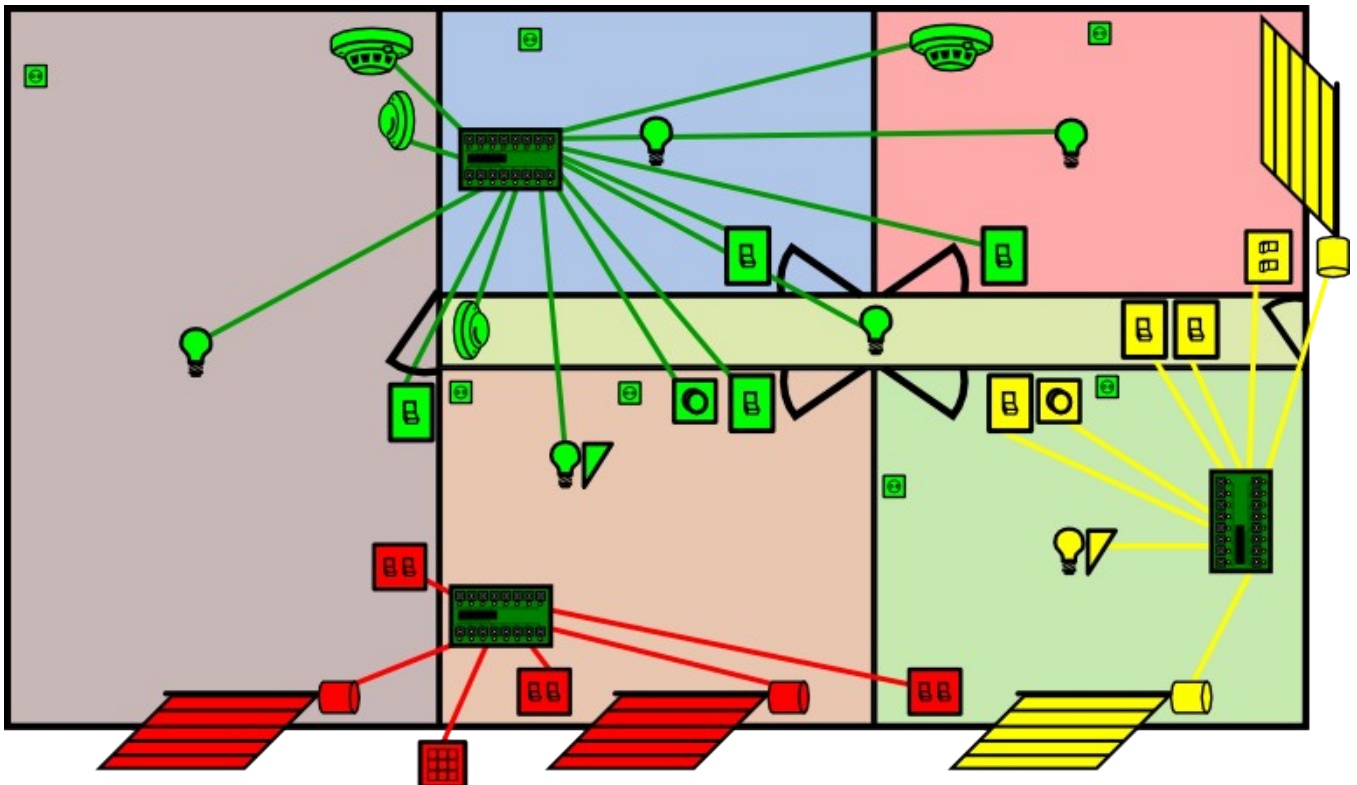


Ilustración 34: Conexión de elementos a panStamp

Para la implementación del proyecto contaremos con las tres placas de prototipos montadas, que diferenciaremos por el color de los terminales que se han montado. El primer paso será comprobar las entradas y salidas necesarias en cada módulo, definiendo la función de cada puerto y configurando los jumpers de la placa según su función:

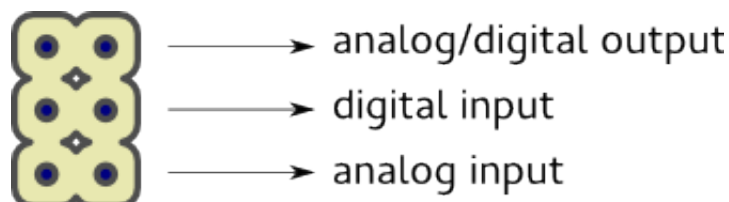


Ilustración 35: Selección de función por jumper

La primera placa contendrá los siguiente elementos:

GreenBoard	Tipo de puerto	Función
Port 00	Digital Input	Alarma incendios Parking
Port 01	Digital Input	Detector de presencia Parking
Port 02	Analog Output	Lampara regulable Habitación principal
Port 03	Digital Output	Lampara Habitación principal
Port 04	Digital Output	Lampara Parking
Port 05	Digital Input	Interruptor Parking
Port 06	Digital Output	Lampara Pasillo
Port 07	Digital Input	Detector de presencia Pasillo
Port 08	Digital Input	Interruptor Baño
Port 09	Digital Output	Lampara Baño
Port 10	Digital Input	Interruptor Habitación principal
Port 11	Analog Input	Regulador de luz Habitación principal
Port 12	Digital Input	Alarma incendios Cocina
Port 13	Digital Output	Lampara Cocina
Port 14	Digital Input	Interruptor Cocina
Port 15	Digital Output	Control de Tomas de corriente

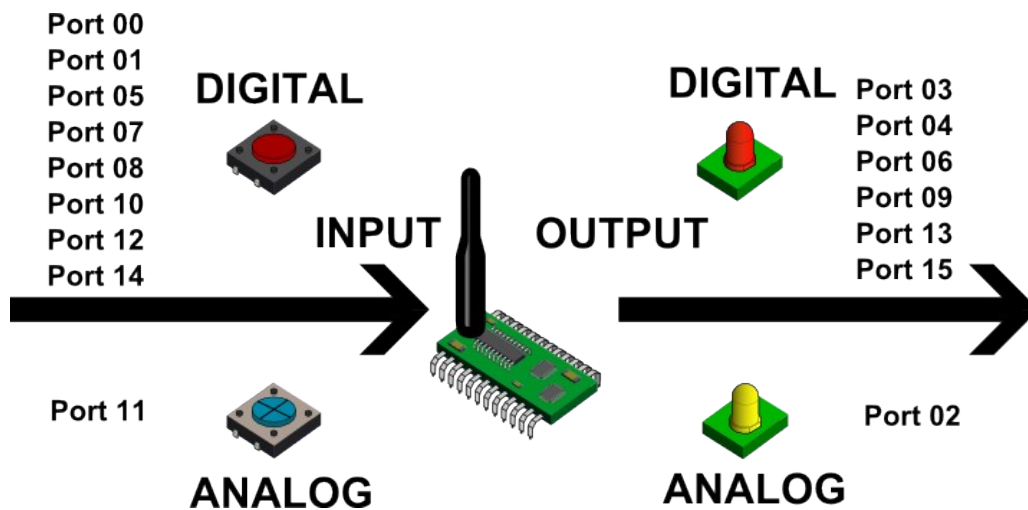
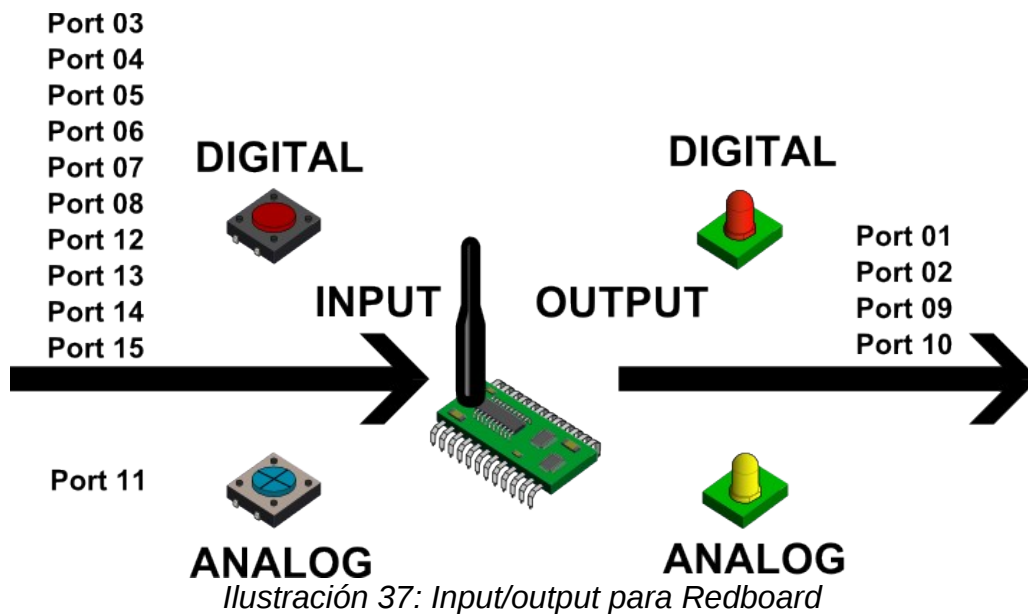


Ilustración 36: Input/output para Greenboard

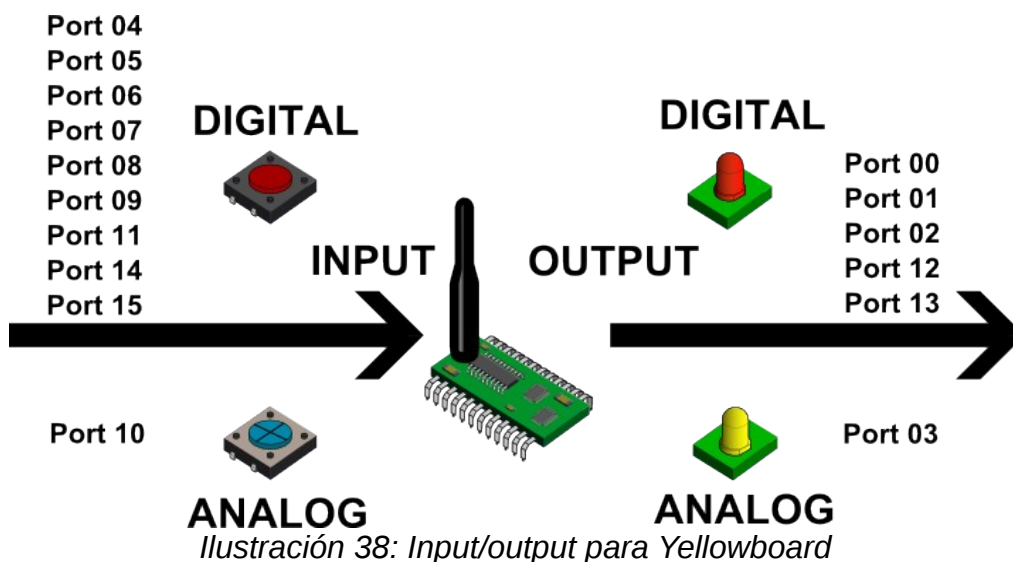
La segunda placa contendrá los siguiente elementos:

RedBoard	Tipo de puerto	Función
Port 00	No usado	
Port 01	Digital Output	Puerta Parking Up
Port 02	Digital Output	Puerta Parking Down
Port 03	Digital Input	Final de Carrera Puerta Parking Up
Port 04	Digital Input	Final de Carrera Puerta Parking Down
Port 05	Digital Input	Interruptor Puerta Parking Up
Port 06	Digital Input	Interruptor Puerta Parking Down
Port 07	Digital Input	Interruptor Persiana Comedor Up
Port 08	Digital Input	Interruptor Persiana Comedor Down
Port 09	Digital Output	Persiana Habitación Up
Port 10	Digital Output	Persiana Habitación Down
Port 11	Analog Input	Sensor de luz solar
Port 12	Digital Input	Final de Carrera Persiana Habitación Up
Port 13	Digital Input	Final de Carrera Persiana Habitación Down
Port 14	Digital Input	Interruptor Persiana Habitación Up
Port 15	Digital Input	Interruptor Persiana Habitación Down



Y la tercera placa contendrá los siguientes elementos:

YellowBoard	Tipo de puerto	Función
Port 00	Digital Output	Persiana Comedor Up
Port 01	Digital Output	Persiana Comedor Down
Port 02	Digital Output	Lampara Comedor
Port 03	Analog Output	Lampara regulable Comedor
Port 04	Digital Input	Final de Carrera Persiana Comedor Up
Port 05	Digital Input	Final de Carrera Persiana Comedor Down
Port 06	Digital Input	Interruptor Comedor
Port 07	Digital Input	Interruptor Persiana Cocina Up
Port 08	Digital Input	Interruptor Persiana Cocina Down
Port 09	Digital Input	Final de Carrera Persiana Cocina Up
Port 10	Analog Input	Regulador de luz Comedor
Port 11	Digital Input	Final de Carrera Persiana Cocina Down
Port 12	Digital Output	Persiana Cocina Up
Port 13	Digital Output	Persiana Cocina Down
Port 14	Digital Input	Interruptor Pasillo
Port 15	Digital Input	Interruptor General



Definición de eventos

Una vez creados los diferentes proyectos mediante la herramienta de generación de nodos simplemente debemos compilar cada sketch de Arduino con la configuración creada y copiar los archivos en el servidor lagarto. Con estos sencillos pasos ya podemos ejecutar el servidor lagarto con los 3 nodos en funcionamiento enviando y recibiendo información correctamente.

The screenshot shows the Lagarto web interface. At the top, there are logos for 'Lagarto by panStamp' and 'SWAP Simple Wireless Abstract Protocol'. Below the logos is a navigation menu with tabs: 'Preferences', 'Gateway', 'Wireless', 'Network', and 'Security'. The 'Endpoint panel' is active, displaying a link to 'View devices' and a table of endpoints.

ID	Location	Name	Value	Command
10.11.0	SWAP	alarmaParking_10	off	
10.12.0	SWAP	DetPresParking_10	off	
10.13.0	SWAP	LRegHabitacion_10	0	<input type="text" value="0"/> <input type="button" value="set"/>
10.14.0	SWAP	LampHabitacion_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
10.15.0	SWAP	LampParking_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
10.16.0	SWAP	IntParking_10	off	
10.17.0	SWAP	LampPasillo_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
10.18.0	SWAP	DetPresPasillo_10	off	
10.19.0	SWAP	IntBano_10	off	
10.20.0	SWAP	LampBano_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
10.21.0	SWAP	IntHabitacion_10	off	
10.22.0	SWAP	RegHabitacion_10	5	
10.23.0	SWAP	AlarmaCocina_10	off	
10.24.0	SWAP	LampCocina_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
10.25.0	SWAP	IntCocina_10	off	
10.26.0	SWAP	TomasCorriente_10	off	<input type="button" value="on"/> <input type="button" value="off"/>
25.12.0	SWAP	MParkingUp_25	off	<input type="button" value="on"/> <input type="button" value="off"/>
25.13.0	SWAP	MParkingDown_25	off	<input type="button" value="on"/> <input type="button" value="off"/>
25.14.0	SWAP	FCParkingUp_25	off	
25.15.0	SWAP	FCParkingDown_25	off	
25.16.0	SWAP	PulParkingUp_25	off	
25.17.0	SWAP	PulParkingDown_25	off	
25.18.0	SWAP	PulComedorUp_25	off	
25.19.0	SWAP	PulComerdorDown 25	off	

Ilustración 39: Interficie de las placas en Lagarto

El ultimo paso para completar nuestra instalación domótica será enlazar los diferentes elementos mediante el modulo lagarto MAX. En el siguiente ejemplo vemos como crear los eventos necesarios para controlar una lampara, por ejemplo la luz del parking.

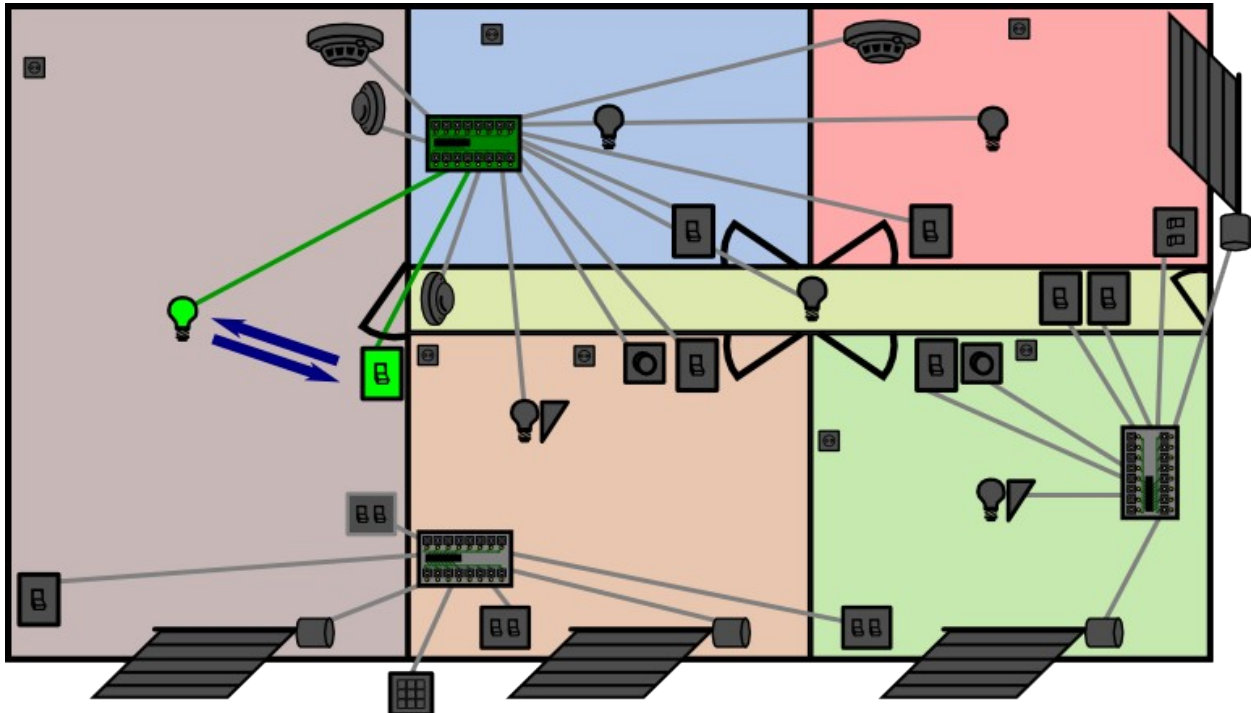


Ilustración 40: Ejemplo de evento a configurar

- Configurar el trigger que lanzará el evento para la activación.

Preferences Events Network Database Graphs Cloud Security

Event editor - Trigger condition

Event name: lampParking
 Type of trigger: Lagarto net

Network condition

Lagarto server: Lagarto-SWAP

Network endpoint: SWAP.IntParking_10

Operator: Equal to

State: on

Network value:

CANCEL SAVE

Ilustración 41: Configuración del trigger

- Configurar la acción del evento para la activación.

Preferences Events Network Database Graphs Cloud Security

Event editor - Action

Event name: lampParking
 Type of action: Lagarto network

Network action

Lagarto server: Lagarto-SWAP
 Network endpoint: SWAP.LampParking_10
 State: on
 Network value:

CANCEL SAVE

Ilustración 42: Configuración de la acción

De igual forma configuramos el trigger y la acción para la desactivación.

Preferences Events Network Database Graphs Cloud Security

Event editor

Event name: lampParkingOFF SAVE

Triggers

Lagarto-SWAP.SWAP.IntParking_10 == "off" ✕ ✎

Add trigger

Additional conditions

Add condition

Actions

Lagarto-SWAP.SWAP.LampParking_10 = off ✕ ✎

Add action

Ilustración 43: Evento de apagado

Una vez implementados ambos eventos podemos comprobar como al pulsar el interruptor de entrada correspondiente en la placa amarilla se enciende el led que simula la salida en la placa verde, y al soltarlo se desactiva.

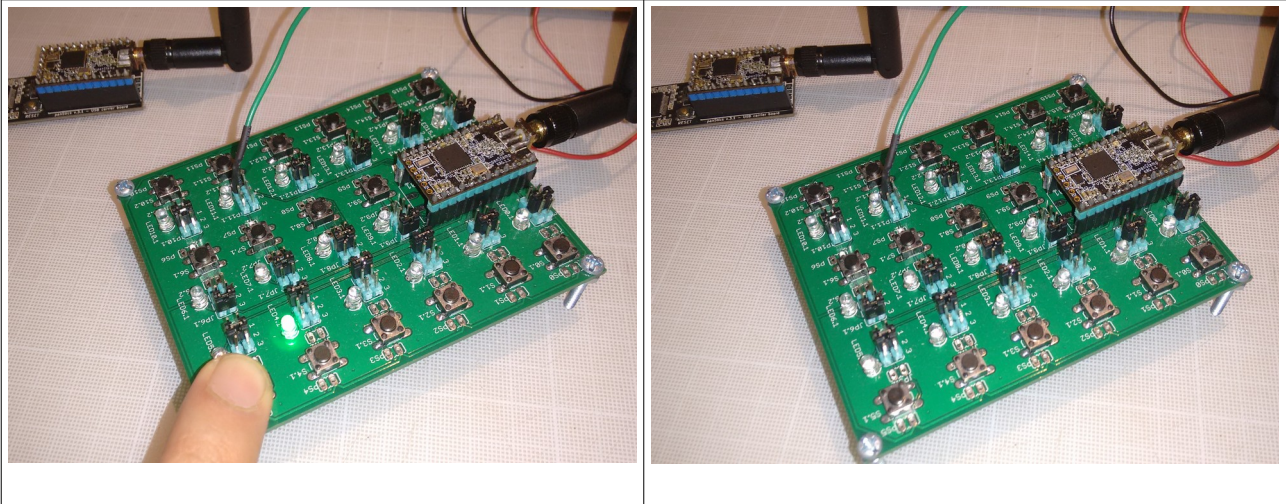


Ilustración 44: Test de funcionamiento del evento

Mediante la línea de comandos de ejecución de Lagarto podemos ver los mensajes enviados a través de la red inalámbrica:

Envío del mensaje de pulsación

```
Rved: (4E30)000A0040000A1001
Register addr= 10 id=16 changed to 01
IntParking_10 in address 10 changed to on
11 Jun 2016 10:45:10 STATUS received from Lagarto-SWAP
11 Jun 2016 10:45:10 SWAP.IntParking_10 True
```

Envío del mensaje de activación provocado por el evento

```
Sent: 0A010040020A0F80
Rved: (4E30)000A0041000A0F80
Register addr= 10 id=15 changed to 80
LampParking_10 in address 10 changed to on
11 Jun 2016 10:45:10 STATUS received from Lagarto-SWAP
11 Jun 2016 10:45:10 SWAP.LampParking_10 True
```

Envío del mensaje tras dejar de pulsar

```
Rved: (4530)000A0042000A1000
Register addr= 10 id=16 changed to 00
IntParking_10 in address 10 changed to off
11 Jun 2016 10:45:12 STATUS received from Lagarto-SWAP
11 Jun 2016 10:45:12 SWAP.IntParking_10 False
```

Envío del mensaje de desactivación provocado por el evento

```
Sent: 0A010042020A0F00
Rved: (4330)000A0043000A0F00
Register addr= 10 id=15 changed to 00
LampParking_10 in address 10 changed to off
11 Jun 2016 10:45:13 STATUS received from Lagarto-SWAP
11 Jun 2016 10:45:13 SWAP.LampParking_10 False
```

De igual manera, podemos generar eventos asociados a pulsadores condicionándolos a otras entradas. Esto sería útil para la configuración de las persianas, teniendo en cuenta los finales de carrera para activar el motor que abriría o cerraría la persiana:

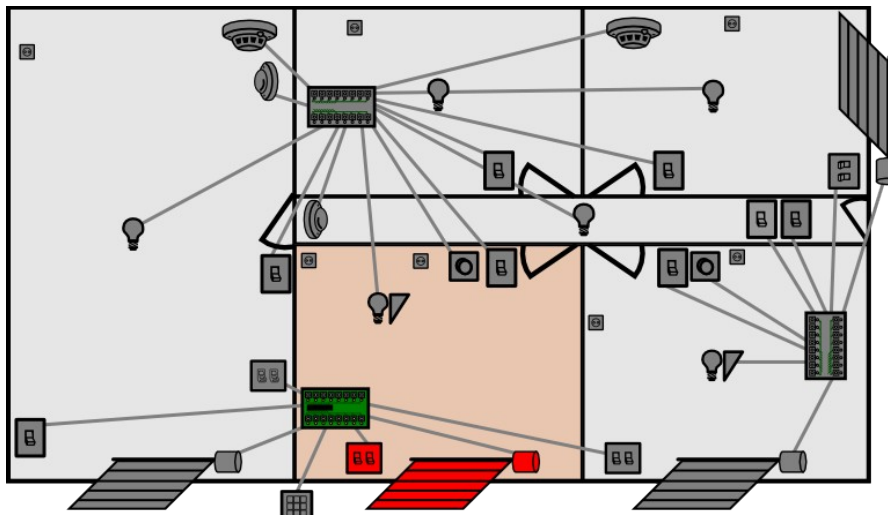


Ilustración 45: Segundo ejemplo de evento a configurar

- Configurar el trigger que lanzará el evento para la activación.
- Condicionar el evento al valor del final de carrera.
- Configurar la acción del evento para la activación.



Ilustración 46: Evento de activación de persiana

Para completar la instalación domótica habría que seguir añadiendo los diferentes eventos que se ejecutarán en nuestra red. Podemos comprobar que podemos configurar cualquier opción independientemente de la placa y registro que estemos usando. A continuación se muestra la lista completa de eventos:

Elemento activación	eventos	Elemento activado	eventos	Condición
Detector de presencia Parking	ON	Lampara Parking	ON	
Detector de presencia Parking	OFF	Lampara Parking	OFF	
Detector de presencia Pasillo	ON	Lampara Pasillo	ON	
Detector de presencia Pasillo	OFF	Lampara Pasillo	OFF	
Interruptor Parking	ON	Lampara Parking	ON	
Interruptor Parking	OFF	Lampara Parking	OFF	
Interruptor Baño	ON	Lampara Baño	ON	
Interruptor Baño	OFF	Lampara Baño	OFF	
Interruptor Cocina	ON	Lampara Cocina	ON	
Interruptor Cocina	OFF	Lampara Cocina	OFF	
Interruptor Pasillo	ON	Lampara Pasillo	ON	
Interruptor Pasillo	OFF	Lampara Pasillo	OFF	
Interruptor Habitación principal	ON	Lampara Habitación principal	ON	
Interruptor Habitación principal	OFF	Lampara Habitación principal	OFF	
Reg. de luz Habitación principal	valor	Lampara reg. Habitación principal	valor	
Interruptor Comedor	ON	Lampara Comedor	ON	
Interruptor Comedor	OFF	Lampara Comedor	OFF	
Regulador de luz Comedor	valor	Lampara regulable Comedor	valor	
Interruptor Puerta Parking Up	ON	Puerta Parking Up	ON	Final de Carrera Puerta Parking Up
Interruptor Puerta Parking Up	OFF	Puerta Parking Up	OFF	
Interruptor Puerta Parking Down	ON	Puerta Parking Down	ON	Final de Carrera Puerta Parking Down
Interruptor Puerta Parking Down	OFF	Puerta Parking Down	OFF	
Interruptor Persiana Comedor Up	ON	Persiana Comedor Up	ON	Final de Carrera Persiana Comedor Up
Interruptor Persiana Comedor Up	OFF	Persiana Comedor Up	OFF	
Interruptor Persiana Comedor Down	ON	Persiana Comedor Down	ON	Final de Carrera Persiana Comedor Down

Interruptor Persiana Comedor Down	OFF	Persiana Comedor Down	OFF	
Interruptor Persiana Habitación Up	ON	Persiana Habitación Up	ON	Final de Carrera Persiana Habitacion Up
Interruptor Persiana Habitación Up	OFF	Persiana Habitación Up	OFF	
Interruptor Persiana Habitación Down	ON	Persiana Habitación Down	ON	Final de Carrera Persiana Habitacion Down
Interruptor Persiana Habitación Down	OFF	Persiana Habitación Down	OFF	
Interruptor Persiana Cocina Up	ON	Persiana Cocina Up	ON	Final de Carrera Persiana Comedor Up
Interruptor Persiana Cocina Up	OFF	Persiana Cocina Up	OFF	
Interruptor Persiana Cocina Down	ON	Persiana Cocina Down	ON	Final de Carrera Persiana Comedor Down
Interruptor Persiana Cocina Down	OFF	Persiana Cocina Down	OFF	
Alarma incendios Parking	ON	msg		
Alarma incendios Cocina	ON	msg		
Interruptor General	OFF	Lampara Parking	OFF	
Interruptor General	OFF	Lampara Baño	OFF	
Interruptor General	OFF	Lampara Cocina	OFF	
Interruptor General	OFF	Lampara Pasillo	OFF	
Interruptor General	OFF	Control de Tomas de corriente	OFF	
Interruptor General	ON	Lampara Parking	ON	
Interruptor General	ON	Lampara Pasillo	ON	
Sensor de luz solar	<valor	Persiana Habitación Down	ON	Final de Carrera Persiana Comedor Down
Sensor de luz solar	<valor	Persiana Habitación Down	ON	Final de Carrera Persiana Habitacion Down
Sensor de luz solar	<valor	Persiana Cocina Down	ON	Final de Carrera Persiana Comedor Down

Automatización mediante node-red

Durante la realización del proyecto se han actualizado las herramientas de panStamp. El cambio mas importante ha sido a nivel interno, dejando de utilizar las librerías de gestión de mensajes ZeroMQ por las librerías MQTT, un protocolo de mensajería estándar diseñado para aplicaciones con ancho de banda limitado. También se ha actualizado la interficie visual.

No obstante el cambio mas significativo es la desaparición de Lagarto MAX, el modulo de automatización del nuestro sistema. A cambio, el nuevo sistema es capaz de utilizar sistemas mas potentes para la gestionar la lógica de nuestra automatización. Una de estas herramientas es node-red, una herramienta visual para enlazar diferentes tipos de hardware y software. Esta herramienta cuenta con acceso a los mensajes MQTT del nuevo sistema, y permite implementar funciones adicionales, ademas de mostrar la información a través de un navegador web de forma muy intuitiva y fácil de gestionar.

Así, para la instalación definitiva de nuestro sistema se usará el nuevo sistema de lagarto.

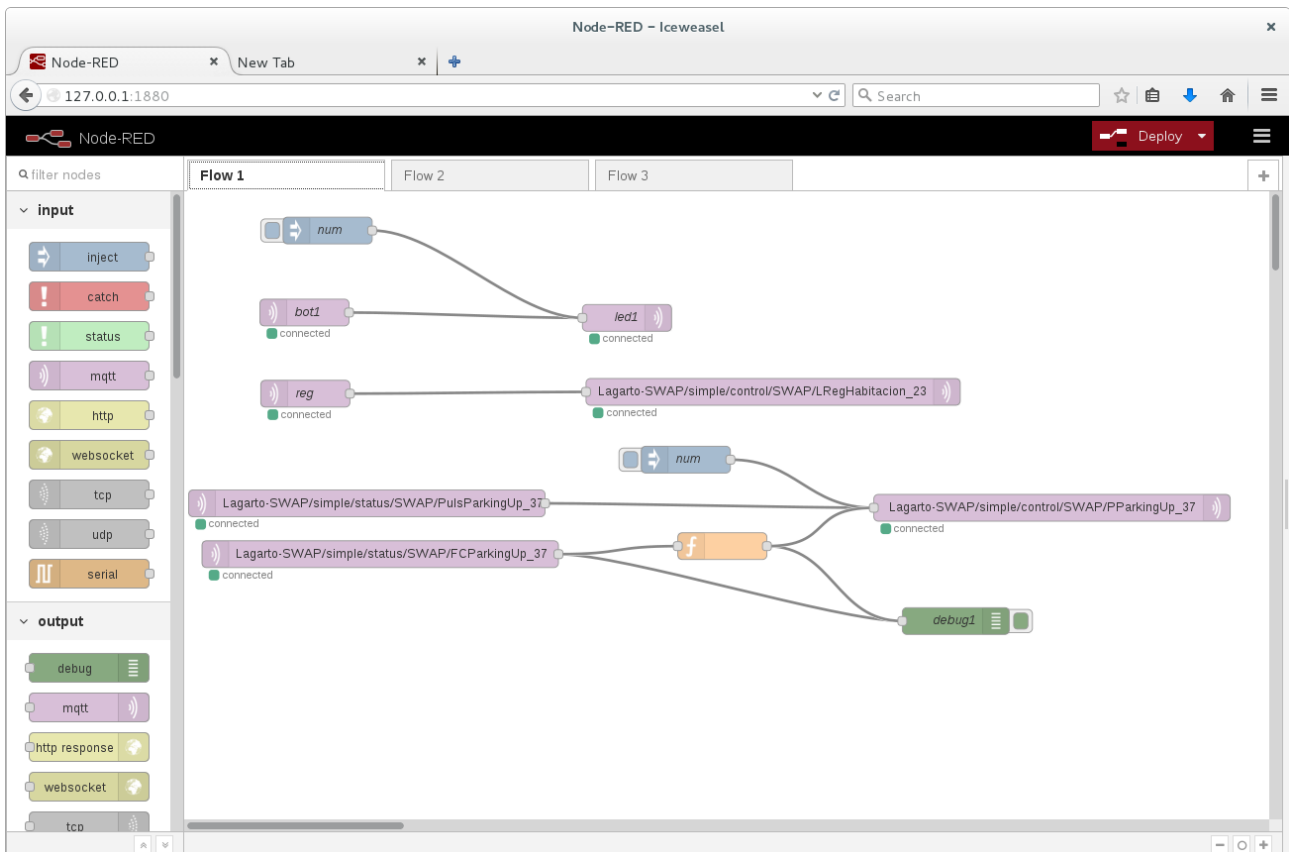


Ilustración 47: Captura de Node-red

Junto con la nueva versión de Lagarto también se han introducido cambios en interficie web. La nueva versión, aparte de utilizar unos controles gráficos mas actuales, tiene un diseño responsive para mejorar el manejo desde dispositivos móviles o tablets. La nueva versión mantiene las configuraciones de la versión anterior, aparte de añadir nuevas configuraciones para el protocolo MQTT:

ID	Location.name	Value	Command
10.11.0	SWAP.alarmaParking_10	off	
10.12.0	SWAP.DetPresParking_10	off	
10.13.0	SWAP.LRegHabitacion_10	0	
10.14.0	SWAP.LampHabitacion_10	off	
10.15.0	SWAP.LampParking_10	off	
10.16.0	SWAP.IntParking_10	off	
10.17.0	SWAP.LampPasillo_10	off	
10.18.0	SWAP.DetPresPasillo_10	off	
10.19.0	SWAP.IntBano_10	off	
10.20.0	SWAP.LampBano_10	off	
10.21.0	SWAP.IntHabitacion_10	off	
10.22.0	SWAP.RegHabitacion_10	5	
10.23.0	SWAP.AlarmaCocina_10	off	

Ilustración 48: Nuevos controles de registros

Value	Command
king_10	off
rking_10	off
acion_10	0
tacion_10	off

Ilustración 49: Nuevos menus de Lagarto

Una vez arrancados los procesos para Lagarto y node-red podemos comenzar a implementar nuestra instalación. Podemos comprobar que el proceso es mucho mas sencillo y fácil de adaptar y modificar. En el siguiente ejemplo vemos la configuración de la lampara del parking que podremos activar mediante el interruptor, o mediante el detector de presencia:

El primer paso será crear una entrada MQTT y enlazarla al registro de Lagarto correspondiente al interruptor:

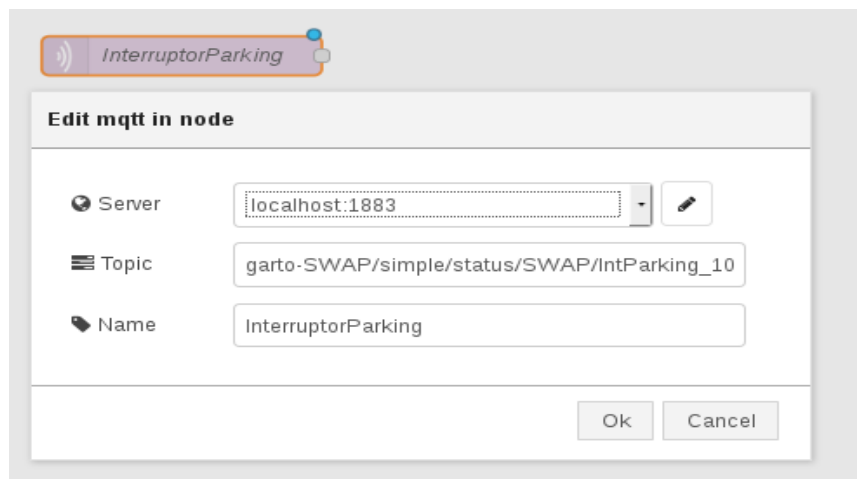


Ilustración 50: Entrada de registro por MQTT

A continuación haremos lo mismo, pero con la salida de la lampara del parking:

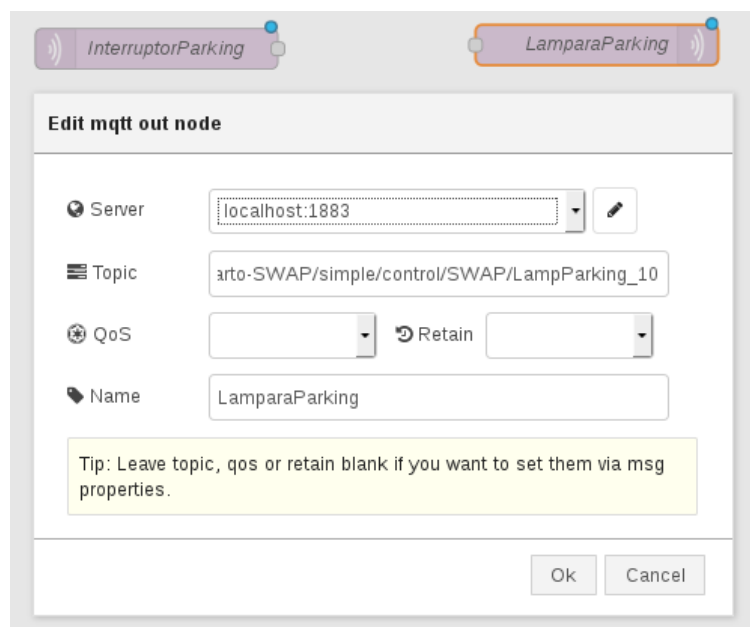


Ilustración 51: Salida a registro por MQTT

Podemos añadir mas bloques, como por ejemplo la entrada del detector de presencia y un bloque de debug que nos permitirá ver los mensajes que se están enviando en la red. El último paso será enlazar los diferentes bloques para que los mensajes se transfieran desde los bloques de entrada a los bloques de salida.

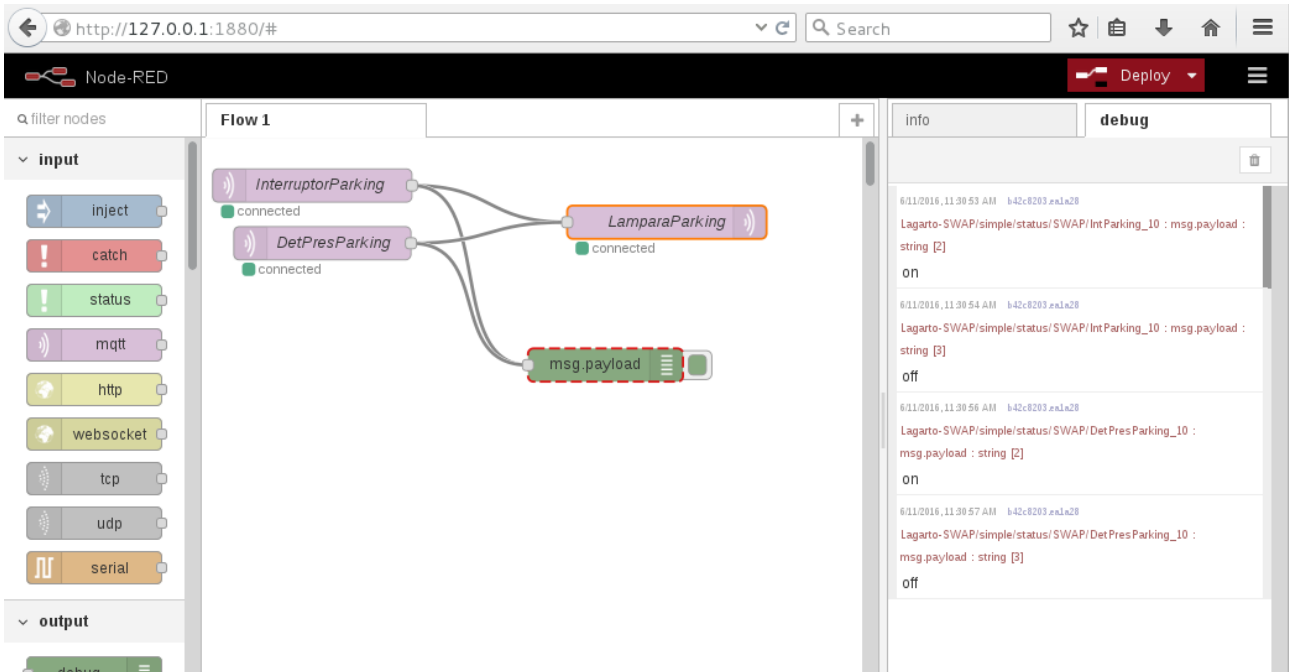


Ilustración 52: Conexión completa de luces del parking

Para finalizar podemos chequear que la implementación es correcta comprobando que la activación de las dos entradas activan la salida correspondiente:

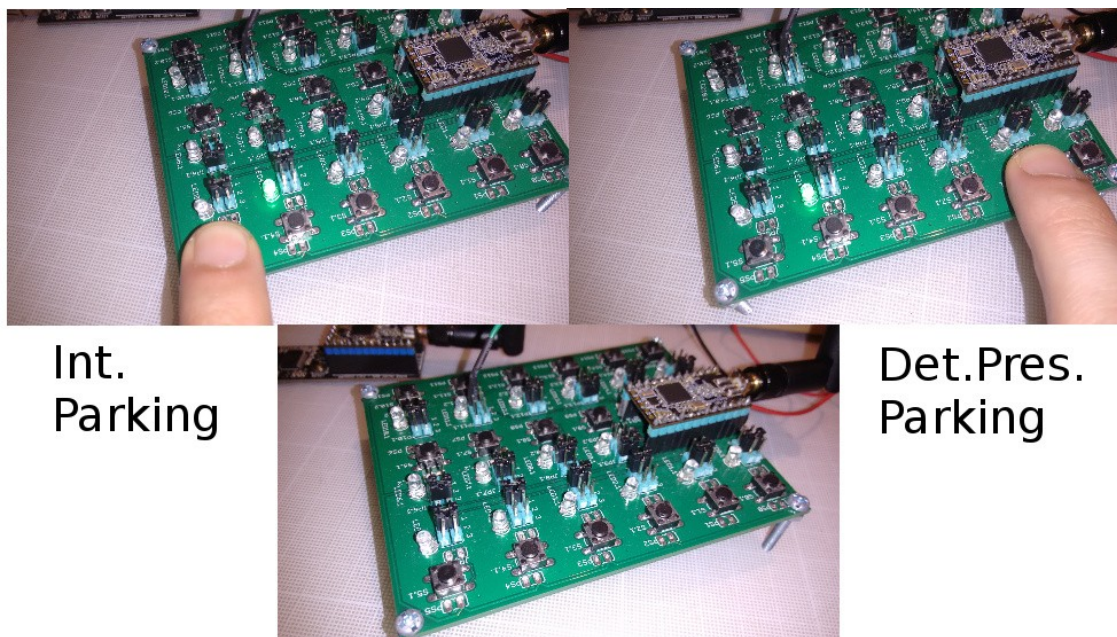


Ilustración 53: Test de funcionamiento de la conexión

También podemos comprobar el funcionamiento de sistema para el control de persianas. En el siguiente ejemplo vemos la implementación de las entradas que activan los motores para subir y bajar la puerta del parking, y la acción de los finales de carrera para desactivar dichos motores. Para que el sistema funcione correctamente añadimos un bloque función para invertir el mensaje de los finales de carrera, es decir, cuando se activa el final de carrera (mensaje ON) deberá enviar un mensaje de desactivación del motor (mensaje OFF):

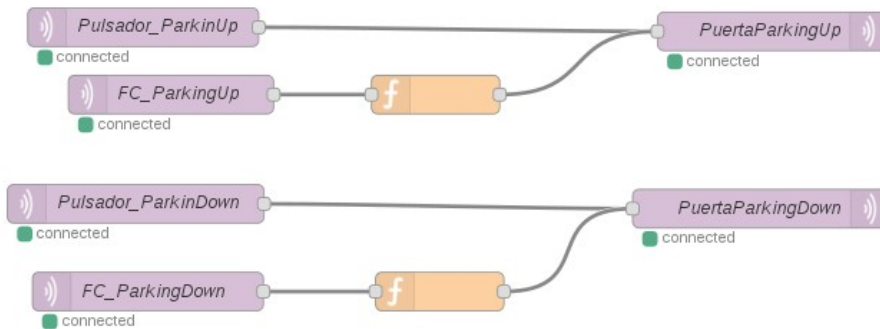


Ilustración 54: Conexión para control de persianas

En el caso de las persianas de la vivienda podemos añadir también como parámetro el valor del sensor de luz solar, lo cual nos permitiría programar la apertura y cierre de persianas por la mañana y por la noche en una instalación real. De esta forma, mediante un bloque función podemos programar la activación de las salidas a partir de ciertos valores.

Edit function node

Name:

Function:

```

1  if (msg.payload>7 & msg.payload>=context.val
2  * {
3      context.value = msg.payload
4      msg.payload="on";
5  * }
6  else
7  * {
8      context.value = msg.payload
9      msg.payload="";
10 * }
11
12 return msg;

```

Outputs: 1

See the Info tab for help writing functions.

Ok Cancel

Ilustración 55: Conexión para control de persianas con sensor de luz

Para completar la instalación simplemente habría que seguir añadiendo los diferentes registros de cada una de las placas de la instalación y realizar las conexiones pertinentes:

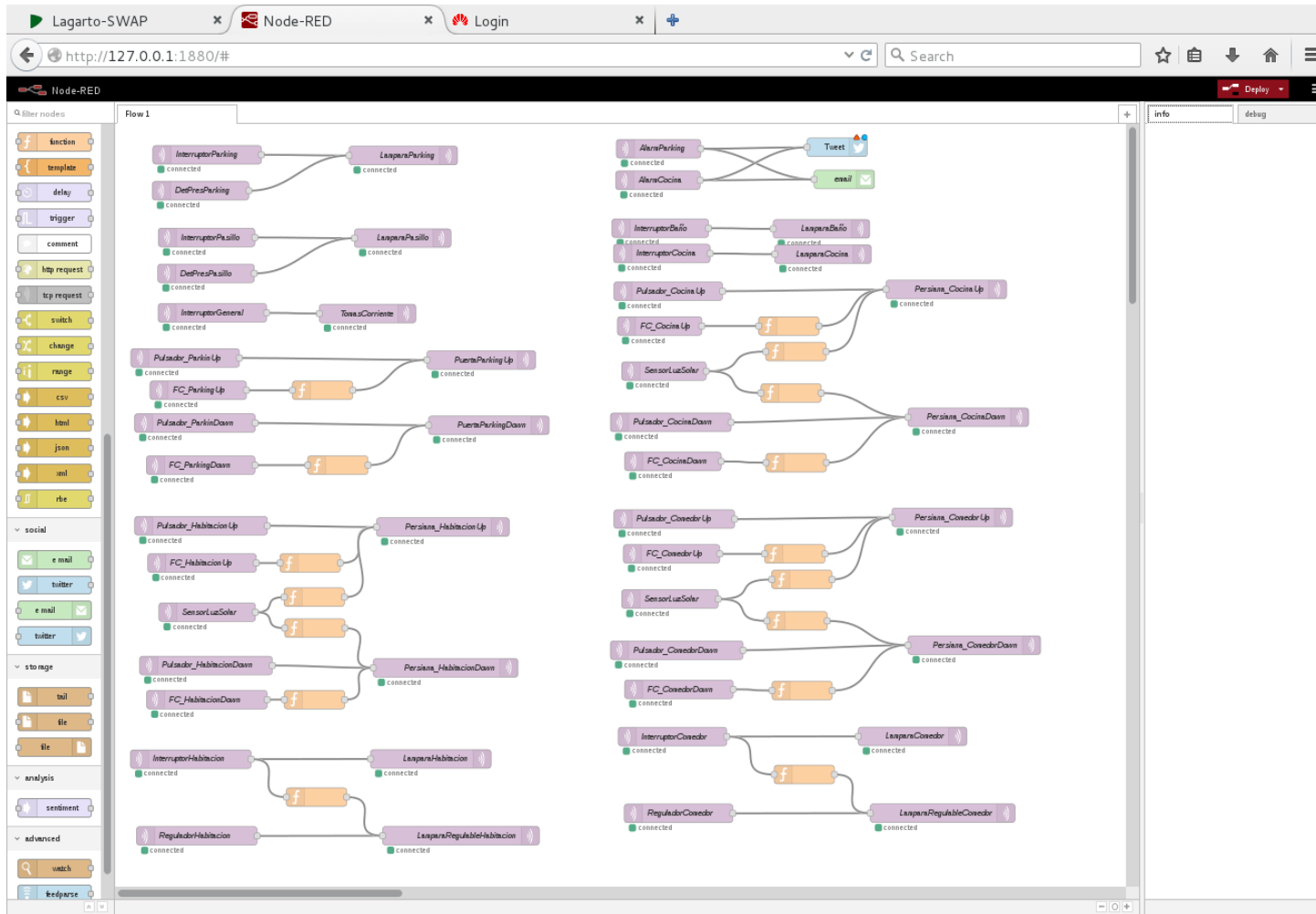


Ilustración 56: Conexión completa de elementos de la vivienda

Conclusiones

Objetivos concluidos

Una vez finalizado el proyecto es hora de revisar los objetivos iniciales y ver si se han cubierto los propósitos que nos habíamos marcado para el mismo:

El objetivo principal de este proyecto era la creación de un procedimiento para facilitar la implementación de instalaciones domóticas mediante panStamp. Creo que el objetivo principal se ha cubierto y podemos asegurar que mediante las herramientas creadas el proceso de creación de nuevas instalaciones se simplifica notablemente. Habiendo trabajado durante estos meses en este proyecto he podido comprobar que las fuentes de fallos y el número de pruebas se multiplican al involucrar tanto partes de software como de hardware, así como el hecho de trabajar con señales analógicas y digitales. El hecho de tener un sketch genérico estable y fiable nos proporciona una gran seguridad y facilidad para la detección de errores en la instalación.

Otro de los objetivos del proyecto era aplicar este procedimiento para mostrar el funcionamiento en un ejemplo práctico de instalación. Durante el proyecto se planteó una posible instalación domótica y se aplicaron las herramientas propias de panStamp, de otros desarrolladores y las creadas durante el proyecto. Creo que este objetivo también se ha cumplido y podemos comprobar en un entorno de test como funcionaría esta instalación en la realidad.

También se pretendía que la instalación fuera fácilmente modificable y escalable. Mediante la tecnología panStamp es posible agregar más nodos a nuestra red inalámbrica de forma automática, por lo que, pese a que nuestro ejemplo se componía únicamente de tres nodos, no supondría ningún obstáculo añadir nuevos nodos con nuevas funcionalidades. Del mismo modo tampoco supondría ningún inconveniente el hecho de modificar alguno de los nodos para añadir, eliminar o modificar un puerto (por ejemplo añadir nuevas lámparas o más interruptores).

En cuanto a las herramientas creadas durante este proyecto hemos de concluir que cumplen su función correctamente, pero teniendo en cuenta que hemos estado trabajando en un entorno de test. Por tanto habría que estudiar si serían necesarias algunas mejoras para la detección y activación de entradas y salidas en un entorno real. Igualmente también se podría evolucionar la herramienta de generación de nodos con una interficie más intuitiva y visual que se pudiese ejecutar en un entorno de escritorio.

Y por último, en el plano personal mis objetivos eran poner en práctica los conocimientos adquiridos durante los estudios del máster. Creo que las asignaturas cursadas y los diferentes proyectos realizados durante el máster me han permitido trasladar esta experiencia a un proyecto real, tanto en el desarrollo como en la planificación del proyecto.

Mejoras

A medida que avanzábamos en el proyecto nos hemos encontrado diferentes problemas, dificultades o mejoras que en algunos casos se podían ir aplicando a medida que se desarrollaba el proyecto, y en otros casos, bien por cuestiones de materiales o tiempo, han quedado para futuras revisiones del proyecto.

Algunas de las mejoras que se podrían implementar serían las siguientes:

- Definiciones de terminales de entradas/salidas: Al inicio del proyecto se tomaron algunas decisiones sobre la cantidad y función de terminales a usar y se realizó el diseño de hardware en función de esta definición. En una nueva revisión se podrían cambiar para aprovechar mejor los terminales disponibles como salidas PWM y entradas analógicas. Así mismo, se podrían “liberar” los pines 1.4 y 1.5 que son los que utiliza la terminal de depuración de Arduino.
- Nuevas funcionalidades: Los cuatro tipos de entradas y salidas disponibles serían las funciones básicas para entradas/salidas analógicas/digitales, pero siempre podríamos implementar nuevas funcionalidades para estas. Se podría implementar una salida temporizada, por ejemplo la lampara de un lugar de paso que una vez encendida se mantenga encendida durante un tiempo predeterminado; detectar la doble pulsación o pulsación continua de algún interruptor, etc... Obviamente estas nuevas funcionalidades se pueden implementar vía software mediante la herramienta de automatización node-red, pero sería interesante estudiar si incluir estas opciones en el sketch Arduino mejoraría el sistema.
- Lectura suave y definición de rango de lectura de sensores: Las entradas analógicas, al ser valores continuos pueden tener cierto ruido que hacen que la lectura de la señal varíe y lance mensajes a la red continuamente. Pese a que ya se ha implementado funciones para la eliminación de ruido en este tipo de entradas sería conveniente mejorar este aspecto. Además, actualmente los valores analógicos se escalan para tomar un rango de 0 a 9. Obviamente el microcontrolador nos da mucha más precisión a la hora de leer valores y se podría incluir una opción para definir el rango de valores que queremos usar dentro de la propia herramienta de generación de nodos.
- Interficie visual más intuitiva. La herramienta de generación de nodos se ha realizado en modo texto para simplificarla lo máximo posible y poder ejecutarla en diferentes entornos. Pero, dado el desarrollo de la aplicación, también se podrían crear nuevas interficies más visuales que, usando las mismas funciones para la gestión de datos y generación de archivos, nos mostrasen por pantalla las diferentes opciones de forma más usable, por ejemplo con aplicaciones de escritorio.

Estas son solo algunas mejoras que se podrían aplicar, pero evidentemente no las únicas, y puesto que los proyectos de software libre son abiertos y en constante desarrollo seguro que a medida que sigamos avanzando encontraremos nuevas ideas y líneas de mejora a nuestro proyecto.

Evolución

Durante el proyecto en todo momento hemos trabajado con placas de desarrollo en un entorno de test. Para comprobar el estado de entradas y salidas hemos estado simulando su comportamiento mediante switches y LEDs. No obstante, la idea final del proyecto es un producto completamente funcional.

El concepto final del producto sería una placa con la electrónica necesaria para alimentación de panStamp, a la que podríamos conectar diferentes módulos en función del tipo de entrada/salida a utilizar. Dependiendo de la función de cada puerto es necesaria una electrónica diferente para adaptar la señal, lo cual se ha resuelto durante nuestro proyecto con unos jumpers que habilitaban ciertas partes del circuito electrónico. En el concepto final toda esta electrónica iría añadida a los pequeños módulos a los que conectaríamos los elementos a controlar.

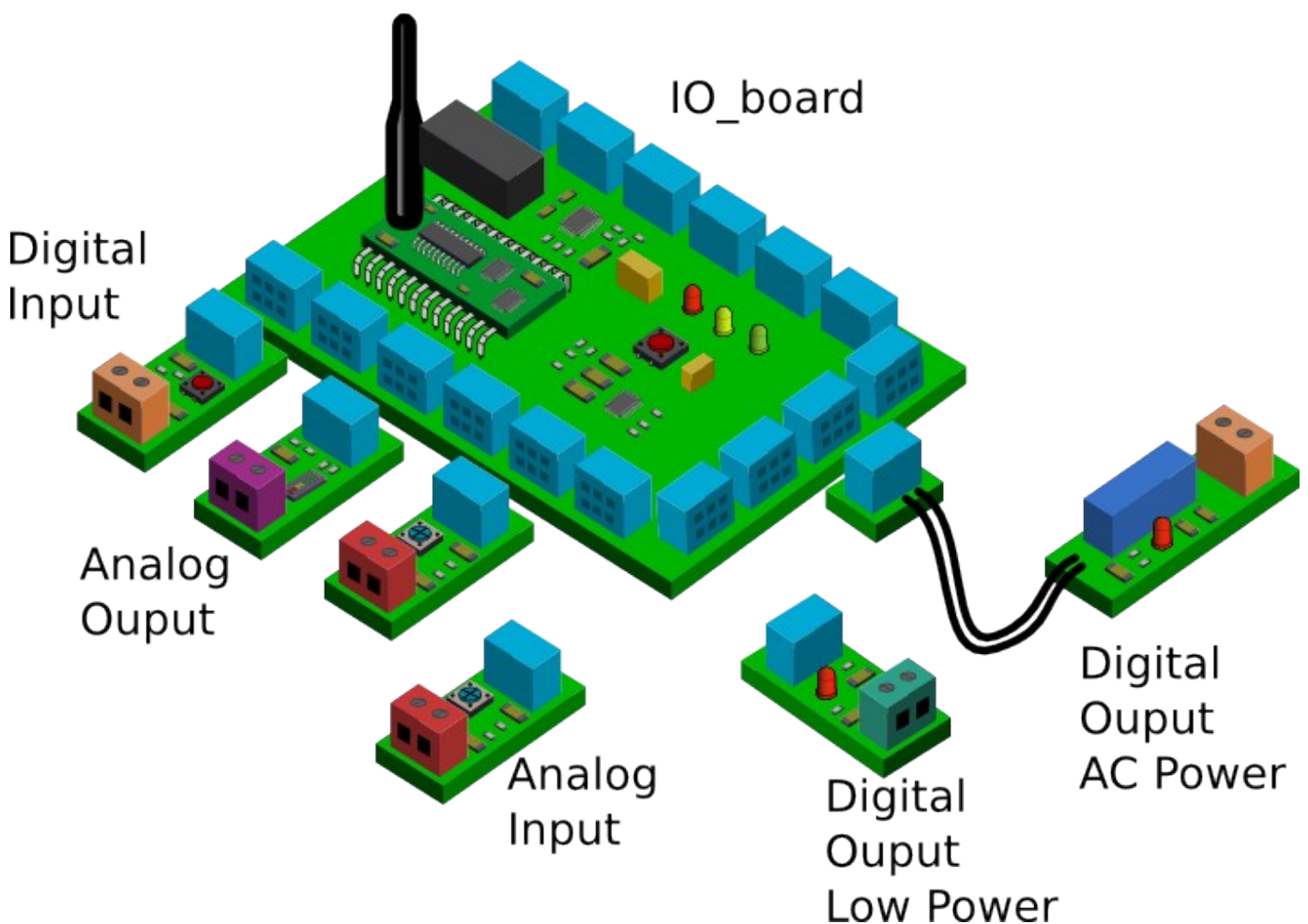


Ilustración 57: Concepto final del producto

Por ejemplo, podríamos diseñar los siguientes módulos:

- Modulo de entrada digital: Sería un modulo al que podríamos conectar un interruptor y añadiría también algún switch de testeo de la señal.

- Modulo de salida digital: Sería un modulo con un relé que activaría un circuito externo, con un led para comprobar el estado de la señal.

- Modulo de entrada analógica: Sería un modulo al que podríamos conectar diferentes tipos de sensores y que añadirían potenciómetros para adaptar la señal correctamente.

-Modulo de salida analógica: Sería un modulo con un pequeño indicador tipo vóumeter para mostrar el nivel de salida y al que conectaríamos cargas analógicas. Podría ser necesaria electrónica adicional para adaptar la señal, por ejemplo para controlar un regulador de luz de 1-10V.

En caso de trabajar con señales de alta potencia, como por ejemplo con las salidas digitales, deberíamos tener en cuenta que debemos aislar correctamente las partes de alta y baja potencia para evitar posibles accidentes de cara al usuario. En estos casos los módulos irán separados en dos partes para que se puedan instalar por una parte en placa de control y por otra en un espacio aislado para las señales a 220V

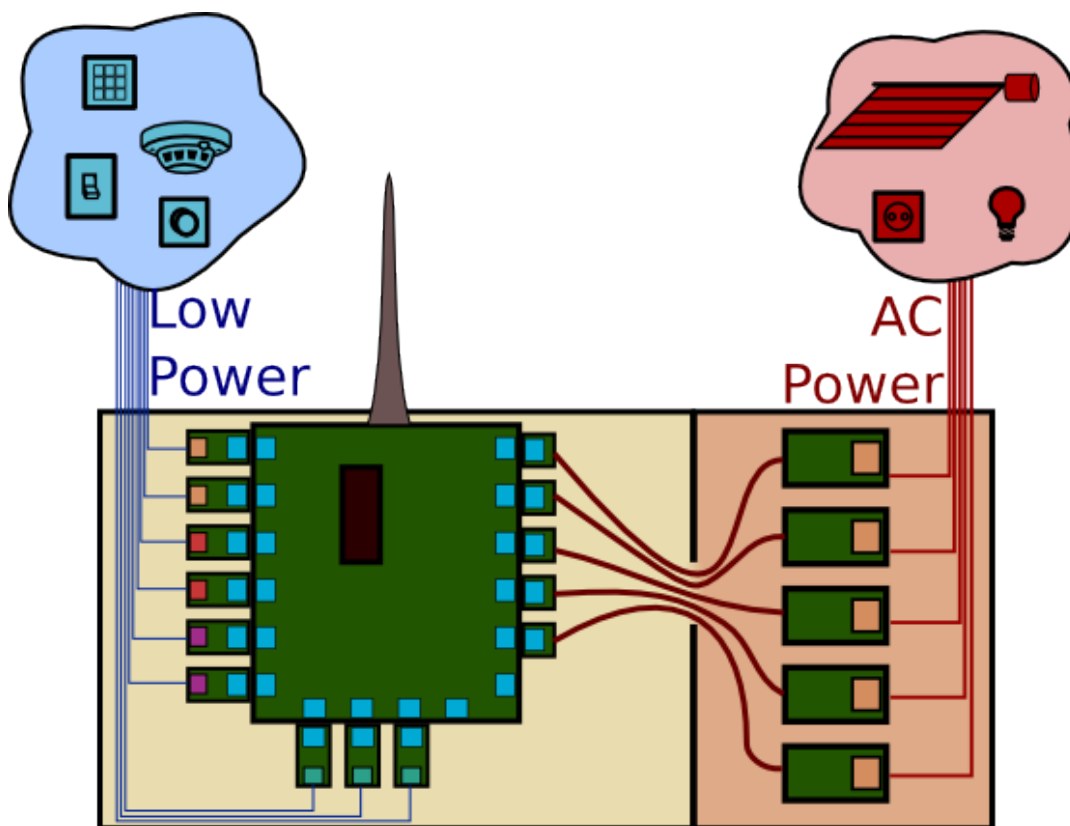


Ilustración 58: Concepto final, separación alta/baja potencia

Valoración personal del proyecto

La valoración personal del proyecto es muy buena. Creo que la realización de este tipo de proyectos como conclusión de unos estudios es muy acertado ya que permite poner en practica los conocimientos adquiridos de forma practica. Personalmente creo que el trabajo desarrollado para la conclusión del proyecto está totalmente justificado y permite valorar el esfuerzo realizado a través de las diferentes asignaturas del máster.

Este proyecto me ha permitido conocer la metodología para el desarrollo de proyectos de Software Libre y el uso de diferentes tecnologías para concluir los objetivos marcados, analizando las ventajas y desventajas de las herramientas disponibles. También he aprendido a investigar las diferentes fuentes de información disponibles en la web, discriminando la información mas útil, estudiando los problemas encontrados por otros usuarios y aplicando las soluciones mas eficientes. El hecho de tener un tutor externo para guiar las practicas me parece muy oportuno ya que puede ver el proyecto desde otras perspectivas.

Por otro lado me ha permitido conocer un proyecto muy interesante en materia domótica, involucrando tanto software como hardware libre, y los beneficios que podemos obtener de la automatización aplicada al hogar. Personalmente creo que es una tecnología con mucho futuro, pero que actualmente está muy controlado por aplicaciones privativas y con un precio muy elevado. Proyectos como panStamp nos permite abrir una pequeña ventana a que este tipos de proyectos sean mas accesibles a usuarios comunes.

Y por ultimo, la valoración del proyecto creo que es también muy positiva. Personalmente creo que puede ser una herramienta muy útil dentro del proyecto panStamp. El proyecto desarrollado ofrece muchas posibilidades a la hora de implementar diferentes tipos de proyecto, tanto si están orientados a instalaciones domóticas como si no. También creo que la idea final del producto podría ser una buena idea y comercialmente viable. Por todo lo anterior estoy muy contento por el trabajo realizado y el resultado final de estas practicas.

Valoración personal del máster

El estudio del máster me ha proporcionado diversas competencias para la realización de este proyecto y las diferentes asignaturas cursadas me han permitido programar las aplicaciones necesarias para el proyecto, planificar las diferentes tareas a realizar y configurar un sistema Linux para el correcto funcionamiento del sistema. En otro ámbito, el máster también me ha permitido conocer el ambiente del software libre y ha incrementado mi capacidad para la búsqueda información y resolución problemas gracias a los diferentes foros, wikis y demás fuentes de información.

Empecé el máster con la idea de que serían unos estudios muy técnicos, prácticos y con mucho trabajo individual pero a través de las diferentes asignaturas, trabajos, proyectos en grupo, conversaciones en los foros y mail, me he dado cuenta que en el software libre lo mas importante es la cooperación y he podido comprobar como los proyectos salen adelante gracias a la experiencia de otros usuarios, las opiniones de compañeros y la ayuda desinteresada de gente que crea documentación en los diferentes proyectos. Al iniciar el máster no tenia demasiados conocimientos sobre software libre, y al acabarlo me he convertido un firme defensor de estos sistemas que, pese a no ser perfectos, creo que tienen un presente y un futuro muy importante en el mundo de la informática.

Links:

PanStamp:

Proyecto y tienda: <http://www.panstamp.com/es/>

Descargas: <https://github.com/panStamp>

Foro: <http://www.panstamp.org/forum/>

Wiki: <https://github.com/panStamp/panstamp/wiki>

Arduino: <http://www.arduino.cc/>

Node-red: <http://nodered.org/>

Ejemplos y tutoriales:

http://www.ply.me.uk/bits_and_pieces/panStamp.html

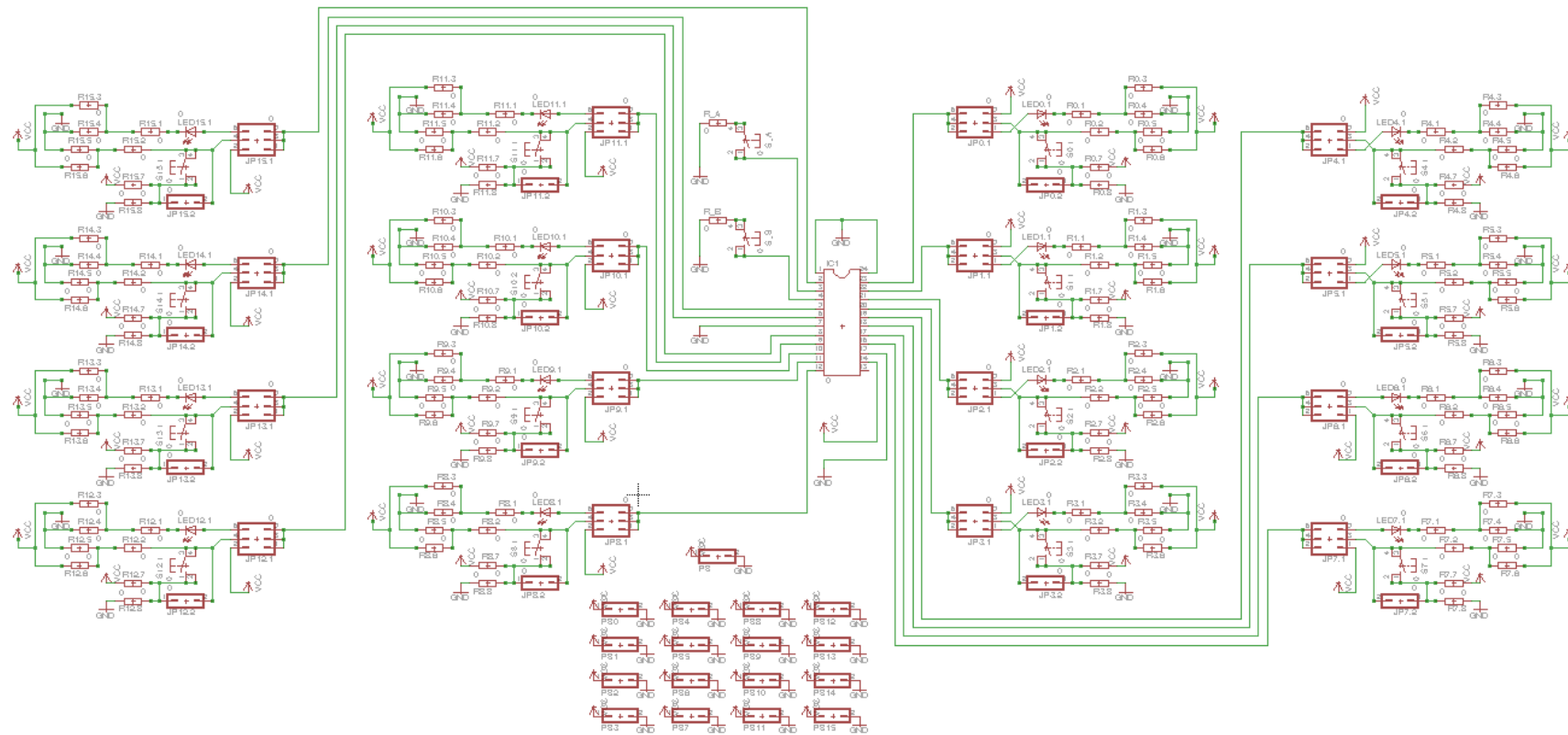
<https://www2.warwick.ac.uk/fac/sci/dcs/people/research/csujbt/panstamp/>

<http://www.dabax.net/PanStamp/>

<http://www.openremote.org/display/docs/Openremote+2.1+How+To+-+PanStamps>

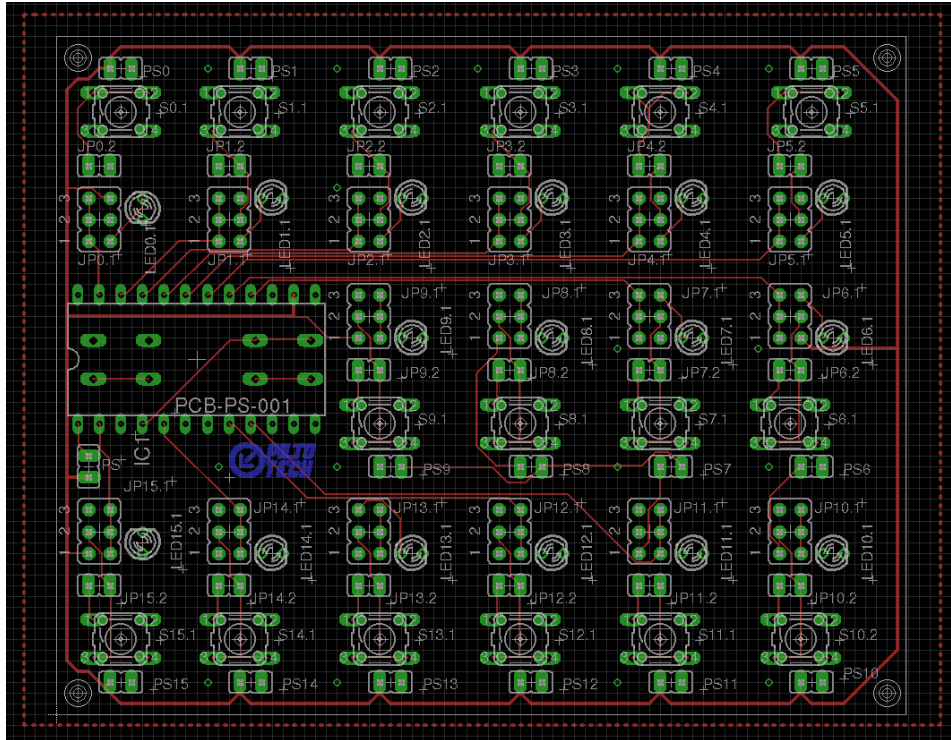
Anexos

Esquema placa



Gerber

TopLayer



Bottom Layer

