DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: DESARROLLO DE UN LENGUAJE DE MODELADO POR BLOQUES PARA LA CREACIÓN DE CÓDIGO DE FORMA AUTOMÁTICA CON PLACAS DE DESARROLLO EN EL DOMINIO IoT.

AUTOR: GARZÓN PARRA, MIGUEL OSWALDO

DIRECTOR: ING. ALULEMA FLORES, DARWIN OMAR

SANGOLQUÍ

2019

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***DESARROLLO DE UN LENGUAJE DE MODELADO POR BLOQUES PARA LA CREACIÓN DE CÓDIGO DE FORMA AUTOMÁTICA CON PLACAS DE DESARROLLO EN EL DOMINIO IoT***" fue realizado por el señor ***GARZÓN PARRA, MIGUEL OSWALDO*** el mismo que ha sido revisado en su totalidad y analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 28 de enero del 2019

ING. DARWIN ALULEMA FLORES
C.C.: 1002493334

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

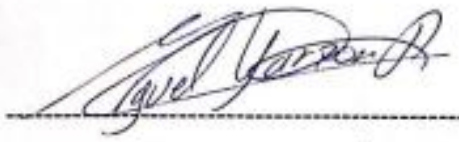
CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORÍA DE RESPONSABILIDAD

Yo, ***GARZÓN PARRA, MIGUEL OSWALDO,*** declaro que el contenido, ideas y criterios del trabajo de titulación: ***DESARROLLO DE UN LENGUAJE DE MODELADO POR BLOQUES PARA LA CREACIÓN DE CÓDIGO DE FORMA AUTOMÁTICA CON PLACAS DE DESARROLLO EN EL DOMINIO IoT*** es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 28 de enero del 2019



MIGUEL OSWALDO GARZÓN PARRA

C.C.: 1716197148

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo, ***GARZÓN PARRA, MIGUEL OSWALDO*** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: ***DESARROLLO DE UN LENGUAJE DE MODELADO POR BLOQUES PARA LA CREACIÓN DE CÓDIGO DE FORMA AUTOMÁTICA CON PLACAS DE DESARROLLO EN EL DOMINIO IoT*** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 28 de enero del 2019

MIGUEL OSWALDO GARZÓN PARRA

C.C.: 1716197148

DEDICATORIA

"Se positivo. Se feliz y no dejes que la negatividad del mundo te deprima"

Germany Kent

Este proyecto de titulación va dedicado a Dios y a la Virgen por estar siempre presente, llenándome de fortaleza, ánimo y sabiduría para enfrentar los problemas.

A mis padres Francisco Garzón y Norma Parra por saber guiarme en todo momento y en caminarme siempre por el camino del bien. Esta tesis es dedicada con mucho cariño y mucho amor hacia ustedes. A mis hermanos y hermanas por su alegría y apoyo constante.

A mis abuelos por siempre ser un apoyo incondicional, mantenernos siempre unidos y confiando siempre en el señor.

A mi hermosa familia Karen y Milena por ser el motor principal para culminar este paso en mi vida.

A mi preciosa hija Milena por ser mi princesa que alegra todos los días de mi vida y el motivo principal por el cual todos los sacrificios valen la pena.

A mi esposa quien es la que siempre me motiva a tener más metas en la vida sin su apoyo esto no hubiera sido posible.

A mi familia en general que ha estado siempre pendiente y aconsejándome en cada paso que doy, dedico esto también a los miembros de mi familia que hoy ya no están con nosotros pero que siempre están brindándome su cariño y protección.

Al Ing. Darwin Alulema y Ing. Darwin Aguilar, por haber sido parte de mi formación universitaria, por su paciencia y entrega total en su labor.

AGRADECIMIENTO

Agradezco a mi madre "Norma (Parrota)" por ser una mujer incondicional y ejemplo de vida, que nos enseñó la bondad, y siempre dar una mano a quien lo necesita, a ser muy agradecidos en la vida. A mi padre "Francisco" que me ha enseñado que la sencillez, humildad y el amor sencillo son parte esencial de una persona, son pequeñas cosas que me motivaron a ser el hombre que soy ahora.

A mi hermana Silvia por enseñarme siempre a ver el lado positivo de las cosas. A mi hermana Anita le agradezco por enseñarme a que todo en la vida se consigue con esfuerzo y dedicación. A mi hermano Daniel por su gran ejemplo de ser familia, de cómo ser buen hijo, buen padre y buena persona con los demás. A mi hermano Cristian, por ser siempre un apoyo con quien puede contar en las buenas y en las malas.

A mis sobrinos Cristino, Emi, Enano, Mile, Emilio, Danito, Cami, Dani, Diego les agradezco con todo mi corazón por que antes de ser padre ya les consideraba mis hijos.

A mi princesa Milena le agradezco siempre porque siempre buscas la forma de alegrarnos con tus pequeñas travesuras, tus juegos, tu sonrisa y encantos, espero siempre estar a tu lado y que puedas buscarme y encontrarme siempre que tú lo necesites. Por todo esto y lo que vendrá te amo y te amare.

A mi esposa y compañera de vida le agradezco por ser siempre una amiga y darme animo en los buenos y malos momentos, agradezco a la vida por tenerte se que la vida nos mantendrá juntos para recorrer los tantos momentos que la vida nos tiene por delante.

A mis cuñados y cuñadas, abuelos y abuelas, tíos y tías, primos y primas, a mi familia en general les agradezco por siempre estar unidos y enseñarme que la familia es familia donde sea que este.

Miguel Oswaldo Garzón Parra

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El Internet en la actualidad está siendo parte fundamental de la vida cotidiana, se encuentra en la gran mayoría de aparatos tecnológicos que se utilizan a diario por ejemplo smartphones, tabletas, etc. Esta integración en la vida de las personas es lo que se denomina el internet en las “cosas”. Pero su gran complejidad al momento de desarrollar aplicaciones en (IoT) se debe a la gran heterogeneidad de las plataformas. Esto dificulta el desarrollo de nuevas aplicaciones debido a que los desarrolladores deben tener conocimientos de varias plataformas. De esta manera en el presente proyecto de investigación se propone un DSL y un Editor Gráfico, para generar código de forma automática mediante transformaciones Model – To Text (M2T), para tarjetas de desarrollo Arduino. En el diseño de la propuesta se utilizó EMF, Sirius y Acceleo. Para validar el funcionamiento, la metodología se ha empleado técnicas de Scaffolding para guiar a los nuevos desarrolladores que no tienen mayor conocimiento en el desarrollo de aplicaciones en IoT. Finalmente se realizaron pruebas en el DSL, en el cual se puede evidenciar que puede ser escalable, y que el código de la aplicación es generado correctamente y de manera rápida.

Palabras claves:

- **LENGUAJE DE DOMINIO ESPECIFICO (DSL)**
- **ECLIPSE MODELING FRAMEWORK (EMF)**
- **SIRIUS**
- **ACCELEO**

ABSTRACT

The Internet is currently a fundamental part of everyday life, is found in the vast majority of technological devices that are used daily for example smartphones, tablets, etc. This integration in the lives of people is what is called the internet in the "things". But its great complexity when developing applications in (IoT) is due to the great heterogeneity of the platforms. This hinders the development of new applications because developers must have knowledge of various platforms. In this way, in the present research project, a DSL and a Graphical Editor are proposed to automatically generate code through Model - To Text (M2T) transformations, for Arduino development cards. EMF, Sirius and Acceleo were used in the design of the proposal. To validate the operation, the methodology has been used Scaffolding techniques to guide new developers who have no greater knowledge in the development of applications in IoT. Finally tests were carried out in the DSL, in which it can be evidenced that it can be scalable, and that the code of the application is generated correctly and quickly.

Keywords:

- **SPECIFIC DOMAIN LANGUAGE (DSL)**
- **ECLIPSE MODELING FRAMEWORK (EMF)**
- **SIRIUS**
- **ACCELEO**

CAPÍTULO 1

1. INTRODUCCIÓN

1.1. Motivación

El ritmo con el que avanza la tecnología es difícil de alcanzar ya que cada minuto en el mundo alguna persona está innovando o desarrollando nuevas aplicaciones o proyectos que satisfagan una necesidad o simplemente haga la vida del ser humano mucho más placentera. Pero para eso está la tecnología descubierta por los seres humanos para hacer uso de ella por los mismos, por lo tanto, he ahí la razón de ya no cubrir una necesidad sino de atesorar la vida. Para cumplir con este fin es importante poner la tecnología en marcha y de esta manera salvaguardar al hombre.

En la actualidad el desarrollo de aplicaciones relacionadas con Internet de las cosas (abreviado del IoT, por sus siglas en inglés *Internet of Things*) ha hecho que la vida del ser humano se facilite, ya que a través de estas se puede tener el control y seguridad de hogares, oficinas, etc; así como nuevos métodos o modelos de aprendizaje. Por ello en el presente proyecto de investigación se pretende desarrollar un lenguaje de modelado por bloques para la creación de código de forma automática con placas de desarrollo (arduino) en el dominio IoT.

De este modo, se trabaja en un sistema capaz de establecer un método de enseñanza de lenguajes de programación propia para placas de desarrollo, pero basado en lenguajes de dominio especifico (abreviado de DSL, por sus siglas en inglés *domain specific language*). Este proyecto de investigación asociado a un trabajo final de grado aprovecha conocimientos adquiridos en el grado de Ingeniería Electrónica y Telecomunicaciones como programación, electrónica básica, circuitos digitales, circuitos de control con sensores y actuadores, así como se encuentra basado

en proyectos de titulación que hacen referencia al desarrollo de prototipos de DSL basado en ingeniería de modelos.

1.2. Justificación e Importancia

El presente proyecto de investigación se encuentra enmarcado con el desarrollo de una herramienta didáctica del IoT, para poder satisfacer las necesidades de las personas que están iniciando en el mundo de la programación de placas de desarrollo y aplicaciones. Permitiendo de esta manera disponer de una herramienta fácil para crear aplicaciones sin tener que detallar conceptos avanzados sobre el lenguaje de programación. Además, al aplicar técnicas de Scaffolding en el diseño del proyecto el usuario se centrará en crear aplicaciones básicas de una forma gráfica sin necesidad de emplear un lenguaje como C++ o JAVA, sino más bien una estructura de bloques.

Dentro de las ventajas que presenta el proyecto se puede encontrar la parte didáctica ya que no es necesario un gran conocimiento en programación para crear aplicaciones IoT, donde la conectividad es muy importante por lo que se han considerado esquemas de comunicación WiFi, Bluetooth y RS232, además de una serie de actuadores y sensores. Por otro lado, permite la capacidad de crear herramientas como editores gráficos dentro de Eclipse o como una aplicación RCP *(Rich Client Plattform)* bajo la licencia Creative Commons.

El desarrollo del presente proyecto de investigación es importante ya que este tiene como finalidad disponer de una herramienta que acelere el proceso de creación de aplicaciones en hardware abierto para IoT, sin necesidad de tener mayor conocimiento de programación. Por otro lado va ayudar a unir dominios de especialización, esto quiere decir que se va integrar dos mundos muy importantes que son el hardware y software, por un lado, el hardware que se

encarga de armar la estructura de la aplicación y el software que se encarga de definir las características que va a tener cada aplicación.

Es importante, mencionar que si bien no es necesario conceptos de programación para desarrollar este tipo de aplicaciones que están basados en modelo *scaffolding*, es requisito tener conocimientos básicos de modos de operación de componentes electrónicos (elementos activos) para poder comprender y desarrollar aplicaciones desde la más básica hasta la más compleja.

1.3. Objetivos

1.3.1. General

Desarrollar un lenguaje de modelado por bloques para la creación de código de forma automática con placas de desarrollo en el dominio de IoT implementando ingeniería de modelos y scaffolding.

1.3.2. Específicos

- Investigar el estado del arte de la ingeniería de modelos y scaffolding.
- Investigar herramientas y software para el Model Driven Architecture (MDA) y MDE.
- Diseñar la Arquitectura del Meta-modelo.
- Definir el proceso de scaffolding.
- Implementar las herramientas gráficas y el motor de generación de código sobre placas de desarrollo.
- Evaluación de usabilidad y el proceso de aprendizaje.
- Documentar la investigación realizada.

1.4. Estado del arte

Para determinar el estado del arte se empleó el método de Revisión Sistemática de la literatura (SLR), (Moguel Marquez, 2018), con el fin de conocer cuáles son las temáticas que se encuentran mejor desarrolladas en base al proyecto de investigación y poder establecer que características necesitan un mayor enfoque el cual aporte a esta investigación y a futuras.

Tabla 1

Preguntas de investigación para la Revisión Sistemática de la literatura

Preguntas de investigación	Motivación
SLR1: ¿Qué tecnología es la más adecuada para desarrollar aplicaciones para IoT con placas de desarrollo?	Conocer los módulos de conexión más adecuados de acuerdo a la placa de desarrollo, para la generación de aplicaciones para IoT.
SLR2: ¿Cuál es la relación que existe entre un DSL con el IoT	Conocer la interacción de los elementos para envió y adquisición de datos

Por lo tanto se partió de la idea de buscar artículos científicos en las diferentes bases de datos disponibles como, *Google Scholar, Dialnet, Scopus y Elsevier;* en estos motores de búsqueda se encuentran una gran cantidad de documentos publicados en áreas como informática, tecnologías de información y comunicaciones. Asimismo, el diseño de la investigación se basa en:

- **Bibliográfica:** se recaba información de diversas fuentes como libros, web, tesis de pregrado y doctorales, etc., todo esto enfocado en el tema del proyecto. Por esta razón se utilizó estas fuentes para desarrollar el marco teórico en relación a lenguaje de dominio específico, internet de las cosas, ingeniería de modelos, *Scaffolding* y tecnologías de comunicación, representando la premisa de la investigación para tener conocimiento específico del tema y desarrollar el proyecto. (Cegarra, 2004).

- **De campo:** en la que el investigador actúa y obtiene información del objeto de estudio, por lo que durante el desarrollo del proyecto se encarga de revisar que las aplicaciones que se desarrollan se ajustan a las características del IoT con el objetivo de cumplir el objetivo del proyecto. (Cegarra, 2004).

1.4.1.1. Selección y clasificación de artículos

Durante la búsqueda de documentos se tuvieron en cuenta publicaciones en revistas arbitradas, cubriendo los trabajos del 2010 hasta noviembre del 2018 (momento en que se finaliza el mapeo). Este período de tiempo fue seleccionado tratando de ubicar estudios de los últimos años con tecnología actual, al mismo tiempo la documentación adquirida fue evaluada por el investigador que lleva a cabo el presente estudio para clasificar (incluir o excluir documentos) cada uno de los estudios referentes a los temas mencionados en el apartado 1.4.1.4. Los documentos que fueron excluidos e incluidos se detallan a continuación:

- Excluidos
    - Artículos que no tengan relación con DSL, MDE, IoT y *Scaffolding*.
    - Presentaciones power point y prezi.
    - Blogs de la web y videos tutoriales.
- Incluidos
    - Artículos completos de *Google Scholar, Dialnet, Scopus* o *Elsevier* y que pertenezcan a la rama de Ingeniería o tecnologías informáticas.
    - Cualquier tipo de publicación, revista, conferencia, tesis de pregrado o doctoral.
    - Publicaciones hasta diciembre del 2017.

A continuación se presenta los artículos tomados en cuenta:

Existen algunos proyectos sobre la ingeniería de modelos (Muñoz Ramirez, Fernandez Lozano, & Gomez de Gabriel, 2016) en su artículo Ingeniería basada en modelos en prácticas de robótica, se recoge la experiencia de usar una herramienta de desarrollo de ingeniería basada en modelos o MDE (Model-Driven Engineering) para la realización de clases prácticas de robótica. La principal característica de los enfoques es que se prescinde de la fase de programación, por lo que se puede pasar del diseño a la implementación de manera automática permitiendo dedicar más tiempo a los objetivos del aprendizaje. Para determinar la eficiencia de estas herramientas se ha utilizado una plataforma robótica móvil y se han realizado experimentos con alumnos utilizando Simulink con generación de código de Arduino y se han medido los resultados de manera subjetiva y objetiva. (Muñoz Ramirez, Fernandez Lozano, & Gomez de Gabriel, 2016)

Según: (R. Brisaboa, Cortinas, R. Luaces, & Pedreira, 2016) en su artículo aplicando Scaffolding en el desarrollo de líneas de producto software, las Líneas de Producto Software (LPS) constituyen una tecnología madura para producir software que ha sido objeto de una gran cantidad de investigación, por lo que existen numerosas técnicas, metodologías y herramientas para crearlas. Sin embargo, es complicado utilizar algunas de estas herramientas en la industria debido a factores como la rápida evolución que han tenido los entornos de desarrollo, lo que provoca que estas herramientas estén obsoletas, la falta de soporte para proyectos que utilizan diferentes lenguajes de desarrollo, o la dificultad en el mantenimiento del código de los productos generados por la LPS. (R. Brisaboa, Cortinas, R. Luaces, & Pedreira, 2016)

Según (Castillejo Parrilla, 2015) el paradigma de IoT es actualmente una realidad y existe una creciente disponibilidad de plataformas comerciales destinadas a ser utilizadas como la columna vertebral de los ecosistemas de IoT. Estos dispositivos son lo suficientemente potentes

para controlar una red de sensores completa y conectarla a Internet. De esta manera, las aplicaciones pueden acceder a los datos adquiridos por los sensores y controlar los actuadores conectados a la red. (Castillejo Parrilla, 2015)

Según (Castillejo Parrilla, 2015) las tecnologías de comunicación inalámbrica Existe un número creciente de protocolos y versiones actualizadas utilizadas en aplicaciones funcionales de redes de sensores inalámbricos. La información recopilada por los sensores aumenta su valor si se puede acceder fuera de la WSN. Por esta razón, la conexión de WSN a Internet es una característica clave. De esta manera, cada dispositivo con conexión a Internet podrá recuperar de forma remota la información recopilada por los sensores y utilizar estos datos para proporcionar aplicaciones útiles para el usuario, por ejemplo: domótica, salud electrónica, sistemas de ahorro de energía, seguridad, defensa, automatización, ciudades inteligentes, etc. (Castillejo Parrilla, 2015)

Por otra parte, la popularidad de la técnica de Scaffolding no ha parado de aumentar entre los desarrolladores de software desde que apareció hace unos años, a pesar de recurrir a alternativas poco valoradas en la academia tales como el uso de preprocesadores. En este trabajo se propone la utilización de la técnica de Scaffolding para implementar una LPS, lo que permite superar algunas de las limitaciones clásicas de otras herramientas LPS

Una aplicación de IoT se tiene el ejemplo de Hello Barbie. Un juguete conectado en el cual se incorpora muchos de los mismos elementos que una aplicación de IoT industrial o médica "adulta": un nodo de borde de baja potencia con sensores; una interfaz hombre-máquina (HMI); conectividad inalámbrica; recopilación y análisis de datos en la nube que conducen a conocimientos accionables; e incluso inteligencia artificial (AI). (Pickering, 2017)

1.4.1.2. Resultados

Tabla 2
Respuestas

Pregunta	Respuesta
SLR1: ¿Qué tecnología es la más adecuada para desarrollar aplicaciones para IoT con placas de desarrollo?	La tecnología más adecuada para las placas de desarrollo en la construcción de aplicaciones en IoT, es WiFi 802.11x, por la cantidad de módulos que se tienen al alcance.
SLR2: ¿Cuál es la relación que existe entre un DSL con el IoT?	Tener un DSL robusto ,escalable, amigable y accesible con el IoT

Como se ha determinado anteriormente hay una gran variedad de nuevas aplicaciones y arquitecturas que proporcionan acceso al IoT. Por lo que de acuerdo con esto se pretende proponer la creación de un meta-modelo basado en la creación de aplicaciones IoT que se ejecuten en una placa desarrollo y cuya metodología de enseñanza se base en un proceso de Scaffolding, que permita al usuario crear aplicaciones sin un mayor conocimiento en programación.

CAPÍTULO 2

2. FUNDAMENTO TEÓRICO Y CONCEPTUAL

2.1. Internet de las cosas (IoT)

El internet de las cosas por sus siglas en ingles IoT se define como la consolidación a través de la red de redes de una "red" que aloja una gran multitud de objetos o dispositivos, es decir, poder tener conectada a esta todas las cosas como podrían ser vehículos, electrodomésticos, dispositivos mecánicos, o simplemente objetos tales como calzado, muebles, maletas, dispositivos de medición, biosensores, o cualquier objeto que se pueda imaginar. (Domodesk 2013)

2.1.1. Definición IoT

Al IoT (por sus siglas del inglés *Internet of Things* que traducido al español significa Internet de las cosas) le han asignado varias definiciones las cuales se detallan a continuación.

- CASAGRAS.- Lo define como una red de alcance global, que abarca elementos físicos y virtuales, brindará la información de los objetos y sensores incluyendo su capacidad de interconexión.
- CERP-IoT (*Cluster of European Research projects con the Internet of Things*).- Lo define como una infraestructura de red con alcance global, interactivo que se puede auto configurar, con la capacidad de dar a los objetos físicos y virtuales atributos virtuales, utilizando estándares y protocolos de comunicación con interfaces inteligentes (Peña & Suquillo, 2016).
- IERC (*European Research Cluster for the Internet of Things*) junto a la ITU (*Telecomunication Standardization*). - Lo definen Como una infraestructura mundial capaz de brindar información de la interconexión de elementos físicos y virtuales gracias

a la funcionabilidad de las tecnologías de comunicación e información (Peña & Suquillo, 2016).

Todas las definiciones para IoT que se mencionan tienen algo en común y es que esta es una red de alcance mundial, la cual se basa en la interpretación de la interconectividad entre los objetos virtuales y los físicos, por lo que se lo podría resumir en ese concepto.

2.1.2. Visión y descripción técnica del IoT

El internet de las cosas o IoT, es la tercera fase de la revolución del internet, en la Figura 1 se puede divisar algo del avance del IoT, en esta etapa se pretende conectar a objetos de la vida cotidiana (físicos) con el mundo virtual, para que estos sean capaces de detectar las distintas situaciones de su alrededor, teniendo la capacidad de responder ante ellas, creando ambientes inteligentes, un ejemplo de los avances que esta tecnología puede brindar es conectar un envase de medicamentos, de tal manera que este indique si alguien se ha olvidado de tomar su medicamento (Peña & Suquillo, 2016).

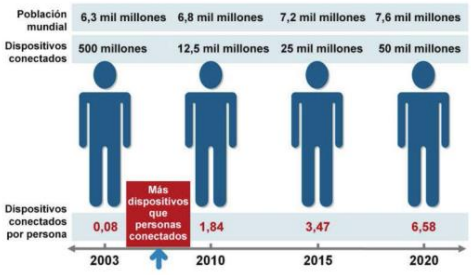


Figura 1. Evolución del IoT.
Fuente: (Peña & Suquillo, 2016).

En el mundo virtual, un objeto físico puede contar con una o más representaciones, no obstante, un elemento virtual no siempre estará asociado a uno físico, sin embargo, intervendrá en las comunicaciones, todo elemento, ya sea este virtual o físico, estará obligado a tener la

capacidad de comunicarse, al mismo tiempo tendrá la opción de detección, almacenamiento o procesamiento de datos; como se representa en la Figura 2 la comunicación de los elementos se puede dar por medio de una red por pasarela, una red sin pasarela o sin usar la red de comunicación (Peña & Suquillo, 2016).

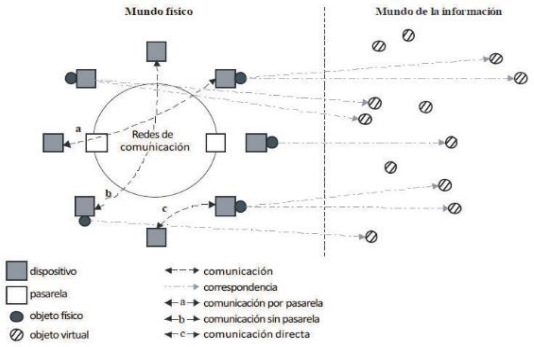


Figura 2. Descripción técnica de IoT
Fuente: (Peña & Suquillo, 2016).

2.1.3. Arquitectura del IoT

Como se especifica en la Figura 3, la arquitectura del IoT tiene tres componentes, los cuales son: Capa de aplicación, Capa de red y la Capa de percepción, mismas que se detallan a continuación.

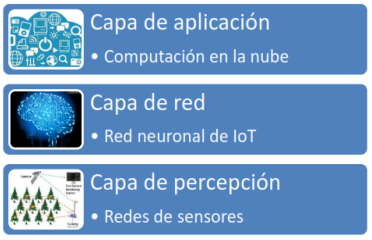


Figura 3. Arquitectura del IoT
Fuente: (García L. , 2014).

2.1.3.1. Capa de Aplicación

En esta capa se utilizan sistemas como la de computación en la nube y demás tecnologías de computación inteligente, con el afán de que estos sistemas puedan captar y procesar la gran cantidad de datos recolectados previamente para la posterior respuesta de los objetos (García L. , 2014).

2.1.3.2. Capa de Red

Esta capa actúa como un sistema neuronal, pero en este caso del IoT, es decir que se encarga principalmente de transmitir y procesar la información, está compuesta por la red de comunicación y la de internet, esta capa recibe los datos obtenidos de la capa de percepción, los procesa y los transmite (García L. , 2014).

2.1.3.3. Capa de Percepción

Se puede asemejar a esta capa con los órganos de los sentidos, ya que es la capa que está a cargo de recolectar información del entorno e identificar a los objetos, esto lo logra mediante sensores, cámaras y demás tipos de artefactos que le permiten analizar el medio en el que se encuentra (García L. , 2014).

2.1.4. Tecnologías del IoT

2.1.4.1. WiFi 802.11x

Como es de conocimiento público la red inalámbrica Wifi opera bajo el estándar IEE 802.11 y sus diversos grupos; la variante IEE 802.11x abarca los grupos 802.11a, 802.11b, 802.11g y 802.11n en la Tabla 1 se especificará la velocidad de cada estándar y la frecuencia en la que operan; los cuales comprenden las capas físicas y la subcapa de control de acceso al medio (Jara & Nazar, 2016).

Tabla 3
Especificaciones de estándares 802.11

Estándar	Velocidad	Frecuencia
802.11a	54 Mbps	5 GHz
802.11b	11 Mbps	2.4 GHz
802.11g	54 Mbps	2.4 GHz
802.11n	600 Mbps	2.4 GHz y 5 GHz

Fuente: (Quitiaquez & De La Torre, 2014).

2.1.5. Infraestructura del IoT

Ante el aumento constante de los teléfonos móviles y sabiendo que las redes de telefonía, a pesar de sus avances, les falta todavía para poder cubrir a todos los clientes nuevos que van apareciendo a diario por lo que baja la calidad de sus servicios, la tecnología IoT ha planteado una nueva red con un único núcleo para servicios múltiples, la red de nueva generación (abreviado de NGN, por sus siglas en inglés *Next Generation Network*), la cual prevé mejorar los servicios y tener el alcance que se necesita para todos los nuevos clientes (García L. , 2014).

2.1.5.1. Protocolos IoT

En la actualidad existen los protocolos de estándar abierto y de protocolo cerrado para desarrollo de IoT, el problema principal que se presenta entre uno y otro es que a futuro el internet de las cosas tendrá un gran impacto que habrá una inmensidad de empresas involucradas en la red de billones de objetos interconectados, de tal manera que será casi imposible que estas impongan un estándar propietario, es decir, una aplicación exclusiva para cada empresa que se encuentre desarrollando IoT. Por tal motivo Mike Simons planteó la solución de que tanto

fabricantes, integradores y usuarios puedan adaptar el código abierto de sus dispositivos como pasa con los teléfonos móviles y *tablets*. De ahí nace la idea de contribuir al proyecto de código abierto de la Fundación Eclipse para precipitar los procesos y ofrecer soporte al desarrollo de la nueva generación de dispositivos móviles inalámbricos para así formar parte de un nuevo estándar abierto de interoperabilidad móvil (Noguera, 2016). Para ello se han estandarizado protocolos como MQTT y XMPP

2.1.5.2. Comunicaciones del IoT

2.1.5.2.1. Serial y USB

Este tipo de comunicación se da en el tipo "en bus" conectando varios periféricos a una computadora, brinda una cierta facilidad ya que se pueden llegar a conectar hasta 127 dispositivos si la necesidad de muchos cables por el tipo de conexión en bus que se realiza, para que un dispositivo pueda transmitir alguna información a la computadora, primero debe ser autorizada por un *host* por lo que este estará pendiente de las transmisiones que quieran enviar los dispositivos que se encuentran conectados (Aguirre, Fernández, & Grossy, 2007).

Basado en esto y refiriéndose a las comunicaciones del IoT, la comunicación entre los sensores de la capa de percepción enviará la información recolectada a través de un cable USB hasta la capa de aplicación, con este sistema se podrá trabajar eficazmente ya que la comunicación será rápida y estarán constantemente intercomunicados los elementos físicos y virtuales del IoT.

2.1.5.2.2. Inalámbrica (WiFi)

El Wifi nace en 1995 cuando la Comisión Federal de Comunicaciones de los Estados Unidos permitió el uso de la banda de 5 GHz sin la necesidad de tener licencias, es una red inalámbrica que funciona mediante el envío de señales por medio del espectro radioeléctrico, al estar expuesta

esa señal a la intemperie se verá afectada por diversos factores como climáticos u obstáculos físicos que le dificulten el traspaso de su señal, opera bajo los estándares IEE 802.11 (Quitiaquez & De La Torre, 2014).

Aplicando este concepto para la comunicación del IoT, los sensores deberán estar conectados a una red Wifi para poder enviar los datos a la capa de aplicación, utilizar esta comunicación tendrá sus beneficios ya que se evita el cableado, los sensores pueden estar lejos o en otro lugar y aun así seguir emitiendo señales, aunque también dependerá de que tan buena sea la conexión Wifi para que la respuesta sea inmediata o dure un tiempo, además si se pierde la señal el IoT quedara inhabilitado.

2.1.6. Arquitectura cliente servidor

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. El modelo Cliente/Servidor permite diversificar el trabajo que realiza cada aplicación, de forma que los Clientes no se sobrecarguen, cosa que ocurriría si ellos mismos desempeñan las funciones que le son proporcionadas de forma directa y transparente. (Marini, 2012)

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. Tanto el Cliente como el Servidor son entidades abstractas que pueden residir en la misma máquina o en máquinas diferentes. (Marini, 2012)

2.2. Ingeniería de modelos

La ingeniería dirigida por modelos surge como la respuesta de la ingeniería de software a la industrialización del desarrollo de software, MDA es la propuesta de la OMG, que centra sus esfuerzos en reconocer, que la interoperabilidad es fundamental y el desarrollo de modelos permite la generación de otros modelos que luego al ser juntados proveerán la solución a todo un sistema e independiza el desarrollo de las tecnologías empleadas. (Stephen, 2004)

2.2.1. Espacios de modelamiento M0, M1, M2, M3

Un metamodelo se define como herramientas que permiten la creación de un modelo; este, a su vez, es una descripción de uno o varios elementos del dominio o el mundo real; finalmente, el meta-metamodelo describe a esos metamodelos planteados y genera un grado de abstracción supremamente alto en el cual coinciden todos los modelos. (Marin, 2017)

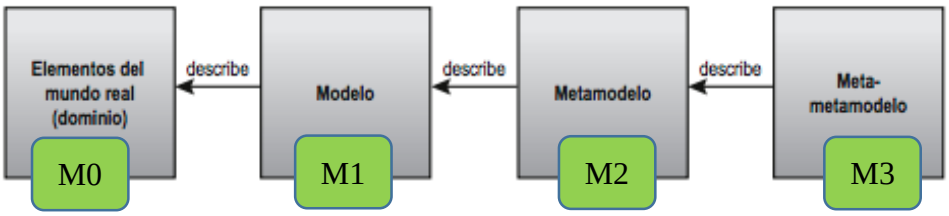


Fig. 1. Modelo, metamodelo y meta-metamodelo

Figura 4. Modelo, Metamodelo y meta-metamodelo
Fuente: (Marin, 2017)

Básicamente hay cuatro espacios de modelamiento,

- Los niveles base M0 que son los elementos del mundo real que tienen impacto en el software. (Garcia Diaz & Cueva Lovelle, 2010).

- Los niveles M1 que son los programas informáticos. (Garcia Diaz & Cueva Lovelle, 2010).
- Los niveles M2 que sería la especificación de UML, ODM, Java, C#, XML u otras y que para este caso será el metamodelo a construir. (Garcia Diaz & Cueva Lovelle, 2010).
- Finalmente están los niveles M3 que son los de mayor abstracción. (Garcia Diaz & Cueva Lovelle, 2010).

La idea de generar estos niveles de abstracción tan altos es proveer un mecanismo común que permita, a través de la transformación de un modelo a otro, la interoperabilidad de los sistemas. . (Marin, 2017)

2.2.2. MDE

MDE (Model-Driven Engineering) o ingeniería dirigida por modelos es una propuesta que se ha venido trabajando desde hace varios años por varios autores, los cuales plantean el uso de modelos como eje fundamental en todo el ciclo de vida de un proyecto de software y que reduce el tiempo y esfuerzo en el desarrollo, una de las aproximaciones de este planteamiento es MDA (Model-Driven Architecture) que es la proposición de la OMG (Object Manage Group) la cuál establece una serie de tecnologías a utilizar en la construcción de software bajo el esquema de MDE. (Montenegro Marin, Gaona Garcia, Cueva Lovelle, & Martinez, 2011)

2.2.3. MDA

La arquitectura dirigida por modelos (MDA, de sus siglas en inglés Model-Driven Architecture) es la propuesta del Object Management Group (OMG) en el contexto de MDE. MDA se basa en otros estándares del OMG, entre los que cabe destacar: Unified Modeling

Language (UML), XML Metadata Interchange (XMI), Meta Object Facility (MOF) o Object Constraint Language (OCL). En MDA, los modelos se clasifican, según su nivel de abstracción, en modelos independientes de la computación, modelos independientes de la plataforma y modelos dependientes de la plataforma. (García, Ruiz, & Vicente Chicote, 2016)

Uno de los conceptos clave en MDA es el de metamodelo. Un metamodelo es un modelo para definir modelos. Cada metamodelo define la sintaxis abstracta de un lenguaje de modelado, esto es, el conjunto de términos del lenguaje (vocabulario), así como las reglas que determinan cómo combinar dichos términos de manera correcta. (García, Ruiz, & Vicente Chicote, 2016)

2.2.4. Lenguajes de dominio específico

"Un lenguaje de dominio específico se enfoca, como bien su nombre lo dice, en operaciones específicas para un dominio, ayudándolo en mejorar diversos aspectos ya que al estar centrado en un solo dominio funcionará óptimamente con el mismo" (Luzza, Berón, Peralta, & Montejano, 2012).

2.2.4.1. Introducción a DSL

Los lenguajes específicos de dominio (abreviado de DSL, por sus siglas en inglés *Domain Specific Language*) no son más que lenguajes de programación que nacen ya que los lenguajes de propósito general (abreviado de GPL, por sus siglas en inglés *Domain Specific Language*), y que si bien son versátiles y útiles para resolver diversos problemas en diversos programas, al estar generalizados comúnmente presentan errores, los cuales para poder solucionar se deberán insertar comandos muchas veces innecesarios para el *software* que se esté manejando, por otro lado utilizando DSL se podrá facilitar el funcionamiento del *software* ya que siendo creados para un dominio específico hará que se optimicen las herramientas utilizadas (Luzza et al., 2012).

Por otra parte, la creación de DSL, ya sean textuales o gráficos, se remonta a los primeros años de la programación, como una forma de incrementar la productividad y simplificar la tarea de programación, habiéndose definido cientos de DSL, algunos muy extendidos como SQL, Excel, make, ant, HTML, XML, etc. Realmente, las especificaciones que uno escribe con un DSL son modelos. (Garcia & Y otros, 2012)

2.3. Transformaciones

El concepto de transformación en el mundo de la informática no es nuevo. De hecho, las transformaciones son algo fundamental en la ingeniería de software, pudiendo verse la computación como un proceso de transformación de datos. (Wimmer & Burgueño, 2013)

Existen dos tipos de transformaciones Model to text (M2T) y Model to Model (M2M) que se explica a continuación:

Las transformaciones M2T se utilizan normalmente para cerrar la brecha entre los lenguajes de modelado y los lenguajes de programación al definir las generaciones de código, pero se pueden emplear de manera genérica para producir texto a partir de modelos como la documentación o las representaciones textuales. Del contenido de un modelo. (Wimmer & Burgueño, 2013)

Estas transformaciones (M2T) se han utilizado para automatizar varias tareas de ingeniería de software, como la generación de documentación, lista de tareas, etc. Por lo tanto las transformaciones M2T en el área de la ingeniería de software basada en modelos están principalmente relacionadas con la generación de código para lograr la transición del nivel del modelo al nivel del código. (Marco, Cabot , & Wimmer, 2012)

Una transformación de modelo a modelo (M2M) es la creación automática de modelos de destino a partir de modelos de origen. (Marco, Cabot , & Wimmer, 2012)

2.4. Herramientas de software

2.4.1. *Eclipse Modeling Framework (EMF)*

Como citó (Saavedra, 2014), Eclipse se lo define como: “un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.”

Eclipse es una plataforma de desarrollo, programación y recopilación de elementos que van desde sitios web hasta aplicaciones de java, esta plataforma es de código libre por lo que brinda gran facilidad para trabajar (Saavedra, 2014).

Ecore consta de 4 clases elementales se puede ver en el meta-modelo de la figura 5.

• EClass: Representa metaclases (clases de un metamodelo). (Garcia & Y otros, 2012)

• EAttribute: Representa metaatributos (atributos de una clase del metamodelo). (Garcia & Y otros, 2012)

• EReference: Representa asociaciones entre clases, incluyendo agregación y referencia. El atributo booleano containment permite distinguir si se trata de una agregación o una referencia. (Garcia & Y otros, 2012)

• EDataType: Representa los tipos de los atributos que son tipos primitivos y tipos de datos objetos definidos en Java. (Garcia & Y otros, 2012)

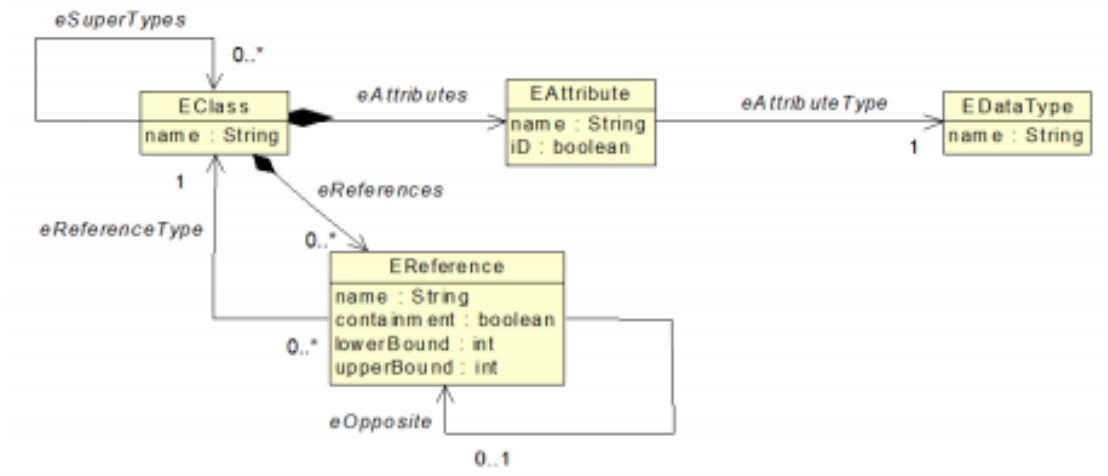


Figura 5. Metamodelo con los elementos básicos de Ecore
Fuente: (Garcia & Y otros, 2012)

Como se muestra en la figura 6 en la paleta Eclipse Modeling Framework se tiene: clases, datatype, atributos, referencias, composiciones, etc. Estos sirven para crear un diagrama UML.

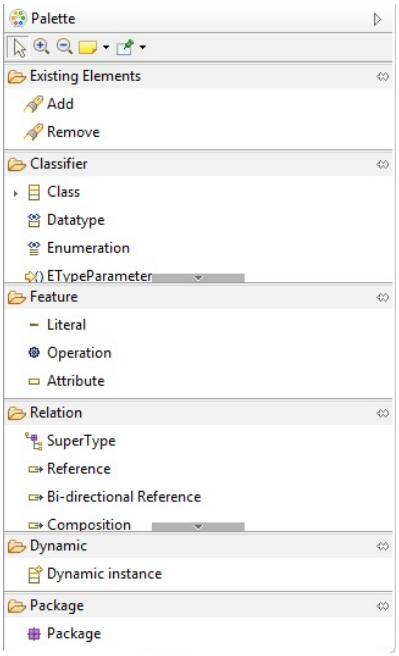


Figura 6. Paleta de eclipse

2.4.2. Sirius

Sirius es un proyecto de Eclipse edificado sobre modelos gráficos mediante tecnologías EMF (*Eclipse Modelling Framework*) y GMF (*Graphical Modeling Framework*) que se creó para desarrollar fácilmente DSLs, definiendo un entorno de trabajo propio para resolver de esta manera problemas específicos, haciendo uso de las tecnologías proporcionadas por EMF y los conocimientos que sustenta la implementación de los lenguajes gráficos de dominio específico. Cabe recalcar que se permite crear una representación de un modelo a partir de la creación de una representación en Sirius a modo de diagrama, tabla o árbol, estas representaciones son denominadas *viewpoints* (Garcia , 2015).

Como se ve en la figura 7 en Sirius Specification Editor se tiene:

- Contenedores: Como su nombre lo explica, sirven para alojar los diferente elementos que se encuentran vinculados a este elemento contenedor.
- Nodos: Sirven para representar las clases del DSL, puede ser de manera gráfica.
- Capas: Sirven para seccionar los diferentes elementos que se encuentran dentro de las diferentes capas que se puedan crear.
- Secciones: Sirven para representar los nodos creados en la paleta del diseño gráfico.
- Relaciones: Se crean para poder unir los diferentes elementos, para esto se debe especificar el orden de inicio y fin.

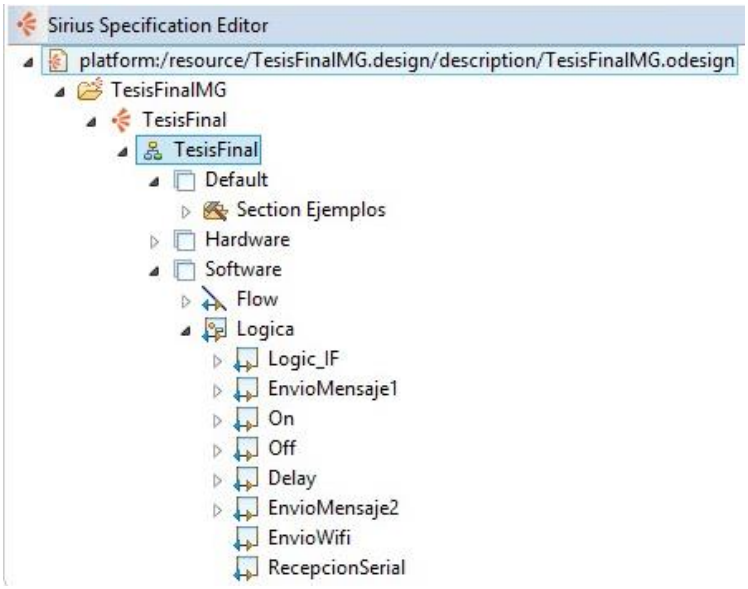


Figura 7. Sirius Specification Editor

2.4.2.1. Creación de editores gráficos

Para este apartado se presenta un ejemplo básico de la utilización y creación de proyectos en Eclipse utilizando Sirius para la generación de plantillas de código fuente mediante Acceleo. Este ejemplo se lo detalla en el Anexo 2.

2.4.3. ACCELEO

Acceleo es una tecnología basada en plantillas que incluye herramientas de creación para crear generadores de código personalizados. Le permite producir automáticamente cualquier tipo de código fuente a partir de cualquier fuente de datos disponible en formato EMF. (Copyright Eclipse Foundation, Inc., 2019)

Como menciona (Copyright Eclipse Foundation, Inc., 2019) ofrece ventajas sobresalientes:

- Alta capacidad para personalizar
- Interoperabilidad

- Inicio fácil

Los códigos más utilizados para este propósito será:

✓ Sentencia template

[template public generateElement(aClass : Class)]

En esta parte se deberá colocar el código perteneciente todo el proyecto que se generará.

Dentro de este segmento, se debe colocar la clase principal del DSL.

[/template]

✓ Sentencia file

[file(aClass.name+'.extensión' false,'UTF-8')]

En esta parte se deberá colocar el código perteneciente todo el proyecto que se generará con el nombre que tenga.

Dentro de este segmento, se debe colocar el nombre del proyecto con la extensión que se requiera en el caso puede ser .txt o .ino, seguido de un false el cual permite abrir el archivo de salida.

[/file]

✓ Sentencia For

[for (nombre_objeto : Clase_pertenece_objeto | nombre_objeto_principal.relacion_clasePrincipal_ClaseObjeto)]

En esta parte se deberá colocar el código perteneciente que se generará por cada elemento de este tipo, que sea arrastrado a la paleta.

Dentro de este segmento, se puede acceder a los atributos de la clase a la cual pertenece el objeto, bajo la siguiente sentencia: [nombre_objeto.Atributo]

[/for]

- ✓ Secuencia de los elementos

Para definir el orden en el que se desea que se ejecuten las sentencias, se debe definir la forma en la que están conectados los elementos, con esta finalidad se debe seguir los siguientes pasos a manera global:

1. Determinar primer elemento
2. Determinar a qué elemento, se encuentra conectado el primer elemento
3. Determinar a qué elemento, se encuentra conectado el elemento que se encuentra conectado al primer elemento
4. Determinar a qué elemento, se encuentra conectado el elemento que se encuentra conectado al elemento que se encuentra conectado al primer elemento
5. Etc.

Con el propósito de cumplir lo descrito en el punto 1, se utiliza la función *eAllContents()->at(1)*, de manera general se la utilizará de la siguiente manera:

```
[for (nombre_objeto :
Clase_pertenece_objeto                                                    |
nombre_objeto_principal.relacion_clasePrincipal_ClaseObjeto)]

 [nombre_objeto.Atributo].eAllContents()->at(1)

[/for]
```

Con el propósito de cumplir lo descrito en el punto 2 al 5, utiliza la función *eCrossReferences()->at(1)*, la cual, entrega el elemento (con sus atributos) al cual se conecta el elemento, de manera general se la utilizará de la siguiente manera:

[for (nombre_objeto : Clase_pertenece_objeto | nombre_objeto_principal.relacion_clasePrincipal_ClaseObjeto)]

[nombre_objeto.Atributo].eAllContents()->at(1). .eCrossReferences()->at(1)

[/for]

2.5. Tarjeta de desarrollo

2.5.1. Introducción

Un arduino es un *Hardware* libre, que se fundamenta en una placa con microcontrolador, para facilitar el control de proyectos electrónicos interactivos, al ser un *Hardware* libre no se necesitan licencias para poder trabajar con el arduino, esta placa cuenta con puertos de entrada y salida, un microcontrolador (como se observa en la figura 6) y el lenguaje de programación que utiliza es el *Processing/wrining* (Morales & Gonzáles, 2013).

Figura 8. Placa Arduino UNO
Fuente: (Rodriguez, 2018)

2.5.2. Características técnicas

Como mencionaron (Piña & Zúñiga , 2017) las características técnicas de un arduino son:

- Tiene 5 pines para entradas analógicas, y contiene 6 pines para salidas analógicas de salidas PWM

- Es Completamente autónomo: ya programado no necesita estar conectado al PC
- Dispone de 1 kbyte de memoria RAM y frecuencia de 16 MHz
- Posee 13 pines para entradas/salidas digitales las cuales con programables
- Posee Memoria Flash 32 KB (2 KB para el bootloader)
- Memoria estática - SRAM 1 KB
- Memoria ROM - EEPROM 512 byte
- Con una Velocidad de reloj 16 MHz
- Voltaje de entrada (recomendado) 7-12 V
- Voltaje de entrada (limite) 6-20 V
- Digital I/O Pines 14 (con 6 salidas PWM)
- Corriente continua (DC en inglés) corriente I/O Pin 40 mA
- Corriente continua (DC en inglés) corriente 3.3V Pin 50 mAs analógicas Pines 6.
- Posee un microprocesador ATmega328
- Tiene 32 kbytes de memoria Flash

2.5.3. Entorno de programación

Como lo mencionaron Morales y Gonzales (2013) el lenguaje de programación propio del arduino es el *Processing*, pero a su vez soporta otros lenguajes, ya que se comunican con la transmisión de datos en formato serie.

2.5.4. Principales aplicaciones

El Arduino al ser un *Hardware* libre presenta una gran versatilidad para crear proyectos, los cuales pueden ir desde relojes, robots, incluso con ingenio, manejando correctamente los lenguajes de programación y contando con los materiales necesarios, puedes crear tu propio

ambiente inteligente, por ejemplo, puedes en tu casa puedes activar cosas como las persianas, o la luz utilizando tu voz; es por esto que en las industrias es muy utilizado para los procesos de automatización que requieren (Yúbal, 2018).

Se puede organizar a los proyectos con Arduino en dos grandes grupos:

- En los que es utilizado como microcontrolador, en estos proyectos el Arduino controla los dispositivos que se le indiquen en base al programa que corre en el ordenador, lo hace mediante el uso de sensores y actuadores (Guía del Arduinomaniaco, 2015).
- El otro grupo es en el que se emplea la placa de desarrollo como interfaz entre un computador u otro dispositivo, en este caso se puede hacer que el programa realice determinada acción, según el Arduino identifique la situación mediante los sensores (Guía del Arduinomaniaco, 2015)

2.5.5. Módulos para ampliación

2.5.5.1. WiFi ESP 8266

El módulo para Arduino ESP 8266, es un chip con conexión Wifi (como se observa en figura 7), compatible con protocolos TCP/IP, este módulo es de gran ayuda para los que quieren trabajar con el IoT, ya que su propósito es brindar acceso a la red a cualquier micro-controlador que lo requiera, además es un módulo accesible económicamente, diferenciándolo de los otros módulos que presentan conexión Wifi (Hernández L. , 2018).

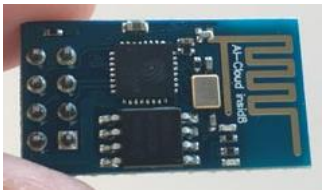


Figura 9. Módulo ESP8266
Fuente: (Hernández L. , 2018).

Tiene 17 puertos GPIO (de propósito general) de los cuales solo se pueden Utilizar 8 o 9, en la figura 8 se tendrán especificados donde se encuentran dichos puertos dentro módulo, esto se los puede configurar con resistencia Pull-up o Pull-down (Hernández L. , 2018).

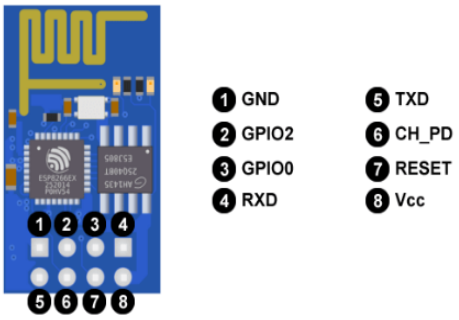


Figura 10. Puertos GPIO.
Fuente: (Hernández L. , 2018).

2.5.5.2. Sensores

Un sensor es un dispositivo que tiene la habilidad de tomar variables de instrumentación (temperatura, distancia, velocidad, presión, humedad, etc.) y transformarlas en variables eléctricas para enviar su información a otro dispositivo par que lo pueda interpretar, los sensores aprovechan las propiedades de elementos para ayudarse con su recolección de información, algo parecido a lo que realiza un termómetro, ya que este se basa en la propiedad del mercurio de expandirse o contraerse con la temperatura (Morales & Gonzáles, 2013).

Para poder lograr el cometido del IoT, un factor muy importante resultan ser los sensores, ya que a través de ellos es que se logra recolectar la información y saber qué es lo que está pasando en el entorno, para que el sistema pueda actuar acorde a estos datos, últimamente se han dado varios avances nanotecnológicos por lo que los sensores cada vez son más imperceptibles, pero a la vez de mejor calidad para manejar los datos en tiempo real (García L. , 2014).

2.5.5.2.1. Tipos de sensores

- **Sensores de temperatura:** Los sensores de temperatura se catalogan en dos series diferentes: TD y HEL/HRTS. Estos sensores consisten en una fina película de resistencia variable con la temperatura (RTD) y están calibrados por láser para una mayor precisión e intercambiabilidad. Las salidas lineales son estables y rápidas. (Molina, 2019)
- **Sensores de humedad:** Los sensores de humedad relativa/temperatura y humedad relativa están configurados con circuitos integrados que proporcionan una señal acondicionada. Estos sensores contienen un elemento sensible capacitivo en base de polímeros que interacciona con electrodos de platino. (Molina, 2019)
- **Sensores de presión:** Los sensores de presión están basados en tecnología piezoresistiva, combinada con microcontroladores que proporcionan una alta precisión, independiente de la temperatura, y capacidad de comunicación digital directa con PC. Las aplicaciones afines a estos productos incluyen instrumentos para aviación, laboratorios, controles de quemadores y calderas, comprobación de motores, tratamiento de aguas residuales y sistemas de frenado. (Molina, 2019)
- **Detectores de ultrasonidos:** Los detectores de ultrasonidos resuelven los problemas de detección de objetos de prácticamente cualquier material. Trabajan en ambientes secos y polvorientos. Normalmente se usan para control de presencia/ausencia, distancia o rastreo. (Molina, 2019)

2.5.5.3. Actuadores

Los actuadores son los objetos que, mediante la implantación de tecnologías inteligentes, llegan a tener inteligencia artificial, obteniendo así la capacidad de emitir una reacción a la

situación que previamente detecto un sensor, estos objetos tienen la habilidad de comunicarse con los usuarios cuando así lo requieran con el fin de terminar la reacción (García L. , 2014).

2.5.5.3.1. Tipos de actuadores

- **Actuador Rele:** Los relés de estado sólido SSR (Solid State Relay) están formados un circuito eléctrónico que contiene en su interior un circuito disparado por nivel, acoplado a un interruptor semiconductor, un transistor o un tiristor. Tienen la ventaja de no producir ruido debido a que el circuito de acoplamiento está basado en componentes no electromecánicos. (Ingenieria de sistemas y automatica , 2016)
- **Actuador led:** El led o diodo emisor de luz es un actuador que se puede encender o apagar desde el microcontrolador. Se uso distintos led estándar de colores de 5mm. (Ingenieria de sistemas y automatica , 2016)

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

El objetivo de desarrollar la arquitectura es representar los conceptos y las relaciones de la meta-modelo para componentes de Arduino que permitan elaborar diferentes programas obteniendo el código de Arduino de forma automática de la aplicación.

Para este propósito se utilizó el software Ecore modeling que es un lenguaje de modelado de eclipse.

3.1. Definición de características y creación del meta-modelo

Primero se definió las características generales que debería tener un programa en una placa de desarrollo como es Arduino, las cuales se ve a continuación:

- Ingreso de datos mediante sensores
- Salida de datos por medio de actuadores
- Escritura por el puerto serial
- Dar secuencia a las condiciones y acciones dentro de Arduino.
- Tener condiciones
- Que los datos en los sensores puedan enviarse a un servicio web (IoT)
- Se pueda obtener los datos del servicio web (IoT) para manipulación en los programas.

Una vez planteadas las características generales para el desarrollo de la meta-modelo, se tiene una idea más clara de las clases necesarias y sus relaciones.

Se procede a la construcción del UML en el software EMF (Eclipse Modeling Framework), luego a crear un proyecto del tipo Ecore Modeling Project, el cual entrega las herramientas para graficar el diagrama de la meta-modelo.

Los elementos que se utilizaron de la paleta fueron: clases, datatype, atributos y composición.

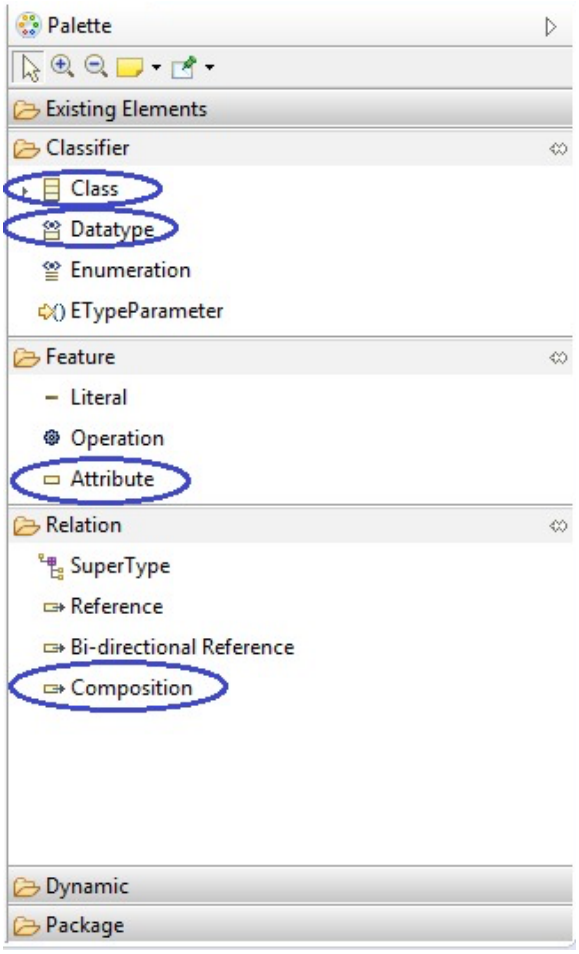


Figura 11. Paleta EMF.

Se diseña un diagrama UML (ver Figura 14) escalable para ampliar la gama de las posibles aplicaciones realizadas en tarjetas Arduino, si en caso de requerir nuevos módulos físicos, sensores, actuadores, etc. será de fácil adaptación al diagrama UML.

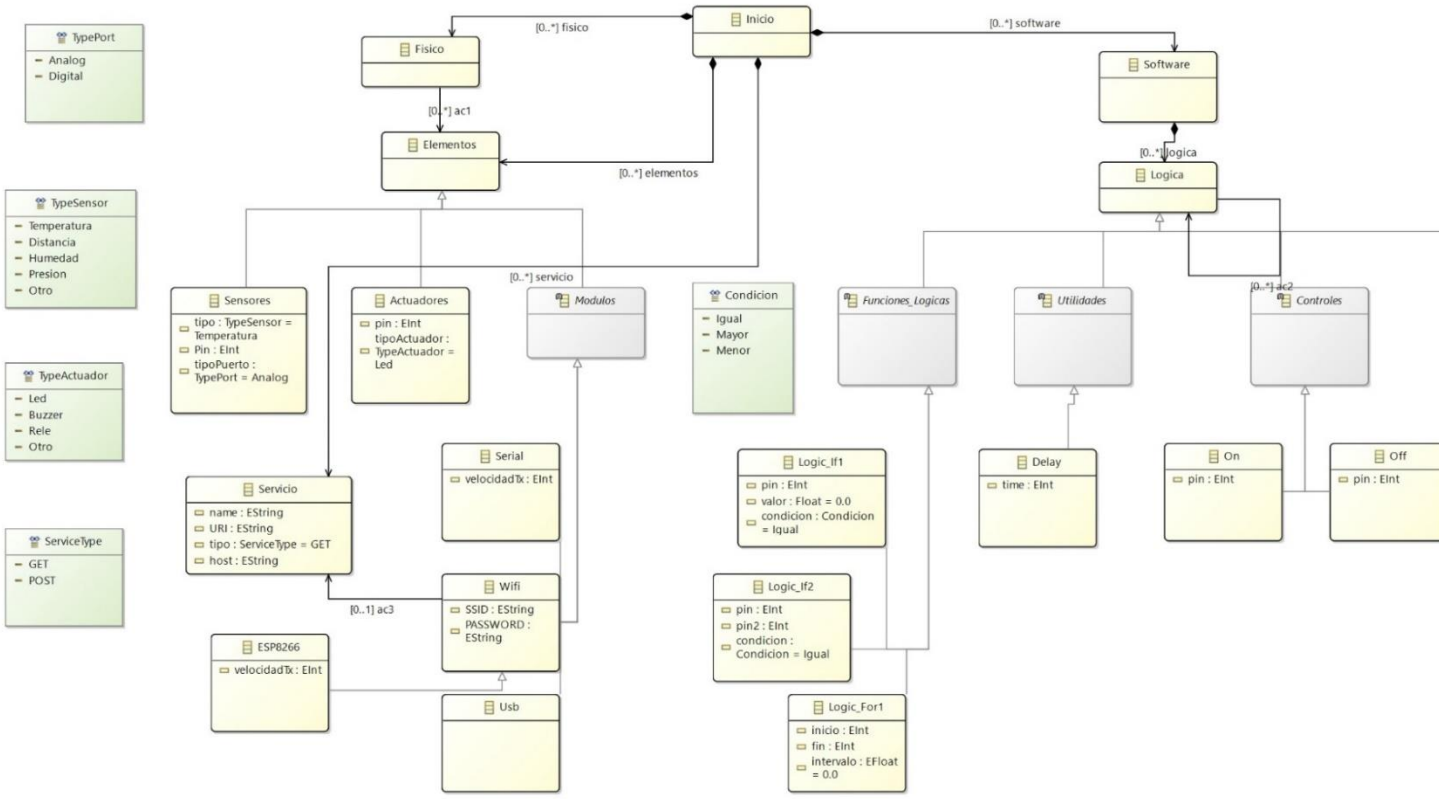


Figura 12. Diagrama UML de la aplicación

En el desarrollo de este UML se analizó que la posible solución era separar la parte física, de la lógica de programación, esta solución permite aprovechar las dos funciones obligatorias que debe llevar todo programa en Arduino, la función *void setup (),* y *void loop().* Siendo utilizado la parte física del diagrama UML para la configuración de la tarjeta Arduino, es decir función *void setup()*, mientras que la sección del UML que describe la funcionalidad lógica, será utilizado para el algoritmo creado en la función *void loop().*

En la Figura 13, se observa la parte física o estructura de hardware, donde se implementó dos clases padre, Físico y Elementos, que servirán para la creación y relación de los componentes que serán graficados en la paleta del software desarrollado.

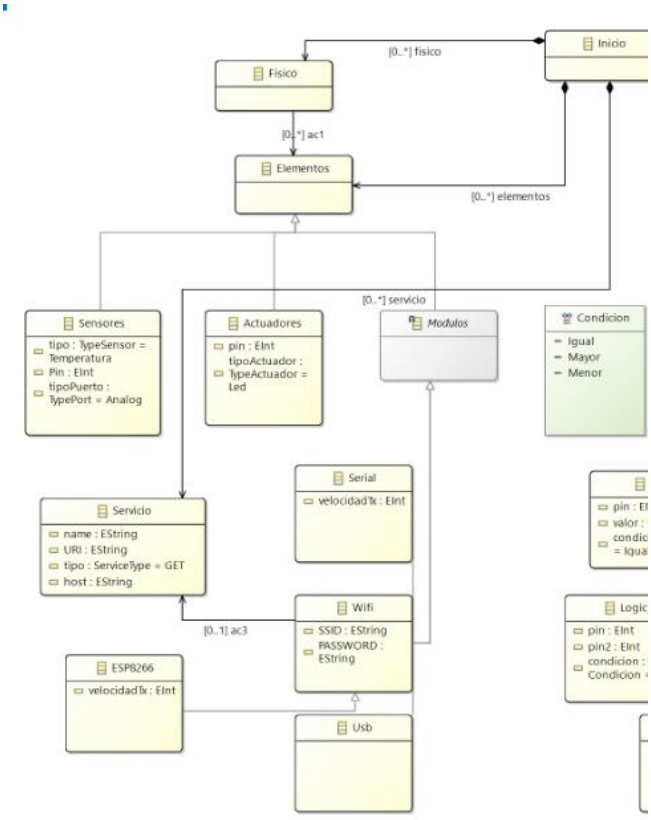


Figura 13. UML parte Hardware

Como se puede visualizar en la Figura 13, tres Clases en el diagrama UML que a partir de la clase elementos, servirán para ejemplificar de manera gráfica los circuitos y componentes electrónicos que serán utilizados por la herramienta, estas clases son, Sensores, Actuadores y Módulos. Bajo la clase Módulos, se implementa las funciones que permitirán configurar las conexiones (serial, usb, wifi) de la tarjeta Arduino hacia la nube u otros dispositivos que serán utilizados para cumplir con el objetivo de interconectar hacia IoT.

Por otra parte, en la Figura 14, se observa la estructura de la parte lógica o software y sus relaciones entre las clases, dicha estructura se basa en cuatro clases padres que serán utilizadas para generar todos y cada uno de las órdenes o sentencias lógicas implementadas en la herramienta. En la clase *"Funciones Lógicas"*, se englobarán las funciones lógicas utilizadas en cualquier algoritmo y lenguaje de programación, como son las sentencias, *if*.

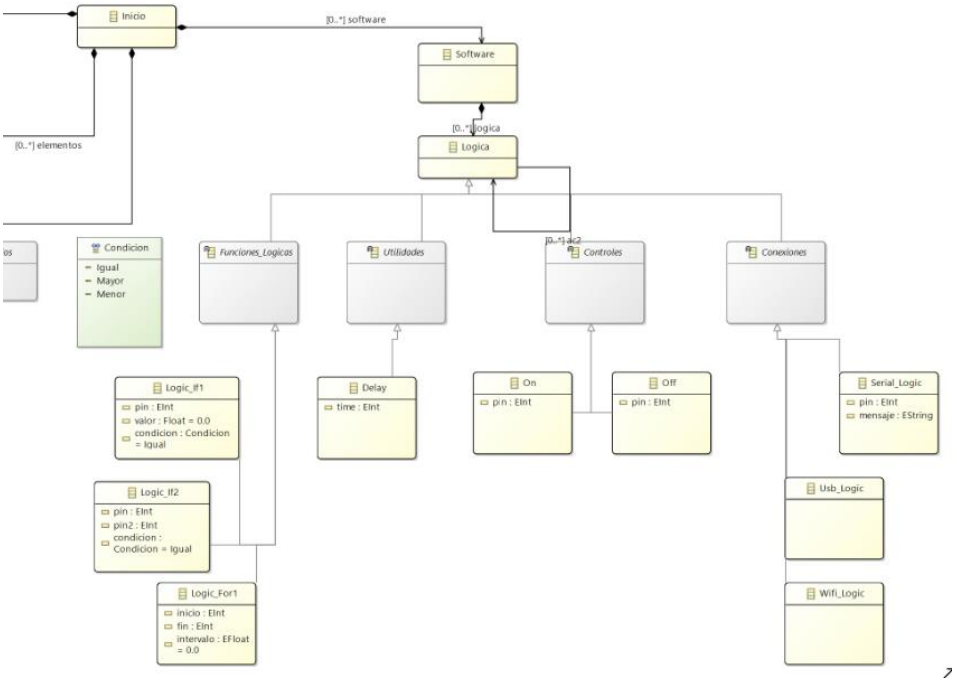


Figura 14 UML parte lógica

La clase padre Utilidades fue desarrollada con el objetivo de englobar a funciones que son utilizadas por los programadores para simplificar el desarrollo de algoritmos, para el alcance de este proyecto fue necesario la implementación de la clase Delay, que realizará la función retardos en el manejo de pines del Arduino. Adicionalmente se tiene la clase Controles, bajo la cual se desprenden las acciones que realizará el Arduino, como controles digitales on, off.

También se cuenta con la clase Conexiones, misma que fue desarrollada con la finalidad de lograr implementar las funciones lógicas necesarias para alcanzar la conectividad de la tarjeta Arduino con diversas tecnologías como wifi, comunicación serial y comunicación usb. Finalmente, como se muestra en la Figura 15, cabe recalcar que fueron implementadas clases con atributos fijos que representan variables que se usaran a lo largo de la implementación

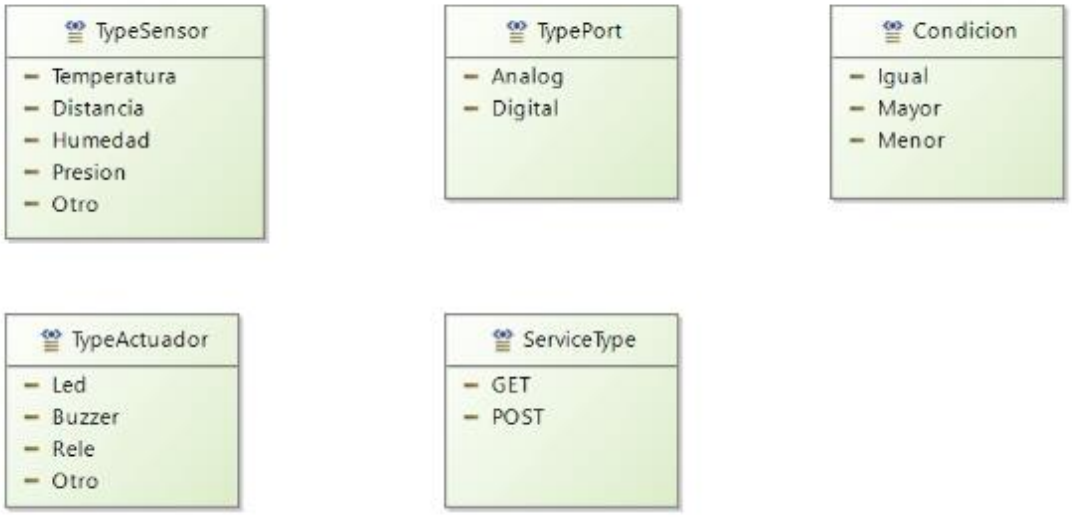


Figura 15. Clases enumeración

3.2. Editor Gráfico

Cuando se creó el diagrama UML en el proyecto “tesis final MG” se crearon una serie de archivos que se deben tomar en cuenta, especialmente el archivo “tesisFinalMG.genmodel” con el cual se ejecutaran un nuevo entorno de eclipse.

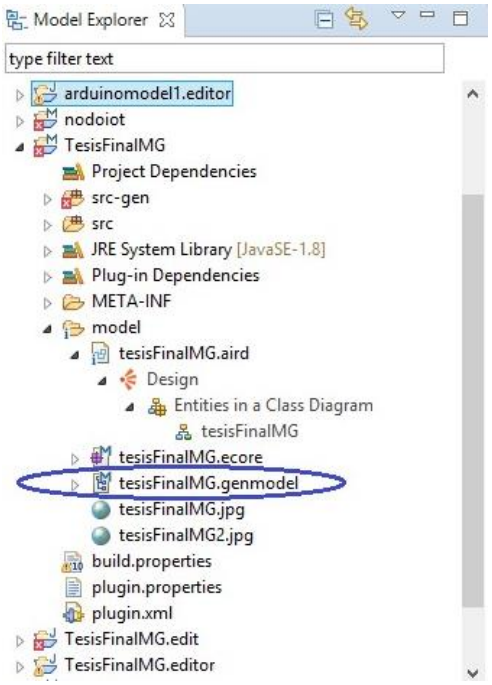


Figura 16. Editor grafico

En el nuevo entorno de eclipse se crea un modeling Project en el cual se escoge la meta-modelo que se creó anteriormente. Y aquí se valida si todas las clases del meta-modelo se crearon correctamente.

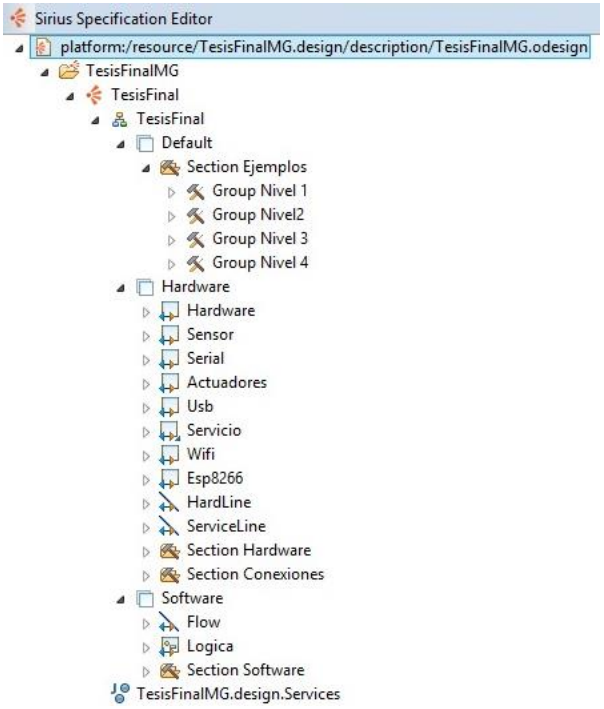


Figura 17. Diagrama UML en forma de árbol

Un *Node Creation* es considerado un objeto, o elemento gráfico que representará a una clase de UML. Una vez estructurado el UML, se continua con el desarrollo de relaciones y elementos gráficos en el Run-Time de Eclipse, para la presente implementación, como se visualiza en la Figura 19, se mantiene la estructura planificada desde el UML, se separó la parte física, creando la capa Hardware, como la parte de software, creando la capa Software.

Los *Node Creation* son creados dentro de cada capa, y mediante sus propiedades podrán ser asociados a la clase a la que se desea vincular. En la Figura 18 se visualiza un ejemplo de cómo asociar el elemento *Node Creation, en este caso sensor,* con una clase de UML.

Para cada node creation es necesario color los tres campos que se ven en la figura 20:

- ID: Nombre que se le da al Node Creation
- DOMAIN CLASS: Aquí es donde se relaciona con la clase del UML

- ❖ SEMANTIC CANDIDATES EXPRESSION: Aquí se coloca la relación que existe desde la clase principal hasta la clase seleccionada.

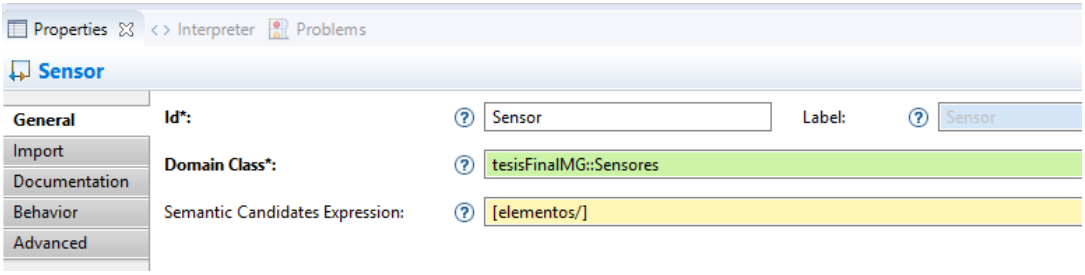


Figura 18. Ejemplo asociación Node Creation

De esta manera se realiza la configuración de cada nodo de creación, dentro de la capa Hardware se cuenta con nodos de creación descritos a continuación en la siguiente tabla:

Tabla 4

Nodos de creación capa Hardware

Nodo de creación	Clase Asociada UML	Utilidad
Hardware	Fisico	Representación gráfica de la tarjeta Arduino
Sensor	Sensores	Representación gráfica de sensores posibles
Serial	Serial	Representación gráfica del módulo de transmisión serial
Actuadores	Actuadores	Representación gráfica de los posibles Actuadores
Usb	Usb	Representación gráfica del módulo de transmisión Usb
Servicio	Servicio	Representación gráfica del servicio necesario para la conectividad con IoT
Wifi	Wifi	Representación gráfica de la conectividad wifi
Esp8266	ESP8266	Representación gráfica de la tarjeta ESP8266

Dentro de la capa Software fueron implementados los siguientes Nodos de creación descritos a continuación en la Tabla 5.

Tabla 5
Nodos de creación capa Software

Nodo de creación	Clase Asociada UML	Utilidad
Lógica	Software	Nodo de creación tipo contenedor que permite lograr la separación entre la parte física (Hardware) y la parte lógica (Software)
Logic_IF1	Logic_if1	Representación gráfica de la función lógica *if*
EnvioSerial	Serial_Logic	Representación gráfica de la trasmisión Serial
EnvioUsb	Usb_Logic	Representación gráfica de la trasmisión Usb
EnvioWifi	Wifi_Logic	Representación gráfica de la trasmisión mediante Wifi
RecepcionSerial	Serial_Logic	Representación gráfica de la recepción Serial
RecepcionUsb	Usb_Logic	Representación gráfica de la recepción Usb
RecepcionWifi	Wifi_Logic	Representación gráfica de la recepción Mediante Wifi
EnvioMensaje (datoAnalogico)	Enviar_Wifi	Representación grafica del envió de datos al internet
Recepción Wifi (Servicio)	Recibir_Wifi	Representación gráfica del recibo de datos al internet
On	On	Representación gráfica de acción On (enviar un “1” lógico a través de Arduino)
Off	Off	Representación gráfica de acción Off (enviar un “0” lógico a través de Arduino)
Delay	Delay	Representación gráfica de la función retardo o delay, necesaria para visualizar las acciones realizadas en la tarjeta

3.2.1. Relaciones

Para poder concatenar secuencialmente los elementos del diagrama (Nodos), es necesario declarar y construir las líneas o segmentos que van a unir a los elementos. Estas relaciones también deben estar especificadas en el UML para que puedan ser utilizadas en la herramienta.

De igual manera como se genera el node creation se crea las relaciones.

Para ejemplificar de mejor manera este tema se observa en la Figura 19, que ambas clases "*Físico"* y "*Elementos"*, se encuentran unidos por la relación ***ac1***, esta relación es la que permitirá graficar la línea que unirá a los elementos gráficos que representan la clase "*Físico"*, con los elementos gráficos que representan la clase "*Elementos"*.

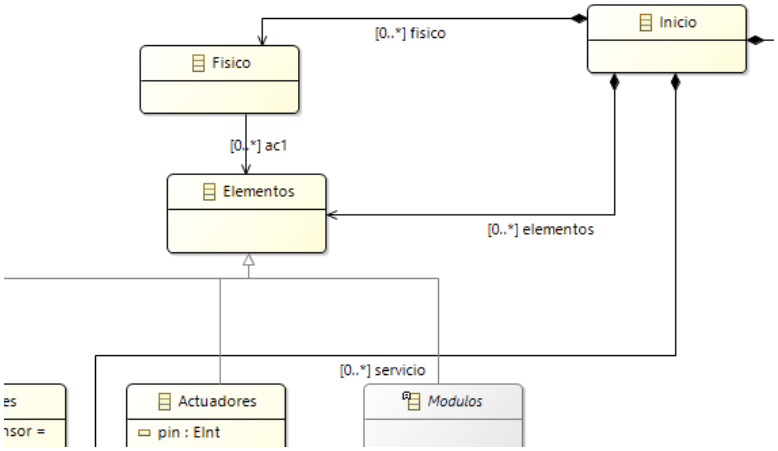


Figura 19. Ejemplo de relaciones en UML

Como se observa en la Figura 20, al declarar los bordes o líneas que permiten unir el nodo Hardware, con los elementos gráficos como sensores, actuadores, conectores, etc., es la previa declaración de la relación *ac1* en el diagrama principal UML.

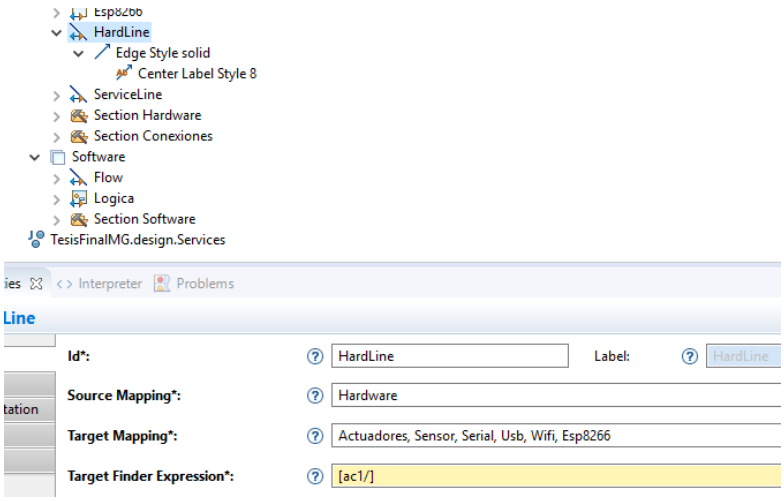


Figura 20. Relación ac1

3.2.2. Diseño de la paleta

Para mostrar la paleta lo que se realiza es crear una nueva herramienta que se llama *section*

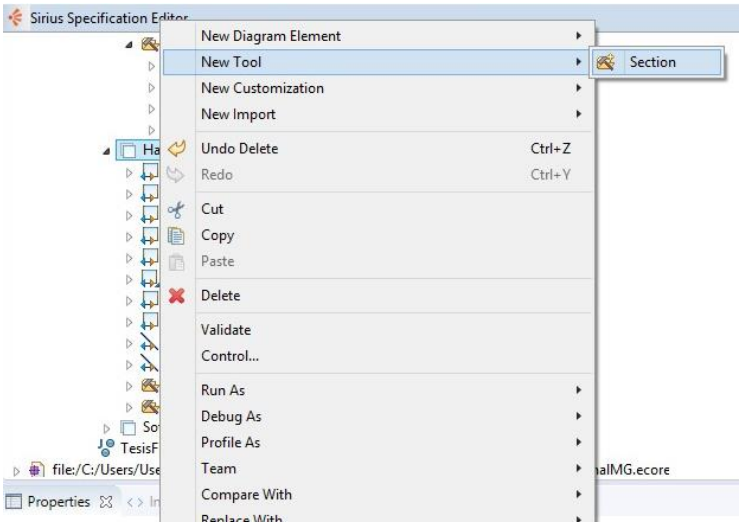


Figura 21. Creación de la paleta

Como se revisó en el apartado anterior, cada *Node creation* debe ser asociado a una clase de UML. De igual forma, cada elemento gráfico que conformará la paleta, debe estar asociada con

un *Node creation*, es así como en la Figura 22 se muestra un ejemplo de la asociación requerida para un elemento.

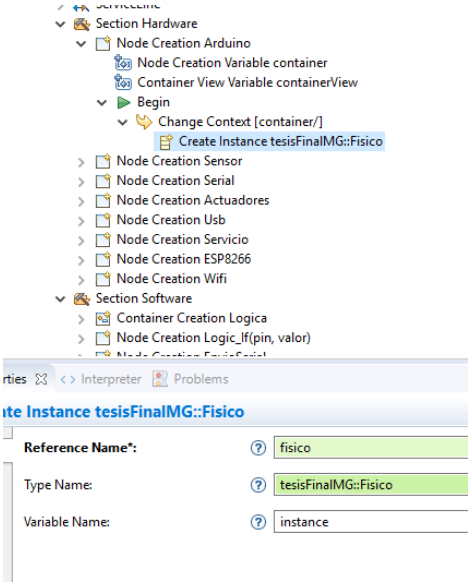


Figura 22. Ejemplo de asociación con Node Creation

Una vez realizada esta asociación descrita, se agrupan los elementos bajo Secciones, que serán útiles para que el usuario final encuentre los elementos reunidos según el tipo de elemento.

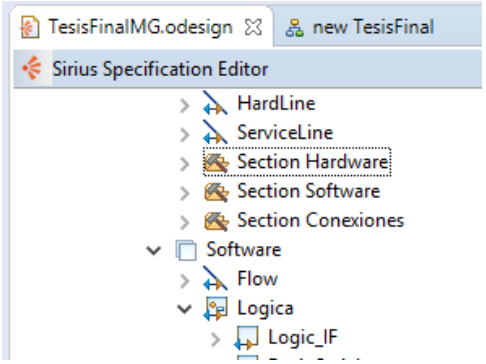


Figura 23. Secciones Paleta

Como se puede observar en la Figura *23*, se crearon tres secciones que agruparan a los elementos disponibles para la ejecución de la herramienta desarrollada. Estas tres secciones son:

- **Hardware**, donde se agrupan los elementos físicos

- **Software**, donde se agrupan los elementos y acciones lógicas
- **Conexiones**, donde se agrupan las posibles relaciones entre los elementos

Finalmente, después de haber cumplido con lo descrito en los apartados anteriores, se obtiene la herramienta gráfica que se observa en la Figura 24 que es el área donde se desarrollarán las aplicaciones que el usuario desee implementar.

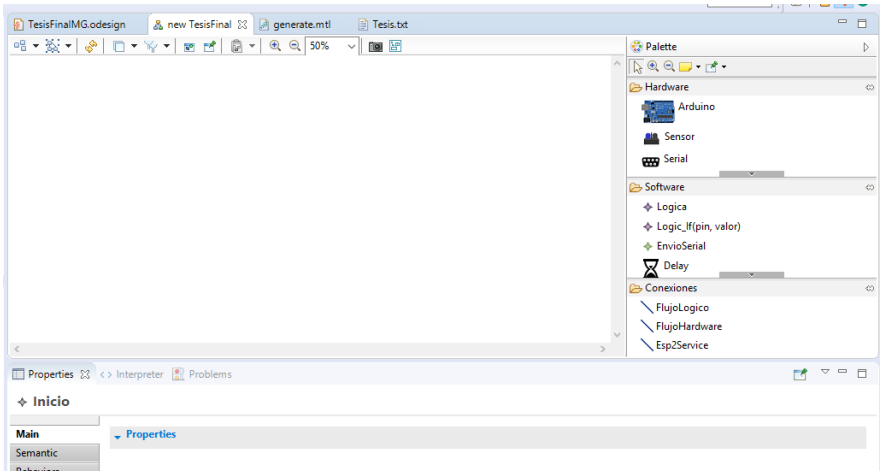


Figura 24. Paleta gráfica

3.3. Transformación M2T

La transformación automática del código fuente mediante Acceleo, se trabajó en congruencia con la metodología planteada desde un inicio en los apartados anteriores, es decir se separó la parte física de la lógica.

De esta manera, y en concatenación en los apartados anteriores, la plantilla desarrollada para la generación automática del código de cualquier aplicación que desee desarrollar el usuario, fue construida utilizando la lógica que se muestra en la figura, donde se observa que tanto para la inicialización de los pines que voy a utilizar para la aplicación que desee construir, como para la carga de librerías se utilizarán los elementos gráficos que fueron desarrollados en la capa de

Hardware, mientras que para rellenar la función de Arduino *loop()*, se utilizan las relaciones entre los elementos de la capa *Software*.

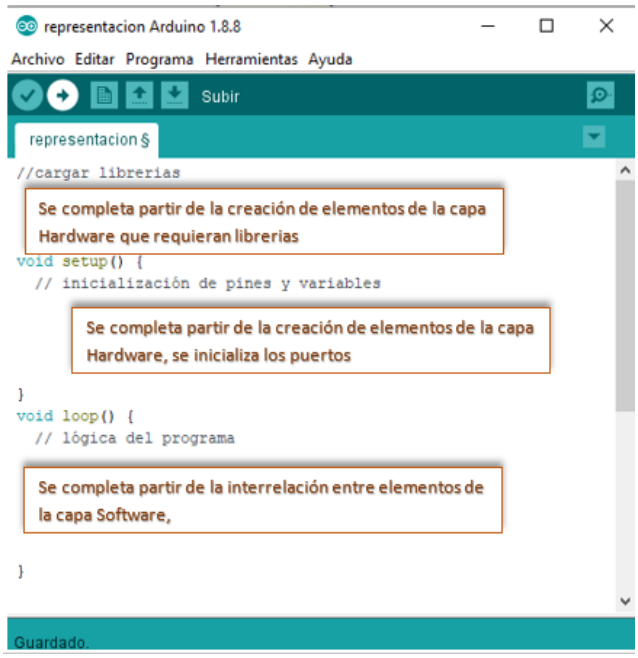


Figura 25. Estructura para generación de la plantilla

El algoritmo completo para la generación automática del código fuente se encuentra en el Anexo1.

3.4. Correspondencia entre UML, Editor gráfico, paleta de diseño y M2T (Acceleo)

Todo lo realizado en los apartados anteriores, tienen relación entre sí, cada elemento es necesario y se relaciona con el otro para poder cumplir las funciones que se especifica en el UML, es decir que toda clase del UML, es utilizada para el editor gráfico, tiene su representación en la paleta de diseño, y generará un determinado segmento de código.

Como se puede observar en la Figura

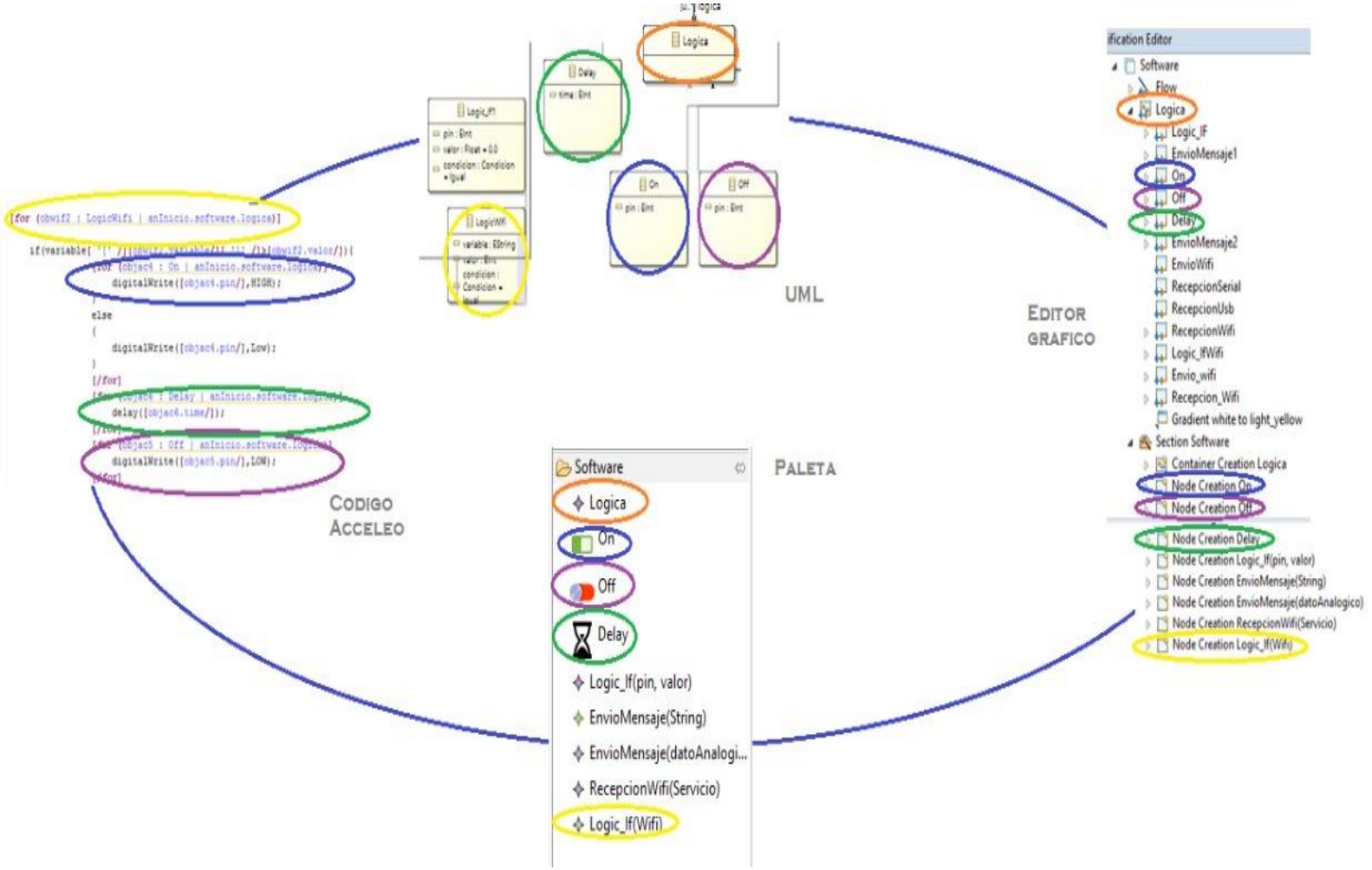


Figura 26 Relación de software dentro de la herramienta

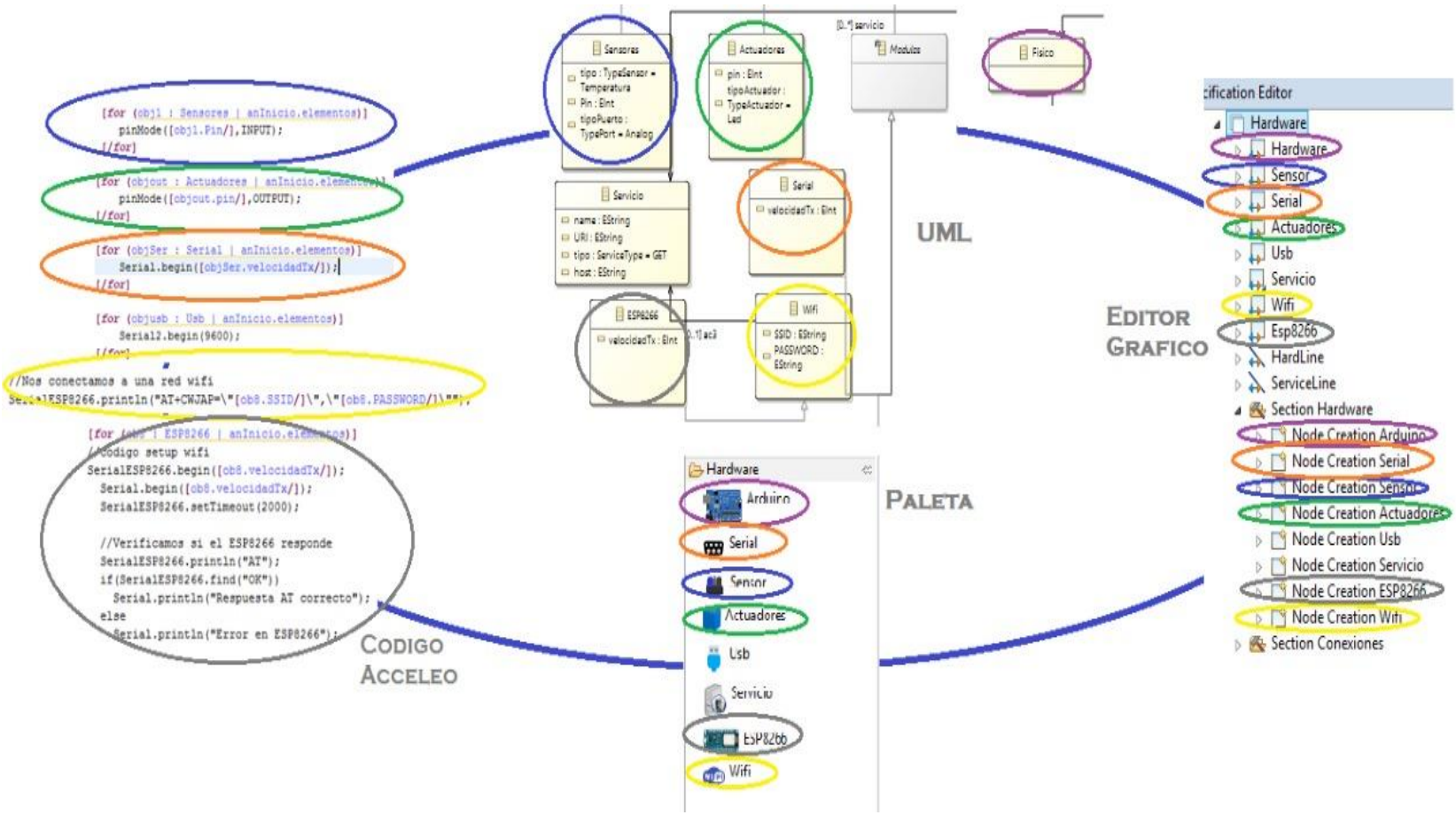


Figura 27 Relación de hardware dentro de la herramienta

3.5. CODIGO DE ACCELEO

En la figura 26 se muestra el código generado en acceleo

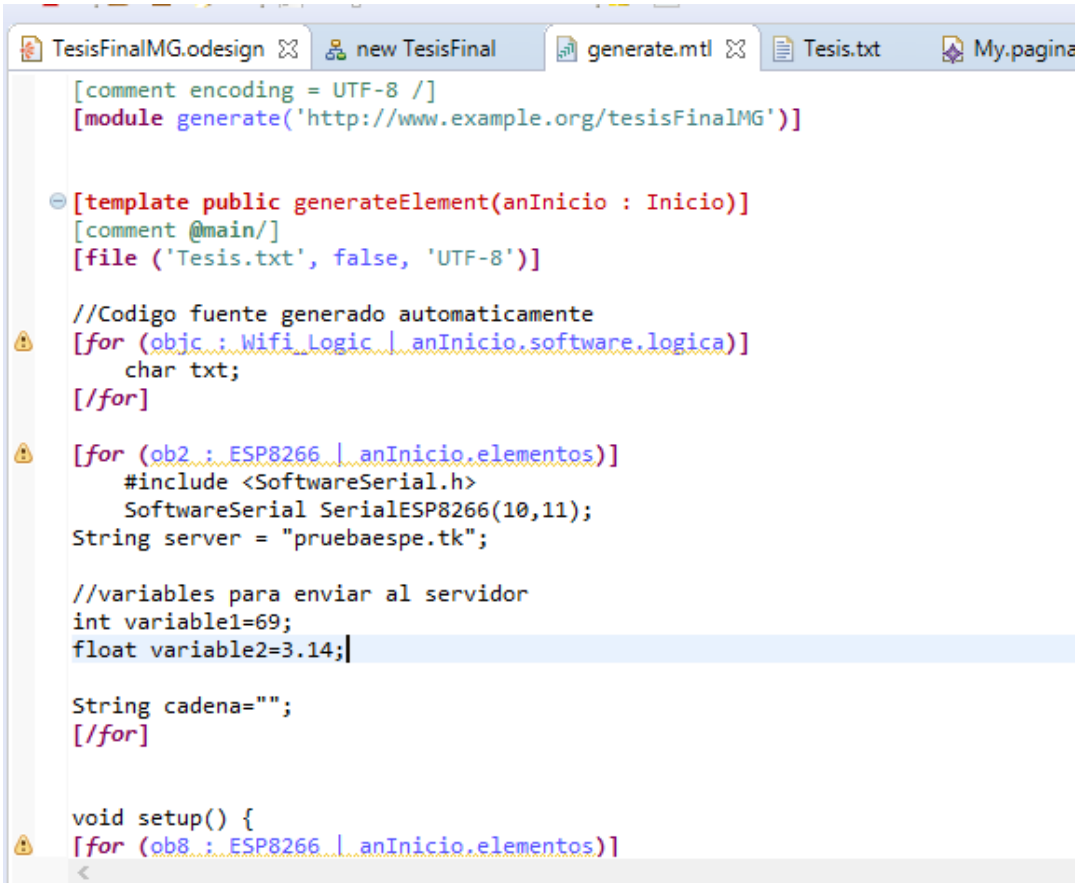


Figura 28 Vista general de acceleo

Para empezar el código en Acceleo se genera de forma automática el siguiente fragmento

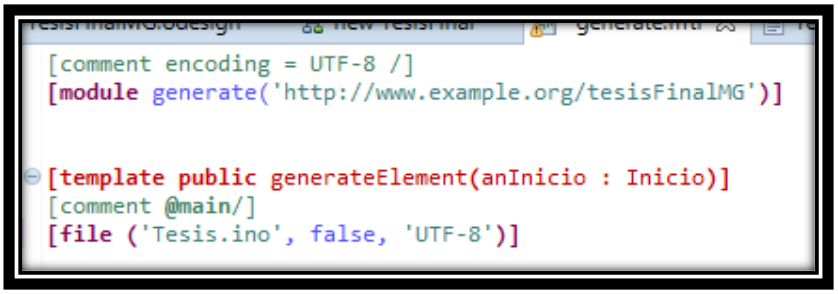


Figura 29 Vista general de acceleo

Donde, dentro del *template public generateElement()*, se declara la clase principal de UML, *Inicio*, es la clase, y se crea un objeto de dicha clase, *anInicio*

A continuación, dentro de la función *file()*, se escribe el nombre del archivo que se generara con su debida extensión, '*Tesis.ino'* (la extensión .ino genera el IDE de Arduino

De manera seguida es necesario declarar todos y cada uno de los elementos que tiene representación gráfica en la paleta, para esto, se precisa crear un nuevo objeto de la clase que representa el elemento gráfico.

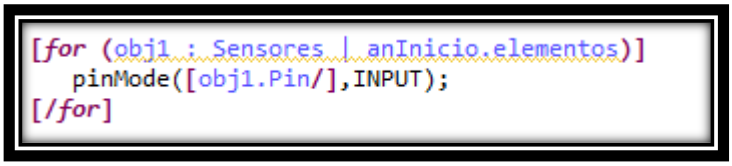


Figura 30 Vista general de acceleo

En la figura que se muestra a continuación se genera un objeto *obj1* de tipo *Sensores* y se accede al atributo *Pin* mediante *[obj1.Pin/]*, esto será utilizado para configurar el pin al cuál se conecta los sensores en Arduino

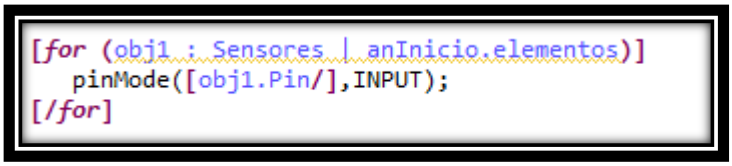


Figura 31 Vista general de acceleo

De forma equivalente, en la figura que se muestra a continuación se genera un objeto *objout* de tipo *Actuadores* y se accede al atributo *pin* mediante *[objout.Pin/]*, esto será utilizado para configurar el pin al cuál se conecta los actuadores en Arduino

```
[for (objout : Actuadores | anInicio.elementos)]
    pinMode([objout.pin/],OUTPUT);
[/for]
```

Figura 32 Vista general de acceleo
Fuente: Elaborado por el Autor

Por otra parte, en la figura que se observa a continuación, se genera un objeto *objSer* de tipo *Serial* y se accede al atributo *velocidadTx* mediante *[objSerial.velocidadTx/]*, esto será utilizado para inicializar el puerto serial y configurar la velocidad de transmisión

```
[for (objSer : Serial | anInicio.elementos)]
    Serial.begin([objSer.velocidadTx/]);
[/for]
```

Figura 33 Vista general de acceleo

De manera similar, en la figura que se muestra a continuación, se genera un objeto *objsb* de tipo *Usb,,* esto será utilizado para inicializar el puerto serial y configurar la velocidad de transmisión a 9600 baudios

```
[for (objusb : Usb | anInicio.elementos)]
    Serial.begin(9600);
[/for]
```

Figura 34 Vista general de acceleo

A continuación, en la figura que se muestra de forma seguida, se genera un objeto *objwifi* de tipo *ESP8266* y se accede a los atributos, *velocidadTx, SSID, PASWORD* mediante *[objwifi.velodicadTx/], [objwifi.SSID/], [objwifi.PASSWORD/]* esto será utilizado para configurar la tarjeta ESP8266 mediante comandos AT,

```
[for (objwifi : ESP8266 | anInicio.elementos)]
    Serial3.begin([objwifi.velocidadTx/]);
    delay(300);
    Serial3.write("AT+CIPMUX=1");
    delay(500);
    Serial3.write("AT+CIPSERVER=1,80");
    delay(500);
    Serial3.write("AT+CWJAP_DEF="[objwifi.SSID/]","[objwifi.PASSWORD/]"");
    delay(500);

[/for]
```

Figura 35 Vista general de acceleo

En la figura que se muestra a continuación, se muestra como determinar el elemento que se encuentra conectado el primer elemento generado dentro del cuadro contenedor *Lógica*.

Para esto, primero se crea un objeto *objlog*, de la clase *Software*, se accede a los objetos que se crearon dentro de este contenedor, utilizando: *objlog.eAllContents()->at(1)*, es necesario convertirlo en un *String* para poder compararlo, con este fin, se utiliza.*toString()*, finalmente se debe realizar las comparaciones del *String*, para determinar el tipo de elemento al cual se interconecta.

```
[for (objlog : Software | anInicio.software)]
[comment determinar primer elemento de la logica /]
    [if (objlog.eAllContents()->at(1).toString().contains('Logic_If1'))]
        [comment cuando se selecciono un if1 /]
            int pin=[objlog.eAllContents()->at(1).toString().substring(
                        objlog.eAllContents()->at(1).toString().indexOf(':')+1,
                        objlog.eAllContents()->at(1).toString().indexOf(',')-1)/];
            int valor=[objlog.eAllContents()->at(1).toString().substring(
                        objlog.eAllContents()->at(1).toString().indexOf(',')).substring(
                        objlog.eAllContents()->at(1).toString().substring(
                        objlog.eAllContents()->at(1).toString().indexOf(',')+1).indexOf(':')+2,
                        objlog.eAllContents()->at(1).toString().substring(
                        objlog.eAllContents()->at(1).toString().indexOf(',')+1).indexOf(','))/];
            [comment determinar condicion /]
            [if (objlog.eAllContents()->at(1).toString().contains('Igual'))]
                if(analogRead(pin)=valor){
            [elseif (objlog.eAllContents()->at(1).toString().contains('Mayor'))]
                if(analogRead(pin)<valor){
            [elseif (objlog.eAllContents()->at(1).toString().contains('Menor'))]
                if(analogRead(pin)>valor){
            [/if]
```

Figura 36 Vista general de acceleo

Por otra parte, en la figura que se muestra a continuación, utilizando el mismo objeto, *objlog*, se accedera al elemento conectado al primer elemento generado, mediante *.eCrossReferences(),* esto se lo puede observar en la sentencia *objlog.eAllContents()->at(1). .eCrossReferences()->at(1)*

```
[if (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('Logic_If1'))]
[elseif (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('Logic_If2'))]
[elseif (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('On'))]
        digitalWrite([objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().substring(
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(':')+2,
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(')')-1)/],HIGH);
[elseif (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('Off'))]
        digitalWrite([objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().substring(
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(':')+2,
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(')')-1)/],LOW);
[elseif (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('Delay'))]
        delay([objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().substring(
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(':')+2,
        objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().indexOf(')')-1)/]);
[elseif (objlog.eAllContents()->at(1).eCrossReferences()->at(1).toString().contains('Serial_Logic'))]
[/if]
```

Figura 37 Vista general de acceleo

CAPÍTULO 4

4. METODOLOGIA PROPUESTA

4.1. Scaffolding

El *Scaffolding* (andamiaje) se trata de un método utilizado para crear aplicaciones, las cuales ayudadas de una base de datos, donde estarán las especificaciones, realizara las acciones que se le predeterminen, esto ayuda al usuario a no tener que estar constantemente actualizando la aplicación sino que con el método de S*caffolding* lo hará más fácilmente (Hincapié, 2016).

El Scaffolding consiste en la generación de código a partir de plantillas predefinidas y de una especificación proporcionada por el desarrollador. Se suele utilizar para generar código repetitivo que se puede especificar fácilmente y generar a partir de una plantilla. De esa forma, el desarrollador se ahorra parte de la codificación ya que solo tendrá que revisar y retocar mínimamente el código generado. Igualmente, el scaffolding puede considerarse como una aplicación informal de los conceptos definidos en el campo de investigación de desarrollo de software dirigido por modelos. (Nieves R. Brisaboa, 2016)

Scaffolding, permite generar productos multilenguaje basados en tecnologías modernas y ampliamente soportadas (HTLM, CSS,JavaScript, etc.) y, mediante el uso de plantillas previamente definidas y bien estructuradas, facilita la generación de código claro, legible y extensible a partir de la especificación del analista. (Nieves R. Brisaboa, 2016)

Scaffolding, un método en el cual ayudara al usuario a entender cómo funciona esta aplicación, se ha visto que en la gran mayoría de aplicaciones graficas-visuales tener un ejemplo ayuda significativamente para familiarizarse con el DSL, es por esto que para esta aplicación se determinó que el utilizar ejemplos ayuda a los principiantes en este caso las personas que vayan a

comenzar a programar puedan hacerlo sin necesidad de generar un código solo con la ayuda de estas plantillas o ejemplos puedan mejorar sus destrezas y llevar cabo tareas muy complicadas.

Según (Nieves R. Brisaboa, 2016) Scaffolding tiene las siguientes funciones

- Dar apoyo
- Servir como herramienta
- Permite alcanzar las metas
- Su manejo es selectivo, cuando es necesario
- Mientras más habilidades la persona adquiera en su uso se puede ir retirando estos ejemplos y la misma persona ir creando nuevos.

Para el desarrollo del presente proyecto se estableció el uso de un método de Scaffolding mediante ejemplos, es por esto, que la paleta de diseño será conformada como se muestra en la **Figura 38**, con una sección específica llamada Ejemplos.

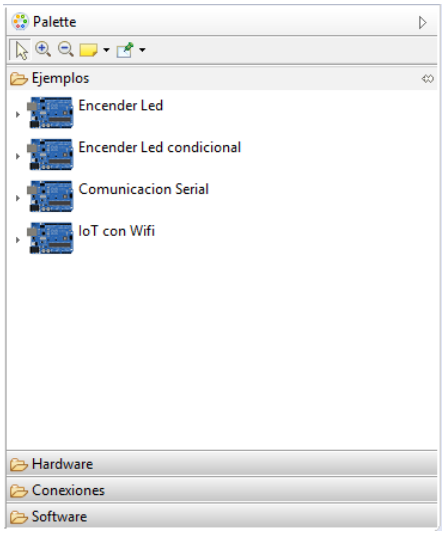


Figura 38 Paleta Ejemplos

Como se observa en la **Figura 39**, al seleccionar uno de los ejemplos de la paleta, se desplegará todos y cada uno de los elementos, con un orden jerárquico en el que se deberán graficar sobre la paleta para realizar la aplicación ejemplificada

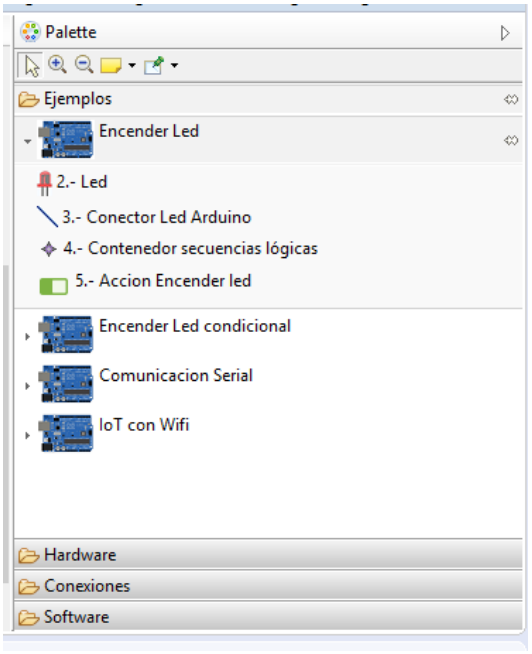


Figura 39 *Scaffolding ejemplo encender led*

Es decir, para realizar una aplicación que Encienda un led (ejemplo 1), será necesario seguir los siguientes pasos:

1. Arrastrar tarjeta Arduino
2. Seleccionar el led
3. Conectar tarjeta Arduino con led
4. Dibujar contenedor de secuencias lógicas
5. Ejecutar acción Encender Led

Realizando de forma secuencial los pasos establecidos en el ejemplo se obtendrá el diagrama mostrado a continuación, mismo que genera la aplicación en la tarjeta de Desarrollo Arduino necesaria para encender un Led.

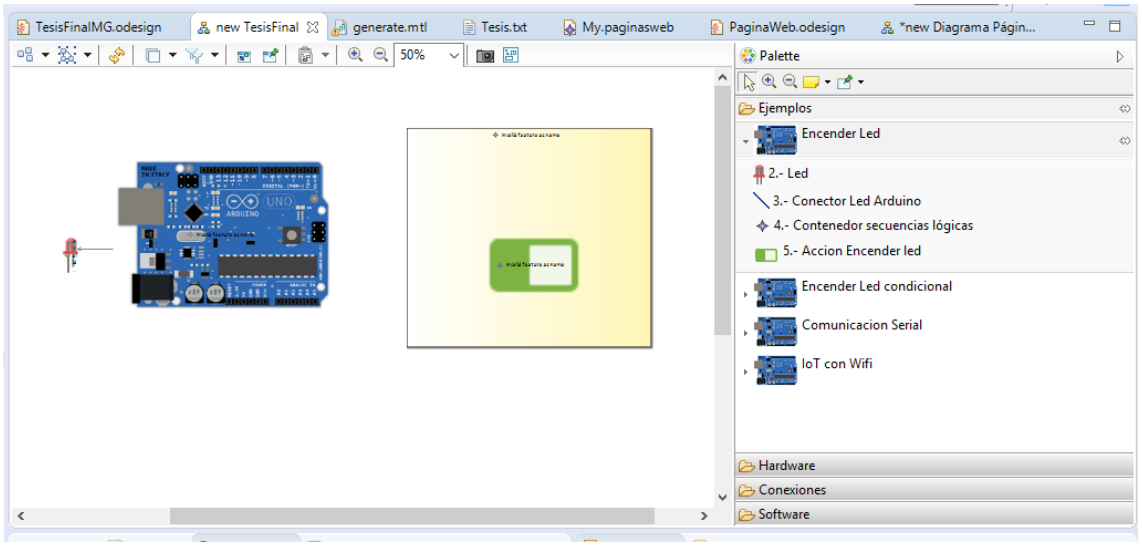


Figura 40 Ejemplo diagrama Scaffolding encender led

4.2. PRACTICA 1

4.2.1. Nombre de la práctica

Control ON/OFF led

4.2.2. OBJETIVO

Desarrollar una aplicación para control on/off de un led mediante la generación de código fuente automático, utilizando la herramienta desarrollada en EMF

4.2.3. LISTADO DE MATERIALES Y HERRAMIENTAS

- Software Eclipse
- Led
- Placa de desarrollo Arduino UNO
- Cables de conexión

4.2.4. INSTRUCCIONES Y ACTIVIDADES A DESARROLLAR

Esta aplicación consiste en controlar el encendido y apagado de un led conectado en el pin 13 del Arduino, el programa deberá cumplir los siguientes parámetros

- Encender el led durante 1 segundo
- Apagar el led durante 1 segundo
- Repetir continuamente la secuencia mencionada
- El led debe estar conectado al pin 13 de arduino

4.2.5. RESULTADOS OBTENIDOS

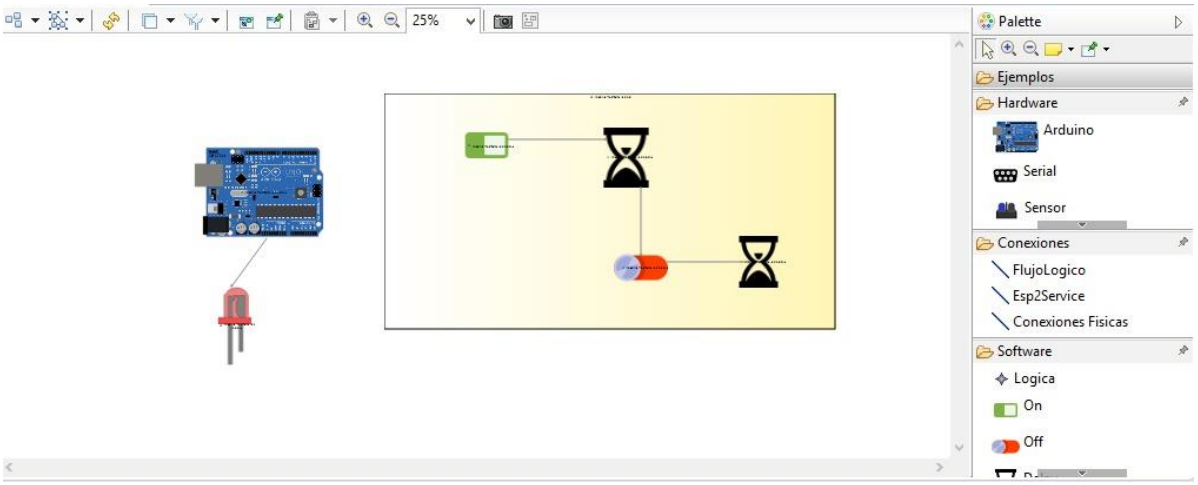


Figura 41 Diagrama Control On/Off

En la Figura 41 se puede observar el diagrama que da solución a la funcionalidad de la aplicación mencionada en este apartado.

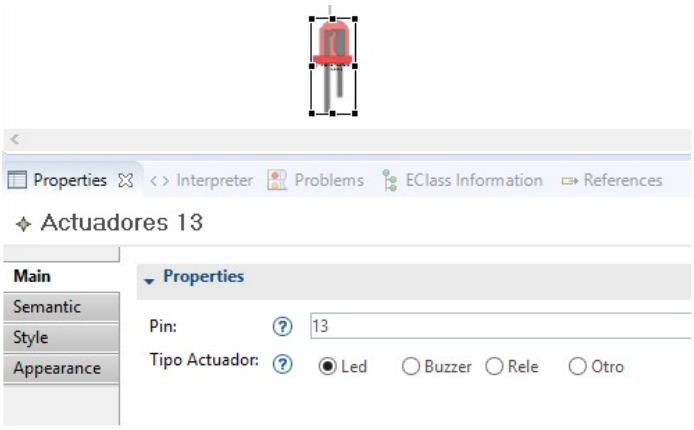


Figura 42 Atributos de los componentes

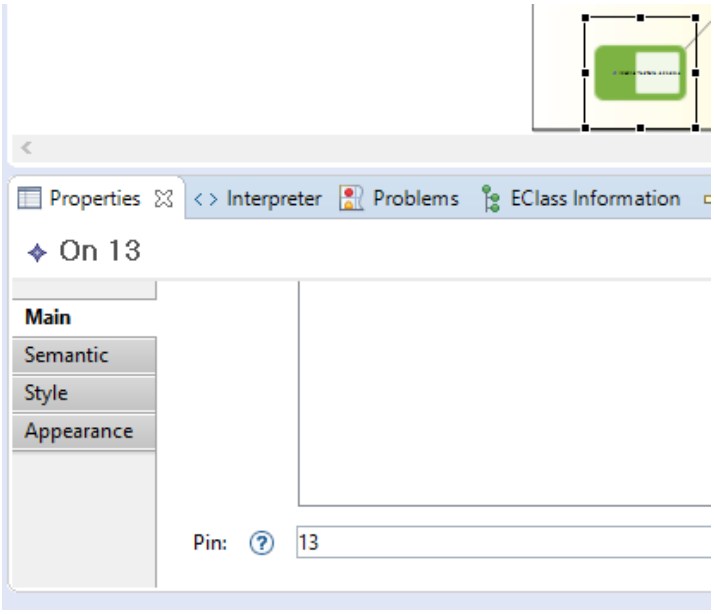


Figura 43 Atributos de los componentes

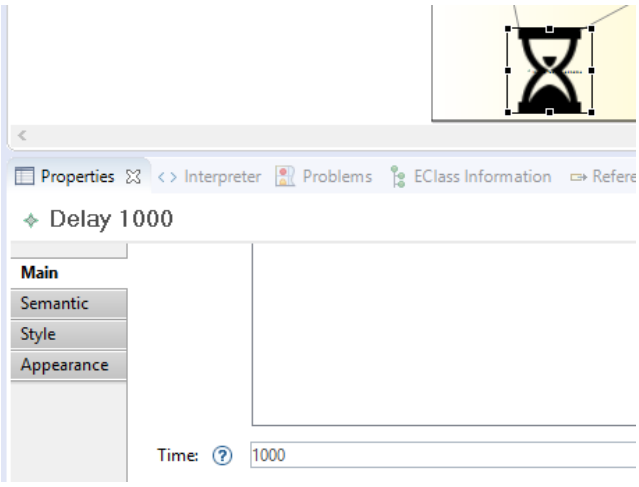


Figura 44 Atributos de los componentes

Aquí se puede cómo deben ser llenados los atributos de los elementos conectados en el programa.

Una vez desarrollado el diagrama mostrado en el Figura 41 Diagrama Control *On/Off*, se obtiene el código fuente que se muestra a continuación. En el cual se puede observar la parte del código generado al colocar el elemento en el diseño del programa.

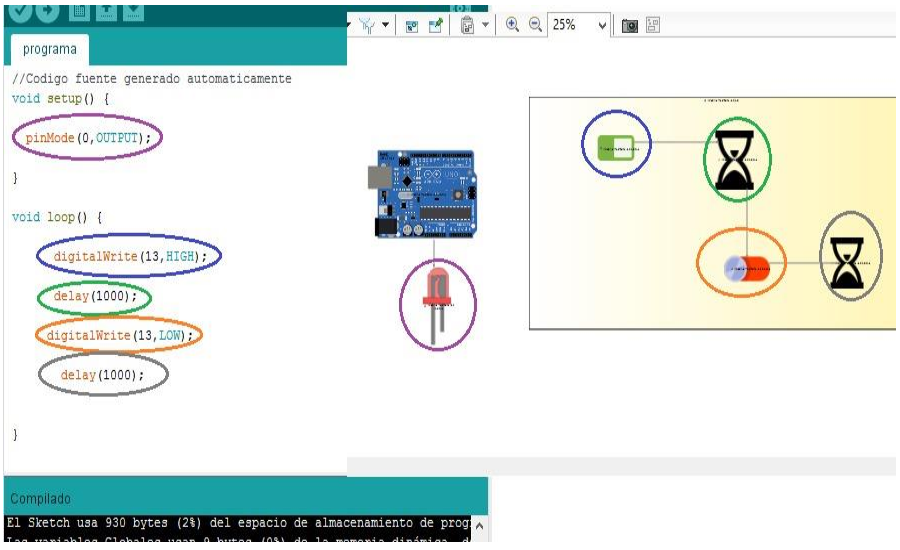


Figura 45 Desarrollo aplicación control ON/OFF

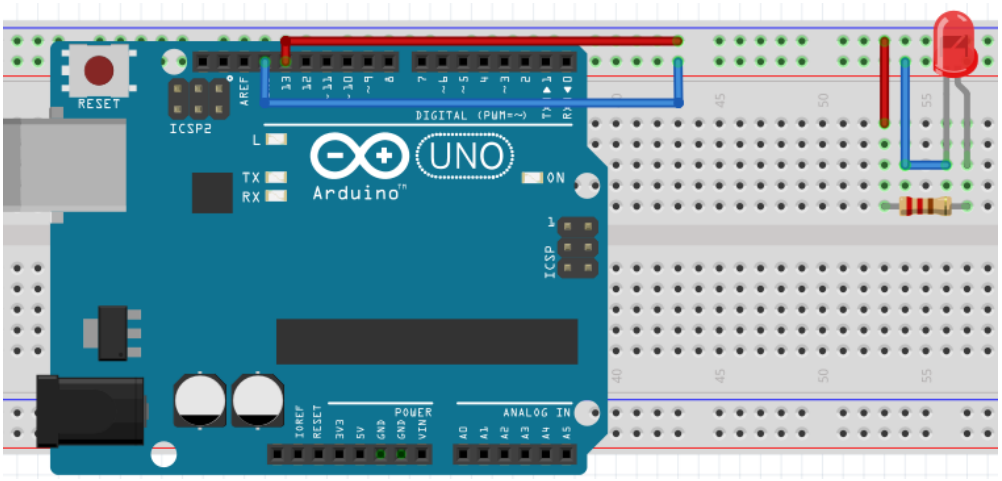


Figura 46 Conexión en el hardware

4.2.6. PREGUNTAS

¿Qué pasaría si conecto en otro pin al led?

4.3. PRACTICA 2

4.3.1. NOMBRE DE LA PRÁCTICA

Control ON/OFF led utilizando operador lógico *if*

4.3.2. OBJETIVO

Desarrollar una aplicación para control on/off de un led con un condicionante if.

4.3.3. LISTADO DE MATERIALES Y HERRAMIENTAS

- Software Eclipse
- Led
- Placa de desarrollo Arduino UNO
- Cables de conexión

4.3.1. INSTRUCCIONES Y ACTIVIDADES A DESARROLLAR

Esta aplicación consiste en controlar de manera automática el encendido de un led mediante la lectura de un sensor de temperatura, el programa deberá cumplir los siguientes parámetros:

- Led conectado al pin 8 de la tarjeta Arduino
- Sensor de temperatura conectado al pin A0 de tarjeta Arduino
- El led deberá encenderse durante 2 segundos cuando la temperatura sobrepase los 30 grados

4.3.2. RESULTADOS OBTENIDOS

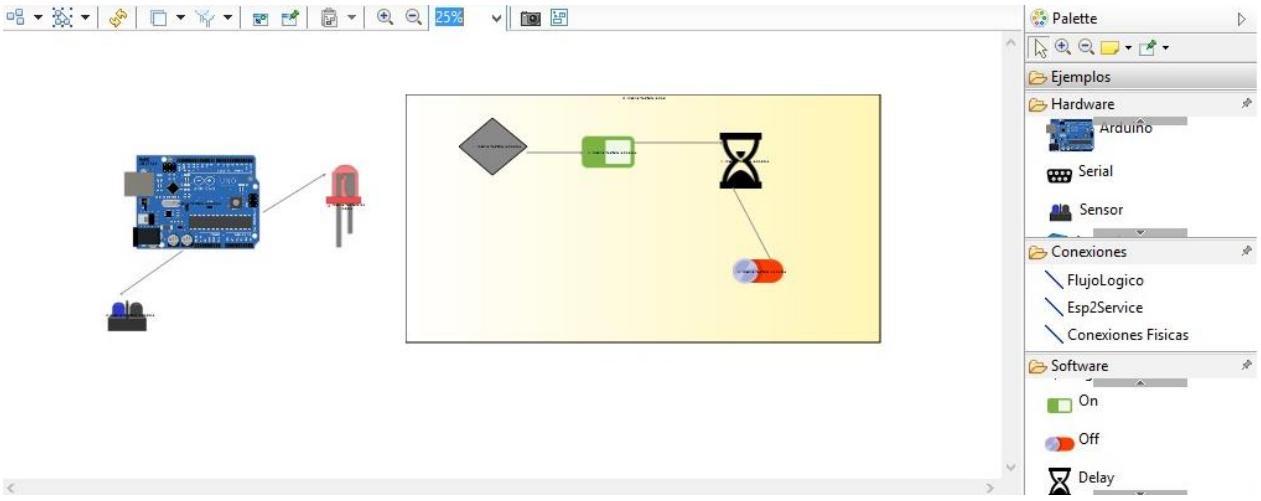


Figura 47 Diagrama Control On/Off utilizando operador lógico if

En la Figura 47 se puede visualizar el diagrama que da solución a la aplicación que se plantea desarrollar.

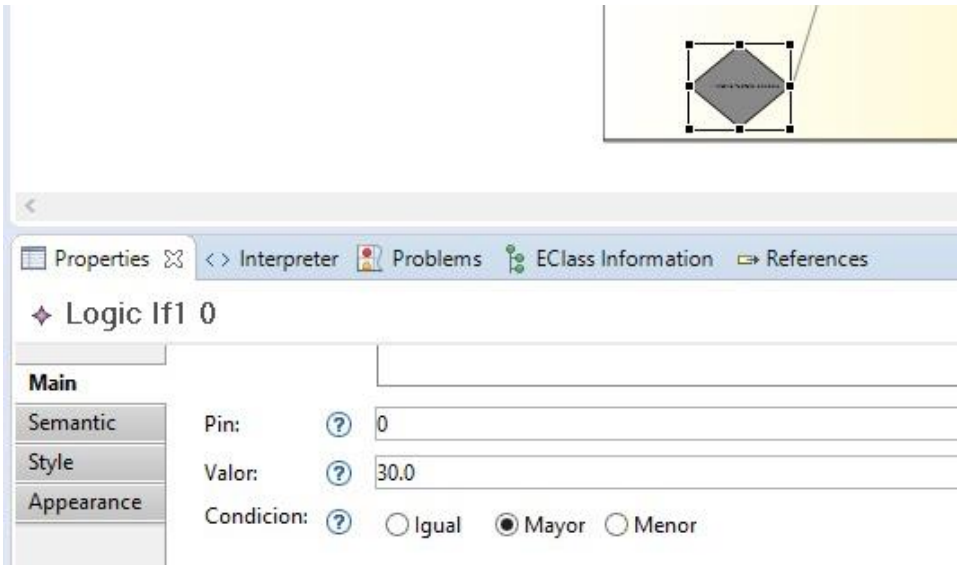


Figura 48 Atributos del componente

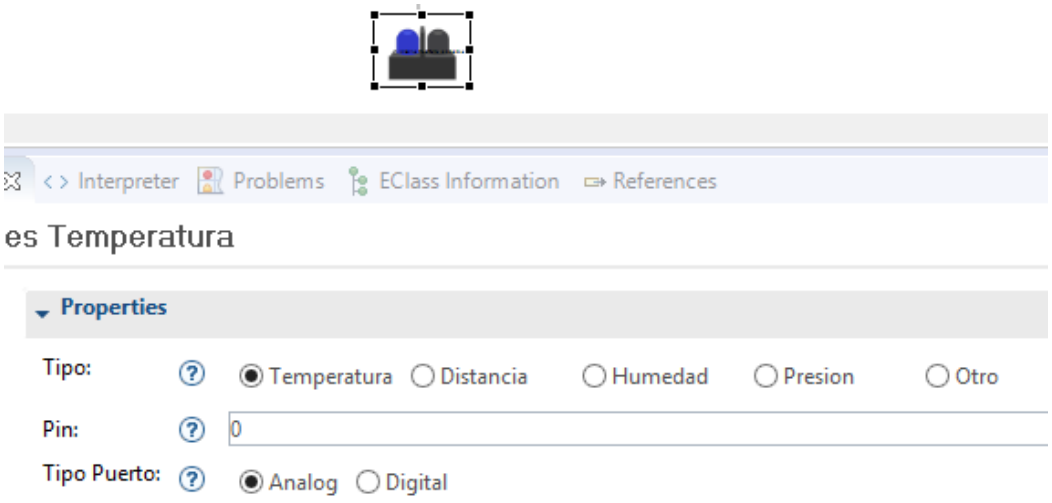


Figura 49 Atributos del componente

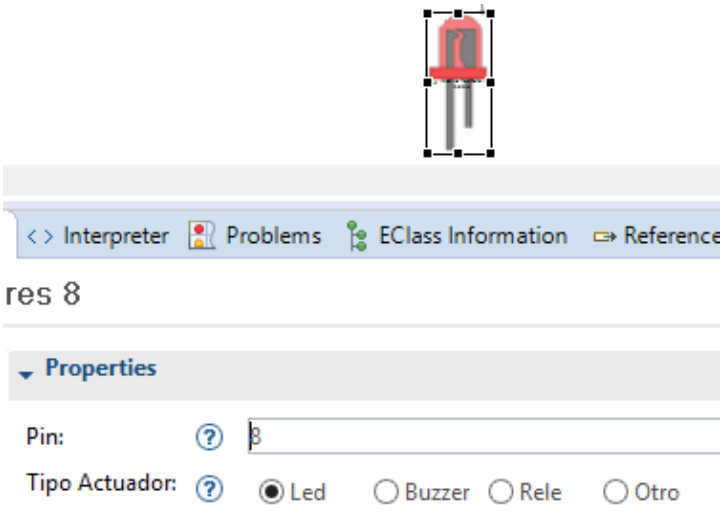


Figura 50 Atributos del componente

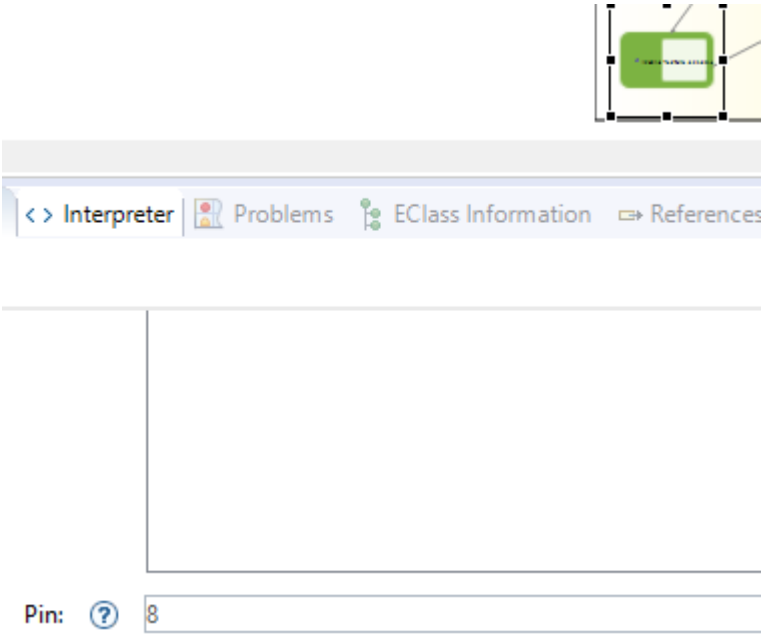


Figura 51 Atributos del componente

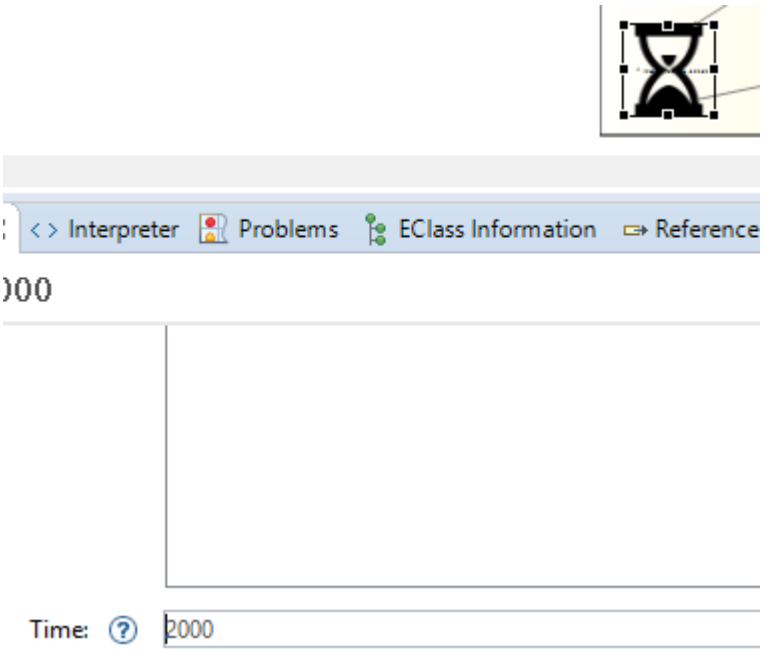


Figura 52 Atributos del componente

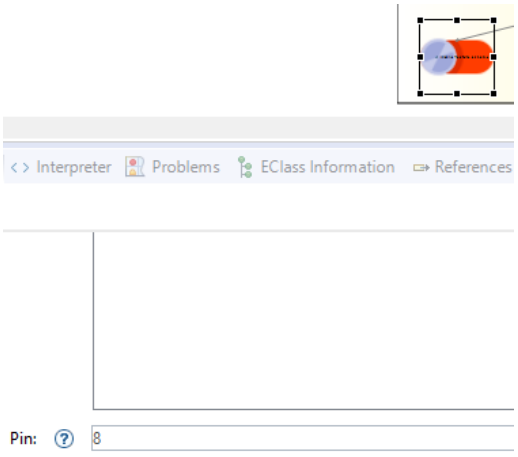


Figura 53 Atributos del componente

Aquí se puede cómo deben ser llenados los atributos de los elementos conectados en el programa.

Una vez desarrollado el diagrama mostrado en el Figura 41 Diagrama Control *On/Off* utilizando operador lógico if, se obtiene el código fuente que se muestra a continuación. En el

cual se puede observar la parte del código generado al colocar el elemento en el diseño del programa.

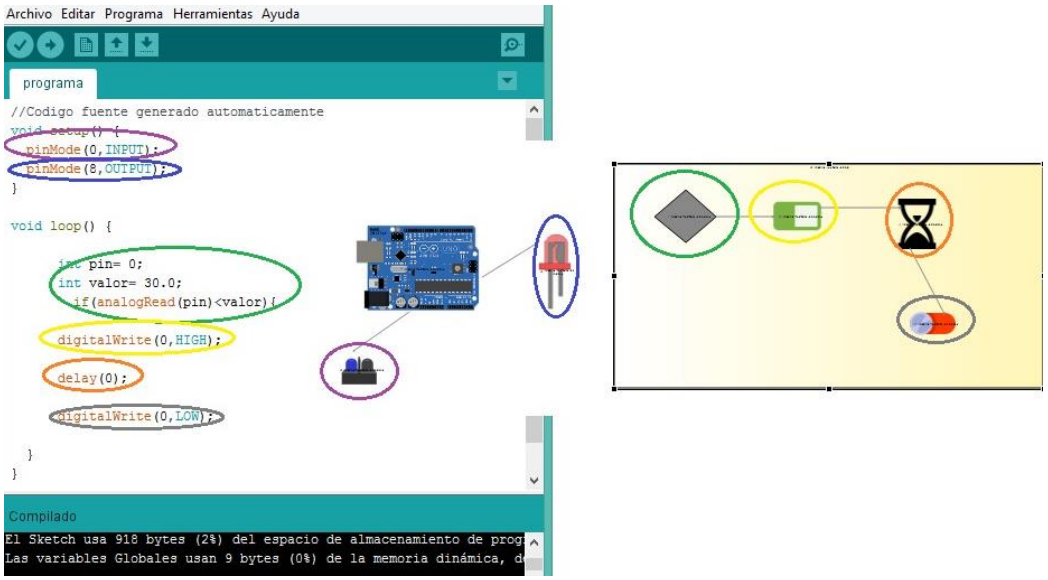


Figura 54 Desarrollo aplicación control ON/OFF utilizando operador lógico if

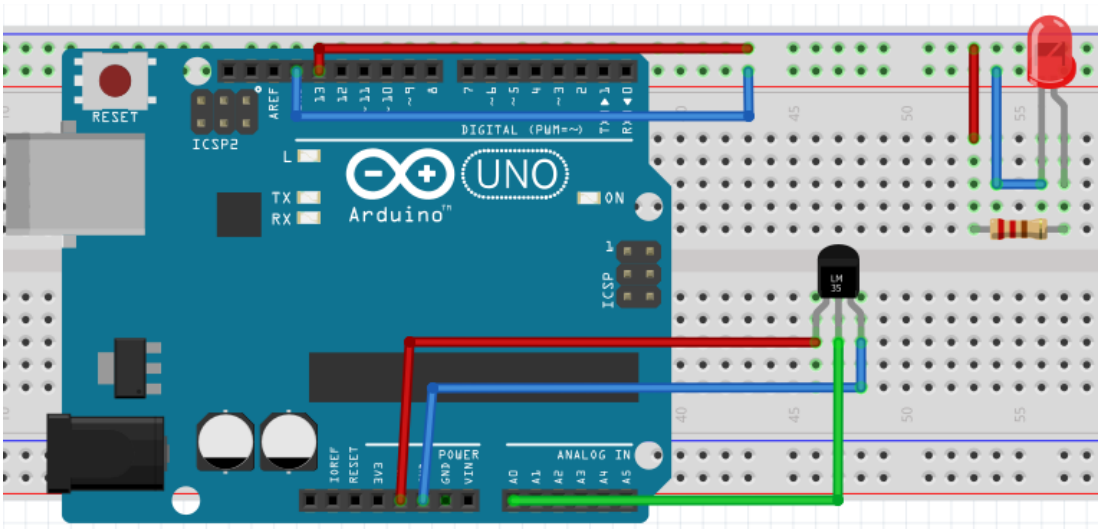


Figura 55 Conexión en el hardware

4.3.3. PREGUNTAS

- ✓ ¿Qué sucede si en el operador lógico if se coloca otra entrada que no es la del sensor?
- ✓ ¿Qué sucedería si en el cuadro lógico si se quita en el elemento off?

4.4. PRACTICA 3

4.4.1. NOMBRE DE LA PRÁCTICA

Comunicación serial

4.4.2. OBJETIVO

Conectar mediante conexión serial para mostrar un mensaje de alerta que diga "Bienvenida".

4.4.3. LISTA DE MATERIALES Y HERRAMIENTAS

- Software Eclipse
- Led
- Placa de desarrollo Arduino UNO
- Cables de conexión

4.4.4. INSTRUCCIONES Y ACTIVIDADES A DESARROLLAR

- Sensor de temperatura conectado en el Pin A2 de la tarjeta Arduino
- Actuador conectado en el pin 9 de la tarjeta de arduino
- Comunicación serial a 9000 baudios
- Enviar mensaje "Bienvenida" utilizando comunicación serial.

4.4.5. RESULTADOS OBTENIDOS

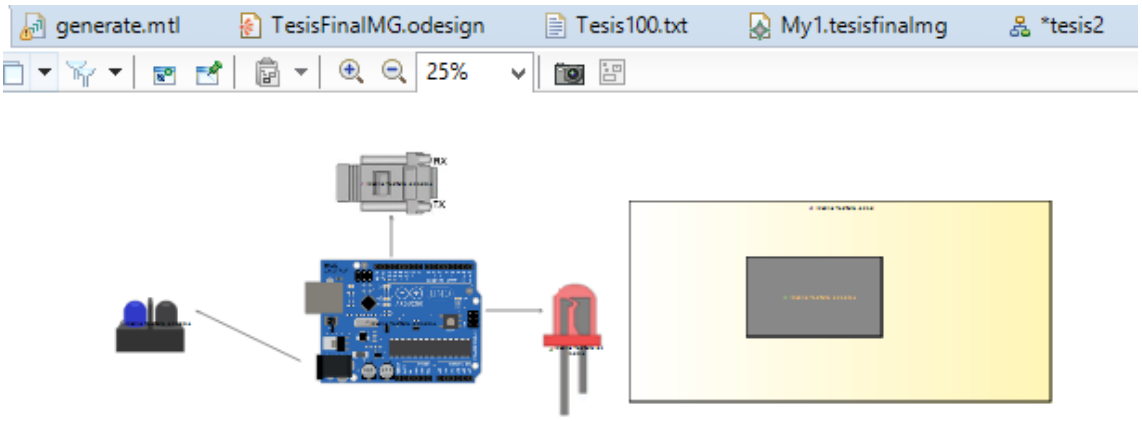


Figura 56 Desarrollo de una aplicación para comunicación serial

En la Figura 47 se puede visualizar el diagrama que da solución a la aplicación que se plantea desarrollar.

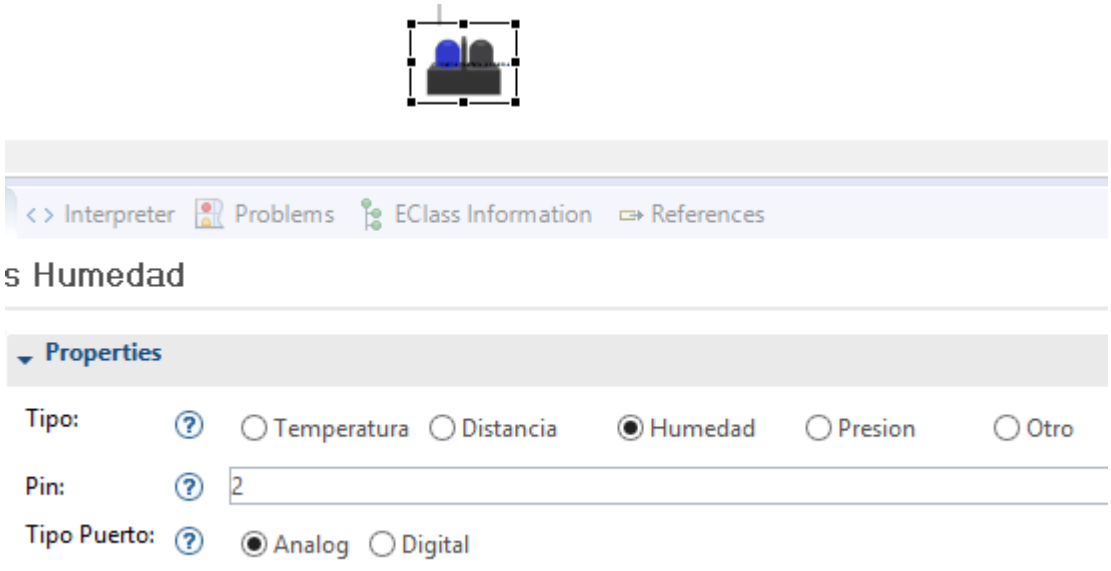


Figura 57 Conexión en el hardware

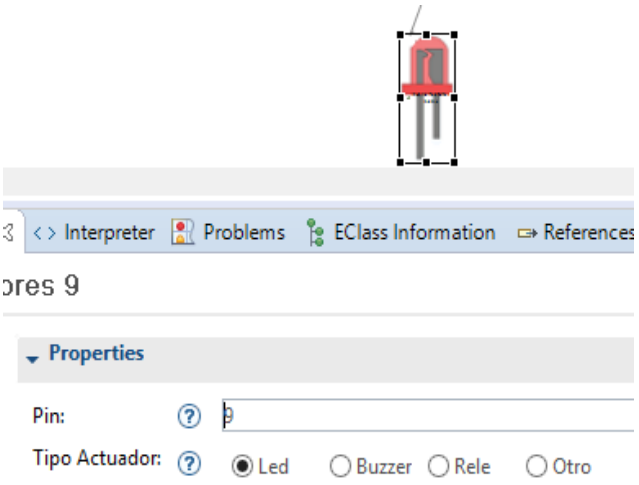


Figura 58 Conexión en el hardware

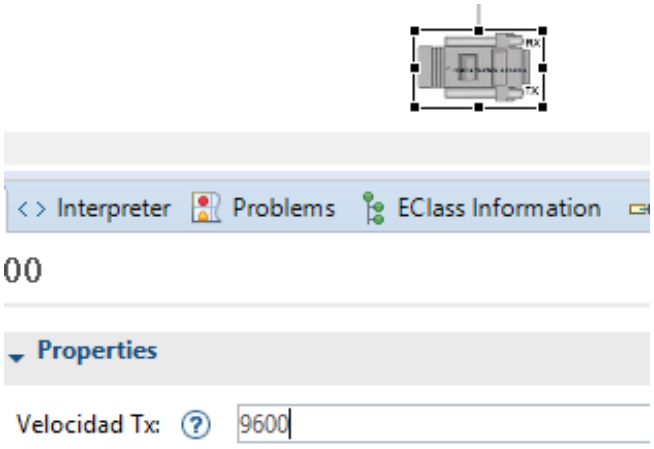


Figura 59 Conexión en el hardware

Figura 60 Conexión en el hardware

Aquí se puede cómo deben ser llenados los atributos de los elementos conectados en el programa.

Una vez desarrollado el diagrama mostrado en el grafico Desarrollo de una aplicación para comunicación serial, se obtiene el código fuente que se muestra a continuación. En el cual se puede observar la parte del código generado al colocar el elemento en el diseño del programa.

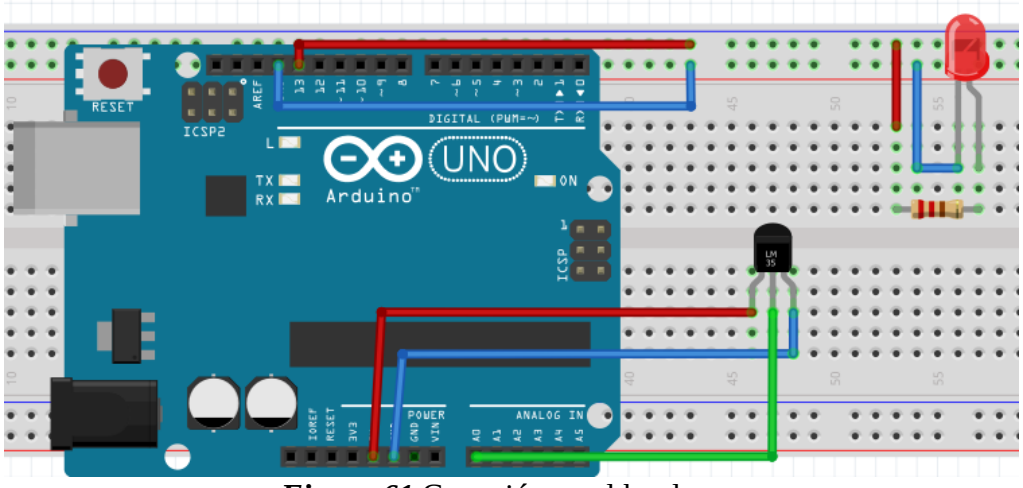


Figura 61 Conexión en el hardware

4.4.6. PREGUNTAS

- ¿Qué sucede si cambio el mensaje que me mostrara en la ventana?

4.5. PRACTICA 4

4.5.1. NOMBRE DE LA PRÁCTICA

Mediante WiFi envió de datos al servidor

4.5.2. OBJETIVO

Desarrollar una aplicación para envió de datos al servidor mediante WiFi

4.5.3. LISTA DE MATERIALES Y HERRAMIENTAS

- Software Eclipse
- Led
- Placa de desarrollo Arduino UNO
- Cables de conexión

4.5.4. INSTRUCCIONES Y ACTIVIDADES A DESARROLLAR

Para esta aplicación se desea efectuar un envió de datos al servidor de manera remota mediante un servicio colocado en la nube, el algoritmo a implementar deberá cumplir los siguientes parámetros:

- El servicio utiliza el método GET
- El módulo ESP8266 requerido para dicha comunicación, deberá conectarse a una red con SSID: “NETLIFE_FLIAGARZON” y PASWORD: “1716197148”
- El módulo ESP8266 se comunicará a una velocidad de transmisión de 9600 baudios
- El servicio enviará el valor del sensor.

- El actuador led debe estar conectado al pin A0 de la tarjeta arduino se deberá ver los datos en el internet.

4.5.4.1. Creación de la base de datos y subida de los archivos php

Para crear una cuenta en el Host “000webhost”, se ingresa a la pagina mostrada en la figura, se ingresa los datos, un correo, una clave, y un nombre para el host

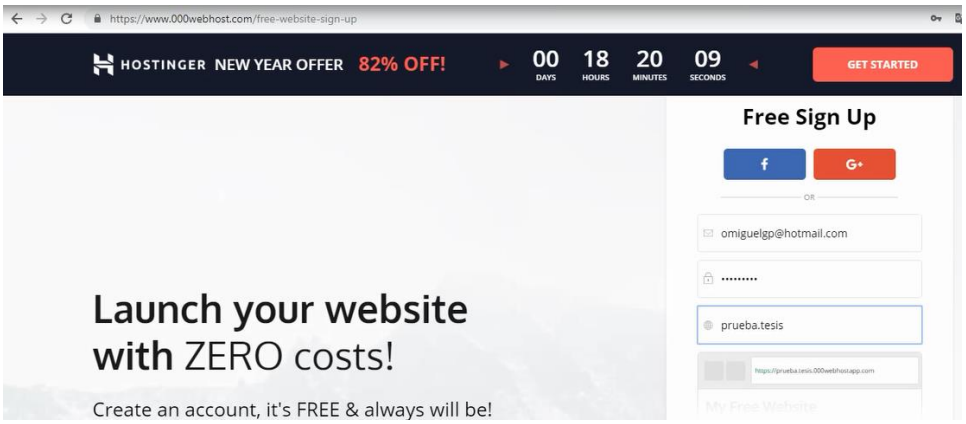


***Figura 62**:* Creación de Host

Luego va a pedir que se confirme un correo.

***Figura 63**:* Email Confirmation

Al momento que se reciba el correo se confirma

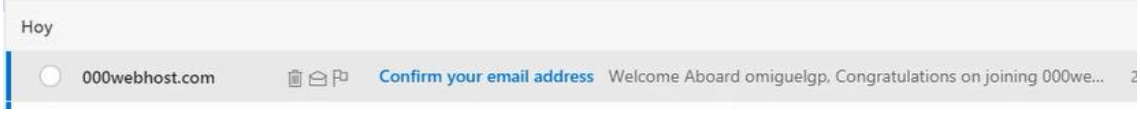


***Figura 64**: Email confirmation*

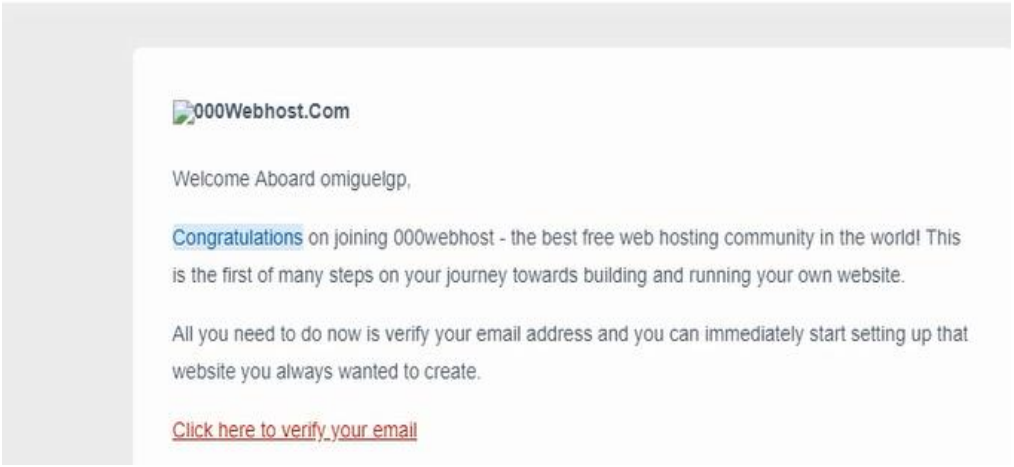


***Figura 65**:* Link de Confirmación

Y se tiene el siguiente mensaje de verificado.

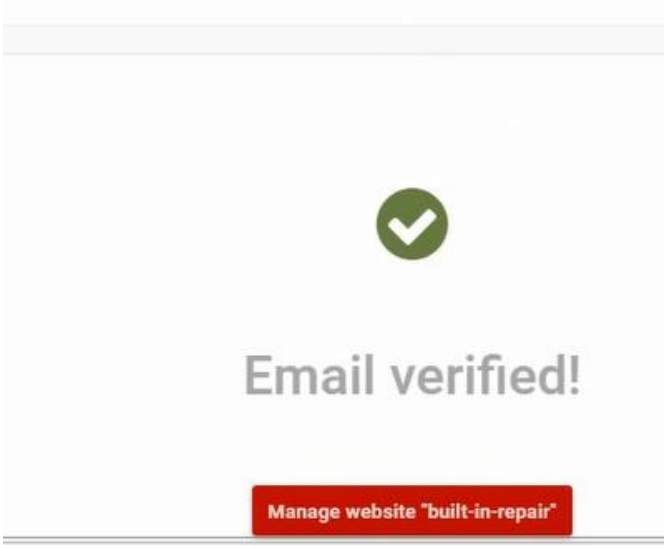


***Figura 66**:* Email verified

Se procede a ingresar nuevamente al Host, y se da click en el icono de +

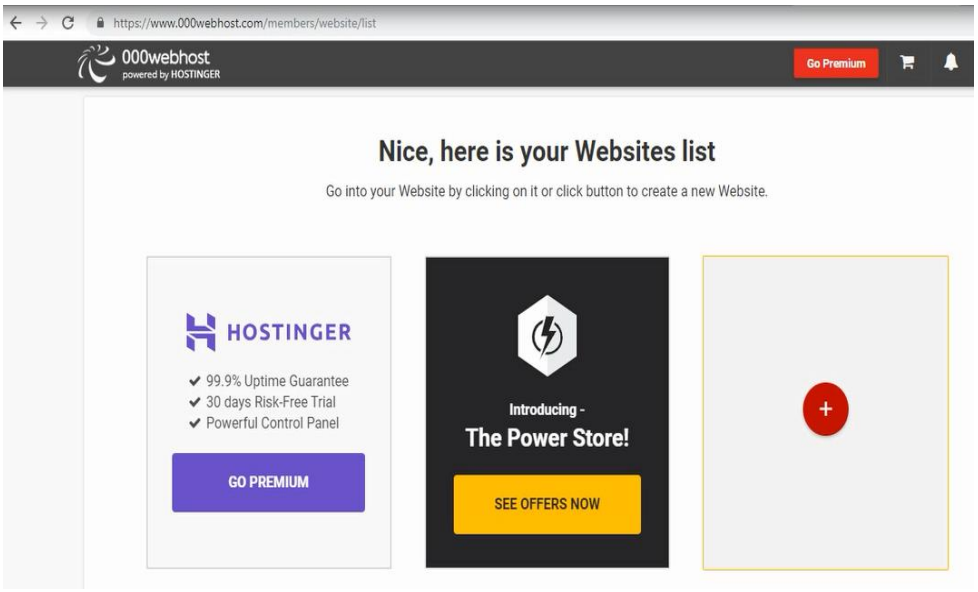


***Figura 67**:* Configuración del host

Se despliega el siguiente mensaje en el cual se escoge la opción que indica en la imagen.

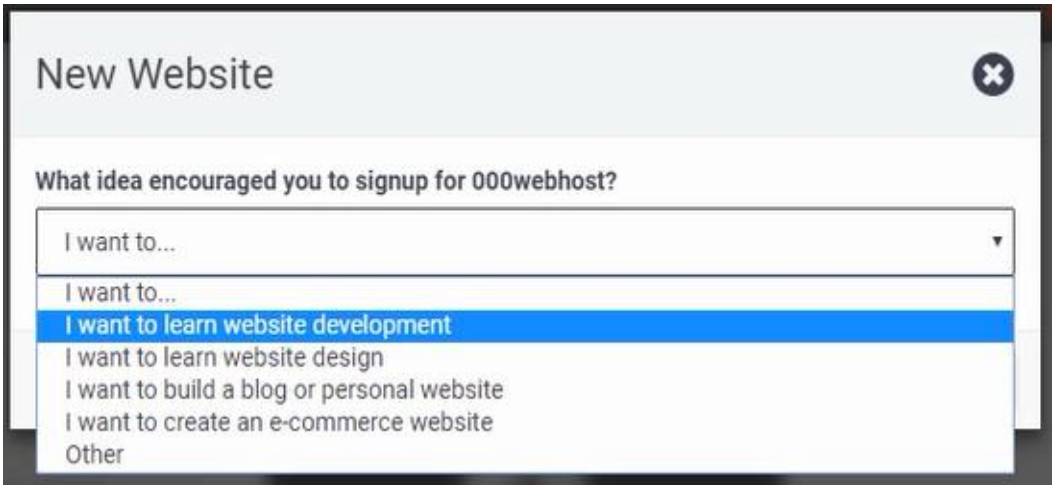


***Figura 68**:* Configuración del host

De igual manera en el siguiente mensaje se coloca un nombre.

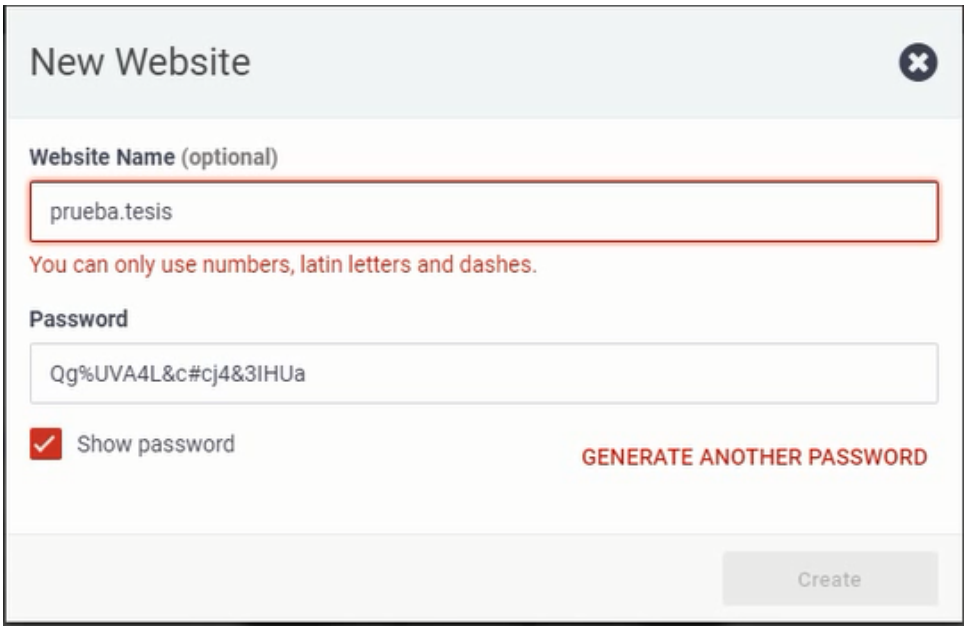


***Figura 69**:* Configuración del host

Una vez configurado el host, se procede a ir al siguiente icono en el cual, se va a crear la base de datos.

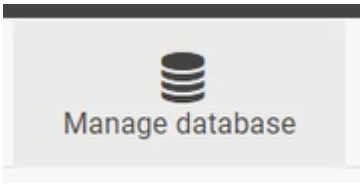


***Figura 70**:* Creación base de datos

Se da click en el botón New Database.

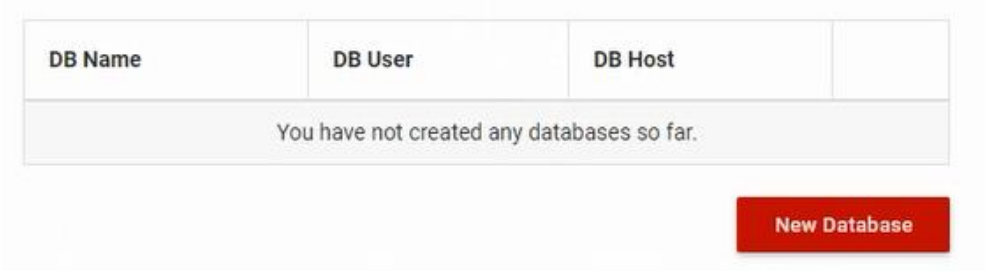


Figura 71: Creación base de datos

Y se procede a configurar los siguientes campos que se ve en la pantalla.

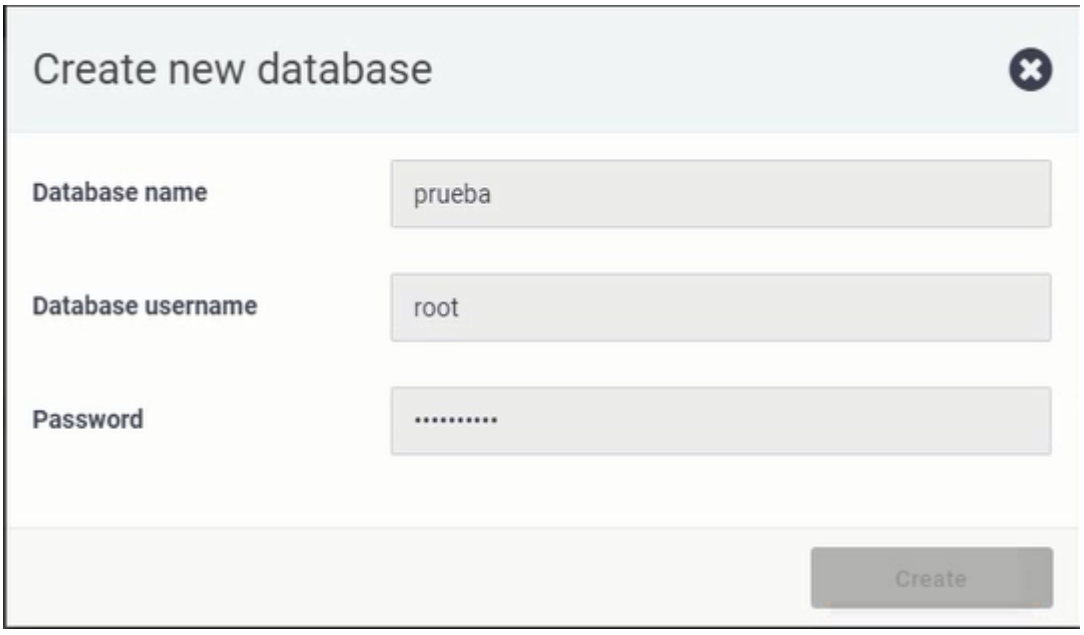


Figura 72: Creación base de datos

La generación de la tabla comienza a cargar, en la cual se ve los nombres con los que la tabla fue creada.

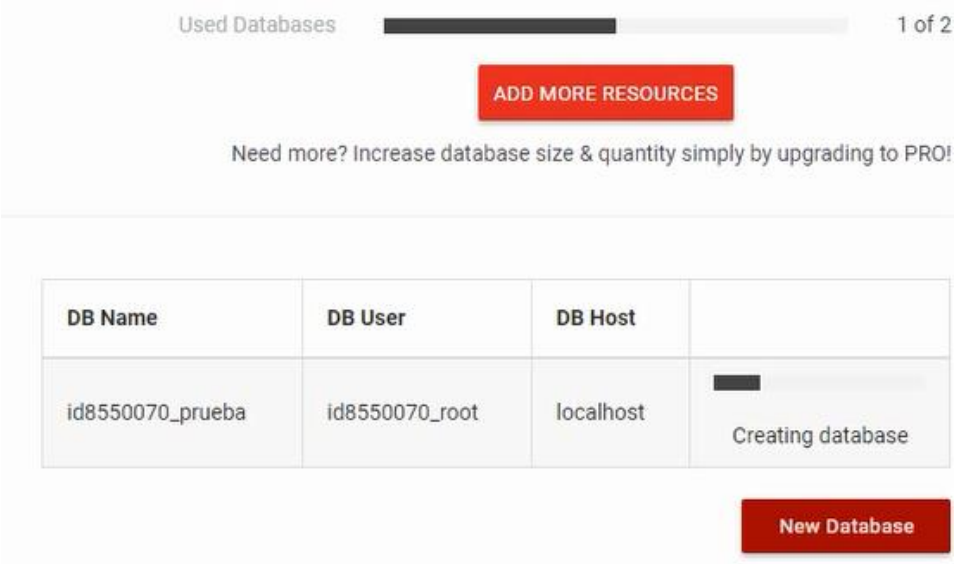


Figura 73: Creación base de datos

Para ingresar a la taba para poder modificar, se pide que se ingrese el idioma, el usuario y clave que se colocó anteriormente.

Figura 74: Creación base de datos

Una vez que se carga la generación de la tabla se procede a configurar los campos que va a requerir en la tabla.

Figura 75: Creación base de datos

Se guardan los cambios y queda lista.

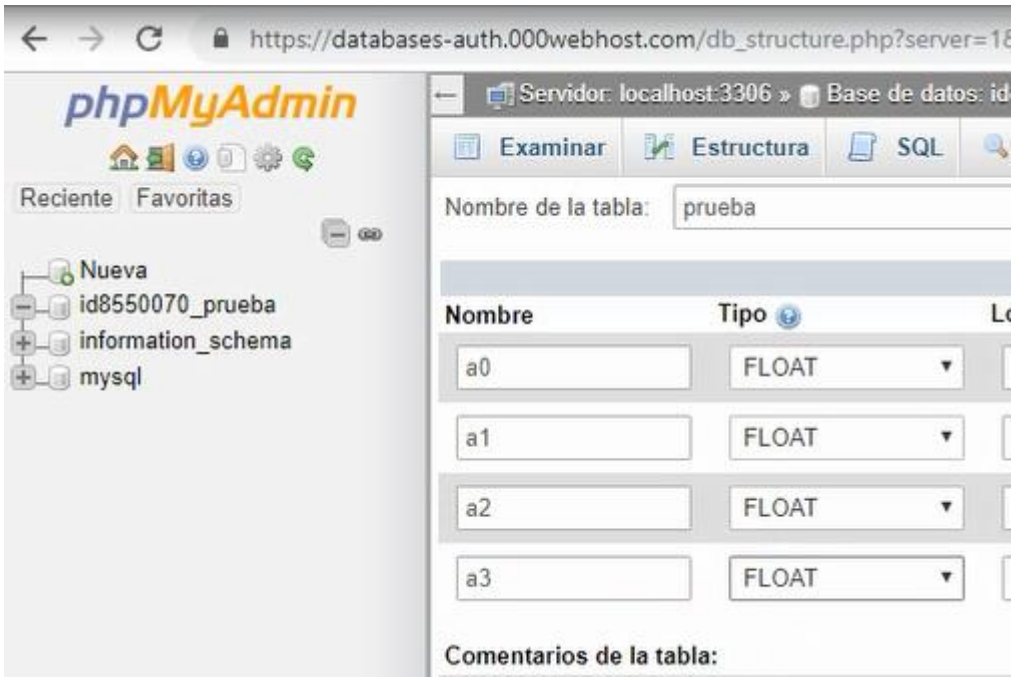


***Figura 76**:* Creación base de datos

Para que todo funcione con normalidad se procede a ir a Settings, en el cual se escoge la opción general.

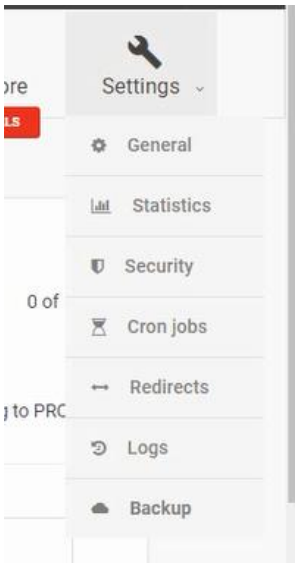


***Figura 77**:* Configuración de la versión

En la versión, se escoge la versión PHP 5.3

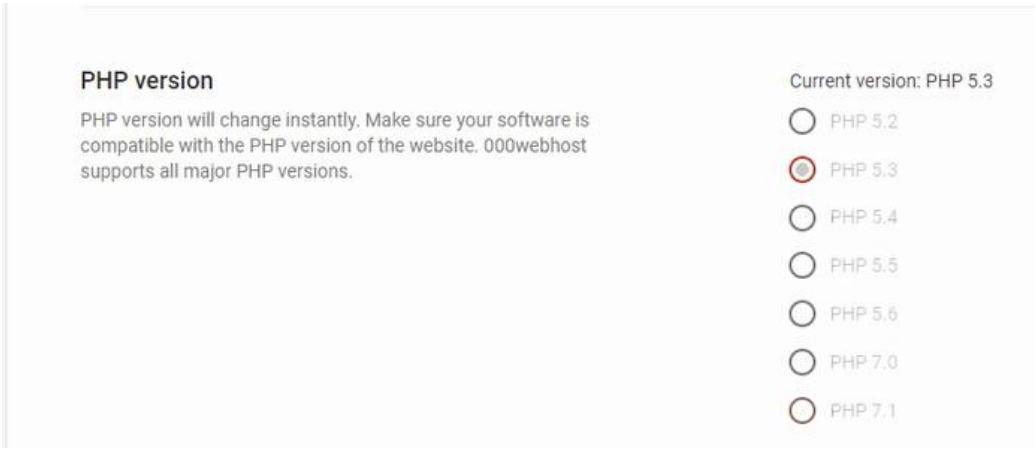


***Figura 78**:* Configuración de la versión

Luego, se ingresa a file manager y se suben los archivos PHP, y 1 de ajax

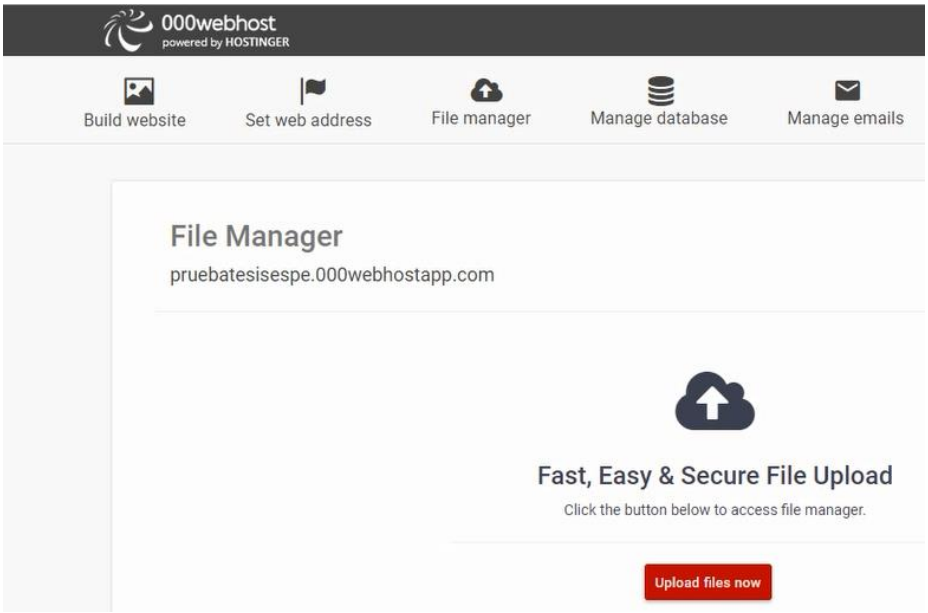


***Figura 79**:* File Manager

Los archivos que se deben subir son los siguientes:

/public_html/php/config2.php

```php
<?php



$host_db = "localhost";
 $user_db = "id8550070_root";
 $pass_db = "123456789a";
 $db_name = "id8550070_prueba";



mysql_connect("$host_db", "$user_db", "$pass_db")or die("Cannot Connect to Data Base.");
    mysql_select_db("$db_name")or die("Cannot Select Data Base");

?>

```

Figura 80 Código Config 2.php

/public_html/php/comprobar.php

```php

<?php

require_once("config2.php");


$g=$_GET['q'];
$q=split('/', $g);

$sql = "update prueba set a0='$q[0]',a1='$q[1]',a2='$q[2]',a3='$q[3]'";
$res = mysql_query($sql);
if($res==1){
echo "DATOS ENVIADOS CON EXITO";
}
else{
    echo "ERROR AL ENVIAR EL DATO";
}

?>

```

Figura 81 Código comprobar.php

/public_html/php/miguel2.php

```php
<?php

require_once("config2.php");
$sql = "SELECT * FROM `prueba`";
$res = mysql_query($sql);
$row = mysql_fetch_array($res, MYSQL_ASSOC);
$a0 = $row['a0'];
$a1 = $row['a1'];
$a2 = $row['a2'];
$a3 = $row['a3'];


echo "@".$a0."@".$a1."@".$a2."@".$a3."@";



?>
```

Figura 82 Código miguel2.php

/public_html/php/miguel1.php

```
1
2  <script src="../js/ajax.js"></script>
3  <script type="text/javascript">
4
5
6  function cocina(){
7    loadDoc("","miguel2.php",function(){
8      if (xmlhttp.readyState==4 && xmlhttp.status==200){
9            var a=xmlhttp.responseText;
10  var b=a.split("@");
11
12
13   for (var i=1;i<5;i++){
14 if(b[i]!="-9999"){
15   document.getElementById("a"+i).innerHTML="Sensor"+i+" :"+b[i];
16   }
17   else{
18   document.getElementById("a"+i).innerHTML="Sensor"+i+" : No conectado";
19
20   }
21   }
22
23     }
24 });
25 }
26
27 setInterval(cocina,2000);
28
29 window.onload=cocina();
30
31 </script>
32
33 <center>
34
35 <h1 id="a1"></h1>
36 <h1 id="a2"></h1>
37 <h1 id="a3"></h1>
38 <h1 id="a4"></h1>
39 <center><div id="myDiv4"></div></center>
40
41
42 </center>
43
44
```

Figura 83 Códigomiguel1.php

/public_html/js/ajax.js

```
var xmlhttp;
function loadDoc(string,url,cfunc)
{
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=cfunc;
xmlhttp.open("POST",url,true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send(string);
}
```

Figura 84 Código Ajax.js

Una vez subidos los programas se configura el programa config2.pp

Figura 85*:* 00webhost

Se configura con los nuevos datos de la tabla.

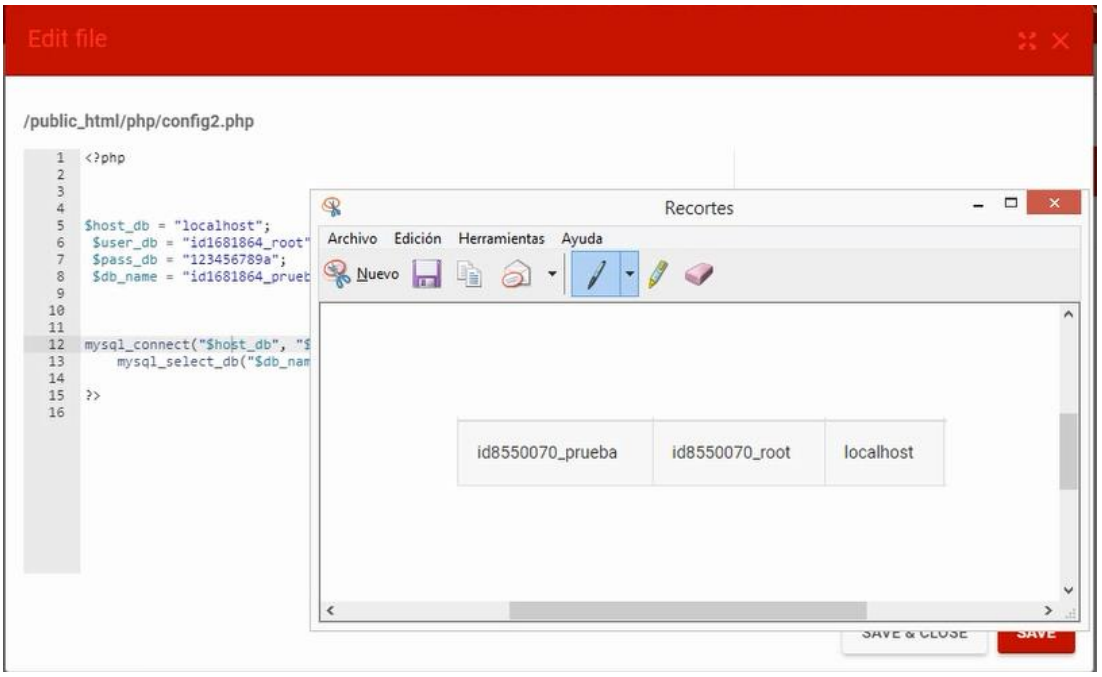


***Figura 86**:* Edit file

Como el programa no realiza ingresos a la tabla, lo que realiza es un update, se debe ingresar a la base de datos y colocar el siguiente código que se ve en la figura

***Figura 87**:* phpMyAdmin

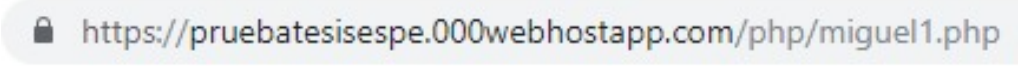


Figura 88 URL de la aplicación en IoT

4.5.5. RESULTADOS OBTENIDOS

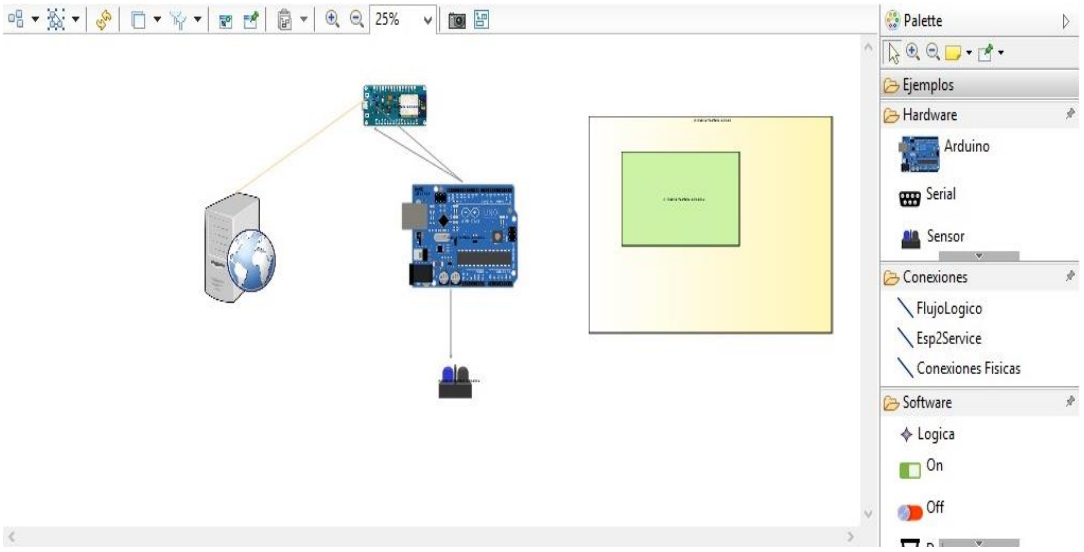


Figura 89 Desarrollar una aplicación para envió de datos al servidor mediante WiFi

En la Figura 89 se puede visualizar el diagrama que da solución a la aplicación que se plantea desarrollar.

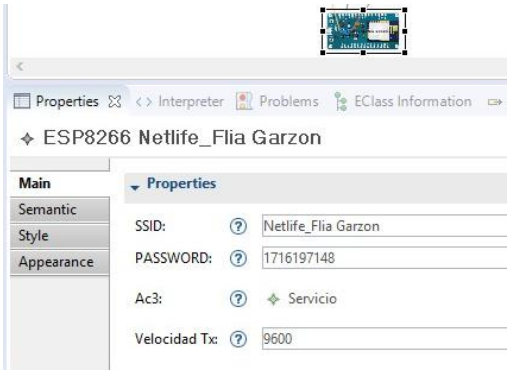


Figura 90 Atributos de los elementos

Aquí se puede cómo deben ser llenados los atributos de los elementos conectados en el programa.

Una vez desarrollado el diagrama mostrado en el grafico Desarrollar una aplicación para envió de datos al servidor mediante WiFi, se obtiene el código fuente que se muestra a continuación. En el cual se puede observar la parte del código generado al colocar el elemento en el diseño del programa.

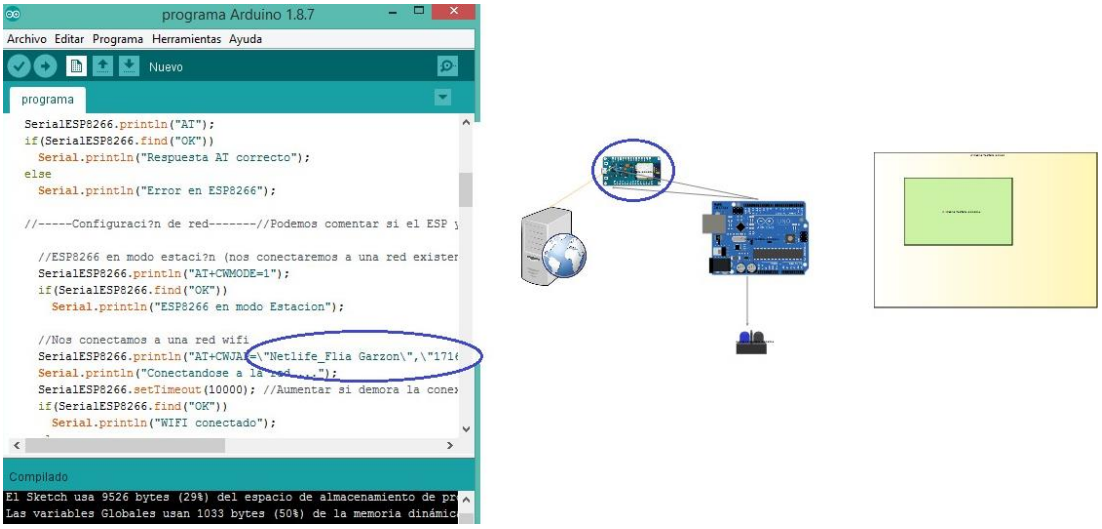


Figura 91 Desarrollo ejemplo de aplicación utilizando Wifi

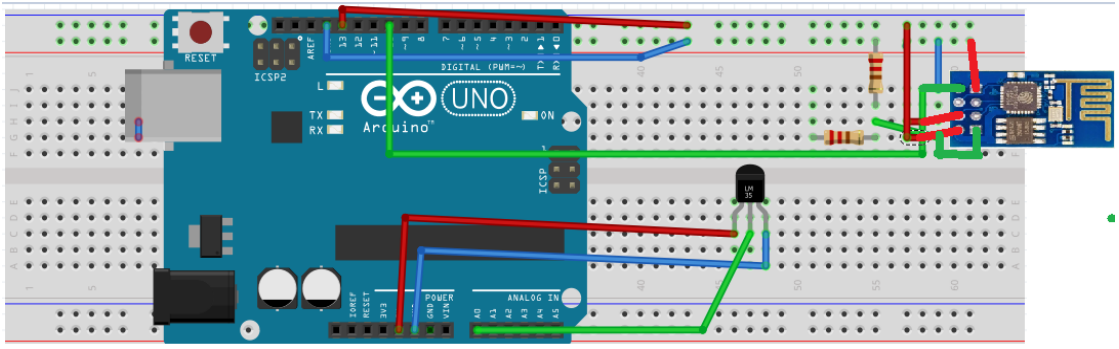


Figura 92 Conexiones hardware

En la figura 42 se muestra la URL en la cual se puede observar los datos obtenidos del sensor

En la figura 43 se ve los datos que el sensor envió al servicio web

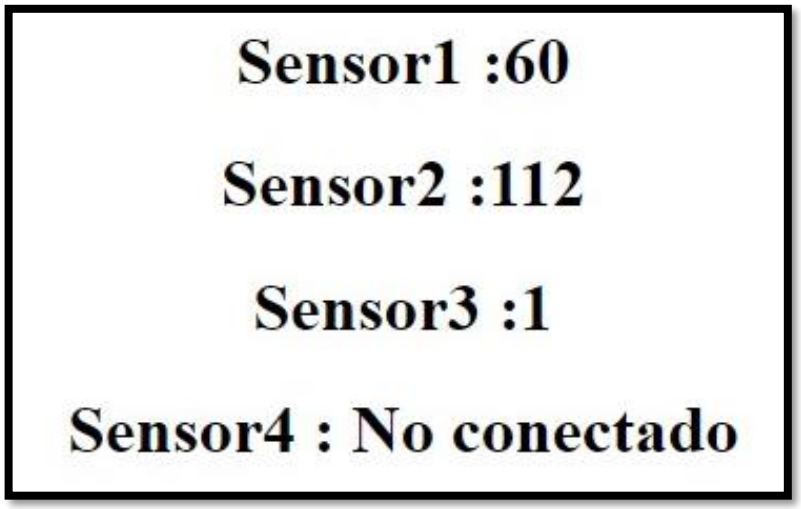


Figura 93 Datos enviados del sensor

4.5.6. PREGUNTAS

- ✓ ¿Qué sucede si se coloca mal el dato SSID, existe conexión con el servidor?

4.6. PRACTICA 5

4.6.1. NOMBRE DE LA PRÁCTICA

Mediante WiFi recibo de datos desde el servidor

4.6.2. OBJETIVO

Desarrollar una aplicación para recibo de datos al servidor mediante WiFi

4.6.3. LISTA DE MATERIALES Y HERRAMIENTAS

- ➢ Software Eclipse
- ➢ Led
- ➢ Placa de desarrollo Arduino UNO
- ➢ Cables de conexión

4.6.4. INSTRUCCIONES Y ACTIVIDADES A DESARROLLAR

- El servicio utiliza el método get
- El modulo requerido para dicha comunicación deberá conectarse a una red. Con SSID: “Familia Garzon” Pasword: 12345678
- El led actuar debe estar conectado al pin 13 de la tarjeta arduino.

4.6.4.1. Creación de la base de datos y subida de los archivos php

Para crear una cuenta en el Host “000webhost”, se ingresa a la pagina mostrada en la figura, se ingresan los datos, un correo, una clave, y un nombre para el host

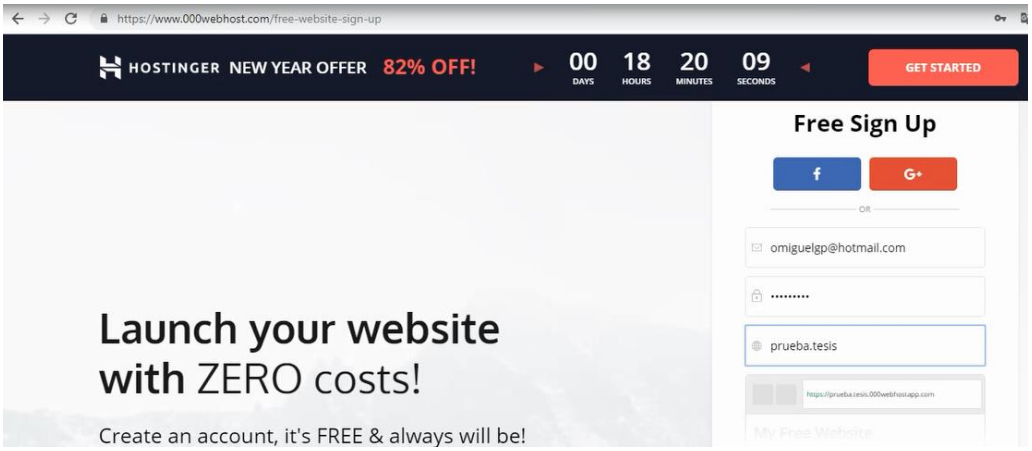


***Figura 94**: Creación de Host*

Luego se va a pedir que se confirme un correo.

***Figura 95**: Email Confirmation*

Al momento que se recibe el correo se confirma

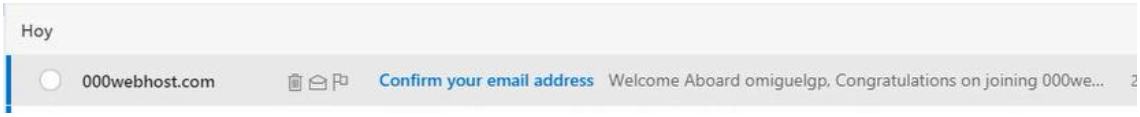


***Figura 96**: Email confirmation*

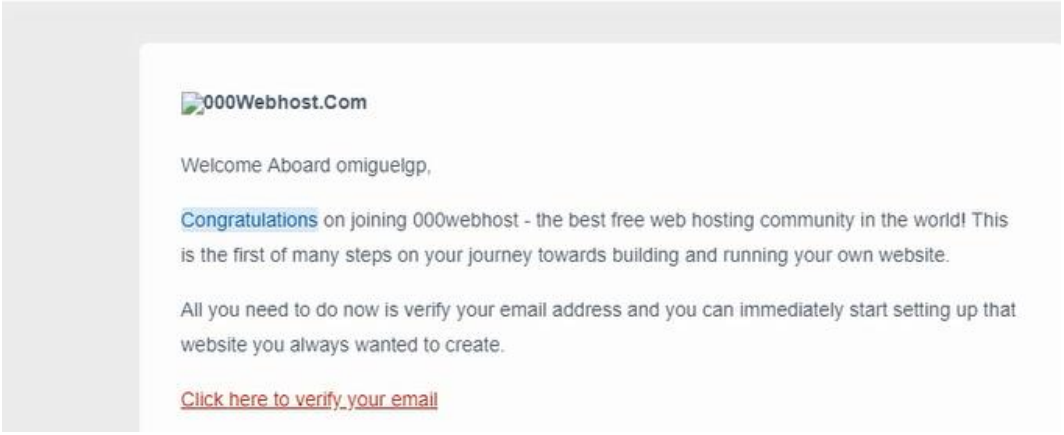


Figura 97: Link de Confirmación

Y se tiene el siguiente mensaje de verificado.

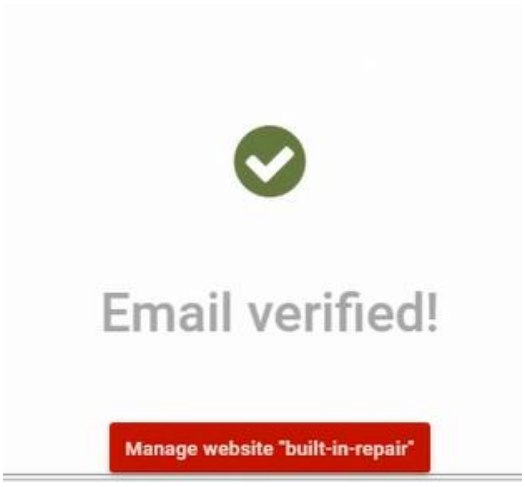


***Figura 98**: Email verified*

Se procede a ingresar nuevamente al Host, y se da click en el icono de +

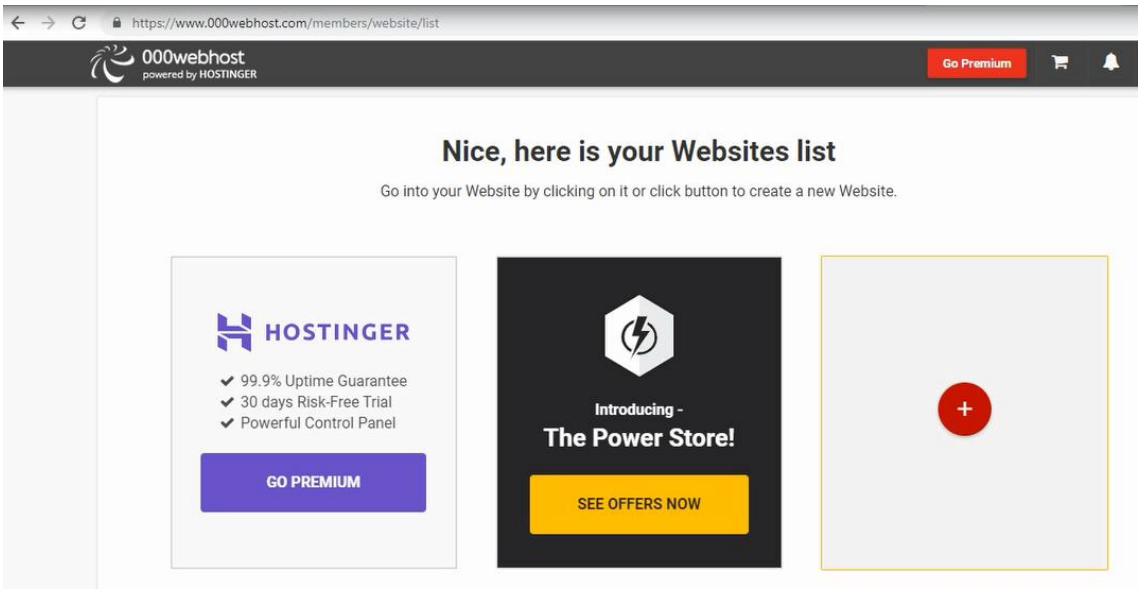


Figura 99: Configuración del host

Se despliega el siguiente mensaje en el cual se escoge la opción que se indica en la imagen.

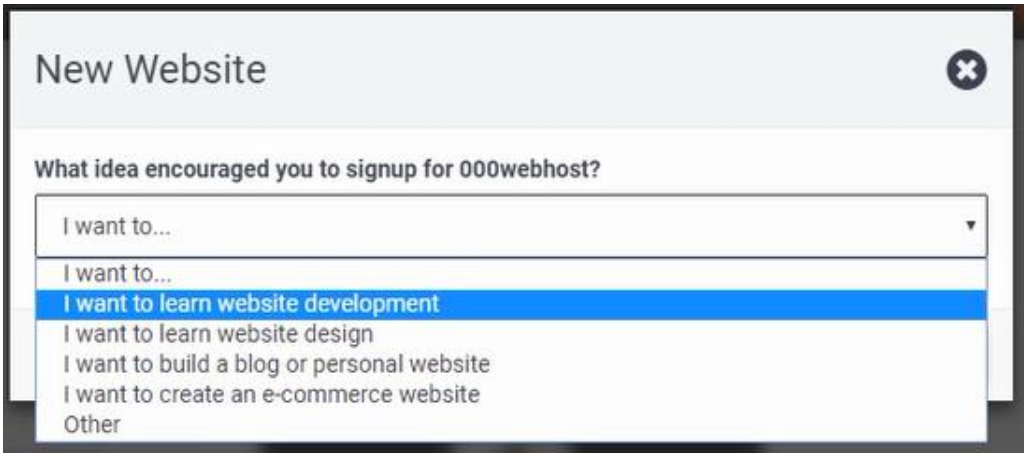


Figura 100*:* Configuración del host

De igual manera en el siguiente mensaje se coloca un nombre.

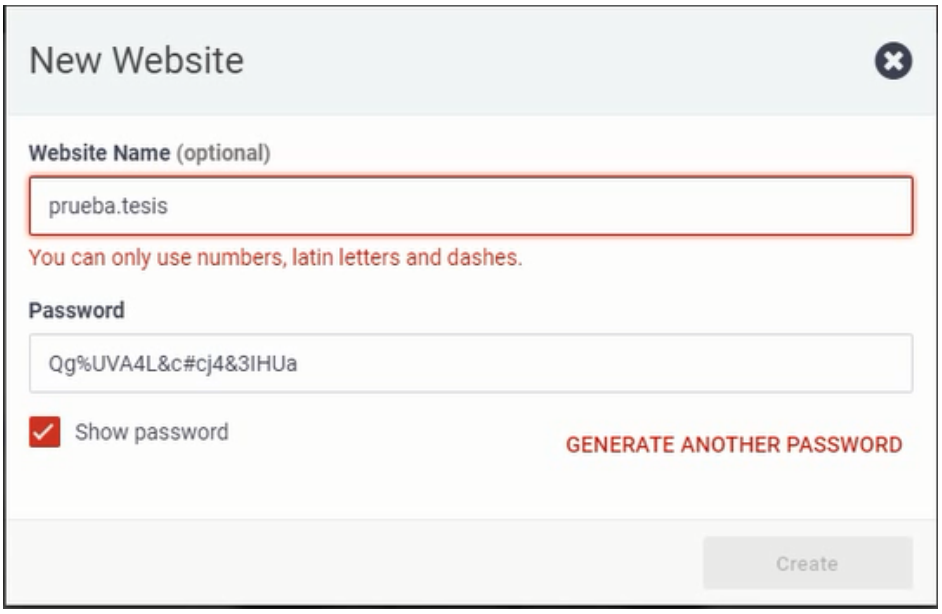


***Figura 101**:* Configuración del host

Una vez configurado el host, se procede a ir al siguiente icono en el cual, se va a crear la base de datos.

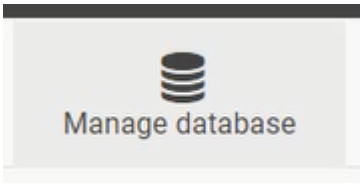


***Figura 102**:* Creación base de datos

Se da click en el botón New Database.

***Figura 103**:* Creación base de datos

Y se procede a configurar los siguientes campos que se ve en la pantalla.

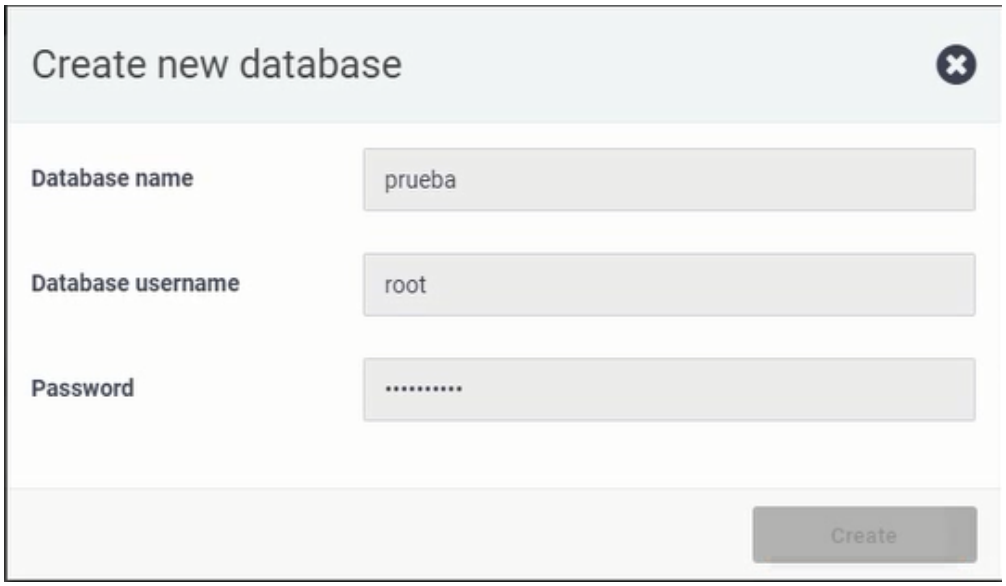


***Figura 104**:* Creación base de datos

La generación de la tabla comienza a cargar, en la cual se muestra los nombres con los que la tabla fue creada.

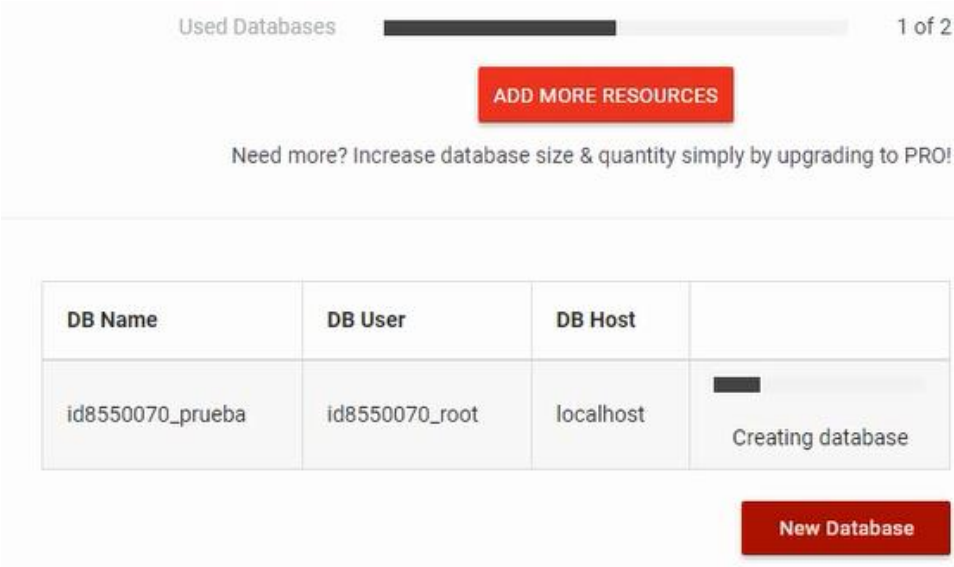


***Figura 105**:* Creación base de datos

Para ingresa a la tabla para poder modificar, se pide que se ingrese el idioma, el usuario y clave que se colocó anteriormente.

***Figura 106**:* Creación base de datos

Una vez que se carga la generación de la tabla se procede a configurar los campos que va a requerir en la tabla.

***Figura 107**:* Creación base de datos

Se guardan los cambios y queda lista.

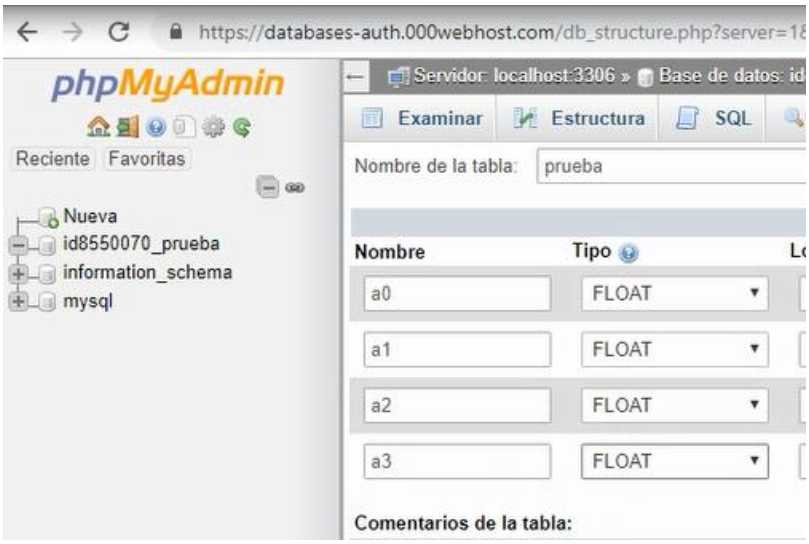


***Figura 108**:* Creación base de datos

Para que todo funcione con normalidad se procede a ir a Settings, en el cual se escoge la opción general.

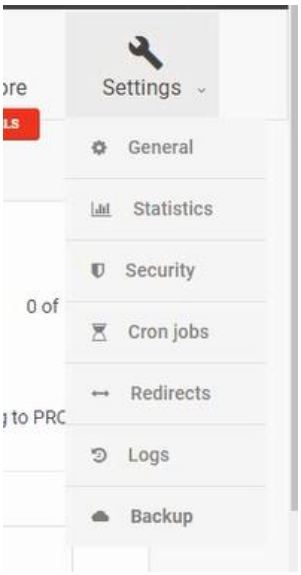


***Figura 109**:* Configuración de la versión

En la versión, se escoge la versión PHP 5.3

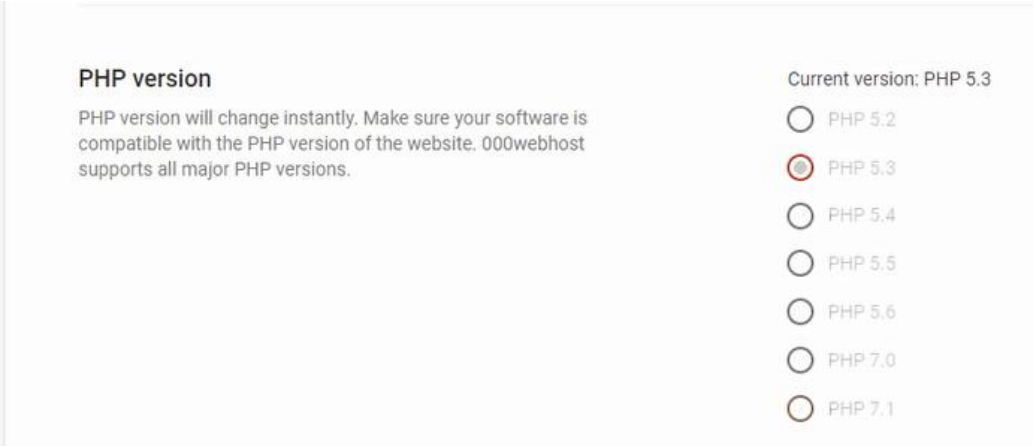


***Figura 110**:* Configuración de la versión

Luego, se ingresa a file manager y se suben los archivos PHP, y 1 archivo de ajax

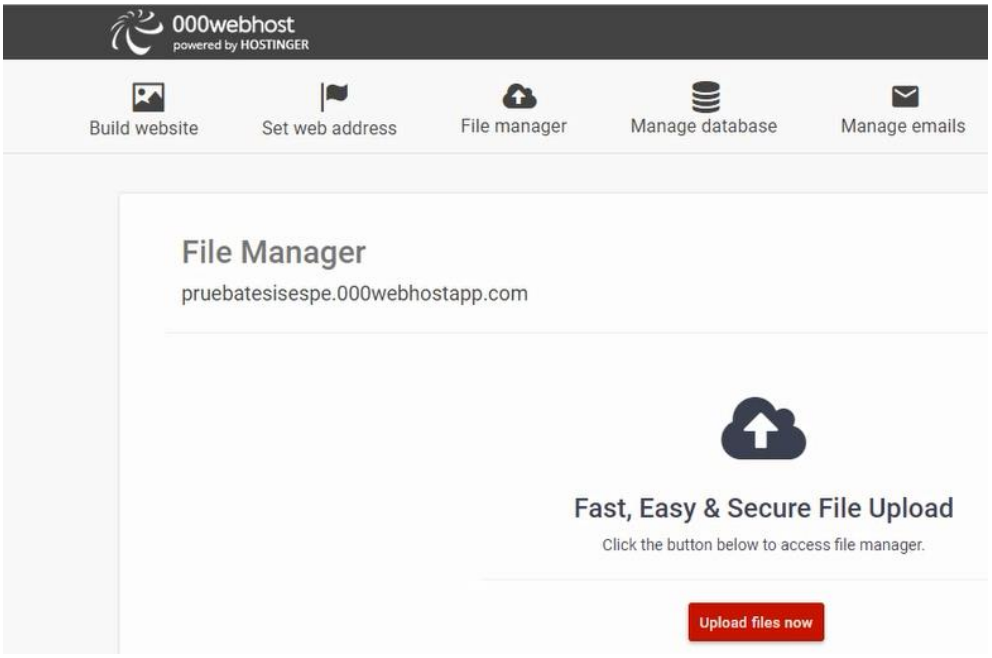


Figura 111: File Manager

Los archivos que se deben subir son los siguientes:

/public_html/php/config2.php

```php
<?php



$host_db = "localhost";
 $user_db = "id8550070_root";
 $pass_db = "123456789a";
 $db_name = "id8550070_prueba";



mysql_connect("$host_db", "$user_db", "$pass_db")or die("Cannot Connect to Data Base.");
    mysql_select_db("$db_name")or die("Cannot Select Data Base");

?>

```

Figura 112 Código config2.php

/public_html/php/comprobar.php

```php

<?php

require_once("config2.php");


$g=$_GET['q'];
$q=split('/', $g);

$sql = "update prueba set a0='$q[0]',a1='$q[1]',a2='$q[2]',a3='$q[3]'";
$res = mysql_query($sql);
if($res==1){
echo "DATOS ENVIADOS CON EXITO";
}
else{
    echo "ERROR AL ENVIAR EL DATO";
}

?>

```

Figura 113 Código comprobar.php

/public_html/php/miguel2.php

```php
<?php

require_once("config2.php");
$sql = "SELECT * FROM `prueba`";
$res = mysql_query($sql);
$row = mysql_fetch_array($res, MYSQL_ASSOC);
$a0 = $row['a0'];
$a1 = $row['a1'];
$a2 = $row['a2'];
$a3 = $row['a3'];


echo "@".$a0."@".$a1."@".$a2."@".$a3."@";



?>
```

Figura 114 Código miguel2.php

/public_html/php/miguel1.php

```
<script src="../js/ajax.js"></script>
<script type="text/javascript">


function cocina(){
  loadDoc("","miguel2.php",function(){
    if (xmlhttp.readyState==4 && xmlhttp.status==200){
        var a=xmlhttp.responseText;
 var b=a.split("@");


 for (var i=1;i<5;i++){
if(b[i]!="-9999"){
 document.getElementById("a"+i).innerHTML="Sensor"+i+" :"+b[i];
 }
 else{
 document.getElementById("a"+i).innerHTML="Sensor"+i+" : No conectado";

 }
 }

    }
});
}

setInterval(cocina,2000);

window.onload=cocina();

</script>

<center>

<h1 id="a1"></h1>
<h1 id="a2"></h1>
<h1 id="a3"></h1>
<h1 id="a4"></h1>
<center><div id="myDiv4"></div></center>


</center>
```

Figura 115 Código miguel1.php

/public_html/js/ajax.js

```
var xmlhttp;
function loadDoc(string,url,cfunc)
{
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=cfunc;
xmlhttp.open("POST",url,true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send(string);
}
```

Figura 116 Código config2.php

Una vez subidos los programas se configura el programa config2.pp

Figura 117: Archivos php

Se configura con los nuevos datos de la tabla.

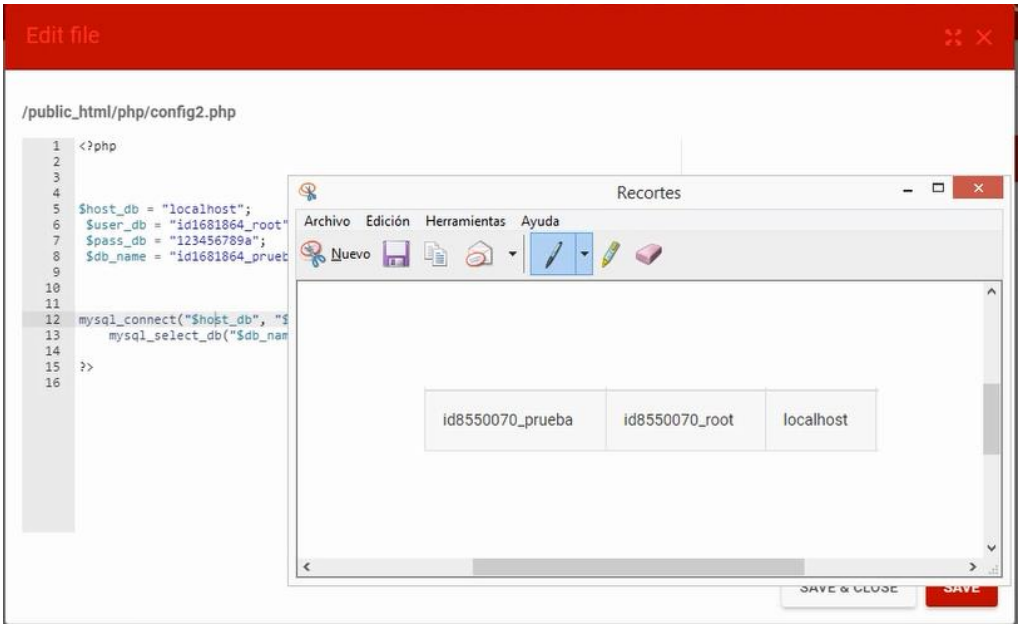


Figura 118: Configuración de config2.php

Como el programa no realiza ingresos a la tabla, lo que realiza es un update, se debe ingresar a la base de datos y colocar el siguiente código que se ve en la figura

***Figura 119**:* Configuración de base de datos

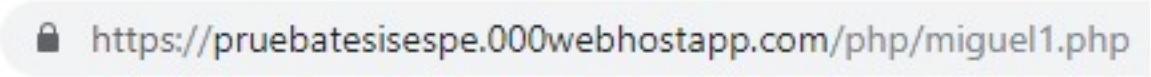


Figura 120 URL de la aplicación en IoT

4.6.1. RESULTADOS OBTENIDOS

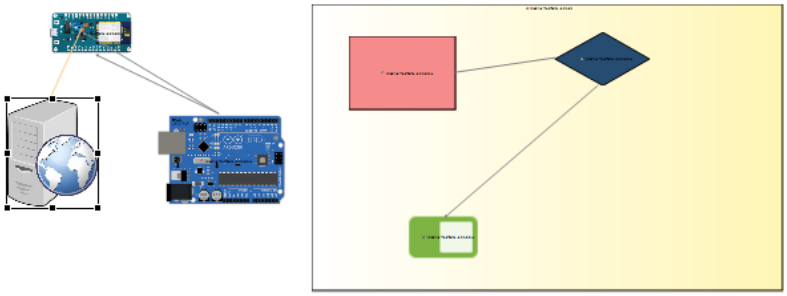


Figura 121 Diagrama para recibo de datos desde IoT

En la Figura 47 se puede visualizar el diagrama que da solución a la aplicación que se plantea desarrollar.

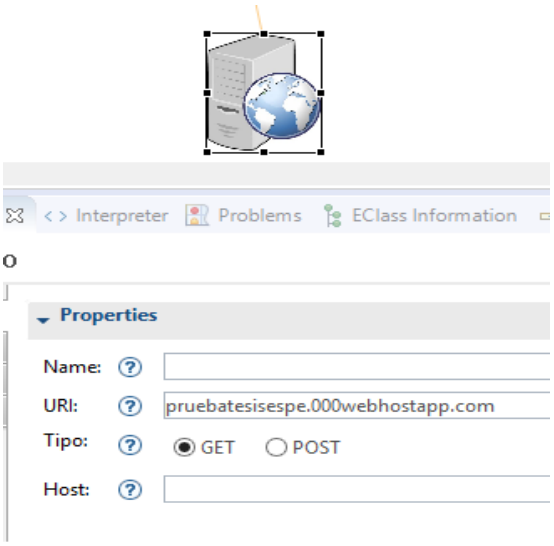


Figura 122 Atributos de componentes

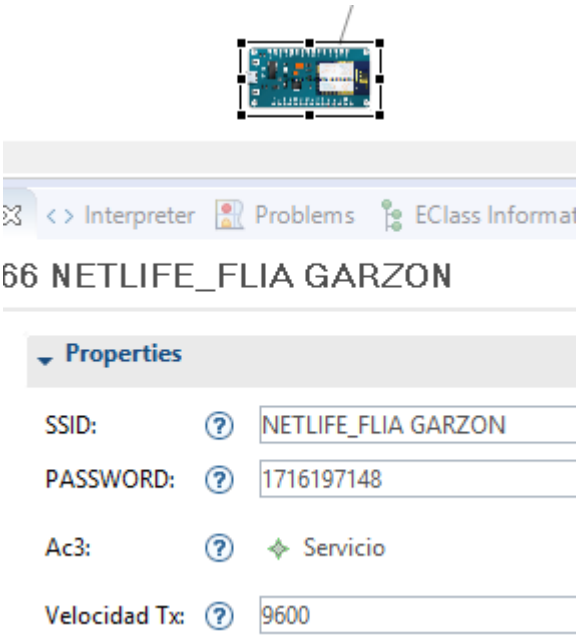


Figura 123 Atributos de componentes

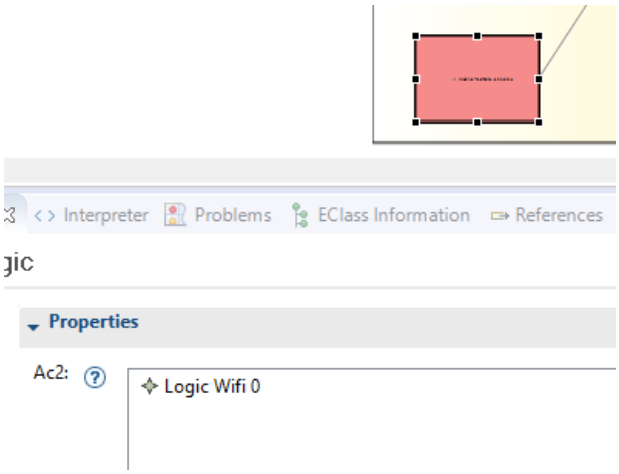


Figura 124 Atributos de componentes

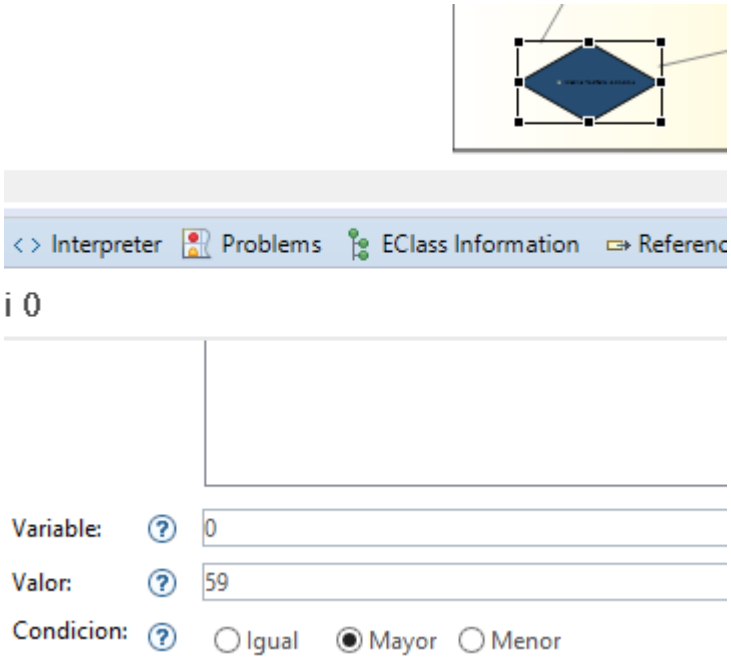


Figura 125 Atributos de componentes

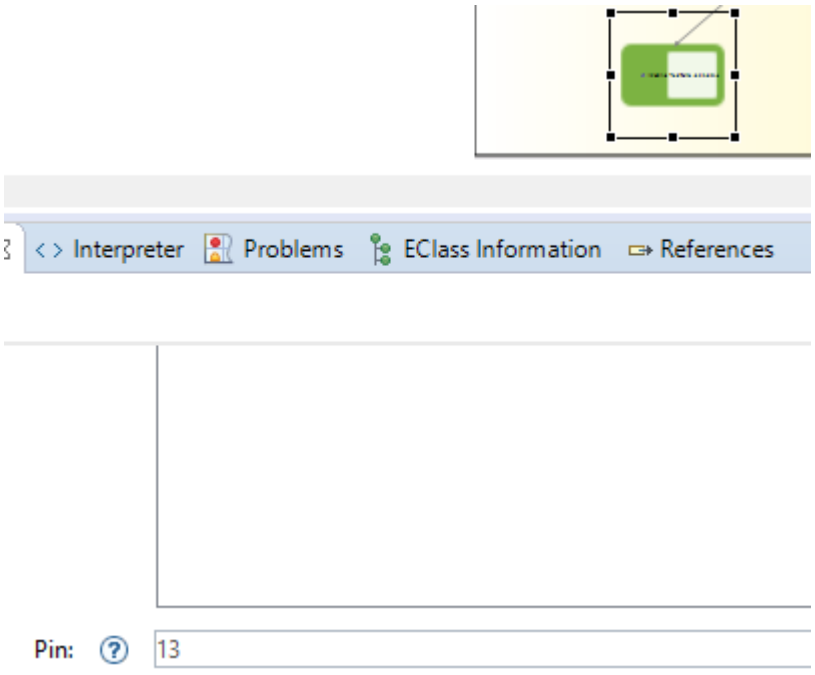


Figura 126 Atributos de componentes

Aquí se puede cómo deben ser llenados los atributos de los elementos conectados en el programa.

Una vez desarrollado el diagrama mostrado en el grafico Desarrollar una aplicación para recibo de datos desde el servidor mediante WiFi, se puede obtener el código fuente que se muestra a continuación. En el cual se puede observar la parte del código generado al colocar el elemento en el diseño del programa.

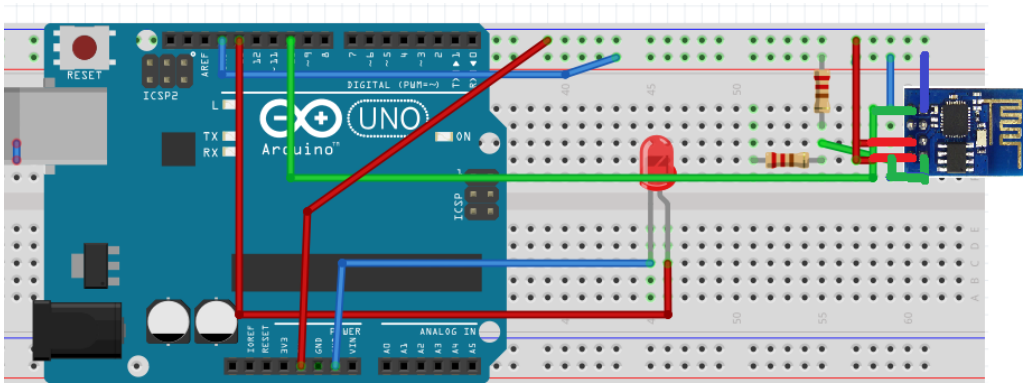


Figura 127 Conexiones hardware

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS

5.1. COCOMO II

COCOMO II es un modelo que considera diferentes enfoques para el desarrollo de software, como el de la construcción de prototipos, el desarrollo basado en componentes y el uso de programación con bases de datos. (Dixon, 2007)

Este modelo está compuesto por tres sub-modelos:

5.1.1. Composición de aplicación

Se utiliza cuando el sistema es creado mediante componente reutilizables, scripts y programación de bases de datos, fue diseñado para realizar estimaciones de desarrollo de prototipos. Este modelo utiliza como variable de medida del tamaño del producto los puntos objeto. (Dixon, 2007)

5.1.2. Diseño inicial

Este sub-modelo tiene en cuenta la exploración de diferentes arquitecturas del sistema y conceptos de operación. Se aplica en etapas tempranas del diseño del sistema, después que los requerimientos hayan sido establecidos. Utiliza como tamaño del producto las Líneas de Código, cuando están disponibles. (Dixon, 2007)

5.1.3. Post- Arquitectura

Este submodelo puede ser utilizado cuando se ha completado el diseño del sistema a desarrollar y se cuenta con información detallada. Utiliza puntos de función y/o líneas de código fuente como entrada para el parámetro del tamaño del producto. (Dixon, 2007)

Dentro del proyecto se utilizara la propuesta de Albrecth-Gaffney para el cálculo de puntos de función desajustados el cual tiene 5 características funcionales del sistema a desarrollar y cada una de ellas un tipo de complejidad con un peso asociado a está. . (Dixon, 2007)

5.2. Pruebas de Rendimiento

Para verificar la eficacia y rendimiento de la utilización de la herramienta del presente proyecto de titulación, en el desarrollo de aplicaciones por personas sin conocimientos de programación se realizaron las siguientes pruebas.

Para las pruebas de rendimiento se solicitó a un total de 4 personas sin mayores conocimientos de programación, pero con una idea básica de elementos electrónicos, que realicen 3 aplicaciones en la tarjeta Arduino, para cada aplicación se pidió realizar utilizando la herramienta desarrollada en el presente proyecto, y también que la aplicación se la desarrolle de forma tradicional, en el IDE de Arduino, las aplicaciones que se pidió realizar fueron:

- Encender y apagar un led
- Enviar un mensaje vía serial
- Subir datos a un servidor web mediante wifi

5.3. CALCULO DEL TAMAÑO DEL SISTEMA SEGÚN COCOMO II

El tamaño del sistema en este proyecto realizado ya se conoce en término de líneas de código. Para poder calcular el esfuerzo por medio de los modelos es indispensable el tamaño de los mismos, esto quiere decir un dato de entrada, así que se supone este tamaño desconocido y se calcula a través de puntos de función que se encuentran en el sistema. (Dixon, 2007)

Para realizar la tabla se tomaron como referencia los datos que se presentan en la siguiente tabla:

Lenguaje	LDC/UPF
Ada	71
A1 Shell	49
APL	32
Assembly	320
Assembly (macro)	213
ANSI / Quick / Turbo Basic	64
Basic - Compiled	91
Basic – Interpreted	128
C	128
C++	29
PHP	25
Visual Basic	32
ANSI Cobol 85	91
Fortran 77	105
Fort	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91
Prolog	64
Report Generator	80
SpreadSheet	6
Java	23

Tabla 11. Conversión de los puntos de función a líneas de código.

Figura 128: Conversión de los puntos de función a líneas de código

Autor: (Dixon, 2007)

Tabla 6

Tamaño del sistema (Encender y apagar un led)

Puntos de función sin ajustar				
Descripción	Complejidad			Total
	Simple	Media	Compleja	
Entradas externas	-	-	-	0
Salidas externas	1 x 4	-	-	4
Ficheros lógicos internos	-	-	-	0
Interfaces lógicas externas	-	-	-	0
Consultas externas	-	-	-	0
Total de puntos de función sin ajustar				4

Fuente: (Dixon, 2007)

Tamaño estimado = 4 x 23 = **92 líneas de código**

Tabla 7

Tamaño del sistema (Enviar un mensaje vía serial)

Puntos de función sin ajustar				
Descripción	Complejidad			Total
	Simple	Media	Compleja	
Entradas externas	-	1 x 4	-	4
Salidas externas	1 x 4	-	-	4
Ficheros lógicos internos	-	-	-	0
Interfaces lógicas externas	1 x 5	-	-	5
Consultas externas	-	-	-	0
Total de puntos de función sin ajustar				13

Fuente: (Dixon, 2007)

Tamaño estimado = 13 x 23 = **299 líneas de código**

Tabla 8

Tamaño del sistema (Subir un dato a un servidor web mediante WiFi)

Puntos de función sin ajustar				
Descripción	Complejidad			Total
	Simple	Media	Compleja	
Entradas externas	-	1 x 4	-	4
Salidas externas	-	-	-	0
Ficheros lógicos internos	1 x7	-	-	7
Interfaces lógicas externas	-	1 x7	-	7
Consultas externas	-	1 x 4	-	4
Total de puntos de función sin ajustar				22

Fuente: (Dixon, 2007)

Tamaño estimado = 22 x 23 = **506 líneas de código**

Tabla 9

Pruebas de rendimiento

Persona	Aplicación	Utilizando la herramienta		Programación Tradicional		Tamaño estimado según cocomo II
		Cumplimiento	**Líneas de código**	**Cumplimiento**	**Líneas de código**	
1	Encender y apagar un led	X	10	X	13	92
1	Enviar un mensaje vía serial	X	13	X	18	299
1	Subir datos a un servidor web mediante wifi	X	200	X	287	506

CONTINUA

2	Encender y apagar un led	X	10	X	15	92
2	Enviar un mensaje vía serial	X	15	X	21	299
2	Subir datos a un servidor web mediante wifi	X	200			506
3	Encender y apagar un led	X	10	X	8	92
3	Enviar un mensaje vía serial	X	15	X	13	299
3	Subir datos a un servidor web mediante wifi	X	200	X	256	506
4	Encender y apagar un led	X	10	X	14	92
4	Enviar un mensaje vía serial	X	15	X	16	299
4	Subir datos a un servidor web mediante wifi	X	200			506

5.4. Resultados

Según los resultados obtenidos, lo que se evidencia es que mientras la complejidad de los programas es menor, la cantidad de código que se genera no es igual según Cocomo II. Pero cuando la dificultad de la aplicación que se quiere realizar es mayor, la cantidad de líneas de código entre la generación de código, en el IDE de Arduino y según Cocomo II los datos cada vez se van pareciendo más, por lo tanto se puede llegar a decir que Cocomo II ayuda a ver la cantidad de líneas de código para aplicaciones o programas que ya tienen un nivel de dificultad mayor.

CAPITULO 6

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- La necesidad de los usuarios actualmente de crear aplicaciones con información procedente del internet para realizar acciones en base a estos datos es muy grande. En base a esta problemática se desarrolló un DSL y transformación M2T el cual a partir de obtener datos del Internet tenga una secuencia en sus elementos y de esta forma ayudar a los usuarios a crear aplicaciones de una forma gráfica y rápida.
- En base a la información adquirida sobre Ingeniería de Modelos y Scaffolding se evidenció que en ningún proyecto utilizan de manera conjunta estas dos herramientas para solventar problemas. En este proyecto se logró integrar tanto ingeniería de modelos como el aprendizaje escalonado, con el fin de que el usuario se familiarice más rápido con el programa.
- El meta modelo separa la parte de software de la de hardware, esto permitió desarrollar una herramienta modular para que sea escalable en el tiempo en la cual se puedan incluir nuevos elementos con sus respectivos atributos en la parte de hardware, y sentencias en la parte de software, introduciéndolas ya sea en clases padre o en diferentes condiciones establecidas en el diagrama UML, sin la necesidad de un cambio significativo en el DSL.
- En el Editor Gráfico se separó por capas la parte de hardware y software para que así el usuario pueda tener una mejor idea de cómo armar el proyecto sin que la parte de software interfiera, además se creó contenedores en el cual solo la parte de software pueda ser

graficada dentro del contenedor, con el fin de que el usuario no pueda mezclar el hardware del software y de esta manera el usuario no tenga problemas en la compilación del código.

- Para aumentar la cantidad de las posibles aplicaciones y algoritmos creados utilizando la herramienta presentada en este proyecto de titulación, se tomó en cuenta todas las posibles combinaciones de los elementos y sus posibles interacciones, logrando de esta manera una plantilla general en Acceleo la cual cuenta con códigos que sirven para tener secuencia lógica y las adquisiciones de datos del Editor Gráfico las cuales sirven para que el código que se va a generar tenga los datos que el usuario necesita para la aplicación creada, lo cual conlleva a la obtención de una herramienta robusta y funcional en todo sentido.
- En la recolección de información se pudo investigar y añadir una arquitectura de modelos que permite la construcción de software para placas de desarrollo, en la creación de aplicaciones orientado a las IoT, en este caso la validación fue realizado en la placa de desarrollo Arduino.
- En el proceso de pruebas se pudo identificar que el uso de la arquitectura en la generación del código que se ejecuta en la placa de desarrollo Arduino junto con el módulo de la IoT, hace posible cumplir el correcto funcionamiento del código.

6.2. Recomendaciones

- Se recomienda agregar el DSL al proyecto creado en viewpoint para crear el Editor Gráfico ya que al momento de diseñar la aplicación puede generar errores o bloquear la herramienta.
- Tomando como referencia artículos presentados en el estado del arte con arquitectura de modelos es recomendable utilizar herencias, mediante clases padre, también es recomendable utilizar clases abstractas todo esto para que este proyecto pueda ser escalable es decir que se pueda agregar o eliminar elementos dependiendo de los requerimientos del usuario.

- Como se muestra en la gran mayoría de proyectos con la herramienta EMF hay que tener precaución al momento de escribir el código en Acceleo ya que si se crea mal un objeto o se llama mal a una variable esto repercutirá directamente en el código generado y por ende la aplicación no funcionara correctamente.

6.3. Trabajos futuros

- Al no tener una carga directa con la placa de desarrollo sería un buen trabajo futuro el de implementar la interconectividad con el compilador de Arduino de forma directa ya que de esta manera se podría compilar y cargar el programa desde el mismo proyecto generado.
- Al querer tener la domótica de un lugar específico, el número de entradas para los sensores no bastara con las que posee la placa de desarrollo utilizada en este proyecto, por lo que se plantea un trabajo a futuro de conectar dos placas de desarrollo con el fin de tener más entradas y salidas para los sensores y actuadores respectivamente.
- Al querer estar más conectados al mundo del Internet es necesario tener acceso a más servicios web con el fin de recibir datos y poder crear aplicación es por eso que se plantea en un trabajo a futuro la creación de servicios web
- Al querer realizar aplicaciones más desarrolladas y avanzadas se propone en un trabajo a futuro incorporar funciones más complejas para la lógica de programación como sentencias while, switch, case, else if, etc.

Referencias

Hernández, R., Fernández, C., & Baptista, P. (2016). *Metodología de la investigación* (Sexta ed.). CDMX: McGraw Hill.

Cegarra, J. (2004). *Metodología de la investigación científica y tecnológica* . Madrid: Díaz de Santos .

Benítez, A. (2013). *Construcción de un Lenguaje Específico de Dominio para el Desarrollo de Interfaces de Usuario Multiplataforma en Aplicaciones para Dispositivos Móviles*.

Luzza, M., Berón, M., Peralta, M., & Montejano, G. (2012). Diseño y Construcción de Lenguajes Específicos del dominio. *XIV Workshop de Investigadores en Ciencias de la Computación* (14).

Pineda, B., & Hernán , J. (2015). *Análisis, diseño e implementación de un prototipo de un lenguaje de dominio específico (dsl) interno, orientado al modelado de problemas de programación lineal*.

Peña, J., & Suquillo, G. (2016). *Estudio del modelo de referencia del internet de las cosas (iot), con la implementación de un prototipo domótico*.

Gómez, A. (2011). *Implementación de un lenguaje de definición de operaciones complejas en Gestión de Modelos para la herramienta MOMENT*.

Cañete, L. (2014). Arquitectura de software. *Tino* (41).

García, L. (2014). *Estudio del impacto técnico y económico de la transición de internet al internet de las cosas (iot) para el caso colombiano*.

Rucci, M. (2015). Servidores WEB.

Filippi, S. (2009). *¿Cómo funciona exactamente un servidor Web?* Obtenido de Internetlab: https://www.internetlab.es/post/628/como-funciona-exactamente-un-servidor-web/

Granada, S. (2014). *Diagrama de funcionamiento servidor web*. Obtenido de kreativo: https://sebastiangranada.wordpress.com/2014/11/13/diagrama-de-funcionamiento-servidor-web/

Aguirre, A., Fernández, P., & Grossy, C. (2007). *Interfaz USB genérica para comunicación con dispositivos electrónicos*.

Quitiaquez, H., & De La Torre, J. (2014). *Analisis y diseño de una red WiFi y WiMax para el centro de investigación científicaen telecomunicacions tegnológicas de la información y las comunicaciones- CITIC*.

Jara, P., & Nazar, P. (2016). *Estándar IEEE 802.11 X de las WLAN*. Tucumán: Universidad tecnológica Nacional.

Morales, R., & Gonzáles, J. (2013). *Control de tráfico vehicular por medio de semaforos inteligentes*.

Rodriguez, E. (2018). *Qué comprar y leer para empezar con Arduino*. Obtenido de xataka: https://www.xataka.com/ecommerce/que-comprar-leer-para-empezar-arduino

Piña, J., & Zúñiga , G. (2017). *Análisis comparativo del sistema tradicional de semaforización vs una propuesta de semaforización inteligente, para la reducción del congestionamiento vehicular, en la ciudad de guayaquil.*

Yúbal, F. (2018). *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. Obtenido de xataka: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

Hernández, L. (2018). *ESP8266 todo lo que necesitas saber del módulo WiFi para Arduino*. Obtenido de programarfacil: https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

Saavedra, Y. (2014). *El lenguaje dealgoritmos je con soporte para eclipse*.

Castillo, C., & Ortiz, F. (2017). *Editor de diagramas para modelos de características con atributos y cardinalidades (EDC).*

Mayorga, J. (2015). *Estudio de la herramienta acceleo y sus beneficios respecto a la programación tradicional*.

AG Electrónica. (2017). kit-sen-act: Kit De Sensores Y Actuadores Compatibles Con Ardu. *¿Que vamos a innovar hoy?*

Hincapié, E. (2016). *Asistente web para la generación y personalización de crudel en el framework php core*.

Guía del Arduinomaniaco. (2015). Obtenido de xataka: https://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino

Noguera, J. (2016). *Sistema de diálogo basado en mensajería instantánea para el control de dispositivos en el internet de las cosas*.

Domenech, F., Jiménez, R., & Jiménez , J. (2009). *Un entorno de modelado y desarrollo para sistemas sociales*.

Garcia , A. (2015). *Desarrollo de DSL visuales con Sirius*. Obtenido de https://legacy.gitbook.com/book/galba/desarrollo-de-dsl-visuales-con-sirius/details

Nieves R. Brisaboa, A. C. (2016). *Aplicando scaffolding en el desarrollo de Lineas de Producto Software.* Obtenido de http://lbd.udc.es/Repository/Publications/Drafts/1472199879813_CEDI_2016_paper_80.pdf

Montenegro Marin, C. E., Gaona Garcia, P. A., Cueva Lovelle, J. M., & Martinez, O. (2011). *Aplicación de ingeniería dirigida por modelos (mda), para la construcción de una herramienta de modelado de dominio específico (dsm) y la creación de módulos en sistemas de gestión de aprendizaje (lms) independientes de la plataforma*. Obtenido de file:///Users/miguelgarzon/Downloads/20390-91550-1-PB.pdf

García, M., Ruiz, M., & Vicente Chicote, C. (2016). *Cómo usar MDE para obtener Modelos de Simulación a partir de Modelos de Negocio.* Universidad de Cádiz, Departamento de Lenguajes y Sistemas Informáticos.

Moreno, A. T. (2013). *Ingeniería dirigida a modelos. sistemas de transformación modelo a texto. complemento de refactorización de código c++ a c++11 para elcipse*. Tesis, Universidad carlos iii de madrid, Departamento de Informatica, Madrid.

Cobo, A., Gomez , P., Perez , D., & Rocha, R. (2005). *PHP y MySQL Tecnologías para el desarrollo de aplicaciones web.* España: Diaz de santos.

Bazán, P. (2007). *AJAX: un análisis tecnológico y posibilidades metodológicas.* Facultad de Informática – UNLP Universidad Nacional de La Plata , Laboratorio de Investigación en Nuevas Tecnologías Informáticas.

Marini, I. E. (2012). *El Modelo Cliente/Servidor*. (I. E. Marini, Productor) Obtenido de https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf

Stephen, J. M. (2004). *Distilled: Principles of Model-Driven Architecture* . Addison Wesley.

Garcia Diaz, V., & Cueva Lovelle, J. (2010). *Ingenieria dirigida por modelos.* Ed. Oviedo.

Garcia, & Y otros. (2012). *Desarrollo de Software Dirigido por Modelos Conceptos, Métodos y Herramientas.* España: RA-MA.

Wimmer, M., & Burgueño, L. (2013). *Testing M2T/T2M Transformations.* Universidad de Malaga. Spain: Gisum/atenea Research Group.

Marco, B., Cabot , J., & Wimmer, M. (2012). *Model-to-model transformations.* usa: Morgan & Claypool.

Molina. (2019). *Sensores y traductores*. Obtenido de Sensores: http://www.profesormolina.com.ar/tecnologia/sens_transduct/index.htm

Ingenieria de sistemas y automatica . (2016). *ACTUADORES*. Obtenido de http://isa.uniovi.es/docencia/AutomEdificios/transparencias/actuadores.pdf

Dixon, V. (2007). *Estimacion de esfuerzo y costo en la produccion de software hecho en venezuela.* Universidad de los Andes.