# UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

## Facultad de Ingeniería Civil

## Programa de Maestría en Ingeniería de los Recursos Hídricos

## "Sistema Automático de Información Hidrológica de Morelia"

# TESIS

Que para obtener el grado académico de maestro en ingeniería de los recursos hídricos


Presenta:
**Daniel Rodriguez Licea**
**Ingeniero Civil**

Director de Tesis:
**Sonia Tatiana Sánchez Quispe**
**Doctora en Ingeniería de Caminos, Canales y Puertos**

Codirector de Tesis:
**Francisco Javier Domínguez Mota**
**Doctor en Ciencias Matemáticas**



Morelia, Michoacán, México noviembre 2020

# Índice

# Índice de figuras

## Lista de abreviaturas

SAIH: Sistema Automático de Información Hidrológica.
EMABC: Estación meteorológica automática de bajo costo.
App: Aplicación móvil.
IoT: Internet de las Cosas (del inglés Internet of Things).
HTTP: Protocolo de transferencia de hipertextos (del inglés Hypertext Transfer Protocol).
SMS: Mensaje corto de texto (del inglés Short Message Servise).
GSM: Sistema global para las comunicaciones móviles (del inglés Global System for Mobile comunications).
GPRS: Servicio general de paquetes vía radio (del inglés General Packet Radio Service).

## Resumen

En países en desarrollo los altos costos de dispositivos de monitoreo de variables meteorológicas afectan negativamente a la gestión de riesgos ambientales (sequias, avenidas, cambio climático) debido a la baja densidad de ubicación de estaciones meteorológicas, así como intervalos de tiempo inadecuados en la adquisición de los datos, unido a una actualización tardía de las bases de datos en los que se incorporan. El clima es muy susceptible a cambios, por ello el interés de generar información suficiente en cantidad, calidad y en tiempo real. Como solución a este problema, en esta tesis se propone desarrollar un Sistema Automático de Información Hidrológica SAIH, constituido por una red de monitoreo formada por estaciones meteorológicas construidas con microcontroladores y sensores de bajo costo a través de Arduino, un servicio de base de datos en la nube que permite la administración, manipulación y análisis de datos en gran escala enviados por las estaciones meteorológicas, un Sistema que permita no solamente adquirir datos, sino también generar alertas a través de SMS, Twitter y Facebook (cuando las variables estén fuera del rango de tolerancia), una plataforma web y un APP que le permite al usuario la visualización de la información espacial y temporalmente y la descarga de las bases de datos de precipitación, temperatura y humedad de cada estación. Para ser confiables, los sensores de las estaciones meteorológicas que recogen los datos deben pasar por un proceso de calibración a través de climas controlados en laboratorio y por comparación de observaciones con estación cercana, posteriormente validados bajo las normas de la Organización Meteorológica Mundial y la Norma Mexicana NMX-AA-166/1-SCFI-2013. El SAIH propuesto debe entregar información en cantidad y calidad, y formará parte de los sistemas de alerta temprana para la toma de decisiones y generación de acciones necesaria para la gestión de riesgos ambientales.

**Palabras clave**: SAIH, tiempo real, bajo costo, arduino

## Abstract

In developing countries, the high costs of monitoring devices of meteorological variables negatively affect the management of environmental risks (droughts, avenues, climate change) due to the low density in the location of the meteorological stations and wide time intervals of the data, together with a late update of the databases. The climate is very susceptible to changes, so, given the interest of generating sufficient information in quantity, quality and real time, as a solution to this problem, in this thesis the development of an Automatic Hydrological Information System SAIH is proposed. The system will be constituted by a monitoring network formed by weather stations, built with low-cost microcontrollers and sensors through Arduino, as well as a cloud database service that allows the administration, manipulation and analysis of large-scale data sent by weather stations, as well as generating alerts through SMS, Twitter and Facebook (when the variables are outside the tolerance range), a web platform and an APP that allows the user to view spatial and temporal information and download the precipitation, temperature and humidity databases of each station . The sensors of the meteorological stations that collect the data have undergone through a calibration process through controlled climates in the laboratory and by comparison of observations with a nearby station, and subsequently validated under the standards of the World Meteorological Organization and the Mexican Standard NMX-AA -166 / 1-SCFI-2013. The SAIH delivers information in quantity and quality, is part of the early warning systems for decision making and generation of actions necessary for environmental risk management.

**Keywords:** SAIH, real time, low cost, arduino

# 1. Introducción

La sociedad siempre ha prestado especial atención al estudio del clima debido a que es un factor muy importante para las personas y sus actividades cotidianas, mismas que pueden verse afectadas o favorecidas dependiendo de las condiciones climáticas.

Generalmente el objetivo es tratar de "predecir" el comportamiento del clima, a través de la medición de parámetros que generan información sobre las condiciones meteorológicas de un lugar, además, con la ayuda de registros pasados se puede hacer un estimado de lo que podría suceder en las siguientes horas y así prever cualquier evento que pueda presentarse. Sin embargo, el clima puede ser afectado por muchos factores y cambiar repentinamente, de ahí la dificultad de generar un pronóstico preciso (Escuela Politécnica Nacional (Quito et al., 2017).

El clima y el tiempo atmosférico afectan en forma directa el entorno del ser humano, ya que intervienen en diversos aspectos tales como biológicos, sociales y económicos. Por un lado, el clima permite conocer el promedio a largo plazo de los elementos del tiempo atmosférico, y este último por su parte, permite conocer el comportamiento atmosférico a corto plazo (uno o varios días). De ahí la importancia de su estudio para contar con información representativa que pueda ayudar en el desarrollo de diversas actividades tales como: desarrollo urbano, agricultura, ganadería, transporte, salud, esparcimiento, prevención de desastres, pronóstico del tiempo, calentamiento global y cambio climático, entre otros (Instituto Nacional de Investigaciones Forestales et al., 2010).

Algunas paradojas importantes de los actuales sistemas económicos en torno a los problemas medioambientales y la desigualdad social son, por ejemplo, la creciente necesidad de hacer frente a los requerimientos del mercado (mayores bienes/servicios) y la presión que ello genera al medio ambiente, así como la ampliación de brechas sociales y tecnológicas, por mencionar algunas; de ahí la necesidad de buscar soluciones innovadoras que hagan compatible los intereses entre la esfera social, económica y ambiental (Alvarado López & Alvarado López, 2018).

Los procesos de innovación y cambio técnico han mostrado, históricamente y en lo general, un comportamiento excluyente y se han localizado en gran medida en algunas regiones,

países o sectores productivos. Sin embargo, es importante que estos beneficios se propaguen hacia países y sectores de la población menos favorecidas con el fin de garantizar una mayor igualdad social, un medio ambiente sano, acceso a la educación y salud de calidad.

Últimamente se han realizado muchos avances en el campo de la meteorología, contando con equipos sofisticados que recogen datos exactos de un lugar y permiten conocer el comportamiento del clima de una manera más acertada. Sin embargo, los equipos que se utilizan generalmente son costosos y se encuentran en estaciones meteorológicas fijas de instituciones dedicadas al estudio del clima.

Se puede definir una estación meteorológica como una instalación en la que se tienen una serie de instrumentos destinados a la recolección y registro de las variables meteorológicas según su tipo, ya sean estas, climáticas, sinópticas o marinas ("World Meteorological Organization Extranet | www.wmo.int," n.d.).

Una de las preguntas más fundamentales en meteorología está relacionada con el hecho de cómo el cambio tecnológico puede afectar los registros de datos y la comparación entre los sistemas tradicionales y los actuales. La estaciones automáticas permiten el acceso a la información meteorológica en tiempo real de lugares alejados y de difícil acceso, este procedimiento es imposible de realizar mediante la utilización de estaciones convencionales a menos que sean atendidas por un observador en forma permanente y que además el observados cuente con la tecnología que le permita trasmitir la información a un centro de acopio de información, que es este caso sería un sistema automático de información geográfica (SAIG). (Rica, Estatal, & Rica, 2011)

Un sistema automático de información hidrológica (SAIH) se sustenta de un SAIG el cual permite la lectura, edición, almacenamiento, gestión de datos espaciales, análisis de dichos datos (esto puede incluir desde consultas sencillas a la elaboración de modelos complejos, y puede llevarse a cabo tanto sobre la componente espacial de los datos como sobre la componente temática) y la generación de resultados tales como mapas, informes, gráficos, etc., en tiempo real. La diferencia se encuentra en el área de conocimiento: el SAIH está directamente enfocado al recurso hídrico.

## 2. Antecedentes

En la literatura se han encontrado diversos estudios e investigaciones que conllevan el desarrollo de sistemas de información a través del uso de microcontroladores y sensores de bajo costo para la creación de dispositivos de medición de variables meteorológicas, la implementación de servicios de bases de datos en la nube y la visualización de la información en App's y plataformas web. Estos sistemas incorporan en el estudio los conceptos de tiempo real, bajo costo, alerta temprana, IoT (Internet of Things) y Big Data los cuales relacionan directamente al proyecto que se desarrolla en esta tesis.

(Imtiaz, Omar, & Ali, 2018) implementan una estación basada en Arduino con el uso de sensores de bajo costo de temperatura, humedad, presión atmosférica y velocidad del viento. La transmisión de información se hace mediante un protocolo HTTP (Hyper Text Transfer Protocol) a través de un shield de Ethernet para Arduino para establecer la conexión con la base de datos de MySQL (My Structured Query Language), a partir de esta se diseña un sitio web que permite al usuario visualizar y descargar los datos históricos y en tiempo real.

(Srivastava, Singh, & Singh, 2018) desarrollan un sistema de monitoreo del aire en tiempo real basado en IoT con la ayuda de Arduino, este se forma a través de estaciones de monitoreo de aire que envían información de temperatura, humedad, gas ozono, presión, gas CO, gas $NO_2$, presencia de polvo, gas $CO_2$, gas metano y ruido a un canal en ThingSpeak a través de un modulo de WiFi (Wireless Fidelity IEEE 802.11x). El canal de ThingSpeak permite la visualización y descarga de la información temporalmente accediendo a través de la red de internet desde un ordenador o teléfono inteligente.

(Muñoz, Yumang, Japitana, Medina, & Tibayan, 2018) diseñan, prueban e implementan un dispositivo de adquisición de datos meteorológicos en tiempo real basado en Arduino que mide temperatura, humedad relativa y la precipitación. El dispositivo transmite la información a una base de datos a través de GPRS 2G (General Packet Radio Service, 2 generation) posteriormente se enlazan a un sitio web que permite la visualización temporal de la información y la descarga de esta. Los sensores que se han utilizado para la adquisición

de datos se someten a un proceso de calibración y validación con climas controlados en laboratorio, obteniendo como resultado la efectividad del sistema hasta un 93.36%.

(Netto & Arigony-Neto, 2019) describen el diseño y la implementación de una estación meteorológica automática de bajo costo desarrollada a través de Arduino, es un equipo electromecánico utilizado para recopilar, almacenar y transmitir datos meteorológicos de una región determinada de forma autónoma. Este dispositivo registra datos de presión atmosférica, temperatura del aire, precipitación liquida, humedad, radiación solar, velocidad y dirección del viento. Almacena los datos en una tarjeta SD y en algunos casos es capaz de enviar los datos a servidores remotos.

(Strigaro, Cannata, & Antonovic, 2019) desarrollan una estación meteorológica de bajo costo basada en Arduino la cual transmite datos de precipitación, temperatura, humedad del aire y suelo, velocidad y dirección del viento e intensidad de luz solar a través de GPRS 2G a una base de datos OSGEO (Open Source Geospatial Foundation). Evalúan cuantitativamente la calidad de los datos, mediante la comparación de observaciones de la estación meteorológica de bajo costo con una estación comercial cercana de marca Trevano, obteniendo como resultado la fiabilidad del proyecto de tal manera que los países en desarrollo establezcan sistemas de monitoreo con tecnología a bajo costo.

(Haque, MD. Shatil, Tusar, Hossain, & Rahman, 2019) desarrollan una estación meteorológica autosustentable energéticamente basada en Arduino, devuelve datos de temperatura, humedad, gotas de lluvia, monóxido de carbono, humo, GLP en el medio ambiente y presión barométrica a través de SMS por medio de un modulo GSM que transmite los datos a los usuarios por medio del protocolo de comunicación GPRS 2G.

(Farid, Prawito, Susila, & Yuniarto, 2017) desarrolla un sistema de alerta temprana de preparación nuclear basado en Arduino, obteniendo mediciones de radiación ambiental que se transmiten por GPRS 2G a una base de datos y visualizados en un sitio web.

## 3. Justificación

Los SAIH, al recoger datos automáticamente, y proporcionar información en tiempo real, generan alertas tempranas para prevención de riesgos, ahorran tiempo de captura y transcripción de datos, reducen errores de digitación y normalizan la representación de los datos.

## 4. Planteamiento del problema

En los países en desarrollo los altos costos de implementación de sistemas automáticos de información debido a los equipos de medición de variables meteorológicas tales como las estaciones meteorológicas automáticas conlleva a que la densidad de estaciones para el monitoreo del clima sea baja, lo cual obliga al uso de estaciones meteorológicas convencionales donde la recolección de información requiere de recurso humano y tecnológico desde la medición hasta el respaldo de la misma en bases de datos oficiales, de esta forma la actualización de las bases de datos es discontinua y tardía (años). Al no conocer el comportamiento del clima históricamente como en tiempo real, afectan negativamente a la gestión de riesgos ambientales (sequias, avenidas, cambio climático).

## 5. Hipótesis

Empleando tecnologías de última generación y bajo costo, es posible desarrollar un Sistema Automático de Información Hidrológica de bajo costo que devuelva datos de precipitación, temperatura y humedad relativa en cantidad y calidad, que será capaz de emitir alertas tempranas para la prevención de riesgos ambientales.

## 6. Objetivo

### 6.1 Objetivo General

Crear un Sistema Automático de Información Hidrológica (SAIH) de bajo costo capaz de medir y soportar la captura, administración, manipulación, análisis, y visualización temporal y espacial de datos de precipitación, temperatura y humedad relativa en tiempo real, constituyendo un sistema de alerta temprana para la prevención de riesgos ambientales ((sequias, avenidas, cambio climático).

### 6.2 Objetivos Específicos

- Crear un dispositivo de medición de precipitación, temperatura y humedad relativa con el uso de tecnologías de bajo costo (estación meteorológica automática de bajo costo EMABC), que envíe automáticamente los datos medidos a la base de datos del SAIH.
- Implementar un servicio de base de datos que soporte la captura, administración, manipulación y análisis de datos en gran escala, además de permitir la generación y envío de alertas tempranas para la prevención de riesgos ambientales.
- Diseñar una interfaz web y un App que permita la visualización temporal y espacial y la descarga de datos.
- Construir una red de monitoreo replicando e instalando las EMABC.

## 7. Materiales y Métodos

### 7.1 Materiales

En este apartado se describen los materiales y métodos utilizados para el desarrollo del proyecto desde la recolección de datos hasta la entrega de estos al usuario.

Los materiales se han clasificado en cuatro grupos, el primero, instrumentación: en este grupo se describen los microcontroladores, sensores y sistemas requeridos para la construcción de las EMABC. El segundo, programación: en este grupo se describen los softwares para el desarrollo de algoritmos utilizados para el funcionamiento de las EMABC. Tercer grupo, almacenamiento de datos y vinculación: en este grupo se describen los servicios de bases de datos y de vinculación con las plataformas de visualización y de publicación de alertas tempranas. Cuarto y ultimo grupo, visualización: en este grupo se describen las herramientas para el desarrollo de las plataformas de visualización de información y las plataformas utilizadas para la publicación de alertas tempranas.

Los métodos se dividen en dos clases: primarios y secundarios. Dentro de los primarios se explica la metodología del ciclo de la vida de desarrollo de sistemas, siendo esta la metodología principal del desarrollo del proyecto. Esta metodología se divide en fases, siete

para ser exactos, dentro de las cuales se requieren métodos secundarios complementarios para la culminación de estas. En el apartado 7.2 se explican a detalle.

### 7.1.1 Instrumentación

7.1.1.1 Arduino UNO

"Arduino es una plataforma *open hardware* microcontrolada para el desarrollo de productos y proyectos electrónicos. Está basado en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Su hardware es ampliable mediante módulos y además es de código abierto. Arduino está basado en los microcontroladores ATMEGA168, ATMEGA328 y ATMEGA1280. Dispone de varios pines que pueden configurarse como entrada o salida y a los que pueden conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de 0 y 5V. También dispone de entradas y salidas analógicas, para obtener datos de sensores o enviar señales de control PWM a otros dispositivos." (Enciso & Vargas, 2018) Arduino uno se basa en el microcontrolador ATMEGA328 (Singh, Bhojwani, Reddy, & Sultana, 2019).

7.1.1.2 NodeMCU

NodeMCU es una placa de desarrollo *open source* con un módulo ESP8266 integrado, el beneficio de esta placa de desarrollo es tiene un módulo WiFi que conecta el sistema a internet (Negara, Tulloh, Nandy Hadiansyah, & Zahra, 2019), cuenta con pines digitales y analógicos para la conexión de diversos módulos y sensores (Janpla & Jewpanich, 2019), tiene un procesador SoC y 16 líneas de GPIO y se alimenta con 5 V (Arun Raj, Nambi Srinivasan, & Vimalathethan, 2019).

7.1.1.3 Módulo GSM SIM900

GSM SIM900 es un módulo de entrada y salida de datos para Arduino UNO, se basa en el protocolo de comunicación GSM (Global System for Mobile Communications) (Satria, Yana, Yusibani, Syahreza, & Zulfan, 2019), se conecta a la placa Arduino UNO por un circuito RS22 (Salahuddin, Basyir, & Tusannalaila, 2019), la alimentación de energía se

transfiere a través del Arduino por medio de pines de tierra y corriente (5 V) o por alimentación directa en el puerto del módulo.

7.1.1.4 DHT22

“El DHT22 es una unidad de sensor avanzada que proporciona una salida de señal digital calibrada, costa de dos partes: un sensor de temperatura con termistor, y un sensor de humedad capacitivo. Esta equipado con un microcontrolador de 8 bits y tiene un tiempo de respuesta corto. Tiene un error relativo de ± 0.5 ℃ en medición de temperatura y ±2% en medición de humedad. El rango de medición de temperatura es de: -40 ℃ a 80 ℃, el rango de medición de humedad es de: 0 a 100%, el voltaje de funcionamiento es desde 3.5 a 5.5 V, la corriente de funcionamiento es de 60 uA, y la resolución de sus mediciones son de 0.1 ℃ y de 1% para temperatura y humedad respectivamente.” (Taştan & Gökozan, 2019)

7.1.1.5 Pluviógrafo de cubeta basculante

“El pluviógrafo de cubeta basculante (también llamado de balancín) se utiliza para medir la intensidad de lluvia, así como los totales acumulados. El principio en el que se basa el funcionamiento de este instrumento es muy sencillo. Un pluviógrafo de cubeta basculante utiliza un depósito con dos compartimentos gemelos de metal o plástico para medir el agua entrante en partes del mismo peso. Cuando un compartimento se llena, el centro de la masa se desplaza del eje y se produce un movimiento de basculación, con lo que se vacía el agua recogida y el segundo compartimento se coloca en la posición de captación.” (OMM, 2014)
El movimiento de inclinación muve un imán sonre un interruptor magnético y hace que un el circuito se conecte electricamente lo cual envia una señal digital que equivale a una inclinación de una de las cubetas. El volumen de cada cubeta equivale a 0.2 mm la resolución de medicion del sensor es de 0.2 mm.

### 7.1.2 Programación

7.1.2.1 Arduino IDE

“Arduino IDE es un entorno de desarrollo integrado, una aplicación para Linux, Mac OS y para Windows. Principalmente, el lenguaje de programación JAVA se utiliza para desarrollar

aplicaciones. Para Arduino, se utiliza el lenguaje IDE C y C ++, siguiendo la regla de estructuración del código. Los códigos de la versión 2 para IDE, que es lanzada por GNU (Licencia pública general), IDE entrega solo las entradas y salidas similares y es suministrada por la biblioteca de software.

Solo se requieren dos funciones básicas para que el usuario inicie el boceto y el bucle del programa principal que se compilan y vinculan con el programa mediante stub main (), que se convierte en un programa ejecutable con la ayuda de la herramienta GNU. El programa IDE se convierte en codificación hexadecimal por código ejecutable, y estos códigos se cargan en el IDE ARDUINO por el programa cargador en el firmware de la placa." (Vijaya Rajan, Babu, Karthik Pandiyan, Venkatragavan, & Shanmugam, 2019)

### 7.1.3 Almacenamiento de datos y vinculación

7.1.3.1 Ubidots

Ubidots es un servicio de Hosting IoT con experiencia especializada en el desarrollo de hardware y software que permite a la plataforma integrar sin esfuerzo la producción de dispositivos, la computación en la nube y la implementación (Anggraini, Effendy, Ihsan Al Hafiz, & Ojeda Luviano, 2018). Ubidots permite alamacenar e interpretar informacion de sensores en tiempo real haciendo posible la creación de aplicaiones IoT facil y rapida (Enciso & Vargas, 2018).

7.1.3.2 ThingSpeak

ThingSpeak es una plataforma basada en IoTCloud que muestra datos recogidos por diversos sensores de manera remota y se envian a traves de internet (Srivastava et al., 2018). Proporciona visualizaciones instantáneas de los datos publicados por sus dispositivos en ThingSpeak. A menudo se usa para la creación de prototipos y sistemas de prueba de concepto de IoT que requieren análisis (Radhika & Aruna Kumari, 2019).

7.1.3.2 IFTTT

"IFTTT (''IfThis, Then That'') es un servicio web gratuito para crear cadenas de declaraciones condicionales simples que también se denominan applets. ''Esto'' y ''Eso'' son el desencadenante y la acción de un applet, respectivamente. En otras palabras, una

aplicación IFTTT (applet) funciona de la manera que ‘‘ Si se observa un desencadenante, realice una acción’’. IFTTT también puede concatenar diferentes servicios populares de Internet, como Gmail, Instagram, Facebook y SmartThings.” (Xu et al., 2019)

### 7.1.4 Visualización

7.1.4.1 Wix.com

“Wix.com es una plataforma de desarrollo web basada en la nube que permite a los usuarios crear sitios web HTML5 mediante arrastrando y pegando. Los usuarios pueden agregar funciones como enlaces de redes sociales, comercio electrónico, contacto formularios, marketing por correo electrónico y foros de la comunidad a sus sitios utilizando una variedad de aplicaciones Wix y de terceros.” (Iskandar & Adhayani, 2018)

7.1.4.2 Google Maps API

Google Maps API es una función de aplicación emitida por Google para facilitar a los usuarios que desean integrar Google Maps en sus respectivos sitios web (Faizah et al., 2019).

7.1.4.3 AppsGeyser

AppsGeyser es un app builder que permite entre otras posibilidades: realizar la versión móvil de una página web, integrar navegador propio, insertar videos, trabajar con código html, etc. (Gabino, Fuertes, Lopresti, Defranco, & Lara, 2015), permite a cada usuario desarrollar aplicaciones en un formato Android personalizado (Sinyo, 2019).

7.1.4.4 Twitter

“Twitter, que puede clasificarse como una forma específica de actividad en las redes sociales, la plataforma se utiliza principalmente para la interacción social, el intercambio de información, la búsqueda de información, la autodocumentación y la autoexpresión. La función de microblogging de Twitter permite a los usuarios publicar sus ideas y opiniones en formato de "mensajes en tiempo real" escribiendo tweets limitados a cierto número de caracteres.” (Malik, Heyman-Schrum, & Johri, 2019)

7.1.4.5 Facebook

“Facebook es el panorama de redes sociales más predominante y visitado en todo el mundo. Hoy en día, Facebook es muy popular entre los adolescentes y los adultos jóvenes debido a varios aspectos cruciales, incluida su naturaleza amigable para el usuario, mejor interfaz, seguridad de primera clase de las cuentas, y proporciona comunicación instantánea y oportunidades de comentarios inmediatos.” (Abrahim, Mir, Suhara, Mohamed, & Sato, 2019)

## 7.2 Métodos

### 7.2.1 Ciclo de la vida de desarrollo de sistemas

El desarrollo de sistemas de información está basado bajo una metodología sistemática con el cual se lleva a cabo el análisis y diseño de los sistemas de información. Es una metodología en fases para el análisis y diseño, de acuerdo con la cual los sistemas se desarrollan mejor al utilizar un ciclo especifico de actividades (Kendall, 2011). La metodología se divide en siete fases como se muestra en la Figura 1.

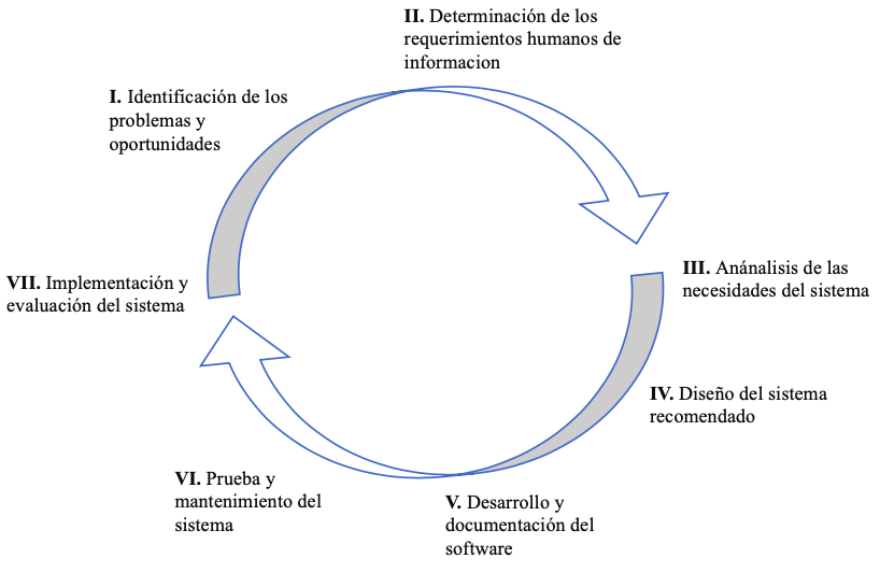


Figura 1: Las siete fases del ciclo de la vida de desarollo de sistemas de información

Fuente: Daniel Rodriguez Licea

I. Identificación de los problemas y oportunidades: en esta primera fase se debe identificar correctamente los, problemas y oportunidades de los sistemas actuales dentro de la zona de

estudio. Se debe realizar un análisis con honestidad de los que está ocurriendo con los sistemas. Las oportunidades residen dentro de las situaciones que se creen poder mejorar mediante el uso de los sistemas automáticos de información.

II. Determinación de los requerimientos humanos de información: determinar las necesidades de los usuarios involucrados, mediante el uso de varias herramientas, para comprender la forma en que interactúan con los sistemas actuales. Se deben de realizar muestreos e investigación de datos duros, entrevistas etc.

III. Análisis de las necesidades del sistema: se realiza un análisis de los requerimientos para preparar una propuesta en la que se sintetice todo lo que se ha averiguado de los sistemas actuales y usuarios.

IV. Diseño del sistema recomendado: se utiliza la información recolectada y analizada para realizar el diseño del sistema, desde los procedimientos entrada y salida de datos, bases de datos, controles y procedimientos para proteger el sistema y los datos y lo más importante, la interfaz ya que esta es quien conecta con el usuario.

V. Desarrollo y documentación del hardware y software: en esta fase se desarrolla el software, manuales de procedimientos y prototipos. La parte de la documentación indica a los usuarios como deben usar el software y que deben hacer en caso de que ocurran problemas.

VI. Prueba y mantenimiento del sistema: antes de utilizar el sistema, se debe probar para detectar problemas o errores en el sistema antes de entregar a los usuarios. En esta fase entran los procesos de calibración del sistema utilizando datos reales de sistemas alternativos. El mantenimiento es una de las partes que se llevan a cabo durante toda la vida del sistema por lo cual deben desarrollarse manuales de mantenimiento para llevar acabo el mantenimiento adecuado del sistema.

VII. Implementación y evaluación del sistema: en esta etapa se implementa el sistema de información, y se aplican métodos de validación de la información al igual que una evaluación de su funcionalidad.

### 7.2.2 Creación de prototipos

La creación de prototipos toma un enfoque iterativo al proceso de desarrollo de sistemas. Durante cada iteración se identifican y analizan los requerimientos y soluciones alternativas al problema, se diseñan nuevas soluciones y se implementa una porción del sistema. Entonces se prueba el prototipo y se evalúa. La creación de prototipos comienza con la creación del modelo preliminar de un subsistema principal o una versión a escala de todo el sistema (Stair & Reynolds, 2010). En la figura 2 se muestra la metodología en todas sus fases.


Figura 2: Creación de prototipos

Fuente: Daniel Rodriguez Licea

### 7.2.3 Calibración de sensores por climas controlados

(Muñoz et al., 2018) hace mención que antes de instalar las estaciones meteorológicas experimentales en campo, los sensores deben calibrarse mediante un clima o entorno controlado para que estos recojan datos de calidad. La calibración mediante clima controlado

consiste en simular un clima de dimensión conocida al que se someterá el sensor experimental junto con un sensor o dispositivo de comparación (equipo de laboratorio o dispositivos comerciales precisos). Los sensores deberán probarse en condiciones mínimas, medias y máximas, baja, media, alta y ligera, moderada y fuerte para temperatura, humedad y precipitación respectivamente durante un periodo de 5 días con intervalos de 1 hora, posteriormente analizar los datos y obtener la precisión de los sensores experimentales. Siendo el caso de que la precisión del sensor no cumpla con la deseada debe ser ajustado en la programación mediante un factor de corrección.

### 7.2.4 Método de validación de sensores por comparación de observaciones con estación cercana

Como (Strigaro et al., 2019) explica en su investigación: el objetivo de la validación de sensores no es más que comparar la calidad de los datos provenientes de una estación meteorológica experimental con una estación meteorológica comercial (dispositivo de adquisición de datos meteorológicos que entrega datos precisos), el método consiste en la instalación de la estación meteorológica experimental en el mismo sitio que la comercial y recolectar información durante un periodo de tiempo para su posterior análisis estadístico de las series de datos medios mínimos y máximos como lo indica la (OMM, 2014). La información del sensor es valida cuando esta este dentro del rango de tolerancia de presión correspondiente al tipo de sensor como se indica en la Guía de Instrumentos y Métodos de Observación Meteorológicos de la OMM.

## 8. Resultados

En este apartado se presentan los resultados de la investigación desarrollando las siete fases de la metodología de del desarrollo de sistemas de información, iniciando con la caracterización de la zona de estudio y el análisis de los sistemas actuales, posteriormente el análisis de las necesidades del usuario, con base a lo anterior se genera una problemática para la cual se determina una propuesta como solución a esta. Definida la solución se detalla el diseño del sistema seguido de las pruebas y mantenimiento concluyendo con la validación del sistema.

## 8.1 Identificación de los problemas y oportunidades

### 8.1.1 Caracterización de la zona de estudio.

Morelia cuenta con una infraestructura de monitoreo de variables meteorológicas por tres sistemas distintos: EMAS-CONAGUA (3 estaciones), OOAPAS (10 estaciones) y CLICOM (9 estaciones) como se muestra en la figura 3.

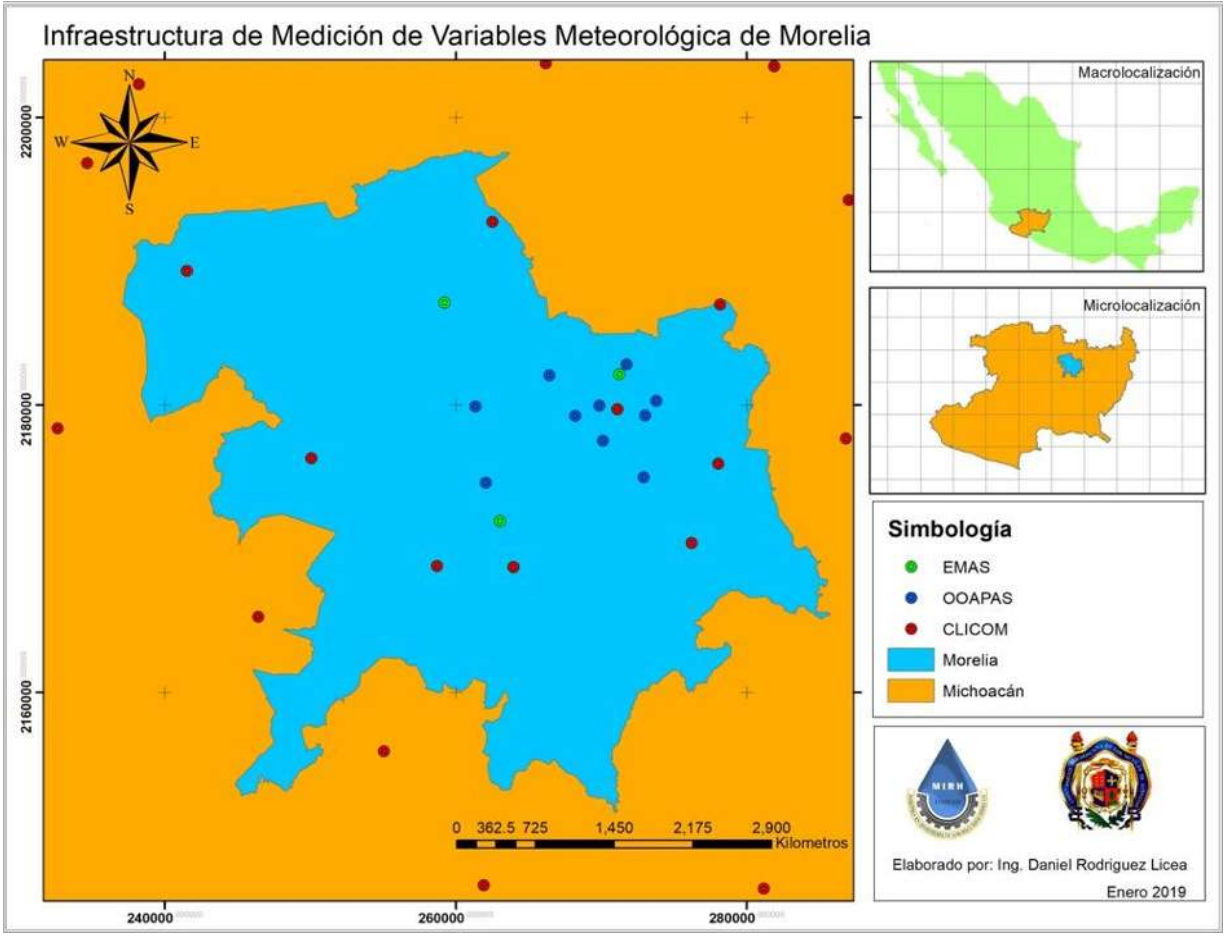


Figura 3: Infraestructura de medición de variables meteorológicas dentro de la zona de estudio.

Fuente: Daniel Rodriguez Licea

### 8.1.2 Caracterización de los sistemas de información de la zona de estudio.

Se analizaron los tres sistemas de información considerando características imprescindibles en los sistemas de alerta temprana además de considerar la implementación de tecnologías de bajo costo y el uso de SIG.

| Característica | EMAS-CONAGUA | OOAPAS | CLICOM |
|---|---|---|---|
| Monitoreo en tiempo real | SI | SI | NO |
| Alertas | NO | NO | NO |
| Generación de bases de datos históricas | SI | NO | SI |
| Información publica | SI | SI | SI |
| Bajo costo | NO | NO | NO |
| Uso de SIG | NO | SI | SI |

Tabla 1: Caracterización de los sistemas de información de la zona de estudio.

Fuente: Daniel Rodriguez Licea

Los problemas identificados en los sistemas de información de Morelia resultan de la carencia de las características imprescindibles de estos, en primer lugar el no ser sistemas automatizados en tiempo real, y su tiempo tan largo en la actualización tienen como consecuencia, el no funcionar como sistemas de alerta temprana, tienen problemas de visualización de la información al no hacer el uso de sistemas de información geográfica (SIG), al no generar bases de datos históricas, es imposible publicar la información en la web, uno de los problemas más importantes es el costo de estos sistemas, los diversos sistemas que existen no solo en Morelia son muy costosos al utilizar dispositivos comerciales para monitoreo.

Cada sistema carece de características importantes que se deberían mejorar, debido al grado de importancia de las instituciones que administran difícilmente podríamos tomar la iniciativa de mejorarlos, en base a las carencias que presentan estos sistemas residen las oportunidades para creer sistemas alternos basados en las características imprescindibles para estos para el óptimo desempeño.

### 8.2 Determinación de los requerimientos humanos de información

Todo un trabajo de campo permite conocer las necesidades entre los usuarios que continuamente hacen uso de los sistemas de información de nuestra zona de estudio, la aplicación de entrevistas estratégicas a usuarios finalmente nos devuelve toda una gama de requerimientos que englobados se direccionan a necesitar sistemas automáticos de

información hidrológica. Usuarios del tipo tomadores de decisiones establecidos en instituciones de gestión de riesgos requieren sistemas que generen información en tiempo real, que generen alertas tempranas, que almacenen información históricamente. Usuarios del tipo planeadores requieren información histórica continua actualizada, para ello es necesario el uso de sistemas de acceso publico donde la información este disponible para su descarga además de visualizarla tanto temporal como espacialmente. Usuarios del tipo consultores requieren de interfaces como sitios web o aplicaciones móviles donde puedan acceder a las consultas del clima fácilmente. Dentro de estos usuarios existen diversos intereses dependiendo del tipo de información que requieren y la aplicación que se le quiera dar, no obstante, son requerimientos que se engloban en un usuario mas general que lo define el objetivo del uso de los sistemas de información.

### 8.3 Análisis de las necesidades del sistema

En este apartado se desarrollará una propuesta que de solución a la problemática que presentan los sistemas de información existentes en la zona de estudio considerando las oportunidades de mejora y satisfacer los requerimientos de los usuarios.

La propuesta de solución se basa en el desarrollo de un sistema automático de información hidrológica capaz de capturar, administrar, manipular, analizar y visualizar temporal y espacialmente información en tiempo real, permitiendo así la funcionalidad como un sistema de alerta temprana. Su aplicación debe permitir el monitoreo de cualquier zona en estudio mediante dispositivos desarrollados con tecnologías de bajo costo que envíen información a bases de datos en la nube basadas en IoT, de las cuales se pueda hacer uso para la vinculación con sitios web o aplicaciones móviles donde la información sea visualizada tanto temporal como espacialmente, incorporando la generación de alertas a través de diferentes medios (SMS, emails, redes sociales, etc.). En la figura 4 se muestra la arquitectura del sistema propuesto.

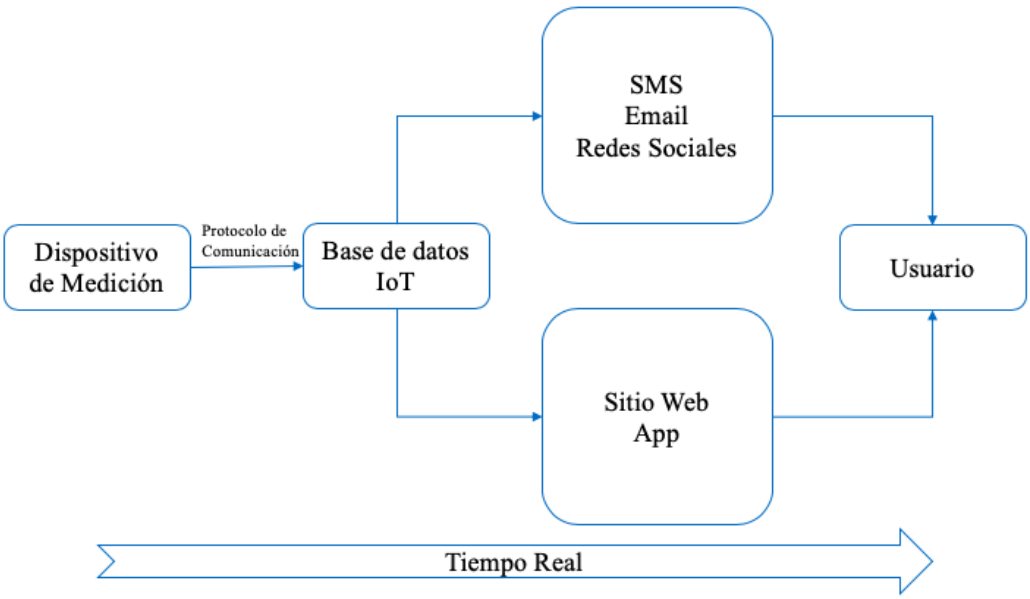


Figura 4: Arquitectura del sistema propuesto.

Fuente: Daniel Rodriguez Licea

### 8.4 Diseño del sistema recomendado

Tomando como base la propuesta de solución se desarrollará el sistema con las características mostradas en la figura 4, en primer lugar, se desarrollará un dispositivo de adquisición de datos de bajo costo, en seguida de la implementación del servicio de base de datos IoT y finalmente el diseño de las plataformas de visualización.

#### 8.4.1 Diseño del dispositivo de adquisición de datos de bajo costo

El dispositivo de adquisición de datos de bajo costo tomará el enfoque hacia una especie de estación meteorológica automática de bajo costo (EMABC) la cual se desarrollará bajo la metodología de creación de prototipos. A continuación, se detalla el diseño de dos estaciones meteorológicas automáticas de bajo costo.

##### 8.4.1.1 EMABC Wifi

La EMABC Wifi debe cumplir con los siguientes requerimientos: medición de precipitación, temperatura y humedad relativa del aire en tiempo real autosustentable energéticamente y el

envió de información a la base de datos sea a través del protocolo de comunicación WiFi. El diseño de la EMABC WiFi ha requerido de 4 iteraciones para llegar al producto final.

8.4.1.1.1 Iteración 1

Las alternativas para el desarrollo de esta iteración parten de la consideración del uso de una placa Arduino UNO como microcontrolador, un sensor DHT11 para la medición de temperatura y humedad del aire y un sensor YFS201 para la medición de la precipitación a través del flujo medido por el sensor. Para la lluvia en particular se consideró utilizar un receptor de lluvia basado en las características de los pluviómetros con la finalidad de aplicar el principio de medición de estos y convertir el gasto y el volumen de flujo del sensor en intensidad y precipitación, para la obtención de fecha y hora se consideró utilizar un Modulo RTC DS3231 y para el envió de datos un shield de Wifi para Arduino UNO.

La EMABC con comunicación wifi se desarrollo bajo la arquitectura que se muestra en la figura 5.

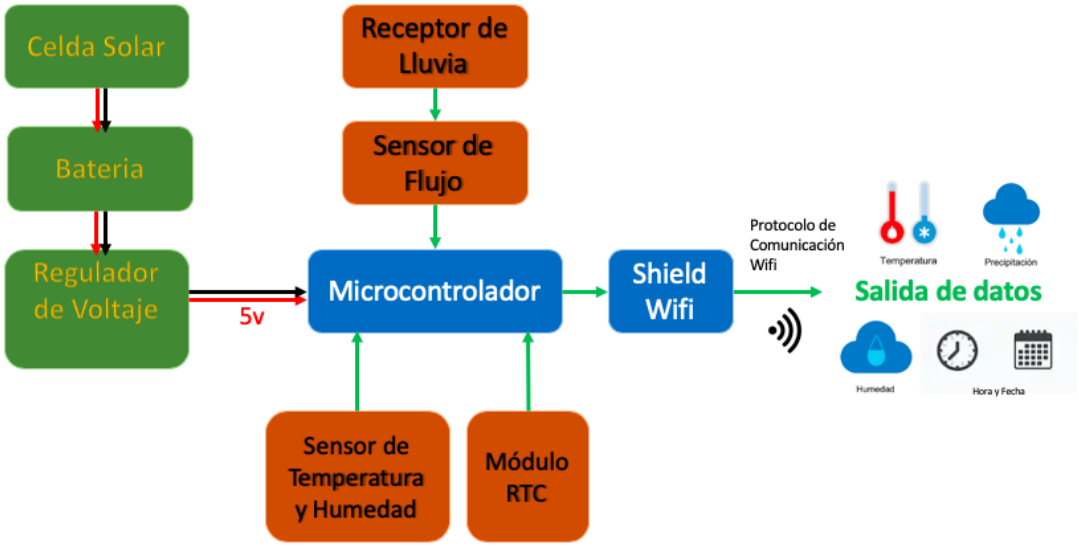


Figura 5: Arquitectura EMABC Wifi Iteración 1.

Fuente: Daniel Rodriguez Licea

En la figura 6 se muestran los circuitos electrónicos de la EMABC Wifi con todos los componentes necesarios para la funcionalidad de este.

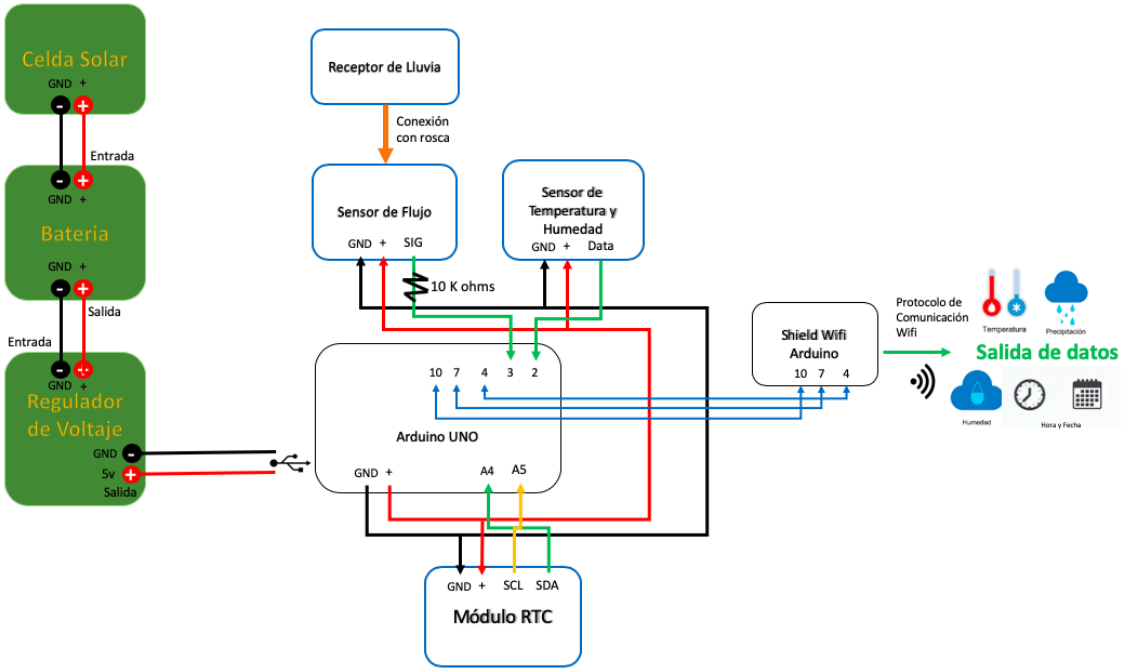


Figura 6: Diagrama de conexión de la EMABC Wifi iteración 1.

Fuente: Daniel Rodriguez Licea

Terminada la primera iteración tenemos un dispositivo de medición de precipitación, temperatura y humedad del aire en tiempo real con comunicación Wifi, que se alimenta energéticamente a través de un sistema de panel solar, este resulta no ser suficiente para el dispositivo debido a los componentes que se forman el dispositivo, la conexión Wifi es inestable y esto conlleva a que la medición sea discontinua. El dispositivo envía datos de fecha y hora adicionales a las variables medidas, realmente esta información es innecesaria ya que capturada la información enviada por el dispositivo en la base de datos esta registra la fecha y hora de captura de la información.

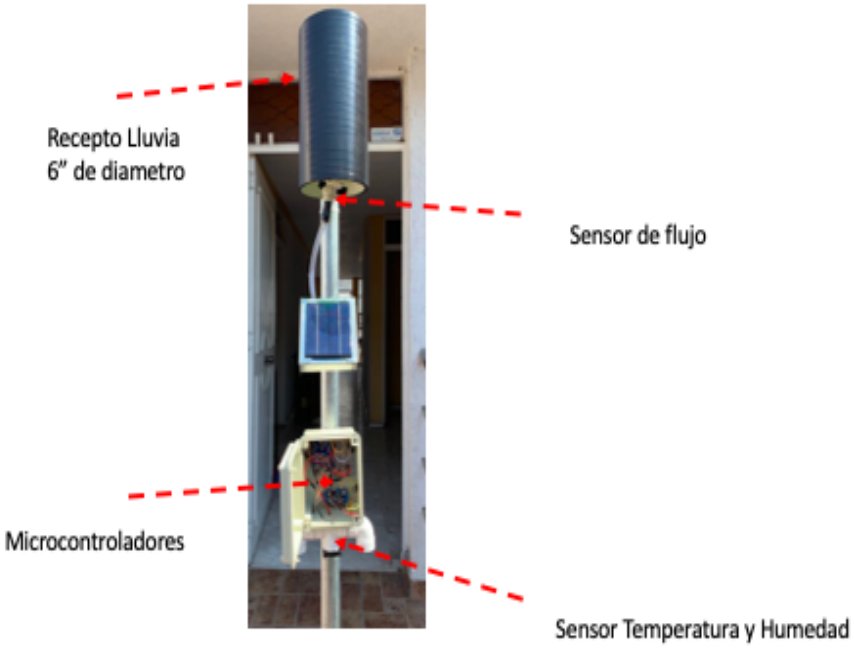


Figura 7: EMABC Wifi Iteración 1.

Fuente: Daniel Rodriguez Licea

8.4.1.1.2 Iteración 2

Con base a los requerimientos y a las pruebas de usuario de la iteración 1 las alternativas para el desarrollo de la EMABC se basarán en una NodeMCU como microcontrolador, un sensor DHT11 para la medición de temperatura y humedad del aire y un sensor YFS201 para la medición de la precipitación a través del flujo medido por el sensor. Para la lluvia en particular se consideró utilizar un receptor de lluvia basado en las características de los pluviómetros con la finalidad de aplicar el principio de medición de estos y convertir el gasto y el volumen de flujo del sensor en intensidad y precipitación. En este caso se han requerido dos microcontroladores debido a que estos nos son capaces de emitir dos señales digitales al mismo tiempo por lo cual se ha colocado un sensor por microcontrolador. En la figura 8 se muestra la arquitectura de la EMABC Wifi en su segunda iteración.

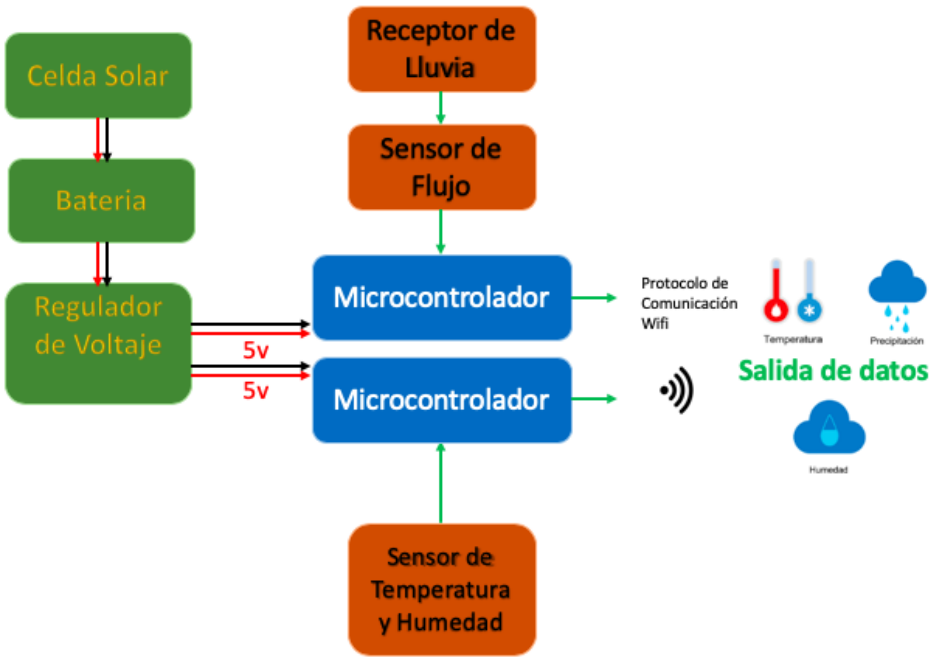


Figura 8: Arquitectura EMABC Wifi Iteración 2.

Fuente: Daniel Rodriguez Licea

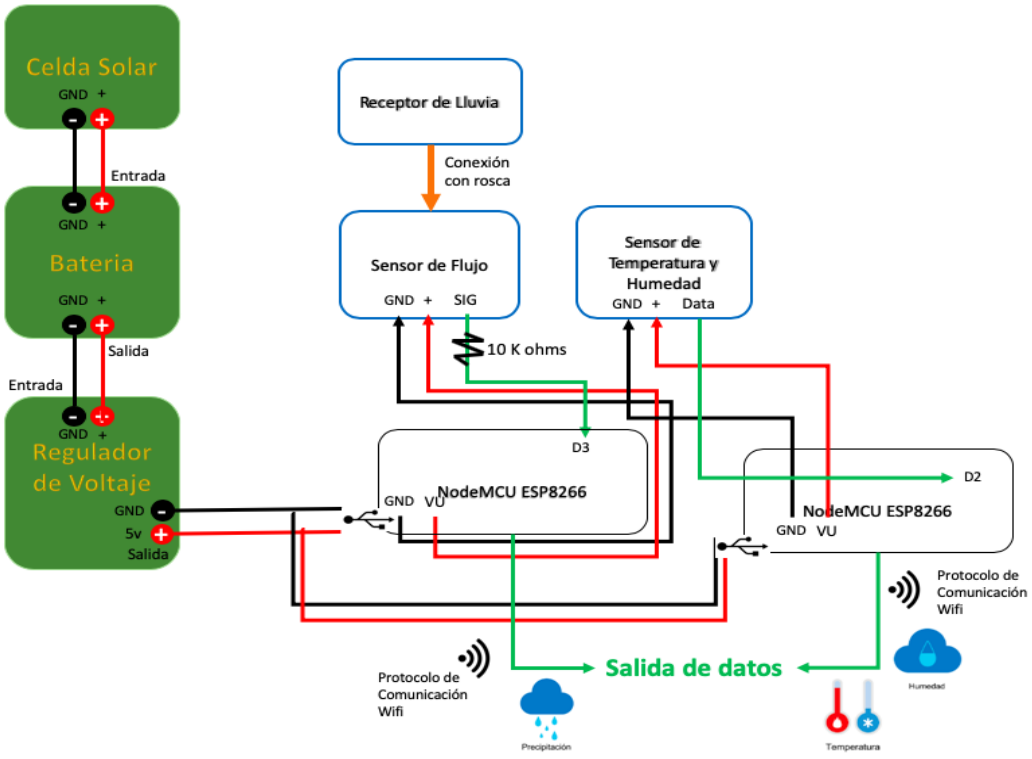


Figura 9: Diagrama de conexión de la EMABC Wifi iteración 2.

Fuente: Daniel Rodriguez Licea

Este segundo dispositivo se sometió a pruebas de funcionalidad y sensibilidad en los sensores resultando un dispositivo con conexión estable con lo cual el envió de información de precipitación, temperatura y humedad del aire es continua, el sistema de alimentación energética autosustentable no es lo suficientemente capaz de mantener el dispositivo. De las pruebas de sensibilidad se ha encontrado una deficiencia en el sensor de medición de flujo para la obtención de precipitación e intensidad, este es un sensor de flujo continuo el cual no es capaz de medir flujos pequeños que representan lluvias débiles de tal forma que estas pequeñas cantidades de lluvia se pierden por no ser registradas por el sensor.


Figura 10: EMABC Wifi Iteración 2.

Fuente: Daniel Rodriguez Licea

8.4.1.1.3 Iteración 3

Analizando los resultados de la iteración 2 y tomando en cuenta los requerimientos, para este dispositivo mantendremos una arquitectura similar, en este caso utilizaremos un solo microcontrolador NodeMCU, un sensor DHT11 de temperatura y humedad del aire y un

sensor YL83 para la medición de la precipitación. El dispositivo contará con un sistema de alimentación energética autosustentable. En la figura 11 y 12 se muestra la arquitectura del dispositivo y su diagrama de conexión respectivamente. El sistema de alimentación eléctrica se sustituyo por uno mas potente capaz de mantener al dispositivo funcionando durante 24 horas y dos días de autonomía extra en caso de no tener días despejados de nubes que no permitan la recarga del sistema (en el apartado 8.4.1.3 se detalla el diseño del sistema de alimentación eléctrica autosustentable).

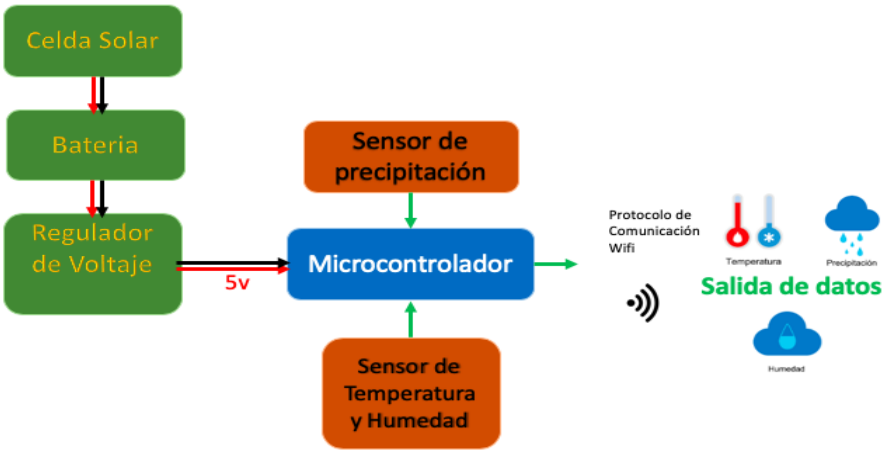


Figura 11: Arquitectura EMABC Wifi Iteración 3.

Fuente: Daniel Rodriguez Licea

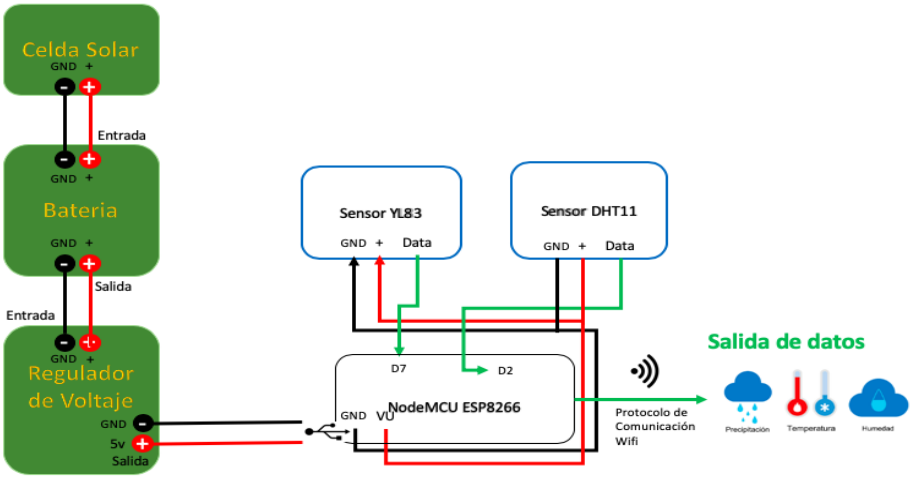


Figura 12: Diagrama de conexión de la EMABC Wifi iteración 3.

Fuente: Daniel Rodriguez Licea

Una vez puesta en marcha se observo el funcionamiento de la misma resultando un dispositivo estable en conexión wifi por lo que el envió de información de precipitación, temperatura y humedad del aire son continuos y logran registrarse en la base de datos sin problema alguno, el sensor YL-83 puede ser utilizado para medir la intensidad de la lluvia y de esta manera emitir alertas si lo es necesario pero, no es suficiente para la medición de cantidad de lluvia ya que este retiene agua aun después de terminar la lluvia. (Strigaro et al., 2019) comprueba en su investigación que el sensor DHT11 no es suficiente para entregar datos de temperatura y humedad con la precisión requerida por la OMM.

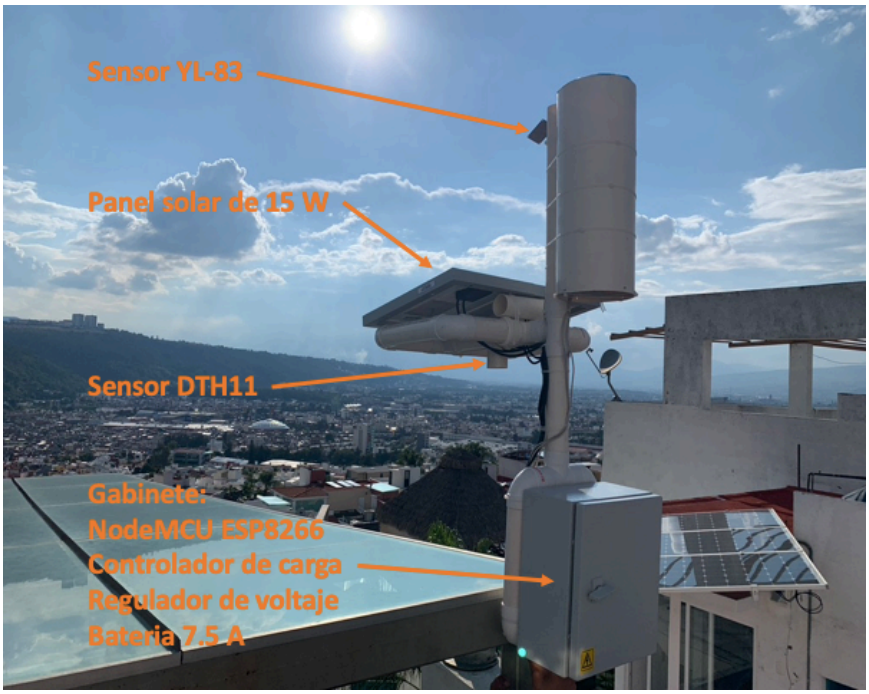


Figura 13: EMABC Wifi Iteración 3.

Fuente: Daniel Rodriguez Licea

8.4.1.1.4 Iteración 4

De la revisión del dispositivo de la iteración 3 y de los requerimientos del dispositivo, para el desarrollo de este en su cuarta iteración mantendremos la arquitectura pasada, utilizando el microcontrolador NodeMCU ESP8266, un sensor de balancín para la medición de precipitación y un sensor DHT22 para la medición de temperatura y humedad del aire. En la

figura 14 y 15 se muestra la arquitectura del dispositivo y su diagrama de conexión respectivamente.

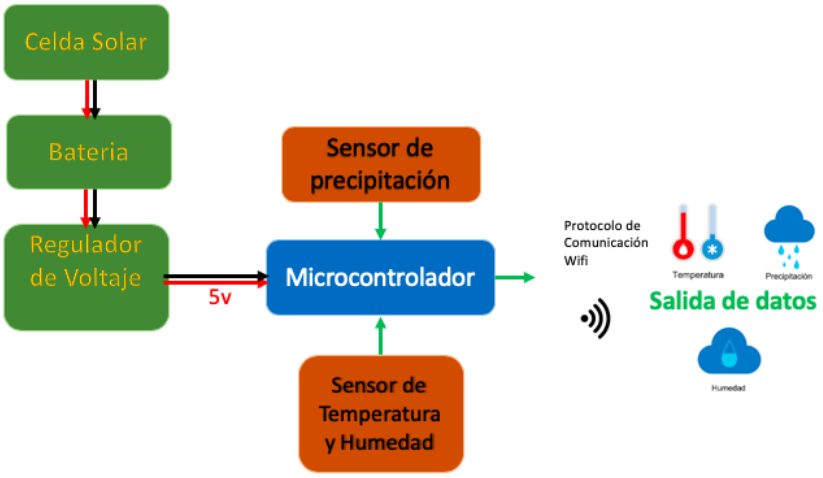


Figura 14: Arquitectura EMABC Wifi Iteración 4.

Fuente: Daniel Rodriguez Licea

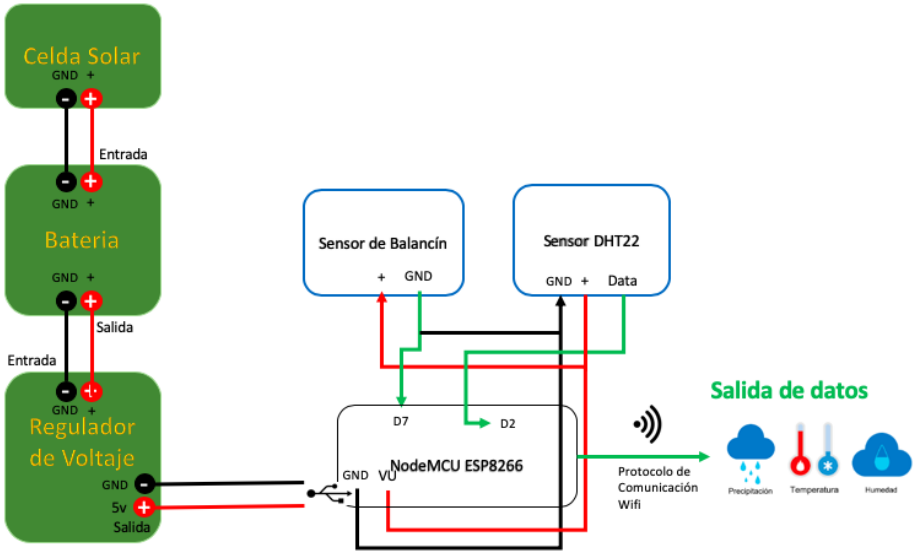


Figura 15: Diagrama de conexión de la EMABC Wifi iteración 4.

Fuente: Daniel Rodriguez Licea

Puesto en marcha el dispositivo se realizo una revisión del mismo para observar el funcionamiento de este, la conexión Wifi es estable por lo cual el envió de información de precipitación temperatura y humedad del aire son continuas y en tiempo real, los sensores envían datos que responden a diversos puntos de sensibilidad (bajo, medio y alto), el sistema

de alimentación eléctrica autosustentable es suficiente para el sistema además de tener la capacidad de alimentar al dispositivo por 2 días en caso de días nublados.

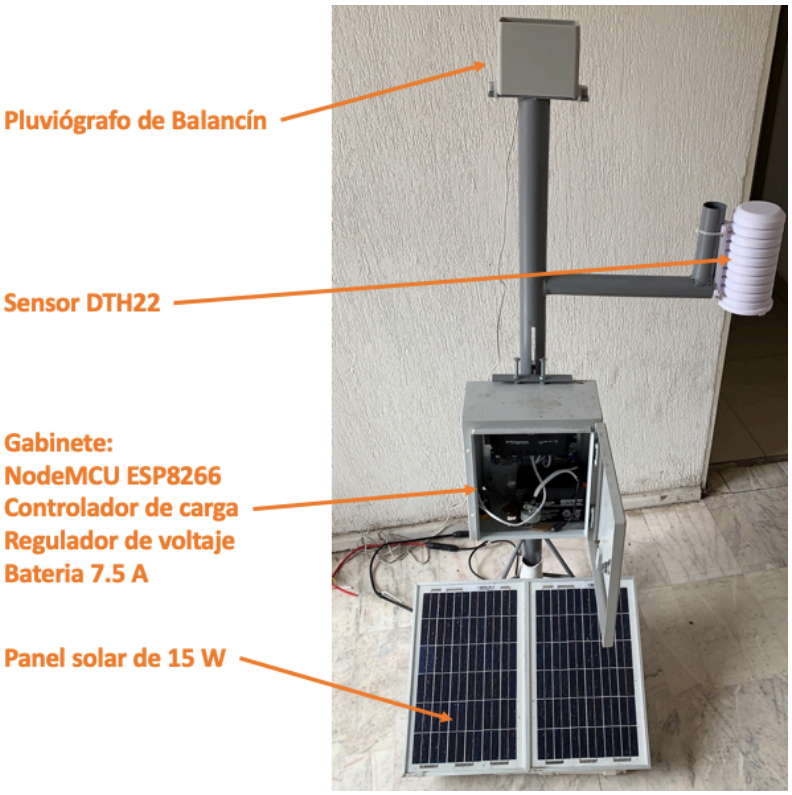


Figura 16: EMABC Wifi Iteración 4.

Fuente: Daniel Rodriguez Licea

8.4.1.2 EMABC GSM

La EMABC GSM debe cumplir con los siguientes requerimientos: medición de precipitación, temperatura y humedad relativa del aire en tiempo real autosustentable energéticamente y el envió de información a la base de datos sea a través del protocolo de comunicación GSM. El diseño de la EMABC Wifi ha requerido de 1 iteraciones para llegar al producto final.

8.4.1.2.1 Iteración 1

Basándonos en la funcionalidad de la EMABC Wifi de la iteración 4 la cual cumple con los requerimientos del dispositivo adaptaremos el mismo de tal forma que pueda comunicarse a

través de la red GSM, para ello utilizaremos un Arduino UNO como microcontrolador principal, un modulo GSM SIM900 para la comunicación GSM, un sensor de balancín para la medición de la precipitación y un sensor DHT22 para la medición de temperatura y humedad del aire. La alimentación del dispositivo será a través del sistema de alimentación eléctrica que se describe en el apartado 8.4.1.3. En la figura 17 y 18 se muestra la arquitectura del dispositivo y su diagrama de conexión respectivamente.

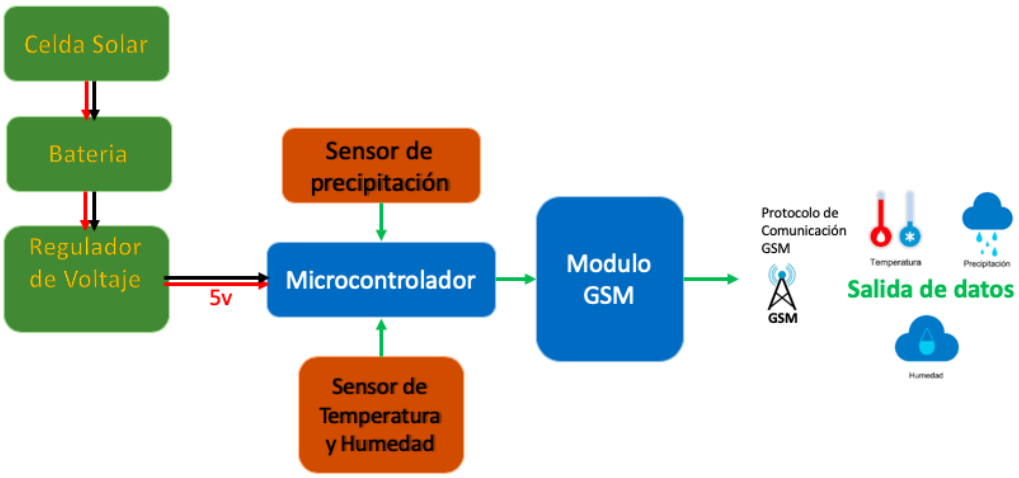


Figura 17: Arquitectura EMABC GSM Iteración 1.

Fuente: Daniel Rodriguez Licea

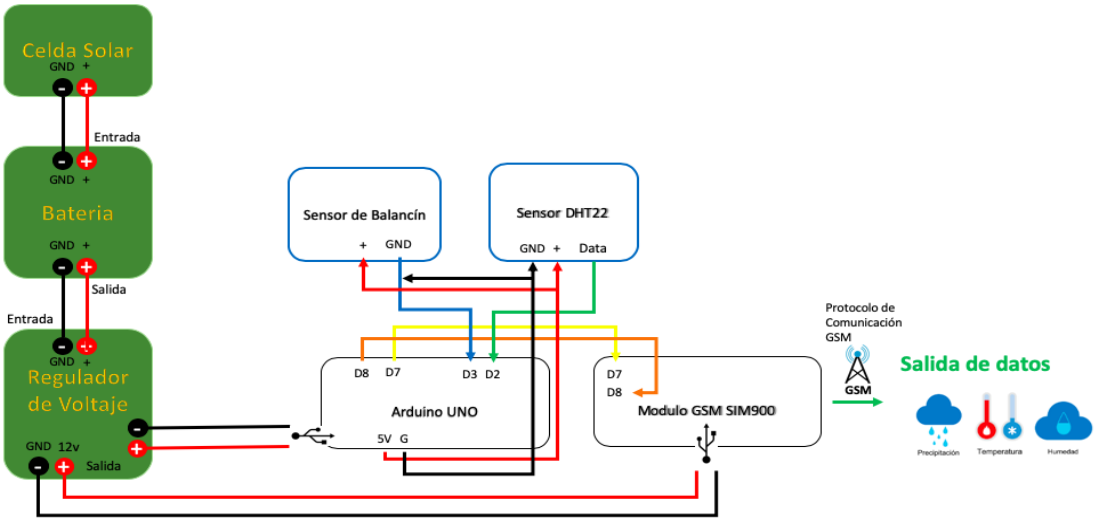


Figura 18: Diagrama de conexión de la EMABC GSM iteración 1

Fuente: Daniel Rodriguez Licea

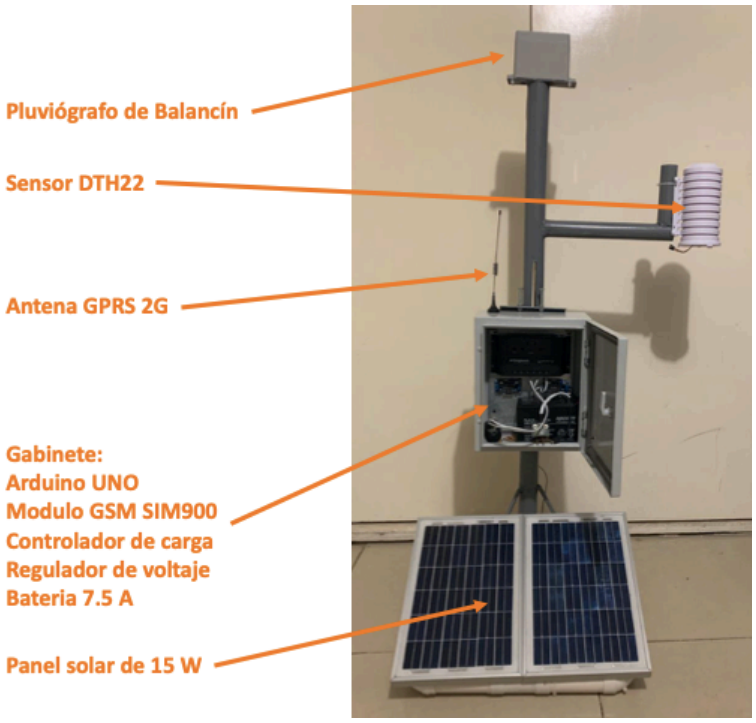


Figura 19: EMABC GSM Iteración 1

Fuente: Daniel Rodriguez Licea

El dispositivo puesto en marcha es un dispositivo que mantiene la conexión con la base de datos al igual que las mediciones continuas y en tiempo real, entrega información de precipitación, temperatura y humedad del aire en tiempo real a través de la comunicación GSM, el sistema de alimentación autosustentable es suficiente para el sistema al igual que con la EMABC Wifi.

8.4.1.3 Sistema de alimentación eléctrica autosustentable

| Dispositivo | Corriente (A) | Potencia (W) | Tiempo de uso (h) | Energía (Wh) |
|---|---|---|---|---|
| 2 Módulos y sensores | 200 mA | 1 Watts | 24 horas | 24 Wh |
| Otros | 100 mA | 0.5 Watts | 24 Horas | 12 Wh |

Tabla 1: Potencia del dispositivo.

Fuente: Daniel Rodriguez Licea

Potencia= Voltaje x Corriente

Energía= Potencia x Tiempo de uso

Capacidad de la batería

$$\text{I}= \frac{36\ Wh}{12\ V} = 3\ \text{Ah}$$

$$C = \frac{Numero\ de\ dias\ de\ autonimía\ x\ Consumo\ diario\ de\ los\ dispositivos}{Máxima\ profundidad\ de\ descarga}$$

Datos:

Número de días de autonomía= 2 días (esto significa que una vez cargada la batería, puede estar funcionando hasta 2 días sin cargar, esto para días lluviosos, si quieres menos o más).

Consumo diario de los dispositivos= 3 Ah (Lo que va a consumir en todo el día el sistema, en este caso estará funcionando las 24 horas, por tanto en tiempo de uso se agregó 24 horas para el cálculo).

Máxima profundidad de descarga= 0.8 (Esto significa que tanto se puede descargar, en este caso de 0.8 significa que se puede descargar solo hasta el 80 % de su capacidad, antes de dañarse).

$$C = \frac{2\ días\ x\ 3\ Ah}{0.8} = 7.5\ A$$

Capacidad del panel solar

$$Potencia\ requerida\ diaria = Potencia\ x\ Tiempo\ de\ uso\ de\ los\ dispositivos$$
$$Potencia\ requerida\ diaria = 1.5\ W\ x\ 24\ Horas = 36\ Wh$$

Potencia que debo generar considerando la eficiencia del controlador de carga

$$= \frac{Potencia\ requerida\ diaria}{Eficiencia\ del\ controlador\ de\ carga}$$

Potencia que debo generar considerando la eficiencia del controlador de carga

$$\text{P}=\frac{36\ Wh}{0.6} = 60\ Wh$$

Capacidad del panel solar

$$= \frac{P\ que\ debo\ generar\ segun\ eficiencia\ del\ c\ de\ carga}{Promedio\ de\ horas\ efectivas\ del\ sol\ segun\ la\ región}$$

$$Capacidad\ del\ panel\ solar = \frac{60\ Wh}{4.2\ h} = 14.285 \approx 15\ \text{watts}$$

### 8.4.2 Implementación del servicio de base de datos IoT

En este apartado se desarrollará el proceso de implementación de las bases de datos IoT que se implementaran en el SAIH. Las EMASBC Wifi o GSM envían información a las bases de datos una vez encendidas estas llevan implícitas instrucciones como la identificación de la base de datos a través de un API KEY y un hosting además de los tiempos de escritura de información a cada minuto.

#### 8.4.2.1 ThingSpeak

Las EMABC's llevan precargado un código (ver el apartado 8.5) que permiten la vinculación con la base de datos para el envió de información. En primer lugar, se creo una cuenta en ThingSpeak.com previamente de la carga del código a la EMABC donde se creo un canal con cuatro campos para la escritura de la información de precipitación (Intensidad y volumen), temperatura y humedad del aire (figura 20), creado el canal se genera un ID (Figura 21) del canal y un API KEY (figura 22) que se incluyen en el código de la EMABC. Puesta en marcha la EMABC inicia con la escritura de información a cada minuto la cual es posible visualizarla en ThingSpeak.com a través de gráficos (figura 23) que posteriormente se comparten en las plataformas de visualización para el usuario.

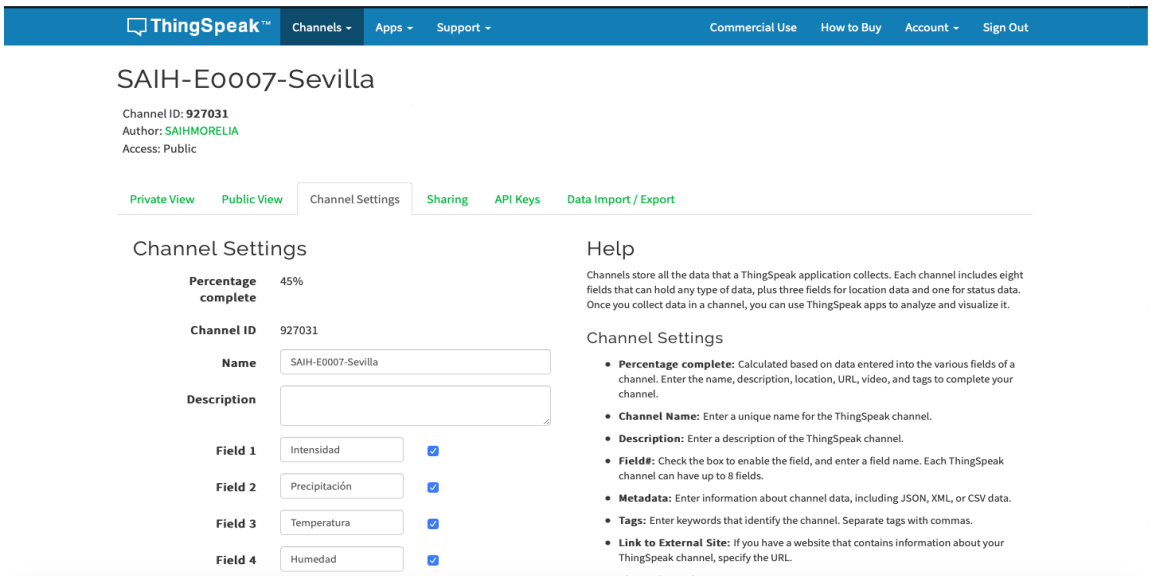


Figura 20: Canal ThingSpeak.

Fuente: Daniel Rodriguez Licea

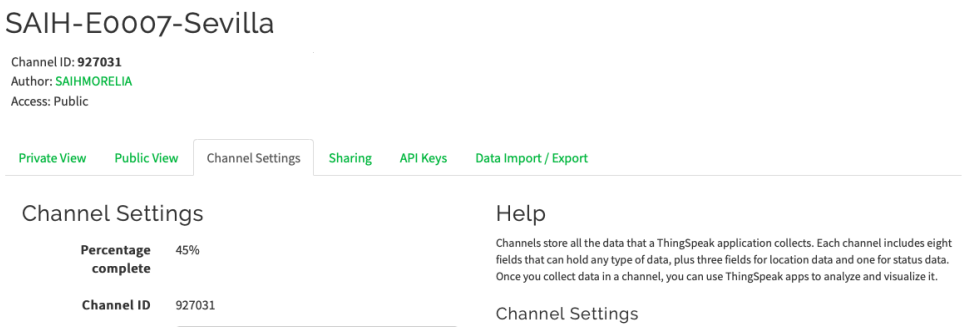


Figura 21: ID del canal ThingSpeak.

Fuente: Daniel Rodriguez Licea

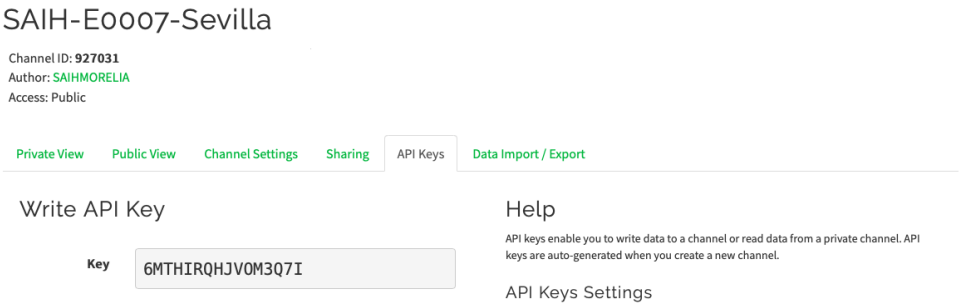


Figura 22: API KEY del canal ThingSpeak.

Fuente: Daniel Rodriguez Licea

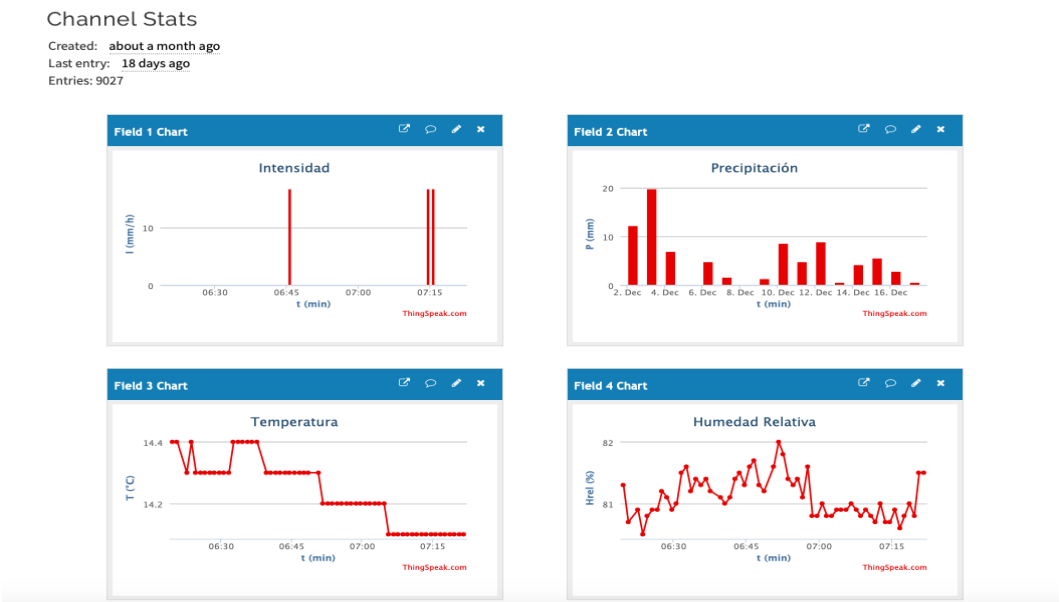


Figura 23: Registro de informacion de la EMABC a la base de datos IoT.

Fuente: Daniel Rodriguez Licea

8.4.2.1 Ubidots

Al igual que en ThingSpeak la EMABC previo de cargar el código requiere de un API KEY que vincula el dispositivo con la base de datos, previamente se creo una cuenta en Ubidots la cual permite obtener una API KEY general para todos los dispositivos a vincular (figura 24). Puesta en marcha, la EMABC envía información de precipitación (intensidad y volumen), temperatura y humedad del aire a cada minuto a la base de datos IoT la cual se visualiza en widges (figura 25) que posteriormente se compartirán en las plataformas de visualización.

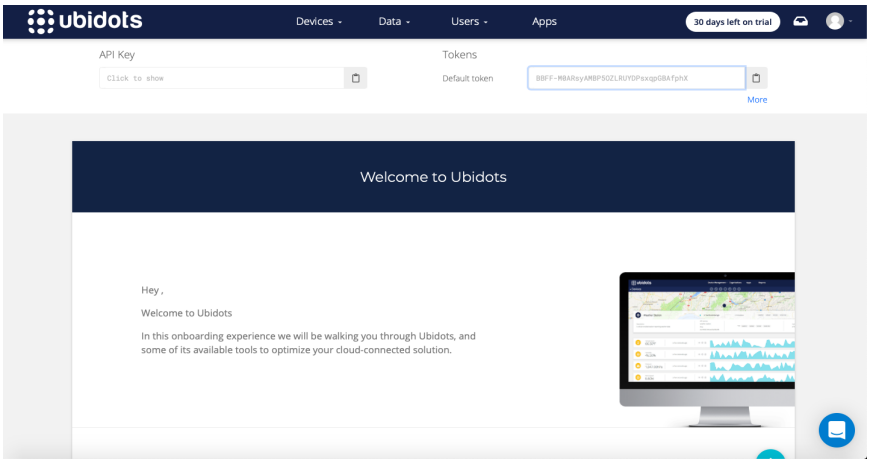


Figura 24: API KEY de Ubidots.

Fuente: Daniel Rodriguez Licea

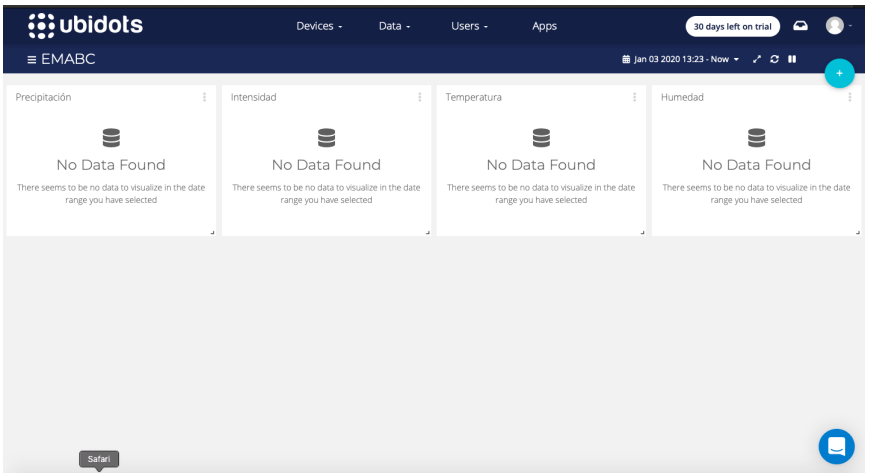


Figura 25: Widges de Visualización de Información enviada por la EMABC a la base de datos IoT.

Fuente: Daniel Rodriguez Licea

### 8.4.3 Diseño de plataformas de visualización

En este apartado de presenta el desarrollo de las plataformas web en sus dos formas: sitio web y App para la visualización espacial y temporal de información de precipitación (intensidad y volumen), Temperatura y humedad del aire; además se detalla el procedimiento de envió y visualización de alertas de riesgos ambientales en las redes sociales (Facebook y Twitter).

#### 8.4.3.1 Desarrollo del Sitio Web

Para el desarrollo de esta interfaz nos basaremos en la metodología de desarrollo de interfaces donde principalmente crearemos un usuario, analizaremos los requerimientos de este para el diseño de la interfaz posteriormente implementaremos el diseño en prototipos de baja y alta fidelidad.

##### 8.4.3.1.1 Creación de usuario

Usuario: Tomador de decisiones

Características: Experiencia en el área, juicio común, creatividad, habilidades cuantitativas, efectos futuros, reversibilidad, impacto, calidad y periodicidad.

Habilidades: Maestro, líder, comunicador, pronosticador, organizador, creador, innovador, analista, estratega, motivador, capacitador, consejero, evaluador, reclutador y visionario.

Cuando se enfrenta a un problema recurre a la experiencia para poder resolverlo, evalúa la información en forma inteligente, es capaz de captar y entender el problema de manera más amplia, emplea técnica presentadas como métodos cuantitativos o investigación de operaciones.

Se anticipa a los cambios y acepta de forma positiva cada cambio que se le presenta. Visualiza y percibe cada cambio como una oportunidad y un reto.

Estas herramientas ayudan al mando a tomar decisiones efectivas, pero es muy importante no olvidar que las habilidades cuantitativas no deben ni pueden reemplazar el buen juicio en el proceso de toma de decisiones. Conocimientos generales de tecnología: uso de computador y Smartphone.

8.4.3.1.2 Análisis de requerimientos del usuario

Es evidente que para estudios de riesgo en el área del recurso hídrico y gestión de proyectos se recurre muy frecuentemente a los SAIH's y bases de datos meteorológicas hasta 3 veces por día, la consulta de estos se lleva a cabo mediante computadoras cuando hablamos de estudios y gestión de proyectos, en el caso de toma de decisiones en campo se utiliza generalmente el teléfono móvil para baja la información y procesarla rápidamente.

Los procesos de descarga de información se vuelven complicados debido a que los puntos de monitoreo no es posible ubicarlos espacialmente, en la mayoría de los casos solo existen bases de datos muy simples que no georreferencian la información por lo cual es ineficaz para la rápida toma de decisiones, ya que ocasiona que el usuario no pueda decidir rápidamente cual punto de monitoreo usar, el no estar georreferenciada la información es imposible conocer la cercanía de los puntos de monitoreo con la zona de estudio. La descarga de información no solo presenta estos problemas, también los sistemas que están georreferenciados generan dificultad de descarga ya que no se puede obtener información directamente de los mapas, si no solo revisar información general del punto de monitoreo para después utilizar los gestores de búsqueda de estos y así descargar la información.

La información en algunos casos es presentada mediante gráficas y tablas, lo cual no resulta eficiente para toma de decisiones ya que la información no se encuentra procesada ni resumida, en otros casos no es posible visualizar la información si no hasta después de la descarga y utilizado herramientas para proceso de datos y visualización como Excel, Matlab, R Studio, etc.

Muy pocos de los sistemas se pueden usar para decidir y tomar acciones cuando ocurren eventos de riesgo, naturalmente, si la zona donde hay que tomar decisiones se encuentra dentro de las redes de monitoreo de los sistemas, ya que estos no son tan amplios. Para la toma de decisiones y la gestión de proyectos se requiere de un procesamiento previo de los datos para obtener parámetros que puedan utilizarse como indicadores y con estos poder

tomar acciones. En la mayoría de los casos el problema se encuentra en la actualización de la información ya que no son sistemas que toman datos en tiempo real.

A manera de comentarios los participantes opinan que lo óptimo sería un sistema fácil de navegar donde la información esté vinculada entre los mapas, existen bases de datos que cubren grandes territorios algunas todo el país, pero estos ofrecen información a pequeña escala, de tal forma que se deben usar métodos para extrapolar la información a las zonas de estudio. Mejorar en cuanto a la actualización frecuente de las bases de datos es una de las partes más importantes de los sistemas si no son en tiempo real que se vuelva periódica la actualización y no esperar años para actualizarlas. Se habla de que la usabilidad de las interfaces de los sistemas no es apta para la rápida consulta.

8.4.3.1.3 Prototipo de baja fidelidad

Este prototipo se genera a través de una simulación de una página web creada en papel, con estas se evalúa la funcionalidad de estas interfaces de tal forma que puedan cumplir con la usabilidad para el usuario.

La interfaz consta de una ventana principal por variable donde se muestra la distribución de estaciones en forma de puntos a través de un SIG, al dar click sobre estas despliegan una etiqueta emergente que permite visualizar la información de la variable en tiempo real además de contener un enlace que lleva a una ventana secundaria donde es posible ver la información temporalmente y descargarla en ficheros CSV. En la figura 26 se muestra el funcionamiento del prototipo de baja fidelidad.

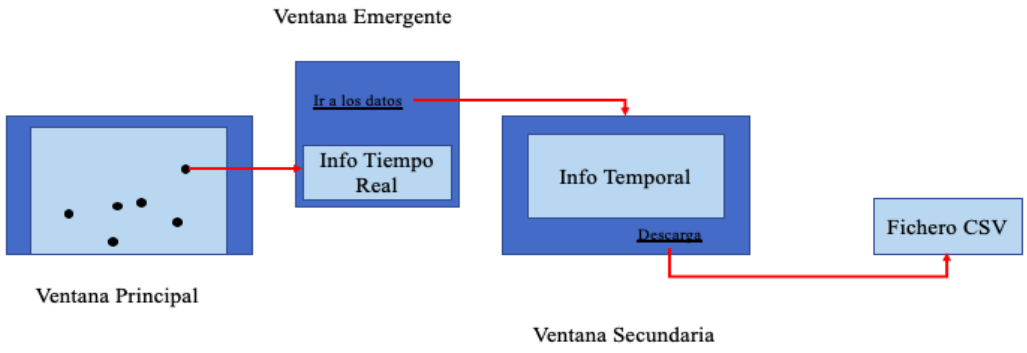


Figura 26: Prototipo de baja fidelidad.

Fuente: Daniel Rodriguez Licea

8.4.3.1.4 Prototipo de alta fidelidad

El prototipo de alta fidelidad se basa en el de baja fidelidad ya que en este se prueba la usabilidad de la plataforma y se replica en un sitio web a través de la herramienta Wix.com que permite la vinculación de la API de google Maps (figura 27) para la visualización espacial y los widgets de Thingspeak y Ubidots para la visualización temporal (figura 28). Previamente se creo una cuenta en wix.com, para obtener un hosting que mantenga el sitio web y un dominio que nos lleve a este (véase en www.saihmorelia.com). Las herramientas de wix.com nos permitieron generar un encabezado donde se colocaron los logos principales de las instituciones que respaldan la investigación, posteriormente en el cuerpo de la interfaz se añadió un elemento HTML para la incorporación del mapa donde se presenta red de estaciones (en el anexo 5 se detalla el código HTML), finalmente se añade un pie de pagina con enlace a las redes sociales, donaciones y contador de visitas. Lo anterior para la ventana principal, en la secundaria se añaden elementos iframe que permiten colocar un código HTML, este proporcionado la base de datos para la visualización temporal de las variables (figura 29 y 30).

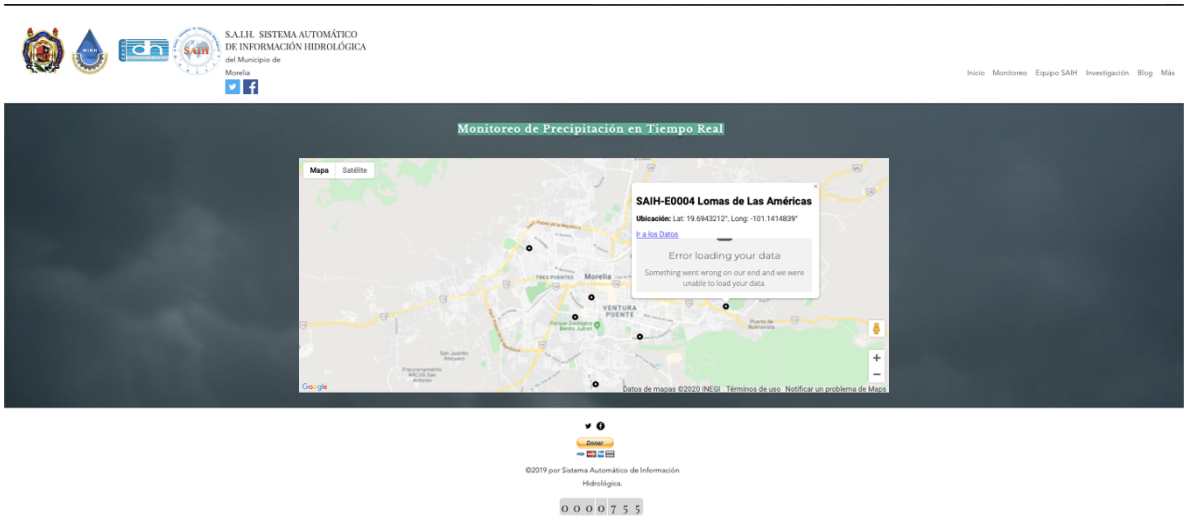


Figura 27: Ventana principal visualización espacial a traves de la Api de Google Maps.

Fuente: Daniel Rodriguez Licea

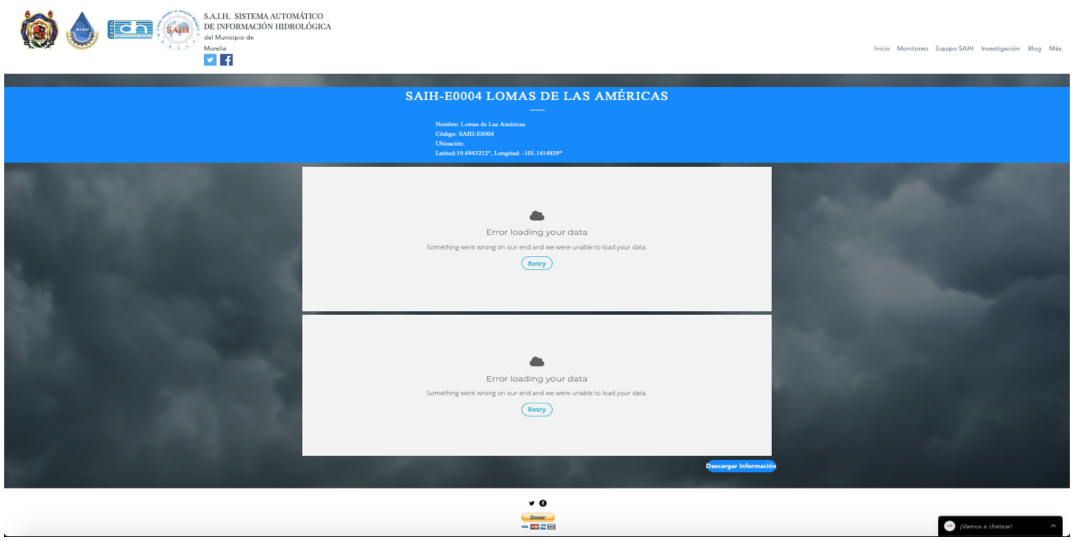


Figura 28: Ventana secundaria de visualizacion temporal atraves de los gidges de ThingSpeak y Ubidots.

Fuente: Daniel Rodriguez Licea

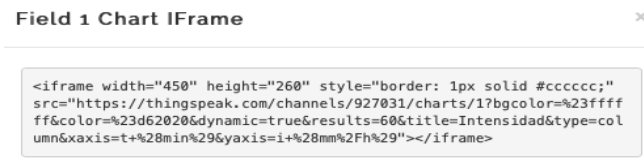


Figura 29: Iframe de ThingSpeak para la vinculación con Wix.com.

Fuente: Daniel Rodriguez Licea

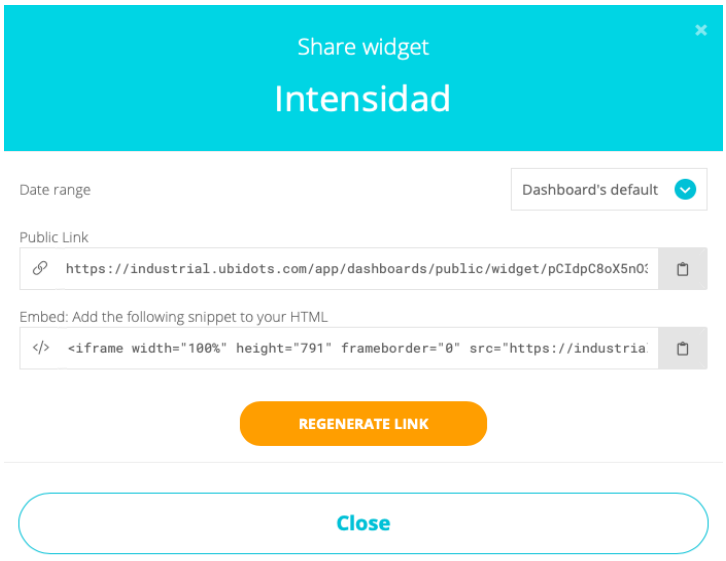


Figura 30: Iframe de Ubidots para la vinculación con Wix.com.

Fuente: Daniel Rodriguez Licea

8.4.3.2 Desarrollo de la App

La App se basó en el sitio web que antes se desarrolló en versión móvil únicamente, de tal forma que pueda ser usada en el móvil sin ningún problema. El desarrollo de la aplicación se llevó a cabo a través de AppsGeyser, una herramienta online para creación de ficheros .jdk, estos son archivos de aplicación para Android que permiten la ejecución de la aplicación en el Smartphone para una instalación directa desde el archivo. Primeramente, se creo una cuenta en AppsGeyser (figura 31) vinculada a una cuenta de Gmail que previamente se creo para la vinculación de las bases de datos.

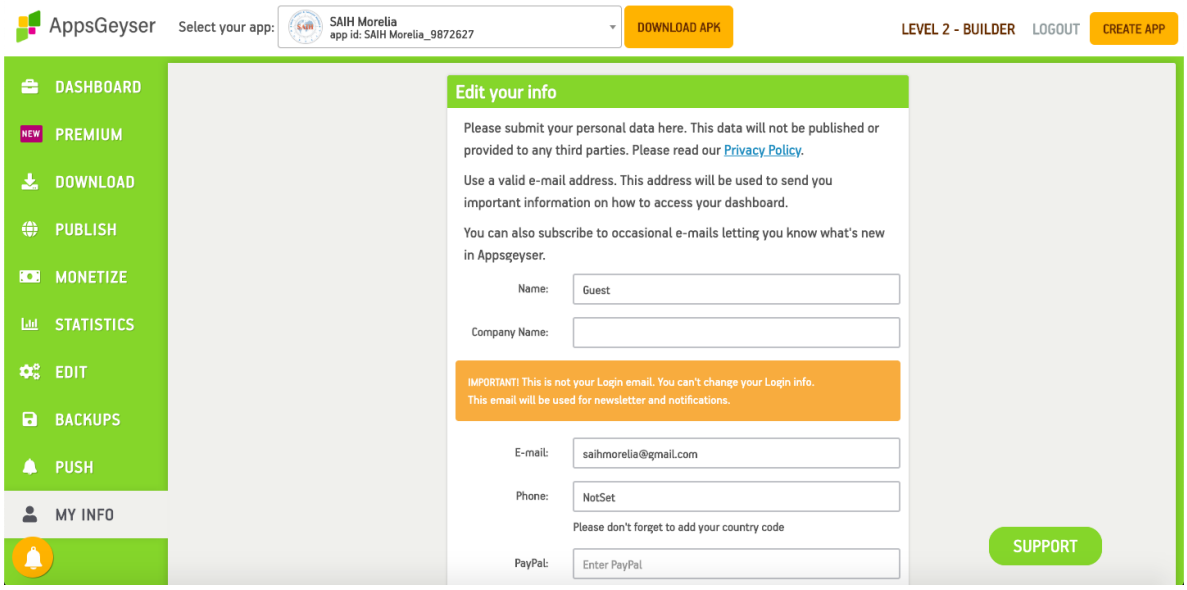


Figura 31: Cuenta AppsGeyser.

Fuente: Daniel Rodriguez Licea

Para el desarrollo de la aplicación elegimos crear una nueva App y después especificamos el tipo de aplicación, en este caso se eligió un tipo blog (figura 32). Finalmente ingresamos el nombre de la aplicación y el URL (SAIH, www.saihmorelia.com) en el que se encuentra el sitio web (figura 33).

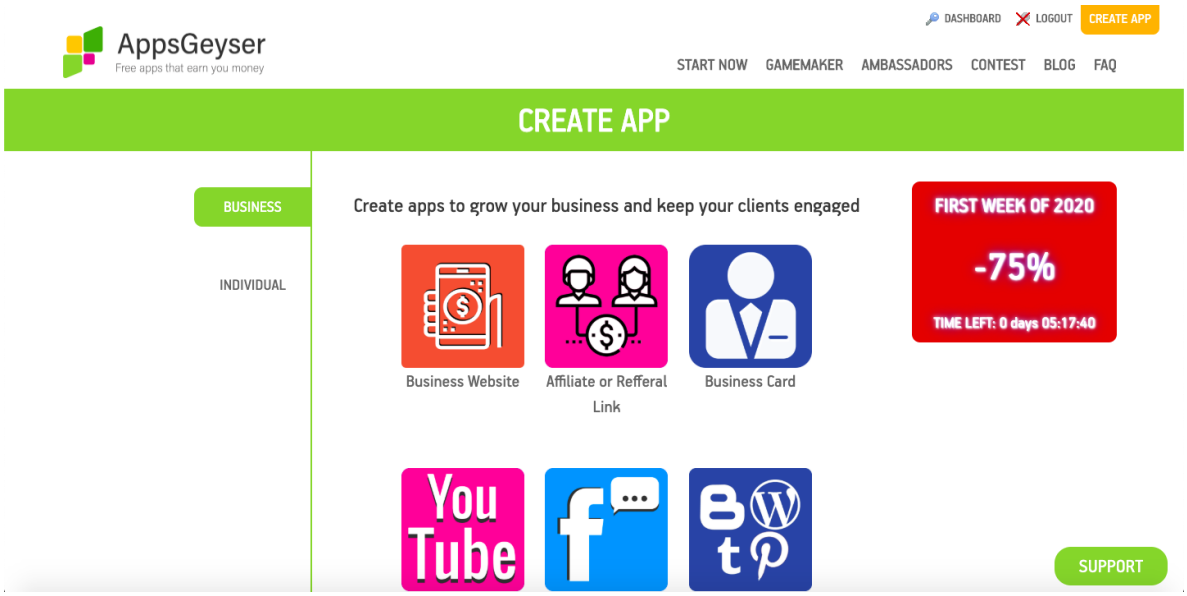


Figura 32: Tipos de aplicación de desarrollo en AppsGeyser.

Fuente: Daniel Rodriguez Licea

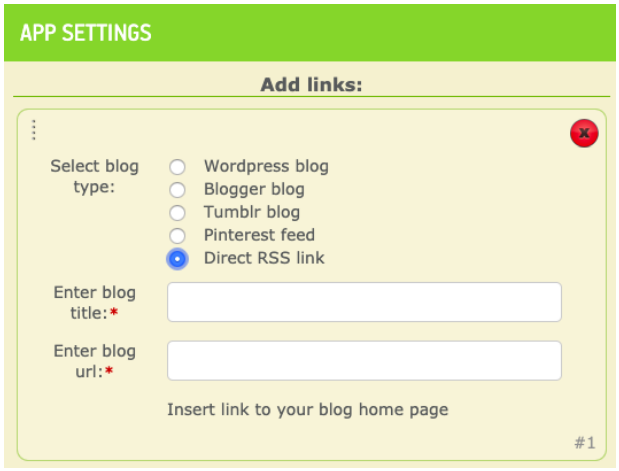


Figura 33: Ingreso de datos para el desarrollo de la App.

Fuente: Daniel Rodriguez Licea

Una vez ingresado los datos y enviada la solicitud de creación de App, AppsGeyser nos redirecciona a una nueva pagina web (figura 34) donde encontramos un código QR y un enlace que nos permiten descargar el .jdk para su posterior instalación el Smartphone.

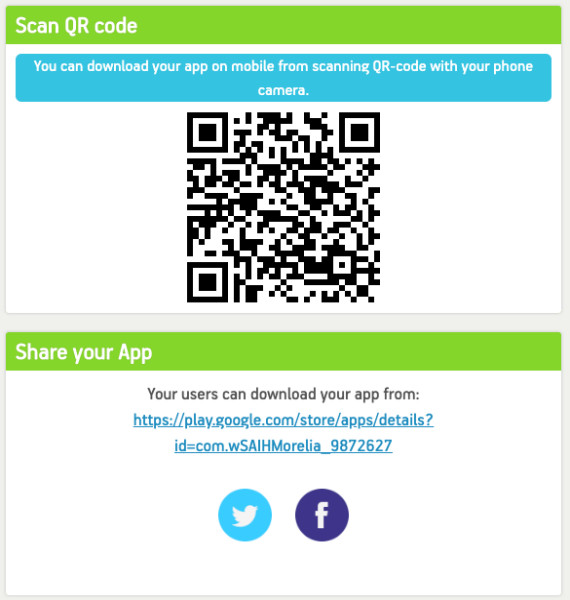


Figura 34: Sitio web de descarga de jdk de la App

Fuente: Daniel Rodriguez Licea

Posteriormente este fichero jdk se subió a Google Play, donde la aplicación paso por revisores de Google para su aprobación como App en la tienda oficial de aplicaciones para Android (figura 35).

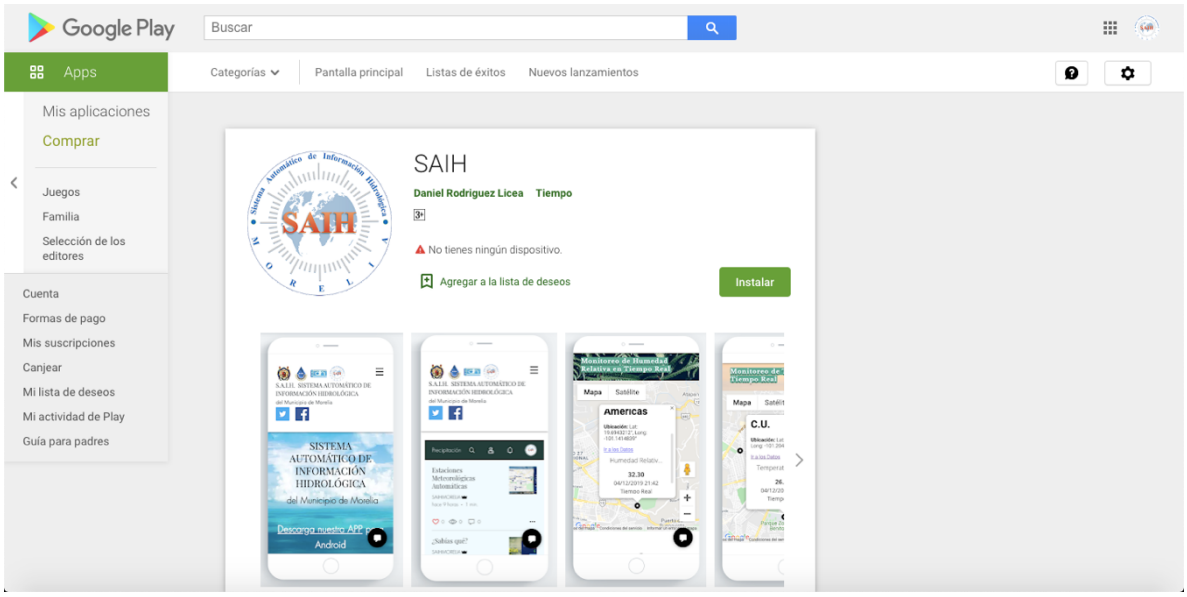


Figura 35: App SAIH en Google Play

Fuente: https://play.google.com/store/apps/details?id=com.wSAIHMorelia_9872627

La funcionalidad del app y la usabilidad de la misma están diseñadas bajo el mismo concepto del sitio web, por ello, la estructura se basa en el mismo concepto del prototipo de baja fidelidad que se explicó en el apartado 8.4.3.1.3: una pantalla principal (por variable) con la red de monitoreo (figura 36), al hacer click sobre los puntos que representan las EMABC devuelve una ventana emergente (figura 37) donde se visualiza la información referente a la estación, la información de las variables en tiempo real y el enlace "ir a los datos" que permite acceder a la información temporal de precipitación (intensidad, volumen), temperatura y humedad del aire, según sea el caso en una pantalla secundaria (figura 38) que además permite la descarga de la información en un fichero CSV.

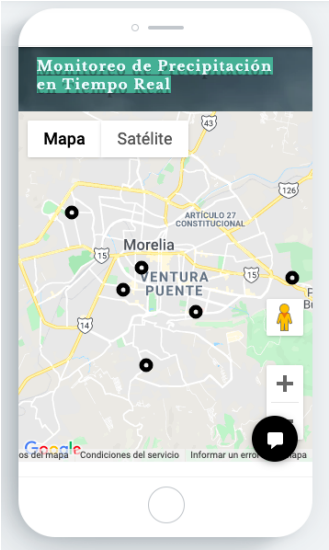


Figura 36: Pantalla principal de la App

Fuente: Daniel Rodriguez Licea


Figura 37: Ventana emergente de la App

Fuente: Daniel Rodriguez Licea

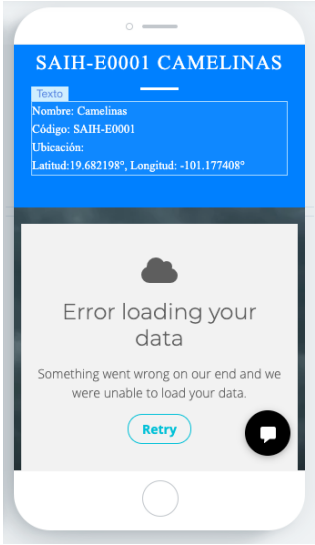


Figura 38: Pantalla secundaria de la App

Fuente: Daniel Rodriguez Licea

### 8.4.3.3 Alertas

Las alertas se emiten cuando la EMABC registra una intensidad de precipitación mayor a 30 mm/h (Lluvia fuerte) con la finalidad de alertar sobre posibles inundaciones.

La emisión de alertas se realiza a través de ThingSpeak o Ubidots usando sus WebHooks con codificación a través de archivos json para el envió de información al servicio IFTTT el cual se encarga de publicar la alerta en las cuentas oficiales de Facebook y Twitter del SAIH a

traves de un post y un tweet respectivamente inmediatamente cuando haya detectado una medición fuera del rango de tolerancia de la variable.

Primeramente se creo una cuenta en el servicio web ifttt.com (figura 39) a través de la cuenta de Gmail del SAIH, accionando la opción “nuevo Applet” (figura 40) en la opción “If This (si esto ocurre) elegimos un WebHooks (figura 41) que se encargará de recibir la información de las alertas enviadas por ThigSpeak o Ubidots, estos enviaran dos tipos de alertas cuando inicie la lluvia u cuando se presente una lluvia superior a los 30mm/h (lluvia fuerte), posteriormente se le asigna un nombre de identificación al WebHooks con referencia al inicio de lluvia o alerta de lluvia fuerte particularmente. Definidos los WebHooks indicamos la acción para el “Then That” (haz esto), elegimos Twitter (figura 42) y posteriormente indicamos la acción de respuesta al WebHooks asignando la publicación de un tweet (figura 43) automáticamente en la cuenta oficial del SAIH (@MoreliaSaih) cuando ocurra cualquier evento ya sea inicio de lluvia o alerta de lluvia fuerte. La estructura de la publicación lleva embebidas variables que toman los datos del mensaje json que envían las bases de datos IoT, en la figura 44 y 45 se muestran las estructuras del mensaje para la publicación Twitter de inicio de lluvia y alerta de lluvia fuerte respectivamente.

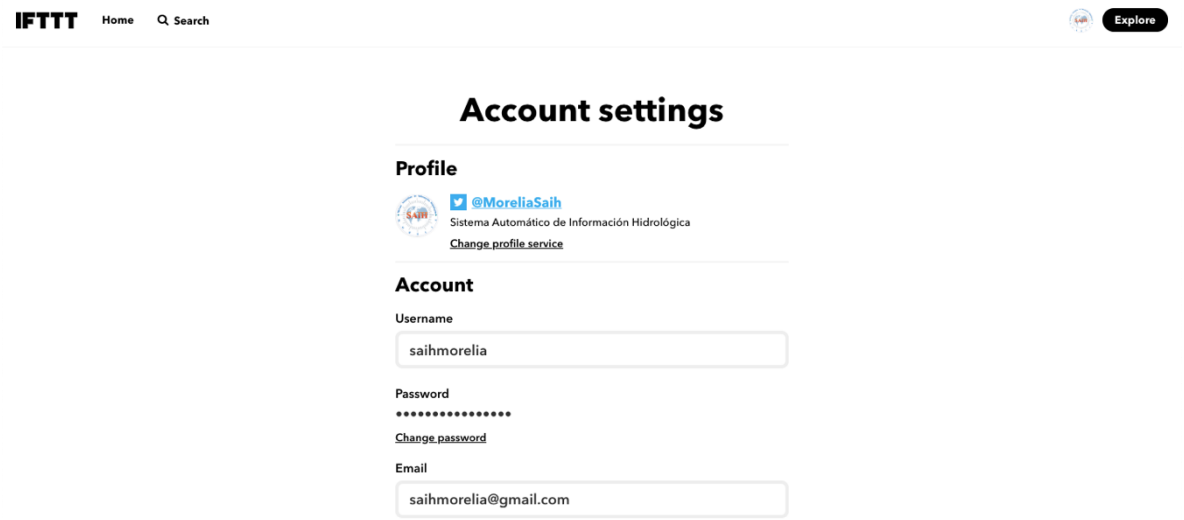


Figura 39: Cuenta en ifttt.com.

Fuente: Daniel Rodriguez Licea

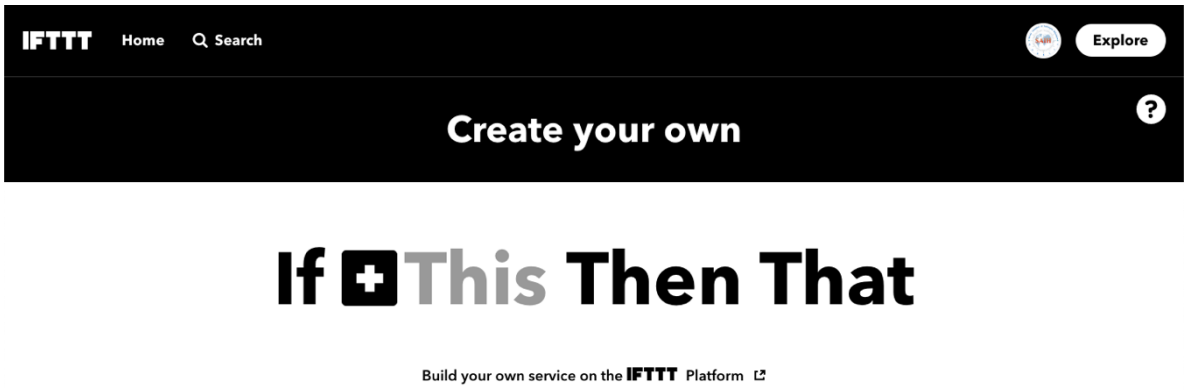


Figura 40: Applet.

Fuente: Daniel Rodriguez Licea

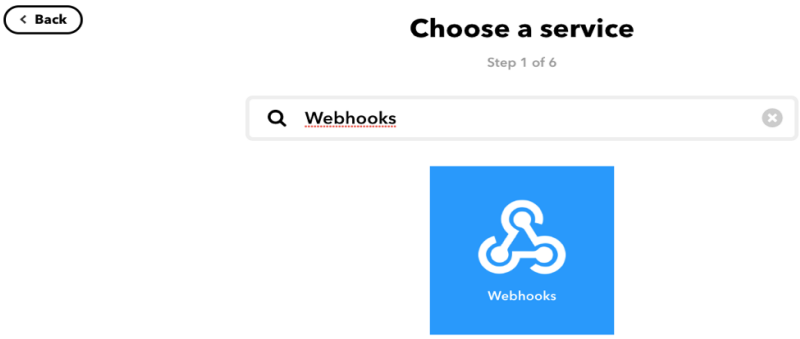


Figura 41:WebHooks.

Fuente: Daniel Rodriguez Licea

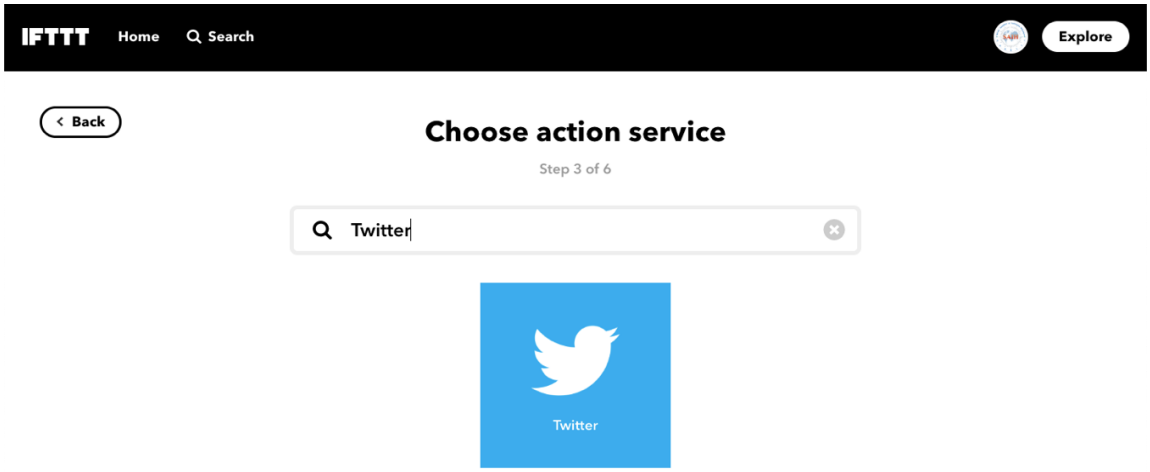


Figura 42: Twitter.

Fuente: Daniel Rodriguez Licea

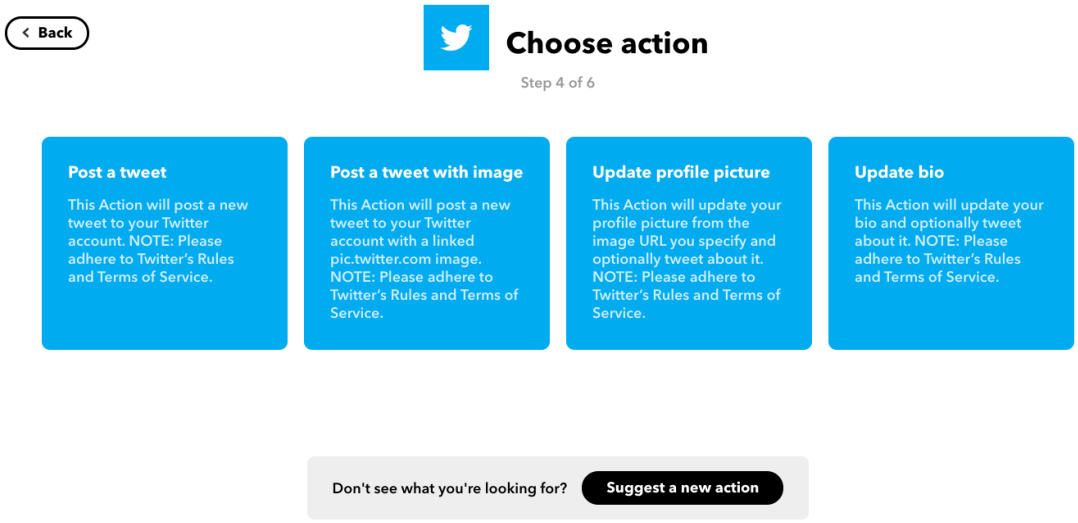


Figura 43: Publicación de Tweet.

Fuente: Daniel Rodriguez Licea


Figura 44: Estructura de tweet de Alerta de inicio de lluvia.

Fuente: Daniel Rodriguez Licea


Figura 45: Estructura de tweet de Alerta de lluvia fuerte.

Fuente: Daniel Rodriguez Licea

Ambas estructuras utilizan la variable 2 y la variable3 para recibir el código y el nombre de la EMABC respectivamente, en el caso de la variable 1 solo se utiliza en las alertas de lluvia fuerte para indicar el valor de la intensidad en el momento.

El análisis de la información se genera en los servicios de las bases de datos a través de la creación de eventos que permiten detectar cuando un valor registrado de intensidad de precipitación es mayor a 0 mm/h y 30 mm/h indicando que la lluvia inició y que existe riesgo por el registro de una lluvia fuerte respectivamente. En el caso Thingspeak se crea primeramente una reacción donde se indica el nombre, la frecuencia de repetición del valor

(inmediatamente), el canal de Thinspeak (depende de la estación), el campo que se analizará (intensidad), la condición lógica (cuando se mayor que), y el valor limite (0 o 30 mm/h). Se crearon dos reacciones una para inicio de lluvia (figura 46) y otra para alerta de lluvia fuerte (figura 47).

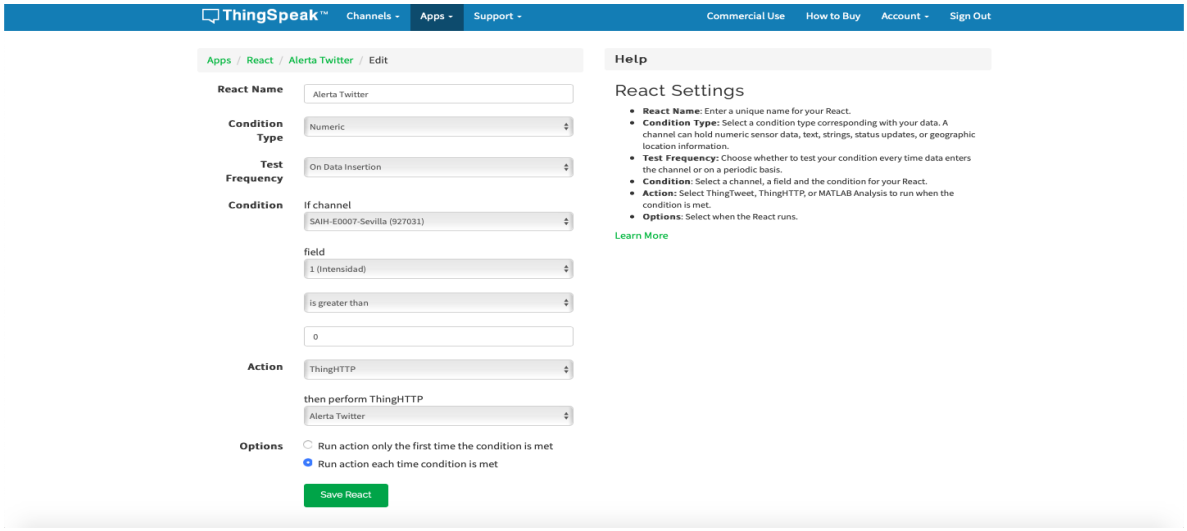


Figura 46: Reacción para inicio de lluvia.

Fuente: Daniel Rodriguez Licea

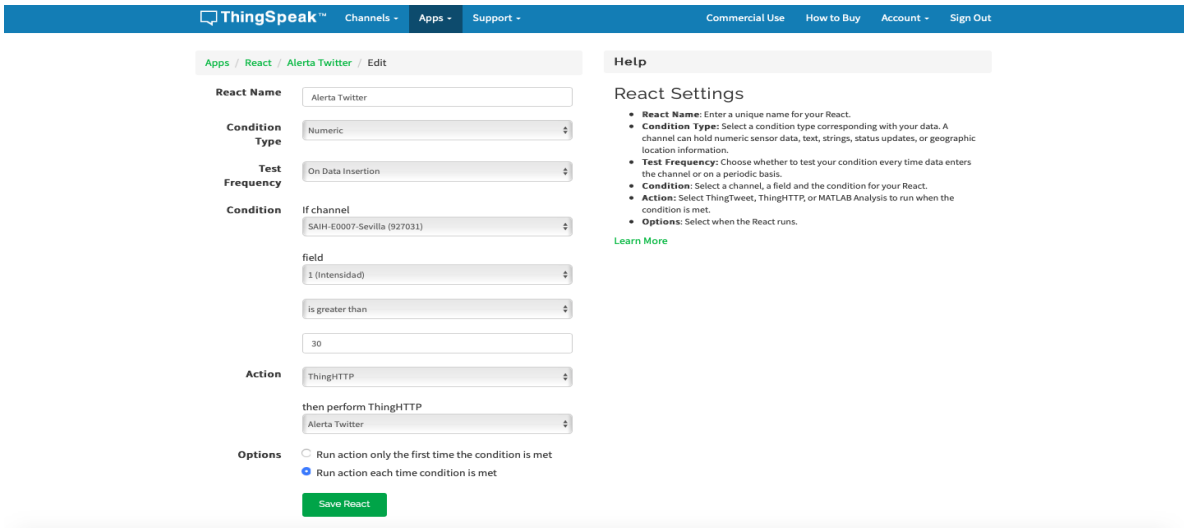


Figura 47: Reacción para alerta de lluvia fuerte.

Fuente: Daniel Rodriguez Licea

Después creamos el ThingHTTP donde se asigna la información de configuración con la cuenta IFTTT indicando el nombre del ThingHTTP, el URL del Webhooks correspondiente, método (publicación) y la cadena en formato json para enviar los datos de la variable 1,2, y 3 que se colocaron en las estructuras de los Tweets, en la figura 48 y 49 se muestra un ejemplo de ThingHTTP para inicio de lluvia y alerta de lluvia fuerte respectivamente.

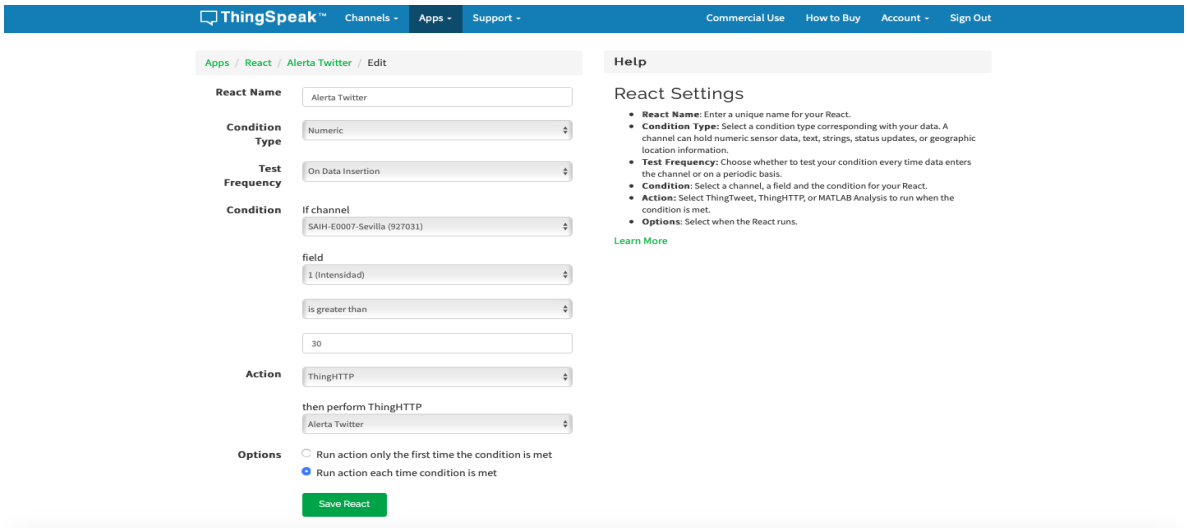


Figura 48: ThingHTTP para inicio de lluvia.

Fuente: Daniel Rodriguez Licea

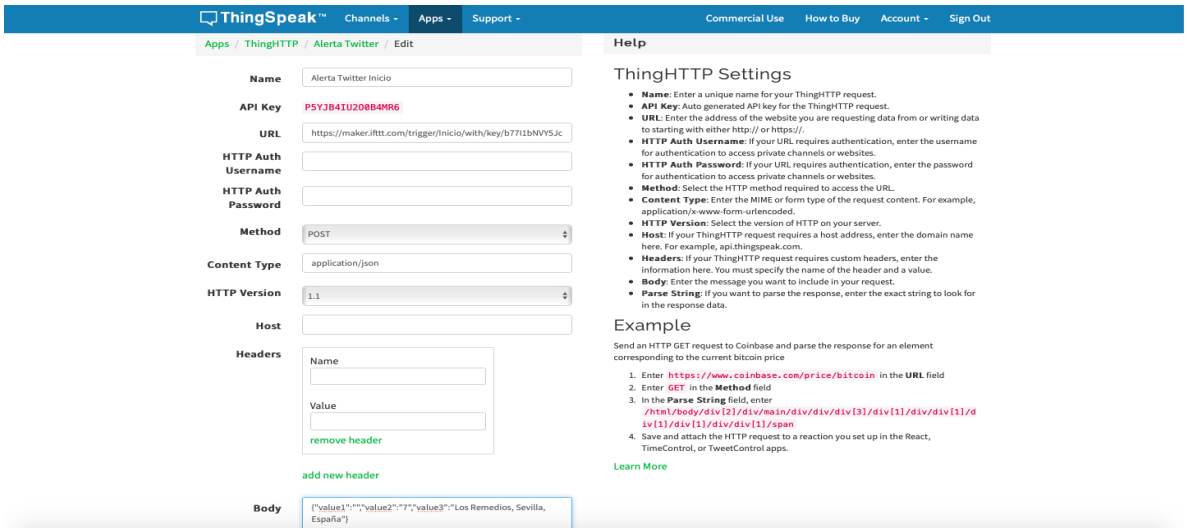


Figura 49: ThingHTTP para alerta de lluvia fuerte.

Fuente: Daniel Rodriguez Licea

En el caso de Ubidots se trata de crear un evento de tipo WebHook (figura 50) donde se indica la variable, la prueba lógica (inicio de lluvia y lluvia fuerte), y la frecuencia de lectura para el envió de alerta en la figura 51 y 52 se muestra un ejemplo de creación de evento tipo WebHook para inicio de lluvia y lluvia fuerte respectivamente. Posteriormente se asigna el URl del Webhook receptor de la cuenta IFTTT y se estructura la información de los valores 1,2, y 3 que están embebidos en el texto de los tweets para ser enviada a través del evento WebHooks de Ubidots, en la figura 53 y 54 se muestra un ejemplo del envío de información a través de una cadena tipo json.

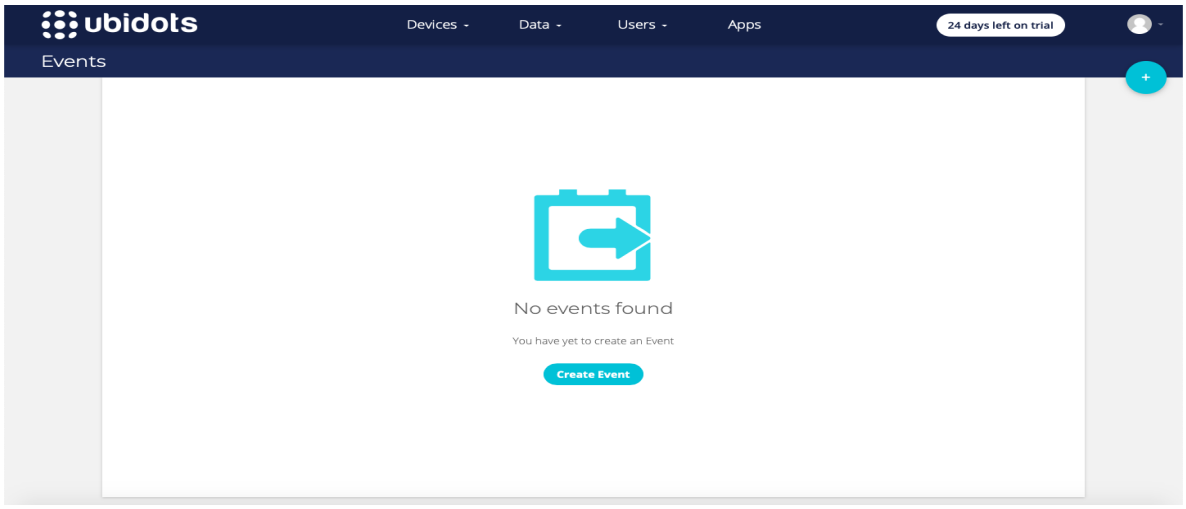


Figura 50: Evento WebHook de Ubidots.

Fuente: Daniel Rodriguez Licea

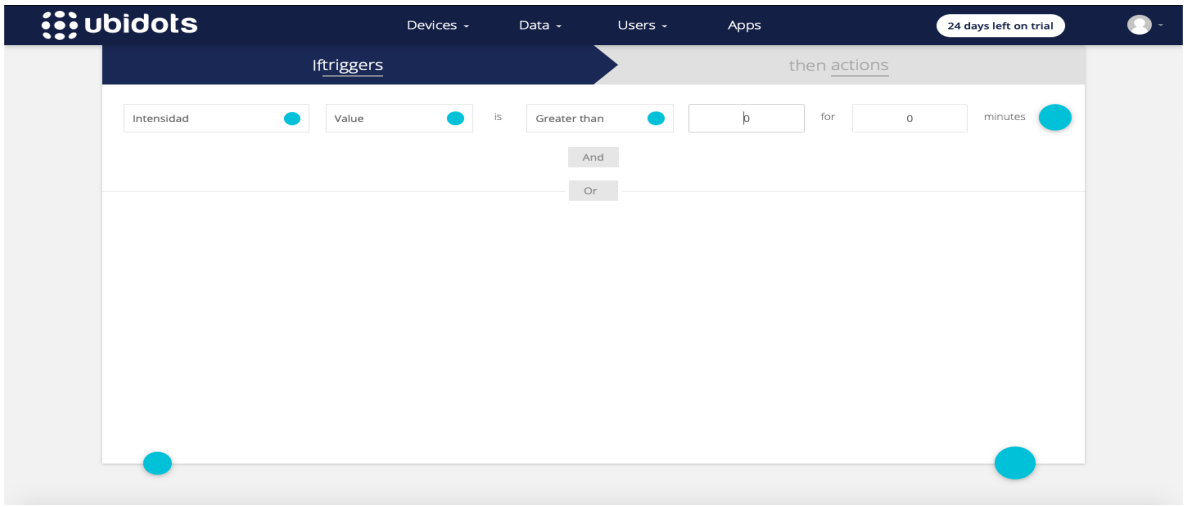


Figura 51: Condiciones para alerta de inicio de lluvia de Ubidots.

Fuente: Daniel Rodriguez Licea

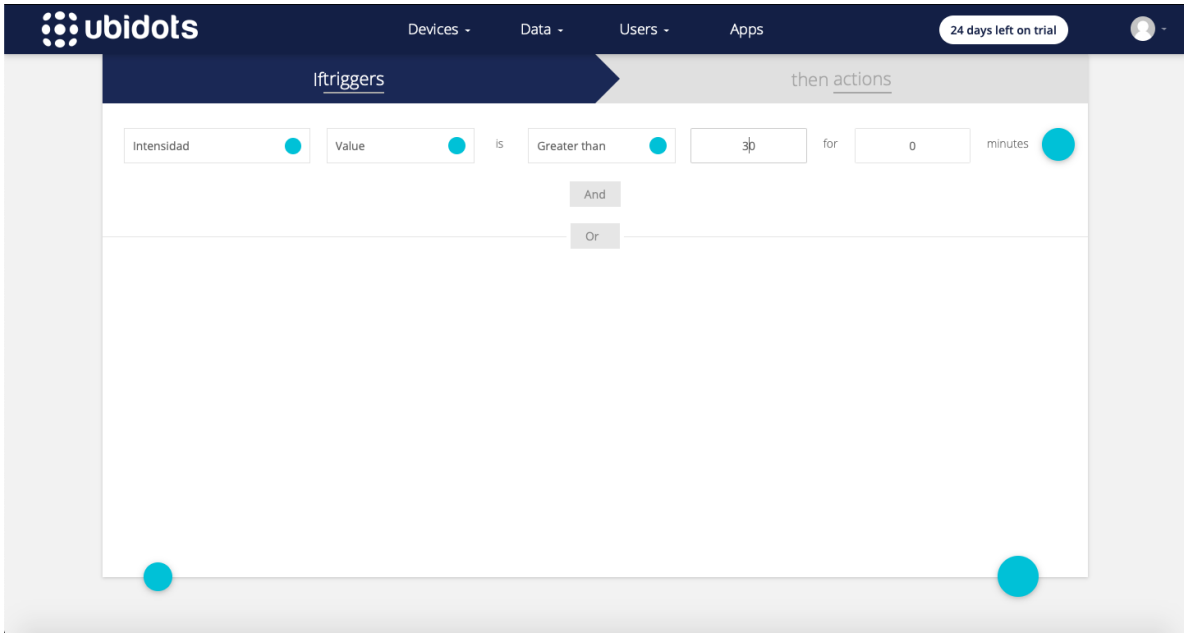


Figura 52: Condiciones para alerta de lluvia fuerte de Ubidots.

Fuente: Daniel Rodriguez Licea

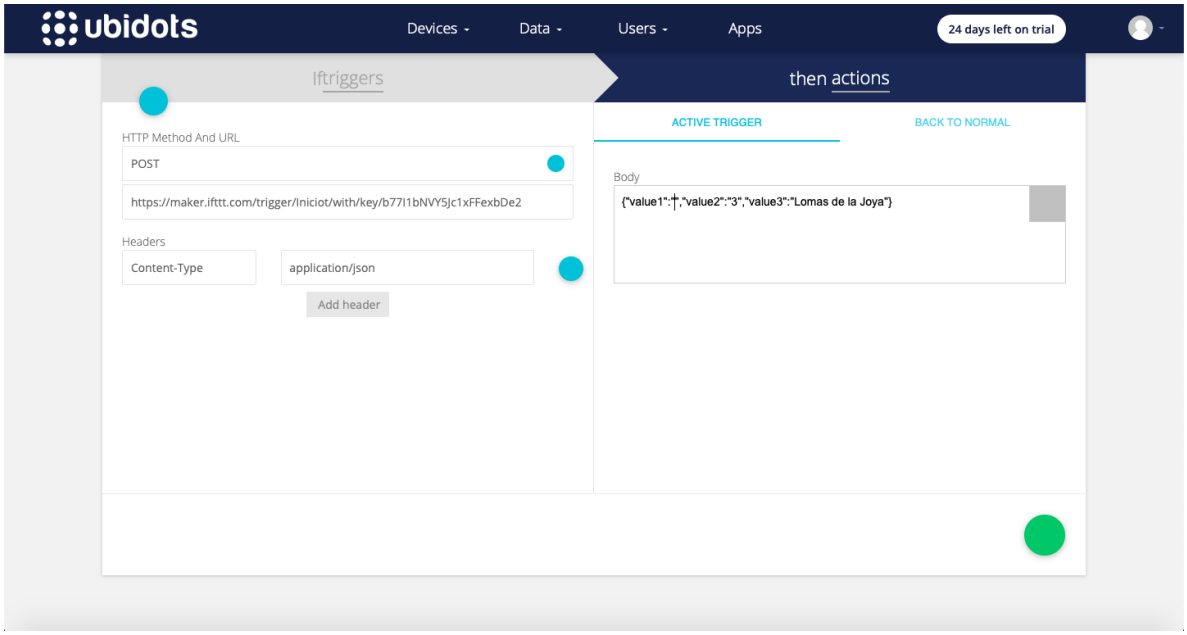


Figura 53: Estructura json para alerta de inicio de lluvia de Ubidots.

Fuente: Daniel Rodriguez Licea

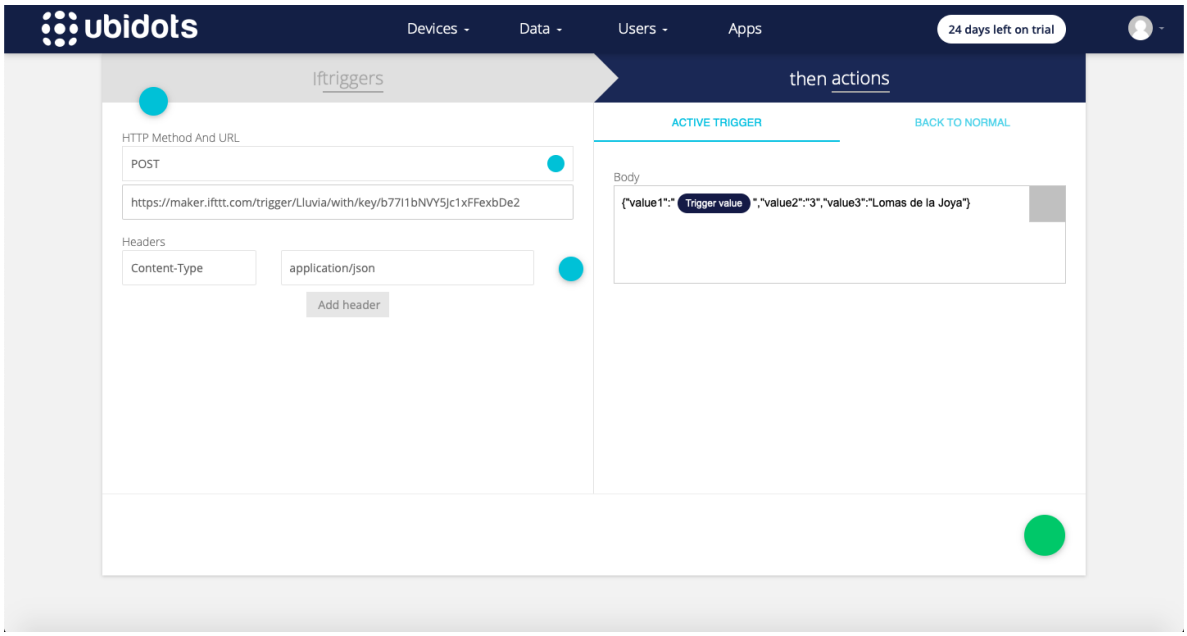


Figura 54: Estructura json para alerta de lluvia fuerte de Ubidots.

Fuente: Daniel Rodriguez Licea

A través de estas herramientas de IFTTT y Twitter se publican alertas automáticamente cuando el servicio de base de datos IoT detecta valores fuera del rango de tolerancia de la precipitación en la figura 55 y 56 se muestran ejemplos de tweets de inicio de lluvia y de alerta de lluvia fuerte respectivamente.


Figura 55: Tweet de Alerta de inicio de lluvia.

Fuente: Daniel Rodriguez Licea


Figura 56: Tweet de Alerta de lluvia fuerte

Fuente: Daniel Rodriguez Licea

Una desarrollado los procedimientos para la automatización de alertas via twitter desarrollamos las de Facebook, para las alertas de Facebook utilizaremos una herramienta de IFTTT que republica todos los tweets de alertas en la pagina Facebook del SAIH. Usamos la herramienta "if new tweet" (figura 57) donde agregamos la cuenta de Facebook (@SAIHMORELIA), después de configurada la herramienta casa que se publique un tweet en la pagina de Twitter este se republicara automáticamente en la pagina de Facebook del SAIH. En la figura 58 y 59 se muestra un ejemplo de publicaciones en Facebook de inicio de lluvia y alerta de lluvia fuerte respectivamente.


Figura 57: If new Tweet.

Fuente: Daniel Rodriguez Licea

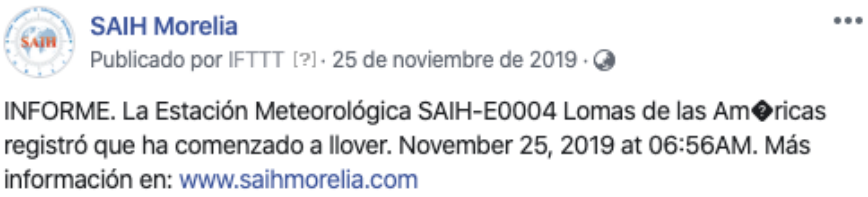


Figura 58: Publicación de inicio de lluvia en Facebook.

Fuente: Daniel Rodriguez Licea


Figura 59: Publicación de Alerta de lluvia fuerte en Facebook.

Fuente: Daniel Rodriguez Licea

### 8.5 Desarrollo y documentación del software

En este apartado se describirán los algoritmos utilizados en los microcontroladores de las EMABC's con comunicación Wifi y GSM. Los algoritmos van desde la obtención de información de precipitación, temperatura y humedad del aire en tiempo real hasta el envió a la base de datos IoT. Dentro esta investigación hemos considerado la utilización de dos bases de datos IoT: ThingSpeak y Ubidots, los algoritmos mantienen la misma estructura, pero la forma de escribir los datos se codifica diferente, en el anexo 1, 2, 3, y 4 se muestran los algoritmos comentados para ThingSpeak con EMABC Wifi y GSM, y con Ubidots con EMABC Wifi y GSM respectivamente.

#### 8.5.1 Algoritmo de la EMABC Wifi

El funcionamiento de las EMABC Wifi se basa en la estructura del algoritmo que se muestra en la figura 60, este obtiene información de precipitación (intensidad, volumen), temperatura y humedad del aire en tiempo real y la envía a la base de datos a cada minuto. Parte de la definición de librerías como la librería DTH para el funcionamiento del sensor de temperatura y humedad del aire, y la librería ESP8266WiFi para la conexión a la red de internet. La definición de pines digitales para la conexión de los sensores, en estos se recibe la señal de los sensores que después leerá y procesará el microcontrolador. Dentro de la declaración de variables existen variables principales las cuales guardan los resultados de las mediciones de los sensores, variables secundarias que contienen información de redes de internet y claves api de los servidores de las bases de datos, y variables complementarias que se utilizan para guardar factores de conversión o hacer algunos cambios de variable necesarios.

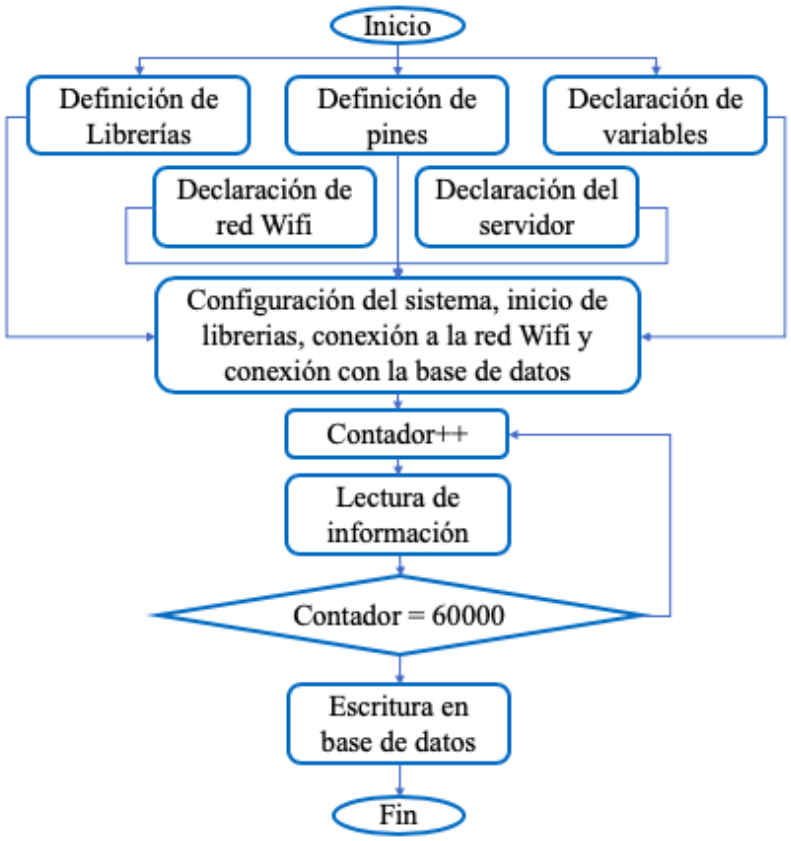


Figura 60: Diagrama de flujo del algoritmo EMABC Wifi.

Fuente: Daniel Rodriguez Licea

La declaración de la red contiene dos variables de tipo carácter que guardan el nombre de la red y su contraseña de acceso. La declaración del servidor contiene dos variables una que almacena una clave API de autentificación para el servidor y una con la dirección URL del servidor. El siguiente paso se encarga de iniciar el sistema, corriendo las librerías, identificando las variables, estableciendo la conexión Wifi y con el servidor. Una vez iniciado el sistema inicia un contador de tiempo en milisegundos, posteriormente la medición de precipitación (intensidad y volumen) recibiendo un pulso eléctrico a través del pin D8 cada vez que cae una de las cubetas del balancín y contándose en una variable que se resetea cada 60000 milisegundos (1 minuto), previamente al reseteo de esta ultima se convierte la cantidad de pulsos que se han sumado durante los 60000 milisegundos a volumen de lluvia multiplicando el contador de pulsos por un factor de conversión con valor de 0.2 mm/pulso, guardándose el resultado en una variable llamada precipitación en mm, posteriormente

generamos una variable llamada intensidad que guarda el volumen de precipitación en un minuto interpretándose como la intensidad de lluvia en mm/min. La temperatura y humedad se obtiene a través de la librería DHT devolviendo la temperatura del aire en grados Celsius y humedad relativa del aire en porcentaje, y guardándola en variables. Teniendo las mediciones de precipitación (intensidad y volumen), temperatura y humedad del aire, el siguiente paso es enviarlos a la base de datos previo a una sentencia de control que evalúa si el contador de tiempo es igual a 60000 milisegundos si esto es verdadero envía la información de las variables a la base de datos y después reinicia el contador de tiempo, en caso contrario aumenta una unidad mas al contador y genera una nueva medición hasta que el tiempo se cumpla para enviar la información.

### 8.5.2 Algoritmo EMABC GSM

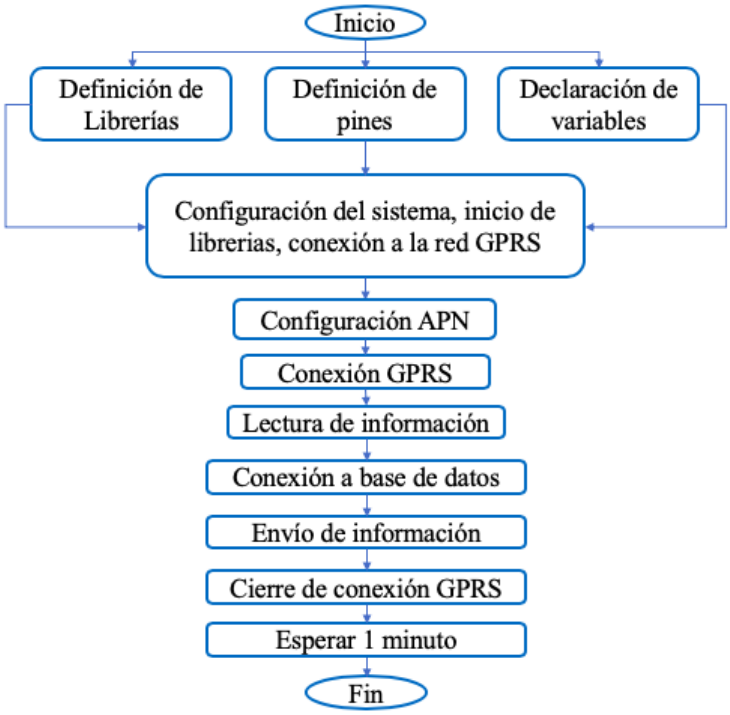


Figura 61: Diagrama de flujo del algoritmo EMABC GSM.

Fuente: Daniel Rodriguez Licea

El funcionamiento de las EMABC GSM se basa en la estructura del algoritmo que se muestra en la figura 61, este obtiene información de precipitación (intensidad, volumen), temperatura

y humedad del aire en tiempo real y la envía a la base de datos a cada minuto. Parte de la definición de librerías como la librería DTH para el funcionamiento del sensor de temperatura y humedad del aire, la librería String para manipular cadenas de texto y la librería SoftwareSerial para la comunicación del modulo GSM SIM900 y el Arduino UNO. La definición de pines digitales para la conexión de los sensores, en estos se recibe la señal de los sensores que después leerá y procesará el microcontrolador. Dentro de la declaración de variables existen variables principales las cuales guardan los resultados de las mediciones de los sensores, variables secundarias que contienen información de redes de internet y claves api de los servidores de las bases de datos, y variables complementarias que se utilizan para guardar factores de conversión o hacer algunos cambios de variable necesarios. El siguiente paso se encarga de iniciar el sistema, corriendo las librerías, identificando las variables, estableciendo la conexión GSM. La configuración APN se realiza mediante comandos AT embebidos en el código donde se declara el acceso APN, el usuario y contraseña de este a través de caracteres. Con los datos anteriores se establece la comunicación GPRS en el modulo SIM900, cuando se tiene esta conexión estable se sigue ejecutando el código. Posteriormente la medición de precipitación (intensidad y volumen) recibiendo un pulso eléctrico a través del pin D8 cada vez que cae una de las cubetas del balancín y contándose en una variable que se resetea cada 60000 milisegundos (1 minuto), previamente al reseteo de esta ultima se convierte la cantidad de pulsos que se han sumado durante los 60000 milisegundos a volumen de lluvia multiplicando el contador de pulsos por un factor de conversión con valor de 0.2 mm/pulso, guardándose el resultado en una variable llamada precipitación en mm, posteriormente generamos una variable llamada intensidad que guarda el volumen de precipitación en un minuto interpretándose como la intensidad de lluvia en mm/min. La temperatura y humedad se obtiene a través de la librería DHT devolviendo la temperatura del aire en grados Celsius y humedad relativa del aire en porcentaje, y guardándola en variables. Teniendo las mediciones de precipitación (intensidad y volumen), temperatura y humedad del aire, el siguiente paso es establecer la conexión con la base de datos mediante un nuevo comando AT y después enviarlos a la base de datos. Enviados los datos la conexión GPRS se desconecta y el sistema se mantiene en espera durante un minuto para volver a iniciar el algoritmo.

## 8.6 Prueba y mantenimiento del sistema

En este apartado se desarrollan las pruebas de precisión de los sensores DTH22 y del pluviógrafo de balancín, y las calibraciones adecuadas para cada sensor.

### 8.6.1 Prueba de precisión y calibración del sensor DHT22

La prueba de precisión del sensor se realizo mediante la comparación de observaciones con estación cerca colocando el sensor DHT22 al lado de una estación comercial de marca NETATMO, durante un periodo del día 10 al 21 de noviembre de 2019, capturando información a cada minuto. Finalizando el periodo de obtención de información se descargaron los datos y graficaron los datos de temperatura (figura 62) como de humedad relativa del aire (figura 63), con la finalidad de analizar el comportamiento de ambos dispositivos respecto uno del otro.

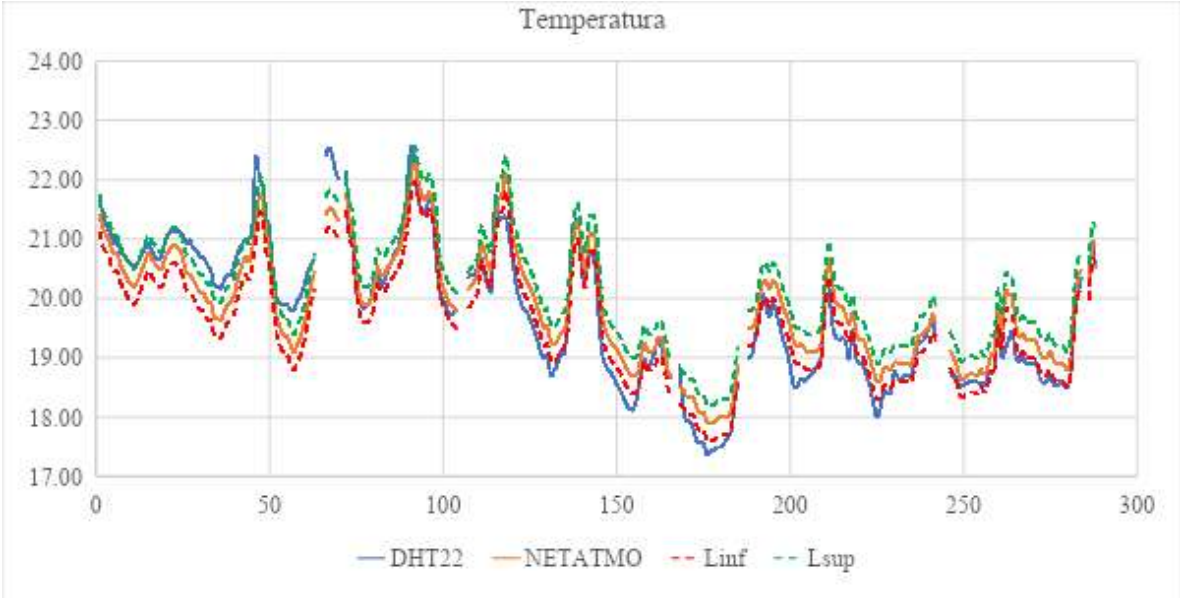


Figura 62:Temperatura, DHT22 Vs NETATMO.

Fuente: Daniel Rodriguez Licea

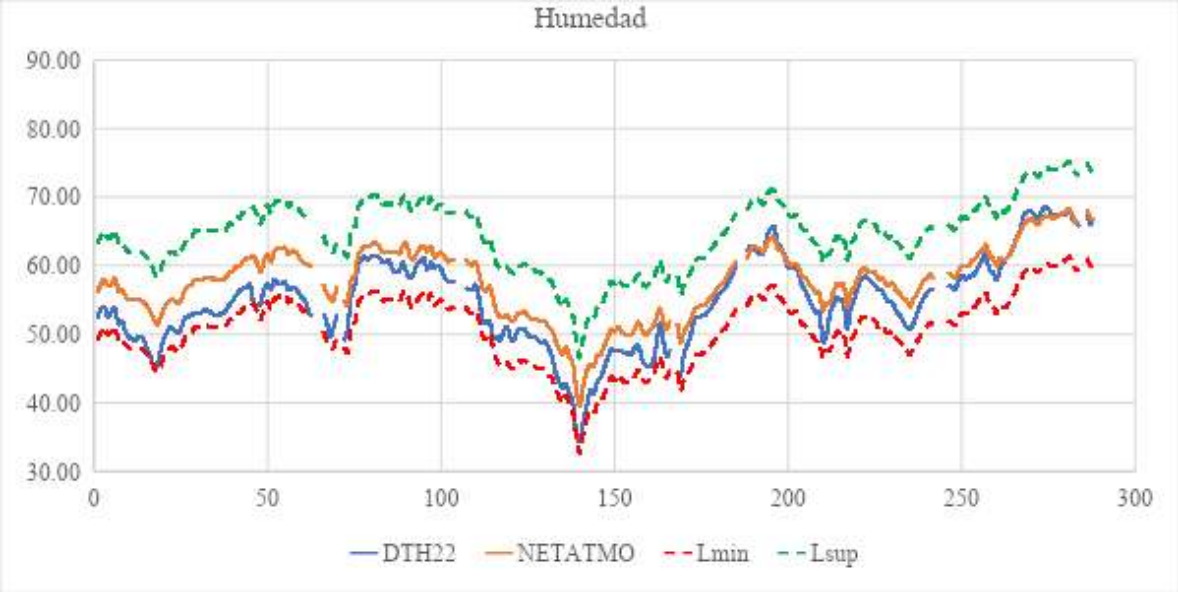


Figura 63:Humedad relativa, DHT22 Vs NETATMO.

Fuente: Daniel Rodriguez Licea

Ambas series se analizaron para obtener el error promedio en el periodo de tiempo de medición. El error promedio obtenido es de 0.19% en la serie de temperatura y de 0.87% en la serie de Humedad relativa.

El sensor DTH22 tiene un rango de medición de -40 ºC a 80 ºC y 0% a 100% en temperatura y humedad relativa respectivamente. Considerando el valor absoluto mayor del rango de temperatura, en este caso 80 ºC obtendremos el máximo error posible en las mediciones de temperatura considerando el error promedio calculado, resultando 0.15 ºC. De la misma manera calculamos el error máximo posible para humedad relativa, resultando 0.87%.

### 8.6.2 Prueba de precisión y calibración del pluviógrafo de balancín

Para la obtención de la precisión del pluviógrafo de balancín se utilizaron en conjunto la metodología de calibración de sensores por climas controlados y el método de validación de sensores por comparación de observaciones con estación cercana.

El pluviógrafo de balancín de la EMABC junto con la estación Davis Ventage Pro2 (estación comercial) se sometieron a un clima controlado de precipitación simulando dos eventos con intensidad de 12 mm/h y 36 mm/h obteniendo mediciones a cada minuto con ambos equipos. Se inició el clima controlado de 12 mm/h y la medición con la estación Davis Ventage Pro2

durante 4 horas (figura 64) posteriormente el pluviógrafo de balancín durante 8 horas (figura 65). Obtenidas las mediciones se han comparado los resultados del pluviógrafo de balancín respecto de la estación Davis Ventage Pro2, resultando 0.03 mm de variación en las mediciones de volumen de precipitación de ambos equipos.

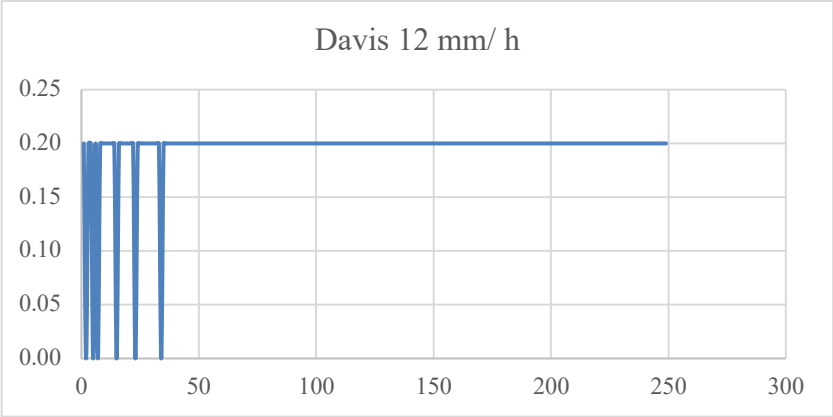


Figura 64. Mediciones de la estación Davis Ventage Pro2, intensidad: 12 mm/h.

Fuente: Daniel Rodriguez Licea

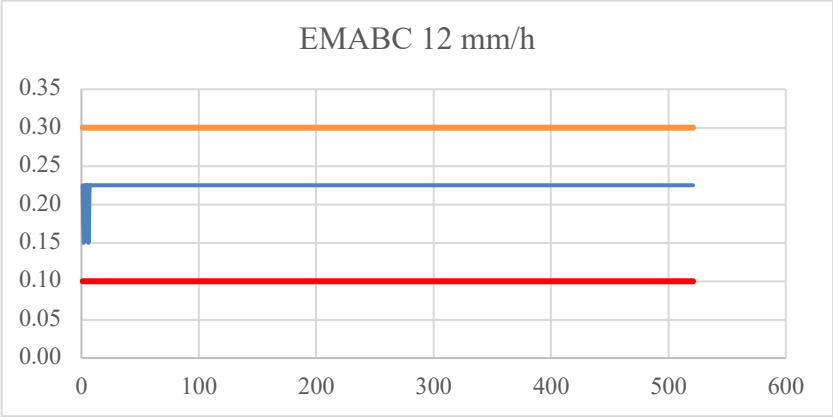


Figura 65. Mediciones del pluviógrafo de balancín, intensidad: 12 mm/h.

Fuente: Daniel Rodriguez Licea

Posteriormente se ha puesto en marcha el clima controlado con intensidad de 36 mm/h y se inició la medición con la estación Davis Ventage Pro2 durante 40 horas (figura 66) y posteriormente el pluviógrafo de balancín durante 12 horas (figura 67). Obtenidas las mediciones se han comparado los resultados del pluviógrafo de balancín respecto de la

estación Davis Ventage Pro2, resultando 0.00 mm de variación en las mediciones de volumen de precipitación de ambos equipos.

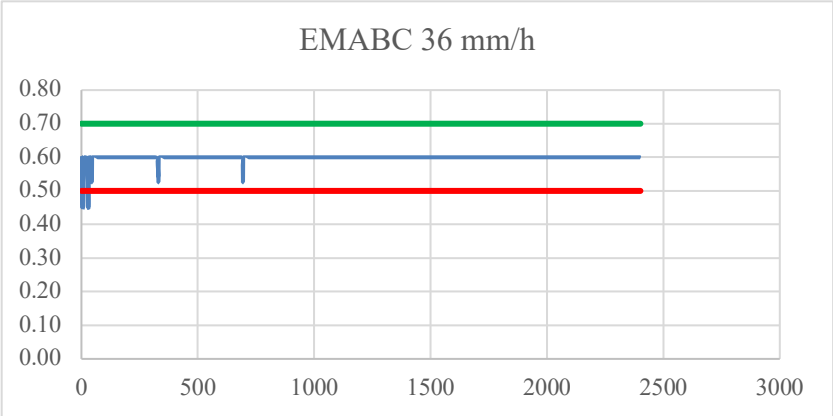


Figura 66. Mediciones de la estación Davis Ventage Pro2, intensidad: 36 mm/h.

Fuente: Daniel Rodriguez Licea

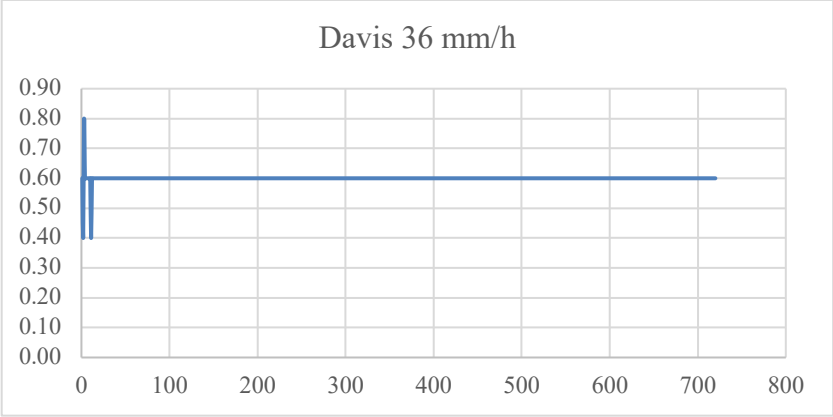


Figura 67. Mediciones del pluviógrafo de balancín, intensidad: 36 mm/h.

Fuente: Daniel Rodriguez Licea

## 8.7 Implementación y evaluación del sistema

En este apartado se presenta la evaluación de la información recogida por las EMABC y la implementación del sistema. La Norma Mexicana NMX-AA-166/1-SCFI-2013 establece la precisión requerida en las mediciones de precipitación, temperatura y humedad por una EMA, basándonos en lo anterior y los resultados de las pruebas de precisión y calibración de los sensores de la EMABC, se evalúa la precisión de la EMABC en función de la norma.

### 8.7.1 Validación del sensor DHT22

La precisión de una EMA (estación meteorológica automática) en la medición de temperatura y humedad relativa del aire lo establece la Norma Mexicana NMX-AA-166/1-SCFI-2013 y sus valores son: ±0.2 ºC y ±2% respectivamente, tomando en cuenta lo anterior evaluamos los resultados de precisión del sensor DHT22 para ambas variables que se obtuvieron en el apartado 8.6.1.

| **Variable** | **DHT22 (Error máximo)** | **NMX-AA-166/1-SCFI-2013** | **Evaluación** |
|---|---|---|---|
| Temperatura | 0.15 ºC | ±0.2 ºC | Válido |
| Humedad Relativa | 0.87% | ±2% | Valido |

Tabla 3: Evaluación de la EMABC en temperatura y humedad.

Fuente: Daniel Rodriguez Licea

### 8.7.2 Validación del pluviógrafo de balancín

La precisión de una EMA (estación meteorológica automática) en la medición de precipitación lo establece la Norma Mexicana NMX-AA-166/1-SCFI-2013 y su valor es de ±0.1 mm, tomando en cuenta lo anterior evaluamos los resultados de precisión del pluviógrafo de balancín siendo un error máximo de 0.03 mm obtenido en el apartado 8.6.2, resultado valida la información que devuelve este equipo.

## 8.8 Construcción de la red de monitoreo

Diseñada la EMABC se replicó alcanzando las 6 unidades las cuales fueron instaladas en lugares estratégicos en la zona de estudio formando una red en la ciudad de Morelia, en la figura 68 se muestra el mapa de distribución de las EMABC.


Figura 68: Red de monitoreo.

Fuente: Daniel Rodriguez Licea

### 8.9 Sistema Automático de Información Hidrológica

En este apartado se presentan los resultados ya integrados de tal forma que se aprecia la vinculación y funcionalidad de cada elemento que constituye al SAIH. En la figura 69 se muestra un SAIH capaz de recoger información de precipitación, temperatura y humedad del aire en tiempo real a través de EMABC y enviarla a ThingSpeak que permite la captura y el análisis de los datos, esta además es capaz de enviar alertas a través de Webhooks cuando existen mediciones fuera del rango de tolerancia de intensidad de precipitación y publicarlas en Facebook y Twitter. ThingSpeak permite compartir la información en sitios web y Apps donde se visualiza la información temporal y espacialmente. El tiempo de actualización de información del SAIH es de un minuto, esto significa que desde la medición de las variables hasta la visualización en el sitio web o App pasa un minuto.

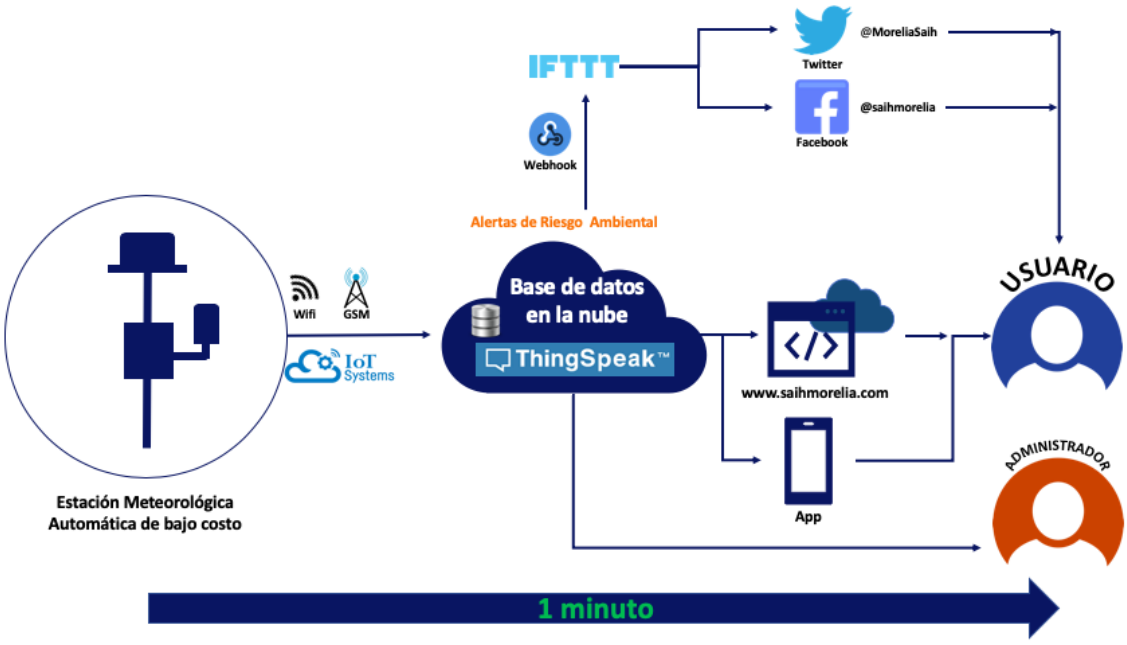


Figura 69: SAIH basado en ThingSpeak.

Fuente: Daniel Rodriguez Licea

En la figura 70 se muestra un SAIH capaz de recoger información de precipitación, temperatura y humedad del aire en tiempo real a través de EMABC y enviarla a Ubidots que permite la captura y el análisis de los datos, esta además es capaz de enviar alertas a través de Webhooks y SMS cuando existen mediciones fuera del rango de tolerancia de intensidad de precipitación y publicarlas en Facebook y Twitter. Ubidots permite compartir la información en sitios web y Apps donde se visualiza la información temporal y espacialmente además de emitir reportes de actividad al usuario administrador cuando existen anomalías en las EMABC. El tiempo de actualización de información del SAIH es de un minuto, esto significa que desde la medición de las variables hasta la visualización en el sitio web o App pasa un minuto.

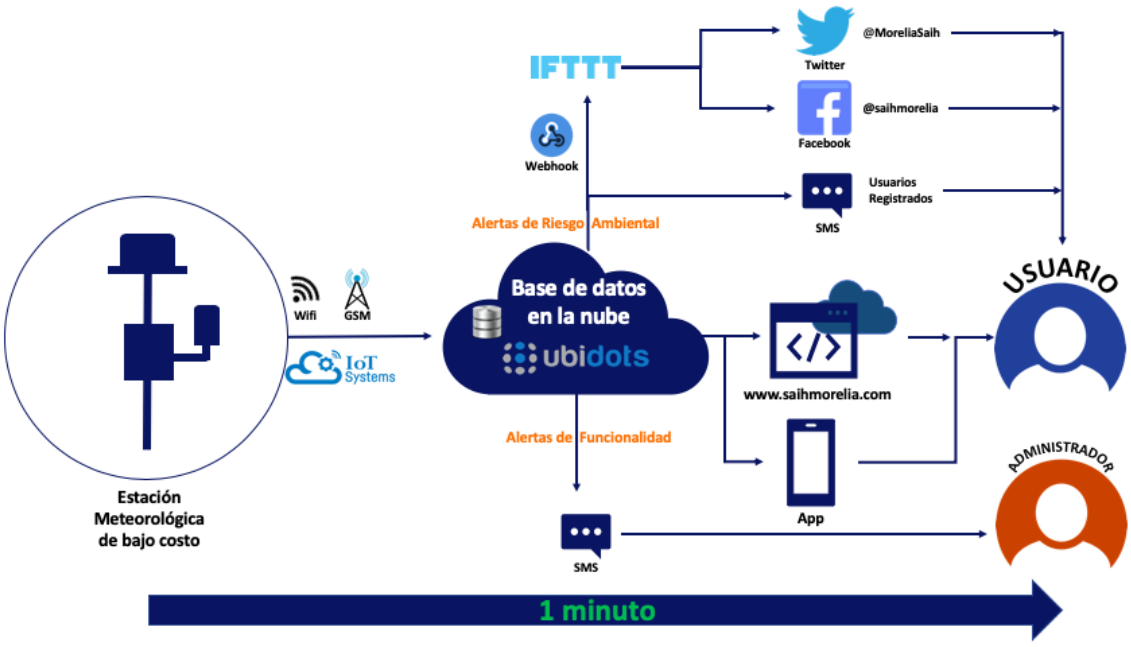


Figura 70: SAIH basado en Ubidots.

Fuente: Daniel Rodriguez Licea

## 9. Discusión de resultados

Como zona de estudio se ha definió la ciudad de Morelia con la finalidad de llevar la aplicación de los SAIH en un entorno donde la sociedad pueda ser beneficiada a través de la información que este genera.

La caracterización de la zona de estudio se realizó con el objetivo de identificar los sistemas de información que actualmente se encuentran en la zona los cuales posteriormente fueron analizados principalmente bajo criterios de caracterización de los sistemas de alerta temprana revisando que estos cumplieran con las características que forman a estos como (Farid et al., 2017) lo mencionan. Con base a los resultados mostrados en la tabla 1 observamos que los tres sistemas no cumplen ampliamente con las características. Hablando de manera global de estos sistemas podemos decir que la funcionalidad de estos no se acerca a la de un SAIH debido a que no monitorean en tiempo real, no emiten alertas tempranas, no generan bases de datos históricas, no es posible obtener la información, sus dispositivos de adquisición de datos son costosos y finalmente la visualización de la información no hace uso de SIG lo cual vuelve a la información difícil de identificar espacialmente.

Hoy en día existen tecnologías de bajo costo lo cual puede aprovecharse para el desarrollo de dispositivos de medición, además de la diversidad de las bases de datos IoT que pueden implementarse junto con los dispositivos de medición para la generación de ficheros de información continua y en tiempo real. Incorporando estas tecnologías es posible utilizar gestores para el desarrollo de sitios web donde la información pueda ser visualizada además de la incorporación de las Apis de google para la visualización temporal y espacial.

En el apartado 8.2 se explican los requerimientos humanos de los usuarios que se sirven de estos sistemas y de su información, particularmente sus necesidades son diversas que dependen del objetivo del uso de estos, ya sea para gestión de riesgo, planeación o una simple consulta del clima, no obstante, es posible uniformizar los requerimientos a un usuario global de tal forma que pueda ser analizado como usuario único.

Analizando los problemas, oportunidades y requerimientos humanos es posible detallar una propuesta la cual, de solución a la problemática planteada, se ha determinado el desarrollo de un SAIH el cual incorpore la funcionalidad de sistema de alerta temprana. Esta propuesta como tal no es una propuesta de reingeniería de los sistemas actuales si no una propuesta de desarrollo de un sistema alterno basado en las características de un sistema automático de información conjuntas de un sistema de alerta temprana.

El desarrollo del sistema consta de tres puntos principales que son: la creación de un dispositivo de medición, la implementación de una base de datos IoT y el desarrollo de plataformas de visualización.

Primeramente, se inició con la creación del dispositivo de medición, donde previo de un análisis se identificó que para el monitoreo de una zona por remota que esta sea debe poderse instalar una red de EMABC’s que devuelvan información a una base de datos con la finalidad de conocer el comportamiento del clima del sitio y además de alertar sobre riesgos ambientales posibles. Considerando lo anterior se ha definido el desarrollo de dos EMABC’s una con comunicación Wifi y la otra con comunicación GSM de tal forma que donde no exista la posibilidad de conectarse a la red de internet pueda utilizarse la red GSM para enviar la información medida por las EMABC’s.

En el apartado 7.2.2 se describe la metodología para la creación de prototipos la cual fue utilizada para el desarrollo de las EMABC's, la metodología involucra un proceso iterativo para lograr un producto funcional. Para el desarrollo de la EMABC Wifi se realizaron cuatro iteraciones las cuales fueron suficientes para obtener un producto funcional. La iteración 1 se desarrolló a través de la incorporación de sensores y placas sin previo análisis, al igual que el sistema de alimentación eléctrica autosustentable, con la finalidad de experimentar con las diversas placas y sensores, a través de esto se abrió camino al perfeccionamiento del dispositivo.

Tomando en cuenta los requerimientos para el desarrollo de la EMABC Wifi y los resultados de las pruebas de la iteración 1, se rediseña el prototipo incorporando nuevos microcontroladores de tal forma que los problemas del dispositivo de la iteración 1 puedan resolverse, por ello se incorporan las placas NodeMCU ESP8266, estas placas incluyen un modulo de Wifi, que hace al dispositivo mas compacto electrónicamente, estabiliza la conexión Wifi para una escritura continua en la base de datos. Estos desarrollos llevan implícito un tanto de seguimiento de investigaciones relacionas de tal forma que facilita la mejora del dispositivo ya que se utilizan componentes recomendados por otros investigadores. Con la incorporación de estos nuevos microcontroladores y la sustitución de los módulos de Wifi y reloj en tiempo real el dispositivo es menos costoso, además de requerir menos energía eléctrica para su funcionamiento. Al reducir el consumo de energía eléctrica el sistema de alimentación eléctrica autosustentable no se rediseña con el objetivo de probar si es suficiente para este, no obstante, resulta lo contrario. En las pruebas de funcionamiento de la iteración 1 no se realizan análisis de sensibilidad de lo sensores por ello no se aprecia el mal funcionamiento del sensor YFS-201, hasta la iteración dos que se prueba con distintos tipos de lluvia desde una débil hasta una muy fuerte, el sensor no fue capaz de medir las lluvias débiles de ahí la dificultad de poder entregar datos precisos de precipitación. El sensor YFS-201 se sustituirá en una tercera iteración por el sensor YL-83 que (Durrani, Khurram, & Khan, 2019) incorpora a un prototipo de estación meteorológica automática de bajo costo. Con la incorporación de este sensor a nuestro prototipo solo es necesario utilizar un microcontrolador el cual puede obtener los datos de los dos sensores ya que se trata de dos señales digitales, de aquí la posibilidad de que el sistema de alimentación eléctrica autosustentable sea suficiente ya que el consumo eléctrico del dispositivo se ha reducido aun

mas. Como se menciona en la revisión del dispositivo en la iteración 3, el sensor YL-83 no es suficiente para las mediciones de precipitación acumulada ya que este retiene agua en su superficie de medición y altera las mediciones, nuevamente el sistema de alimentación eléctrica autosustentable no es suficiente para mantener el dispositivo funcional al menos 24 horas. De las revisiones bibliográficas que continuamente se realizan, (Strigaro et al., 2019) en su investigación prueba diversos sensores para obtener la calidad de la información que estos devuelve a través de esta investigación nos damos cuenta que el sensor DHT11 no es lo suficiente para devolver información precisa como lo requiere la OMM. Debido a lo anterior es necesario desarrollar na cuarta iteración sustituyendo al sensor YL-83 y al DHT11 por un sensor de balancín y un DHT22 respectivamente, además de involucrar un diseño especifico para el sistema de alimentación eléctrica autosustentable. Después de poner en marcha este dispositivo y generando pruebas en laboratorio con distintos climas es que resulta este dispositivo un producto final en su iteración 4, estable de medición continua y en tiempo real y autosustentable eléctricamente.

Basándonos en el dispositivo de la iteración 4 es que se procede al desarrollo de la EMABC GSM rediseñando la arquitectura del prototipo Wifi y llevándolo a la comunicación Wifi utilizando un modulo capaz de conectarse a la red GPRS 2G el cual se controla con un microcontrolador Arduino UNO. Este dispositivo mantiene conexión con la base de datos enviando continuamente sin interrupciones información a la base de datos, los sensores fueron probados anteriormente en el dispositivo Wifi por lo cual no es necesario probar su funcionalidad con el dispositivo GSM ya que el rediseño solo involucra el protocolo de comunicación.

El desarrollo de estos dispositivos con conectividad a internet va mas allá de las antiguas formas de medir el clima, por ello es indispensable la vinculación con bases de datos IoT.

La implementación de las bases IoT mejoran el rendimiento de la captura, análisis y manipulación de la información ya que su compatibilidad y simplicidad de conexión con sistemas basados en Arduino las hace adecuadas para el uso. ThingSpeak es un servicio de base de datos que permite capturar información enviada por dispositivos IoT, visualizarla y descargarla, ofrece un plan gratuito para estudiantes que lo hace realmente relevante para la implementación en un sistema como el que se desarrolla de bajo costo ya que nos mantiene

en este concepto. Ubidots funciona exactamente que la anterior, además de que contiene servicios adicionales como los eventos de llamada, SMS, emails etc., por ello es un servicio de paga no obstante recomendado en la implementación de sistemas de información como (Enciso & Vargas, 2018) lo menciona en su investigación.

Estas bases de datos permiten la vinculación con sitios web a través de iframes vinculados por claves Api de una manera muy sencilla, hoy en día existe una diversidad de hostings en la web que permiten el desarrollo de sitios web, en este caso elegimos wix ya que es uno de los servicios de hosting antiguos que representan seguridad. Wix se ha prestado para plasmar todo un concepto de prototipo de baja fidelidad funcional de un sistema el cual presenta información de precipitación, temperatura y humedad del aire con visualización temporal y espacialmente. El diseño del sitio web se desarrollo considerando un usuario universal que englobaba todas las necesidades de los distintos usuarios. Con base en lo anterior y siguiendo el desarrollo bajo el concepto HCI, se creo una persona (usuario), este considerado como un tomador de decisiones ya que es aquel que se enfrenta hoy a la planificación de acciones contra riesgos. El diseño de la interfaz es el paso mas importante en el desarrollo de sistemas de información ya que esta es la cara del sistema con la cual el usuario interactúa, debido a esto es que se busco la funcionalidad y la usabilidad de esta. Para lograr estos requisitos fue que se inicio con un prototipo de baja fidelidad probado en papel con usuarios reales donde se busco la usabilidad optima. Una vez optimizado el prototipo es que se decidió generar el prototipo de alta fidelidad en wix.com con el uso de sus herramientas de diseño. La representación espacial de las estaciones facilita al usuario la ubicación de los puntos de interés que este requiere ubicar para la obtención de información del sitio. La presentación temporal de los datos, para el usuario le es de fácil entendimiento ya que esta puede presentarse en diversas escalas de tiempo, y facilita el análisis. La descarga de información de las estaciones le permite al usuario utilizar herramientas distintas para un análisis posterior de la información, cuando esto así lo requiera. Hoy en día la tendencia del uso del teléfono móvil va en ascendente crecimiento por ello, es que se planteo el desarrollo de una aplicación móvil, los usuarios pueden acceder a la información con mas facilidad ya que no es necesario acceder desde un computador, ni de un navegador móvil, si no, desde una aplicación oficial que redirecciona al sitio web en un formato de móvil. El desarrollo de la aplicación se llevó

a cabo mediante un software de uso libre ya que se pretende mantener el concepto de bajo costo, existen diversos gestores para desarrollar ficheros jdk, pero por lo general son de paga. Bien se puede tener un fichero jdk que puede ir de mano en mano para la instalación en los móviles, pero esto suele ser de poca confianza para los usuarios ya que pueden instalar un programa maligno y no lo que realmente se esta buscando. Por ello se ha creado una cuenta en google Play con la finalidad de oficializar la App en una tienda donde cualquier usuario pueda acceder para descargarla y que mejor que la tienda oficial de aplicaciones para Android.

La visualización de la información y el fácil acceso es de gran ayuda para los usuarios ya que en segundos pueden obtener información en tiempo real de diversos sitios de Morelia, no obstante, la importancia del uso de la información en tiempo real es la generación de alertas las cuales deben ser visualizadas por los usuarios. En este caso se eligió Twitter ya que es una de las redes sociales de mas rubro social. Twitter se ha convertido en una plataforma importante para la comunicación entre políticos y sus seguidores (Sainudiin, Yogeeswaran, Nash, & Sahioun, 2019). De esta manera postear las alertas de eventos de lluvia fuera de tolerancia, hace una forma de mantener informados a los usuarios y a las personas en general que siguen la cuenta. Con las herramientas existentes de IFTTT es evidente que se pueden realizar tareas automáticas de republicación de las alertas que se envían a twitter y por que no llevarlas a otras redes sociales si los usuarios se encuentran en estas, por ello se usa esta herramienta para publicar en Facebook las mismas alertas que Twitter recibe. La generación de alerta se vuelve posible con el uso de estos servicios IoT como ThingSpeak y Ubidots, ya que permiten el análisis de la información e inmediatamente enviar las alertas si esta se encuentra fuera de los rangos de tolerancia.

Para que la información llegue al usuario, de tras de esto se encuentran los algoritmos que controlan las placas de desarrollo de las EMABC. Estos están desarrollados en un lenguaje de programación C, estos llevan a cabo la instrucción desde la conexión a la red Wifi o GSM, obtención de información y el envió de esta a las bases de datos IoT para su análisis, almacenamiento y publicación a cada minuto. Existen dos algoritmos diferentes ya que tenemos dos protocolos de comunicación con las bases de datos diferentes, y estos requieren diferentes configuraciones para su funcionalidad. Estos fueron mejorados en paralelo con el

desarrollo de los dispositivos ya que cada vez se mejoraban mas hasta llegar a optimizarlos casi al 99%.

Para el SAIH es de gran importancia que las mediciones sean precisas ya que esto representa la confiabilidad de la información que se entrega por ello es por lo que los sensores pasaron por ciertas pruebas de precisión, calibración y validación. En el caso del sensor DHT22 se sometió a una prueba de precisión con comparación de observaciones con estación cercana, de esto resulta que la información medida realmente esta dentro de los rangos de precisión de la estación comercial. Siendo el mismo caso para el pluviógrafo de balancín.

Finalmente se presentan dos modelos de SAIH identificados por la base de datos donde difieren, entiéndase que el uso de ThingSpeak es de uso gratuito lo cual no implica costos adicionales a la construcción del sistema, pero existen limitaciones desde la cantidad de datos a generar y los tiempos de escritura cortos, en nuestro caso no existe problema con ello ya que se generan datos a cada minuto. El modelo que se basa en Ubidots si requiere de costos adicionales dependiendo del plan que se adquiera, este se cobra en base a la cantidad de dispositivos que se quieran instalar y se recomienda utilizarlo cuando se pretenda montar el sistema por una institución gubernamental que tenga los fondos suficientes para cubrir los gastos del SAIH.

## 10. Conclusiones

“El cambio tecnológico, particularmente en los países en desarrollo, no se trata solo de innovar en la frontera, sino también de adaptar los productos y procesos existentes para lograr mayores niveles de productividad según corresponda a sus contextos locales. En este proceso, la capacidad de las empresas globales y empresas locales para acceder a los conocimientos tecnológicos es fundamental para configurar su capacidad de proporcionar productos y servicios, que creemos que son esenciales para mejorar el nivel de vida, y que también podrían promover el crecimiento y la competitividad "(Strigaro et al., 2019). En este sentido, el Sistema Automático de Información Hidrológica de Morelia puede ser una oportunidad para los países en desarrollo de monitorear variables meteorológicas en tiempo real con la finalidad de generar información en cantidad y calidad para la gestión de riesgos ambientales

(inundaciones, sequias, cambio climático) a bajo costo. Este concepto se mantiene cuando con el uso de las tecnologías de bajo costo que se han utilizado se desarrollan los dispositivos de medición con los cuales puede monitorearse tanto zonas urbanas y rurales por muy remotas que sean estas. La implementación del internet de las cosas en el desarrollo de estos dispositivos de medición es una de las oportunidades mas aprovechadas en la investigación ya que con esto fue posible la implementación de bases de datos IoT capaces de recibir información y publicarlas en un sitio web y un App en tiempo real, además de emitir alertas en el caso de que la precipitación este fuera del rango de tolerancia. La visualización espacial y temporal le dan un plus al SAIH en la usabilidad de las plataformas para el usuario ya que es muy fácil de leer la información. Un sistema que emite datos en cantidad y además con calidad es una gran oportunidad de aplicación dentro de la investigación del clima. Esta investigación ha seguido toda una metodología sistemática la cual implica desde el estudio de necesidades y oportunidades para el desarrollo, calibración y validación de un sistema de adquisición de datos meteorológicos en tiempo real. Por ello la importancia de la aplicación de una metodología sistemática que posterior a la puesta en marcha se siga replicando para la reingeniería continua del SAIH.

El SAIH se basa en un sistema automático de información el cual puede ser aplicable en diversas áreas que hoy en día siguen la línea de investigación de los SI con la aplicación de conceptos como bajo costo y tiempo real un ejemplo claro es la medicina en el desarrollo de detección de necesidades de los pacientes a través de pulsos eléctricos que emite el cerebro detectados a través de Arduino.

## 11. Líneas Futuras

El SAIH es una estructura de monitoreo que puede aplicarse a diversas áreas de interés, esto es posible debido a que las arquitecturas de los dispositivos de medición son abiertas a la incorporación de cualquier sensor, y por lo tanto podrían obtenerse dispositivos que devuelvan información diferente al clima. Hoy en día la aplicación de la tecnología basada en Arduino en el área de la salud, agricultura, ganadería, industria y otras áreas ha generado una motivación en la investigación y construcción de dispositivos de adquisición de

información en tiempo real en estas áreas. Por ello es posible que el SAIH sea un base de la que se pueda partir para el monitoreo en cualquier área.

Parte importante en el uso de tecnologías de bajo costo es el análisis de durabilidad de los equipos que se han construido ya que esto demostrará aun mas la factibilidad del uso de estas herramientas.

## 12. Recomendaciones

El SAIH esta desarrollado bajo una metodología sistemática la cual implica que continuamente se monitoree su actividad, de tal forma que se pueda analizar si existe alguna falla en este, para una funcionalidad continua es necesario que este se someta a mantenimiento frecuente desde la revisión de sus dispositivos y la supervisión de las bases de datos IoT, del hosting del sitio web y de la cuenta de desarrollador de google play que mantiene la App.

### 12.1 Monitoreo de Actividad

El monitoreo de actividad se realizará a través de la base de datos bajo un protocolo de revisión horaria de tal forma que pueda analizarse si los dispositivos están activos y enviando información, esto para el modelo de SAIH que se basa en ThingSpeak en el caso del SAIH que se basa en Ubidots uno de sus servicios de SMS se encarga de notificar al administrador si existe inactividad en los dispositivos.

### 12.2 Mantenimiento

El mantenimiento es directamente a los dispositivos ya que se desconoce su durabilidad, requieren de frecuente revisión de tal forma que se pueda analizar cuando es necesario sustituir microcontroladores, sensores o hasta algún elemento del sistema de alimentación eléctrica autosustentable.

### 12.3 Investigación y reingeniería

El SAIH se basa en tecnología que día a día se actualiza y pronto puede quedar descartados por ello es recomendable que se mantenga una constante investigación sobre mejoras de los dispositivos o bases de datos de tal forma que se someta a reingeniería.

## 13. Literatura Citada


Abrahim, S., Mir, B. A., Suhara, H., Mohamed, F. A., & Sato, M. (2019). Structural equation modeling and confirmatory factor analysis of social media use and education. *International Journal of Educational Technology in Higher Education*, *16*(1). https://doi.org/10.1186/s41239-019-0157-y

Alvarado López, R. A., & Alvarado López, R. A. (2018). Ciudad inteligente y sostenible: hacia un modelo de innovación inclusiva. *PAAKAT: Revista de Tecnología y Sociedad*, *7*(13). https://doi.org/10.18381/pk.a7n13.299

Anggraini, D., Effendy, N., Ihsan Al Hafiz, M., & Ojeda Luviano, D. (2018). Research and Development of a Power Monitoring System for the Sustainable Energy Management System Implementation at Green School, Bali, Indonesia. *E3S Web of Conferences*, *43*. https://doi.org/10.1051/e3sconf/20184301021

Arun Raj, V., Nambi Srinivasan, S., & Vimalathethan, K. K. A. (2019). Smart irrigation and management of cash crops using IoT. *International Journal of Recent Technology and Engineering*, *8*(2), 521–525. https://doi.org/10.35940/ijrte.B1583.078219

Durrani, A., Khurram, M., & Khan, H. R. (2019). Smart Weather Alert System for dwellers of different Areas. *Proceedings of 2019 16th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2019*, 333–339. https://doi.org/10.1109/IBCAST.2019.8667190

Enciso, L., & Vargas, A. (2018). Interface with Ubidots for a fire alarm system using WiFi. *Iberian Conference on Information Systems and Technologies, CISTI*, *2018-June*, 1–6. https://doi.org/10.23919/CISTI.2018.8399327

Escuela Politécnica Nacional (Quito, E., Llugsi, R., Lupera, P., Chango, R., Suárez, A., Llugsi, R., … Chango, R. (2017). Politécnica. In *Revista Politécnica* (Vol. 39). Retrieved from http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-01292017000200017

Faizah, S., Novianti, L., Kom, S., Kom, M., Novita, N., E, S., & M, M. (2019). The Implementation of Ant Colony Algorithm in Finding the Shortest Travel Route of Palembang Tourism by Android Based. *Journal of Physics: Conference Series*, *1167*(1). https://doi.org/10.1088/1742-6596/1167/1/012067

Farid, M. M., Prawito, Susila, I. P., & Yuniarto, A. (2017). *Design of early warning system for nuclear preparedness case study at Serpong*. 030067. https://doi.org/10.1063/1.4991171

Gabino, S., Fuertes, L. L., Lopresti, L. A., Defranco, G., & Lara, M. (2015). Aplicaciones para dispositivos móviles: una aproximación en las prácticas de enseñanza de los sistemas de representación. *III Jornadas de Investigación, Transferencia y Extensión de La Facultad de Ingeniería*, *1*(2015–04), 591–598. Retrieved from http://sedici.unlp.edu.ar/handle/10915/47900

Haque, M. I., MD. Shatil, A. H., Tusar, A. N., Hossain, M., & Rahman, M. H. (2019). Renewable Powered Portable Weather Update Station. *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, 374–377. https://doi.org/10.1109/ICREST.2019.8644330

Imtiaz, A., Omar, S. G., & Ali, T. A. (2018). Efficient Design of a Low Cost Portable Weather Station. *2018 International Conference on Computer Communication and Informatics (ICCCI)*, 1–7. https://doi.org/10.1109/ICCCI.2018.8441207

Instituto Nacional de Investigaciones Forestales, A. y P. (Mexico), Quevedo Nolasco, A., Tijerina Chávez, L., Castro Popoca, M., Guzmán Luna, R., Quevedo Nolasco, A., … Castro Popoca, M. (2010). Revista mexicana de ciencias agrícolas. In *Revista mexicana de ciencias agrícolas* (Vol. 6). Retrieved from http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-09342015000801701&lang=pt

Iskandar, M. S., & Adhayani, S. W. (2018). Analysis of Utilizing Website in Designing Online Store Site. *IOP Conference Series: Materials Science and Engineering*, *407*(1). https://doi.org/10.1088/1757-899X/407/1/012178

Janpla, S., & Jewpanich, C. (2019). The architecture of the smart flowerpot by using the internet of things (IoT). *International Journal of Engineering and Advanced Technology*, *9*(1), 6419–6426. https://doi.org/10.35940/ijeat.A2208.109119

Malik, A., Heyman-Schrum, C., & Johri, A. (2019, December 1). Use of Twitter across educational settings: a review of the literature. *International Journal of Educational Technology in Higher Education*, Vol. 16. https://doi.org/10.1186/s41239-019-0166-x

Muñoz, R. C., Yumang, X. S., Japitana, S. J. A., Medina, K. R. C., & Tibayan, J. E. T.

(2018). Micro-weather Station System for Small Geographical Coverage in the Philippines. *IOP Conference Series: Earth and Environmental Science*, *192*, 012064. https://doi.org/10.1088/1755-1315/192/1/012064

Negara, R. M., Tulloh, R., Nandy Hadiansyah, P. N., & Zahra, R. T. (2019). My locker: Loaning locker system based on QR code. *International Journal of Engineering and Advanced Technology*, *9*(1), 12–19. https://doi.org/10.35940/ijeat.A1008.109119

Netto, G. T., & Arigony-Neto, J. (2019). Open-source Automatic Weather Station and Electronic Ablation Station for measuring the impacts of climate change on glaciers. *HardwareX*, *5*, e00053. https://doi.org/10.1016/j.ohx.2019.e00053

OMM. (2014). *Guía de Instrumentos y Métodos de Observación Meteorológicos* (OMM, Ed.).

Radhika, D., & Aruna Kumari, D. (2019). The smart triad: Big data analytics, cloud computing and internet of things to shape the smart home, smart city, smart business & smart country. *International Journal of Recent Technology and Engineering*, *8*(2 Special Issue 11), 3594–3600. https://doi.org/10.35940/ijrte.B1449.0982S1119

Rica, C., Estatal, U., & Rica, C. (2011). *Utilización de estaciones meteorológicas automáticas como nueva alternativa para el registro y transmisión de datos Fernando Ureña Elizondo*. 33–49.

Sainudiin, R., Yogeeswaran, K., Nash, K., & Sahioun, R. (2019). Characterizing the Twitter network of prominent politicians and SPLC-defined hate groups in the 2016 US presidential election. *Social Network Analysis and Mining*, *9*(1). https://doi.org/10.1007/s13278-019-0567-9

Salahuddin, Basyir, M., & Tusannalaila, S. (2019). Design Tools of Electric Applications Kwh Printing by Short Message Service. *IOP Conference Series: Materials Science and Engineering*, *536*(1). https://doi.org/10.1088/1757-899X/536/1/012099

Satria, D., Yana, S., Yusibani, E., Syahreza, S., & Zulfan. (2019). Implementation of the SMS gateway in the flood early warning information system for village warning and community information. *International Journal of Engineering and Advanced Technology*, *8*(6), 4005–4009. https://doi.org/10.35940/ijeat.F9287.088619

Singh, R., Bhojwani, Y., Reddy, R., & Sultana, H. P. (2019). Hand gesture based multi-purpose bot. *International Journal of Innovative Technology and Exploring*

*Engineering*, *9*(1), 4419–4423. https://doi.org/10.35940/ijitee.A5098.119119

Sinyo, M. A. W. (2019). Pengembangan Aplikasi Multimedia Pembelajaran Penulisan Daftar Pustaka Karya Ilmiah Berbasis Android Menggunakan Appsgeyser. *Prosiding Seminar Nasional Teknologi Informasi Dan Komunikasi (SENATIK)*, *2*(1), 23–30.

Srivastava, C., Singh, S., & Singh, A. P. (2018). *IoT-Enabled Air Monitoring System*. 173–180. https://doi.org/10.1007/978-981-13-6095-4_13

Stair, R. M., & Reynolds, G. W. (2010). *Principios de sistemas de información : un enfoque administrativo*. Retrieved from https://www.buscalibre.com.mx/libro-principios-de-sistemas-de-informacion-un-enfoque-administrativo-stair-cengage-learning-editores-s-a-de-c-v/9786074812671/p/17794508?t=test-envio&gclid=CjwKCAjwyqTqBRAyEiwA8K_4O6K029Bt7kYOClNN4hNJ2d0heNxouovn4BjyIDC4UDRF

Strigaro, D., Cannata, M., & Antonovic, M. (2019). Boosting a Weather Monitoring System in Low Income Economies Using Open and Non-Conventional Systems: Data Quality Analysis. *Sensors*, *19*(5), 1185. https://doi.org/10.3390/s19051185

Taştan, M., & Gökozan, H. (2019). Real-time monitoring of indoor air quality with internet of things-based e-nose. *Applied Sciences (Switzerland)*, *9*(16). https://doi.org/10.3390/app9163435

Vijaya Rajan, P., Babu, T., Karthik Pandiyan, G., Venkatragavan, D., & Shanmugam, R. (2019). Auxillary safety locking system of vehicle doors using arduino. *International Journal of Innovative Technology and Exploring Engineering*, *9*(1), 277–281. https://doi.org/10.35940/ijitee.A4022.119119

World Meteorological Organization Extranet | www.wmo.int. (n.d.). Retrieved July 22, 2018, from http://www.wmo.int/pages/index_es.html

Xu, R., Zeng, Q., Zhu, L., Chi, H., Du, X., & Guizani, M. (2019). Privacy Leakage in Smart Homes and Its Mitigation: IFTTT as a Case Study. *IEEE Access*, *7*, 63457–63471. https://doi.org/10.1109/ACCESS.2019.2911202

# 14. Anexos

## 14.1 Anexo 1: Algoritmo EMABC Wifi a ThingSpeak

```
// Developed By M V Subrahmanyam
// Electrinics Innovation

#include <DHT.h> // library for getting data from DHT
#include <ESP8266WiFi.h> //Librarry connecting ESP8266 to connect with Wifi



//----------------------Definición de Variables

int  Contadorlluvia;        //Contador de pulsos magneticos del sensor de lluvia
int  Estatuslluvia;         //Estado de lluvia
unsigned long Tiempo = 0;   //Tiempo inicial
unsigned long Tiempo1 = 0;  //Toma el tiempo actual de la variable Tiempo cada 60000
milisegundos (1 minuto)
unsigned long Tiempo2 = 0;  //Indicador de que ha transcurrido un minuto




// ***********************************************Thingspeak
Credentials*****************
String apiKey = "6MTHIRQHJVOM3Q7I"; //Write API key of your ThingSpeak channel
****se modifica
const char* server = "api.thingspeak.com"; // API for thingspeak

const char *ssid = "Floor 4"; // Wifi SSID of your Internet connection   ****se modifica
const char *pass = "@viverasmus"; // Password  ****se modifica

//*****************************************************************************
***************



//--Pin GPIO 15 de la NodeMCU ESP8266 para conexión del pin
const byte PinLluvia = 15;


#define DHTPIN D2 //pin where the DHT22 is connected
DHT dht(DHTPIN, DHT22);
WiFiClient client;
void setup()
```

```
{
  Serial.begin(9600); // Serial monitor Baudrate
  delay(10);
  //******************PowerSupply to the Sensor**********************

  //*********************************************************
  dht.begin();

  Serial.println("Trying to Connect with ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass); // Connecting ESP8266 with Internet enabled Wifi with above
mentioned credentials
  while (WiFi.status() != WL_CONNECTED)
  {
    // If the connection was unsuccesfull, it will try again and again
    delay(500);
    Serial.print(".");
  }
  // Connection succesfull
  Serial.println("");
  Serial.println("WiFi connected");
}
void loop()
{




  float humedad = dht.readHumidity(); // Reading Temperature form DHT sensor
  float temperatura = dht.readTemperature(); // Reading Humidity form DHT sensor
  if (isnan(humedad) || isnan(temperatura))
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Estatuslluvia = digitalRead(PinLluvia); //Lectura de señal en el pin digital
  if (Estatuslluvia == HIGH) {  //Si existe corriente (esta se genera cuando hay pulsos en el
sensor) es una condicion de la lectura del pin
    Contadorlluvia++; //aumenta una unidad cada que se detecta corriente
    delay(300); // Espera 300 milisegundos para que el balancin vuelva a su posición
  }
  float precipitacion = Contadorlluvia * 0.2794; //Se multiplica las veces que se volteo el
balancín por el volumen de sus contenedores que equivalen a 0.2794 mm cada volteo
  Tiempo = millis(); //Inicia el conteo de tiempo
```

```
  Tiempo2 = Tiempo - Tiempo1;// obtención del Tiempo2 el cual debe llegar a 60000 milisegundos

  if (Tiempo2 == 60000) { // Condicional si el Tiempo2 es igual a 60000 milisegundos se ejecuta lo siguiente
    float Intensidad = precipitacion * 60; //Calculo de la intensidad en mm/hr
    Serial.print("Precipitación: ");
    Serial.print(precipitacion);
    Serial.println(" (mm)");
    Serial.print("Intensidad: ");
    Serial.print(Intensidad);
    Serial.println(" (mm/h)");
    Serial.print("Humedad Relativa: ");
    Serial.print(humedad);
    Serial.println(" (%)");
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.println(" (°C)");

    // Connecting to the Thingspeak API and Posting DATA
    if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com
    {
      // Format of DATA Packet "Write API Key&field1=Temperature data&field2=Humidity Data"
      String postStr = apiKey;
      postStr += "&field1=";
      postStr += String(Intensidad);
      postStr += "&field2=";
      postStr += String(precipitacion);
      postStr += "&field3=";
      postStr += String(temperatura);
      postStr += "&field4=";
      postStr += String(humedad);
      postStr += "\r\n\r\n";

      client.print("POST /update HTTP/1.1\n");
      client.print("Host: api.thingspeak.com\n");
      client.print("Connection: close\n");
      client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
      client.print("Content-Type: application/x-www-form-urlencoded\n");
      client.print("Content-Length: ");
      client.print(postStr.length());
      client.print("\n\n");
      client.print(postStr);
      client.flush();
      Serial.println(postStr);
```

```
    Serial.println("Data has been sussecfully sent to Thingspeak.");
   }
   client.stop();


   Contadorlluvia = 0; //Se reinicia el contador
   Tiempo1 = Tiempo; //Se le asigna el tiempo actual de Tiempo a Tiempo1
  }

}
```

### 14.2 Anexo 2: Algoritmo EMABC Wifi a Ubidots

```
//----------------------Definición de Librerías

//--Librería DHT para sensor DHT11 o DHT22
#include "DHT.h"

//--Librería Ubidots para envío de datos a base de datos
#include "UbidotsMicroESP8266.h"

//--------------------------------------------------------------------



//----------------------Definición de Pines

//--Pin D2 de la NodeMCU ESP8266 para conexión del pin de datos del sensor DHT22
#define DHTPIN D2 //Defición de pines por el modo de etiqueta

//--Pin GPIO 15 de la NodeMCU ESP8266 para conexión del pin
const byte PinLluvia = 15;
//-------------------------------------------------------------



//----------------------ADICIONALES DHT22

//--Definición del tipo de sensor en este caso DHT22
#define DHTTYPE DHT22

//--Configuración de librería DHT
DHT dht(DHTPIN, DHTTYPE);

//-------------------------------------------------------------

//----------------------Definición de Variables

int  Contadorlluvia;        //Contador de pulsos magneticos del sensor de lluvia
int  Estatuslluvia;         //Estado de lluvia
unsigned long Tiempo = 0;   //Tiempo inicial
unsigned long Tiempo1 = 0;  //Toma el tiempo actual de la variable Tiempo cada 60000
milisegundos (1 minuto)
unsigned long Tiempo2 = 0;  //Indicador de que ha transcurrido un minuto
unsigned long Tiempo3 = 0;  //Toma el tiempo actual de la variable Tiempo cada 900000
milisegundos (15 minutos)
unsigned long Tiempo4 = 0;  //Indicador de que ha transcurrido 15 minutos
```

```
//-------------------------------------------------------------------------------------------------------
----

//----------------------ADICIONALES UBIDOTS

//--Configuración de red y Credenciales de UBIDOTS
#define TOKEN  "BBFF-
pENnzPXSgP2y8ekXIAb6oHtYkzirIRMYS7py3coQIacuHudW4ye2sNg"  //Se define el
TOKEN de la cuenta de UBIDOTS
#define WIFISSID "Floor 4"  //Se define el nombre de la red inalambrica de internet a la
que se conectará la placa
#define PASSWORD "@viverasmus"  //Se define la contraseña de la red inalambrica de
internet a la que se conectára la placa

//--Vinculación con los servicios de UBIDOTS
Ubidots client(TOKEN);

//-------------------------------------------------------------------------------------------------------
----


void setup() {
  Serial.begin(9600); // Inicio de la comunicación serial
  dht.begin();  // Inicio de la librería DHT
  client.wifiConnection(WIFISSID, PASSWORD); //Conexión a la red inalambrica y al
servidor UBIDOTS
  delay(5000);  //Esperar 5 segundos
}

void loop() {
  Estatuslluvia = digitalRead(PinLluvia); //Lectura de señal en el pin digital
  if (Estatuslluvia == HIGH) {  //Si existe corriente (esta se genera cuando hay pulsos en el
sensor) es una condicion de la lectura del pin
    Contadorlluvia++; //aumenta una unidad cada que se detecta corriente
    delay(300); // Espera 300 milisegundos para que el balancin vuelva a su posición
  }
  float precipitacion = Contadorlluvia * 0.2794; //Se multiplica las veces que se volteo el
balancín por el volumen de sus contenedores que equivalen a 0.2794 mm cada volteo

  float humedad = dht.readHumidity(); //Lectura de Humedad
  float temperatura = dht.readTemperature();// Lectura de Temperatura

  Tiempo = millis(); //Inicia el conteo de tiempo
  Tiempo2 = Tiempo - Tiempo1;// obtención del Tiempo2 el cual debe llegar a 60000
milisegundos
```

```
  if (Tiempo2 == 60000) { // Condicional si el Tiempo2 es igual a 60000 milisegundos se
ejecuta lo siguiente
    float Intensidad = precipitacion * 60; //Calculo de la intensidad en mm/hr
    Serial.print("Precipitación: ");
    Serial.print(precipitacion);
    Serial.println(" (mm)");
    Serial.print("Intensidad: ");
    Serial.print(Intensidad);
    Serial.println(" (mm/h)");
    client.add("Precipitación (mm)", precipitacion); //se etiqueta la variable precipitación
para enviar
    client.add("Intesidad (mm/h)", Intensidad); //se etiqueta la variable Intensidad para
enviar
    client.sendAll(true); //Envío de variables a la base de datos de UBIDOTS
    Contadorlluvia = 0; //Se reinicia el contador
    Tiempo1 = Tiempo; //Se le asigna el tiempo actual de Tiempo a Tiempo1
  }

  Tiempo3 = Tiempo - Tiempo4; // obtención del Tiempo3 el cual debe llegar a 900000
milisegundos
  if (Tiempo3 == 900000) {// Condicional si el Tiempo3 es igual a 900000 milisegundos se
ejecuta lo siguiente
    Serial.print("Humedad Relativa: ");
    Serial.print(humedad);
    Serial.println(" (%)");
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.println(" (°C)");
    client.add("humedad Relativa (%)", humedad);  //se etiqueta la variable humedad para
enviar
    client.add("Temperatura (°C)", temperatura);  //se etiqueta la variable temperatura para
enviar
    client.sendAll(true);//Envío de variables a la base de datos de UBIDOTS
    Tiempo4 = Tiempo;   //Se le asigna el tiempo actual de Tiempo a Tiempo4
  }


}
```

### 14.3 Anexo 3: Algoritmo EMABC GSM a ThingSpeak

```
//--Librería DHT para sensor DHT11 o DHT22
#include "DHT.h"
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial Sim900Serial(7, 8);//Configarión de los pines serial por software
// Incluya las bibliotecas que necesitamos para controlar el sensor ds18b20

//Configure una instancia oneWire para comunicarse con cualquier dispositivo OneWire

//----------------------Definición de Pines

//--Pin D2 de la NodeMCU ESP8266 para conexión del pin de datos del sensor DHT22
#define DHTPIN D2 //Defición de pines por el modo de etiqueta

//--Pin GPIO 15 de la NodeMCU ESP8266 para conexión del pin
const byte PinLluvia = 15;
//---------------------------------------------------------------


//----------------------ADICIONALES DHT22

//--Definición del tipo de sensor en este caso DHT22
#define DHTTYPE DHT22

//--Configuración de librería DHT
DHT dht(DHTPIN, DHTTYPE);

//---------------------------------------------------------------
//----------------------Definición de Variables

int  Contadorlluvia;       //Contador de pulsos magneticos del sensor de lluvia
int  Estatuslluvia;        //Estado de lluvia


void setup()
{
Sim900Serial.begin(19200);//Arduino se comunica con el SIM900 a una velocidad de
19200bps
Serial.begin(19200);//Velocidad del puerto serial de arduino

//Encendido del módulo por sofware
digitalWrite(9, HIGH);
delay(1000);
```

```
digitalWrite(9, LOW);
delay(20000);//Tiempo prudencial para el escudo inicie sesión de red con tu operador
}
void loop(){
comandosAT();//Llama a la función comandosAT
if(Sim900Serial.available())//Verificamos si hay datos disponibles desde el SIM900
Serial.write(Sim900Serial.read());//Escribir datos
}
void comandosAT(){
Sim900Serial.println("AT+CIPSTATUS");//Consultar el estado actual de la conexión
delay(2000);
Sim900Serial.println("AT+CIPMUX=0");//comando configura el dispositivo para una
conexión IP única o múltiple 0=única
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CSTT=\"internet\",\"orange\",\"orange\"");//comando configura
el APN, nombre de usuario y contraseña."gprs.movistar.com.ar","wap","wap"->Movistar
Arg.
delay(1000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIICR");//REALIZAR UNA CONEXIÓN INALÁMBRICA
CON GPRS O CSD
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIFSR");// Obtenemos nuestra IP local
delay(2000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSPRT=0");//Establece un indicador '>' al enviar datos
Variables();
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");//Indicam
os el tipo de conexión, url o dirección IP y puerto al que realizamos la conexión
delay(6000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSEND");//ENVÍA DATOS A TRAVÉS DE una
CONEXIÓN TCP O UDP
delay(4000);
mostrarDatosSeriales();
String datos="GET
https://api.thingspeak.com/update?api_key=DVB7QCNA6XX75PR8&field1=0" +
String(variables);
Sim900Serial.println(datos);//Envía datos al servidor remoto
delay(4000);
mostrarDatosSeriales();
Sim900Serial.println((char)26);
```

```
delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
Sim900Serial.println();
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
delay(5000);
mostrarDatosSeriales();
}
void mostrarDatosSeriales()//Muestra los datos que va entregando el sim900
{
while(Sim900Serial.available()!=0)
Serial.write(Sim900Serial.read());
}
Variables (){
  Estatuslluvia = digitalRead(PinLluvia); //Lectura de señal en el pin digital
  if (Estatuslluvia == HIGH) {  //Si existe corriente (esta se genera cuando hay pulsos en el sensor) es una condicion de la lectura del pin
    Contadorlluvia++; //aumenta una unidad cada que se detecta corriente
    delay(300); // Espera 300 milisegundos para que el balancin vuelva a su posición
  }
  float precipitacion = Contadorlluvia * 0.2794; //Se multiplica las veces que se volteo el balancín por el volumen de sus contenedores que equivalen a 0.2794 mm cada volteo
  float Intensidad = precipitacion * 60; //Calculo de la intensidad en mm/hr
  float humedad = dht.readHumidity(); //Lectura de Humedad
  float temperatura = dht.readTemperature();// Lectura de Temperatura
  }
```

### 14.4 Anexo 4: Algoritmo EMABC GSM a Ubidots

```
//--Librería DHT para sensor DHT11 o DHT22
#include "DHT.h"
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial Sim900Serial(7, 8);//Configarión de los pines serial por software
// Incluya las bibliotecas que necesitamos para controlar el sensor ds18b20

//Configure una instancia oneWire para comunicarse con cualquier dispositivo OneWire

//----------------------Definición de Pines

//--Pin D2 de la NodeMCU ESP8266 para conexión del pin de datos del sensor DHT22
#define DHTPIN D2 //Defición de pines por el modo de etiqueta

//--Pin GPIO 15 de la NodeMCU ESP8266 para conexión del pin
const byte PinLluvia = 15;
//---------------------------------------------------------------


//----------------------ADICIONALES DHT22

//--Definición del tipo de sensor en este caso DHT22
#define DHTTYPE DHT22

//--Configuración de librería DHT
DHT dht(DHTPIN, DHTTYPE);

//---------------------------------------------------------------
//----------------------Definición de Variables

int  Contadorlluvia;       //Contador de pulsos magneticos del sensor de lluvia
int  Estatuslluvia;        //Estado de lluvia


void setup()
{
Sim900Serial.begin(19200);//Arduino se comunica con el SIM900 a una velocidad de
19200bps
Serial.begin(19200);//Velocidad del puerto serial de arduino

//Encendido del módulo por sofware
digitalWrite(9, HIGH);
delay(1000);
```

```
digitalWrite(9, LOW);
delay(20000);//Tiempo prudencial para el escudo inicie sesión de red con tu operador
}
void loop(){
comandosAT();//Llama a la función comandosAT
if(Sim900Serial.available())//Verificamos si hay datos disponibles desde el SIM900
Serial.write(Sim900Serial.read());//Escribir datos
}
void comandosAT(){
Sim900Serial.println("AT+CIPSTATUS");//Consultar el estado actual de la conexión
delay(2000);
Sim900Serial.println("AT+CIPMUX=0");//comando configura el dispositivo para una
conexión IP única o múltiple 0=única
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CSTT=\"internet\",\"orange\",\"orange\"");//comando configura
el APN, nombre de usuario y contraseña."gprs.movistar.com.ar","wap","wap"->Movistar
Arg.
delay(1000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIICR");//REALIZAR UNA CONEXIÓN INALÁMBRICA
CON GPRS O CSD
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIFSR");// Obtenemos nuestra IP local
delay(2000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSPRT=0");//Establece un indicador '>' al enviar datos
Variables();
delay(3000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSTART=\"TCP\",\"api.ubidots.com\",\"80\"");//Indicamos
el tipo de conexión, url o dirección IP y puerto al que realizamos la conexión
delay(6000);
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSEND");//ENVÍA DATOS A TRAVÉS DE una
CONEXIÓN TCP O UDP
delay(4000);
mostrarDatosSeriales();
String datos="GET https://api.ubidots.com/update?api_key=BBFF-
pENnzPXSgP2y8ekXIAb6oHtYkzirIRMYS7py3coQIacuHudW4ye2sNg=0" +
String(variables);
Sim900Serial.println(datos);//Envía datos al servidor remoto
delay(4000);
mostrarDatosSeriales();
Sim900Serial.println((char)26);
```

```
delay(5000);//Ahora esperaremos una respuesta pero esto va a depender de las condiones de la red y este valor quizá debamos modificarlo dependiendo de las condiciones de la red
Sim900Serial.println();
mostrarDatosSeriales();
Sim900Serial.println("AT+CIPSHUT");//Cierra la conexión(Desactiva el contexto GPRS PDP)
delay(5000);
mostrarDatosSeriales();
}
void mostrarDatosSeriales()//Muestra los datos que va entregando el sim900
{
while(Sim900Serial.available()!=0)
Serial.write(Sim900Serial.read());
}
Variables (){
  Estatuslluvia = digitalRead(PinLluvia); //Lectura de señal en el pin digital
  if (Estatuslluvia == HIGH) {  //Si existe corriente (esta se genera cuando hay pulsos en el sensor) es una condicion de la lectura del pin
    Contadorlluvia++; //aumenta una unidad cada que se detecta corriente
    delay(300); // Espera 300 milisegundos para que el balancin vuelva a su posición
  }
  float precipitacion = Contadorlluvia * 0.2794; //Se multiplica las veces que se volteo el balancín por el volumen de sus contenedores que equivalen a 0.2794 mm cada volteo
  float Intensidad = precipitacion * 60; //Calculo de la intensidad en mm/hr
  float humedad = dht.readHumidity(); //Lectura de Humedad
  float temperatura = dht.readTemperature();// Lectura de Temperatura
  }
```

### 14.5 Anexo 5: Codigo HTML Mapa de red de monitoreo

```
<!DOCTYPE html>
<html>
  <head>

    <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
    <meta charset="utf-8">
    <title>Info Windows</title>
    <style>
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }
      /* Optional: Makes the sample page fill the window. */
      html, body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
                                <script src="../js_popup/jquery.js"></script>
<script src="../js_popup/fancybox/jquery.fancybox-1.3.4.pack.js"></script>
<link rel="stylesheet" type="text/css" href="../js_popup/fancybox/jquery.fancybox-
1.3.4.css">

 <script>
    function abre(nombre){
        $.fancybox({
                'href'                          : nombre,
                'autoScale'                     : true,
                'transitionIn'          : 'none',
                'transitionOut'                 : 'none',
                'width'                                 : 700,
                'height'                        : 490,
                'modal'                                 : false,
                'type'                                  : 'iframe'

            });
    }
</script>
  </head>
  <body>

    <div id="map"></div>
```

```
<script>

  // This example displays a marker at the center of Australia.
  // When the user clicks the marker, an info window opens.

  function initMap() {
        var Centrar= {lat: 19.701806, lng: -101.196550};
    var Camelinas = {lat: 19.682198, lng: -101.177408};
            var CU = {lat: 19.690119, lng: -101.204754};
            var Lomasdelajoya = {lat: 19.6635582, lng: -101.1960197};
    var Lomasamericas = {lat: 19.6943212, lng: -101.1414839};
    var Wenceslao = {lat: 19.7172263, lng: -101.2239849};
    var Centro = {lat: 19.6979361, lng: -101.1978357};
    var map = new google.maps.Map(document.getElementById('map'), {
      zoom: 13,
      center: Centrar
    });



    var contentString = '<div id="content">'+
      '<div id="siteNotice">'+
      '</div>'+
      '<h1 id="firstHeading" class="firstHeading">SAIH-E0001  Camelinas</h1>'+
      '<div id="bodyContent">'+
      '<p><b>Ubicación: </b>' +
      'Lat:      19.682198°, '+
      'Long:  -101.177408°</p>'+
                  '<div>'+
                  '<a href="https://www.saihmorelia.com/copia-de-p-saih-e0002-3"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
                  '</div>'+
                  '<div>'+
                  '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/08VNzgBu7UIjZlFbQj3J
RkcJ92k?embed=true"></iframe>'+
                  '</div>'+
                  '<div>'+
                  //'<a><img src="../Imagenes/Estacion1.jpeg"  class="img_c"></a>'+
                  '</div>'+
      '</div>'+
      '</div>';

            var contentString2 = '<div id="content">'+
      '<div id="siteNotice">'+
      '</div>'+
```

```
        '<h1 id="firstHeading" class="firstHeading">SAIH-E0002  C.U.</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Ubicación: </b>' +
        'Lat:      19.690119°, '+
        'Long:  -101.204754°</p>'+
                        '<div>'+
                        '<a href="https://www.saihmorelia.com/saih-e0002"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
        '</div>'+
                        '<div>'+
                        '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/ABNbxLv1h68rmVsVb8
VhVJ_2DS4?embed=true"></iframe>'+
                        '</div>'+
                        '<div>'+
                //      '<a><img src="../Imagenes/Estacion2.jpeg"  class="img_c"></a>'+
                        '</div>'+
        '</div>'+
        '</div>';

                var contentString3 = '<div id="content">'+
        '<div id="siteNotice">'+
        '</div>'+
        '<h1 id="firstHeading" class="firstHeading">SAIH-E0003  Lomas de la
Joya</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Ubicación: </b>' +
        'Lat:      19.663558°, '+
        'Long:  -101.204754°</p>'+
                        '<div>'+
                        '<a href="https://www.saihmorelia.com/copia-de-p-saih-e0002-2"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
        '</div>'+
                        '<div>'+
                        '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/RtRYMlQB6J4Utf2RWI
79ir3_bdg?embed=true"></iframe>'+
                        '</div>'+
                        '<div>'+
                        //'<a><img src="../Imagenes/Estacion3.jpeg"  class="img_c"></a>'+
                        '</div>'+
        '</div>'+
        '</div>';

   var contentString4 = '<div id="content">'+
```

```
        '<div id="siteNotice">'+
        '</div>'+
        '<h1 id="firstHeading" class="firstHeading">SAIH-E0004  Lomas de Las
Américas</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Ubicación: </b>' +
        'Lat:  19.6943212°, '+
        'Long:  -101.1414839°</p>'+
        '<div>'+
        '<a href="https://www.saihmorelia.com/copia-de-p-saih-e0002-1"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
        '</div>'+
        '<div>'+
        '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/gH8-
A15GVwjaNUrMDx-6GhnOoSI?embed=true"></iframe>'+
        '</div>'+
        '<div>'+
        //'<a><img src="../Imagenes/Estacion3.jpeg"  class="img_c"></a>'+
        '</div>'+
        '</div>'+
        '</div>';

   var contentString5 = '<div id="content">'+
        '<div id="siteNotice">'+
        '</div>'+
        '<h1 id="firstHeading" class="firstHeading">SAIH-E0005  Wenceslao
Victoria</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Ubicación: </b>' +
        'Lat:      19.7172263°, '+
        'Long:  -101.2239849°</p>'+
                        '<div>'+
                        '<a href="https://www.saihmorelia.com/copia-de-p-saih-e0003"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
        '</div>'+
                        '<div>'+
                        '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/iUdq8j1gC7KcK86ZtnZJ
bniGeX8?embed=true"></iframe>'+
                        '</div>'+
                        '<div>'+
                        //'<a><img src="../Imagenes/Estacion3.jpeg"  class="img_c"></a>'+
                        '</div>'+
        '</div>'+
```

```
        '</div>';

      var contentString6 = '<div id="content">'+
        '<div id="siteNotice">'+
        '</div>'+
        '<h1 id="firstHeading" class="firstHeading">SAIH-E0006  Centro
Histórico</h1>'+
        '<div id="bodyContent">'+
        '<p><b>Ubicación: </b>' +
        'Lat:      19.6979361°, '+
        'Long:  -101.1978357°</p>'+
                        '<div>'+
                        '<a href="https://www.saihmorelia.com/copia-de-p-saih-e0005"
onclick="window.open(this.href, this.target, width=600, height=400); return false;">Ir a los
Datos</a>'+
        '</div>'+
                        '<div>'+
                        '<iframe width="100%" height="130" frameborder="0"
src="https://industrial.ubidots.com/app/dashboards/public/widget/2VEjmUzDjTh3Kmgx6_
UraZMBAyI?embed=true"></iframe>'+
                        '</div>'+
                        '<div>'+
                        //'<a><img src="../Imagenes/Estacion3.jpeg"  class="img_c"></a>'+
                        '</div>'+
        '</div>'+
        '</div>';



      var infowindow = new google.maps.InfoWindow({
        content: contentString
      });


      var infowindow2 = new google.maps.InfoWindow({
        content: contentString2
      });

                var infowindow3 = new google.maps.InfoWindow({
        content: contentString3
      });

    var infowindow4 = new google.maps.InfoWindow({
        content: contentString4
      });

    var infowindow5 = new google.maps.InfoWindow({
```

```
    content: contentString5
  });

var infowindow6 = new google.maps.InfoWindow({
    content: contentString6
  });


    var marker = new google.maps.Marker({
    position: Camelinas,
                icon: {
                path: google.maps.SymbolPath.CIRCLE,
                scale: 5
               },
    map: map,
    title: 'SAIH-E0001'
  });
  marker.addListener('click', function() {
    infowindow.open(map, marker);
  });

                var marker2 = new google.maps.Marker({
    position: CU,
                icon: {
                path: google.maps.SymbolPath.CIRCLE,
                scale: 5
               },
    map: map,
    title: 'SAIH-E0002'
  });
  marker2.addListener('click', function() {
    infowindow2.open(map, marker2);
  });

              var marker3 = new google.maps.Marker({
    position: Lomasdelajoya,
                icon: {
                path: google.maps.SymbolPath.CIRCLE,
                scale: 5
               },
    map: map,
    title: 'SAIH-E0003'
  });
  marker3.addListener('click', function() {
    infowindow3.open(map, marker3);
  });
```

```
    var marker4 = new google.maps.Marker({
        position: Lomasamericas,
     icon: {
     path: google.maps.SymbolPath.CIRCLE,
     scale: 5
    },
        map: map,
        title: 'SAIH-E0004'
      });
      marker4.addListener('click', function() {
        infowindow4.open(map, marker4);
      });

    var marker5 = new google.maps.Marker({
        position: Wenceslao,
     icon: {
     path: google.maps.SymbolPath.CIRCLE,
     scale: 5
    },
        map: map,
        title: 'SAIH-E0005'
      });
      marker5.addListener('click', function() {
        infowindow5.open(map, marker5);
      })
    var marker6 = new google.maps.Marker({
        position: Centro,
     icon: {
     path: google.maps.SymbolPath.CIRCLE,
     scale: 5
    },
        map: map,
        title: 'SAIH-E0006'
      });
      marker6.addListener('click', function() {
        infowindow6.open(map, marker6);
      });
     }
    </script>
    <script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAYWglNL6JQImeRoUiJB4E9
GGi7dK3JCZc&callback=initMap">
    </script>
  </body>
</html>
```