



**INSTITUTO POLITÉCNICO NACIONAL**  
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS  
AVANZADAS  
**ACUSE REPORTE FINAL**  
**INGENIERÍA TELEMÁTICA**



El presente documento hace constar que se ha entregado el Reporte Final de Trabajo Terminal II (Trabajo Terminal I/Trabajo Terminal II) al Jurado para su defensa en la presentación Ordinaria (Ordinaria/Extraordinaria/ETS), del proyecto denominando:

NÚMERO DE REGISTRO (Número asignado en la Constancia de Registro de Protocolo)	TTT-2021/1-03
TÍTULO	
Modelos predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil.	

Proyecto integrado por los siguientes alumnos:

	NOMBRE COMPLETO (Iniciando por apellidos y después nombres)
Alumno 1	Chávez Galván Camilo Israel
Alumno 2	
Alumno 3	
Alumno 4	

A continuación firman de recibido todos los miembros del Jurado del proyecto:

 Escriba el texto aquí	Cédula: 5302307	Cédula: 11411519
M. en C. Paola Nayeli Cortez Herrera		M. en C. y T.E. Carlos Hernández Mejía
<b>PRESIDENTE</b> Nombre Completo del Presidente (Incluir el Grado Académico)		<b>SECRETARIO</b> Nombre Completo del Secretario (Incluir el Grado Académico)

 Cédula: 4369607	Cédula: 11050111
PhD. Miguel Félix Mata Rivera	M. en C. Roberto Eswart Zagal Flores
<b>ASESOR 1</b> Nombre Completo del Asesor 1 (Incluir el Grado Académico)	<b>ASESOR 2</b> Nombre Completo del Asesor 2 (Incluir el Grado Académico)
Interno <input checked="" type="checkbox"/> Externo <input type="checkbox"/>	Interno <input type="checkbox"/> Externo <input checked="" type="checkbox"/>

		Cédula: 08800765
		M. en C. Carlos Hernandez Nava
<b>ASESOR 3</b> Nombre Completo del Asesor 3 (Incluir el Grado Académico)	<b>SUPLENTE</b> Nombre Completo del Suplente (Incluir el Grado Académico)	
Interno <input type="checkbox"/> Externo <input type="checkbox"/>		

**NOTAS IMPORTANTES:**

- El Acuse de Reporte Final deberá entregarse en original al Secretario, este deberá incluir todas las firmas de los miembros del Jurado, incluyendo al Suplente, sin excepciones, en caso contrario el Departamento de Tecnologías Avanzadas no podrá programar la presentación del proyecto.
- Los alumnos tendrán la responsabilidad de notificar a todo el Jurado, incluyendo al Suplente, en caso de que el proyecto no se presentará en la evaluación correspondiente, esto lo deberán realizar con al menos dos días de anticipación a la fecha programada.
- En caso de tener menos de tres asesores, deberá dejar el espacio vacío.

“Modelos predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil”



PRESENTA  
CHÁVEZ GALVÁN CAMILO ISRAEL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA  
Y TECNOLOGÍAS AVANZADAS  
INSTITUTO POLITÉCNICO NACIONAL

PROYECTO TERMINAL

Que para obtener el título de:  
Ingeniero en Telemática

Ciudad de México, Diciembre 2021

“Modelos predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil”



PRESENTA  
CHÁVEZ GALVÁN CAMILO ISRAEL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y  
TECNOLOGÍAS AVANZADAS  
INSTITUTO POLITÉCNICO NACIONAL

PROYECTO TERMINAL

Que para obtener el título de:

Ingeniero en Telemática

Paola N. Cortez Herrera

Presidenta:  
M. en C. Paola Nayeli Cortez Herrera

Secretario:  
M. en C. y T.E. Carlos Hernández Mejía

Asesor:  
PhD. Miguel Félix Mata Rivera

Asesor:  
M. en C. Roberto Zagal Flores

# Índice General

<b>RESUMEN .....</b>	<b>10</b>
<b>ABSTRACT.....</b>	<b>11</b>
<b>AGRADECIMIENTOS.....</b>	<b>12</b>
<b>INTRODUCCIÓN .....</b>	<b>13</b>
<i>Pero ¿Qué es la contaminación del aire? .....</i>	<i>13</i>
<i>Monitoreo de niveles contaminantes en la Ciudad de México.....</i>	<i>13</i>
<b>CAPÍTULO I.....</b>	<b>15</b>
<b>PLANTEAMIENTO DEL PROBLEMA.....</b>	<b>15</b>
1.2.    PROPUESTA DE SOLUCIÓN .....	16
1.3.    FUNCIONAMIENTO GENERAL .....	20
1.4.    ALCANCES .....	20
1.5.    OBJETIVOS .....	21
1.5.1. <i>Objetivo general.....</i>	<i>21</i>
1.5.2 <i>Objetivos específicos.....</i>	<i>22</i>
1.6.    JUSTIFICACIÓN .....	22
1.7.    METODOLOGÍA .....	23
1.7.1. <i>Revisión Documental.....</i>	<i>23</i>
1.7.2. <i>Recolección de Datos.....</i>	<i>23</i>
1.7.3. <i>Procesamiento de Información .....</i>	<i>24</i>
1.7.4. <i>Ejecución de Pruebas .....</i>	<i>24</i>
1.8.    ESCENARIO DE PRUEBAS.....	24
<b>CAPÍTULO II.....</b>	<b>27</b>
<b>ESTADO DEL ARTE .....</b>	<b>27</b>
2.1    RED AUTOMÁTICA DE MONITOREO ATMOSFÉRICO.....	27
2.2    BREATHE LONDON (RESPIRA LONDRES).....	28
2.3    OAKLAND ENVIRONMENTAL DEFENSE FUND PROJECT.....	29
2.4    ESTUDIO DE LA CONTAMINACIÓN AMBIENTAL DE LA CIUDAD DE MÉXICO MEDIANTE LOS MODELOS REDES NEURONALES ARTIFICIALES Y ALGORITMOS GENÉTICOS.....	30
2.5    TABLA COMPARATIVA DE CADA UNO DE LOS ENUNCIADOS EN EL ESTADO DEL ARTE.....	31
<b>CAPÍTULO III .....</b>	<b>33</b>
<b>MARCO TEÓRICO.....</b>	<b>33</b>
3.1    IOT (INTERNET DE LAS COSAS).....	33
3.1.1 <i>Red 4G y 5G.....</i>	<i>33</i>
3.2    MACHINE LEARNING.....	34

3.2.1	<i>Ensemble Learning (Aprendizaje conjunto)</i> .....	35
3.3	LA NUBE (CLOUD) .....	35
3.3.1	<i>Almacenamiento en la nube</i> .....	35
3.4	MONITOREO DE CALIDAD DEL AIRE.....	35
3.4.1	<i>Dióxido de Nitrógeno</i> .....	36
3.4.2	<i>Partículas Suspendidas</i> .....	36
3.4.3	<i>Ozono</i> .....	36
3.4.4	<i>Monóxido de carbono y Dióxido de Azufre</i> .....	37
3.5	IMPLEMENTACIÓN DEL MARCO TEÓRICO.....	37
<b>CAPITULO IV .....</b>		<b>39</b>
<b>ANÁLISIS.....</b>		<b>39</b>
4.1	INTRODUCCIÓN .....	39
4.2	IDENTIFICACIÓN DE REQUERIMIENTOS.....	39
4.3	ANÁLISIS DE REQUERIMIENTOS.....	40
4.3.1	<i>Ámbito del sistema</i> .....	40
4.3.2	<i>Definiciones, acrónimos y abreviaturas</i> .....	41
4.3.3	<i>Perspectiva del producto</i> .....	41
4.3.4	<i>Funcionamiento del producto</i> .....	42
4.3.5	<i>Características de los usuarios</i> .....	43
4.3.6	<i>Restricciones</i> .....	43
4.3.7	<i>Suposiciones y dependencias</i> .....	44
4.3.8	<i>Requisitos Futuros</i> .....	44
4.4	REQUISITOS ESPECÍFICOS.....	44
4.4.1	<i>Hardware</i> .....	44
4.4.2	<i>Interfaz de comunicación</i> .....	45
4.4.3	<i>Requerimientos Funcionales</i> .....	45
4.4.4	<i>Requerimientos No Funcionales</i> .....	51
4.5	DIAGRAMAS DE CASO DE USO .....	52
4.6	ANÁLISIS DE HARDWARE A UTILIZAR .....	60
4.6.1	<i>Sensor de PM<sub>2.5</sub>/PM<sub>10</sub></i> .....	61
4.6.3	<i>Módulo GPS</i> .....	62
4.6.4	<i>Placa Base WiFi</i> .....	63
4.6.5	<i>Placa Base GSM</i> .....	64
4.7	ANÁLISIS DE SERVICIOS IoT EN LA NUBE.....	65
4.8	ANÁLISIS DEL MODELO DE MACHINE LEARNING .....	66
4.8.1	<i>Recurrent Neural Network (RNN)</i> .....	66
4.8.2	<i>Long Short-Term Memory (LSTM)</i> .....	67
4.8.3	<i>Stacking Ensemble</i> .....	69
<b>CAPÍTULO V.....</b>		<b>70</b>
<b>DISEÑO.....</b>		<b>70</b>
5.1	DISEÑO DE LA ARQUITECTURA DEL SISTEMA .....	70
5.2	ESTRUCTURA DEL DOCUMENTO JSON ALMACENADO EN LA BASE DE DATOS.....	71
5.2.1	<i>Diccionario de datos del documento JSON</i> .....	72
5.3	DIAGRAMAS DE SECUENCIA .....	73

5.4 DISEÑO DE LAS ESTACIONES DE MONITOREO .....	75
<b>CAPÍTULO VI .....</b>	<b>77</b>
<b>IMPLEMENTACIÓN.....</b>	<b>77</b>
6.1 ESTACIONES DE MONITOREO (CÓDIGO).....	77
6.1.1 <i>Estación de monitoreo estacionaria (WiFi)</i> .....	78
6.1.2 <i>Estación de monitoreo móvil (GSM)</i> .....	82
6.2 CONFIGURACIÓN DE AZURE IoT HUB .....	86
6.2.1 <i>Configuración de Azure IoT Hub (Broker)</i> .....	86
6.2.2 <i>Base de Datos No Relacional CosmosDB</i> .....	88
6.2.3 <i>Procesamiento de datos con Stream Analytics Job</i> .....	90
6.3 MODELO DE MACHINE LEARNING .....	93
6.3.1 <i>Preparación de los datos</i> .....	93
6.3.2 <i>Ambiente en Python</i> .....	94
<b>CAPÍTULO VII.....</b>	<b>99</b>
<b>PRUEBAS Y RESULTADOS.....</b>	<b>99</b>
7.1 ESTACIONES DE MONITOREO .....	99
7.2 TRANSMISIÓN DE DATOS MEDIANTE PROTOCOLO MQTT .....	104
7.3 RECEPCIÓN Y MONITOREO DE DATOS EN AZURE IoT HUB. ....	105
7.4 RECEPCIÓN DE DATOS EN COSMOSDB. ....	107
7.5 RESULTADOS DE LAS PRUEBAS .....	108
7.5.1 <i>Resultados estación de monitoreo móvil (recorridos)</i> .....	110
7.6 MODELO PREDICTIVO (LSTM).....	111
7.6.1 <i>Estación de monitoreo Fija (Evelina)</i> .....	111
7.6.2 <i>Estación de monitoreo móvil (Ariatne)</i> .....	114
<b>CAPÍTULO VIII.....</b>	<b>118</b>
<b>VALIDACIÓN .....</b>	<b>118</b>
8.1 OBJETIVOS ESPECÍFICOS.....	118
8.2 OBJETIVO GENERAL .....	122
<b>CONCLUSIONES .....</b>	<b>124</b>
<b>TRABAJOS FUTUROS .....</b>	<b>125</b>
<b>REFERENCIAS.....</b>	<b>127</b>

# Índice de Ilustraciones

Ilustración 1 Fuente: base de datos histórica del Sistema de Monitoreo Atmosférico de la Ciudad de México.....	15
Ilustración 2 Interacción entre los componentes del sistema.....	18
Ilustración 3 Recorrido propuesto para estación de monitoreo móvil.....	25
Ilustración 4 Distribución de las redes de monitoreo del SIMAT, Fuente: SEDEMA	28
Ilustración 5 Mapa de la red de monitoreo de Breathe London en tiempo real, Fuente: Breathe London.....	29
Ilustración 6 Mapa de muestras de Carbono Negro en la ciudad de Oakland, Fuente: EDF .....	30
Ilustración 7 Caso de uso Monitoreo de la estación .....	52
Ilustración 8 Caso de Uso Recepción de mensajes.....	54
Ilustración 9 Caso de uso Procesamiento y almacenamiento de datos .....	55
Ilustración 10 Caso de uso modelo de machine learning .....	57
Ilustración 11 Caso de uso Análisis de datos en Azure .....	59
Ilustración 12 Red Neuronal Persistente con Iteración Fuete: Understanding LSTM. [30] .....	67
Ilustración 13 Iteración del módulo en una RNN, Fuente: Understanding LSTM. [30] .....	68
Ilustración 14 Representación de una LSTM y su módulo de iteraciones. Fuente: Understanding LSTM. [30] .....	68
Ilustración 15 Arquitectura del sistema .....	70
Ilustración 16 Descripción del documento JSON .....	71
Ilustración 17 Diagrama de secuencia de la estación de monitoreo .....	73
Ilustración 18 Diagrama de secuencia de la arquitectura de Azure .....	74
Ilustración 19 Diagrama de secuencia del entrenamiento del modelo.....	75
Ilustración 20 Diseño de estación de monitoreo WiFi y GSM.....	76
Ilustración 21 Certificado de la tarjeta de Arduino.....	86
Ilustración 22 Creación de recurso de IoT Hub.....	87
Ilustración 23 Configuración del IoT Hub mediante el hash de las estaciones de monitoreo.....	87
Ilustración 24 Vista de la consola y recepción de datos .....	88
Ilustración 25 Selección de API para CosmosDB.....	89
Ilustración 26 Creación de recurso de CosmosDB .....	89

Ilustración 27 Creación de Base de Datos en el Data Explorer de CosmosDB .....	90
Ilustración 28 Configuración del Stream Analytics Job .....	91
Ilustración 29 Configuración de la entrada para el Stream Analytics .....	91
Ilustración 30 Configuración para la salida del Stream Analytics .....	92
Ilustración 31 Creación de query para el Stream Analytics Job.....	92
Ilustración 32 Verificación de conexión a CosmosDB mediante el connection string usando la herramienta de Data Migration Tool.....	93
Ilustración 33 Paso 1 del modelo LSTM, Fuente: Understanding LSTM. [30] .....	96
Ilustración 34 Paso 2 del modelo LSTM, Fuente: Understanding LSTM. [29] .....	97
Ilustración 35 Paso 3 del modelo LSTM, Fuente: Understanding LSTM. [29] .....	97
Ilustración 36 Paso Final del modelo LSTM, Fuente: Undestanding LSTM. [29] .....	98
Ilustración 37 Estación de monitoreo estacionaria.....	100
Ilustración 38 Estación de monitoreo Evelina.....	101
Ilustración 39 Estación de monitoreo Inaella. ....	101
Ilustración 40 Estación de monitoreo móvil.....	102
Ilustración 41 Estación de monitoreo móvil sobre el vehículo a utilizar para los recorridos.....	103
Ilustración 42 Panel de información de Hologram.....	104
Ilustración 43 Conexiones de la estación de monitoreo a la red GSM.....	105
Ilustración 44 Recepción de cadena de mensajes en consola.....	106
Ilustración 45 Monitoreo de mensajes usados por el broker. ....	106
Ilustración 46 Monitoreo de mensajes en el broker. ....	107
Ilustración 47 Data Explorer de CosmosDB .....	107
Ilustración 48 Mapa de estaciones en la Ciudad de México. ....	109
Ilustración 49 Promedio de lecturas estación de monitoreo móvil.....	110
Ilustración 50 Muestras de PM25 en estación de monitoreo fija. ....	111
Ilustración 51 Datos de validación de PM25 para estación de monitoreo fija. ....	112
Ilustración 52 Resultados del desempeño del modelo para pm25 en estación fija....	112
Ilustración 53 Historia de muestras PM10 para la estación de monitoreo fija.....	113
Ilustración 54 Resultados del entrenamiento para PM10 en estación fija, verde: predicción, naranja: valores reales. ....	113
Ilustración 55 Perdida obtenida del modelo para PM10 con la estación fija. ....	114
Ilustración 56 Historia de datos de PM25 para estación móvil. ....	114
Ilustración 57 Datos de validación de PM25 para estación móvil. Verde: predicción, Naranja: valores reales.....	115

Ilustración 58 Perdida del modelo para PM25 en la estación móvil .....	115
Ilustración 59 Historia de datos de PM10 para la estación de monitoreo móvil.....	116
Ilustración 60 Validación de datos del modelo con PM10 en la estación de monitoreo móvil. Verde: predicciones y Naranja: valores reales.....	116
Ilustración 61 Perdida obtenida en el modelo para PM10 en la estación móvil.....	117
Ilustración 62 Estaciones de monitoreo móvil y fija.....	119
Ilustración 63 Dashboard de Hologram usando la red de AT6T para el envío de datos usando la red 3G/4G.....	120
Ilustración 64 Arquitectura de Azure .....	120
Ilustración 65 Data Explorer de CosmosDB .....	121
Ilustración 66 Modelo LSTM, datos de validación.....	121
Ilustración 67 Parte del código del modelo LSTM.....	122

# Índice de Tablas

Tabla 1 Tabla comparativa de los elementos del Estado del Arte .....	32
Tabla 2 - Características del usuario Administrador .....	43
Tabla 3 - Características del usuario Supervisor.....	43
Tabla 4 - Requerimiento Funcional 01 .....	45
Tabla 5 - Requerimiento Funcional 02 .....	46
Tabla 6 - Requerimiento Funcional 03 .....	46
Tabla 7 - Requerimiento Funcional 04 .....	46
Tabla 8 - Requerimiento Funcional 05 .....	47
Tabla 9 - Requerimiento Funcional 06 .....	47
Tabla 10 - Requerimiento Funcional 07.....	48
Tabla 11 - Requerimiento Funcional 08.....	48
Tabla 12 - Requerimiento Funcional 09.....	49
Tabla 13 - Requerimiento Funcional 10.....	49
Tabla 14 - Requerimiento Funcional 11.....	50
Tabla 15 - Requerimiento Funcional 12.....	50
Tabla 16 - Requerimiento no funcional 01 .....	51
Tabla 17 - Requerimiento No Funcional 02 .....	51
Tabla 18 - Descripción de caso de uso CU01 .....	53
Tabla 19 Descripción de caso de uso CU02.....	55
Tabla 20 Descripción del caso de uso 03.....	56
Tabla 21 Descripción de caso de uso 04.....	58
Tabla 22 Descripción de caso de uso 05.....	60
Tabla 23 Descripción de sensor de partículas suspendidas.....	61
Tabla 24 Descripción del módulo GPS .....	62
Tabla 25 Descripción de placa base WiFi.....	64
Tabla 26 Descripción de placa base GSM.....	65
Tabla 27 Descripción de servicios de IoT en la nube .....	66
Tabla 28 Diccionario de datos con descripción del documento JSON .....	73
Tabla 29 Resultados estaciones de monitoreo.....	108
Tabla 30 Relación de calidad del aire.....	110

# Resumen

El presente documento describe la propuesta del diseño e implementación de un sistema capaz de crear modelos predictivos a partir de técnicas de machine learning mediante dos módulos de monitoreo de calidad de aire móviles que logran recabar información de los niveles de contaminación a nivel de calle, ya que actualmente no se cuentan con estos datos debido a que la red de monitoreo de la CDMX lo efectúa en alturas superiores a los 3 metros. Los módulos están diseñados en dos tipos diferentes; el primero es fijo a nivel de calle y el segundo montado sobre un vehículo, esto con la finalidad de obtener una muestra amplia del comportamiento de las partículas contaminantes en un área delimitada. Los datos son transmitidos en tiempo real usando una red de cobertura 4G implementando la tecnología del IoT (Internet de las cosas) hacia una nube de almacenamiento para su después procesamiento usando técnicas de *machine learning* que permitan crear modelos predictivos, contribuyendo así con una alternativa de monitoreo del medio ambiente en la Ciudad de México.

## Palabras clave

Monitoreo de calidad del aire, IoT, Machine Learning, Nube, 4G/5G

# Abstract

This work describes the design and implementation of a system capable of creating predictive models based on machine learning techniques using two mobile air quality monitoring modules that sense pollution contaminants at street level, currently street level pollution data is not available since the Mexico City monitoring network measures air quality index at heights of 3 meters. There are two types of modules, the first is fixed at street level in public spaces whereas the second is mounted on a vehicle to obtain a better Air Quality Index samples in the city. Data is transmitted in real time using 4G coverage area network implementing IoT (Internet of Things) to store the data on the cloud for later processing using machine learning techniques that will help to create predictive models and therefore being able to provide a new alternative of air quality monitoring through new technologies in Mexico City.

## Keywords

Air Quality Index monitoring, IoT, Machine Learning, Cloud, 4G/5G

## Agradecimientos

A mi papá, mi mamá y mis hermanas, que han sido un apoyo emocional incondicional durante cada una de las etapas de mi vida, porque a pesar de todas las circunstancias y adversidades siempre creyeron en mi y apoyaron cada una de mis decisiones, porque gracias a ustedes estoy en el lugar que quiero estar.

A mis profesores, en la UPIITA y en cada una de las etapas de mi vida académica, que gracias a ustedes he aprendido grandes lecciones que sin duda forman parte de lo que he logrado hasta ahora.

Y al Instituto Politécnico Nacional por brindarme las herramientas para crecer como persona y como profesional.

Agradecimiento especial al proyecto CONACYT 7051.

# Introducción

A pesar de décadas de esfuerzos en el combate para la reducción de partículas contaminantes en el aire en todo el mundo, parece ser que éstos se encuentran en declive como una tendencia mundial. Las razones de ello, entre otras, son propiciadas por el cambio climático, incendios forestales y el incremento desmedido del consumo humano impulsado por el incremento poblacional. [1]

Pero ¿Qué es la contaminación del aire?

Es una combinación de partículas y gases que pueden alcanzar concentraciones peligrosas tanto en ambientes abiertos como cerrados, los efectos negativos como consecuencia de estos niveles van desde riesgos de salud para la población hasta el incremento de temperaturas. Dentro de los principales contaminantes de los cuales se necesita un monitoreo constante se encuentran el dióxido de azufre ( $\text{SO}_2$ ), monóxido de carbono (CO), dióxido de nitrógeno ( $\text{NO}_2$ ), ozono ( $\text{O}_3$ ), partículas suspendidas ( $\text{PM}_{10}$ ,  $\text{PM}_{2.5}$ ), entre otras.

Con datos de la Organización Mundial de la Salud (OMS) se sabe que por consecuencia de la contaminación del aire mueren en un estimado 7 millones de personas mundialmente cada año, y que 9 de cada 10 personas respira aire que excede las pautas establecidas por la OMS, los países más pobres y en vías de desarrollo son los más afectados. [2]

Monitoreo de niveles contaminantes en la Ciudad de México

Al igual que en ciudades de gran densidad, especialmente aquellas localizadas en valles como lo es la Ciudad de México, se perciben problemas en el control de los niveles contaminantes dadas sus características geográficas, principalmente con Ozono y partículas suspendidas, gracias a los esfuerzos gubernamentales se ha logrado reducir el

nivel de emisiones desde la década de 1990 de forma gradual, sin embargo, el crecimiento poblacional ha limitado la tasa de reducción en los últimos años. [3]

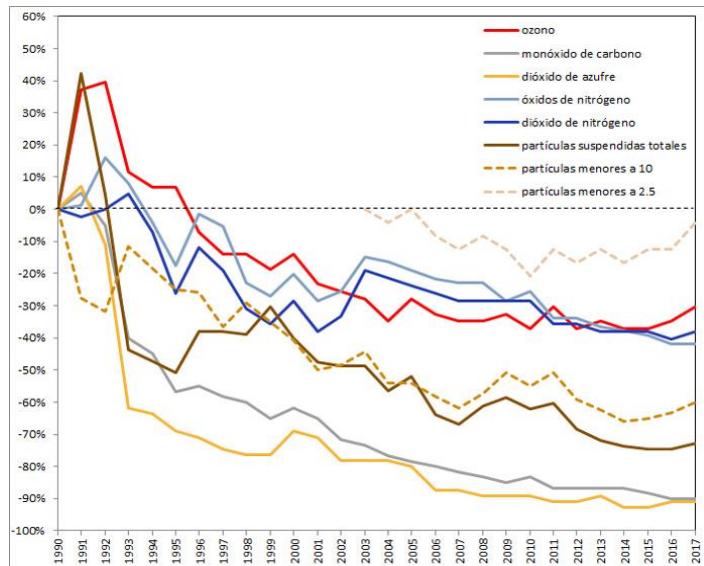
Actualmente la Ciudad de México cuenta con el Sistema de Monitoreo Atmosférico (SIMAT) cuya implementación responde a los esfuerzos por cumplir con las normativas en cuanto al control de los niveles contaminantes y prevenir oportunamente los riesgos derivados de altas concentraciones de contaminantes que puedan poner en riesgo a la población, éste a su vez cuenta con el subsistema de Red Automática de Monitoreo Atmosférico (RAMA) cuya función es la medición de dióxido de azufre, monóxido de carbono, dióxido de nitrógeno, ozono, PM<sub>10</sub> y PM<sub>2.5</sub> y está integrada por 34 estaciones, así como el Centro de Información de la Calidad del Aire (CICA) que es el repositorio de todos los datos generados por el Sistema de Monitoreo Atmosférico y es responsable de la validación, procesamiento y difusión de la información generada por el programa de monitoreo. [4]

Esta red permite generar un mapa del comportamiento de los niveles de contaminación, mediante estaciones fijas a determinadas alturas, este sistema se ha usado durante décadas y no ha cambiado de manera significativa en contraste con el acelerado crecimiento de la ciudad y sus constantes cambios a lo largo de los años, lo que pone al actual sistema de monitoreo incógnitas sobre su escalabilidad y precisión.

# Capítulo I

## Planteamiento del problema

Si bien los actuales sistemas de monitoreo de la Ciudad de México han logrado reducir en cierta medida los niveles de contaminación desde su implementación como se muestra en la *ilustración 1*, es claro que en los últimos 10 años se ha visto una desaceleración en la disminución de niveles contaminantes y en algunos casos ha habido un ligero incremento, lo cual plantea incógnitas sobre su actual efectividad frente a los presentes desafíos que representa el rápido crecimiento de la ciudad. [4]



*Ilustración 1* Fuente: base de datos histórica del Sistema de Monitoreo Atmosférico de la Ciudad de México.

Esta red ha logrado que se genere un espectro suficiente del nivel de calidad de aire de la ciudad sobre los principales contaminantes perjudiciales para sus habitantes, pero impide una medición a mayor profundidad y no permite tener una visión más clara de lo que ocurre en los niveles de mayor importancia; las actuales 34 estaciones de monitoreo de la Ciudad de México no se encuentran con respecto al nivel de calle, que

es el nivel donde las personas respiran los contaminantes, de aquí la necesidad de implementar sistemas que logren generar un mapa más amplio del nivel de contaminación donde más se le necesita. [5]

En una ciudad cuya área metropolitana tiene una población de 27,782,378 personas, lo cual representa un aumento del 0.51% respecto al año anterior (2019) y general representa un incremento del 0.41% anual [6], el seguir contando con medidas de monitoreo iguales a las de hace tres décadas que no presentan una evolución al mismo ritmo que el campo de estudio, hace pensar que probablemente sea tiempo de implementar un sistema que tenga un alcance mayor, y una precisión en niveles de los cuales no se tienen datos. Durante años el monitoreo ha estado limitado al uso de estaciones fijas cuyo alcance se ve condicionado y rebasado, esto atribuye una perspectiva limitada de lo que sucede en una ciudad que crece vertiginosamente.

Ante el escenario planteado se llega a la siguiente cuestión: ¿Cómo diseñar e implementar un sistema de monitoreo de calidad del aire en cuanto al nivel de calle que sea escalable y permita generar predicciones del comportamiento de los contaminantes?

## 1.2. Propuesta de Solución

Con base en el escenario anteriormente descrito, es evidente la necesidad de la implementación de un sistema que pueda adaptarse al constante cambio que presentan las urbes, como lo es el caso de la Ciudad de México, de tal manera que se logre obtener escalabilidad, confiabilidad y precisión en los datos monitoreados para así crear modelos predictivos que permitan describir de mejor forma el comportamiento de los contaminantes en el aire de la ciudad.

La era de las ciudades inteligentes cada vez está más presente en la vida de todos y con ello el concepto de Internet de las cosas (IoT), esto significa que se puede aprovechar el crecimiento de la red para incrementar la cobertura, reducir costos y proveer mayores niveles de flexibilidad en cuanto a los tradicionales sistemas de monitoreo de contaminantes las cuales como se ha mencionado en la lectura están limitadas a estaciones fijas de grandes dimensiones.

El avance de la tecnología permite acceder a sensores de contaminantes cada vez menos costosos y fáciles de manipular, la propuesta radica en la implementación de un sistema de monitoreo móvil que puede instalarse en vehículos automotores que en general tengan un constante movimiento por la urbe, el cual puede ser el caso de vehículos de uso gubernamental como patrullas, vehículos de pasajeros pertenecientes al sistema de movilidad integrada de la ciudad, o incluso vehículos particulares cuyo uso sea constante en la metrópoli, por ejemplo, Uber<sup>1</sup> o inclusive el vehículo de Google Street View<sup>2</sup>, de tal manera que se asegure la cobertura en la urbe y de esta manera crear un mapa que contenga un espectro mucho más grande de los contaminantes comparado al que se tiene actualmente; así como pequeñas estaciones que pueden ser instaladas en áreas públicas como parques, plazas y hospitales y que obtengan un rango mucho más amplio de los contaminantes que llegan directamente a los ciudadanos.

Dichas estaciones móviles tienen integrado un sistema de comunicación para el envío de datos mediante la red de cobertura 4G de la ciudad que permite la fácil integración del sistema a la ciudad inteligente y así concentrar los datos de las estaciones móviles para poder ser procesados.

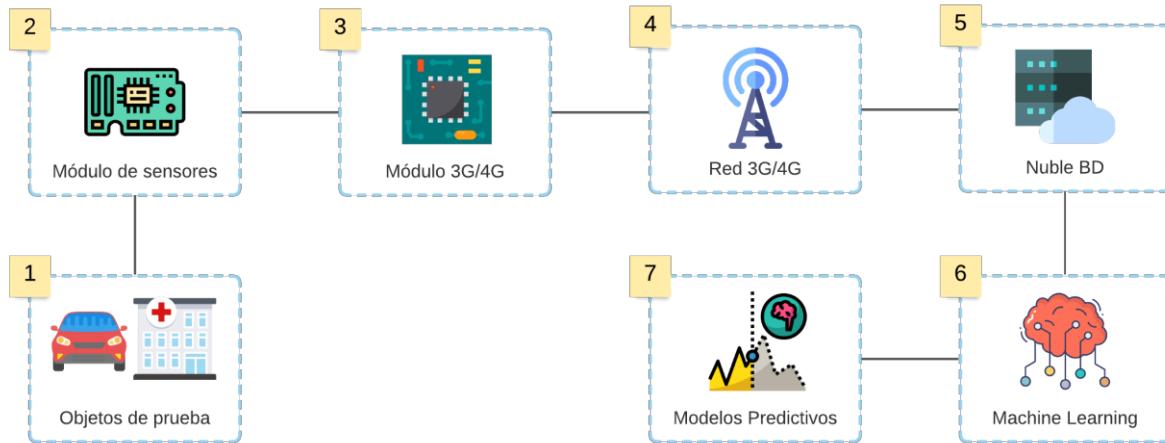
La cantidad de datos recibida en el modelo propuesto abre las puertas para la creación de modelos predictivos mediante técnicas de *machine learning* que permiten pronosticar el comportamiento de los contaminantes con base en los patrones recabados por el sistema.

---

<sup>1</sup> Uber: Plataforma de movilidad privada para las masas.

<sup>2</sup> Vehículo de Google Street View: Vehículo cuyo propósito es crear una vista virtual de las calles y caminos del mundo y que recorre de manera constante los mismos.

En la *Ilustración 2* se observa el diagrama de interacción del sistema entre cada una de las etapas que lo componen.



*Ilustración 2 Interacción entre los componentes del sistema*

## 1. Objetos de prueba

Los objetos de prueba como se mencionó anteriormente pueden ser vehículos que tengan un recorrido constante en las calles de la ciudad, así como zonas públicas en las que se pueden instalar el módulo para el monitoreo de los contaminantes.

## 2. Módulo de sensores

La tarea de este módulo es realizar la toma de muestras de partículas suspendidas ( $PM_{10}$ ,  $PM_{2.5}$ ), cada hora durante todo el día para la estación fija y en un promedio de 2 muestras por día para la estación móvil, el rango de toma de muestras seleccionado para esta última se definió de esa manera debido a que se necesita un constante monitoreo y verificación de los datos obtenidos por los sensores al encontrarse en movimiento, dicha supervisión de la estación podría evitarse si se tuviese acceso a sensores de mayor calidad cuyo precio está fuera del presupuesto del presente proyecto, pero que se puede tomar como mejora a futuro.

En Proyecto Terminal I, se definió el uso de un sensor de CO (Monóxido de Carbono) aunado al sensor de partículas suspendidas, en las pruebas iniciales se implementó dicho sensor, pero se presentaron múltiples problemas con la calibración del mismo lo cuál podría afectar la confiabilidad del sistema, es por ello que se optó por eliminarlo del sistema y continuar solamente con el sensor de partículas suspendidas que si mostraba confiabilidad en la toma de muestras.

### 3. Módulo 4G

Aquí el módulo tiene la tarea de enviar las muestras tomadas por el módulo de sensores a la red.

### 4. Red 4G

Los datos enviados a la red por el módulo 4G viajarán por la red de cobertura a utilizar para poder ser transportados en tiempo real.

### 5. Nube BD

Aquí se hace uso de un servicio en la Nube como es el de Azure que sirve como Hub para la recepción de datos por las estaciones de monitoreo y una Base de Datos de toda la información recabada por los módulos de monitoreo, se seleccionó dicho proveedor de la Nube debido a la fácil implementación, escalabilidad y precio que esta supone para efectos del presente trabajo.

### 6. Machine Learning

Usando técnicas de Machine Learning se realizó un procesado de la información para determinar modelos predictivos que sirvan como referencia para saber el comportamiento de los contaminantes.

### 7. Modelos Predictivos

Por último, se presentan los modelos predictivos resultantes del monitoreo constante de cada uno de los módulos para su interpretación y evaluación, dichas predicciones no son comparables entre si debido a la diferencia de toma de muestras ya que no representan un sistema de monitoreo equiparable (entre las estaciones fija y móvil).

### 1.3. Funcionamiento General

Con base en el diagrama de la ilustración 1 se describe de manera general el funcionamiento del sistema.

1. Se realizó la instalación del módulo de monitoreo que está conformado por el módulo de sensores y el módulo 4G sobre el objeto de prueba, que está dividido en dos estaciones, una a nivel de calle y un módulo de monitoreo adaptado sobre un automóvil.
2. La tarea de estas estaciones de monitoreo móvil es el de enviar datos en tiempo real usando la red de cobertura 4G de las muestras obtenidas por las estaciones de monitoreo.
3. Los datos son recibidos en la nube para su procesamiento y posterior almacenamiento.
4. El procesamiento y análisis de los datos se hizo mediante la implementación de técnicas de *machine learning* que permiten crear modelos predictivos a partir de los datos que serán suministrados por nuestra Base de Datos.
5. Se presentan los resultados del algoritmo para crear el modelo predictivo que logre interpretar el futuro comportamiento de los niveles de contaminantes.

### 1.4. Alcances

- Se creó un prototipo de los módulos de monitoreo que serán instalados en los objetos de prueba.
- Se realizó la instalación de un solo prototipo de estación de monitoreo móvil en un vehículo no motorizado que tendrá la tarea de tomar hasta 2 muestras por día.
- Se implementaron tres estaciones de monitoreo con respecto al nivel de calle, una de ellas fue instalada en la fachada de una casa mientras que las dos restantes fueron instaladas en la UPIITA (Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas) y el CIC (Centro de Investigación en Computación) respectivamente para la toma de muestras de manera fija y periódica.

- Las estaciones con respecto al nivel de calle funcionan con una red Wi-Fi o en su defecto la red 4G que garantice la cobertura en la ciudad usando el servicio de AT&T México.
- Se implementó un servicio en la nube que permite realizar la recepción y procesamiento de las muestras proporcionadas por las estaciones de monitoreo móvil, así como su almacenamiento, el servicio a elegido es el de Azure de Microsoft dadas las facilidades que representan su instalación, escalabilidad y desempeño.
- Se realizó la implantación de un algoritmo para la creación de modelos predictivos mediante técnicas de *machine learning* (*Long Short Term Memory*) que incluye diversas técnicas de clasificadores para lograr un modelo más preciso.
- Se presenta la interpretación del algoritmo con un tiempo de tomas de muestras de aproximadamente 2 meses.
- 

## 1.5. Objetivos

Los objetivos del presente trabajo pretenden indicar los alcances que el sistema propuesto pretende alcanzar y definir las limitaciones a las que se atendrá.

### 1.5.1. Objetivo general

Implementar un sistema de monitoreo de calidad de aire móvil que sea escalable, y permita reducir costos que den como resultado un espectro mucho más amplio del comportamiento de los contaminantes criterio con respecto al nivel calle de forma periódica (24 veces al día) para la estación fija y hasta 2 tomas de muestras por día para la estación de monitoreo móvil en un área delimitada, y que con base en los datos recabados se creen modelos predictivos mediante técnicas de *Machine Learning* que logren medir la confiabilidad de los datos recabados y la confiabilidad del sistema en el presente proyecto.

### 1.5.2 Objetivos específicos

1. Integración de un módulo de sensores que realicen tomas de muestras de manera periódica de partículas suspendidas ( $PM_{10}$ ,  $PM_{2.5}$ ).
2. Se integró al módulo de sensores un módulo que permita transmitir la información obtenida por los sensores en tiempo real usando la red 4G o en su caso WiFi.
3. Implementación de un servicio de Base de Datos en la nube usando la plataforma de Azure que almacene todas las muestras recolectadas por el sistema de monitoreo.
4. Creación de un algoritmo para obtener modelos predictivos mediante técnicas de Machine Learning.
5. Interpretar los datos obtenidos del modelo predictivo para comprobar la eficacia del sistema.

### 1.6. Justificación

Durante décadas, la Ciudad de México ha sido considerada una ciudad con un nivel alto de contaminación del aire [4], y aunque si bien los esfuerzos por reducir los volúmenes de contaminantes han sido grandes, el problema sigue creciendo no solo en el país, sino en todo el mundo [2], la motivación de un proyecto de esta naturaleza surge de la creciente demanda por nuevas tecnologías como lo son el IoT y el Machine Learning para la resolución de problemas [7], y como su implementación puede mejorar significativamente la vida de miles de personas, inspirado en los esfuerzos de organizaciones no gubernamentales para crear sistemas inteligentes de monitoreo de calidad del aire como lo son el del *Environmental Defense Fund* en Londres [5] es que este proyecto propone la creación de un sistema de IoT que logre crear una red de monitoreo inteligente de contaminantes en el aire en tiempo real, que sea escalable y logre contribuir como una alternativa en la mejora de estrategias para la reducción de los niveles de contaminación en la Ciudad de México.

## 1.7. Metodología

En este apartado se explica la metodología que se siguió para el desarrollo del sistema propuesto.

### 1.7.1. Revisión Documental

Esta investigación partió de una revisión extensa de artículos científicos, tesis y proyectos alrededor del mundo cuyo propósito sean los de crear sistemas e implementación de técnicas para lograr un impacto en los esfuerzos por reducir nuestro impacto ambiental en el planeta, esto con la finalidad de contar con referencias de proyectos e investigaciones que fundamenten el propuesto en este documento. El criterio de búsqueda al final se redujo a aquellos proyectos e investigaciones que involucraran una implementación de IoT y Machine Learning como propuesta para el control y monitoreo de los niveles de contaminación en las ciudades.

Dentro de los objetivos de la revisión están la identificación de mejoras que se pueden aplicar a un sistema para hacerlo escalable, accesible y tenga un impacto dentro de las propuestas para la reducción de contaminación.

### 1.7.2. Recolección de Datos

Los datos de entrada para el presente proyecto son niveles de PM<sub>10</sub> y PM<sub>2.5</sub> proporcionados por los sensores en la placa base, un módulo GPS que proporciona las coordenadas de las mediciones cada hora, en conjunto con datos del sensor y la hora en que se enviaron los paquetes.

Los umbrales de precisión de los sensores y todo lo relacionado a la forma en la que estos realizan las mediciones se detallan en el Capítulo IV: Análisis.

La transmisión de datos se realizó usando el protocolo MQTT y la red WiFi / GSM disponible para cada estación de monitoreo.

### 1.7.3. Procesamiento de Información

El procesamiento de la información de los datos se describe de manera más detallada en el Capítulo IV: Análisis, sin embargo como preámbulo a este capítulo el procesamiento de los datos recabados por los sensores será usando una Arquitectura en la Nube mediante un servicio de Azure llamado IoT Hub que permite la recepción de datos, el procesamiento y análisis usando Azure Stream Analytics y el almacenamiento usando una Base de Datos no relacional, para su después procesamiento alimentando el modelo de machine learning propuesto para la creación del modelo predictivo.

### 1.7.4. Ejecución de Pruebas

Las pruebas realizadas para comprobar el funcionamiento del sistema se describen en la sección 1.8. Escenario de Pruebas.

Finalmente, la validación de los resultados se realiza mediante una comparación entre los resultados obtenidos del escenario de pruebas con los objetivos tanto generales como particulares propuestos en el presente trabajo.

## 1.8. Escenario de pruebas

El escenario de pruebas está diseñado para transmitir mediciones de datos mediante el módulo de monitoreo que está integrado por los sensores de medición y el módulo de acceso a la red WiFi / GSM, la transmisión de datos fue hecha en una frecuencia de 24 veces al día para la estación fija y un recorrido de 2 veces al día cada 5 segundos para la estación móvil, es decir que en el caso de la estación móvil cada hora se recibieron datos en tiempo real a la nube de almacenamiento durante todo el tiempo que duraron las pruebas y en el caso de la estación móvil cada 5 segundos durante el tiempo que tomó realizar el recorrido.

El recorrido propuesto para la estación de monitoreo móvil es el siguiente:

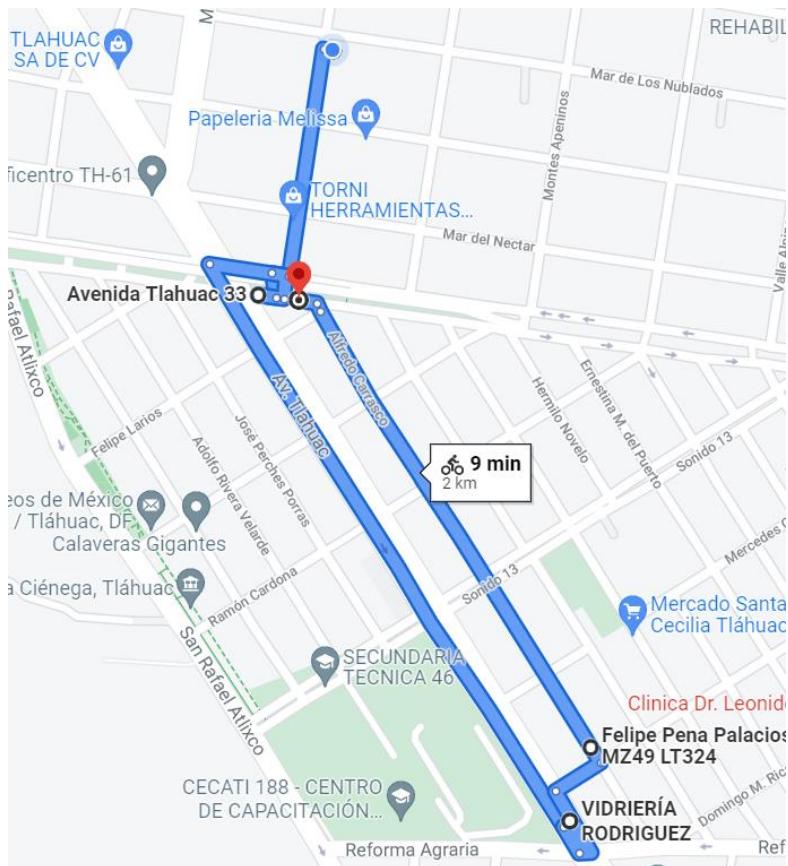


Ilustración 3 Recorrido propuesto para estación de monitoreo móvil.

El recorrido fue realizado con un vehículo no motorizado (bicicleta).

Los datos por monitorear como ya se ha mencionado son los de partículas suspendidas PM<sub>10</sub> y PM<sub>2.5</sub>, éstos fueron obtenidos mediante los sensores que se implementaron en el módulo principal y que han sido transmitidos a través de la red.

Se instalaron cuatro dispositivos de monitoreo, tres se encuentran respecto al nivel de calle montados en la fachada de una casa, así como las fachadas de los edificios, éstos usan de ser posible la conexión vía Wi-Fi, el último prototipo fue montado en un vehículo no motorizado que transmite los datos en tiempo real usando la conexión GSM/3G del proveedor AT&T, al ser un proveedor que cuenta con una cobertura lo suficientemente confiable para el caso de estudio.

La nube de almacenamiento nos permite almacenar cada uno de los datos enviados por ambas estaciones de monitoreo desde la primera hora y hasta el final de la fase de prueba.

El algoritmo de *machine learning* que procesa los datos obtenidos para la creación de los modelos predictivos una vez se obtuvieron los suficientes datos (al menos dos meses de recopilación de datos) para poder entrenar al sistema.

Sobre la marcha de la adquisición de datos el algoritmo fue sujeto a prueba para mejorar su eficiencia en la predicción.

Finalmente se muestran los resultados obtenidos por cada una de las estaciones de monitoreo, se interpretan los resultados del algoritmo para comprobar la eficiencia del sistema y serán estos los presentados para la evaluación del presente proyecto, así como una demostración del funcionamiento de una de las estaciones.

# Capítulo II

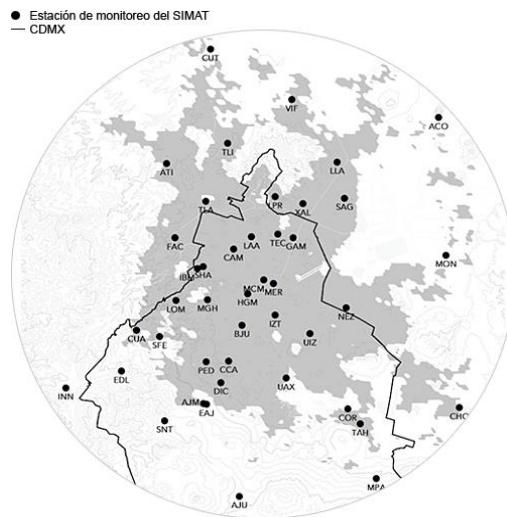
## Estado del Arte

En este capítulo se describen algunos proyectos de índole académico e investigación relacionados con el presente sistema, permitiéndonos así analizar similitudes y diferencias con respecto a la propuesta.

### 2.1 Red Automática de Monitoreo Atmosférico

El Sistema de Monitoreo Atmosférico de la Ciudad de México (SIMAT) cuenta con un subsistema denominado Red Automática de Monitoreo Atmosférico (RAMA) cuyo propósito es el empleo de equipos continuos fijos cuya función es la medición de dióxido de azufre, monóxido de carbono, dióxido de nitrógeno, ozono, PM<sub>10</sub> y PM<sub>2.5</sub> y está integrada por 34 estaciones, así como el Centro de Información de la Calidad del Aire (CICA) que es el repositorio de todos los datos generados por el Sistema de Monitoreo Atmosférico y es responsable de la validación, procesamiento y difusión de la información generada por el programa de monitoreo. [4]

En la *ilustración 3* se muestra la distribución de las estaciones de monitoreo del SIMAT incluyendo las 34 subestaciones dedicadas al monitoreo de ozono, dióxido de nitrógeno, dióxido de azufre, monóxido de carbono y partículas suspendidas.



*Ilustración 4 Distribución de las redes de monitoreo del SIMAT, Fuente: SEDEMA*

## 2.2 Breathe London (Respira Londres).

*Breathe London (Respira Londres)* que es un programa desarrollado por el grupo de investigación del medio ambiente del Colegio Imperial de Londres ha implementado un sistema con análisis de datos para comprender mejor la exposición de los londinenses a la contaminación del aire.

El sistema implementado en la ciudad de Londres, Reino Unido, es un sistema que combina la implementación de una red de 100 sensores instalados en lámparas y edificios de la ciudad que logran transmitir en tiempo real datos de monitoreo, así como vehículos de Google Street View especialmente equipados con sensores de monitoreo de calidad del aire para tener una comprensión más amplia de los niveles en las calles, así como en conjunto con el *Greater London Authority, King's College* que dotó sensores portátiles a niños y maestros para que puedan realizar mediciones durante su trayecto a la escuela.

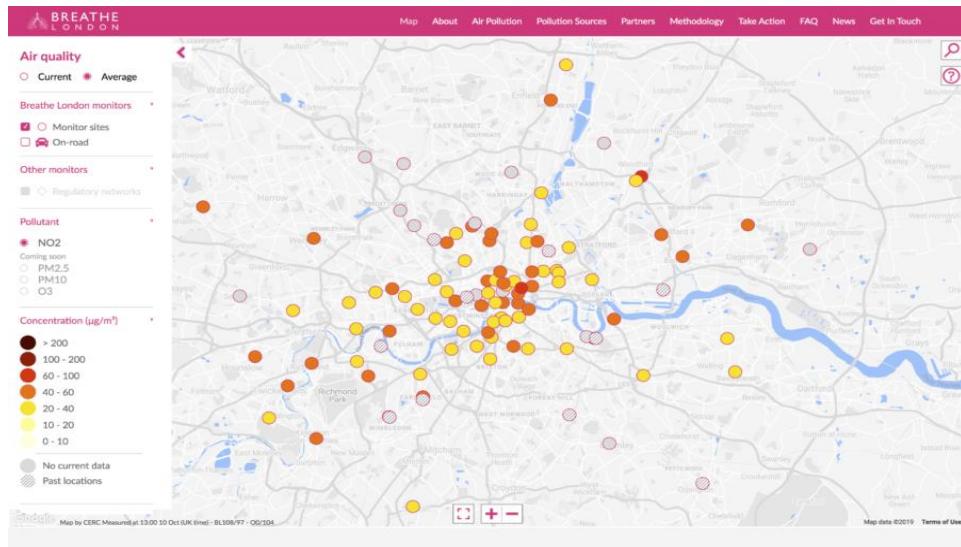
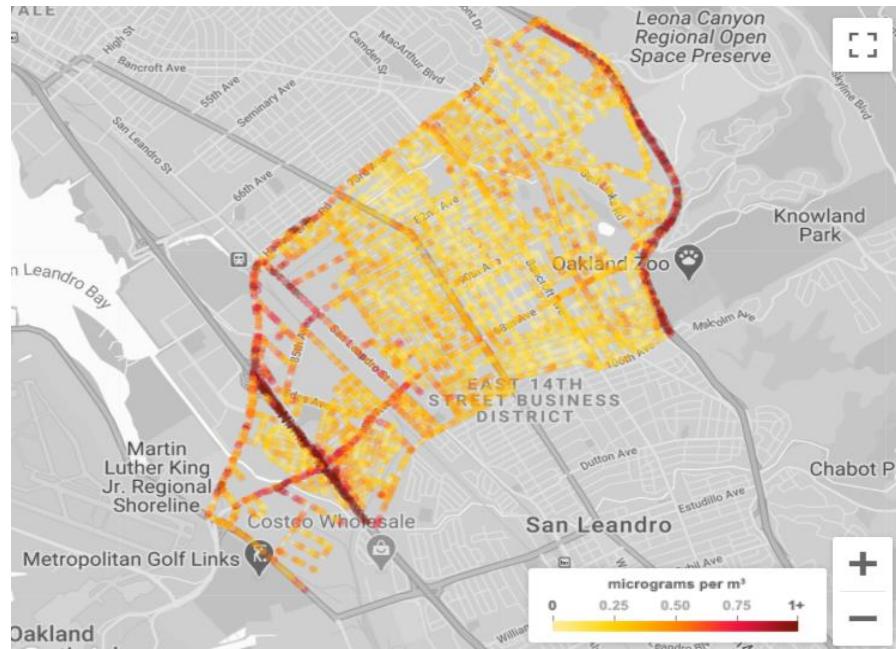


Ilustración 5 Mapa de la red de monitoreo de Breathe London en tiempo real, Fuente: Breathe London

## 2.3 Oakland Environmental Defense Fund Project

El proyecto dirigido por el *Environmental Defense Fund* en la ciudad de Oakland, California, Estados Unidos que tomó 11 meses de datos y utilizando vehículos especialmente equipados con sensores de monitoreo de calidad del aire que circulaban por distintos puntos de la ciudad en un promedio de 30 veces y que logró arrojar datos sobre cómo está distribuida la contaminación en las diferentes áreas de la ciudad.

En la siguiente ilustración 5 se puede observar el recorrido y las muestras obtenidas por el proyecto en cuanto a carbono negro, que es una partícula suspendida ( $PM_{2.5}$ ).



*Ilustración 6 Mapa de muestras de Carbono Negro en la ciudad de Oakland,  
Fuente: EDF*

## 2.4 Estudio de la contaminación ambiental de la Ciudad de México mediante los modelos Redes Neuronales Artificiales y Algoritmos genéticos.

El trabajo desarrollado por alumnos de la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional presenta el estudio de los registros del Índice Mexicano de la Calidad del Aire (IMECA) para la creación de un modelo predictivo de comportamiento de los contaminantes criterio (NO<sub>2</sub>, CO, O<sub>3</sub> y PM10) mediante la implementación de una red neuronal artificial entrenada mediante algoritmos genéticos, para obtener predicciones de hasta con 24 horas de antelación a la lecturas del IMECA, dichos datos fueron comprendidos en un periodo de Enero de 2012 a Agosto de 2016 para la Ciudad de México. [8]

## 2.5 Tabla comparativa de cada uno de los enunciados en el Estado del Arte

Nombre	Red Automática de Monitoreo	Breathe London	Oakland EDF Project	Estudio de la contaminación ambiental de la Ciudad de México mediante los modelos Redes Neuronales Artificiales y Algoritmos genéticos.	Modelos predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil
¿Es escalable fácilmente?	No	Si	Si	Si	Si
¿Tiene un bajo costo	No	Si	Si	Si	Si
¿Su implementación usa una red de vehículos?	No	Si	Si	No	Si
¿Su implementación combina mediciones móviles con estacionarias?	No	Si	No	No	Si
¿Realiza mediciones a nivel de calle?	No	Si	Si	No	Si
¿Utiliza el concepto de IoT?	No	Si	No	No	Si
¿Transmite datos en tiempo real?	No	Si	No	No	Si
¿Crea modelos predictivos con los datos?	Si	Si	No	Si	Si
¿Esta implementado	Si	No	No	Si	Si

<b>en la Ciudad de México?</b>					
------------------------------------	--	--	--	--	--

Tabla 1 Tabla comparativa de los elementos del Estado del Arte

En la Tabla Comparativa 1 se observan las diferencias entre cada uno de los sistemas en cuanto a su implementación.

# Capítulo III

## Marco teórico

Después de la revisión de la literatura que comprende el desarrollo del proyecto denominado “Modelos predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil”, se plantean a continuación los conocimientos necesarios para la comprensión de éste.

1. IoT
2. Machine Learning
3. Nube
4. Monitoreo de contaminación del aire

### 3.1 IoT (Internet de las cosas)

El Internet de las cosas se refiere al concepto de conectar cualquier dispositivo que recopile algún tipo de información y es enviada al Internet y hacia otros dispositivos conectados en la red, de tal forma que se crea una red de dispositivos y personas que intercambian información de su uso y de lo que sucede a su alrededor.

El objetivo principal de crear una red tan grande como lo propone el concepto del IoT es el de entender a niveles de detalle precisos todo lo que ocurre en una ciudad, de esta manera los algoritmos que la integren se harán cargo de mejorar los dispositivos conectados en ella, así como los servicios. [9]

#### 3.1.1 Red 4G y 5G

La red celular es el orquestador en de la tecnología del IoT, en la actualidad las redes 4G admiten transmisión de datos masivas basadas en LTE (Long-Term Evolution), si bien existen algunas opciones de menor consumo en energía (LP-

WAN) que son ideales para la implementación de IoT, su presencia en países como México aún no es suficiente, por ello este caso se concentra en el uso de la red 4G y su transición a la red 5G que gracias a sus capacidades de transmisión y baja latencia son idóneas para la transmisión de datos en tiempo real. [10]

### 3.2 Machine Learning

El término *Machine Learning* es un subcampo de IA (Inteligencia Artificial) y a su vez éste pertenece a las ciencias de la computación (aunque también se puede decir que el machine learning pertenece al estudio de la estadística), es decir que esta permite soluciones adaptativas de IA, y se puede dar una definición más concisa a este concepto de la siguiente forma:

“Sistema que mejora su desempeño en una determinada tarea mientras consume más y más información y genera experiencia.” [11]

Las raíces de todo se encuentran en la estadística, se puede ver análogamente como el arte de extraer información a partir de datos, con especial referencia a métodos como lo son la regresión lineal y las estadísticas Bayesianas. [11]

Una categorización amplia de esta área de estudio puede ser planteada de la siguiente manera:

- Aprendizaje supervisado: Se da una entrada, por ejemplo, una fotografía con una señal de tráfico, y la tarea es predecir la salida o etiqueta correcta, por ejemplo, qué señal de tráfico está en la imagen (límite de velocidad, señal de alto, etc.). En los casos más simples, las respuestas están en forma de sí / no (a estos los llamamos problemas de clasificación binaria).
- Aprendizaje no supervisado: no hay etiquetas ni resultados correctos. La tarea consiste en descubrir la estructura de los datos: por ejemplo, agrupando elementos similares para formar conglomerados o reduciendo los datos a un pequeño número de "dimensiones" importantes.
- Aprendizaje por refuerzo: se usa comúnmente en situaciones en las que un agente de inteligencia artificial, como un automóvil autónomo, debe operar en un entorno y donde la retroalimentación sobre buenas o malas decisiones está

disponible con cierta demora. También se usa en juegos donde el resultado puede decidirse solo al final del juego.

### 3.2.1 Ensemble Learning (Aprendizaje conjunto)

El aprendizaje conjunto ayuda a mejorar los resultados del machine learning al combinar diversos modelos, este enfoque permite tener un desempeño predictivo más acertado al compararlo con un modelo individual. Éstos son meta algoritmos que al combinar distintas técnicas en un modelo predictivo logran reducir la varianza, sesgos y robustece las predicciones. [11]

## 3.3 La nube (Cloud)

En pocas palabras, la nube es la entrega de servicios computacionales, incluidos servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia, a través de Internet ("la nube") para ofrecer una innovación más rápida, recursos flexibles y economías de escala. Por lo general, los servicios en la nube funcionan con la premisa de pagar por lo que se consume, ayudando a la reducción de costos y recursos.

### 3.3.1 Almacenamiento en la nube

El almacenamiento en la nube permite guardar datos y archivos en una locación remota a la que se accede a través de una red pública o privada, la transferencia de datos a la nube se vuelve responsabilidad del proveedor, quien almacena, administra, asegura y mantiene los servidores asociados a la infraestructura que asegura el acceso a la información cuando se requiere. [12]

## 3.4 Monitoreo de calidad del aire

El monitoreo de los niveles de contaminación de las áreas urbanas es crítico para mantener condiciones de vida dentro de los estándares internacionales de la Organización Mundial de la Salud en cuanto a los límites permisibles de estas partículas

en el aire, históricamente el monitoreo de los agentes contaminantes se ha hecho mediante estaciones de medición fijas que usualmente cuentan con sensores de los principales contaminantes que afectan la salud de millones de personas anualmente.

### 3.4.1 Dióxido de Nitrógeno

La principal fuente de dióxido de nitrógeno (NO<sub>2</sub>) en los entornos urbanos es la quema de combustibles fósiles. En las ciudades, esto se asocia comúnmente con los gases de escape de los automotores de combustión interna. Las áreas con redes de carreteras de alta circulación cercanas a grandes poblaciones tienen mayor riesgo de sobreexposición. Los sitios industriales también producen altas concentraciones de NO<sub>2</sub>. [13]

### 3.4.2 Partículas Suspendidas

Las partículas suspendidas (PM) en el aire se clasifican en fracciones de diferentes tamaños. El total de partículas suspendidas (TSP) incluye todos los tamaños de partículas. Las fuentes clave que son PM<sub>10</sub> y PM<sub>2.5</sub> provienen de las actividades humanas como lo son: emisiones de automotores de combustión interna, quema de combustibles como diésel y carbón, y actividades industriales. [13]

### 3.4.3 Ozono

El ozono no se emite directamente, más bien se forma típicamente a través de reacciones fotoquímicas que involucran óxidos de nitrógeno (NO<sub>x</sub>) y compuestos orgánicos volátiles (COV), ambos típicamente emitidos por vehículos y fuentes industriales. A medida que aumenta la temperatura en una tarde calurosa de verano, la eficiencia de esta reacción aumenta y se forma el ozono. [13]

Las grandes áreas urbanas a menudo tienen la combinación perfecta de emisiones y temperaturas cálidas, lo que permite que la reacción fotoquímica cree el manto marrón de "smog" que es una característica de muchas grandes ciudades

en todo el mundo. Las concentraciones de ozono son generalmente más altas en ciudades y áreas urbanas. [13]

### 3.4.4 Monóxido de carbono y Dióxido de Azufre

La principal fuente de CO son los procesos de combustión del transporte y la actividad industrial. La combustión de combustibles fósiles en las centrales eléctricas es la mayor fuente de emisión de SO<sub>2</sub> a la atmósfera. Otras fuentes incluyen la extracción de metales del mineral y la quema de combustibles con alto contenido de azufre en barcos, trenes y maquinaria. Sin embargo, en los últimos años el SO<sub>2</sub> como contaminante del aire urbano ha ido disminuyendo debido a la desulfuración del combustible. [13]

## 3.5 Implementación del Marco Teórico

La importancia de la literatura que comprende los temas necesarios para lo comprensión del presente trabajo radica en el hecho de su directa aplicación en el sistema propuesto que pretende usar Machine Learning en primera instancia para crear modelos predictivos usando una técnica llamada *Long Short Term Memory* cuyo objetivo es implementar distintos algoritmos para crear una predicción más robusta logrando reducir la varianza y bias.

El concepto del IoT nos ofrece la implementación de una red robusta de consumo de datos para mejorar servicios y calidad de vida en las ciudades, una red de IoT se comprende primordialmente de dispositivos conectados a Internet que proporcionan censado de datos, en nuestro caso de los niveles contaminantes en el aire mediante la implementación de pequeñas estaciones de monitoreo móviles a nivel de calle. Para esto es necesario conocer la teoría en cuanto a los principales contaminantes se refiere y entender el impacto de ellos, en el presente trabajo se detallará el uso de dos sensores que son de CO y PM<sub>2.5</sub>/PM<sub>10</sub> y que nos permitirán tener comprensión de los niveles de contaminación en un rango de 24 horas al día por 7 días a la semana.

El ultimo concepto por aplicar es la nube cuya función es la de servir como Hub de entrada para los datos proporcionados por nuestras estaciones de monitoreo que a su vez serán almacenados en una Base de Datos no relacional en la nube y nos permitirá tener un control granular de aquellos datos recabados por los sensores de prueba.

# Capítulo IV

## Análisis

### 4.1 Introducción

El presente apartado describe los requerimientos de herramientas de hardware para la implementación de las estaciones de monitoreo, arquitectura de la red e implementación de algoritmos de Machine Learning para la creación de modelos predictivos.

### 4.2 Identificación de requerimientos

Para la identificación de requerimientos se necesitó comprender el problema existente en la Ciudad de México respecto a los niveles contaminantes, actualmente no se realizan mediciones a niveles de calle y el desafío radica en proporcionar un sistema que nos provea de un espectro mayor con respecto a lo que sucede con el índice de calidad del aire en la ciudad.

Como resultado del análisis se obtuvo lo siguiente:

Datos de sensores

- Mediciones de PM<sub>2.5</sub>/PM<sub>10</sub>
- Lecturas de GPS

Placa Base

- Arduino MKR WiFi 1010 (Módulo estacionario)
- Arduino MKR GSM 1400 (Módulo móvil)

Comunicación de Datos

- MQTT (Protocolo de transmisión de mensajes para IoT)

- Red WiFi (Módulo estacionario) y GSM (Módulo móvil)

#### Datos de la Arquitectura

- Suscripción de Azure
- IoT Hub de Azure
- CosmosDB para almacenamiento de datos no relacional.
- Azure Stream Analytics para activar trigger en la recepción de datos y posterior almacenamiento en CosmosDB

#### Datos para el modelo predictivo

- Datos de las mediciones de los sensores de PM<sub>2.5</sub>/PM<sub>10</sub>
- Datos de las coordenadas obtenidos por el módulo GPS
- Datos del módulo de transmisión
- Datos de fecha y hora de transmisión del paquete.

### 4.3 Análisis de requerimientos

En este apartado, se explica la especificación de requisitos del sistema en cuanto al funcionamiento de cada elemento y módulo en específico.

#### 4.3.1 Ámbito del sistema

El sistema “Modelos Predictivos de contaminación del aire mediante la implementación de un sistema de monitoreo móvil” tiene como objetivo principal ofrecer una alternativa para el monitoreo de calidad del aire en la Ciudad de México que nos permita obtener un mayor espectro del comportamiento de los contaminantes a nivel de calle con dos módulos de sensores uno móvil y otro estacionario y que mediante *Long Short Term Memory* que es una técnica de Machine Learning se creen modelos predictivos para la obtener mejores estrategias en la reducción de contaminantes.

El sistema funciona de la siguiente manera:

- Obtiene mediciones de los contaminantes de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, así como de las coordenadas de la estación mediante un módulo GPS.
- Envío de paquetes de datos usando el protocolo MQTT mediante la red WiFi o GSM disponible al servicio de Azure.
- Mediante el servicio de Azure IoT Hub gestionar los datos recibidos y almacenarlos en una base de datos no relacional.
- Uso de algoritmo de *Long Short Term Memory* para crear modelos predictivos de los contaminantes recabados por el sistema.

#### 4.3.2 Definiciones, acrónimos y abreviaturas

- RF: Requerimiento Funcional
- RH: Requerimiento de Hardware
- CU: Casos de Uso
- MQTT: protocolo de transmisión de datos.
- Administrador: administrador de toda la arquitectura del sistema y la implementación de los modelos predictivos.
- Supervisor: usuario que analizara los datos obtenidos por el modelo de machine learning.
- Broker: Mantiene el registro de los clientes conectados y se encarga de recibir los mensajes.

#### 4.3.3 Perspectiva del producto

El sistema hace uso de hardware y una arquitectura en la nube de Azure que permite interactuar con los datos recabados por las estaciones de monitoreo y brinda una alternativa para la medición de los contaminantes criterio en la Ciudad de México mediante mediciones a nivel de calle y haciendo uso de

modelos predictivos mediante técnicas Machine Learning se logra obtener un sistema que proporcione un espectro amplio de los contaminantes.

El sistema es auto administrado desde el punto de vista del manejo de la información, recolección y envío de paquetes. El análisis de los modelos predictivos obtenidos de la implementación de Machine Learning, así como el funcionamiento de la arquitectura supervisados por el usuario administrador.

#### 4.3.4 Funcionamiento del producto

El sistema se encarga de almacenar los datos obtenidos por las estaciones de monitoreo usando el concepto de IoT para su después procesamiento mediante Machine Learning para obtener modelos predictivos.

La placa base de la estación de monitoreo estacionaria cuenta con tarjeta Arduino MKR WiFi 1010 que integra los sensores principales de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, así como el módulo GPS, esta tarjeta cuenta con un módulo WiFi NINA-W10 que opera a 2.4GHz y nos permite enviar los paquetes recabados por los sensores y el módulo GPS usando el protocolo MQTT directamente a la nube de Azure usando una red de WiFi.

La placa base para la estación de monitoreo móvil cuanta con una tarjeta Arduino MKR GSM 1400 que al igual que la estación estacionaria integra los sensores de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, así como el módulo GPS, esta tarjeta cuenta con un chipset SARA-U201 que opera usando la red GSM / 3G y nos permite de igual manera enviar paquetes usando el protocolo MQTT directamente a la nube de Azure.

Los eventos generados por la estación de monitoreo son enviados a la capa de Azure IoT Hub como una cadena de mensajes almacenados como particiones cada hora, los datos son eventualmente almacenados en CosmosDB como un documento JSON.

Los datos generados en la Base de Datos CosmosDB fueron utilizados para crear el modelo de machine learning que nos permitió crear modelos predictivos, una vez terminados los modelos después del periodo de prueba fueron analizados por el administrador del sistema.

#### 4.3.5 Características de los usuarios

<b>Tipo de Usuario</b>	Administrador
<b>Experiencia</b>	Conocimiento de la Arquitectura de la nube de Azure
<b>Tareas</b>	Administrar el funcionamiento de la arquitectura

Tabla 2 - Características del usuario Administrador

<b>Tipo de Usuario</b>	Supervisor
<b>Experiencia</b>	Conocimientos de Machine Learning
<b>Tareas</b>	Crear y analizar los datos obtenidos del modelo predictivo.

Tabla 3 - Características del usuario Supervisor

#### 4.3.6 Restricciones

- El sistema requiere de una conexión WiFi para la estación de monitoreo estacionaria y red GSM/3G para la estación de monitoreo móvil.
- Los datos son almacenados en la nube de Azure, por lo tanto, necesita una suscripción a Azure para visualizar la información.
- La estación de monitoreo móvil requiere supervisión constante para su calibración debido a las limitaciones técnicas de los sensores.

#### 4.3.7 Suposiciones y dependencias

El sistema requiere de una suscripción a Azure y acceso a internet para la visualización de los datos, para las estaciones de monitoreo se necesita de conexión WiFi y GSM/3G, así como una fuente de alimentación.

#### 4.3.8 Requisitos Futuros

Incrementar el número de estaciones de monitoreo y adquisición de mejores sensores para las estaciones, así como la mejora del modelo de machine learning.

### 4.4 Requisitos Específicos

#### 4.4.1 Hardware

Estación de monitoreo estacionaria compuesta de:

- Placa Arduino MKR WiFi 1010
- Sensor de PM<sub>2.5</sub>/PM<sub>10</sub>
- Módulo GPS

Estación de monitoreo móvil compuesta por:

- Placa Arduino MKR GSM 1400
- Sensor de PM<sub>2.5</sub>/PM<sub>10</sub>
- Módulo GPS

#### 4.4.2 Interfaz de comunicación

La comunicación se realiza usando el protocolo MQTT mediante una red WiFi y GSM/3G para el envío de paquetes.

#### 4.4.3 Requerimientos Funcionales

En este apartado se presentan tablas con la descripción de los requerimientos funcionales del sistema, para una mejor comprensión son divididos por la fase que integra al sistema.

##### 4.4.3.1 Requerimientos de las estaciones de monitoreo

<b>Identificador del requerimiento</b>	RF01
<b>Nombre del requerimiento</b>	Monitoreo de PM <sub>2.5</sub>
<b>Características</b>	Censar la concentración de partículas suspendidas en el ambiente.
<b>Descripción del requerimiento</b>	La estación de monitoreo debe monitorear las mediciones de partículas suspendidas proporcionadas por el sensor cada hora.
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 4 - Requerimiento Funcional 01

<b>Identificador del requerimiento</b>	RF02
<b>Nombre del requerimiento</b>	Monitoreo de PM <sub>10</sub>
<b>Características</b>	Censar la concentración de partículas suspendidas en el ambiente.
<b>Descripción del requerimiento</b>	La estación de monitoreo debe monitorear las mediciones de partículas suspendidas proporcionadas por el sensor cada hora.
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 5 - Requerimiento Funcional 02

<b>Identificador del requerimiento</b>	RF03
<b>Nombre del requerimiento</b>	Monitoreo de coordenadas
<b>Características</b>	Registro de las coordenadas de la posición de la estación.
<b>Descripción del requerimiento</b>	La estación de monitoreo debe registrar las coordenadas proporcionadas por el módulo GPS.
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 6 - Requerimiento Funcional 03

<b>Identificador del requerimiento</b>	RF04
<b>Nombre del requerimiento</b>	Envío de paquetes vía WiFi
<b>Características</b>	Envío de paquetes usando la red WiFi local en la estación de monitoreo estacionaria.
<b>Descripción del requerimiento</b>	La estación de monitoreo estacionaria debe enviar paquetes cada hora usando el protocolo MQTT mediante el chipset WiFi integrado en la placa.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF01, RF02, RF03

Tabla 7 - Requerimiento Funcional 04

<b>Identificador del requerimiento</b>	RF05
<b>Nombre del requerimiento</b>	Envío de paquetes vía GSM
<b>Características</b>	Envío de paquetes usando la red GSM/3G en la estación de monitoreo móvil.
<b>Descripción del requerimiento</b>	La estación de monitoreo móvil debe enviar paquetes cada hora usando el protocolo MQTT mediante el chipset GSM integrado en la placa.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF01, RF02, RF03

Tabla 8 - Requerimiento Funcional 05

#### 4.4.3.2 Requerimientos de la Arquitectura de Azure

<b>Identificador del requerimiento</b>	RF06
<b>Nombre del requerimiento</b>	Recepción de Paquetes IoT Hub
<b>Características</b>	El IoT Hub debe hacer la recepción de los paquetes.
<b>Descripción del requerimiento</b>	El IoT Hub debe de estar configurado para recibir los mensajes proporcionados por las estaciones de monitoreo cada hora.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF05, RF04

Tabla 9 - Requerimiento Funcional 06

<b>Identificador del requerimiento</b>	RF07
<b>Nombre del requerimiento</b>	Procesamiento en Stream Analytics
<b>Características</b>	Stream Analytics Job deberá hacer el procesamiento de mensajes.
<b>Descripción del requerimiento</b>	Stream Analytics Job procesa los mensajes en tiempo real proporcionados por el IoT Hub basado en la lógica de negocio y enviando los datos a la capa de almacenamiento.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF06

Tabla 10 - Requerimiento Funcional 07

<b>Identificador del requerimiento</b>	RF08
<b>Nombre del requerimiento</b>	Capa de Almacenamiento en CosmosDB
<b>Características</b>	CosmosDB funge como la capa de almacenamiento de datos.
<b>Descripción del requerimiento</b>	CosmosDB recibe los datos procesados por Stream Analytics Job y los almacena en documentos JSON.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF07

Tabla 11 - Requerimiento Funcional 08

#### 4.4.3.3 Requerimientos del modelo de machine learning

<b>Identificador del requerimiento</b>	RF09
<b>Nombre del requerimiento</b>	Descarga de datos de CosmosDB
<b>Características</b>	CosmosDB debe permitir la descarga de los datos almacenados.
<b>Descripción del requerimiento</b>	Se realiza la descarga de los datos para alimentar al modelo de machine learning.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF08

Tabla 12 - Requerimiento Funcional 09

<b>Identificador del requerimiento</b>	RF10
<b>Nombre del requerimiento</b>	Entrenamiento del modelo de machine learning
<b>Características</b>	Se proporcionan los datos al modelo.
<b>Descripción del requerimiento</b>	Se proporcionan los datos obtenidos y almacenados en la base de datos al modelo para empezar el entrenamiento del modelo.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF09

Tabla 13 - Requerimiento Funcional 10

<b>Identificador del requerimiento</b>	RF11
<b>Nombre del requerimiento</b>	Análisis del modelo predictivo
<b>Características</b>	Análisis detallado del modelo de machine learning y sus resultados.
<b>Descripción del requerimiento</b>	Se realiza el análisis de los datos obtenidos por el modelo para verificar su correcta implementación.
<b>Prioridad</b>	Alta
<b>Relación</b>	RF10

Tabla 14 - Requerimiento Funcional 11

#### 4.4.3.4 Requerimientos de Supervisión

<b>Identificador del requerimiento</b>	RF12
<b>Nombre del requerimiento</b>	Verificar datos almacenados Azure
<b>Características</b>	Se realiza una supervisión de las analíticas que proporciona Stream Analytics Job y los datos almacenados en CosmosDB
<b>Descripción del requerimiento</b>	La supervisión de la arquitectura nos permite identificar el progreso de la recepción de datos por parte de las estaciones de monitoreo y la identificación problemas
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 15 - Requerimiento Funcional 12

#### 4.4.4 Requerimientos No Funcionales

Las siguientes tablas representan los requerimientos No Funcionales del sistema.

<b>Identificador del requerimiento</b>	RNF01
<b>Nombre del requerimiento</b>	Disponibilidad de red
<b>Características</b>	Se debe garantizar la disponibilidad de la red para la transmisión de paquetes.
<b>Descripción del requerimiento</b>	Debido a que se implementa una arquitectura en tiempo real se necesita garantizar el envío de paquetes en tiempo real en todo momento por lo que el acceso a la red WiFi o GSM es de vital importancia.
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 16 - Requerimiento no funcional 01

<b>Identificador del requerimiento</b>	RNF02
<b>Nombre del requerimiento</b>	Accesibilidad
<b>Características</b>	Se debe garantizar la accesibilidad de los datos en todo momento
<b>Descripción del requerimiento</b>	La arquitectura de Azure creada mediante el portal de IoT Hub debe proporcionar los datos en todo momento.
<b>Prioridad</b>	Alta
<b>Relación</b>	

Tabla 17 - Requerimiento No Funcional 02

## 4.5 Diagramas de caso de uso

En esta sección se presentan los casos de uso que conforman al sistema y que permiten describir las acciones e interacciones entre los distintos actores y componentes del sistema.

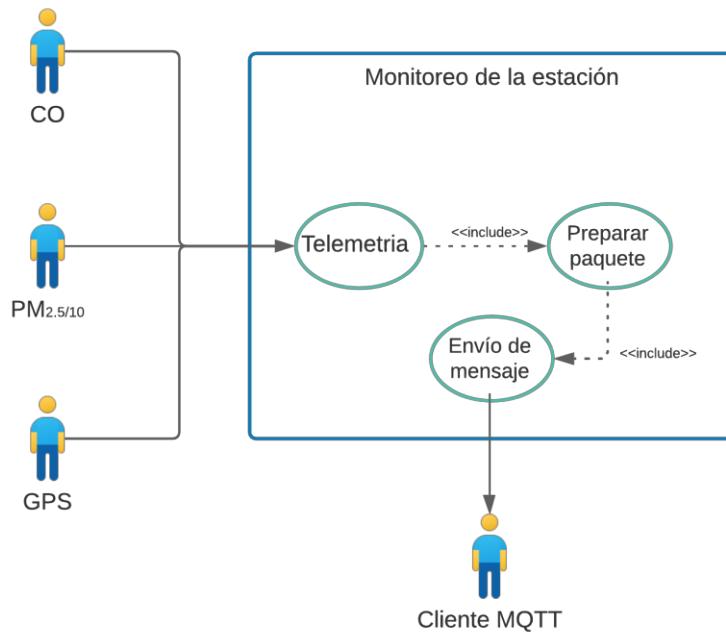


Ilustración 7 Caso de uso Monitoreo de la estación

Identificación del caso de uso	CU01
Nombre del caso de uso	Monitoreo de la Estación
Características	Los sensores obtienen datos de censado
Descripción del requerimiento	La estación de monitoreo realiza el censado de los niveles de PM <sub>2.5</sub> /PM <sub>10</sub> y los prepara para publicarlos usando el cliente MQTT
Precondición	Sensores correctamente calibrados, alimentación de corriente a la placa base y conexión WiFi / GSM

	Paso	Acción del actor	Acción Sistema
<b>Secuencia Normal</b>	1	Registrar niveles de CO y PM <sub>2.5</sub> /PM <sub>10</sub>	Recepción de telemetría
	2		Preparar paquete
	3		Envío de mensaje al cliente MQTT
	4	Cliente MQTT	Publica el mensaje al Broker
<b>Excepciones</b>	1. Error en el censado de datos 2. Error en la publicación del paquete al Broker.		
<b>Requerimiento Funcional</b>	RF01-RF05		
<b>Requerimiento No Funcional</b>	RFN01-RNF02		
<b>Prioridad de Requerimiento</b>	ALTA		

Tabla 18 - Descripción de caso de uso CU01

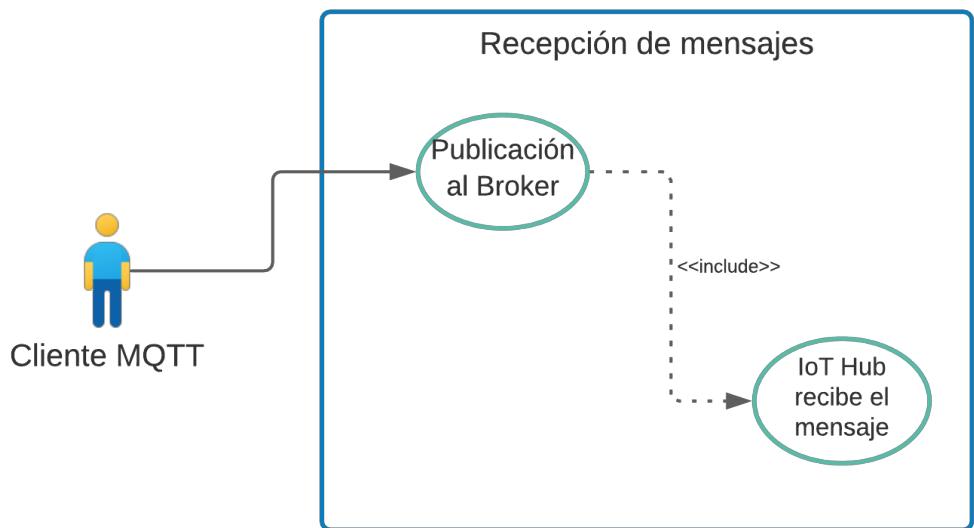


Ilustración 8 Caso de Uso Recepción de mensajes

Identificación del caso de uso	CU02		
Nombre del caso de uso	Recepción de menajes		
Características	Cliente MQTT publica el mensaje		
Descripción del requerimiento	El cliente MQTT publica el mensaje al MQTT Broker (IoT Hub) que de acuerdo con la documentación de Azure no funciona completamente como un MQTT Broker, pero realiza la función de despachar los mensajes recibidos por el cliente, IoT Hub recibe el mensaje como un stream y se prepara para la siguiente fase.		
Precondición	Conexión a la red y acceso a una cuanta de Azure		
Secuencia Normal	Paso	Acción del actor	Acción Sistema
	1	Cliente MQTT publica el	Publicación de mensaje en el bróker de IoT Hub

		mensaje al Broker	
	2		Recepción del stream de mensajes
<b>Excepciones</b>		1. Error en la publicación del mensaje 2. Error en la recepción del mensaje	
<b>Requerimiento Funcional</b>		RF06	
<b>Requerimiento No Funcional</b>		RFN01-RNF02	
<b>Prioridad de Requerimiento</b>		ALTA	

Tabla 19 Descripción de caso de uso CU02

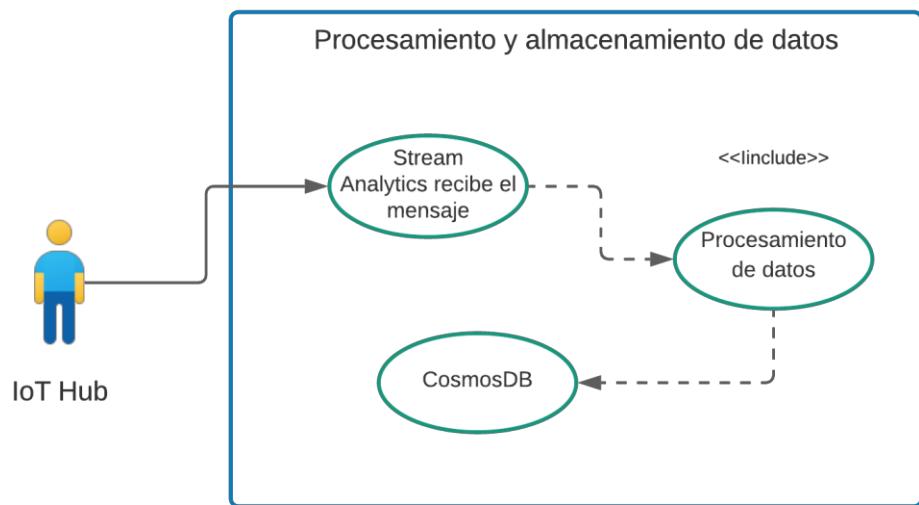


Ilustración 9 Caso de uso Procesamiento y almacenamiento de datos

<b>Identificación del caso de uso</b>	CU03		
<b>Nombre del caso de uso</b>	Procesamiento y Almacenamiento de datos		
<b>Características</b>	Procesamiento y almacenamiento en CosmosDB		
<b>Descripción del requerimiento</b>	El IoT Hub despacha los mensajes a Stream Analytics Job que proporciona analíticas de los mensajes y envía los datos a CosmosDB para que sean guardados como un documento JSON		
<b>Precondición</b>	Acceso a una cuenta de Azure		
<b>Secuencia Normal</b>	Paso	Acción del actor	Acción Sistema
	1	IoT Hub procesa el mensaje a Stream Analytics	Procesamiento del mensaje en Stream Analytics
	2		CosmosDB recibe el stream de mensajes como un documento JSON
<b>Excepciones</b>	1. Error en La arquitectura de IoT Hub		
<b>Requerimiento Funcional</b>	RF07 – RF08		
<b>Requerimiento No Funcional</b>	RNF02		
<b>Prioridad de Requerimiento</b>	ALTA		

Tabla 20 Descripción del caso de uso 03

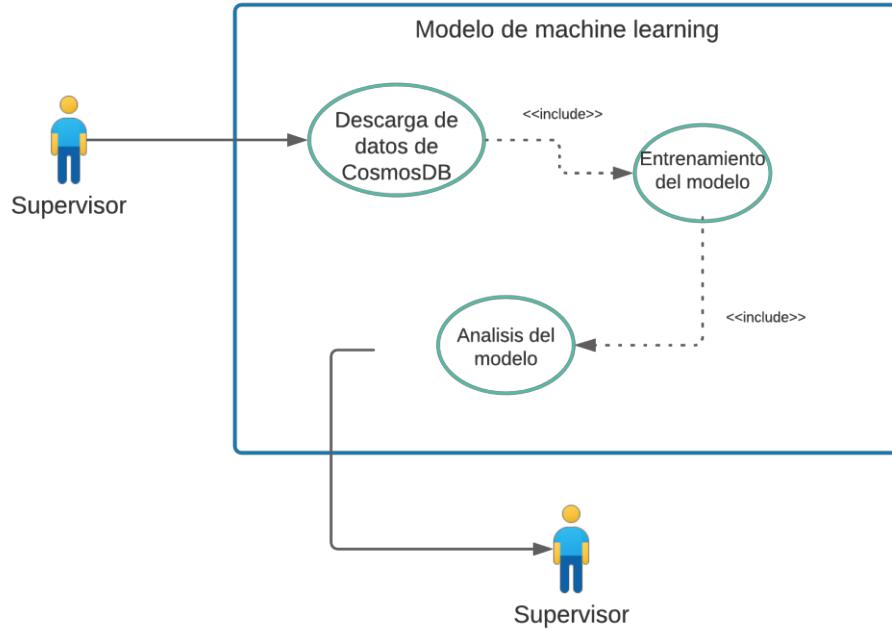


Ilustración 10 Caso de uso modelo de machine learning

Identificación del caso de uso	CU04		
Nombre del caso de uso	Modelo de machine learning		
Características	Entrenamiento y análisis del modelo de machine learning		
Descripción del requerimiento	Se entrena al modelo de machine learning usando los datos obtenidos por CosmosDB, después de la fase de entrenamiento de realizará el análisis de los datos obtenidos.		
Precondición	Acceso a una cuenta de Azure		
Secuencia Normal	Paso	Acción del actor	Acción Sistema
	1	Descargar datos de Cosmos DB	CosmosDB deberá permitir la

			descarga de los datos
2	Entrenamiento del modelo		El modelo de machine learning empezará su entrenamiento con los datos proporcionados
3	Analizar datos		El modelo presentara los datos obtenidos derivados del modelo de machine learning
<b>Excepciones</b>	1. Datos no suficientes		
<b>Requerimiento Funcional</b>	RF09 – RF011		
<b>Requerimiento No Funcional</b>			
<b>Prioridad de Requerimiento</b>	ALTA		

Tabla 21 Descripción de caso de uso 04

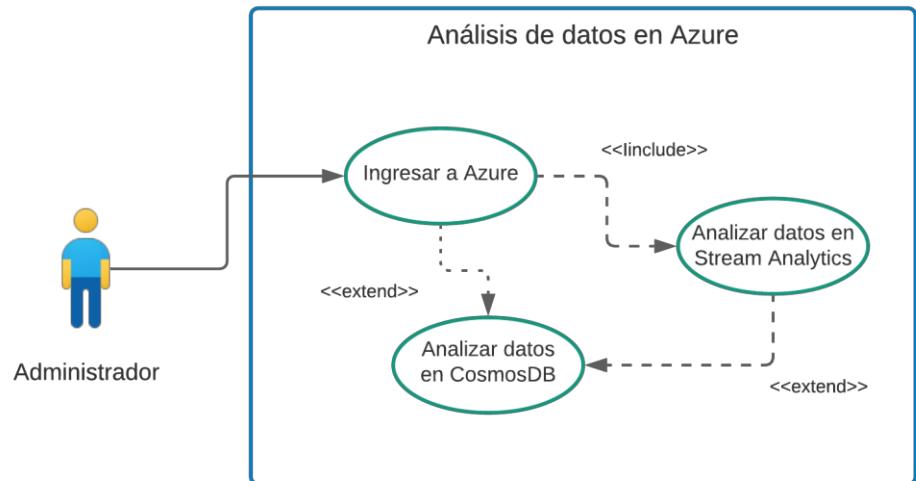


Ilustración 11 Caso de uso Análisis de datos en Azure

Identificación del caso de uso	CU05		
Nombre del caso de uso	Análisis de datos en Azure		
Características	Análisis de datos en la arquitectura de Azure		
Descripción del requerimiento	Este requerimiento pretende proporcionar análisis de los datos que están ingresando a la arquitectura de Azure, tanto en Stream Analytics como en CosmosDB.		
Precondición	Paso	Acción del actor	Acción Sistema
Secuencia Normal	1	Acceder a Azure	Ingresar al sistema de Azure
	2	Solicitar analíticas de los datos	Stream Analytics proporcionara datos del análisis de

			mensajes que llegaron
	3	Solicitar acceso a los datos de CosmosDB	CosmosDB proporcionara visualización de los documentos JSON
<b>Excepciones</b>	1. Sin acceso a Azure		
<b>Requerimiento Funcional</b>	RF012		
<b>Requerimiento No Funcional</b>	RNF02		
<b>Prioridad de Requerimiento</b>	ALTA		

Tabla 22 Descripción de caso de uso 05

#### 4.6 Análisis de Hardware a utilizar

En esta sección se presentan los sensores a usar para hacer las mediciones de contaminantes, así como el módulo GPS y la placa base necesaria para la construcción de las estaciones de monitoreo.

#### 4.6.1 Sensor de PM<sub>2.5</sub>/PM<sub>10</sub>

El sensor de partículas suspendidas es uno de los componentes criterio en el presente trabajo y a continuación se detallan las características que este debe de tener.

Componente	Módulo	Funcionamiento	Precios
<b>PM2.5</b>	PM <sub>2.5</sub> /PM <sub>10</sub>	Sensor de partículas suspendidas PM <sub>2.5</sub> /PM <sub>10</sub>	MXN ~ 600
Especificaciones		Comentarios	
Voltaje de funcionamiento: 4.95 ~ 5.05V Corriente eléctrica máxima: 120mA Diámetro pm de medición: 0.3-1.0, 1.0-2.5, 2.5-10(um) Rango de medición pm: 0 ~ 999 ug / m <sup>3</sup> Corriente de espera: ≤200 uA Tiempo de respuesta: ≤10 s Rango de temperatura de funcionamiento: -20 ~ 50C Rango de humedad de funcionamiento: 0 ~ 99% RH Tamaño máximo: 65 × 42 × 23 (mm) MTBF: >= 5 años Respuesta rápida Salida de palabras de entrada serie estándar Curva de calibración multipunto de segundo orden El tamaño mínimo de la resolución de 0,3 micras		Sensor con compatibilidad con Arduino.	

Tabla 23 Descripción de sensor de partículas suspendidas

### 4.6.3 Módulo GPS

La tarea del módulo GPS es proporcionar coordenadas de movimiento en todo momento en que la estación de monitoreo este activa para crear un mapa del análisis.

Componente	Módulo	Funcionamiento	Precios
<b>NEO-6M</b>	GPS	Módulo de GPS	MXN ~ 300
<b>Especificaciones</b>			Comentarios
Comunicación serial Voltaje de alimentación: (3.5 – 5) VDC Antena cerámica activa incluida LED indicador de señal Tamaño de antena 22x22mm Tamaño de modulo 23x30mm Batería incluida BAUDRATE: 9600 EEPROM para guardar configuración de parámetros Sistema de coordenadas: WGS-84 Sensibilidad de captura -148dBm Sensibilidad de rastreo: -161 dBm Máxima altura medible: 18000 Máxima velocidad 515 m/s Exactitud: 1micro segundo Frecuencia receptora: L1 (1575.42 MHz) Código C/A 1.023 MHz Tiempo de inicio primera vez: 38s en promedio Tiempo de inicio: 35s en promedio			Modulo GPS para Arduino.

Tabla 24 Descripción del módulo GPS

#### 4.6.4 Placa Base WiFi

A continuación se describen las características de la placa base de Arduino con funcionalidad WiFi que incorporara los sensores de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, así como el módulo GPS y que se encarga de transmitir los mensajes usando el protocolo MQTT.

Componente	Módulo	Funcionamiento	Precios
<b>MKR 1010 WiFi</b>	Arduino	Placa Base	MXN ~ 700
<b>Especificaciones</b>			Comentarios
Microcontrolador SAMD21 Cortex®-M0+ 32bit de bajo consumo ARM MCU Módulo de radio u-blox NINA-W102 Alimentación de poder (USB/VIN) 5V Batería Li-Po Single Cell, 3.7V, 1024mAh Voltaje 3.3V Digital I/O Pins 8 PWM Pins 13 (0 .. 8, 10, 12, 18 / A3, 19 / A4) UART 1 Pines de entrada analógica7 (ADC 8/10/12 bits) Pines de salida analógica1 (DAC de 10 bits) Interrupciones externas10 (0, 1, 4, 5, 6, 7, 8, 9, 16 / A1, 17 / A2) Corriente de CC para E/S Pin7 mA Memoria Flash de la CPU256 KB (interna) SRAM32 KB Velocidad de reloj32,768 kHz (RTC), 48 MHz	Placa base WiFi para estación de monitoreo fija.		

LED_BUILTIN6 Dispositivo USB de velocidad rápida y host integrado Longitud 61,5 mm Ancho 25 mm Peso 32 gr.	
--	--

Tabla 25 Descripción de placa base WiFi

#### 4.6.5 Placa Base GSM

A continuación, se describen las características de la placa base de Arduino con funcionalidad GSM que incorpora los sensores de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, así como el módulo GPS y que se encarga de transmitir los mensajes usando el protocolo MQTT.

Componente	Modulo	Funcionamiento	Precios
<b>MKR 1400 GSM</b>	Arduino	Placa Base	MXN ~ 1700
<b>Especificaciones</b>		Comentarios	
Microcontrolador SAMD21 Cortex®-M0+ MCU ARM de baja potencia de 32 bits Radio moduleu-blox SARA-U201 Secure Element ATECC508 Fuente de alimentación de la placa (USB/VIN)5V Batería compatible Li-Po de una sola celda, 3.7V, 2500mAh mínimo Voltaje de funcionamiento del circuito 3.3V Pines de E/S digitales 8 Pines PWM 13 (0.. 8, 10, 12, 18 / A3, 19 / A4) UART1 Pines de entrada analógica 7 (ADC 8/10/12 bits)		Placa base GSM/3G para estación de monitoreo móvil.	

Pines de salida analógica1 (DAC de 10 bits)	
Interrupciones externas10 (0, 1, 4, 5, 6, 7, 8, 9, 16 / A1, 17 / A2)	
Corriente de CC para E/S Pin7 mA	
Memoria Flash256 KB (interna)	
SRAM32 KB	
Velocidad de reloj32,768 kHz (RTC), 48 MHz	
LED_BUILTIN6	
Dispositivo USB de velocidad completa y host integrado	
Ganancia de antena2dB	
Frecuencia portadora GSM 850 MHz, E-GSM 1900 MHz, DCS 1800 MHz, PCS 1900 MHz	
Región de trabajo Global	
Tarjeta SIMMicroSIM (no incluida con la placa)	
Longitud67,64 mm	
Ancho25 mm	
Peso32 gr.	

Tabla 26 Descripción de placa base GSM

#### 4.7 Análisis de servicios IoT en la nube

Aquí se describe de manera general las características de los servicios de IoT en la nube que fueron considerados, se seleccionó el servicio de Azure al tener características y una relación calidad/precio más componentes extras que pueden lograr que el análisis y mantenimiento de los datos sea más efectivo.

Proveedor	Calidad del servicio	Costo aproximado	Servicios Extra	Configuración	Documentación
Azure IoT	Alta	MXN ~ 500/mes.	Azure Data Bricks CosmosDB Machine Learning	Media	Amplia
Google IoT	Alta	MXN ~ 500/mes.	DataFlow BigQuery Vertex AI	Media	Amplia
AWS IoT	Alta	MXN ~ 480/mes.	AWS IoT Analytics	Media	Amplia

Tabla 27 Descripción de servicios de IoT en la nube

## 4.8 Análisis del modelo de machine learning

Para la elección del modelo de machine learning a utilizar en el presente proyecto se realizó la revisión de distintas fuentes de información como lo son tesis y artículos científicos, de los cuales la gran mayoría en el caso específico de modelos predictivos de calidad del aire concluían en que las técnicas de redes de Memoria de Corto y Largo Plazo (Long Short Term Memory) en general dan resultados considerablemente precisos, en este documento se usará como referencia para el análisis de la técnica de machine learning a utilizar el paper titulado “Air Pollution Prediction Using Long Short-Term Memory (LSTM) and Deep Autoencoder (DAE) Models” publicado el 2 de Febrero de 2020 por Thanongsak Xayasouk, HwaMin Lee y Giyeol Lee. [14]

### 4.8.1 Recurrent Neural Network (RNN)

Una Red Neuronal Recurrente o *RNN* por sus siglas en inglés es un tipo de red neuronal que se basa en el hecho de que los humanos no iniciamos nuestros pensamientos desde cero todo el tiempo, es decir que al ir leyendo este proyecto es posible entender cada palabra escrita basándonos en la comprensión que obtuvimos de

palabras anteriores, en pocas palabras no deseamos lo que aprendemos si no que nuestros pensamientos son persistentes.

Las redes neuronales comunes no son capaces de utilizar la persistencia, este problema es resuelto usando iteraciones dentro de cada nodo que permiten que la información persista.

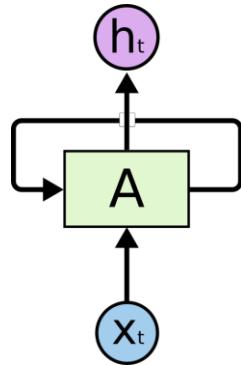


Ilustración 12 Red Neuronal Persistente con Iteración Fuente: Understanding LSTM. [30]

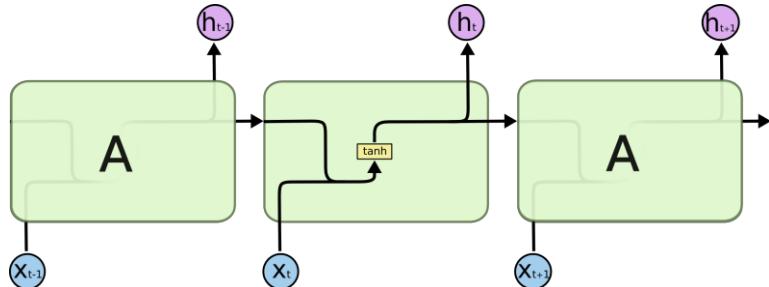
En la Ilustración 12 se logra apreciar una parte de una red neuronal, el nodo A, tiene una entrada  $x_t$  y una salida con un valor  $h_t$ . La iteración permite pasar la información de un paso al siguiente, es posible pensar sobre una red neuronal recurrente como múltiples copias de la misma red, cada una enviando un mensaje al siguiente. Sin embargo, uno de los problemas de este tipo de redes neuronales son las dependencias a largo plazo, al tener una red cada vez más grande y la brecha entre nodos es mayor la red neuronal recurrente presenta problemas al tratar de conectar la información, sin embargo y como se expone en el siguiente apartado las redes de memoria de corto y largo plazo son capaces de trabajar con el problema de la dependencia a largo plazo.

#### 4.8.2 Long Short-Term Memory (LSTM)

El método de Long Short-Term Memory usualmente denominado LSTM es un tipo de Red Neuronal Recurrente, capaz de aprender dependencias a largo plazo, éstas fueron introducidas por Hochreiter y Schmidhuber en 1997 [15], este tipo de redes

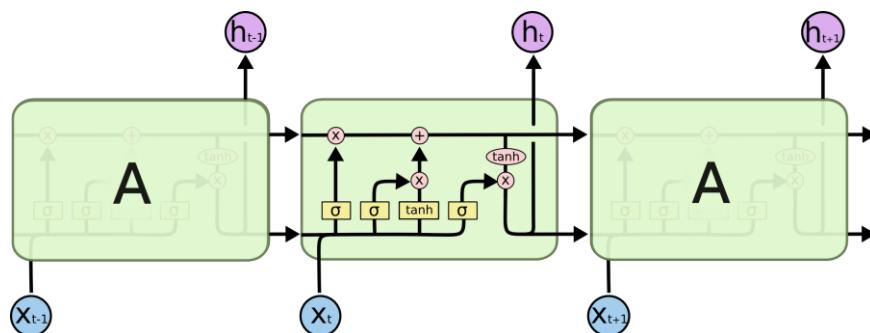
funcionan excelente en una gran variedad de problemas y son usadas en la actualidad en distintas aplicaciones.

Todas las redes neuronales recurrentes tienen forma de una cadena con módulos repetidos de una red neuronal, este módulo tiene una estructura bastante sencilla.



*Ilustración 13 Iteración del módulo en una RNN, Fuente: Understanding LSTM. [30]*

Las redes LSTM también tienen una estructura similar, pero, el módulo de iteraciones tiene una estructura diferente, tenemos cuatro capas de redes neuronales en vez de una que interactúan de una manera especial.



*Ilustración 14 Representación de una LSTM y su módulo de iteraciones. Fuente: Understanding LSTM. [30]*

En la Ilustración 14 cada conector es un vector de transferencia, desde la salida de un nodo a la entrada del siguiente, los círculos rosas representan operaciones como la de adición, mientras que las cajas amarillas son capas del aprendizaje de la red neuronal, los conectores que convergen denotan concatenación, mientras que las líneas que se separan representan la información que está siendo copiada de un nodo a otro.

En pocas palabras las redes de LSTM, tienen la habilidad de remover o agregar información a el estado de un nodo, siendo este proceso administrado por estructuras llamadas compuertas, estas nos permiten la entrada o no de información, compuesta por la operación de multiplicación, de esta manera la red regula la información que está siendo transmitida de un nodo a otro y en caso de que el contexto lo necesite retiene la información necesaria para futuros procesos.

#### 4.8.3 Stacking Ensemble

En la literatura presentada referente al Análisis del Proyecto Terminal I se propuso el uso de Stacking Ensemble para construir el modelo predictivo de machine learning, sin embargo después de una investigación más exhaustiva de información se optó por utilizar Long Short Term Memory ya que este método nos permite trabajar de una mejor manera con datos que sólo tienen una característica y múltiples muestras, caso en el que se presenta este proyecto ya que sólo se trabajará con muestras de partículas suspendidas, el modelo de Stacking Ensemble trabaja mejor con múltiples características y por ende no es la mejor opción para los propósitos de este proyecto.

# Capítulo V

## Diseño

En esta sección se describe el diseño del sistema, para el cual se emplearon diagramas, esquemas, tablas y figuras para describir la arquitectura de la solución.

### 5.1 Diseño de la arquitectura del sistema

En esta sección se describe el diseño de la arquitectura del sistema, descripción de los componentes y características de su funcionamiento.

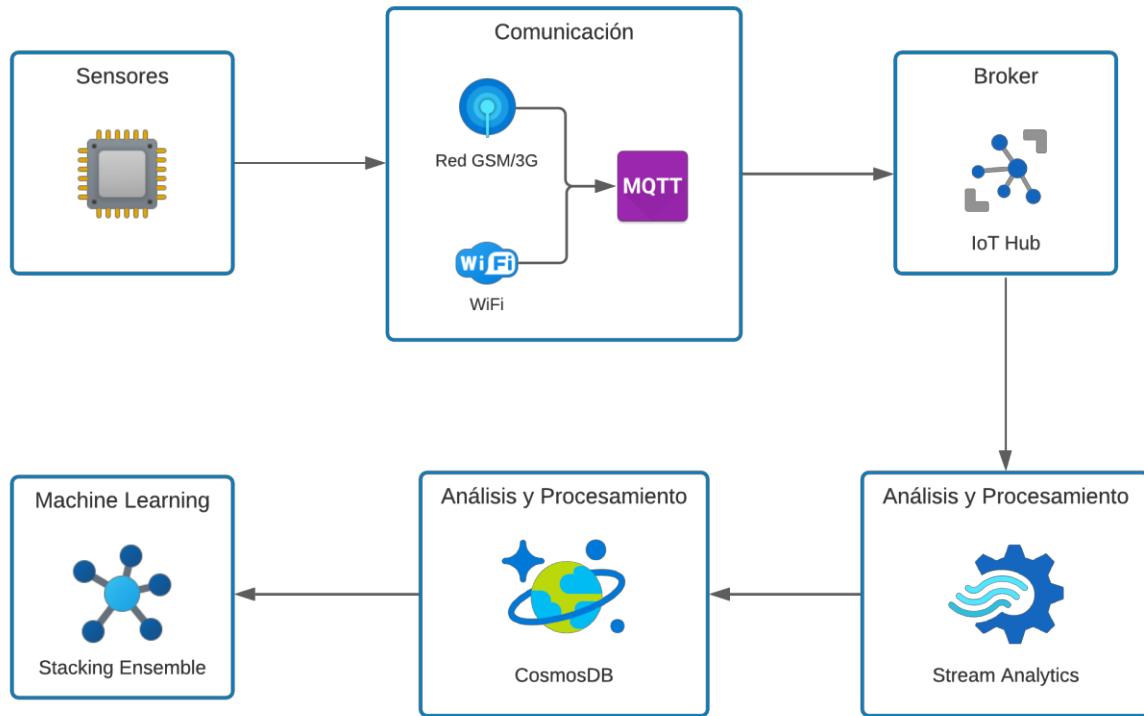


Ilustración 15 Arquitectura del sistema

La arquitectura del sistema se compone de la estación de monitoreo que incluye los sensores y la comunicación del sistema que se encarga de enviar los mensajes usando el protocolo MQTT, quien a su vez publica los mensajes al Broker (IoT Hub) y envía los datos a Stream Analytics Job para su análisis y procesamiento, para que después sean almacenados en forma de documentos JSON en la base de datos de CosmosDB, una vez en la base de datos se pueden descargar para empezar a entrenar el modelo de Long Short Term Memory y obtener resultados del modelo predictivo.

## 5.2 Estructura del documento JSON almacenado en la Base de Datos

En esta sección se describe el documento JSON que será almacenado en la base de datos de Azure CosmosDB, la siguiente imagen describe su estructura.



```
1  [
2   {
3     "timestamp": 1634567732,
4     "stationid": "Ariatne",
5     "latitude": 19.28424,
6     "longitude": -99.0089,
7     "parameters-pm25": {
8       "idparameter": "pm25",
9       "value": 23,
10      "unit": "ug/m3"
11    },
12    "parameters-pm10": {
13      "idparameter": "pm10",
14      "value": 25,
15      "unit": "ug/m3"
16    },
17    "EventProcessedUtcTime": "2021-10-18T19:35:34.0934279Z",
18    "PartitionId": 2,
19    "EventEnqueuedUtcTime": "2021-10-18T19:35:34.0100000Z",
20    "IoTHub": {
21      "MessageId": null,
22      "CorrelationId": null,
23      "ConnectionDeviceId": "ARIATNE",
24      "ConnectionDeviceGenerationId": "637701798117896687",
25      "EnqueuedTime": "2021-10-18T19:35:33.8390000Z"
26    },
27    "id": "efeedb64-cffc-4ac6-b692-fa57a6807ccd"
28  },
```

Ilustración 16 Descripción del documento JSON

### 5.2.1 Diccionario de datos del documento JSON

Documento	Nombre variable	Tipo de Dato	Descripción
general	timestamp	String	Identificador de evento con fecha y hora
	stationid	String	Identificador de la estación
	parameters	Object	Documento con información de los parámetros del evento
	latitude	Float	Coordenadas del evento
	longitude	Float	Coordenadas del evento
parameters	idparameter	String	Identificador del parámetro
	value	Int	Valor de la lectura del sensor
	unit	Int	Unidad de la lectura del sensor
autogenerated*	EventProcessedUtcTime*	String	Tiempo de procesamiento de la cadena
	PartitionId*	Int	Id Partición del Hub
	EventEnqueuedUtcTime*	String	Tiempo de procesamiento en cola de la cadena
	MessageId*	null	Id del mensaje
	CorrelationId*	null	Id de relación IoT Hub
	ConnectionDeviceId*	String	Id del dispositivo
	ConnectionDeviceGenerationId*	String	Id de conexión
	EnqueuedTime*	String	Tiempo de procesamiento en cola
<b>*datos generados automáticamente por el IoT Hub</b>			

	<b>Id*</b>	<b>String</b>	<b>Id del mensaje en el documento</b>
--	------------	---------------	---------------------------------------

Tabla 28 Diccionario de datos con descripción del documento JSON

### 5.3 Diagramas de secuencia

Un diagrama de secuencia es un tipo de diagrama que describe cómo y en qué orden interactúan un grupo de objetos que funcionan en conjunto a través del tiempo, dentro de este diagrama se indican los módulos o clases que forman parte del programa, así como las llamadas que hace cada uno de ellos para realizar una tarea.

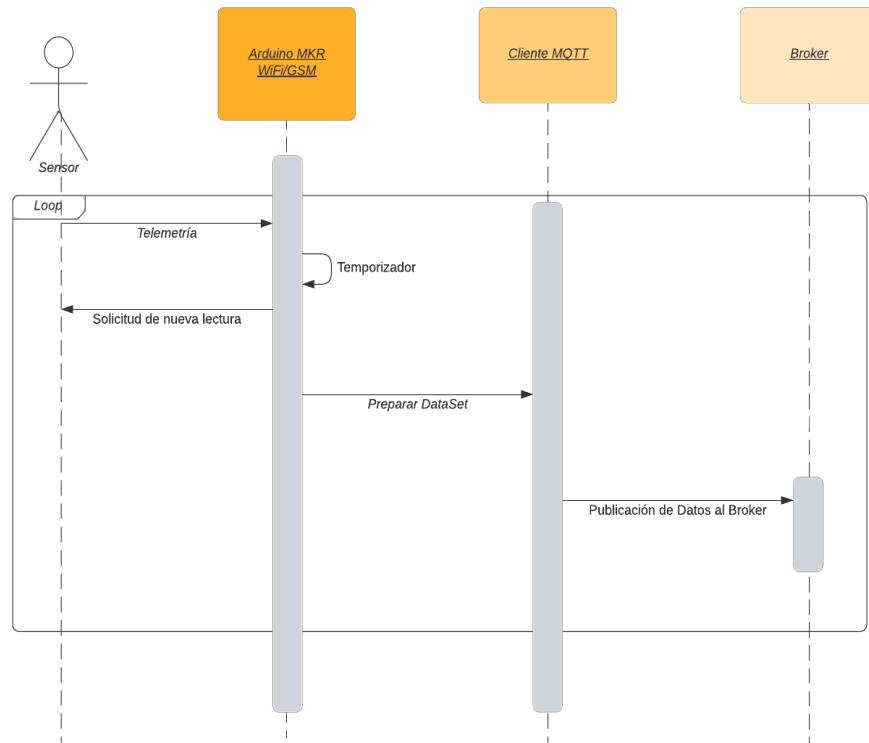


Ilustración 17 Diagrama de secuencia de la estación de monitoreo

El diagrama de secuencia de la Ilustración 13 nos indica el proceso que hará la estación de monitoreo para que en cada evento se obtengan los datos del sensor y estos sean preparados para ser publicados por cliente MQTT al Broker, la secuencia de la placa de Arduino solicitará una nueva lectura al sensor en los intervalos ya definidos.

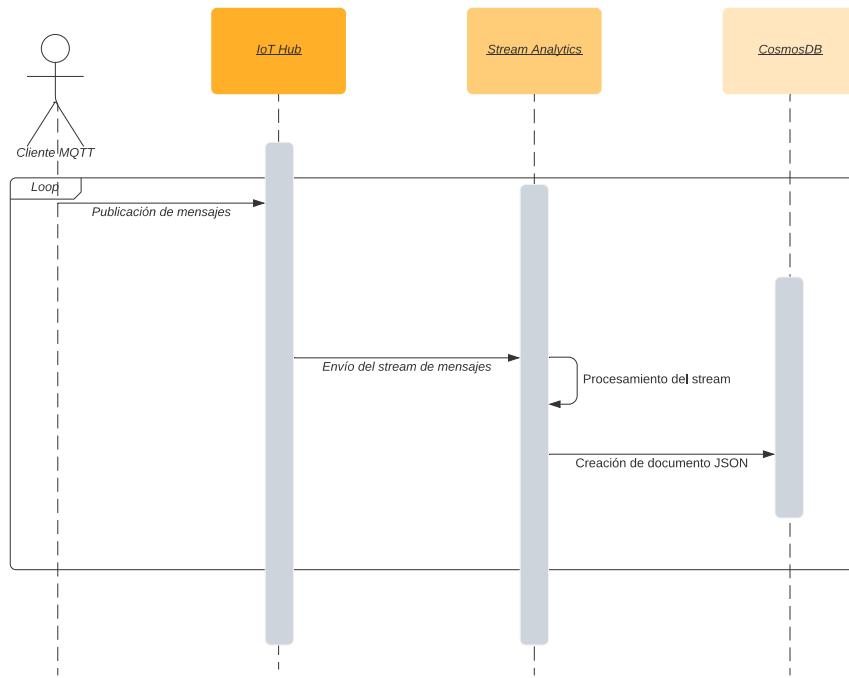
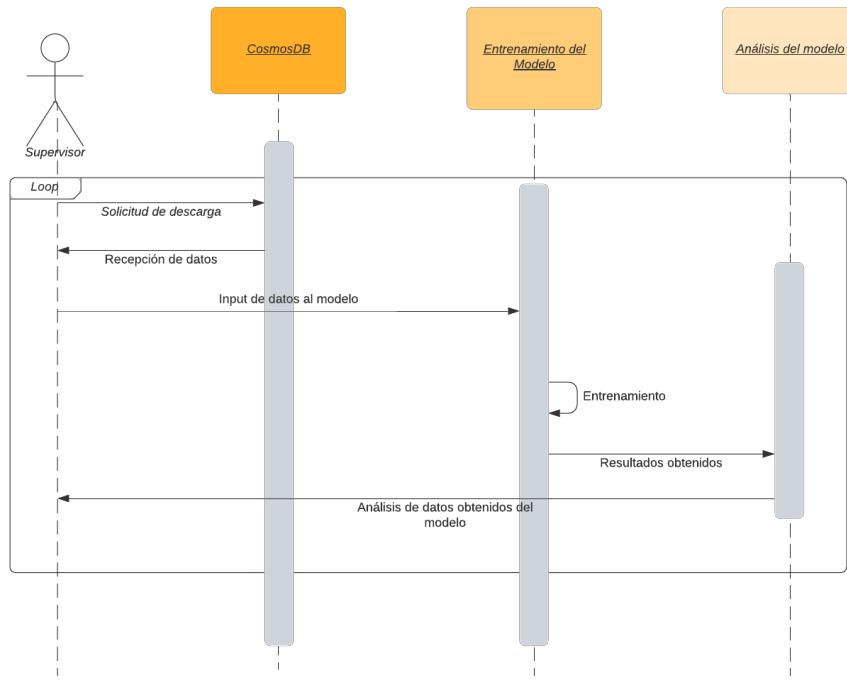


Ilustración 18 Diagrama de secuencia de la arquitectura de Azure

El diagrama de secuencia de la ilustración 14 nos muestra el proceso que tendrá que realizar la arquitectura de Azure para recibir la publicación del cliente MQTT al Broker en el IoT Hub y preparar el stream de mensajes para enviarlos a su procesamiento en Data Bricks, quien se encargará de enviarlos finalmente a CosmosDB para que sean guardados como un documento JSON.

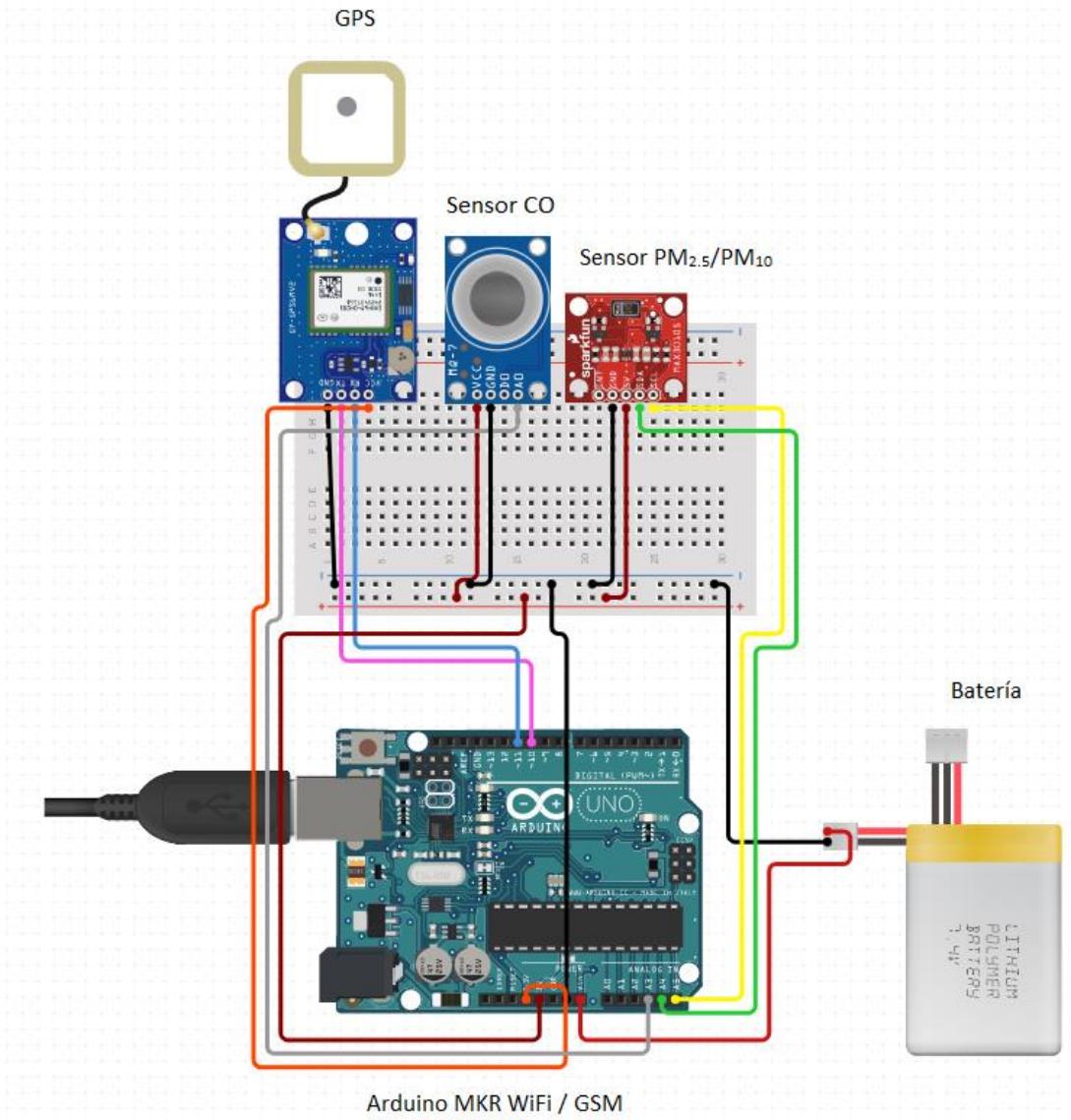


*Ilustración 19 Diagrama de secuencia del entrenamiento del modelo*

El diagrama de secuencia de la ilustración 15 indica el proceso que tendrá que seguir el supervisor para la descarga de los datos de la base de datos de CosmosDB, y la entrada de datos al modelo para su entrenamiento y posterior análisis de datos obtenidos.

#### 5.4 Diseño de las estaciones de monitoreo

Esta sección describe el diseño de las estaciones de monitoreo, la cual incluye la incorporación de sensores de PM<sub>2.5</sub>/PM<sub>10</sub> y un módulo GPS que se incorpora con dos placas base distintas, una para la estación de monitoreo móvil que incorporara la tarjeta Arduino MKR 1400 GSM que utiliza el protocolo MQTT para el envío de datos y la otra para la estación de monitoreo estacionaria que incorpora una tarjeta Arduino MKR 1010 WiFi que de igual manera utiliza el protocolo MQTT.



*Ilustración 20 Diseño de estación de monitoreo WiFi y GSM*

# Capítulo VI

## Implementación

En el presente capítulo se describe la implementación de las estaciones de monitoreo móviles y estacionarias, así como la arquitectura en la nube y el algoritmo de machine learning usado para la creación de modelos predictivos. Se incluyen los fragmentos de código y configuraciones de la arquitectura en la nube más relevantes, así como código de las estaciones de monitoreo y el algoritmo de machine learning.

### 6.1 Estaciones de Monitoreo (código)

En esta sección se describen los fragmentos de código más importantes que fueron usados para la construcción de las estaciones de monitoreo y que incluye el procesamiento y manejo de datos transmitidos por los sensores que incorporan las estaciones, así como el envío de datos usando el protocolo de MQTT.

El desarrollo del código para las estaciones de monitoreo se realizó utilizando Arduino debido a que ambas tarjetas de las estaciones de monitoreo son un Arduino MKR WiFi 1010 y Arduino MKR GSM 1400 respectivamente, se utilizaron distintas librerías que cumplen propósitos distintos, dentro de los que destacan la manipulación del módulo GPS, sensor de Partículas Suspendidas, creación de documentos JSON, manipulación de modulo GPS y protocolo MQTT.

Debido a que la implementación de las estaciones de monitoreo WiFi y GSM es distinta y usa librerías distintas se describen por separado en los apartados 6.1.1 y 6.1.2.

### 6.1.1 Estación de monitoreo estacionaria (WiFi)

Para la implementación de la estación de monitoreo WiFi se hace uso de la librería WiFiNINA [16] que nos ayuda a trabajar con el módulo WiFi de la tarjeta MKR WiFi 1010 y es capaz de soportar WEP y WPA2.

Inicialización de sensores y configuración de conexiones.

El primer paso en la configuración de la estación es definir las claves de acceso a la red WiFi, así como las conexiones a nuestro bróker.

```
1. #define SECRET_SSID      "SSID"
2. #define SECRET_PASS      "Password"
3. #define SECRET_BROKER    "Ariatne.azure-devices.net"
4. #define SECRET_DEVICE_ID "MKR-02"
```

Código 6.1

Una vez definidas las conexiones principales el siguiente paso consiste en inicializar la tarjeta de Arduino con los sensores y la configuración de los certificados de autenticación para poder conectarnos a nuestro broker.

En este caso solo estamos inicializando el sensor de partículas suspendidas mediante el Serial1 de la tarjeta de Arduino y obteniendo el certificado de autenticación a través de la librería ECCX08 [17] para poder conectarse al broker mediante el protocolo MQTT.

```
1. void setup() {
2.
3.   //LED SEND TO BROKER
4.   pinMode(7, OUTPUT);
5.
6.   //inicializa sensores
7.   Serial1.begin(9600);
8.
9.   if (!ECCX08.begin()) {
10.     //Verifica precencia de ECCX0
11.     while (1);
12.   }
13.
14.  // obtener certificado self signed
15.  ECCX08SelfSignedCert.beginReconstruction(0, 8);
16.  ECCX08SelfSignedCert.setCommonName(ECCX08.serialNumber());
17.  ECCX08SelfSignedCert.endReconstruction();
18.
19.  // validacion del certificado en el servidor
```

```

20. ArduinoBearSSL.onGetTime(getTime);
21.
22. sslClient.setEccSlot(0, ECCX08SelfSignedCert.bytes(), ECCX08SelfSignedCert.length());
23.
24. //cliente id para MQTT como device id
25. mqttClient.setId(deviceId);
26.
27. // Definicion de username "<broker>/<device id>/api-version=2018-06-30" y password vacio
28. String username;
29.
30. username += broker;
31. username += "/";
32. username += deviceId;
33. username += "/api-version=2018-06-30";
34.
35. mqttClient.setUserNamePassword(username, "");
36.
37. mqttClient.onMessage(onMessageReceived);
38. }
39.

```

Código 6.2

Lectura de muestras y conexión al broker mediante el protocolo MQTT.

En esta sección del código se realiza la verificación de la conexión a WiFi, y el cliente MQTT, y mediante la librería PMS [18] se obtiene lectura de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>, se crea un offset para publicar el mensaje cada 60 minutos.

```

1. void loop() {
2.   if (WiFi.status() != WL_CONNECTED) {
3.     connectWiFi();
4.   }
5.
6.   //Serial1.listen();
7.   dataTimer3 = millis();
8.   while (millis() - dataTimer3 <= 1000) {
9.     pms.readUntil(data);
10.    pm25 = data.PM_AE_UG_2_5;
11.    pm10 = data.PM_AE_UG_10_0;
12.  }
13.
14.  if (!mqttClient.connected()) {
15.    // conectar cliente MQTT si esta desconectado
16.    connectMQTT();
17.  }
18.
19.  // poll para mensajes MQTT y heartbeats
20.  mqttClient.poll();
21.
22.  // publicar mensaje por primera vez
23.  if (flagstart == 0) {
24.    flagstart = 1;
25.  }

```

```

26.     publishMessage();
27. }
28.
29. //3600000
30. // offset: 3599998
31. // publicar mensaje
32. if (millis() - lastMillis > 3599999) {
33.     lastMillis = millis();
34.
35.     publishMessage();
36. }
37. }
38.
39. unsigned long getTime() {
40.     // time en el modulo wifi
41.     return WiFi.getTime();
42. }
```

Código 6.3

#### Método *connectWiFi()*

El método *connectWiFi()* nos permite utilizar las credenciales ssid y pass para conectarse a la red, verificar la conexión y en su defecto reintentar la conexión.

```

1. void connectWiFi() {
2.
3.     while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
4.         // reintetar si hubo fallo
5.         delay(5000);
6.     }
7. }
```

Código 6.4

#### Método *connectMQTT()*

El método *connectMQTT()* realiza la conexión al broker del IoT Hub mediante el protocolo MQTT, este paso es importante para poder publicar la cadena de mensajes desde nuestra estación de monitoreo.

```

1. void connectMQTT() {
2.
3.     while (!mqttClient.connect(broker, 8883)) {
4.         delay(5000);
5.     }
6.     // subscripcion a un topic MQTT
7.     mqttClient.subscribe("devices/" + deviceId + "/messages/devicebound/#");
8. }
```

Código 6.5

Publicación del mensaje al Broker de Azure IoT Hub.

Finalmente se publica el mensaje con los datos obtenidos por la estación, se hace uso de la librería Arduino JSON [19] para crear el documento con la estructura JSON deseada, se realiza la serialización y se publica el mensaje al cliente MQTT del Broker.

```
1. void publishMessage() {
2.
3.     digitalWrite(7, HIGH);
4.
5.     timestamp = getTime();
6.     timestamp = timestamp - 18000;
7.
8.     StaticJsonDocument<256> doc;
9.     doc["timestamp"] = timestamp;
10.    doc["stationid"] = stationid;
11.    doc["latitude"] = latitude;
12.    doc["longitude"] = longitude;
13.
14.    JsonObject obj = doc.createNestedObject("parameters-pm25");
15.    obj["idparameter"] = idparameter25;
16.    obj["value"] = pm25;
17.    obj["unit"] = unit;
18.
19.    JsonObject obj1 = doc.createNestedObject("parameters-pm10");
20.    obj1["idparameter"] = idparameter10;
21.    obj1["value"] = pm10;
22.    obj1["unit"] = unit;
23.
24.    //payload = serializeJson(doc, Serial);
25.
26.    int b = serializeJson(doc, out);
27.
28.    mqttClient.beginMessage("devices/" + deviceId + "/messages/events/");
29.    mqttClient.print(out);
30.    //mqttClient.publish("test", out);
31.    //mqttClient.print("Hello ");
32.    //mqttClient.print(millis());
33.    mqttClient.endMessage();
34.
35.    digitalWrite(7, LOW);
36.
37. }
```

Código 6.6

### 6.1.2 Estación de monitoreo móvil (GSM)

La estación de monitoreo móvil hace uso de la librería MKRGSM [20] que está especialmente diseñada para el Arduino MKR GSM 1400, con capacidad de conexión a internet sobre la red GPRS usando un módulo que opera con 3G/4G.

Para la conexión a la red se usó la red de Hologram [21] especialmente diseñada para trabajar con dispositivos de IoT y que usa la red de AT&T en México, debido a esto debemos definir las credenciales de acceso a la red celular y al broker de la siguiente manera.

```
1. // GSM settings
2. #define SECRET_PINNUMBER      ""
3. #define SECRET_GPRS_APN        "hologram"
4. #define SECRET_GPRS_LOGIN      ""
5. #define SECRET_GPRS_PASSWORD   ""
6. #define SECRET_BROKER          "Ariatne.azure-devices.net"
7. #define SECRET_DEVICE_ID        "ARIATNE"
8.
```

Código 6.7

Configuración de sensores y conexiones.

De manera similar que, en la configuración de la estación WiFi, se inicializan sensores tanto de partículas suspendidas como módulo GPS usando Serial1 y SERCOM [22] que es una forma de agregar más interfaces seriales a la tarjeta de Arduino, de manera similar se verifica el certificado de autenticación que nos servirá para la conexión al Broker del IoT Hub.

```
1. void setup() {
2.
3.     //LED SEND TO BROKER
4.     pinMode(7, OUTPUT);
5.
6.     //inicializa sensores
7.     Serial1.begin(9600);
8.     mySerial.begin(GPSBaud);
9.
10.    //ASIGNACION DE PERIFERICOS SERCOM EN 0 y 1
11.    pinPeripheral(1, PIO_SERCOM); //Assign RX function to pin 1
12.    pinPeripheral(0, PIO_SERCOM); //Assign TX function to pin 0
13.
```

```

14. if (!ECCX08.begin()) {
15.     while (1);
16. }
17. ECCX08SelfSignedCert.beginReconstruction(0, 8);
18. ECCX08SelfSignedCert.setCommonName(ECCX08.serialNumber());
19. ECCX08SelfSignedCert.endReconstruction();
20.
21. ArduinoBearSSL.onGetTime(getTime);
22.
23. sslClient.setEccSlot(0, ECCX08SelfSignedCert.bytes(), ECCX08SelfSignedCert.length());
24. mqttClient.setId(deviceId);
25.
26. String username;
27. username += broker;
28. username += "/";
29. username += deviceId;
30. username += "/api-version=2018-06-30";
31.
32. mqttClient.setUsernamePassword(username, "");
33. mqttClient.onMessage(onMessageReceived);
34. }
35.

```

Código 6.8

Lectura de muestras y conexión al broker mediante el protocolo MQTT.

En esta sección se verifica el acceso a la red GPRS y una vez hecho esto se procede a tomar las lecturas del sensor de partículas suspendidas PMS [18] y se hace uso de la librería TyniGPSplus [23] para usar el módulo GPS y recibir lecturas de latitud y longitud, finalmente se publica el mensaje cada 5 segundos.

```

1. void loop() {
2.     if (gsmAccess.status() != GSM_READY || gprs.status() != GPRS_READY) {
3.         connectGSM();
4.     }
5.     if (!mqttClient.connected()) {
6.         connectMQTT();
7.     }
8.
9.     //Serial1.listen();
10.    dataTimer3 = millis();
11.    while (millis() - dataTimer3 <= 1000) {
12.        pms.readUntil(data);
13.        pm25 = data.PM_AE_UG_2_5;
14.        pm10 = data.PM_AE_UG_10_0;
15.    }
16.
17.    dataTimer2 = millis();
18.    while (millis() - dataTimer2 <= 1000) {
19.        while (mySerial.available() > 0)
20.        {
21.            if (gps.encode(mySerial.read())){
22.                if (gps.location.isValid())
23.                {
24.                    latitude = gps.location.lat();

```

```

25.         longitude = gps.location.lng();
26.     }
27. }
28. }
29. }
30.
31. mqttClient.poll();
32.
33. if (millis() - lastMillis > 5000) {
34.     lastMillis = millis();
35.
36.     publishMessage();
37. }
38. }
39.

```

Código 6.9

### Método *connectGSM()*

El método *connectGSM()* nos permite utilizar las credenciales apn, login y password para conectarse a la red, verificar la conexión y en su defecto reintentar la conexión.

```

1. void connectGSM() {
2.     //Serial.println("Attempting to connect to the cellular network");
3.
4.     while ((gsmAccess.begin(pinnumber) != GSM_READY) ||
5.             (gprs.attachGPRS(gprs_apn, gprs_login, gprs_password) != GPRS_READY)) {
6.         // failed, retry
7.         delay(1000);
8.     }
9. }
10.

```

Código 6.10

### Método *connectMQTT()*

El método *connectMQTT()* realiza la conexión al broker del IoT Hub mediante el protocolo MQTT, este paso es importante para poder publicar la cadena de mensajes desde nuestra estación de monitoreo.

```

1. void connectMQTT() {
2.
3.     while (!mqttClient.connect(broker, 8883)) {
4.         delay(5000);
5.     }
6.     // subscripcion a un topic MQTT
7.     mqttClient.subscribe("devices/" + deviceId + "/messages/devicebound/#");
8. }
9.

```

*Código 6.11*

Publicación de mensaje al Broker de IoT Hub.

De manera similar a la estación de monitoreo WiFi se publica el mensaje serializando los datos en un formato JSON y publicando a el broker del IoT Hub mediante el protocolo MQTT.

```
1. void publishMessage() {  
2.  
3.     digitalWrite(7, HIGH);  
4.  
5.     timestamp = getTime();  
6.     timestamp = timestamp - 18000;  
7.  
8.     StaticJsonDocument<256> doc;  
9.     doc["timestamp"] = timestamp;  
10.    doc["stationid"] = stationid;  
11.    doc["latitude"] = latitude;  
12.    doc["longitude"] = longitude;  
13.  
14.    JsonObject obj = doc.createNestedObject("parameters-pm25");  
15.    obj["idparameter"] = idparameter25;  
16.    obj["value"] = pm25;  
17.    obj["unit"] = unit;  
18.  
19.    JsonObject obj1 = doc.createNestedObject("parameters-pm10");  
20.    obj1["idparameter"] = idparameter10;  
21.    obj1["value"] = pm10;  
22.    obj1["unit"] = unit;  
23.  
24.  
25.    int b = serializeJson(doc, out);  
26.    mqttClient.beginMessage("devices/" + deviceId + "/messages/events/");  
27.    mqttClient.print(out);  
28.    mqttClient.endMessage();  
29.  
30.    digitalWrite(7, LOW);  
31. }  
32.
```

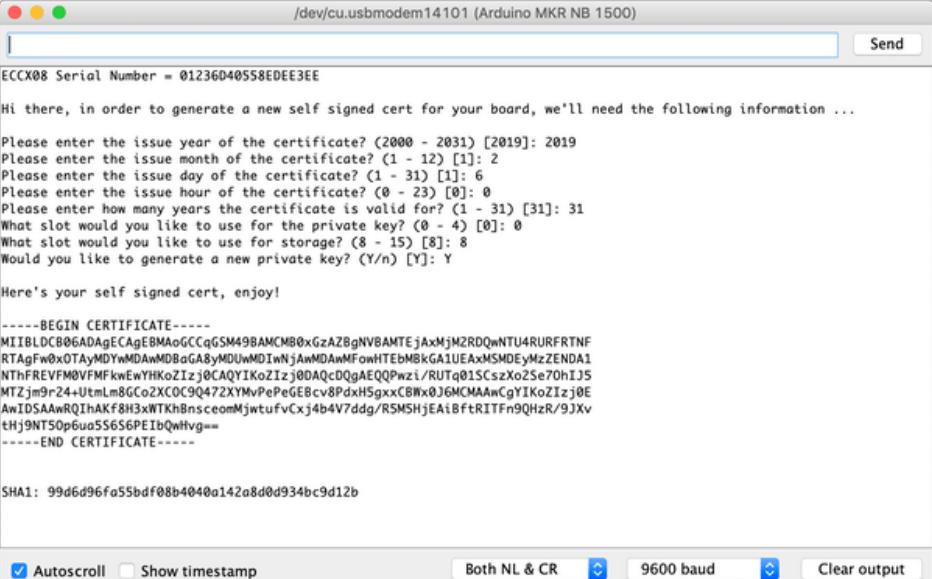
*Código 6.12*

## 6.2 Configuración de Azure IoT Hub

La Arquitectura propuesta para la implementación del procesamiento de los datos en la nube como fue presentada en el Capítulo V consta de un broker, Stream Analytics y una base de datos no relacional usando CosmosDB, en esta sección se detallan las configuraciones realizadas para lograr esta arquitectura.

### 6.2.1 Configuración de Azure IoT Hub (Broker)

Para la configuración del Azure IoT Hub es necesario en primera instancia obtener el certificado de autenticación de la tarjeta de Arduino, esto se hace mediante la librería ArduinoECCX08 [17] que nos permite obtener el Hash que usaremos para conectar la tarjeta al Broker.



The terminal window shows the following output:

```
ECCX08 Serial Number = 01236040558EDEE3EE
Hi there, in order to generate a new self signed cert for your board, we'll need the following information ...
Please enter the issue year of the certificate? (2000 - 2031) [2019]: 2019
Please enter the issue month of the certificate? (1 - 12) [1]: 2
Please enter the issue day of the certificate? (1 - 31) [1]: 6
Please enter the issue hour of the certificate? (0 - 23) [0]: 0
Please enter how many years the certificate is valid for? (1 - 31) [31]: 31
What slot would you like to use for the private key? (0 - 4) [0]: 0
What slot would you like to use for storage? (8 - 15) [8]: 8
Would you like to generate a new private key? (Y/n) [Y]: Y
Here's your self signed cert, enjoy!
-----BEGIN CERTIFICATE-----
MIIBLDCB0GADAgEAgEBMAoGCCqGSM49BAMCMB0xGzAZBgNVBAMTEjAxMjM2RDQmNTU4RURFRTNF
RTAgFw0xOTAYMDtWMDAwMDBaG8yMDUwMDIwNjAwMDAwMFonHTEbMBkGA1UEAxMSMDEyMzZENDA1
NThfREVFMD0VFMFkwEYHKKoZIZj0CAQYIKoIzj0DQcDQgAEQPwzi/RUTq01Scszk0Zse70hIJS
MTZjm9r24-UtmLm8GCo2XCOC90472XYMvPePeGE8cv8PdxH5gxxC8Wx0J6CMMAwCgYIKoIzj0E
AwIDSAwRQ1hAKF8H3xwTKh8nsceomMjwtufvCxj4b4V7ddg/R5MSHjEAiBftRITFn9QHz/9JXv
tHj9NT50p6uo5S6S6PEIbQwifvg==
-----END CERTIFICATE-----
SHA1: 99d6d96fa55bd08b4040a142a8d0d934bc9d12b
```

At the bottom of the terminal window, there are checkboxes for "Autoscroll" and "Show timestamp", and dropdown menus for "Both NL & CR" and "9600 baud". There is also a "Clear output" button.

Ilustración 21 Certificado de la tarjeta de Arduino

Lo siguiente en la lista es crear un nuevo recurso de IoT Hub dentro de Azure, creamos el grupo de recurso y establecemos un nombre para crear el despliegue.

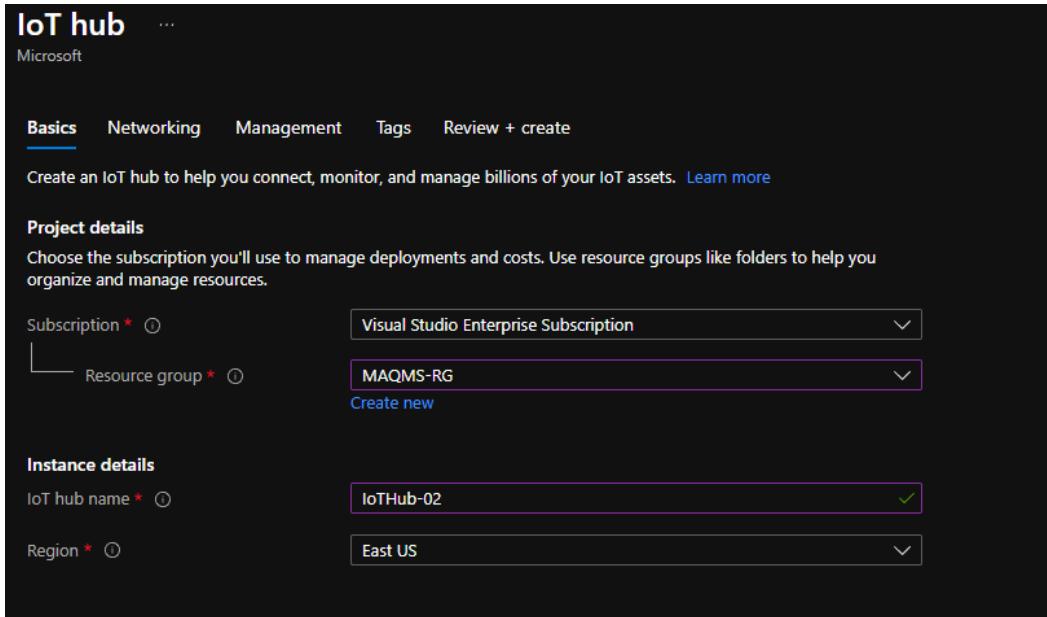


Ilustración 22 Creación de recurso de IoT Hub

Una vez hecho esto podemos agregar el hash al crear un nuevo dispositivo dentro del Azure IoT Hub.

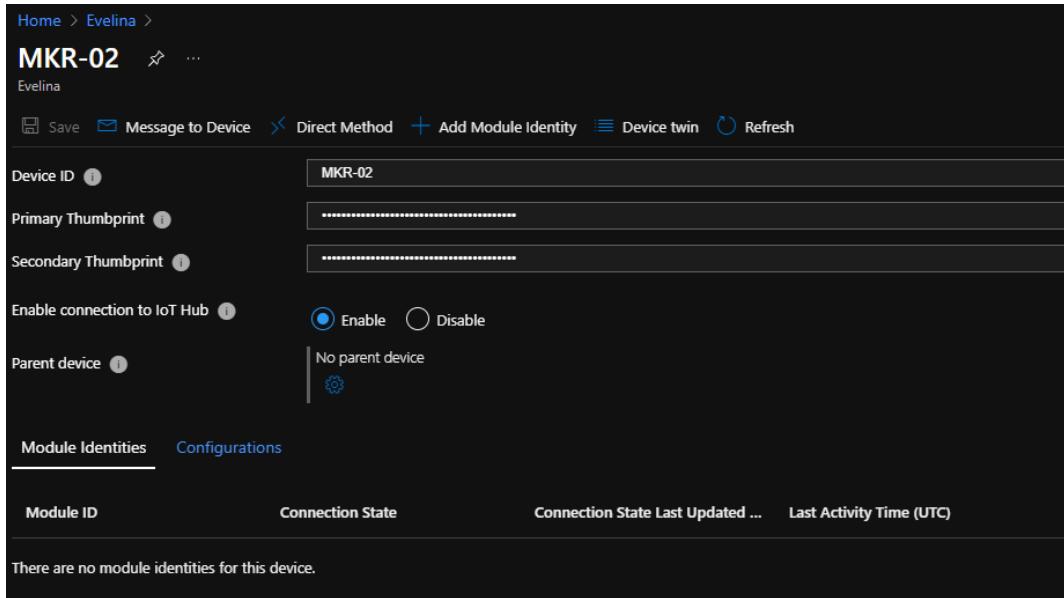
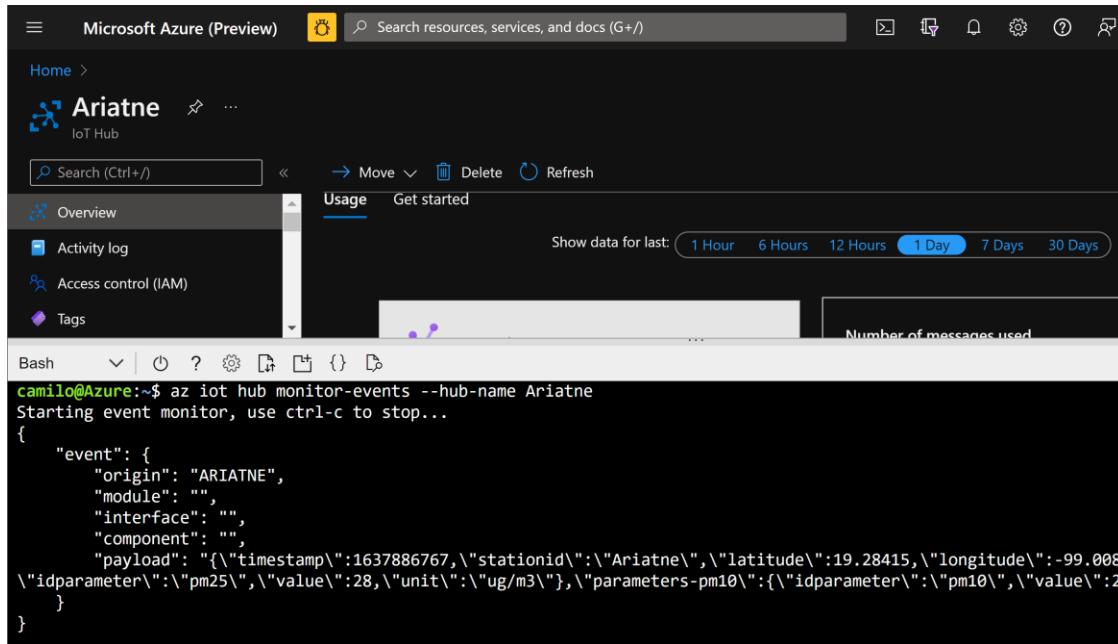


Ilustración 23 Configuración del IoT Hub mediante el hash de las estaciones de monitoreo.

Teniendo estos pasos y conectando la tarjeta de Arduino a la red podemos verificar que se reciben los mensajes por parte de la estación de monitoreo al Azure IoT Hub mediante la consola de PowerShell de Azure.



```
camilo@Azure:~$ az iot hub monitor-events --hub-name Ariatne
Starting event monitor, use ctrl-c to stop...
{
  "event": {
    "origin": "ARIATNE",
    "module": "",
    "interface": "",
    "component": "",
    "payload": "{\"timestamp\":1637886767,\"stationid\":\"Ariatne\",\"latitude\":19.28415,\"longitude\":-99.0088,\"idparameter\":\"pm25\",\"value\":28,\"unit\":\"ug/m3\"},\"parameters-pm10\":{\"idparameter\":\"pm10\",\"value\":28}
  }
}
```

Ilustración 24 Vista de la consola y recepción de datos

Sin embargo, los datos se están recibiendo de una manera cruda y necesitamos un procesamiento previo para poderlos enviar a nuestra base de datos no relacional.

### 6.2.2 Base de Datos No Relacional CosmosDB

La creación de nuestra base de datos no relacional se hace de manera transparente usando la interfaz de Azure, el primer paso es seleccionar la API correcta para nuestra base de datos no relacional, para este proyecto se usó la API Core (SQL) debido a que se usa T-SQL para las consultas y procesamiento de datos desde el Azure IoT Hub.

The screenshot shows the 'Select API option' page for creating a new Azure Cosmos DB account. It features two main options: 'Core (SQL) - Recommended' and 'Azure Cosmos DB API for MongoDB'. Both options include a 'Create' button and a 'Learn more' link.

**Core (SQL) - Recommended**

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

**Create**   [Learn more](#)

**Azure Cosmos DB API for MongoDB**

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

**Create**   [Learn more](#)

Ilustración 25 Selección de API para CosmosDB

Se crea la base de datos dentro de un grupo de recursos y seleccionando la región que debe de ser la misma que en la que se encuentra el Azure IoT Hub.

The screenshot shows the 'Create Azure Cosmos DB Account - Core (SQL)' configuration page. It includes sections for Project Details, Instance Details, and Capacity mode.

**Project Details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*: CE-chavezgal

Resource Group \*: MAQMS  
Create new

**Instance Details**

Account Name \*: newdatabaseaqms

Location \*: (US) East US

Capacity mode:  Provisioned throughput  Serverless  
[Learn more about capacity mode](#)

With Azure Cosmos DB free tier, you will get the first 1000 RU/s and 25 GB of storage for free in an account. You can enable free tier on up to one account per subscription. Estimated \$64/month discount per account.

Ilustración 26 Creación de recurso de CosmosDB

Una vez realizado el despliegue es posible crear las bases de datos para cada uno de nuestros dispositivos, como se puede observar se provisionaron 4 bases de datos para cada una de las estaciones de monitoreo, que estarán listas para recibir los documentos JSON una vez se haga el procesamiento de los datos.

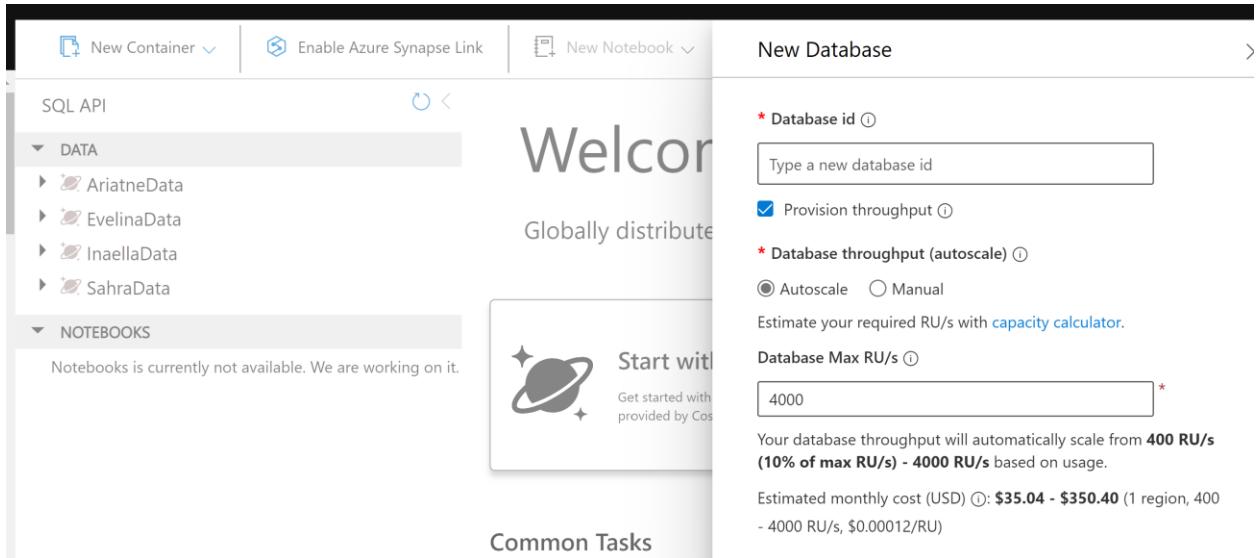


Ilustración 27 Creación de Base de Datos en el Data Explorer de CosmosDB

### 6.2.3 Procesamiento de datos con Stream Analytics Job

Se proporciona un recurso de Stream Analytics para poder procesar los datos en Azure IoT Hub y enviarlos correctamente a AzureCosmosDB, es importante que este sea creado dentro del mismo grupo de recursos y dentro de la misma región.

EvelinaVisualizationJob

Stream Analytics job

Search (Ctrl+ /)

Start Stop Delete

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Properties

Locks

Job topology

Running

Resource group (Move)  
MAQMS

Status  
Running

Location  
East US

Subscription (Move)  
CE-chavezgal

Subscription ID  
6cde05d1-f274-4be1-b155-7cffa93f509e

Created  
Monday, October 25, 2021, 6:45:20 PM

Started  
Monday, October 25, 2021, 6:48:19 PM

Output watermark  
Thursday, November 25, 2021, 11:48:14 PM

Cluster  
Shared

Hosting environment  
Cloud

Ilustración 28 Configuración del Stream Analytics Job

Con esto es posible conectar Azure IoT Hub con CosmosDB, el primer paso es establecer una entrada de datos para el Stream Analytics, y configurar el stream de datos desde el IoT Hub correcto.

+ Add stream input ▾ + Add reference input ▾

Inputs can't be added or edited while a job is running. You can stop the job to add or edit.

Name	Source type
EvelinaVisualizationInput	Stream

Subscription information not needed

IoT Hub ⓘ Ariatne

Consumer group ⓘ iothub-streaming

Shared access policy name ⓘ iothubowner

Ilustración 29 Configuración de la entrada para el Stream Analytics

Se establece la salida de datos apuntando hacia Azure CosmosDB y se configura correctamente el contenedor y la base de datos para guardar nuestro stream.

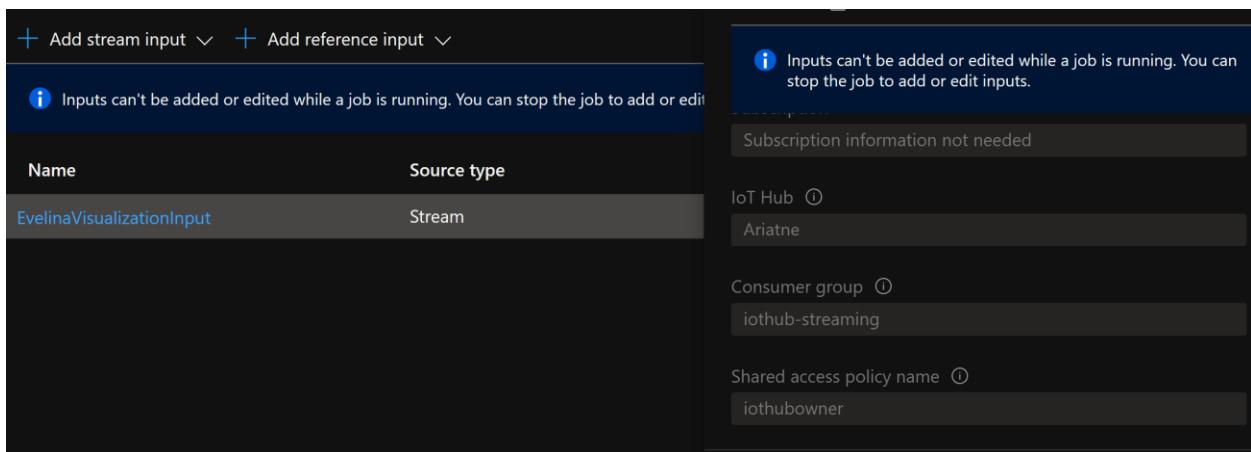


Ilustración 30 Configuración para la salida del Stream Analytics

Se creó un Query para poder tomar los datos desde el IoT Hub y seleccionar que datos se requieren usando la cláusula **where** para que estos sean procesados a CosmosDB.

The screenshot shows the 'Test query' editor with the following T-SQL query:

```
1  SELECT
2      *
3  INTO
4      EvelinaVisualizationOutput
5  FROM
6      EvelinaVisualizationInput
7  WHERE stationid = 'Evelina'
```

Ilustración 31 Creación de query para el Stream Analytics Job

De esta manera se logró conectar los datos desde el IoT Hub y enviarlos a CosmosDB, todo el procesamiento de datos se hace de manera transparente por el Stream Analytics.

## 6.3 Modelo de Machine Learning

### 6.3.1 Preparación de los datos

En primera instancia se necesitan descargar los datos de cada una de las estaciones de monitoreo registradas en el IoT Hub y en la base de datos de CosmosDB.

Este proceso se realiza mediante una herramienta proporcionada por Microsoft llamada Data Migration Tool [24], la cual entre muchas características nos permite realizar una conexión a la Base de Datos de CosmosDB usando un Connection String y posteriormente convertir toda la colección de la base de datos a un JSON que podremos descargar y utilizar.

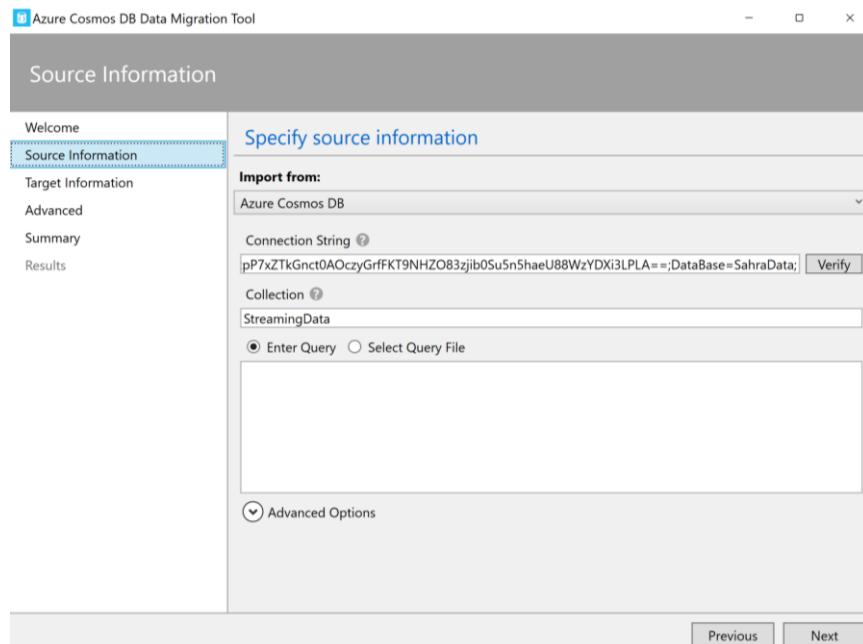


Ilustración 32 Verificación de conexión a CosmosDB mediante el connection string usando la herramienta de Data Migration Tool

Una vez teniendo nuestro archivo JSON, el siguiente paso es convertirlo a un archivo CSV para poder consumirlo en nuestro algoritmo de Machine Learning.

### 6.3.2 Ambiente en Python

Para la implementación del modelo de machine learning se usó Anaconda [25] mediante Jupyter Notebook, las librerías y hardware usado para este proceso se describen a continuación.

Características del hardware:

- GPU: NVIDIA GTX 1660ti 6GB VRAM.
- CPU: Intel Core i7 9na generación.
- RAM: 32 GB.

Librerías:

Las librerías usadas son las siguientes:

- Tensorflow
- Pandas
- Keras
- Sklearn
- Numpy
- Matplotlib

En el caso de Tensorflow y el soporte para el uso de GPU (*Graphic Process Unit*) se requirió de la instalación de drivers de Nvidia para el uso de CUDA que es una plataforma de cómputo paralelo que nos permite acelerar procesos computacionales requeridos por Tensorflow.

Manipulación de los datos

Los datos fueron manipulados y limpiados para que puedan usarse con la red LSTM, como se indicó en el Capítulo IV de Análisis, este modelo trabaja bien realizando

predicciones de una sola dimensión, es decir, con una sola característica y múltiples muestras, que es el caso del presente proyecto, el modelo predictivo se usará para predecir las muestras de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub> por lo tanto se usaron las funciones `df()` de la librería Pandas para dejar nuestro archivo de muestras con sólo dos columnas, el timestamp transformado a un formato de tipo fecha y el valor de las partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub>.

```
1. df = df[['date', 'pm25']]  
2. df.head()
```

Código 6.13

## Normalización

El método que se usó para la normalización de los datos fue *min-max scaler* de la librería sklearn, este método de normalización nos permite transformar nuestras características (es decir las columnas de nuestros datos [PM<sub>2.5</sub>/PM<sub>10</sub>]) dentro de un valor mínimo y un máximo que en este caso es 0 y 1.

La idea principal de la normalización está dada por el hecho de la parcialidad (bias), esta se presenta al tener variables que son medidas a diferentes escalas y por lo tanto no contribuyen de igual manera al modelo, por lo tanto, la normalización es un proceso importante a la hora de preparar nuestros datos para ser entrenados.

La función matemática para la normalización min-max scaler está dada por la siguiente expresión:

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

El siguiente código representa el uso de la normalización con la función `min_max_scaler` en Python:

```
1. scaler=MinMaxScaler(feature_range=(0,1))  
2. data.index=data.Date  
3. data.drop("Date",axis=1,inplace=True)  
4. final_data = data.values  
5. train_data=final_data[0:200,:]  
6. valid_data=final_data[200:,:]  
7. scaler=MinMaxScaler(feature_range=(0,1))  
8. scaled_data=scaler.fit_transform(final_data)  
9. x_train_data,y_train_data=[],[ ]  
10. for i in range(60,len(train_data)):
```

```

11.     x_train_data.append(scaled_data[i-60:i,0])
12.     y_train_data.append(scaled_data[i,0])

```

Código 6.14

En el Código 6.14 se define la función `MinMaxScaler()` con un rango de características entre 0 y 1, se define el índice de nuestros datos como Date en la función `index.data` y posteriormente se eligen los segmentos a utilizar para nuestros datos de entrenamiento y validación mediante `train_data` y `valid_data`, finalmente se guardan como un arreglo.

Definición del modelo LSTM.

Se define el modelo de LSTM usando la librería de Tensorflow asignando un modelo secuencial.

```

1. lstm_model=Sequential()
2. lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(np.shape(x_train_data)[1],1)
   ))
3. lstm_model.add(LSTM(units=50))
4. lstm_model.add(Dense(1))
5.

```

Código 6.15

De manera general el modelo LSTM funciona de la siguiente manera:

El primer paso es decidir qué información vamos a desechar en el estado del nodo, esta decisión es creada por la capa llamada ‘forget gate layer’, de la salida de  $h_{t-1}$  y  $x_t$  se obtienen los rangos entre 0 y 1 que nos dicen si la información es transmitida al siguiente nodo o no, 1 significa que necesita la información y 0 que debe desecharla.

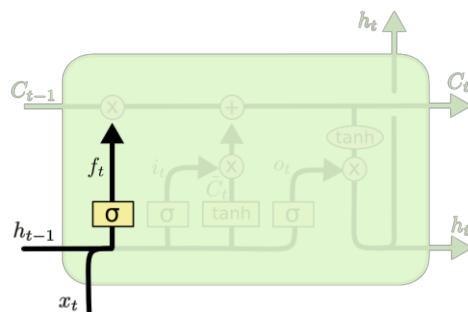


Ilustración 33 Paso 1 del modelo LSTM,  
Fuente: Understanding LSTM. [30]

El siguiente paso es decidir qué información nueva almacenaremos en el estado de nuestro nodo, este paso está dividido en dos, el primero definido por la función *sigma* que define que valores vamos a actualizar y *tanh* que crea un vector con los nuevos valores de los candidatos que pueden ser almacenados en el estado del nodo.

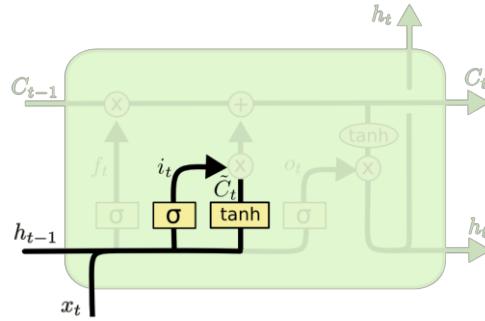


Ilustración 34 Paso 2 del modelo LSTM,  
Fuente: Understanding LSTM. [29]

El tercer paso es realizar lo que el anterior paso decidió, multiplicamos el anterior estado por  $f_t$  olvidando la información que decidimos anteriormente y sumando los nuevos candidatos.

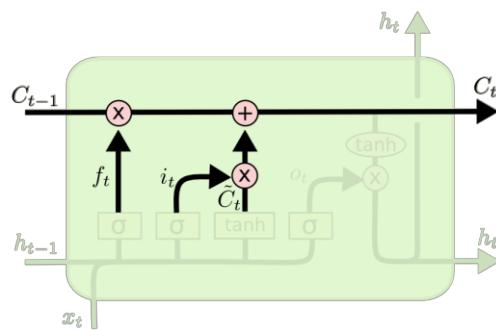


Ilustración 35 Paso 3 del modelo LSTM, Fuente:  
Understanding LSTM. [29]

Finalmente decidimos que mandaremos a la salida, esto está basado en el estado de nuestro nodo, la función *sigma* decide que parte del estado de nuestro nodo mandaremos a la salida, después forzamos a los valores a estar dentro de -1 y 1 con la

función  $tanh$  y lo multiplicamos por la salida de la función  $sigma$  de esta manera nos aseguramos que solo mandamos a la salida los datos que fueron decididos en el proceso.

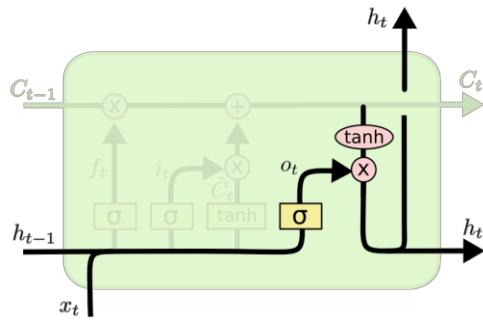


Ilustración 36 Paso Final del modelo LSTM,  
Fuente: Understanding LSTM. [29]

## Entrenamiento de los datos

Se asignan a la función de `lstm_model.compile` los argumentos de perdida (error cuadrático medio) y optimizador (adam), el optimizador ‘adam’ nos permite definir el algoritmo de optimización en el proceso de actualización de pesos en las iteraciones de los nodos, una vez hecho esto se asignan los datos de entrenamiento  $x$  y  $y$  del modelo y se define el rango y el arreglo de los datos de validación.

```

1. lstm_model.compile(loss='mean_squared_error',optimizer='adam')
2. lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
3. X_test=[]
4. for i in range(60,model_data.shape[0]):
5.     X_test.append(model_data[i-60:i,0])
6. X_test=np.array(X_test)
7. X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
8.

```

Código 6.16

Finalmente se produce la función de predicción, los resultados del modelo, así como los datos que fueron usados se definen en el Capítulo VII de Pruebas y Resultados.

# Capítulo VII

## Pruebas y Resultados

En este apartado se presentan los resultados de las pruebas ejecutadas de acuerdo con la sección 1.8 Escenario de Pruebas, propuesto en el Capítulo I. Se incluyen capturas de pantalla, imágenes y gráficas que muestran los resultados obtenidos en el presente proyecto.

### 7.1 Estaciones de monitoreo.

Como se describió en el escenario de pruebas las estaciones de monitoreo realizaron distintos periodos de toma de muestras.

#### Estación de monitoreo estacionaria.

La estación de monitoreo estacionaría fue diseñada para tomar lecturas de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub> y coordenadas (latitud y longitud) cada hora durante las 24 horas del día durante 2 meses de pruebas, el código de la estación que implementó una tarjeta Arduino MKR WiFi 1010 espera 1 hora para mandar la cadena de mensajes mediante el protocolo MQTT y toma en cuenta el periodo que necesitan los sensores para tomar muestras correctas que es de 5 segundos.

La estación de monitoreo estacionaria cómo se describe en el Capítulo V: Diseño, está constituida por una placa Arduino MKR WiFi 1010, un módulo de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub> y un módulo de GPS, la construcción de esta estación fue hecha como se muestra en la *ilustración 36* y que de esta manera le permite ser fijada por las cuerdas a una base y ser atornillada en la fachada del lugar a ser colocada.

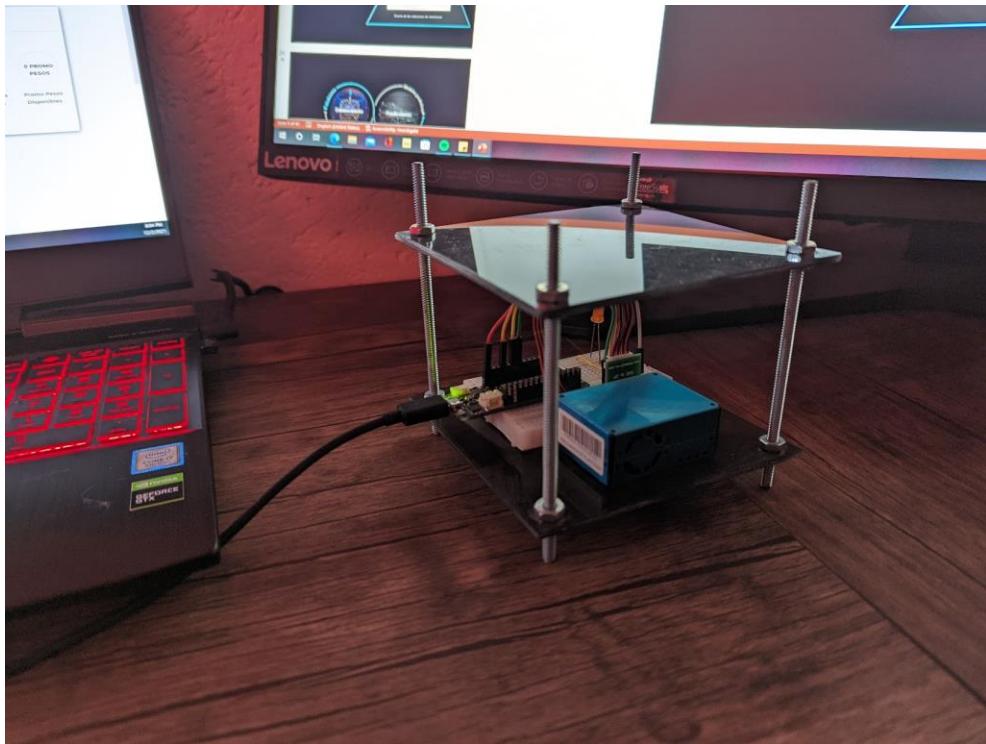


Ilustración 37 Estación de monitoreo estacionaria.

Cómo se describe en el Capítulo I, fueron construidas 3 estaciones de monitoreo fijas, las cuales fueron colocadas en la fachada del domicilio ubicado en Calle Montes Pirineos de la Colonia Selene, Tláhuac, en la caseta de policía a la entrada de la UPIITA y en el CIC dentro de las instalaciones, respectivamente.

En la siguiente ilustración se aprecia la primera estación de monitoreo denominada con el id 'Evelina' en el domicilio de la Calle Montes Pirineos, esta estación de monitoreo fue colocada el 30 de septiembre de 2021 y continuo realizando lecturas cada hora hasta el 26 de Noviembre de 2021, obtuvo un total de 1015 lecturas.



*Ilustración 38 Estación de monitoreo Evelina.*

La siguiente ilustración muestra a la estación de monitoreo con el id 'Inaella' colocada y funcionando en la caseta de policías de la UPIITA.

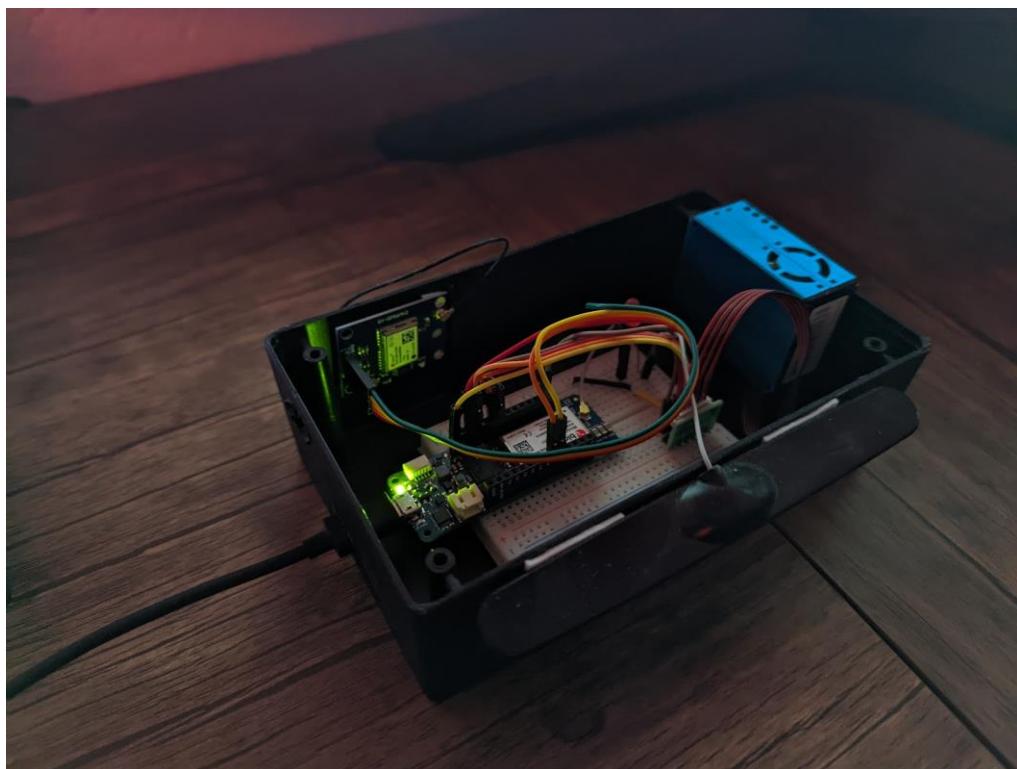


*Ilustración 39 Estación de monitoreo Inaella.*

## Estación de monitoreo móvil.

La estación de monitoreo fija fue diseñada para tomar lecturas de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub> y coordenadas (latitud y longitud) cada 5 segundos durante el tiempo que toma realizar el recorrido propuesto en el Capítulo I, el código de la estación que implementó una tarjeta Arduino MKR GSM 1400 espera 5 segundos para mandar la cadena de mensajes mediante el protocolo MQTT y toma en cuenta el periodo que necesitan los sensores para tomar muestras correctas que es de 5 segundos.

La estación de monitoreo móvil cómo se describe en el Capítulo V: Diseño, está constituida por una placa Arduino MKR GSM 1400, un módulo de partículas suspendidas PM<sub>2.5</sub>/PM<sub>10</sub> y un módulo de GPS, la construcción de esta estación fue hecha como se muestra en la *ilustración 40*.



*Ilustración 40 Estación de monitoreo móvil.*

El diseño de esta estación de monitoreo fue precisamente hecho para que pueda ser transportado en un vehículo, la siguiente ilustración muestra cómo esta fue montada sobre una bicicleta para realizar el recorrido propuesto en el Capítulo I.



*Ilustración 41 Estación de monitoreo móvil sobre el vehículo a utilizar para los recorridos.*

El tiempo promedio de toma de muestras en cada recorrido fue de 8 a 10 minutos y se realizó dos veces al día durante 5 días a la semana (Lunes - Viernes), se usó una batería portátil de 10000 mAh para dar energía a la estación de monitoreo.

## 7.2 Transmisión de datos mediante protocolo MQTT.

Para el proceso de la transmisión de datos se hizo uso del protocolo MQTT como se indica en el Capítulo V: Diseño, este protocolo permite la comunicación entre la estación de monitoreo y el broker de la nube de Azure, en el caso de la estación de monitoreo fija, la transmisión se hace usando la red WiFi disponible.

Estación de monitoreo móvil.

En el caso de la transmisión de datos mediante la red GSM se usó Hologram que nos permite usar la red bajo demanda aprovechando la infraestructura de AT&T, este portal nos permite enviar datos y cobrarlos por kilobyte, lo que nos permite reducir costos en el envío de paquetes a través de la red móvil.

El portal de Hologram (Hologram.io) nos permite monitorear el costo y el envío de paquetes usando la red GSM.

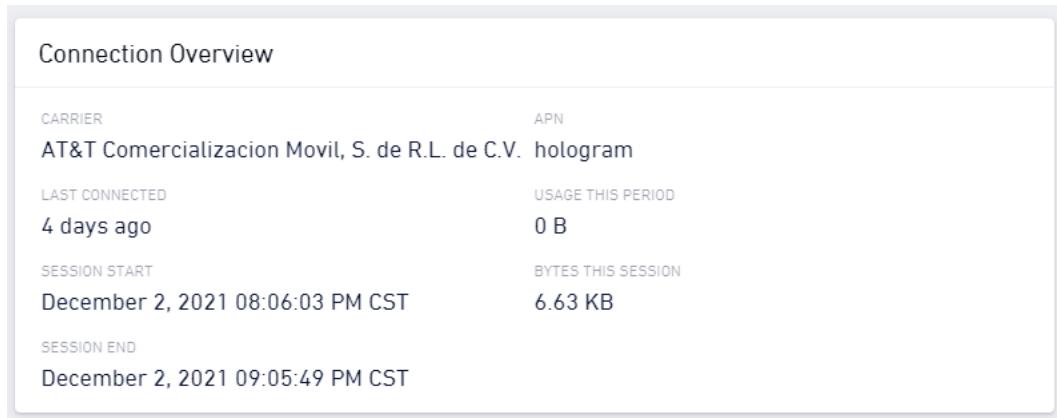
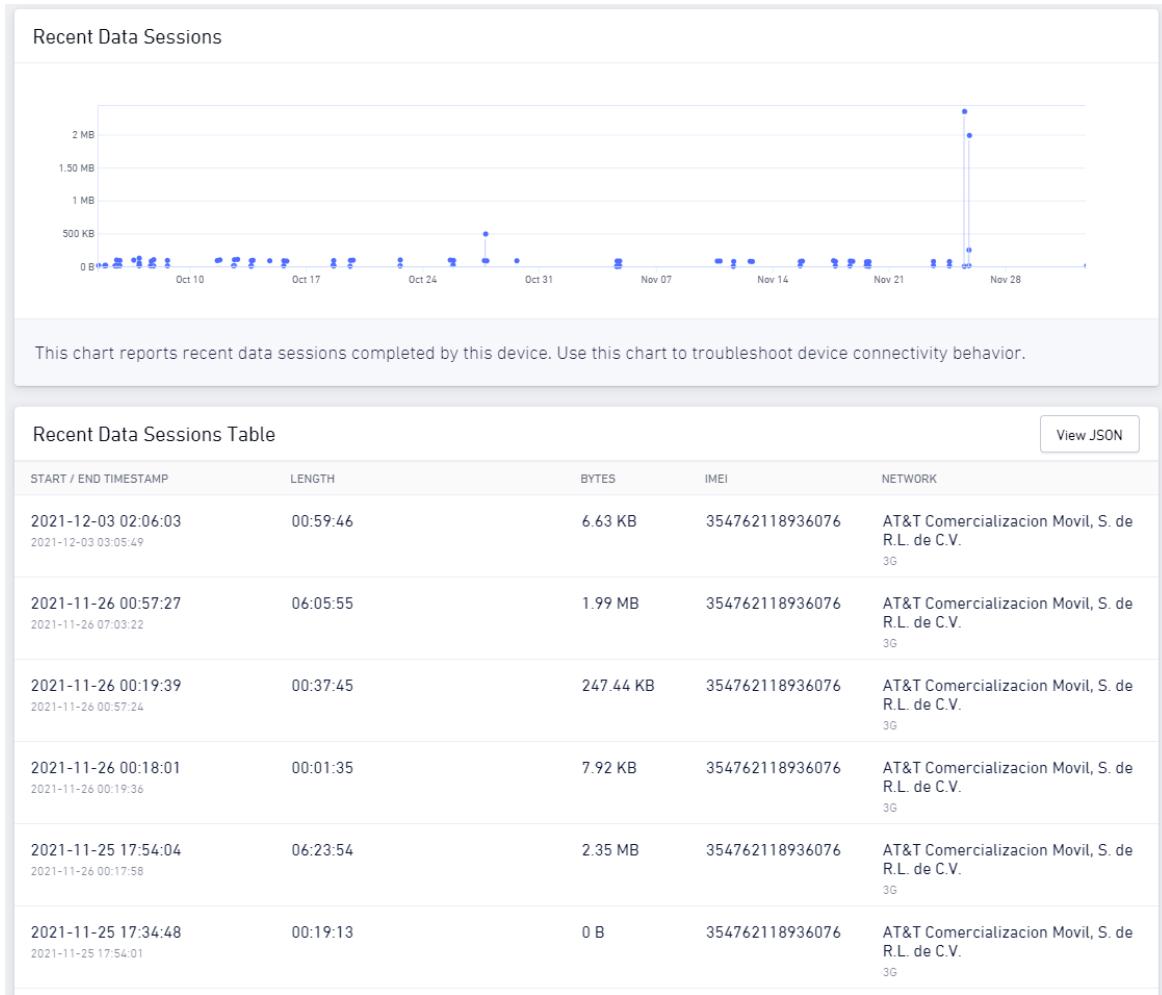


Ilustración 42 Panel de información de Hologram

La siguiente ilustración nos permite visualizar cada una de las conexiones que realizó la estación de monitoreo a la red y el tamaño del mensaje en kilobytes.



*Ilustración 43 Conexiones de la estación de monitoreo a la red GSM.*

### 7.3 Recepción y monitoreo de datos en Azure IoT Hub.

La recepción de los datos en el Azure IoT Hub puede ser monitoreado de dos maneras, mediante la consola y de manera gráfica usando el *dashboard* del IoT Hub.

Para visualizar los datos que van llegando al broker podemos abrir la consola y correr el siguiente comando:

```
1. az iot hub monitor-events -hub-name Ariatne
```

*Código 7.1*

```

camilo@Azure:~$ az iot hub monitor-events --hub-name Ariatne
Starting event monitor, use ctrl-c to stop...
{
  "event": {
    "origin": "ARIATNE",
    "module": "",
    "interface": "",
    "component": "",
    "payload": "{\"timestamp\":1637886767,\"stationid\":\"Ariatne\",\"latitude\":19.28415,\"longitude\":-99.0088,\"idparameter\":\"pm25\",\"value\":28,\"unit\":\"ug/m3\"},\"parameters-pm10\":{\"idparameter\":\"pm10\",\"value\":28}
  }
}

```

Ilustración 44 Recepción de cadena de mensajes en consola.

Cómo se puede observar en la Ilustración 44, el monitoreo nos muestra el paquete de datos enviado por la estación en tiempo real, el *payload* no tiene formato y por ello es necesario procesar los datos por el Stream Analytics antes de poder almacenarlos en la base de datos no relacional CosmosDB.

El monitoreo también puede ser de manera gráfica y la siguiente ilustración permite observar los datos que podemos monitorear en este *dashboard*.

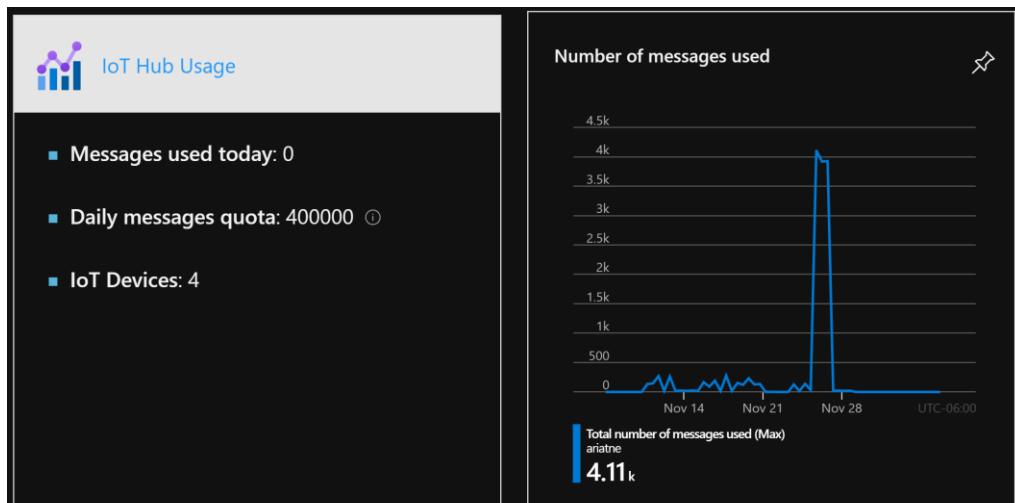


Ilustración 45 Monitoreo de mensajes usados por el broker.

De manera similar, los datos de conexión y el número de conexiones dispositivo-broker pueden ser medidos.

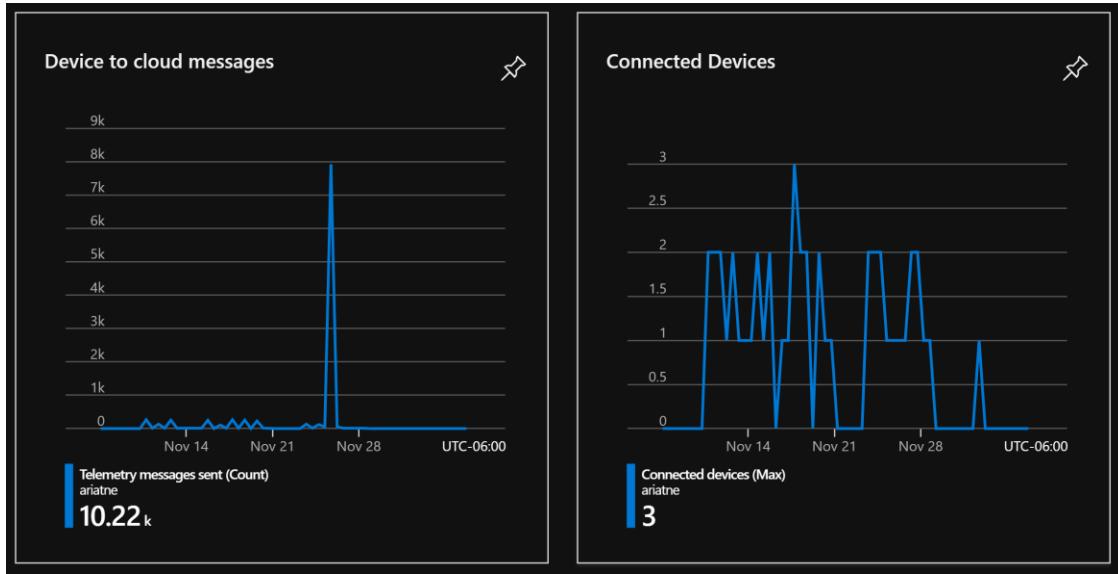


Ilustración 46 Monitoreo de mensajes en el broker.

## 7.4 Recepción de datos en CosmosDB.

Una vez que el Trigger de Stream Analytics procesa los datos, estos son almacenados en la base de datos no relacional de CosmosDB, la siguiente ilustración permite visualizar cómo se pueden ver los datos almacenados dentro del explorador de CosmosDB.

The screenshot shows the Azure Cosmos DB Data Explorer interface. On the left, there's a navigation pane with "SQL API" selected, followed by "DATA" which contains "AriatneData", "EvelinaData", "InaellaData", "SahraData", and "StreamingData". "StreamingData" is expanded, showing "Items", "Scale & Settings", "Stored Procedures", "User Defined Functions", and "Triggers". Below this is a "NOTEBOOKS" section with a note: "Notebooks is currently not available. We are working on it." At the top right, there's a search bar with "Items" and a "SELECT \* FROM c" query. Below the query is an "Edit Filter" button. The main area displays a JSON document with the following content:

```

1  "timestamp": 1635186618,
2  "stationid": "Sahra",
3  "latitude": "",
4  "longitude": "",
5  "parameters-pm25": {
6      "idparameter": "pm25",
7      "value": 0,
8      "unit": "ug/m3"
9  },
10 "parameters-pm10": {
11     "idparameter": "pm10",
12     "value": 0,
13     "unit": "ug/m3"
14 },
15 "EventProcessedUtcTime": "2021-10-25T23:30:20.800671"
16

```

Ilustración 47 Data Explorer de CosmosDB

## 7.5 Resultados de las pruebas

Las pruebas realizadas por las 4 estaciones de monitoreo fueron comprendidas en un periodo de 2 meses del 30 de Septiembre de 2021 al 26 de Noviembre de 2021, la primera estación de monitoreo denominada Evelina (estación fija) estuvo en todo el periodo mencionado anteriormente, así como la estación de monitoreo Ariatne (estación móvil), en el caso de las estaciones de monitoreo Sahra e Inaella (estaciones fijas), estas estuvieron instaladas y mandando mensajes por un periodo comprendido del 26 de Octubre al 26 de Noviembre y del 7 de Octubre al 26 de Noviembre respectivamente.

Número de muestras por estación:

El total de muestras obtenidas por cada una de las estaciones, así como el periodo se muestra en la siguiente tabla.

<b>idEstación</b>	<b>Ubicación</b>	<b>Número de muestras</b>	<b>Periodo</b>	<b>Tiempo</b>
<b>Evelina</b>	Tláhuac	1015	30 septiembre – 26 noviembre.	2 meses.
<b>Ariatne</b>	Tláhuac	14016	30 septiembre – 26 noviembre.	2 meses.
<b>Inaella</b>	UPIITA	89	7 octubre – 26 noviembre.	1 mes y medio.
<b>Sahra</b>	CIC	99	26 octubre – 26 noviembre.	1 mes.

*Tabla 29 Resultados estaciones de monitoreo.*

La estación de monitoreo fija Evelina estuvo conectada a una red WiFi privada y por ende tuvo una estabilidad de conexión ALTA, lo que resultó en tomas de muestras constantes y sin interrupciones durante todo el periodo de pruebas.

La estación de monitoreo móvil Ariatne estuvo conectada a la red GSM lo cual garantiza la conexión en cada uno de los recorridos realizados y por lo tanto, una estabilidad de conexión ALTA y tomas constantes y sin interrupciones en todo el periodo de pruebas.

Por otro lado, las estaciones de monitoreo fijas Sahra e Inaella fueron colocadas en lugares con conexiones públicas, lo que resultó en una inestabilidad ALTA de la conexión y por lo tanto lecturas con interrupciones constantes.

El siguiente mapa muestra la ubicación de todas las estaciones de monitoreo en la Ciudad de México.

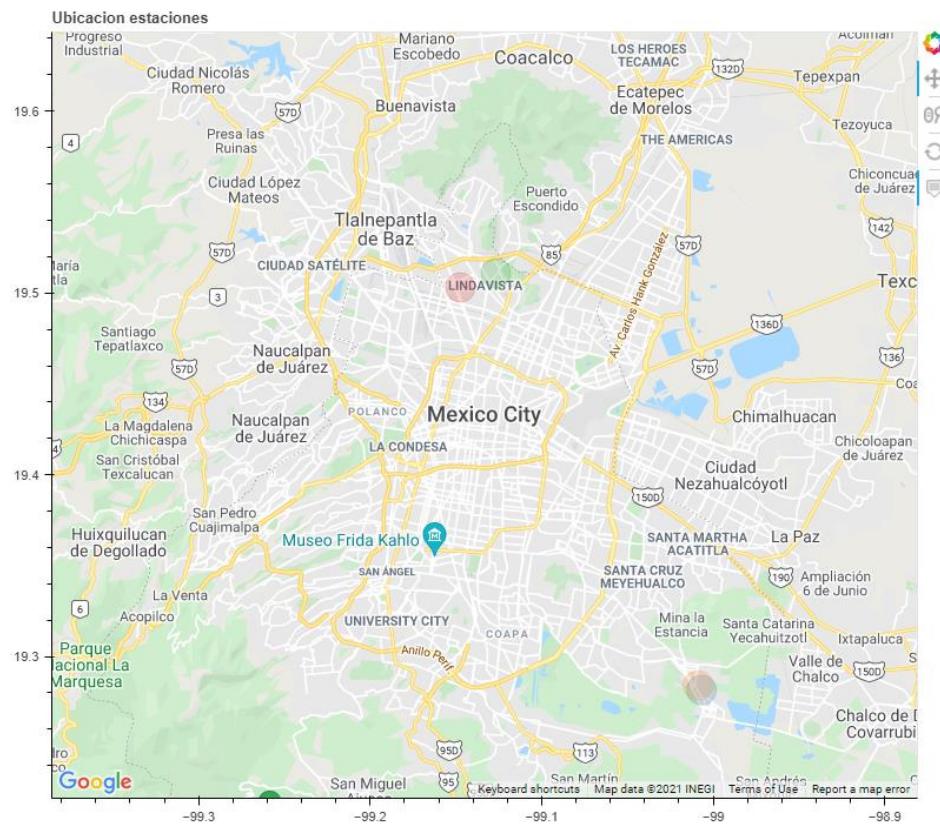


Ilustración 48 Mapa de estaciones en la Ciudad de México.

### 7.5.1 Resultados estación de monitoreo móvil (recorridos)

La estación de monitoreo móvil recabó 14016 muestras en un periodo de 2 meses, de acuerdo con la SEDEMA [3] las regulaciones para los niveles de calidad del aire respecto a las partículas suspendidas se categorizan de la siguiente manera.

Contaminante	Valores	Calidad
<b>PM<sub>2.5</sub>/PM<sub>10</sub></b>	0-50	Buena
<b>PM<sub>2.5</sub>/PM<sub>10</sub></b>	51-100	Regular
<b>PM<sub>2.5</sub>/PM<sub>10</sub></b>	101-150	Mala
<b>PM<sub>2.5</sub>/PM<sub>10</sub></b>	151-200	Muy Mala
<b>PM<sub>2.5</sub>/PM<sub>10</sub></b>	201-300	Extremadamente Mala

Tabla 30 Relación de calidad del aire.

En este caso la siguiente ilustración permite observar el promedio de valores obtenidos por la estación de monitoreo Ariatne (móvil) en todos sus recorridos.

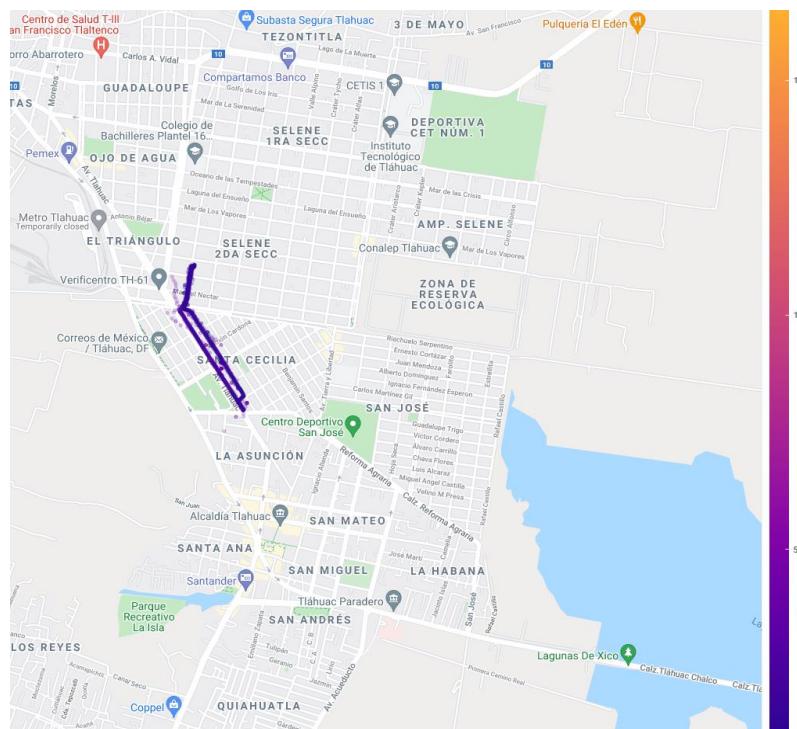


Ilustración 49 Promedio de lecturas estación de monitoreo móvil.

Cómo se logra observar, el promedio de lecturas siempre ronda entre los 30 a 50 puntos, lo cual nos indica un nivel de calidad del aire adecuado para las personas.

## 7.6 Modelo Predictivo (LSTM)

Para el entrenamiento y validación de nuestro modelo se usaron los datos de las estaciones que obtuvieron mayor número de muestras, de acuerdo con la *Tabla 29* estas fueron la estación de monitoreo móvil (Ariatne) y la estación de monitoreo fija (Evelina) con 14016 y 1015 muestras respectivamente.

### 7.6.1 Estación de monitoreo Fija (Evelina)

Se realizó la limpieza de los datos para preparar los datos de entrenamiento y validación, dejando sólo los datos que interesan para el propósito del modelo de machine learning, que son *timestamp* y *parameters-pm25/value* y *parameters-pm10/value*, el valor de la columna *timestamp* fue transformado a tipo *date* y la ejecución se realizó en dos fases, la primera con los valores de partículas suspendidas PM<sub>2.5</sub> y la segunda con los valores de partículas suspendidas PM<sub>10</sub>, la siguiente ilustración indica 1015 muestras en un periodo de 2 meses para la estación de monitoreo con valores de PM<sub>2.5</sub>.

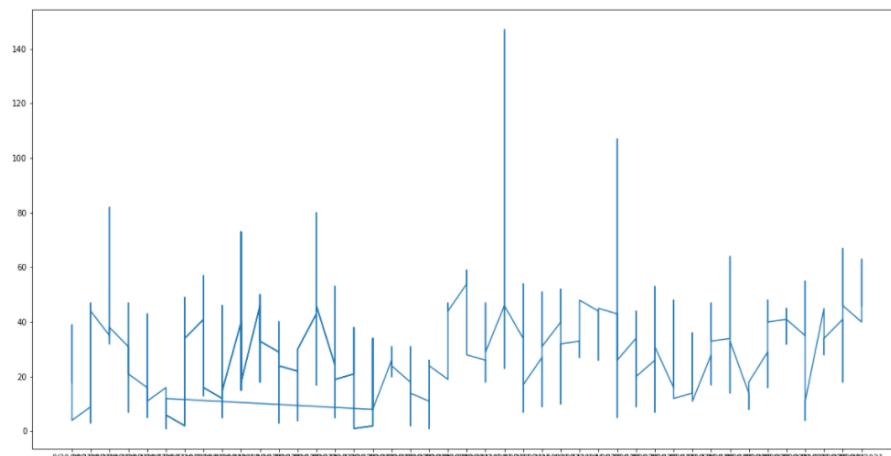


Ilustración 50 Muestras de PM25 en estación de monitoreo fija.

El entrenamiento del modelo usó 2/3 partes de las muestras obtenidas y 1/3 parte para los datos de validación, con ello se obtuvo el siguiente resultado.

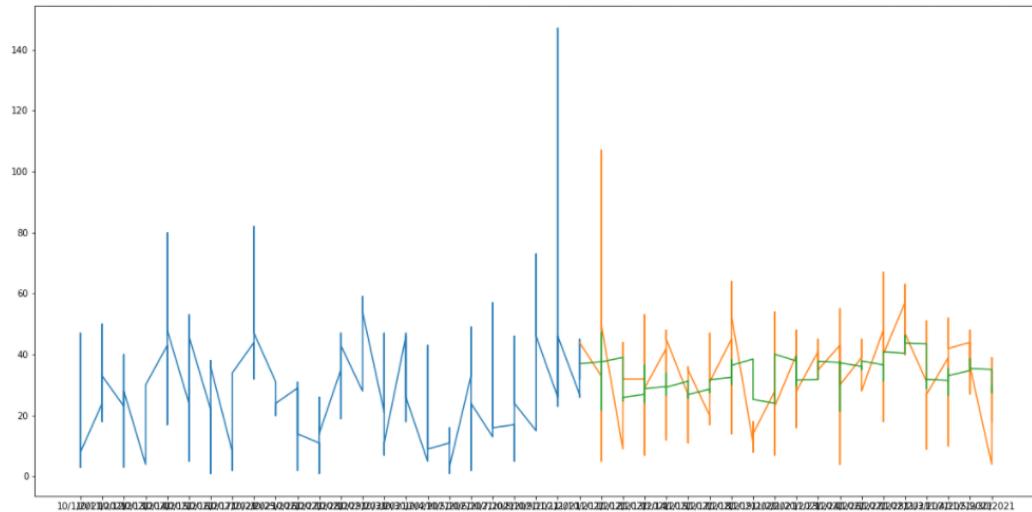


Ilustración 51 Datos de validación de PM25 para estación de monitoreo fija.

La línea azul representa los datos de entrenamiento, mientras que la línea naranja son los datos actuales y la línea verde los valores de la predicción del modelo.

Derivado del entrenamiento realizado se obtuvo una pérdida de 0.0098, la pérdida está definida por el valor cuadrático medio e indica el desempeño del modelo de acuerdo con el porcentaje de muestras que se usaron para la validación que fueron incorrectas, este valor está entre 0 y 1, donde 0 es un mejor desempeño y mientras más nos acerquemos a 1 significa un peor desempeño.

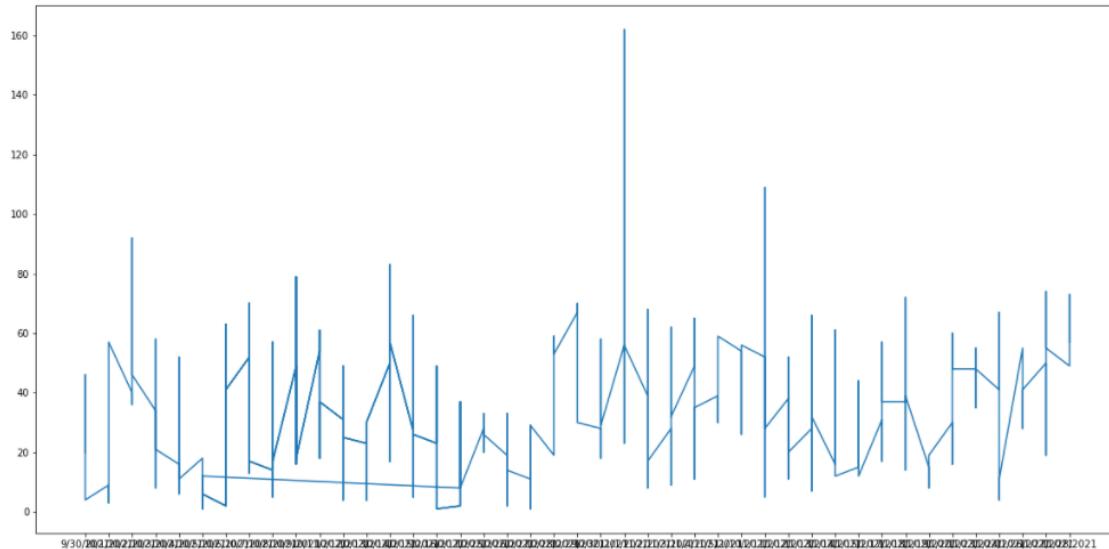
Los datos son 617 muestras para el set de entrenamiento, al modelo le tomó 10 segundos completar el entrenamiento y la pérdida de 0.0098.

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

617/617 - 10s - loss: 0.0098
```

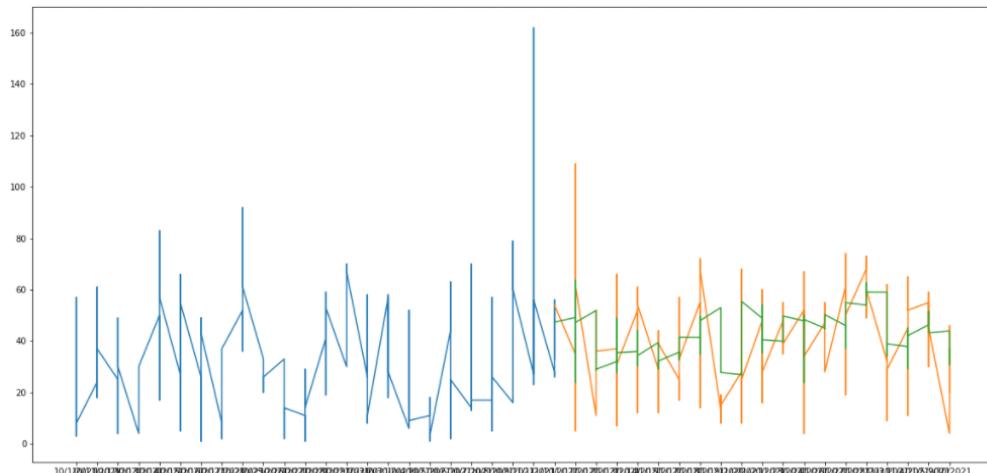
Ilustración 52 Resultados del desempeño del modelo para pm25 en estación fija.

El siguiente paso fue repetir el mismo proceso, pero esta vez con los valores de partículas suspendidas PM<sub>10</sub>. La siguiente ilustración muestra las muestras de PM<sub>10</sub> en un periodo de dos meses para la estación de monitoreo fija (Evelina).



*Ilustración 53 Historia de muestras PM10 para la estación de monitoreo fija.*

El entrenamiento de manera similar al anterior usó 2/3 partes de las muestras obtenidas y 1/3 parte para los datos de validación, con ello se obtuvo el siguiente resultado.



*Ilustración 54 Resultados del entrenamiento para PM10 en estación fija, verde: predicción, naranja: valores reales.*

El entrenamiento fue realizado con 617 muestras en un tiempo de 10 segundos y con una pérdida de 0.0113 que es ligeramente peor que el obtenido con PM<sub>2.5</sub>.

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

617/617 - 10s - loss: 0.0113
```

Ilustración 55 Perdida obtenida del modelo para PM10 con la estación fija.

## 7.6.2 Estación de monitoreo móvil (Ariatne)

De manera similar que en la preparación del modelo anterior se realizó la limpieza de los datos para el entrenamiento y validación, dejando sólo los datos que interesan para el propósito del modelo de machine learning que son *timestamp* y *parameters-pm25/value* y *parameters-pm10/value*, el valor de la columna *timestamp* fue transformado a tipo *date* y la ejecución se realizó en dos fases, con los valores de partículas suspendidas PM<sub>2.5</sub> y la segunda con los valores de partículas suspendidas PM<sub>10</sub>, la siguiente ilustración indica 14016 muestras en un periodo de 2 meses para la estación de monitoreo con valores de PM<sub>2.5</sub>.

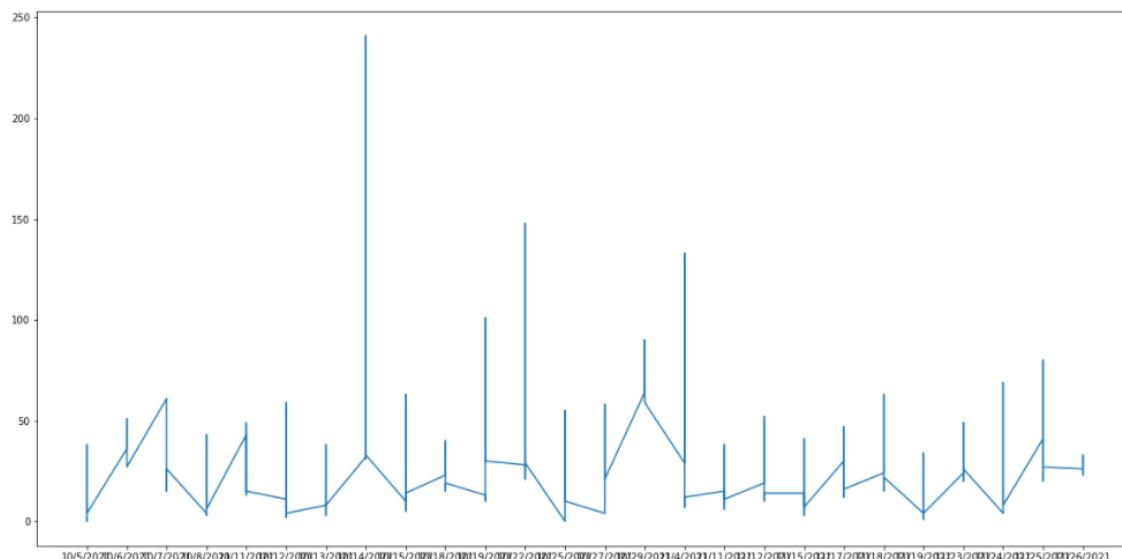
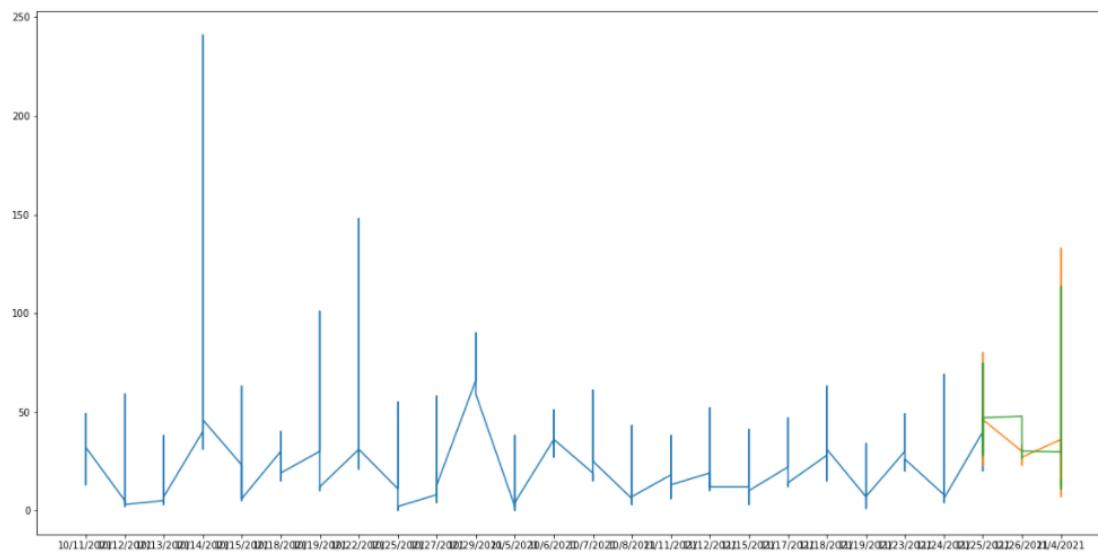


Ilustración 56 Historia de datos de PM25 para estación móvil.

El modelo utilizó 2/3 partes para el set de datos de entrenamiento, es decir, 9284 muestras y 1/3 parte restante para el set de datos de validación.



*Ilustración 57 Datos de validación de PM25 para estación móvil. Verde: predicción, Naranja: valores reales.*

Podemos concluir con la *ilustración 58* el éxito del modelo basándonos en los resultados obtenidos, con 9284 muestras utilizadas para entrenamiento y un tiempo de 129 segundos se obtuvo una pérdida de 0.0006676 definida por el valor cuadrático medio, representando un valor mínimo de errores que tuvo el modelo respecto a los valores reales.

```

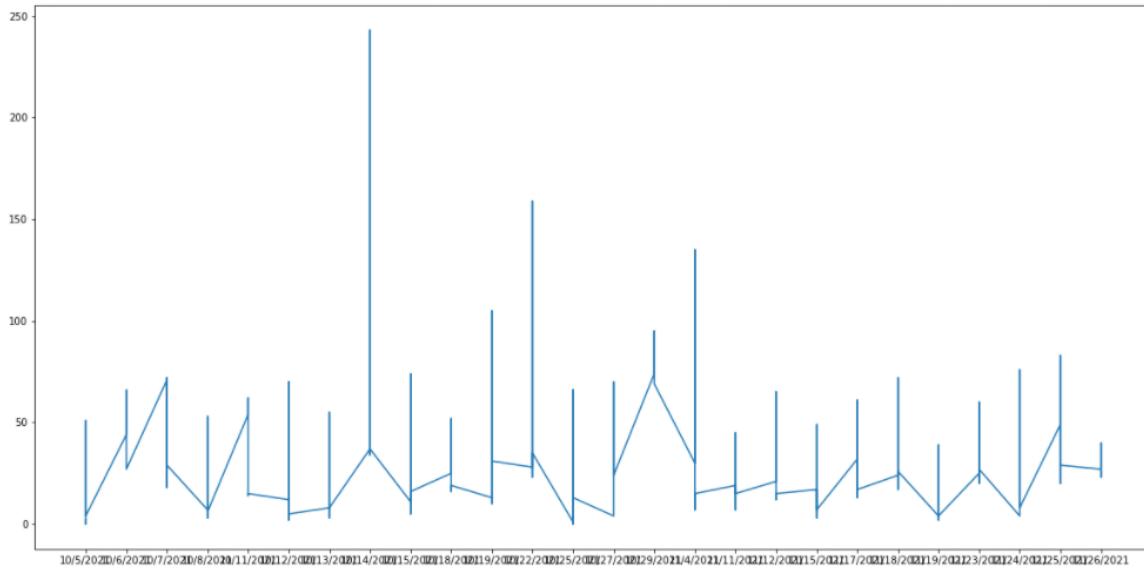
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

```

9284/9284 - 129s - loss: 6.6676e-04

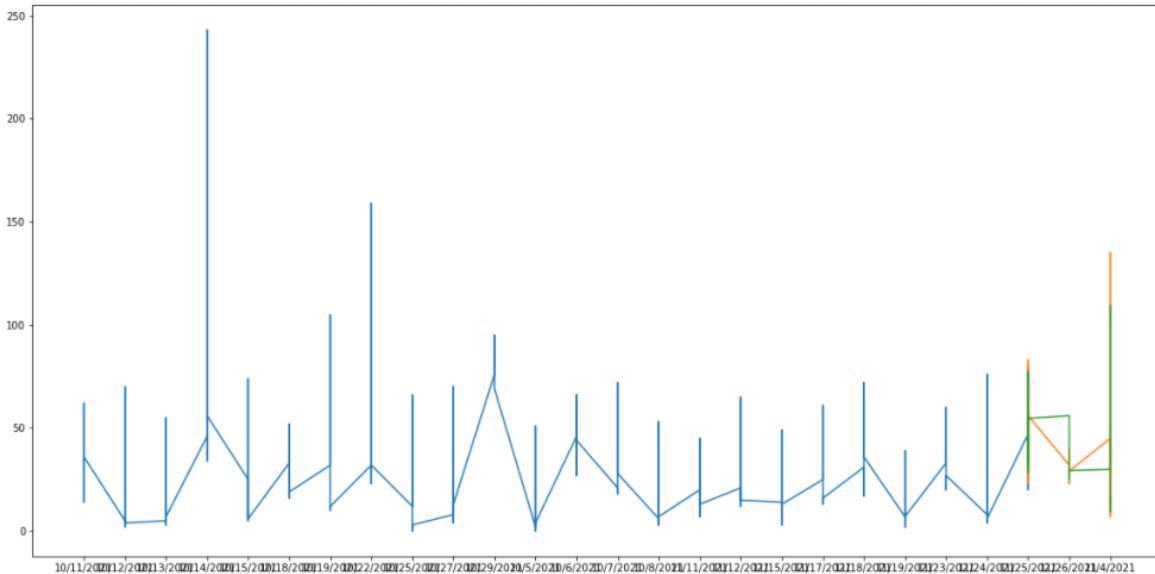
*Ilustración 58 Perdida del modelo para PM25 en la estación móvil.*

El último paso en la validación de nuestro modelo es con el set de datos de partículas suspendidas PM<sub>10</sub> de la estación de monitoreo móvil.



*Ilustración 59 Historia de datos de PM<sub>10</sub> para la estación de monitoreo móvil.*

La ilustración 59 muestra la historia de datos de PM<sub>10</sub> obtenida en un periodo de 2 meses y con 14016 muestras.



*Ilustración 60 Validación de datos del modelo con PM<sub>10</sub> en la estación de monitoreo móvil. Verde: predicciones y Naranja: valores reales.*

Finalmente se muestran en la siguiente ilustración las métricas de desempeño de nuestro modelo para las partículas suspendidas PM<sub>10</sub>, con 9284 muestras para el set de entrenamiento, en un tiempo de 129 segundos y con una pérdida de 0.0008674 definida por el valor cuadrático medio que nos indica el porcentaje de error que obtuvo el modelo, siendo que el valor se aproxima a 0 significa que nuestro modelo obtuvo una precisión ALTA en las predicciones obtenidas.

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

9284/9284 - 129s - loss: 8.6746e-04
```

*Ilustración 61 Perdida obtenida en el modelo para PM10 en la estación móvil.*

# Capítulo VIII

## Validación

En este Capítulo se presenta una comparación entre los objetivos planteados inicialmente y los resultados obtenidos luego del desarrollo e implementación del presente proyecto, cuyo propósito es el de verificar el cumplimiento de los objetivos planeados en el Capítulo I.

### 8.1 Objetivos Específicos.

Se presenta la verificación de los objetivos específicos del proyecto frente a las implementaciones realizadas.

*Integración de un módulo de sensores que realicen tomas de muestras de manera periódica de partículas suspendidas ( $PM_{10}$ ,  $PM_{2.5}$ ).*

La construcción de las estaciones de monitoreo implicó la implementación de un sensor de partículas suspendidas de  $PM_{10}$  y  $PM_{2.5}$  que realiza tomas de muestras de manera periódica en diferentes escenarios, en el caso de las estaciones de monitoreo móviles la toma de muestras se realizó cada 5 segundos por el tiempo del recorrido de la estación, este tomando en promedio 8 a 10 minutos, por otro lado, las estaciones de monitoreo fijas realizan un lectura de muestras cada hora durante las 24 horas del día, el periodo comprendido de pruebas fue de 2 meses entre el 30 de septiembre de 2021 y 26 de Noviembre de 2021.

La siguiente ilustración muestra a ambas estaciones de monitoreo integrando el sensor de partículas suspendidas de  $PM_{10}$  y  $PM_{2.5}$  que es la caja de color azul con modelo PMS5002.



Ilustración 62 Estaciones de monitoreo móvil y fija.

*Se integró al módulo de sensores un módulo que permita transmitir la información obtenida por los sensores en tiempo real usando la red 4G o en su caso WiFi.*

La integración al módulo de sensores con las placas de Arduino MKR WiFi 1010 y Arduino MKR GS 1400 permiten transmitir los datos obtenidos por las estaciones de una manera periódica usando la red disponible WiFi o GSM/4G respectivamente, en el caso de la estación de monitoreo fija, se construyeron 3, una de ellas funcionando con una red privada y las dos restantes con un red pública, los datos y resultados obtenidos nos indican que la mejor opción es una red privada de WiFi para obtener un mayor disponibilidad y confiabilidad en la toma de muestras.

Para la estación de monitoreo móvil se usó el proveedor Hologram que nos permite usar la red de AT&T para enviar los datos usando una conexión 3G/4G, los resultados obtenidos indican que de esta forma se puede garantizar la alta disponibilidad en el envío de datos a la red.

Connection Overview	
CARRIER	APN
AT&T Comercializacion Movil, S. de R.L. de C.V.	hogram
LAST CONNECTED	USAGE THIS PERIOD
4 days ago	0 B
SESSION START	BYTES THIS SESSION
December 2, 2021 08:06:03 PM CST	6.63 KB
SESSION END	
December 2, 2021 09:05:49 PM CST	

Ilustración 63 Dashboard de Hogram usando la red de AT&T para el envío de datos usando la red 3G/4G

Implementación de un servicio de Base de Datos en la nube usando la plataforma de Azure que almacene todas las muestras recolectadas por el sistema de monitoreo.

La arquitectura presentada en el presente proyecto consta de un broker (Azure IoT Hub), un recurso de procesamiento de datos y analíticas que permiten trabajar con el stream recibido por el broker y procesarlo para almacenar los datos en una base de datos no relacional usando la API SQL de CosmosDB.

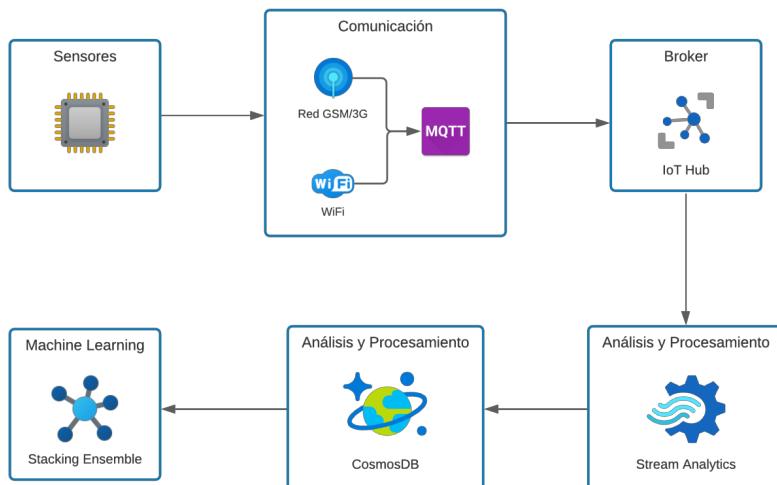
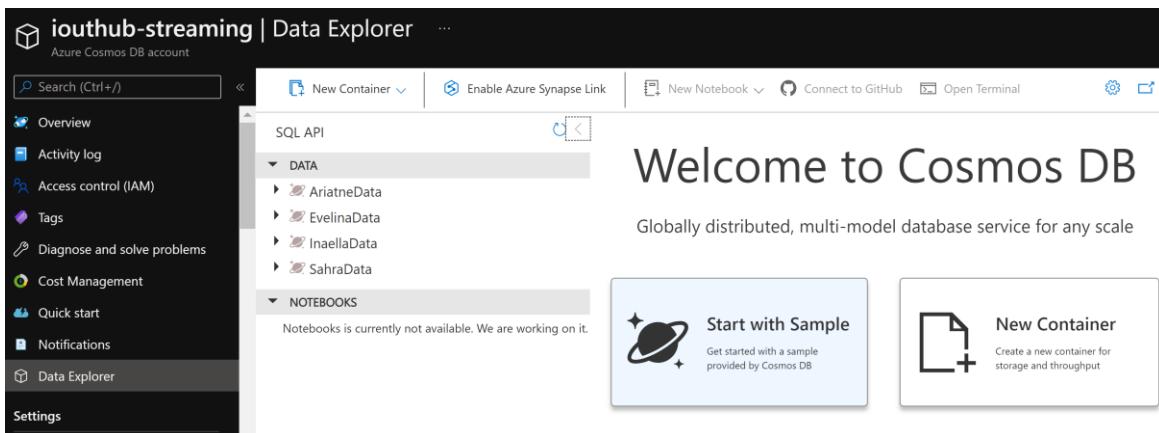


Ilustración 64 Arquitectura de Azure

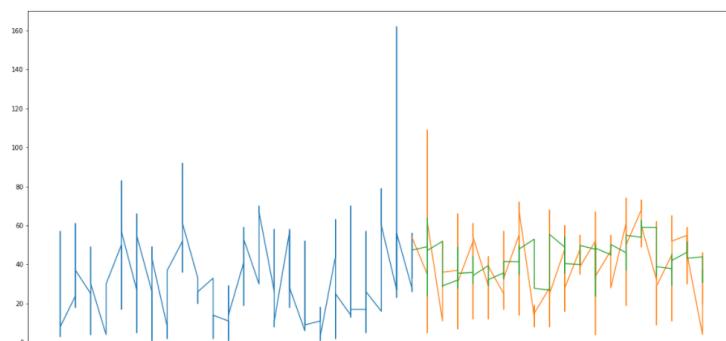


*Ilustración 65 Data Explorer de CosmosDB*

La *ilustración 65* nos muestra el recurso provisionado de CosmosDB llamado iothub-streaming y que nos permite administrar las bases de datos creadas para cada una de las estaciones y sus contenedores y documentos JSON correspondientes.

*Creación de un algoritmo para obtener modelos predictivos mediante técnicas de Machine Learning.*

El modelo de machine learning creado usando la técnica de LSTM (Long Term-Short Memory) propuesta en el Capítulo I, fue creada usando Anaconda y mediante Jupyter Notebook, se usaron las librerías de Tensorflow, Keras, Pandas, Numpy y Matplotlib y la creación de este se presenta con más detalle en el Capítulo VI, el uso de esta técnica nos permite tomar ventaja de dependencias a largo plazo que nos permiten crear modelos predictivos con una mayor precisión.



*Ilustración 66 Modelo LSTM, datos de validación.*

*Interpretar los datos obtenidos del modelo predictivo para comprobar la eficacia del sistema.*

Finalmente, la *ilustración 67* permite visualizar parte de la representación del modelo LSTM y analizar el performance obtenido en una de las ejecuciones para predecir las partículas suspendidas, derivado de estos resultados podemos corroborar la eficacia del modelo LSTM que es parte del sistema propuesto.

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

9284/9284 - 129s - loss: 8.6746e-04
```

*Ilustración 67 Parte del código del modelo LSTM.*

En general y de acuerdo con la *tabla 29* se logra apreciar que el sistema cumple con la función de enviar los datos por los periodos e intervalos propuestos, y que estos son recibidos en todos los casos por el broker para poder ser eventualmente procesados y almacenados en CosmosDB, en total se obtuvieron 14016 muestras para la estación de monitoreo móvil y 1015 en el caso de la estación fija, casos que tuvieron una alta disponibilidad de red, en el caso de las estaciones de monitoreo fijas restantes debido a la baja cantidad de resultados obtenidos no pudieron ser sujetas a crear modelos predictivos con el modelo LSTM, sin embargo el sistema cumple en todos los casos con los objetivos específicos en condiciones que pueden ser controladas, como es el caso de la disponibilidad de la red.

## 8.2 Objetivo General.

*Implementar un sistema de monitoreo de calidad de aire móvil que sea escalable, y permita reducir costos que den como resultado un espectro mucho más amplio del comportamiento de los contaminantes criterio con respecto al nivel calle de forma periódica (24 veces al día) para la estación fija y hasta 2 tomas de muestras por día para la estación de monitoreo móvil en un área delimitada, y que con base en los datos*

*recabados se creen modelos predictivos mediante técnicas de Machine Learning que logren medir la confiabilidad de los datos recabados y la confiabilidad del sistema en el presente proyecto.*

Al finalizar el desarrollo del proyecto propuesto para el cumplimiento del objetivo antes descrito, se determinó que este ha sido cumplido satisfactoriamente.

Las estaciones de monitoreo cumplen con la función de ser mucho menos costosas que una estación de monitoreo común y comercial, si bien no existen datos disponibles al público por parte de la SEDEMA [3], de acuerdo con diversos artículos este precio ronda los 10 mil dólares [26], estas estaciones de monitoreo cuentan con diversos sensores como lo son Azufre, Ozono, Partículas Suspendidas, Monóxido de Carbono, entre otros. En contraste las estaciones de monitoreo propuestas en el presente proyecto y para propósito de pruebas fueron construidas solo con sensores de partículas suspendidas y su precio fue de 2500 pesos mexicanos para la estación WiFi y 3500 pesos mexicanos para la estación de monitoreo GSM, aunado a ello la capacidad de crear una red IoT móvil, que no es posible con el actual sistema de monitoreo de la Ciudad de México.

Las estaciones de monitoreo cumplen con el propósito de realizar lecturas a nivel de calle y en intervalos de 1 hora las 24 horas del día para la estación fija y 5 segundos por 2 periodos de recorridos al día para la estación móvil, se implementó una arquitectura en Azure que nos permite recibir el stream de datos mediante un broker, procesarlos mediante Stream Analytics y almacenarlos mediante CosmosDB, para finalmente usarlos en el entrenamiento del modelo LSTM que resultó en predicciones con un nivel de perdidas (errores en la predicción) bajo, comprobando la eficiencia del sistema y el modelo propuestos.

## Conclusiones

Uno de los retos más importantes de las últimas décadas han sido los esfuerzos por reducir el impacto ambiental que creamos como especie al contaminar nuestro aire, es un fenómeno que anualmente toma la vida de miles de personas [2] y que sin duda afecta cada vez más a las grandes ciudades, no es coincidencia que cada vez existan más organizaciones y proyectos independientes que se han unido a la causa por buscar alternativas que logren reducir los niveles de contaminación en nuestro planeta.

Es nuestra obligación como ciudadanos contribuir con la conservación de nuestro hogar, el presente proyecto busca sumarse a esta ola de esfuerzos por reducir y encontrar alternativas para controlar los niveles de emisiones en el aire de las grandes ciudades, en específico el de la Ciudad de México, y crear más eco para que nuestros líderes tomen cartas en el asunto, la literatura presentada en el Capítulo III: Marco Teórico, del presente trabajo muestra algunos de los proyectos relevantes en cuanto a las tecnologías que se buscan implementar en este proyecto.

Los conceptos del internet de las cosas y el machine learning cada vez toman más relevancia en nuestras vidas, son tecnologías que sin duda representan el futuro de nuestra sociedad, a lo largo de los últimos años hemos observado implementaciones de estas tecnologías que décadas atrás parecían muy distantes, la capacidad del IoT de crear una red que haga de una ciudad una ciudad inteligente y del machine learning que logre dentro de sus muchas aplicaciones crear predicciones precisas de una cantidad de datos impresionantes son cada vez más vividas, este proyecto busca juntar ambas tecnologías para sumarse a los esfuerzos que buscan crear alternativas en el monitoreo y control de los niveles de contaminación en las grandes ciudades y ser un futuro referente que lidere los esfuerzos para reducir los niveles de contaminación en nuestro mundo.

Un proyecto cuyo reto yace en la unión de estas tecnologías y su correcta implementación que dé como resultado una alternativa viable para el bien de la sociedad y de la humanidad, aún es necesario seguir trabajando y mejorando el proyecto, pero la importancia de crear los cimientos para que no sólo este trabajo puede tener un impacto si no que pueda servir como referencia para futuros proyectos, está hecha.

## Trabajos Futuros

Las bases y la arquitectura de cómo debe funcionar el sistema propuesto han sido presentados en este proyecto, al ser un sistema que trabaja con software y plataformas nuevas es relativamente sencillo implementar funcionalidades a este sistema que pueden mejorar su implementación, así como los alcances que este puede lograr, a continuación se describen los trabajos a futuro que pueden ser implementados.

Implementación de una interfaz de usuario, el desarrollo de una aplicación web usando como base la arquitectura en la nube permitiría al público observar los datos en tiempo real de cada una de las estaciones de monitoreo, y así mismo proporcionar datos que nos permitan tener un control sobre la red de IoT creada, así como el modelo de machine learning.

Las estaciones de monitoreo fueron puestas a prueba durante un periodo de 2 meses, sin embargo, se considera que para tener una mejor proyección sobre el sistema propuesto es recomendable realizar periodos de prueba más largos, 1 año sería el idóneo para analizar los datos de las estaciones de monitoreo con mayor detalle.

Es posible implementar seguridad en los datos mediante Blockchain, lo que nos permitiría asegurar la integridad de las estaciones de monitoreo y de los datos enviados usando redes públicas WiFi o GSM.

Alta Disponibilidad, como se describió en el Capítulo VII: Pruebas y Resultados, es posible garantizar la alta disponibilidad del sistema cuando se tiene control sobre variables como lo son la conexión a la red, en el caso de las estaciones de monitoreo fijas instaladas en la UPIITA y el CIC, se presentaron diversos problemas de disponibilidad en la conexión, lo que sugiere evitar el uso una red WiFi pública o en su defecto usar la red GSM cuando una red WiFi privada no esté disponible.

Mojaras al prototipo, las estaciones de monitoreo fueron probadas con un solo sensor de partículas suspendidas, que dio resultados satisfactorios, sin embargo, es necesaria la implementación de más sensores de diferentes contaminantes para ampliar el espectro y funcionalidad del sistema, así como buscar sensores de mayor calidad que no puedan comprometer la calidad de los datos que se esperan.

Teniendo la implementación de una aplicación web es posible consumir estos datos desde una aplicación móvil disponible para Android e iOS, lo que incrementaría el impacto del proyecto propuesto y pondría al alcance de todos los datos proporcionados por la red de estaciones de monitoreo.

## Referencias

- [1] C. NUNEZ, «Air Pollution Explained,» National Geographic, 4 Febrero 2014. [En línea]. Available: <https://www.nationalgeographic.com/environment/global-warming/pollution/>.
- [2] World Health Organization, «Health Topics,» 2020. [En línea]. Available: <https://www.who.int/health-topics/air-pollution>.
- [3] SEDEMA, «Calidad del Aire,» SEDEMA, 2018. [En línea]. Available: <http://www.data.sedema.cdmx.gob.mx/breatheLife/calidadAire.html>.
- [4] Gobierno de la Ciudad de México, «Dirección de Monitoreo Atmosférico,» [En línea]. Available: <http://www.aire.cdmx.gob.mx/default.php?opc=%27ZaBhnMI=%27>.
- [5] Environmental Defense Fund, «Mapping Air Pollution,» Environmental Defense Fund, [En línea]. Available: <https://www.edf.org/airqualitymaps>.
- [6] Macrotrends, «Mexico Metro Area Population 1950-2020,» 2020. [En línea]. Available: <https://www.macrotrends.net/cities/21853/mexico-city/population>.
- [7] S. E. Rashidy, «Orange Business,» Orange, 18 Mar 2018. [En línea]. Available: <https://www.orange-business.com/es/blogs/conectividad-demanda-iot-para-sacar-las-ciudades-inteligentes-del-papel>. [Último acceso: 24 05 2020].
- [8] V. H. Torres, «Repositorio de Tesis IPN,» 17 01 2017. [En línea]. Available: <http://tesis.ipn.mx/handle/123456789/20461>. [Último acceso: 20 12 2020].
- [9] Oracle, «Internet of Things,» Oracle, 2020. [En línea]. Available: <https://www.oracle.com/internet-of-things/what-is-iot/>.
- [1] Ericsson, «Cellular IoT in the 5G era,» Ericsson, 2020. [En línea]. Available: <https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-iot-in-the-5g-era>.

- [1] IBM, «Machine Learning,» IBM Cloud Education, 15 Julio 2020. [En línea].
- 1] Available: <https://www.ibm.com/cloud/learn/machine-learning>.
  
- [1] IBM, «Cloud Storage,» IBM Cloud Education, 2020. [En línea]. Available:
- 2] <https://www.ibm.com/cloud/learn/cloud-storage>.
  
- [1] Aeroqual, «Urban Air Pollution Monitoring,» Aeroqual, 2013. [En línea].
- 3] Available: <https://www.aeroqual.com/outdoor-air-quality-monitors/urban-air-monitoring>.
  
- [1] H. L. y. G. L. Thanongsak Xayasouk, «Air Pollution Prediction Using Long
- 4] Short-Term,» *Multidisciplinary Digital Publishing Institute*, vol. 2570, nº 12, p. 17, 2020.
  
- [1] S. H. y. J. Schmidhuber, «Long Short Term Memory,» *Neural Computation* , vol. 5] 9, nº 8, p. 32, 1997.
  
- [1] Arduino LLC, «WiFiNINA library,» Arduino LLC, 24 12 2019. [En línea].
- 6] Available: <https://www.arduino.cc/en/Reference/WiFiNINA>. [Último acceso: 24 7 2021].
  
- [1] Arduino LLC, «ArduinoECCX08,» Arduino LLC, 2018. [En línea]. Available:
- 7] <https://github.com/arduino-libraries/ArduinoECCX08>. [Último acceso: 26 8 2021].
  
- [1] Mariusz Kacki, «PMS Library,» Arduino, 26 Jun 2018. [En línea]. Available:
- 8] <https://www.arduino.cc/reference/en/libraries/pms-library/>. [Último acceso: 25 7 2021].
  
- [1] Benoît Blanchon, «Arduino JSON,» Benoît Blanchon, 26 8 2021. [En línea].
- 9] Available: <https://arduinojson.org/>. [Último acceso: 26 8 2021].
  
- [2] Arduino, «Arduino,» Arduino, 2021. [En línea]. Available:
- 0] <https://www.arduino.cc/en/Reference/MKRGSM>. [Último acceso: 26 8 2021].
  
- [2] Hologram, «Hologram io,» Hologram io, 2021. [En línea]. Available:
- 1] <https://www.hologram.io/>. [Último acceso: 26 8 2021].

- [2] Arduino, «Adding more Serial Interfaces to SAMD microcontrollers (SERCOM),»
- 2] Arduino LLC, 2021. [En línea]. Available:  
<https://www.arduino.cc/en/Tutorial/SamdSercom>. [Último acceso: 26 8 2021].
- [2] mikalhart, «TinyGPSPlus,» TinyGPSPlus, 2019. [En línea]. Available:  
3] <https://github.com/mikalhart/TinyGPSPlus>. [Último acceso: 26 8 2021].
- [2] Microsoft, «Use Data migration tool to migrate your data to Azure Cosmos DB,»
- 4] Microsoft, 2021. [En línea]. Available: <https://docs.microsoft.com/en-us/azure/cosmos-db/import-data#export-to-json-file>. [Último acceso: 27 6 2021].
- [2] Anaconda org, «Anaconda,» Anaconda org, 2021. [En línea]. Available:  
5] <https://anaconda.org/>. [Último acceso: 27 6 2021].
- [2] aqicn, «Air Quality Monitoring Stations,» aqicn, 21 Jun 2021. [En línea].  
6] Available: <https://aqicn.org/products/monitoring-stations/#:~:text=One%20can%20expect%20arround%20%2410,the%20quotation%20from%20other%20vendors..> [Último acceso: 3 Dec 2021].
- [2] L. Horwitz, «Can smart city infrastructure alleviate the strain of city growth?,»
- 7] CISCO, [En línea]. Available: <https://www.cisco.com/c/en/us/solutions/internet-of-things/smart-city-infrastructure.html>.
- [2] Breathe London, «Breathe London,» Enviromental Defense Fund Europe, 2018.  
8] [En línea]. Available: <https://www.breathelondon.org/about/>.
- [2] Y. M. A. H.-L. C. y. J. R. C. J. Yun-Chia Liang, «Machine Learning-Based  
9] Prediction of Air Quality,» *applioed sciences*, vol. 9151, p. 17, 2020.
- [3] O. y. Hakyll, «Undestanding LSTM,» colah, 27 Agosto 2015. [En línea].  
0] Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Último acceso: 27 6 2021].