

Sistema de Monitoreo y Generación de Alertas de Prevención Contra Inundaciones con IOT

Trabajo de grado para optar al título de
Ingeniero Electrónico

Estudiantes:

Sebastián Sánchez Vargas Código:20172005158
Jose Gabriel Ramirez Bernal Código:20182005158

Director:

Leonardo Plazas Nossa, Ing.

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería Electrónica
2024

Índice general

1. Introducción **5**

2. Planteamiento del problema **6**

3. Justificación **7**

4. Objetivos **8**
4.1. Objetivo General 8
4.2. Objetivos Específicos 8

5. Marco teórico **9**
5.1. Inundación 9
5.1. Variabilidad y Cambio Climático 13
5.2. Definicion de llanura de inundación 14
5.3. ¿Qué es una ribera, un cauce y una faja marginal? 14
5.4. Clases de alerta 15
5.5. CONDICIONES TÉCNICAS PARA LA ELABORACIÓN DE ESTUDIOS BÁSICOS 18
5.6. Sistemas de alerta temprana 19
5.7. Elementos de un SAT 20

6. Estado del arte **23**

7. Plan de trabajo **29**

8. Desarrollo **32**

9. Resultados **87**

10. Conclusiones **96**

11. Trabajo futuro **97**

12. Referencias **98**

13. Anexos **100**
13.1. Código del sistema de monitoreo 100
13.2. Código del sistema de alerta 106
13.3. Código de la generación de la gráfica del historial del nivel del agua 108
13.4. Código de la generación de la gráfica de la cantidad de alertas activadas 111
13.5. Código de la generación del mapa 113
13.6. Código de la generación de los datos meteorológicos 116
13.7. Códigos en bloques para la aplicación 119

Índice de figuras

5.1. HC-SR04, tomado de [17] 21
5.2. ESP32, tomado de [18] 22

6.1. Dispositivo implementado en la práctica, tomado de [20] 23
6.2. Modelo 3D de un nodo fijo RiverCore., tomado de [21] 24
6.3. Diagrama de bloques del prototipo propuesto, tomado de [22] 25
6.4. Información recopilada vizualizada, tomado de [22] 25
6.5. El prototipo del sistema y la configuración del experimento, tomado de [23] . 26
6.6. Diagrama de bloques, tomado de [24] 27
6.7. Prototipo final, tomado de [24] 27
6.8. Diagrama de la arquitectura del sistema, tomado de [25] 28
6.9. Prototipo funcionando, tomado de [25] 28
7.1 Diagrama de bloques del proyecto propuesto. 29
7.2 Esquemático del funcionamiento del proyecto propuesto. 29
8.2.1: Medidas para la base de la estructura. 35
8.2.2: Medidas de los rectángulos de madera. 35
8.2.3: Propuesta de maqueta (fase inicial). 36
8.4.1: Etapa inicial de la estructura. 37
8.6.1: Circuito accionador de la sirena. 38
8.7.1: Estructura adecuada para realizar las primeras pruebas. 42
8.7.2: Estructura con un nivel de agua de aproximadamente 9 cm. 42
8.7.3: Estructura con un nivel de agua de aproximadamente 12 cm. 43
8.7.4: Estructura con un nivel de agua de aproximadamente 12.5 cm. 43
8.7.5: Estructura con un nivel de agua de aproximadamente 14.4 cm. 44
8.7.6: Estructura con un nivel de agua de aproximadamente 16.7 cm. 44
8.7.7: Activación del sensor de lluvia e indicación de intensidad de la misma. 45
8.7.8: Maqueta con su máxima capacidad de llenado. 45
8.9.1. Modulo celular SIM800L GSM GPRS con antena- Shield SIM card para Arduino. 46
8.10.1. Panel Solar 5v 300 mA Fotovoltaico. 49
8.10.2. Cargador Batería De Litio Lipo 1a Tipo C 5v Tp4056. 50
8.10.3. Baterías recargables li-ion de a 1500 mAh. 50
8.10.4. Low-dropout regulator MCP1700-3302E 51
8.10.5. Diagrama de bloques funcional del MCP1700. 51
8.10.6. Esquemático del sistema de monitoreo. 52
8.10.7. Vista del circuito en 3D. 53
8.10.8. Vista de los caminos del circuito en 3D. 53
8.10.9. Vista del circuito en 3D desde otro ángulo. 54
8.10.10. Vista superior del circuito impreso. 54
8.10.11. Vista de los caminos del circuito impreso. 55
8.10.12. Circuito ubicado en el exterior para hacer las pruebas de autonomía. 55
8.10.13. Comportamiento de la batería desde la noche hasta el mediodía del otro día. 56

8.10.14. Comportamiento de la batería en los últimos 2 días.....57
8.10.15. Pared frontal de la caja. 58
8.10.16. Pared con agujero para los cables de los paneles solares. 58
8.10.17. Base de la caja. 59
8.10.18. Una de las paredes de la caja. 59
8.10.19. Pared lateral izquierda con abertura para el sensor de lluvia. 59
8.10.20. Diseño en 3D de la caja (vista inferior). 60
8.10.21. Diseño en 3D de la caja (vista superior). 60
8.10.22. Caja en físico (vista superior). 61
8.10.23. Caja en físico (vista superior desde otro ángulo). 61
8.10.24. Caja en físico (vista inferior). 62
8.10.25. Caja en físico (vista del interior). 62
8.11.1. Panel Solar 12v 125 mA Fotovoltaico. 63
8.11.2. Modulo cargador de pilas. 63
8.11.3. Regulador lineal de “Low-Dropout”- LM1117T-3.3. 64
8.11.4. Transistor Npn TIP31C. 64
8.11.5. Sirena 12 V 10W (HPS102) 65
8.11.6. Esquemático del sistema de alarma. 65
8.11.7. Vista del circuito en la herramienta de diseño de PCB. 66
8.11.8. Vista del circuito en 3D. 67
8.11.9. Vista de los caminos del circuito en 3D. 67
8.11.10. Vista de los caminos del circuito impreso. 68
8.11.11. Vista superior del circuito impreso. 68
8.11.12. Pared frontal, derecha y superior de la caja del sistema de alarma. 69
8.11.13. Pared izquierda de la caja del sistema de alarma. 69
8.11.14. Pared posterior de la caja del sistema de alarma. 69
8.11.15. Base de la caja del sistema de alarma. 70
8.11.16. Diseño 3D de la caja visión lateral izquierda. 70
8.11.17. Caja en físico (vista desde el lateral izquierdo). 71
8.11.18. Caja en físico (vista superior sin la tapa). 71
8.12.1. Realtime Database del proyecto. 74
8.13.1. Pantalla de carga de la aplicación. 76
8.13.2. Configuración de los bloques para el funcionamiento de la pantalla de carga. 77
8.13.3. Pantalla de inicio de la aplicación (vista de diseño). 77
8.13.4. Configuración de los bloques para el funcionamiento de la pantalla de inicio. 78
8.13.5. Configuración de los bloques para el funcionamiento de la pantalla de inicio. 78
8.13.6. Configuración de los bloques para el funcionamiento de la pantalla de inicio. 79
8.13.7. Grafica del historial de niveles de agua con valores de algunas pruebas. 80
8.13.8. Grafica del historial de niveles de agua con valores de algunas pruebas. 82
8.13.9. Visualización del mapa y el área de afectación. 84
8.13.10. Visualización del mapa y el área de afectación. 85
8.13.11. Configuración de los bloques para el cambio de pantalla. 86
9.1: Estructura adecuada para comprobar el funcionamiento del sistema completo. 87
9.2: Sistema de alarma para comprobar el funcionamiento del sistema completo. 88
9.3: Aplicativo en funcionamiento mostrando la información del nivel del rio. 88

9.4: Aplicativo en funcionamiento mostrando la información de las gráficas. 89
9.5: Aplicativo en funcionamiento mostrando la información del mapa. 90
9.6: Aplicativo en funcionamiento mostrando la información de los datos meteorológicos. 90
9.7: Estructura con el nivel del rio a 12.5 cm. 91
9.8: Aplicativo en funcionamiento mostrando la información de la alerta amarilla activada.92
9.9: Estructura con el nivel del rio a 15 cm. 93
9.10: Aplicativo en funcionamiento mostrando la información de la alerta naranja activada.93
9.11: Estructura con el nivel del rio a 16.5cm. 94
9.12: Aplicativo en funcionamiento mostrando la información de la alerta roja activada. 94
9.13: Aplicativo en funcionamiento mostrando la información del área de afectación en el mapa. 95

Capítulo 1

Introducción

Las inundaciones constituyen uno de los fenómenos naturales más devastadores y frecuentes en Colombia, un país caracterizado por su diversidad geográfica y climática. Este tipo de eventos naturales no solo representa una amenaza para la vida humana, sino que también conlleva significativas pérdidas económicas y daños ambientales [1]. A lo largo de las décadas, Colombia ha enfrentado numerosos episodios de inundaciones, los cuales han evidenciado la vulnerabilidad de su territorio frente a estas catástrofes naturales. La complejidad de este fenómeno en Colombia se debe a varios factores, incluyendo su topografía variada, la presencia de múltiples cuencas hidrográficas, y un clima que favorece precipitaciones intensas y prolongadas. La problemática de las inundaciones en Colombia se ve agravada por el cambio climático, que incrementa la frecuencia e intensidad de las lluvias en ciertas regiones, así como por la intervención humana en los ecosistemas naturales [2]. La deforestación, la expansión urbana descontrolada en zonas de riesgo, la mala gestión de los recursos hídricos y el inadecuado manejo de los residuos, son algunos de los factores que han exacerbado la vulnerabilidad del país ante las inundaciones. Estos eventos no solo afectan la infraestructura y la economía, sino que también tienen un impacto profundo en el tejido social y en la salud pública, desplazando comunidades enteras y dejando tras de sí un legado de pobreza y enfermedad [3]. En respuesta a estos desafíos, Colombia ha iniciado la implementación de diversas estrategias de gestión del riesgo y adaptación al cambio climático. Estas medidas incluyen desde la construcción de infraestructuras resilientes hasta la promoción de prácticas agrícolas y urbanísticas sostenibles. Sin embargo, la efectividad de estas iniciativas depende en gran medida de la cooperación entre los gobiernos locales, nacionales, la comunidad internacional, y los propios ciudadanos [4].

En Colombia contamos con diferentes entidades encargadas de monitorear, predecir y alertara las comunidades acerca de posibles desastres que puedan ocurrir en diferentes sectores. Pe- ro en varios casos aislados donde puede llegar a suceder un desastre de manera muy rápida e inesperada estos sistemas no alcanzan a alertar a las comunidades, por lo que las comu- nidades necesitan sistemas de alerta temprana (SAT) los cuales actúen en estos casos para que avisen a la comunidad con el suficiente tiempo y esta actúe de manera oportuna y se usede igual forma para salvaguardar la vida de las personas [5].

En este documento lo que se propuso hacer es proponer un anteproyecto el cual está enfocado en el análisis, diseño y simulación de un sistema de alerta temprana el cual cumpla con los estandares establicidos por el UNGRD, cuente con una aplicación movil y que haga gran uso de todo lo que se ha aprendido a lo largo de la carrera tanto de elementos actua- dores como de redes móviles con los cuales podamos llevar a cabo el proyecto de manera satisfactoria [6]

Capítulo 2

Planteamiento del problema

En Colombia se tienen catástrofes debido a las diversas condiciones climáticas que son comunes y que aumentan debido a la posición geo estacionaria y el relieve del terreno, entre ellas se encuentran las inundaciones, respecto a estas, adicionalmente, en Colombia es difícil pronosticarlas o ubicar el sitio exacto dónde se puedan generar, a pesar de que se cuentan con varias entidades gubernamentales y privadas, entre la cuales se encuentran el IDEAM, DIMAR, Corporación OSSO y SCG, las cuales ayudan a predecir estos desastres, informar a la poblacion e intentan minimizar los daños que puedan ocurrir [7].

Sin embargo, aun teniendo estas entidades que promueven la prevención de catástrofes, como se mencionó anteriormente, se presenta en algunas regiones, cambios muy rápidos en los caudales de agua de los ríos y donde los sistemas de monitoreo no alertan a la comunidad con el suficiente tiempo y la misma comunidad deba tener que alertar a las entidades gubernamentales, la pregunta que nos hacemos es: ¿Es posible diseñar una guía para un sistema de alerta temprana (SAT) el cual cuente con elementos característicos como son: bajo en costo, comunicación rápida y efectiva, siendo esta un apoyo a la comunidad? [8]

Capítulo 3

Justificación

En Colombia aún se presentan catástrofes en diversas comunidades, apartadas o en las que no se tiene un monitoreo en tiempo real y no se les podría comunicar sobre alguna eventualidad. Es por lo que estas comunidades deben gestionar o construir sus propios sistemas de alerta temprana y capacitar a algunos miembros de la comunidad para que tomen las medidas necesarias y así poder minimizar los daños que se pudieran presentar.

Pensamos que en el área de electrónica se trabaja en varias ocasiones con sensores, microcontroladores y varios elementos los cuales son útiles para poder determinar cambios con diferentes variables, se puede diseñar un sistema de monitoreo y alerta para una zona con posibilidades de inundación el cual debería incluir una guía para que sea usado por una comunidad en particular, el cual cuente con una serie de requerimientos mínimos los cuales aseguren su veracidad y que la comunidad pueda ser participe por medio una aplicación.

Capítulo 4

Objetivos

4.1. Objetivo General

Diseñar un sistema de monitoreo y generación de alertas de prevención contra inundaciones con IoT.

4.2. Objetivos Específicos

- Determinar las condiciones iniciales en las que se aplicará el sistema de alerta de riesgos.
- Implementación de la estructura en la que se van a realizar las mediciones.
- Determinar los sensores más adecuados considerando las limitaciones de la estructura y el presupuesto para el proyecto.
- Analizar y determinar el sistema de procesamiento de datos que cumpla con las condiciones necesarias para una correcta implementación.
- Diseñar e implementar una aplicación movil capaz de alertar en cualquier momento a cualquier persona.
- Simular con la estructura las condiciones especificadas con el sistema de monitoreo.

Capítulo 5

Marco teórico

5.1. Inundación

El IDEAM, dentro del Comité Nacional de Conocimiento para la Gestión del Riesgo de Desastres, define lo siguinete: inundación es la acumulación temporal de agua fuera de los cauces y áreas de reserva hídrica de las redes de drenaje (naturales y construidas). Se presentan debido a que los cauces de escorrentía superan la capacidad de retención e infiltración del suelo y/o capacidad de transporte de los canales. Las inundaciones son eventos propios y periódicos de la dinámica natural de las cuencas hidrográficas. [9]

La precipitación es un elemento clave en la ocurrencia de inundaciones. El origen del agua en los ríos se debe principalmente a la lluvia que fluye desde las partes altas de las cuencas, influenciada por factores como la topografía, la cobertura vegetal y las prácticas de uso del suelo. Asimismo, el aporte de agua a los ríos se ve complementado por corrientes subterráneas, que dependen de las propiedades del suelo y los niveles de las aguas subterraneas. En nuestro país, muchas ciudades y centros administrativos se encuentran cerca de ríos y lagos para beneficiarse del acceso al agua y a las rutas de transporte por agua, pero esta cercanía también los expone a posibles inundaciones.

Dependiendo de las causas que generan una inundación, estas se pueden clasificar en general en:

- Inundación por desbordamiento: Ocurre cuando el agua de un cuerpo hídrico, como un río, lago o canal, excede su capacidad normal o sus límites y se extiende sobre las áreas adyacentes que normalmente están secas. Este tipo de inundación suele suceder cuando hay un exceso de precipitaciones, lo que lleva a que el cuerpo de agua reciba un volumen de agua mayor al que puede contener dentro de sus márgenes naturales o artificiales. Como resultado, el agua se derrama e inunda las áreas circundantes, que pueden incluir zonas residenciales, agrícolas y urbanas. Las inundaciones por desbordamiento pueden

causar daños significativos a la infraestructura, la agricultura, y representan un riesgo para la seguridad de las personas en las áreas afectadas.

- Inundación por encharcamiento: También conocida como inundación pluvial o inundación por acumulación, ocurre cuando la lluvia cae más rápido de lo que puede ser absorbida por el suelo o drenada por el sistema de drenaje existente. Este tipo de inundación es común en áreas donde el suelo está ya saturado, donde la capacidad de drenaje es insuficiente o en áreas urbanas con superficies impermeables como concreto y asfalto, que impiden la absorción natural del agua.

  Las inundaciones por encharcamiento pueden suceder rápidamente, a menudo durante o poco después de una tormenta intensa. Aunque generalmente son menos devastadoras que las inundaciones por desbordamiento de ríos o mares, pueden causar daños significativos a propiedades, infraestructuras urbanas y pueden afectar la movilidad y seguridad de las personas en las áreas urbanas y suburbanas. Estas inundaciones son más comunes en zonas urbanas debido a la gran cantidad de superficies impermeables y a menudo están relacionadas con problemas en los sistemas de alcantarillado y drenaje.

- Inundación por reflujo: También conocida como inundación por remanso, se produce cuando el agua fluye hacia atrás desde su curso normal debido a un obstáculo o una condición que impide su flujo normal. Este tipo de inundación es común en áreas donde los sistemas de drenaje o los cursos de agua están bloqueados, o donde hay una confluencia de cuerpos de agua, como la unión de un río con otro o con el mar.

  El reflujo puede ser causado por varios factores, como mareas altas, que empujan el agua de mar hacia arriba en la desembocadura de un r´ıo, impidiendo que el agua del río fluya normalmente hacia el mar. También puede ocurrir cuando un río se desborda y el exceso de agua no encuentra una vía de escape rápida, lo que provoca que fluya hacia atrás o hacia zonas normalmente no inundadas.

  Las inundaciones por reflujo pueden causar daños importantes en áreas donde el drenaje normal se ve comprometido. Este tipo de inundación es particularmente problemático en áreas bajas y planas cerca de la costa o en las desembocaduras de los ríos, donde la influencia de las mareas puede ser significativa.

- Inundación por rompimiento de jarillones o presas: Ocurre cuando estas estructuras, construidas para contener el agua, fallan o se rompen, liberando grandes volúmenes de agua de manera súbita y sin control. Los jarillones, también conocidos como diques o terraplenes, y las presas, son estructuras diseñadas para regular el flujo de agua, contener inundaciones o almacenar agua para diversos usos como riego, consumo humano o generación de energía.

  La ruptura de estas estructuras puede deberse a varios factores, como mal mantenimiento, diseño deficiente, condiciones climáticas extremas que provocan un aumento

del nivel del agua más allá de la capacidad de la presa o del jarillón, o eventos geológicos como terremotos.

Cuando un jarillón o una presa se rompe, el agua acumulada se libera repentinamente, inundando las áreas aguas abajo. Estas inundaciones pueden ser extremadamente peligrosas debido a la velocidad y cantidad de agua liberada, lo que puede resultar en inundaciones catastróficas con poco tiempo de advertencia, causando daños significativos a las comunidades, la infraestructura y el medio ambiente en las áreas afectadas. La preparación y el mantenimiento adecuados de estas estructuras son cruciales para prevenir tales desastres.

- Inundación por marejadas: También conocida como inundación costera, ocurre cuando el nivel del mar aumenta significativamente por encima de lo normal, inundando las áreas costeras. Este fenómeno es usualmente causado por fuertes vientos de tormentas, huracanes, o ciclones que empujan el agua del mar hacia la costa. Además, las marejadas pueden verse agravadas por la marea alta, aumentando aún más el riesgo y el impacto de la inundación.

  Durante una marejada, el agua del mar puede invadir áreas bajas y planas a lo largo de la costa, causando inundaciones en zonas habitadas, dañando propiedades y infraestructura, y poniendo en peligro la vida de las personas. La fuerza de las marejadas puede ser devastadora, arrastrando escombros, erosionando playas y costas, y afectando ecosistemas costeros.

  Las marejadas son particularmente peligrosas porque la combinación del nivel elevado del mar y las olas fuertes puede causar un impacto mucho más severo que una elevación del nivel del mar sin olas. Las comunidades costeras son especialmente vulnerables a este tipo de inundaciones, y su frecuencia y severidad pueden aumentar debido a los efectos del cambio climático y el aumento del nivel del mar [10].

Tambien se establece, que de acuerdo con las caracter´ısticas de las cuencas y la rapidez con la cual se producen los desbordamientos, las inundaciones se pueden clasificar en:

- Inundación rápida o avenida torrencial: Es un tipo de inundación que ocurre de manera muy súbita y con gran intensidad, usualmente como resultado de fuertes lluvias en un corto periodo de tiempo, especialmente en áreas con terrenos empinados o en cuencas pequeñas. Este tipo de inundación también puede ser provocada por la ruptura de una presa o un dique, o por el deshielo rápido.

  Las avenidas torrenciales se caracterizan por la rápida subida del nivel del agua, a menudo en cuestión de minutos o horas, lo que deja poco tiempo para advertencias o evacuaciones. El agua fluye con gran fuerza, arrastrando lodo, rocas, escombros y otros materiales, lo que puede causar daños severos a estructuras, carreteras y puentes, así

como riesgos significativos para la vida humana.

Este tipo de inundación es particularmente peligroso en áreas montañosas o en regiones áridas y semiáridas donde la tierra, a menudo seca y dura, no puede absorber rápidamente el agua de la lluvia, resultando en un rápido escurrimiento superficial. Las avenidas torrenciales también pueden ocurrir en áreas urbanas, donde la gran cantidad de superficies impermeables contribuye a un rápido escurrimiento del agua de lluvia.

- Inundación lenta: También conocida como inundación de llanura de inundación o inundación fluvial, es un tipo de inundación que se desarrolla gradualmente a lo largo de varios días o incluso semanas. Este fenómeno ocurre generalmente cuando los ríos desbordan sus cauces debido a un prolongado periodo de lluvias intensas o al deshielo en las regiones montañosas, lo que resulta en un incremento lento y sostenido del nivel del agua.

  A diferencia de las inundaciones rápidas o torrenciales, las inundaciones lentas dan más tiempo para la preparación y evacuación, pero pueden cubrir extensas áreas geográficas, afectando grandes extensiones de terreno. Este tipo de inundación es especialmente común en regiones con grandes llanuras de inundación y ríos de caudal lento.

  Las inundaciones lentas pueden tener impactos significativos a largo plazo en las comunidades y en la agricultura, ya que el agua puede permanecer estancada durante un tiempo prolongado, afectando los cultivos, las infraestructuras y las viviendas. Además, pueden causar problemas de salud pública debido a la contaminación del agua y el aumento de las poblaciones de insectos vectores de enfermedades. Aunque menos dramáticas que las inundaciones repentinas, las inundaciones lentas requieren una gestión cuidadosa y una planificación a largo plazo para mitigar sus efectos [12].

Las inundaciones suelen ser vistas como eventos negativos debido a sus impactos inmediatos, como daños a la infraestructura, pérdidas económicas, desplazamiento de personas y riesgos para la salud humana. Sin embargo, desde una perspectiva ecológica y a largo plazo, las inundaciones pueden tener también efectos beneficiosos:

1. Renovación de Ecosistemas: Las inundaciones pueden aportar nutrientes esenciales a los ecosistemas, especialmente en llanuras aluviales y humedales, mejorando la fertilidad del suelo. Esto es crucial para mantener la biodiversidad en estos ecosistemas.
2. Recarga de Acuíferos: Las inundaciones pueden contribuir a la recarga de acu´ıferos subterráneos, lo que es vital para el suministro de agua dulce.
3. Control de Especies: Las inundaciones pueden ayudar a controlar la población de ciertas especies que podrían volverse problemáticas si no se controlan naturalmente, como ciertos tipos de insectos.
4. Formación de Suelo: Las inundaciones depositan sedimentos que pueden formar suelo fértil, especialmente en las áreas cercanas a los ríos.

5. Mantenimiento de Humedales: Los humedales dependen de las inundaciones periódicas para mantener sus características ecológicas únicas, que son esenciales para muchas especies de plantas y animales.

A pesar de estos beneficios, es importante tener en cuenta que las actividades humanas, como la urbanización y la modificación de los cursos de agua, han aumentado la frecuencia y la severidad de las inundaciones, exacerbando sus impactos negativos. Por lo tanto, es crucial gestionar y planificar cuidadosamente el uso del suelo y las infraestructuras para mitigar los riesgos de inundación y aprovechar sus beneficios potenciales de manera sostenible [10].

5.2. Variabilidad y Cambio Climático

Mientras que los cambios climáticos se miden en promedios durante periodos específicos, como entre 40 y 50 años, la variabilidad climática incluye fases extremas como periodos más lluviosos o secos de lo usual. Estas fases extremas son cruciales para entender y regular fenomenos como sequías y precipitaciones inusuales. Por ejemplo, la variación en la cantidad de lluvia de un año a otro puede llevar a inundaciones en años de máxima lluvia y a sequías en años de mínima precipitación.

Estudios sobre la variabilidad climática intraestacional se han realizado globalmente y específicamente en América del Sur, incluyendo análisis detallados para Colombia. Según estos estudios, las fases extremas de las fluctuaciones climáticas dentro de una estación pueden influir en la frecuencia e intensidad de eventos extremos como lluvias intensas y desbordamientos, así como en la severidad temporal de fenómenos como sequías e inundaciones, impactando significativamente en los aspectos socioeconómicos del país.

Respecto a los eventos extremos y su diferencia con el clima promedio, se destaca que los extremos están vinculados a procesos físicos distintos a los que determinan los promedios a largo plazo. Los cambios en la precipitación promedio están mayormente ligados a cambios en la circulacion atmosférica, mientras que los eventos extremos son más sensibles a las condiciones termodinámicas y específicas del momento. Es crucial, entonces, contrastar las tendencias y proyecciones en los promedios climáticos con los de eventos raros. Los eventos extremos ocurren bajo ciertas precondiciones, como la necesidad de un transporte de humedad elevado, altas temperaturas y una atmósfera inestable. Estas condiciones son poco frecuentes, pero bajo el cambio climático, su ocurrencia puede aumentar, especialmente la de temperaturas más altas. Esto es relevante para la precipitación, ya que un incremento en la temperatura del aire aumenta su capacidad de retener humedad, lo que podría resultar en lluvias más intensas.

La localización exacta de la precipitación extrema es incierta, ya que depende de las condiciones locales y a veces de la topografía. En resumen, los eventos de precipitación extrema pueden mostrar cambios más significativos que los promedios de lluvia, y en un mundo más cálido, la probabilidad de precipitaciones intensas aumenta, lo que podría incrementar el riesgo de inundaciones. Sin embargo, en áreas donde disminuye la precipitación, esta tendencia podría ser menos pronunciada. [11]

5.3. Definición de llanura de inundación

Las crecidas de los ríos son fenómenos habituales que se presentan con una frecuencia aproximada de cada dos años y un tercio. Estos sucesos se originan cuando las precipitaciones intensas o prolongadas superan la capacidad del suelo para absorber agua y la de los ríos y arroyos para contenerla, provocando así el desbordamiento de estos cuerpos de agua e inundaciones en las zonas adyacentes.

Las áreas propensas a estas inundaciones, conocidas como zonas de inundación, se caracterizan por estar situadas cerca de ríos y están frecuentemente sujetas a estas situaciones. Dependiendo del enfoque, estas zonas pueden ser vistas como terrenos planos junto a ríos, áreas formadas por sedimentos fluviales, o regiones que regularmente experimentan inundaciones.

En cuanto a su medición, las inundaciones se describen a menudo en términos de probabilidades. Un término común es la ïnundación de 100 años", que se refiere a una probabilidad del 1 % de que ocurra una inundación de cierta magnitud en un año dado. Este concepto es clave para identificar áreas de alto riesgo y para la planificación de medidas de prevención y mitigación de inundaciones. Los planificadores pueden usar diferentes escalas de tiempo, como 5, 20, 50 o 500 años, para evaluar y manejar los riesgos asociados.

Los factores que influyen en la frecuencia de las inundaciones incluyen las condiciones climáticas, la composición del suelo en las riberas del río y la inclinación del cauce. En regiones con lluvias estacionales o donde el deshielo anual es común, las inundaciones pueden ser un evento casi anual. Por otro lado, en áreas sin largos periodos de congelación, las inundaciones tienden a ocurrir durante las temporadas de mayor lluvia, mientras que, en zonas de deshielo, suelen darse en la primavera o a principios del verano. [12]

5.4. ¿Qué es una ribera, un cauce y una faja marginal?

Numerosas comunidades a lo largo del país han sufrido graves perjuicios debido a inundaciones, la activacion de quebradas y la aparición de huaicos a causa de fuertes lluvias. Estas comunidades afectadas comparten la característica de haber establecido sus hogares en áreas adyacentes a ríos o dentro de lechos fluviales que, tras recibir un volumen excesivo de agua, terminaron desbordándose, así como en quebradas que se reactivaron debido a las precipitaciones.

Es crucial subrayar la importancia de evitar el asentamiento en zonas cercanas a las orillas de los ríos o dentro de los cauces de quebradas, que aunque pueden permanecer secas durante periodos, eventualmente se reactivan. Según su intensidad, esto puede causar enormes daños al llevarse consigo todo a su paso.

La ribera define la franja que bordea ríos, arroyos, torrentes, lagos y lagunas, situada entre el nivel más bajo y el más alto que el agua puede alcanzar durante sus crecidas.

Por otro lado, el cauce representa el espacio por donde fluyen las aguas de un río en su punto más alto, típicamente durante la temporada de lluvias. Durante los períodos de sequía o cuando el río cambia su curso, se liberan áreas que son aprovechadas por el ser humano para la agricultura o para establecer asentamientos.

La faja marginal, ubicada justo por encima de la ribera, es una zona donde naturalmente se desarrollan bosques y áreas verdes, las cuales son frecuentemente utilizadas por las personas para la vivienda y la agricultura.

Hay numerosos asentamientos humanos, desde poblados hasta urbanizaciones, que se han establecido en estas zonas ribereñas, fajas marginales, y cauces de ríos y quebradas, que inicialmente estaban secos y disponibles. Sin embargo, esta ocupación no ha tomado en cuenta que, con el tiempo, los cursos de agua tienden a recuperar su cauce natural. [13]

5.5. Clases de alerta

Con el propósito de ilustrar a los Comités para la Prevención y Atención de Desastres sobre la declaratoria de las alertas frente a la temporada invernal, la Dirección deGestión del Riesgo, se permite hacer las siguientes precisiones:

Las alertas son medidas de pronóstico y preparación, relacionadas con dos aspectos: la información previa que existe sobre la evolucion de un fenómeno, y las acciones y disposiciones que deben ser asumidas por los Comités para la Prevención y Atención de Desastres para enfrentar la situación que se prevé.

La responsabilidad directa para declarar los diferentes grados de alerta recae sobre los Comités Locales y Regionales, dependiendo del ámbito de la situación, con base en la información técnica suministrada por una entidad competente, para el caso, por el Instituto de Hidrología, Meteorología y Estudios Ambientales-IDEAM, y el comportamiento local, que esta supeditado a las condiciones y el nivel de protección y mitigación focal. Esto implica, la necesidad de contar con preparativos para la atención de emergencias por parte de los Comités, entre otros, mecanismos de alarma, información, evacuación, alojamiento temporal, elementos básicos, así como recursos económicos, según lo establece el **Decreto Ley 919 de 1989.**

Las alertas hidrometeorologicas tienen la siguiente connotación:

Alerta Amarilla

Se declara cuando la persistencia e intensidad de las lluvias puede ocasionar desbordamiento de los ríos en los próximos días o semanas.

Las acciones que implica la declaratoria de la alerta amarilla son las siguientes:

Convocar al Comité para la Prevención y Atención de desastres.
Ubicar los puntos críticos y definir los mecanismos de vigilancia, alerta máxima y evacuación, con base en los censos y mapas de riesgo.
Realizar un inventario de recursos humanos, técnicos, económicos, en equipos, en instalaciones e insumos de emergencia.

Alerta Naranja

Se declara cuando la tendencia ascendente de los niveles de los ríos y la persistencia de las lluvias indican la posibilidad de que se presenten desbordamientos en las próximas horas.

Las acciones que implica la declaratoria de la alerta naranja son las siguientes:

Preparar los operativos para una posible evacuación.
Informar a la comunidad sobre los sistemas de aviso en caso de emergencia.
Establecer alistamiento de equipos y personal.
Coordinar alojamiento temporal.
Revisar planes de emergencia, incluyendo las actividades en salud, transporte, remoción de escombros, adecuación vial.

Alerta Roja

Se declara cuando el nivel de los ríos alcanza alturas criticas que hacen inminente el desbordamiento, o cuando ya se ha iniciado la inundación.

Las acciones que implica la declaratoria de la alerta roja son las siguientes:

Activar las alarmas preestablecidas.
Evacuar y asegurar a la población afectada.
Movilizar los operativos según los planes de emergencia.
Atender a la población afectada en sus necesidades básicas.

Finalmente la Dirección General para la Atención y Prevención de Desastres les reitera la importancia de que existan estrechas relaciones de coordinación, interacción y comunicación permanentes sobre las medidas asumidas y situaciones de emergencia registradas, entre las diferentes instancias. [14]

Normas e indicaciones de alertas de riesgo contra inundaciones en Colombia

En Colombia, para el caso de inundaciones y deslizamientos la entidad encargada para activar

estas alertas tempranas es el Ideam, entidad que hace parte de las instituciones técnicas del SNGRD.

LA UNGRD recibe los comunicados y alertas hidrometeorológicas por parte del Ideam, y como coordinadora del SNGRD lidera la operatividad para avisar de forma oportuna e implementar los planes de contingencia y emergencia necesarios, en conjunto con los concejos departamentales y municipales de gestión del riesgo de desastres, por lo que los eventos que puedan llegar a desencadenar un desastre.

Como fue mencionado anteriormente, el Instituto de Hidrología, Meteorología y Estudios Ambientales de Colombia (Ideam), hace parte del SNGRD, y en este sentido contempla dentro de la Misión " recursos naturales y condiciones hidrometeorológicas de todo el país para la toma de decisiones de la población, autoridades, sectores económicos y sociales de Colombia y para el SNGRD en lo concerniente a la prevención y reducción del riesgo asociado a la ocurrencia de fenómenos hidrometeorológicos extremos" 17 . Así mismo, el último objetivo misional del Ideam define prestar el servicio de pronósticos, predicciones climáticas y alertas de eventos naturales que puedan tener impacto en la población y sectores productivos colombianos.

Sistema de Alerta Temprana (SAT)

El Sistema de Alerta Temprana (SAT) en Colombia está diseñado para proporcionar información oportuna y efectiva a través de instituciones identificadas, con el fin de minimizar el impacto de desastres naturales, incluyendo las inundaciones. El SAT incluye:

- **Monitoreo Continuo**: Involucra la vigilancia constante de las condiciones meteorológicas e hidrológicas, esto por medio de sensores de bajo costo.

- **Difusión de Alertas**: Las alertas se difunden a través de diferentes medios de comunicación para asegurar que la información llegue a todas las personas en riesgo, para el caso de este trabajo se cuenta con alarmas y mensajes por medio de una aplicación, y también por medio de la activación de una sirena para situaciones de emergencia.

- **Acción Coordinada**: La implementación de acciones coordinadas entre entidades gubernamentales y comunidades para responder eficazmente a las alertas.

5.6. CONDICIONES TÉCNICAS PARA LA ELABORACIÓN DE ESTUDIOS BÁSICOS

Artículo 2.2.2.1.3.2.1.2. Estudios básicos de amenaza de inundación. Para determinar las condiciones de amenaza por inundación en suelos urbanos, de expansión urbana y rural, los estudios básicos tienen las siguientes especificaciones mínimas:

1. **Área de estudio:** Las zonas en las cuales exista la posibilidad de presentarse una inundación sean aledañas o no a ríos, caños, quebradas, humedales y otros cuerpos de agua o aquellas que hagan parte de su área de influencia. En el análisis se deben considerar los casos en los que existan precedentes de mecanismos generadores de inundaciones tales como encharcamiento por lluvias intensas sobre áreas planas, encharcamiento por deficiencia de drenaje, inundaciones costeras entre otros. Los municipios o distritos con un suelo rural superior a 1.500 km2, para los cuales no exista base cartográfica e insumos a 1:25.000, podrán realizar los estudios para esta clase de suelo a escala 1:100.000 o 1:50.000. En aquellas áreas rurales donde se presenten inundaciones recurrentes, con presencia de elementos expuestos, deben realizar los estudios básicos a 1:25.000.

2. **Insumos:** Se debe utilizar como mínimo los siguientes insumos:

   2.1 Geomorfología. Identificación de las diferentes subunidades geomorfológicas asociadas a los paisajes aluviales, con especial énfasis en las geoformas correspondientes a la llanura de inundación.

   2.2 Modelo de elevación digital del terreno.

   2.3 Identificación de las zonas inundables e inundadas (registro de eventos). A partir de información de las diferentes entidades a nivel nacional, regional o local, interrelacionada con la información de la comunidad identificar cuales áreas han sufrido afectaciones por inundación y en qué fecha.

   2.4 Hidrología. Caracterización del comportamiento del régimen hidrológico en la region a la cual pertenece el municipio mediante un análisis de los eventos hidroclimaticos máximos identificando para cuales períodos de retorno se están presentando las afectaciones y las áreas afectadas para los mismos.

3. **Alcance:** Para la zonificación de la amenaza se emplean tres categorías: alta, media y baja, teniendo en cuenta el registro de eventos, la recurrencia de los mismos y la intensidad (niveles alcanzados) de la inundación. Para el suelo urbano, de expansión urbana y rural se utilizará, como mínimo, análisis de tipo histórico y geomorfológico. De acuerdo con la información disponible se podrán complementar con análisis hidrológico-hidráulicos y métodos asistidos por sensores remotos y sistemas de información geográfica. En todo caso, los análisis se realizan en función de la magnitud de la amenaza, su intensidad, consecuencias y la disponibilidad de información.

4. **Productos:** Como resultado de los estudios, se elaboran mapas de zonificación de amenaza por inundaciones, según lo dispuesto en el presente artículo. Se debe elaborar un documento técnico que contenga la metodología empleada y los resultados obtenidos.

(Decreto 1807 de 2014, artículo 9°)

Artículo 2.2.2.1.3.2.1.4. Delimitación y zonificación de las áreas con condición de amenaza. Con fundamento en la delimitación y zonificación de amenazas, se delimitan y zonifican aquellas áreas sin ocupar del suelo urbano, de expansión urbana, rural suburbano o centros poblados rurales en las que en la revisión o en la expedición de un nuevo POT se proponga su desarrollo. La identificación de estas áreas se realizará a partir del análisis de las áreas zonificadas como de amenaza alta y media sin ocupar en los estudios básicos con aquellas que se consideren como objeto de desarrollo. En todo caso el desarrollo de las zonas de amenaza media y alta sin ocupar quedará sujeto a los resultados de los estudios detallados. Con esta información se elabora el mapa con la delimitación y zonificación de las áreas con condición de amenaza y se establecen los criterios para la caracterización y delimitación de las unidades de análisis en las áreas que serán objeto de estudios detallados. [15]

(Decreto 1807 de 2014, artículo 11)

5.7. Sistemas de alerta temprana

Los Sistemas de Alerta Temprana (SAT) son herramientas clave para proporcionar información pertinente y efectiva a través de entidades especializadas en ciencia, tecnología y comunidades. Estos sistemas utilizan diversos métodos y herramientas que facilitan a las personas en riesgo la toma de decisiones para minimizar o evitar peligros, as´ı como prepararse para responder adecuadamente según sus habilidades.

La EIRD menciona que es esencial tener un conjunto de habilidades para crear y distribuir alertas de manera oportuna y efectiva, lo cual ayuda a individuos, comunidades y organizaciones a prepararse y actuar de forma adecuada y con el tiempo necesario para disminuir la probabilidad de daños o pérdidas.

Los SAT habilitan a personas y comunidades amenazadas para actuar a tiempo y de forma correcta, minimizando así el riesgo de lesiones, pérdida de vidas, daños a propiedades y al medio ambiente.

La alerta temprana provee soporte técnico a comunidades o individuos, permitiéndoles actuar de manera oportuna y apropiada para reducir el riesgo de daños personales, pérdida de vidas, daños a propiedades y al entorno, frente a eventos adversos que podrían desencadenar situaciones de peligro.

Siguiendo las directrices internacionales, la UNISDR (NOAA) sugiere cuatro componentes esenciales para la alerta temprana:

1. Detección y pronóstico de amenazas.
2. Evaluación de riesgos y manejo de información.
3. Comunicación oportuna, fiable y clara.
4. Planeación, preparación y capacitación en respuesta a todos los niveles (institucional y comunitario).

Esto asegura que cualquier medida tomada para la alerta temprana esté coordinada con la planificación en gestión de riesgo, así como con las acciones de respuesta y recuperación. [16]

5.8. Elementos de un SAT

1. **Sensor:** Los eventos naturales pueden ser monitoreados visualmente o a través de sensores a distancia, empleando diversas tecnologías de comunicación. El monitoreo mediante sensores puede llevarse a cabo de forma local o de manera automática y remota. Asimismo, la determinación de las alertas y su nivel puede ser establecida por la autoridad hidrometeorológica (como el IDEAM en Colombia) para ciertos eventos, por el SGC para otros, o por distintas entidades como comunidades que estén debidamente entrenadas, verificadas y coordinadas por las autoridades nacionales pertinentes. Miden y señalizan cualquier incremento peligroso en el nivel de los cursos de agua, embalses y lodazales, en algunos lugares que necesitan monitorizar continuamente el nivel de agua, no tiene por qué haber conexión eléctrica, los sensores de monitorización utilizados están provistos de baterías. Estos sensores pueden funcionar incluso durante varios años. Como medida alternativa pueden utilizarse también baterías que se recargan a partir de colectores solares.

   a) **HC-SR04:** Este es un sensor ultrasónico de bajo costo y ampliamente utilizado en proyectos DIY y educativos. Es ideal para prototipos y experimentos iniciales. Sin embargo, su rango de medición es limitado a aproximadamente 4 metros, y puede no ser tan robusto o confiable para aplicaciones en exteriores a largo plazo.
      - Voltaje de trabajo: 5V (CC)
      - Corriente estática: menos de 2 mA.
      - Señal de salida: Señal de frecuencia eléctrica, alto nivel 5V, bajo nivel 0V.
      - Ángulo del sensor: no más de 15 grados.
      - Distancia de detección: 2cm-450cm.
      - Alta precisión: hasta 0,3 cm
      - Señal de disparo de entrada: 10us impulso TTL
      - Señal de eco: señal de salida TTL



Figura 5.1: HC-SR04, tomado de [17]

2. **Microcontrolador:** Estos elementos son los encargados de recibir toda la información proveniente de los sensores y realizar el procesamiento de la información para comunicarse con otros dispositivos o con una central de procesamiento a través de módulos inalámbricos o cableados. En el contexto de IoT, esto suele hacerse mediante Wi-Fi, Bluetooth, GSM o LPWAN.

   a) **ESP32:** Es muy útil por su conectividad inalámbrica integrada, procesamiento relativamente potente y bajo costo. Ideal para proyectos de IoT (Internet de las Cosas), wearables, automatización del hogar y cualquier proyecto que requiera comunicación inalámbrica.

   - Procesador Tensilica Xtensa 32bits LX6 hasta 240MHz.
   - Wi-Fi: 802.11b/g/n/e/i (802.11n @ 2.4 Ghz hasta 150 Mbit/s).
   - Bluetooth: v4.2 BR/EDR y bluetooth Low Energy (BLE).
   - Rom:448 KiB.
   - SRAM: 520 KiB.
   - RTC slow SRAM: 8 KiB.
   - RTC fast SRAM: 8 KiB.
   - eFuse: 1 Kbit.
   - Flash embebida: 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips); 2 MiB (ESP32-D2WD chip); 4 MiB (ESP32-PICO-D4 SIP module).
   - Periféricos compatibles: ADC, DAC, I2C, UART, Interfaz CAN 2.0, SPI, I2S, RMII y PWM entre otros.
   - Seguridad tipo IEEE 802.11, WFA, WPA/WPA2 y WAPI.
   - Encriptación de memoria Flash.
   - Criptografía soportada por acelerador de hardware: AES, SHA-2, RSA, ECC,

RNG.

- Voltaje de trabajo 3.3VDC.
- Energía y datos via conector microUSB 5VDC.



Figura 5.2: ESP32, tomado de [18]

1. **Servidor:** Para un proyecto que requiere enviar datos a una aplicación móvil para su visualización y control, necesitas un servidor o una plataforma en la nube que pueda

   manejar la recepción, almacenamiento y procesamiento de estos datos. Existen varias opciones, desde servidores y plataformas especializadas en Internet de las Cosas (IoT) hasta soluciones de cloud computing más genéricas.

   - **Firebase de Google:** Firebase es una plataforma de desarrollo de aplicaciones móviles y web que ofrece una base de datos en tiempo real, autenticación de usuarios, hosting y muchas otras características que facilitan la creación y gestión de aplicaciones ricas e interactivas. Presenta ventajas como la Integración con otras herramientas y servicios de Google, escalabilidad, y una amplia gama de características para el desarrollo de aplicación. [19]

Capítulo 6

Estado del arte

An IoT-based scalable river level monitoring platform [20]. En el documento se describe un sistema de monitoreo de ríos en Antioquia, implementado para alertar tempranamente sobre crecidas y prevenir tragedias. Utilizando el Internet de las cosas, el sistema cuenta con unas 45 estaciones automatizadas que miden el nivel del agua y la precipitación, enviando datos en tiempo real. El proyecto es una colaboración entre Cornare y la Pontificia Universidad Javeriana, y utiliza ultrasonido para detectar cambios en el nivel del agua, ayudando a las comunidades a prepararse para posibles inundaciones.



Figura 6.1: Dispositivo implementado en la práctica, tomado de [20]

Además de generar la alerta inmediata, la interfaz gráfica que está en la nube permite visualizar los puntos donde se está generando el riesgo: "Si en el punto X a las 5:21 se presenta un desbordamiento, significa que a las 6:57 se reflejará en el punto Y aguas abajo de la misma quebrada", explica Trujillo. Si se desborda el río y puede ocurrir una inundación, el sistema avisa con tiempo. Henao lo complementa: "No solo es importante poder monitorear el momento de una crecida o en que puede presentarse un evento; queremos conocer qué pasa con las fuentes hídricas, poder saber históricamente cómo se vienen comportando en tiempos

de menos lluvias, en tiempos de más lluvias, para que nosotros hagamos las correlaciones con modelos matemáticos y climatológicos, y poder empezar a caracterizar e identificar mejor la problemática de nuestras quebradas en cuanto al mismo caudal, a la oferta hídrica". Se trata, en suma, de "tener información más consolidada".

RiverCore: IoT Device for River Water Level Monitoring over Cellular Communications [21]. El documento describe el desarrollo de RiverCore", un dispositivo IoT para monitorear el nivel del agua en ríos utilizando comunicaciones celulares. Se centra en el diseño del hardware, la implementación en una región hidrológica específica en Colima, México, y las características de adquisicion y procesamiento de datos. El sistema incluye nodos fijos y móviles (drifters), una plataforma de adquisición de datos basada en la web integrada con técnicas de IoT para recopilar datos a través de redes celulares 3G, y utiliza el protocolo MQTT para la transmisión segura de paquetes de datos en tiempo real.

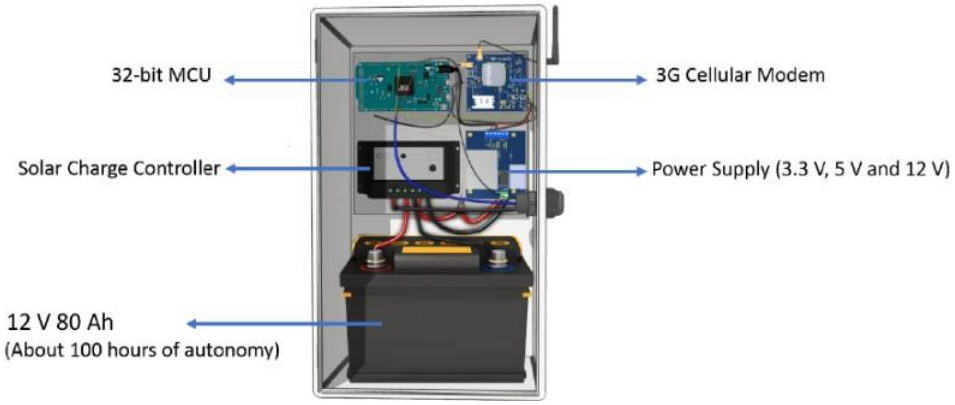


Figura 6.2: Modelo 3D de un nodo fijo RiverCore., tomado de [21]

La imagen muestra un modelo 3D de un nodo fijo RiverCore, el cual es parte del dispositivo IoT diseñado para el monitoreo del nivel del agua en ríos. El nodo incluye un controlador de carga solar, un MCU de 32 bits, un módem celular 3G y una fuente de alimentación con múltiples salidas de voltaje. También se visualiza una batería de 12 V y 80 Ah que proporciona aproximadamente 100 horas de autonomía, lo que implica que el dispositivo está diseñado para operar de forma autónoma y en ubicaciones remotas sin acceso a la red eléctrica.

Design of Information Monitoring System Flood Based Internet of Things (IoT) [22]. Aqui se describe el desarrollo de un prototipo de sistema de monitoreo de inundaciones basado en IoT. Este sistema tiene como objetivo facilitar a los usuarios el acceso a información sobre niveles de agua y condiciones meteorológicas lluviosas mediante sensores ultrasónicos y de lluvia. La información recopilada por los sensores se procesa con un microcontrolador Arduino Uno para generar datos en formato HTML, que se distribuyen a través de un módulo Ethernet actuando como servidor web, integrado con un enrutador inalámbrico como puerta de enlace para el usuario. Este trabajo resalta la creación de un sistema de monitoreo de inundaciones en tiempo real, accesible v´ıa web.

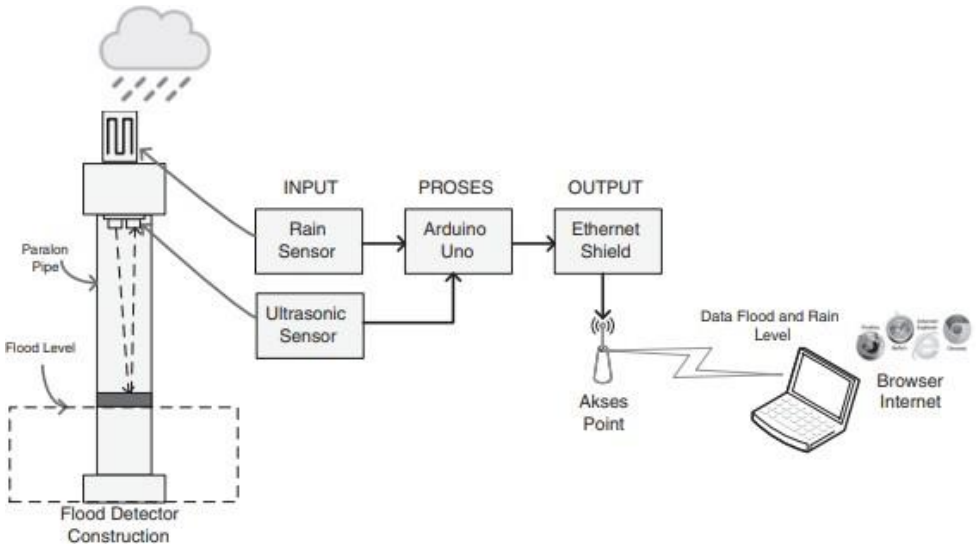


Figura 6.3: Diagrama de bloques del prototipo propuesto, tomado de [22]

En la figura 6.3 se muestra de un sistema de monitoreo de inundaciones basado en IoT. Muestra un sensor de lluvia y un sensor ultrasónico como entradas (INPUT), que se conectan a un microcontrolador Arduino Uno. El procesamiento (PROSES) se realiza en el Arduino, y los datos resultantes se envían a través de un Ethernet Shield. Luego, los datos sobre el nivel de lluvia y de inundación se transmiten a un punto de acceso (Akses Point) y se ponen a disposición a través de Internet para ser visualizados en un navegador web (Browser Internet). Esto permite la observacion remota de los niveles de inundación y lluvia en tiempo real.

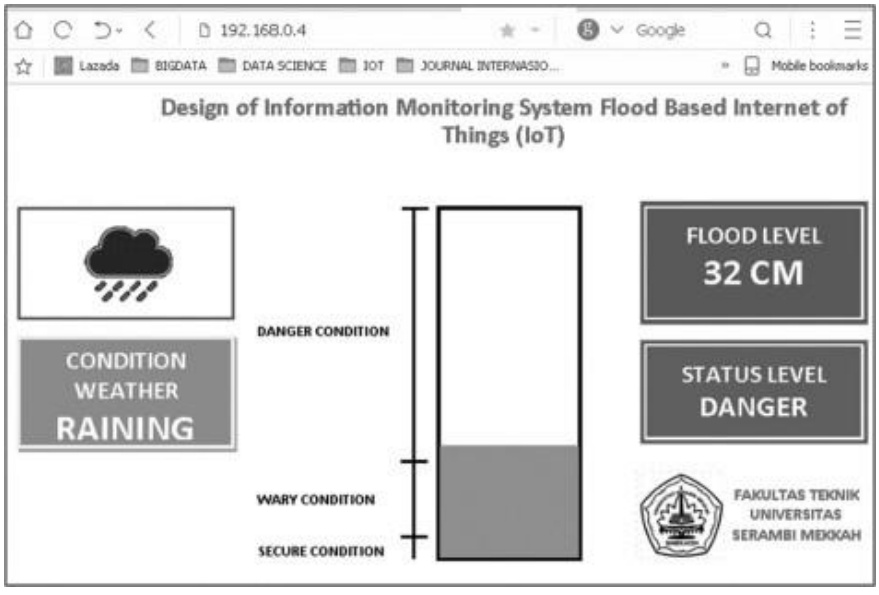


Figura 6.4: Información recopilada vizualizada, tomado de [22]

En la figura 6.4 se puede observar cómo se consiguió satisfactoriamente la visualización de los datos recogidos por los sensores en tiempo real.

The Implementation of an IoT-Based Flood AlertSystem [23]. El documento presenta el desarrollo de un sistema de alerta de inundaciones utilizando IoT, que detecta el nivel del agua y la velocidad de su ascenso, enviando alertas por SMS. Se adopta el modelo Waterfall para el desarrollo, utilizando Raspberry Pi para la recopilación de datos de sensores de agua y un módulo GSM para la transmisión de alertas. Se realiza un análisis para integrar el sistema con smartphones y se prueba en dos entornos diferentes para asegurar precisión y fiabilidad. Este enfoque alinea la tecnología IoT con la Revolucion Industrial 4.0, apoyando la infraestructura de los sistemas ciberfísicos.

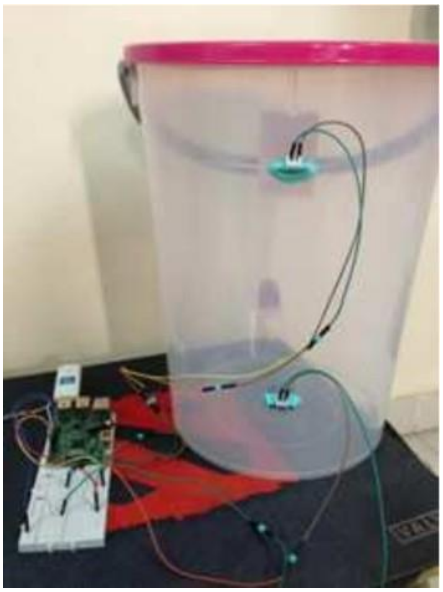


Figura 6.5: El prototipo del sistema y la configuración del experimento, tomado de [23]

Designing Early Warning Flood Detection and Monitoring System via IoT [24]. El documento aborda el desarrollo de un sistema avanzado para la detección temprana y monitoreo de inundaciones usando tecnología IoT. Propone un mecanismo para proporcionar alertas tempranas sobre niveles críticos de agua mediante el uso de sensores ultrasónicos, un sistema de válvulas para el manejo de flujos de agua, y comunicación en tiempo real a través de aplicaciones móviles. Este enfoque busca mitigar los efectos adversos de las inundaciones, facilitando una respuesta más eficaz ante estas emergencias.

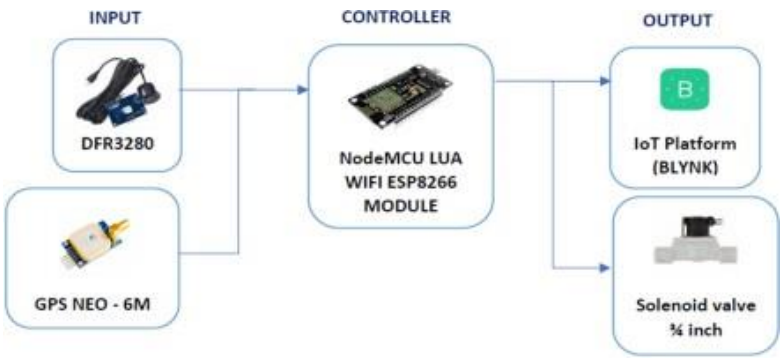


Figura 6.6: Diagrama de bloques, tomado de [24]

La figura 6.6 muestra el diagrama de flujo del sistema IoT para la gestión de inundaciones. En el lado de la entrada (INPUT), hay un sensor de agua DFR3280 y un módulo GPS NEO-6M. Estos se conectan a un controlador NodeMCU LUA WiFi ESP8266 (CONTROLLER). En la salida (OUTPUT), los datos se envían a una plataforma IoT (BLYNK) y activan una válvula de solenoide de ¼ de pulgada. En conjunto se tiene un sistema que puede detectar la ubicación y el nivel de agua, y luego actuar sobre esos datos a través de una plataforma de IoT y un mecanismo de control físico.



Figura 6.7: Prototipo final, tomado de [24]

Flood Alert Management System Using IoT and Microcontroller [25]. El documento describe el diseño y la implementación de un sistema de alerta de inundaciones que integra tecnología IoT y microcontroladores para monitorear y gestionar niveles de agua en embalses, lagos y áreas de almacenamiento de agua. Utiliza sensores avanzados para detectar los niveles de agua y un sistema observatorio de inundaciones compuesto por microcontroladores ATMEGA328P, sensores ultrasónicos y de nivel de agua, y módulos de comunicación como Bluetooth y una pantalla LCD para mostrar los cambios en los niveles de agua. El sistema es capaz de actuar automáticamente al abrir compuertas para liberar agua excedente

durante condiciones anormales de nivel de agua, demostrando un enfoque automatizado y sin intervención humana para la gestión de inundaciones.

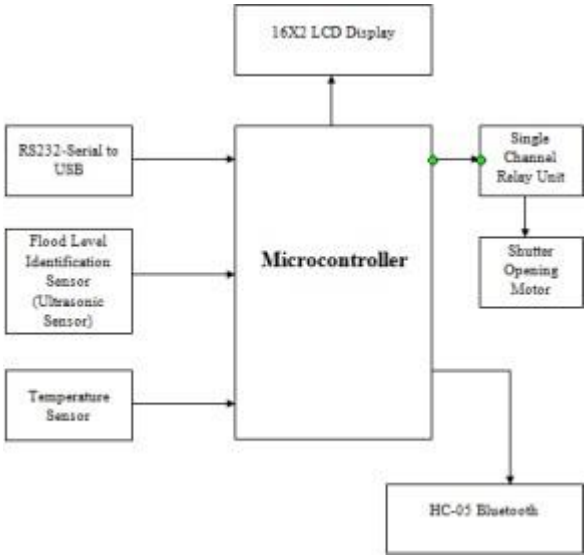


Figura 6.8: Diagrama de la arquitectura del sistema, tomado de [25]

La figura 6.8 muestra el esquema del sistema de alerta de inundaciones, centrado en un microcontrolador conectado a varios dispositivos y sensores. Los componentes incluyen un sensor de nivel de agua (ultrasónico), un sensor de temperatura, un módulo RS232-Serial a USB, y un módulo Bluetooth HC-05. El microcontrolador también está conectado a un display LCD 16x2 y a una unidad de relé de canal único que controla un motor de compuerta. Este sistema permite la recopilación de datos del nivel de agua y la temperatura, la visualizacion de estos en una pantalla y la activación automática de mecanismos de respuesta como la apertura de compuertas para controlar el flujo de agua.

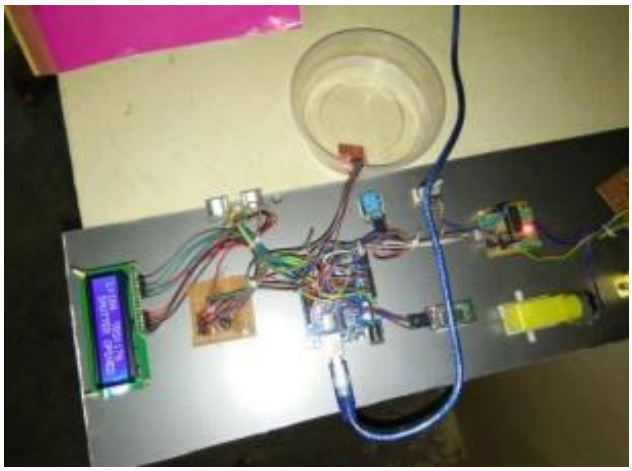


Figura 6.9: Prototipo funcionando, tomado de [25]

Capítulo 7

Plan de trabajo

La figura 7.1 se tiene un diagrama de bloques general para un sistema de monitoreo y generación de alertas de prevención contra inundaciones con IOT, consta de un bloque inicial que incluye sensores de nivel de agua, y otros sensores ambientales en caso de ser necesarios, su función es recoger datos físicos del entorno, el bloque del microcontrolador se encarga de procesar los datos recogidos por los sensores, posteriormente se transmiten los datos procesados vía Wi-Fi o Bluetooth para la conectividad IoT, el siguiente bloque recibe los datos transmitidos y los almacena, finalmente se tiene el punto final para los usuarios, donde reciben alertas basadas en los datos procesados, la app debe poder mostrar información en tiempo real, históricos de datos y alertas basadas en criterios específicos. En la figura 7.2 se presenta une esquemático en donde se pretende ilustrar el funcionamiento anteriormente descrito.

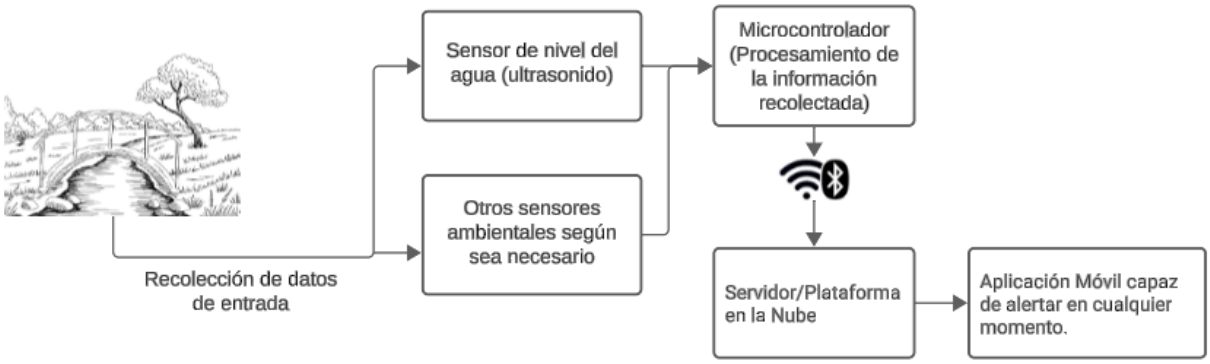


Figura 7.1: Diagrama de bloques del proyecto propuesto.

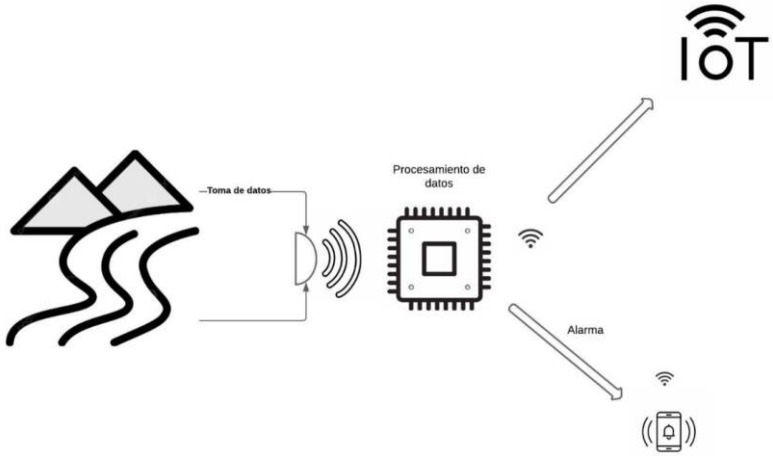


Figura 7.2: Esquemático del funcionamiento del proyecto propuesto.

Objetivo 1: Determinar las condiciones iniciales en las que se aplicará el sistema de alerta de riesgos.

Actividades:

- 1.1 Definir las dimensiones que va a tener la estructura y qué condiciones se van a simular en la misma.

Objetivo 2: Implementación de la estructura en la que se va a realizar las mediciones.

Actividades:

- 2.1 Identificar que materiales van a ser necesarios para construir la maqueta.
- 2.2 Construir de la estructura.
- 2.3 Realizar diferentes pruebas para comprobar el funcionamiento óptimo de la estructura.

Objetivo 3: Determinar los sensores más adecuados considerando las limitaciones de la estructura y el presupuesto para el proyecto.

Actividades:

- 3.1 Seleccionar los sensores que se adecuen mejor a la estructura.
- 3.2 Verificación de los sensores para observar sus limitantes.
- 3.3 Realizar una adaptación a la estructura para acoplar los sensores dentro de lo diseñado.

Objetivo 4: Analizar y determinar el sistema de procesamiento de datos que cumpla con las condiciones necesarias para una correcta implementación.

Actividades:

- 4.1 Seleccionar el mejor sistema de procesamiento de datos teniendo en cuenta las necesi-dades que se tengan en cuanto a entradas digitales, conexión a WI-FI
- 4.2 Adaptar el sistema de procesamiento de datos para su facil manejo.

Objetivo 5: Diseñar e implementar una aplicación movil capaz de alertar en cualquier momento a cualquier persona.

Actividades:

- 5.1 Definir que funcionalidades específicas va a tener la aplicación.
- 5.2 Elegir las plataformas de desarrollo móvil (iOS/Android), lenguajes de programación y herramientas.

- 5.3 Conectar la aplicación con las fuentes de datos para poder realizar el monitoreo en tiempo real.
- 5.4 Realizar las pruebas necesarias para verificar el correcto funcionamiento de la aplica-ción.

Objetivo 6: Simular con la estructura condiciones especificadas con el sistema de monitoreo.

Actividades:

- 6.1 Hacer pruebas de creciente subita y de una creciente que aumenta lentamente paraverificar el funcionamiento del sistema de alerta y monitoreo.
- 6.2 Observar en la aplicación que se emite una alarma cuando el sistema monitoreo se lo indique.
- 6.3 Registrar los resultados obtenidos.

Capítulo 8

Desarrollo

8.1 Condiciones iniciales en las que se aplicara el sistema de alerta temprana.

Niveles de Alerta y Condiciones para su Activación en Colombia

En Colombia, los niveles de alerta para inundaciones están claramente definidos y se activan bajo condiciones específicas para garantizar una respuesta adecuada a las emergencias. Aquí se describen los niveles de alerta y las circunstancias bajo las cuales se activan, basados en las normativas del IDEAM y otras entidades de gestión del riesgo:

1. Alerta Amarilla

Condiciones para Activación:

- **Nivel del Agua**: Se activa cuando el nivel del agua del río alcanza un 70% de la capacidad máxima del cauce.
- **Precipitaciones Intensas**: Pronósticos de lluvias continuas e intensas que podrían aumentar significativamente el nivel del río.
- **Indicadores de Saturación del Suelo**: Suelo ya saturado por lluvias previas, lo que aumenta el riesgo de deslizamientos de tierra y escorrentía rápida.

Acciones a Tomar:

- **Monitoreo Continuo**: Intensificación del monitoreo hidrológico y meteorológico.
- **Preparación Comunitaria**: Informar a las comunidades y autoridades locales sobre la posibilidad de una emergencia.
- **Revisión de Infraestructura**: Verificación del estado de infraestructuras críticas, como puentes y diques, para asegurar que estén en buenas condiciones.

2. Alerta Naranja

Condiciones para Activación:

- **Nivel del Agua**: Se activa cuando el nivel del agua del río alcanza un 85% de la capacidad máxima del cauce.
- **Aumento Rápido del Nivel del Agua**: Incremento rápido y sostenido del nivel del agua en un corto período de tiempo.
- **Desbordamiento Parcial**: Indicios de desbordamiento en áreas específicas del río.

Acciones a Tomar:

- **Evacuación Preventiva**: Evacuación de personas en áreas de alto riesgo.
- **Activación de Equipos de Respuesta**: Movilización de equipos de emergencia y recursos necesarios.
- **Información Continua**: Proveer actualizaciones continuas a la comunidad y autoridades sobre la evolución del riesgo.

3. Alerta Roja

Condiciones para Activación:

- **Nivel del Agua**: Se activa cuando el nivel del agua del río alcanza o supera el 95% de la capacidad máxima del cauce.
- **Desbordamiento Completo**: Desbordamiento del río afectando áreas extensas y pobladas.
- **Condiciones Meteorológicas Extremas**: Pronósticos de tormentas o ciclones que exacerben las condiciones actuales.

Acciones a Tomar:

- **Evacuación Total**: Evacuación completa de las áreas afectadas.
- **Respuesta de Emergencia**: Implementación de planes de emergencia y socorro.

- **Apoyo Logístico y Humanitario**: Provisión de refugios temporales, alimentos, y atención médica.

Determinación de Umbrales:

Utilizando la maqueta para simular diferentes niveles de agua para comprender las condiciones bajo las cuales se activarán los diferentes niveles de alerta. Esto implica:

Dado que la estructura contará con las siguientes características:

La altura que va a definir la capacidad máxima del cauce será de 20 cm, entonces, se decidió colocar el sensor ultrasónico a una distancia de 60 cm del suelo del cauce, por medio de código se va a ajustar la medición para que mida desde 0 cm, que es cuando el río este sin nada de agua, hasta 20 cm que sería cuando se presente el desbordamiento, teniendo esto en cuenta, para el caso de este trabajo las condiciones iniciales bajo las cuales se activarán los diferentes niveles de alerta (posteriormente visualizados y escuchados mediante una aplicación móvil y una sirena), son las siguientes:

Alerta amarilla: Se activa cuando el nivel del agua del río alcance los 12 cm.

Alerta naranja: Se activa cuando el nivel del agua del río alcance los 14.45 cm.

Alerta roja: Se activa cuando el nivel del agua del río alcance los 16 cm.

Dado que también se hará uso de un sensor de lluvia, esta característica será tenida en cuenta para activar algún nivel de alerta antes, dependiendo de la intensidad de la lluvia que indique el sensor.

8.2 Dimensiones de la estructura propuesta.

Se proponen las siguientes dimensiones para la estructura, en primera instancia, la base va a ser de 150 cm de largo, esto para tener un buen tramo de la estructura en el cual realizar el censado, y un ancho de 50 cm, para poder replicar lo mejor posible las características como por ejemplo las riberas.

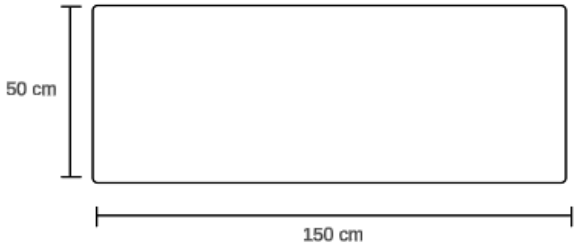


Figura 8.2.1: Medidas para la base de la estructura.

Teniendo en cuenta que el nivel del agua tiene que subir de manera que lo sensores detecten cambios considerables, se plantea una altura de 20 cm para el crecimiento del nivel de agua del rio, se van a utilizar rectángulos de madera de 20 cm x 20 cm para lograr dicha altura a lo largo de la estructura, adicionalmente se van a utilizar rectángulos de las mismas dimensiones para reforzar un poco mas las paredes que van aguantar el agua y también para que se noten mejor las riberas en nuestra estructura.

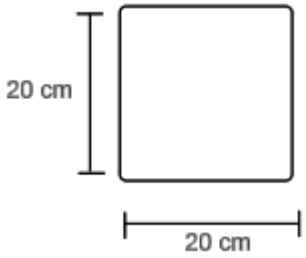


Figura 8.2.2: Medidas de los rectángulos de madera.

La altura de 20 cm se escoge, teniendo en cuenta que un rio tiene una altura variable a lo largo de su cause, tomando como referencia una altura de 20 metros, por lo que la escala es de 100:1.

Se va a orientar la colocación de los rectángulos de madera para que se vea cierta curvatura de la maqueta del rio, el uso de bombas de agua va a permitir visualizar el caudal del rio.

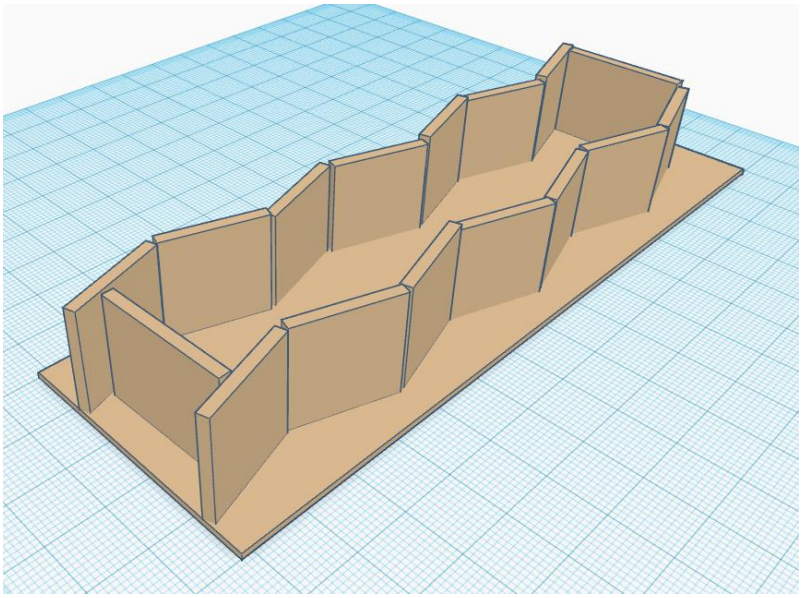


Figura 8.2.3: Propuesta de maqueta (fase inicial).

8.3 Materiales utilizados para la construcción de la estructura.

- Una base de madera de 150 cm de largo y 50 cm de ancho.
- 16 rectángulos de madera de 20 cm x 20 cm cada uno.
- 28 rectángulos de ipocor de 20 cm de ancho por 18 cm de largo cada uno.
- L´s y tornillos de rosca.
- Plástico.
- Pintura color ocre.
- Cable.
- 2 motobombas de 3 V- 6 V
- 1 motobomba sumergible pump de 110 V-120 V
- 2 cargadores de 5 V
- Materiales para decorar la estructura y dar relieve.

8.4 Construcción de la estructura.

La base va a ser de madera, para la construcción de las paredes que aguanten el agua, se va a utilizar madera e icopor, con ayuda de L´s y tornillos de rosca de van a fijar dichos rectángulos de madera a la base, para que el agua pueda circular sin que se presenten fugas, posteriormente se cubren las paredes de madera con rectángulos de icopor para poder

pintarlos después y que se noten un poco las riberas, inicialmente la estructura queda como se observa en la figura 7.4.

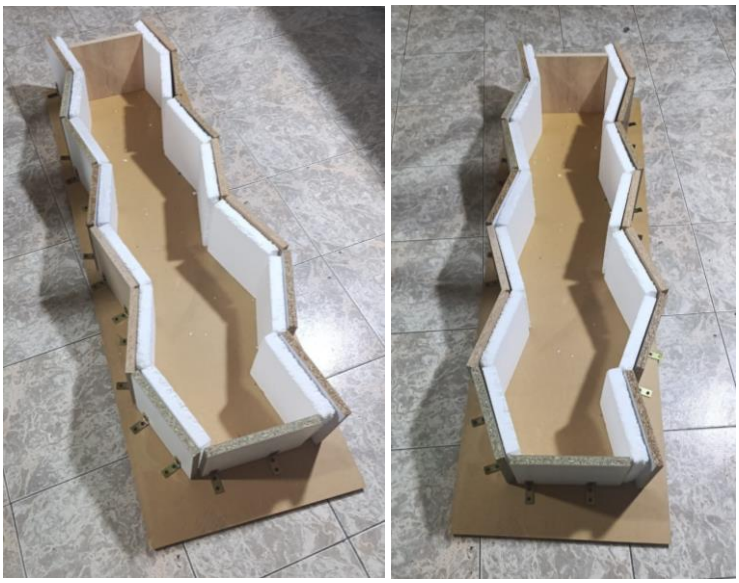


Figura 8.4.1: Etapa inicial de la estructura.

Posteriormente se pintan los rectángulos de icopor con vinilo de color ocre, y se procede a recubrir toda la maqueta con plástico, ahora se necesitan las bombas de agua que ayuden a recircular el agua y a simular el caudal de la maqueta, con ayuda de las 2 motobombas de 3 V- 6 V, las cuales se dejan sus terminales soldados a los cargadores de 5 V y las mangueras que van a llevar el agua al inicio del rio se acomodan para que puedan trabajar correctamente, finalmente se ajustan dos tapas en las zonas donde van a estar los motores paraque estos no sean visibles, y por ultimo se agregan los detalles de relieve a la maqueta para que quede bien decorada, como se puede ver en la figura 7.5.

8.5 Sensores adecuados.

Las principales consideraciones para la selección de los sensores que se van a utilizar en este trabajo tienen en cuenta tanto los costos como la efectividad en términos de lo que se necesita al usar los mismos.

8.5.1 Nivel de agua del río

Para medir el nivel de agua del río se escogió el sensor ultrasonido HC-SR04, es adecuado para la estructura, dada su capacidad de medir distancias con precisión y su utilidad en la detección de niveles de agua a corta distancia. Siendo un sensor de bajo costo, lo que lo hace accesible para nuestro proyecto.

El HC-SR04 típicamente tiene un rango de medición efectivo de aproximadamente 2 cm a 400 cm. Cualquier objeto fuera de este rango no se detectará correctamente,

El HC-SR04 tiene un ángulo de detección que puede llevar a lecturas imprecisas si el objeto no está bien alineado con el sensor. El ángulo de detección relativamente estrecho requiere una alineación precisa con la superficie del agua, lo cual puede ser un problema si nuestra maqueta no es muy ancha, pero en la puesta en práctica en un río real no debería haber problemas.

8.5.2 Detección de lluvia y medición de su intensidad

Para determinar cuándo se tengan condiciones de lluvia y la intensidad de la misma, se escogió el sensor de lluvia YL-83, este sensor es fácil de integrar con los microcontroladores propuestos en el trabajo y no requiere configuraciones complejas, es muy sensible a la presencia de agua, lo que permite detectar rápidamente la acumulación de agua, siendo ideal para nuestro proyecto. Es un sensor de bajo costo, lo que lo hace perfecto para nuestra propuesta de trabajo. A pesar de su precio, ofrece una buena precisión y fiabilidad y otro factor clave es que consume muy poca energía, perfecto ya que nuestro sistema está pensado para funcionar con baterías que se cargan con ayuda de paneles solares. Proporciona una señal digital y una señal analógica. La señal digital se activa cuando se detecta agua, mientras que la señal analógica proporciona una medición más granular de la cantidad de agua presente.

8.6 Actuadores.

Se decidió incorporar una sirena, por lo que esta se verá activada para el caso en que se presente una alerta roja a partir de las mediciones de los sensores.

Se propone utilizar una sirena de 6-12 Voltios, 10 W (HPS102), aclarando que en la implementación en un escenario real lo ideal sería desplegar varias réplicas del circuito alrededor del área que es vulnerable a la inundación. El esquema inicial para el funcionamiento de la sirena es el siguiente:

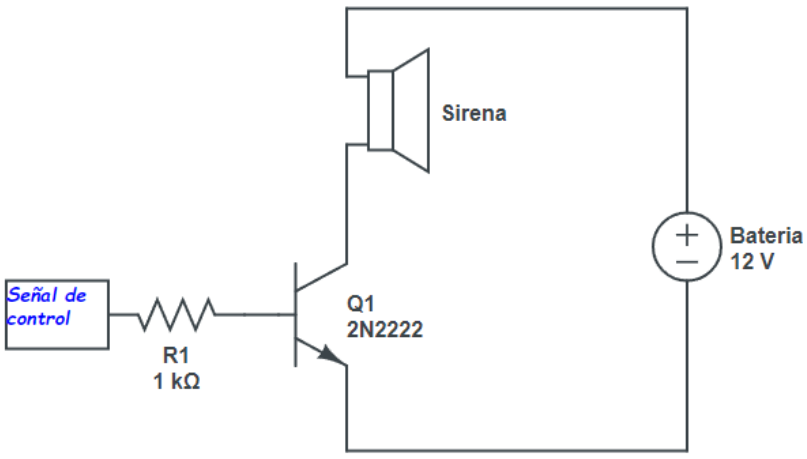


Figura 8.6.1: Circuito accionador de la sirena.

8.7. Pruebas iniciales del funcionamiento del sistema

Con los sensores y el circuito para la sirena definidos, se hacen unas pruebas iniciales de cómo debería funcionar el sistema de monitoreo y la generación de la alarma, para esta prueba se decidió utilizar cualquier microcontrolador a la mano, en este caso un ESP32, y se empieza a programar con ayuda de la IDE de Arduino.

```
const int trigPin = 12; // GPIO 12 para el pin Trig
const int echoPin = 14; // GPIO 14 para el pin Echo
const int llueve = 23; // Pin digital del YL-83 conectado al pin 23 de la ESP32
const int intensidadLluvia = 34; // Pin analógico del YL-83 conectado al pin 34 de la ESP32
const int outputPin = 15; // Pin para activar o desactivar la sirena


void setup() {

 Serial.begin(115200); // Inicia la comunicación serial

  pinMode(trigPin, OUTPUT); // Configura el pin Trig como salida
  pinMode(echoPin, INPUT);  // Configura el pin Echo como entrada
  pinMode(llueve, INPUT);   // Configura el pin digital del YL-83 como entrada
   pinMode(outputPin, OUTPUT);  // Configura el pin de activacion de la sirena como salida
  digitalWrite(outputPin, LOW); // Inicialmente apagado

}
```

Código 1. Configuración inicial de los pines de entrada y salida.

En el código 1 se escogen los pines necesarios para operar los sensores y la sirena.

```
float measureDistance() {
  // Envía un pulso ultrasonico
  digi
  talWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Recibe el eco
  long duration = pulseIn(echoPin, HIGH);
  float distance = 60 - (duration * 0.034 / 2); // Calcula la distancia

  if (distance < 0) {
    distance = 0;
  }

  Serial.print("Distance: ");
  Serial.println(distance);
  return distance;
}
```

Código 2. Función para realizar la medición de la distancia.

La función del código 2 envía un pulso ultrasónico y calcula el tiempo que tarda en recibir el eco, utilizando esta información se calcula la distancia al nivel del rio, se le resta 60 al resultado debido a que el sensor va estar ubicado a 60 cm del nivel más bajo de la maqueta construida.

```
void loop() {

float distance = measureDistance();

// Leer la salida digital
  int digitalValue = digitalRead(llueve);
  // Leer la salida analógica
  int analogValue = analogRead(intensidadLluvia);

  // Convertir el valor analógico a porcentaje (0-100%)
  float rainIntensityPercentage = map(analogValue, 0, 4095, 0, 100);

  // Mostrar los valores en el monitor serie
  if (digitalValue == 1) {
        Serial.print("Sin lluvia");;
      } else {
        Serial.print("Con lluvia");
      }

  Serial.print(" | Valor analógico: ");
  Serial.print(analogValue);
  Serial.print(" | Intensidad de lluvia: ");
  Serial.print(-rainIntensityPercentage+100);
  Serial.println("%");
```

Código 3. Obtención de la distancia, lectura y visualización del sensor de lluvia.

En el loop se mide la distancia usando el sensor y la función mencionada en el código 2, se leen los valores del sensor de lluvia, para un valor de la señal digital de 1 se define que no está lloviendo, mientras que para el valor de 0, se define que está lloviendo, para el valor analógico que entrega el sensor, se convierte a un porcentaje para indicar la intensidad de lluvia siendo 0% el valor más bajo de intensidad y 100% el valor más alto de intensidad de lluvia, finalmente se muestran todos los resultados en el monitor serie.

```
56 if (average >= 12 && average < 14.45) {
57         Serial.println("Nivel de alerta activo: Alerta amarilla.");
58       } else if (average >= 14.45 && average < 16){
59         Serial.println("Nivel de alerta activo: Alerta naranja.");
60       } else if (average >= 16){
61         Serial.println("Nivel de alerta activo: Alerta roja.");
62       }
63
64
65 // Verificar si el nivel es mayor o igual a 16 (alerta roja)
66       if (average >= 16) {
67         digitalWrite(outputPin, HIGH); // Activar el pin de salida
68         Serial.println("Pin de salida activado. Sirena encendida.");
69       } else {
70         digitalWrite(outputPin, LOW); // Desactivar el pin de salida
71         Serial.println("Pin de salida desactivado. Sirena apagada.");
72       }
```

Código 4. Configuración de niveles de alerta y control de la sirena.

En el código 4 se establecen los niveles de alerta basados en la distancia medida, dependiendo de la distancia, se imprime un mensaje de alerta correspondiente (amarilla, naranja o roja), además, si el nivel del rio llega a 19 cm, se activa una sirena para indicar el estado de alerta roja.

Una vez se tiene todo configurado, se realizan las pruebas correspondientes a esta primera parte con el fin de comprobar el funcionamiento de los elementos con los parámetros previamente definidos, se deja la maqueta de tal forma que se pueda ir llenando de agua y vaciando para verificar que el circuito este tomando las medidas correspondientes.



Figura 8.7.1: Estructura adecuada para realizar las primeras pruebas.

Una vez se tiene todo listo, se ocupa la estructura con algo de gua para verificar que no haya ningún escape de agua y poder ocupar la capacidad total del cauce de la maqueta del rio para hacer las pruebas correspondientes.



Figura 8.7.2: Estructura con un nivel de agua de aproximadamente 9 cm.

Con ayuda de una motobomba de empieza a subir el nivel del agua de la maqueta, y simultáneamente se observa el comportamiento del sistema de monitoreo en el monitor serial de la IDE de Arduino como se ve en la siguiente figura:

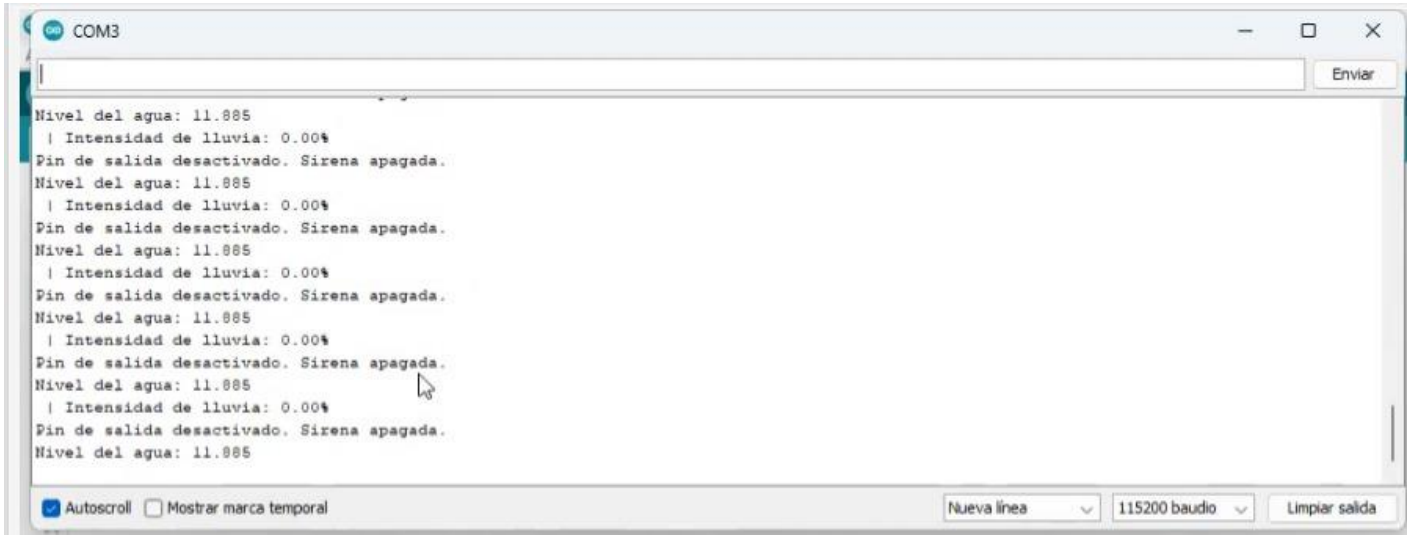


Figura 8.7.3: Estructura con un nivel de agua de aproximadamente 12 cm.

Al seguir incrementando el nivel del agua, al alcanzar el nivel para la activación de la alerta amarilla, se ve como en el monitor se observa la indicación del mismo, como se ve en la figura 8.7.4:



Figura 8.7.4: Estructura con un nivel de agua de aproximadamente 12.5 cm.

Se sigue incrementando del nivel del agua para poder visualizar la activación del nivel de alerta de alarma naranja.

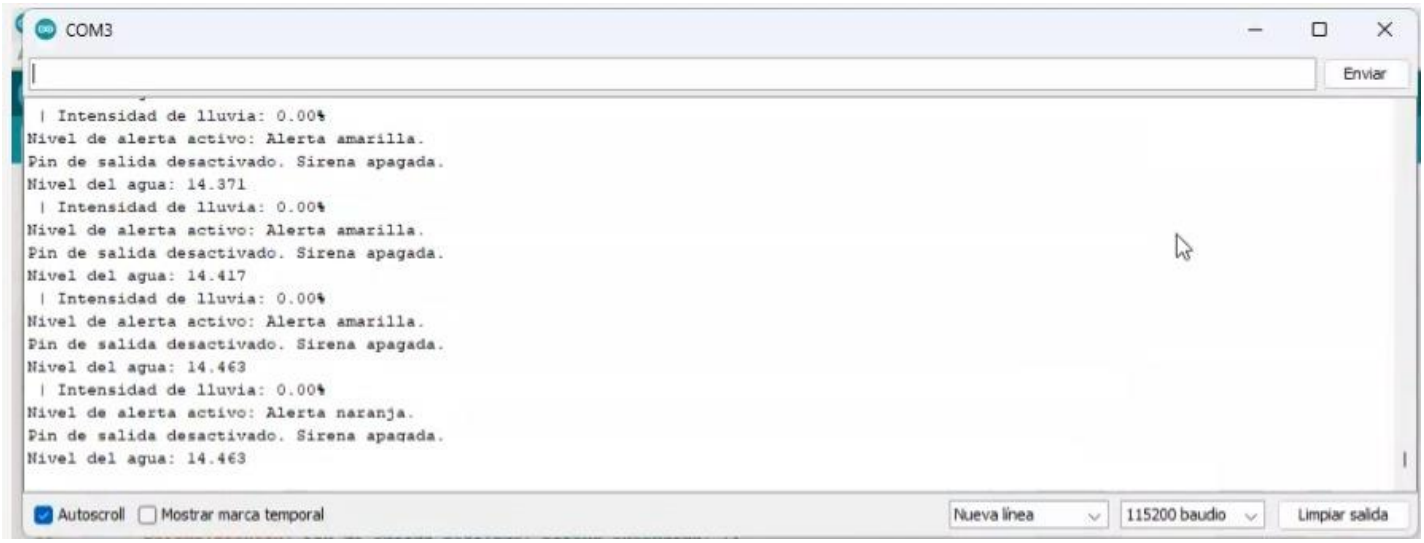


Figura 8.7.5: Estructura con un nivel de agua de aproximadamente 14.4 cm.

Finalmente se incrementa lo suficiente para poder ver la activación de la alarma roja, y con ella un mensaje en donde se indica que la sirena esta encendida, igualmente se comprobó en la practica que la alarma suena correctamente para este nivel de alerta, como se puede ver en la figura 8.7.6.

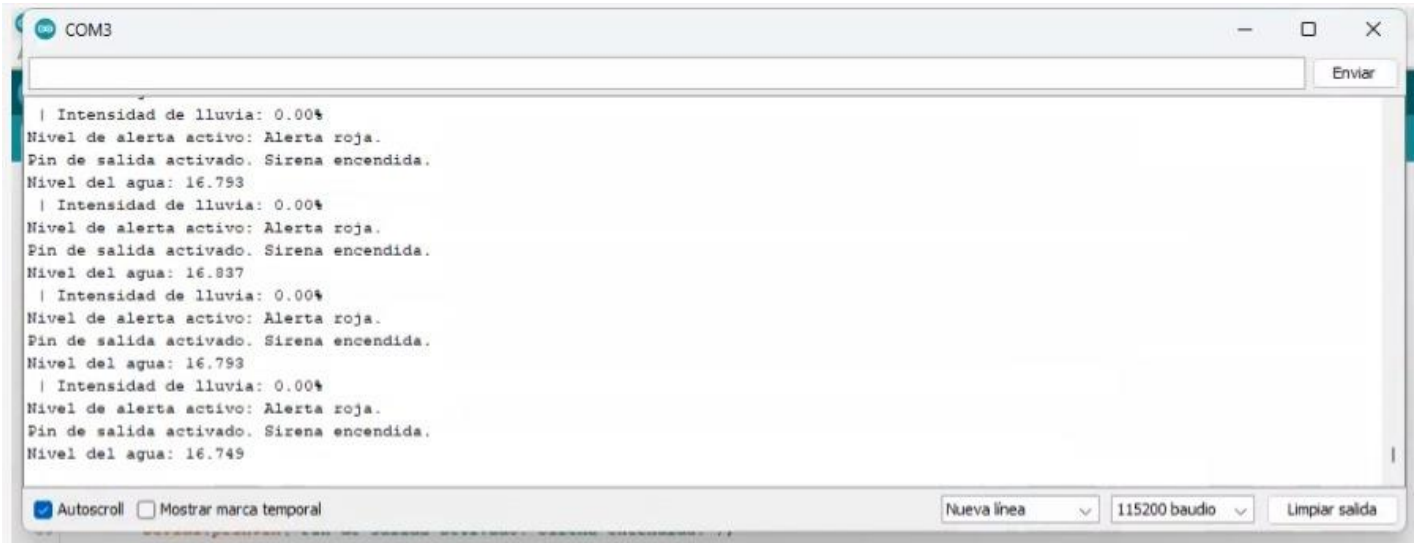


Figura 8.7.6: Estructura con un nivel de agua de aproximadamente 16.7 cm.

Se realiza una última prueba con el fin de comprobar el funcionamiento del sensor de lluvia, arrojándole algo de agua, y se puede observar en la figura 8.7.7 que se activa un mensaje que indica que esta lloviendo, y se indica el nivel de intensidad.

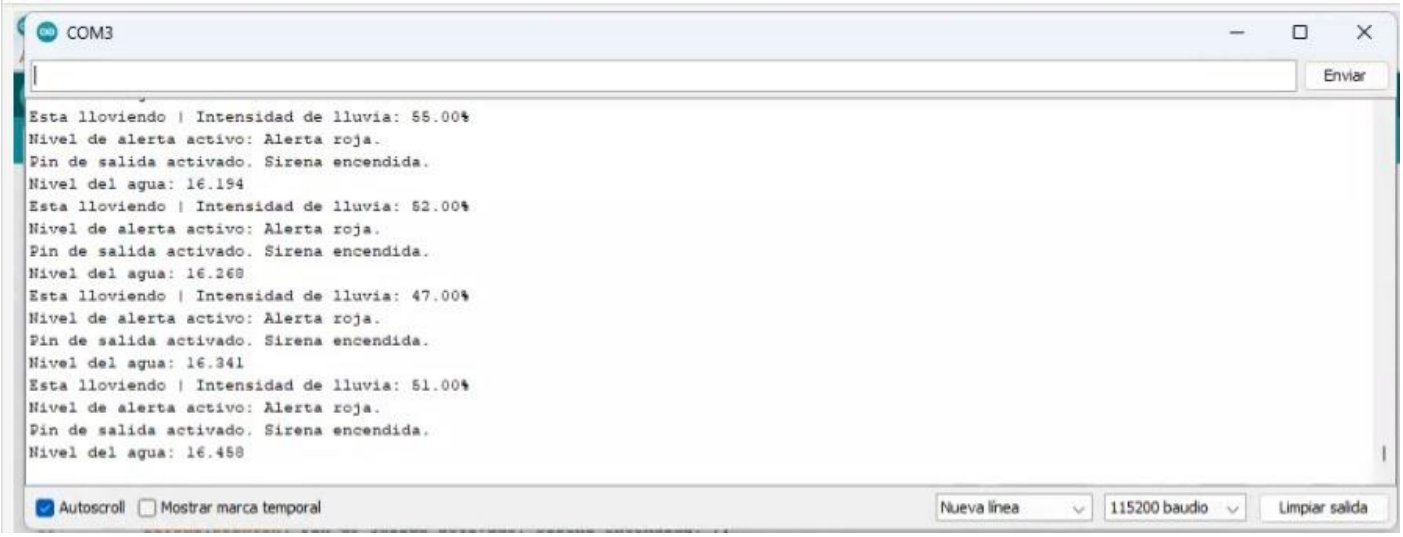


Figura 8.7.7: Activación del sensor de lluvia e indicación de intensidad de la misma.


Figura 8.7.8: Maqueta con su máxima capacidad de llenado.

Como se observa en la figura 8.7.8 la maqueta esta casi a rebosar, y el nivel de alerta roja esta activo, por lo que los umbrales que se definieron para este caso en particular funcionan correctamente, si se quisieran hacer cambios en los umbrales de activación de los diferentes niveles de alerta, solo basta con cambiar algo de código, y tener en cuenta la limitación de distancia que maneja el sensor que se está utilizando para este proyecto, que es máximo de 4.5 m.

8.8. Microcontrolador

De los microcontroladores propuestos en el marco teórico, el que mejor se ajusta a las necesidades de conectividad, procesamiento de datos, eficiencia energética y bajo costo para el proyecto es el ESP32, tiene capacidades Wi-Fi integradas que nos van a facilitar conectar el sistema de monitoreo a la red. Esto es crucial para transmitir datos en tiempo real y acceder a ellos de forma remota. Es compatible con varios entornos de programación, como Arduino IDE y Espressif IoT Development Framework (ESP-IDF), lo que nos simplifica la programación debido a la experiencia con dichos entornos. Otro elemento clave es la eficiencia energética ya que puede operar en modos de bajo consumo de energía, lo que es importante para aplicaciones de IoT que funcionan con baterías o donde el consumo de energía debe mantenerse al mínimo. también es un microcontrolador que cuenta con gran escalabilidad ya que permite la fácil Integración en redes más amplias, pudiendo aumentar el número de sensores o dispositivos conectados sin un gran incremento en el costo.

8.9. Solución de conectividad a internet

Dado que el sistema de monitoreo va a encontrarse en una situación en al que no va a tener una red de Wi-Fi cercana, se plantea la solución de usar un modulo GSM en miniatura el cual permite conectarse a internet a traes de GPRS, TCP/IP.



Figura 8.9.1. Modulo celular SIM800L GSM GPRS con antena- Shield SIM card para Arduino.

El SIM800L es compatible con la ESP32, proporciona conectividad GSM (2G), lo que permite el acceso a redes de telefonía móvil en casi cualquier parte del mundo, también soporta GPRS (General Packet Radio Service) para la transmisión de datos, GPRS es una tecnología de transmisión de datos en redes GSM (2G) que permite la transferencia de datos en paquetes, GPRS permite la transmisión intermitente de datos en paquetes optimizando el uso del ancho de banda y los recursos de la red, a través de comandos AT (conjunto de instrucciones para

controlar módems y módulo de comunicación como el SIM800L, desde un microcontrolador o computadora, facilitando la comunicación con redes de telefonía móvil y otros servicios de comunicación), se puede manejar configurar el módulo para establecer una conexión GPRS y manejar la transmisión y recepción de datos.

Para utilizar GPRS, se necesitan configurar varios parámetros en el modulo SIM800L, aspectos como la configuración APN (Access Point Name) de la red del proveedor de servicios móviles, nombre de usuario y contraseña y la activación del contexto PDP (Packet Data Protocol).

```
AT+SAPBR=3,1,"CONTYPE","GPRS"   // Configurar tipo de conexión GPRS
AT+SAPBR=3,1,"APN","_____"      // Configurar APN
AT+SAPBR=1,1                    // Activar el contexto PDP
AT+SAPBR=2,1                    // Verificar el estado de la conexión
AT+HTTPINIT                     // Inicializar el servicio HTTP
AT+HTTPPARA="CID",1             // Configurar CID para HTTP
AT+HTTPPARA="URL","_____"       // Configurar URL de destino
AT+HTTPACTION=0                 // Iniciar solicitud HTTP GET
```

Cuando se inicia la conexión GPRS, el modulo puede enviar y recibir datos a través de HTTP, HTTPS, FTP o TCP/IP. El SIM800L esta diseñado para ser eficiente en el consumo de energía, se va a implementar usando una SIM que cuente con plan de datos.

Realizando la configuración en la ESP32, hay que hacer uso de los pones TXD y RXD que realizan la comunicación serie, la ESP32 envía comandos AT al SIM800L a través del pin TXD para configurar la conexión GPRS, el SIM800L responde a estos comandos y envía datos de vuelta a la ESP32 a través del pin RXD, esto incluye respuestas de estado, datos de red y confirmaciones de las acciones que se soliciten, el código encargado de configurar el modulo SIM800L es el siguiente:

```
#include <TinyGsmClient.h>
#include <SoftwareSerial.h>

// Definir el tipo de módulo
#define TINY_GSM_MODEM_SIM800

// Configurar los pines de la ESP32
#define MODEM_RST 5
#define MODEM_PWRKEY 4
#define MODEM_POWER_ON 23
#define MODEM_TX 17
#define MODEM_RX 16

// Configurar Serial para depuración y GSM
#define SerialMon Serial
#define SerialAT Serial2

// Configurar APN (Access Point Name)
const char apn[]  = "internet.comcel.com.co"; // APN de Claro Colombia
const char user[] = "comcelweb"; // Nombre de usuario
const char pass[] = "comcelweb"; // Contraseña

// Crear instancia del cliente GSM
TinyGsm modem(SerialAT);
TinyGsmClient client(modem);
```

Código 5. Configuración APN.

En el código 5 se utilizan las librerías necesarias para el funcionamiento del modulo SIM800L, se definen los pines a utilizar para a la conexión, y se configura el APN, el nombre de usuario y la contraseña necesarios para conectarse a la red de datos de Claro, información que se encuentra en la página del operador.

```
void setup() {
  // Iniciar el monitor serial
  SerialMon.begin(115200);
  delay(10);

  // Iniciar el Serial para el módulo SIM800L
  SerialAT.begin(9600, SERIAL_8N1, MODEM_RX, MODEM_TX);
  delay(3000);

  // Reiniciar el módulo
  pinMode(MODEM_PWRKEY, OUTPUT);
  digitalWrite(MODEM_PWRKEY, LOW);
  delay(1000);
  digitalWrite(MODEM_PWRKEY, HIGH);
  delay(2000);
  digitalWrite(MODEM_PWRKEY, LOW);
  delay(3000);

  // Iniciar el módem
  SerialMon.println("Iniciando módem...");
  modem.restart();
  SerialMon.println("Módem iniciado.");

  // Conectarse a la red GPRS
  SerialMon.print("Conectando a la red GPRS...");
  if (!modem.gprsConnect(apn, user, pass)) {
    SerialMon.println("Error al conectar a la red GPRS.");
    while (true);
  }
  SerialMon.println("Conectado a la red GPRS.");
```

Código 6. Inicialización del modem y conexión de la red GPRS.

En el código 6 se realiza la inicialización del modem, reiniciando el mismo y configurándolo para la comunicación con la ESP32, finalmente se establece la conexión GPRS utilizando los detalles del APN proporcionados. Este código establece la conexión a internet y la mantiene, permitiendo que la ESP32 permanezca conectada a la red de datos móviles de claro.

8.10. Circuito de alimentación para el sistema de monitoreo

Con los sensores, actuadores, microcontrolador y demás elementos escogidos para implementar el sistema de alerta temprana, lo primero a solucionar, es cómo se va a alimentar el sistema, teniendo en cuenta que éste debe poder operar de forma ininterrumpida, se debe diseñar de tal forma que pueda replicarse de fácil manera con varios sensores y actuadores y tenga buena autonomía.

Para el caso de los elementos que van a efectuar el monitoreo continuo se decidió utilizar paneles solares, dado que el sistema de medición se diseña para ser desplegado en ubicaciones remotas y donde el acceso a la red eléctrica es limitado o inexistente, utilizar paneles solares proporciona una fuente de energía autónoma y aunque la inversión inicial en paneles solares puede ser alta, a largo plazo se reducen los costos de operación y los paneles requieren poco mantenimiento, adicionalmente tienen una vida útil larga lo cual da seguridad para que puedan funcionar de manera continua y fiable con poca intervención, finalmente también es de nuestro interés que el uso de tecnologías limpias contribuya a la concienciación sobre la importancia de la energía renovable.



Figura 8.10.1. Panel Solar 5v 300 mA Fotovoltaico.

Se escogió el panel solar ilustrado en la figura 8.10.1, teniendo en cuenta que se pretende alimentar el ESP32 con un arreglo de 4 baterías en paralelo, esto con el fin de que se tenga una buena capacidad de almacenamiento de energía para poder operar el sistema durante largos periodos de tiempo, y así mismo cargar dichas pilas con ayuda de un módulo y dos de los paneles solares propuestos conectados en paralelo para tener una buena capacidad de carga a las baterías durante las horas de sol.

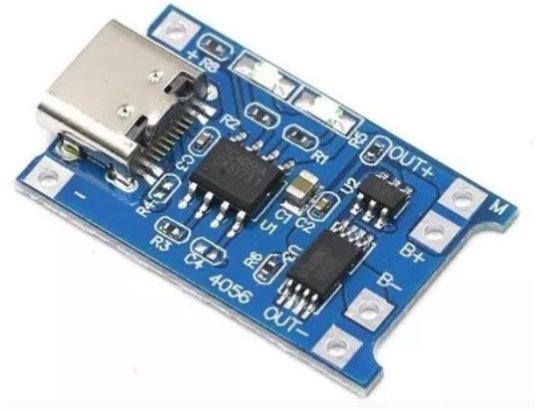


Figura 8.10.2. Cargador Batería De Litio Lipo 1a Tipo C 5v Tp4056.

Para cargar las baterías con ayuda de los paneles solares se decidió utilizar un Módulo Cargador Batería De Litio Lipo 1a Tipo C 5v Tp4056, como se ilustra en la figura 8.10.2, el módulo incluye características de protección y control de carga, lo que asegura una carga segura y eficiente de las baterías, esto incluye protección contra sobrecarga, descarga excesiva y cortocircuitos además de que es un módulo de bajo costo.



Figura 8.10.3. Baterías recargables li-ion de a 1500 mAh.

Las baterías ilustradas en la figura 8.10.3 son una buena opción debido a su alta densidad de energía, lo que permite que puedan almacenar una gran cantidad de energía en un tamaño compacto, cuentan con un ciclo de vida largo, mantienen su voltaje constante durante la mayor parte de sui ciclo de descarga, lo que da seguridad que el sistema no deje de funcionar en ningún momento, también cuentan con mecanismos de seguridad contra sobrecarga y descarga excesiva. se decidió usar 4 baterías en paralelo, por lo que se cuenta con una capacidad total de 6000 mAh.

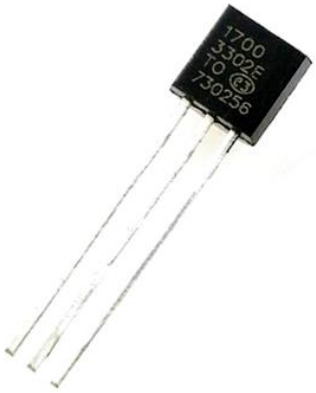


Figura 8.10.4. Low-dropout regulator MCP1700-3302E.

El MCP1700 es una buena opción debido a su bajo voltaje de caída (aproximadamente 178 mV a 250 mA), lo cual es crucial cuando se utilizan baterías, ya que maximiza el uso de la energía almacenada, cuenta con una alta eficiencia energética, estabilidad y precisión también tiene protección térmica y contra sobre corriente y es un elemento de bajo costo.

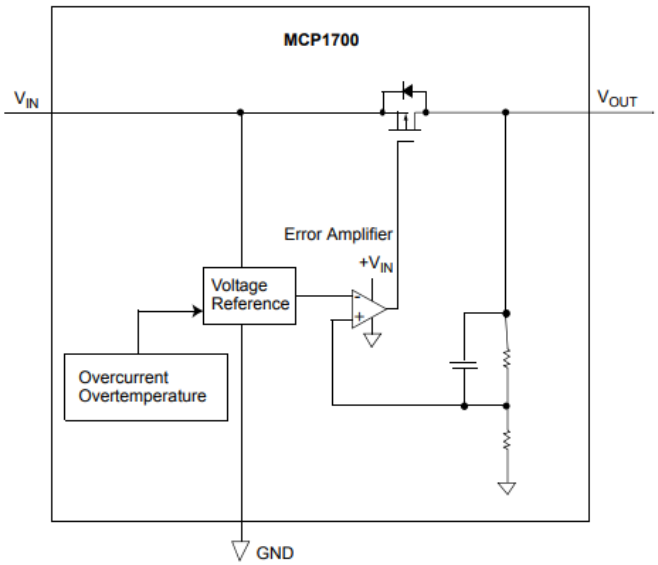


Figura 8.10.5. Diagrama de bloques funcional del MCP1700.

El diagrama cuenta con un voltaje de entrada que se aplica al terminal de entrada del regulador, la hoja de datos especifica que este voltaje puede ser mayor que el voltaje de salida deseado, pero no exceder el límite máximo especificado en la hoja de datos que es de aproximadamente 6,5 V, El transistor MOSFET se encarga de regular el voltaje de salida, controlando la cantidad de corriente que pasa a través de él para mantener el voltaje de salida estable, el amplificador de error compara una muestra del voltaje de salida con una referencia de voltaje interna muy precisa, si el voltaje de salida cae por debajo del valor deseado, el

amplificador de error ajusta el transistor de paso para permitir más corriente, aumentando el voltaje de salida, si el voltaje de salida es demasiado alto, el amplificador de error reduce la conducción del transistor de paso, disminuyendo la salida.

Con todos los elementos listos se realiza el esquemático del circuito propuesto en simulación, con la ayuda del programa EasyEDA.

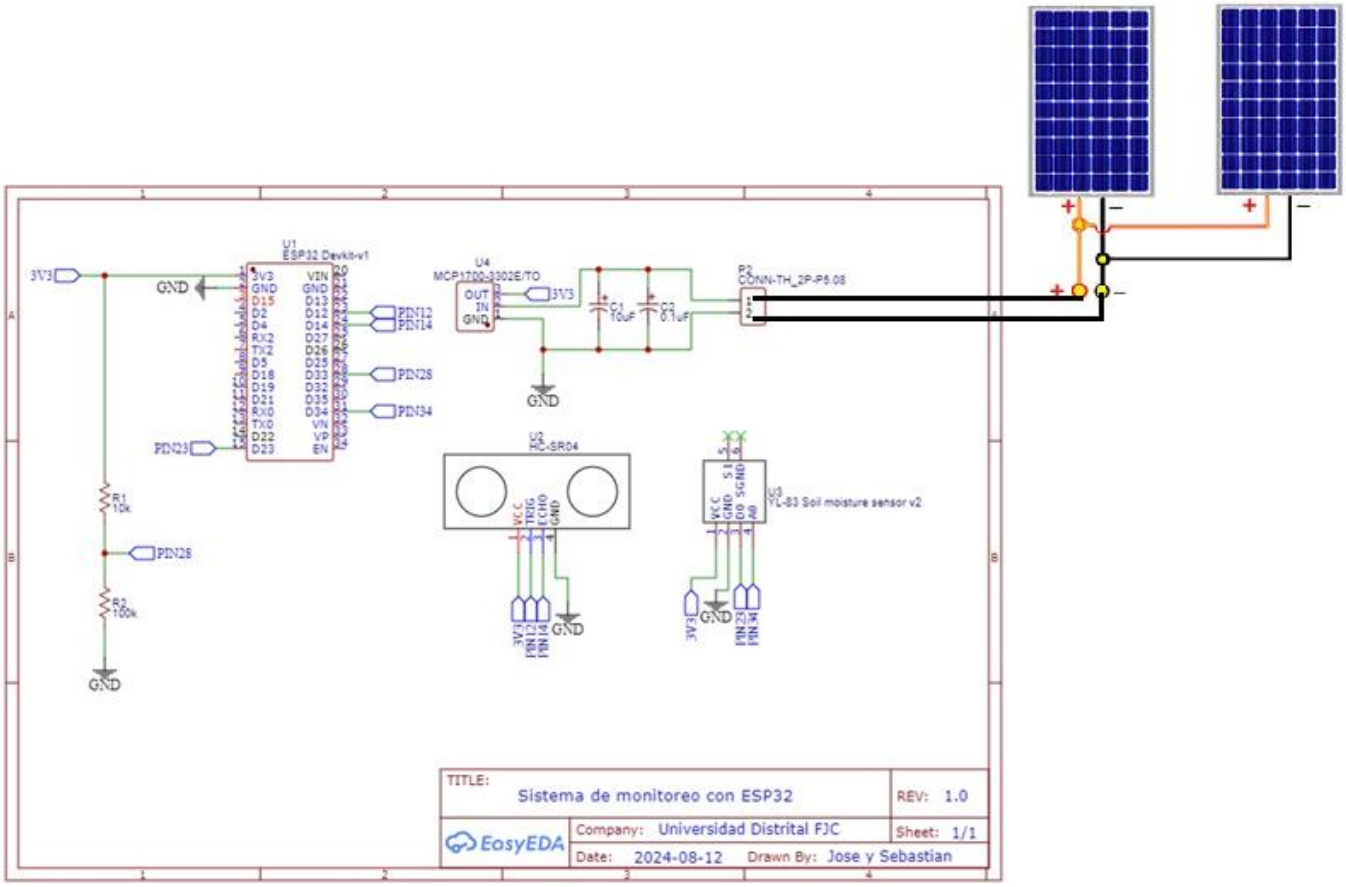


Figura 8.10.6. Esquemático del sistema de monitoreo.

El esquemático muestra los dos paneles solares conectados en paralelo como se comentó anteriormente para proporcionar una mayor corriente de salida, se tiene el módulo cargador de batería TP4056 cuyas el cual recibe la alimentación de los paneles solares y carga las baterías, la salida que proporciona el TP4056 con ayuda de las baterías proporciona energía al regulador el cual proporciona los 3,3 Voltios necesarios para operar la ESP32, el regulador cuenta con dos condensadores de 100 nF (cerámico) y 100 uF (electrolítico) para proteger el mismo, se tienen las cuatro baterías conectadas en paralelo para aumentar la capacidad total a 6000 mAh, los pines de la ESP32 están conectados a los sensores y demás periféricos que son el sensor ultrasonido HC-Sr04 y al sensor de lluvia YL-83 (este con ayuda de una regleta hembra de 4 slots), y al módulo SIM800L, finalmente se conectan 2 resistencias de 100k y 27k entre la tensión de las baterías y el pin D23 de la ESP32 con el fin de tener información del estado de carga de las baterías.

Con el fin de organizar el circuito para cuando se vaya montar en físico se realiza la organización del mismo mediante el asistente de diseño de PCB que ofrece EasyEDA, quedando como resultado la siguiente distribución de los componentes a excepción de los paneles solares.

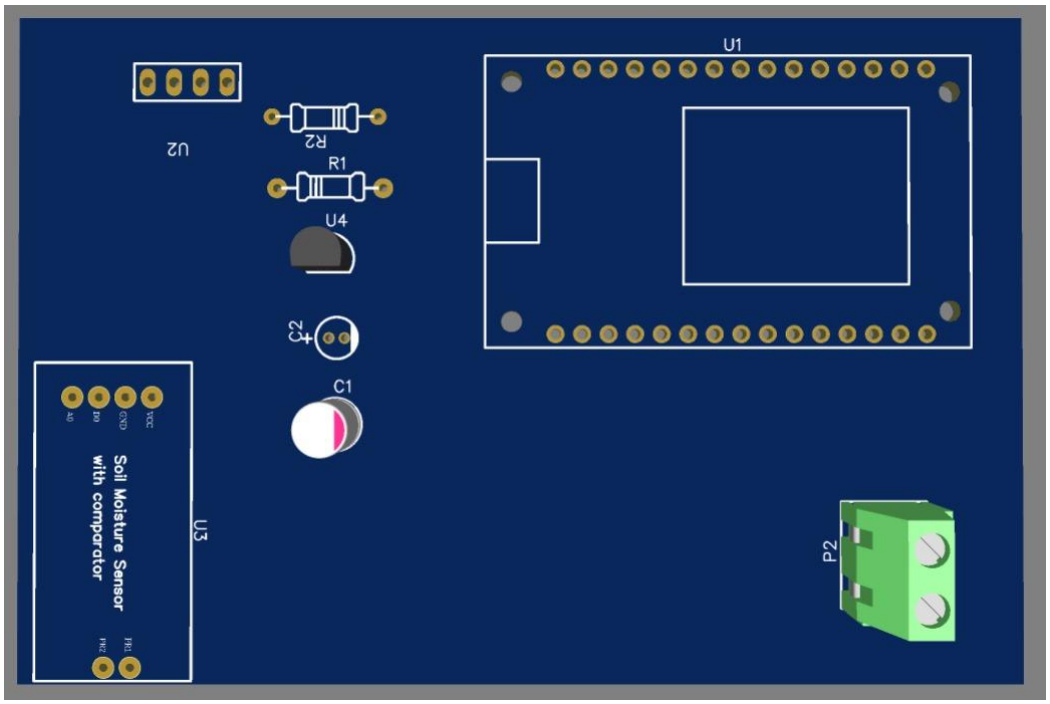


Figura 8.10.7. Vista del circuito en 3D.

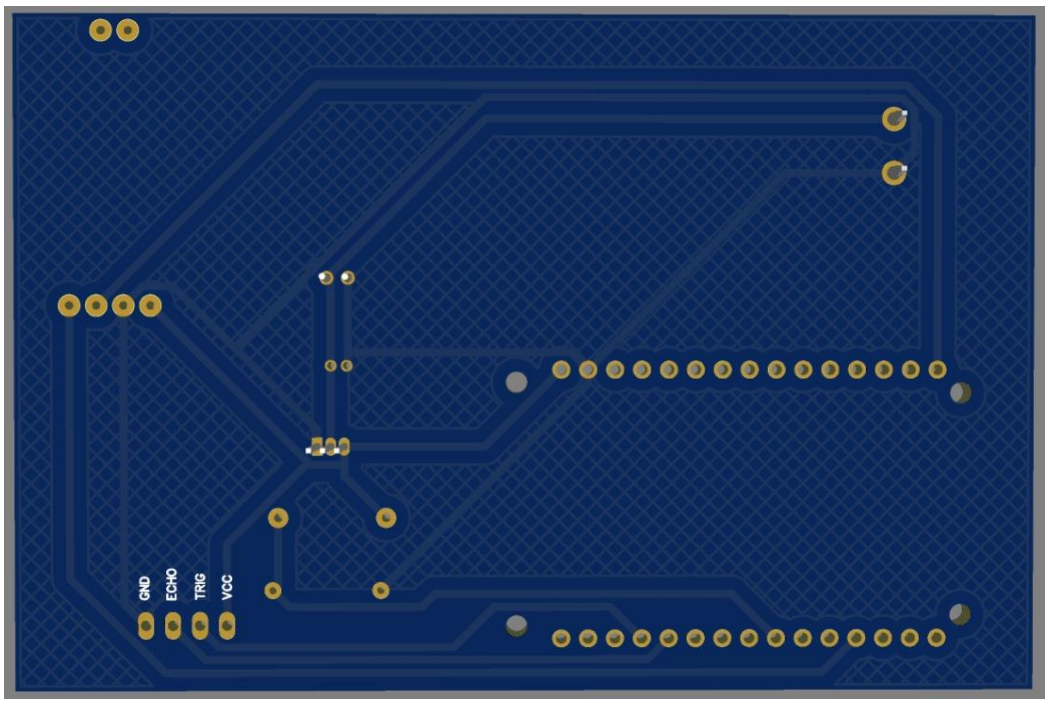


Figura 8.10.8. Vista de los caminos del circuito en 3D.

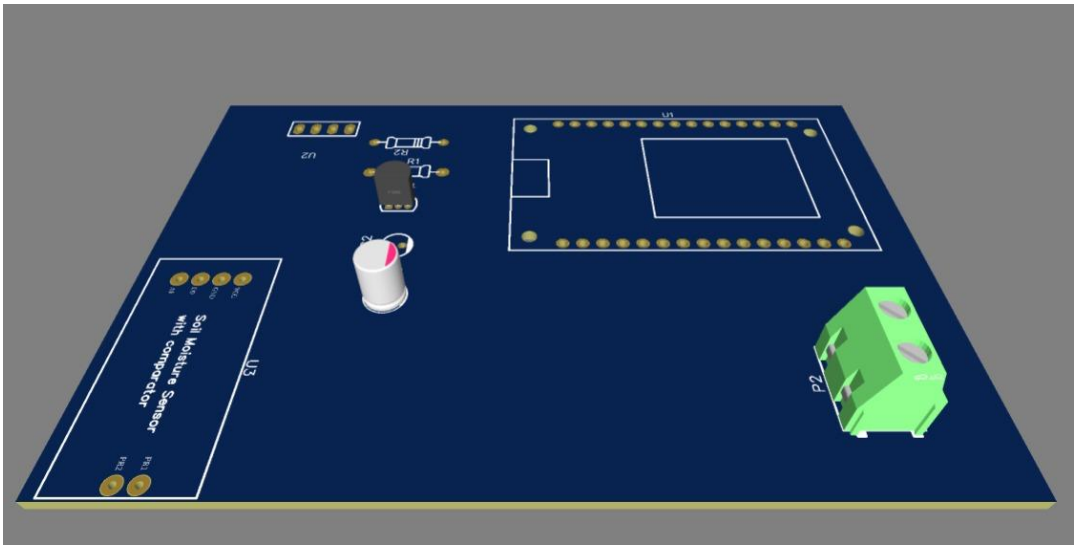


Figura 8.10.9. Vista del circuito en 3D desde otro ángulo.

El circuito impreso muestra el espacio que se designó para los componentes y los caminos que se hicieron en al parte de abajo para que todos los elementos tengan las debidas conexiones, la bornera sirve como entrada de la alineación del sistema, las pilas, que también son los mismos pines de salida del modulo TP4056.

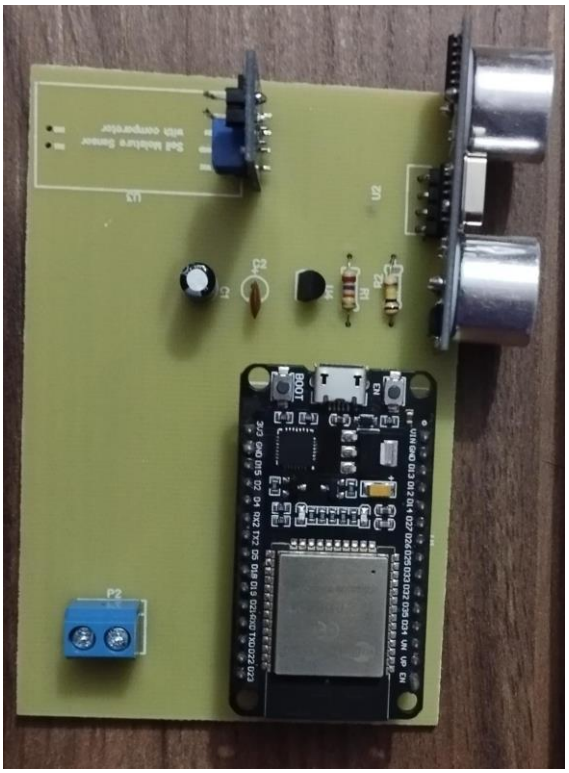


Figura 8.10.10. Vista superior del circuito impreso.

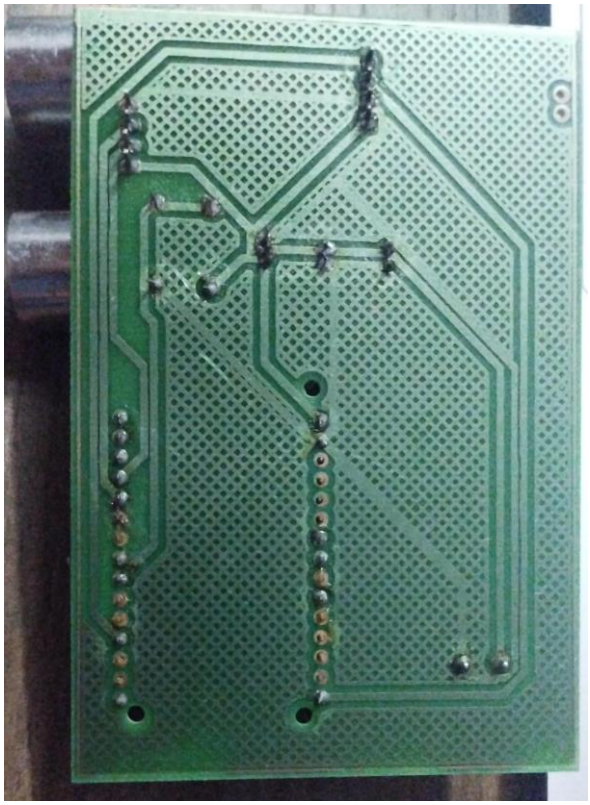


Figura 8.10.11. Vista de los caminos del circuito impreso.

Para verificar que el sistema puede operar durante largos periodos de tiempo, se hicieron pruebas con el circuito funcionando en una terraza, se dejó el circuito conectado y se estuvo monitoreando el estado de las baterías por 4 días.



Figura 8.10.12. Circuito ubicado en el exterior para hacer las pruebas de autonomía.

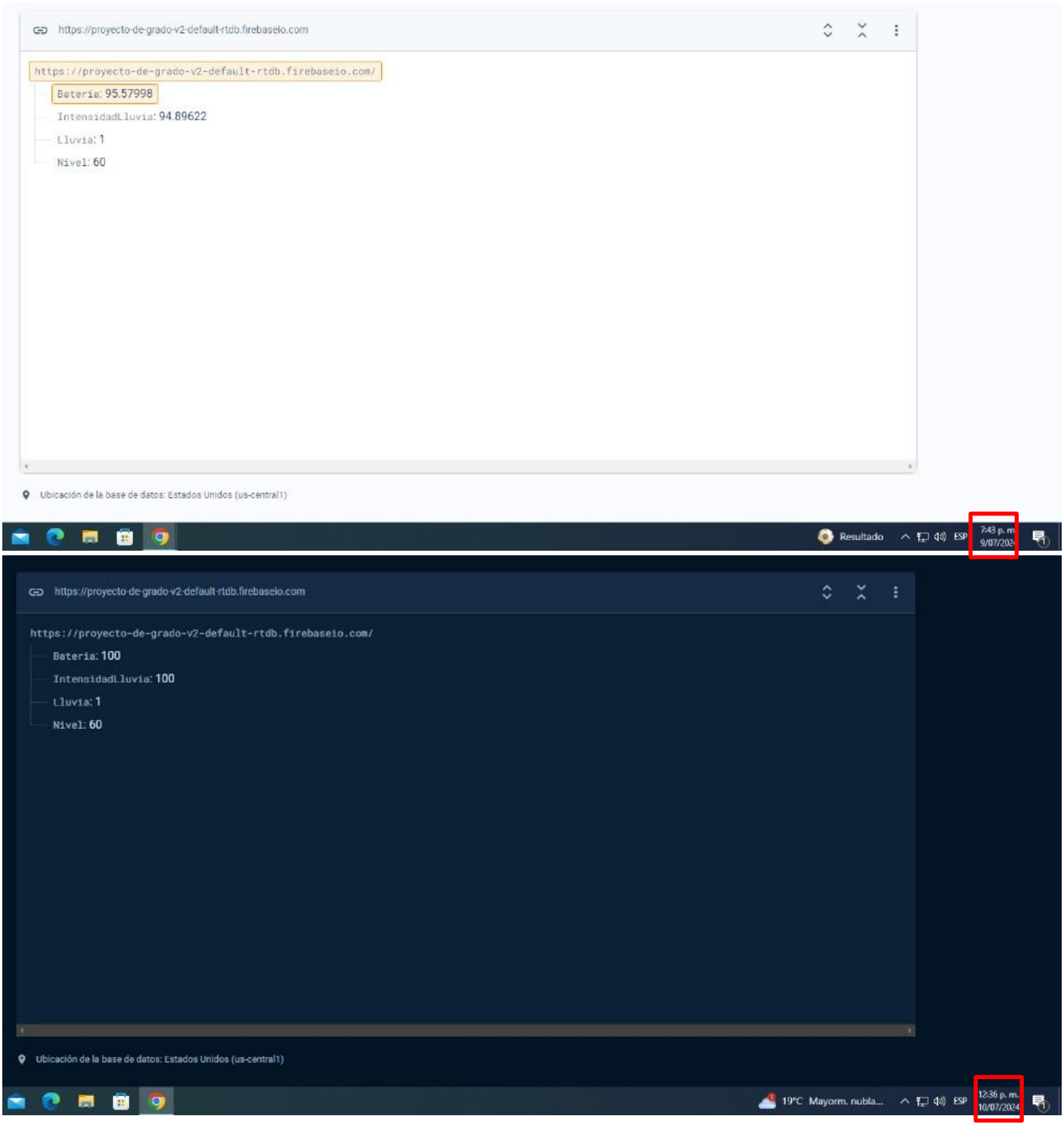


Figura 8.10.13. Comportamiento de la batería desde la noche hasta el mediodía del otro día.

Se verifico la batería el día 9 de julio a las 7:43 pm, se observa que va descargándose lentamente pasando del 100% con las baterías recién compradas a un 95 %, posteriormente se comprueba el nivel de batería al día siguiente a las 12 y 36 pm, se observa que las baterías lograron cargarse en la mañana llegando nuevamente al 100%.

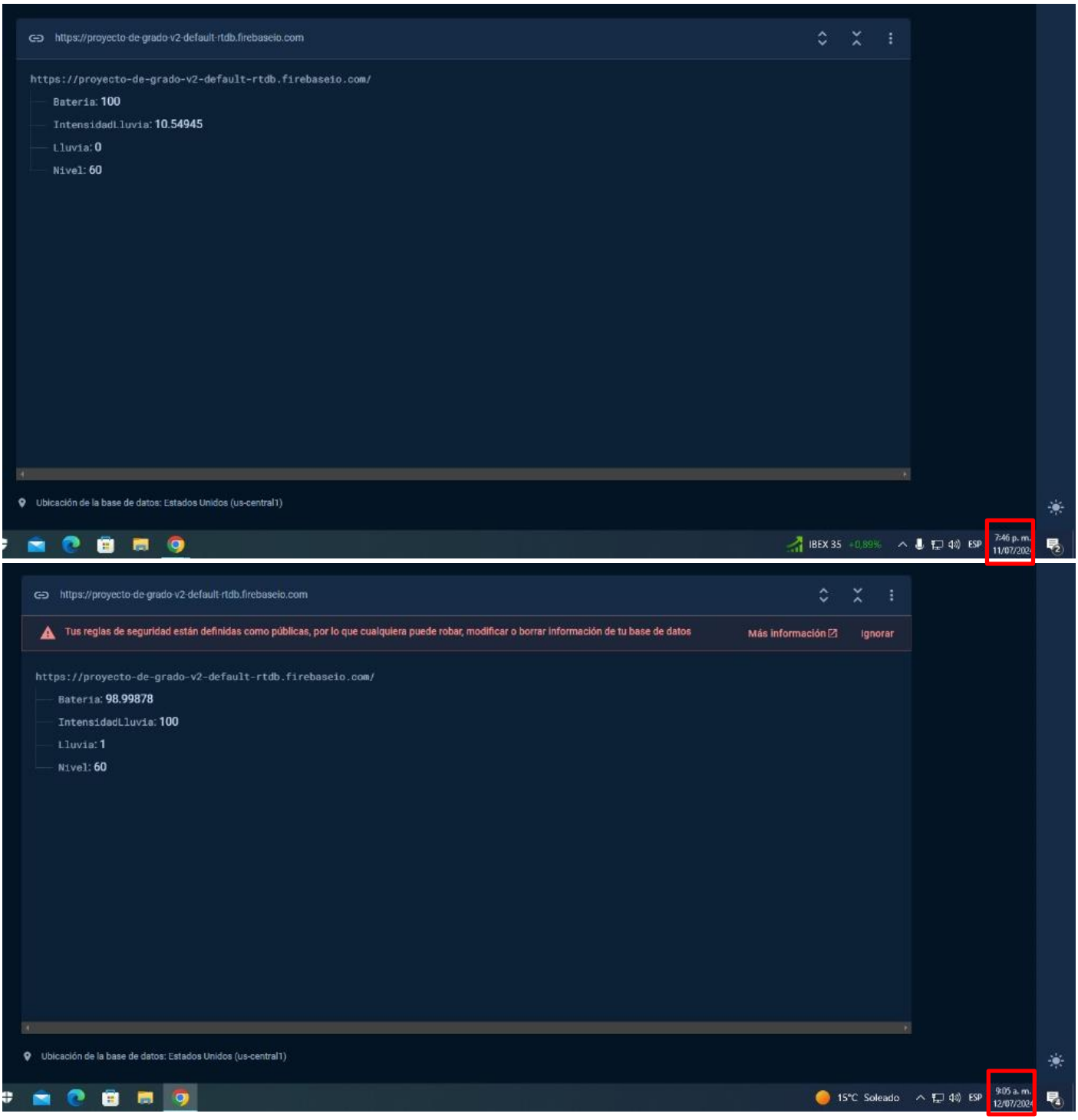


Figura 8.10.14. Comportamiento de la batería en los 2 últimos días.

Se observa que para el 11 de julio, a las 7:45 pm, la batería se encuentra cargada en su mayoría, y para el otro día en la mañana, posiblemente después de haberse descargado un poco, a hora de las 9:05 am ya se encuentra nuevamente casi al 100% de su capacidad.

Una vez se tiene el diseño le circuito de monitoreo completo, hace falta diseñar un contenedor en el cual se pueda guardar el circuito para que este protegido contra las condiciones climáticas (a excepción del sensor de lluvia que necesita estar afuera para efectos de la medición de la intensidad de lluvia y los paneles solares para recibir la energía proveniente del sol), por lo cual se plantea lo siguiente:

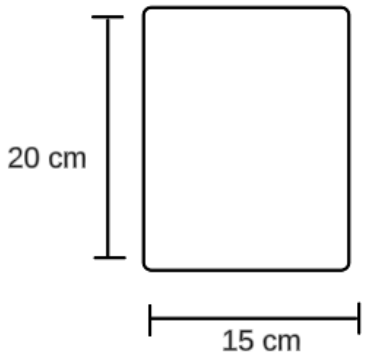


Figura 8.10.15. Pared frontal de la caja.

La figura 8.10.15 muestra las medidas de la pared frontal de la caja, se usaron 2 como la de la figura, la segunda seria la tapa de la caja.

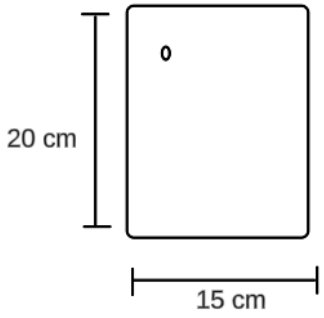


Figura 8.10.16. Pared con agujero para los cables de los paneles solares.

La figura 8.10.16 muestra la pared que está más cerca a la porta pilas contraria a la pared de la figura anterior (8.10.12), la cual cuenta con un agujero por el cual pasan los cables que van a los paneles solares.

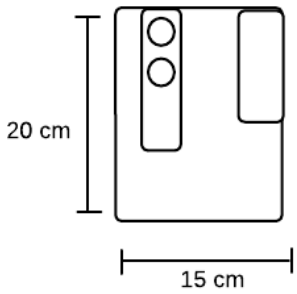


Figura 8.10.17. Base de la caja.

En la figura 8.10.17 se muestra la base de la caja la cual cuenta con 2 añadiduras que generan un desnivel sobre las cuales se coloca el circuito impreso que contiene el sensor de ultrasonido en la de la izquierda y la de la derecha en la que se va a colocar el porta pilas que está siendo alimentado por los paneles solares.

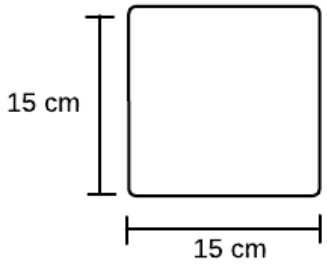


Figura 8.10.18. Una de las paredes de la caja.

La figura 8.10.18 muestra la pared a la que está más cerca al agujero de dónde vienen los cables de los paneles solares.

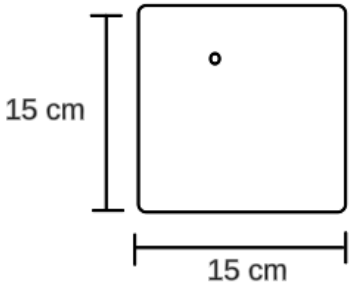


Figura 8.10.19. Pared lateral izquierda con abertura para el sensor de lluvia.

La figura 16 muestra la pared que se encuentra al lado contrario de la pared de la figura 8.10.19

Para tener una mejor visión de la caja que se quiere hacer, se realiza un diseño en 3D con ayuda de Tinkercat, el cual permite tener una mejor aproximación de lo que se quiere hacer y poder mandar a hacer un corte láser con piezas de acrílico, el cual es un material resistente al medio ambiente, perfecto para dejar en la intemperie.

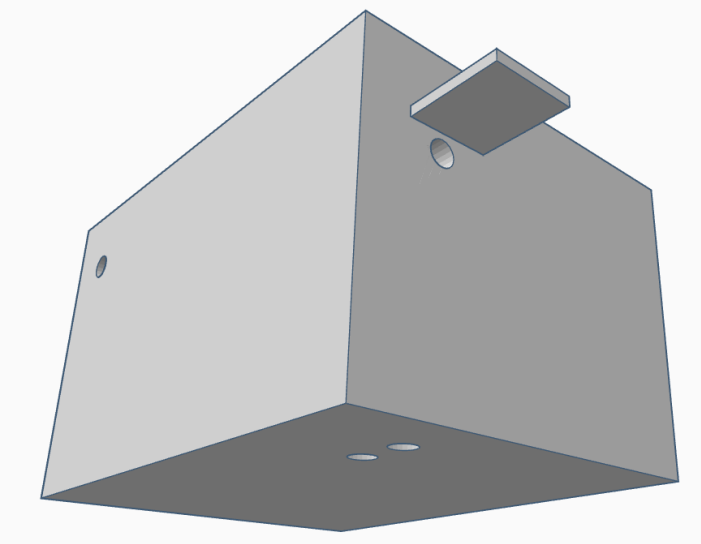


Figura 8.10.20. Diseño en 3D de la caja (vista inferior).

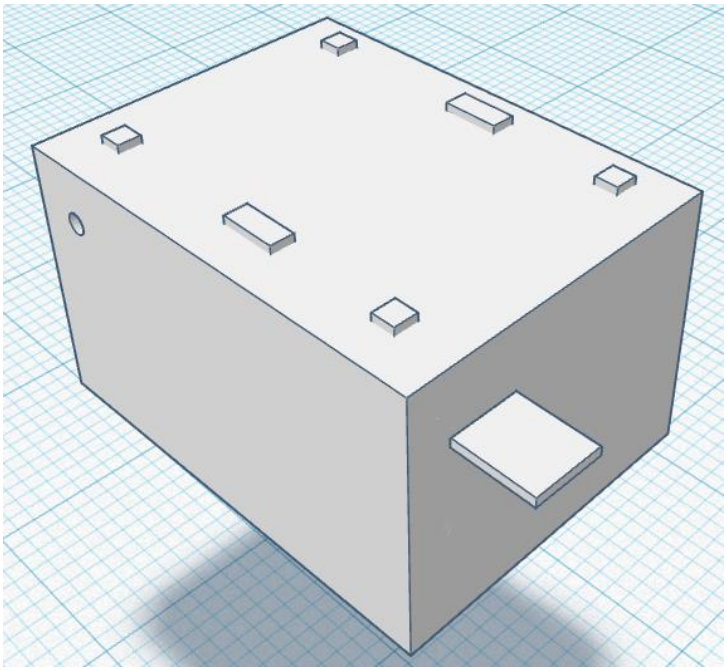


Figura 8.10.21. Diseño en 3D de la caja (vista superior).

El diseño del contenedor se realizó a partir de la necesidad de mantener el circuito electrónico protegido frente al ambiente, las medidas que se escogieron fueron para que el circuito impreso tuviera suficiente espacio y también contara con el espacio para las baterías y cualquier elemento extra que se necesitara en futuros trabajos, su ancho es

debido a que los paneles solares van a estar ubicados en la parte superior de la caja, también cuenta con una rejilla para colocar el sensor de lluvia, finalmente el contenedor quedo de la siguiente manera:

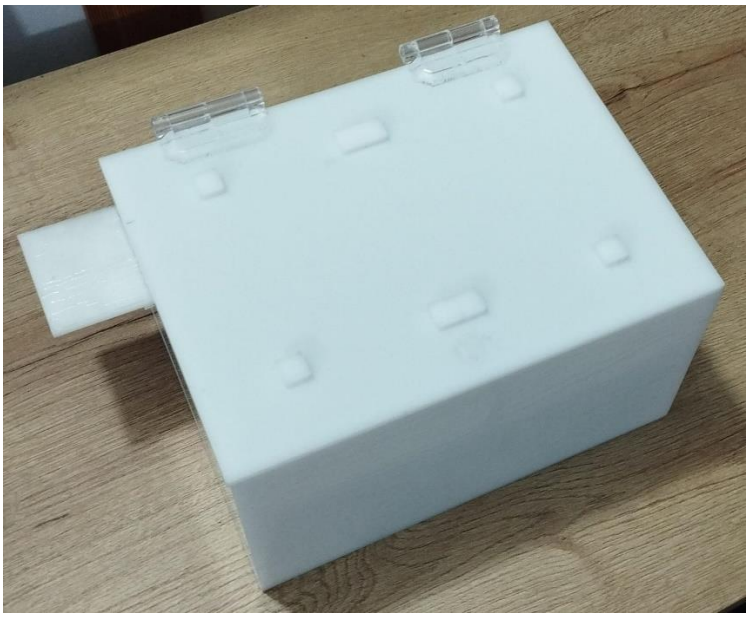


Figura 8.10.22. Caja en físico (vista superior).



Figura 8.10.23. Caja en físico (vista superior desde otro ángulo).



Figura 8.10.24. Caja en físico (vista inferior).



Figura 8.10.25. Caja en físico (vista del interior).

8.11. Circuito de alimentación para el sistema de alarma físico

Debido a que se planteó utilizar una sirena para notificar en caso de que se presente una alerta roja por medio del sistema de monitoreo, este sistema de alarma también necesita estar operativo en todo momento, por lo que se plantea un sistema similar al del sistema de monitoreo, pero hay que tener en cuenta que la parte de potencia para este circuito es diferente ya que se necesita alimentar una sirena de 6-12 voltios, por lo que se van a utilizar los siguientes componentes para este caso:



Figura 8.11.1. Panel Solar 12v 125 mA Fotovoltaico.

Se escogió el panel solar ilustrado en la figura 8.11.1, ya que es ideal para cargar las baterías Li-ion en serie, es un elemento con buena eficiencia y bajo costo siendo perfecto para el este trabajo.



Figura 8.11.2. Modulo cargador de pilas.

Se decidió utilizar el módulo ilustrado en la figura 8.11.2, primero por sus protecciones criticas para asegurar el alto tiempo de uso y la seguridad de las baterías, la configuración 3s significa que 3 baterías de 3,7 voltios se conectan en serie, proporcionando un voltaje de 12,6 voltios cuando las baterías estén totalmente cargadas, tensión adecuada para operar la sirena cuando sea necesario.



Figura 8.11.3. Regulador lineal de “Low-Dropout”- LM1117T-3.3.

El regulador LM1117T - 3.3Ves un dispositivo diseñado para proporcionar una salida estable de 3.3 V a partir de una tensión de entrada ligeramente superior, con un voltaje de dropout típico de 1.1 V.



Figura 8.11.4. Transistor Npn TIP31C.

Para el manejo de la sirena, se optó por utilizar el transistor ilustrado en la figura 8.11.4 dado que es suficiente para manejar la carga de la sirena y es bastante económico.



Figura 8.11.5. Sirena 12 V 10W (HPS102).

La sirena ilustrada en la figura 8.11.5 cuenta con un consumo de energía que se adapta a los elementos previamente discutidos, tiene una alta intensidad de sonido, durabilidad, los cuales son factores que aseguran que el sistema de alarma sea efectivo, fiable y eficiente energéticamente.

Con todos los elementos listos se realiza el esquemático del circuito propuesto en simulación, con la ayuda del programa EasyEDA.

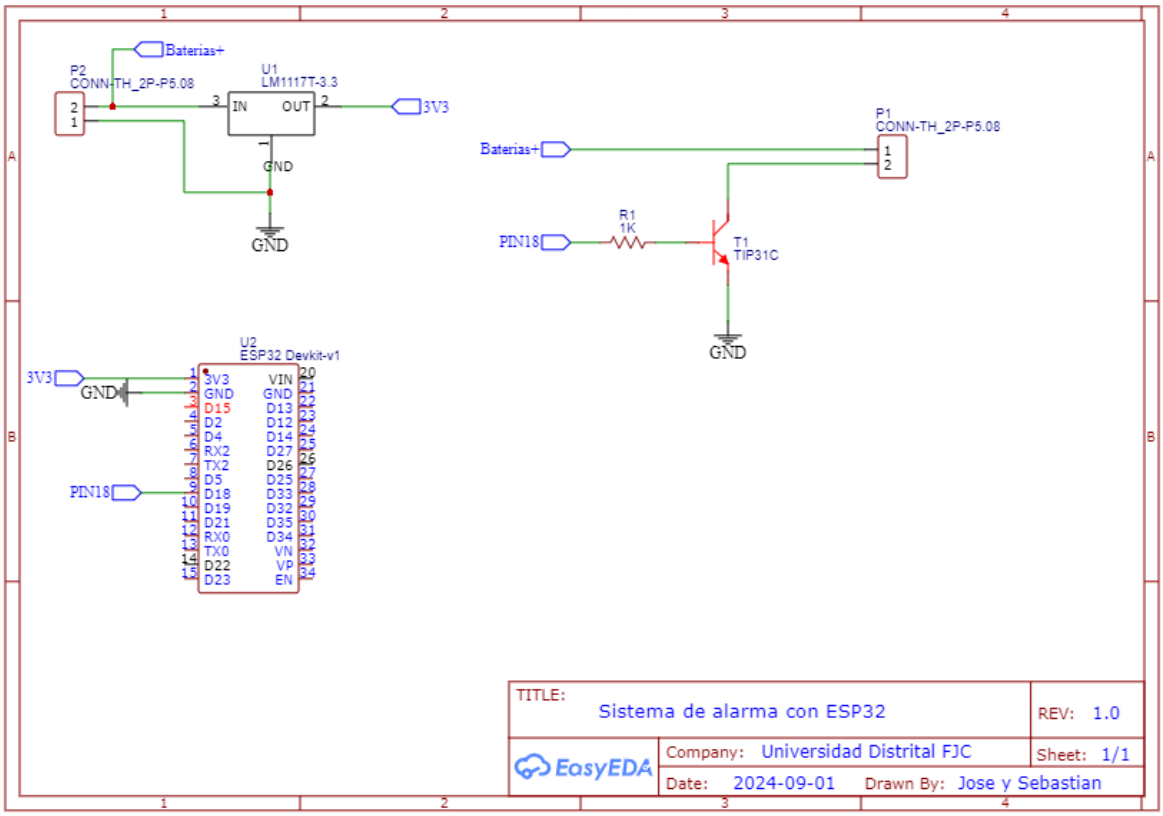


Figura 8.11.6. Esquemático del sistema de alarma.

El esquemático muestra el panel solar de 12 V, 125 mA, conectado a los terminales positivo y negativo del módulo de carga BMS para baterías Li-ion, las salidas del módulo van a las baterías configuradas en serie, se tienen 3 baterías de 3.7 V conectadas en serie para proporcionar un voltaje total de 12.6 V cuando estén totalmente cargadas, el regulador recibe como entrada la tensión de las baterías y entrega a la salida un voltaje de 3,3 V para alimentar la ESP32, la esp32 envía una señal por el pin D18 para poner en funcionamiento o parar el funcionamiento del transistor NPN TIP31C y así activar o desactivar la sirena, nos e tiene en cuenta un módulo de internet como el SIM800L para este caso debido a que la sirena debe encontrarse en un lugar poblado en donde se debería contar con acceso a internet por medio del módulo Wi-Fi de la ESP32.

Con el fin de organizar el circuito para cuando se vaya montar en físico se realiza la organización del mismo mediante el asistente de diseño de PCB que ofrece EasyEDA, quedando como resultado la siguiente distribución de los componentes a excepción del panel solar.

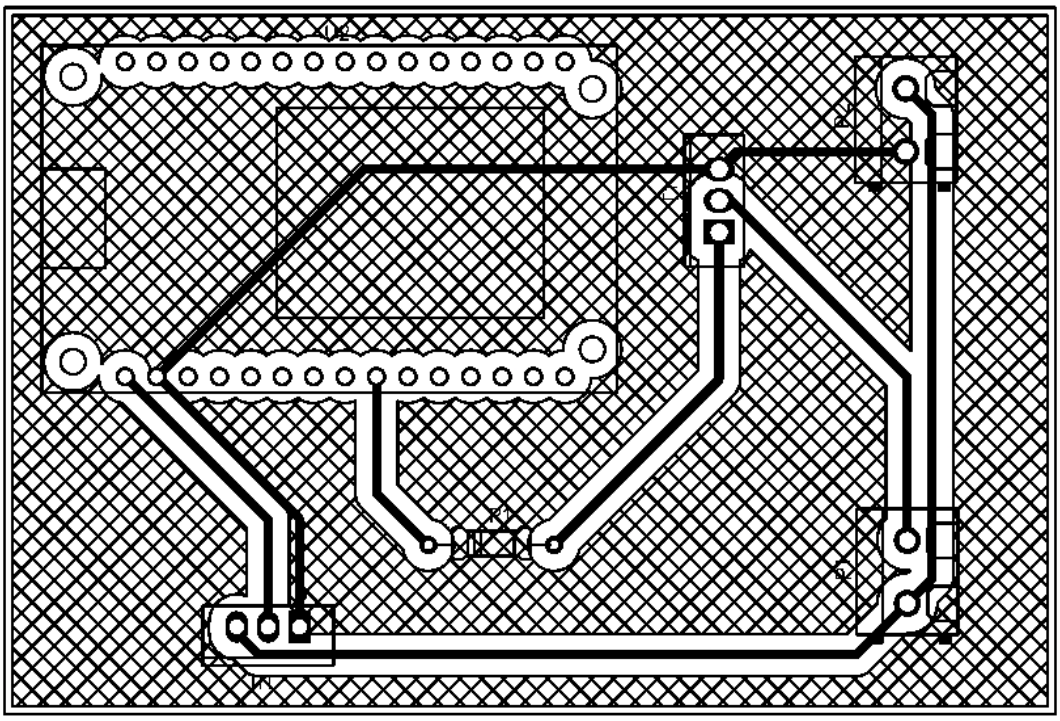


Figura 8.11.7. Vista del circuito en la herramienta de diseño de PCB.

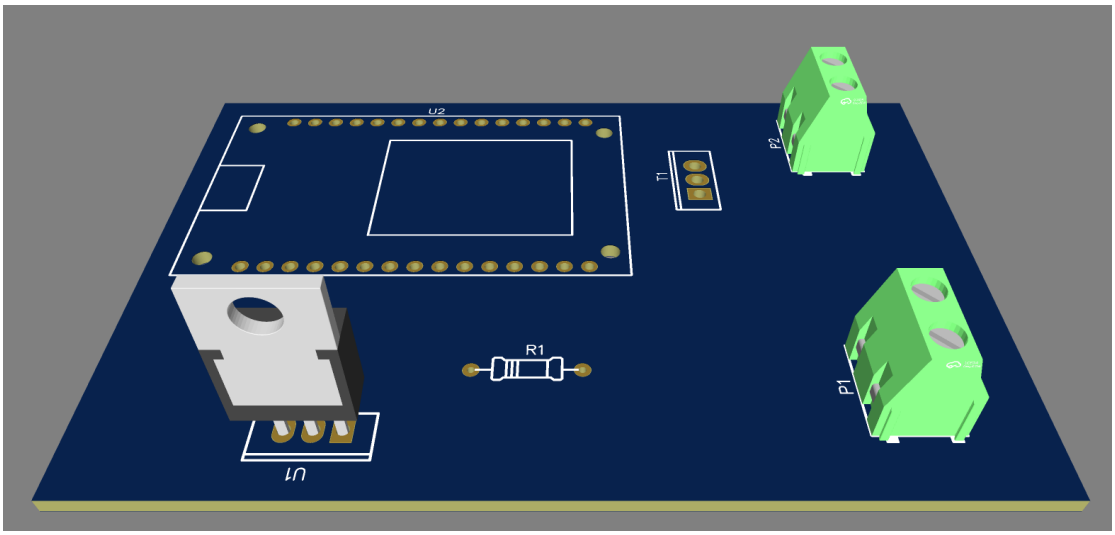


Figura 8.11.8. Vista del circuito en 3D.

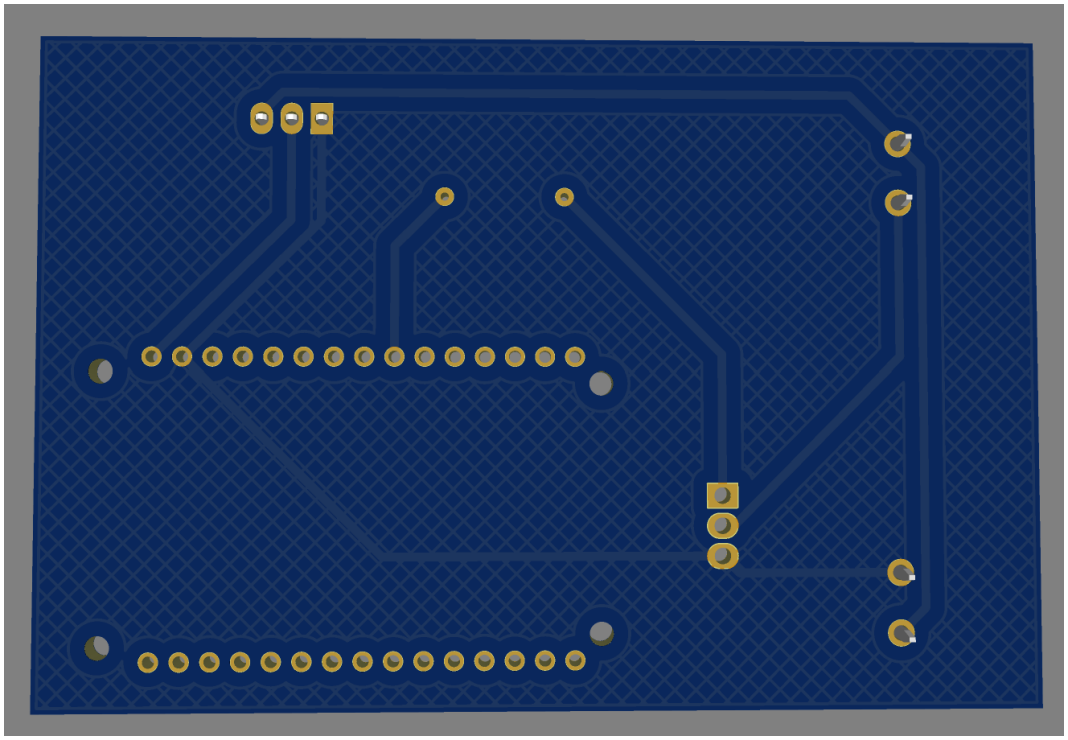


Figura 8.11.9. Vista de los caminos del circuito en 3D.

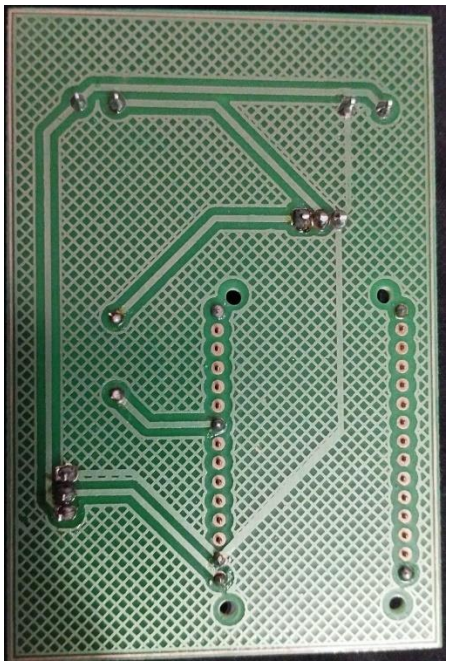


Figura 8.11.10. Vista de los caminos del circuito impreso.

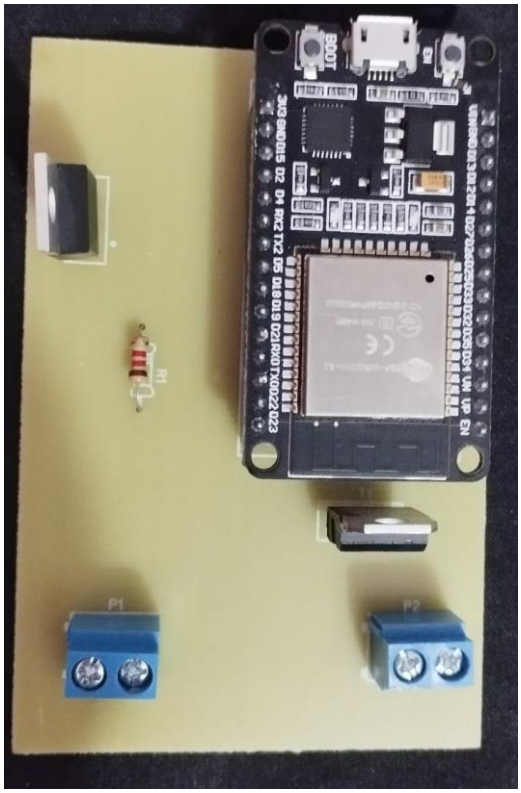


Figura 8.11.11. Vista superior del circuito impreso.

Una vez se tiene el diseño le circuito de alarma completo, hace falta diseñar un contenedor en el cual se pueda guardar el circuito para que este protegido contra las condiciones

climáticas (a excepción del panel solar para recibir la energía proveniente del sol), por lo cual se plantea lo siguiente:

DISEÑO DE LA CAJA:

Con el diseño del circuito impreso para el sistema de alarma físico pasamos a realizar el diseño de un contenedor el cual contenga el circuito ante las adversidades climáticas, tenga en la parte superior un panel solar para que sea independiente energéticamente y una salida para los cables conectados a la alarma por lo que las diferentes paredes de la caja son las siguientes:

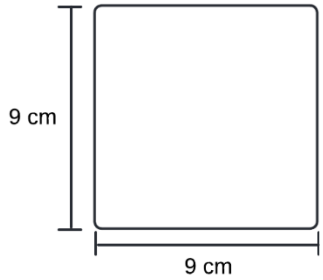


Figura 8.11.12. Pared frontal, derecha y superior de la caja del sistema de alarma.

De la figura 8.11.12 se muestran las medidas de la pared frontal, lateral derecha y la tapa de la caja, se usaron tres de estas para la implementación.

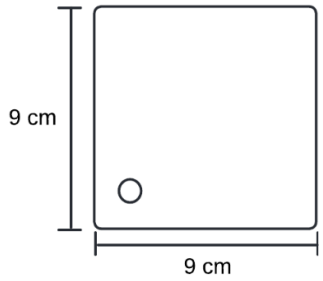


Figura 8.11.13. Pared izquierda de la caja del sistema de alarma.

De la figura 8.11.13 muestra la parad de lado izquierdo el cual cuenta con una salida por la cual pueden ingresar los cables de la alarma a las borneras del circuito.

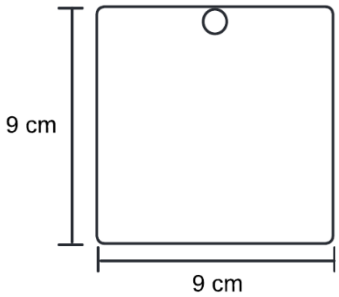


Figura 8.11.14. Pared posterior de la caja del sistema de alarma.

De la figura 8.11.14 muestra la pared de la parte posterior la cual cuenta con un espacio para hacer ingreso de los cables del panel solar, solo se tiene una de esta para su implementación.

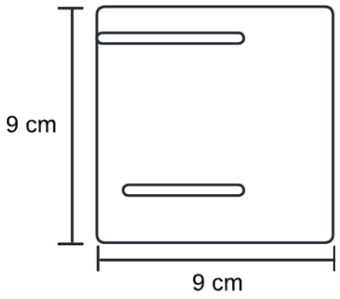


Figura 8.11.15. Base de la caja del sistema de alarma.

La figura 8.11.15 muestra la pared inferior y base de la caja la cual cuenta con dos elevaciones las cuales se usan como soporte del circuito impreso el cual se ajusta colocando unos tornillos sobre una de estas elevaciones, solo se tiene una de estas para la implementación de la caja.

Se realiza un diseño en 3D con ayuda de Tinkercat, el cual permite tener una mejor aproximación de lo que se quiere hacer y poder mandar a hacer un corte láser con piezas de acrílico, de igual material que la caja anterior perfecto para dejarlo a la intemperie:

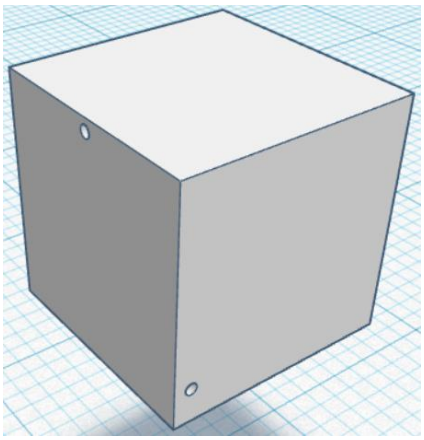


Figura 8.11.16. Diseño 3D de la caja visión lateral izquierda.

El diseño para el contenedor se realizó a partir de la necesidad proteger el circuito impreso del ambiente, y que este sea capaz de ser ubicado en el exterior, las medidas que se escogieron son para que el circuito se encuentre lo más compacto posible contando con un banco de baterías y las entradas de los cables, finalmente el contenedor quedo de la siguiente forma:

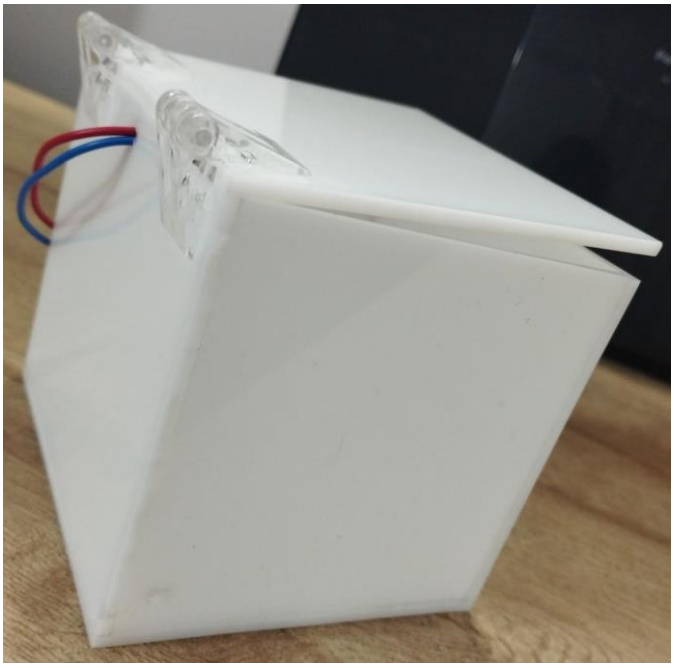


Figura 8.11.17. Caja en físico (vista desde el lateral izquierdo).



Figura 8.11.18. Caja en físico (vista superior sin la tapa).

8.12 Base de datos

Con la solución de conectividad a internet implementada para el sistema de monitoreo con ayuda del módulo SIM800L, ahora se va a enviar la información recolectada de los sensores y el estado de la batería del sistema a la base de datos, que para este trabajo va a ser Firebase, específicamente se va a utilizar la característica **Realtime Database** la cual es una base de datos NoSQL alojada en la nube que permite almacenar y sincronizar datos entre sus aplicaciones de manera instantánea, aquí hay algunas características importantes de esta funcionalidad de Firebase:

- **Sincronización en tiempo real:** Los datos se sincronizan en tiempo real, lo cual es clave en nuestro trabajo, dado que cualquier cambio en la base de datos se releja inmediatamente en todas las aplicaciones conectadas a la base de datos.
- **Estructura de datos JSON:** La base de datos almacena a información en un formato JSON, lo que facilita la manipulación de los datos jerárquicos y anidados, esto brinda un acceso directo a cualquier nodo de la información, facilitando la lectura y escritura de datos específicos, lo que es bastante útil para hacer pruebas con la aplicación sin que el sistema de monitoreo tenga que estar en funcionamiento.
- **Persistencia de datos:** Los datos se almacenan localmente en el dispositivo cuando no hay conexión y se sincronizan automáticamente con el servidor cuando la conexión se restablece.
- **Seguridad y reglas de acceso:** Permite definir reglas de seguridad basadas en el contenido y la autenticación del usuario.
- **Escalabilidad y desempeño:** escala automáticamente a medida que la aplicación crece, manejando grandes volúmenes de datos y usuarios, cuenta con baja latencia y alto rendimiento.

Inicialmente se observa la conexión a Firebase por medio de la ESP32:

```
#include <FirebaseESP32.h>
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
#define API_KE
#define DATABASE_URL "https://proyecto-de-grado-v2-default-rtdb.firebaseio.com"
#define USER_EMAIL "s
#define USER_PASSWORD "

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
```

Código 7. Bibliotecas, contantes y objetos requeridos.

Se incluyen las bibliotecas necesarias para la conexión y manipulación de datos en Firebase, la generación de tokens de autenticación y el uso de funciones adicionales para trabajar con Firebase, también se incluyen los datos necesarios para poder acceder a la base de datos y se inicializan los objetos para poder usar las operaciones de lectura y escritura de datos en la base de datos.

```
16 void setup() {
17   Serial.begin(115200); // Inicia la comunicación serial
18
19
20   // Configuración de Firebase
21
22   Serial.printf("Firebase client", FIREBASE_CLIENT_VERSION);
23   config.api_key = API_KEY;
24   auth.user.email = USER_EMAIL;
25   auth.user.password = USER_PASSWORD;
26   config.database_url = DATABASE_URL;
27   config.token_status_callback = tokenStatusCallback;
28   Firebase.begin(&config, &auth);
29
30 }
```

Código 8. Configuración de Firebase.

Se imprime un mensaje en el monitor serial para indicar la versión del cliente de Firebase que se está utilizando, posteriormente se asignan las credenciales de acceso a la base de datos para su autenticación, se asigna la URL de la base de datos, y se asigna una función callback que maneja los eventos relaciones al estado del token de autenticación para finalmente inicializar la conexión a Firebase utilizando la configuración y detalles de autenticación previamente mencionados.

```
32 void loop() {
33
34 // Verificar y reconectar a Firebase si es necesario
35   if (!Firebase.ready()) {
36     reconnectFirebase();
37   }
38
39   Firebase.setFloat(fbdo,"Nivel", distance);
40   Firebase.setFloat(fbdo,"Bateria", batteryLevel);
41   Firebase.setFloat(fbdo,"Lluvia", digitalValue);
42   Firebase.setFloat(fbdo,"IntensidadLluvia", rainIntensityPercentage);
43
44   delay(2000);
45 }
```

Código 9. Verificación de conexión y envió de información a la base de datos.

En el loop se verifica si la conexión a Firebase está estable, de lo contrario vuelve a ejecutar la conexión, y se realiza la actualización de datos en Firebase cada 2 segundos, se transmite la información que se obtiene por medio del sensor ultrasonido, el sensor de lluvia y el estado de la batería del sistema.



Figura 8.12.1. Realtime Database del proyecto.

Una vez creada la base da tos y poniendo a funcionar el sistema de monitoreo se comprueba que efectivamente se está enviando correctamente la información de monitoreo a la base de datos.

Con los datos ya almacenados en la nube, se programa la esp32 que trabaja con la sirena para acceder a la información:

```
60  void loop() {
61
62    if (Firebase.ready()) {
63      // Leer el valor de "nivelDeAlerta" desde Firebase
64      if (Firebase.getFloat(fbdo, "/nivelDeAlerta")) {
65        float nivelDeAlerta = fbdo.floatData();
66        Serial.print("Nivel de Alerta: ");
67        Serial.println(nivelDeAlerta);
68
69
70  // Verificar si el nivel es mayor o igual a 16 (alerta roja)
71        if (nivelDeAlerta == 3) {
72          digitalWrite(outputPin, HIGH); // Activar el pin de salida
73          Serial.println("Pin de salida activado. Sirena encendida.");
74        } else {
75          digitalWrite(outputPin, LOW); // Desactivar el pin de salida
76          Serial.println("Pin de salida desactivado. Sirena apagada.");
77        }
78
79
80
81      delay(1000);
82  }
```

Código 10. Codificación parta la lectura de datos de firebase.

Finalmente se lee el valor de nivelDealerta en firebase, y si esté es igual a 3 “alerta roja activa”, pone a funcionar la sirena con ayuda del circuito discutido anteriormente.

8.13 Aplicación

Lo primero definir que funcionalidades va a tener la aplicación, a continuación, se describen las 4 pestañas propuestas para la aplicación y sus funcionalidades:

- **Primera pestaña (información del nivel del rio y del estado del clima):** La primera pestaña cumple la función de mostrar la información referente al nivel del rio, asi mismo con el nivel de alerta activo en el que se encuentre, adicional se informará si en la zona monitoreada se están presentado lluvias o no.
- **Segunda pestaña (Gráficas):** La segunda pestaña mostrara dos gráficas, una mostrando el historial del nivel del rio, la segunda grafica indicara la cantidad de veces que cada nivel de alerta se ha activado.

- **Tercera pestaña (Geolocalización con alerta de área afectada):** La tercera pestaña es la encargada de mostrar un mapa, el cual estará centrado en la zona de monitoreo, adicionalmente este mostrará una rea roja cuando se presente el nivel de alerta rojo, indicando la posible área de afectación.
- **Cuarta pestaña (Estado del tiempo):** La última pestaña mostrara información adicional recolectada de sistemas de monitoreo del clima cercano al área, indicando algunas variables de interés del clima.

Se decide utilizar MIT App inventor, el cual presenta una codificación en bloques, por lo que es fácil de implementar y debido a que no se necesitan funcionalidades muy complejas, es perfecta para desarrollar la aplicación. [26]

Se diseña primero la página de carga al ingresar a la aplicación, esto con ayuda de la interfaz de MIT app inventor que nos permite organizar los elementos de la aplicación, finalmente la pantalla de inicio queda de la siguiente manera:



Figura 8.13.1. Pantalla de carga de la aplicación.

Con el la pantalla configurada visualmente, se pasa a programar con los bloques.

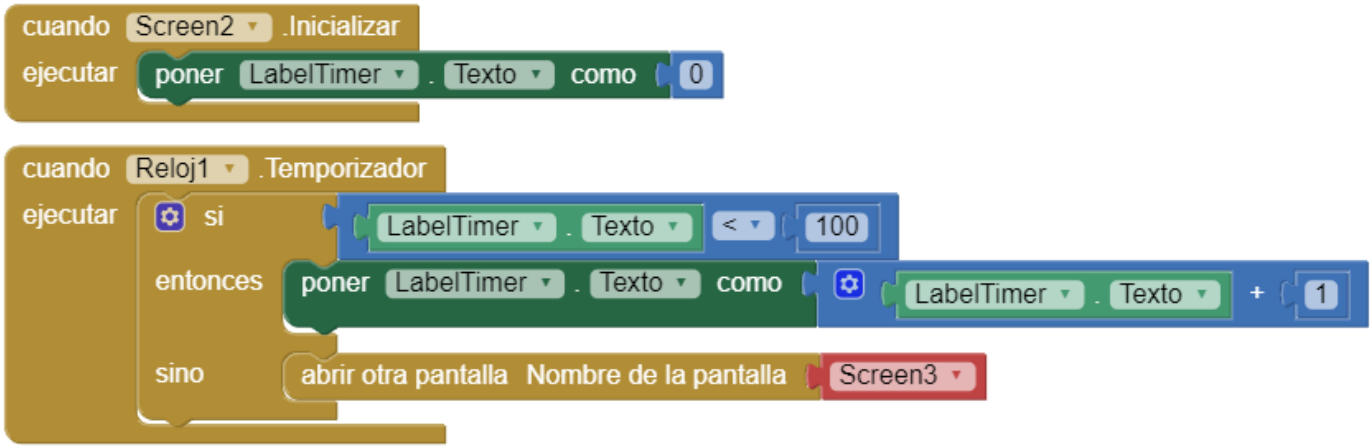


Figura 8.13.2. Configuración de los bloques para el funcionamiento de la pantalla de carga.

En la figura 8.13.2, el primer bloque se encarga de inicializar el número que indica la carga de la pantalla en 0, el segundo bloque configura el reloj y la carga del número que indica la carga de la pantalla (en este caso labelTimer) para que cuando esté llegue a 100 % haga el cambio a la siguiente pantalla (Screen3).

Posteriormente se inicia con la primera pestaña de la aplicación, que contendrá la información del nivel del agua, estado del nivel de alerta y estado del clima, se propone la siguiente interfaz de visualización:

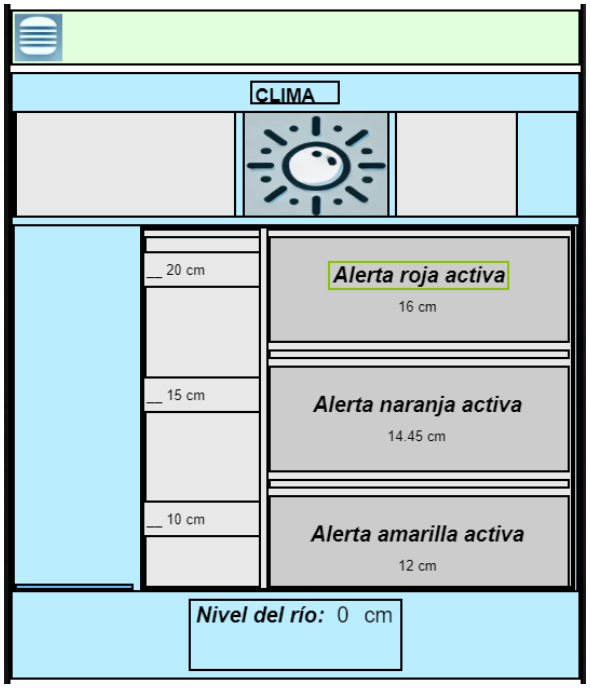


Figura 8.13.3. Pantalla de inicio de la aplicación (vista de diseño).

En la figura 8.13.3 se ve la pantalla de inicio en la cual se aprecia el nivel de rio de manera numérica y con un medido vertical, adicionalmente cuando se activa un nivel de alerta, se puede apreciar con ayuda de los bloques de cada nivel correspondiente iluminándose con el color correspondiente, y finalmente una imagen haciendo referencia al clima, simplemente para indicar se en lugar de monitoreo hay presencia de lluvia o no.

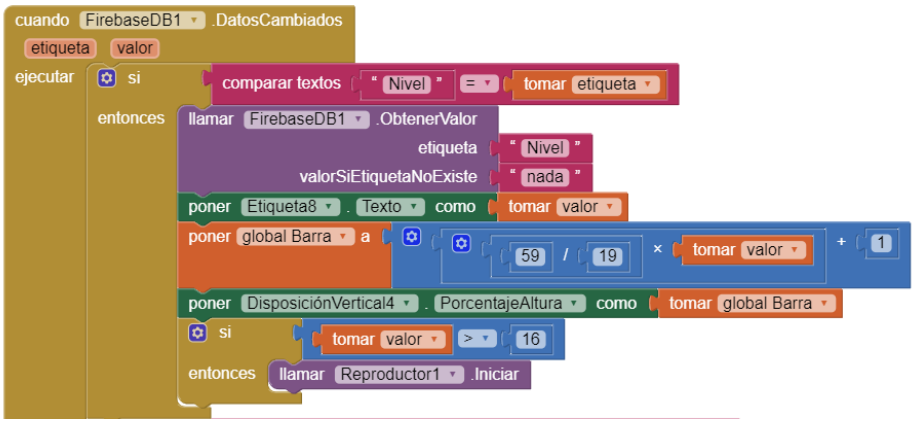


Figura 8.13.4. Configuración de los bloques para el funcionamiento de la pantalla de inicio.

Esta parte de los bloques se encarga de que cuando se detecte un cambio en el base de datos, primero verifique si el cambio de presentó en la etiqueta de Nivel en la base de datos, y si esto se cumple, pasa a ejecutar la función de obtener el valor que se esté leyendo de la base de datos, este valor de asigna a una etiqueta para poder visualizar el valor en la aplicación, también se ejecuta la instrucción para que se llene el medidor, también se configura la alarma sonora del celular, la cual solo se activara si el nivel de rio alcanza los 16 cm.

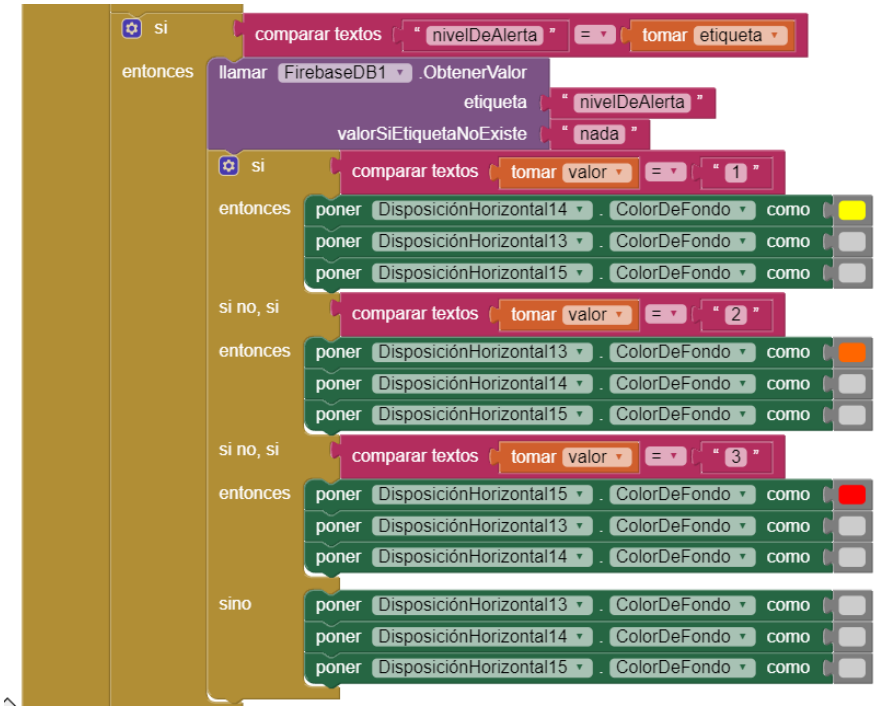


Figura 8.13.5. Configuración de los bloques para el funcionamiento de la pantalla de inicio.

Esta la parte de los bloques de la figura 8.13.5 se vuelve a consultar la base de datos, y cuando se presente un cambio en la etiqueta de nivelDealerta se va obtener el valor de esta etiqueta en la base de datos, y dependiendo del valor se va iluminar el nivel de alerta correspondiente.

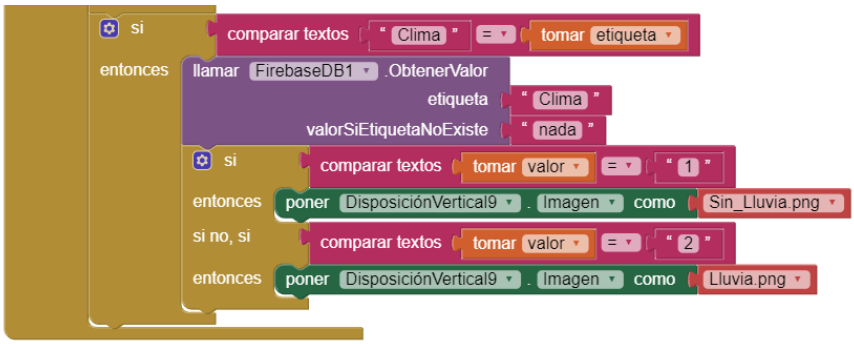


Figura 8.13.6. Configuración de los bloques para el funcionamiento de la pantalla de inicio.

Finalmente, de configura la parte de la imagen que indica el estado del tiempo, si se lee el valor de la etiqueta Clima en la base de datos, se tiene que para un 1, se pueda ver la imagen del sol, y para el valor de 2 se vea la imagen relacionada a la lluvia previamente subida.

Para las gráficas del histórico del nivel del agua y la cantidad de alertas de cada tipo que se han activado, se hace uso de Google chart, utilizando codificación en HTML, para lo cual se plantea el siguiente código: [27]

```
// Inicializa Firebase
firebase.initializeApp(firebaseConfig);
const db = firebase.database();

// Cargar Google Charts
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawChart);

let dataPoints = [['Fecha y Hora', 'Nivel de Agua']]; // Inicializar los puntos de la gráfica

// Función para dibujar la gráfica
function drawChart() {
  var data = google.visualization.arrayToDataTable(dataPoints);
  var options = {
    title: 'Historial de Niveles de Agua',
    curveType: 'function',
    legend: { position: 'bottom' },
    hAxis: {
      title: 'Fecha y Hora'
    },
    vAxis: {
      title: 'Nivel de Agua'
    }
  };

  var chart = new google.visualization.LineChart(document.getElementById('curve_chart'));
  chart.draw(data, options);
}
```

Código 11. Código para la dibujar la gráfica de historial de nivel de agua.

El código 11 se encarga de inicializar firebase, y Google charts, para posteriormente definir la función para dibujar la gráfica.

```
// Escuchar en tiempo real los cambios en Firebase para la gráfica
db.ref('historialDeNiveles').on('value', (snapshot) => {
  dataPoints = [['Fecha y Hora', 'Nivel de Agua']]; // Reiniciar los puntos de la gráfica

  snapshot.forEach((childSnapshot) => {
    let fechaHora = childSnapshot.key; // La clave es la fecha y hora
    let nivel = childSnapshot.val();   // El valor es el nivel de agua

    // Agregar los datos a los puntos de la gráfica
    dataPoints.push([fechaHora, nivel]);
  });

  // Redibujar la gráfica con los nuevos datos
  drawChart();
```

Código 12. Código para la dibujar la gráfica de historial de nivel de agua.

En el código 12 de tiene la lectura de los valores de Firebase con el fin de ir dibujando la gráfica a medida que se van viendo nuevos datos en Firebase, para el proyecto se realizaron pruebas con lectura de 2 minutos, pero en la práctica este valor debería leerse cada varias horas, para no saturar la base de datos.

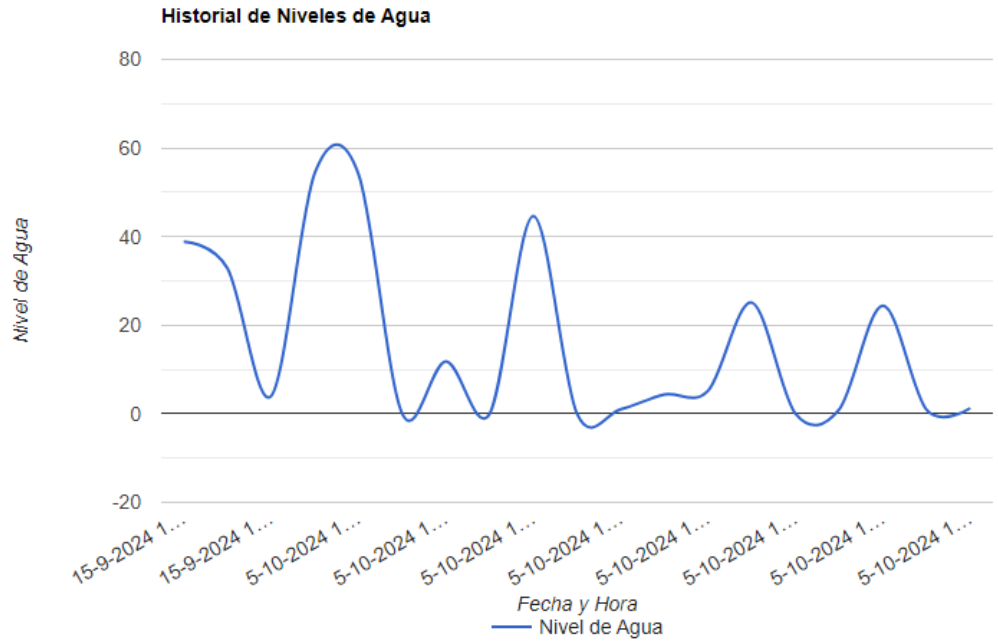


Figura 8.13.7. Grafica del historial de niveles de agua con valores de algunas pruebas.

En la figura 8.13.7 se puede ver cómo queda dibujada la gráfica, esta se va actualizando con cada lectura que se haga cada cierto tiempo según el código, para no saturarla de información esta grafica se limitó a 50 lecturas, por lo que cuando alcance este valor va a eliminar los valores más viejos para dar espacio a cada nuevo valor, por lo que se pensó esto para hacer lecturas cada varias horas para un amplio tiempo de almacenamiento de datos.

```
function actualizarDatos() {
    database.ref("/").once("value", function (snapshot) {
        var data = snapshot.val();
        var contadorAmarilla = parseInt(data.contadorAlertaAmarilla) || 0;
        var contadorNaranja = parseInt(data.contadorAlertaNaranja) || 0;
        var contadorRoja = parseInt(data.contadorAlertaRoja) || 0;

        var chartData = google.visualization.arrayToDataTable([
            ["Nivel de Alerta", "Veces Activado", { role: "style" }],
            ["Alerta Amarilla", contadorAmarilla, "#f1c40f"],
            ["Alerta Naranja", contadorNaranja, "#e67e22"],
            ["Alerta Roja", contadorRoja, "#e74c3c"],
        ]);

        var options = {
            title: "Gráfico de Alertas de Inundación",
            chartArea: { width: "60%" },
            hAxis: {
                title: "Número de Veces Activado",
                minValue: 0,
                textStyle: { fontName: "Arial", fontSize: 12 },
            },
            vAxis: {
                title: "Niveles de Alerta",
                textStyle: { fontName: "Arial", fontSize: 12 },
            },
            titleTextStyle: {
                fontName: "Arial",
                fontSize: 18,
                bold: true,
                color: "#2c3e50",
            },
            legend: { position: "none" },
            backgroundColor: "#f4f6f7",
            bar: { groupWidth: "80%" },
        };

        var chart = new google.visualization.BarChart(
            document.getElementById("barchart_values")
        );
        chart.draw(chartData, options);
    });
```

Código 13. Código para la dibujar el grafico de barras.

Para la gráfica de barras que muestra cuantas veces se ha activado cada nivel de alerta, se usa un código similar, para este caso se utiliza la función actualizarDatos para actualizar el gráfico con

los datos más recientes almacenados en Firebase, luego con ayuda de las funciones de Google charts se estructuran los datos para el gráfico en un formato de tabla. Cada fila representa un tipo de alerta con su correspondiente contador de veces activado y un color, para finalmente generar un gráfico de barras que muestra cuántas veces se han activado las alertas amarilla, naranja y roja, obteniendo los datos en tiempo real desde Firebase.

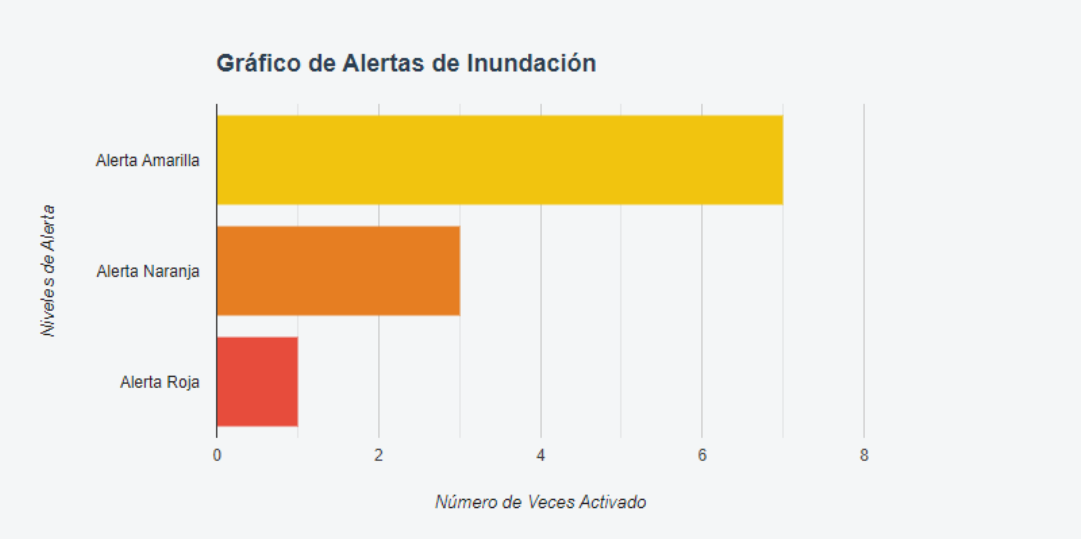


Figura 8.13.8. Grafica del historial de niveles de agua con valores de algunas pruebas.

En la figura 8.13.8 se puede ver cómo queda la gráfica propuesta en el código 13, indicando que tanto se han activado los diferentes niveles de alerta desde que el sistema de monitoreo empieza a funcionar.

Posteriormente se pasa a la pestaña para el mapa, el cual también se va a poder visualizar con ayuda de código HTML:

```
// Inicializar el mapa en Bogotá
var map = L.map('map').setView([4.630477, -74.083623], 15); // Usamos las coordenadas de la distri

// Cargar la capa de OpenStreetMap
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  maxZoom: 18,
  attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

var circle; // Variable para el círculo de alerta

// Leer los datos directamente desde la raíz de Firebase
db.ref().on('value', (snapshot) => {
  const data = snapshot.val();


  const latitud = data ? data.Latitud || 4.630477 : 4.630477; // Coordenada de la Universidad Distrital
  const longitud = data ? data.Longitud || -74.083623 : -74.083623; // Coordenada de la Universidad Distrital
  const nivelDeAlerta = data ? parseInt(data.nivelDeAlerta) : 0;

  // Mostrar las coordenadas y el nivel de alerta en la consola para verificación
  console.log("Coordenadas obtenidas:", latitud, longitud);
  console.log("Nivel de Alerta:", nivelDeAlerta);
```

Código 14. Código para mostrar el mapa de la zona monitoreada según datos de Firebase.

El código 14 inicializa un mapa centrado en Bogotá (para las pruebas) usando la biblioteca Leaflet y OpenStreetMap, y coloca un marcador interactivo basado en datos en tiempo real obtenidos desde Firebase. Se utilizan coordenadas predeterminadas (las de la Universidad Distrital) para latitud y longitud, y el nivel de alerta se obtiene desde Firebase. Además, se imprime en la consola las coordenadas y el nivel de alerta para verificación.

```
      // Si el nivel de alerta es 3, dibujar el círculo
      if (nivelDeAlerta === 3) {

        if (circle) {
          map.removeLayer(circle);
        }

        // Crear un círculo rojo para el posible área de afectación
        circle = L.circle([latitud, longitud], {
  color: 'red',
  fillColor: '#f03',
  fillOpacity: 0.5,
  radius: 500 // Radio de 500 metros para un área pequeña
}).addTo(map);

        // Centrar el mapa en las coordenadas del sensor
        map.setView([latitud, longitud], 15); // Centrar el mapa en las coordenadas de la Universidad Distrital
      } else {
        // Si no hay alerta roja y ya hay un círculo, eliminarlo
        if (circle) {
          map.removeLayer(circle);
        }
      }
    });
```

Código 15. Código para mapear la posible zona de afectación en caso de alerta roja.

El código 15 verifica si el nivel de alerta es 3 (alerta roja) y, si es así, dibuja un círculo rojo en el mapa para indicar un área de posible afectación con un radio de 500 metros. Si ya existe un círculo previamente dibujado, primero lo elimina para evitar duplicados. Además, centra el mapa en las coordenadas del sensor. Si el nivel de alerta no es 3 y hay un círculo previamente dibujado, este se elimina del mapa.



Figura 8.13.9. Visualización del mapa y el área de afectación.

Finalmente, para la pestaña con la información adicional del estado el tiempo, se hace uso de una API de Tomoorow.io que brinda algo de información recolectada por los distintos sistemas de monitoreo del clima del mundo, para este caso de prueba, para Bogotá en las coordenadas cercanas a la universidad distrital, el código que permite visualizar esto por el navegador web y con ayuda de GitHub es el siguiente: [28]

```
async function obtenerClima() {
    try {
        const response = await fetch(url);
        const data = await response.json();
        const clima = data.data.timelines[0].intervals[0].values;

        // Actualizar la interfaz con los datos obtenidos
        document.getElementById('temp').innerText = clima.temperature;
        document.getElementById('humidity').innerText = clima.humidity;
        document.getElementById('windSpeed').innerText = clima.windSpeed;
        document.getElementById('pressure').innerText = clima.pressureSeaLevel;
        document.getElementById('cloudCover').innerText = clima.cloudCover;

    } catch (error) {
        console.error('Error al obtener los datos del clima:', error);
    }
}
```

Código 16. Código para mostrar los datos obtenidos de Tomorrow.io.

Se recogen los datos con ayuda de la API y se mapean en el navegador web con la interfaz definida en el navegador web, luego se sube a GitHub y se agrega con un visor web a MIT app inventor.

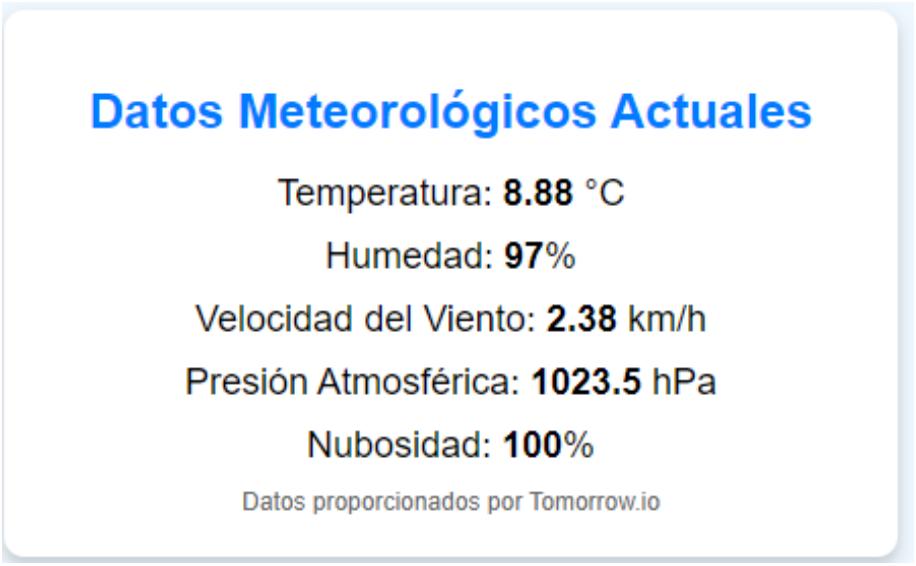


Figura 8.13.10. Visualización del mapa y el área de afectación.

Finalmente, en MIT app inventor se configuran los bloques necesarios para poder hacer el cambio de pantalla y visualizar cada una con ayuda de los siguientes bloques:

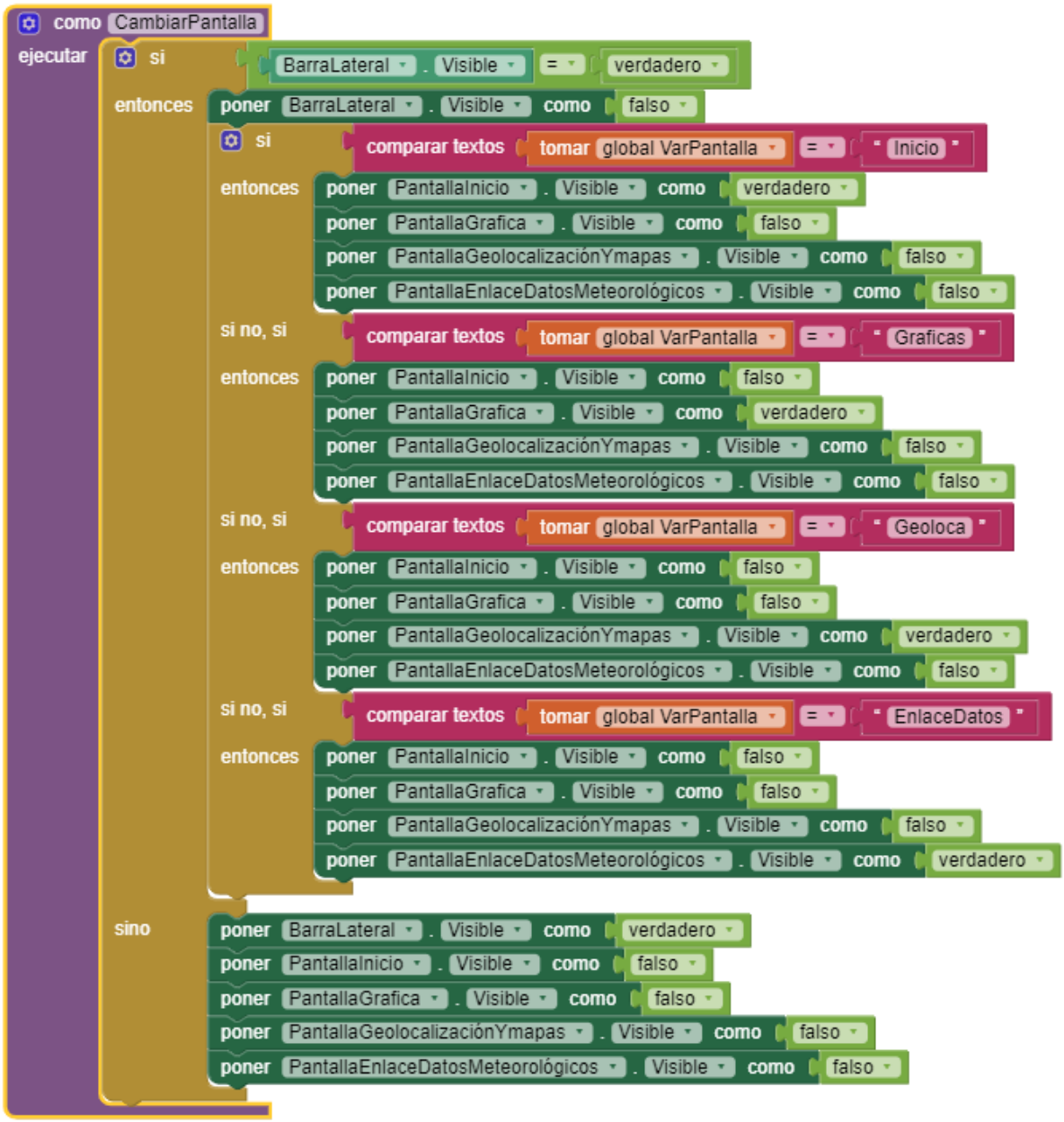


Figura 8.13.11. Configuración de los bloques para el cambio de pantalla.

Capítulo 9

Resultados

Para realizar las pruebas, como se hizo en la primera prueba de funcionamiento del sistema, se colocó la caja diseñada previamente con el circuito como se puede apreciar en las siguientes imágenes:



Figura 9.1: Estructura adecuada para comprobar el funcionamiento del sistema completo.

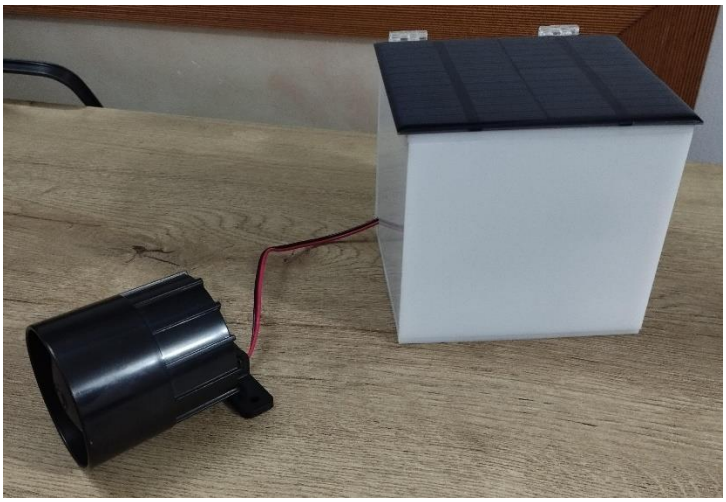


Figura 9.2: Sistema de alarma para comprobar el funcionamiento del sistema completo.

Inicialmente se sube un poco el nivel del rio, para verificar que el sensor, la lectura de datos en Firebase y en la aplicación funciona correctamente, por o que se deja en 6.5 cm aproximadamente.

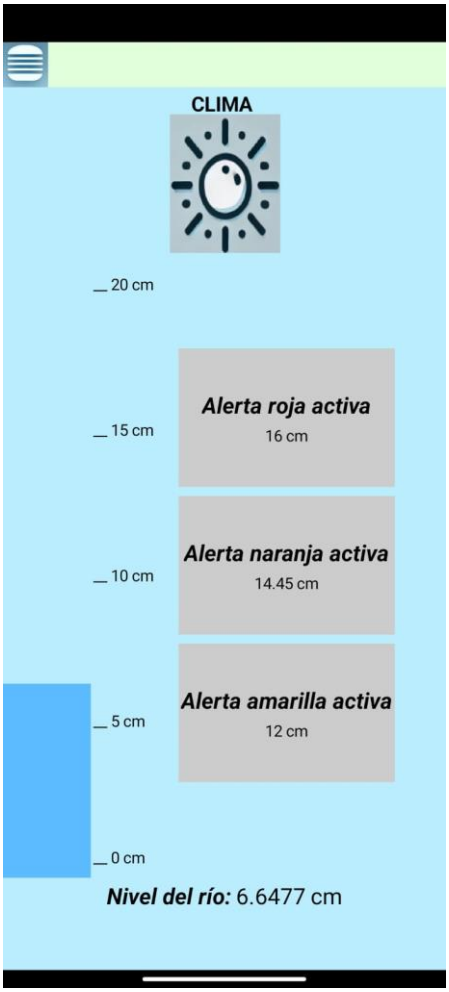


Figura 9.3: Aplicativo en funcionamiento mostrando la información del nivel del rio.

La lectura se efectúa de manera correcta, la barra del medidor también se ajusta al nivel correspondiente del nivel del rio y ya que el sensor de lluvia no se ha mojado el clima se muestra correctamente.

Se verifican las pestañas correspondientes a las gráficas:

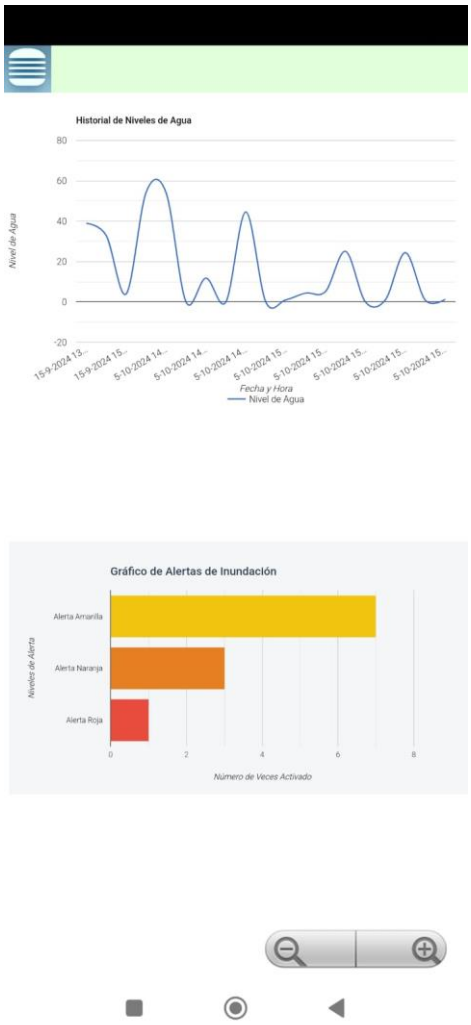


Figura 9.4: Aplicativo en funcionamiento mostrando la información de las gráficas.

Los datos que se pueden apreciar en las gráficas son debido a las pruebas que se hicieron previamente para comprobar que estuviera recopilando los datos correctamente tanto de lo sensores como de Firebase.

Se verifica ahora la pestaña con el mapa.



Figura 9.5: Aplicativo en funcionamiento mostrando la información del mapa.

Dado que no se ha llegado al nivel de alerta rojo, el mapa no tiene dibujada el área de afectación, ahora se verifica la pestaña de los datos meteorológicos:



Figura 9.6: Aplicativo en funcionamiento mostrando la información de los datos meteorológicos.

La información se está mostrando con éxito con ayuda de la API, por lo que todas las pestañas funcionan de manera correcta, luego se incrementa el nivel del rio para ver el comportamiento de todo el sistema, se sube hasta que se llegue a la alerta amarilla como se aprecia en la siguiente figura:



Figura 9.7: Estructura con el nivel del rio a 12.5 cm.

Una vez se alcanza el umbral de la alerta amarilla se puede visualizar en el aplicativo como se ilumina el bloque relacionado a este nivel de alerta, Adicionalmente después de comprobar que el nivel de alerta amarillo se active correctamente, se le aplica algo de agua al sensor de lluvia para ver que la imagen que indica el estado del clima cambie como se espera, pudiendo ver esto en las siguientes figuras:



Figura 9.8: Aplicativo en funcionamiento mostrando la información de la alerta amarilla activada.

Se sigue incrementando el nivel del rio hasta el umbral de alerta naranja:



Figura 9.9: Estructura con el nivel del rio a 15 cm.

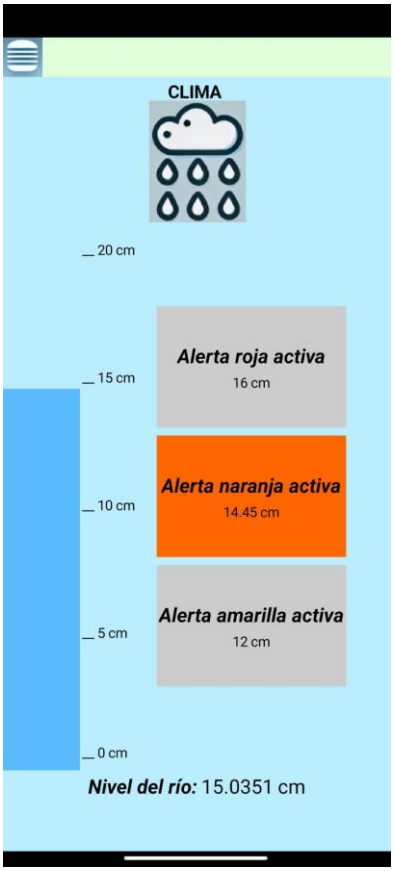


Figura 9.10: Aplicativo en funcionamiento mostrando la información de la alerta naranja activada.

Finalmente, para verificar el estado de alerta roja se sube el nivel hasta el umbral correspondiente:



Figura 9.11: Estructura con el nivel del rio a 16.5cm.

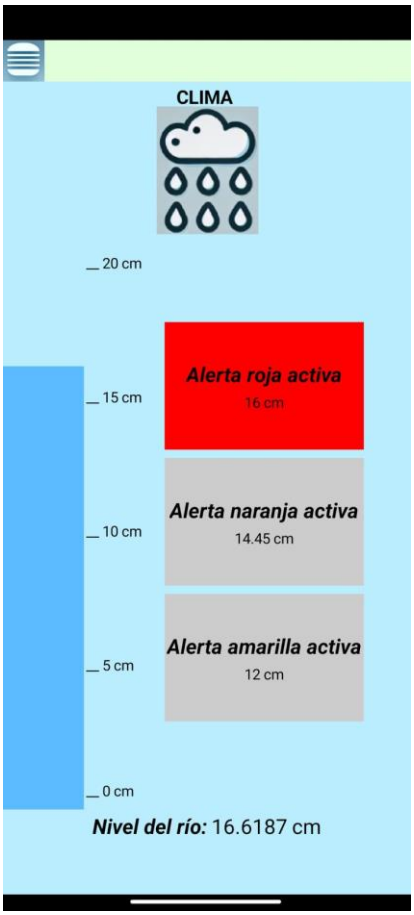


Figura 9.12: Aplicativo en funcionamiento mostrando la información de la alerta roja activada.

Al llegar al estado de alerta roja, la alarma configurada en el aplicativo comienza a sonar, igualmente el sistema de alarma se activa, así mismo la pestaña del mapa nos muestra la posible área de afectación como se puede ver en la siguiente figura:

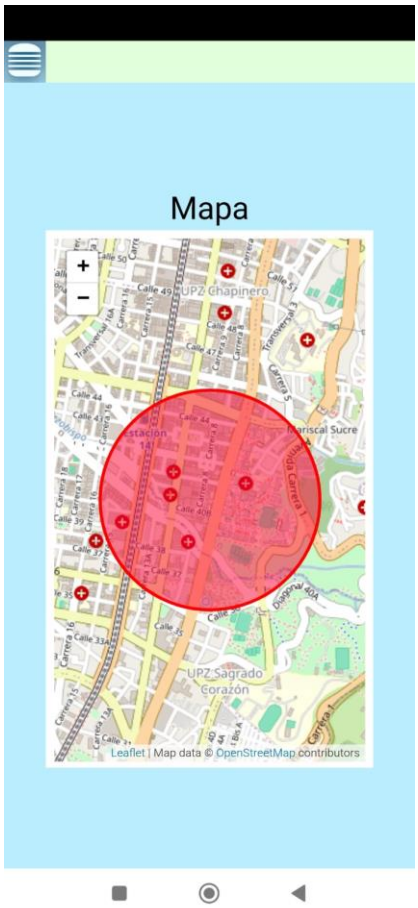


Figura 9.13: Aplicativo en funcionamiento mostrando la información del área de afectación en el mapa.

Capítulo 10

Conclusiones

- La comprobación del funcionamiento del sistema se realizó teniendo en cuenta las condiciones de la maqueta, la cual tiene 20 cm para verificar el aumento del nivel del agua, el sistema está pensando para poder implementarse en condiciones reales en un rio, teniendo en cuenta esto, los parámetros y variables que hacen funcionar el sistema, son fácilmente ajustables a un entorno real, y deben funcionar en el mismo de manera eficiente, teniendo en cuenta la limitación de máxima distancia de 4 metros que tiene el sensor de ultrasonido.

- El prototipo cumple con la mayoría de los aspectos propuestos, ya que puede monitorear constantemente el nivel del agua de un río y es autosuficiente, gracias a los paneles solares que lo mantienen cargado. Además, cuenta con una aplicación móvil que, sin necesidad de estar conectada a la misma red, puede alertar sobre diferentes situaciones. Otro aspecto importante es que este proyecto fue diseñado con elementos de bajo costo que hemos utilizado a lo largo de nuestra carrera.

- El uso de paneles solares permitió dar solución de manera sostenible al problema de alimentación del sistema de alerta, y con las pruebas realizadas de la autonomía del sistema se comprobó que la solución es efectiva y es capaz de proveer energía por mucho tiempo de manera ininterrumpida mientras las baterías y los paneles solares trabajen sin inconvenientes.

- Para la alimentación, se utilizaron pilas en paralelo, ya que una sola pila con el panel solar no tenía suficiente energía para mantener encendido de manera autónoma el sistema de monitoreo durante la noche. Al colocar las pilas en paralelo, se mantiene el mismo voltaje y se aumenta la capacidad de entrega de corriente, lo que permite mantener el dispositivo encendido toda la noche y gran parte de la mañana sin necesidad de los paneles solares.

- El valor del nivel de agua que se visualiza en la aplicación es un promedio, ya que el sensor a veces mostraba valores incorrectos en relación con el nivel real del agua, para luego retornar a una medición correcta. Al calcular el promedio, evitamos que estos valores erróneos activaran alarmas por error o generaran confusión al visualizarse en la aplicación.

- El uso de MIT app inventor, a pesar de no contar con una personalización o programación de funcionalidades muy complejas, fue suficiente para diseñar una aplicación que cumpliera con los objetivos planteados para el trabajo.

- El sensor ultrasonido HC-SR04 que se utilizó para este trabajo, presento ciertos inconvenientes cuando se hacían pruebas para medir el nivel del agua, dado que, en unas pocas lecturas, se tenían valores que no correspondían al valor real del agua, esto se debe a que el agua no siempre refleja de manera eficiente las ondas ultrasónicas hacia el sensor que se utilizó, por lo que fue necesario tomar un valor promedio e todas la mediciones que realizaba el sensor, con el fin de que esas pocas mediciones erradas, no afectaran significativamente el valor real del nivel del rio, lo cual resolvió este problema y sirvió para este contexto académico.

Capítulo 11

Trabajo futuro

En futuras mejoras del sistema de alerta temprana contra inundaciones con IoT, la idea es ampliar la cobertura del sistema añadiendo más sensores en distintos puntos a lo largo del río. Con estos sensores distribuidos, se podrá monitorear varias zonas al mismo tiempo y en tiempo real, lo que mejoraría mucho la capacidad de respuesta ante riesgos de inundación en áreas más vulnerables y hacer el sistema de alerta más robusto y eficiente.

Para mejorar la comunicación entre los sensores que están en el río y los sistemas de alerta en el poblado, se está pensando en usar tecnologías como LoRa y Sigfox, que son ideales para largas distancias y consumen muy poca energía. LoRa puede enviar datos a más de 10 km de distancia con un consumo mínimo, lo que viene perfecto para los dispositivos que funcionan con baterías y paneles solares. Sigfox, por su parte, es una opción confiable para mandar pequeños paquetes de datos a grandes distancias usando redes diseñadas para el Internet de las Cosas (IoT).

Otra mejora interesante sería implementar un sistema de análisis predictivo que, aprovechando los datos almacenados en Firebase, pueda anticipar posibles inundaciones basándose en patrones históricos. Este sistema se podría conectar con la bocina que ya tenemos para emitir alertas en caso de emergencia, y se podrían ajustar la intensidad y frecuencia de las alertas según el nivel de riesgo que se detecte.

Capítulo 12

Referencias

[1] Las inundaciones en Colombia se estarían dando en mayor porcentaje del que se creía, según la UNAL, 2017, https://www.iagua.es/noticias/universidad-nacional-colombia/inundaciones-colombia-se-estarian-dando-mayor-porcentaje-que.

[2] Sistema Nacional de información para la gestión de riesgo de desastres, https://gestiondelriesgo.gov.co/snigrd/pagina.aspx?id=144

[3] ¿Por qué ocurren inundaciones en el bajo cauca?, 2021, https://periodico.unal.edu.co/articulos/por-que-ocurren-inundaciones-en-el-bajo-cauca/

[4] G. Borizi, C.Francisco. Methodological proposal for the study of floods in plains areas with a lack of information, 2020.

[5] Planear y conocer los territorios para prevenir el riesgo ante desastres naturales, 2022, https://unimedios.medellin.unal.edu.co/bitacora/conexion/1027-planear-y-conocer-los-territorios-para-prevenir-el-riesgo-ante-desastres-naturales.html

[6] Doce municipios de Cundinamarca, bajo monitoreo permanente por deslizamientos e inundaciones, 2023, https://www.semana.com/actualidad/articulo/doce-municipios-de-cundinamarca-bajo-monitoreo-permanente-por-deslizamientos-e-inundaciones/202310/

[7] Al menos 371 han resultado damnificadas por inundaciones en Colombia, 2016, https://www.defensoria.gov.co/-/al-menos-371-familias-han-resultado-damnificadas-por-inundaciones-en-colombia

[8] Colombia: Inundacion a gran escala en la Mojana – Reporte de situación SITREP No. 01 del 25 de agosto de 2023, https://reliefweb.int/report/colombia/colombia-inundacion-gran-escala-en-la-mojana-reporte-de-situacion-sitrep-no-01-del-25-de-agosto-de-2023

[9] Inundaciones tienen aisladas a varias familias en el Meta, 2020, https://www.eltiempo.com/colombia/otras-ciudades/inundaciones-en-villavicencio-dejan-varias-familias-damnificadas-522142

[10] TANIA LORENA RUIZ MORALES LAURA VANESSA SALAZAR DUARTE. *DETERMINACIÓN DE POSIBLES ZONAS EN AMENAZA POR INUNDACIÓN EN EL MUNICIPIO DE MOMPÓS – BOLÍVAR, MEDIANTE EL USO DE PROCESAMIENTO DIGITAL DE IMÁGENES Y HERRAMIENTAS SIG, Y SU POSTERIOR PUBLICACIÓN COMO DATOS ABIERTOS (OPEN DATA)*. 2017.

[11] María Alejandra Guerrero Morillo Guillermo Eduardo Armenta Porras Xavier Corredor Llano. *LA VARIABILIDAD CLIMÁTICA Y EL CAMBIO CLIMÁTICO EN COLOMBIA*. 2018.

[12] *Manual Sobre el Manejo de Peligros Naturales en la Planificación para el Desarrollo Regional Integrado*. Capitulo 8. Definición de llanura de inundación y evaluación del peligro de inundaciones.

[13] Cuidado con las riberas de los ríos, primero la vida, UNGRD.

[14] *https://www.gestiondelriesgo.gov.co/snigrd/pagina.aspx?id=109*. Sistema Nacional de Información para la Gestión del Riesgo de Desastres.

[15] *https://curadoresurbanos.org/decreto-1077-de-2015-publico-pagina-3/*. Decreto 1077 de 2015 (2021) – Página 3.

[16] *GUÍA PARA EL DESARROLLO DE SISTEMAS DE ALERTA TEMPRANA*. SISTEMA NACIONAL DE GESTIÓN DEL RIESGO DE DESASTRES.

[17] *https://www.sigmaelectronica.net/producto/hc-sr04/*. Sigma Electrónica.

[18] *https://www.sigmaelectronica.net/producto/esp-32/*. Sigma Electrónica.

[19] https://firebase.google.com/?hl=es

[20] Daniel Moreno Juan Acosta Diego Mendez. *An IoT-based scalable river level monitoring platform*. International Journal of Sensor Networks, 2022.

[21] José Ibarreche Carlos Moreno 1 Raúl Aquino. *RiverCore: IoT Device for River Water Level Monitoring over Cellular Com- munications*. 2018.

[22] Rizal Munadi Dedi Satria Syaifuddin Yana. *Design of Information Monitoring System Flood Based Internet of Things (IoT)*. Emerald publishing, 2017.

[23] A.A. Shahrin Wahidah Md. Shah F. Arif† y Aslinda Hassan. *The Implementation of an IoT-Based Flood AlertSystem*. International Journal of Advanced Computer Science y Applications, 2018.

[24] F. Yakub MI. Hadi. *Designing Early Warning Flood Detection and Monitoring System via IoT*. The 7th AUN/SEED-Net Regional Conference on Natural Disaster, 2019.

[25] R.Kingsy Grace M.Madhumathi. *Flood Alert Management System Using IoT and Microcontroller*. International Journal of Innovative Research in Computer y Communication Engineering.

[26] The MIT App Inventor Library: Documentation & Support, https://appinventor.mit.edu/explore/library

[27] https://developers.google.com/chart?hl=es-419

[28] Documentación de GitHub, https://docs.github.com/es

Capítulo 13

Anexos

13.1 Código del sistema de monitoreo

```
#include <FirebaseESP32.h>
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
#include <WiFiUdp.h>
#include <NTPClient.h>
#include <WiFi.h>
#include <TimeLib.h>

#define API_KEY "*****************"
#define DATABASE_URL "*****************"
#define USER_EMAIL "*****************"
#define USER_PASSWORD "*****************


const int numReadings = 10;  // Cantidad de lecturas
float readings[numReadings];  // Array para almacenar las lecturas
int currentIndex = 0;              // Índice actual del array
float sum = 0.0;            // Suma total de las lecturas
float average = 0.0;        // Promedio calculado
float clima = 0;
float nivelDeAlerta = 0;
float contadorAlertaAmarilla = 0;
float contadorAlertaNaranja = 0;
float contadorAlertaRoja = 0;


FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Configuración de NTP para obtener la fecha y hora actuales
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", -5*3600, 60000); // Sin zona horaria,
actualiza cada 60 segundos
```

```
unsigned long ultimoEnvioHistorial = 0; // Variable para almacenar el tiempo del último
envío al historial
unsigned long intervaloHistorial = 7200000; // Intervalo de 2 horas (en milisegundos)
int maximoDatos = 72; // Máximo de datos (6 días, 12 datos por día)

// Función para contar el número de registros en Firebase
int contarRegistrosHistorial() {
  Firebase.getInt(fbdo, "/historialDeNiveles/contador");
  return fbdo.intData();
}


const char* ssid = "*****************";
const char* password = "*****************";


// Definición de pines
const int trigPin = 12; // GPIO 12 para el pin Trig
const int echoPin = 14; // GPIO 14 para el pin Echo
const int bateria = 33; // GPIO 33 para el pin bateria
const int llueve = 23; // Pin digital del YL-83 conectado al pin 23 de la ESP32
const int intensidadLluvia = 34; // Pin analógico del YL-83 conectado al
pin 34 de la ESP32



void setup() {

  // Inicializa el array de lecturas con ceros
 for (int i = 0; i < numReadings; i++) {
   readings[i] = 0.0;
 }


  Serial.begin(115200);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(llueve, INPUT);


  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }


  Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());



  // Configuración de Firebase
  Serial.printf("Firebase client", FIREBASE_CLIENT_VERSION);
  config.api_key = API_KEY;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);

  // Inicializar el cliente NTP para obtener la fecha y hora actuales
  timeClient.begin();
  timeClient.update();

}

void loop() {
  // Verificacion y reconeciond e firebase.
  if (!Firebase.ready()) {
    reconnectFirebase();
  }

  // Medir la distancia del sensor de nivel de agua
  float distance = measureDistance();


  analogRead(33);

float batteryLevel = map(analogRead(33), 0.0f, 4095.0f, 0, 100);

 Serial.print("Bateria: ");
  Serial.println(analogRead(33));
  Serial.println(batteryLevel);



// Leer la salida digital
  int digitalValue = digitalRead(llueve);
```

```
  // Leer la salida analógica
  int analogValue = analogRead(intensidadLluvia);

  // Convertir el valor analógico a porcentaje (0-100%)
  float rainIntensityPercentage = map(analogValue, 0, 4095, 0, 100);


  if ((-rainIntensityPercentage+100) >= 70){
        clima = 2;
      }
  else {clima = 1;
    }



if (average < 12){
    nivelDeAlerta = 0;
  }

  if (average >= 12 && average < 14.45) {
       if (nivelDeAlerta == 0){
        nivelDeAlerta = 1;
        contadorAlertaAmarilla = contadorAlertaAmarilla +1;
        } else {
          nivelDeAlerta = 1;
          }
      }


      if (average >= 14.45 && average < 16){
        if (nivelDeAlerta == 1){
        nivelDeAlerta = 2;
        contadorAlertaNaranja = contadorAlertaNaranja +1;
        } else {
          nivelDeAlerta = 2;
          }
      }


      if (average >= 16){
        if (nivelDeAlerta == 2){
        nivelDeAlerta = 3;
        contadorAlertaRoja = contadorAlertaRoja +1;
        } else {
          nivelDeAlerta = 3;
          }
```

```
      }



  // Mostrar los valores en el monitor serie
  Serial.print("Valor digital: ");
  Serial.print(digitalValue);
  Serial.print(" | Valor analógico: ");
  Serial.print(analogValue);
  Serial.print(" | Intensidad de lluvia: ");
  Serial.print(rainIntensityPercentage);
  Serial.println("%");




  // Verificar si ha pasado suficiente tiempo para enviar el historial (cada 2 horas)
  if (millis() - ultimoEnvioHistorial > intervaloHistorial) {
    // Actualizar la hora desde NTP
    timeClient.update();

  // Obtener el timestamp actual y convertirlo a una fecha legible
    unsigned long epochTime = timeClient.getEpochTime();
    setTime(epochTime);
    String currentDate = String(day()) + "-" + String(month()) + "-" + String(year());
    String currentTime = String(hour()) + ":" + String(minute()) + ":" +
String(second());
    String claveHistorial = currentDate + " " + currentTime; // Formato de la clave
con fecha y hora

    // Contar el número de registros en el historial
    int numRegistros = contarRegistrosHistorial();

    // Si ya hay más de 72 registros, sobrescribir el dato más antiguo
    if (numRegistros >= maximoDatos) {
      // Eliminar el registro más antiguo
      Firebase.getJSON(fbdo, "/historialDeNiveles");
      FirebaseJson &json = fbdo.jsonObject();
      FirebaseJsonData data;
      String claveMasAntigua;
      json.get(data, "0"); // Obtener la primera entrada (la más antigua)
      claveMasAntigua = data.stringValue;
      Firebase.deleteNode(fbdo, "/historialDeNiveles/" + claveMasAntigua);
      Serial.println("Dato más antiguo eliminado: " + claveMasAntigua);
    }
```

```
    // Enviar el valor de nivel del agua al historial en Firebase
    if (Firebase.setFloat(fbdo, "/historialDeNiveles/" + claveHistorial, distance)) {
      Serial.println("Historial de nivel actualizado con éxito");
    } else {
      Serial.println("Error al actualizar el historial");
      Serial.println(fbdo.errorReason());
    }

    // Actualizar el tiempo del último envío al historial
    ultimoEnvioHistorial = millis();
  }

  float newReading = distance;
  sum = sum - readings[currentIndex] + newReading;
  readings[currentIndex] = newReading;
  currentIndex = (currentIndex + 1) % numReadings;
  average = sum / numReadings;
  Serial.print("Nivel del agua: ");
  Serial.print(average, 3 );
  Serial.println(" cm");


  // Enviar el nivel del agua en tiempo real
  Firebase.setFloat(fbdo, "Nivel", average);
  Firebase.setFloat(fbdo,"Bateria", batteryLevel);
  Firebase.setFloat(fbdo,"Lluvia", digitalValue);
  Firebase.setFloat(fbdo,"IntensidadLluvia", rainIntensityPercentage);
  Firebase.setFloat(fbdo,"Clima", clima);
  Firebase.setFloat(fbdo,"nivelDeAlerta", nivelDeAlerta);
  Firebase.setFloat(fbdo,"contadorAlertaAmarilla", contadorAlertaAmarilla);
  Firebase.setFloat(fbdo,"contadorAlertaNaranja", contadorAlertaNaranja);
  Firebase.setFloat(fbdo,"contadorAlertaRoja", contadorAlertaRoja);



  delay(1000); // Esperar 1 segundos antes de la siguiente lectura en tiempo real
}

// Función para medir la distancia del sensor de nivel de agua
float measureDistance() {
  // Envía un pulso ultrasonico
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Recibe el eco
  long duration = pulseIn(echoPin, HIGH);
  float distance = 21.13 - (duration * 0.034 / 2); // Calcula la distancia

  if (distance < 0) {
    distance = 0;
  }

  Serial.print("Distance: ");
  Serial.println(distance);
  return distance;
}

void reconnectFirebase() {
  Serial.printf("Firebase client", FIREBASE_CLIENT_VERSION);
  config.api_key = API_KEY;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}
```

Anexo 13.1. Código sistema de monitoreo.

13.2 Código del sistema de alerta

```
#include <FirebaseESP32.h>
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
#include <WiFiUdp.h>
#include <NTPClient.h>
#include <WiFi.h>
#include <TimeLib.h>

#define API_KEY "********"
#define DATABASE_URL "********"
#define USER_EMAIL "********"
```

```
#define USER_PASSWORD "********"

const int outputPin = 19; // Pin para activar o desactivar la sirena

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

const char* ssid = "********";
const char* password = "****************";


void setup() {

 Serial.begin(115200); //

  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");

   pinMode(outputPin, OUTPUT);
  digitalWrite(outputPin, LOW);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }


  Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());



  // Configuración de Firebase
  Serial.printf("Firebase client", FIREBASE_CLIENT_VERSION);
  config.api_key = API_KEY;
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;
  config.database_url = DATABASE_URL;
  config.token_status_callback = tokenStatusCallback;
  Firebase.begin(&config, &auth);
```

```
}

void loop() {

  if (Firebase.ready()) {
    // Leer el valor de "nivelDeAlerta" desde Firebase
    if (Firebase.getFloat(fbdo, "/nivelDeAlerta")) {
      float nivelDeAlerta = fbdo.floatData();
      Serial.print("Nivel de Alerta: ");
      Serial.println(nivelDeAlerta);


// Verificar si el nivel es mayor o igual a 16 (alerta roja)
      if (nivelDeAlerta == 3) {
        digitalWrite(outputPin, HIGH); // Activar el pin de salida
        Serial.println("Pin de salida activado. Sirena encendida.");
      } else {
        digitalWrite(outputPin, LOW); // Desactivar el pin de salida
        Serial.println("Pin de salida desactivado. Sirena apagada.");
      }



    delay(1000);
}

  }

}
```

Anexo 13.2. Código sistema de alerta.

13.3 Código de la generación de la gráfica del historial del nivel del agua

```
<!DOCTYPE html>
<html>
 <head>
  <title>Gráfica del Nivel de Agua - Historial en Tiempo Real</title>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>
```

```
<script src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">
  // Configuración de Firebase
  const firebaseConfig = {
    apiKey: "*******",
    authDomain: "*******",
    databaseURL: "*******",
    projectId: "*******",
    storageBucket: "*******",
    messagingSenderId: "*******",
    appId: "*******"
  };

  // Inicializa Firebase
  firebase.initializeApp(firebaseConfig);
  const db = firebase.database();

  // Cargar Google Charts
  google.charts.load('current', {packages: ['corechart']});
  google.charts.setOnLoadCallback(drawChart);

  let dataPoints = [['Fecha y Hora', 'Nivel de Agua']]; // Inicializar los puntos de la gráfica

  // Función para dibujar la gráfica
  function drawChart() {
    var data = google.visualization.arrayToDataTable(dataPoints);
    var options = {
      title: 'Historial de Niveles de Agua',
      curveType: 'function',
      legend: { position: 'bottom' },
      hAxis: {
        title: 'Fecha y Hora'
      },
      vAxis: {
        title: 'Nivel de Agua'
      }
```

```
      };

      var chart = new google.visualization.LineChart(document.getElementById('curve_chart'));
      chart.draw(data, options);
    }

    // Escuchar en tiempo real los cambios en Firebase para la gráfica
    db.ref('historialDeNiveles').on('value', (snapshot) => {
      dataPoints = [['Fecha y Hora', 'Nivel de Agua']]; // Reiniciar los puntos de la gráfica

      snapshot.forEach((childSnapshot) => {
        let fechaHora = childSnapshot.key; // La clave es la fecha y hora
        let nivel = childSnapshot.val();   // El valor es el nivel de agua

        // Agregar los datos a los puntos de la gráfica
        dataPoints.push([fechaHora, nivel]);
      });

      // Redibujar la gráfica con los nuevos datos
      drawChart();
    });
  </script>
 </head>
 <body>
  <div id="curve_chart" style="width: 900px; height: 500px"></div>
 </body>
</html>
```

Anexo 13.3. Código Generación gráfica del historial del nivel del agua.

13.4 Código de la generación de la gráfica de la cantidad de alertas activadas

```
<!DOCTYPE html>
<html lang="es">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Gráfico de Alertas de Inundación</title>
  <script src="https://www.gstatic.com/charts/loader.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-database.js"></script>

  <script type="text/javascript">
   // Configuración de Firebase
   const firebaseConfig = {
    apiKey: "*******",
    authDomain: "*******",
    databaseURL: "*******
    projectId: "*******",
    storageBucket: "*******",
    messagingSenderId: "*******",
    appId: "*******"
   };
   firebase.initializeApp(firebaseConfig);

   google.charts.load("current", { packages: ["corechart"] });
   google.charts.setOnLoadCallback(drawChart);

   // Dentro de la función `drawChart`
function drawChart() {
  var database = firebase.database();

  function actualizarDatos() {
    database.ref("/").once("value", function (snapshot) {
      var data = snapshot.val();
      var contadorAmarilla = parseInt(data.contadorAlertaAmarilla) || 0;
```

```
var contadorNaranja = parseInt(data.contadorAlertaNaranja) || 0;
var contadorRoja = parseInt(data.contadorAlertaRoja) || 0;

var chartData = google.visualization.arrayToDataTable([
  ["Nivel de Alerta", "Veces Activado", { role: "style" }],
  ["Alerta Amarilla", contadorAmarilla, "#f1c40f"],
  ["Alerta Naranja", contadorNaranja, "#e67e22"],
  ["Alerta Roja", contadorRoja, "#e74c3c"],
]);

var options = {
  title: "Gráfico de Alertas de Inundación",
  chartArea: { width: "60%" },
  hAxis: {
    title: "Número de Veces Activado",
    minValue: 0,
    textStyle: { fontName: "Arial", fontSize: 12 },
  },
  vAxis: {
    title: "Niveles de Alerta",
    textStyle: { fontName: "Arial", fontSize: 12 },
  },
  titleTextStyle: {
    fontName: "Arial",
    fontSize: 18,
    bold: true,
    color: "#2c3e50",
  },
  legend: { position: "none" },
  backgroundColor: "#f4f6f7",
  bar: { groupWidth: "80%" },
};

var chart = new google.visualization.BarChart(
  document.getElementById("barchart_values")
);
chart.draw(chartData, options);
```

```
      });
    }

    // Llamar la función inmediatamente
    actualizarDatos();

    // Actualizar los datos cada 15 segundos
    setInterval(actualizarDatos, 15000);
}

    </script>
  </head>
  <body>
    <div id="barchart_values" style="width: 800px; height: 400px"></div>
  </body>
</html>
```

Anexo 13.4. Código Generación gráfica de la cantidad de alertas activadas.

13.5 Código de la generación del mapa

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mapa con Coordenadas desde Firebase</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
    <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>
    <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>
    <style>
      #map {
        height: 500px;
        width: 100%;
      }
    </style>
```

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

  <script>
   // Configuración de Firebase
    const firebaseConfig = {
     apiKey: "*****",
     authDomain: "*****",
     databaseURL: "*****
     projectId: "*****",
     storageBucket: "*****",
     messagingSenderId: "*****",
     appId: "*****"
    };

    // Inicializa Firebase
    firebase.initializeApp(firebaseConfig);
    const db = firebase.database();

    // Inicializar el mapa en Bogotá
    var map = L.map('map').setView([4.630477, -74.083623], 15); // Usamos las
coordenadas de la distri

    // Cargar la capa de OpenStreetMap
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
     maxZoom: 18,
     attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);

    var circle; // Variable para el círculo de alerta

    // Leer los datos directamente desde la raíz de Firebase
    db.ref().on('value', (snapshot) => {
     const data = snapshot.val();
```

```
    const latitud = data ? data.Latitud || 4.630477 : 4.630477; // Coordenada de la Universidad Distrital
    const longitud = data ? data.Longitud || -74.083623 : -74.083623; // Coordenada de la Universidad Distrital
    const nivelDeAlerta = data ? parseInt(data.nivelDeAlerta) : 0;

    // Mostrar las coordenadas y el nivel de alerta en la consola para verificación
    console.log("Coordenadas obtenidas:", latitud, longitud);
    console.log("Nivel de Alerta:", nivelDeAlerta);

    // Si el nivel de alerta es 3, dibujar el círculo
    if (nivelDeAlerta === 3) {

      if (circle) {
        map.removeLayer(circle);
      }

      // Crear un círculo rojo para el posible área de afectación
      circle = L.circle([latitud, longitud], {
 color: 'red',
 fillColor: '#f03',
 fillOpacity: 0.5,
 radius: 500 // Radio de 500 metros para un área pequeña
}).addTo(map);

      // Centrar el mapa en las coordenadas del sensor
      map.setView([latitud, longitud], 15); // Centrar el mapa en las coordenadas de la Universidad Distrital
    } else {
      // Si no hay alerta roja y ya hay un círculo, eliminarlo
      if (circle) {
        map.removeLayer(circle);
      }
    }
   });
  </script>
 </body>
```

```
</html>
```

Anexo 13.5. Código Generación gráfica del mapa.

13.6 Código de la generación de los datos meteorológicos

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Clima Actual - Tomorrow.io</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f8ff;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }
    .weather-container {
      background-color: white;
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
      text-align: center;
      max-width: 400px;
      width: 100%;
    }
    h1 {
      font-size: 24px;
```

```
        color: #007BFF;
      }
      p {
        font-size: 18px;
        margin: 10px 0;
      }
      .data {
        font-weight: bold;
      }
      footer {
        font-size: 12px;
        margin-top: 10px;
        color: #666;
      }
    </style>
</head>
<body>
    <div class="weather-container">
      <h1>Datos Meteorológicos Actuales</h1>
      <p>Temperatura: <span id="temp" class="data">Cargando...</span> °C</p>
      <p>Humedad: <span id="humidity" class="data">Cargando...</span>%</p>
      <p>Velocidad del Viento: <span id="windSpeed" class="data">Cargando...</span>
km/h</p>
      <p>Presión Atmosférica: <span id="pressure" class="data">Cargando...</span>
hPa</p>
      <p>Nubosidad: <span id="cloudCover" class="data">Cargando...</span>%</p>
      <footer>Datos proporcionados por Tomorrow.io</footer>
    </div>

    <script>
      const apiKey = '68OZC7L4oCjPx7R7r7oNhLb7tK8i3Z2f';
      const latitud = 4.60971;  // Latitud de Bogotá
      const longitud = -74.08175;  // Longitud de Bogotá
      const                                    url                                    =
`https://api.tomorrow.io/v4/timelines?location=${latitud},${longitud}&fields=temperat
ure,humidity,windSpeed,pressureSeaLevel,cloudCover&timesteps=current&units=metri
c&apikey=${apiKey}`;
```

```
    async function obtenerClima() {
      try {
        const response = await fetch(url);
        const data = await response.json();
        const clima = data.data.timelines[0].intervals[0].values;

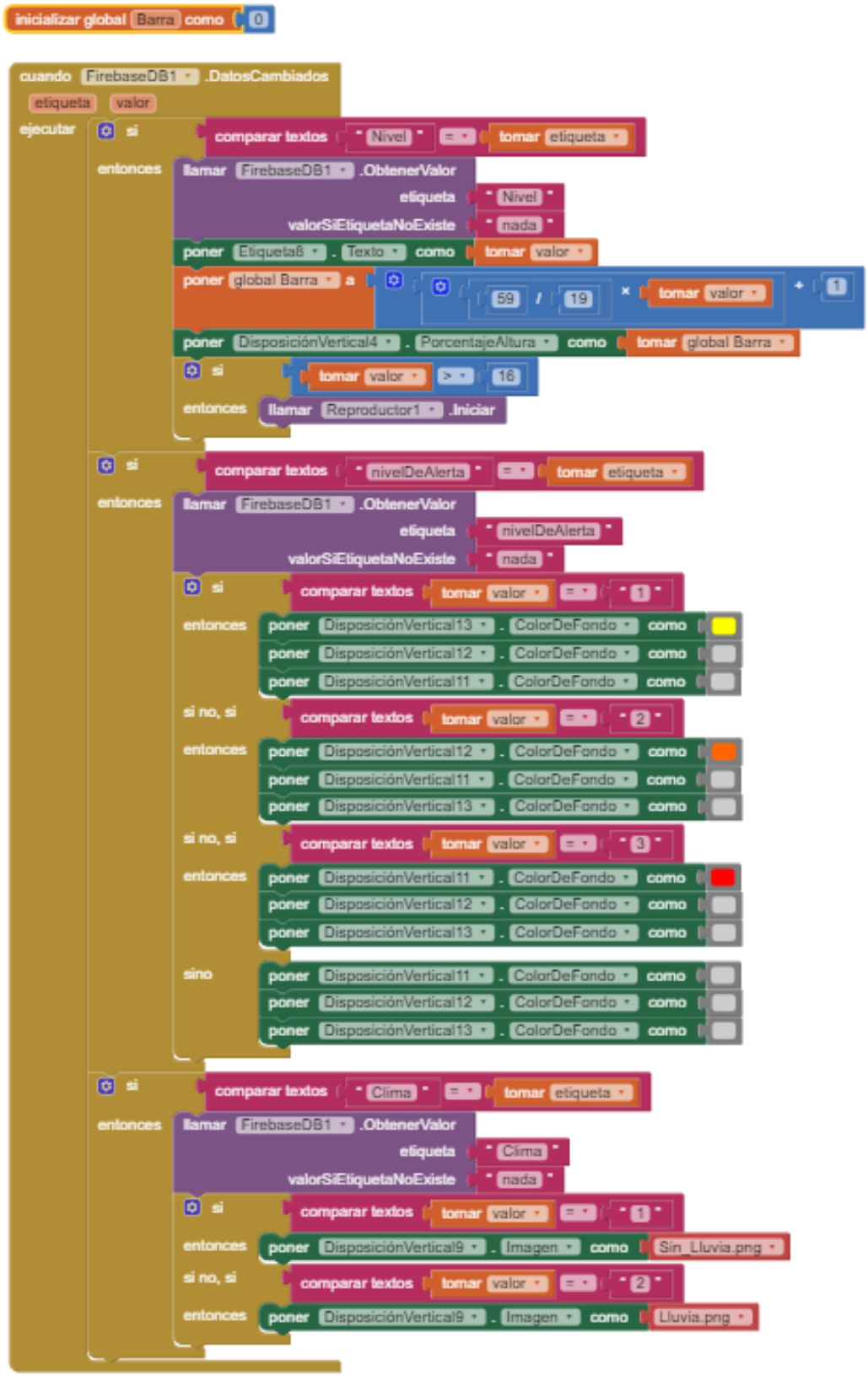
        // Actualizar la interfaz con los datos obtenidos
        document.getElementById('temp').innerText = clima.temperature;
        document.getElementById('humidity').innerText = clima.humidity;
        document.getElementById('windSpeed').innerText = clima.windSpeed;
        document.getElementById('pressure').innerText = clima.pressureSeaLevel;
        document.getElementById('cloudCover').innerText = clima.cloudCover;

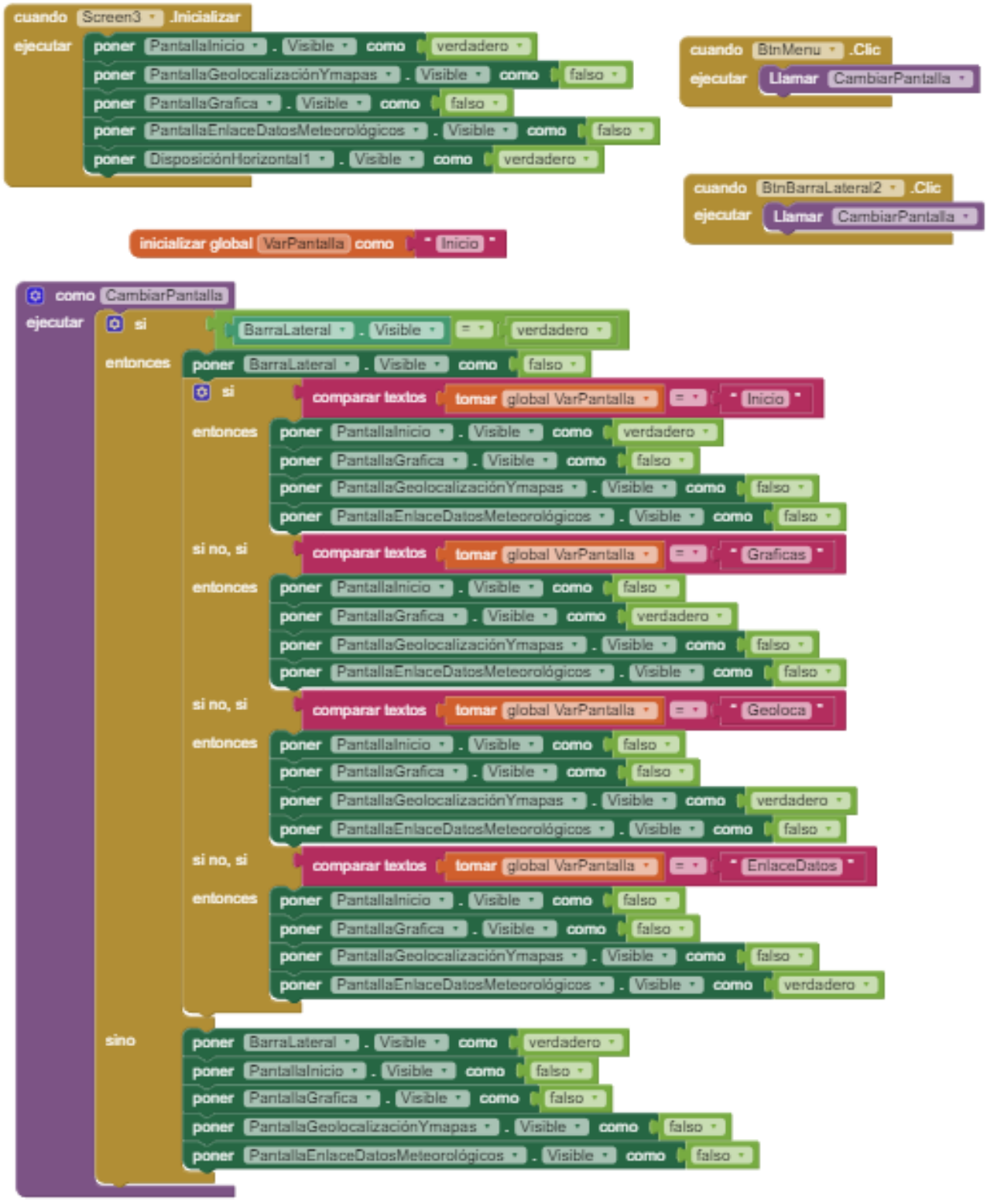
      } catch (error) {
        console.error('Error al obtener los datos del clima:', error);
      }
    }

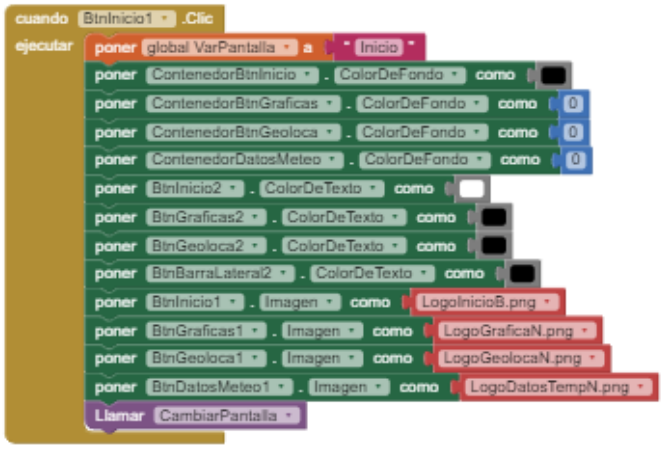
    obtenerClima();  // Llamar a la función para obtener el clima al cargar la página
  </script>
</body>
</html>
```

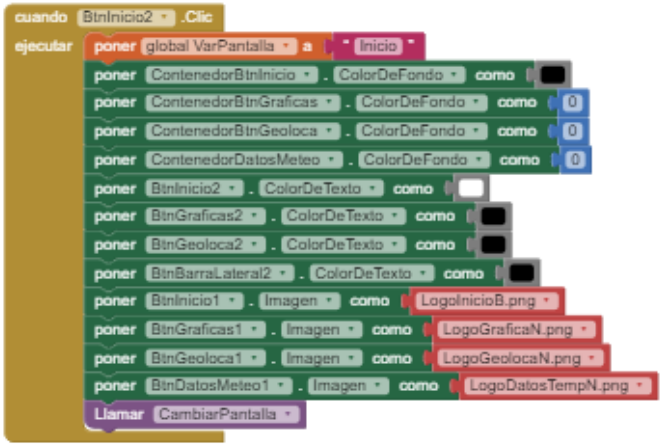
Anexo 13.6. Código Generación de los datos meteorológicos.

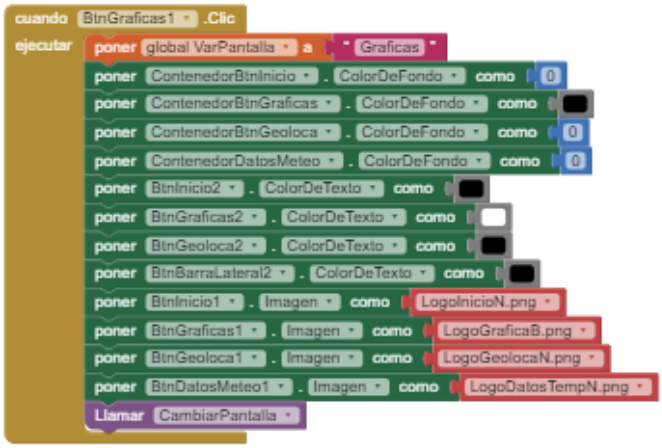
13.7 Códigos en bloques para la aplicación

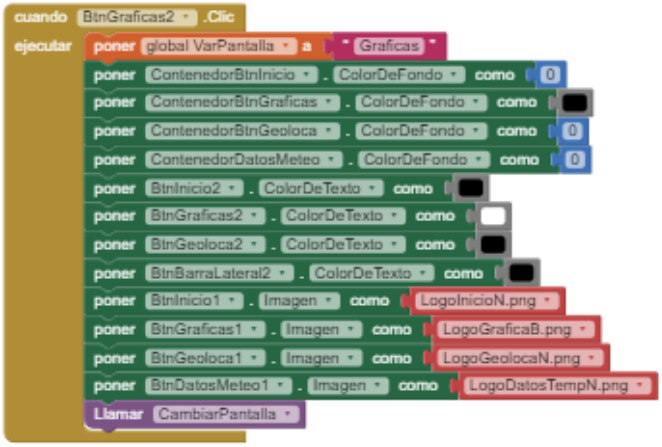


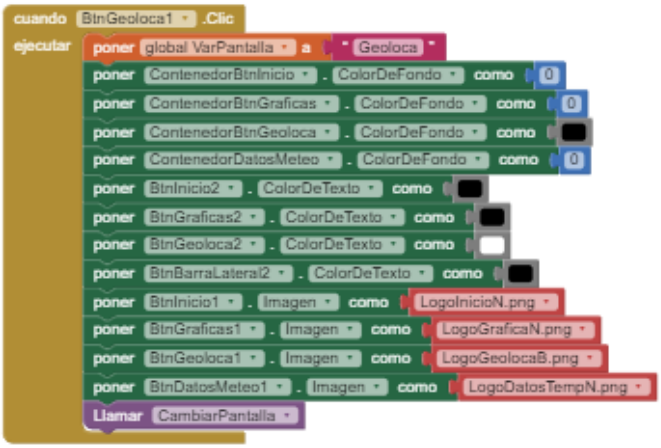

cuando Screen3 .Inicializar
ejecutar
poner PantallaInicio . Visible como verdadero
poner PantallaGeolocalizaciónYmapas . Visible como falso
poner PantallaGrafica . Visible como falso
poner PantallaEnlaceDatosMeteorológicos . Visible como falso
poner DisposiciónHorizontal1 . Visible como verdadero
cuando BtnMenu .Clic
ejecutar Llamar CambiarPantalla
cuando BtnBarraLateral2 .Clic
ejecutar Llamar CambiarPantalla
inicializar global VarPantalla como " Inicio "
como CambiarPantalla
ejecutar
si BarraLateral . Visible = verdadero
entonces
poner BarraLateral . Visible como falso
si comparar textos tomar global VarPantalla = " Inicio "
entonces
poner PantallaInicio . Visible como verdadero
poner PantallaGrafica . Visible como falso
poner PantallaGeolocalizaciónYmapas . Visible como falso
poner PantallaEnlaceDatosMeteorológicos . Visible como falso
si no, si comparar textos tomar global VarPantalla = " Graficas "
entonces
poner PantallaInicio . Visible como falso
poner PantallaGrafica . Visible como verdadero
poner PantallaGeolocalizaciónYmapas . Visible como falso
poner PantallaEnlaceDatosMeteorológicos . Visible como falso
si no, si comparar textos tomar global VarPantalla = " Geoloca "
entonces
poner PantallaInicio . Visible como falso
poner PantallaGrafica . Visible como falso
poner PantallaGeolocalizaciónYmapas . Visible como verdadero
poner PantallaEnlaceDatosMeteorológicos . Visible como falso
si no, si comparar textos tomar global VarPantalla = " EnlaceDatos "
entonces
poner PantallaInicio . Visible como falso
poner PantallaGrafica . Visible como falso
poner PantallaGeolocalizaciónYmapas . Visible como falso
poner PantallaEnlaceDatosMeteorológicos . Visible como verdadero
sino
poner BarraLateral . Visible como verdadero
poner PantallaInicio . Visible como falso
poner PantallaGrafica . Visible como falso
poner PantallaGeolocalizaciónYmapas . Visible como falso
poner PantallaEnlaceDatosMeteorológicos . Visible como falso

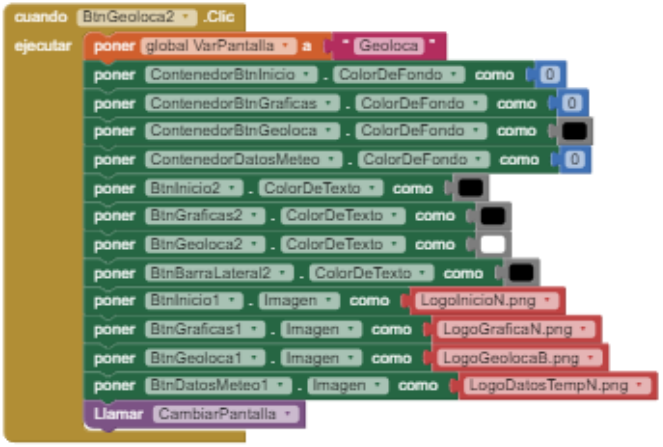

cuando BtnInicio1 .Clic
ejecutar
poner global VarPantalla a Inicio
poner ContenedorBtnInicio . ColorDeFondo como
poner ContenedorBtnGraficas . ColorDeFondo como 0
poner ContenedorBtnGeoloca . ColorDeFondo como 0
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioB.png
poner BtnGraficas1 . Imagen como LogoGraficaN.png
poner BtnGeoloca1 . Imagen como LogoGeolocaN.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla

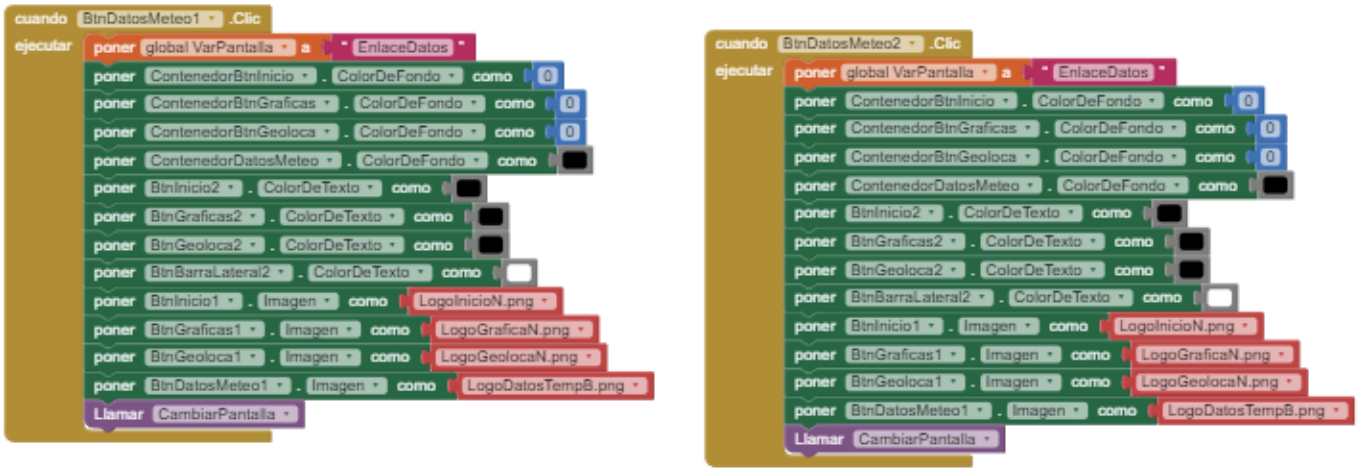

cuando BtnInicio2 .Clic
ejecutar
poner global VarPantalla a Inicio
poner ContenedorBtnInicio . ColorDeFondo como
poner ContenedorBtnGraficas . ColorDeFondo como 0
poner ContenedorBtnGeoloca . ColorDeFondo como 0
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioB.png
poner BtnGraficas1 . Imagen como LogoGraficaN.png
poner BtnGeoloca1 . Imagen como LogoGeolocaN.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla


cuando BtnGraficas1 .Clic
ejecutar
poner global VarPantalla a Graficas
poner ContenedorBtnInicio . ColorDeFondo como 0
poner ContenedorBtnGraficas . ColorDeFondo como
poner ContenedorBtnGeoloca . ColorDeFondo como 0
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioN.png
poner BtnGraficas1 . Imagen como LogoGraficaB.png
poner BtnGeoloca1 . Imagen como LogoGeolocaN.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla


cuando BtnGraficas2 .Clic
ejecutar
poner global VarPantalla a Graficas
poner ContenedorBtnInicio . ColorDeFondo como 0
poner ContenedorBtnGraficas . ColorDeFondo como
poner ContenedorBtnGeoloca . ColorDeFondo como 0
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioN.png
poner BtnGraficas1 . Imagen como LogoGraficaB.png
poner BtnGeoloca1 . Imagen como LogoGeolocaN.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla


cuando BtnGeoloca1 .Clic
ejecutar
poner global VarPantalla a Geoloca
poner ContenedorBtnInicio . ColorDeFondo como 0
poner ContenedorBtnGraficas . ColorDeFondo como 0
poner ContenedorBtnGeoloca . ColorDeFondo como
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioN.png
poner BtnGraficas1 . Imagen como LogoGraficaN.png
poner BtnGeoloca1 . Imagen como LogoGeolocaB.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla


cuando BtnGeoloca2 .Clic
ejecutar
poner global VarPantalla a Geoloca
poner ContenedorBtnInicio . ColorDeFondo como 0
poner ContenedorBtnGraficas . ColorDeFondo como 0
poner ContenedorBtnGeoloca . ColorDeFondo como
poner ContenedorDatosMeteo . ColorDeFondo como 0
poner BtnInicio2 . ColorDeTexto como
poner BtnGraficas2 . ColorDeTexto como
poner BtnGeoloca2 . ColorDeTexto como
poner BtnBarraLateral2 . ColorDeTexto como
poner BtnInicio1 . Imagen como LogoInicioN.png
poner BtnGraficas1 . Imagen como LogoGraficaN.png
poner BtnGeoloca1 . Imagen como LogoGeolocaB.png
poner BtnDatosMeteo1 . Imagen como LogoDatosTempN.png
Llamar CambiarPantalla



Anexo 13.7. Códigos en bloques para la aplicación.