Sistema Electrónico y Automático para la determinación de los porcentajes de las variables de Nitrógeno, Fósforo y Potasio (NPK) en suelos

Luiza Fernanda Narváez Timaná
Luis Fernando Cobo Mendez

Proyecto de Grado
Ingeniería en Automática Industrial

Director: M.S.c. Fabio Hernán Realpe Martínez
Co-Director: M.S.c. Judy Cristina Realpe Chamorro

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Control e Instrumentación
Popayán, Diciembre de 2021

Luiza Fernanda Narváez Timaná
Luis Fernando Cobo Mendez

Sistema Electrónico y Automático para la determinación de los porcentajes de las variables de Nitrógeno, Fósforo y Potasio (NPK) en suelos

Proyecto de Grado presentado a:
Facultad de Ingeniería Electrónica y Telecomunicaciones
de la Universidad del Cauca

Para obtener el título de:
Ingeniero/a en Automática Industrial

Director: M.S.c. Fabio Hernán Realpe Martínez
Co-Director: M.S.c. Judy Cristina Realpe Chamorro

Popayán, Diciembre de 2021

Agradecimientos

... A Dios y el Universo, por siempre guiarnos hacia el camino correcto y brindarnos las oportunidades para alcanzar esta meta

... A nuestros padres Fernando y Lucy, Luperli y Luis Alfredo quienes nos han acompañado incondicionalmente en este proyecto con infinito amor y paciencia

... A nuestros hermanos, por su apoyo sincero; en especial a mi hermana por sus consejos, su constante motivación y amor

... A mi esposo, por su amor leal, paciencia y apoyo en cada momento

... A toda nuestra familia, aquí presente y los que ya no nos acompañan, amigos y maestros por ser parte esencial de este proceso

... A mi compañero/a de tesis, por su apoyo y dedicación para que juntos culmináramos esta meta

... A nuestro director de tesis MSc. Fabio Hernán Realpe Martínez por su acompañamiento en la trayectoria de la carrera y de esta investigación

... A nuestra Alma Mater, La Universidad del Cauca, que nos forjó como unos profesionales íntegros y capaces

Resumen

Los agricultores requieren monitorear y analizar constantemente los niveles de macronutrientes del suelo con el fin tomar decisiones acertadas en el uso de fertilizantes y el proceso que conlleva el desarrollo del cultivo; entre dichos macronutrientes se encuentran, el Nitrógeno, el Fósforo y el Potasio (NPK) que son importantes al momento de garantizar el manejo sostenible y el incremento de la capacidad de producción de los cultivos, contribuyendo a la satisfacción de la demanda en relación a la cantidad y calidad de los productos agrícolas. El objetivo de este proyecto es determinar mediante un sistema electrónico, el porcentaje de las variables Nitrógeno, Potasio y Fósforo NPK en sitio, para verificar las propiedades intrínsecas de un terreno. Se utilizó una metodología conformada por tres fases: la implementación de un sistema que permitiera adquirir las variables NPK en un terreno de prueba, la evaluación de las propiedades del terreno de prueba con respecto a las variables NPK y la determinación del grado de fertilidad del terreno de prueba. A través de una prueba de campo en una parcela agrícola se obtuvo los datos a través del hardware y el software diseñado (Aplicación móvil), obteniendo dentro de los resultados las variables NPK del suelo, así como los posibles errores en la implementación del cultivo a fin de contribuir con mejoras en la producción agrícola. Se pudo concluir que el dispositivo permite verificar la ausencia o el exceso a partir del análisis de las características físicas de la planta y la fertilidad del suelo, a fin de brindar oportunidades de proyección y rendimiento para futuros cultivos.

Palabras Clave: Aplicación movil, NPK, fertilidad, suelos, macronutrientes.

Abstract

Farmers need to constantly monitor and analyze the levels of macronutrients in the soil in order to make the right decisions in the use of fertilizers and the process involved in the development of the crop; among these macronutrients are Nitrogen, Phosphorus and Potassium (NPK), which are important to ensure sustainable management and increase the production capacity of crops, contributing to the satisfaction of the demand in relation to the quantity and quality of agricultural products. The objective of this project is to determine, by means of an electronic system, the percentage of the variables Nitrogen, Potassium and Phosphorus NPK on site, to verify the intrinsic properties of a soil. A methodology consisting of three phases was used: the implementation of a system to acquire the NPK variables in a test plot, the evaluation of the properties of the test plot with respect to the NPK variables, and the determination of the degree of fertility of the test plot. Through a field test in an agricultural plot, the data was obtained through the designed hardware and software (Mobile App), obtaining within the results the NPK variables of the soil, as well as the possible errors in the implementation of the crop in order to contribute with improvements in agricultural production. It was possible to conclude that the device allows to verify the absence or excess from the analysis of the physical characteristics of the plant and soil fertility, in order to provide opportunities for projection and yield for future crops.

Key Words: Mobile App, NPK, fertility, soil, macronutrients.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El presente trabajo pretende realizar una contribución importante al progreso de los sectores agrícolas, específicamente en los procesos de monitorización de los niveles de los macronutrientes NPK, necesarios para una adecuada toma de decisiones en el proceso del desarrollo de un cultivo.

Se implementarán procesos más ágiles en las condiciones actuales de los procedimientos para la verificación de las variables NPK, apoyadas en la correcta gestión de la información obtenida a partir de un Sistema Electrónico y Automático.

En este sentido, el proyecto presentará aportes al crecimiento del departamento en la adopción de tecnologías IOT y métodos de gestión de información que incrementarán la eficiencia, la seguridad, y que también permitirán el análisis de datos en las actividades de cultivo en el sector agrícola. Además, de dar paso a futuros proyectos que busquen apoyar diferentes procesos donde se puedan desarrollar dispositivos electrónicos configurados para la medición de propiedades intrínsecas similares, así como el desarrollo de aplicaciones móviles y la gestión de bases de datos.

1.1. Trabajos Relacionados

En los últimos años la agricultura mundial se ha vuelto significativamente efectiva, en efecto de la mejora en sus sistemas de producción; entre los desafíos que enfrenta

el sector de la agricultura se destacan: el manejo sostenible y el incremento en la capacidad de producción en los cultivos [1], esto con el fin de generar producciones que satisfagan la demanda en relación de cantidad y calidad a nivel local, nacional e internacional. No obstante, para cubrir esta necesidad, se debe hacer una gestión de las variables fisicoquímicas en los suelos de cultivo importantes para su desarrollo [2]; entre ellas, las más representativas en su dimensión nutricional son el pH y los macronutrientes Nitrógeno, Fósforo y Potasio (NPK) [3] en conjunto con las variables agroclimáticas como: la temperatura, precipitación, humedad, radiación solar, etc.[1] En particular, las variables NPK juegan un papel fundamental, puesto que sus concentraciones varían según la región y el entorno circundante al área de la planta[4]; estos macronutrientes le permiten desarrollarse óptimamente [5]: el nitrógeno es el encargado de apoyar el adecuado crecimiento de la planta; el fósforo es un elemento relativamente estable en el suelo, ayuda a la transferencia de la energía de la luz solar a la planta, estimulando su madurez y el crecimiento temprano de sus raíces; el potasio incrementa la protección contra posibles enfermedades y plagas, ayudando a la formación y circulación de compuestos orgánicos, que estimulan el tamaño del fruto y/o de la flor [6][7]. En el mismo sentido, el rendimiento en la producción de los cultivos está relacionado directamente con la fertilidad de su suelo, por lo tanto, es primordial disponer de la adecuada gestión de las variables previamente mencionadas, debido a que en las temporadas de crecimiento, los cultivos requieren de porcentajes variables en sus nutrientes[8], para lo cual es necesario realizar toma de muestras del suelo y posteriormente efectuar un costoso y extenso proceso de laboratorio que puede llevar desde uno hasta dos meses[9]; es por esta razón que es poco común que los agricultores lo lleven a cabo de esta manera, generalmente, optan por realizar un análisis empírico para la toma de decisiones, y en consecuencia ajustan inoportunamente los niveles de nutrientes en el suelo[3], impactando negativamente su entorno que en efecto, no solo contamina la tierra y eventualmente las plantas, sino que también amenaza la salud humana[10], disminuye la calidad del suelo y altera el rendimiento de los insumos, acción reflejada en una falta de control en el presupuesto [2][11][12], por lo que es necesario un sistema de medición que permita verificar el contenido de nutrientes NPK de una forma rápida y precisa. [5] Sumado a esto, la falta de desarrollos tecnológicos que permitan gestionar la información de los procesos agrícolas en su totalidad, se está convirtiendo en un limitante para el desarrollo de las tareas agrícolas.[13] Sin embargo, los países en vía de desarrollo buscan que sus agricultores se adapten a maneras más inteligentes para efectuar sus

labores en campo apoyados en tecnologías IOT y métodos de detección NPK tales como: ópticos, electroquímicos, acústicos, eléctricos, electromagnéticos, y mecánicos [14][15], dando paso a un aumento en el número de proyectos afines, para apoyar las actividades agrícolas, algunos de ellos han sido consignados en la *Figura 1.1*.

Nombre	Lugar de origen	Costo	Precisión Promedio	Funcionamiento en Campo	Portabilidad	Interfaz de Usuario (Mobile App)	Tiempo promedio de procesamiento	Hardware y/o software	Tecnologías incorporadas
Prototipo de medición de nutrientes NPK en suelos	Departamento de ciencias de computación y electrónica de la Universidad Gadjah Mada, Indonesia	No definido	84,16%	No experimentado	No	No	0,64 s	Ambos	Redes neuronales artificiales, reconocimiento óptico, procesamiento de imágenes LBP
Monitoreo Paramétrico del ambiente de las granjas	Departamento de Electrónica e Ingeniería de Comunicación de la Universidad PES, Bangalore, India	No definido	95%	Sí	No	Sí	Tiempo Real	Ambos	LoRa, Arduino, Transductor óptico, Páneles solares
Analizador automático de macronutrientes en el suelo usando sistemas embebidos	Departamento de Electrónica e ingeniería de Telecomunicación K. J. Somaiya Instituto de Ingeniería, Mumbai, India.	No definido	No definida	No	No	Display LCD	No definido	Hardware	Atmel, sensor de color, sensor de Ph
Agribot basado en microcontroladores para la fertirrigación y la plantación	Departamento de Instrumentación y Control, Instituto de Ingeniería, Pune, India.	No definido	No definida	Sí	Sí	Sí	No definido	Sí	Arduino, bluetooth, Wi-Fi, comunicación GSM, Blink, sensor NPK, sensor de humedad
Circuito de comparación de voltaje basado en un sensor de color policromático del dispositivo de control de pH y nutrientes del suelo para la recomendación de fertilizantes	Universidad Politécnica de Filipinas, Manila, Filipinas	Económico	93.75%	No	No	Display LCD	45.4 s – 5 m	Hardware	Raspberry Pi, Sensor policromático
Detección de nutrientes de nitrógeno, fósforo y potasio (NPK) del suelo usando un transductor óptico	Facultad de Ingeniería Eléctrica Universidad Teknologi Mara (UiTM) Shah Alam, Selangor, Malasia	Económico	No definida	No	No	Display LCD	No definido	Hardware	Arduino, Transductor óptico
Sensor óptico integrado para la detección de nutrientes de suelo NPK	Facultad de Ingeniería Eléctrica Universidad Teknologi Mara (UiTM) Shah Alam, Selangor, Malasia	No definido	80%	No	No	Monitor	No definido	Hardware	Arduino, Sensor óptico
Detección de fertilizantes N, P, K en suelos agrícolas con técnica de absorción láser NIR 1	Universidad de Mumbai Mumbai, India.	No definido	No definida	No	No	Monitor	> 5 h	Ambos	DAQ Board, Fotodetector, IR Laser, Interferómetro

Figura 1.1: Trabajos relacionados, Fuente: Propia

Por otro lado, el sector agricultor, manifiesta la necesidad de incorporar sistemas basados en GPS, esta importante tecnología hace parte de sistemas más complejos como los Sistemas de Información Geográfica (SIG) los cuales almacenan, analizan y gestionan grandes cantidades de datos y atributos asociados a análisis espaciales haciendo uso de referencias geográficas, que en este caso brindarían un soporte más sólido proporcionando una mejor caracterización del terreno de cultivo. [16][17][18] La investigación del estado del arte permite conocer gran cantidad de estudios con relación al tema, planteando diferentes soluciones a las necesidades de los agricultores permitiendo un mejor análisis del tema, además se evidencia el esfuerzo por mejorar los sistemas de gestión de cultivos agrícolas y es en este sentido que la propuesta actual pretenda contribuir con el tema en una ciudad donde este tipo de estudios son limitados.

1.2. Planteamiento del problema

Se prevé que para el 2050 la población mundial crecerá en un 34 %, lo que equivale a una cifra total de 9100 millones de personas, esto se verá reflejado en el incremento de la demanda de los recursos que suplen las necesidades vitales de subsistencia, como la salud y la alimentación [19] En el caso de la alimentación, la mayoría de la población mundial localizada en países en vía de desarrollo, depende actualmente de la agricultura local, cada zona es apta para producir una gran variedad de cultivos según sus condiciones climáticas y la composición de los macronutrientes en sus suelos: nitrógeno, fósforo y potasio (NPK) [20] esta última, está directamente relacionada con su fertilidad, por lo tanto, es primordial disponer de su adecuada gestión [8]; sin embargo, el potencial de los recursos locales para responder a la creciente demanda alimentaria, se encuentra limitado a causa de las condiciones tecnológicas existentes [20]. Las formas tradicionales de tratar los cultivos han mantenido la calidad y cantidad del producto en un mismo punto [13]; para estar en una constante evolución, algunos grandes productores han comenzado a implementar modelos de agricultura inteligente que presentan información de compleja interpretación, limitada e incompleta respecto a diferentes necesidades como geolocalización, portabilidad, almacenamiento de datos, entre otras [15]. Sin embargo, los pequeños agricultores, son pertenecientes al 75 % de la población de bajos recursos de los países en vía de desarrollo, viven en zonas rurales y su economía depende directa e indirectamente de la agricultura [19]; por lo tanto, acceder a este tipo de tecnologías, resulta en una situación difícil para ellos, por cuestiones de costos, dificultad para manejar adecuadamente estos sistemas e interpretación de la información obtenida; lo que significa a largo plazo realizar grandes inversiones de tiempo y dinero que demandan los complejos estudios de suelos en laboratorio; por eso, en su lugar, seguir utilizando métodos tradicionales y empíricos que obstaculizan el proceso de producción [21][22]. En este contexto, teniendo identificado el problema y para dar claridad acerca de las necesidades identificadas en el proyecto se plantea la siguiente pregunta de investigación: ¿Cómo monitorear las variables NPK (Nitrógeno, Fósforo y Potasio) en sitio para interpretar las características de un terreno respecto a métodos convencionales como pruebas de suelos en laboratorios especializados para la verificación de los porcentajes de NPK?

1.3. Objetivos

1.3.1. Objetivo General

Determinar mediante un sistema electrónico el porcentaje de las variables Nitrógeno, Potasio y Fosforo NPK en sitio para verificar las propiedades intrínsecas de un terreno.

1.3.2. Objetivos Especificos

- Implementar un sistema que permita adquirir las variables NPK en un terreno de prueba.
- Evaluar las propiedades del terreno de prueba con respecto a las variables NPK.
- Determinar el grado de fertilidad del terreno de prueba.

1.4. Estructura de la monografía

Este documento se divide en 4 capítulos: Introducción, Materiales y métodos, Experimentación y resultados, y finalmente, Discusión y conclusiones, en los cuales se explica detalladamente los aspectos más relevantes para este trabajo.

De esta manera, el capítulo 1 contiene la introducción del trabajo donde se contextualiza la problemática de investigación, el planteamiento del problema, objetivos y trabajos relacionados con el tema abordado. En el capítulo 2 se describen los materiales y métodos utilizados para el desarrollo de los objetivos, así como la selección del software, bases de datos y diagramas de modelado para el software. En el capítulo 3 se presenta la evidencia de la aplicación del software con los resultados de las pruebas aplicadas a la población de estudio, así como el análisis de cada uno. Finalmente, en el capítulo 4, se describe la discusión y las conclusiones derivadas del trabajo de investigación.

Capítulo 2

Materiales y métodos

2.1. Metodología empleada para el desarrollo del trabajo

El desarrollo del trabajo se estructura sobre una metodología que consta de tres fases, cada una de las cuales tiene gran importancia en el momento de cumplir con los objetivos planteados en el proyecto; las fases se ejecutaron de manera secuencial, cumpliendo con una lógica funcional y paralelo a ellas se desarrollaron actividades asociadas al ordenamiento, documentación y redacción del proyecto. A continuación se describen las actividades de cada fase.

Fase 1

Objetivo: Implementar un sistema que permita adquirir las variables NPK en un terreno de prueba.

Actividades:

1. Búsqueda, selección y análisis de información relacionada con dispositivos electrónicos para la medición de las variables
2. Definición de los requerimientos técnicos.
3. Definición de los componentes a integrar en el dispositivo

4. Definición del diseño
5. Adquirir los componentes necesarios
6. Incorporación de los componentes
7. Incorporación de estructura de protección al dispositivo
8. Documentación del prototipo
9. Análisis de requerimientos para el desarrollo de la aplicación móvil
10. Desarrollo del software
11. Selección del terreno de prueba
12. Pruebas de funcionamiento lógico

Fase 2

Objetivo: Evaluar las propiedades del terreno de prueba con respecto a las variables NPK.

Actividades:

1. Programación de la salida de campo para toma de muestra
2. Procesamiento de muestra en el laboratorio
3. Procesamiento de muestra con el dispositivo
4. Registro de la prueba
5. Verificación de errores
6. Corrección y ajuste de resultados

Fase 3

Objetivo: Determinar el grado de fertilidad del terreno de prueba.

Actividades:

1. Programación de la salida de campo
2. Puesta en marcha del dispositivo en campo
3. Adquisición de datos para geolocalización en mapa satelital
4. Mapeo cromático del terreno para determinación de porcentajes NPK
5. Programación de la salida de campo para ajuste final
6. Corrección y ajuste de resultados
7. Documentación Oficial del funcionamiento
8. Desarrollo de los documentos requeridos por parte de la institución.
9. Redacción del informe final.

2.2. Diseño e implementación del Sistema

2.2.1. Definición de requerimientos Hardware e Ingeniería de producto

Para que el dispositivo cumpla su objetivo principal, es necesario que el usuario final (agricultor), tenga la posibilidad de interactuar con un conjunto de interfaces gráficas que le permitan recolectar la información, visualizarla e interpretarla de una forma sencilla e intuitiva; contenidas en un entorno de fácil manejo, que apoye en cualquier momento, la resolución de una tarea determinada, la toma de decisiones y el seguimiento a las medidas tomadas a lo largo del proceso de cultivo.

Para el desarrollo del sistema hardware se hizo uso de una metodología de diseño y desarrollo de productos, basada en 5 pasos que permite identificar el conjunto de necesidades del usuario con el objetivo de asegurar que el producto se enfoque en sus necesidades, tanto latentes u ocultas, como explícitas, especificando una base de datos que justifique las especificaciones del producto, creando un registro de archivos de la actividad de necesidades del proceso de desarrollo, asegurando que no falte o no se olvide ninguna necesidad crítica del usuario[23]. El proceso de identificar las necesidades del usuario, es parte integral del proceso de desarrollo del producto; los pasos para realizar adecuadamente esta identificación son:

1. Recopilar datos sin procesar de los usuarios.
2. Interpretar los datos sin procesar en términos de las necesidades de usuarios.
3. Organizar las necesidades en una jerarquía de necesidades primarias, secundarias y, de ser necesario, terciarias.
4. Establecer la importancia relativa de las necesidades.
5. Reflexionar en los resultados y el proceso.

Como canal de información, se decidió realizar entrevistas en el ambiente de uso del sistema para entender profundamente las necesidades del usuario, con el fin de obtener una mejor perspectiva y definir los requerimientos de la manera más acertada para el correcto desarrollo del producto.

Entrevistas: Se consideró de suma importancia recolectar información de las necesidades de los agricultores desde la comunicación directa con ellos, a través de la *Asociación de Productores Orgánicos Nuevo Futuro*, para esto se diseñó una entrevista, mostrada en *Tabla 2.1*

Entrevista dirigida a los agricultores de la Asociación de Productores Orgánicos Nuevo Futuro para el desarrollo del Proyecto: **"Sistema electrónico y automático para la determinación de los porcentajes de las variables de Nitrógeno, Fósforo y Potasio (NPK) en suelos"**
Nombre: Dirección: Teléfono: Entrevistador(es): Luis Cobo y Luiza Narváez Fecha: Dispuesto a otra llamada ○Sí ○No
¿Cuáles cree usted, son los nutrientes más importantes para el buen desarrollo de sus cultivos?
Rta:
¿Qué otros factores o variables usted considera importantes y que se deben tener en cuenta para saber el estado de su cultivo?
Rta:
¿Qué técnicas emplea en el momento de definir la cantidad de fertilizante que requiere su cultivo?
Rta:
¿Con qué frecuencia realiza usted pruebas de laboratorio para confirmar los niveles de las propiedades presentes en su cultivo?
Rta:
¿Que factores lo limitan a realizar recurrentes pruebas de laboratorio a su cultivo?
Rta:
¿Con qué facilidad interpreta los resultados de las pruebas de laboratorio?
Rta:
¿Conoce usted algún otro sistema que sea capaz de medir estas variables y entregar una información certera de ellas? ¿Qué ventajas y desventajas considera usted que tienen estos sistemas?
Rta:

Tabla 2.1: Formato para entrevista, Fuente:Propia

Después de realizar un análisis de las necesidades identificadas a partir de las respuestas, los entrevistados manifestaron que los nutrientes más importantes para el desarrollo de un cultivo son los macro-nutrientes nitrógeno, fósforo y potasio (NPK), además de factores como la temperatura, acidez y humedad.

Por otra parte, exponen que los métodos para verificar los porcentajes de estas variables, basados en las pruebas de laboratorio son costosos y en el tiempo que transcurre entre la toma de la prueba y la entrega de los resultados, las variables presentes en

el suelo se alteran ligeramente y ya no corresponden a los valores actuales exactos a la fecha.

Además de esto expusieron conocer acerca de otros sistemas que funcionan para la verificación de estas variables, sin embargo, estos presentan desventajas en otras características que los hacen productos incompletos y difíciles de adquirir.

A partir de la interpretación de las respuestas obtenidas se definieron los siguientes requerimientos:

- Sistema para medir las variables Nitrógeno, fósforo y potasio
- Sistema para medir temperatura y humedad.
- Tratamiento de datos para una correcta interpretación
- Rápida respuesta
- Sistema económico, ergonómico, liviano, resistente y apariencia como herramienta de calidad profesional.
- Larga duración de la batería.
- Fácil de usar.

2.2.2. Diseño Electrónico

Para construir el hardware es importante tener en cuenta los requisitos identificados en el capitulo anterior, después de un análisis de requisitos se pudieron identificar las características que debía tener el dispositivo hardware para cumplir con los objetivos del proyecto. El dispositivo hardware debe ser capaz de medir las variables: Nitrógeno, fósforo, potasio, temperatura y humedad; procesar estos datos, mostrarlos por medio de una pantalla LCD y sincronizarlos con la aplicación móvil, de igual forma, este debe adaptarse a la escasez de conectividad tipo Wi-Fi en algunos terrenos localizados en lugares donde la conexión a internet es intermitente o nula, y permitirle al usuario almacenar los datos sin importar el estado actual de su conectividad a internet.

Por otro lado, el dispositivo debe ser portátil, con baterías recargables y de alta duración. Sus dimensiones deben ser adecuadas para su ensamble a la cubierta que lo convierte en un dispositivo robusto para las condiciones a las cuales será expuesto (altas temperaturas, agua, y otros factores presentes en los campos de prueba que pueden afectar su correcto funcionamiento).

Inicialmente, se debían encontrar las opciones más adecuadas y económicas en cuanto a sensores correspondientes a la medición de las variables mencionadas. De igual forma, para todos los dispositivos requeridos que permiten el correcto funcionamiento del dispositivo. Después de un análisis de costos y teniendo en cuenta el presupuesto del proyecto se hizo la inversión pertinente para conseguir los siguientes dispositivos:

- Sensor NPK
- Sensor de humedad Hd - 38
- Modulo RS485 Modbus
- Fuente 24V DC
- Modulo i2c para LCD
- LCD 16 X 2
- Modulo para tarjeta MicroSD
- Pulsadores
- Modulo ESP32 WROOM DEV KIT 1
- Sensor de temperatura MAX6675
- Placa de desarrollo Arduino Nano

El sensor NPK, mostrado en *Figura 2.1* no requiere un reactivo químico para realizar la prueba de suelo, contrario a las pruebas de laboratorio; además tiene una rápida respuesta, y puede ser usado en conjunto con un amplia gama de microcontroladores.

El sensor requiere de una alimentación de 9-24v DC , es un dispositivo de bajo consumo y utiliza un módulo de comunicación RS485 MODBUS para la transferencia de datos al microcontrolador[24].

	Line Color	Description
Power	Brown	Power supply Positive (12-24V DC)
	Black	Power supply Negative
Communication	Yellow (grey) color	485-A
	Blue	485-B

Figura 2.1: Sensor NPK. Fuente: [24]

Como se ha mencionado anteriormente, el sensor utiliza un protocolo de comunicación RS485 Modbus-RTU a IC/232 por lo que la información hace uso de ciertos parámetros de comunicación específicos que definen el formato de la trama para el envío de datos[25].

La trama definida es:

0x01,0x03, 0x00, 0x1e, 0x00, 0x01, 0xe4, 0x0c;

0x01,0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc;

0x01,0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xc0;

Donde cada fila corresponde a las tramas de las variables N, P, K

Por la razón anterior, se debe contar con un modulo RS485 Modbus, *(ver Figura 2.2)*, el cual se eligió con la característica de que pudiera conectarse a tarjetas de microcontroladores relativamente económicas.

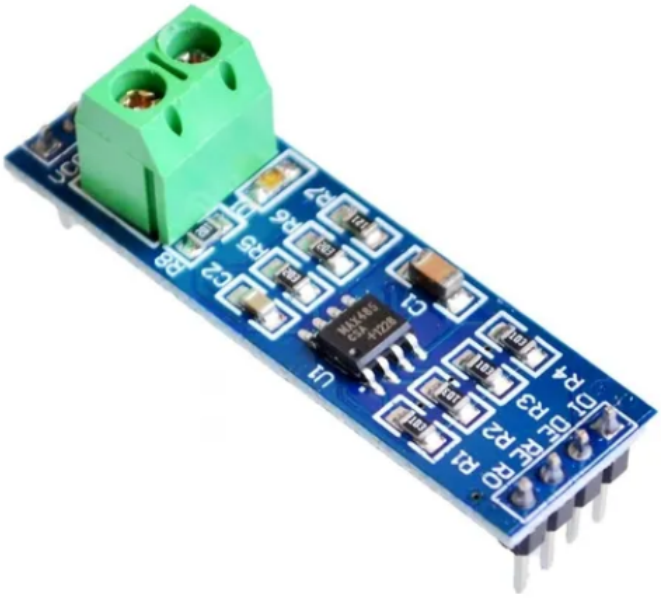


Figura 2.2: Módulo conversor RS485. Fuente:[25]

El módulo debe comunicarse a una tarjeta de desarrollo que procese los datos del sensor NPK; la placa elegida fue Arduino Nano, (*Figura 2.3*), debido a su pequeña dimensión y funcionalidad[26]. Su programación se basó en obtener los datos mediante el RS485 y almacenarlos en una variable de 8 bytes, procesando los datos y usando la trama definida anteriormente para registrar los valores convertidos de hexadecimal a decimal.

Figura 2.3: Tarjeta de desarrollo Arduino Nano. Fuente:[26]

Del mismo modo, se hizo uso de un sensor de temperatura conformado por una termocupla y un módulo de transmisión MAX6675, el cual convierte la señal analógica en digital. También posee una interfaz de comunicación SPI (Serial Pheriperal Interface) para conectarse al MCU[27].

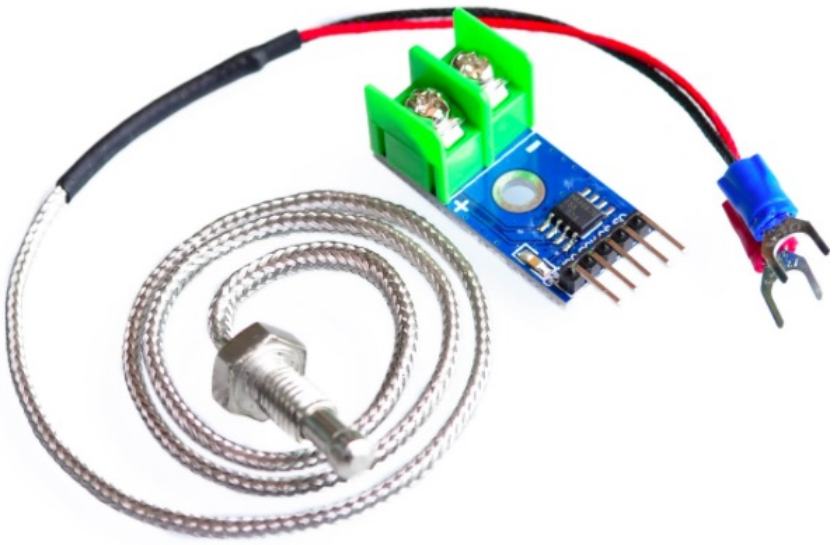


Figura 2.4: Módulo con termocupla MAX6675. Fuente:[27]

Por otra parte, se hizo la conexión de un sensor de humedad, el cual se conectó a un modulo ESP32; para ello, se eligió el módulo ESP32 WROOM DEV KIT debido a que este módulo permite el uso de Wi-Fi y diferentes interfaces como la SPI, entradas analógicas y digitales. Del mismo modo es de bajo costo y consumo ademas de utilizar el IDE de arduino.[28], por tales motivos, fue la opción mas favorable para la realización del proyecto.

Figura 2.5: Módulo ESP32. Fuente:[28]

En cuanto al sensor de humedad, específico para suelos, este funciona a un voltaje directo de 5 voltios y permite la lectura de humedad relativa siendo directamente conectado una entrada analógica del módulo ESP32[29].

Figura 2.6: Sensor de Humedad. Fuente:[29]

Una vez habiendo medido las variables, era importante mostrarle al usuario los valores obtenidos, para eso se hizo uso de una pantalla LCD.

Al contar con dos placas diferentes y teniendo en cuenta que en cada una habían distintos dispositivos integrados, estos se conectaron formando una estructura maestro - esclavo, donde el ESP32 funcionaba como maestro y usaba comunicación serial para la transferencia de datos al Arduino Nano; la pantalla LCD se conectó al ESP32 mediante el uso de un módulo interfaz I2C para el funcionamiento de la misma, y que ocupa una menor cantidad de pines en el microcontrolador[30].

Figura 2.7: Módulo I2C conectado a pantalla LCD. Fuente:[30]

Según[31], las zonas rurales al estar dispersas en las regiones, hacen que la cobertura a internet existente sea insuficiente, efecto de las bajas inversiones de las empresas de telecomunicaciones en estas zonas. Por tal motivo, el dispositivo fue diseñado para almacenar datos independientemente de si el usuario cuenta o no con una conexión a internet estable, integrando un módulo para lectura y escritura de tarjetas micro SD almacenando los datos en un formato .txt para posteriormente hacer uso de ellos. Del mismo modo, el dispositivo funciona conectado a internet y almacena automáticamente los valores en la base de datos[32].

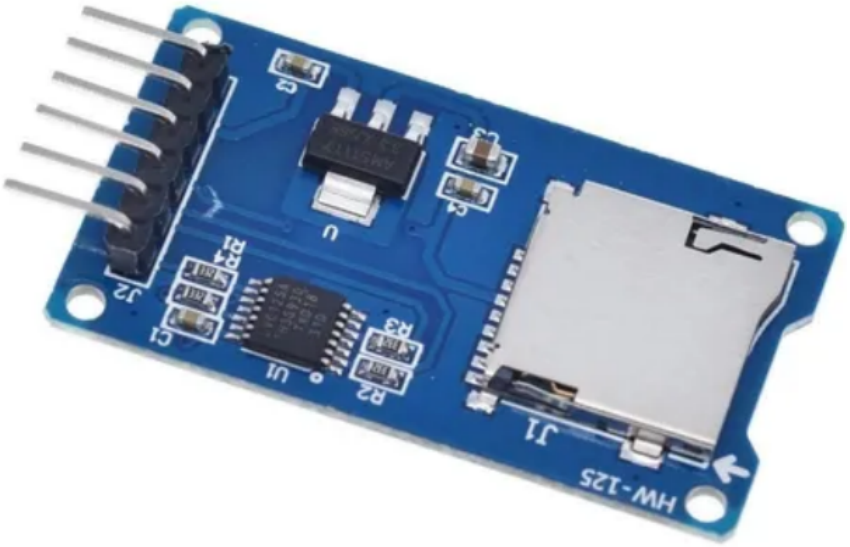


Figura 2.8: Módulo para chip MicroSD. Fuente[32]

Para almacenar un dato, ya sea on-line u off-line, se usó una interrupción por medio de un botón programado con una resistencia interna que permite almacenar los datos en el formato mostrado en *Tabla 2.3*:

Fecha	N	P	K	Temperatura	Humedad
-	-	-	-	-	-
-	-	-		-	-

Tabla 2.3: Formato para datos almacenados. Fuente: Propia

Mientras que en la nube se almacena en formato JSON *(ver Figura 2.9)*.


```
{
  "crops": [],
  "Create_at": "7/12/2021",
  "_id": "61aff7224f10c331a06b4172",
  "Humedad": 70.92086,
  "N": 3,
  "P": 1,
  "K": 0,
  "Temp": 23.5,
  "__v": 0
},
```


Figura 2.9: Formato JSON. Fuente: Propia

En *Figura 2.10* se muestra el diagrama de conexión del dispositivo:

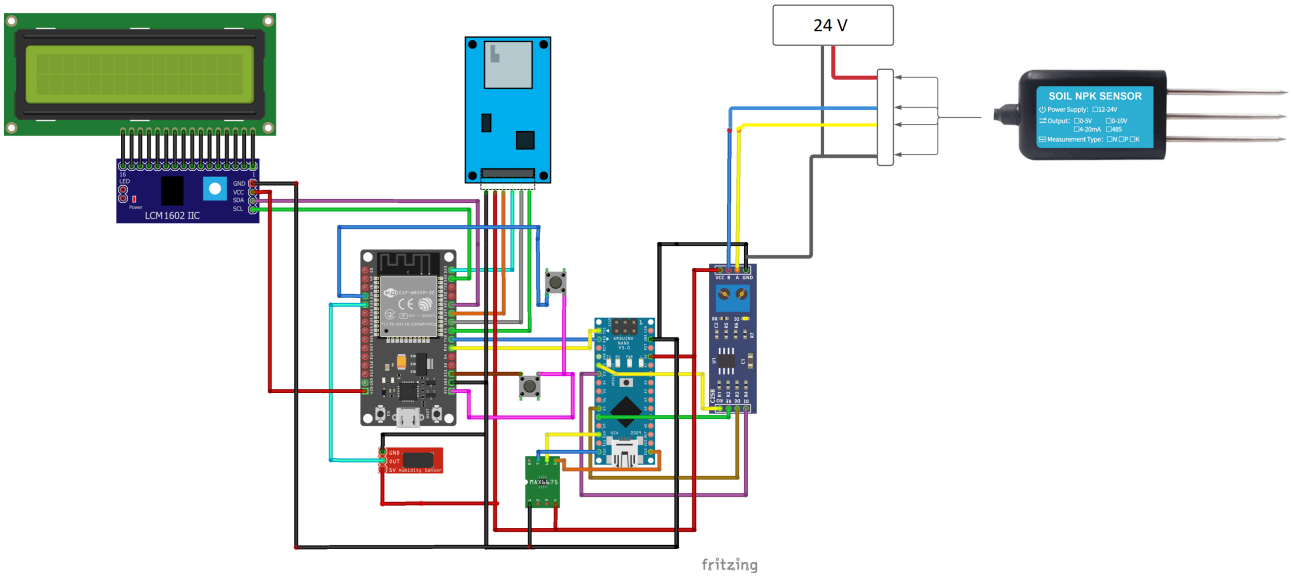


Figura 2.10: Diagrama de conexión del dispositivo. Fuente: Propia

2.2.3. Calibración de sensores

Para la calibración de los sensores se hizo uso de diferentes métodos que permitieron ajustar las lecturas por medio de factores de conversión, regresión lineal, prácticas experimentales, entre otros, los cuales son mostrados a continuación:

- **Sensor NPK:**

Para el sensor NPK, se requiere calibrar cada variable (Nitrógeno, fósforo y potasio). El laboratorio de la Secretaria de Agricultura y Desarrollo Rural facilitó resultados de análisis de suelos de las muestras 48173, 48172, 827, 828 y de abono 47761; así mismo ejemplares de las muestras, dispuestas posteriormente en vasos y tubos de ensayo para así comparar las medidas con el sensor y ajustar cada uno de los elementos NPK. La toma de muestras es evidenciadas en *Figura 2.11* y *Figura 2.12*

(a) Prueba N° 47761 (b) Prueba N° 827

Figura 2.11: Pruebas tomadas en el laboratorio de la Secretaria de Agricultura y Desarrollo Rural. Fuente: Propia

(a) Empaque de pruebas

(b) Fotografía en el laboratorio

Figura 2.12: Pruebas tomadas en el laboratorio de la Secretaria de Agricultura y Desarrollo Rural. Fuente: Propia

El proceso de medición fue realizado con asesoría de personal del laboratorio.

A continuación se presenta la comparación de los valores entregados por el sensor NPK y las pruebas realizadas en el laboratorio para cada uno de los elementos:

1. **Nitrógeno**

   Inicialmente se realizó la conversión de los valores registrados por el sensor, puesto que estos son entregados en términos de 10^4 y en unidades de mg/kg según [24] (*Ver Tabla 2.4*):

Nitrógeno						
Unidades/Muestra	47761	481742	48173	827	828	Abono
Laboratorio (%)	93,05	6,25	5,19	8,79	81,06	91,5
Sensor (%)	87	3	5	10	62	98

Tabla 2.4: Calibración Nitrógeno. Fuente: Propia

Posteriormente, se compararon con los datos obtenidos en el laboratorio de la siguiente manera para realizar el cálculo del factor de conversión:

$$\frac{93,05}{87} = 1,067 \tag{2.1}$$

$$\frac{6,25}{3} = 2,08 \tag{2.2}$$

$$\frac{5,19}{5} = 1,038 \tag{2.3}$$

$$\frac{8,79}{10} = 0,879 \tag{2.4}$$

$$\frac{81,06}{62} = 1,3 \tag{2.5}$$

$$\frac{91,5}{98} = 0,934 \tag{2.6}$$

Promedio:

$$\frac{1,067 + 2,08 + 1,038 + 0,879 + 1,3 + 0,934}{6} = 1,216 \tag{2.7}$$

2. **Fósforo** *(Ver Tabla 2.5)*

Fósforo						
Método/Muestra	47761	481742	48173	827	828	Abono
Laboratorio (mg/kg)	33	0,62	1	5,97	30	58
Sensor NPK (mg/kg)	18	1	2	3	17	32

Tabla 2.5: Calibración Fósforo. Fuente: Propia

Para este elemento se realizó el cálculo del factor de conversión de la siguiente manera:

$$\frac{33}{18} = 1,83 \tag{2.8}$$

$$\frac{1,62}{1} = 1,62 \tag{2.9}$$

$$\frac{3,9}{2} = 1,95 \tag{2.10}$$

$$\frac{5,97}{3} = 1,99 \tag{2.11}$$

$$\frac{30}{17} = 1,76 \tag{2.12}$$

$$\frac{58}{32} = 1,81 \tag{2.13}$$

Promedio:

$$\frac{1,62 + 1,95 + 1,99 + 1,76 + 1,81}{6} = 1,826 \tag{2.14}$$

3. **Potasio**

   Los valores entregados por el laboratorio se encuentran en meq/100g de tierra y deben ser convertidos a mg/kg (unidades del sensor) *(Ver Tabla 2.6)*:

$$1meq/100gdetierra = 391mg/kg \tag{2.15}$$

Potasio						
Unidad/Muestra	47761	481742	48173	827	828	Abono
Laboratorio (meq/100g de suelo)	0,568	0,0098	0,028	0,546	0,299	0,543
Conversión a mg/kg	222,08	3,86	11,05	213,38	117,16	212,44

Tabla 2.6: Calibración Potasio. Fuente: Propia

Posteriormente se compararon estos valores con el sensor *(Ver Tabla 2.7)*:

Potasio						
Método/Muestra	47761	481742	48173	827	828	Abono
Laboratorio (mg/kg)	222,08	3,86	11,05	213,38	117,16	212,44
Sensor (mg/kg)	43	1	2	44	26	43

Tabla 2.7: Calibración Potasio. Fuente: Propia

Para este elemento se realizó el cálculo del factor de conversión de la siguiente manera:

$$\frac{222,08}{43} = 5,16 \tag{2.16}$$

$$\frac{3,86}{1} = 3,86 \tag{2.17}$$

$$\frac{11,05}{2} = 5,525 \tag{2.18}$$

$$\frac{213,38}{44} = 4,85 \tag{2.19}$$

$$\frac{117,16}{26} = 4,5 \tag{2.20}$$

$$\frac{212,44}{43} = 4,94 \tag{2.21}$$

Promedio:

$$\frac{5,16 + 3,86 + 5,525 + 4,85 + 4,5 + 4,94}{6} = 4,805 \tag{2.22}$$

- **Sensor de Humedad del suelo Hd-38:**

Para la calibración del sensor de humedad, se siguió el procedimiento del paño húmedo[1]: inicialmente se obtuvo el valor del sensor en un ambiente seco para fijar el cero correspondiente al 0 % de humedad relativa, más adelante, se dejó reposar el sensor por aproximadamente una hora, envuelto en un paño completamente húmedo, hasta que el valor se estabilizara, así se identificó el rango de operación del sensor. Los valores obtenidos se presentan en *Tabla 2.8*:

Parámetros de calibración - Sensor de Humedad Hd - 38	
Parámetro	**Valor**
Rango	780 - 4095
Span	3315

Tabla 2.8: Parametros de calibración - sensor humedad. Fuente: Propia

Es importante resaltar que este sensor tiene una conducta inversamente proporcional a la variable medida, por lo tanto su comportamiento resulta en una curva con pendiente negativa, donde cada unidad porcentual equivale a 33,15 según su span:

$$\frac{3315}{100} = 33,15 \tag{2.23}$$

Por lo tanto, la estructura de la ecuación que representa el valor ajustado de una forma directamente proporcional es:

$$a = (n - 780) \tag{2.24}$$

$$100 - (a/33,15) \tag{2.25}$$

Donde n es el valor correspondiente a la salida del sensor

[1]tomado de: https://labexco.com/como-calibrar-un-higrometro/

- **Termocupla Tipo K Max6675:**

Para poder calibrar la Termocupla MAX6675 con el termopar tipo K, se sugiere usar una regresión lineal para programarla en el microcontrolador.[2]

Se incluyó en el código la calibración sugerida, ajustando el coeficiente de posición en 18 para nuestro caso; la ecuación final de calibración se prensenta a continuación:

$$b/2,021142857 - 18 \tag{2.26}$$

Donde b es el valor correspondiente a la salida de la termocupla

Una vez realizada la calibración, era importante convertir el hardware en un dispositivo portátil, por lo tanto se agregó un modulo para regular voltaje, debido a que se requería una alimentación de 12-24v DC para la alimentación del circuito en general[33].

Figura 2.13: Módulo regulador de voltaje DC-DC LM2596. Fuente:[33]

[2]tomado de: https://https://controlautomaticoeducacion.com/microcontroladores-pic/termopar-tipo-k-con-pic/

Para definir la capacidad de la batería encargada de alimentar el circuito integrado, se hizo un análisis de consumo de los dispositivos mostrado en *Tabla 2.9*.

Dispositivo	**Consumo (mA)**
Arduino Nano	15
Esp32 con WIFI	180
Sensor NPK	20
Modulo SD	180
Modulo RS485	0.5
Modulo I2C LCD	250
Modulo MAX6675	1.5
Sensor de Humedad	2.5
Módulo Regulador	10
Módulo Cargador	10
Consumo Total	669.5

Tabla 2.9: Tabla de consumos de los dispositivos. Fuente: Propia

Sabiendo el consumo de mA y voltaje, se seleccionaron las baterías adecuadas para el suministro de energía portátil a la placa. Posteriormente, fue pertinente saber en cuánto tiempo las baterías perderían su autonomía; para esto, se usó una fórmula matemática [3] que encuentra el cociente entre la potencia de la batería y la potencia del circuito, correspondiente la duración de la batería en horas. Las potencias son representadas en *Tabla 2.10*.

	Voltajes (V)	**Corriente (A)**	**Potencia (W)**
Batería	22.2	2	44.4
Circuito	17.5	0.66	11.55

Tabla 2.10: Potencias calculadas. Fuente: Propia

[3] *https://coelectrix.com/calcular-la-autonomia-de-una-bateria*

Se calculó que la duración de la batería en horas es aproximadamente de:

$$\frac{PotenciadelCircuito(VcxIc)}{PotenciadelaBateria(VbxIb)} = \frac{11{,}55}{4{,}44} = 2{,}6horas$$

A partir del análisis anterior, se seleccionaron dos baterías recargables tipo LiPo de 11.1V y 1000mA [34] conectadas en paralelo para obtener el voltaje total necesario.

Figura 2.14: Batería LiPo. Fuente:[34]

Una vez teniendo las baterías, se adquirió un módulo encargado de recargarlas: el módulo cargador de bateria stepdown XL4015[35] *(Ver Figura 2.15)*:

Figura 2.15: Módulo cargador XL4015. Fuente:[35]

Teniendo estos dispositivos funcionando correctamente, se optó por crear una placa PCB para implementar el circuito, con el objetivo de hacerlo más compacto y robusto. El diseño de la placa mostrado en *(Figura 2.16 y Figura 2.17)* fue hecho en el software Eagle.

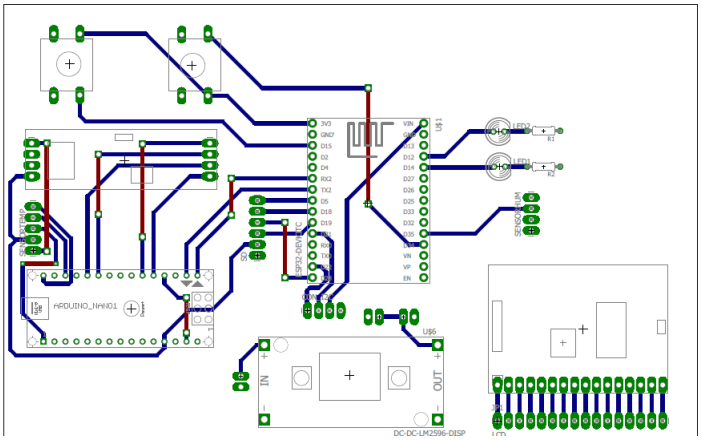


Figura 2.16: Diseño realizado en Eagle. Fuente: Propia

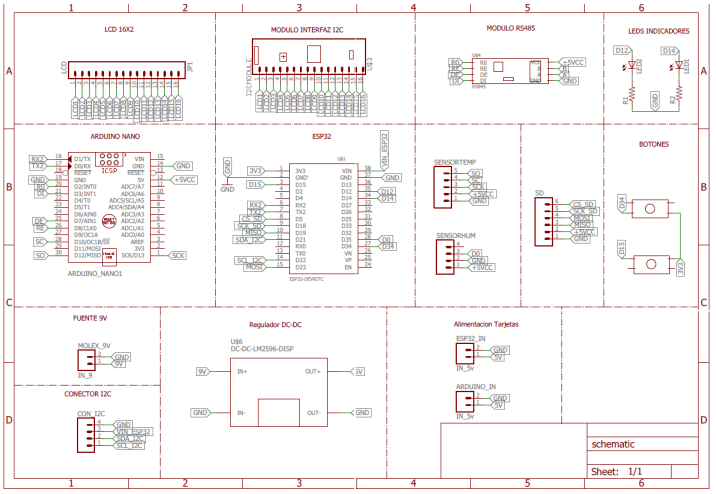


Figura 2.17: Diseño esquemático. Fuente: Propia

Después del proceso de impresión del circuito, soldadura y ensamble de componentes, se midió la continuidad entre los dispositivos y se verificó que todos funcionaran al mismo tiempo, entregando los voltajes necesarios y procesando los datos obtenidos desde los sensores correctamente *(Ver Figura 2.18)*.

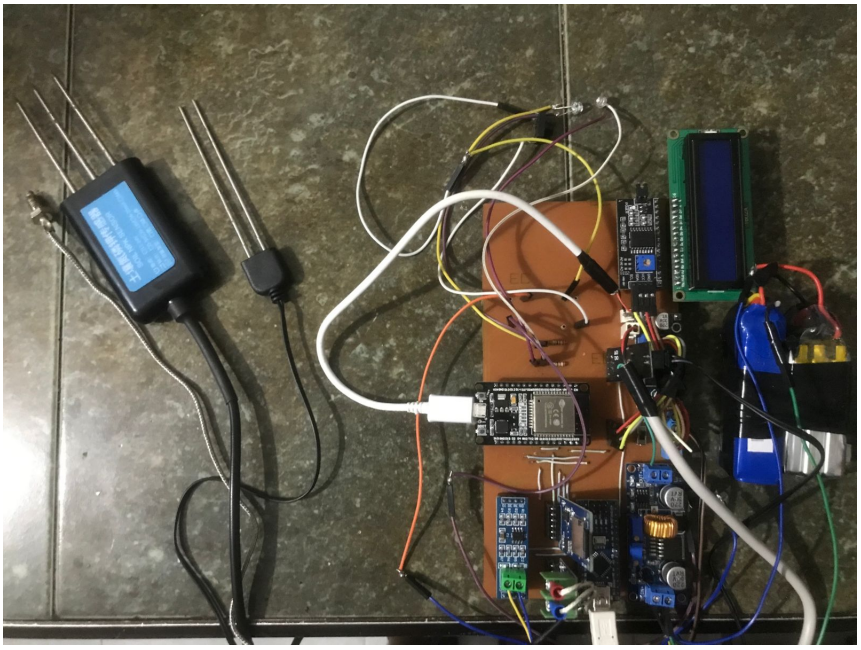


Figura 2.18: Placa PCB ensamblada. Fuente: Propia

Ahora bien, era importante tener una cubierta para la placa con el fin de asegurar la portabilidad cómoda del dispositivo en campo. Para esto se definió como medio de transporte un canguro de doble compartimiento distribuido de la siguiente manera:

- Compartimiento principal: Circuito completo, incorporado en una carcasa de protección contra caídas, polvo y humedad.
- Compartimiento secundario: Salida retráctil únicamente de los extremos de los sensores para facilitar su uso en campo: el usuario solo debe abrir este compartimiento para hacer uso del sistema. Los diseños realizados en SolidWorks se evidencian en *Figura 2.19 y Figura 2.20*

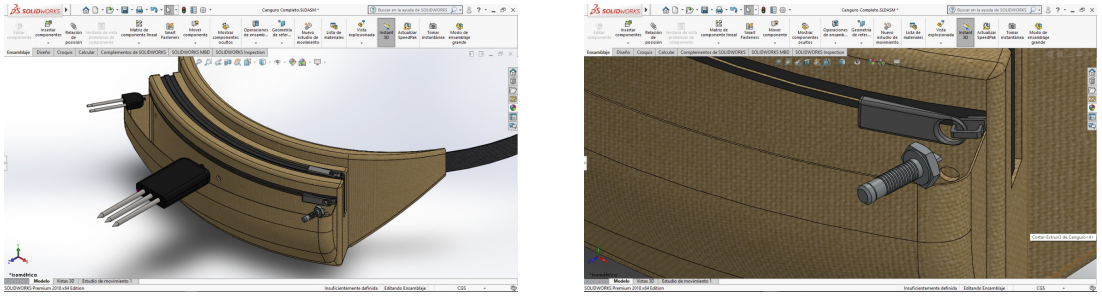


Figura 2.19: Prototipo principal diseñado en SolidWorks. Fuente: Propia

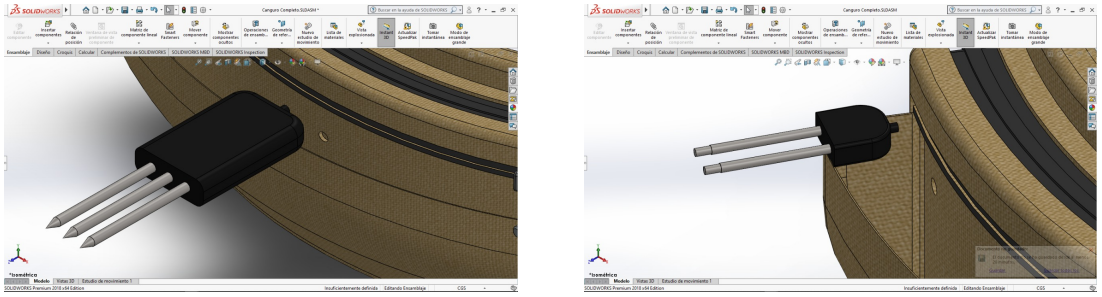


Figura 2.20: Prototipo principal diseñado en SolidWorks. Fuente: Propia

2.2.4. Definición de requerimientos Software

Para el desarrollo de la aplicación móvil, se partió de un análisis de requerimientos basado en la especificación de los requisitos del software (SRS), esta herramienta permite definir las características operacionales del software (función, datos y rendimientos) y establecer las restricciones que debe cumplir, a partir de la obtención de requisitos y el análisis enfocados sólo en la visión del sistema que tiene el usuario. [36] [37]

Las técnicas propuestas por la SRS para recolectar los datos en bruto son:

- Entrevsitas
- Talleres
- Observación
- Encuestas

- Revisión Documental

Para complementar la revisión documental realizada previamente, donde se encontraron necesidades ya especificadas, se decidió proseguir con la realización de encuestas y un proceso de observación por parte de los desarrolladores de la aplicación, para tener una mejor perspectiva y definir los requerimientos de la manera más acertada para el usuario final.

Encuestas: Se consideró de suma importancia recolectar información de las necesidades de los agricultores desde la comunicación directa con ellos, a través de la *Asociación de Productores Orgánicos Nuevo Futuro*, para esto se diseñó una encuesta mostrada en *Figura 2.11*.

Encuesta para los agricultores de la Asociación de Productores Orgánicos Nuevo Futuro para el desarrollo del Proyecto:
"Sistema electrónico y automático para la determinación de los porcentajes de las variables de Nitrógeno, Fósforo y Potasio (NPK) en suelos"

Lea detenidamente cada ítem.
En las preguntas cerradas, se le pide que elija entre varias posibilidades, entonces sólo tendrá que poner una "X" dentro de los cuadrados de las respuestas que haya elegido, es posible elegir más de una opción.
En las preguntas abiertas, usted tiene la posibilidad de expresar libremente su opinión.

¿Cuáles de las siguientes propiedades son las más importantes para el desarrollo de sus cultivos?
○ % Nutrientes ○Textura ○Color ○ % Humedad ○pH
Otros:

¿Cuenta con un dispositivo portátil inteligente como Smartphone, Tablet, teléfono inteligente industrial, entre otros?
○Sí ○No

¿Con qué facilidad maneja usted un dispositivo inteligente?
○Muy Fácil ○Fácil ○Medio ○Difícil ○Muy Difícil
Comentarios:

Independientemente de su habilidad para manejar algún dispositivo inteligente, ¿estaría usted dispuesto a recibir capacitaciones para el manejo de la aplicación?
○Sí ○No
Comentarios:

¿Qué características le gustaría en una aplicación móvil para la gestión de su cultivo? Respuesta:
Para usted, ¿cuál sería la mejor forma de visualizar los datos de su cultivo en pantalla? Respuesta:
¿Alguna vez a utilizada una aplicación móvil similar para monitorear su cultivo? ○Sí ○No En caso de que sí, escriba el nombre de la aplicación:
¿Cómo maneja la información de su cultivo? Respuesta:
¿Cada cuánto actualiza la información de su cultivo y cómo la almacena? Respuesta:
¿Considera útil contar poder observar a través de un mapa satelital, información sobre su cultivo? ○Sí ○No Qué tipo de información?:
¿Cuenta con una conexión estable a internet? ○Sí ○No

Tabla 2.11: Estructura de la encuesta realizada. Fuente: Propia

A partir de la comparación de los resultados obtenidos en las 3 técnicas empleadas podemos identificar las necesidades más importantes del usuario:

- Fácil manejo de la aplicación
- Almacenamiento de datos
- Fácil interpretación de la información almacenada
- Versatilidad del software
- Funcionamiento sin conexión a internet

Basado en la información anterior y para la definición de los requerimientos/requisitos del sistema, se describen los servicios ofrecerá el sistema y las restricciones asociadas a su funcionamiento.
En *Tabla 2.12 se definen los requerimientos funcionales y no funcionales de la aplicación*:

Requisitos del Sistema			
ID	**Funcionales**	**ID**	**No Funcionales**
RF1	Sistema de registro, recuperación de contraseña e inicio de sesión.	RNF1	Disponible para Sistemas operativos con Android 8.0 o superiores.
RF2	Registro de la información básica del cultivo en una Base de Datos	RNF2	La aplicación necesita de mínimo 1 Gb de almacenamiento disponible para ser instalada
RF3	Visualización de la información del cultivo en diferentes intervalos de tiempo, y presentaciones estadísticas	RNF3	Requiere activación y acceso a los servicios GPS del teléfono
RF4	Modificación de la información registrada del cultivo	RNF4	La aplicación debe ser fácil de utilizar
RF5	Eliminar información de los datos del cultivo	RNF5	Disponible en la Play Store
RF6	Opción de visualizar constantemente las instrucciones para el acople del dispositivo con la aplicación por medio de Bluetooth y/o Wi-Fi	RNF6	La aplicación debe ser fácil de descargar e instalar
RF7	Disponibilidad de un Sistema de información geográfico para la interpretación de los datos con mapa de colores según los porcentajes NPK en diferentes intervalos de tiempo	RNF7	La aplicación debe proporcionar tiempos de respuesta rápido
RF8	Opción de descarga de archivos en formatos .csv de la información en diferentes intervalos de tiempo	RNF8	Las interfaces de la aplicación deben ser amigables e intuitivas
RF9	Opción de sugerencias de los porcentajes NPK ideales según el tipo de cultivo	RNF9	La aplicación debe mantener los datos almacenados seguros y protegidos

Tabla 2.12: Tabla de requerimientos. Fuente: Propia

2.2.5. Selección de Software y Base de Datos

1. **Elección de base de datos**
   Actualmente el sector de las bases de datos evoluciona rápidamente debido a todo el flujo de información y crecimiento del internet. Ahora, los nuevos gestores de bases de datos permiten trabajar sobre diferentes opciones como: relacionales, no relacionales e híbridas, entre otros.
   Para la selección del tipo de base de datos a utilizar se identificaron las diferencias entre las bases de datos relacionales (SQL) y no relacionales (noSQL). Una base de datos relacional es una colección de elementos de datos que se organizan en un conjunto de tablas desde las cuales se puede acceder y hacer uso de esos datos sin tener que reorganizar dichas tablas. Del mismo modo, estas se basan en la organización de la información en partes pequeñas que son integradas mediante identificadores.

   Por otro lado, las bases de datos no relacionales están orientadas a modelos de datos específicos que poseen esquemas flexibles para la creación de aplicaciones modernas. Estas bases de datos no tienen un identificador que permita crear una relación entre datos de diferentes conjuntos, sin embargo cada estructura de datos compleja denominada "documento", se empareja con una clave única, estos documentos permiten ser anidados con otros documentos permitiendo crear una relación entre diferentes conjuntos de datos (colecciones) lo que la hace una base de datos altamente escalable y de mayor rendimiento para aplicaciones que requieran almacenar grandes volúmenes de datos estructurados, semi estructurados y no estructurados. [4]

   Debido a que el hardware relacionado con el proyecto recolecta datos de diferentes sensores, y está pensado para ser actualizado a diferentes versiones, donde se integren más instrumentos de medición, habrá un incremento en el volumen de datos; por lo cual se requerirá de una alta capacidad de procesamiento.

   Por estos requerimientos y teniendo en cuenta las definiciones expuestas previamente se escogió una base de datos NoSQL para el desarrollo del proyecto. Una vez escogida la base de datos se optó por obtener información de cuales

[4] *tomado de: https://www.oracle.co m/co/database/what-is-database/ https://www.mongodb.com/es/nosql-explained*

de las bases de datos NoSQL desarrolladas por diferentes compañías, ofrecía y permitía la adaptación de nuestro modelo de base de datos a un software. Después de una investigación, se encontraron las más conocidas actualmente: MongoDB, Apache Cassandra, CouchDB, Redis y Neo4j, siendo MongoDB la ideal debido a su alto nivel de escalabilidad y velocidad de procesamiento, al igual que por su gran cantidad de documentación disponible.[5]

El software para la base de datos permite implementar comandos SQL (GET, POST, PUT, DELETE) para hacer usos específicos de los datos almacenados en ella de diferentes clientes. Del mismo modo, crear diferentes conjuntos de datos (Colleciones en mongodb).

Así mismo es importante tener en cuenta que la base de datos va a ser controlada por un cliente, lo que genera la pregunta de cómo comunicar el cliente con la base de datos, la opción más adecuada fue la creación de un API Rest, ya que estas son las plataformas de servicios mas utilizadas para proporcionar contenido a las aplicaciones ejecutadas en dispositivos pequeños como celulares y tablets.[38] Su desarrolló se implementó en el software Node.js el cual es un entorno de ejecución para JavaScript orientado a eventos asíncronos, y diseñado para la creación de APIs Rest. El API Rest posibilita la creación de un localhost relacionado con la base de datos creada en MongoDB para el almacenamiento de los datos y el procesamiento de las solicitudes de los clientes conectados a esa plataforma de servicios; en esta plataforma se deben implementar instrucciones para la creación de modelos con los datos de los usuarios, tales como: Nombre, E-mail, Contraseña, Creación del dato. De igual forma para las variables N, P , K, Temperatura y Humedad el cual es un modelo diferente al de los usuarios. En el mismo sentido, se deben crear funciones para crear rutas HTTP entre el cliente y la base de datos, usando diferentes librerías y creando paralelamente validaciones que permitan la seguridad de esos datos haciendo uso de algoritmos para la encriptación de contraseñas y generadores de cadenas de largas longitudes, compuestas de caracteres aleatorios o también conocidos como Tokens (JSON WEB TOKENS), los cuales permiten realizar autenticaciones de las credenciales de un usuario especifico, y acceder a su respectiva información. [6] *(Ver Figura 2.21)*

[5] *tomado de: https://cassandra.apache.org/doc/latest/, https://docs.couchdb.org/en/stable, https://redis.io/documentation/*

[6] *Tomado de: https://nodejs.org/es/*

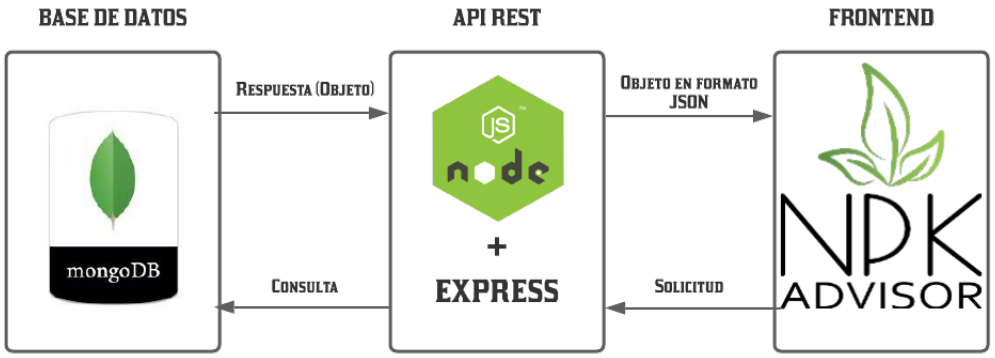


Figura 2.21: Plataforma de servicios Api Rest. Fuente: Propia

Teniendo una vez creada la base de datos y una plataforma de servicios que permitiera la comunicación entre el cliente y la base de datos, era pertinente implementar el cliente, el cual es una Aplicación Móvil con nombre NPK Advisor y que debía cumplir con los requerimientos mencionados en el capitulo 1, teniendo en cuenta lo anterior se decidió implementar la aplicación móvil para un sistema operativo Android.[7]

Android es un sistema operativo libre y abierto basado en Linux, es principalmente usado para el desarrollo de aplicaciones móviles, permitiéndole a los desarrolladores un servicio de calidad en desarrollo de software[39].

Para el diseño de la aplicación debían implementarse funciones y métodos que permitieran diferentes tareas tales como:

- **Creación de usuario:** Permite el registro de un usuario a la aplicación, obteniendo su nombre, su correo electrónico, su contraseña, verificación y a su vez, guardando esos datos en la base de datos.
- **Menú de usuario:** Menú desplegable que permite escoger entre las opciones (Mis cultivos, Registrar información de cultivo, Mapas y Gráficas).
- **Mis cultivos:** permite agregar un nuevo cultivo con nombre y área, actualizar un cultivo, eliminar un cultivo y visualizar los datos registrados dentro de un rango de fechas en una tabla.

[7] *Tomado de: https://www.android.com/what-is-android//*

- **Registrar información de cultivo:** Muestra las instrucciones de uso del dispositivo Hardware
- **Mapas:** permite la visualización de la ubicación en tiempo real del dispositivo a través de la herramienta de geolocalización Google Maps.
- **Graficas:** permite la visualización de los datos dentro de un rango de fechas en diagramas de barras, circulares y lineales.

Además de otras funcionalidades como la autenticación de credenciales de usuario, y la manipulación de los datos obtenidos en la plataforma de servicios.

En *Figura 2.22* se pueden observar los diseños realizados para para las actividades Login y Registro de Usuario.
En *Figura 2.23* se evidencian las diferentes opciones para la gestión de la información obtenida.
Del mismo modo, en *Figura 2.24* se muestra el diseño del menú principal y la actividad Agregar un nuevo cultivo.
En *Figura 2.25* se puede observar el diseño de la actividad Modificar la información de los cultivos.
En *Figura 2.26* se hacen referencia a los diseños de las actividades Eliminar y Ver la información de un cultivo respectivamente.
En *Figura 2.27* se encuentran los diseños para las actividades de visualización de los datos en una tabla y los mapas para la visualización de la ubicación del dispositivo.
En *Figura 2.28* se pueden visualizar las instrucciones para el uso del dispositivo Hardware.
Finalmente, en *Figura 2.29* y *Figura 2.30* se indican los diseños referentes a las actividades que imprimen las gráficas.

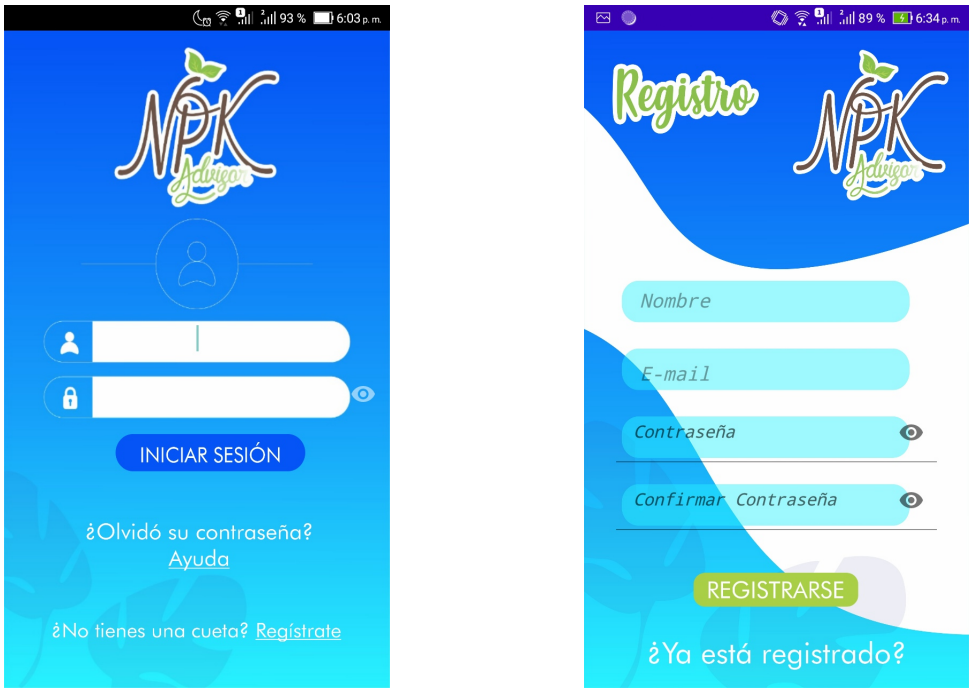


Figura 2.22: Actividades Login y registro de usuario. Fuente: Propia

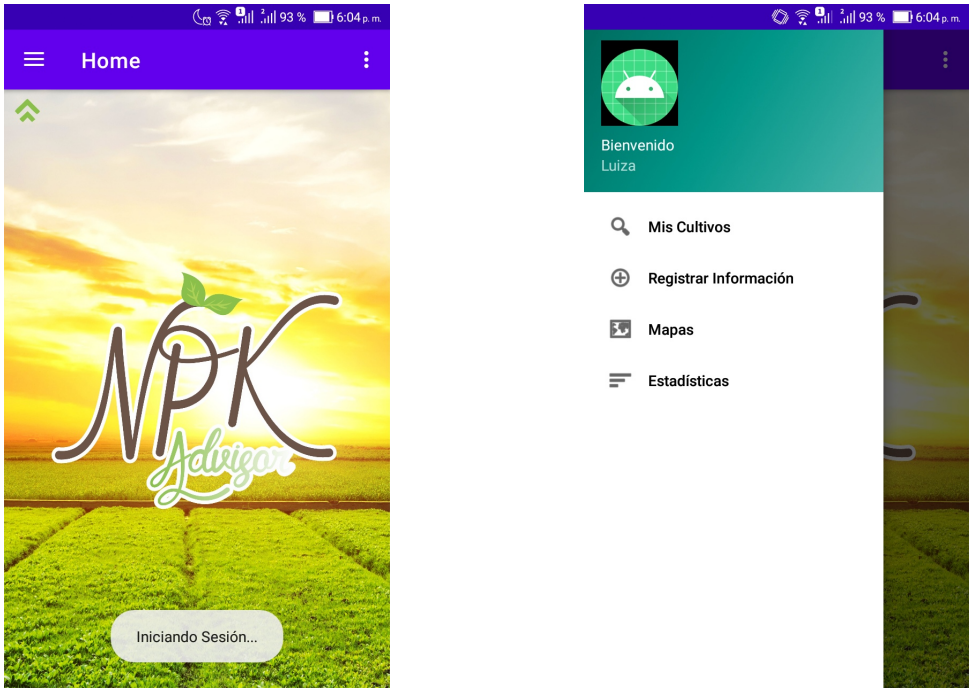


Figura 2.23: Actividad Home. Fuente: Propia

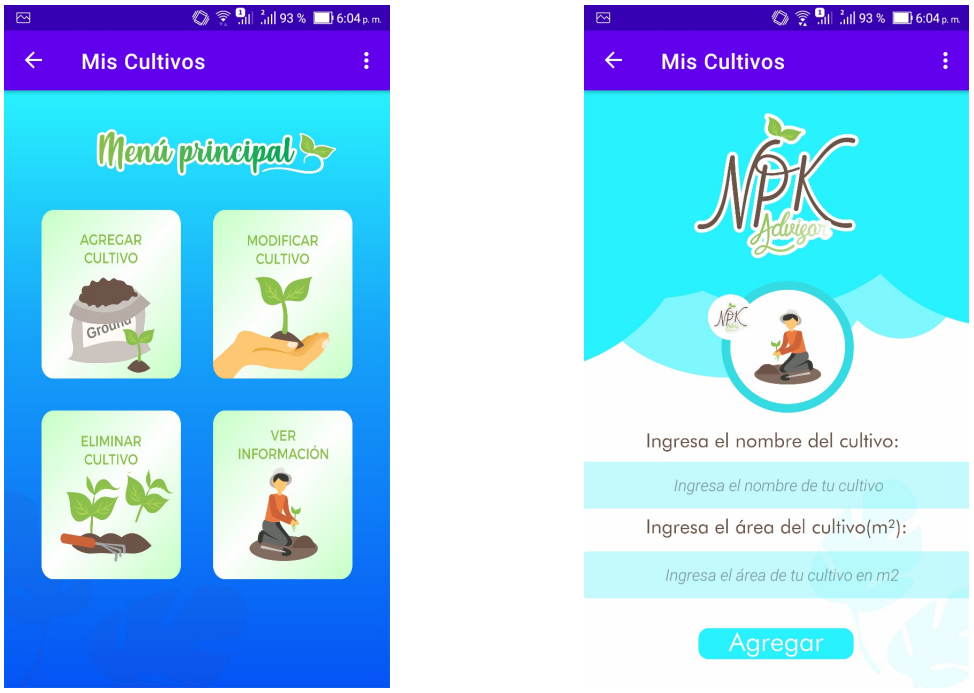


Figura 2.24: Actividad Menú y Agregar un nuevo cultivo. Fuente: Propia

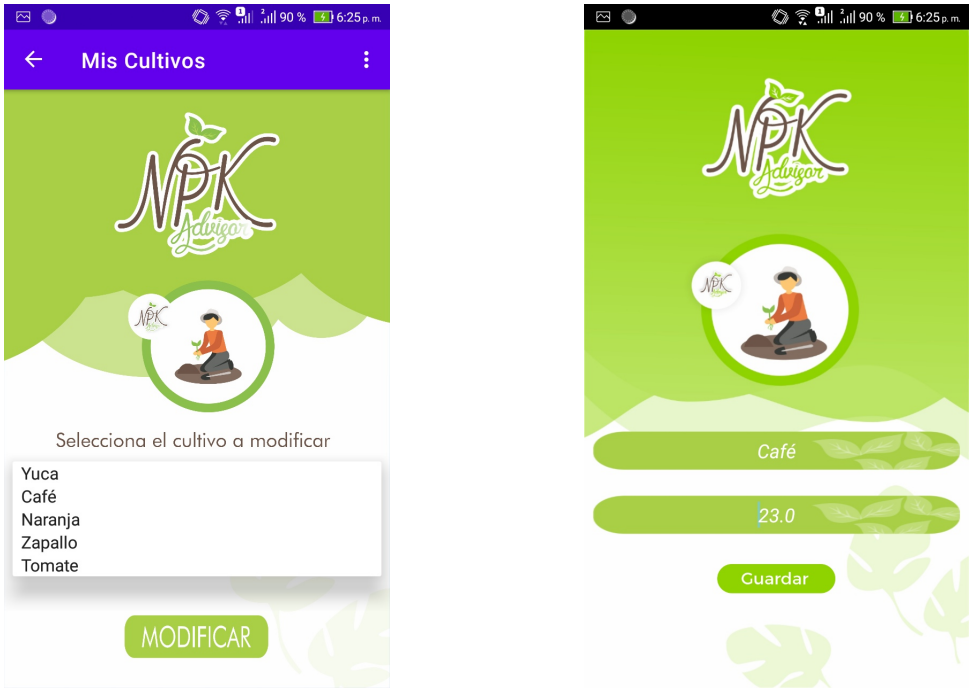


Figura 2.25: Actividad Modificar cultivo. Fuente: Propia

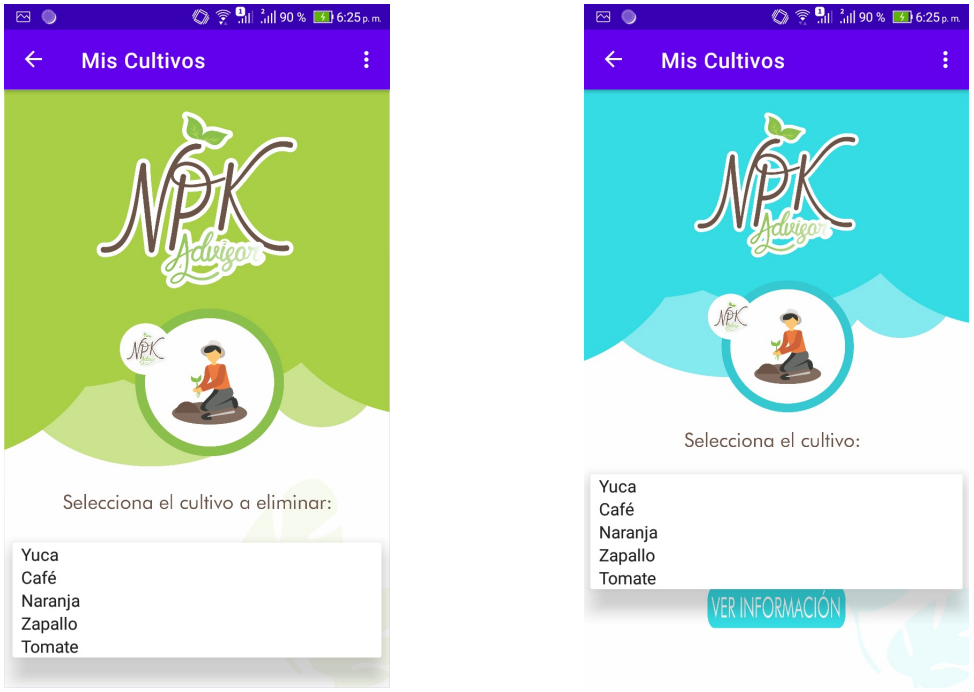


Figura 2.26: Actividad Eliminar y Ver información del cultivo. Fuente: Propia

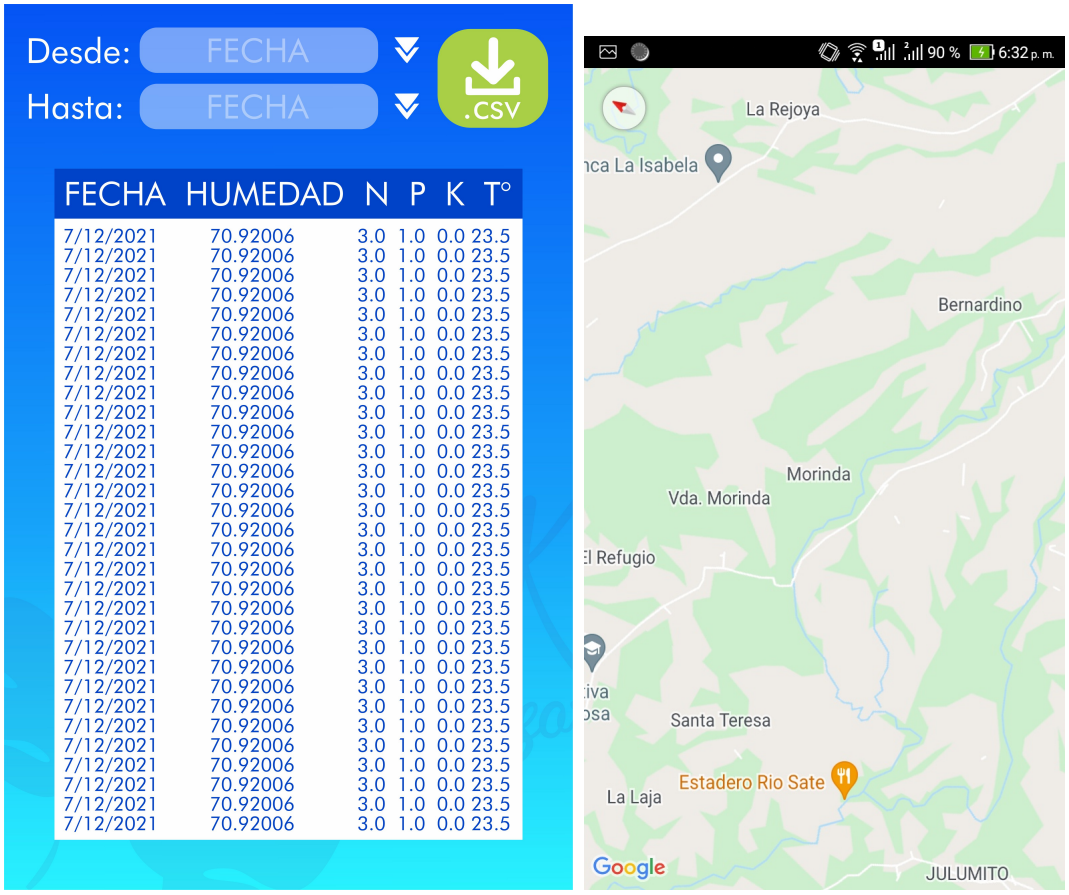


Figura 2.27: Actividad Información del cultivo y Mapas. Fuente: Propia

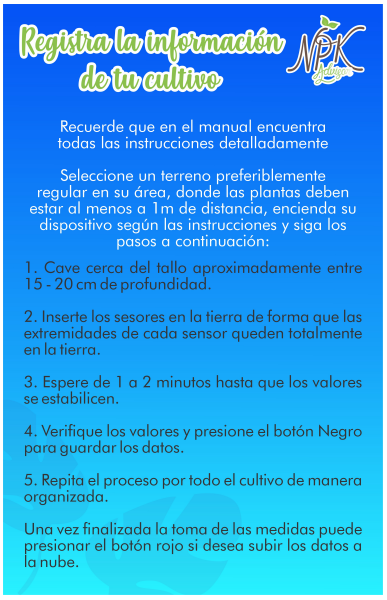


Figura 2.28: Actividad Registro de información del cultivo. Fuente: Propia

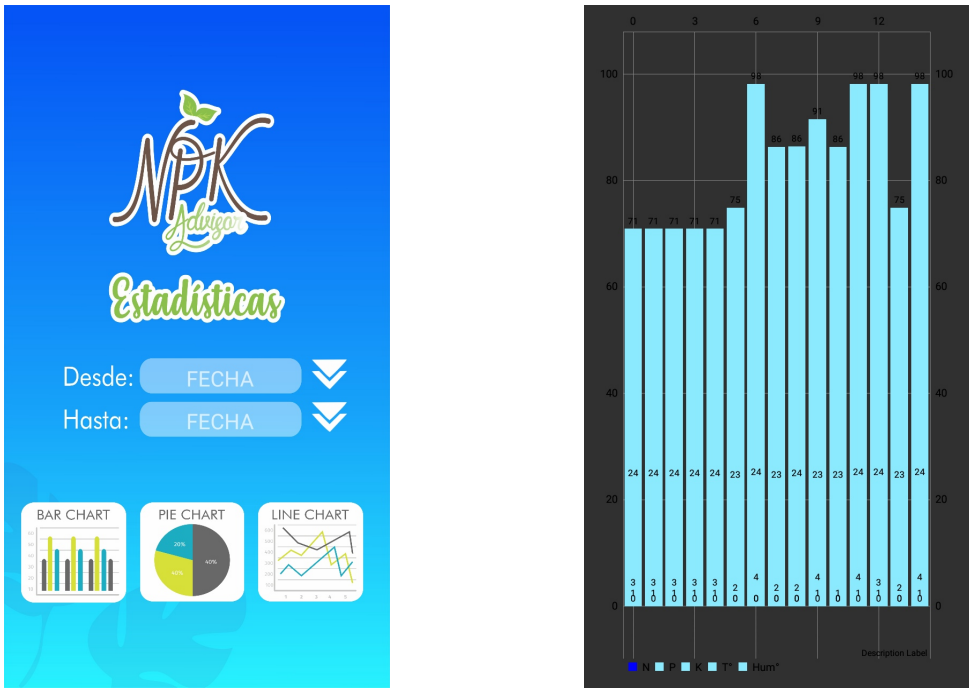


Figura 2.29: Actividad Gráficas. Fuente: Propia

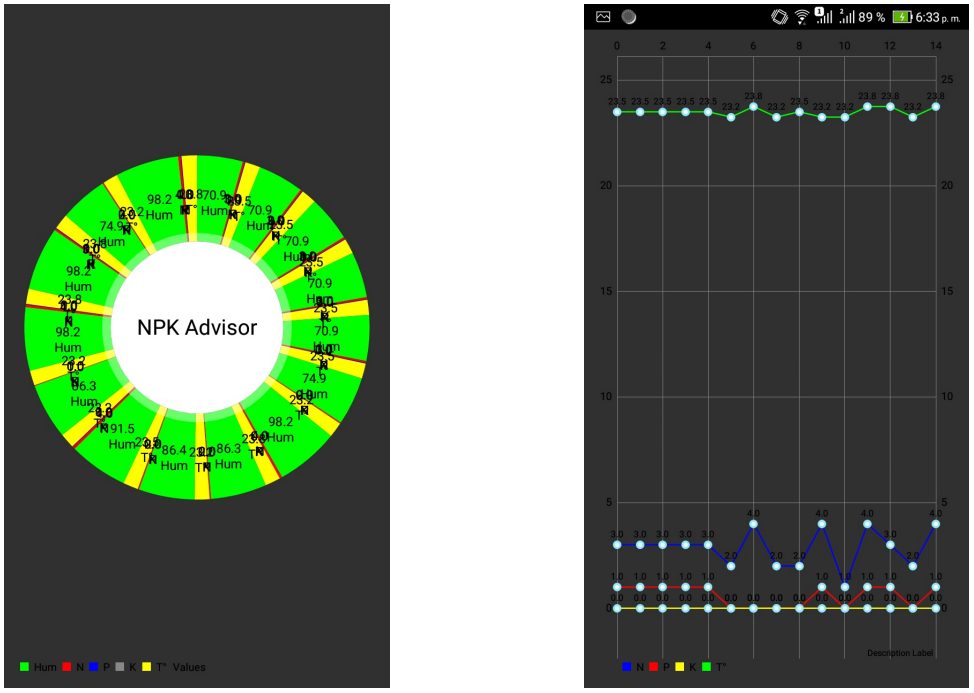


Figura 2.30: Actividad Gráficas. Fuente: Propia

Una vez implementado el programa, este consta de 27 clases *.java*, 23 archivos *.xml* y 1 interfaz; Esta interfaz hace uso de la librería *Retrofit* la cual le permite al usuario hacer peticiones de tipo HTTP con los metodos (*GET, POST, DELETE, PUT*) a la plataforma de servicios API Rest, accediendo por medio de una URL a la base de datos donde están almacenados los datos solicitados, estos datos son enviados y recibidos en un formato ligero de intercambio de datos JSON[8] para posteriormente hacer diferentes operaciones deseadas con ellos. Por otro lado, en los archivos .xml se diseñaron las diferentes interfaces de usuario haciendo uso de contenedores, botones, textos, campos de texto...etc. De tal manera que fueran intuitivas y agradables a la vista; La mayoría de los diseños de los objetos de las diferentes interfaces fueron figuras procesadas en el programa Corel Draw haciendo uso de bancos y repositorios de imágenes en diferentes formatos. Finalmente, las clases contienen todos los objetos y las instrucciones asociadas a estos, haciendo uso de las diferentes librerías y funciones, y de igual forma solicitando permiso para acceder a diferentes características del sistema operativo (Ubicación, almacenamiento, conexión a internet, etc.) Los códigos son evidenciados en la parte de Anexos de esta monografía.

2.2.6. Diagramas de modelado de la arquitectura Software

Para una fácil interpretación de la estructura que se va a utilizar en el desarrollo de la aplicación, se modeló el algoritmo gráficamente, a través de una serie de pasos estructurados y vinculados que permiten su revisión como un todo en un diagrama de flujo.[40]

A partir del análisis de requerimientos realizado anteriormente, se definieron unas secciones principales para la estructuración del funcionamiento del desarrollo de aplicación, estas son: inicio de sesión, administración de datos del cultivo y visualización de los datos del cultivo.

[8] *https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format*

Una descripción más detallada de cada sección, es mostrada a continuación.

- Inicio de Sesión

Para usuarios nuevos, la aplicación presentará en su interfaz de inicio la opción de un nuevo registro, también es posible acceder con cuentas alternas como Google.

Para usuarios ya registrados, podrán iniciar sesión validando sus credenciales, y en caso de no tenerlas, tendrá la opción de acceder a un proceso de recuperación.

- Administración de datos del cultivo

Esta sección hace referencia al manejo de los datos de los cultivos como registro, modificación de su información principal y la eliminación de todo lo asociado a este.

- Visualización de los datos del cultivo.

En esta sección se presentará toda la información registrada acerca de los porcentajes de los macronutrientes NPK de un cultivo seleccionado con la posibilidad de seleccionar un intervalo de tiempo, haciendo uso de las principales gráficas estadísticas para una interpretación más asequible e intuitiva del usuario final. Los diagramas elaborados se exponen en *Figura 2.31 y Figura 2.32*

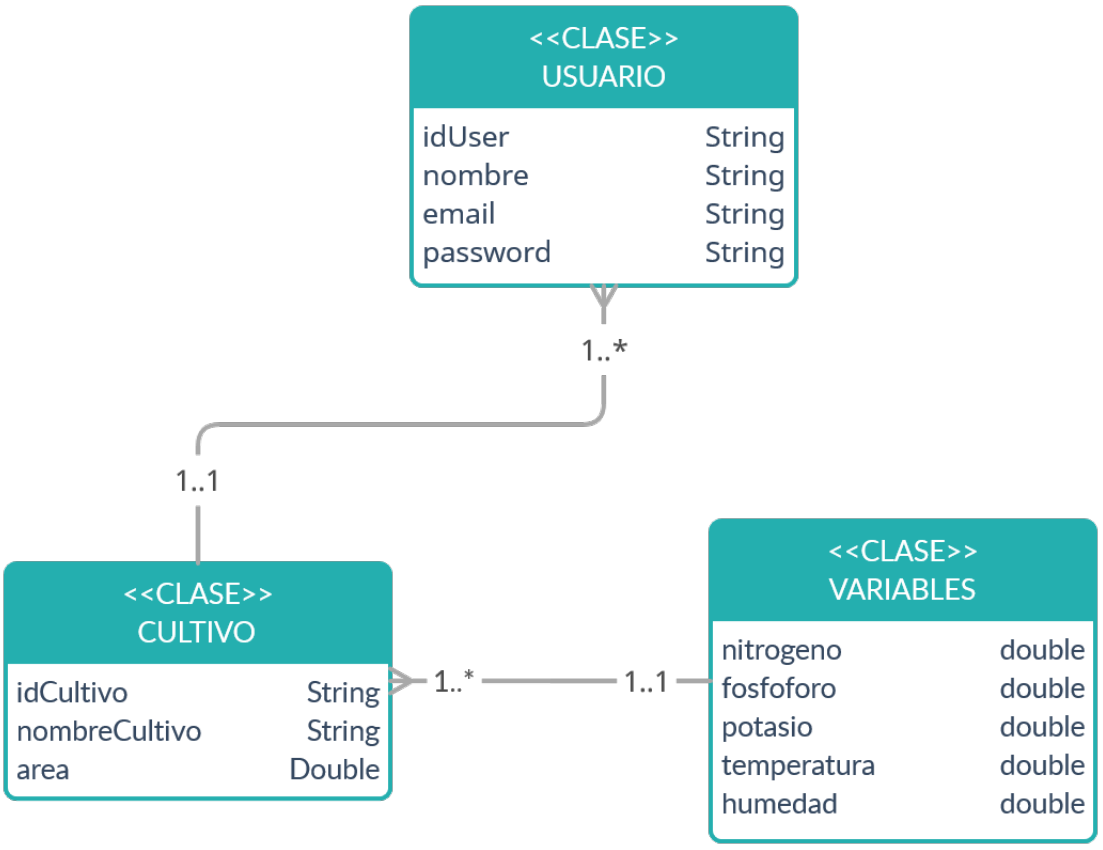


Figura 2.31: Diagrama UML. Fuente: Propia

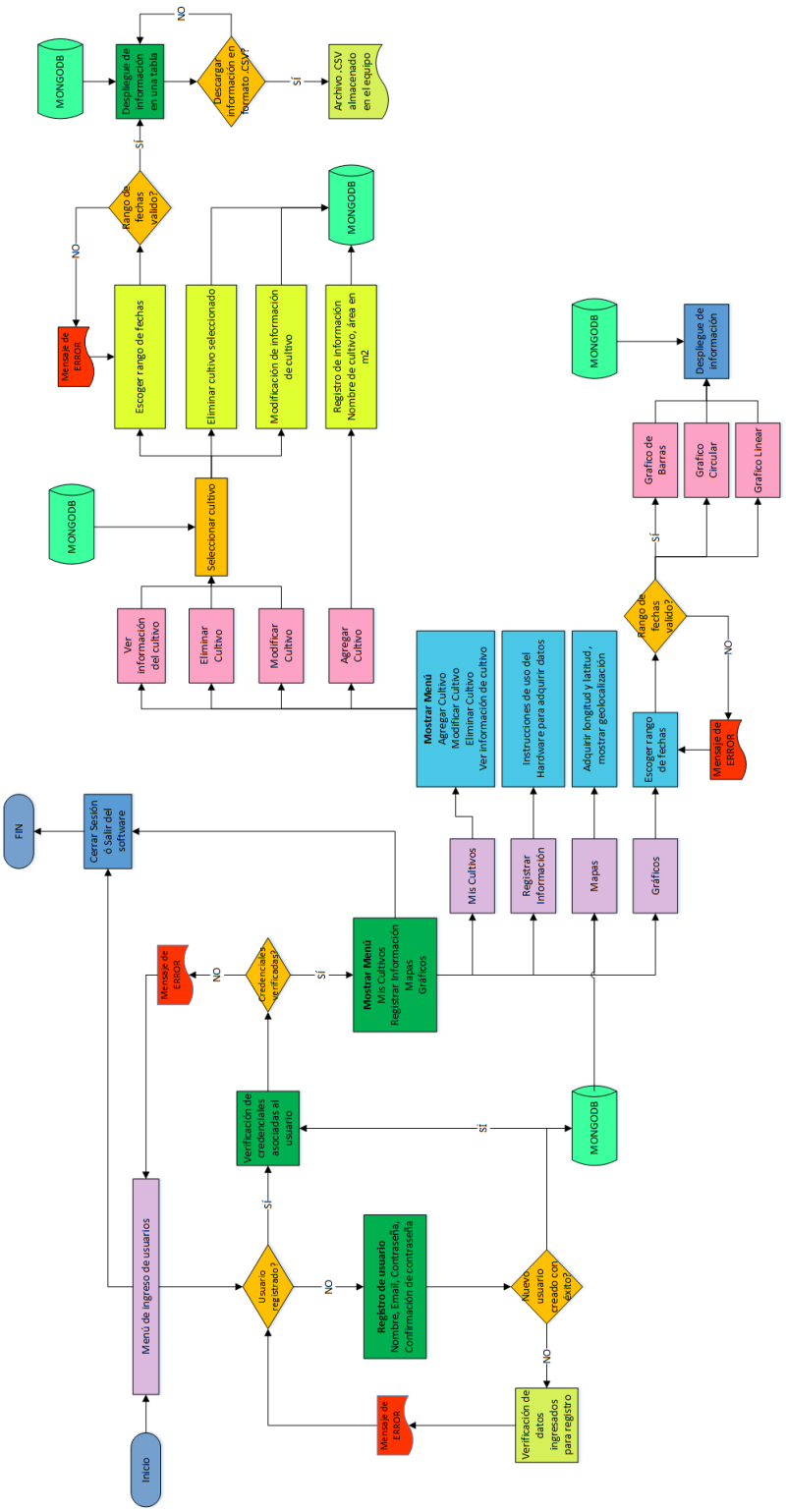


Figura 2.32: Diagrama de Flujo. Fuente: Propia

Capítulo 3

Experimentación y resultados

3.1. Toma y análisis de muestras

Hay suelos donde el desarrollo de los cultivos es significativamente mejor que en otros, según su fertilidad o la capacidad del suelo para aportar nutrientes. La fertilidad del suelo se determina por sus características intrínsecas y como cambia debido a disturbios externos, tanto físicos como químicos que ocurren de forma constante debido a las condiciones atmosféricas presentes. Los componentes presentes en el suelo al sufrir modificaciones pueden tanto aportar como intervenir en el saludable desarrollo del cultivo, por lo tanto el conocimiento de la cantidad de estos componentes por parte del agricultor es fundamental para mantener su productividad. Conocer estas variables permite aprovecharlo mejor[41].

El suelo agrícola se forma de diferentes partículas minerales, existen diferentes tipos de suelos clasificados por su textura, estructura, color y diversas características que le brindan un grado de fertilidad. Del mismo modo, en cuanto a su productividad la fertilidad cumple un papel fundamental, para que un suelo sea fértil debe contener los nutrientes necesarios, agua, aire y materia orgánica para el desarrollo de la planta del cultivo. Sin embargo, el porcentaje de estos componentes al tener variaciones hace a unos cultivos distintamente más productivos que otros. No obstante, tener mediciones de estos componentes permite precisar un buen desarrollo de la planta a través del tiempo, con base en los estudios de suelos presentes en la actualidad que definen si un suelo es fértil para diferentes tipos de plantas realizando pruebas de

suelos ya sea en laboratorios haciendo uso de químicos, o haciendo uso de un tipo de sensor o dispositivo electrónico[42].

De los mas de 90 elementos que una planta puede contener, solo 16 son indispensables para su crecimiento y reproducción, están divididos en tres grupos denominados: macro elementos, elementos secundarios y micro elementos, este proyecto fue enfocado a la medición de los macro elementos entre ellos está el nitrógeno, el fósforo y el potasio. Siendo el nitrógeno el elemento más importante en el crecimiento de las plantas, debido a que favorece a la descomposición de materia orgánica presente en la superficie siendo una constituyente de la proteína de la planta. Cuando el nitrógeno es adecuado el follaje es verde profundo, y la planta se ve fuerte y saludable[42].

Después del nitrógeno, el fósforo es el segundo nutriente más importante ya que este elemento promueve la formación y crecimiento temprano de las raíces, estimula la floración y acelera la madurez.

El tercero más importante es el potasio, desempeñando un papel importante en el movimiento del agua dentro de la planta, influyendo en el color y solidez de la fruta; cuando el potasio está en la cantidad correcta las plantas brinda como resultados frutos grandes y de buen color.

Si los suelos son incapaces de producir una cantidad favorable de estos elementos, el cultivo no podrá beneficiarse al máximo de una temperatura y humedad adecuada. por tal razón, conocer los porcentajes de estos elementos y así identificar sus necesidades, para corregir deficiencias es muy importante para obtener buenas resultados en sus respectivas cosechas[42].

Según las concentraciones de los macro elementos presentes en el suelo las características físicas de las plantas pueden variar, por ejemplo cuando falta nitrógeno las hojas de la planta se tornan de un color opaco, hay pocas flores y menos frutos, se puede ver como el proceso de crecimiento es retardado. Del mismo modo, cuando hace una falta fósforo las hojas que crecen son mas pequeñas y se tornan de un tono verde azulado. por el lado del potasio las hojas comienzan a presentar un rizado en el borde de las hojas, seguido de la muerte de las orillas de la hoja.

La importancia de conocer las propiedades del suelo, permite adicionalmente asegurar un uso eficaz de los fertilizantes ya que las dosis a implementar debe estar

fundamentada en un análisis de suelo previo. Al saber la cantidad exacta de fertilizante que se debe aplicar al cultivo, no solo para evitar gastos innecesarios si no también, para evitar el exceso de este, el cual puede provocar un desequilibro en el proceso de cultivo[42].

Para los análisis de resultados es importante conocer como se puede clasificar si el suelo de un cultivo es fértil para el desarrollo de ese cultivo. El análisis de los datos se hizo comparando tablas de rangos obtenidos a partir de tablas obtenidas por otras investigaciones experimentales los cuales definen la concentración de nutrientes NPK adecuados para un saludable desarrollo del cultivo[43][44][45][46][47][48]
Los valores adecuados de la variables medidas se encuentran en, *Tabla 3.1* para cultivos de Café, *Tabla 3.2* para cultivos de Yuca, *Tabla 3.3* para cultivos de Zapallo, *Tabla 3.4* para cultivos de Naranja y *Tabla 3.5* para cultivos de Tomate.

Análisis	Unidad	Bajo	Normal	Alto
N	%	<0.4	0.4-0.8	>0.8
P	ppm	<5	5-15	>15
K	meq/100g	<0.2	0.2-0.7	>0.7
Temperatura	°C	<18	18-23	>23
Humedad	%	<70	70-85	>85

Tabla 3.1: Valores adecuados para un cultivo fertil de café. Fuente:[43][45]

Análisis	Unidad	Bajo	Normal	Alto
N	%	<0.25	0.25-0.5	>0.5
P	ppm	<15	15-40	>40
K	meq/100g	<0.3	0.3-0.6	>0.6
Temperatura	°C	<20	20-30	>30
Humedad	%	<70	70-85	>85

Tabla 3.2: Valores adecuados para un cultivo fertil de Yuca. Fuente:[44]

Análisis	Unidad	Bajo	Normal	Alto
N	%	<0.1	0.1-0.2	>0.2
P	ppm	<20	20-40	>40
K	meq/100g	<0.38	0.38	>0.38
Temperatura	°C	<15	15-30	>30
Humedad	%	<70	70-80	>80

Tabla 3.3: Valores adecuados para un cultivo fertil de Zapallo. Fuente:[46]

Análisis	Unidad	Bajo	Normal	Alto
N	%	<2.4	2.4-2.6	>3
P	ppm	<0.12	0.13-0.16	0.2
K	meq/100g	<0.40	0.90	>1.15
Temperatura	°C	<15	15-30	>30
Humedad	%	<75	75-95	>95

Tabla 3.4: Valores adecuados para un cultivo fertil de Naranja. Fuente:[48]

Análisis	Unidad	Bajo	Normal	Alto
N	%	<4	4-6	>6
P	ppm	<20	20-40	>40
K	meq/100g	<0.3	0.3-0.5	>0.5
Temperatura	°C	<18	18-30	>30
Humedad	%	<60	60-80	>80

Tabla 3.5: Valores adecuados para un cultivo fertil de Tomate. Fuente:[47]

Una vez teniendo integrados el hardware y el software listos para su uso fue necesario hacer la respectiva prueba de campo; por tal motivo, se realizó una visita a una parcela agrícola ubicada a las afueras de la ciudad de Popayán y al centro de Agricultura del SENA, donde se permitió hacer pruebas en sus suelos para posteriormente sacar conclusiones sobre el funcionamiento del dispositivo electrónico y automático en pruebas reales, y así mismo poder brindar una evaluacion en cuanto a la fertilidad del suelo. Del mismo modo, se logró encontrar posibles errores que pudiera tener la implementación, y depurarlo, con el fin de brindar datos confiables y precisos.

Resultados de la salida de campo:

Se visitó la parcela, y el centro de agricultura del SENA,donde para realizar la prueba de campo inicialmente se definieron diferentes cultivos para tomar muestras en diferentes plantas de su suelo circundante con el fin de observar la variación de los valores N,P,K.temperatura y humedad al área del cultivo. Las pruebas se tomaron formando un área con una forma de matriz 4x3 definiendo cada punto en el plano cartesiano con su eje x,y. *(Ver Figura 3.1)*

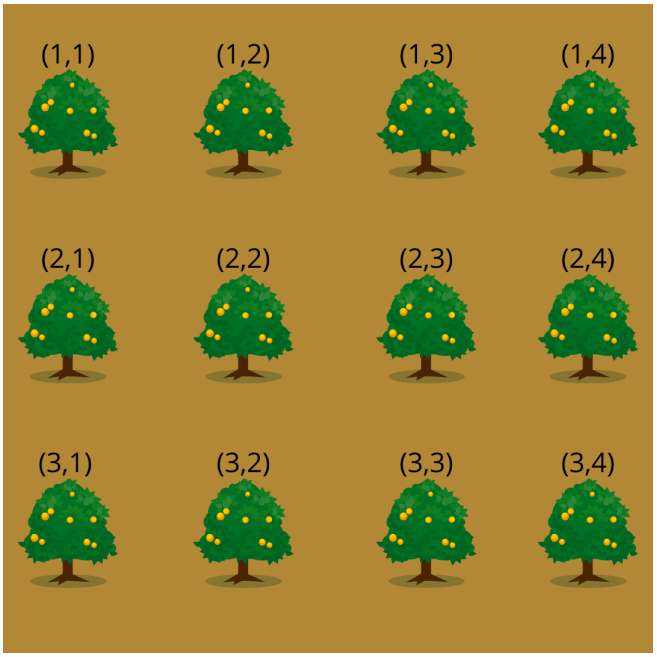


Figura 3.1: Matríz de distribución del cultivo. Fuente: Propia

Los cultivos elegidos fueron:

- Cultivo de Naranja
- Cultivo de Café
- Cultivo de Zapallo
- Cultivo de Yuca
- Cultivo de Tomate

Para cada uno de los cultivos se tomaron doce muestras. Los elementos necesarios para la toma de muestras fueron:

- Una cubeta de plástico
- Una Pala pequeña
- Vaso desechable
- Etiquetas adheribles

- Marcador

Inicialmente se definió el área de las muestras teniendo en cuenta plantas que tenían diferentes características debido a sus concentraciones en los elementos NPK, en tamaño, edad, color de su follaje, tamaño de su tronco, etc. De esta manera se realizó una comparación entre los valores obtenidos y sus características físicas explicadas anteriormente. Llegando a este punto, se introdujeron los sensores, obteniendo una muestra (N P K Temperatura y humedad de cada área), estos se etiquetaron con los nombres de cada cultivo usando etiquetas adheribles. Para el cultivo de tomate, los registros se hicieron directamente introduciendo el dispositivo en el suelo.

Los resultados obtenidos experimentalmente se muestran en *Tabla 3.6* para cultivos de Café, *Tabla 3.7* para cultivos de Yuca, *Tabla 3.9* para cultivos de Zapallo, *Tabla 3.8* para cultivos de Naranja y *Tabla 3.10* para cultivos de Tomate. haciendo referencia a la distribución del cultivo en X,Y y en los valores obtenidos:

X,Y	**N %**	**P(ppm)**	**K(meq/100g)**	**T°C**	**H %**
1,1	3	1	0	23,5	70,9200603
1,2	2	0	0	23,25	74,8717949
1,3	4	1	0	23,75	98,1598793
1,4	2	0	0	23,25	86,3348416
2,1	2	0	0	23,5	86,3650075
2,2	4	1	0	23,25	91,4932127
2,3	1	0	0	23,5	86,3348416
2,4	4	1	0	23,75	98,1598793
3,1	3	1	0	23,75	98,1598793
3,2	2	0	0	23,25	74,8717949
3,3	2	0	0	23,25	74,8717949
3,4	4	1	0	23,75	98,1598793

Tabla 3.6: Muestras cultivo de café. Fuente: Propia

X,Y	N %	P(ppm)	K(meq/100g)	T°C	H %
1,1	6	1	0	23,25	72,51885
1,2	1	0	0	23,25	68,98944
1,3	3	1	1	23,25	64,67572
1,4	2	0	0	23,25	54,44947
2,1	2	0	0	23,25	61,35747
2,2	0	0	0	23,76	96,16893
2,3	2	0	0	23,6	69,26094
2,4	1	0	0	23,5	90,6184
3,1	3	1	0	27	71,82504
3,2	3	1	1	23,5	58,7632
3,3	4	1	0	23,25	61,35747
3,4	2	1	0	23,75	69,26094

Tabla 3.7: Muestras cultivo de Yuca. Fuente: Propia

X,Y	N %	P(ppm)	K(meq/100g)	T°C	H %
1,1	3	1	0	23,5	97,1644
1,2	1	0	0	23,75	97,91855
1,3	1	0	0	23,25	99,33635
1,4	2	0	0	22,75	93,42383
2,1	4	1	0	23,75	71,73454
2,2	2	0	0	23,5	84,49472
2,3	3	1	0	23	81,71946
2,4	3	1	0	23	91,04072
3,1	4	1	0	23,5	84,94721
3,2	6	1	0	23	97,94872
3,3	4	1	0	23,75	91,04072
3,4	2	1	0	23	84,49472

Tabla 3.8: Muestras cultivo de Naranja. Fuente: Propia

X,Y	N %	P(ppm)	K(meq/100g)	T°C	H %
1,1	15	7	1	23,5	67,33032
1,2	12	5	1	24,5	71,43288
1,3	8	3	0	23,75	70,22624
1,4	3	1	0	23,75	72,45852
2,1	9	3	0	23,25	73,21267
2,2	11	5	1	23,75	71,01056
2,3	55	30	5	23,5	72,85068
2,4	8	3	0	23,75	72,88084
3,1	9	5	1	23,25	77,22474
3,2	15	7	1	23,5	72,88084
3,3	8	5	5	23,5	72,85068
3,4	11	3	1	23,75	73,21267

Tabla 3.9: Muestras cultivo de Zapallo. Fuente: Propia

X,Y	N %	P(ppm)	K(meq/100g)	T°C	H %
1,1	5	18	0	27,25	67,33032
1,2	5	16	0	23,5	71,43288
1,3	4	20	0	24	70,22624
1,4	5	22	0	23,25	72,45852
2,1	4	17	0	23,75	73,21267
2,2	5	17	1	23,5	71,01056
2,3	3	17	1	23,25	72,85068
2,4	1	22	1	23,75	72,88084
3,1	4	23	0	23,75	77,22474
3,2	6	23	0	23,5	72,88084
3,3	4	25	0	24	72,85068
3,4	4	25	0	23,75	73,21267

Tabla 3.10: Muestras cultivo de Tomate. Fuente: Propia

En *Figura 3.2 y Figura 3.3* se encuentran las imágenes relacionadas con la salida de campo.

Figura 3.2: Cultivos muestreados. Fuente: Propia

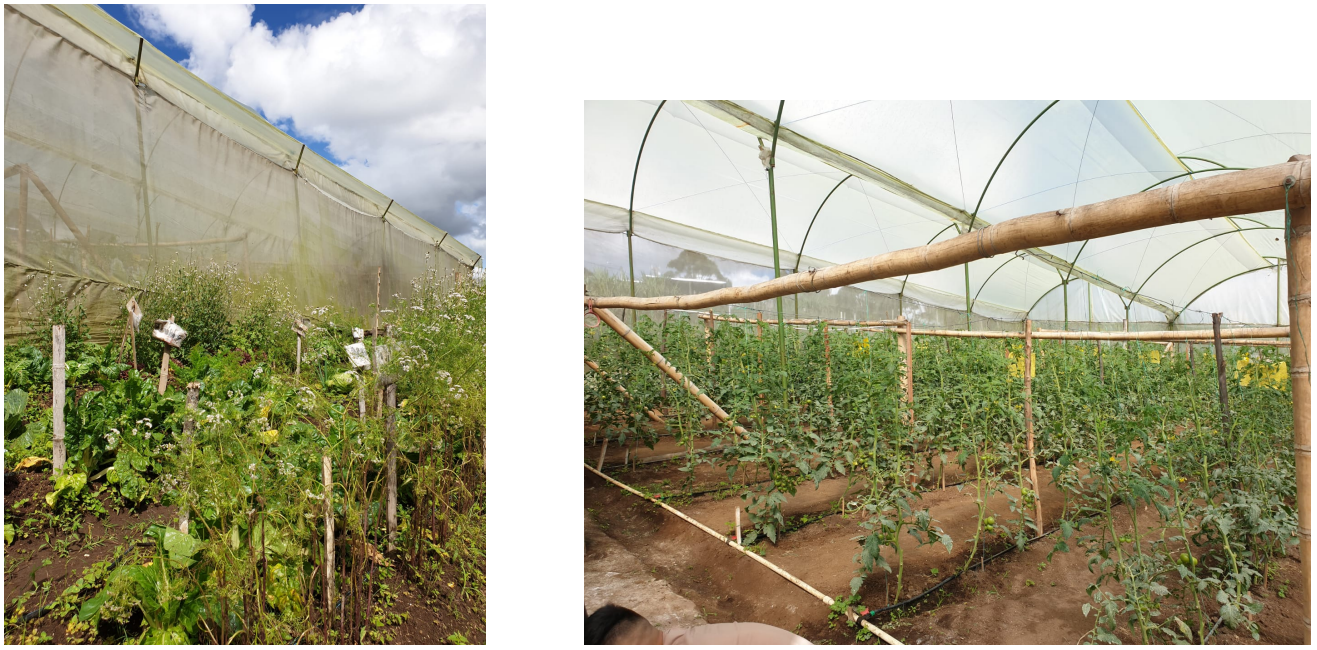


Figura 3.3: Cultivos muestreados. Fuente: Propia

en *Figura 3.4* y *Figura 3.5* se evidencia la toma de muestras de suelo

Figura 3.4: Toma de muestras. Fuente: Propia

Figura 3.5: Toma de muestras. Fuente: Propia

y finalmente la puesta en marcha del dispositivo en un entorno real *(Ver Figura 3.6 y Figura 3.7)*

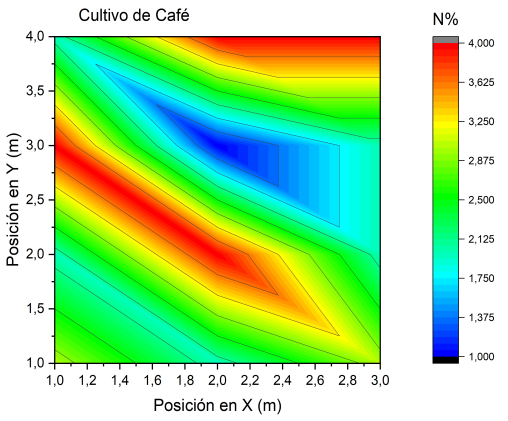
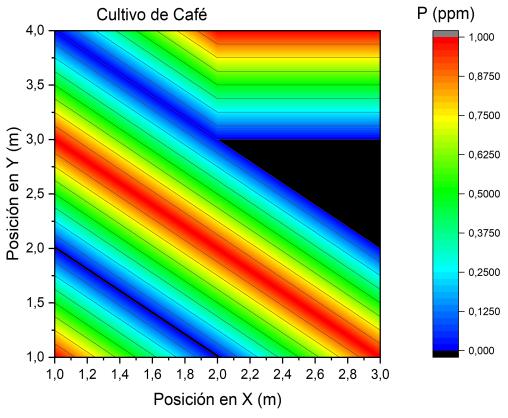
Figura 3.6: Toma de muestras. Fuente: Propia

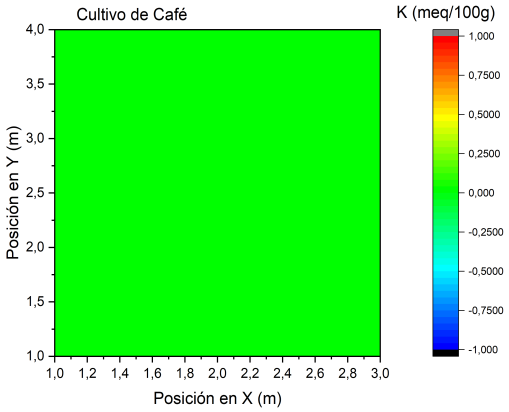
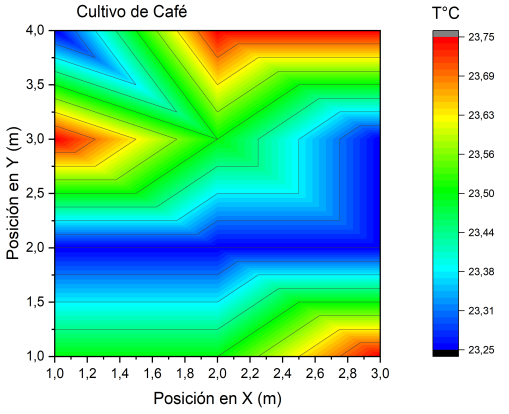
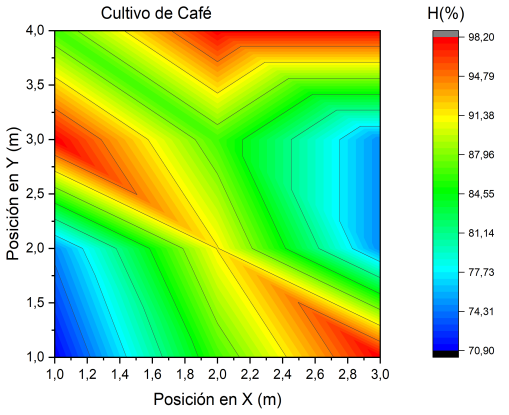
Figura 3.7: Dispositivo. Fuente: Propia

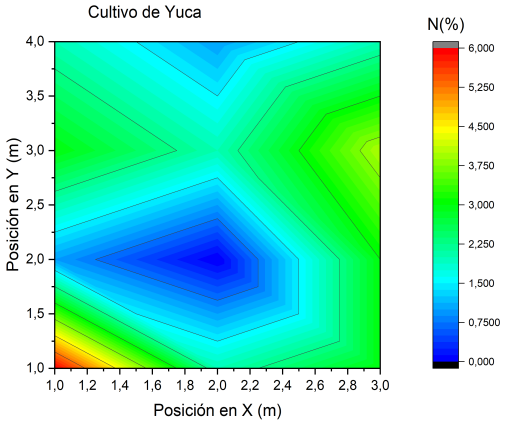
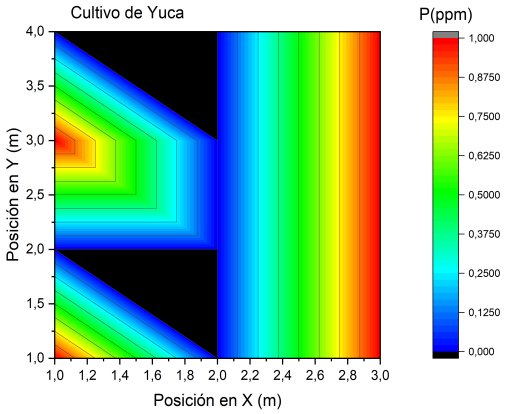
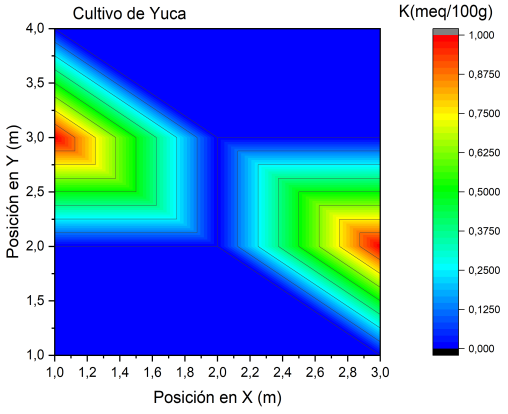
3.2. Resultados

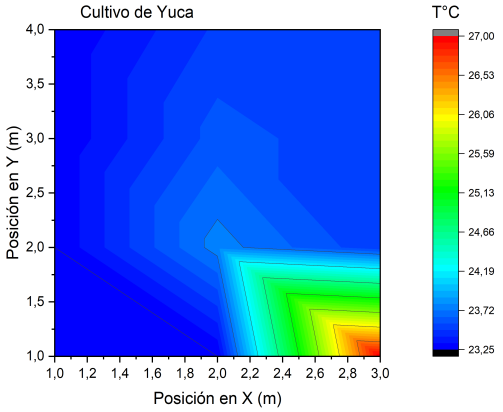
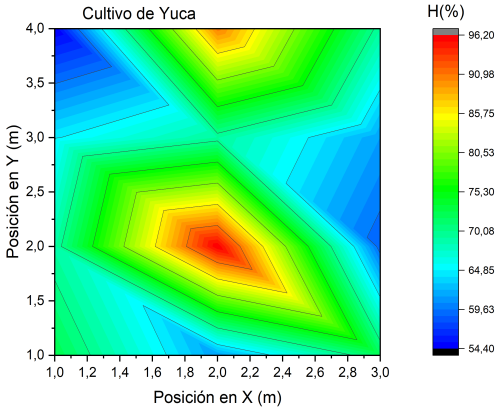
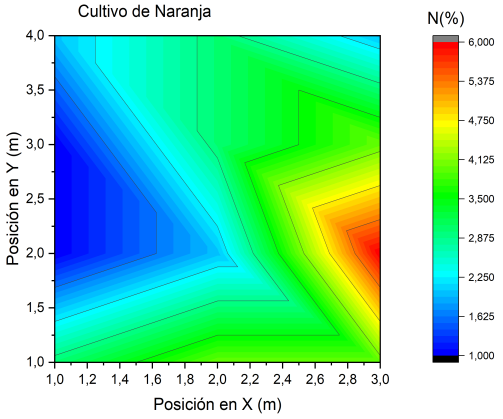
3.2.1. Verificación de las variables NPK

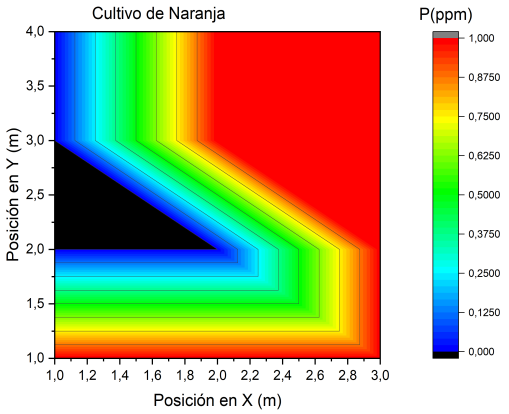
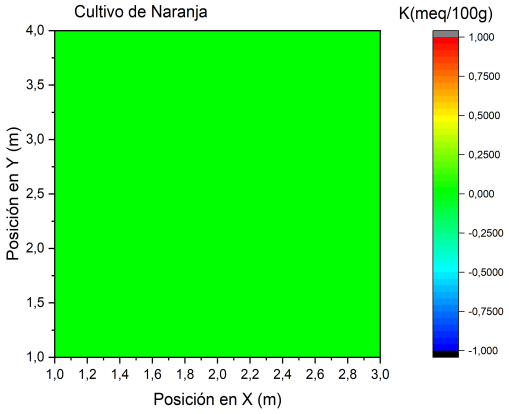
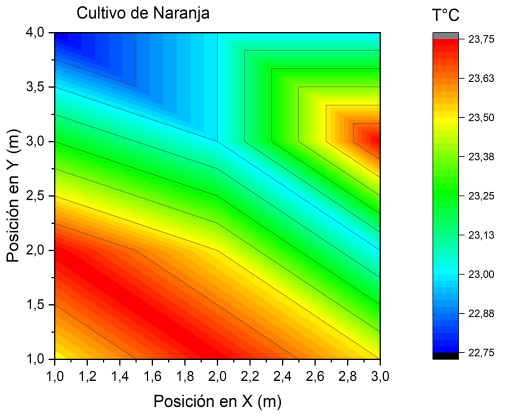
Para la verificación de las variables NPK, se hizo una comparación entre las concentraciones sugeridas y las obtenidas por medio del dispositivo.El uso de gráficos de contorno para el análisis del trabajo permitió observar de una manera más didáctica la cantidad de concentración de las diferentes variables. Las gráficas fueron elaboradas mediante el Software Origin Pro 2021 y son presentadas con sus respectivos análisis en *(Tabla 3.11)*.

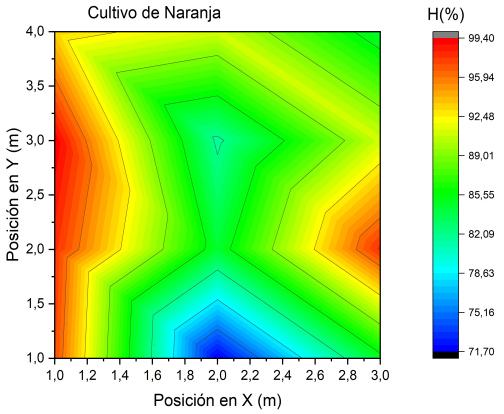
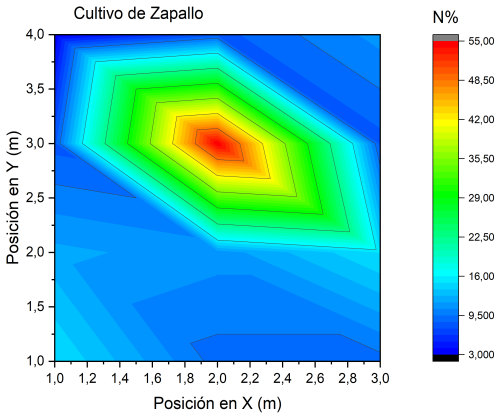
Cultivo	Índice	Resultado	Análisis
Café	N		Todos los valores obtenidos en el cultivo superan el límite máximo indicado para la concentración de Nitrógeno en el café, esto explica porqué las plantas presentan una apariencia débil y con escasez de frutos.
	P		Ninguno de los valores alcanza el límite mínimo indicado para la concentración de fósforo, esto explica porqué la planta en general y sus hojas presentan un tamaño evidentemente menor a la de un cultivo normal, además de un color verde azulado en sus hojas.

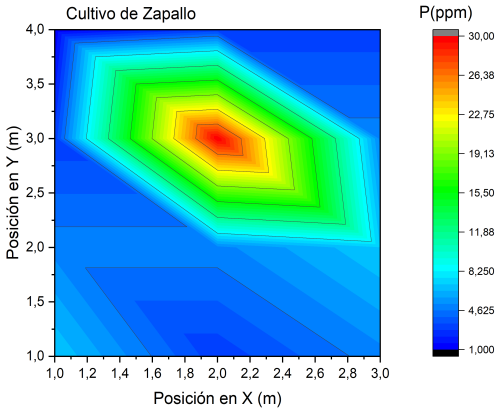
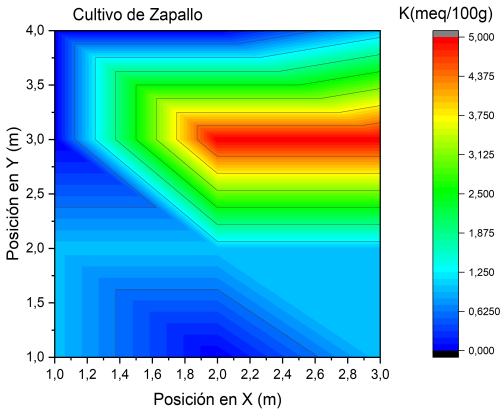
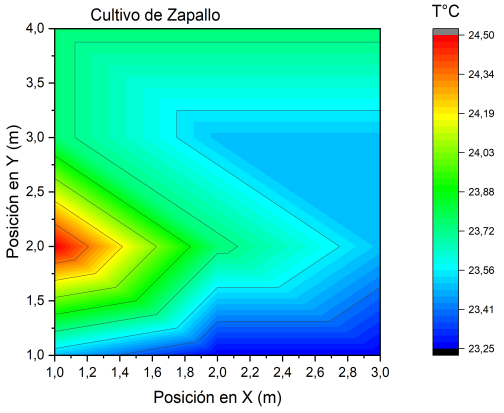
	K		Todos los valores obtenidos para la concentración de potasio en el cultivo corresponden a 0 mg/kg, puesto que el sensor maneja una resolución de 1 mg/kg no es posible saber el valor exacto. Sin embargo, se intuye que el valor oscila en el rango adecuado, dado que las características de los bordes de las hojas son los apropiados: lisos y vitales.
	T°		Todos los valores obtenidos se encuentran dentro del rango indicado para la temperatura en el café, sin embargo existen lugares en el cultivo cercanos al límite máximo, que pueden afectar los procesos vitales de la planta.
	Hum		La mayoría de los valores obtenidos superan el limite máximo indicado para la humedad en el café, esto sugiere un alto riesgo en cuanto a la propagación de plagas, sin embargo, el cultivo se encuentra sano.

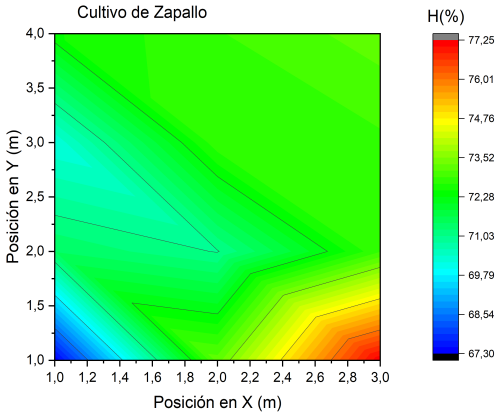
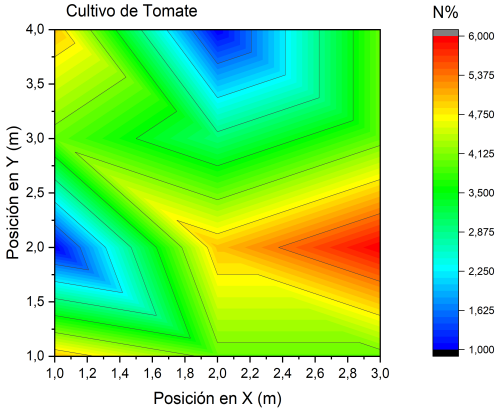
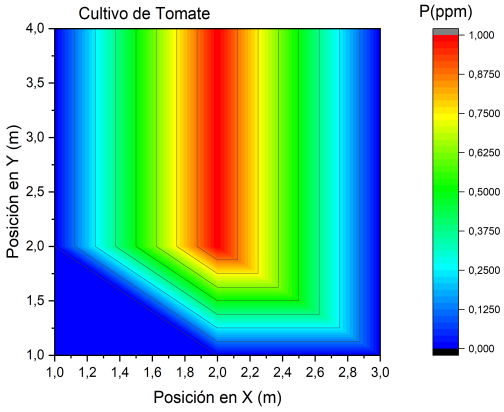
Yuca	N		Todos los valores obtenidos en el cultivo superan el límite máximo indicado para la concentración de Nitrógeno en la yuca, esto explica porqué las plantas presentan una apariencia parcialmente débil.
	P		Ninguno de los valores alcanza el límite mínimo indicado para la concentración de fósforo, esto explica porqué la planta presenta hojas pálidas y tamaño en los tallos inferior al normal.
	K		Algunos valores obtenidos para la concentración de potasio en el cultivo superan el límite máximo indicado para la yuca, por lo tanto se prevé que la planta rendirá frutos con madurez prematura y sus hojas tendrán una apariencia amarillenta.

	T°		Todos los valores obtenidos se encuentran dentro del rango indicado para la temperatura en la yuca, esto beneficia todos los procesos vitales del cultivo.
	Hum		Los valores obtenidos oscilan tanto fuera como dentro del rango indicado para la humedad en la yuca, esto indica un terreno inestable para que las plantas puedan absorber sus nutrientes, provocando el riesgo de marchitarse.
Naranja	N		Todos los valores obtenidos en el cultivo superan el límite máximo indicado para la concentración de Nitrógeno en la naranja, esto explica porqué las plantas presentan una apariencia débil y con escasez de frutos.

	P		Algunos valores obtenidos para la concentración de fósforo en el cultivo superan el límite máximo indicado para la naranja, esto se evidencia en la clorosis presentada en la mayoría del cultivo.
	K		Todos los valores obtenidos para la concentración de potasio en el cultivo corresponden a 0 mg/kg, puesto que el sensor maneja una resolución de 1 mg/kg no es posible saber el valor exacto. Sin embargo, se intuye que la planta se encuentra escasa de este macronutriente, dado que presenta hojas rizadas y amarillentas.
	T°		Todos los valores obtenidos se encuentran dentro del rango indicado para la temperatura en la naranja, esto beneficia todos los procesos vitales del cultivo.

	Hum		La mayoría de los valores obtenidos se encuentran dentro del rango indicado para la humedad de la naranja, sin embargo, algunas superan el límite máximo, esto sugiere un alto riesgo en cuanto a la propagación de plagas.
Zapallo	N		Todos los valores obtenidos en el cultivo superan considerablemente el límite máximo indicado para la concentración de Nitrógeno en el Zapallo, consecuencia de un alta presencia de maleza, ambiente ideal para un alto volumen bacterias que se descomponen rápidamente; aumentando los niveles de nitrógeno, esto explica porqué las plantas presentan una apariencia débil y con escasez de frutos.

	P		Los valores obtenidos para la concentración de fósforo en el cultivo se encuentran por debajo del rango indicado para el Zapallo, a excepción de una planta, esto explica porqué la planta en general y sus hojas presentan un tamaño evidentemente menor a la de un cultivo normal, además de unas hojas pálidas.
	K		Algunos valores obtenidos para la concentración de potasio en el cultivo superan considerablemente el límite máximo indicado para el zapallo, por lo tanto la planta presenta hojas amarillentas y se prevé que rendirá frutos con madurez prematura.
	T°		Todos los valores obtenidos se encuentran dentro del rango indicado para la temperatura del zapallo, esto beneficia todos los procesos vitales del cultivo.

	Hum		Todos los valores obtenidos se encuentran dentro del rango indicado para la humedad del zapallo, beneficiando a la planta en su proceso de fotosíntesis.
Tomate	N		Todos los valores obtenidos en el cultivo se encuentran dentro del rango indicado para la concentración de nitrógeno en el tomate, favoreciendo el crecimiento general de las plantas que presenta un follaje verde profundo.
	P		Todos los valores obtenidos en el cultivo se encuentran dentro del rango indicado para la concentración de fósforo en el tomate, favoreciendo la salud general del cultivo, en especial la formación de raíces y semillas.

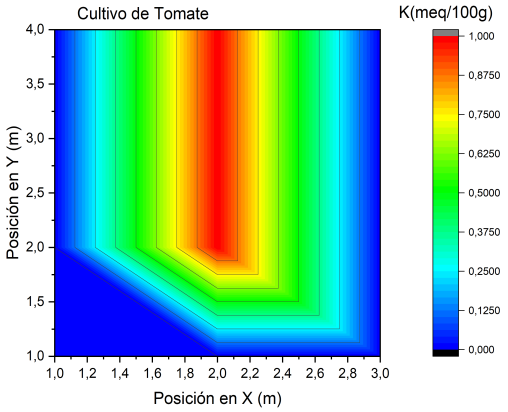
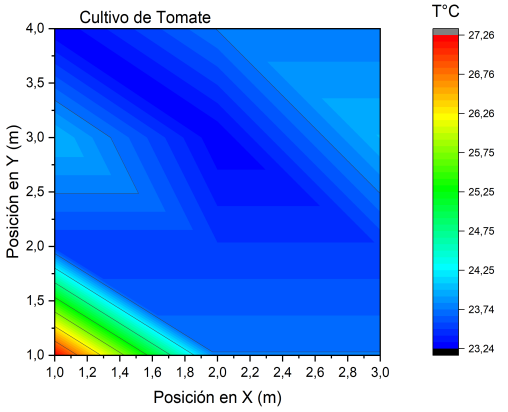
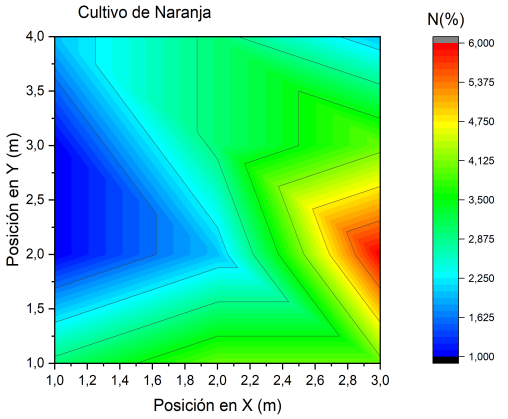
	K		Todos los valores obtenidos en el cultivo se encuentran dentro del rango indicado para la concentración de potasio en el tomate, favoreciendo las propiedades intrínsecas y estéticas del fruto; además de conservar la buena salud de la planta.
	T°		Todos los valores obtenidos se encuentran dentro del rango indicado para la temperatura del tomate, esto beneficia todos los procesos vitales del cultivo.
	Hum		Todos los valores obtenidos se encuentran dentro del rango indicado para la humedad del tomate, beneficiando a la planta en su proceso de fotosíntesis

Tabla 3.11: Análisis de resultados. Fuente: Propia

3.2.2. Análisis de Fertilidad del cultivo

A partir del análisis de los resultados obtenidos en el apartado anterior, se pueden definir las condiciones del suelo para el desarrollo de futuros cultivos en el mismo y brindar un apoyo en la asesoría acerca de las debilidades que presenta.

Los suelos correspondientes a los cultivos de café, yuca, naranja y zapallo son parcialmente fértiles, puesto que presentan una variabilidad en algunas concentraciones de sus macro-nutrientes, que oscilan fuera de los rangos para ser considerado como un cultivo fértil, y aunque bajo estas condiciones las plantas pueden dar frutos, no se debe considerar como un ambiente adecuado para producciones de medio y alto rendimiento que requieran conservar la buena salud de las plantas y obtener frutos con altos estándares de calidad para su comercialización en el mercado.

Por otro lado, el cultivo de tomate presenta las características que hacen de su suelo un suelo fértil donde, en las condiciones ideales, se prevé que las plantas tendrán un desarrollo saludable, y las cosechas se darán en el tiempo y de la forma esperada.

A continuación se especificará el análisis realizado para cada uno de los cultivos:

- **Café:** puede germinar la semilla, se pueden dar los frutos, planta deficiente, apariencia débil, escasez de frutos.
- **Yuca:** plantas débiles, hojas pálidas, tallos débiles, crecimiento inadecuado, hojas amarillentas, frutos con madurez prematura, riesgo de marchitarse, buena temperatura.
- **Naranja:** apariencia débil, escasez de frutos, clorosis, hojas rizadas amarillentas, buena temperatura, excesiva humedad, riesgo de plagas.
- **Zapallo:** apariencia débil, escasez de frutos, hojas pequeñas y pálidas, hojas amarillas, frutos con madurez prematura, buena temperatura, buena humedad.
- **Tomate:** follaje verde profundo y saludable, buena formación de raíces y semillas, buenos frutos, buena temperatura, buena humedad.

Capítulo 4

Discusión y conclusiones

Este trabajo tuvo como objetivo general, determinar mediante un sistema electrónico, el porcentaje de las variables Nitrógeno, Potasio y Fósforo NPK en sitio para verificar las propiedades intrínsecas de un terreno. Inicialmente, este trabajo encontró como problemática relevante para los agricultores, la dificultad en la identificación de los macro nutrientes nitrógeno, fósforo y potasio (NPK) que son los más importantes para el desarrollo de los cultivos, así como de los factores de temperatura,y humedad, reflejando la necesidad de contar con un método que les permitiera verificar los porcentajes de estas variables, que fuera asequible, económico y rentable, ya que la mayoría de métodos que existen actualmente para el análisis de estas variables, son costosos y toman tiempo para generar los resultados de las pruebas, encontrándose valores inexactos para el proceso del cultivo, al momento de obtenerlos. Por lo tanto, para dar respuesta al objetivo general se estructuraron tres objetivos específicos y se desarrollaron los requerimientos para el sistema e ingeniería del producto, el diseño electrónico, la calibración del sensor NPK, la definición de los requerimientos del software, la selección del software y la base de datos, y los diagramas de modelado de la arquitectura software.

De esta forma, se buscó responder a la pregunta, ¿cómo monitorear las variables NPK (Nitrógeno, Fósforo y Potasio) en sitio, para interpretar las características de un terreno, respecto a métodos convencionales como pruebas de suelos en laboratorios especializados para la verificación de los porcentajes de NPK?, y, a través del desarrollo del sistema y la comparación respecto a pruebas de un análisis de suelo

real, se pudo concluir que, en un cultivo, es posible monitorear las variables en sitio con el Sistema Electrónico y Automático para determinar los porcentajes de las variables de Nitrógeno, Fósforo y Potasio en suelos, desarrollado en este trabajo; sin embargo, este sistema puede ser utilizado para suelos que no requieran un análisis exacto, debido a la resolución del sensor NPK.

Por consiguiente, con el dispositivo se puede verificar la ausencia o exceso a partir de los datos obtenidos por el sistema y la observación de las características físicas de la planta y el análisis de la fertilidad en el suelo, las cuales permiten realizar la verificación correcta de las variables NPK con el fin de hacer proyecciones adecuadas a futuros cultivos de alto rendimiento y así, facilitar a los agricultores el uso de tecnologías que entregan información en tiempos oportunos, inferiores a los de las pruebas convencionales de laboratorio para garantizar la producción y el rendimiento del cultivo.

Por otro lado, a través de los resultados obtenidos por el dispositivo se genera un informe de resultados en la aplicación que es de fácil acceso para el agricultor, el cual le permite analizar los datos y presentarlos fácilmente porque es un dispositivo versátil y de uso sencillo. En este punto, es importante recalcar la importancia que tiene conocer las propiedades del suelo, porque además de asegurar un uso eficaz de los fertilizantes en relación con las dosis a implementar a partir de una análisis de suelo previo, al saber la cantidad exacta de fertilizante que se debe aplicar al cultivo, no solo se logra evitar gastos innecesarios sino también se evita su uso en exceso, el cual puede provocar un desequilibro en el proceso de cultivo[33].

Finalmente, entre las limitaciones se encuentra que, el dispositivo desarrollado dura 2 horas en campo, por lo tanto, es importante tener en cuenta la necesidad de expandir la batería. Además, es importante precisar que este dispositivo permite analizar la fertilidad del suelo solamente en relación con las variables NPK pero se requiere evaluar otras variables que inciden en el cultivo, como el calcio, zinc, entre otras, con el fin de obtener el estado de fertilidad total del suelo.

4.1. Trabajos futuros

En esta sección se muestran los trabajos que pueden ser ejecutados con base en este proyecto

- Adicionar sensores que midan diferentes nutrientes del suelo
- Implementar redes de sensores en un área de cultivo obteniendo un estudio de todos los puntos en tiempo real
- Realizar un módulo basado en sistemas de información geográfico

Bibliografía

[1] G. Archbold Taylor, H. Beltran Torres, F. Ruiz, M. Narducci Marin, D. Mendez Chaves, L. Trujillo Arboleda, C. Parra, H. Carrillo, and A. M. Mouazen, "PH Measurement IoT System for Precision Agriculture Applications," *IEEE Lat. Am. Trans.*, vol. 17, no. 5, pp. 823–832, 2019.

[2] L. Ma, T. Duan, and J. Hu, "Application of a universal soil extractant for determining the available NPK: A case study of crop planting zones in central China," *Sci. Total Environ.*, vol. 704, p. 135253, 2020. [Online]. Available: https://doi.org/10.1016/j.scitotenv.2019.135253

[3] G. Lavanya, C. Rani, and P. Ganeshkumar, "An automated low cost IoT based Fertilizer Intimation System for smart agriculture," *Sustain. Comput. Informatics Syst.*, no. 2018, pp. 1–12, 2019. [Online]. Available: https://doi.org/10.1016/j.suscom.2019.01.002

[4] K. Liu, L. E. Sollenberger, M. L. Silveira, J. M. Vendramini, and Y. C. Newman, "Distribution of nutrients among soil-plant pools in 'tifton 85' bermudagrass pastures grazed at different intensities," *Crop Sci.*, vol. 51, no. 4, pp. 1800–1807, 2011.

[5] R. Sumiharto and R. Hardiyanto, "NPK Soil nutrient measurement prototype based on local binary pattern and back-propagation," *Proc. - 2018 IEEE Int. Conf. Internet Things Intell. Syst. IOTAIS 2018*, pp. 23–28, 2019.

[6] H. M. Khairnar and S. S. Kulkarni, "Automated Soil Macro-Nutrient Analyzer Using Embedded Systems," *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–3, 2018.

[7] A. Y. Kachor and K. Ghodinde, "Design of microcontroller based agribot for fertigation and plantation," *2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019*, no. Iciccs, pp. 1215–1219, 2019.

[8] M. A. Coutinho, F. d. O. Alari, M. M. Ferreira, and L. R. Amaral, "Influence of soil sample preparation on the quantification of NPK content via spectroscopy," *Geoderma*, vol. 338, no. December 2018, pp. 401–409, 2019. [Online]. Available: https://doi.org/10.1016/j.geoderma.2018.12.021

[9] M. Masrie, M. S. A. Rosman, R. Sam, and Z. Janin, "Detection of nitrogen, phosphorus, and potassium (NPK) nutrients of soil using optical transducer," *2017 IEEE Int. Conf. Smart Instrumentation, Meas. Appl. ICSIMA 2017*, vol. 2017-Novem, no. November, pp. 1–4, 2018.

[10] L. Ma, S. Feng, P. Reidsma, F. Qu, and N. Heerink, "Identifying entry points to improve fertilizer use efficiency in Taihu Basin, China," *Land use policy*, vol. 37, pp. 52–59, 2014.

[11] A. M. A. Fonacier, R. Claire Manaol, R. C. C. Parillon, M. M. Villena, and G. P. Tan, "Design of a Polychromatic Color Sensor - Based Voltage Comparator Circuit of Soil pH and Nutrient Management Device for Fertilizer Recommendation," *2019 IEEE 11th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2019*, 2019.

[12] T. Raj, T. A. Johny, S. Khetawat, B. Rajeshwari, and S. Prasad, "Ambient Parametric Monitoring of Farms Using Embedded IoT LoRa," *2019 IEEE Bombay Sect. Signat. Conf. IBSSC 2019*, vol. 2019Januar, pp. 2–7, 2019.

[13] A. Liopa-Tsakalidi, D. Tsolis, P. Barouchas, A.-E. Chantzi, A. Koulopoulos, and N. Malamos, "Application of Mobile Technologies through an Integrated Management System for Agricultural Production," *Procedia Technol.*, vol. 8, no. Haicta, pp. 165–170, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.protcy.2013.11.023

[14] M. Y. Kulkarni, K. K. Warhade, and S. Bahekar, "Primary Nutrients Determination in the Soil Using UV Spectroscopy," *Int. J. Emerg. Eng. Res. Technol.*, vol. 2, no. 2, pp. 198–204, 2014.

[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[16] X. Y. Huang, J. S. Ma, and Q. Tang, "Geographic information system conspectus," *High. Educ. Press. Beijing*, 2001.

[17] G. A. Tang and M. D. Zhao, "Geographic information system," *Sci. Publ. house*, 2000.

[18] M. F. Goodchild, "Geographic information systems," *Prog. Hum. Geogr.*, vol. 12, no. 4, pp. 560–566, 1988.

[19] Organización de las Naciones Unidas para la Alimentación y la Agricultura, "Cómo alimentar al mundo en el 2050," *Voluntas*, vol. 3, no. 1, pp. 120–123, 1992.

[20] M. Mundial, "SUMMARY OF 1993 WHO/ISH GUIDELINES FOR THE MANAGEMENT OF MILD HYPERTENSION: MEMORANDUM FROM A WHO/ISH MEETING : Guidelines Sub-committee of WHO/ISH Mild Hypertension Liaison Committee," *Clin. Exp. Pharmacol. Physiol.*, vol. 20, no. 12, pp. 801–808, 1993.

[21] A. A. B. Ruíz, "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ADQUISICIÓN DE DATOS CON SENSORES: 808H5V5, MCP9700A, WATERMARK, MPX4115A, SQ-110, COMUNICACIÓN MEDIANTE PROTOCOLO ZIGBEE Y MySQL, PARA UN CULTIVO DE TOMATE EN SUTAMARCHÁN, BOYACÁ," vol. 3, no. 2, pp. 54–67, 2015. [Online]. Available: http://repositorio.unan.edu.ni/2986/1/5624.pdf

[22] M. Masrie, A. Z. M. Rosli, R. Sam, Z. Janin, and M. K. Nordin, "Integrated optical sensor for NPK Nutrient of Soil detection," *2018 IEEE 5th Int. Conf. Smart Instrumentation, Meas. Appl. ICSIMA 2018*, no. November, pp. 1–4, 2019.

[23] Ulrich Karl and Eppinger Steven, *Diseño y desarrollo de productos*, 2013.

[24] S. Nitrogen and P. Three-in one, "Instruction Manual Soil Nitrogen , Phosphorus and Potassium Three-in-One," *Soil Nitrogen,Phosphorus Potassium Three-in-One Transm. Type Rs485*, vol. VER 2.0, p. 15, 2021.

[25] [Online]. Available: https://naylampmechatronics.com/blog/rs485conarduino.html

[26] U. Manual, "Arduino Nano V2.3 User Manual," *Arduino*, pp. 1–5, 2008. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno

[27] G. Description, O. Information, P. Configuration, and T. A. Circuit, "to-Digital Converter (0 ° C to + 1024 ° C) to-Digital Converter (0 ° C to + 1024 ° C) SYMBOL CONDITIONS," pp. 1–8.

[28] E. Systems, "ESP32-WROOM-32 Datasheet," pp. 1–3, 2013.

[29] [Online]. Available: https://www.instructables.com/How-to-Use-the-Soil-Hygrometer-Module-Arduino-Tuto/

[30] U. Guide, "Handson Technology I2C Serial Interface 1602 LCD Module," pp. 1–8.

[31] A. Espinal, "Retos de conectividad a internet en instituciones educativas rurales de Colombia," *Univ. EAFIT*, pp. 1–7, 2018.

[32] M.-s. C. Adapter and M. Sd, "Micro SD Card Module for Arduino," pp. 3–4.

[33] O. N. Semiconductor, "Regulator 150 kHz Fixed Frequency Internal Oscillator," 2008.

[34] [Online]. Available: https://demosspro.com/cb/inicio/61-bateria-lipo-turnigy-111v-3s-1000mah-3s-20c.html

[35]

[36] J. Pressman, "Análisis de requisitos del software," p. 9, 2002. [Online]. Available: https://tesuva.edu.co/phocadownloadpap/Anlisis de requisitos del software.pdf

[37] S. S. D. Reservas, "Especificación de Requerimientos," pp. 1–21, 2011.

[38] A. Belkhir, M. Abdellatif, R. Tighilt, N. Moha, Y. G. Gueheneuc, and E. Beaudry, "An observational study on the state of REST API uses in android mobile applications," *Proc. - 2019 IEEE/ACM 6th Int. Conf. Mob. Softw. Eng. Syst. MOBILESoft 2019*, pp. 66–75, 2019.

[39] S. Guo-Hong, "Application development research based on android platform," *Proc. - 7th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2014*, vol. 1, pp. 579–582, 2015.

[40] H. Gonzales, "diagrama de pareto ejemplo – Calidad & Gestion – Consultoría para Empresas," 2012. [Online]. Available: https://calidadgestion.wordpress.com/tag/diagrama-de-pareto-ejemplo/

[41] M. stationery office, *Fertiliser Recommendations*, 1986.

[42] L. Lesur, A. Martinez, and P. Celis, *Manual de fertilización y productividad del suelo agrícola*, 2006.

[43] I. Agronómica and G. A. Torres, "Ciencia Unisalle Implementación de una hectárea de yuca (Manihot esculenta crantz) con fines comerciales , en el municipio de Nunchía , Casanare," 2019.

[44] Fao, "La yuca," *Fao-Ciat*, p. 18, 2002.

[45] O. L. Ocampo López, K. Castañeda Peláez, and J. J. Vélez Upegui, "Caracterización de los ecotopos cafeteros colombianos," *Perspect. Geográfica*, vol. 22, no. 1, pp. 89–108, 2017. [Online]. Available: http://revistas.uptc.edu.co/index.php/perspectiva/article/view/6100

[46] N. S. Ávila Pinilla, "EL CULTIVO DE AUYAMA (Cucurbita moschata) HÍBRIDO BÁRBARA UN MODELO DEMOSTRATIVO Y PRODUCTIVO A CORTO PLAZO EN LA VEREDA LA UNIÓN DEL MUNICIPIO DE PUERTO LLERAS META." p. 50, 2017. [Online]. Available: https://ciencia.lasalle.edu.co/ingenieria_agronomica

[47] M. Allende C, L. Salinas P, N. Olivares P, J. Riquelme S, A. Antúnez B, J. P. martinez C, P. Corradini S, P. Abarca R, A. Guzmán L, and S. Felmer E, "Manual de cultivo de tomate en invernadero," p. 112, 2017.

[48] O. Figueroa, "Fertilización de cítricos. ," *Univ. Nac. Agrar. La Molina*, pp. 1–30, 2011.

Anexos

A. Documentación Software A continuación se describen mediante este anexo las funcionas mas importantes para el desarrollo del Sistema:

- **RegInfo():** Permite el registro de nuevos usuarios

```
1  public class RegInfo extends AppCompatActivity {
2      Button pruebaGet;
3      TextView Humedad;
4      TextView N;
5      TextView P;
6      TextView K;
7      TextView Temp;
8      TextView horas;
9      TextView fecha;
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_reg_info);
15         pruebaGet = findViewById(R.id.pruebaget);
16         Humedad = findViewById(R.id.pruebagetT);
17         N = findViewById(R.id.n);
18         P = findViewById(R.id.p);
19         K = findViewById(R.id.k);
20         Temp = findViewById(R.id.temp);
21         horas = findViewById(R.id.time);
22         fecha = findViewById(R.id.date);
23         horas = findViewById(R.id.time);
24         pruebaGet.setOnClickListener(new View.OnClickListener() {
```

```
25              @Override
26              public void onClick(View v) {
27                  Index();
28              }
29          });
30      }
31
32      public void Index() {
33
34          Calendar rightNow = Calendar.getInstance(); //FORMATO DATE
35          int hour = rightNow.get(Calendar.HOUR_OF_DAY);
36          int min = rightNow.get(Calendar.MINUTE);
37
38          //https://mkyong.com/java/java-date-and-calendar-examples/
39          SimpleDateFormat sdf = new SimpleDateFormat("dd/M/yyyy");
40          String date = sdf.format(new Date());
41
42
43
44              Call<IndexResponse> indexResponseCall = ApiClient.
                    getUserService().findIndex1();
45              indexResponseCall.enqueue(new Callback<IndexResponse>() {
46                  @Override
47                  public void onResponse(Call<IndexResponse> call,
                        Response<IndexResponse> response) {
48                      if (response.isSuccessful()) {
49                          ArrayList<IndexResponse2> IndexReponses =
                                response.body().getInfoIndex();
50                          for (int i = 0; i < IndexReponses.size(); i++)
                                {
51                              Humedad.setText((IndexReponses.get(i).
                                    getHumedad()).toString());
52                              N.setText(IndexReponses.get(i).getN().
                                    toString());
53                              P.setText((IndexReponses.get(i).getP()).
                                    toString());
54                              K.setText((IndexReponses.get(i).getK()).
                                    toString());
55                              Temp.setText((IndexReponses.get(i).getTemp
                                    ()).toString());
56                              horas.setText(hour +":"+ min);
57                              fecha.setText(date);
```

```
58                          }
59
60                      } else {
61                          Toast.makeText(com.example.npkadvisorfinal.
                                RegInfo.this, "Verifique su conexi n a
                                Internet", Toast.LENGTH_LONG).show();
62                      }
63                  }
64
65                  @Override
66                  public void onFailure(Call<IndexResponse> call,
                        Throwable t) {
67                      Toast.makeText(com.example.npkadvisorfinal.RegInfo.
                            this, "Request Failed", Toast.LENGTH_LONG).show
                            ();
68
69                  }
70              });
71          }
72      }
```

- **MainActivity():** Permite la autenticación de los usuarios y la navegación hacía el menú principal

```
1  public class MainActivity extends AppCompatActivity {
2      EditText username1;
3      EditText password1;
4      ImageView start_user;
5      ImageView register;
6      private static String token;
7      private String email;
8
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14         username1 = findViewById(R.id.username1);
15         password1 = findViewById(R.id.password1);
```

```
16          start_user = findViewById(R.id.btn_inicio);
17          register = findViewById(R.id.register);
18
19
20          start_user.setOnClickListener(new View.OnClickListener() {
21              @Override
22              public void onClick(View view) {
23                  if (validate()) {
24                      SignIn();
25                  }
26              }
27          });
28
29          register.setOnClickListener(new View.OnClickListener() {
30              @Override
31              public void onClick(View v) {
32                  OpenRegister();
33              }
34          });
35      }
36
37
38      public void Open() {  //Lleva a la actividad de Men  despu s de
            autenticar las credenciales.
39          startActivity(new Intent(MainActivity.this, MainMenu2.class));
40      }
41
42      public void OpenRegister() {  //Lleva a la actividad de Men
            despu s de autenticar las credenciales.
43          startActivity(new Intent(MainActivity.this, SignUpActivity.
                class));
44      }
45
46      @NonNull
47      private Boolean validate() { //VALIDAR QUE LOS CAMPOS DEL LOGIN NO
            ESTEN VACIOS
48          Boolean result = false;
49          String name = username1.getText().toString();
50          String password = password1.getText().toString();
51          if (name.isEmpty() && password.isEmpty()) {
52              Toast.makeText(this, "Por favor ingrese sus credenciales",
                    Toast.LENGTH_SHORT).show();
```

```
53          } else {
54              result = true;
55          }
56          return result;
57      }
58
59      public void SignIn() {
60          if (validate()) {
61              email = username1.getText().toString();
62              Login login = new Login(username1.getText().toString(),
                    password1.getText().toString());
63              Call<LoginModel> userResponseCall = ApiClient.
                    getUserService().login(login);
64              userResponseCall.enqueue(new Callback<LoginModel>() {
65                  @Override
66                  public void onResponse(Call<LoginModel> call, Response<
                        LoginModel> response) {
67                      if (response.isSuccessful()) {
68                          token = response.body().getToken();
69                          getSecret();
70                          Enviar();
71                          //Toast.makeText(MainActivity.this, "Iniciando
                                Sesi n...", Toast.LENGTH_LONG).show();
72
73                      } else {
74                          Toast.makeText(MainActivity.this, "Verifique
                                sus credenciales", Toast.LENGTH_LONG).show
                                ();
75                      }
76                  }
77
78                  @Override
79                  public void onFailure(Call<LoginModel> call, Throwable
                        t) {
80                      Toast.makeText(MainActivity.this, "Verifique su
                            conexi n a internet", Toast.LENGTH_LONG).show
                            ();
81
82                  }
83              });
84          }
85      }
```

```
86
87      private void getSecret() {
88          Call<ResponseBody> call = ApiClient.getUserService().getSecret(
                token);
89          call.enqueue(new Callback<ResponseBody>() {
90              @Override
91              public void onResponse(Call<ResponseBody> call, Response<
                    ResponseBody> response) {
92                  if (response.isSuccessful()) {
93                      Toast.makeText(MainActivity.this, "Iniciando
                            Sesi n...", Toast.LENGTH_LONG).show();
94                  } else {
95                      Toast.makeText(MainActivity.this, "Verifica tus
                            credenciales", Toast.LENGTH_LONG).show();
96                  }
97              }
98
99              @Override
100             public void onFailure(Call<ResponseBody> call, Throwable t)
                    {
101
102             }
103         });
104
105     }
106
107     public void Enviar() {
108         Bundle bundle = new Bundle();
109         Intent intent = new Intent(MainActivity.this, MainMenu2.class);
110         // Agregas la informaci n del EditText al Bundle
111         bundle.putString("email", email);
112         // Agregas el Bundle al Intent e inicias ActivityB
113         intent.putExtras(bundle);
114         MainActivity.this.startActivity(intent);
115     }
116 }
```

- **SignUpActivity():** Permite el registro de nuevos usuarios

```
public class SignUpActivity extends AppCompatActivity {
    EditText name;
    EditText username;
    EditText password;
    EditText passwordc;
    ImageView btn_registro;
    String mail;
    boolean flag = false;
    boolean flag1 = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        name = findViewById(R.id.name);
        username = findViewById(R.id.username);
        password = findViewById(R.id.password);
        passwordc = findViewById(R.id.passwordc);
        btn_registro = findViewById(R.id.btn_registro);
        btn_registro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                saveUser(createRequest());
            }
        });
    }

    public void Open() {  //ABRIR UNA NUEVA ACTIVIDADj
        startActivity(new Intent(com.example.npkadvisorfinal.
            SignUpActivity.this, MainActivity.class));
    }

    @NonNull
    public UserRequest createRequest() {
        UserRequest userRequest = new UserRequest();
        userRequest.setNombre(name.getText().toString());
        userRequest.setUsername(username.getText().toString());
        userRequest.setPassword(password.getText().toString());
        userRequest.setPasswordC(passwordc.getText().toString());
        return userRequest;
```

```
40      }
41
42      @NonNull
43      private Boolean validate() { //VALIDAR QUE LOS CAMPOS DEL LOGIN NO
            ESTEN VACIOS
44          Boolean result = false;
45          if (name.getText().toString().isEmpty() || username.getText().
                toString().isEmpty() || password.getText().toString().
                isEmpty() ||
46                  password.getText().toString().isEmpty()) {
47              Toast.makeText(this, "Por favor complete todos los campos",
                    Toast.LENGTH_LONG).show();
48          } else {
49              result = true;
50          }
51          return result;
52      }
53
54      public void saveUser(@NonNull UserRequest userRequest) {
55          mail = username.getText().toString();
56          if (validate()) {
57              if ((userRequest.getPassword().equals(userRequest.
                    getPasswordC()))) {
58                  Call<UserResponse> userResponseCall2 = ApiClient.
                        getUserService().FindUser();
59                  userResponseCall2.enqueue(new Callback<UserResponse>()
                        {
60                      @Override
61                      public void onResponse(Call<UserResponse> call,
                            Response<UserResponse> response) {
62                          ArrayList<UserRequest> cropResponses = response
                                .body().getUsuariosBuscados();
63                          if(response.isSuccessful()) {
64                              for (int i = 0; i < cropResponses.size(); i
                                    ++) {
65                                  if (cropResponses.get(i).getUsername().
                                        equalsIgnoreCase(mail)) {
66                                      Toast.makeText(com.example.
                                            npkadvisorfinal.SignUpActivity.
                                            this, "El email ya est  en uso
                                            ", Toast.LENGTH_LONG).show();
67                                      flag = false;
```

```
68                                  }
69                              }
70                          }
71
72                      }
73
74                      @Override
75                      public void onFailure(Call<UserResponse> call,
                            Throwable t) {
76                          Toast.makeText(com.example.npkadvisorfinal.
                                SignUpActivity.this, "Verifique su
                                conexi n a internet", Toast.LENGTH_LONG).
                                show();
77
78                      }
79                  });
80
81                  if(flag) {
82                      Call<UserResponse> userResponseCall = ApiClient.
                            getUserService().saveUser(userRequest);
83                      userResponseCall.enqueue(new Callback<UserResponse
                            >() {
84                          @Override
85                          public void onResponse(Call<UserResponse> call,
                                Response<UserResponse> response) {
86                              if (response.isSuccessful()) {
87                                  Toast.makeText(com.example.
                                        npkadvisorfinal.SignUpActivity.this
                                        , "Registro exitoso, por favor
                                        inicie sesi n", Toast.LENGTH_LONG)
                                        .show();
88                              }
89                          }
90
91                          @Override
92                          public void onFailure(Call<UserResponse> call,
                                Throwable t) {
93                              Toast.makeText(com.example.npkadvisorfinal.
                                    SignUpActivity.this, "Fall ", Toast.
                                    LENGTH_LONG).show();
94
95                          }
```

```
                        });
                        flag = false;
                    }
                }
            } else {
                Toast.makeText(com.example.npkadvisorfinal.SignUpActivity.
                    this, "Las contrase as no coinciden", Toast.LENGTH_
                    LONG).show();
            }
        }
}
```

- **CropsMenu():** Menú que permite la gestión de cultivos

```
public class CropsMenu extends Fragment {
    ImageView buttonadd, buttonmodify, buttondelete, buttonverify;

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public CropsMenu() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment CropsMenu.
     */
    // TODO: Rename and change types and number of parameters
```

```
26      public static CropsMenu newInstance(String param1, String param2) {
27          CropsMenu fragment = new CropsMenu();
28          Bundle args = new Bundle();
29          args.putString(ARG_PARAM1, param1);
30          args.putString(ARG_PARAM2, param2);
31          fragment.setArguments(args);
32          return fragment;
33      }
34
35      @Override
36      public void onCreate(Bundle savedInstanceState) {
37          super.onCreate(savedInstanceState);
38          if (getArguments() != null) {
39              mParam1 = getArguments().getString(ARG_PARAM1);
40              mParam2 = getArguments().getString(ARG_PARAM2);
41          }
42      }
43
44      @Override
45      public View onCreateView(LayoutInflater inflater, ViewGroup
            container,
46                               Bundle savedInstanceState) {
47          // Inflate the layout for this fragment
48          View vista = inflater.inflate(R.layout.fragment_crops_menu,
                container, false);
49          buttonadd = vista.findViewById(R.id.vercultivos);
50          buttonmodify = vista.findViewById(R.id.reginfo);
51          buttondelete = vista.findViewById(R.id.maps);
52          buttonverify = vista.findViewById(R.id.graphics);
53
54          buttonadd.setOnClickListener(new View.OnClickListener() {
55              @Override
56              public void onClick(View v) {
57                  // Create new fragment and transaction
58                  FragmentManager fragmentManager = getActivity().
                        getSupportFragmentManager();
59                  FragmentTransaction transaction = fragmentManager.
                        beginTransaction();
60                  transaction.setReorderingAllowed(true);
61                  // Replace whatever is in the fragment_container view
                        with this fragment
62                  transaction.replace(R.id.InicialFragment, add_.
```

```
                newInstance("", ""));
63              // Commit the transaction
64              transaction.commit();
65          }
66      });
67
68      buttonmodify.setOnClickListener(new View.OnClickListener() {
69          @Override
70          public void onClick(View v) {
71              // Create new fragment and transaction
72              FragmentManager fragmentManager = getActivity().
                    getSupportFragmentManager();
73              FragmentTransaction transaction = fragmentManager.
                    beginTransaction();
74              transaction.setReorderingAllowed(true);
75              // Replace whatever is in the fragment_container view
                    with this fragment
76              transaction.replace(R.id.InicialFragment, Add_Modify.
                    newInstance("", ""));
77              // Commit the transaction
78              transaction.commit();
79          }
80      });
81
82      buttondelete.setOnClickListener(new View.OnClickListener() {
83          @Override
84          public void onClick(View v) {
85              // Create new fragment and transaction
86              FragmentManager fragmentManager = getActivity().
                    getSupportFragmentManager();
87              FragmentTransaction transaction = fragmentManager.
                    beginTransaction();
88              transaction.setReorderingAllowed(true);
89              // Replace whatever is in the fragment_container view
                    with this fragment
90              transaction.replace(R.id.InicialFragment, Delete_Check.
                    newInstance("", ""));
91              // Commit the transaction
92              transaction.commit();
93          }
94      });
95
```

```
96          buttonverify.setOnClickListener(new View.OnClickListener() {
97              @Override
98              public void onClick(View v) {
99                  // Create new fragment and transaction
100                 FragmentManager fragmentManager = getActivity().
                        getSupportFragmentManager();
101                 FragmentTransaction transaction = fragmentManager.
                        beginTransaction();
102                 transaction.setReorderingAllowed(true);
103                 // Replace whatever is in the fragment_container view
                        with this fragment
104                 transaction.replace(R.id.InicialFragment, Check.
                        newInstance("", ""));
105                 // Commit the transaction
106                 transaction.commit();
107             }
108         });
109
110         return vista;
111     }
112
113
114 }
```

- **ApliClient:** Permite la solicitud y envío de consultas hacía la plataforma de servicios.

```
1  public class ApiClient() {
2
3      @NonNull
4      private static Retrofit getRetrofit(){
5          HttpLoggingInterceptor httpLoggingInterceptor = new
               HttpLoggingInterceptor();
6          httpLoggingInterceptor.setLevel(HttpLoggingInterceptor.Level.
               BODY);
7          OkHttpClient okHttpClient = new OkHttpClient.Builder().
               addInterceptor(httpLoggingInterceptor).build();
8          Retrofit retrofit = new Retrofit.Builder()
```

```
                .baseUrl("http://2a78-201-190-121-53.ngrok.io/") //URL
                    del servidor
                .addConverterFactory(GsonConverterFactory.create())
                .client(okHttpClient)
                .build();
        return retrofit;
    }

    @NonNull
    public static UserService getUserService(){
        UserService userService = getRetrofit().create(UserService.
            class);
        return userService;
    }
}
```

- **add():** Permite el registro de nuevos cultivos

```
public class add_ extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;
    private EditText cropnombre;
    private EditText croparea;
    private ImageView button;

    public add_() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
```

```
22          *
23          * @param param1 Parameter 1.
24          * @param param2 Parameter 2.
25          * @return A new instance of fragment add_.
26          */
27         // TODO: Rename and change types and number of parameters
28         @NonNull
29         public static add_ newInstance(String param1, String param2) {
30             com.example.npkadvisorfinal.add_ fragment = new com.example.
                   npkadvisorfinal.add_();
31             Bundle args = new Bundle();
32             args.putString(ARG_PARAM1, param1);
33             args.putString(ARG_PARAM2, param2);
34             fragment.setArguments(args);
35             return fragment;
36         }
37
38         @Override
39         public void onCreate(Bundle savedInstanceState) {
40             super.onCreate(savedInstanceState);
41             if (getArguments() != null) {
42                 mParam1 = getArguments().getString(ARG_PARAM1);
43                 mParam2 = getArguments().getString(ARG_PARAM2);
44             }
45         }
46
47         @Override
48         public View onCreateView(@NonNull LayoutInflater inflater,
               ViewGroup container,
49                                  Bundle savedInstanceState) {
50             // Inflate the layout for this fragment
51             View view = inflater.inflate(R.layout.fragment_add, container,
                   false);
52
53             cropnombre = view.findViewById(R.id.cropname);
54             croparea = view.findViewById(R.id.croparea);
55             button = view.findViewById(R.id.btn_cropadd);
56             button.setOnClickListener(new View.OnClickListener() {
57                 @Override
58                 public void onClick(View view) {
59                     saveCrop(createRequest());
60                 }
```

```
61          });
62
63          return view;
64      }
65
66      @NonNull
67      public CropResponse2 createRequest(){
68          CropResponse2 cropRequest = new CropResponse2();
69          cropRequest.setCNombre(cropnombre.getText().toString());
70          cropRequest.setCArea( Double.parseDouble(croparea.getText().
                toString()));
71          return cropRequest;
72      }
73
74      public void saveCrop(CropResponse2 cropRequest) {
75                  Call<CropResponse> userResponseCall = ApiClient.
                        getUserService().saveCrop(cropRequest);
76                  userResponseCall.enqueue(new Callback<CropResponse>() {
77                      @Override
78                      public void onResponse(Call<CropResponse> call,
                            Response<CropResponse> response) {
79                          if (response.isSuccessful()) {
80
81                                  Toast.makeText(getContext(), "Registro
                                        exitoso", Toast.LENGTH_LONG).show()
                                        ;
82                                  cropnombre.setText("");
83                                  croparea.setText("");
84                                  button.setEnabled(false);
85
86                          }
87                      }
88
89                      @Override
90                      public void onFailure(Call<CropResponse> call,
                            Throwable t) {
91                          Toast.makeText(getContext(), "Verifique su
                                conexi n a internet", Toast.LENGTH_LONG).
                                show();
92                      }
93                  });
94      }
```

```
95
96
97 }
```

- **AddModify():** Permite modificar información de cultivos ya existente.

```
1  public class Add_Modify extends Fragment {
2
3      // TODO: Rename parameter arguments, choose names that match
4      // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
5      private static final String ARG_PARAM1 = "param1";
6      private static final String ARG_PARAM2 = "param2";
7
8      // TODO: Rename and change types of parameters
9      private String mParam1;
10     private String mParam2;
11     private Spinner spinnermodify;
12     private ImageButton btn_modify;
13     private String ID;
14     private String cropname;
15     private String croparea;
16     private ArrayList<String> cropss = new ArrayList<>();
17
18     public Add_Modify() {
19         // Required empty public constructor
20     }
21
22     /**
23      * Use this factory method to create a new instance of
24      * this fragment using the provided parameters.
25      *
26      * @param param1 Parameter 1.
27      * @param param2 Parameter 2.
28      * @return A new instance of fragment Add_Modify.
29      */
30     // TODO: Rename and change types and number of parameters
31     @NonNull
32     public static com.example.npkadvisorfinal.Add_Modify newInstance(
           String param1, String param2) {
```

```
33          com.example.npkadvisorfinal.Add_Modify fragment = new com.
                example.npkadvisorfinal.Add_Modify();
34          Bundle args = new Bundle();
35          args.putString(ARG_PARAM1, param1);
36          args.putString(ARG_PARAM2, param2);
37          fragment.setArguments(args);
38          return fragment;
39      }
40
41      @Override
42      public void onCreate(Bundle savedInstanceState) {
43          super.onCreate(savedInstanceState);
44          if (getArguments() != null) {
45              mParam1 = getArguments().getString(ARG_PARAM1);
46              mParam2 = getArguments().getString(ARG_PARAM2);
47          }
48      }
49
50      @Override
51      public View onCreateView(@NonNull LayoutInflater inflater,
            ViewGroup container,
52                               Bundle savedInstanceState) {
53          // Inflate the layout for this fragment
54          View view = inflater.inflate(R.layout.fragment_add__modify,
                container, false);
55
56
57          ShowCrop();
58          spinnermodify = view.findViewById(R.id.spinnermodify);
59          btn_modify = view.findViewById(R.id.modifyy);
60          btn_modify.setOnClickListener(new View.OnClickListener() {
61              @Override
62              public void onClick(View v) {
63                  Enviar(view);
64
65              }
66          });
67          return view;
68
69      }
70
71      public void ShowCrop() {
```

```
72          Call<CropResponse> cropResponseCall = ApiClient.getUserService
                ().findAllC();
73          cropResponseCall.enqueue(new Callback<CropResponse>() {
74              @Override
75              public void onResponse(Call<CropResponse> call, Response<
                    CropResponse> response) {
76                  if (response.isSuccessful()) {
77                      ArrayList<CropResponse2> cropResponses = response.
                            body().getCultivosBuscados();
78                      for (int i = 0; i < cropResponses.size(); i++) {
79                          cropss.add(cropResponses.get(i).getCNombre());
80                      }
81                      ArrayAdapter<CharSequence> adaptador = new
                            ArrayAdapter(getActivity().getBaseContext(),
                            android.R.layout.simple_spinner_item, cropss);
82                      spinnermodify.setAdapter(adaptador);
83                      spinnermodify.setOnItemSelectedListener(new
                            AdapterView.OnItemSelectedListener() {
84                          @Override
85                          public void onItemSelected(AdapterView<?>
                                parent, View view, int position, long id) {
86                              ID = cropResponses.get(position).getId();
87                              cropname = cropResponses.get(position).
                                    getCNombre();
88                              croparea = cropResponses.get(position).
                                    getCArea().toString();
89                          }
90
91                          @Override
92                          public void onNothingSelected(AdapterView<?>
                                parent) {
93
94                          }
95                      });
96                  }
97              }
98              @Override
99              public void onFailure(Call<CropResponse> call, Throwable t)
                    {
100                 Toast.makeText(getContext(), "Verifique su conexi n a
                        internet", Toast.LENGTH_LONG).show();
101                 //System.out.println("causes" + t.fillInStackTrace());
```

```
            }
        });
    }
    public void Enviar(View view){
        Intent intent = new Intent(getActivity(), Modify.class);
        intent.putExtra("id", ID);
        intent.putExtra("name", cropname);
        intent.putExtra("area", croparea);
        getActivity().startActivity(intent);
    }



}
```

- **DeleteCheck():** Permite eliminar cultivos ya registrados

```
public class Delete_Check extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;
    private Spinner spinnerdelete1;
    private ImageView btn_delete;
    private String ID;
    private ArrayList<String> cropss = new ArrayList<>();

    public Delete_Check() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
```

```
23          *
24          * @param param1 Parameter 1.
25          * @param param2 Parameter 2.
26          * @return A new instance of fragment Delete_Check.
27          */
28         // TODO: Rename and change types and number of parameters
29         @NonNull
30         public static com.example.npkadvisorfinal.Delete_Check newInstance(
               String param1, String param2) {
31             com.example.npkadvisorfinal.Delete_Check fragment = new com.
                   example.npkadvisorfinal.Delete_Check();
32             Bundle args = new Bundle();
33             args.putString(ARG_PARAM1, param1);
34             args.putString(ARG_PARAM2, param2);
35             fragment.setArguments(args);
36             return fragment;
37         }
38
39         @Override
40         public void onCreate(Bundle savedInstanceState) {
41             super.onCreate(savedInstanceState);
42             if (getArguments() != null) {
43                 mParam1 = getArguments().getString(ARG_PARAM1);
44                 mParam2 = getArguments().getString(ARG_PARAM2);
45             }
46         }
47
48         @Override
49         public View onCreateView(@NonNull LayoutInflater inflater,
               ViewGroup container,
50                                  Bundle savedInstanceState) {
51             View view = inflater.inflate(R.layout.fragment_delete__check,
                   container, false);
52             spinnerdelete1 = view.findViewById(R.id.spinnerdelete);
53             btn_delete = view.findViewById(R.id.btn_eliminar);
54             DeleteCrop();
55             btn_delete.setOnClickListener(new View.OnClickListener() {
56                 @Override
57                 public void onClick(View v) {
58                     deletecropdb();
59                     btn_delete.setEnabled(false);
60                 }
```

```
61          });
62          // Inflate the layout for this fragment
63          return view;
64      }
65
66       public void DeleteCrop() {
67          Call<CropResponse> cropResponseCall = ApiClient.getUserService
                ().findAllC();
68
69          cropResponseCall.enqueue(new Callback<CropResponse>() {
70              @Override
71              public void onResponse(Call<CropResponse> call, Response<
                    CropResponse> response) {
72                  if (response.isSuccessful()) {
73                      ArrayList<CropResponse2> cropResponses2 = response
                            .body().getCultivosBuscados();
74                      for (int i = 0; i < cropResponses2.size(); i++) {
75                          cropss.add(cropResponses2.get(i).getCNombre())
                                ;
76                      }
77                      ArrayAdapter<CharSequence> adaptador = new
                            ArrayAdapter(getActivity().getBaseContext(),
                            android.R.layout.simple_spinner_item, cropss);
78                      spinnerdelete1.setAdapter(adaptador);
79                      spinnerdelete1.setOnItemSelectedListener(new
                            AdapterView.OnItemSelectedListener() {
80                          @Override
81                          public void onItemSelected(AdapterView<?>
                                parent, View view, int position, long id)
                                {
82                              ID = cropResponses2.get(position).getId();
83                          }
84
85                          @Override
86                          public void onNothingSelected(AdapterView<?>
                                parent) {
87
88                          }
89                      });
90                  }
91              }
92              @Override
```

```
93                public void onFailure(Call<CropResponse> call, Throwable t
                      ) {
94                    Toast.makeText(getContext(), "Verifique su conexi n a
                          internet", Toast.LENGTH_LONG).show();
95                    //System.out.println("causes" + t.fillInStackTrace());
96                }
97            });
98
99        }
100
101       public void deletecropdb(){
102
103           Call<CropResponse> cropResponseCall = ApiClient.getUserService
                  ().delete(ID);
104           ArrayList<String> cropssId = new ArrayList<>();
105           cropResponseCall.enqueue(new Callback<CropResponse>() {
106               @Override
107               public void onResponse(Call<CropResponse> call, Response<
                      CropResponse> response) {
108                   if (response.isSuccessful()) {
109                       Toast.makeText(getContext(), "Registro Eliminado",
                              Toast.LENGTH_LONG).show();
110                   }
111               }
112               @Override
113               public void onFailure(Call<CropResponse> call, Throwable t
                      ) {
114                   Toast.makeText(getContext(), "Verifique su conexi n a
                          internet", Toast.LENGTH_LONG).show();
115               }
116           });
117
118
119       }
120
121   }
```

- **History:** Permite consultar toda la información de un cultivo filtrado por fechas y las muestra en una tabla

```
public class History extends AppCompatActivity {
    ImageButton ChooseDate;
    ImageButton ChooseDateHasta;
    Button csvExport;
    Button Saveb;
    TextView ChooseT1;
    TextView ChooseT;
    TableRow fila;
    TextView Humedad;
    TextView N;
    TextView P;
    TextView K;
    TextView Temp;
    TextView Datex;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_history);
        ChooseDate = findViewById(R.id.choosedate);
        ChooseDateHasta = findViewById(R.id.choosedate1);
        ChooseT = findViewById(R.id.chooseText);
        ChooseT1 = findViewById(R.id.choosedateT1);
        csvExport = findViewById(R.id.csv);
        Saveb = findViewById(R.id.saveDate);

        Saveb.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Save();
                Saveb.setEnabled(false);
            }
        });

        RequestPermissions();

        csvExport.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                datalist();
```

```
40                  Toast.makeText(com.example.npkadvisorfinal.History.this
                        ,"Se creo existosamente" ,Toast.LENGTH_SHORT).show
                        ();
41                  //humedadlist();
42              }
43          });
44
45          ChooseDateHasta.setOnClickListener(new View.OnClickListener() {
46              @RequiresApi(api = Build.VERSION_CODES.N)
47              @Override
48              public void onClick(View v) {
49                  ChooseDate1();
50              }
51          });
52          ChooseDate.setOnClickListener(new View.OnClickListener() {
53              @RequiresApi(api = Build.VERSION_CODES.N)
54              @Override
55              public void onClick(View v) {
56                  ChooseDate();
57              }
58          });
59
60      }
61      public void Save(){
62          TableRow.LayoutParams layoutFila = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
63          TableRow.LayoutParams layouthumedad = new TableRow.LayoutParams
                (TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
64          TableRow.LayoutParams layoutN = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
65          TableRow.LayoutParams layoutP = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
66          TableRow.LayoutParams layoutK = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
67          TableRow.LayoutParams layoutTemp = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
```

```
68          TableRow.LayoutParams layoutDate = new TableRow.LayoutParams(
                TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.
                WRAP_CONTENT);
69          Call<IndexResponse> indexResponseCall = ApiClient.
                getUserService().findIndex();
70          indexResponseCall.enqueue(new Callback<IndexResponse>() {
71              @Override
72              public void onResponse(Call<IndexResponse> call, Response<
                    IndexResponse> response) {
73                  if (response.isSuccessful()) {
74                      TableLayout lista = findViewById(R.id.table);
75                      ArrayList<IndexResponse2> IndexReponses = response.
                            body().getInfoIndex();
76                      for (int i = 0; i < IndexReponses.size(); i++) {
77                          SimpleDateFormat sdf = new SimpleDateFormat("dd
                                /MM/yyyy");
78                          try {
79                              Date date1 = sdf.parse(ChooseT.getText().
                                    toString());
80                              Date date2 = sdf.parse(ChooseT1.getText().
                                    toString());
81                              Date datedb = sdf.parse(IndexReponses.get(i
                                    ).getCreateAt());
82
83                              if ((datedb.after(date1)||datedb.equals(
                                    date1)) && (datedb.before(date2)||
                                    datedb.equals(date2))) {
84                                  fila = new TableRow(com.example.
                                        npkadvisorfinal.History.this);
85                                  fila.setLayoutParams(layoutFila);
86                                  if (i == 0) {
87                                      Datex = new TextView(com.example.
                                            npkadvisorfinal.History.this);
88                                      Datex.setText("FECHA");
89                                      Datex.setGravity(Gravity.CENTER);
90                                      Datex.setBackgroundColor(Color.
                                            BLACK);
91                                      Datex.setTextColor(Color.WHITE);
92                                      Datex.setPadding(10, 10, 10, 10);
93                                      Datex.setLayoutParams(layoutDate);
94                                      fila.addView(Datex);
95
```

```
96                  Humedad = new TextView(com.example.
                        npkadvisorfinal.History.this);
97                  Humedad.setText("HUMEDAD");
98                  Humedad.setGravity(Gravity.CENTER);
99                  Humedad.setBackgroundColor(Color.
                        BLACK);
100                 Humedad.setTextColor(Color.WHITE);
101                 Humedad.setPadding(10, 10, 10, 10);
102                 Humedad.setLayoutParams(
                        layouthumedad);
103                 fila.addView(Humedad);
104
105                 N = new TextView(com.example.
                        npkadvisorfinal.History.this);
106                 N.setText("N");
107                 N.setGravity(Gravity.CENTER);
108                 N.setBackgroundColor(Color.BLACK);
109                 N.setTextColor(Color.WHITE);
110                 N.setPadding(10, 10, 10, 10);
111                 N.setLayoutParams(layoutN);
112                 fila.addView(N);
113
114                 P = new TextView(com.example.
                        npkadvisorfinal.History.this);
115                 P.setText("P");
116                 P.setGravity(Gravity.CENTER);
117                 P.setBackgroundColor(Color.BLACK);
118                 P.setTextColor(Color.WHITE);
119                 P.setPadding(10, 10, 10, 10);
120                 P.setLayoutParams(layoutP);
121                 fila.addView(P);
122
123                 K = new TextView(com.example.
                        npkadvisorfinal.History.this);
124                 K.setText("K");
125                 K.setGravity(Gravity.CENTER);
126                 K.setBackgroundColor(Color.BLACK);
127                 K.setTextColor(Color.WHITE);
128                 K.setPadding(10, 10, 10, 10);
129                 K.setLayoutParams(layoutK);
130                 fila.addView(K);
131
```

```
132                     Temp = new TextView(com.example.
                            npkadvisorfinal.History.this);
133                     Temp.setText("T  ");
134                     Temp.setGravity(Gravity.CENTER);
135                     Temp.setBackgroundColor(Color.BLACK
                            );
136                     Temp.setTextColor(Color.WHITE);
137                     Temp.setPadding(10, 10, 10, 10);
138                     Temp.setLayoutParams(layoutTemp);
139                     fila.addView(Temp);
140                     lista.addView(fila);
141                 } else {
142                     Datex = new TextView(com.example.
                            npkadvisorfinal.History.this);
143                     Datex.setText(IndexReponses.get(i).
                            getCreateAt());
144                     Datex.setPadding(10, 10, 10, 10);
145                     Datex.setGravity(Gravity.CENTER);
146                     Datex.setBackgroundColor(Color.BLUE
                            );
147                     Datex.setLayoutParams(layoutDate);
148                     fila.addView(Datex);
149
150                     Humedad = new TextView(com.example.
                            npkadvisorfinal.History.this);
151                     Humedad.setText(IndexReponses.get(i
                            ).getHumedad().toString());
152                     Humedad.setPadding(10, 10, 10, 10);
153                     Humedad.setGravity(Gravity.CENTER);
154                     Humedad.setBackgroundColor(Color.
                            BLUE);
155                     Humedad.setLayoutParams(
                            layouthumedad);
156                     fila.addView(Humedad);
157
158                     N = new TextView(com.example.
                            npkadvisorfinal.History.this);
159                     N.setText(IndexReponses.get(i).getN
                            ().toString());
160                     N.setPadding(10, 10, 10, 10);
161                     N.setGravity(Gravity.CENTER);
162                     N.setBackgroundColor(Color.BLUE);
```

```
163                                 N.setLayoutParams(layoutN);
164                                 fila.addView(N);
165
166                                 P = new TextView(com.example.
                                        npkadvisorfinal.History.this);
167                                 P.setText(IndexReponses.get(i).getP
                                        ().toString());
168                                 P.setPadding(10, 10, 10, 10);
169                                 P.setGravity(Gravity.CENTER);
170                                 P.setBackgroundColor(Color.BLUE);
171                                 P.setLayoutParams(layoutP);
172                                 fila.addView(P);
173
174                                 K = new TextView(com.example.
                                        npkadvisorfinal.History.this);
175                                 K.setText(IndexReponses.get(i).getK
                                        ().toString());
176                                 K.setPadding(10, 10, 10, 10);
177                                 K.setGravity(Gravity.CENTER);
178                                 K.setBackgroundColor(Color.BLUE);
179                                 K.setLayoutParams(layoutK);
180                                 fila.addView(K);
181
182                                 Temp = new TextView(com.example.
                                        npkadvisorfinal.History.this);
183                                 Temp.setText(IndexReponses.get(i).
                                        getTemp().toString());
184                                 Temp.setPadding(10, 10, 10, 10);
185                                 Temp.setGravity(Gravity.CENTER);
186                                 Temp.setBackgroundColor(Color.BLUE)
                                        ;
187                                 Temp.setLayoutParams(layoutTemp);
188                                 fila.addView(Temp);
189
190                                 lista.addView(fila);
191                             }
192                         }
193                     } catch (ParseException e) {
194                         e.printStackTrace();
195                     }
196                 }
197             }
```

```
198             }
199             @Override
200             public void onFailure(Call<IndexResponse> call, Throwable t
                    ) {
201                 Toast.makeText(com.example.npkadvisorfinal.History.this
                        , "Request Failed", Toast.LENGTH_LONG).show();
202
203             }
204         });
205     }
206
207
208     @RequiresApi(api = Build.VERSION_CODES.N)
209     public void ChooseDate(){
210         Calendar rightNow = Calendar.getInstance(); //FORMATO DATE
211         int day = rightNow.get(Calendar.DAY_OF_MONTH);
212         int month = rightNow.get(Calendar.MONTH);
213         int year = rightNow.get(Calendar.YEAR);
214
215         DatePickerDialog datePickerDialog = new DatePickerDialog(this,
                new DatePickerDialog.OnDateSetListener() {
216             @Override
217             public void onDateSet(DatePicker view, int year, int
                    monthOfYear, int dayOfMonth) {
218                 ChooseT.setText(dayOfMonth+ "/" +(monthOfYear+1)+"/"+
                        year);
219             }
220         }
221         ,day,month,year);
222         datePickerDialog.show();
223     }
224
225     public void ChooseDate1(){
226         Calendar rightNow = Calendar.getInstance(); //FORMATO DATE
227         int day1 = rightNow.get(Calendar.DAY_OF_MONTH);
228         int month1 = rightNow.get(Calendar.MONTH);
229         int year1 = rightNow.get(Calendar.YEAR) - 2021;
230
231         DatePickerDialog datePickerDialog = new DatePickerDialog(this,
                new DatePickerDialog.OnDateSetListener() {
232             @Override
233             public void onDateSet(DatePicker view, int year, int
```

```
                monthOfYear, int dayOfMonth) {
234                 ChooseT1.setText(dayOfMonth+ "/" +(monthOfYear+1)+"/"+
                        year);
235             }
236         }
237                 ,day1,month1,year1);
238         datePickerDialog.show();
239     }
240
241     public void RequestPermissions(){
242
243         if(ContextCompat.checkSelfPermission(com.example.
                npkadvisorfinal.History.this, Manifest.permission.WRITE_
                EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
244             ActivityCompat.requestPermissions(com.example.
                    npkadvisorfinal.History.this, new String []{Manifest.
                    permission.READ_EXTERNAL_STORAGE, Manifest.permission.
                    WRITE_EXTERNAL_STORAGE},0);
245         }
246     }
247     public void datalist(){
248         File file = new File(Environment.getExternalStorageDirectory()
                + "/CSV");
249         String archivo = file.toString()+"/"+"NPK.csv";
250
251         boolean isCreate = false;
252         if(!file.exists()){
253             isCreate = file.mkdir();
254         }
255         Call<IndexResponse> indexResponseCall = ApiClient.
                getUserService().findIndex1();
256         ArrayList<String> data = new ArrayList<>();
257         indexResponseCall.enqueue(new Callback<IndexResponse>() {
258             @Override
259             public void onResponse(Call<IndexResponse> call, Response<
                    IndexResponse> response) {
260                 if (response.isSuccessful()) {
261                     ArrayList<IndexResponse2> IndexReponses = response.
                            body().getInfoIndex();
262
263                     try {
264                         FileWriter fileWriter = new FileWriter(archivo)
```

```
                ;
265             fileWriter.write("N");
266             fileWriter.write(",");
267             fileWriter.write("P");
268             fileWriter.write(",");
269             fileWriter.write("K");
270             fileWriter.write(",");
271             fileWriter.write("Hum");
272             fileWriter.write(",");
273             fileWriter.write("T  ");
274             fileWriter.write("\n");
275
276
277             for (int i = 0; i < IndexReponses.size(); i++)
                {
278                 SimpleDateFormat sdf = new SimpleDateFormat
                        ("dd/MM/yyyy");
279                 try {
280                     Date date1 = sdf.parse(ChooseT.getText
                            ().toString());
281                     Date date2 = sdf.parse(ChooseT1.getText
                            ().toString());
282                     Date datedb = sdf.parse(IndexReponses.
                            get(i).getCreateAt());
283                     if ((datedb.after(date1) || datedb.
                            equals(date1)) && (datedb.before(
                            date2) || datedb.equals(date2))) {
284                         fileWriter.write(IndexReponses.get(
                                i).getN().toString());
285                         fileWriter.write(",");
286                         fileWriter.write(IndexReponses.get(
                                i).getP().toString());
287                         fileWriter.write(",");
288                         fileWriter.write(IndexReponses.get(
                                i).getK().toString());
289                         fileWriter.write(",");
290                         fileWriter.write(IndexReponses.get(
                                i).getHumedad().toString());
291                         fileWriter.write(",");
292                         fileWriter.write(IndexReponses.get(
                                i).getTemp().toString());
293                         fileWriter.write("\n");
```

```
294                                 }
295                             } catch (Exception e) {
296                                 e.printStackTrace();
297                             }
298                         }
299                         fileWriter.flush();
300                         fileWriter.close();
301                     } catch (IOException e) {
302                         e.printStackTrace();
303                     }
304                 }
305             }
306             @Override
307             public void onFailure(Call<IndexResponse> call, Throwable t
                    ) {
308                 Toast.makeText(com.example.npkadvisorfinal.History.this
                        , "Request Failed", Toast.LENGTH_LONG).show();
309             }
310         });
311     }
312
313
314     public void humedadlist(){
315         File file = new File(Environment.getExternalStorageDirectory()
                + "/CSV");
316         String archivo = file.toString()+"/"+"Humedad.csv";
317
318         boolean isCreate = false;
319         if(!file.exists()){
320             isCreate = file.mkdir();
321         }
322         Call<IndexResponse> indexResponseCall = ApiClient.
                getUserService().findIndex1();
323         ArrayList<String> data = new ArrayList<>();
324         indexResponseCall.enqueue(new Callback<IndexResponse>() {
325             @Override
326             public void onResponse(Call<IndexResponse> call, Response<
                    IndexResponse> response) {
327                 if (response.isSuccessful()) {
328                     ArrayList<IndexResponse2> IndexReponses = response.
                            body().getInfoIndex();
329                     try {
```

```
330                         FileWriter fileWriter = new FileWriter(archivo)
                                ;
331                         int count = 0;
332                         for (int i = 0; i < IndexReponses.size(); i++)
                                {
333                             count++;
334                             fileWriter.write(IndexReponses.get(i).
                                    getHumedad().toString());
335                             fileWriter.write(",");
336                             if (count == 5){
337                                 fileWriter.write("\n");
338                                 count = 0;
339                             }
340                         }
341                         fileWriter.flush();
342                         fileWriter.close();
343                     }
344                     catch (Exception e){}
345                 }
346
347             }
348
349             @Override
350             public void onFailure(Call<IndexResponse> call, Throwable t
                    ) {
351                 Toast.makeText(com.example.npkadvisorfinal.History.this
                        , "Request Failed", Toast.LENGTH_LONG).show();
352             }
353         });
354     }
355 }
```

- **GraficBar():** Muestra los datos filtrados por fecha en un grafico de barras, la lógica es similar para las gráficas LineChart y PieChart haciendo uso de sus respectivos comandos según la librería

```
public class GraficBar extends AppCompatActivity {

    private BarChart bar;
    int[] colorClassArray = new int[]{Color.RED, Color.YELLOW, Color.
        GREEN, Color.WHITE, Color.CYAN};
    private String desde;
    private String hasta;

    @Override
    protected void onCreate(android.os.Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grafic_bar);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN
            , WindowManager.LayoutParams.FLAG_FULLSCREEN);
        Bundle bundle = getIntent().getExtras();
        desde =  bundle.getString("desde");
        hasta = bundle.getString("hasta");
        bar = findViewById(R.id.barChart);
        Databar();

    }

    public void Databar() {
        ArrayList<BarEntry> datan = new ArrayList<>();
        ArrayList<BarEntry> datap = new ArrayList<>();
        ArrayList<BarEntry> datak = new ArrayList<>();
        ArrayList<BarEntry> datat = new ArrayList<>();
        ArrayList<BarEntry> datah = new ArrayList<>();

        Call<IndexResponse> cropResponseCall = ApiClient.getUserService
            ().findIndex();
        cropResponseCall.enqueue(new Callback<IndexResponse>() {
            @Override
            public void onResponse(Call<IndexResponse> call, Response<
                IndexResponse> response) {
                if (response.isSuccessful()) {
                    ArrayList<IndexResponse2> indexResponse = response.
                        body().getInfoIndex();
```

```
34                  for (int i = 0; i < indexResponse.size(); i++) {
35                      SimpleDateFormat sdf = new SimpleDateFormat("dd
                            /MM/yyyy");
36                      try {
37                          Date date1 = sdf.parse(desde);
38                          Date date2 = sdf.parse(hasta);
39                          Date datedb = sdf.parse(indexResponse.get(i
                                ).getCreateAt());
40
41                          if ((datedb.after(date1) || datedb.equals(
                                date1)) && (datedb.before(date2) ||
                                datedb.equals(date2))) {
42                              //android.util.Log.d(TAG, "onResponse:
                                    \n " +
43                              //        "Cultivo " + indexResponse.get(
                                    i).getHumedad());
44                              datan.add(new BarEntry(i, Float.
                                    parseFloat(indexResponse.get(i).
                                    getN().toString())));
45                              datap.add(new BarEntry(i, Float.
                                    parseFloat(indexResponse.get(i).
                                    getP().toString())));
46                              datak.add(new BarEntry(i, Float.
                                    parseFloat(indexResponse.get(i).
                                    getK().toString())));
47                              datat.add(new BarEntry(i, Float.
                                    parseFloat(indexResponse.get(i).
                                    getTemp().toString())));
48                              datah.add(new BarEntry(i, Float.
                                    parseFloat(indexResponse.get(i).
                                    getHumedad().toString())));
49                          }
50                      }catch (ParseException e){
51                          e.printStackTrace();
52                      }
53                  }
54                  BarDataSet set1,set2,set3,set4,set5;
55
56                  set1= new BarDataSet(datan,"N");
57                  set1.setColor(Color.BLUE);
58                  set2= new BarDataSet(datap,"P");
59                  //set2.setColor(Color.RED);
```

```
60                      set3= new BarDataSet(datak,"K");
61                      //set3.setColor(Color.YELLOW);
62                      set4= new BarDataSet(datat," T  ");
63                      //set4.setColor(Color.GREEN);
64                      set5= new BarDataSet(datah,"Hum ");
65                      //set5.setColor(Color.CYAN);
66                      BarData barData =  new BarData(set1,set2,set3,set4,
                            set5);
67                      bar.setData(barData);
68                      bar.invalidate();
69                  }
70              }
71              @Override
72              public void onFailure(Call<IndexResponse> call, Throwable t
                    ) {
73                  Toast.makeText(GraficBar.this, "Verifique su conexi n
                        a internet", Toast.LENGTH_LONG).show();
74                  //System.out.println("causes" + t.fillInStackTrace());
75              }
76          });
77      }
78  }
```

- **Mapas():** Permite la gestión de información con el Api Key de Google MAPS

```
1  public class Mapas extends AppCompatActivity implements View.
       OnClickListener {
2
3      private FusedLocationProviderClient mFusedLocationClient;
4      private int MY_PERMISSION_REQUEST_READ_CONTACTS;
5      private Button maps;
6
7      @SuppressLint("WrongViewCast")
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_mapas);
12         mFusedLocationClient = LocationServices.
               getFusedLocationProviderClient(this);
```

```
13          LatLong();
14          maps = findViewById(R.id.mapa);
15          maps.setOnClickListener(this);
16      }
17
18      private void LatLong() {
19          if (ActivityCompat.checkSelfPermission(this, Manifest.
                permission.ACCESS_FINE_LOCATION) != PackageManager.
                PERMISSION_GRANTED
20                  && ActivityCompat.checkSelfPermission(this, Manifest.
                        permission.ACCESS_COARSE_LOCATION) !=
                        PackageManager.PERMISSION_GRANTED) {
21              ActivityCompat.requestPermissions(com.example.
                    npkadvisorfinal.Mapas.this, new String []{Manifest.
                    permission.ACCESS_FINE_LOCATION}, MY_PERMISSION_REQUEST
                    _READ_CONTACTS);
22              return;
23          }
24          mFusedLocationClient.getLastLocation()
25                  .addOnSuccessListener(this, new OnSuccessListener<
                        Location>() {
26                      @Override
27                      public void onSuccess(Location location) {
28                          if (location != null) {
29                              Log.e("Latitud",+location.getLatitude()+"
                                    Longitud"+location.getLongitude());
30                              Toast.makeText(Mapas.this, "Medidas
                                    Obtenidas", Toast.LENGTH_LONG).show();
31
32
33                          }
34                          else
35                          {
36                              System.out.println("da ado");
37                          }
38                      }
39                  });
40      }
41
42      @Override
43      public void onClick(View v) {
44          switch(v.getId()){
```

```
45              case R.id.mapa : Intent intent = new Intent(Mapas.this,
                    MapsActivity.class);
46                              startActivity(intent);
47                              break;
48          }
49      }
50  }
```

- **Model():** Modelo POJO para la gestión de información, se creó para los modelos de persona, cultivo, e información de las variables

```
1  public class UserRequest {
2
3      @SerializedName("_id")
4      @Expose
5      private String id;
6
7      @SerializedName("Nombre")
8      @Expose
9      private String Nombre;
10
11     @SerializedName("Username")
12     @Expose
13     private String username;
14
15     @SerializedName("Password")
16     @Expose
17     private String password;
18
19     @SerializedName("PasswordC")
20     @Expose
21     private String passwordC;
22
23     @SerializedName("__v")
24     @Expose
25     private Integer v;
26
27     @SerializedName("Create_at")
28     @Expose
```

```
29      private String createat;
30
31      public String getId() {
32          return id;
33      }
34
35      public void setId(String id) {
36          this.id = id;
37      }
38
39      public String getNombre() {
40          return Nombre;
41      }
42
43      public void setNombre(String nombre) {
44          Nombre = nombre;
45      }
46
47      public String getUsername() {
48          return username;
49      }
50
51      public void setUsername(String username) {
52          this.username = username;
53      }
54
55      public String getPassword() {
56          return password;
57      }
58
59      public void setPassword(String password) {
60          this.password = password;
61      }
62
63      public String getPasswordC() {
64          return passwordC;
65      }
66
67      public void setPasswordC(String passwordC) {
68          this.passwordC = passwordC;
69      }
70
```

```
71      public Integer getV() {
72          return v;
73      }
74
75      public void setV(Integer v) {
76          this.v = v;
77      }
78
79      public String getCreateat() {
80          return createat;
81      }
82
83      public void setCreateat(String createat) {
84          this.createat = createat;
85      }
86      @NonNull
87      @Override
88      public String toString() {
89          return "UserRequest{" +
90                  "id='" + id + '\'' +
91                  ", Nombre='" + Nombre + '\'' +
92                  ", username='" + username + '\'' +
93                  ", password='" + password + '\'' +
94                  ", passwordC='" + passwordC + '\'' +
95                  ", v=" + v +
96                  ", createat='" + createat + '\'' +
97                  '}';
98      }
99  }
```

- **UserService():** Interfaz para configuración de solicitudes HTTP al servidor por medio de rutas URL

```
1   public interface UserService {
2
3       //A adir un nuevo usuario
4       @NonNull
5       @POST("persona")
6       Call<UserResponse> saveUser(@Body UserRequest userRequest);
```

```
7
8       //Encontrar usuarios
9       @NonNull
10      @GET("persona")
11      Call<UserResponse> FindUser();
12
13      //Login Authentication
14      @NonNull
15      @POST("persona/login")
16      Call<LoginModel>login(@Body Login login);
17
18      //Secret token authentication
19      @NonNull
20      @GET("cultivo")
21      Call<ResponseBody> getSecret(@Header("Authorization")String
            authToken);
22
23
24      //a adir un cultivo
25      @NonNull
26      @POST("cultivo")
27      Call<CropResponse>saveCrop(@Body CropResponse2 cropRequest);
28
29      //consultar todos los cultivos
30      @NonNull
31      @GET("cultivo")
32      Call<CropResponse>findAllC();
33
34      //Consultar variables Humedad, Npk, Temperatura
35      @NonNull
36      @GET("index")
37      Call<IndexResponse>findIndex();
38
39      //Consultar variables Humedad, NPK, Temperatura
40      @NonNull
41      @GET("index")
42      Call<IndexResponse>findIndex1();
43
44      //Eliminar cultivo
45      @NonNull
46      @DELETE("cultivo/{Id}")
47      Call<CropResponse>delete(@Path("Id") String cropId);
```

```
48
49      //Actualizar cultivos
50      @Headers({"Content-Type: application/json"})
51      @PUT("cultivo/{id}")
52      Call<CropResponse> updateCrop (@Path("id") String id, @Body
            CropResponse2 body);
53
54
55  }
```

- **Codigo implementado en la tarjeta de desarollo Arduino Nano:**

```
1  #include <SoftwareSerial.h>
2  #include <Wire.h>
3  #include <max6675.h>
4
5  #define RE 8
6  #define DE 7
7
8  // ThermoCouple
9  int thermo_so_pin  = 12;
10 int thermo_cs_pin  = 10;
11 int thermo_sck_pin = 13;
12 MAX6675 thermocouple(thermo_sck_pin, thermo_cs_pin, thermo_so_pin);
13
14 //PRUEBA1
15 //const byte code[]= {0x01, 0x03, 0x00, 0x1e, 0x00, 0x03, 0x65, 0xCD};
16 const byte nitro[] = {0x01,0x03, 0x00, 0x1e, 0x00, 0x01, 0xe4, 0x0c};
17 const byte phos[] = {0x01,0x03, 0x00, 0x1f, 0x00, 0x01, 0xb5, 0xcc};
18 const byte pota[] = {0x01,0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xc0};
19
20 byte values[11];
21 SoftwareSerial mod(2,3);
22
23
24 void setup() {
25   Serial.begin(9600);
26   mod.begin(9600);
27   pinMode(RE, OUTPUT);
```

```
28    pinMode(DE, OUTPUT);
29
30  }
31
32  void loop() {
33    byte val1,val2,val3;
34    val1 = nitrogen();
35    delay(250);
36    val2 = phosphorous();
37    delay(250);
38    val3 = potassium();
39    delay(250);
40
41    //Serial.print("Nitrogen: ");
42    Serial.print(val1);
43    Serial.print(",");
44    Serial.print(val2);
45    Serial.print(",");
46    Serial.print(val3);
47    Serial.print(",");
48    Serial.print((thermocouple.readCelsius())/2.021142857-18);
49    Serial.print('\n');
50
51   //Serial.print("Phosphorous: ");
52   //Serial.println(val2);
53    //Serial.println(" ppm");
54    //Serial.print("Potassium: ");
55    //Serial.println(val3);
56    //Serial.println(" meq/100g");
57    //Serial.println(thermocouple.readCelsius());
58    //Serial.println(" C ");
59    delay(1000); //3000
60
61  }
62
63  byte nitrogen(){
64    digitalWrite(DE,HIGH);
65    digitalWrite(RE,HIGH);
66    delay(10);
67    if(mod.write(nitro,sizeof(nitro))==8){
68      digitalWrite(DE,LOW);
69      digitalWrite(RE,LOW);
```

```
70      for(byte i=0;i<7;i++){
71      //Serial.print(mod.read(),HEX);
72      values[i] = (mod.read()/10000)*1.216;
73      //Serial.print(values[i],HEX);
74      }
75      //Serial.println();
76    }
77    return values[4];
78  }
79
80  byte phosphorous(){
81    digitalWrite(DE,HIGH);
82    digitalWrite(RE,HIGH);
83    delay(10);
84    if(mod.write(phos,sizeof(phos))==8){
85      digitalWrite(DE,LOW);
86      digitalWrite(RE,LOW);
87      for(byte i=0;i<7;i++){
88      //Serial.print(mod.read(),HEX);
89      values[i] = mod.read()*1.826;
90      //Serial.print(values[i],HEX);
91      }
92      //Serial.println();
93    }
94    return values[4];
95  }
96
97  byte potassium(){
98    digitalWrite(DE,HIGH);
99    digitalWrite(RE,HIGH);
100   delay(10);
101   if(mod.write(pota,sizeof(pota))==8){
102     digitalWrite(DE,LOW);
103     digitalWrite(RE,LOW);
104     for(byte i=0;i<7;i++){
105     //Serial.print(mod.read(),HEX);
106     values[i] = (mod.read()*391)*4.805;
107     //Serial.print(values[i],HEX);
108     }
109     //Serial.println();
110   }
111   return values[4];
```

```
}
```

- **Codigo implementado en el Módulo ESP32:**

```
//Codigo Dispositivo NPK Advisor
// Realizado por:
// Luiza Fernanda Narvaez Timan , Luis Fernando Cobo Mendez


#include <LiquidCrystal_I2C.h> // Libreria para usar LCD con modulo i2c
#include<Wire.h>
#include "FS.h"               // SD Card Library
#include "SD.h"               // SD Card Library
#include <SPI.h>              // SPI Interface
#include <WiFi.h>             // Librer a WiFi
#include <ArduinoJson.h>      // Librer a para convertir datos en objectos
    JSON
#include <HTTPClient.h>       // Libreria para conexi n con la plataforma
    de servicios
#include <Separador.h>        // Librer a para separar datos enviados
    desde el Arduino Nano
#define SD_CS 5               // Pin CD del modulo SD
#define RXp2 16               // Comunicaci n serial ESP32 - Arduino Nano
#define TXp2 17               // Comunicaci n serial ESP32 - Arduino Nano
#define btn1 15               // Pin Pulsador
#define btn2 34               // Pin Pulsados
#define redLed 12             // Pin Led Rojo
#define greenLed 14           // Pin Led Verde
#define DELAY_BTN 500         // Retardo bot n a 500ms


LiquidCrystal_I2C lcd(0x27,16,2);

Separador s;

const char* ssid = "UNE_HFC_51F0";
const char* password = "ACA5CFBE";
char jsonOutput[128];
long lastTime = 0;
```

```
unsigned long lcdTimer = 0;
unsigned long lcdInterval = 500;
String dataMessage;
String humedad;
String datosrecibidos;
String n;
String p;
String k;
String tem;
int btnPress = 0;
int nitro;
int fosf;
int pota;
int tempe;
int humity;


//Interrupci n
void ISR(){
  if (digitalRead(btn1)) btnPress = 1;
  else if (digitalRead(btn2)) btnPress = 2;
  else btnPress = 0;
}

void setup() {
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  WiFi.begin(ssid,password);
  Serial.print("Conectando...");
  delay(1000);
  //lcd.setCursor(0,0);
  pinMode(btn1, INPUT_PULLDOWN);          // INPUT_PULLDOWN/INPUT_PULLUP
  pinMode(btn2, INPUT_PULLDOWN);          // INPUT_PULLDOWN/INPUT_PULLUP
  pinMode(greenLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  attachInterrupt(btn1, ISR, RISING);   // LOW/HIGH/FALLING/RISING/
      CHANGE
  attachInterrupt(btn2, ISR, RISING);   // LOW/HIGH/FALLING/RISING/
      CHANGE
```

```
73    while(WiFi.status() != WL_CONNECTED){
74      delay(5000);
75      break;
76    }
77    if(WiFi.status() != WL_CONNECTED){
78      delay(500);
79      digitalWrite(redLed, HIGH);
80      digitalWrite(greenLed, LOW);
81      Serial.print("Sin conexi n a internet....");
82    }else{
83    //Serial.print("Conectado con  xito , mi IP es : ");
84    //Serial.println(WiFi.localIP());
85    digitalWrite(redLed, LOW);
86    digitalWrite(greenLed, HIGH);
87    }
88     // Inicializaci n SD CARD
89    SD.begin(SD_CS);
90    if(!SD.begin(SD_CS)) {
91      //Serial.println("Card Mount Failed");
92      return;
93    }
94    uint8_t cardType = SD.cardType();
95    if(cardType == CARD_NONE) {
96      //Serial.println("No SD card attached");
97      return;
98    }
99    Serial.println("Initializing SD card...");
100   if (!SD.begin(SD_CS)) {
101     //Serial.println("ERROR - SD card initialization failed!");
102     return;    // init failed
103   }
104
105   // If the data.txt file doesnt exist
106   // Create a file on the SD card and write the data labels
107   File file = SD.open("/data.txt");
108   if(!file) {
109     //Serial.println("File doens't exist");
110     //Serial.println("Creating file...");
111     writeFile(SD, "/data.txt","");
112   }
113   else {
114     //Serial.println("File already exists");
```

```
115         //Serial.println("Leyendo datos del archivo... ");
116         for(int i=0;i<file.size();i++) // Lee caracter a caracter.
117         {
118           //Serial.println(file.read());
119         }
120    }
121    file.close();
122 }
123 void loop() {
124    if(Serial2.available()){
125    humedad = analogRead(35);
126    humity = humedad.toInt();
127    Serial2Event();
128    }
129    if (millis() - lcdTimer >= lcdInterval)
130     {
131       lcd.setCursor(0,0);
132       lcd.print("N:"+n);
133       lcd.setCursor(5,0);
134       lcd.print("%");
135       lcd.setCursor(7,0);
136       lcd.print("P:"+p);
137       lcd.setCursor(12,0);
138       lcd.print("%");
139       lcd.setCursor(0,1);
140       lcd.print("K:"+k);
141       lcd.setCursor(5,1);
142       lcd.print("%");
143       lcd.setCursor(7,1);
144       lcd.print("T:"+tem);
145       lcd.setCursor(11,1);
146       lcd.print("\337C");
147       lcd.setCursor(12,1);
148       lcd.print("C");
149       delay(5000);
150       lcd.clear();
151       lcd.setCursor(0,0);
152       lcd.print("H:"+humedad);
153       delay(3000);
154       lcd.clear();
155 }
156    if (btnPress == 1)
```

```
157   {
158     if (millis()-lastTime > DELAY_BTN)
159     {
160       if(WiFi.status()== WL_CONNECTED){
161          //Serial.println("Estoy conectado a WiFi");
162          HTTPClient http;
163          const size_t CAPACITY = JSON_OBJECT_SIZE(5);
164          StaticJsonDocument<CAPACITY> doc;
165          JsonObject object = doc.to<JsonObject>();
166          object["Humedad"]= humity;
167          object["N"]= nitro;
168          object["P"]= fosf;
169          object["K"]= pota;
170          object["Temp"]=tempe;
171          serializeJson(doc, jsonOutput);
172          http.begin("http://f344-191-95-173-170.ngrok.io/index/items");
173          //http.addHeader("Content-Type","application/x-www-form-
                 urlencoded");
174          http.addHeader("Content-Type","application/json");
175
176          int codigo_respuesta = http.POST(String(jsonOutput));
177          if(codigo_respuesta == 200){
178             String cuerpo_respuesta = http.getString();
179             //Serial.println("El servidor respondi ");
180             //Serial.println(cuerpo_respuesta);
181             lcd.clear();
182             lcd.setCursor(0,0);
183             lcd.print("Datos Subidos");
184             delay(3000);
185             lcd.clear();
186        }else{
187             //Serial.println("Error en la conexi n WiFi");
188             }
189        delay(2000);
190        }
191       lastTime = millis();
192     }
193    btnPress = 0;
194   }
195
196  if (btnPress == 2)
197  {
```

```
198     if (millis()-lastTime > DELAY_BTN)
199     {
200       logSDCard();
201       lcd.clear();
202       lcd.setCursor(0,0);
203       lcd.print("DATOS GUARDADOS");
204       delay(3000);
205       lcd.clear();
206     }
207    lastTime = millis();
208     }
209     btnPress = 0;
210 }
211
212 void Serial2Event(){
213
214    datosrecibidos = Serial2.readStringUntil('\n');
215
216    n = s.separa(datosrecibidos,',',0);
217    p = s.separa(datosrecibidos,',',1);
218    k = s.separa(datosrecibidos,',',2);
219    tem = s.separa(datosrecibidos,',',3);
220    nitro = n.toInt();
221    fosf = p.toInt();
222    pota = k.toInt();
223    tempe = tem.toInt();
224
225 }
226
227 // Write the sensor readings on the SD card
228 void logSDCard() {
229
230   dataMessage = String(humedad)+","+String(datosrecibidos) + "\r\n";
231   //Serial.println(dataMessage);
232   appendFile(SD, "/data.txt", dataMessage.c_str());
233 }
234
235 // Write to the SD card (DONT MODIFY THIS FUNCTION)
236 void writeFile(fs::FS &fs, const char * path, const char * message) {
237   //Serial.printf("Writing file: %s\n", path);
238
239   File file = fs.open(path, FILE_WRITE);
```

```
240   if(!file) {
241     //Serial.println("Failed to open file for writing");
242     return;
243   }
244   if(file.print(message)) {
245     //Serial.println("File written");
246   } else {
247     //Serial.println("Write failed");
248   }
249   file.close();
250 }
251
252 // Append data to the SD card (DONT MODIFY THIS FUNCTION)
253 void appendFile(fs::FS &fs, const char * path, const char * message) {
254   Serial.printf("Appending to file: %s\n", path);
255
256   File file = fs.open(path, FILE_APPEND);
257   if(!file) {
258     //Serial.println("Failed to open file for appending");
259     return;
260   }
261   if(file.print(message)) {
262     //Serial.println("Message appended");
263   } else {
264     //Serial.println("Append failed");
265   }
266   file.close();
267 }
```

B. Manual de usuario

En esta sección se incluye instrucciones detalladas de cómo hacer un uso correcto del sistema expuesto en el proyecto.

- **Logín y registro de usuario**

  En esta sección el usuario puede iniciar sesión, autenticando las credenciales una cuenta previamente creada, o en su defecto, registrar una nueva. En la sección de registro, el usuario debe completar todos los campos de texto y presionar el botón guardar.

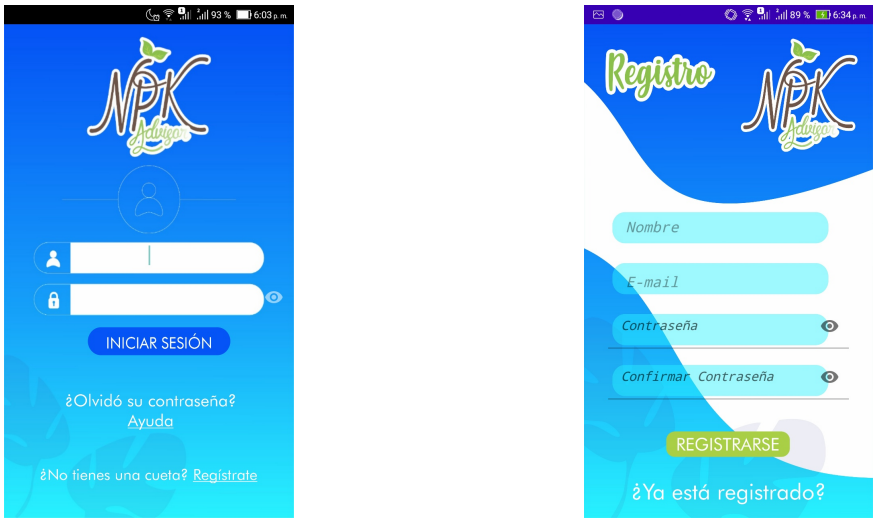


Figura 4.1: Registro y login de usuario

- **Screen Home**

  Una vez iniciada la sesión, el usuario puede desplegar un menú que le permitirá hacer uso de diferentes diferentes opciones, tales como: gestión de cultivo, registro de información de cultivo, mapas y grafícas.

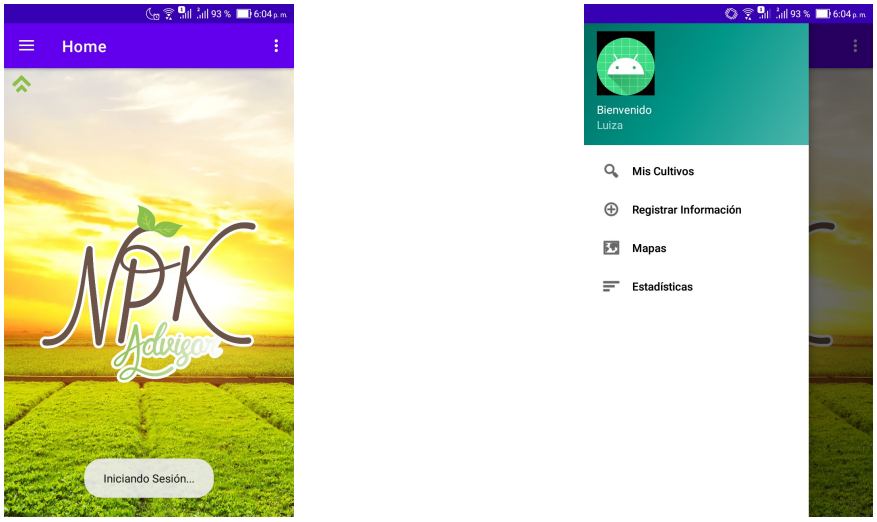


Figura 4.2: Screen Home

- **Gestión de cultivos**
  En esta sección el usuario tendra la posibilidad de agregar un nuevo cultivo, registrando el nombre del mismo y el área, del mismo modo podrá modificarlo, eliminarlo, o visualizar sus respectivos datos y descargalos en un archivo .CSV

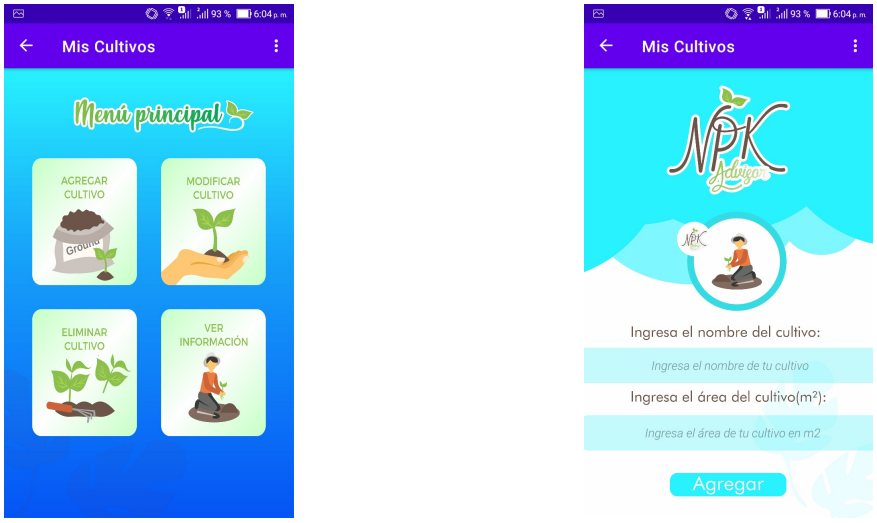


Figura 4.3: Menú para la gestión de cultivos y agregar un nuevo cultivo

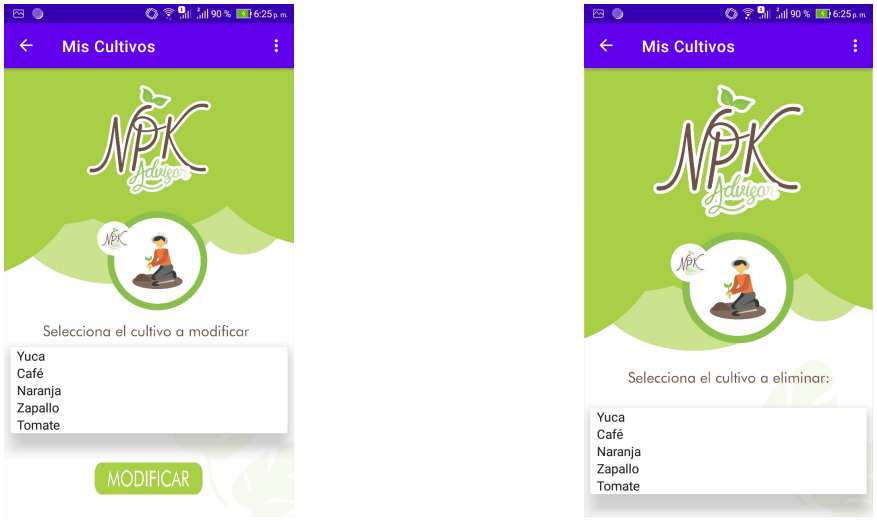


Figura 4.4: Eliminar y modificar un cultivo

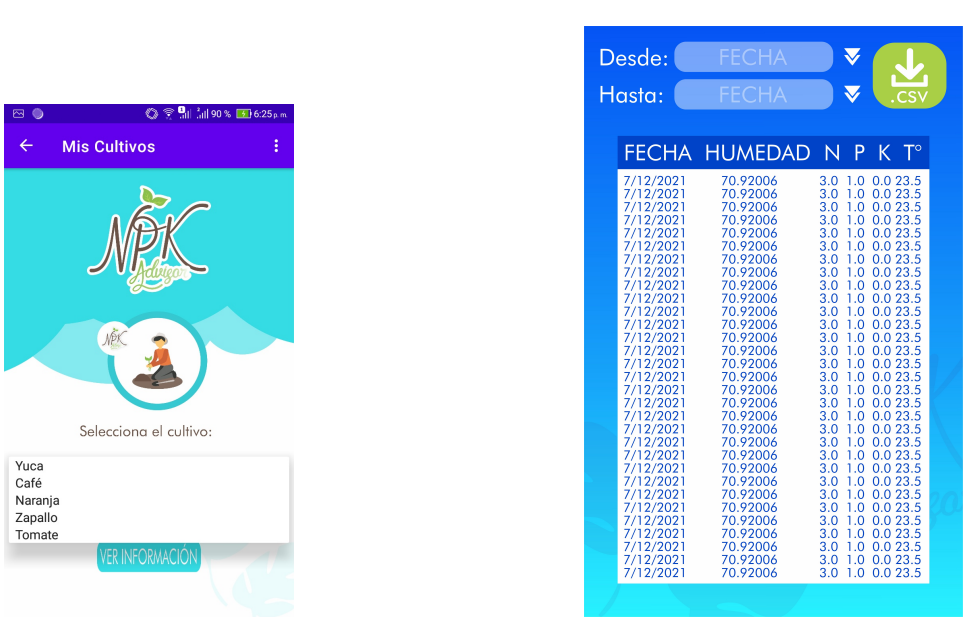


Figura 4.5: Visualizar datos del cultivo

- **Registrar información de cultivo**
  Al seleccionar esta opción se mostrarán las instrucciones para tomar los datos por medio del sistema hardware. A continuación se hace un breve resumen del uso de hardware

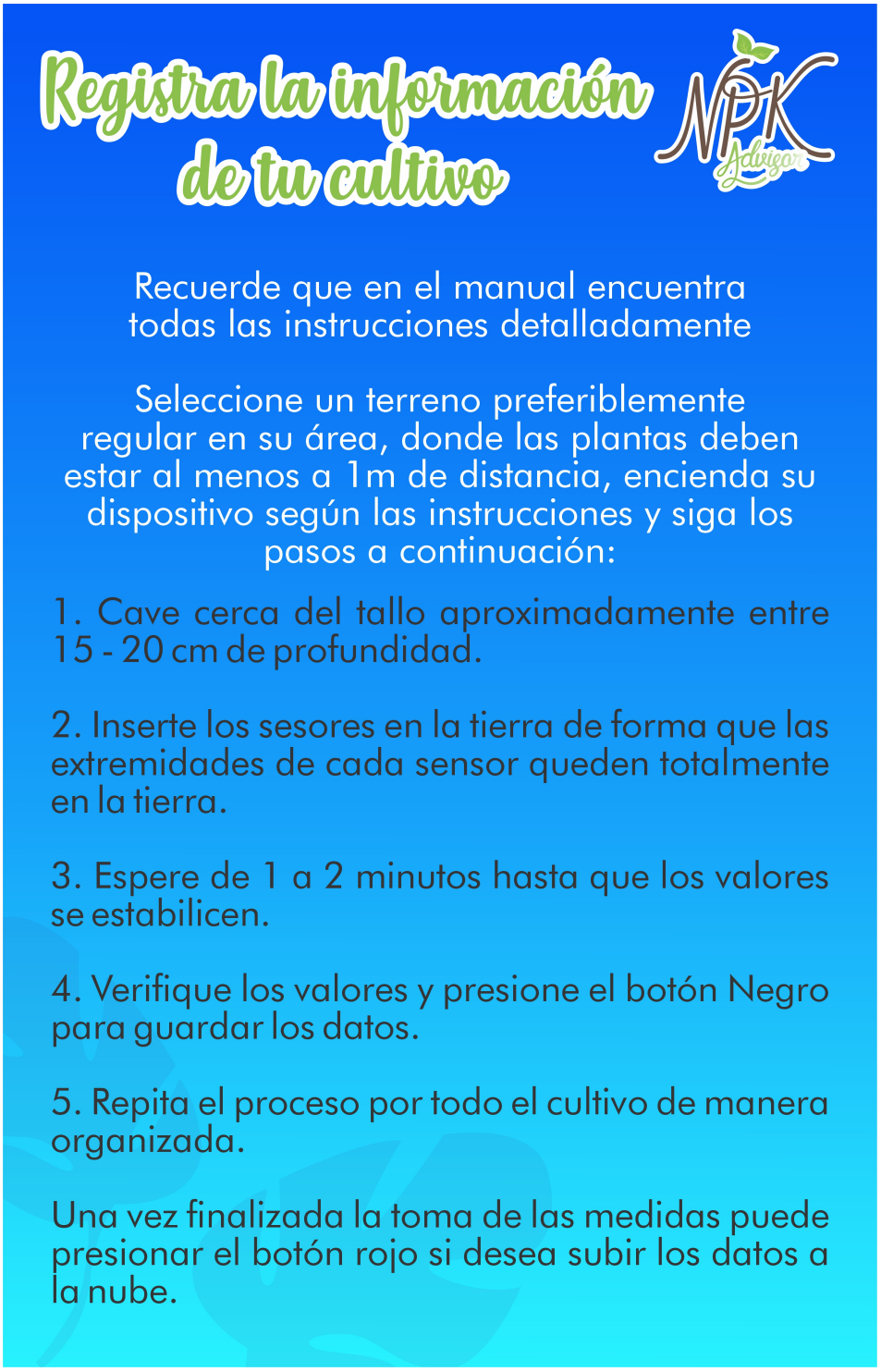


Figura 4.6: Instrucciones de uso hardware

- **Mapas y graficas**
  En la sección del mapa se tomarán los datos de longitud y latitud del usuario y se desplegarán en un mapa de Google maps que mostrará la ubicación del dispositivo. Por otro lado, para la generación de las gráficas el usuario debe ingresar una fecha de inicio y una fecha final, con el objetivo de obtener todos los valores en diferentes tipos de gráficas.

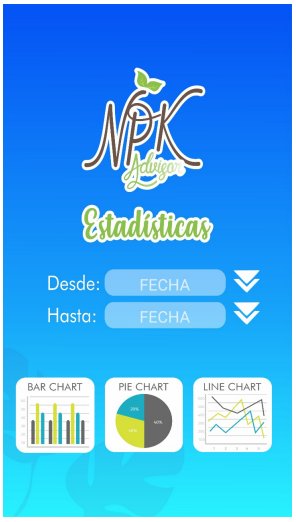
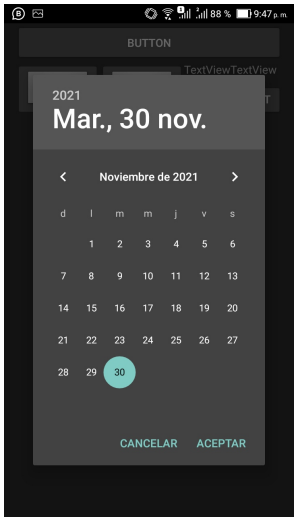


Figura 4.7: Mapas y gráficas



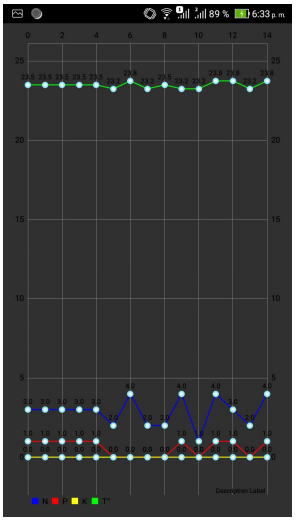


Figura 4.8: Fechas y gráfica

C. Comparación de costos del dispositivo con respecto a pruebas de laboratorio.

Comparación de costos del dispositivo con respecto a pruebas de laboratorio		
Método de medición	**Prueba de laboratorio**	**NPK Advisor**
Costo	$100.000 por muestra	$600.000