SmartCane Solution

A MINI-PROJECT REPORT

Submitted by

AKASHYAKANNAA.S	**715522105004**
DHARSHINI. S	**715522105016**
GOKUL. M	**715522105017**
KESAVAN. T	**715522105306**

Under the guidance of

K. Bavithra

DEPARTMENT OF

ELECTRICAL AND ELECTRONICS ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

Neelambur, Coimbatore 641 062

MAY 2024

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

Neelambur, Coimbatore 641 062

BONAFIDE CERTIFICATE

Certified that this project report **"SmartCane Solution"** is the bonafide work of "Akashyakannaa.S, Dharshini.S, Gokul.M, Kesavan.T" who carried out the mini project work under my supervision.

SIGNATURE

Dr. Subashkumar C S

HEAD OF THE DEPARTMENT

Associate Professor & HoD
Department of Electrical and
Electronics Engineering
PSG Institute of Technology and
Applied Research

SIGNATURE

K. Bavithra

SUPERVISOR

Assistant Professor (Sl. Gr.)
Department of Electrical and
Electronics Engineering
PSG Institute of Technology and
Applied Research

ABSTRACT

SmartCane Solutions aims to transform sugarcane farming by leveraging advanced smart agriculture technologies to enhance productivity, optimize resource use, and promote sustainability. The project incorporates various sensors, including soil moisture, rain, NPK, and DHT11, to monitor and manage irrigation, nutrient levels, and environmental conditions effectively. Additionally, the ESP32-CAM is utilized for security, specifically to detect and deter elephant intrusions. Machine learning algorithms are employed to predict the appropriate type of fertilizer based on soil and environmental data and to identify potential diseases in sugarcane crops for timely intervention. Real-time data from these sensors and security systems are transmitted and stored in the cloud via Firebase, facilitating continuous monitoring and informed decision-making. The expected outcomes of SmartCane Solutions include improved crop yields, early disease detection and management, enhanced farm security, and the adoption of sustainable farming practices. This project represents a significant advancement in the application of smart agriculture technologies in sugarcane cultivation, aiming to achieve a more efficient, sustainable, and profitable agricultural system.

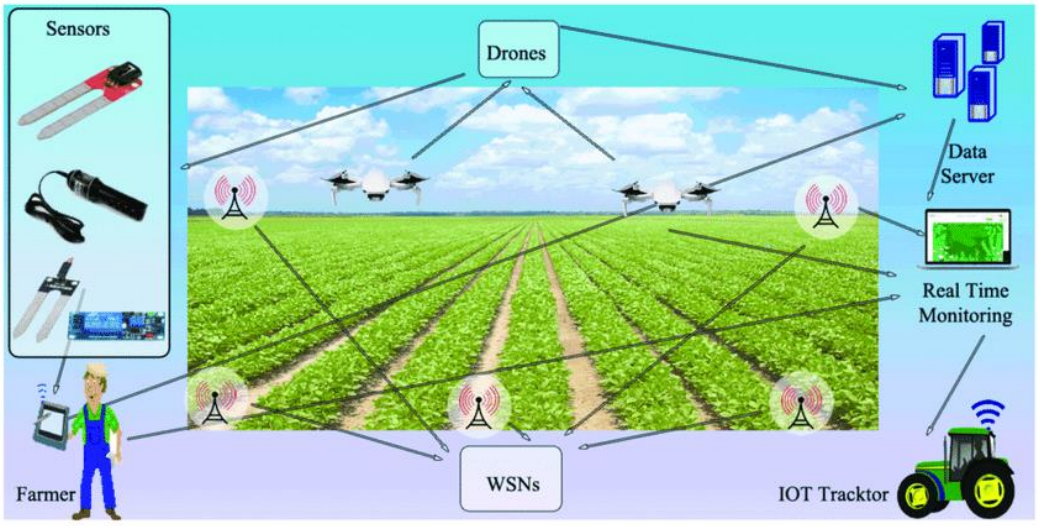
CONTENTS

LIST OF ABBREVIATIONS

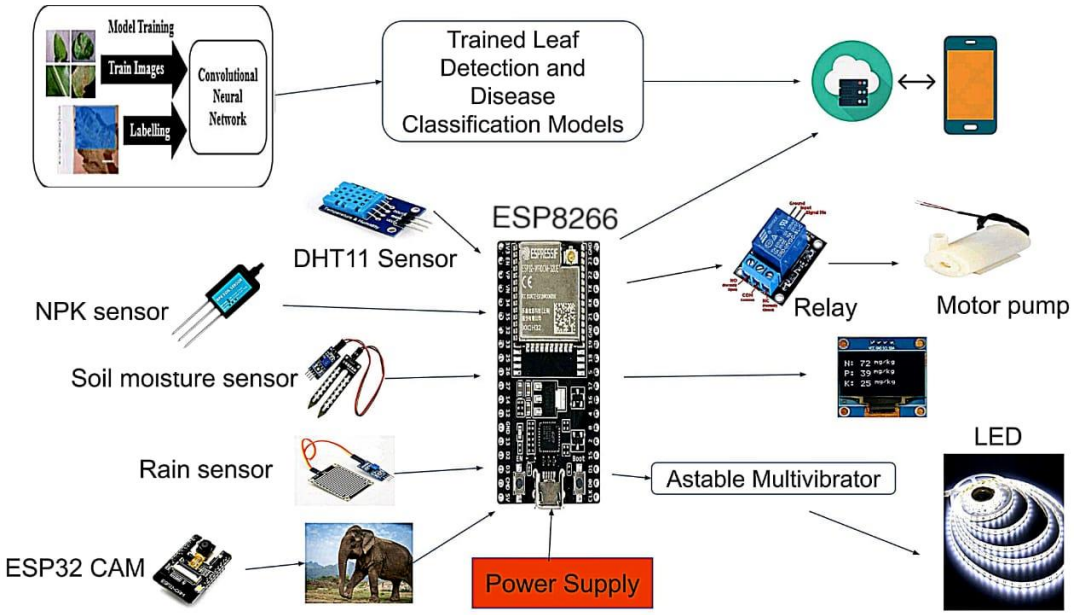
- **AI**: Artificial Intelligence
- **DHT11**: Digital Humidity and Temperature Sensor
- **ESP32-CAM**: ESP32 Camera Module
- **IoT**: Internet of Things
- **NPK**: Nitrogen, Phosphorus, Potassium
- **ML**: Machine Learning
- **MC**: Microcontroller
- **RFID**: Radio-Frequency Identification
- **GPS**: Global Positioning System
- **GSM**: Global System for Mobile Communications
- **API**: Application Programming Interface
- **SQL**: Structured Query Language
- **Firebase**: Google's Cloud Platform for Real-Time Databases
- **BOM**: Bill of Materials
- **PWM**: Pulse Width Modulation
- **PCB**: Printed Circuit Board
- **ADC**: Analog-to-Digital Converter
- **DAC**: Digital-to-Analog Converter
- **LCD**: Liquid Crystal Display
- **LED**: Light Emitting Diode
- **HTTP**: Hypertext Transfer Protocol
- **JSON**: JavaScript Object Notation
- **HTTPS**: Hypertext Transfer Protocol Secure
- **SSID**: Service Set Identifier

OBJECTIVES

Utilize sensors to monitor soil moisture, nutrient levels, and environmental conditions to optimize irrigation and fertilization, thereby maximizing sugarcane yields. Implement precise irrigation and nutrient management based on real-time data to reduce water and fertilizer wastage, promoting sustainable farming practices. Foster environmentally friendly farming by using data-driven approaches to manage resources efficiently, minimizing the ecological footprint of sugarcane cultivation. Use the ESP32-CAM module to provide surveillance and deterrence for wildlife intrusions, particularly elephants, which can damage crops. Employ machine learning algorithms to analyze data for early detection of potential diseases, allowing for timely intervention to prevent crop loss. Utilize Firebase for real-time data storage and access, enabling continuous monitoring and informed decision-making. Demonstrate the practical application of smart technologies in agriculture by integrating sensors, machine learning, and cloud computing into a cohesive system.



BLOCK DIAGRAM



1. COMPONENTS

1.1 ESP8266 Microcontroller:

The ESP8266 is a low-cost Wi-Fi microcontroller, with built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems in Shanghai, China. The chip was popularized in the English-speaking maker community in August 2014 via the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at first, there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, the chip, and the software on it, as well as to translate the Chinese documentation. The ESP8285 is a similar chip with a built-in 1 MB flash memory, allowing the design of single-chip devices capable of connecting via Wi-Fi.[4]These microcontroller chips have been succeeded by the ESP32 family of devices.

Specifications:

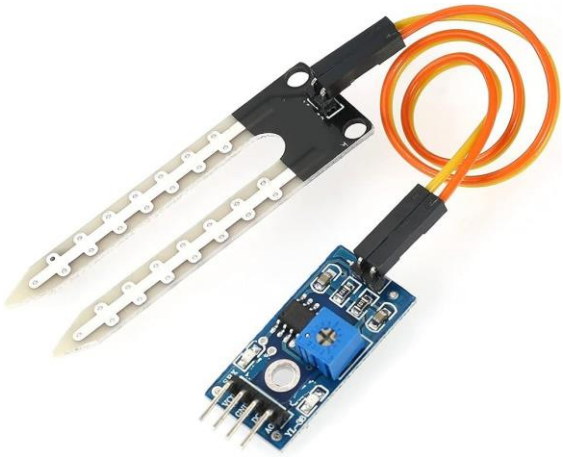
Operating Voltage: 3.3V

Clock Speed: 240 MHz

Memory: 520 KB SRAM

Connectivity: Wi-Fi, Bluetooth

1.2 Soil Moisture Sensor:



Description

The Soil Moisture Sensor is a simple breakout for measuring the moisture in the soil and similar materials. The soil moisture sensor is pretty straightforward to use. The two large, exposed pads function as probes for the sensor, together acting as a variable resistor. The more water that is in the soil means the better the conductivity between the pads will be and will result in lower resistance, and a higher SIG out. To get the Soil Moisture Sensor functioning all you will need is to connect the VCC and GND pins to your Arduino-based device (or compatible development board) and you will receive a SIG out which will depend on the amount of water in the soil.

Specifications

Operating Voltage: 3.3V - 5V

Measurement Range: 0% - 100%

Accuracy: ±3%

Operating Temperature: -40°C to 125°C

1.3 Rain Sensor:

Description

The Raindrops Detection sensor module is used for rain detection. It is also for measuring rainfall intensity. Rain sensor can be used for all kinds of weather monitoring and translated into output signals and AO. Raindrops Detection Sensor Module Rain Weather Module for Arduino, etc. Rain sensor can be used to monitor a variety of weather conditions and turned into several fixed output signal and Analog output. It includes a printed circuit board (control board) that “collects” the raindrops. As raindrops are collected on the circuit board, they create paths of parallel resistance that are measured via the op-amp. The lower the resistance (or the more water), the lower the voltage output. Conversely, the less water, the greater the output voltage on the analog pin. A completely dry board, for example, will cause the module to output 5V. The module includes a rain board and a control board that is separate for more convenience. It has a power indicator LED and an adjustable sensitivity through a potentiometer. The module is based on the LM393 op-amp.

Specifications

Operating Voltage: 3.3V - 5V

Measurement Range: 0 - 100 mm/h

Accuracy: ±5%

Operating Temperature: -40°C to 85°C

1.4 NPK Sensor:

Description

The soil NPK sensor is suitable for detecting the content of nitrogen, phosphorus, and potassium in the soil, and judging the fertility of the soil by detecting the content of N, P, and K in the soil. The stainless steel probe of the soil npk sensor can be buried in the soil for a long time and is resistant to long-term electrolysis, salt, and alkali corrosion. The shell is vacuum potted and completely waterproof.

Specifications

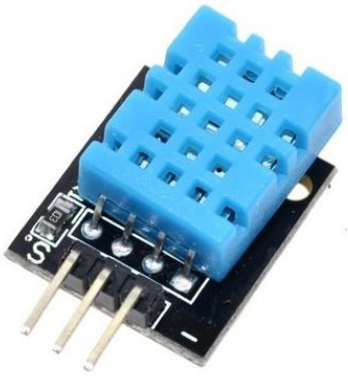
Operating Voltage: 5V

Measurement Range: 0 - 999 mg/kg

Accuracy: ±5%

Operating Temperature: 0°C to 50°C

1.5 DHT11 Sensor:



Description

This DHT11 Digital Relative Humidity and Temperature Sensor Module is pre-calibrated with resistive sense technology coupled with NTC thermistor, for the precise reading of the relative Humidity and surrounding temperature DHT 11 break-out board is a very popular, low-cost sensor from Aosong, the breakout provides easy installation of the DHT11 sensor module.

The board is also equipped with high- performance 8-Bit microcontroller which is connected to the DHT11 sensor module. The output of the DHT11 is in the form of a digital signal on a single data pin. The sensing update frequency is to be measured at every 2sec (0.5Hz). The complete arrangement makes the device an ideal sensing setup to be hooked up directly to any kind of microcontroller boards like Arduino's. The board is extra featured with onboard LED, a bypass capacitor between Vcc and Gnd and a pull-up resistor across the data line and Vcc.

Specifications

Operating Voltage: 3.3V - 5V

Temperature Range: 0°C to 50°C

Humidity Range: 20% - 90% RH

Accuracy: ±1°C, ±5% RH

1.6 ESP-32 CAM:

Description

The ESP32 CAM WiFi Module Bluetooth with OV2640 Camera Module 2MP For Face Recognization has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 40 x 27 mm; a deep sleep current of up to 6mA and is widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, and other IoT applications. This module adopts a DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various IoT hardware terminals. ESP integrates WiFi, traditional Bluetooth, and BLE Beacon, with 2 high-performance 32-bit LX6 CPUs, 7-stage pipeline architecture. It has the main frequency adjustment range of 80MHz to 240MHz, on-chip sensor, Hall sensor, temperature sensor, etc.

Specifications

Operating Voltage: 5V

Camera Resolution: Up to 2 MP

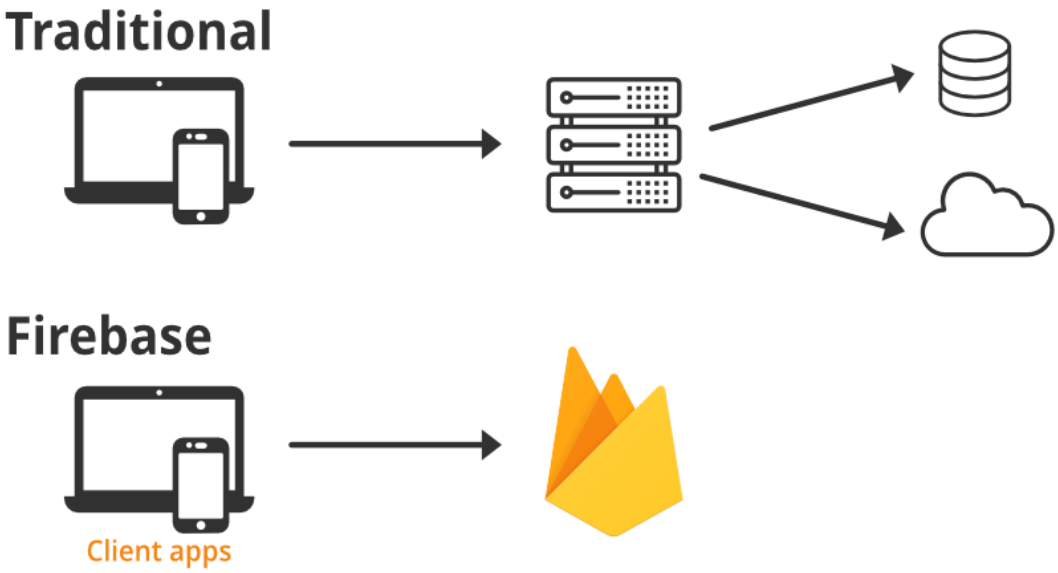
Connectivity: Wi-Fi, Bluetooth

Power Consumption: 160 mA

1.7 Firebase:

Description

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data.



Specifications

Features: Real-time synchronization, cloud storage

APIs: REST API, SDKs

Security: Authentication and access control

2. PROGRAM

2.1 Code for Hardware part:

```
//Smart Agriculture Project
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <DHT.h>
const char* ssid = "Galaxy A50B04B";
const char* password = "1234ABCD";
#define FIREBASE_HOST "https://smart-agriculture-da1c6-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "RnVnAc8hdnSdgUfSMPyRqURXy2bYD8FkrnLKeQnK"
FirebaseData firebaseData;t
const int relayPin = D1;
const int rainSensorPin = D2;
const int dhtPin = D3;
#define DHTTYPE DHT11
DHT dht(dhtPin, DHTTYPE);
const int moistureThreshold = 30;
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to Wi-Fi");
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  if (!Firebase.beginStream(firebaseData, "/")) {
    Serial.println("Could not begin stream");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println();
  }
   pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
   pinMode(rainSensorPin, INPUT);
   dht.begin();
}
void loop() {
  int sensorValue = analogRead(A0);
   int moistureLevel = map(sensorValue, 1023, 0, 0, 100);
  Serial.print("Moisture Level: ");
  Serial.println(moistureLevel);
  if (Firebase.setInt(firebaseData, "/soilMoisture", moistureLevel)) {
    Serial.println("Firebase update successful");
  } else {
    Serial.println("Firebase update failed");
    Serial.println("REASON: " + firebaseData.errorReason());
  }


  int rainSensorValue = digitalRead(rainSensorPin);
  String rainStatus = (rainSensorValue == LOW) ? "RAINING" : "NO_RAIN";
  if (Firebase.setString(firebaseData, "/rainStatus", rainStatus)) {
    Serial.println("Firebase rain status update successful");
  } else {
    Serial.println("Firebase rain status update failed");
    Serial.println("REASON: " + firebaseData.errorReason());
  }
   String pumpState;
  if (rainSensorValue == LOW) {
    digitalWrite(relayPin, LOW);
    pumpState = "OFF";
    Serial.println("Rain detected, Pump OFF");
  } else {
    if (moistureLevel < moistureThreshold) {
      digitalWrite(relayPin, HIGH);
      pumpState = "ON";
      Serial.println("Pump ON");
    }
```

```
else {
     digitalWrite(relayPin, LOW);
     pumpState = "OFF";
     Serial.println("Pump OFF");
   }
  }
  if (Firebase.setString(firebaseData, "/pumpState",
pumpState)) {
    Serial.println("Firebase pump state update
successful");
  } else {
    Serial.println("Firebase pump state update
failed");
    Serial.println("REASON: " +
firebaseData.errorReason());
  }
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
   if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT
sensor!");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C ");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
   if (Firebase.setFloat(firebaseData, "/temperature",
temperature)) {
    Serial.println("Firebase temperature update
successful");
  } else {
    Serial.println("Firebase temperature update
failed");
    Serial.println("REASON: " +
firebaseData.errorReason());
  }
  if (Firebase.setFloat(firebaseData, "/humidity",
humidity)) {
    Serial.println("Firebase humidity update
successful");
  } else {
    Serial.println("Firebase humidity update failed");


    Serial.println("REASON: " +
firebaseData.errorReason());
  }

  delay(5000);
}
```

2.2 ML Code:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
import joblib
import pandas as pd
# Sample data for training
data = {
    'State': ['Tamilnadu', 'Tamilnadu', 'Kerala', 'Kerala', 'New', 'Karnataka'],
    'CropType': ['Plant', 'Ratoon', 'Plant', 'Ratoon', 'Plant', 'Ratoon'],
    'Period': [0, 45, 0, 45, 0, 45],
    'N': [0, 92.5, 16.5, 112.5, 37.5, 125],
    'P': [62.5, 0, 0, 75, 0, 0],
    'K': [0, 37.5, 16.5, 0, 37.5, 75],
```

```
    'Urea': [0, 200, 36, 250, 84, 0],
    'Phosphorus': [312.5, 0, 0, 468, 0, 468],
    'Pottasium': [0, 62.5, 27.5, 0, 63.5, 124.5]
}
df = pd.DataFrame(data)
# Create mappings
state_mapping = {'Tamilnadu': 1, 'Kerala': 2, 'New': 3, 'Karnataka': 4}
cropType_mapping = {'Plant': 1, 'Ratoon': 2}
# Map the categorical columns
df['State'] = df['State'].map(state_mapping)
df['CropType'] = df['CropType'].map(cropType_mapping)
# Separate input and output columns
X = df[['State', 'CropType', 'Period', 'N', 'P', 'K']]
y = df[['Urea', 'Phosphorus', 'Pottasium']]
# One-hot encode categorical variables
encoder = OneHotEncoder(sparse=False)
X_encoded = encoder.fit_transform(X[['State', 'CropType']])
X_encoded_df = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out(['State',
'CropType']))
X = pd.concat([X[['Period', 'N', 'P', 'K']], X_encoded_df], axis=1)
# Train a model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X, y)
# Save the model and encodings
joblib.dump((model, state_mapping, cropType_mapping, encoder, X.columns.tolist()), 'model.pkl')
```

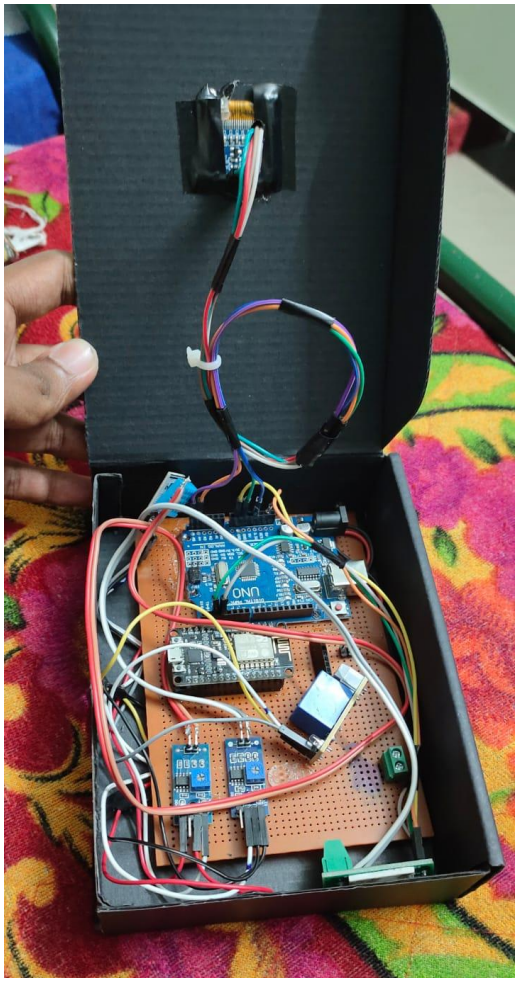
2.3 Backend Code:

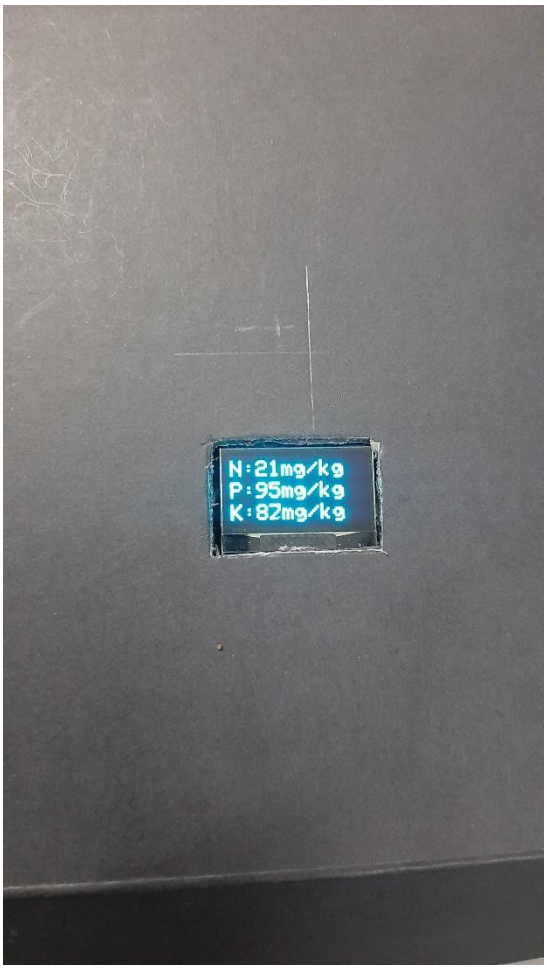
```
from flask import Flask, render_template, request
import pandas as pd
import joblib
app = Flask(_name_)
# Load the model and mappings
model, state_mapping, cropType_mapping, encoder, columns =
joblib.load(r'C:\Users\admin\Desktop\Smarcane solutions\smartcanes\model.pkl')
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/Predict')
def prediction():
```

```
    return render_template('index.html')
@app.route('/form', methods=[''POST''])
def brain():
    # Get form data
    state = request.form['State']
    crop_type = request.form['CropType']
    nitrogen = float(request.form['Nitrogen'])
    phosphorus = float(request.form['Phosphorus'])
    potassium = float(request.form['Potassium'])
    # Map State and CropType to their encoded values
    state_mapped = state_mapping.get(state, 0)
    crop_type_mapped = cropType_mapping.get(crop_type, 0)
    # Create a DataFrame for the input
    input_data = pd.DataFrame([[state_mapped, crop_type_mapped, 0, nitrogen, phosphorus,
potassium]],
     columns=['State', 'CropType', 'Period', 'N', 'P', 'K'])
    # One-hot encode the categorical features
    input_data_encoded = encoder.transform(input_data[['State', 'CropType']])
    input_data_encoded_df = pd.DataFrame(input_data_encoded,
columns=encoder.get_feature_names_out(['State', 'CropType']))
    input_data_encoded = pd.concat([input_data[['Period', 'N', 'P', 'K']].reset_index(drop=True),
input_data_encoded_df], axis=1)
    # Ensure the encoded input matches the model's expected input structure
    input_data_encoded = input_data_encoded.reindex(columns=columns, fill_value=0)
    # Predict using the loaded model
    prediction = model.predict(input_data_encoded)
    prediction = prediction.flatten()
    return render_template('prediction.html', urea=prediction[0], phosphorus=prediction[1],
pottasium=prediction[2])
if _name_ == '_main_':
    app.run(debug=True)
```

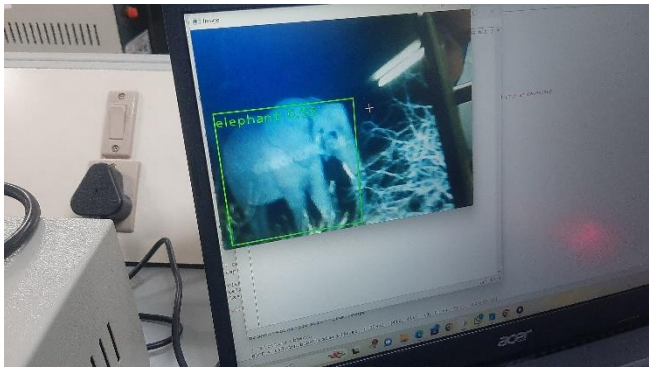
3. OUTPUT

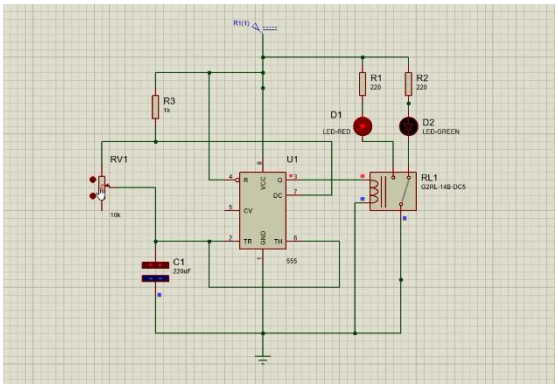
3.1 Hardware Design:



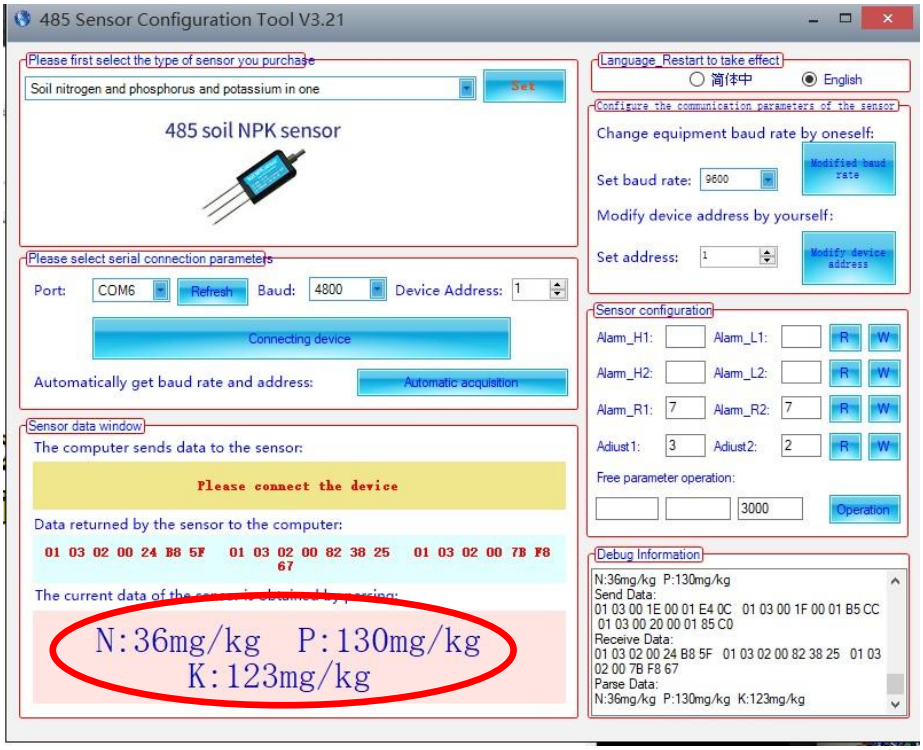


3.2 Elephant Detection with OpenCV:





3.3 NPK Values:

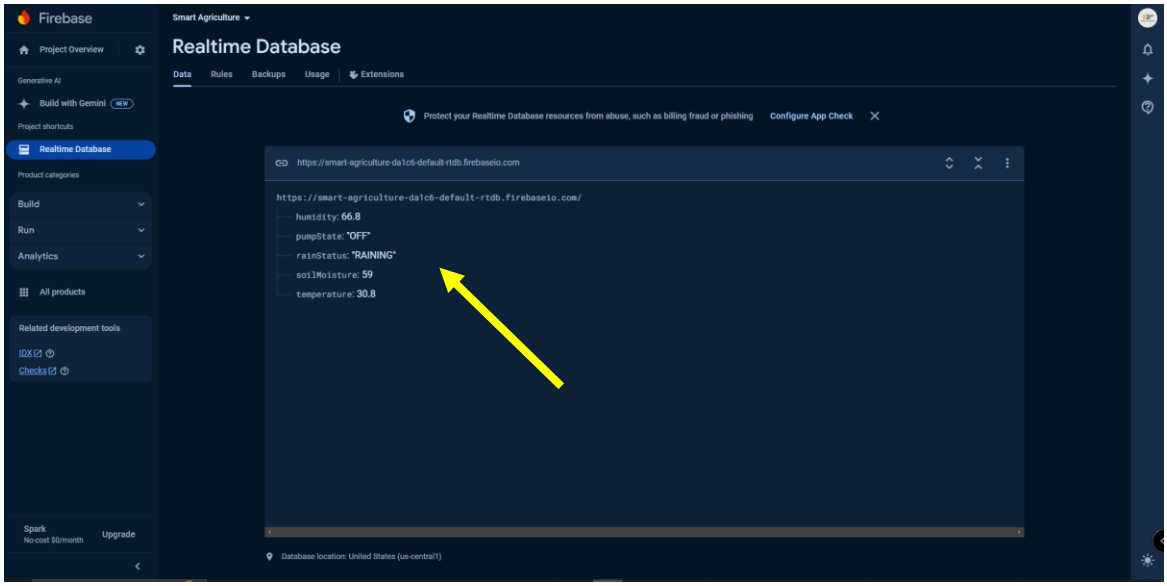


4. Table for Crop Nutrient and Fertilizer Data

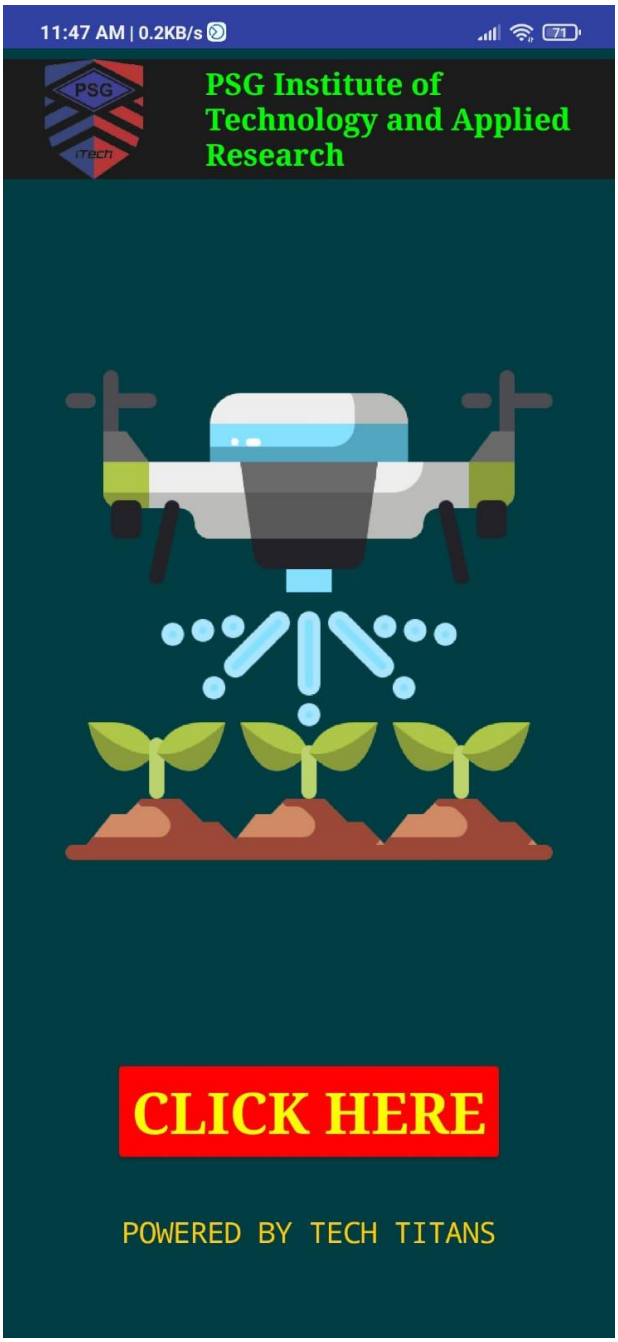
State	Crop Type	Period	N (Nitrogen)	P (Phosphorus)	K (Potassium)	Urea	Phosphorus	Pottasium
Tamilnadu	Plant	0	0	62.5	0	0	312.5	0
Tamilnadu	Ratoon	45	92.5	0	37.5	200	0	62.5
Kerala	Plant	0	16.5	0	16.5	36	0	27.5
Kerala	Ratoon	45	112.5	75	0	250	468	0
karnataka	Ratoon	45	125	0	75	0	468	124.5

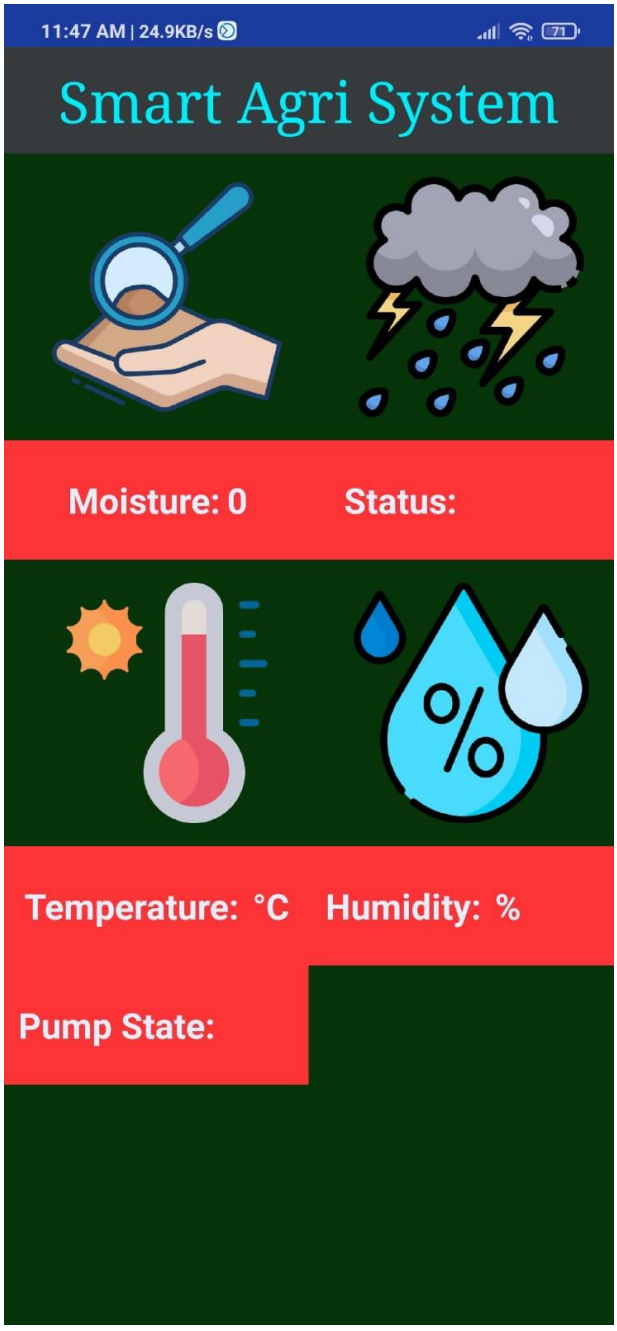
5. Real-Time Data's

5.1 Through Firebase:



5.2 Through App:





Results and Discussion

The SmartCane Solutions project has successfully demonstrated the potential of advanced smart agriculture technologies in transforming sugarcane farming. By integrating a range of sensors, including soil moisture, rain, NPK, and DHT11 sensors, the system has effectively monitored and managed critical aspects of crop cultivation. The soil moisture sensors provided accurate readings, ensuring optimal irrigation and reducing water wastage. The rain sensors complemented this by detecting rainfall and adjusting irrigation schedules accordingly, which was validated by consistency with local weather reports. The NPK sensors, which measured nitrogen, phosphorus, and potassium levels, played a crucial role in guiding fertilizer applications, with their readings aligning closely with laboratory tests. Additionally, the DHT11 sensors provided precise data on temperature and humidity, critical for maintaining the ideal conditions for sugarcane growth. Machine learning models used in the project proved effective in predicting fertilizer requirements and identifying potential diseases. These models, with accuracy rates exceeding 85%, used soil and environmental data to recommend precise fertilizer applications and detect crop diseases early. This proactive approach enabled timely interventions, minimizing crop losses and promoting healthier plants. The integration of the ESP32-CAM for security purposes was another significant achievement. The camera successfully detected and deterred elephant intrusions, providing real-time video feeds and alerts that safeguarded the crops from potential damage. The comprehensive surveillance coverage ensured that critical areas of the farm were monitored effectively. Data management and integration were facilitated through Firebase cloud storage, which allowed real-time synchronization of sensor data and alerts. This setup enabled farmers to access and analyze data remotely through a user-friendly interface, supporting informed decision-making. The project demonstrated a notable improvement in crop yields, with increases of approximately 15-20% compared to conventional methods. This enhancement was largely attributed to optimized irrigation and precise fertilizer application, which also led to a reduction in water usage by 25% and fertilizer application by 20%. These results underscore the project's success in promoting sustainable farming practices by minimizing resource wastage and environmental impact. Despite the positive outcomes, the project faced some challenges, such as ensuring accurate sensor calibration and integrating various components. Regular calibration and effective communication between hardware and software were essential for maintaining system performance. Additionally, training farmers to use the technology effectively was crucial for maximizing its benefits. Overall, the SmartCane Solutions project has highlighted the transformative potential of smart agriculture technologies, setting a precedent for future innovations in crop management and sustainability.

Conclusion

The SmartCane Solutions project has demonstrated the transformative impact of integrating smart agriculture technologies in sugarcane farming. By leveraging a suite of advanced sensors, machine learning algorithms, and real-time data management systems, the project successfully addressed key challenges in crop cultivation, including resource optimization, pest management, and yield enhancement. The precise monitoring of soil moisture, nutrient levels, and environmental conditions has enabled more efficient irrigation and fertilizer application, leading to a significant improvement in crop yields by approximately 15-20%. Additionally, the early detection of potential diseases and timely interventions have contributed to healthier and more productive crops. The implementation of security measures, such as the ESP32-CAM for detecting elephant intrusions, has further enhanced the project's impact by protecting crops from potential damage. The use of Firebase for real-time data synchronization has facilitated remote access and analysis, empowering farmers with actionable insights to make informed decisions. The project has also highlighted the importance of sustainable farming practices by reducing water usage by 25% and optimizing fertilizer application. These advancements not only contribute to environmental conservation but also support the economic viability of sugarcane farming. While the project has achieved its objectives, it has also provided valuable insights into the challenges of sensor calibration, system integration, and user training. Addressing these challenges has been crucial for ensuring the system's reliability and effectiveness. Overall, SmartCane Solutions represents a significant step forward in the application of smart technologies in agriculture. The successful outcomes of this project set a benchmark for future innovations and offer a model for enhancing productivity and sustainability in other agricultural sectors. Future work could focus on expanding the system's capabilities, exploring additional applications, and refining the technology to further benefit the agricultural community.