Embedded Edge ML for Fire Prediction

Using Arduino Nano 33 BLE Sense Rev-2

CONTENT

GROUP MEMBERS

1)SWAGATIKA SAHOO(220301130013)

2)SHIVANI BHARTI(220301130006)

3) AMBAR BARIK(220301130012)

4)SURYA PRATAP SARANGI(220301130011)

5)PRAKASH PANDEY(220301130004)

INTRODUCTION

Welcome to the presentation on "*fire prediction system*"

In this presentation, we will explore how the combination of Arduino Nano BLE Sense and advanced machine learning techniques can revolutionize fire prediction. By leveraging real- time sensor data and powerful algorithms, we can significantly improve the accuracy of fire prediction models.

ABOUT ARDUINO NANO BLE BOARD

The Arduino Nano BLE Sense is a powerful development board equipped with various sensors including temperature, humidity, gas, and motion sensors. These sensors provide real-time data that can be used to detect and predict fire incidents. With its compact size and wireless capabilities, the Arduino Nano BLE Sense is an ideal platform for fire prediction applications

THE NEED FOR ACCURATE FIRE PREDICTION

Accurate fire prediction is crucial for early detection and efficient response. Traditional methods often fall short due to limited data and outdated techniques. By incorporating Arduino Nano BLE Sense and advanced machine learning algorithms, we can overcome these limitations and achieve higher accuracy in fire prediction. This can save lives, reduce property damage, and enable proactive fire management strategies.

BACKGROUND STUDY

HARDWARE USED	IDE USED	CLOUD	MODEL	SENSORS	ACCURACY	DATASET
Arduino nano 33ble sense	Edge Impulse Studio	Not used	Tinyml	Microphone	97%	From Online
Bio Medical circuit	Edge Impulse	Not used	CNN	Microphone	95.1%-99.5%	Created
Not used	Edge Impulse	AWS	DNN & HMM	Laptop Microphone	82%-88%	Created
Arduino nano 33ble sense	Edge Impulse	Not used	Fuzzy classification	Microphone & Proximity sensor	90%	From Online
Not used	Kaldi open source	Not used	GMM-HMM	Microphone	96%	Created

OBJECTIVE

In this project we built a model for fire prediction using Arduino 33 ble sense and machine learning.

Data Capturing:-First we Collect the data in the form of RGB(Red, Green, Blue) through the APDS9960 sensor. Then we collect two data for the model i.e. Fire & No fire.

Training Model:- For Training the data set we use GOOGLE COLLAB. Then we setup the environment and importing the data sets(fire & no fire).

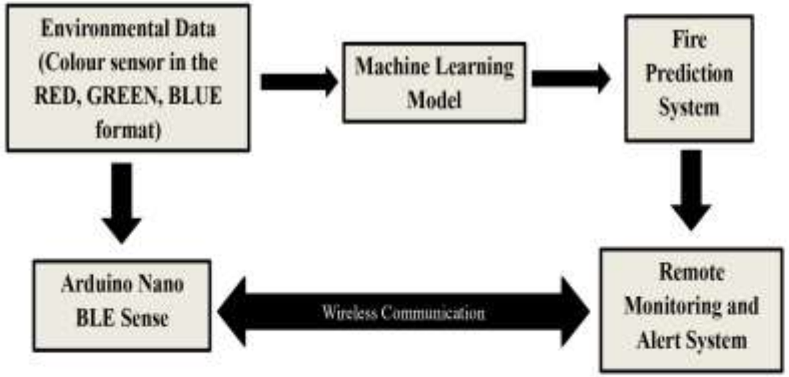
1.We have to set up the environment in collab with installing required libraries and Tensorflow.

2.After it Import the required library and check the TensorFlow version and load the datasets from files folder to our program.

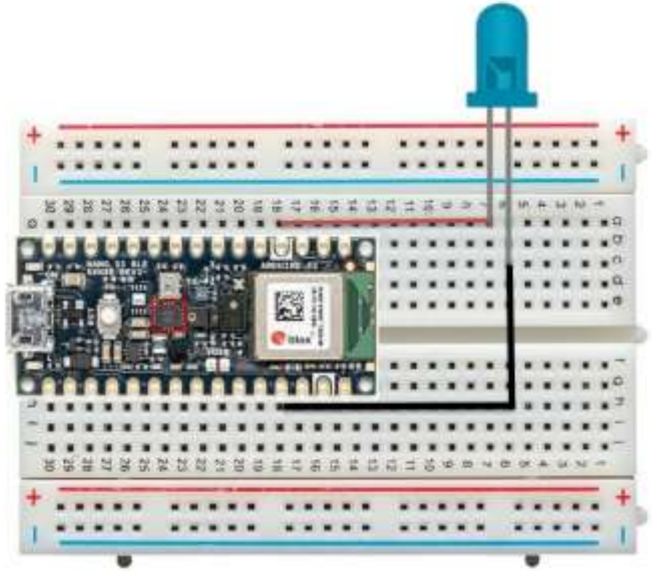
3.Create a one-hot encoding for representing categorical data to more expressive format and read one by one dataset and check how many samples are present.
4. the graph was plot for all the samples and the one-hot encoder was implemented 5. Data set randomization for training was done and splitted into test and train data 6. Define the model
7. after training the model using the test data to predict the data and test your model 8. converting the tensor flow model into TensorFlow lite to be implemented into our microcontroller board
9. after converting the model the model.h file should be seen in the file folder and download the model from there. Implementing our trained

model :- Move the model.h file with Arduino code folder open it and upload the code to ble sense. Then see the output.

BLOCK & CIRCUIT DIAGRAM


Environmental Data (Colour sensor in the RED, GREEN, BLUE format)
Machine Learning Model
Fire Prediction System
Arduino Nano BLE Sense
Wireless Communication
Remote Monitoring and Alert System

Block Diagram



Circuit Diagram

COMPONENTS USED:

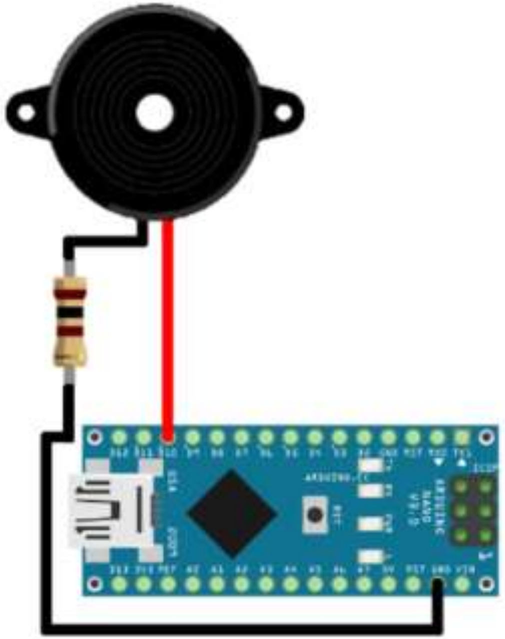
ARDUINO

HARDWARE USED:

1. Arduino nano 33 Ble sense
2. Buzzer
3. USB type b
4. Jumper wires

SOFTWARE USED

1. Arduino Ide
2. Google colab

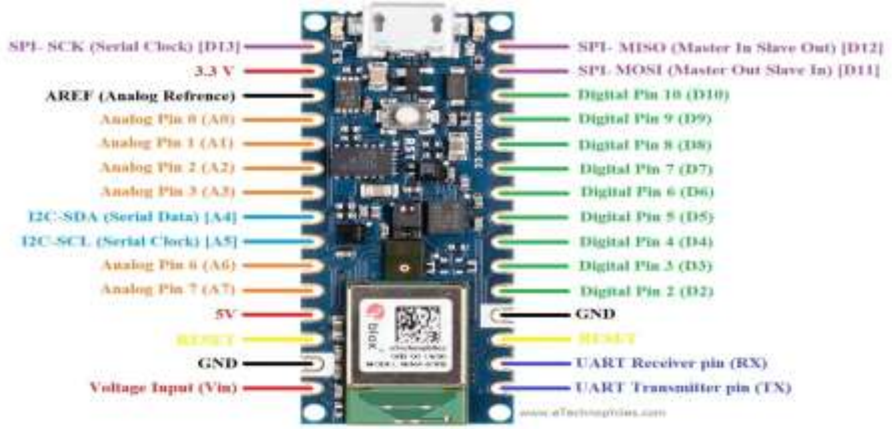


HARDWARE USED

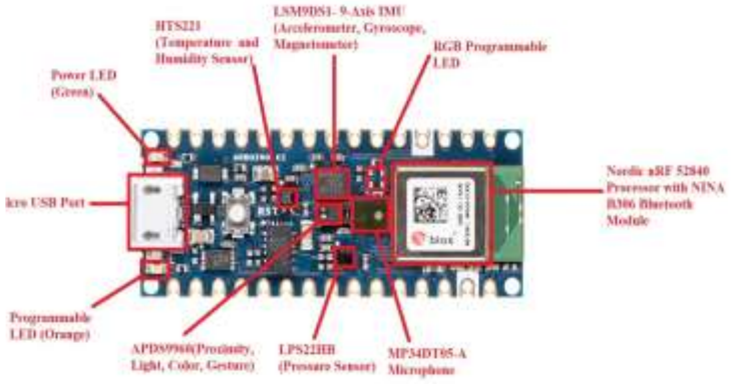
① Arduino Nano 33 BLE Sense

The Arduino Nano 33 BLE Sense is a compact development board equipped with an ARM Cortex-M4 processor and a variety of sensors, including an accelerometer, gyroscope, temperature sensor, humidity sensor, microphone, and color and gesture recognition sensors, making it ideal for IoT and sensor-based projects.

BRIEF ABOUT ARDUINO BLE NANO 33



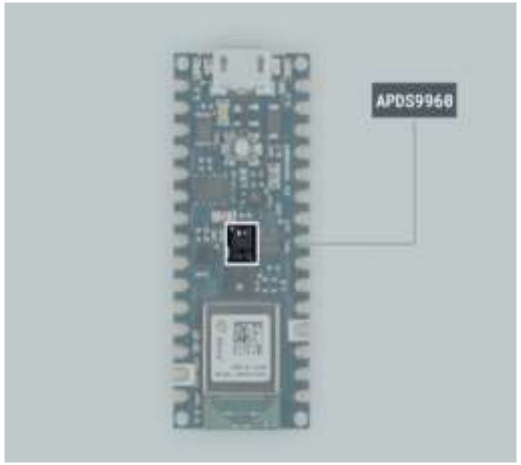
Different pins of Arduino Nano BLE 33


HTS221 (Temperature and Humidity Sensor)
LSM9DS1- 9-Axis IMU (Accelerometer, Gyroscope, Magnetometer)
RGB Programmable LED
Power LED (Green)
kro USB Port
Nordic nRF 52840 Processor with NINA B306 Bluetooth Module
Programmable LED (Orange)
APDS9960(Proximity, Light, Color, Gesture)
LPS22HB (Pressure Sensor)
MP34DT05-A Microphone

Different sensors of Arduino Nano BLE 33

Arduino ble nano for: fire prediction system

- ❑ The Arduino Nano 33 BLE Sense is equipped with various sensors, including a APDS9960 sensor.
- ❑ The APDS9960 sensor detech data from the surroundings.
- ❑ A fire prediction algorithm processes this data.
- ❑ It analyzes fire-specific pattern like color
- ❑ This system aid to create a fire prediction system by collecting data with onboard sensors, analyzing for anomalies, and triggering alerts in case of potential fire events.



BUZZER

A buzzer is an electromechanical device that generates a buzzing or beeping sound when an electrical current passes through it. It is commonly used in alarms, timers, and signaling systems to audibly alert or notify users of specific events or conditions.

JUMBER WIRES

Jumper wires are flexible, insulated electrical wires typically used in electronics and prototyping. They connect components on a breadboard, circuit board, or between various points in a circuit, enabling easy testing, troubleshooting, and customization of electronic projects.

USB TYPE MINI:

A USB Type B Mini connector is a small, rectangular USB interface commonly used for connecting various devices, including cameras, external hard drives, and some older smartphones, to computers or power sources. It features a compact, trapezoidal shape with a slightly wider top and is known for its versatility and compatibility.

SOFTWARE USED

GOOGLE COLAB

Google Colab, short for "Google Colaboratory," is a free cloud-based platform for writing, running, and sharing Python code collaboratively. It provides access to a virtual machine with GPU support, allowing users to develop machine learning models, conduct data analysis, and run Python notebooks without the need for local installations. Users can share and collaborate on projects in real-time, leveraging Google Drive integration. Google Colab has become popular in the machine learning community for its ease of use and resource availability, making it an attractive tool for research and experimentation

ARDUINO IDE

The Arduino Integrated Development Environment (IDE) is an open-source software platform used for programming and developing applications for Arduino microcontroller boards. It provides a user-friendly interface for writing, uploading, and debugging code on Arduino hardware. The IDE includes a text editor, compiler, and libraries to simplify programming tasks, making it accessible for both beginners and experienced developers. It supports the C/C++ programming languages, allowing users to create interactive projects and prototypes for various applications, from robotics to IoT devices. Additionally, the IDE offers a vibrant community and extensive documentation, fostering learning and innovation in the maker and electronics community.

SOFTWARE PACKAGES FOR COLAB

1. **MATPLOTLIB**

Matplotlib is a Python library used for creating high-quality, customizable, and interactive data visualizations, such as charts, graphs, and plots. It provides a wide range of functions and tools for generating publication-ready graphics, making it a popular choice for data scientists and researchers to visualize their data and findings.

2. **NUMPY**

NumPy is a fundamental Python library for numerical and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays. NumPy is crucial for tasks like data analysis, machine learning, and scientific research, enabling efficient data manipulation and computation.

3.**PANDAS**

Pandas is a powerful Python library for data manipulation and analysis. It offers data structures, such as dataframes, that enable efficient handling of structured data, including filtering, grouping, and statistical operations. Pandas simplifies data preprocessing tasks and integrates seamlessly with other data science libraries. It's a fundamental tool for data scientists and analysts for data exploration, cleaning, and transformation.

4)**TensorFlow**

It is an open-source machine learning framework developed by Google. It enables the creation and training of deep neural networks and is widely used for tasks like image and speech recognition, natural language processing, and more. TensorFlow provides a flexible and efficient platform for building and deploying machine learning models, making it a fundamental tool for AI and data science applications.

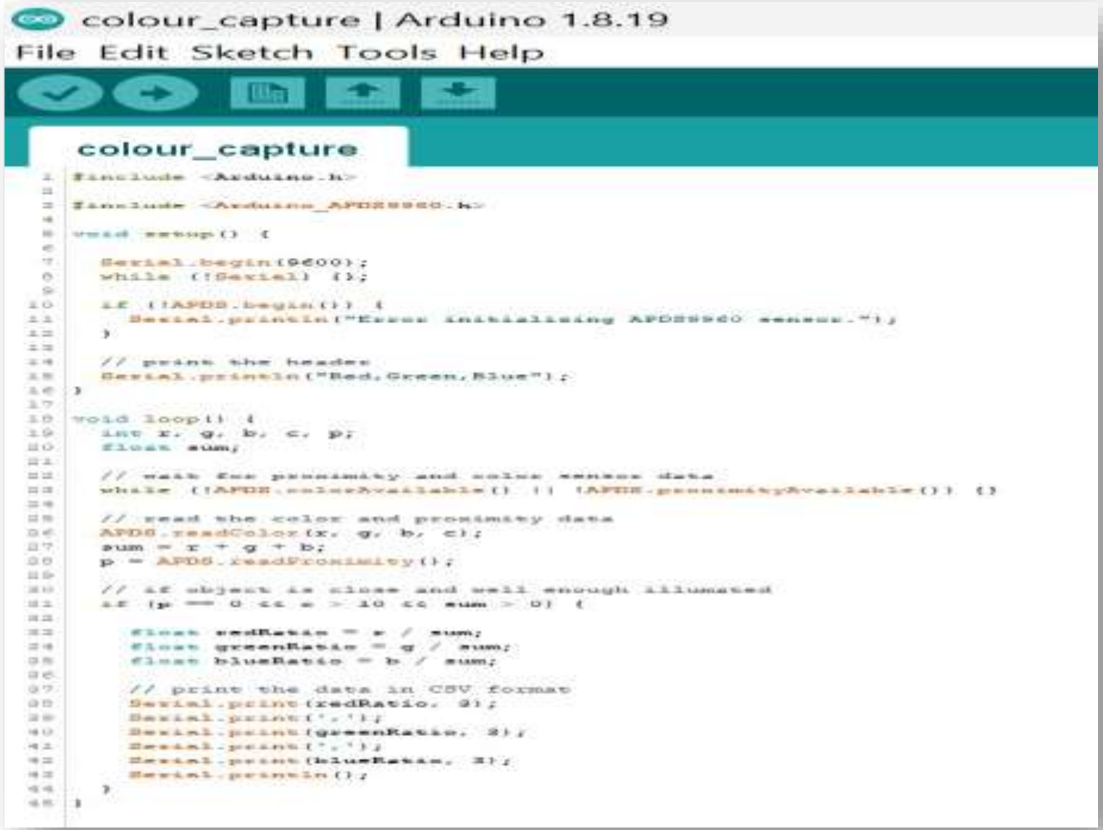
1)TENSORFLOWLITE.H

tensorflowlite.h is a header file used in Arduino when working with TensorFlow Lite, a framework for running machine learning models on microcontrollers and embedded systems. It contains essential functions and declarations for deploying machine learning models, making it easier to integrate AI and ML capabilities into Arduino-based projects, like object recognition or sensor data analysis.

2)<arduino_APDS9960.h>

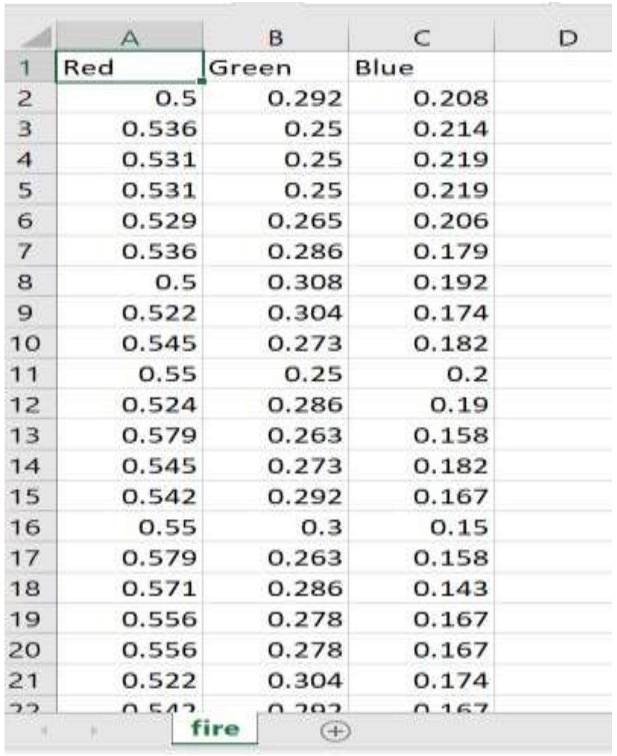
It is a library for Arduino that provides support for the APDS-9960 sensor module, which combines ambient light, RGB color, and gesture detection. This library simplifies the interfacing and usage of the sensor, allowing Arduino developers to incorporate gesture-based controls and ambient light sensing in their projects.

data
collection



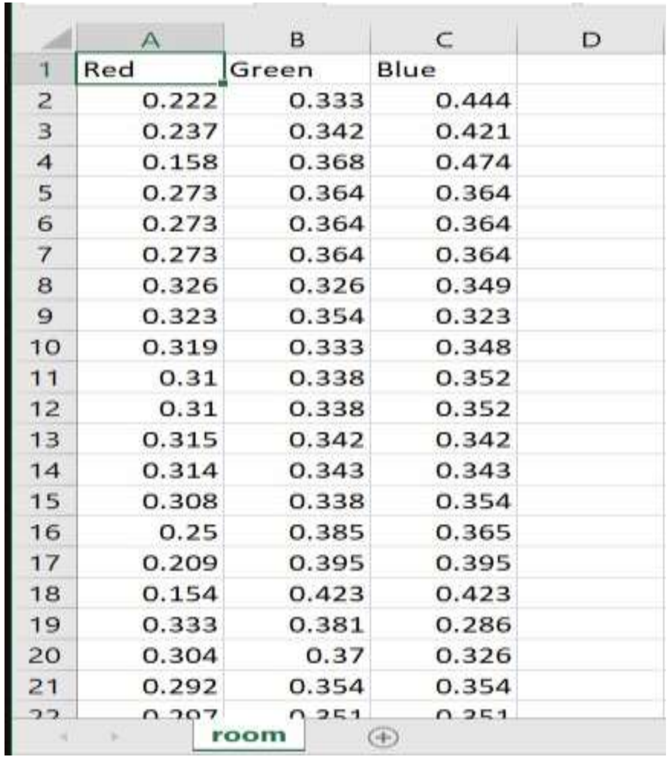
```
#include <Arduino.h>

#include <Arduino_APDS9960.h>

void setup() {

  Serial.begin(9600);
  while (!Serial) {};

  if (!APDS.begin()) {
    Serial.println("Error initializing APDS9960 sensor.");
  }

  // print the header
  Serial.println("Red,Green,Blue");
}

void loop() {
  int r, g, b, c, p;
  float sum;

  // wait for proximity and color sensor data
  while (!APDS.colorAvailable() || !APDS.proximityAvailable()) {}

  // read the color and proximity data
  APDS.readColor(r, g, b, c);
  sum = r + g + b;
  p = APDS.readProximity();

  // if object is close and well enough illumated
  if (p == 0 && c > 10 && sum > 0) {

    float redRatio = r / sum;
    float greenRatio = g / sum;
    float blueRatio = b / sum;

    // print the data in CSV format
    Serial.print(redRatio, 3);
    Serial.print(',');
    Serial.print(greenRatio, 3);
    Serial.print(',');
    Serial.print(blueRatio, 3);
    Serial.println();
  }
}
```

CODE FOR CAPTURING THE DATA IN RGB FORMAT



	A	B	C	D
1	Red	Green	Blue	
2	0.5	0.292	0.208	
3	0.536	0.25	0.214	
4	0.531	0.25	0.219	
5	0.531	0.25	0.219	
6	0.529	0.265	0.206	
7	0.536	0.286	0.179	
8	0.5	0.308	0.192	
9	0.522	0.304	0.174	
10	0.545	0.273	0.182	
11	0.55	0.25	0.2	
12	0.524	0.286	0.19	
13	0.579	0.263	0.158	
14	0.545	0.273	0.182	
15	0.542	0.292	0.167	
16	0.55	0.3	0.15	
17	0.579	0.263	0.158	
18	0.571	0.286	0.143	
19	0.556	0.278	0.167	
20	0.556	0.278	0.167	
21	0.522	0.304	0.174	

fire

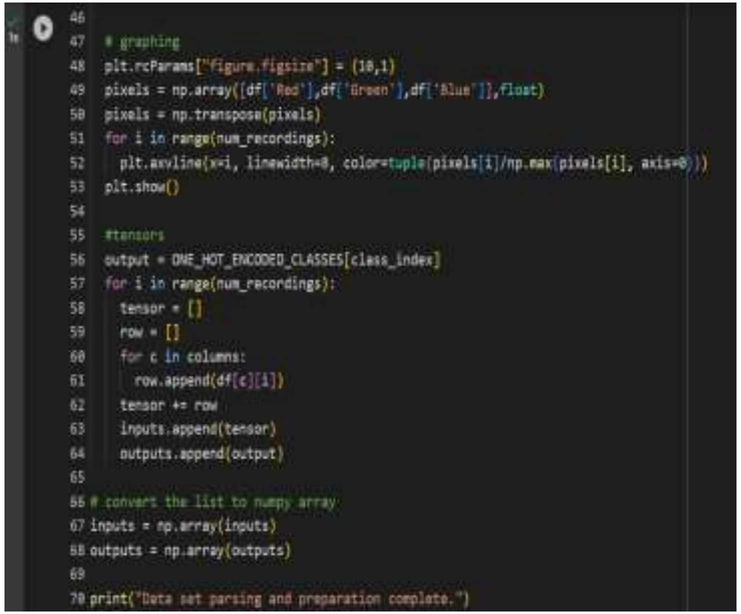


	A	B	C	D
1	Red	Green	Blue	
2	0.222	0.333	0.444	
3	0.237	0.342	0.421	
4	0.158	0.368	0.474	
5	0.273	0.364	0.364	
6	0.273	0.364	0.364	
7	0.273	0.364	0.364	
8	0.326	0.326	0.349	
9	0.323	0.354	0.323	
10	0.319	0.333	0.348	
11	0.31	0.338	0.352	
12	0.31	0.338	0.352	
13	0.315	0.342	0.342	
14	0.314	0.343	0.343	
15	0.308	0.338	0.354	
16	0.25	0.385	0.365	
17	0.209	0.395	0.395	
18	0.154	0.423	0.423	
19	0.333	0.381	0.286	
20	0.304	0.37	0.326	
21	0.292	0.354	0.354	

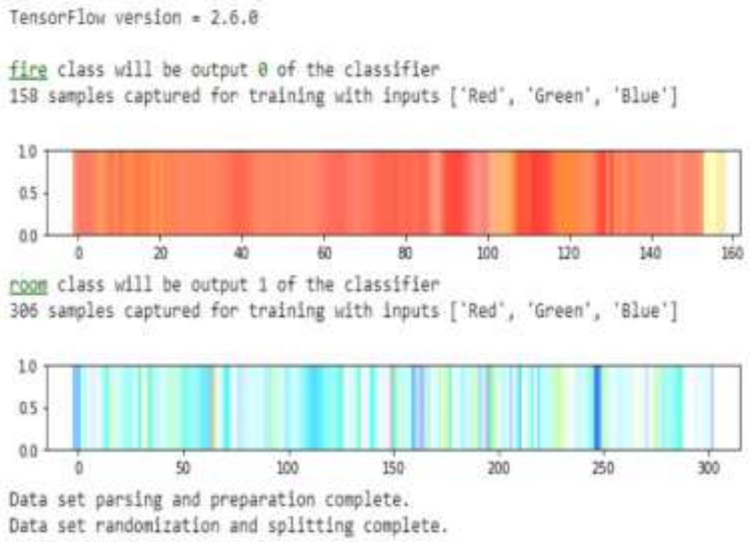
room

FIRE DATA NON FIRE DATA

PLOTING THE GRAPH FOR DATA



```python

  # graphing
  plt.rcParams["figure.figsize"] = (10,1)
  pixels = np.array([df['Red'],df['Green'],df['Blue']],float)
  pixels = np.transpose(pixels)
  for i in range(num_recordings):
    plt.axvline(x=i, linewidth=8, color=tuple(pixels[i]/np.max(pixels[i], axis=0)))
  plt.show()

  #tensors
  output = ONE_HOT_ENCODED_CLASSES[class_index]
  for i in range(num_recordings):
    tensor = []
    row = []
    for c in columns:
      row.append(df[c][i])
    tensor += row
    inputs.append(tensor)
    outputs.append(output)

# convert the list to numpy array
inputs = np.array(inputs)
outputs = np.array(outputs)

print("Data set parsing and preparation complete.")
```



```
TensorFlow version = 2.6.0

fire class will be output 0 of the classifier
158 samples captured for training with inputs ['Red', 'Green', 'Blue']
```

```
room class will be output 1 of the classifier
306 samples captured for training with inputs ['Red', 'Green', 'Blue']
```

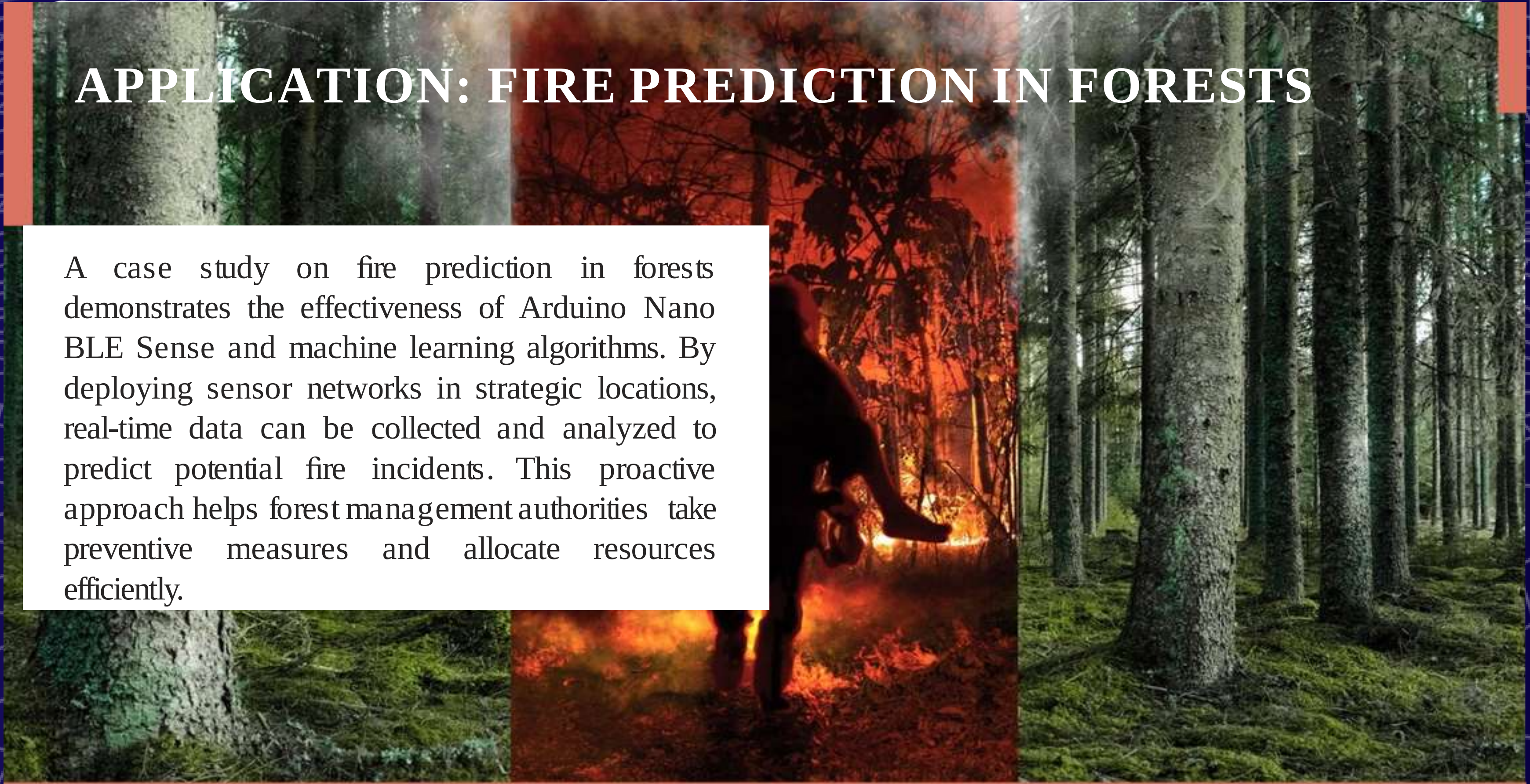
```
Data set parsing and preparation complete.
Data set randomization and splitting complete.
```

Dataset samples graph and data info

FUTURE SCOPE

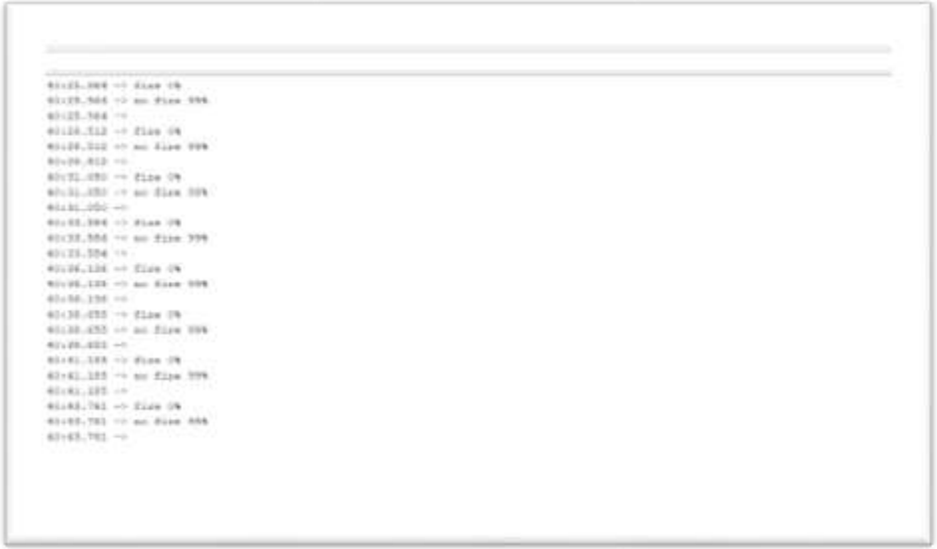
The future scope of fire prediction using Arduino Nano 33 BLE holds significant promise. With advances in sensor technology, machine learning, and IoT integration, this approach can become a valuable tool for early fire detection. It has potential applications in industrial settings, smart homes, and wildfire monitoring systems. Real-time data transmission and analysis can enhance safety measures and help mitigate the destructive impact of fires. As technology evolves, the integration of AI and cloud-based solutions can further improve prediction accuracy and response times, making it a crucial tool for fire prevention and control in the future.

APPLICATION: FIRE PREDICTION IN FORESTS

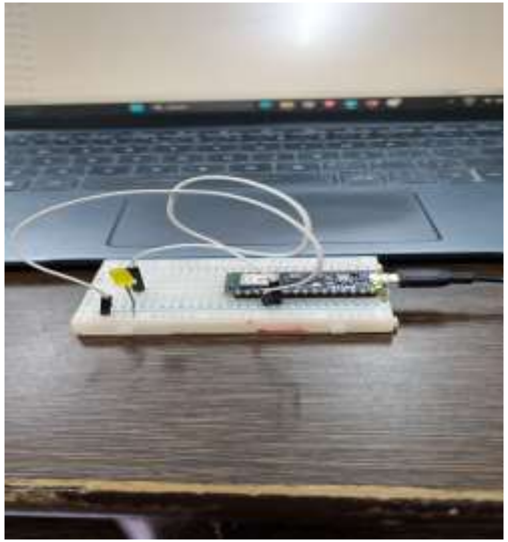


A case study on fire prediction in forests demonstrates the effectiveness of Arduino Nano BLE Sense and machine learning algorithms. By deploying sensor networks in strategic locations, real-time data can be collected and analyzed to predict potential fire incidents. This proactive approach helps forest management authorities take preventive measures and allocate resources efficiently.

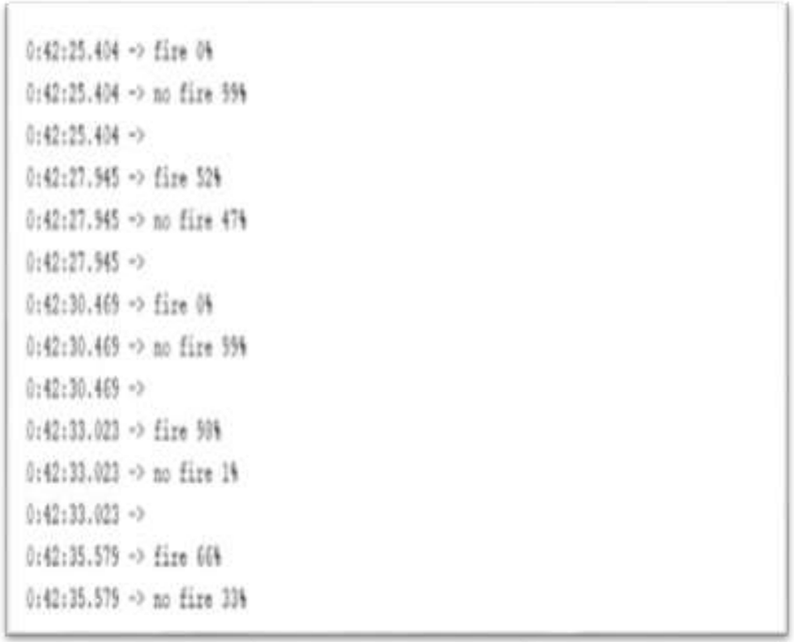
RESULT



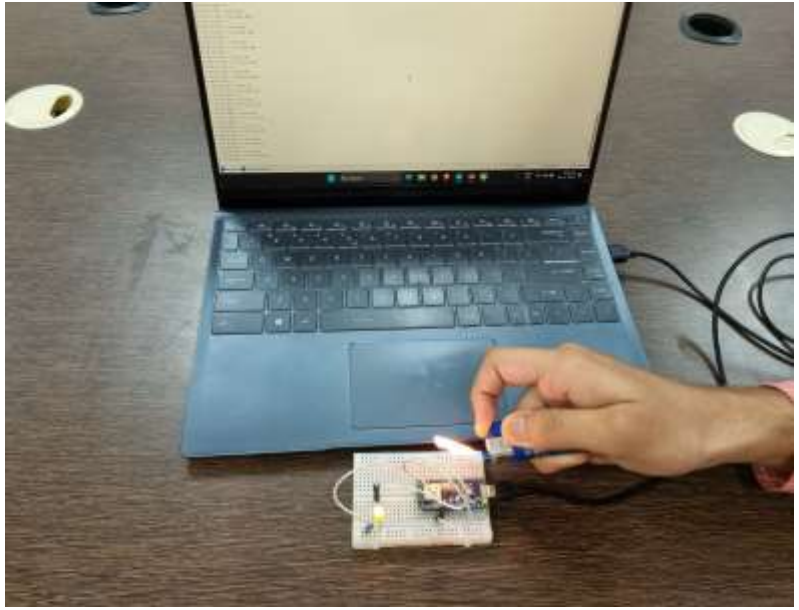
```
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
[illegible] -> fire 0%
[illegible] -> no fire 99%
[illegible] ->
```



NO FIRE DETECTION



```
0:42:25.404 -> fire 0%
0:42:25.404 -> no fire 59%
0:42:25.404 ->
0:42:27.945 -> fire 52%
0:42:27.945 -> no fire 47%
0:42:27.945 ->
0:42:30.469 -> fire 0%
0:42:30.469 -> no fire 59%
0:42:30.469 ->
0:42:33.023 -> fire 50%
0:42:33.023 -> no fire 1%
0:42:33.023 ->
0:42:35.579 -> fire 66%
0:42:35.579 -> no fire 33%
```



FIRE DETECTED

CONCLUSION

In conclusion, leveraging Arduino Nano BLE Sense and advanced machine learning algorithms can significantly enhance fire prediction accuracy. This technology enables early detection, proactive management, and timely response, ultimately saving lives and mitigating the devastating effects of fires. With further research and development, we can revolutionize fire prediction and make our communities safer.

REFERENCES

1. Pang, Y. et al. (2022) ‘Forest fire occurrence prediction in China based on machine learning methods’, Remote Sensing, 14(21), p. 5546. doi:10.3390/rs14215546.
2. W. Ma, Z. Feng, Z. Cheng, S. Chen, and F. Wang, "Identifying Forest Fire Driving Factors and Related Impacts in China Using Random Forest Algorithm," Forests, vol. 11, p. 507, 2020.
3. K. J. Maingi and M. C. Henry, "Factors influencing wildfire occurrence and distribution in eastern Kentucky, USA," Int. J. Wildland Fire, vol. 16, pp. 23-33, 2007.
4. K. L. Pew and C. P. S. Larsen, "GIS analysis of spatial and temporal patterns of human-caused wildfires in the temperate rainforest of Vancouver Island, Canada," Forest Ecol. Manag., vol. 140, pp. 1-18, 2001.
5. Z. S. Pourtaghi, H. R. Pourghasemi, R. Aretano, and T. Semeraro, "Investigation of general indicators influencing on forest fire and its susceptibility modeling using different data mining techniques," Ecol. Indic., vol. 64, pp. 72-84, 2016.
6. Open Data Nepal, "Forest fire dataset throughout Nepal - March 2021," Open Data Nepal, Available: https://opendatanepal.com/dataset/forest-fire-dataset-throughout-nepal-march-2021

TIMELINE

SL. NO.	TASK	TIME	STATUS
01	Group making	3 days	Complete
02	Topic choosing	7 days	Complete
03	Purchasing hardware	6 days	Complete
04	Collecting data	2 days	Complete
05	Training data	3 days	Complete
06	Code implementation	15 days	Complete
07	upload code in Arduino	1 days	Complete
08	Testing hardware	1 days	Complete
09	Live display	3 days	Compete

THANK YOU