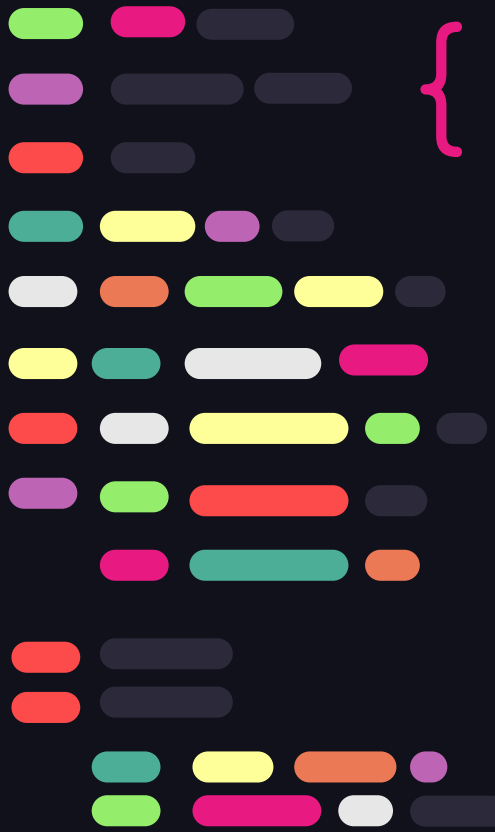


Detector de fuego con Arduino nano para predecir incendios con Machine Learning

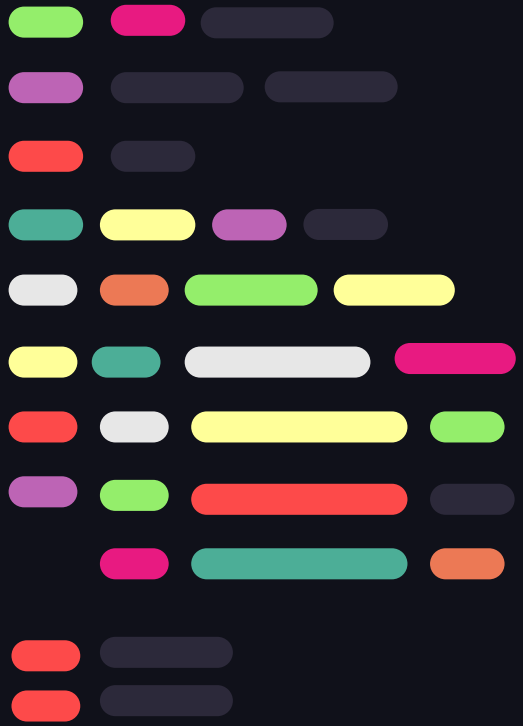
Amy Herrera Mora B53473
Isaac Rojas H B76693

01 { ..

Objetivos

} ..

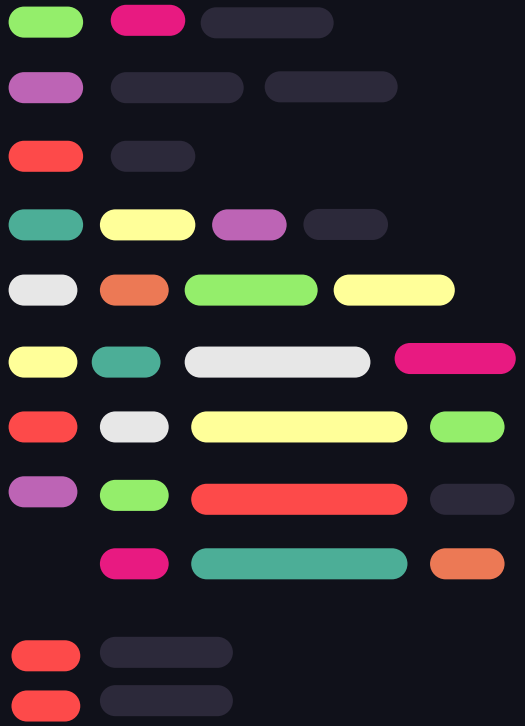
Objetivos



Objetivo General:

- Desarrollar un detector de fuego con un Arduino Nano para predecir incendios utilizando Machine Learning.

Objetivos



Objetivos específicos:

- Implementar un circuito junto al microcontrolador Arduino Nano que sirva como alerta.
- Construir un modelo de red neuronal y entrenarlo con los datos obtenidos con ayuda de bibliotecas disponibles en Python para la detección de fuego.
- Exportar la red neuronal creada a un modelo TensorFlow lite para su utilización en el Arduino Nano.
- Implementar un programa que utilice la red neuronal para detectar fuego y alertar de la presencia del mismo.

02 { ..

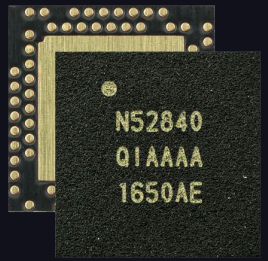
Elementos utilizados

} ..

Elementos utilizados

Arduino Nano 33 BLE Sense Lite

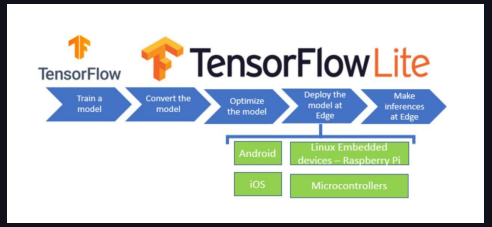
nRF52840



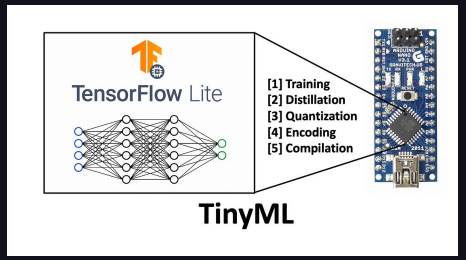
APDS-9960

Elementos utilizados

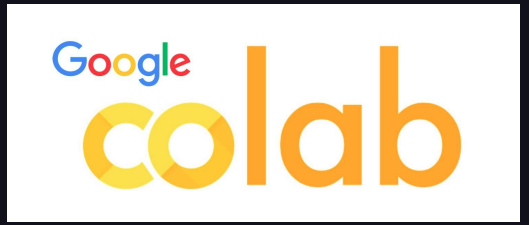
TensorFlow Lite



TinyMl



Google Colab



03 { ..

Código para la captura de datos

} ..

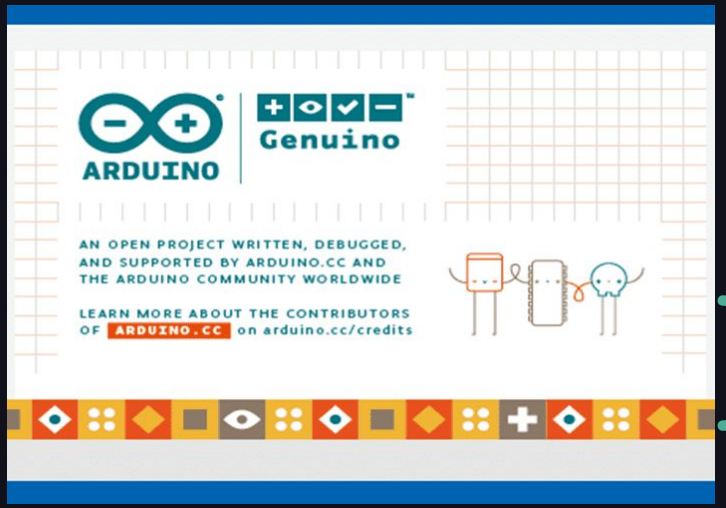
Código para el Arduino

Se incluye biblioteca APDS9960

Dentro del setup() se establece el baudrate, se verifica inicialización y se imprime encabezado

En el loop `APDS.colorAvailable()` para ver si hay color, `APDS.proximityAvailable()` para verificar cercanía y con esto guardar en las variables con `APDS.readColor`

Con el if se verifica proximidad y de acuerdo a esto, se calcula la relación RGB



Código para el Arduino

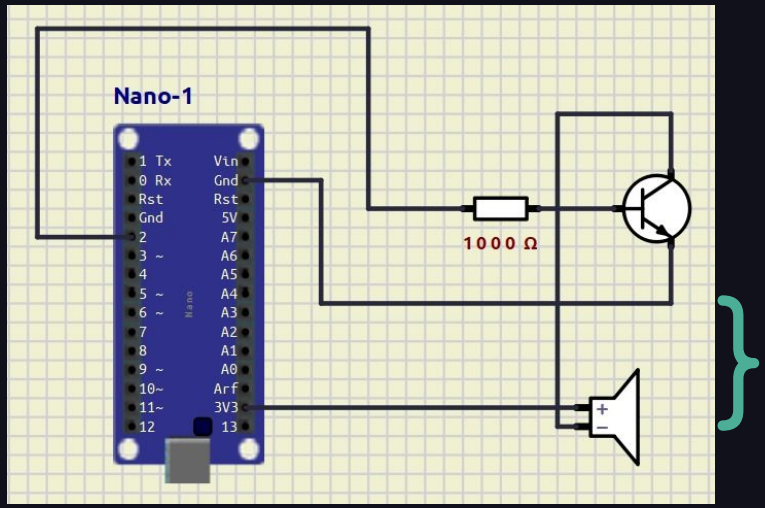
```
#include <Arduino_APDS9960.h>

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

  if (!APDS.begin()) {

    Serial.println("No se inicio correctamente");

    }

  Serial.println("Rojo, Verde, Azul");
}

void loop() {
  int r, v, a, c, p;
  float sum;

  while (!APDS.colorAvailable() || !APDS.proximityAvailable()) {}


  APDS.readColor(r,v,a,c);
  sum = r + v + a;

  p = APDS.readProximity();

  if(p >= 0 && c > 10 && sum >0){

   float redRatio = r/sum;
   float greenRatio = v/sum;
   float blueRatio = a/sum;

   Serial.print(redRatio, 3);
   Serial.print(',');
   Serial.print(greenRatio, 3);
   Serial.print(',');
   Serial.print(blueRatio, 3);
   Serial.println();


  }
}
```

Diseño del circuito

Transistor BC547 que actúa como interruptor para el Buzzer

Resistencia controlando la corriente de saturación del transistor

Buzzer de 5V activado por el transistor en saturación

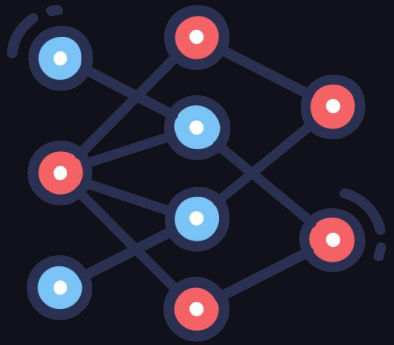


04 { ..
Diseño de la red neuronal

} ..

Diseño de la Red Neuronal

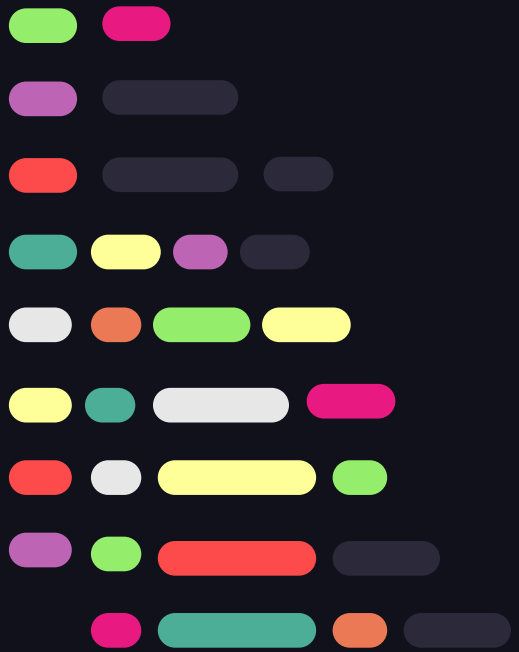
Modelo Secuencial: Capas apiladas, la salida es la entrada de la siguiente



Diseño de la Red Neuronal

Capa 1: 8 neuronas con modelo ReLU
Capa 2: 5 neuronas con modelo ReLU
Capa 3: Capa de 2 neuronas, una por cada predicción (Fuego o No Fuego)

Entrenamiento de la red



Script en Google Colab, que implementó la creación, el entrenamiento y la exportación del modelo de red neuronal.

Se utilizaron 6300 datos capturados para cada set:

- Habitaciones sin fuego
- Habitaciones con fuego

05 { ..

Resultados

} ..

Resultados

Ambiente sin fuego:

```
fuego 16%
no fuego 83%

fuego 16%
no fuego 83%

fuego 5%
no fuego 94%

fuego 5%
no fuego 94%

fuego 5%
no fuego 94%

fuego 5%
no fuego 94%

fuego 10%
no fuego 89%
```

Ambiente con fuego:

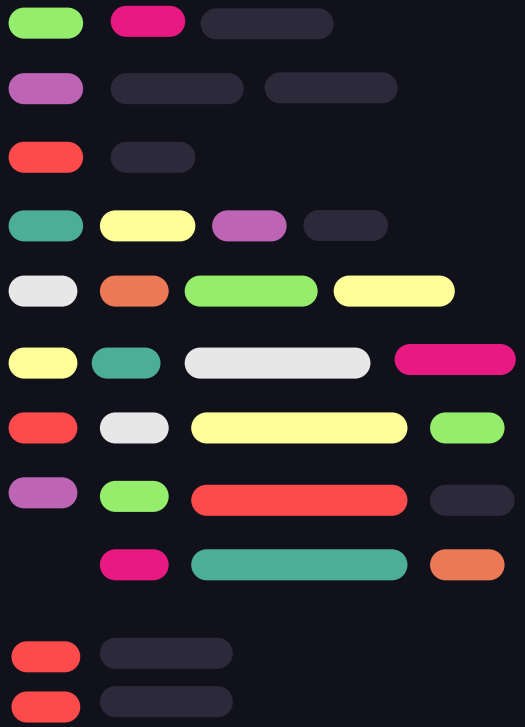
```
fuego 92%
no fuego 7%

fuego 97%
no fuego 2%

fuego 96%
no fuego 3%

fuego 96%
no fuego 3%

fuego 96%
no fuego 3%

fuego 97%
no fuego 2%

fuego 97%
no fuego 2%
```

06 { ..

Conclusiones

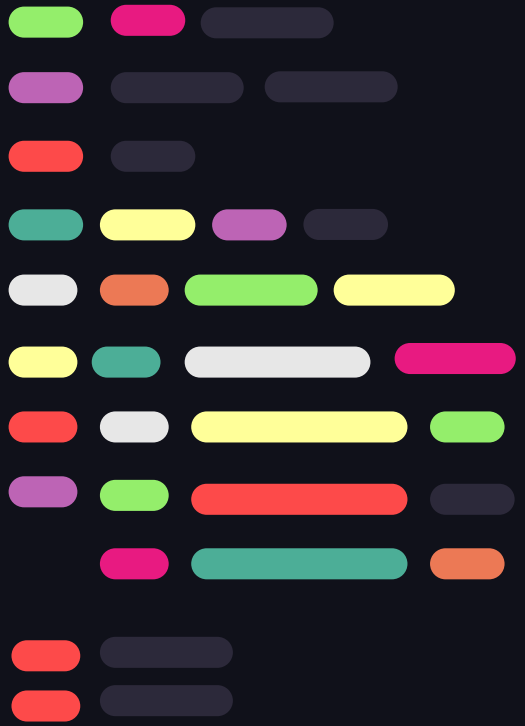
} ..

Conclusiones



- Implementar un modelo de red neuronal que provea datos de predicción e identificación adecuado, requiere de una captura de datos de entrenamiento extenso.
- Crear modelos de redes neuronales buenos, requieren de mucha potencia computacional.
- Ejecutar un modelo de redes neuronales no requiere de mucha potencia computacional si se realiza de forma correcta.

Conclusiones



- El uso de la herramienta de desarrollo de google collab, para desarrollar, instanciar, entrenar y exportar redes neuronales es de suma importancia.
- Herramientas como TensorFlow Lite son indispensables para desarrollar un proyecto de este tipo considerando que se tienen recursos limitados.
- Utilizar bibliotecas reconocidas permiten una mejor implementación de AI por toda la documentación disponible y trabajos previamente realizados.