Universidad Autónoma de Chihuahua

Facultad de Ingeniería

APLICACIONES EMBEBIDAS PARA INTERNET DE LAS COSAS

Grupo: 9HW1

PRACTICA: Proyecto final

Maestro: Alberto Pacheco González

Integrantes:

Kevin Daniel Lazcano Ortiz 339006

Hugo Alonso Mendoza Ponce 339020

Fernando García Morales 343454

Contenido

Resumen del proyecto

El proyecto desarrollado con el microcontrolador ESP8266 tiene como objetivo principal la detección precisa de la frecuencia en hertzios (Hz) de sirenas. La implementación se centra en aprovechar las capacidades del ESP8266 para captar señales de audio y analizarlas en tiempo real.

En primer lugar, se diseñó un sistema de adquisición de audio que permite al ESP8266 recibir las señales sonoras provenientes de su entorno. A través de un micrófono incorporado o conectado al microcontrolador, se logra capturar eficientemente las frecuencias presentes en el sonido ambiente.

Posteriormente, se implementó un algoritmo de procesamiento digital de señales (DSP) en el ESP8266 para analizar el espectro de frecuencias de las señales de audio recibidas. Este algoritmo permite identificar de manera precisa la frecuencia predominante, facilitando así la detección de la sirena.

Además, se incorporó una interfaz de usuario amigable para visualizar los resultados obtenidos. Esta interfaz puede ser accedida a través de un dispositivo móvil o una computadora, brindando información en tiempo real sobre la frecuencia detectada, así como otros datos relevantes.

El proyecto ofrece diversas aplicaciones prácticas, como la detección temprana de vehículos de emergencia en situaciones de tráfico intenso o la integración con sistemas de seguridad para alertar sobre la presencia de sirenas en áreas específicas. La versatilidad del ESP8266 y la precisión del sistema de detección hacen de este proyecto una herramienta eficiente para mejorar la seguridad y la respuesta ante situaciones de emergencia en entornos urbanos.

En resumen, el proyecto de detector de Hz de sirenas con ESP8266 representa una solución innovadora y tecnológica para la detección y análisis de frecuencias sonoras, ofreciendo aplicaciones prácticas y mejorando la eficiencia en la gestión de emergencias.

Introducción

En un mundo cada vez más urbanizado y dinámico, la gestión efectiva de emergencias y la seguridad ciudadana se han vuelto imperativas. En este contexto, el proyecto de desarrollo basado en el microcontrolador ESP8266 emerge como una solución innovadora para la detección precisa de frecuencias sonoras, centrándose específicamente en la identificación de las señales emitidas por sirenas.

Este proyecto tiene como objetivo principal aprovechar las capacidades avanzadas del ESP8266 para capturar, procesar y analizar las señales acústicas presentes en el entorno. La implementación de un sistema de adquisición de audio, junto con algoritmos de procesamiento digital de señales, permite la detección eficiente de las frecuencias en hertzios (Hz) asociadas a las sirenas de vehículos de emergencia.

A medida que las ciudades crecen y la movilidad urbana se intensifica, la capacidad de identificar rápidamente la presencia de vehículos de emergencia se vuelve crucial. Este proyecto no solo busca ofrecer una solución tecnológica precisa y eficiente, sino también proporcionar una herramienta versátil que pueda integrarse en sistemas de seguridad existentes.

A lo largo de este documento, exploraremos en detalle el diseño y la implementación del proyecto, destacando sus componentes clave, como el sistema de adquisición de audio, los algoritmos de procesamiento digital de señales, y la interfaz de usuario. Además, se discutirán las posibles aplicaciones prácticas de este sistema en entornos urbanos, contribuyendo así a mejorar la seguridad y la capacidad de respuesta ante situaciones de emergencia. Este proyecto representa un paso significativo hacia la aplicación de la tecnología para abordar desafíos contemporáneos en el ámbito de la seguridad ciudadana.

Desarrollo del proyecto

El programa está diseñado para la adquisición de datos a través de un micrófono analógico utilizando un microcontrolador ESP8266 y realiza análisis de señales utilizando la Transformada Rápida de Fourier (FFT) para detectar frecuencias y amplitudes en la señal capturada, más específicamente para poder detectar sirenas de vehículos de emergencia tomando en cuenta las frecuencias de sus tonos a si como patrones de repetición.

El código busca patrones específicos en la secuencia de frecuencias capturadas por el sensor de sonido. Cuando se detecta un patrón que sugiere la presencia de una sirena de vehículo de emergencia, se toman medidas como encender un LED, se podría realizar cualquier otra acción como por ejemplo bajar el volumen de un estéreo en un automóvil para asi evitar la distracción del conductor u otras acciones como encender las luces intermitentes del vehículo automáticamente.

```
#include "arduinoFFT.h"
#include <iostream>
#include <regex>
#include <vector>
#include <string>
#include <ESP8266WiFi.h>
#define SAMPLES 256//256
#define SAMPLING_FREQUENCY 10000

arduinoFFT FFT = arduinoFFT();

unsigned int sampling_period_us;
unsigned long microseconds;
double vReal[SAMPLES];
double vImag[SAMPLES];
unsigned long newTime, oldTime;
/*detectar la presencia de al menos 2 "A" seguidas de al menos 2 "B" o al menos 2 "B"
 seguidas de al menos 2 "A" en cualquier parte del string.*/
std::regex hi_lo("A{2,}B{2,}|B{2,}A{2,}", std::regex_constants::ECMAScript |
std::regex_constants::icase);
std::vector<float>  signals;

float FA_MIN=700;//700
float FA_MAX=1500;//1500
```

```
float FB_MIN=200;//200
float FB_MAX=650;//600

void setup() {
  Serial.begin(115200);
  pinMode(2,OUTPUT);
  digitalWrite(2,HIGH);// led integrado
  sampling_period_us = round(1000000 * (1.0 / SAMPLING_FREQUENCY));
}

void loop() {
  for (int i = 0; i < SAMPLES; i++) {
    newTime = micros() - oldTime;
    oldTime = newTime;
    vReal[i] = analogRead(A0);
    vImag[i] = 0;
    while (micros() < (newTime + sampling_period_us)) { /* do nothing to wait */ }
  }
  FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
  FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

  for (int i = 2; i < (SAMPLES / 2); i++) {
    if (vReal[i]>200) {
      std::cout<<"Frecuencia:"<<i * (SAMPLING_FREQUENCY / SAMPLES)<<" Hz, amplitud:
"<<vReal[i]<<std::endl;
      //detectar si el frame de 20 tonos tiene una sirena
      if(signals.size()==20){
        detectSiren(signals);
        signals.clear();
      }else{
        //llenamos
        signals.push_back(i * (SAMPLING_FREQUENCY / SAMPLES));
      }
    }
  }

}
void detectSiren(std::vector<float>signals){
  if(regex_search(get_hilo(signals),hi_lo)){
    digitalWrite(2,LOW);
    std::cout<<"Sirena detectada "<<get_hilo(signals)<<std::endl;
    delay(5000);
    digitalWrite(2,HIGH);
    ESP.restart();// evitar lecturas incorrectas de frecuencias
```

```
  }else{
    digitalWrite(2,HIGH);
    std::cout<<"no detectada "<<get_hilo(signals)<<std::endl;
  }
}
bool isFa(float freq) {return (FA_MIN <= freq && freq <= FA_MAX);}
bool isFb(float freq) {return (FB_MIN <= freq && freq <= FB_MAX);}

std::string get_hilo(std::vector<float> freqs){
  std::string patron="";
  for(float freq: freqs){
    if(isFa(freq)) patron+="A";
    else if(isFb(freq)) patron+="B";
    else patron+="-";
  }
  return patron;
}
```

Librerías utilizadas:

- arduinoFFT.h: Se utiliza para realizar la transformada de Fourier rápida (FFT), que ayuda a analizar las frecuencias presentes en la señal de sonido.
- <iostream>, <regex>, <vector>, <string>: Librerías estándar de C++ utilizadas para entrada/salida, manipulación de cadenas y trabajo con expresiones regulares.

Constantes y Variables Globales:

- SAMPLES: Número de muestras utilizadas en cada iteración de lectura del sensor de sonido.
- SAMPLING_FREQUENCY: Frecuencia de muestreo del sensor de sonido.
- arduinoFFT FFT: Objeto de la clase arduinoFFT utilizado para realizar la transformada de Fourier rápida.
- vReal y vImag: Arreglos que almacenan la parte real e imaginaria de las muestras de la señal de sonido.
- newTime y oldTime: Variables para medir el tiempo transcurrido entre lecturas del sensor.
- hi_lo: Expresión regular para detectar patrones específicos en la secuencia de frecuencias.

Parámetros de Frecuencia:

- FA_MIN, FA_MAX, FB_MIN, FB_MAX: Rangos de frecuencias que se considerarán como "A" y "B".

Configuración Inicial (setup):

- Inicialización de la comunicación serial.
- Configuración de un pin digital (2) como salida para un LED integrado.
- Configuración del periodo de muestreo en microsegundos.

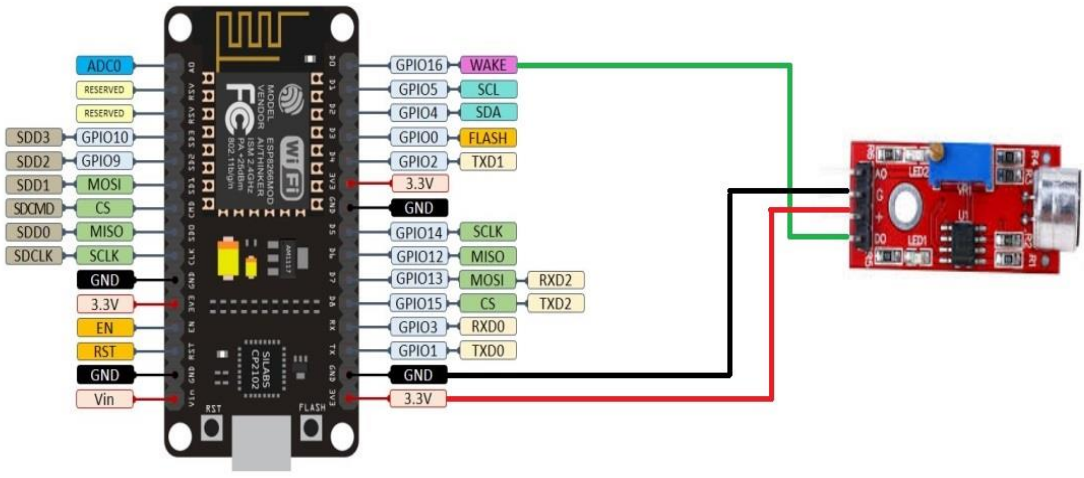
Loop Principal (loop):

- Se realiza un bucle para leer SAMPLES muestras del sensor de sonido.
- Se aplica una ventana de Hamming y se realiza la FFT para analizar las frecuencias presentes en la señal.
- Se calcula la magnitud de las frecuencias y se imprime la frecuencia y amplitud si la amplitud es mayor a 200.
- Se busca un patrón de 20 frecuencias consecutivas que se interpreta como un posible sonido de sirena.

Funciones Adicionales:

- detectSiren(std::vector<float> signals): Detecta si la secuencia de frecuencias cumple con el patrón predefinido mediante una expresión regular. En caso afirmativo, apaga un LED, muestra un mensaje y reinicia el microcontrolador.
- isFa(float freq), isFb(float freq): Verifican si una frecuencia dada está dentro de los rangos definidos para "A" o "B".
- get_hilo(std::vector<float> freqs): Convierte una secuencia de frecuencias en una cadena de caracteres según los rangos de frecuencia definidos.

Diagrama de conexiones



RESULTADOS

Resultados obtenidos:

1. **Detección de Sirenas Exitosa:** Uno de los logros más destacados del proyecto fue la capacidad de detectar señales de sirenas de manera efectiva. Utilizando un micrófono y el ESP8266, logramos implementar un algoritmo de detección que respondía a las frecuencias características de las sirenas de emergencia.

2. **Alertas en Tiempo Real:** El ESP8266 pudo procesar las señales del micrófono y enviar alertas en tiempo real cuando detectaba una sirena. Esto podría ser extremadamente útil en situaciones de emergencia y seguridad.

Problemas enfrentados:

1. **Bugs de Detección Inicial**: Al principio, el proyecto enfrentó problemas de detección, como la no coincidencia de las señales del micrófono con las frecuencias esperadas de las sirenas. Esto se debió a la necesidad de afinar y calibrar el algoritmo de detección para mejorar la precisión.
2. **Sensibilidad del Micrófono:** La sensibilidad del micrófono resultó ser un desafío, ya que a veces generaba falsas alarmas debido a ruidos ambientales o a señales no deseadas que se asemejaban a sirenas. Ajustar el umbral de sensibilidad fue crucial para mitigar este problema.
3. **Limitaciones de Frecuencia:** Uno de los mayores desafíos fue la limitación de la frecuencia máxima del ESP8266. Las sirenas modernas utilizan frecuencias altas que a veces superaban la capacidad del ESP8266 para detectarlas con precisión. Esto requirió una selección cuidadosa de micrófonos y ajustes en el hardware para mejorar la capacidad de detección en frecuencias más altas.

Limitaciones encontradas:

1. **Recursos Limitados del ESP8266:** El ESP8266, aunque potente, tiene recursos limitados en términos de potencia de procesamiento y memoria. Esto

limitó la complejidad del algoritmo de detección que podíamos implementar y la cantidad de datos que podíamos procesar en tiempo real.

2. **Requiere Conexión a Internet:** Para enviar alertas en tiempo real o registrar eventos, el proyecto dependía de una conexión a Internet constante, lo que podría ser una limitación en áreas sin acceso a la red.
3. **Configuración Inicial Compleja:** La configuración inicial y la calibración del sistema eran complejas y requerían conocimientos técnicos, lo que limitaba su accesibilidad a usuarios no técnicos.

Enlace github:

https://github.com/Hamp001/DETECTOR-DE-SIRENAS

Primeras pruebas:

https://drive.google.com/file/d/1GsUk0WAGNs0SEYOi2drcRzhaQpGPQYMo/view?usp=sharing

https://drive.google.com/file/d/1Gj23FkOVl9lII8DJQRLtBnCkM9Za9aG-/view?usp=drive_link


```
Frecuencia:78 Hz, amplitud: 75.0657
Frecuencia:78 Hz, amplitud: 68.9815
Frecuencia:1248 Hz, amplitud: 104.067
Frecuencia:1287 Hz, amplitud: 68.8008
Frecuencia:78 Hz, amplitud: 79.451
Frecuencia:1248 Hz, amplitud: 192.511
Frecuencia:1287 Hz, amplitud: 218.984
Frecuencia:78 Hz, amplitud: 69.7889
Frecuencia:1287 Hz, amplitud: 119.466
Frecuencia:1326 Hz, amplitud: 160.98
Frecuencia:78 Hz, amplitud: 74.8076
Frecuencia:78 Hz, amplitud: 78.315
Frecuencia:78 Hz, amplitud: 74.1281
Frecuencia:78 Hz, amplitud: 73.4546
Frecuencia:78 Hz, amplitud: 83.9625
Frecuencia:78 Hz, amplitud: 69.4657
Frecuencia:78 Hz, amplitud: 82.1423
Frecuencia:78 Hz, amplitud: 69.5005
Frecuencia:1482 Hz, amplitud: 85.8601
Frecuencia:1521 Hz, amplitud: 62.6127
Frecuencia:78 Hz, amplitud: 103.37
Frecuencia:1521 Hz, amplitud: 76.5585
Frecuencia:117 Hz, amplitud: 124.288
Frecuencia:1521 Hz, amplitud: 84.152
```

```
Frecuencia:663 Hz, amplitud: 248.788
Frecuencia:702 Hz, amplitud: 543.847
Frecuencia:741 Hz, amplitud: 232.447
Frecuencia:663 Hz, amplitud: 271.153
Frecuencia:702 Hz, amplitud: 393.507
Frecuencia:624 Hz, amplitud: 225.706
Frecuencia:624 Hz, amplitud: 221.516
Frecuencia:663 Hz, amplitud: 409.826
Frecuencia:702 Hz, amplitud: 535.174
Frecuencia:663 Hz, amplitud: 213.912
Frecuencia:702 Hz, amplitud: 294.072
Frecuencia:741 Hz, amplitud: 386.944
Frecuencia:741 Hz, amplitud: 550.011
Frecuencia:780 Hz, amplitud: 455.076
Frecuencia:741 Hz, amplitud: 339.285
Frecuencia:780 Hz, amplitud: 599.835
Frecuencia:819 Hz, amplitud: 318.16
Frecuencia:780 Hz, amplitud: 332.639
Frecuencia:819 Hz, amplitud: 519.767
Frecuencia:819 Hz, amplitud: 413.14
no detectada A-AA-ABB-A-AAAAAAAAA
```

```
Frecuencia:663 Hz, amplitud: 228.52
Frecuencia:624 Hz, amplitud: 295.495
Frecuencia:663 Hz, amplitud: 303.37
Frecuencia:624 Hz, amplitud: 294.815
Frecuencia:663 Hz, amplitud: 281.568
Frecuencia:585 Hz, amplitud: 261.475
Frecuencia:624 Hz, amplitud: 315.765
Frecuencia:585 Hz, amplitud: 224.623
Frecuencia:585 Hz, amplitud: 228.463
Frecuencia:702 Hz, amplitud: 264.53
Frecuencia:702 Hz, amplitud: 387.091
Frecuencia:741 Hz, amplitud: 553.068
Frecuencia:780 Hz, amplitud: 333.886
Frecuencia:780 Hz, amplitud: 445.667
Frecuencia:819 Hz, amplitud: 339.118
Frecuencia:780 Hz, amplitud: 329.05
Frecuencia:819 Hz, amplitud: 539.183
Frecuencia:819 Hz, amplitud: 357.084
Frecuencia:858 Hz, amplitud: 252.999
Frecuencia:819 Hz, amplitud: 239.905
Sirena detectada A-B-B-BBBBAAAAAAAAAA
```

Funcionamiento final:

https://drive.google.com/drive/folders/1IapllYaWcf48EnuTXbB-4YpKm_KiIxBy

Conclusiones

Kevin Daniel Lazcano Ortiz 339006

El proyecto de detector de frecuencias de sirenas con ESP8266 ofrece una contribución significativa a la mejora de la seguridad en entornos urbanos. La capacidad de identificar y analizar las señales acústicas de sirenas de vehículos de emergencia en tiempo real brinda una herramienta valiosa para optimizar la gestión de situaciones críticas, como accidentes de tráfico o emergencias médicas. La tecnología implementada en este proyecto puede ser integrada de manera efectiva en sistemas de seguridad existentes, fortaleciendo así la respuesta ante eventos imprevistos.

Hugo Alonso Mendoza Ponce 339020

La combinación de un sistema de adquisición de audio y algoritmos de procesamiento digital de señales en el ESP8266 demuestra una eficiencia notable en la detección de frecuencias sonoras específicas de las sirenas. Esto no solo facilita la identificación temprana de vehículos de emergencia, sino que también reduce la posibilidad de falsas alarmas al analizar de manera precisa el espectro de frecuencias. La aplicación práctica de esta tecnología puede tener un impacto directo en la eficacia de los servicios de emergencia y en la seguridad general de la comunidad.

Fernando García Morales 343454

La versatilidad del ESP8266 como microcontrolador permite una fácil integración en diversos entornos y sistemas. La interfaz de usuario proporciona una experiencia intuitiva para el monitoreo en tiempo real de las frecuencias detectadas, lo que amplía las posibilidades de aplicación. Desde la gestión del tráfico hasta la integración en sistemas de seguridad de edificios, este proyecto destaca por su potencial de adaptación a diferentes escenarios, brindando una solución tecnológica robusta y escalable.

Referencias:

kosme. (2023, 24 de octubre). Fast Fourier Transform for Arduino. GitHub. https://github.com/kosme/arduinoFFT