



UNIVERSIDAD DE CUENCA

Universidad de Cuenca
Facultad de Ingeniería
Carrera de Electrónica y Telecomunicaciones

Evaluación y Comparación de Algoritmos para la Detección Automática de Ondas Sísmicas.

Trabajo de titulación previo a la obtención del título de Ingeniero en
Electrónica y Telecomunicaciones

Autor:

Alvaro Eduardo Armijos Sarango
C.I. 1104533482
alvaro.armijoss@gmail.com

Director de Tesis:

Ing. Santiago Renán González Martínez, PhD.
C.I. 0103895934

Codirector:

Ing. Iván Santiago Palacios Serrano, MSc.
C.I. 0105169429

Cuenca - Ecuador
28 de octubre de 2021

Resumen

La detección temprana de eventos sísmicos permite reducir daños materiales, el número de personas afectadas e incluso salvar vidas. En particular, la actividad sísmica en Ecuador es alta dado que se encuentra en el denominado Cinturón de Fuego del Pacífico. En tal contexto, la presente tesis tiene como objetivo la comparación de algoritmos para la detección automática de eventos sísmicos. Dicha comparación se realizó con respecto a la funcionalidad y configuración de los parámetros requeridos para cada algoritmo. Además, la implementación se llevó a cabo sobre una plataforma computacional tipo SBC (Single Board Computer) con el objetivo de obtener una herramienta portable, económica y de bajo costo computacional. Los métodos comparados son: Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA, Z-detector, Baer-and Kradolfer-picker y AR-AIC (Autoregressive-Akaike-Information-Criterion-picker). Para la evaluación y comparación se desarrollaron múltiples experimentos empleando registros sísmicos reales proporcionados por la Red Sísmica del Austro (RSA), disponibles como fuente de entrada a los algoritmos. Como resultado se obtuvo que el algoritmo Classic STA/LTA presentó el mejor rendimiento, ya que del total de eventos reales (58), solo un evento no fue detectado. Además, se determinaron 6 eventos falsos, logrando un 90 % de efectividad. Finalmente para la implementación en tiempo real, se utilizó el algoritmo Classic STA/LTA sobre una plataforma SBC (Single Board Computer). Cabe destacar que el software desarrollado para la comparación de los algoritmos y la detección en tiempo real está disponible de forma libre.

Palabras clave: SBC. Classic STA/LTA. Recursive STA/LTA. Delayed STA/LTA. Z-detector. Baer and Kradolfer-picker. AR-AIC. Eventos sísmicos.

Abstract

Early event detection can reduce the number of people affected and save lives. In particular, the seismic activity in Ecuador is high, given that it is located along the zone named the Pacific Belt of Fire. Following this purpose, this work presents a solution in order to compare algorithms for detecting seismic events. This comparison was performed both in relation to the functionality and the configuration of the parameters required for each algorithm. This solution was implemented on an SBC platform (Single Board Computer) for the purpose of obtaining a portable, economical and low-cost computational tool. The methods compared were: Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA, Z-detector, Baer-and Kradolfer-picker and AR-AIC (Autoregressive-Akaike-Information-Criterion-picker). For the evaluation and comparison, multiple experiments were carried out using real data provided by the Regional Seismological Network (RSA, Red Sísmica del Austro). In particular, such registers were used as the input source to the seismic algorithms. Results reveal, the algorithm with the best performance was the Classic STA/LTA, since of the total number of real events (58), only one event was not detected. In addition, 6 false events were obtained, achieving 90 % of effectiveness. Finally, for the real-time implementation, the Classic STA / LTA algorithm is used on an SBC (Single Board Computer) platform. The software developed for algorithm comparison and real-time detection has been released for free usage which represents another contribution of this work in the context of seismic analysis.

Keywords:: SBC. Classic STA/LTA. Recursive STA/LTA. Delayed STA/LTA. Z-detector. Baer and Kradolfer-picker. AR-AIC. Seismic events.

Índice General

Lista de Figuras	VI
Lista de Tablas	VIII
Acrónimos	XII
1 Introducción	1
1.1 Definición del problema	1
1.2 Justificación y alcance	1
1.3 Objetivos	2
1.3.1 Objetivo general	2
1.3.2 Objetivos específicos	2
1.4 Contribuciones	2
2 Sustento teórico	3
2.1 Conceptos de sismología	3
2.1.1 Ondas de cuerpo	3
2.1.1.1 Ondas P	3
2.1.1.2 Ondas S	3
2.1.2 Ondas superficiales	3
2.1.2.1 Ondas Rayleigh	4
2.1.2.2 Ondas Love	4
2.1.3 Magnitud	4
2.1.4 Intensidad	7
2.1.5 Energía	7
2.1.6 Ruido sísmico	7
2.2 Métodos de detección automática de eventos sísmicos	7
2.2.1 Método STA/LTA (Short Time Average / Long Time Average)	7
2.2.1.1 STA	8
2.2.1.2 LTA	8
2.2.1.3 Umbral de activación	8
2.2.1.4 Umbral de desactivación	8
2.2.2 Z Detector	9
2.2.3 Baer and Kradolfer picker	10
2.2.4 AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)	11
2.3 Técnicas de procesamiento de señales	11
2.3.1 Filtrado de señales	11
2.3.1.1 Respuesta al impulso, frecuencia y de fase de un filtro	11



2.3.1.2	Tipos de filtros	12
2.3.1.3	Filtro de Butterworth	13
2.4	Formatos de datos sísmicos	13
2.4.1	SEED	14
2.4.2	SAC	14
2.4.3	GSE	14
2.4.4	IRIS	15
2.5	Estación sísmica	15
3	Estado del arte	16
3.1	Algoritmos de detección automática de eventos sísmicos	16
3.2	Detección de eventos en tiempo real	17
3.3	Nuevos enfoques	17
3.4	Software para el análisis sísmico	19
4	Detección y procesamiento de eventos sísmicos	20
4.1	La librería ObsPy	20
4.1.1	Instalación y configuración	20
4.1.2	Métodos de detección disponibles en la librería ObsPy	21
4.1.2.1	Classic STA/LTA	21
4.1.2.2	Recursive STA/LTA	22
4.1.2.3	Delayed STA/LTA	23
4.1.2.4	Z detector	24
4.1.2.5	Baer and Kradolfer picker	24
4.1.2.6	AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)	25
4.2	Diseño de una herramienta para el análisis de eventos	25
4.2.1	Características y funcionalidades	25
4.2.2	Selección de los métodos	28
4.2.3	Visualización y presentación de resultados	28
4.3	Comparación de algoritmos de detección	28
4.3.1	Metodología	29
4.3.2	Análisis de parámetros y métricas	31
4.3.3	Configuración de métodos de detección	31
4.3.3.1	Classic STA/LTA	31
4.3.3.2	Delayed STA/LTA	32
4.3.3.3	Recursive STA/LTA	32
4.3.3.4	Z Detector	32
4.3.3.5	Baer and Kradolfer picker	33
4.3.3.6	AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)	33
4.3.4	Procesamiento de eventos reales mediante los métodos de detección	34
4.3.4.1	Classic STA/LTA	34
4.3.4.2	Delayed STA/LTA	35
4.3.4.3	Recursive STA/LTA	37
4.3.4.4	Z Detector	38
4.3.4.5	Baer and Kradolfer picker	40
4.3.4.6	AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)	41



4.3.5	Análisis de resultados	42
4.4	Eficiencia computacional	43
4.5	Conclusiones	43
5	Detección de eventos en tiempo real	45
5.1	Selección del algoritmo de detección	45
5.2	Diseño de una herramienta para el análisis de eventos en tiempo real	46
5.2.1	Arquitectura	46
5.2.2	Características y funcionalidades	47
5.2.3	Visualización y presentación de resultados	48
5.3	Eficiencia computacional	49
5.4	Conclusiones	50
	Conclusiones	51
6.1	Conclusiones	51
6.2	Recomendaciones	51
6.3	Trabajos futuros	52
	Bibliografía	53
A	Manual de usuario de la herramienta para la comparación de algoritmos	57
A.1	Introducción	57
A.2	Primeros pasos	57
A.2.1	Prerrequisitos	57
A.2.2	Descarga de la herramienta	57
A.2.3	Compilación	57
A.2.4	Recomendación	57
A.3	Métodos disponibles	58
A.3.1	Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA y Z Detector	58
A.3.2	Baer and Kradolfer picker	58
A.3.3	AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)	59
A.4	Visualización y presentación de resultados	59
B	Manual de usuario de la herramienta en tiempo real	65
B.1	Introducción	65
B.2	Primeros pasos	65
B.2.1	Prerrequisitos	65
B.2.2	Descarga de la herramienta	65
B.2.3	Compilación	65
B.2.4	Recomendación	66
B.3	Classic STA/LTA	66
B.4	Visualización y presentación de resultados	66

Lista de Figuras

2.1.1	Representación de la propagación de la onda P en a) resorte y b) tierra [1]	4
2.1.2	Representación de la propagación de la onda S en a) cuerda y b) tierra [1]	4
2.1.3	Representación del movimiento de las ondas Rayleigh [1]	5
2.1.4	Representación del movimiento de las ondas Love [1]	5
2.2.1	Variables del algoritmo STA/LTA	8
2.2.2	Ejemplo del algoritmo Z Detector	9
2.2.3	Ejemplo del algoritmo Baer y Kradolfer	10
2.2.4	Ejemplo de Baer y Kradolfer [2]	10
2.2.5	Ejemplo del método AR-AIC [2]	11
2.3.1	Tipos de filtros.	13
2.3.2	Respuesta del filtro Butterworth	13
2.5.1	Estaciones instaladas en la Red Sísmica del Austro	15
4.1.1	Diagrama de la instalación de la librería ObsPy en la Raspberry Pi	21
4.2.1	Diagrama de clases de la herramienta	27
4.2.2	Características y funcionalidades de la herramienta	27
4.2.3	Casos de uso de la herramienta	28
4.3.1	Metodología	29
4.3.2	Ejemplo de la aplicación de filtros a la señal	30
4.3.3	Resultados obtenidos con el algoritmo Classic STA/LTA	34
4.3.4	Gráfica de los eventos obtenidos con el mejor resultado en el algoritmo Classic STA/LTA	35
4.3.5	Resultados obtenidos con el algoritmo Delayed Sta/Lta	36
4.3.6	Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Delayed Sta/Lta	36
4.3.7	Resultados obtenidos con el algoritmo Recursive STA/LTA	38
4.3.8	Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Recursive STA/LTA	38
4.3.9	Resultados obtenidos con el algoritmo Z Detector	39
4.3.10	Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Z Detector	39
4.3.11	Resultados obtenidos con el algoritmo Baer and Kradolfer picker	40
4.3.12	Gráfica de uno de los eventos obtenidos con el algoritmo Baer and Kradolfer picker	41
4.3.13	Resultados obtenidos con el algoritmo AR-AIC	42
4.3.14	Gráfica de uno de los eventos obtenidos con el algoritmo AR-AIC	42
4.4.1	Uso de CPU en la herramienta de comparación de algoritmos	43



5.1.1	Evento sísmico obtenido con el algoritmo Classic STA/LTA	46
5.2.1	Arquitectura del sistema	47
5.2.2	Diagrama de clases de la herramienta en tiempo real	47
5.2.3	Características y funcionalidades de la herramienta en tiempo real	48
5.2.4	Casos de uso de la herramienta en tiempo real	48
5.2.5	Gráfica de un evento sísmico detectado en tiempo real	49
5.3.1	Uso de CPU en la herramienta en tiempo real	50
A.3.1	Herramienta al momento de ejecutar	58
A.3.2	Interfaz gráfica para métodos STA/LTA	58
A.3.3	Interfaz gráfica para el método Baer and Kradolfer picker	58
A.3.4	Interfaz gráfica para el método AR-AIC	59
A.4.1	Interfaz sin parámetros	59
A.4.2	Seleccionar archivo	60
A.4.3	Mensaje de advertencia cuando se tiene parámetros vacíos	60
A.4.4	Gráfica de eventos	61
A.4.5	Guardar gráfica de eventos	61
A.4.6	Exportar eventos	62
A.4.7	Opciones para generar archivos miniSeed	62
A.4.8	Mensaje de confirmación para generar archivos miniSeed	63
A.4.9	Mensaje de confirmación luego de extraer un archivo miniSeed	63
A.4.10	Resultado del algoritmo AR-AIC	64
A.4.11	Resultado del algoritmo Baer and Kradolfer picker	64
B.3.1	Interfaz gráfica para el método Classic STA/LTA	66
B.4.1	Interfaz sin parámetros	67
B.4.2	Conexión al servidor	68
B.4.3	Llegada de datos sísmicos	68
B.4.4	Llegada datos sísmicos y buffer completo	69
B.4.5	Detección de eventos sísmicos	69
B.4.6	Registros sísmicos luego de un evento	70
B.4.7	Guardar gráficas	70
B.4.8	Desconectarse del servidor	71

Lista de Tablas

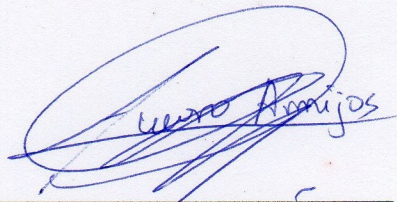
2.1.1	Escala Richter.	6
2.1.2	Escala de Mercalli Modificada.	6
4.1.1	Parámetros del algoritmo Classic Sta/Lta	22
4.1.2	Parámetros del algoritmo Recursive STA/LTA	23
4.1.3	Parámetros del algoritmo Delayed STA/LTA	23
4.1.4	Parámetros del algoritmo Z-detector	24
4.1.5	Parámetros del algoritmo Baer-and Kradolfer-picker	24
4.1.6	Parámetros del algoritmo AR-AIC	25
4.3.1	Parámetros del algoritmo Classic STA/LTA en cada prueba	32
4.3.2	Parámetros del algoritmo Delayed STA/LTA en cada prueba	32
4.3.3	Parámetros del algoritmo Recursive STA/LTA en cada prueba	32
4.3.4	Parámetros del algoritmo Z Detector en cada prueba	33
4.3.5	Parámetros del algoritmo Baer and Kradolfer picker en cada prueba	33
4.3.6	Parámetros del algoritmo AR-AIC en cada prueba	33
4.3.7	Resultados del algoritmo Classic STA/LTA	34
4.3.8	Resultados del algoritmo Delayed STA/LTA	35
4.3.9	Resultados del algoritmo Recursive STA/LTA	37
4.3.10	Resultados del algoritmo Z Detector	39
4.3.11	Resultados del algoritmo Baer and Kradolfer picker	40
4.3.12	Resultados del algoritmo AR-AIC	41
5.1.1	Mejores resultados de cada algoritmo	45

**Cláusula de licencia y autorización
para publicación en el Repositorio Institucional**

Alvaro Eduardo Armijos Sarango en calidad de autor y titular de los derechos morales y patrimoniales de trabajo de titulación Evaluación y Comparación de Algoritmos para la Detección Automática de Ondas Sísmicas, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fin estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 28 de octubre de 2021

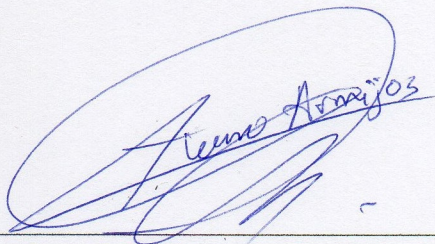


Alvaro Eduardo Armijos Sarango
C.I. 1104533482

Cláusula de Propiedad Intelectual

Alvaro Eduardo Armijos Sarango, autor del trabajo de titulación Evaluación y Comparación de Algoritmos para la Detección Automática de Ondas Sísmicas, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 28 de octubre de 2021



Alvaro Eduardo Armijos Sarango
C.I. 1104533482



Agradecimientos

A mi familia por todo el apoyo que me han brindado, los ánimos cuando hizo falta y la fortaleza que me transmiten para continuar adelante. Y sobre todo por siempre estar ahí cuando los necesitaba.

A todos los profesores que con su paciencia y grandes conocimientos han formado parte integral de mi educación, gracias por transmitirme todos sus conocimientos, valores y enseñarme que a pesar de que el camino no es fácil vale totalmente la pena.

A la Red Sísmica del Austro por todo el apoyo brindado para la realización de esta propuesta de titulación.

Un especial agradecimiento al Ing. Santiago Renán González Martínez, PhD, y al Ing. Iván Santiago Palacios Serrano, MSc. por todo el tiempo dedicado para apoyar el desarrollo de esta tesis.

Alvaro Eduardo Armijos Sarango

Acrónimos

ADC Analog-to-Digital Converter. [29](#)

ANN Artificial Neural Network. [18](#)

AR Autoregressive Process. [11](#)

AR-AIC Autoregressive-Akaike-Information-Criterion-picker. [16](#)

BER Balanced Error Rate. [17](#)

BP Band Pass. [12](#)

CF Characteristic Functions. [17](#)

CNN Convolutional Neural Network. [18](#)

CPIC Phase-Identification Classifier. [18](#)

FDSN Federation of Digital Seismographic Networks. [14](#)

GSE Group of Scientific Experts. [13](#), [14](#)

GSN Global Seismograph Network. [14](#)

HP High Pass. [12](#)

IP Internet Protocol. [46](#)

LP Low Pass. [12](#)

LP Long Period. [17](#)

MF Matched Filter. [17](#)

MLP Multilayer Perceptron. [18](#)

MW Moment magnitude. [5](#)

NCC Normalized Cross-Correlation. [17](#)

NOTCH Notch. [12](#)



- PBEE** Performance-based earthquake engineering. [19](#)
- PSD** Power Spectral Density. [17](#)
- RBF** Base Radial Function. [18](#)
- RMSE** Root-Mean-Square Error. [18](#)
- SAC** Seismic Analysis Code. [13](#), [14](#)
- SBC** Single Board Computer. [29](#)
- SD** Subspace Detection. [17](#)
- SEED** Standard for the Exchange of Earthquake Data. [13](#), [14](#)
- SNR** Signal-to-Noise Ratio. [17](#)
- STA/LTA** Short-Time-Average Through Long-Time-Average Trigger. [16–18](#), [21–23](#)
- TCP** Transmission Control Protocol. [46](#)
- VT** Volcano Tectonic. [17](#)
- WCEDS** Waveform Correlation Event-Detection and Location System. [17](#)

Capítulo 1

Introducción

1.1. Definición del problema

Los movimientos sísmicos son una causa importante de destrucción de vida y propiedad. Acorde a los registros y reportes disponibles, en Ecuador, el historial sísmico da cuenta de eventos a lo largo de cinco siglos en la región Sierra y al menos un siglo para la zona del Litoral [3]. Si bien eventos de magnitudes considerables han tenido lugar en la región del Litoral debido a la zona de subducción de las placas continentales, es en la región de la Sierra donde se han producido las mayores devastaciones, como es el caso de los terremotos acontecidos en las ciudades de Riobamba (1797) y Ambato (1949), como se describe en [3].

En la Región Andina, las principales fuentes sísmicas son producto de la actividad volcánica [4], así como de las numerosas fallas geológicas existentes [5]. En particular, en la provincia del Azuay existen cuatro fallas geológicas (Paute, Girón, Gualaceo y Tarqui) y un historial de al menos cuatro sismos importantes. El más significativo ocurrió en 1887 y afectó considerablemente ciertas edificaciones de la ciudad de Cuenca. Dicho sismo fue de origen superficial y en una de las fallas geológicas cercanas [6].

Por lo tanto, el daño causado por un sismo depende del grado de preparación y de la capacidad de respuesta de la sociedad. En concreto, la detección de eventos es fundamental para mitigar los detrimentos ocasionados por los sismos. Sin embargo, la detección automática de eventos sísmicos, es un desafío por el ruido sísmico existente en las señales capturadas por las estaciones, como se describe en [7].

En tal contexto, cabe indicar que la presente tesis se enmarca dentro de las tareas de investigación que se llevan a cabo en el Proyecto “Tecnologías IoT y Redes Inalámbricas de Sensores aplicados a la Monitorización de Salud Estructural en Edificios Esenciales de la Ciudad de Cuenca”, en el cual participa personal de la RSA así como del DEET (Departamento de Ingeniería Eléctrica, Electrónica y Telecomunicaciones).

1.2. Justificación y alcance

La detección automática de eventos sísmicos es una de las tareas fundamentales durante el procesamiento de las señales adquiridas por estaciones sísmicas (v.g. sismógrafos, acelerógrafos). En particular, dicha información es crucial para el desarrollo de sistemas de monitorización de salud estructural.

En tal contexto, la RSA adscrita a la Facultad de Ingeniería monitorea poblaciones en el Sur del Ecuador. Para tal objetivo, disponen de múltiples estaciones sísmicas de geófonos y acelerógrafos ubicadas en distintos sitios de la región. Además, la transmisión de los datos de los sensores se realiza en tiempo real mediante comunicación por radiofrecuencia. El procesamiento de esta información sigue siendo ambiguo y el personal a cargo debe revisarla de forma manual para obtener datos sobre eventos sísmicos. Este procesamiento manual conlleva un tiempo considerable de trabajo humano y es propenso a errores, por lo que resulta de gran interés el desarrollo de soluciones que permitan automatizar el análisis.

Por lo tanto, el presente tema de tesis se enfoca en realizar una evaluación y comparación de diferentes algoritmos empleados para la detección automática de ondas sísmicas tipo P y S. En concreto, se propone evaluar y comparar los siguientes algoritmos: Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA, Z-detector, Baer- and Kradolfer-picker y AR-AIC disponibles en la librería ObsPy que consiste en un proyecto



open source desarrollado en el lenguaje de programación Python. Finalmente, a partir de la evaluación se elegirá el algoritmo que presente mejores resultados y se realizará la implementación del mismo en tiempo real sobre una plataforma SBC (Single Board Computer)

1.3. Objetivos

1.3.1. Objetivo general

Evaluar y comparar algoritmos de detección automática de eventos sísmicos aplicando técnicas de procesamiento de señales.

1.3.2. Objetivos específicos

- Diseñar e implementar una herramienta para el análisis de eventos sísmicos empleando diferentes algoritmos de detección.
- Determinar los parámetros óptimos para cada uno de los algoritmos, en función de los datos disponibles.
- Definir métricas de comparación de algoritmos de detección de eventos sísmicos
- Implementar en tiempo real, en la plataforma SBC, el algoritmo que presente mejores resultados.

1.4. Contribuciones

Las principales contribuciones del presente trabajo de tesis son:

- Evaluación y comparación de algoritmos para la detección automática de ondas sísmicas
- Herramienta para el análisis de eventos sísmicos empleando diferentes algoritmos de detección
- Herramienta para la detección de eventos en tiempo real
- Artículo científico: “Evaluación y comparación de algoritmos para la detección automática de eventos sísmicos”

Capítulo 2

Sustento teórico

En este capítulo se describen los principales conceptos relacionados con el presente trabajo de tesis, tales como fundamentos de sismología, técnicas de procesamiento de señales, formatos empleados en los sistemas de registro continuo así como instrumentación sísmica.

2.1. Conceptos de sismología

Un sismo es un fenómeno que se produce por la liberación de energía acumulada en forma de ondas sísmicas, lo que provoca una sacudida pasajera de la corteza terrestre. Las ondas sísmicas se clasifican en ondas de cuerpo o internas y ondas superficiales. Las ondas internas son aquellas que se propagan desde su origen hasta la superficie de la tierra y se subdividen en ondas P (ondas primarias) y ondas S (ondas secundarias). Por otra parte, las ondas superficiales son las que se propagan sobre la superficie de la tierra, y se subdividen en ondas Rayleigh y ondas Love.

2.1.1. Ondas de cuerpo

Las ondas de cuerpo llegan antes que las ondas superficiales. Estas viajan a través de la parte interna hasta llegar a la superficie de la tierra. Hay dos tipos diferentes de ondas corporales: ondas P y ondas S, ver [7].

2.1.1.1. Ondas P

Las ondas P (ondas primarias) se denominan así porque son las primeras en llegar a la superficie de la Tierra y por ende las primeras en ser registradas por las estaciones sísmicas. Su velocidad de propagación es de aproximadamente unos 7,5 kilómetros por segundo. Como se observa en la Figura 2.1.1 las ondas P son ondas longitudinales que se propagan produciendo oscilaciones en el mismo sentido en el cual se propagan [8].

2.1.1.2. Ondas S

Las ondas S (ondas secundarias) llegan a la superficie de la tierra después de las ondas P, en segundo lugar. Las ondas S tienen una velocidad de propagación de alrededor de 4,2 kilómetros por segundo. Las ondas S son ondas transversales, como se puede apreciar en la Figura 2.1.2. Las mismas se transmiten produciendo movimientos perpendiculares a la dirección en que se propagan, a través del material en que se transmiten.

2.1.2. Ondas superficiales

Las ondas superficiales viajan a través de la corteza. Tienen una frecuencia más baja que las ondas de cuerpo. En consecuencia, son más lentas que las ondas de cuerpo, pero causan mayor daño y destrucción. Hay dos tipos diferentes de ondas superficiales: Ondas Love y Ondas Rayleigh [7].

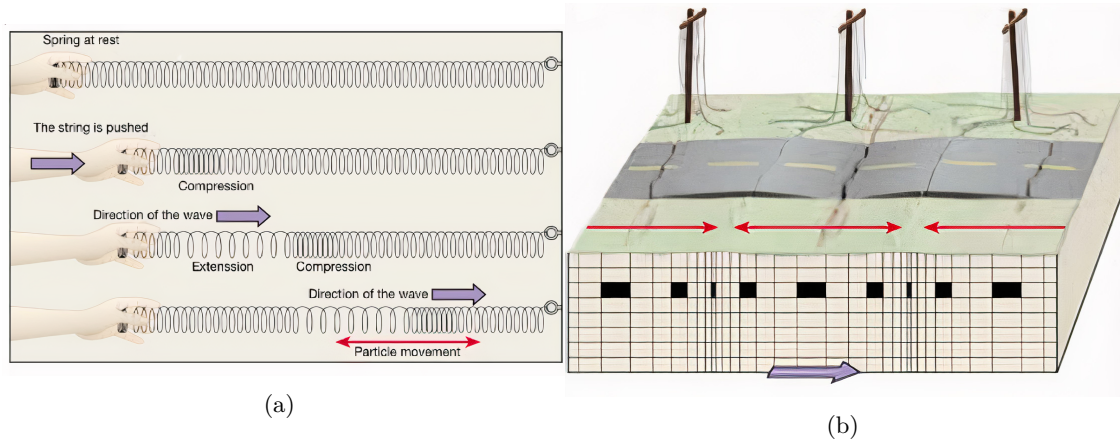


Figura 2.1.1: Representación de la propagación de la onda P en a) resorte y b) tierra [1]

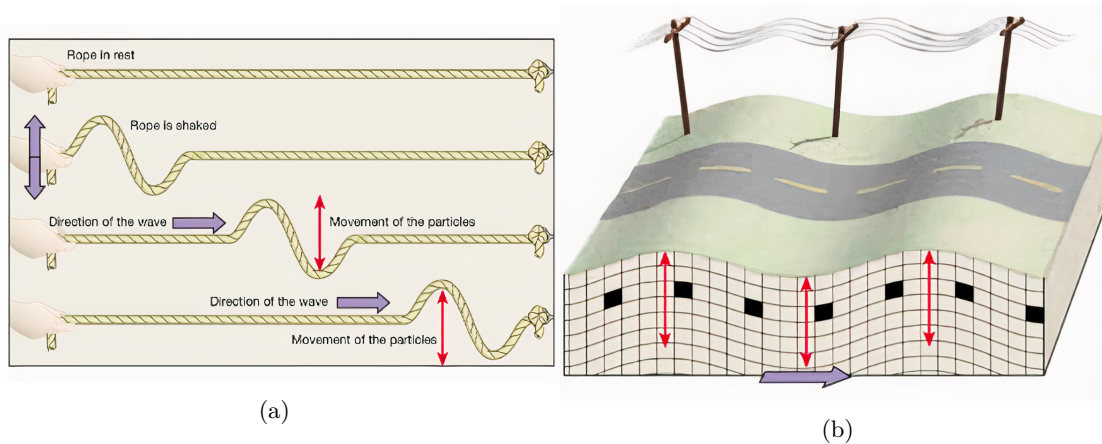


Figura 2.1.2: Representación de la propagación de la onda S en a) cuerda y b) tierra [1]

2.1.2.1. Ondas Rayleigh

Las ondas Rayleigh, son ondas superficiales que ruedan por el suelo y tienen un movimiento vertical similar al de las olas de mar, ya que mueven el suelo hacia arriba y hacia abajo y de lado a lado en la misma dirección en que se propaga la onda, esto se puede apreciar en la Figura 2.1.3. La existencia de estas ondas fue predicha por John William Strutt, Lord Rayleigh, en 1885. Son ondas más lentas que las ondas de cuerpo y su velocidad de propagación es casi un 90 % de la velocidad de las ondas S [8].

2.1.2.2. Ondas Love

Las ondas Love son ondas superficiales que producen un movimiento horizontal de corte en la superficie (2.1.4). Se denominan así en honor al matemático Augustus Edward Hough Love del Reino Unido. La velocidad de las ondas Love es un 90 % de la velocidad de las ondas S y es ligeramente superior a la velocidad de las ondas Rayleigh.

2.1.3. Magnitud

El concepto de magnitud fue introducido en 1935 por Charles Francis Richter, para medir los terremotos locales y así poder estimar la energía liberada a fin de ser comparados con otros terremotos. La magnitud está asociada a una función logarítmica calculada a partir de la amplitud de la señal registrada por el sismógrafo (ML, Ms, mb) o a partir de su duración (MD) sobre el sismograma [8]. Generalmente, en

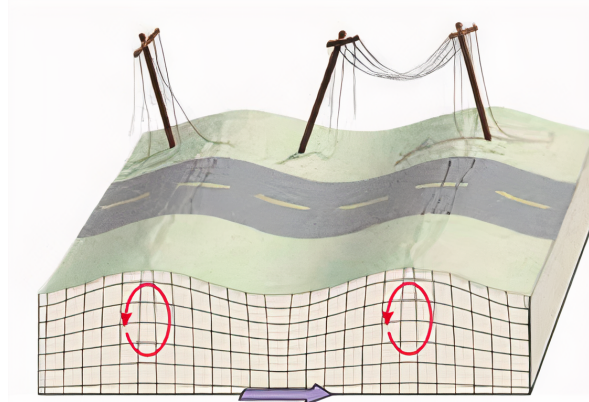


Figura 2.1.3: Representación del movimiento de las ondas Rayleigh [1]

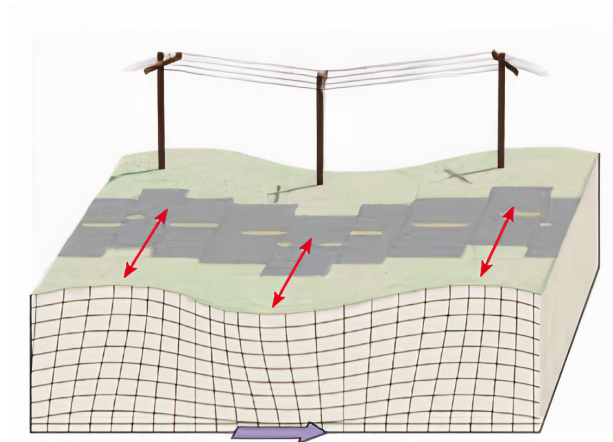


Figura 2.1.4: Representación del movimiento de las ondas Love [1]

diferentes partes del mundo se usa la **Magnitud Richter** ya que esta escala ha sido calibrada para ser empleada con registros de diferentes instrumentos. En la Tabla 2.1.1 se muestran las escalas de la Magnitud Richter. Sin embargo, la tendencia actual es calcular la magnitud a partir del momento sísmico (**Moment magnitude (MW)**) del terremoto. Esta magnitud se determina a partir del momento sísmico, que es una cantidad proporcional al área de ruptura y al deslizamiento que ocurra en la falla, su estimación es compleja y puede llevarse a cabo empleando diversos métodos y tipos de datos.



Tabla 2.1.1: Escala Richter.

Magnitud, escala Richter	Efectos del sismo o terremoto
Menos de 3,5	Generalmente no se siente, pero es registrado.
3,5-5,4	A menudo se siente, pero solo causa daños menores.
5,5-6,0	Ocasiona daños ligeros a edificios.
6,1-6,9	Puede ocasionar daños severos en áreas donde vive mucha gente.
7,0-7,9	Terremoto mayor. Causa graves daños
8 o mayor	Gran terremoto. Destrucción total a comunidades cercanas.

Tabla 2.1.2: Escala de Mercalli Modificada.

Grado	Descripción
I	Sacudida sentida por muy pocas personas en condiciones especialmente favorables.
II	Sacudida sentida sólo por pocas personas en reposo, especialmente en los pisos altos de los edificios. Los objetos suspendidos pueden oscilar.
III	Sacudida sentida claramente en los interiores, especialmente en los pisos altos de los edificios, muchas personas no lo asocian con un temblor. Los vehículos de motor estacionados pueden moverse ligeramente. Vibración como la originada por el paso de un carro pesado. Duración estimable.
IV	Sacudida sentida durante el día por muchas personas en los interiores, por pocas en el exterior. Por la noche algunas despiertan. Vibración de vajillas, vidrios de ventanas y puertas; los muros crujen. Sensación como de un carro pesado chocando contra un edificio, los vehículos de motor estacionados se balancean claramente.
V	Sacudida sentida casi por todo el mundo; muchos despiertan. Algunas piezas de vajillas, vidrios de ventanas, etcétera, se rompen; pocos casos de agrietamiento de aplanados; caen objetos inestables. Se observan perturbaciones en los árboles, postes y otros objetos altos. Se detienen relojes de péndulo.
VI	Sacudida sentida por todo mundo; muchas personas atemorizadas huyen hacia afuera. Algunos muebles pesados cambian de sitio; pocos ejemplos de caída de aplanados o daño en chimeneas. Daños ligeros.
VII	Advertido por todos. La gente huye al exterior. Daños sin importancia en edificios de buen diseño y construcción. Daños ligeros en estructuras ordinarias bien construidas; daños considerables en las débiles o mal planeadas; ruptura de algunas chimeneas. Estimado por las personas conduciendo vehículos en movimiento.
VIII	Daños ligeros en estructuras de diseño especialmente bueno; considerable en edificios ordinarios con derrumbe parcial; grande en estructuras débilmente construidas. Los muros salen de sus armaduras. Caída de chimeneas, pilas de productos en los almacenes de las fábricas, columnas, monumentos y muros. Los muebles pesados se vuelcan. Arena y lodo proyectados en pequeñas cantidades. Cambio en el nivel del agua de los pozos. Pérdida de control en las personas que guían carros de motor.
IX	Daño considerable en las estructuras de diseño bueno; las armaduras de las estructuras bien planeadas se desploman; grandes daños en los edificios sólidos, con derrumbe parcial. Los edificios salen de sus cimientos. El terreno se agrieta notablemente. Las tuberías subterráneas se rompen.
X	Destrucción de algunas estructuras de madera bien construidas; la mayor parte de las estructuras de mampostería y armaduras se destruyen con todo y cimientos; agrietamiento considerable del terreno. Las vías del ferrocarril se tuercen. Considerables deslizamientos en las márgenes de los ríos y pendientes fuertes. Invasión del agua de los ríos sobre sus márgenes.
XI	Casi ninguna estructura de mampostería queda en pie. Puentes destruidos. Anchas grietas en el terreno. Las tuberías subterráneas quedan fuera de servicio. Hundimientos y derrumbes en terreno suave. Gran torsión de vías férreas.
XII	Destrucción total. Ondas visibles sobre el terreno. Perturbaciones de las cotas de nivel. Objetos lanzados en el aire hacia arriba.

2.1.4. Intensidad

La intensidad no permite medir el movimiento del suelo, pero sí los efectos que ellos producen en la superficie en donde causan daños al hombre y a las construcciones. A fin de no confundir magnitud e intensidad, dos terremotos de igual magnitud pueden generar en superficie intensidades máximas muy diferentes [8].

La escala de intensidad que se usa actualmente se llama escala de Mercalli Modificada que consta de doce grados. Los valores bajos por lo general están asociados con la forma como las personas sintieron el sismo, mientras que valores mayores con la forma como fue afectado el paisaje o las construcciones hechas por el hombre. La Tabla 2.1.2 define los grados de la escala de intensidad de Mercalli.

Para entender mejor la diferencia entre magnitud e intensidad, se tiene por ejemplo que el terremoto de Pedernales-Ecuador tuvo una magnitud de momento M_w 7,8 y una intensidad máxima de IX. El terremoto ocasionó una gran cantidad de víctimas y una extensa destrucción especialmente en la costa norte y centro del Ecuador. Es así, como consecuencia directa del terremoto y según los datos oficiales emitidos por las autoridades, se contabilizaron alrededor de 700 personas fallecidas, más de 7000 heridos, 22000 personas refugiadas, millares de edificaciones destruidas o inhabitables y pérdidas económicas estimadas en alrededor de tres mil millones de dólares [9].

2.1.5. Energía

La energía total liberada por un terremoto es difícil de calcular con precisión, debido a que es la suma de la energía disipada en forma térmica por la deformación en la zona de ruptura y la energía emitida como ondas sísmicas, la única que puede ser estimada a partir de los sismogramas [8].

2.1.6. Ruido sísmico

El ruido sísmico es un conjunto de pequeñas vibraciones que pueden ser ocasionadas por la superficie de la tierra o por la actividad humana. Estas vibraciones se introducen en las señales que obtienen las estaciones sísmicas y afectan su desempeño.

2.2. Métodos de detección automática de eventos sísmicos

Existen varios algoritmos para la detección de eventos sísmicos. Los algoritmos que usan métodos de energía son los más utilizados, se basan en el estudio del cuadrado de la velocidad de partícula (STA/LTA). También están los métodos que hacen uso de la morfología de las ondas, como por ejemplo el algoritmo Z detector [10]. Además, existen algoritmos de selección. Un algoritmo de selección ampliamente utilizado es el propuesto por Baer y Kradolfer [11]. Otra forma de detectar el inicio de la onda P es utilizar estadística de orden superior, como es el caso del algoritmo Autoregressive-Akaike-Information-Criterion-picker (AR-AIC) que fue propuesto por Sleeman y Van Eck [12].

2.2.1. Método STA/LTA (Short Time Average / Long Time Average)

El algoritmo STA/LTA detecta sismos comparando los niveles de movimiento del suelo a corto plazo con los niveles de movimiento del suelo a largo plazo, de esta manera se mejora significativamente el registro de sismos de baja magnitud en comparación con los algoritmos de activación de umbral de amplitud. También reduce el número de registros falsos provocados por el ruido sísmico natural y provocado por el hombre [7].

El método STA/LTA requiere cuatro variables diferentes, una ventana de tiempo corto promedio (STA) que mide la amplitud instantánea de una señal sísmica buscando posibles sismos, una ventana de tiempo largo promedio (LTA) que se encarga de la amplitud promedio de ruido sísmico actual [13], un umbral de activación y un umbral de desactivación o *detrigger*. Todas las variables se ilustran en la Figura 2.2.1.

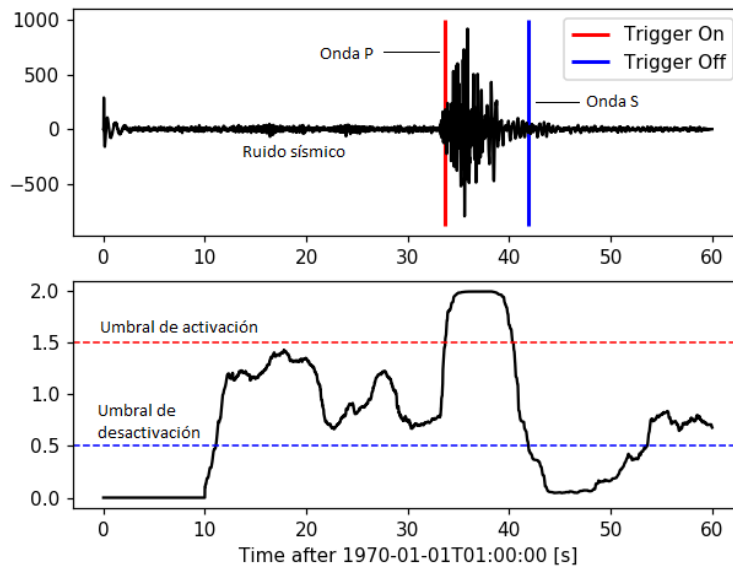


Figura 2.2.1: Variables del algoritmo STA/LTA

2.2.1.1. STA

La ventana de tiempo corto promedio mide el valor instantáneo de una señal sísmica. Normalmente, la ventana STA debe ser más larga que algunos períodos de una señal sísmica típicamente esperada. Si la ventana STA es demasiado corta, el promedio de la señal sísmica no será preciso, ya que se verá influenciado por períodos individuales de la señal sísmica. Por otro lado, la longitud de la ventana STA debe ser más corta que la duración más corta del evento que se quiere capturar. Cuanto más corta sea la duración de la ventana, mayor será la sensibilidad de activación a los sismos locales en comparación con los sismos distantes de larga duración y de menor frecuencia. Por lo tanto, al cambiar la longitud de la ventana STA, se puede priorizar la captura de eventos distantes o locales [7].

2.2.1.2. LTA

La ventana de tiempo largo promedio mide el ruido sísmico de amplitud promedio. La ventana debe ser más larga que algunos períodos de fluctuaciones de ruido sísmico típicamente irregulares. Una ventana LTA corta permite que el valor de LTA se adapte más o menos a la amplitud que aumenta lentamente de las ondas sísmicas. Por lo tanto, la relación STA/LTA sigue siendo pequeña a pesar del aumento de STA. Como consecuencia, una ventana de LTA corta disminuye la sensibilidad del disparador a eventos distantes con una gran distancia epicentral. Por otro lado, el uso de una ventana de LTA larga aumenta la sensibilidad del disparador a los sismos emergentes, ya que el valor de LTA no se ve tan rápidamente influenciado por la señal emergente [13].

2.2.1.3. Umbral de activación

El umbral de activación determina qué eventos se registrarán y cuáles no, ya que establece el límite de la relación STA/LTA sobre el que se produce la activación. Por un lado, cuanto mayor sea el valor que se establece, más sismos se perderán, pero se producirán menos eventos falsos. Por otra parte, cuanto más bajo se establezca el valor del umbral de activación de STA/LTA, más eventos se registrarán [13].

2.2.1.4. Umbral de desactivación

Cuando la relación STA/LTA alcanza el umbral de activación, se evita que el sistema se active de nuevo hasta que la relación STA/LTA caiga por debajo del umbral de desactivación, estableciendo el nivel por

debajo del cual se restablece la rutina de activación. Cuando el valor del *dettrigger* es demasiado alto, la rutina de activación se reinicia con mucha frecuencia y, como resultado, un solo evento puede producir varios activadores. Por otro lado, si el valor de *dettrigger* es demasiado bajo, el disparador puede estar activo durante mucho tiempo, evitando que el sistema vuelva a dispararse [13].

Para una mayor comprensión de estas variables, en la Figura 2.2.1 se muestra un ejemplo donde se nota claramente que el disparador se activa cuando el valor de la relación STA/LTA excede 1,5 y se desactiva cuando el valor de la relación STA/LTA cae por debajo de 0,5. De igual forma se detecta la llegada de la onda P y onda S.

El método STA/LTA calcula la amplitud absoluta de cada muestra de datos de una señal entrante. A continuación, calcula el promedio de amplitudes absolutas en ambas ventanas. En un paso adicional, calcula una relación de ambos valores (relación STA/LTA). Esta relación se compara continuamente con un valor de umbral seleccionado por el usuario: nivel de umbral de activación STA/LTA. Si la relación supera este umbral, se registra el evento sísmico. Cuando la relación STA/LTA actual cae por debajo de otro parámetro seleccionado por el usuario: el nivel de umbral de desactivación del disparador STA/LTA, el registro del evento sísmico se detiene [14]. Se han desarrollado diferentes variaciones del algoritmo STA/LTA [15], como el STA/LTA recursivo y el STA/LTA retardado.

2.2.2. Z Detector

El algoritmo Z Detector es una modificación del método STA/LTA. Como se discute en [16], las fluctuaciones de ruido de STA pueden ocasionar falsas alarmas y por tanto inestabilidad de los métodos convencionales. En algunos casos, conduce al funcionamiento inestable de los métodos convencionales. En algunos casos, tal inestabilidad hace que el detector se encienda o se apague durante largos períodos de tiempo. Z Detector fue diseñado para resolver este problema ajustando continuamente el umbral de STA/LTA a un número fijo de desviaciones estándar de STA/LTA.

En la Figura 2.2.2 es posible apreciar una señal con un evento sísmico. En la gráfica superior, la línea de color rojo marca el tiempo de inicio de la alerta o disparo, en azul el tiempo de fin de la alerta. En la gráfica inferior, la línea de color rojo marca el umbral de activación y si el valor del detector supera este nivel significa el comienzo de un evento sísmico. En azul se indica el umbral al que debe descender la señal de disparo para dar por concluida la alerta [17].

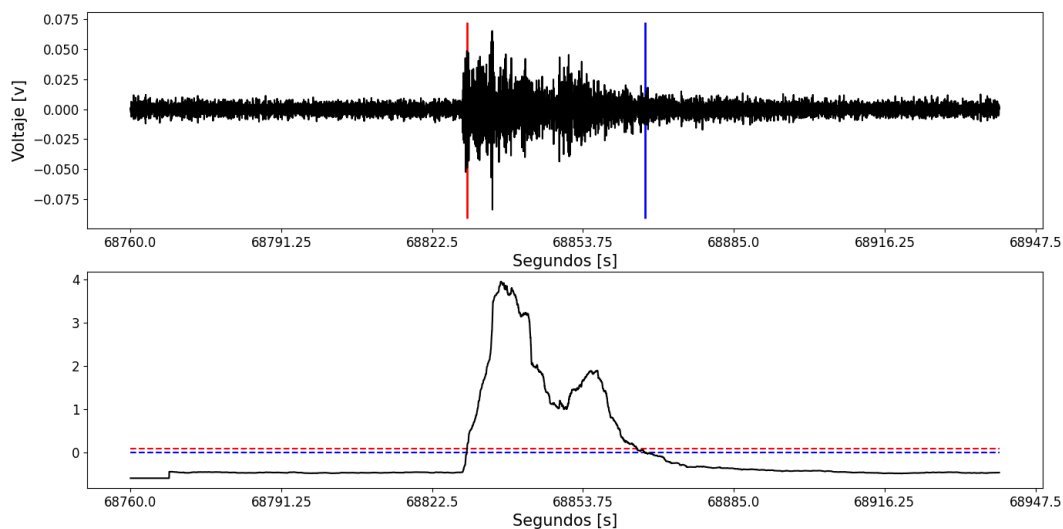


Figura 2.2.2: Ejemplo del algoritmo Z Detector

2.2.3. Baer and Kradolfer picker

Baer y Kradolfer [11] mejoraron la función envolvente de Allen, incorporando un umbral de detección de señal dinámico. Este algoritmo marcó un hito en la identificación automática de fases sísmicas, la cual, si es combinada con una evaluación sofisticada de calidad, puede identificar tiempos de llegada de onda P con alta precisión [18]. A diferencia de la función de envolvente cuadrada de Allen, esta función característica (CF) es sensible a los cambios de amplitud, frecuencia y fase como se muestra en la Figura 2.2.3.

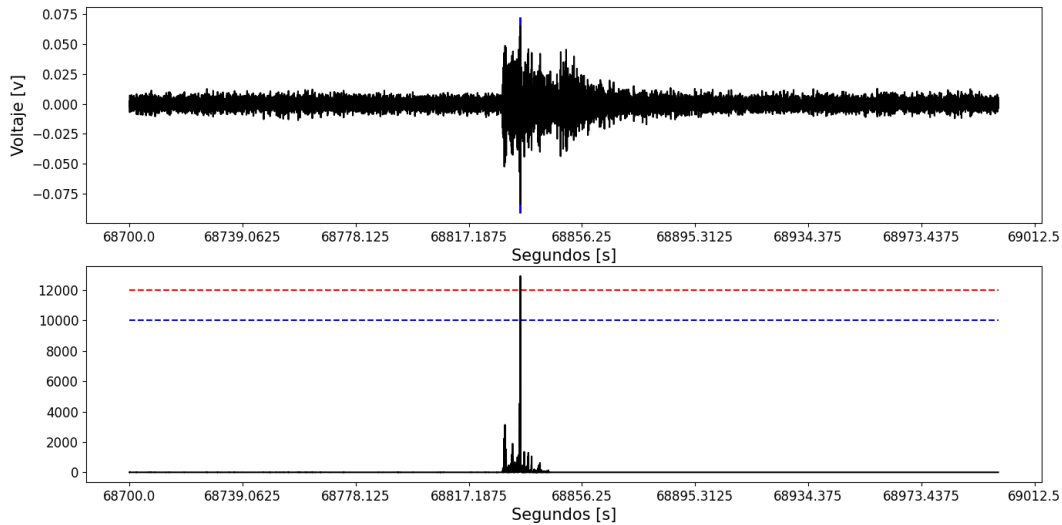


Figura 2.2.3: Ejemplo del algoritmo Baer y Kradolfer

En concreto, se establece una bandera de selección si CF supera un umbral. Para evitar el incremento en la detección de eventos falsos a corto plazo causados por el ruido, una señal solo se acepta si el CF no cae por debajo del umbral de la señal durante tiempos mayores que el período dominante. Si el CF disminuye dentro de un cierto tiempo t_{up} , la selección provisional se borra. Sin embargo, debido a la complejidad de las señales sísmicas, se permiten caídas del CF por debajo del umbral y durante segundos de t_{down} [2]. La Figura 2.2.4 muestra un ejemplo del método Baer y Kradolfer (rojo) calculado para una forma de onda de evento local (negro). La línea vertical azul indica el inicio de la onda P detectada automáticamente, mientras la línea verde la detección manual.

El método de Baer y Kradolfer es un algoritmo rápido y robusto. Baer y Kradolfer no proponen una evaluación automática de la calidad. Sin embargo, este algoritmo produce picos de alta calidad y se complementó con el sofisticado sistema de evaluación de calidad MannekenPix [19].

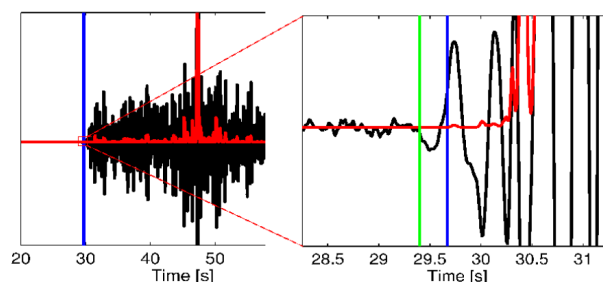


Figura 2.2.4: Ejemplo de Baer y Kradolfer [2]

2.2.4. AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

El método AR-AIC fue propuesto por Sleeman y Van Eck [12]. En concreto, se divide una serie de tiempo más larga en dos segmentos localmente estacionarios, cada uno modelado por un proceso AR (Autoregressive Process (AR)). El primer segmento representa ruido y el segundo segmento contiene la señal. El algoritmo de selección se puede describir mediante cinco pasos: (1) filtrado paso banda del sismograma, (2) detección de una fase sísmica usando un detector STA/LTA, (3) estimación de dos conjuntos de parámetros AR para el ruido y la señal, respectivamente, (4) cálculo de dos errores de predicción utilizando los parámetros AR para ruido y señal, respectivamente, (5) el mínimo del AIC, el cual indica el tiempo de llegada. En la Figura 2.2.5 se puede observar en la forma de onda (negra), el método STA/LTA (línea roja, discontinua) y el inicio P determinado automáticamente (línea roja). Verde: la función AR-AIC.

El selector AR-AIC es un algoritmo de selección altamente sofisticado basado en la teoría de la información. El algoritmo presenta una alta demanda computacional y, por lo tanto, es mucho más lento que otros métodos. El inicio de la onda P se deriva de un detector STA/LTA, el cual podría perder las llegadas P emergentes o las llegadas de la fase P dominadas por cambios instantáneos en la frecuencia. Esto puede limitar el rendimiento de este algoritmo de selección. Por lo tanto un defecto de este método es la dependencia del algoritmo STA/LTA [2].

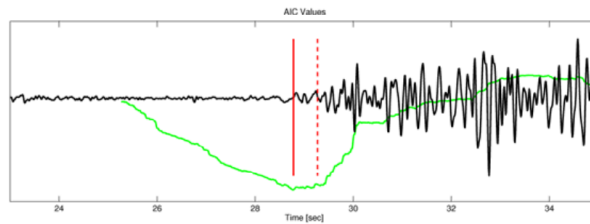


Figura 2.2.5: Ejemplo del método AR-AIC [2]

2.3. Técnicas de procesamiento de señales

El procesamiento de señales es la aplicación de una serie de operaciones matemáticas a un conjunto de datos provenientes de una señal con el objetivo de tener una estimación del contenido de la información y de esta forma analizar, representar, transformar, y manipular señales. Por lo tanto se abordan temas sobre filtros y sus tipos, además de su respuesta al impulso, fase y frecuencia.

2.3.1. Filtrado de señales

El filtrado es un proceso computacional o algoritmo mediante el cual una señal digital es transformada en una segunda secuencia de muestras o señal digital de salida con el objetivo de reducir o eliminar el ruido de una señal. [20].

2.3.1.1. Respuesta al impulso, frecuencia y de fase de un filtro

La respuesta al impulso es la reacción de un filtro a un impulso $\delta[n]$ que se envía a su entrada. La respuesta al impulso caracteriza a un filtro en el dominio temporal. Dicha respuesta al impulso estará discretizada en el tiempo y por tanto definida por una serie de muestras, como se indica en la Ecuación 2.1:

$$h[n] \quad (2.1)$$

La transformada de Fourier de una respuesta al impulso de un filtro corresponde a su función de transferencia o representación en frecuencia, definida en la Ecuación 2.2, que caracteriza al filtro en el dominio de la frecuencia. Dicha caracterización se realiza a través de su espectro de amplitud (Ecuación 2.3) y de su espectro de fase (Ecuación 2.4).

$$H(f) \tag{2.2}$$

$$\textit{Amplitud} : |H(f)| \tag{2.3}$$

$$\textit{Fase} : \angle H(f) \tag{2.4}$$

Sea una señal digital de entrada $x[n]$ que se procesa con un filtro para generar una señal de salida $y[n]$. El espectro de la señal de salida $Y(f)$ se obtiene multiplicando el espectro de entrada $X(f)$ por la respuesta en frecuencia del filtro $H(f)$, esto se muestra en la Ecuación 2.5:

$$Y(f) = X(f) \cdot H(f) \tag{2.5}$$

Esto equivale a la operación de convolución (representada con un $*$) entre las señales en el dominio temporal (Ecuación 2.6):

$$y[n] = x[n] * h[n] \tag{2.6}$$

Los filtros tienen también un efecto importante en la fase de las señales. El filtrado en sí mismo es una aplicación de los retardos (modificando la fase de la señal), lo que explica su comportamiento en el dominio temporal y su implantación digital [20].

2.3.1.2. Tipos de filtros

Los filtros más usuales son los filtros paso bajo (**Low Pass (LP)**), paso alto (**High Pass (HP)**), paso banda (**Band Pass (BP)**) y los filtros rechazo de banda (o paso no banda) (**Notch (NOTCH)**). En la Figura 2.3.1 se representan estos 4 tipos de filtros mediante su respuesta en frecuencia o espectro de amplitud. Cada punto de la respuesta en frecuencia indica la atenuación a la que se someterá una señal a una frecuencia determinada.

En este contexto, mientras los filtros paso bajo dejan pasar las frecuencias que están por debajo de una determinada frecuencia. Los filtros paso alto dejan pasar las frecuencias que están por encima de una determinada frecuencia. Estos dos tipos de filtros están definidos por su frecuencia de corte, que es la frecuencia a la cual la amplitud de la señal se reduce a $\frac{1}{\sqrt{2}}$ de su valor máximo, es decir, sufre 3 dB de atenuación. Por otra parte los filtros paso banda dejan pasar las frecuencias que están situadas en una determinada banda de frecuencias, es decir, entre dos determinadas frecuencias. Los filtros rechazo de banda dejan pasar todas las frecuencias excepto las que están situadas en una determinada banda de frecuencias, es decir, entre dos determinadas frecuencias f_1 y f_2 . Estas son las frecuencias a las que la amplitud de la señal se reduce a $\frac{1}{\sqrt{2}}$ de su valor máximo, es decir, sufre 3 dB de atenuación.

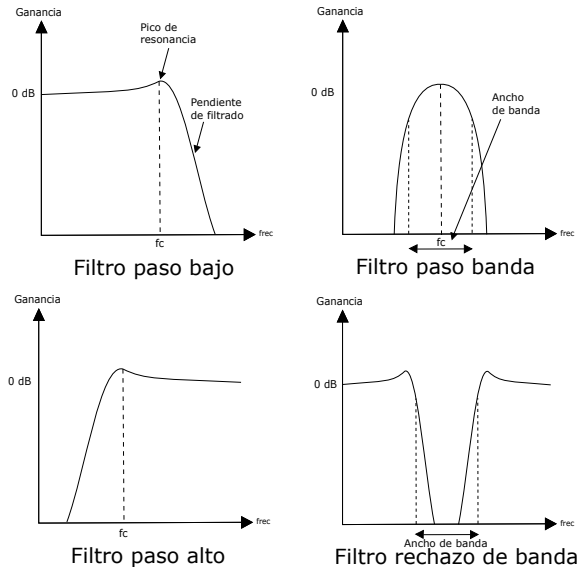


Figura 2.3.1: Tipos de filtros.

2.3.1.3. Filtro de Butterworth

Este filtro tiene una respuesta plana en la banda de paso (llamada máximamente plana), a expensas de la respuesta en la región de transición, la cual es de 20 dB/Década por polo. El módulo de la respuesta en frecuencia del filtro pasabajos, para ganancia G , y frecuencia de corte w_c esta dado en la Ecuación 2.7.

$$|H(W)| = \frac{G}{\sqrt{1 + \left(\frac{w}{w_c}\right)^{2n}}} \quad (2.7)$$

Donde $n = 1, 2, \dots, k$. es el orden. En la Figura 2.3.2 se presentan ejemplos de la respuesta de este tipo de filtro para distintos valores de n [21].

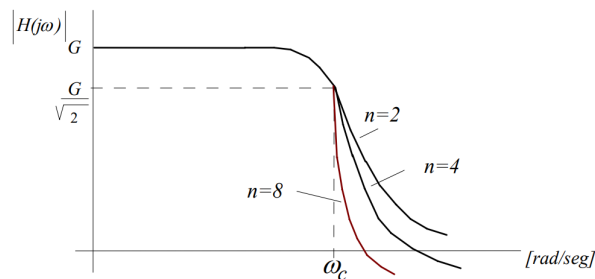


Figura 2.3.2: Respuesta del filtro Butterworth

2.4. Formatos de datos sísmicos

Los formatos sísmicos más extendidos para el intercambio de datos de formas de onda son los formatos [Standard for the Exchange of Earthquake Data \(SEED\)](#), [Seismic Analysis Code \(SAC\)](#) y [Group of Scientific Experts \(GSE\)](#). La información en todos los formatos de datos se divide en dos partes: encabezado y datos

de forma de onda. El encabezado puede incluir información sobre el sitio, la estación, el canal, la red, la hora, la respuesta del instrumento, información sobre el sistema de grabación, el tipo de datos de forma de onda, etc. A continuación se ofrece la descripción de algunos de los formatos de datos más utilizados.

2.4.1. SEED

SEED ha sido adoptado por la [Federation of Digital Seismographic Networks \(FDSN\)](#) como formato para los datos de la Red Global de Sismógrafos ([Global Seismograph Network \(GSN\)](#)), principalmente para el intercambio de datos de formas de onda sin procesar. Varios de los sistemas de adquisición de datos de la generación actual producen datos en formato miniSEED (volumen **SEED** de datos únicamente).

SEED funciona para datos de forma de onda continuos, orientados a estaciones, observatorios, redes o arreglos individuales. La **SEED** está autodefinida, es decir que los datos de cada canal de estación incluyen toda la información necesaria, incluida la información de respuesta, la referencia al orden de bytes y el lenguaje de descripción de datos (entero, binario, comprimido).

El archivo **SEED** consta de varios volúmenes lógicos. Hay tres tipos de volúmenes lógicos: orientados a la estación de campo, a la red de estaciones y a la red de eventos. La estructura de cada volumen lógico es la misma, pero la interpretación de algunos campos de datos es diferente. Los datos en el formato se dividen en dos partes: encabezados de control (volumen **SEED** sin datos) formateados en el Código Estándar Americano para el intercambio de información ASCII y series de tiempo (volumen **SEED** solo de datos o miniSEED) que son binarios y sin formato.

Los encabezados de control contienen información auxiliar sobre el volumen, los canales de la estación y los datos. Debido a que los encabezados de control están codificados en caracteres ASCII imprimibles, se pueden leer directamente y los analistas pueden controlarlos visualmente. Se definen cuatro grupos de encabezados de control. El primer grupo se denomina encabezados de identificador de volumen. Contiene información sobre la hora de los datos, la longitud del registro lógico, la versión de formato del volumen lógico, etc. El segundo se denomina encabezados del diccionario de abreviaturas. El cual contiene las definiciones de abreviaturas utilizadas en otros encabezados de control. El tercer grupo se denomina encabezados de estación. El mismo proporciona características operativas relevantes para una estación y todos sus canales, incluida la ubicación de la estación, los tipos de instrumentos y las funciones de transferencia de canales [22].

2.4.2. SAC

SAC es un programa interactivo de propósito general diseñado para el estudio de señales secuenciales en el tiempo. Un archivo de datos **SAC** contiene un solo componente de datos registrado en una sola estación sísmica. Cada archivo de datos también contiene un registro de encabezado que describe el contenido de ese archivo. Deben estar presentes ciertas entradas de encabezado (por ejemplo, el número de puntos de datos, el tipo de archivo, etc.). Otros siempre están presentes para ciertos tipos de archivos (por ejemplo, intervalo de muestreo, hora de inicio, etc.). Algunas variables de encabezado son simplemente informativas y el programa no las utiliza directamente. Aunque el software de análisis **SAC** solo se ejecuta en plataformas Unix y el formato general es binario, también existe una versión ASCII que se puede utilizar en cualquier plataforma [23].

2.4.3. GSE

GSE consta de una línea de identificación de forma de onda seguida de la línea de la estación (STA2), la información de la forma de onda en sí (DAT2) y una suma de comprobación de los datos (CHK2) para cada sección DAT2. La longitud de línea predeterminada es 132 bytes. Ninguna línea puede tener más de 1024 bytes. El tipo de datos de respuesta permite dar la información completa como una serie de grupos que se pueden conectar en cascada. La descripción de la respuesta se compone de la línea de identificación CAL2 más una o más de las secciones de PAZ2, FAP2, GEN2, DIG2 y FIR2 en cualquier orden. La línea de identificación de forma de onda WID2 da la fecha y hora de la primera muestra de datos; los códigos de estación, canal y auxiliar; el subformato de los datos, el número de muestras y la frecuencia de muestreo;

la calibración del instrumento representada como el número de nanómetros por conteo digital en el período de calibración; el tipo de instrumento y la orientación horizontal y vertical [23].

2.4.4. IRIS

El sistema de recuperación de datos de acceso telefónico IRIS se puede utilizar para buscar, visualizar y escribir datos de las estaciones IRIS GSN que están equipadas con capacidades de acceso telefónico. Las formas de onda digitales se pueden escribir en ASCII utilizando los diversos comandos en línea. Estos archivos contienen dos tipos de registros: de encabezado (uno por archivo) y de datos. El encabezado contiene información sobre la estación y el instrumento, la hora de inicio del registro de datos y el número de muestras. El archivo de datos contiene el número de registro, 8 valores de muestra y una suma de comprobación. Este formato usa un archivo separado para cada componente de cada estación [23].

2.5. Estación sísmica

Una estación sísmica es una suma de varios instrumentos que sirven para registrar eventos sísmicos. La misma consta de un sensor que puede ser triaxial (componentes: vertical y 2 horizontales) ó uniaxial (componente vertical). Además, dispone de un equipo de registro que permite la adquisición de la señales para su almacenamiento y transmisión en tiempo real desde el terreno o región de estudio hacia el Centro de Adquisición de Datos. Generalmente estas estaciones son de operación autónoma debido al bajo consumo de energía de sus elementos.

Existen varios tipos de estaciones sísmicas que se definen por el tipo de aplicación, como pueden ser: Estación Sísmica Permanente, Estación de Estructuras (edificios, puentes, presas, torres, etc.), Estaciones en Minas y Pozos profundos, Estaciones de Fondo Marino. En la Figura 2.5.1 se muestran los equipos instalados en el edificio de la Empresa Eléctrica Regional Centro Sur para la adquisición de datos sísmicos en la Red Sísmica del Austro (RSA) dentro del proyecto “Tecnologías IoT y Redes Inalámbricas de Sensores aplicados a la Monitorización de Salud Estructural en Edificios Esenciales de la Ciudad de Cuenca”.

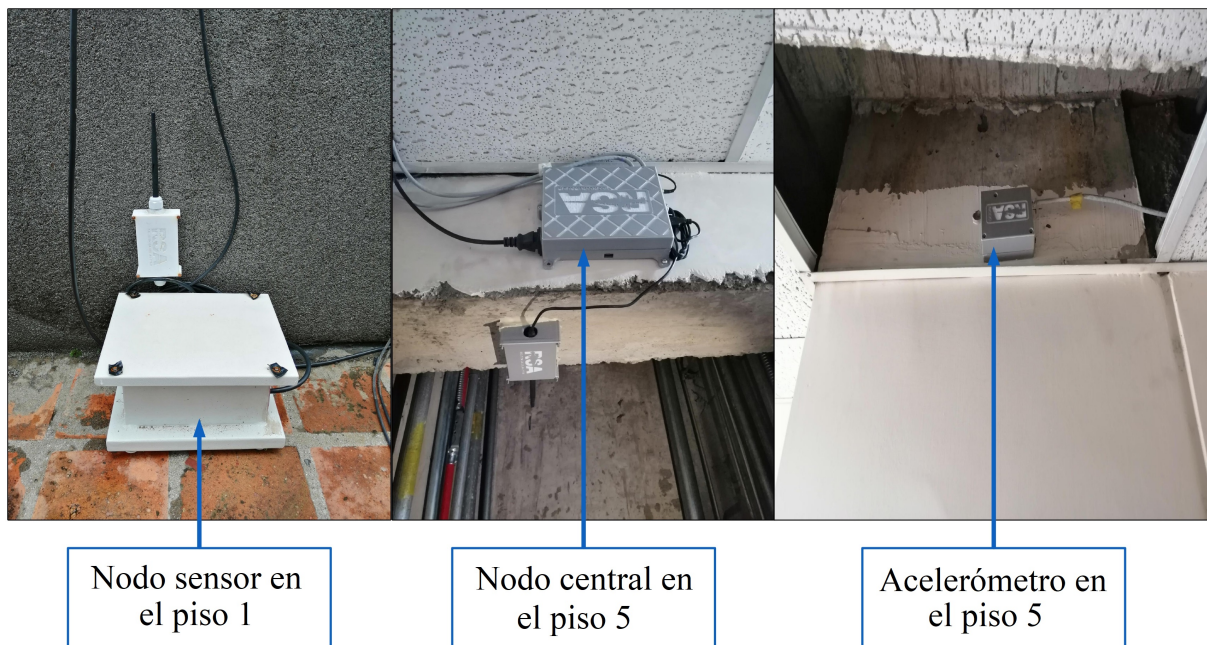


Figura 2.5.1: Estaciones instaladas en la Red Sísmica del Austro

Capítulo 3

Estado del arte

En el presente capítulo se describen los principales trabajos disponibles en la literatura y relacionados con la temática de la tesis. Para una mejor comprensión del problema y cómo abordarlo, se revisaron numerosos trabajos relacionados y se consideraron los diversos enfoques donde se emplean herramientas de software/hardware para la detección de eventos sísmicos. Asimismo se analizaron distintos estudios donde se describen varios métodos utilizados para comparar algoritmos y también diferentes métodos empleados en la detección de eventos en tiempo real. Específicamente, se profundizaron los métodos basados en STA/LTA ya que son los más usados, pero también se buscaron trabajos sobre nuevos métodos que se utilizan actualmente.

3.1. Algoritmos de detección automática de eventos sísmicos

El daño causado por un sismo depende del grado de preparación y de la capacidad de respuesta de la sociedad. Por lo tanto, la detección de eventos es fundamental para mitigar los detrimentos ocasionados por los sismos. Tradicionalmente, la detección de un evento sísmico se realizaba de forma visual, es decir basándose en las amplitudes y los cambios de la forma de onda de la señal. Actualmente, existen diferentes algoritmos de detección automática que permiten realizar este análisis de forma más sencilla, con mayor precisión y detalle. Sin embargo, la detección automática de eventos sísmicos, en un registro continuo de datos de una red de estaciones sísmicas, es un desafío por el ruido sísmico que existe junto a las señales, como se describe en [7].

En tal contexto, existen métodos que ocupan como discriminador la energía cinética registrada por los sismómetros o alguna función simple de ésta. Los algoritmos que usan métodos de energía son los más utilizados, se basan en el estudio del cuadrado de la velocidad de partícula. Una de las funciones más comunes es el denominado [Short-Time-Average Through Long-Time-Average Trigger \(STA/LTA\)](#). También están los métodos que hacen uso de la morfología de las ondas. Los métodos morfológicos son aquellos destinados a remarcar una firma particular de los eventos, como por ejemplo, el arribo repentino de la onda P, la gran amplitud de la onda S o la elipticidad de las ondas de superficie. Un ejemplo de este mecanismo, es el algoritmo Z-detector [10]. También existen algoritmos de selección, que se utilizan para estimar de forma más precisa los tiempos de llegada de las ondas P y S. Un algoritmo de selección ampliamente utilizado es el propuesto por Baer y Kradolfer [11]. El algoritmo de Baer y Kradolfer produce selecciones de alta calidad, además, es de fácil aplicación debido a la baja cantidad de parámetros a configurar. Otra forma de detectar el inicio de la onda P es utilizar estadística de orden superior. El algoritmo llamado [Autoregressive-Akaike-Information-Criterion-picker \(AR-AIC\)](#) fue propuesto por Sleeman y Van Eck [12]. Este algoritmo tiene una fuerte sensibilidad a los cambios en la forma de onda por lo que representa una herramienta robusta para la identificación del primer arribo de la onda P [2]. La librería ObsPy cuenta con todos estos algoritmos para la detección de eventos sísmicos.

En la actualidad, existe una diversidad de algoritmos de detección, que van desde un tipo de umbral de amplitud muy simple hasta los más sofisticados basados en enfoques de reconocimiento de patrones. Por lo tanto, en cuanto a la comparación de algoritmos, en [24] se presentan varios enfoques para detectar señales de eventos en ruido de fondo y se discute su precisión. De igual forma, en [25] se realiza la comparación



entre el algoritmo [STA/LTA](#) y un método que se basa en las mediciones de [Power Spectral Density \(PSD\)](#) para la detección de eventos microsísmicos. Como resultado de este estudio, se encontró que la técnica [PSD](#) supera al método [STA/LTA](#) al detectar un mayor número de eventos microsísmicos débiles que se pierden con el ruido de fondo. Cuando se aplicaron los datos sin filtrar, el método [PSD](#) detectó aproximadamente un 55,2% más eventos que el [STA/LTA](#). Incluso cuando se aplicaron datos filtrados, el método [PSD](#) todavía detectó aproximadamente un 37,7% más eventos. Otra comparación se realiza en [26], donde además de comparar los métodos, se obtiene el rendimiento y las reglas de selección para la longitud de la ventana, el umbral de activación y la función característica de los algoritmos: [STA/LTA](#), *weighting factor method*, el algoritmo *multi-window* y el algoritmo *modified energy ratio*. Como resultado se obtuvo que el método *weighting factor* puede reducir la tasa de error de la detección microsísmica. El algoritmo *multi-window* y el algoritmo *modified energy ratio* pueden mejorar la precisión de la detección de eventos para datos microsísmicos de baja relación señal/ruido.

En este contexto, para el desarrollo de un sistema [Waveform Correlation Event-Detection and Location System \(WCEDS\)](#) en [15], se consideraron algoritmos [STA/LTA](#), estadística Z, algoritmos de polarización y de frecuencia transitoria. Sin embargo, se encontró que ningún algoritmo fue claramente óptimo en todas las condiciones probadas, pero el método [STA/LTA](#) proporcionó una salida que cumple mejor con los requisitos de una correlación global, sistema de localización y detección de eventos.

3.2. Detección de eventos en tiempo real

Por otra parte, en lo que respecta a los sistemas de monitoreo de alerta temprana, estos requieren la detección de eventos en tiempo real para minimizar el posible impacto de desastres naturales. En [27] se propone implementar un detector de eventos de periodo largo ([Long Period \(LP\)](#)) en tiempo real y un detector volcano-tectónico ([Volcano Tectonic \(VT\)](#)) basado en algoritmos de detección de actividad de voz. Para determinar la eficiencia del detector propuesto, se utilizó para las pruebas una base de datos que contiene 436 eventos sísmicos ([LP](#) y [VT](#)) adquiridos en el volcán Cotopaxi en Ecuador. Luego se calcularon los principales parámetros de rendimiento como exactitud (A), precisión, sensibilidad, especificidad y la tasa de error balanceada ([Balanced Error Rate \(BER\)](#)), finalmente también se consideró el tiempo de procesamiento requerido por el algoritmo. Los resultados obtenidos sugieren un rendimiento similar en comparación con los algoritmos de detección de eventos desarrollados previamente para el mismo conjunto de datos, pero con mucha menos complejidad computacional, logrando una exactitud de 95,2% y [BER](#) de 0,005. En [28] también se presenta un algoritmo automático de detección de ondas P y S, que utiliza funciones características derivadas de curtosis ([Characteristic Functions \(CF\)](#)) y descomposiciones de valores propios en datos sísmicos de tres componentes. Se modificó la [CF](#) de curtosis para mejorar la precisión de la selección calculando la [CF](#) en varios anchos de banda de frecuencia, tamaños de ventana y parámetros de suavizado. Una vez que se seleccionan las fases, el algoritmo determina el tipo de inicio (P o S) utilizando parámetros de polarización, elimina las selecciones incorrectas mediante un procedimiento de agrupación y la relación señal-ruido ([Signal-to-Noise Ratio \(SNR\)](#)) y asigna un índice de calidad de selección basado en la [SNR](#).

De igual forma, en [29] se presenta un algoritmo de filtro adaptado rápido ([Matched Filter \(MF\)](#)), que pretende resolver el problema de eficiencia de los métodos [STA/LTA](#), [PSD](#) y subespacio ([Subspace Detection \(SD\)](#)). El sistema propuesto se basa en una técnica de correlación cruzada normalizada rápida ([Normalized Cross-Correlation \(NCC\)](#)). Este método detecta eventos en los datos en función de su similitud con los eventos de plantilla, comparando los coeficientes [NCC](#) entre los eventos de plantilla y los datos con un umbral específico definido por el usuario. Como resultado se obtuvo que el algoritmo [MF](#) es más eficiente en la detección de eventos con menos falsas alarmas, mayor probabilidad de detección y menor tiempo de procesamiento que el [STA/LTA](#), especialmente cuando se trata de conjuntos de datos grandes y ruidosos.

3.3. Nuevos enfoques

En cuanto a nuevos enfoques se tiene que algunas de las herramientas para la detección de eventos sísmicos más recientes y prometedoras provienen del campo del aprendizaje automático. En [30] se com-



bina un método reciente de detección y ubicación sísmica con la clasificación de redes neuronales. En los experimentos se analizaron datos registrados por 76 estaciones a lo largo de 4 meses, para lo cual se dividieron los conjuntos de datos en 234.240 ventanas de 3 minutos de duración con un 75 % de superposición. El modelo de red neuronal predijo que 2.522 ventanas de tiempo contienen eventos sísmicos, de los cuales se localizaron 1.192 eventos únicos. De igual forma, en [31], se describe un Clasificador de Identificación de Fase ([Phase-Identification Classifier \(CPIC\)](#)) basado en Redes Neuronales Convolucionales ([Convolutional Neural Network \(CNN\)](#)) diseñado para la detección de fase y selección de conjuntos de datos de entrenamiento de tamaño pequeño a mediano. Cuando se entrenó en 30.146 fases etiquetadas y se aplicó a un mes de grabaciones continuas durante las secuencias de réplicas del terremoto de Wenchuan MW7,9 de 2008 en Sichuan, China, [CPIC](#) logró detectar el 97,5 % de las fases seleccionadas manualmente en el catálogo estándar y predijo sus tiempos de llegada con una mejora de cinco veces sobre el selector de AR de ObsPy. Alternativamente, los avances recientes están relacionados con la aplicación de Redes Neuronales Artificiales ([Artificial Neural Network \(ANN\)](#)), un subconjunto de técnicas de aprendizaje automático que está impulsando el desarrollo de la investigación en diversos campos de estudio. Por ejemplo, en [32] se examinan las últimas aplicaciones de [ANN](#) para la interpretación automática de datos sísmicos, con un enfoque especial en la detección de terremotos y la estimación de tiempos establecidos.

Por otra parte, se han diseñado muchas herramientas y métodos para analizar datos sísmicos. Sin embargo, los métodos dominantes en la mayoría de los mecanismos son los basados en [STA/LTA](#). Estos mecanismos tienen un umbral considerable en términos de rango de terremoto, por lo que muchos microsismos se ignoran como ruido. En [33] los datos registrados de varios experimentos sísmicos se clasificaron mediante un modelo híbrido. Por lo tanto, el objetivo fue mejorar la autenticidad de los datos basados en la aplicación de variables efectivas. Esto se llevó a cabo mediante el uso de un método difuso y un algoritmo de red neuronal integrado, que involucró al perceptrón ([Multilayer Perceptron \(MLP\)](#)) y la red radial de ([Base Radial Function \(RBF\)](#)) en forma de un sistema de aprendizaje colectivo con el fin de identificar eventos sísmicos a pequeña escala. Según los resultados, en comparación con los métodos básicos, el método propuesto mejoró significativamente utilizando los criterios de error real y error de raíz cuadrática media ([Root-Mean-Square Error \(RMSE\)](#)).

En lo que se refiere a nuevos métodos, en [34] se implementa una propuesta que usa una técnica de aprendizaje automático para la clasificación y detección de eventos en redes sísmicas. En concreto, se emplean los atributos de polarización y frecuencia como parámetros de entrada para el cálculo de la regresión. Estos atributos se extraen en los tiempos de llegada previstos para las ondas P y S utilizando solo un modelo de velocidad aproximado, ya que los atributos se calculan en períodos de tiempo grandes. Se demostró que este método de caracterización es capaz de distinguir entre explosiones y terremotos con una precisión del 99 % empleando una red de 13 estaciones ubicadas en el sur de California. Este método proporciona una forma muy segura de detectar y localizar eventos. Además, una gran cantidad de eventos sísmicos se pueden detectar automáticamente logrando pocas alarmas falsas, lo que permite un catálogo de eventos automático más grande con un alto grado de confianza. Por otro lado, en [35], se explora una extensión del método de [matched-filter](#) para detectar terremotos de baja frecuencia en escalas locales a regionales. Se muestra que es posible aumentar el número de detecciones en comparación con los métodos estándar basados en energía, con bajas tasas de detección falsa, usando conjuntos de formas de onda sintéticas como plantillas.

También existen nuevos algoritmos que utilizan como base el método [STA/LTA](#). Uno de ellos es el [STAFD/LTAFD](#) ([Short-Time-Average Fractal Dimension Through Long-Time-Average Fractal Dimension](#)) desarrollado en [36]. De forma similar, en [37] se presenta un algoritmo adaptativo que utiliza [STA](#), [LTA](#) y los modelos ocultos de Markov ([HMM](#)). Otra implementación realizada en [38], utiliza el algoritmo [STA/LTA](#) y el modelo Hadoop MapReduce (modelo de programación para dar soporte a la computación paralela sobre grandes colecciones de datos) para acelerar el proceso de detección al reducir el tiempo de procesamiento. Por otra parte, en [39], se propone una modificación al flujo de trabajo convencional [STA/LTA](#). En lugar de usar la amplitud o energía de la forma de onda de entrada, se usa una característica de [k-means](#), función que es más sensible a las variaciones de la señal en relación con el ruido de fondo. Como resultado se obtuvo que, para un nivel de ruido de fondo constante, el [STA/LTA](#) convencional responde menos y, por lo tanto, no detecta 2 eventos débiles. Sin embargo, [STA/LTA](#) aplicado en este flujo de trabajo propuesto detectó exitosamente cada uno de los 11 eventos al costo de un falso positivo.



3.4. Software para el análisis sísmico

En cuanto a las herramientas de software disponibles para análisis sísmicos, existen diferentes propuestas. Por ejemplo, PICOSS es una plataforma grafica en Python para la detección, segmentación y clasificación de datos sísmicos-volcánicos. El usuario puede seleccionar flujos de trabajo automáticos o manuales, incluidas las redes neuronales profundas como se explica en [40]. También hay métodos analíticos en la ingeniería sísmica basados en el rendimiento ([Performance-based earthquake engineering \(PBEE\)](#)) generalmente emplean conjuntos de registros de movimiento del suelo para obtener métricas de rendimiento estructural. En este contexto, EaRL, ver [41], es un software basado en Matlab que tiene como objetivo facilitar los cálculos [PBEE](#) proporcionando una interfaz gráfica, varias opciones de visualización de datos y además es de código abierto. De igual forma, R2R-EU es una herramienta [PBEE](#) que implementa numéricamente varios métodos para estimar la fragilidad sísmica específica de una estructura. El usuario puede elegir el método de análisis entre Bootstrap no paramétrico y el método delta, como se detalla en [42]. Finalmente, APASVO, ver [43], es una herramienta grafica en Python de código abierto que permite la detección automática de la onda P y de eventos sísmicos. La aplicación utiliza los algoritmos STA/LTA, AMPA y AIC.

Al comparar las herramientas descritas anteriormente con el software implementado para el análisis de eventos en el presente trabajo de Tesis, cabe destacar que es de código abierto en Python que tiene las funcionalidades de análisis en intervalos de tiempo, generación de gráficas de los resultados, exportación de los resultados de detección de eventos, creación de nuevos archivos miniSeed con los datos de los intervalos de interés y además permite elegir entre varios algoritmos de detección de eventos. Todas estas funcionalidades se desarrollaron en una plataforma SBC (Raspberry Pi 3), la cual es ampliamente utilizada en múltiples aplicaciones por sus capacidades de versatilidad, robustez, con prestaciones actuales y disponibilidad de diversos periféricos. En consecuencia, se implementó una herramienta portátil, escalable, de bajo costo y uso mínimo de CPU para optimizar el rendimiento del dispositivo SBC.

Capítulo 4

Detección y procesamiento de eventos sísmicos

En este capítulo se presentan las etapas para la detección y procesamiento de eventos sísmicos. Con tal objetivo, en primer lugar, se describe la librería ObsPy, la misma que proporciona las herramientas necesarias para la detección de eventos sísmicos. A continuación, se estudian los principales algoritmos disponibles en la librería, a partir de lo cual se propone una herramienta para el análisis de eventos sísmicos. Finalmente, se emplea dicha herramienta para comparar los diferentes algoritmos y determinar la mejor opción para implementar en la detección en tiempo real.

4.1. La librería ObsPy

La herramienta ObsPy proporciona un conjunto de funcionalidades desarrolladas en Python, las mismas que facilitan el procesamiento de señales sísmicas. En ObsPy se implementan y están listas para usar las siguientes rutinas esenciales de procesamiento sísmológico: lectura y escritura de datos en los formatos SEED/MiniSEED y Dataless SEED, XML-SEED, GSE2 y SAC, así como filtrado, simulación de instrumentos, activación y trazado. También hay soporte para recuperar datos de ArcLink (un protocolo de solicitud de datos distribuidos para acceder a datos de forma de onda archivados) o una base de datos SeisHub [44]. Recientemente, se añadieron módulos para leer los archivos de datos DESANO y para recuperar datos con el protocolo de interfaz de gestión de datos (DHI).

Adicionalmente, ObsPy permite el uso de módulos de programación numérica de matrices como NumPy [45] o SciPy [46]. Los resultados se pueden visualizar utilizando módulos como matplotlib (2D) [47] o MayaVi (3D)[48]. Dado que Python y sus módulos mencionados son de código abierto, no hay restricciones debido a las licencias. Esta es una clara ventaja sobre el producto patentado MATLAB [49]. Además, Python es conocido por su sintaxis intuitiva. Es independiente de la plataforma, y su popularidad en rápido crecimiento se extiende más allá de la comunidad sísmológica. Python se utiliza en varios campos porque su completa biblioteca estándar proporciona herramientas para todo tipo de tareas (por ejemplo, los servidores web completos se pueden escribir en unas pocas líneas con módulos estándar) [50].

4.1.1. Instalación y configuración

La instalación de ObsPy se realizó en una Raspberry Pi 3 como se indica en la Figura 4.1.1.

1. Instalación de la librería ObsPy

```
sudo pip install obspy
```

La implementación se realiza en Python 3. *Sudo* es para instalar en el directorio de Python: */usr/local/lib/python3.7/dist-packages*. *Pip3* sirve para instalar la librería de Python 3.

2. Después se comprueba que todo este correcto con el comando:

```
sudo obspy-runttests
```

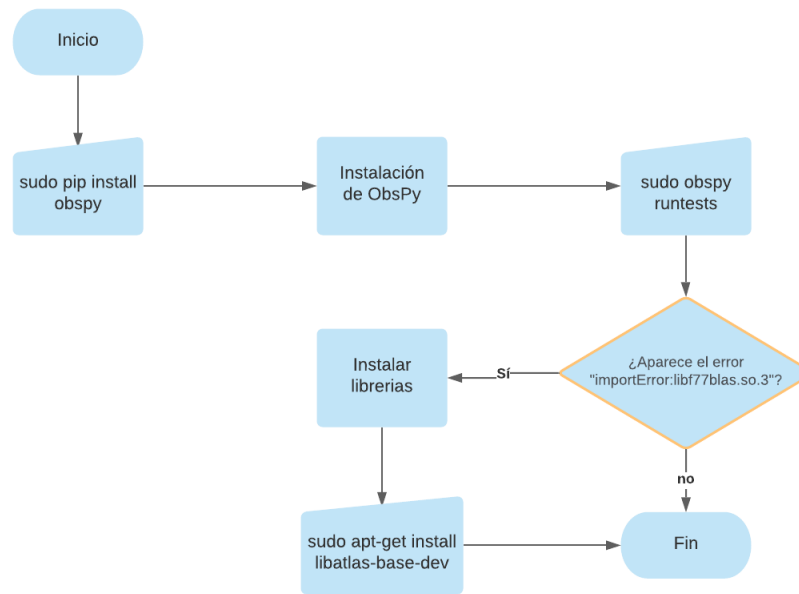


Figura 4.1.1: Diagrama de la instalación de la librería ObsPy en la Raspberry Pi

3. En caso de que aparezca el siguiente error: *ImportError: libf77blas.so.3: cannot open shared object file: No such file or directory*

Es necesario instalar:

```
sudo apt-get install libatlas-base-dev
```

4. Al realizarlo se ejecuta nuevamente *sudo obspy-runtests*, el cual es un script de prueba *runtests.py* (en el directorio */usr/local/bin*)

4.1.2. Métodos de detección disponibles en la librería ObsPy

Los métodos para la detección automática de eventos sísmicos disponibles en ObsPy son: Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA, Z Detector, Baer and Kradolfer picker y Autoregressive-Akaike-Information-Criterion-picker. Las características y parámetros necesarios para cada algoritmo se detallan a continuación.

4.1.2.1. Classic STA/LTA

La idea detrás del método *STA/LTA* es simple; la relación *STA/LTA* se calcula continuamente en cada tiempo t para cada k -ésimo canal de datos como $R = \frac{STA}{LTA}$. STA es la amplitud media a corto plazo y LTA es el promedio a largo plazo de las señales.

El valor STA mide el nivel de amplitud instantáneo de la señal sísmica y observa los eventos, mientras que el LTA se encarga de la amplitud promedio actual del ruido sísmico. Cuando la relación (R) excede un umbral predeterminado (seleccionado por el usuario) τ , inicia la detección de un evento. El disparador está activo hasta que la proporción cae por debajo de un umbral de *detrigger*. Aunque pueden ser diferentes, los umbrales de activación y desactivación se suelen considerar iguales y se denominan simplemente umbral de detección ($\tau > 1$).

Aunque el detector *STA/LTA* puede identificar los cambios de amplitud de manera efectiva, es menos preciso al reconocer las llegadas de fase [51].

Método en ObsPy:

`classic_sta_lta(a, nsta, nlta)`

Esta función calcula el [STA/LTA](#) estándar a partir de una matriz de datos sísmicos de entrada a determinada. La longitud de la ventana STA es dada por $nsta$ en muestras y la longitud de la LTA dada por $nlta$ en muestras [52].

Los parámetros del algoritmo de activación de [STA/LTA](#) se muestran en la Tabla 4.1.1.

Tabla 4.1.1: Parámetros del algoritmo Classic Sta/Lta

Parámetro	Tipo de dato	Descripción
<code>a</code>	<code>ndarray</code>	Serie temporal de datos sísmicos
<code>nsta</code>	<code>int</code>	Duración de la ventana promedio de tiempo corto en muestras
<code>nlta</code>	<code>int</code>	Duración de la ventana promedio de tiempo largo en muestras

Tipo de devolución:

- NumPy =>ndarray

Devuelve:

- Función característica del STA/LTA clásico

4.1.2.2. Recursive STA/LTA

El primer paso consiste en el cálculo de las funciones características. Por lo general, estas son sensibles a los cambios en la energía, el contenido de frecuencia, la polarización u otras características de la señal en relación con el ruido de fondo en cada estación sísmica individual [53]. Aquí se calculan dos funciones características, una primera sensible a la llegada de la onda P y una segunda sensible a la llegada de la onda S. Los datos sísmicos de tres componentes pueden verse como series de tiempo discretas. Luego se denota la componente Este con $x(j)$, la componente Norte con $y(j)$ y la Vertical con $z(j)$. El valor entero $j = 1$ es el índice de tiempo de la serie. Siguiendo el mismo enfoque propuesto por [54], la función característica CF^P (Ecuación 4.1) se define como la energía de la componente vertical de la traza sísmica:

$$CF^P(j) = z^2(j) \quad (4.1)$$

Para calcular la función de característica CF^S , primero calculamos las trazas analíticas de ambas componentes horizontales definidas en la Ecuación 4.2 y 4.3:

$$X(j) = x(j) + iH\{x(j)\} \quad (4.2)$$

$$Y(j) = y(j) + iH\{y(j)\} \quad (4.3)$$

Donde $i^2 = -1$ y H es la transformada de Hilbert. Entonces se puede calcular la matriz de covarianza instantánea $Q(j)$ como se muestra en la Ecuación 4.4:

$$Q(j) = \begin{pmatrix} X(j)\hat{X}(j) & X(j)\hat{Y}(j) \\ Y(j)\hat{X}(j) & Y(j)\hat{Y}(j) \end{pmatrix} \quad (4.4)$$

donde el sombrero $\hat{}$ denota conjugación compleja. Dado que la matriz $Q(j)$ es hermitiana, tiene, para cada muestra j , dos valores propios positivos reales λ_1 y λ_2 (con $\lambda_1 \geq \lambda_2$). Las funciones características S se definen entonces como se indica en la Ecuación 4.5:

$$CF^S(j) = \lambda_1(j)^2 + \epsilon \quad (4.5)$$

El término ϵ es un pequeño número positivo necesario para superar los problemas numéricos relacionados con el cálculo de STA/LTA cuando $\lambda_1(j)$ tiende a cero. Finalmente, calculamos el valor de STA/LTA usando

las funciones características CF^P y CF^S por separado. El algoritmo **STA/LTA** original de Allen [55] se modifica aquí, mediante la adopción de uno recursivo, que reduce los requisitos de memoria y resulta más suave que el **STA/LTA** estándar en ausencia de señal. Si denotamos como n_s y n_l el número de muestras de las ventanas de tiempo corto y largo, respectivamente, el algoritmo **STA/LTA** recursivo se describe mediante la Ecuación 4.6 y 4.7:

$$STA(j) = k_s[CF(j)] + (1 - k_s)STA(j - 1) \quad (4.6)$$

$$LTA(j) = k_l[CF(j - n_s - 1)] + (1 - k_l)LTA(j - 1) \quad (4.7)$$

donde el índice j varía en el rango entre $h = n_s + n_l$ y la última muestra n_m de la función característica [56].

Si el **STA/LTA** es igual o mayor que el valor preestablecido para la condición verdadera del evento, entonces el evento se declara verdadero y se registran los datos sísmicos.

Método en ObsPy del **STA/LTA** recursivo [57].

```
recursive_sta_lta(a, nsta, nlta)
```

Los parámetros del algoritmo se presentan en la Tabla 4.1.2.

Tabla 4.1.2: Parámetros del algoritmo Recursive STA/LTA

Parámetro	Tipo de dato	Descripción
a	ndarray	Serie temporal de datos sísmicos
nsta	int	Duración de la ventana promedio de tiempo corto en muestras
nlta	int	Duración de la ventana promedio de tiempo largo en muestras

Tipo de devolución:

- NumPy =>ndarray, dtype=float64

Devuelve:

- Función característica del STA/LTA recursivo

4.1.2.3. Delayed STA/LTA

El **STA/LTA** retardado proporciona picos en más fases que los otros métodos en el dominio del tiempo, pero su respuesta al ruido no es tan suave como el **STA/LTA** recursivo y los métodos de envolvente analítica [15].

Método en ObsPy del **STA/LTA** retardado [58].

```
delayed_sta_lta(a, nsta, nlta)
```

En la Tabla 4.1.3 se muestran los parámetros necesarios del algoritmo.

Tabla 4.1.3: Parámetros del algoritmo Delayed STA/LTA

Parámetro	Tipo de dato	Descripción
a	ndarray	Serie temporal de datos sísmicos
nsta	int	Duración de la ventana promedio de tiempo corto en muestras
nlta	int	Duración de la ventana promedio de tiempo largo en muestras

Tipo de devolución:

- NumPy => ndarray

Devuelve:

- Función característica del STA/LTA clásico.

4.1.2.4. Z detector

El detector Z estima la distancia de los datos a la media en unidades de la desviación estándar. Tiene la ventaja de un ajuste automático a la variación del ruido de fondo. Si la variación de fondo es pequeña, se requiere un pequeño cambio en la entrada para un gran cambio en la respuesta. Si la variación de fondo es grande, se requiere un gran cambio de entrada para un cambio de salida significativo. La ventaja de este algoritmo es su ajuste adaptativo al nivel de ruido de fondo. Sin embargo, no mejora la desactivación en el sistema de registro sísmico luego de la llegada de las ondas P [24].

Método en ObsPy del Detector z [59].

```
z_detect(a, nsta)
```

Los parámetros del algoritmo se muestran en la Tabla 4.1.4.

Tabla 4.1.4: Parámetros del algoritmo Z-detector

Parámetro	Tipo de dato	Descripción
a	ndarray	Serie temporal de datos sísmicos
nsta	int	Duración de la ventana promedio de tiempo corto en muestras

4.1.2.5. Baer and Kradolfer picker

El selector de Baer y Kradolfer es una rutina rápida y robusta. Baer y Kradolfer no proponen una evaluación automática de la calidad. Sin embargo, este algoritmo produce selecciones de alta calidad [2].

Método Baer and Kradolfer picker en ObsPy [60].

```
pk_baer(reltrc, samp_int, tdownmax, tupevent, thr1, thr2, preset_len, p_dur,
return_cf=False)
```

En la Tabla 4.1.5 se muestran los parámetros necesarios del algoritmo.

Tabla 4.1.5: Parámetros del algoritmo Baer-and Kradolfer-picker

Parámetro	Descripción
reltrc	Serie temporal de datos sísmicos
samp_int	Número de muestras por segundo
tdownmax	Tiempo permitido para que la función característica caiga por debajo del umbral
tupevent	Tiempo para que la función característica permanezca por encima del umbral
thr1	Umbral para activar la selección
thr2	Umbral para actualizar thr1
preset_len	Número de puntos tomados para la estimación de la varianza
p_dur	Define el intervalo de tiempo para el que se evalúa la amplitud máxima.

Devuelve:

- (pptime, pfm, [cf]) pptime: número de muestra en la que inicia la onda P; dirección pfm del primer movimiento, opcionalmente también la función característica



4.1.2.6. AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

Este algoritmo selecciona los tiempos de inicio mediante un método de criterio de información de Akaike de regresión automática (AR-AIC). Los intervalos de detección se reducen sucesivamente con la ayuda de las relaciones STA/LTA, así como los cálculos de diferencia STA-LTA. Una característica importante de este algoritmo es que requiere relativamente poca configuración de ajustes y específicos del sitio y, por lo tanto, es aplicable a grandes y diversos conjuntos de datos [61].

Método AR-AIC en ObsPy:

```
ar_pick(a, b, c, samp_rate, f1, f2, lta_p, sta_p, lta_s, sta_s, m_p, m_s, l_p, l_s,
s_pick=True)
```

Los parámetros del algoritmo se observan en la Tabla 4.1.6.

Tabla 4.1.6: Parámetros del algoritmo AR-AIC

Parámetro	Tipo de dato	Descripción
a	ndarray	Datos de la señal Z.
b	ndarray	Datos de la señal N.
c	ndarray	Datos de la señal E.
samp_rate	float	Número de muestras por segundo.
f1	float	Frecuencia de la ventana de paso de banda inferior.
f2	float	Frecuencia de la ventana de paso de banda superior.
lta_p	float	Duración de LTA para la llegada de la onda P en segundos.
sta_p	float	Duración del STA para la llegada de la onda P en segundos.
lta_s	float	Duración de LTA para la llegada de la onda S en segundos.
sta_s	float	Duración del STA para la llegada de la ondas S en segundos.
m_p	int	Número de coeficientes AR para la llegada de la onda P.
m_s	int	Número de coeficientes AR para la llegada de la onda S.
l_p	float	Longitud de la ventana de varianza para la llegada de la onda P en segundos.
l_s	float	Longitud de la ventana de varianza para la llegada de la onda S en segundos.
s_pick	bool	Si es True selecciona la fase de la onda S

4.2. Diseño de una herramienta para el análisis de eventos

Esta sección tiene como objetivo el diseño y desarrollo de una herramienta para el análisis de eventos sísmicos empleando los diferentes algoritmos de detección disponibles en ObsPy. El desarrollo de la herramienta se realizó en Python. Esta herramienta permite graficar eventos en diferentes intervalos de tiempo, obtener la información de los eventos sísmicos en un archivo de texto y exportar un archivo miniSeed entre los intervalos de tiempo seleccionados. A continuación se describen detalladamente todas las características.

4.2.1. Características y funcionalidades

En primer lugar, el *software* desarrollado permite seleccionar el algoritmo de detección de una lista desplegable (**seleccionar_metodo()**). Dependiendo del tipo de algoritmo, se solicitan diferentes parámetros. Luego se debe seleccionar un archivo miniSeed para el análisis desde el gestor de archivos, se debe ingresar el factor de conversión y el canal del cual se quiere obtener los eventos. Opcionalmente, se puede agregar hora de inicio y fin del análisis. En caso que se deje en blanco se considera la duración total del archivo de eventos. Con todos estos parámetros se puede obtener la gráfica de los eventos (**graficar_eventos()**). Esta gráfica es interactiva, es decir se puede hacer zoom o guardar la imagen. También permite exportar la información de los eventos obtenidos (hora de inicio y duración) en un archivo de texto (**obtener_eventos()**)



y segmentar el archivo miniSeed entre las horas seleccionadas (**obtener_miniSeed()**). Todas estas funciones se pueden observar en las Figura [4.2.1](#) y [4.2.2](#). Cabe indicar que el desarrollo se liberó para su uso y experimentación y se encuentra disponible en [\[62\]](#).

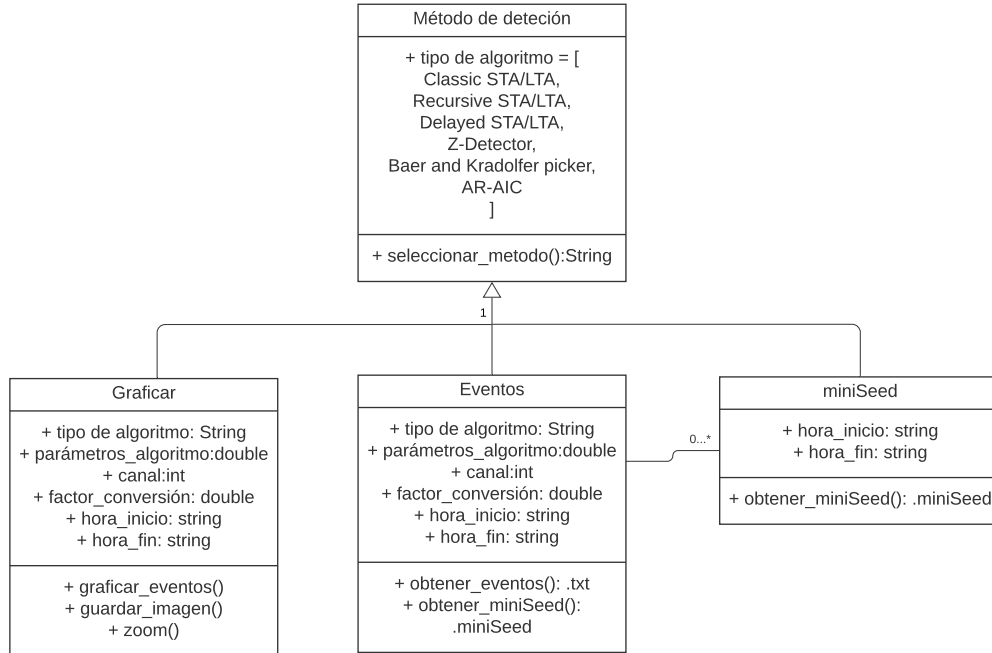


Figura 4.2.1: Diagrama de clases de la herramienta



Figura 4.2.2: Características y funcionalidades de la herramienta

Los casos de uso se muestran en la Figura 4.2.3. En particular, la herramienta es el sistema y el actor el usuario. Para seleccionar el algoritmo únicamente interviene el usuario, pero para los otros casos de uso

como graficar, obtener eventos y miniSeed también interviene la herramienta ya que es la que “prepara” todo para dar los resultados al usuario.

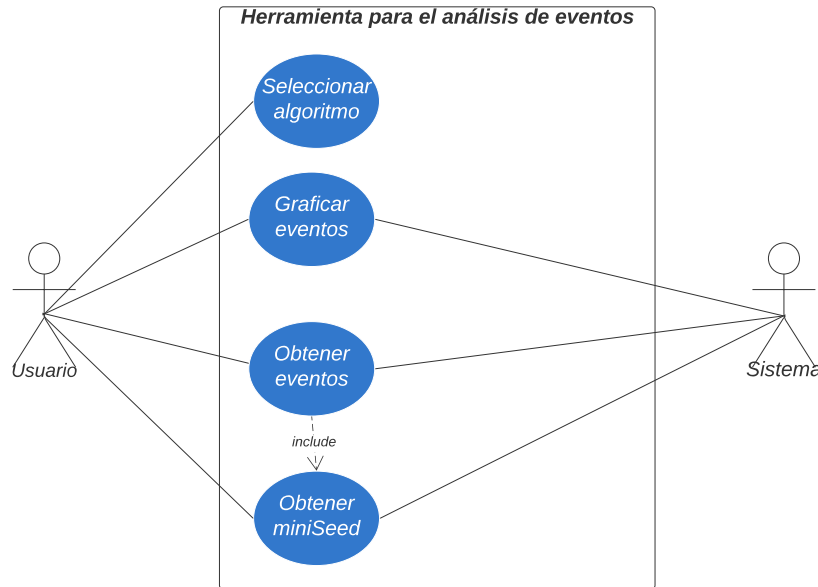


Figura 4.2.3: Casos de uso de la herramienta

4.2.2. Selección de los métodos

Los métodos disponibles en la herramienta son los siguientes:

- Classic STA/LTA
- Recursive STA/LTA
- Delayed STA/LTA
- Z detector
- Baer and Kradolfer picker.
- AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

Para más detalles, dirigirse al Apéndice [A.3](#)

4.2.3. Visualización y presentación de resultados

Luego de que se ingresan todos los parámetros, se presiona la opción Graficar evento, además en la parte inferior se cuenta con una barra de herramientas para interactuar con la gráfica, por ejemplo hacer un zoom o guardar la imagen. En el Apéndice [A](#), se presenta un manual de usuario de la herramienta donde se detallan las funcionalidades y opciones descritas.

4.3. Comparación de algoritmos de detección

Todos los métodos de detección disponibles en ObsPy tienen ventajas y desventajas, por lo que se realizó una comparación tanto en relación a la funcionalidad y configuración de los parámetros requeridos

por cada algoritmo, así como en cuanto a la demanda de recursos computacionales para poder implementar esta solución en una plataforma **Single Board Computer (SBC)**. En concreto, la evaluación y comparación se realizó mediante múltiples experimentos empleando datos disponibles como fuente de entrada de los algoritmos. Estos datos consisten en ficheros recopilados previamente por estaciones acelerográficas. Para una correcta evaluación, se usan registros con eventos sísmicos verdaderos y falsos para que la comparación sea relevante.

4.3.1. Metodología

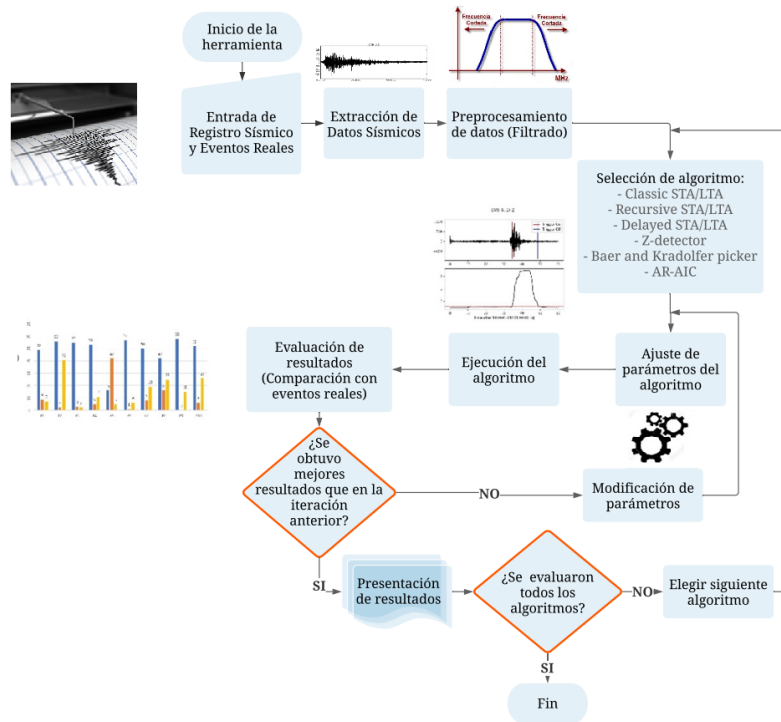


Figura 4.3.1: Metodología

En la Figura 4.3.1 se muestra la metodología utilizada para la comparación de cada algoritmo. En primera instancia se tienen eventos sísmicos previamente obtenidos con estaciones acelerográficas de la Red Sísmica del Austro en formato miniSeed. A estos datos se les realiza un preprocesamiento, que consiste primero en encontrar el factor de conversión adecuado para estos datos ya que el formato miniSeed almacena la información como números enteros, por lo que hay que dividir cada muestra por este factor. Dicho factor se calcula usando la Ecuación 4.8

$$f_c = \frac{max_{valor}}{v_{adquisicion}} \quad (4.8)$$

donde max_{valor} son los valores enteros máximos del sistema de adquisición en función del número de bits del ADC (**Analog-to-Digital Converter (ADC)**) y $v_{adquisicion}$ es el voltaje de adquisición del sistema. A continuación, se aplica un filtro pasa banda para eliminar frecuencias altas y también para excluir el offset que tienen ciertos registros sísmicos generados por los sensores o por la instrumentación. En Obspy los filtros son de tipo Butterworth con la función `FILTER()`. Para filtrar en pasa banda se utiliza la opción “bandpass”. Los parámetros de entrada son “*freqmin*” y “*freqmax*”, que son la esquina inferior y superior en frecuencia en Hz respectivamente. Por defecto el filtro es de 4to orden, se puede especificar con el parámetro “*corners*”. En este caso, se empleó el filtro pasa banda entre 5 Hz y 20 Hz ya que las frecuencias de las señales sísmicas usualmente están en el rango de 5 a 20 Hz. Se podría ampliar desde 2 Hz hasta 30 o máximo

40 Hz, pero valores inferiores o superiores no corresponden a datos sísmicos. Por ejemplo, un filtro paso alto de 40 Hz no es de utilidad en el análisis sísmico, porque se pierden todas las frecuencias de interés. La línea de código utilizada para aplicar este filtro se indica en la Ecuación 4.9:

$$\text{tr.filter('bandpass', freqmin=5, freqmax=20)} \quad (4.9)$$

En la Figura 4.3.2 se muestra el efecto de aplicar diferentes filtros sobre la señal. Con el filtro pasa banda (4.3.2b) se logra eliminar componentes del ruido de la señal original. En cambio con el filtro pasa alto, se afecta directamente el nivel de voltaje de la señal (4.3.2c) ya que se reduce de 0,15V a 0,03V.

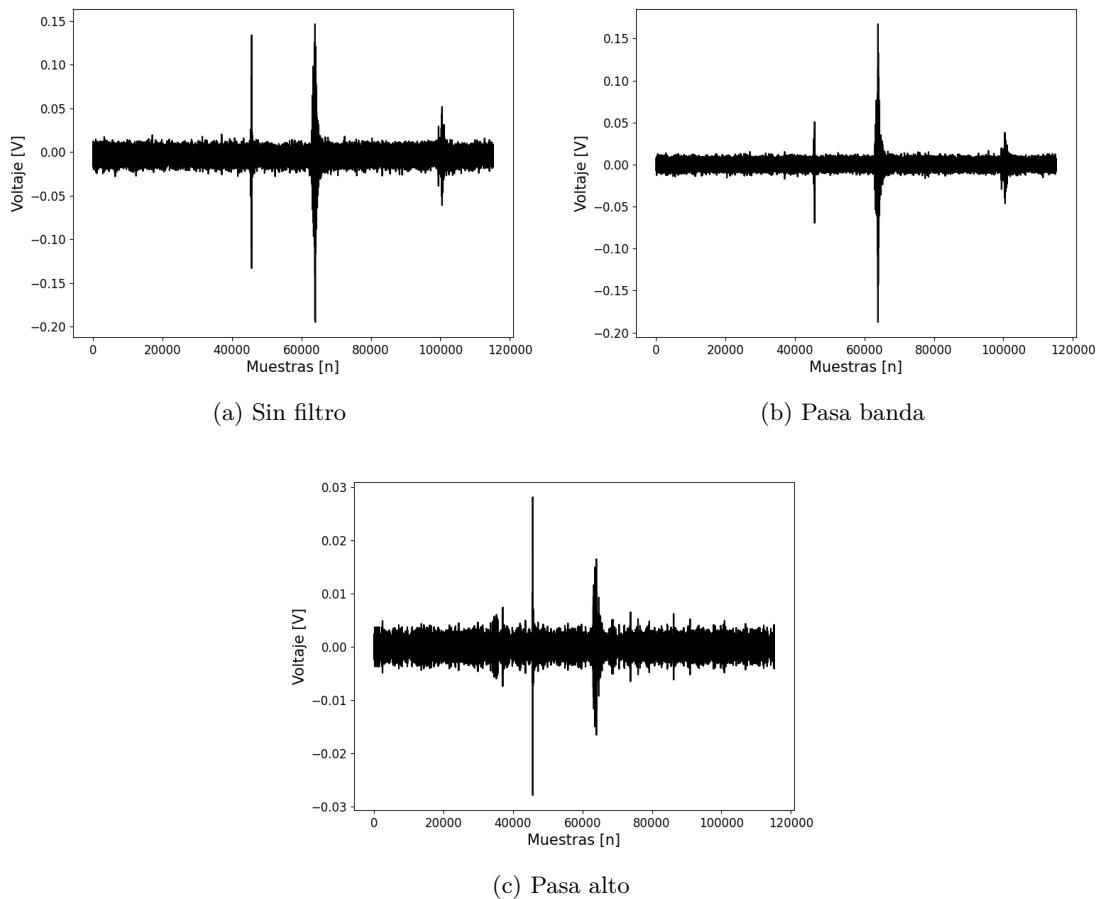


Figura 4.3.2: Ejemplo de la aplicación de filtros a la señal

En el procesamiento se utilizan los diferentes métodos de detección en ObsPy. Para cada método se prueban diferentes parámetros de STA y LTA y se obtienen los primeros resultados. Se analiza la capacidad de estos para detectar eventos sísmicos y en base a esto se ajustan los parámetros para obtener mejores resultados. Una vez que se realizó este ajuste, se obtienen los eventos detectados y se los compara con eventos reales detectados. Con esto se tiene información de eventos totales detectados, no detectados (eventos sísmicos que no fueron detectados), eventos falsos (ruido detectado como evento) y el total de eventos obtenidos. En base a estos resultados se elige el algoritmo óptimo para ser implementado en la detección en tiempo real.

4.3.2. Análisis de parámetros y métricas

Los parámetros utilizados para evaluar a los algoritmos son los siguientes: total de eventos obtenidos, eventos detectados, eventos no detectados (en comparación con los eventos reales previamente obtenidos) y eventos falsos detectados (eventos marcados como ruido originalmente pero que se detectan con los algoritmos).

Además, se tiene el porcentaje de eventos detectados, que viene a ser la relación entre los eventos detectados por el algoritmo y el total de eventos reales (Ecuación 4.10):

$$\%Eventos_Detectados = \frac{Eventos_detectados}{Eventos_reales} \times 100\% \quad (4.10)$$

Por otra parte, el porcentaje de eventos detectados correctamente es la relación entre los eventos detectados con el método y el total de eventos obtenidos con el algoritmo de detección (Ecuación 4.11):

$$\%Eventos_Detectados_Correctamente = \frac{Eventos_detectados}{Total_de_eventos_obtenidos} \times 100\% \quad (4.11)$$

Finalmente el error es la fracción entre los eventos que no se lograron detectar con el algoritmo y el total de eventos reales (Ecuación 4.12):

$$\%Error = \frac{Eventos_no_detectados}{Eventos_reales} \times 100\% \quad (4.12)$$

4.3.3. Configuración de métodos de detección

Los valores adecuados de los parámetros para cada algoritmo dependen de diferentes factores, como el nivel de ruido promedio, la sensibilidad de los sensores sísmicos, la distancia a la que se encuentran las estaciones sísmicas, etc. Por eso los valores adecuados no van a ser siempre los mismos en todos los casos. Sin embargo, existen recomendaciones que se pueden tomar como referencia para empezar a probar los parámetros. Los valores iniciales de los parámetros se basan en lo que se menciona en [13] y en los ejemplos iniciales de ObsPy [63]. Según Trnkoczy, el parámetro STA sirve como un filtro de señal, y cuanto menor sea la duración seleccionada mayor será la sensibilidad y viceversa. Para eventos regionales, un valor típico de STA está entre 1 y 2 segundos y para eventos locales es más común valores entre 0,5 y 0,3 s. De igual forma, un LTA corto puede mitigar los disparos falsos debidos al ruido generado por las personas. Un valor común para este parámetro es de 60 segundos. Hay que tener en cuenta que los algoritmos disponibles en ObsPy reciben los parámetros NSTA y NLTA en número de muestras, por lo que los valores recomendados se deben convertir tomando en cuenta la frecuencia de muestreo de los eventos. Pero la herramienta desarrollada recibe los parámetros de STA y LTA en segundos.

Por otra parte, el umbral de *trigger* determina en mayor medida qué eventos se registrarán y cuáles no. Cuanto mayor sea el valor que se establezca, más eventos no se registrarán, pero se producirán menos disparos falsos. Un nivel de umbral inicial de STA/LTA es de 4. De igual forma, un nivel de umbral de *detrigger* STA/LTA demasiado bajo es ocasionalmente peligroso. El mismo puede causar registros muy largos o incluso interminables. Un valor inicial típico del nivel de umbral de *detrigger* es de 2 a 3.

4.3.3.1. Classic STA/LTA

Se realizaron 10 pruebas con diferentes parámetros, como se observa en la Tabla. 4.3.1. Como valores iniciales se tomaron en cuenta los ejemplos disponibles en [63].



Tabla 4.3.1: Parámetros del algoritmo Classic STA/LTA en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
trigger_on	1,5	1,3	1,2	1,15	1,1	1,15	1,15	1,15	1,15	5
trigger_off	0,5	0,7	0,8	0,85	0,85	0,85	0,85	0,85	0,5	0,5
sta	5	5	50	50	500	40	40	40	10	0,5
lta	10	10	100	100	1000	70	200	500	70	70

4.3.3.2. Delayed STA/LTA

Para este algoritmo se tuvo varios problemas para encontrar los valores adecuados de sta y lta. Se realizaron más pruebas de las que se muestran en la Tabla 4.3.2, sin embargo, solo se publicaron los más relevantes. Lo que se hizo es obtener los eventos en los intervalos de tiempo conocidos de los eventos para mejorar los resultados.

Tabla 4.3.2: Parámetros del algoritmo Delayed STA/LTA en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
trigger_on	5	1	1	18	1.4	103	2	103	124	101
trigger_off	10	5	20	30	1.5	100	3	100	121	100
sta	2,1	5,2	21	1,7	80	20	300	4	10	5
lta	2,3	5,5	23	2	90	2000	350	400	1200	500

4.3.3.3. Recursive STA/LTA

De igual forma para este método, se realizaron 10 pruebas variando los parámetros en cada una. Los valores usados en cada prueba se muestran en la Tabla 4.3.3. En este caso, para los valores iniciales de los parámetros se tomaron como referencia experimentos desarrollados anteriormente con otros eventos y los valores de sta=8,5 y lta =14 arrojaron mejores resultados.

Tabla 4.3.3: Parámetros del algoritmo Recursive STA/LTA en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
trigger_on	1,5	1,15	2	1,3	1,2	1,1	1,1	1,1	1,1	1,1
trigger_off	0,5	0,5	0,5	0,5	0,7	0,8	0,8	0,8	0,8	0,95
sta	8,5	100	1000	10	20	40	40	20	6	20
lta	14	200	3000	70	50	80	120	100	12	40

4.3.3.4. Z Detector

Este método no recibe como entrada el parámetro lta, como los algoritmos basados en STA/LTA, pero de igual forma se realizaron 10 experimentos variando los parámetros en cada una. Los valores usados en cada prueba se muestran en la Tabla 4.3.4. Algo que se debe considerar en este algoritmo es el tiempo de procesamiento. Mientras en los algoritmos previos el tiempo de procesamiento fue en promedio 1-2 minutos. Por otra parte, en el método Z-Detector, conforme se incrementaba el valor de sta, el tiempo en el que se ejecutaba el algoritmo aumentaba. Por esta razón se agregó este parámetro a la Tabla 4.3.4. El tiempo es un parámetro importante ya que, si se desea implementar un algoritmo en tiempo real, no se puede esperar que un resultado se demore 30 minutos. El valor inicial de las pruebas se tomó como referencia del ejemplo disponible en [63].



Tabla 4.3.4: Parámetros del algoritmo Z Detector en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
trigger_on	1	2	1	1	0,5	0,5	0,5	0,2	0,1	0,1
trigger_off	0,5	1	0,5	0,5	0,1	0,1	0,1	0,1	0	0
sta	10	50	200	2	0,5	1	7	2	26	8
tiempo	56 s	10 min 56s	32 min 34s	44s	23s	30s	1 min 43s	44s	4 min 25s	1 min 30s

4.3.3.5. Baer and Kradolfer picker

Para este algoritmo se tienen más parámetros que para los métodos basados en STA/LTA. Lo difícil de este algoritmo es que una vez que se encuentra un parámetro que puede dar buenos resultados, se debe segmentar el archivo de eventos por intervalos de tiempo. Para este caso, como ya se conocen los eventos que se debe detectar y la hora en que se producen, lo que se hace es segmentar el archivo en los intervalos donde se conoce que existe un evento. Por lo que para probar este algoritmo se requiere una cantidad de tiempo considerable. Los parámetros utilizados para cada prueba se muestran en la Tabla 4.3.5.

Tabla 4.3.5: Parámetros del algoritmo Baer and Kradolfer picker en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
trigger_on	1E+08	7	1E+08	1E+08	7	10	7	1E+08	12000	20
trigger_off	2E+06	12	2E+06	2E+06	12	12	12	2E+06	10000	30
thr1	10	7	10	300	10	7	10	10	7	7
thr2	2	12	200	2	20	12	20	2	12	15
tdownmax	1	20	1	1	10	20	10	1	20	20
tupevent	640	60	640	640	2	60	97	640	60	60
preset_len	640	100	640	640	384	100	100	6	200	100
p_dur	6	100	6	6	100	100	100	6	200	100

4.3.3.6. AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

En lo que respecta a AR-AIC, inicialmente se probó con los valores disponibles en [63] y en base a los resultados obtenidos se empezaron a modificar los diferentes parámetros. De igual forma que en el algoritmo Baer and Kradolfer-picker, se debe segmentar el archivo de eventos por intervalos de tiempo. Por lo tanto, este algoritmo también requiere una cantidad de tiempo considerable. Los parámetros utilizados para cada prueba se muestran en la Tabla 4.3.6.

Tabla 4.3.6: Parámetros del algoritmo AR-AIC en cada prueba

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
f1	1	1	1	1	1	1	1	1
f2	20	20	20	20	8	20	20	20
lta_p	1	2	10	100	1	8	8	8
sta_p	0,1	0,2	1	10	0,1	2	2	2
lta_s	4	8	40	400	4	6	6	6
sta_s	1	2	10	100	1	3	3	3
m_p	2	4	20	200	2	2	10	2
m_s	8	16	80	800	8	8	15	8
l_p	0,1	0,2	1	10	0,1	0,1	0,1	1
l_s	0,2	0,4	2	20	0,2	0,2	0,2	3

4.3.4. Procesamiento de eventos reales mediante los métodos de detección

A continuación se presentan los resultados obtenidos con los parámetros configurados en la sección 4.3.3 para cada método de detección automática.

4.3.4.1. Classic STA/LTA

En la Tabla 4.3.7 se observan los resultados obtenidos con diferentes parámetros del algoritmos classic STA/LTA.

Tabla 4.3.7: Resultados del algoritmo Classic STA/LTA

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
Detectados	49	56	55	53	16	57	50	42	58	52
No detectados	9	2	3	5	42	1	8	16	0	6
Eventos falsos detectados	7	41	2	11	5	6	19	25	15	26
Total eventos reales	58	58	58	58	58	58	58	58	58	58
Total eventos obtenidos	56	97	57	64	21	63	69	67	73	78
% Eventos detectados	84 %	97 %	95 %	91 %	28 %	98 %	86 %	72 %	100 %	90 %
% Detectados correctamente	88 %	58 %	96 %	83 %	76 %	90 %	72 %	63 %	79 %	67 %
% Error	16 %	3 %	5 %	9 %	72 %	2 %	14 %	28 %	0 %	100 %

El análisis gráfico de los resultados obtenidos con diferentes parámetros se presenta en la Figura 4.3.3. Con la prueba 9 (P.9) se logró detectar todos los eventos reales, sin embargo, el número de eventos obtenidos es de 73, por lo que tenemos un total de 15 eventos falsos, lo que no sería conveniente. Considerando el análisis de esta manera, la prueba que mejores resultados ofrece es la P.6 ya que solo faltó detectar un evento verdadero, se detectaron 6 eventos falsos y se obtuvieron un total de 57 eventos reales. En la Figura 4.3.4 se observa la gráfica de los eventos obtenidos con P.6.

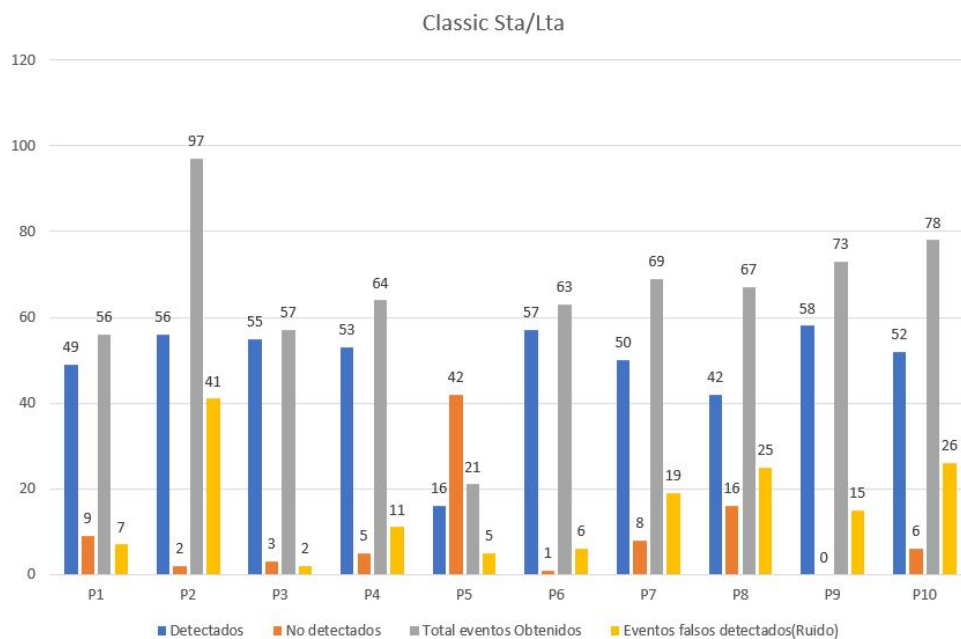


Figura 4.3.3: Resultados obtenidos con el algoritmo Classic STA/LTA

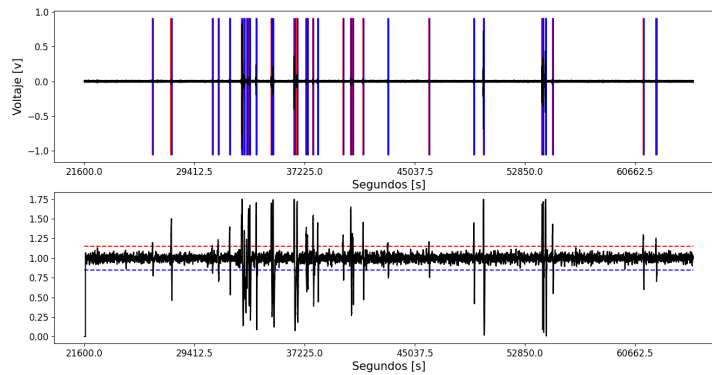


Figura 4.3.4: Gráfica de los eventos obtenidos con el mejor resultado en el algoritmo Classic STA/LTA

4.3.4.2. Delayed STA/LTA

Los resultados obtenidos con los diferentes parámetros de la Tabla 4.3.2, se presentan en la Figura 4.3.5. En la Tabla 4.3.8 se muestran los resultados de cada prueba. El mejor resultado se obtuvo con P.10, con un 40 % de eventos detectados. Del total de eventos detectados, se tiene un 60 % de error respecto a todos los eventos.

Tabla 4.3.8: Resultados del algoritmo Delayed STA/LTA

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
Detectados	21	15	20	19	2	5	12	13	15	23
No detectados	37	43	38	39	56	53	46	45	43	35
Eventos falsos detectados	37	43	38	39	56	53	46	45	43	35
Total eventos reales	58	58	58	58	58	58	58	58	58	58
Total eventos obtenidos	58	58	58	58	58	58	58	58	58	58
% Eventos detectados	36 %	26 %	34 %	33 %	3 %	9 %	21 %	22 %	26 %	40 %
% Detectados correctamente	36 %	26 %	34 %	33 %	3 %	9 %	21 %	22 %	26 %	40 %
% Error	64 %	74 %	66 %	67 %	97 %	91 %	79 %	78 %	74 %	60 %

Los resultados obtenidos con el algoritmo Delayed STA/LTA no son los mejores, como se observa en la Figura 4.3.5. En P.10 se obtuvieron un total de 58 eventos, de los cuales 23 son correctos, por lo tanto, se tiene 35 eventos falsos. En la Figura 4.3.6 se muestra la gráfica de los eventos obtenidos con P.10.

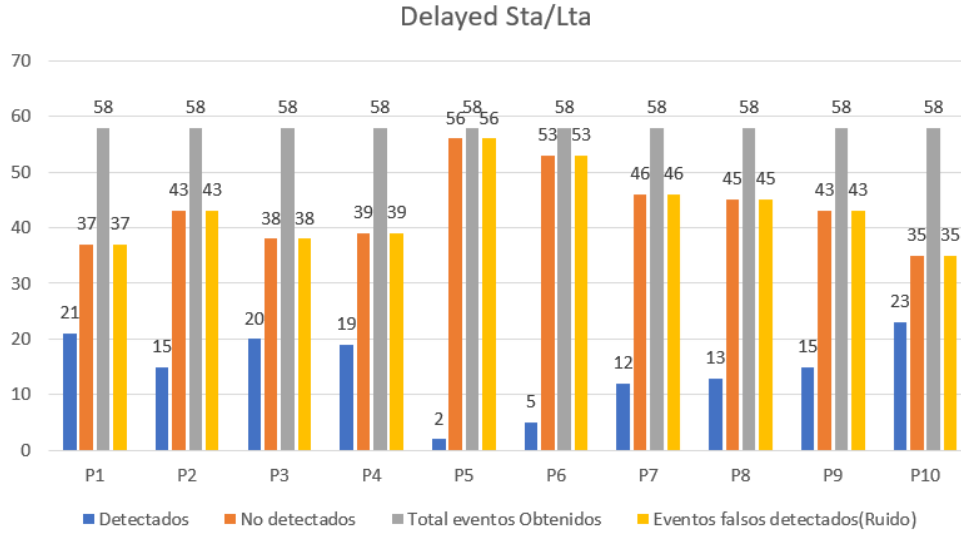


Figura 4.3.5: Resultados obtenidos con el algoritmo Delayed Sta/Lta

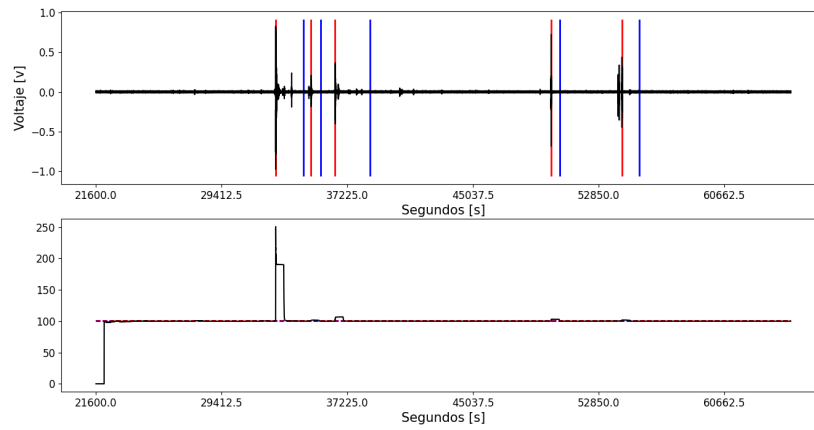


Figura 4.3.6: Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Delayed Sta/Lta

**4.3.4.3. Recursive STA/LTA**

Los resultados obtenidos con este algoritmo se muestran en la Figura 4.3.7 y en la Tabla 4.3.9. En base a los valores de los parámetros utilizados en P.1 se ajustaron los parámetros hasta mejorar los resultados. Se observa claramente que P.10 fue la prueba que mejores resultados dió, ya que se obtuvieron 64 eventos en total, de los cuales 49 eran eventos correctos, y 9 no se detectaron. Esto nos da un 84% de eventos detectados, y un 77% de efectividad (porcentaje de eventos reales detectados de acuerdo con el total de eventos obtenidos, Tabla 4.3.9). En la Figura 4.3.8 se muestran los eventos obtenidos con P.10 en el intervalo de 6:00 a 18:00 horas.

Tabla 4.3.9: Resultados del algoritmo Recursive STA/LTA

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
Detectados	14	9	5	22	23	21	22	24	20	49
No detectados	44	49	53	36	35	37	36	34	38	9
Eventos falsos detectados	2	3	0	5	5	4	7	16	29	15
Total eventos reales	58	58	58	58	58	58	58	58	58	58
Total eventos obtenidos	16	12	5	27	28	25	29	40	49	64
% Eventos detectados	24 %	16 %	9 %	38 %	40 %	36 %	38 %	41 %	34 %	84 %
% Detectados correctamente	88 %	75 %	100 %	81 %	82 %	84 %	76 %	60 %	41 %	77 %
% Error	76 %	84 %	91 %	62 %	60 %	64 %	62 %	59 %	66 %	16 %

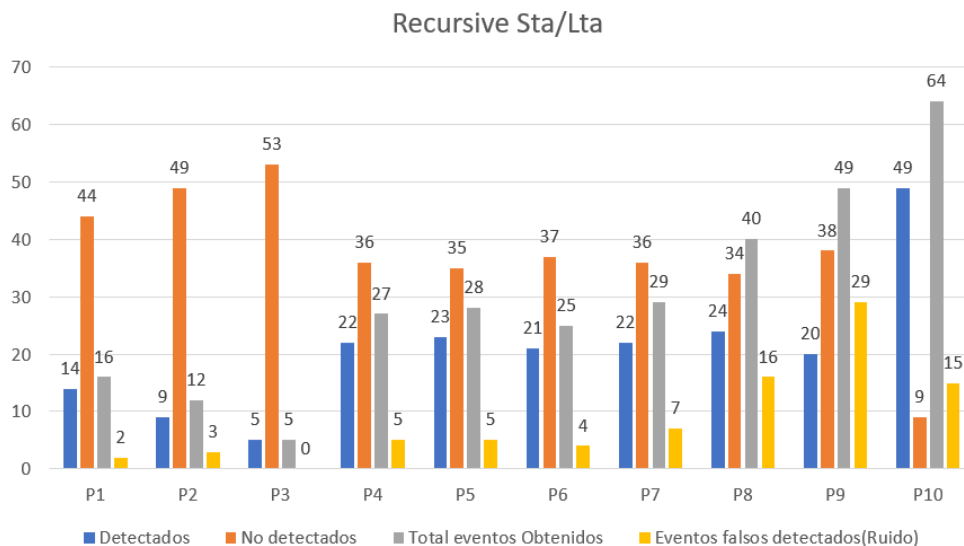


Figura 4.3.7: Resultados obtenidos con el algoritmo Recursive STA/LTA

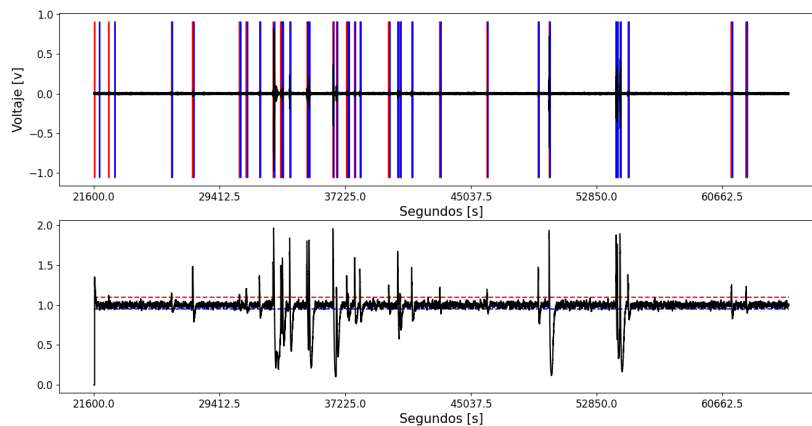


Figura 4.3.8: Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Recursive STA/LTA

4.3.4.4. Z Detector

Los resultados obtenidos con este algoritmo se observan en la Tabla 4.3.10 y en la Figura 4.3.9. En general, todas las pruebas tuvieron un porcentaje de eventos detectados bajo, entre 19 % y 38 %. La máxima cantidad de eventos detectados (22) de un total de 24 eventos obtenidos, fue en P.10. Esta da un 92 % de eventos detectados correctamente. En relación con el total de eventos reales, se tiene un 62 % de error, lo que es un valor alto, pero para este algoritmo es el mejor resultado obtenido. En la Figura 4.3.10 se muestran los eventos obtenidos en P.10 en el intervalo de 6:00 a 18:00 horas.

Tabla 4.3.10: Resultados del algoritmo Z Detector

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P. 9	P. 10
Detectados	11	6	5	16	19	21	18	21	19	22
No detectados	47	52	53	42	39	37	40	37	39	36
Eventos falsos detectados	1	0	0	3	76	8	2	6	2	2
Total eventos reales	58	58	58	58	58	58	58	58	58	58
Total eventos obtenidos	12	6	5	19	95	29	20	21	21	24
% Eventos detectados	19 %	10 %	9 %	28 %	33 %	36 %	31 %	36 %	33 %	38 %
% Detectados correctamente	92 %	100 %	100 %	84 %	20 %	72 %	90 %	100 %	90 %	92 %
% Error	81 %	90 %	91 %	72 %	67 %	64 %	69 %	64 %	67 %	62 %

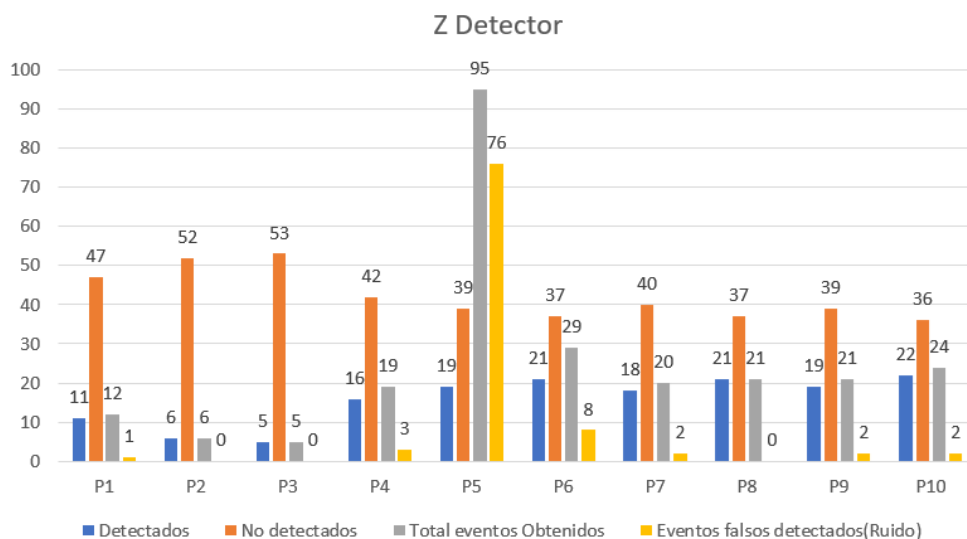


Figura 4.3.9: Resultados obtenidos con el algoritmo Z Detector

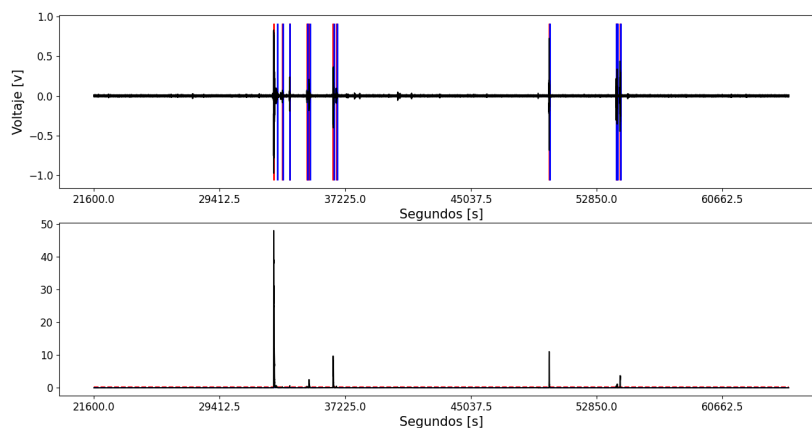


Figura 4.3.10: Gráfica de los eventos obtenidos con mejor resultado en el algoritmo Z Detector



4.3.4.5. Baer and Kradolfer picker

Los resultados del algoritmo Baer and Kradolfer picker se observan en la Tabla 4.3.11 y en la Figura 4.3.11. El mejor resultado con este método se obtuvo en P.9. En esta prueba se detectaron 39 eventos de un total de 58, por lo que se tiene un error de 33%. En la Figura 4.3.12 se muestra uno de los eventos más relevantes detectados con P.9.

Tabla 4.3.11: Resultados del algoritmo Baer and Kradolfer picker

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8	P.9	P. 10
Detectados	5	7	2	3	10	36	25	6	39	30
No detectados	53	51	56	55	48	22	33	52	19	28
Eventos falsos detectados	7	47	0	0	2	22	33	52	19	28
Total eventos reales	58	58	58	58	58	58	58	58	58	58
Total eventos obtenidos	12	54	2	3	12	58	58	58	58	58
% Eventos detectados	9 %	12 %	3 %	5 %	17 %	62 %	43 %	10 %	67 %	52 %
% Detectados correctamente	42 %	13 %	100 %	100 %	83 %	62 %	43 %	10 %	67 %	52 %
% Error	91 %	88 %	97 %	95 %	83 %	38 %	57 %	90 %	33 %	48 %

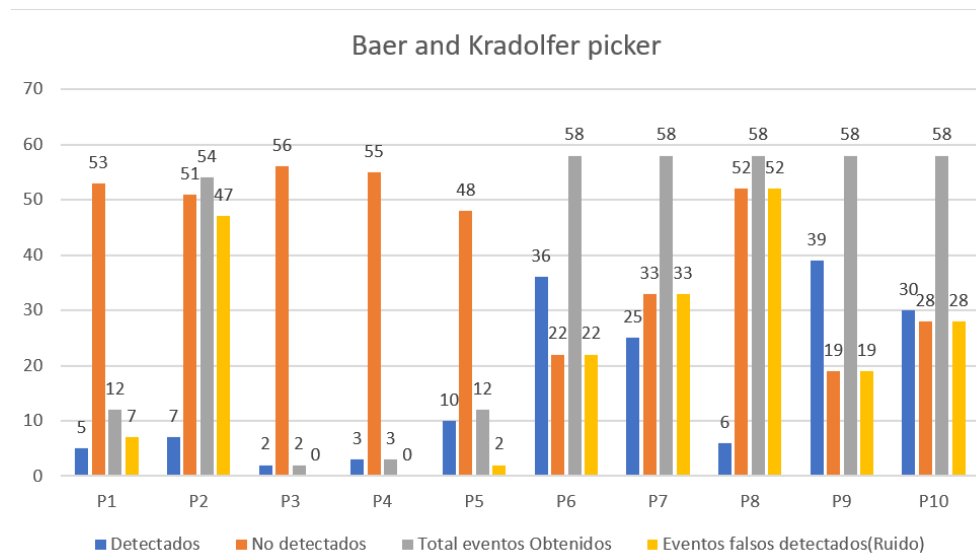


Figura 4.3.11: Resultados obtenidos con el algoritmo Baer and Kradolfer picker

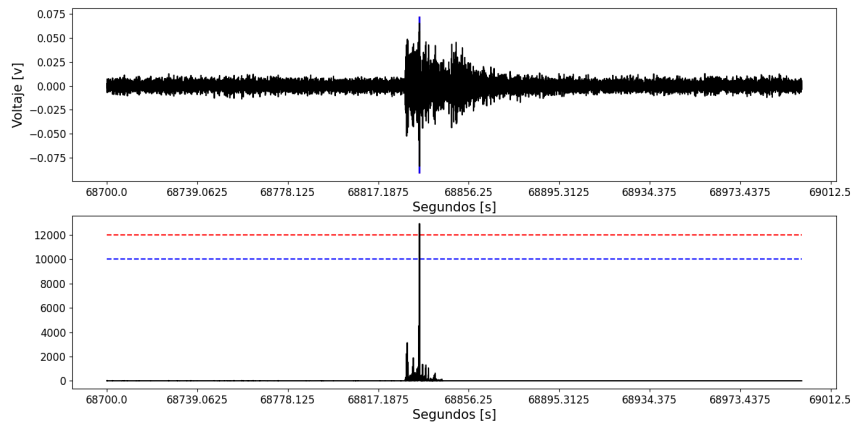


Figura 4.3.12: Gráfica de uno de los eventos obtenidos con el algoritmo Baer and Kradolfer picker

4.3.4.6. AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

Los resultados obtenidos con este algoritmo se muestran en la Tabla 4.3.12 y en la Figura 4.3.13. Se puede observar que se realizaron 8 pruebas, esto debido a que en promedio se tenían resultados parecidos, a excepción de P.4 y P.7. El mejor resultado se da en P.6 ya que se lograron detectar 28 eventos. Por lo tanto, se tiene 48 % de eventos detectados del total. Y del total de eventos obtenidos con el algoritmo, el 78 % de eventos es correcto. Esto se observa claramente en la Figura 4.3.13. En la figura 4.3.14 se muestra uno de los eventos obtenidos con el algoritmo AR-AIC.

Tabla 4.3.12: Resultados del algoritmo AR-AIC

Parámetro	P.1	P. 2	P. 3	P. 4	P. 5	P. 6	P. 7	P. 8
Detectados	24	26	26	6	22	28	0	27
No detectados	34	32	32	52	36	30	58	31
Eventos falsos detectados	13	20	11	3	25	8	5	17
Total eventos reales	58	58	58	58	58	58	58	58
Total eventos obtenidos	37	46	37	9	47	36	5	44
% Eventos detectados	41 %	45 %	45 %	10 %	38 %	48 %	0 %	47 %
% Detectados correctamente	65 %	57 %	70 %	67 %	47 %	78 %	0 %	61 %
% Error	59 %	55 %	55 %	90 %	62 %	52 %	100 %	53 %

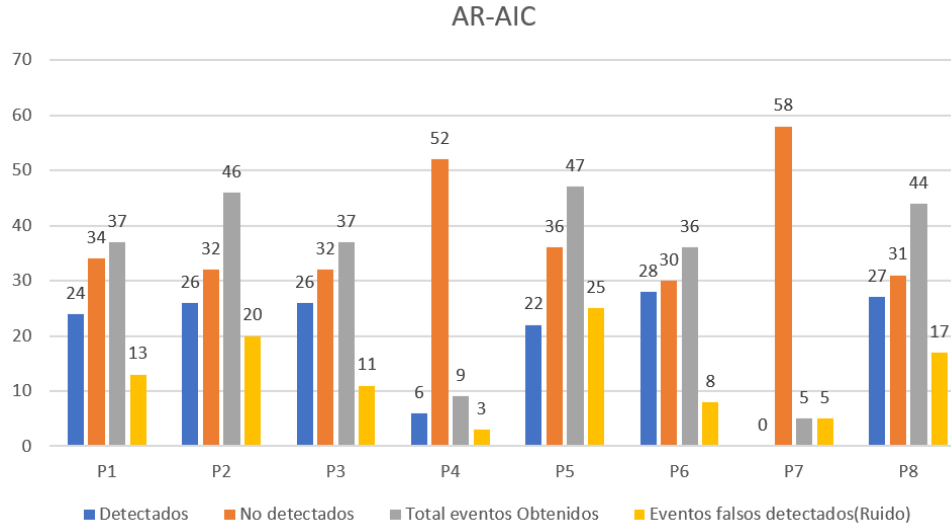


Figura 4.3.13: Resultados obtenidos con el algoritmo AR-AIC

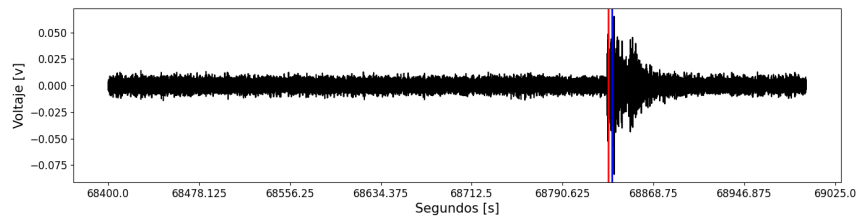


Figura 4.3.14: Gráfica de uno de los eventos obtenidos con el algoritmo AR-AIC

4.3.5. Análisis de resultados

Los resultados del algoritmo Classic STA/LTA se muestran en la Figura 4.3.3. Como se puede observar, con la prueba 9 (P.9) se detectaron todos los eventos reales, sin embargo, el número de eventos obtenidos es de 73, por lo que se tiene un total de 15 eventos falsos, lo cual no es conveniente. Con este análisis, la prueba que mejores resultados ofrece es la P.6, en la cual solo un evento no se detectó, se detectaron 6 eventos falsos y se obtuvo un total de 57 eventos reales. El algoritmo Delayed STA/LTA generó los resultados mostrados en la Figura 4.3.5. El mejor resultado se obtuvo en el experimento P.10, con un 40% de eventos detectados. Del total de eventos detectados, se tiene un 60% de error respecto a todos los eventos. Los resultados obtenidos con el algoritmo Delayed STA/LTA no son los mejores. En P.10 se detectó un total de 58 eventos, de los cuales 23 son correctos, por lo tanto, se tienen 35 eventos falsos.

Con el último método basado en STA/LTA, el recursivo, se consiguieron los resultados de la Figura 4.3.7. En base a los valores utilizados en P.1 se ajustaron los parámetros hasta mejorar los resultados. Se observa claramente que P.10 fue la prueba con mejores resultados, ya que se lograron 64 eventos en total, de los cuales 49 eran eventos correctos, y 9 no se detectaron. Esto nos da un 84% de eventos detectados, y un 77% de efectividad (porcentaje de eventos reales detectados de acuerdo con el total de eventos obtenidos). Continuando con la evaluación experimental, los resultados obtenidos con el algoritmo Z Detector se observan en la Figura 4.3.9. En general, todas las pruebas tuvieron un porcentaje de eventos detectados bajo, entre 19% y 38%. La máxima cantidad de eventos detectados (22) de un total de 24 eventos obtenidos, fue en P.10. Esta da un 92% de eventos detectados correctamente. En relación con

el total de eventos reales, se tiene un 62 % de error, lo que es un valor alto, comparado con los demás algoritmos.

Para el algoritmo Baer and Kradolfer picker se generaron los resultados de la Figura 4.3.11. El mejor resultado con este método se obtuvo en P.9. En esta prueba se detectaron 39 eventos de un total de 58, por lo que se tiene un error de 33 %. Finalmente, para el algoritmo AR-AIC se puede observar que se realizaron 8 pruebas, esto debido a que los resultados fueron parecidos a pesar de modificar los parámetros, a excepción de P.4 y P.7. El mejor resultado se obtuvo en P.6 (Figura 4.3.13) ya que se lograron detectar 28 eventos. Por lo tanto, se tiene 48 % de eventos detectados del total. Además, del total de eventos obtenidos con el algoritmo, el 78 % de eventos es correcto.

4.4. Eficiencia computacional

Para concluir el capítulo y evaluar la eficiencia computacional, en la Figura 4.4.1 se muestra el uso de la CPU del dispositivo SBC para cada método de detección. Para obtener estos resultados se empleó un registro sísmico de 30 minutos para cada algoritmo. Además, se utilizaron los mejores parámetros obtenidos en la sub sección anterior y, finalmente, se capturó el uso de la CPU de la Raspberry Pi durante 30 segundos ya que en promedio el tiempo de procesamiento de este registro fue de 10 segundos. Como se puede apreciar en las curvas, los algoritmos que más costo computacional demandan fueron el Z Detector y AR-AIC. Mientras que el método Classic STA/LTA, que generó los mejores resultados, presentó un pico máximo de 18 % de uso de CPU. En cuanto al método que menor costo computacional requiere es el de Baer and Kradolfer picker, en este caso el máximo valor obtenido fue de 17,1 %. Finalmente, cabe destacar que en ningún caso se presentaron sobrecargas en la CPU, por consiguiente, el uso de la plataforma SBC es adecuado para la comparación de los algoritmos de detección de eventos sísmicos.

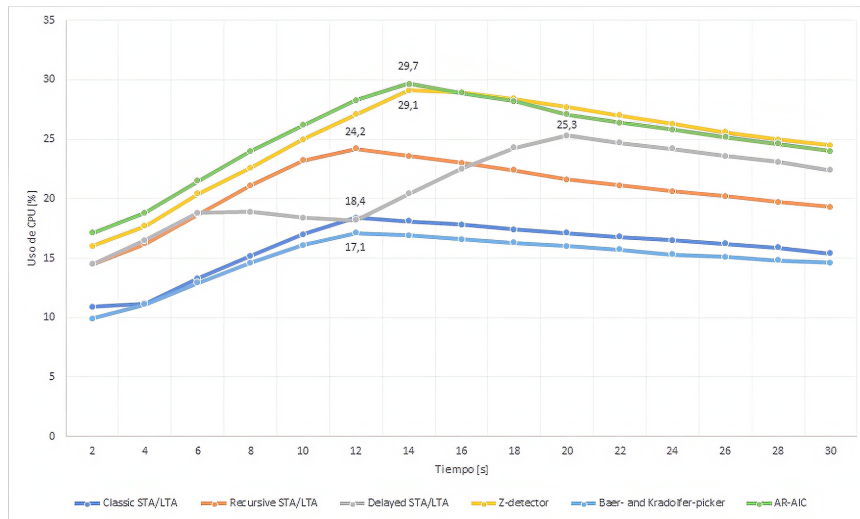


Figura 4.4.1: Uso de CPU en la herramienta de comparación de algoritmos

4.5. Conclusiones

En este capítulo se presentó un estudio de evaluación y comparación de algoritmos para la detección de eventos sísmicos. Con tal propósito se desarrolló una herramienta de *software* que permite leer un registro sísmico, detectar con diversos algoritmos los eventos sísmicos de dicho registro y la comparación de resultados de forma rápida y sencilla. Además, dispone de las funcionalidades de análisis en intervalos de tiempo, obtener gráficas de los resultados, exportación de los resultados y creación de nuevos archivos miniSeed con los datos de los intervalos de interés. La decisión de liberar la herramienta tiene la finalidad



de que siga creciendo y mejorando con aportes de personas interesadas en este ámbito y que pueda ser usada para desarrollar más estudios sobre la detección de eventos sísmicos.

Con respecto a los experimentos realizados para cada algoritmo, se obtuvo como resultado que el método Classic STA/LTA es el que mejores resultados generó, bajo las condiciones de los registros sísmicos de la Red Sísmica del Austro (nivel de ruido promedio, sensibilidad de los sensores sísmicos, distancia a la que se encuentran las estaciones sísmicas, magnitud). Con este algoritmo se detectaron el 98 % de eventos reales. Por otro lado, para otros algoritmos como el Baer and Kradolfer Picker se obtuvo una baja tasa de detección debido a que varios de los eventos tienen una baja magnitud.

En cuanto al uso de la plataforma SBC (Raspberry Pi 3) se encontró que, los algoritmos que más costo computacional requieren son el Z Detector y AR-AIC. Mientras que el método que menor uso de CPU tiene es el de Baer and Kradolfer picker. Además, mientras los otros algoritmos toman en promedio 1 minuto para generar resultados, en el Z Detector se obtuvo como máximo 32 minutos de procesamiento para obtener resultados. Por lo que no sería óptimo para una implementación en tiempo real.

Capítulo 5

Detección de eventos en tiempo real

En el presente capítulo se describe la herramienta utilizada para la detección de eventos en tiempo real. Una vez que en el Capítulo anterior se realizó la comparación de algoritmos, se selecciona el que mejor rendimiento obtuvo. Este algoritmo se utiliza para realizar la implementación en tiempo real. Para esto se desarrolló una herramienta de software. Finalmente, se prueba la eficiencia computacional de la herramienta, evaluando la carga de CPU sobre la Raspberry Pi.

5.1. Selección del algoritmo de detección

En base a los resultados obtenidos en el Capítulo 4, en la Tabla 5.1.1 se resumen los mejores resultados obtenidos para cada algoritmo. De los 6 métodos comparados, el algoritmo Classic STA/LTA obtuvo el mayor rendimiento, ya que se lograron detectar 57 eventos de un total de 58 eventos reales y se obtuvieron 6 eventos falsos, lo cual implica un 90 % de efectividad y 98 % de eventos detectados correctamente. Por lo tanto, para la detección de eventos en tiempo real, se seleccionó el algoritmo Classic STA/LTA

En la Figura 5.1.1 se presenta uno de los eventos sísmicos más relevantes, obtenido con el algoritmo Classic STA/LTA. En la parte superior de la figura se puede observar la llegada de la onda P (primaria) con la línea vertical de color rojo y el final de la onda S (secundaria) con la línea azul. En la parte inferior se muestra la función característica y el umbral On/Off configurado.

Tabla 5.1.1: Mejores resultados de cada algoritmo

Algoritmo	Eventos Detectados	Eventos Detectados (%)	Error (%)	Efectividad (%)
Classic STA/LTA	57	98 %	2 %	90 %
Delayed STA/LTA	23	40 %	60 %	40 %
Recursive STA/LTA	49	84 %	16 %	77 %
Z Detector	22	38 %	62 %	92 %
Baer-and Kradolfer-picker	39	67 %	33 %	67 %
AR-AIC	28	48 %	52 %	78 %

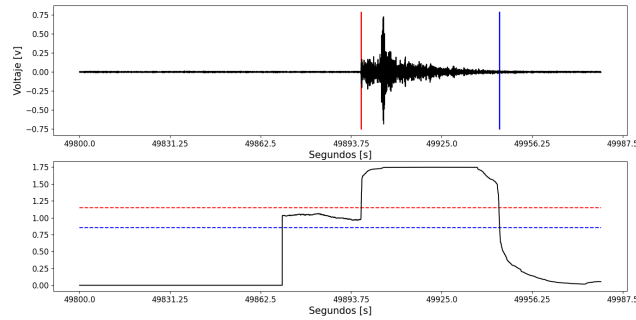


Figura 5.1.1: Evento sísmico obtenido con el algoritmo Classic STA/LTA

5.2. Diseño de una herramienta para el análisis de eventos en tiempo real

Esta sección tiene como objetivo el diseño y desarrollo de una herramienta para el análisis de eventos sísmicos en tiempo real, usando el algoritmo Classic STA/LTA. El desarrollo de la herramienta se realizó en Python. Esta herramienta permite visualizar los eventos en tiempo real que llegan desde un servidor. El servidor también está desarrollado en Python y la librería *socketio*. Se encarga de enviar los registros sísmicos cada 2 segundos. La herramienta cliente recibe estos datos, grafica y aplica el algoritmo Classic STA/LTA en busca de eventos sísmicos. En caso de detectar un evento muestra un indicador (led rojo), la hora del evento y guarda en un archivo de texto la hora de inicio y fin del evento, para tener un registro de todos los eventos sísmicos detectados.

5.2.1. Arquitectura

El sistema desarrollado está basado en la arquitectura cliente servidor. En la Figura 5.2.1 se muestra una representación de la arquitectura del sistema. La herramienta para el análisis de eventos en tiempo real, está implementada sobre una Raspberry Pi, y viene a ser el cliente. Mientras que el servidor está en Ubuntu.

Al inicio lo que se hace es conectarse al servidor mediante *sockets*, usando la IP ([Internet Protocol \(IP\)](#)) del servidor y el puerto 3000. Un *socket* es un proceso existente en el cliente y servidor, que sirve para que el programa servidor y el cliente realicen correctamente la entrega de paquetes a través de un canal de comunicación TCP ([Transmission Control Protocol \(TCP\)](#)), que brinda seguridad en la transmisión de datos, seguridad en la recepción, en la integridad, entre otros. Luego se inicia la solicitud para obtener los eventos mediante el *emmit('event')*. El servidor responde en el *emmit('new-event')*, y envía los datos de registros sísmicos, la frecuencia de muestreo y la longitud de los datos enviados. A continuación, espera 2 segundos, y continúa enviando los datos. Si se desea desconectar del servidor y dejar de recibir los datos se recurre al *disconnect()*.

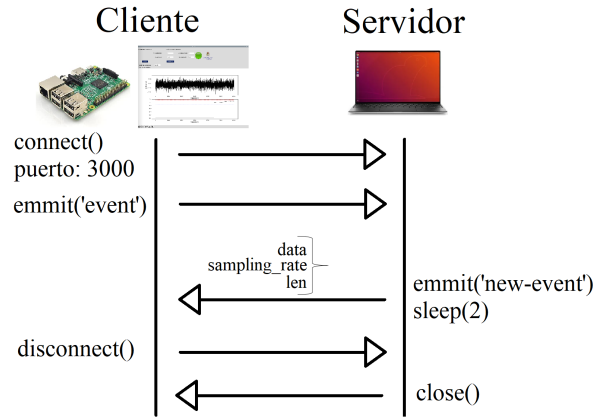


Figura 5.2.1: Arquitectura del sistema

5.2.2. Características y funcionalidades

En la Figura 5.2.2 se muestra la estructura de la herramienta en tiempo real mostrando las clases del sistema, sus atributos y métodos. La herramienta en primer lugar debe conectarse al servidor. Luego se deben ingresar los parámetros de entrada del algoritmo Classic STA/LTA y el factor de conversión. La interfaz tiene indicadores de eventos, rojo cuando detecta un evento y verde mientras está conectado al servidor. También se puede observar la frecuencia de muestreo de los datos y la hora del último evento detectado. La gráfica de los eventos se actualiza en tiempo real. Entre otras funciones que tiene es que permite desconectarse del servidor y guardar la imagen de los eventos obtenidos. Todas estas funciones se observan en la Figura 5.2.3. Este desarrollo también se liberó para su uso y experimentación y se encuentra disponible en [64].

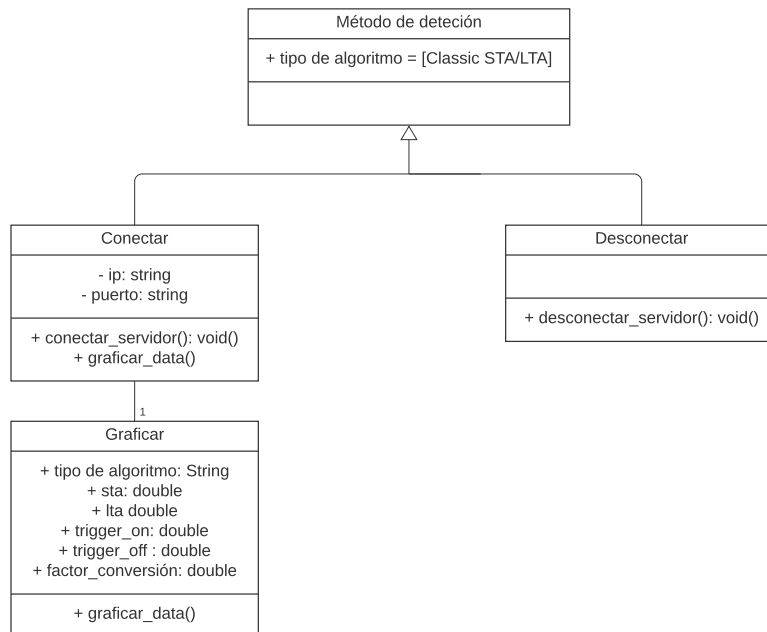


Figura 5.2.2: Diagrama de clases de la herramienta en tiempo real

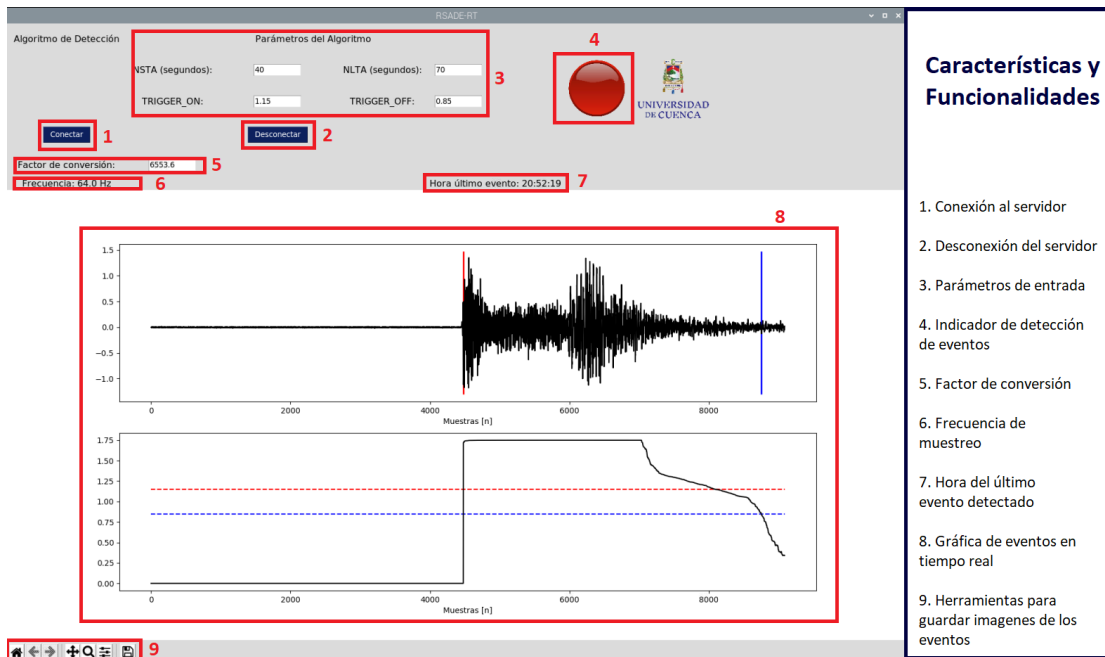


Figura 5.2.3: Características y funcionalidades de la herramienta en tiempo real

Los casos de uso se muestran en la Figura 5.2.4. Específicamente, la herramienta es el sistema y los actores el cliente y el servidor. Para conectarse al servidor interviene el cliente, luego que se da la conexión se empiezan a graficar los registros sísmicos. Para desconectarse del servidor únicamente interviene el cliente. Mientras que el servidor envía continuamente los registros al sistema.

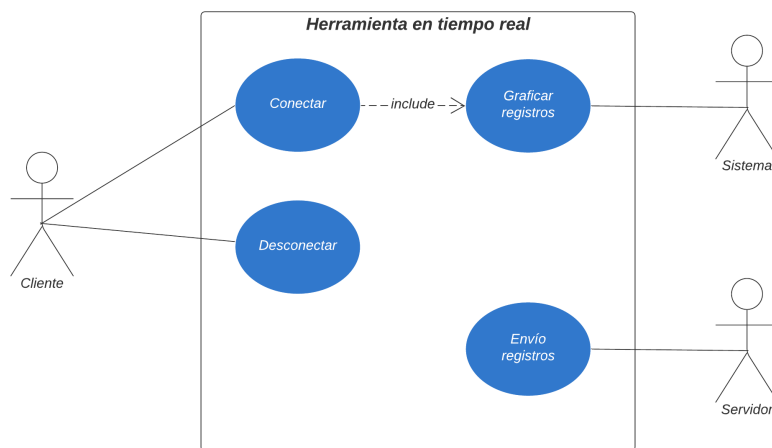


Figura 5.2.4: Casos de uso de la herramienta en tiempo real

La información detallada de esta herramienta se presenta en el Apéndice B

5.2.3. Visualización y presentación de resultados

Luego de que se ingresan todos los parámetros, se presiona el botón **Conectar** para conectarse al servidor. El resultado de la detección de un evento se muestra en la Figura 5.2.5. El indicador de color rojo hace referencia a la detección de un evento, también se muestra la hora del último evento detectado y

la frecuencia de muestreo de los datos. Además, se observa que en la parte inferior se tiene una barra de herramientas para interactuar con la gráfica, con la cual se puede hacer zoom o guardar la imagen. En el Apéndice B, se detalla un manual de usuario de la herramienta desarrollada.

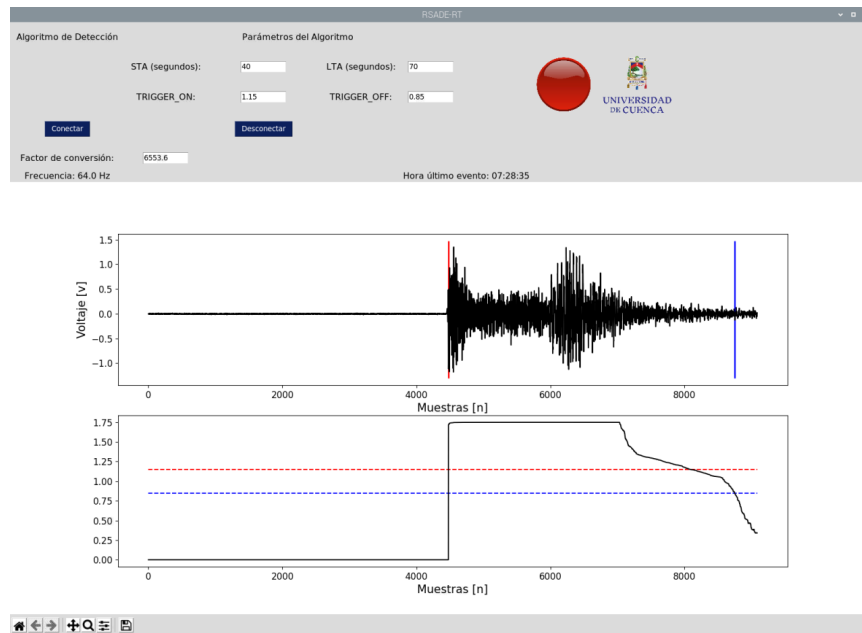


Figura 5.2.5: Gráfica de un evento sísmico detectado en tiempo real

5.3. Eficiencia computacional

Para concluir el capítulo detección de eventos en tiempo real y evaluar la eficiencia computacional, en la Figura 5.3.1 se muestra el uso de CPU del dispositivo SBC en la herramienta desarrollada para la detección de eventos en tiempo real. Como ya se mencionó, se implementó el algoritmo Classic STA/LTA con los mejores parámetros obtenidos en la Sección 5.1 y, finalmente, se capturó el uso de la CPU de la Raspberry Pi durante 30 minutos ya que se considera un tiempo adecuado para ver si se sobrecarga el dispositivo SBC. Como se puede apreciar en la gráfica, en ningún momento se presentaron sobrecargas en la CPU, ya que se mantiene entre 19 % y 21 %. Por consiguiente, el uso de la plataforma SBC es adecuado para la detección de eventos en tiempo real.

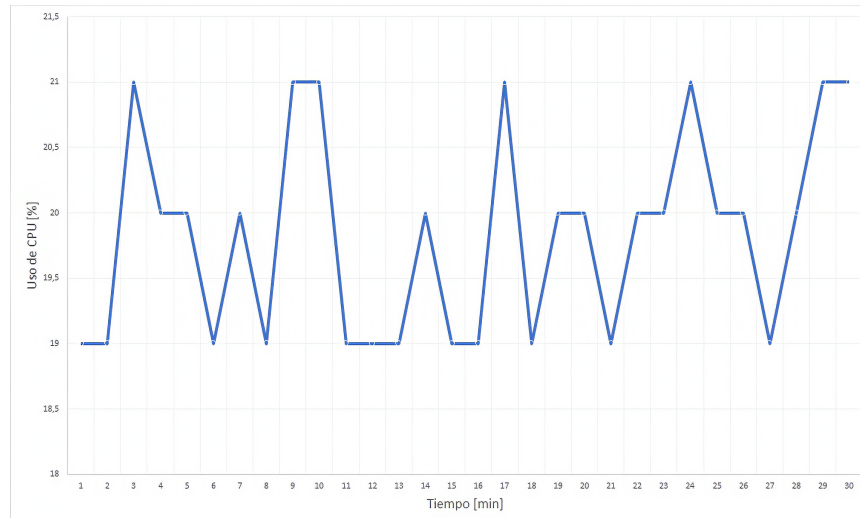


Figura 5.3.1: Uso de CPU en la herramienta en tiempo real

5.4. Conclusiones

En esta sección se presentó una herramienta de software para la detección de eventos sísmicos que permite conectarse a un servidor para obtener los datos en tiempo real y modificar los parámetros de entrada. Además, dispone de las funcionalidades de obtener gráficas de los resultados, indicador de eventos detectados, frecuencia de muestreo de los datos y exportación de todos los eventos detectados con la hora de inicio y fin del mismo. De igual forma, la decisión de liberar la herramienta tiene la finalidad de que siga creciendo y mejorando con aportes de personas interesadas en este ámbito y que pueda ser usada para desarrollar más estudios sobre la detección de eventos sísmicos.

En cuanto al uso de la plataforma SBC (Raspberry Pi 3) se encontró que, durante 30 minutos el uso de CPU se mantiene constante entre 19% y 21%. Por lo que el uso de la plataforma SBC es adecuado para la detección de eventos en tiempo real.

Conclusiones

6.1. Conclusiones

La detección automática de eventos sísmicos es una de las tareas fundamentales durante el procesamiento de las señales adquiridas por estaciones sísmicas. En tal contexto, la RSA monitorea poblaciones en el Sur del Ecuador. Para tal objetivo, disponen de múltiples estaciones sísmicas de geófonos y acelerógrafos ubicadas en distintos sitios de la región. Sin embargo, el procesamiento de esta información sigue siendo ambiguo y el personal a cargo debe revisarla de forma manual para obtener datos sobre eventos sísmicos. Este procesamiento manual conlleva un tiempo considerable de trabajo humano y es propenso a errores, por lo que, en la actualidad con las distintas herramientas disponibles de forma libre como ObsPy, es posible aumentar la eficiencia al momento de procesar datos sismológicos y obtener los eventos sísmicos.

La herramienta propuesta y probada para el análisis de eventos sísmicos puede adaptarse para realizar diferentes estudios o incluso la identificación de distintos eventos sismológicos. Dicha herramienta permite leer un registro sísmico, detectar con diversos algoritmos los eventos sísmicos de tal registro y la comparación de resultados de forma rápida y sencilla. Además, dispone de las funcionalidades de análisis en intervalos de tiempo, obtener gráficas de los resultados, exportación de los resultados y creación de nuevos archivos miniSeed con los datos de los intervalos de interés. Con respecto a los experimentos realizados para cada algoritmo, se obtuvo como resultado que el método Classic STA/LTA es el que generó mejor rendimiento, bajo las condiciones de los registros sísmicos de la Red Sísmica del Austro (nivel de ruido promedio, sensibilidad de los sensores sísmicos, distancia a la que se encuentran las estaciones sísmicas, magnitud). Con este algoritmo se detectaron el 98 % de eventos reales.

En cuanto a la herramienta de software para la detección de eventos sísmicos en tiempo real, permite conectarse a un servidor para obtener los datos y modificar los parámetros de entrada. Además, dispone de las funcionalidades de obtener gráficas de los resultados, indicador de eventos detectados, frecuencia de muestreo de los datos y exportación de todos los eventos detectados con la hora de inicio y fin del mismo.

En lo que se refiere al uso de la plataforma SBC (Raspberry Pi 3) se encontró que, en la herramienta para el análisis de eventos, los algoritmos que más costo computacional requieren son el Z Detector y AR-AIC. Mientras que el método que menor uso de CPU tiene es el de Baer and Kradolfer picker. Además, mientras los otros algoritmos toman en promedio 1 minuto para generar resultados, en el Z Detector se obtuvo como máximo 32 minutos de procesamiento para obtener resultados. Con respecto a la herramienta de *software* para la detección de eventos sísmicos en tiempo real, se detectó que, durante 30 minutos el uso de CPU se mantiene constante entre 19 % y 21 %. Por lo que el uso de la plataforma SBC es adecuado para la detección de eventos en tiempo real.

6.2. Recomendaciones

Apoyado en los resultados obtenidos dentro del alcance de la tesis se destaca que las herramientas desarrolladas tienen las funcionalidades necesarias para ser utilizadas en aplicaciones reales. Dichas herramientas en su estado actual son robustas y tienen libertad para su escalabilidad en cualquiera de sus componentes ya que están liberadas y desarrolladas en Python 3, lo que facilita la modularidad de las mismas. Por lo que se recomienda plantear modificaciones (v.g. filtros, formatos de hora, parámetros necesarios para cada algoritmo, factor de conversión, formato de datos de entrada) a los sistemas acorde a cada necesidad.



6.3. Trabajos futuros

Para mejorar el rendimiento de la herramienta de detección de eventos en tiempo real, se pueden implementar todos los métodos de detección de eventos disponibles en ObsPy, ya que actualmente solo se aplica el algoritmo Classic STA/LTA, que fue el que mejor rendimiento tuvo en los experimentos de comparación de algoritmos. Si se implementa esta mejora, se debe tener en cuenta que también se deben modificar los diferentes parámetros de entrada acorde a los algoritmos agregados. También se puede implementar programación en paralelo para realizar el análisis en tiempo real con varios métodos de detección al mismo tiempo. En tal caso se debe analizar si la Raspberry Pi 3 es capaz de soportar esta carga de CPU o en caso contrario buscar la implementación en una Raspberry Pi 4.

Se puede utilizar matrices de confusión para evaluar el desempeño y la precisión de cada método de detección de eventos en la comparación de algoritmos. Luego, con el análisis de las curvas ROC se puede tener herramientas para seleccionar los métodos posiblemente óptimos y descartar algoritmos subóptimos. Además, sería interesante aplicar técnicas de inteligencia artificial como Redes Neuronales o *Support Vector Machines* para construir modelos que predigan la llegada de eventos sísmicos, proporcionando un conjunto de ejemplos de entrenamiento reales como los utilizados para el análisis de la comparación de algoritmos.

Para evaluar el rendimiento de la herramienta en tiempo real, se pueden usar más parámetros aparte de la carga de CPU sobre el dispositivo SBC. Se puede utilizar el consumo energético ya que para el uso en campo del software sería interesante conocer dicha métrica teniendo en cuenta que estos dispositivos operan con baterías y tienen un tiempo de autonomía limitada. También se puede medir el uso de la RAM cuando la herramienta está en funcionamiento, ya que la Raspberry Pi también está limitada en cuanto a estos recursos. Además, como datos de entrada se plantea emplear registros sísmicos reales de varios institutos de sismología de forma similar a los utilizados en el presente trabajo.

Bibliografía

- [1] E. J. Tarbuck, F. K. Lutgens, D. Tasa, and D. Tasa, *Earth: an introduction to physical geology*. Pearson/Prentice Hall Upper Saddle River, 2005.
- [2] L. Küperkoch, T. Meier, and T. Diehl, “Automated event and phase identification,” in *New Manual of Seismological Observatory Practice 2 (NMSOP-2)*. Deutsches GeoForschungsZentrum GFZ, 2012, pp. 1–52.
- [3] C. Beauval, H. Yepes, W. H. Bakun, J. Egred, A. Alvarado, and J.-C. Singaicho, “Locations and magnitudes of historical earthquakes in the sierra of ecuador (1587–1996),” *Geophysical Journal International*, vol. 181, no. 3, pp. 1613–1633, 2010.
- [4] R. A. Lara-Cueva, D. S. Benítez, E. V. Carrera, M. Ruiz, and J. L. Rojo-Álvarez, “Automatic recognition of long period events from volcano tectonic earthquakes at cotopaxi volcano,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5247–5257, 2016.
- [5] A. Eguez, A. Alvarado, H. Yepes, M. N. Machette, C. Costa, R. L. Dart, and L. Bradley, “Database and map of quaternary faults and folds of ecuador and its offshore regions,” *US Geological Survey Open-File Report*, vol. 3, p. 289, 2003.
- [6] J. Jiménez, J. Cabrera, J. Sánchez, and F. Avilés, “Vulnerabilidad sísmica del patrimonio edificado del centro histórico de la ciudad de cuenca: Lineamientos generales y avances del proyecto,” *Maskana*, vol. 9, no. 1, pp. 59–78, 2018.
- [7] I. Sevilla Echeverría, “Improving automatic earthquake detection in the krafla seismic network,” Ph.D. dissertation.
- [8] D. Z. Otero, “Ondas sísmicas, su importancia para la geofísica y la humanidad,” *Universidad Industrial de Santander*, vol. 3, no. 4, 2018.
- [9] “Cuatro años después del Terremoto de Pedernales: Un testimonio sobre el peligro sísmico en el Ecuador - Instituto Geofísico - EPN.” [Online]. Available: <https://www.igepn.edu.ec/interactuamos-con-usted/1810-cuatro-anos-despues-del-terremoto-de-pedernales-un-testimonio-sobre-el-peligro-sismico-en-el-ecuador>
- [10] P. A. Toledo Peña, “Algoritmo de detección de ondas p invariante de escala: Caso de réplicas del sismo del 11 de marzo de 2010,” 2014.
- [11] M. Baer and U. Kradolfer, “An automatic phase picker for local and teleseismic events,” *Bulletin of the Seismological Society of America*, vol. 77, no. 4, pp. 1437–1445, 1987.
- [12] R. Sleeman and T. Van Eck, “Robust automatic p-phase picking: an on-line implementation in the analysis of broadband seismogram recordings,” *Physics of the earth and planetary interiors*, vol. 113, no. 1-4, pp. 265–275, 1999.
- [13] A. Trnkoczy, “Understanding and parameter setting of sta/lta trigger algorithm,” in *New Manual of Seismological Observatory Practice (NMSOP)*. Deutsches GeoForschungsZentrum GFZ, 2009, pp. 1–20.



-
- [14] D. A. Gínez Ordoñez, “Sistema para determinación de eventos sísmicos en una secuencia de tiempo,” B.S. thesis, PUCE-Quito, 2019.
- [15] M. Withers, R. Aster, C. Young, J. Beiriger, M. Harris, S. Moore, and J. Trujillo, “A comparison of select trigger algorithms for automated global seismic phase and event detection,” *Bulletin of the Seismological Society of America*, vol. 88, no. 1, pp. 95–106, 1998.
- [16] J. Berger and R. L. Sax, “Seismic detectors: the state-of-the-art,” SYSTEMS SCIENCE AND SOFTWARE LA JOLLA CA, Tech. Rep., 1981.
- [17] T. J. Cohen, “Vela network and automatic processing research.” ENSCO INC SPRINGFIELD VA SAR DIV, Tech. Rep., 1980.
- [18] R. Centeno Quico, “Análisis de métodos de identificación automática de llegadas de fases p y su aplicación a las señales sismo-volcánicas del misti (perú),” 2017.
- [19] F. Aldersons, “Toward three-dimensional crustal structure of the dead sea region from local earthquake tomography,” 2004.
- [20] E. G. Gutiérrez, “Introducción al filtrado digital,” *Catalunya: Escola Superior de Musica de Catalunya, Departamento de Sonología*, 2009.
- [21] J. I. Huircán, “Filtros activos, conceptos básicos y diseño,” *Departamento de Ingeniería Eléctrica, Universidad de La Frontera. Araucama, Chile*, 2012.
- [22] R. B. Raykova and S. Nikolova, “Review of digital formats for exchange and processing of seismological data.” *Bulgarian Geophysical Journal*, vol. 26, pp. 1–4, 2000.
- [23] B. Dost, J. Zednik, J. Havskov, R. Willemann, and P. Bormann, “Seismic data formats, archival and exchange,” in *New Manual of Seismological Observatory Practice (NMSOP)*. Deutsches GeoForschungsZentrum GFZ, 2009, pp. 1–20.
- [24] B. Sharma, A. Kumar, and V. Murthy, “Evaluation of seismic events detection algorithms,” *Journal of the Geological Society of India*, vol. 75, no. 3, pp. 533–538, 2010.
- [25] Y. Vaezi and M. Van der Baan, “Comparison of the sta/lta and power spectral density methods for microseismic event detection,” *Geophysical Supplements to the Monthly Notices of the Royal Astronomical Society*, vol. 203, no. 3, pp. 1896–1908, 2015.
- [26] H. Liu and J. Zhang, “Sta/lta algorithm analysis and improvement of microseismic signal automatic detection,” *Progress in Geophysics*, vol. 29, no. 4, pp. 1708–1714, 2014.
- [27] R. A. Lara-Cueva, A. S. Moreno, J. C. Larco, and D. S. Benítez, “Real-time seismic event detection using voice activity detection techniques,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5533–5542, 2016.
- [28] C. Baillard, W. C. Crawford, V. Ballu, C. Hibert, and A. Mangeney, “An automatic kurtosis-based p-and s-phase picker designed for local seismic networks,” *Bulletin of the Seismological Society of America*, vol. 104, no. 1, pp. 394–409, 2014.
- [29] H. Bui and M. van der Baan, “Event detection using a fast matched filter algorithm—an efficient way to deal with big microseismic data sets,” in *SEG Technical Program Expanded Abstracts 2020*. Society of Exploration Geophysicists, 2020, pp. 1325–1329.
- [30] S. G. Mosher and P. Audet, “Automatic detection and location of seismic events from time-delay projection mapping and neural network classification,” *Journal of Geophysical Research: Solid Earth*, vol. 125, no. 10, p. e2020JB019426, 2020.
-



- [31] L. Zhu, Z. Peng, J. McClellan, C. Li, D. Yao, Z. Li, and L. Fang, “Deep learning for seismic phase detection and picking in the aftershock zone of 2008 mw7. 9 wenchuan earthquake,” *Physics of the Earth and Planetary Interiors*, vol. 293, p. 106261, 2019.
- [32] O. Rojas, B. Otero, L. Alvarado, S. Mus, and R. T. Tous, “Artificial neural networks as emerging tools for earthquake detection,” *Computación y Sistemas*, vol. 23, no. 2, pp. 335–350, 2019.
- [33] S. Ghorbani, M. Barari, and M. Hoseini, “Presenting a new method to improve the detection of micro-seismic events,” *Environmental monitoring and assessment*, vol. 190, no. 8, pp. 1–13, 2018.
- [34] A. Reynen and P. Audet, “Supervised machine learning on a network scale: Application to seismic event classification and detection,” *Geophysical Journal International*, vol. 210, no. 3, pp. 1394–1409, 2017.
- [35] C. Chamberlain and J. Townend, “Detecting real earthquakes using artificial earthquakes: On the use of synthetic waveforms in matched-filter earthquake detection,” *Geophysical Research Letters*, vol. 45, no. 21, pp. 11–641, 2018.
- [36] J. Zhang, Y. Tang, and H. Li, “Sta/lta fractal dimension algorithm of detecting the p-wave arrivalsta/lta fractal dimension algorithm of detecting the p-wave arrival,” *Bulletin of the Seismological Society of America*, vol. 108, no. 1, pp. 230–237, 2018.
- [37] J. P. Jones and M. van der Baan, “Adaptive sta–lta with outlier statistics,” *Bulletin of the Seismological Society of America*, vol. 105, no. 3, pp. 1606–1618, 2015.
- [38] Y. Choubik, A. Mahmoudi, M. M. Himmi, and L. El Moudnib, “Sta/lta trigger algorithm implementation on a seismological dataset using hadoop mapreduce,” *IAES International Journal of Artificial Intelligence*, vol. 9, no. 2, p. 269, 2020.
- [39] J. Akram, D. Peter, and D. Eaton, “A k-mean characteristic function to improve sta/lta detection,” *Proceedings of the Geoconvention, Calgary, AB, Canada*, pp. 7–11, 2018.
- [40] A. Bueno, L. Zuccarello, A. Díaz-Moreno, J. Woollam, M. Titos, C. Benítez, I. Álvarez, J. Prudencio, and S. De Angelis, “Picoss: Python interface for the classification of seismic signals,” *Computers & Geosciences*, vol. 142, p. 104531, 2020.
- [41] A. Elkady and D. G. Lignos, “Earl—software for earthquake risk, loss and lifecycle analysis,” *SoftwareX*, vol. 12, p. 100607, 2020.
- [42] R. Baraschino, G. Baltzopoulos, and I. Iervolino, “R2r-eu: Software for fragility fitting and evaluation of estimation uncertainty in seismic risk analysis,” *Soil Dynamics and Earthquake Engineering*, vol. 132, p. 106093, 2020.
- [43] J. E. Romero, M. Titos, Á. Bueno, I. Álvarez, L. García, Á. de la Torre, and M. C. Benítez, “Apasvo: a free software tool for automatic p-phase picking and event detection in seismic traces,” *Computers & Geosciences*, vol. 90, pp. 213–220, 2016.
- [44] R. Barsch, “Web-based technology for storage and processing of multi-component data in seismology,” Ph.D. dissertation, lmu, 2009.
- [45] “The N-dimensional array (ndarray) — NumPy v1.19 Manual.” [Online]. Available: <https://numpy.org/doc/stable/reference/arrays.ndarray.html>
- [46] “SciPy.org — SciPy.org.” [Online]. Available: <https://scipy.org/>
- [47] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [48] “Enthought Tool Suite :: Enthought, Inc.” [Online]. Available: <http://code.enthought.com/pages/mayavi-project.html>



- [49] “MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink.” [Online]. Available: <https://www.mathworks.com/>
- [50] M. Beyreuther, R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann, “Obspy: A python toolbox for seismology,” *Seismological Research Letters*, vol. 81, no. 3, pp. 530–533, 2010.
- [51] Z. Wang and B. Zhao, “Automatic event detection and picking of p, s seismic phases for earthquake early warning and application for the 2008 wenchuan earthquake,” *Soil Dynamics and Earthquake Engineering*, vol. 97, pp. 172–181, 2017.
- [52] “12.2.3. obspy.signal.trigger.classic_sta_lta — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.classic_sta_lta.html#obspy.signal.trigger.classic_sta_lta
- [53] A. Lomax, C. Satriano, and M. Vassallo, “Automatic picker developments and optimization: Filter-picker—a robust, broadband picker for real-time seismic monitoring and earthquake early warning,” *Seismological Research Letters*, vol. 83, no. 3, pp. 531–540, 2012.
- [54] F. Grigoli, S. Cesca, M. Vassallo, and T. Dahm, “Automated seismic event location by travel-time stacking: An application to mining induced seismicity,” *Seismological Research Letters*, vol. 84, no. 4, pp. 666–677, 2013.
- [55] R. V. Allen, “Automatic earthquake recognition and timing from single traces,” *Bulletin of the seismological society of America*, vol. 68, no. 5, pp. 1521–1532, 1978.
- [56] F. Grigoli, S. Cesca, O. Amoroso, A. Emolo, A. Zollo, and T. Dahm, “Automated seismic event location by waveform coherence analysis,” *Geophysical Journal International*, vol. 196, no. 3, pp. 1742–1753, 2014.
- [57] “12.2.1. obspy.signal.trigger.recursive_sta_lta — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.recursive_sta_lta.html#obspy.signal.trigger.recursive_sta_lta
- [58] “12.2.4. obspy.signal.trigger.delayed_sta_lta — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.delayed_sta_lta.html#obspy.signal.trigger.delayed_sta_lta
- [59] “12.2.5. obspy.signal.trigger.z_detect — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.z_detect.html#obspy.signal.trigger.z_detect
- [60] “12.2.6. obspy.signal.trigger.pk_baer — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.pk_baer.html#obspy.signal.trigger.pk_baer
- [61] “12.2.7. obspy.signal.trigger.ar_pick — ObsPy Documentation (1.2.0).” [Online]. Available: https://docs.obspy.org/packages/autogen/obspy.signal.trigger.ar_pick.html#obspy.signal.trigger.ar_pick
- [62] A. Armijos, “alvaroarmijos/RSADE,” original-date: 2020-12-25T16:20:16Z. [Online]. Available: <https://github.com/alvaroarmijos/RSADE>
- [63] 12. trigger/picker tutorial — ObsPy documentation (1.2.0). [Online]. Available: https://docs.obspy.org/tutorial/code_snippets/trigger_tutorial.html
- [64] A. Armijos, “alvaroarmijos/RSADE-RT,” July 2021, original-date: 2021-05-24T14:20:44Z. [Online]. Available: <https://github.com/alvaroarmijos/RSADE-RT>

Apéndice A

Manual de usuario de la herramienta para la comparación de algoritmos

A.1. Introducción

El presente manual pretende ser una guía para utilizar de manera correcta la herramienta desarrollada para la detección de eventos sísmicos. Para el desarrollo del software se utilizó Python 3. Además cada algoritmo necesita diferentes parámetros que son obligatorios para poder obtener resultados y se detallan a continuación.

A.2. Primeros pasos

A.2.1. Prerrequisitos

Se necesitan las siguientes librerías para el funcionamiento de la herramienta:

- ObsPy
- tkinter
- matplotlib
- numpy

A.2.2. Descarga de la herramienta

La herramienta se encuentra en el repositorio de GitHub [62] y es de código abierto. Para descargarla se puede usar git:

```
git clone https://github.com/alvaroarmijos/RSADE.git
```

A.2.3. Compilación

Para ejecutar la herramienta desde la terminal se usa el siguiente comando:

```
python RSADE.py
```

A.2.4. Recomendación

La herramienta inicialmente toma el tamaño completo de pantalla, por lo que si la pantalla que se tiene conectada a la Raspberry no esta ajustada, no se pueden observar bien todos los elementos de la herramienta, por lo que se recomienda ajustar el tamaño de la siguiente manera:

Preferencias/Configuración de Raspberry Pi/Display/Set Resolution/ Aquí elegir la resolución adecuada a su pantalla

A.3. Métodos disponibles

Los métodos disponibles en la herramienta son los siguientes:

- Classic STA/LTA
- Recursive STA/LTA
- Delayed STA/LTA
- Z detector
- Baer and Kradolfer picker.
- AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

Inicialmente, la herramienta se muestra en blanco, hasta seleccionar un algoritmo, como se observa en la Figura A.3.1.

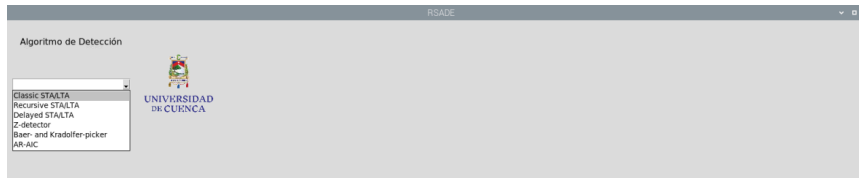


Figura A.3.1: Herramienta al momento de ejecutar

A.3.1. Classic STA/LTA, Recursive STA/LTA, Delayed STA/LTA y Z Detector

Los parámetros que se pide ingresar para estos métodos se muestran en la Figura A.3.2. Para estos algoritmos se puede graficar los eventos, obtener los eventos en un archivo de texto y generar el archivo miniSeed en el intervalo de horas especificado. Los parámetros se detallan en la Sección 4.1.2.1



Figura A.3.2: Interfaz gráfica para métodos STA/LTA

A.3.2. Baer and Kradolfer picker

Los parámetros para el método Baer and Kradolfer picker que se deben ingresar en la herramienta, se muestran en la Figura A.3.3. El detalle de cada parámetro se especifica en la Sección 4.1.2.5.



Figura A.3.3: Interfaz gráfica para el método Baer and Kradolfer picker

A.3.3. AR-AIC (Autoregressive-Akaike-Information-Criterion-picker)

Los parámetros necesarios para este método se observan en la Figura A.3.4. La especificación de cada parámetro se detalla en la Sección 4.1.2.6.




Figura A.3.4: Interfaz gráfica para el método AR-AIC

A.4. Visualización y presentación de resultados

Antes de proceder a obtener resultados, debemos ingresar los siguientes parámetros obligatorios:

- Seleccionar un algoritmo
- Parámetros de cada algoritmo
- Archivo de eventos
- Factor de conversión
- Elegir un canal

El intervalo de hora para obtener los eventos es opcional, pero en caso de ingresarlo se debe hacer en el formato **HH:MM**.

Después de elegir un algoritmo, tenemos una interfaz como se observa en la Figura A.4.1. Si en ese momento intentamos graficar los eventos, sale un mensaje que dice **Debe seleccionar un archivo antes de graficar**. Esto es para indicar al usuario que primero debe seleccionar un archivo de eventos.



Figura A.4.1: Interfaz sin parámetros

Para seleccionar un archivo de eventos, se debe dar clic en el botón **Seleccionar Archivo**. Se abre una ventana como se observa en la Figura A.4.2, donde se puede elegir el archivo.

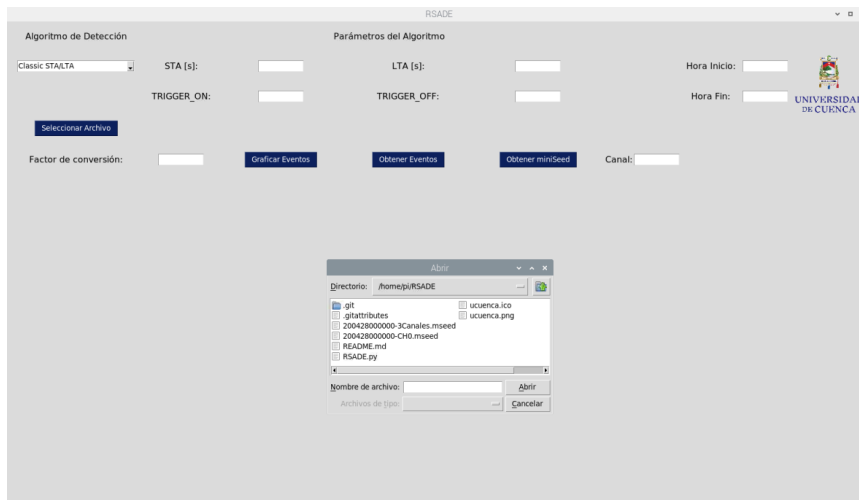


Figura A.4.2: Seleccionar archivo

Si ahora se intenta graficar, de igual forma sale otro mensaje de advertencia. Esto se observa en la Figura A.4.3

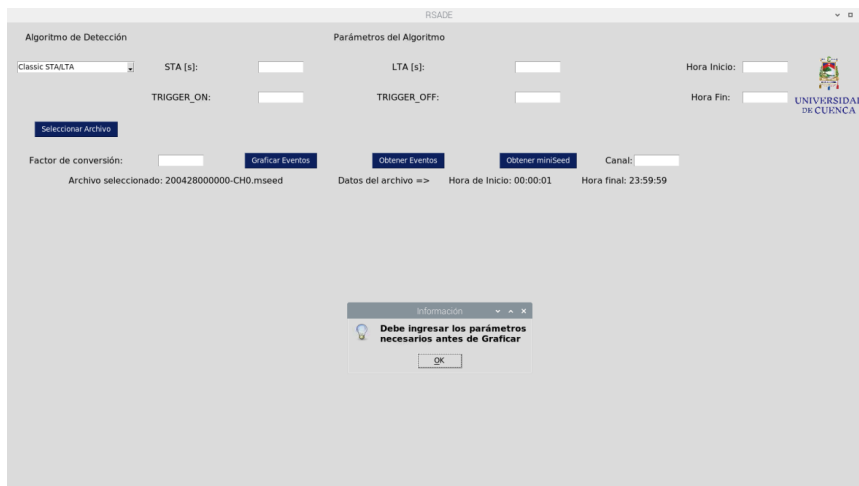


Figura A.4.3: Mensaje de advertencia cuando se tiene parámetros vacíos

Luego de que se ingresan todos los parámetros, se presiona el botón **Graficar evento**. El resultado se muestra en la Figura A.4.4. Aquí se observa que en la parte inferior se tiene una barra de herramientas para interactuar con la gráfica. Desde aquí se puede hacer zoom o guardar la imagen (como se muestra en la Figura A.4.5)

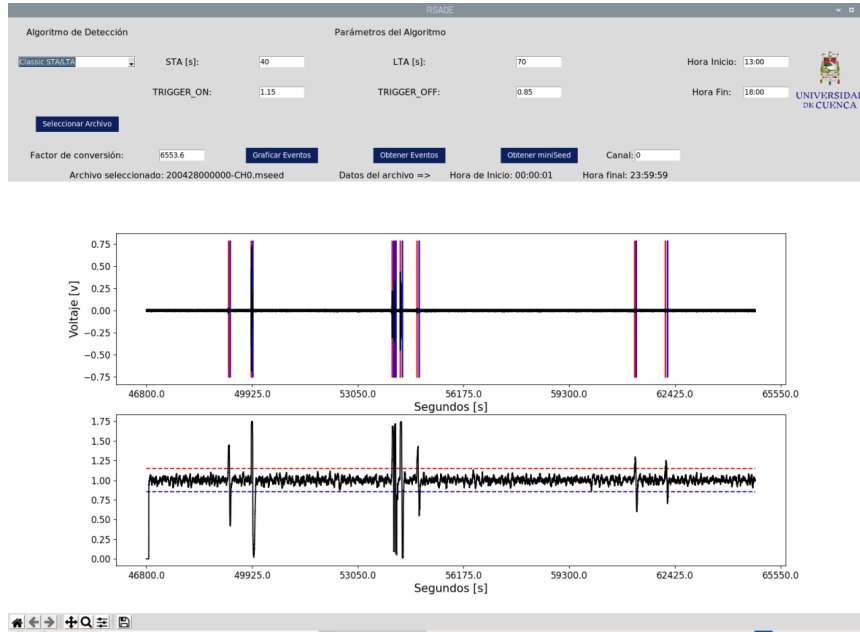


Figura A.4.4: Gráfica de eventos

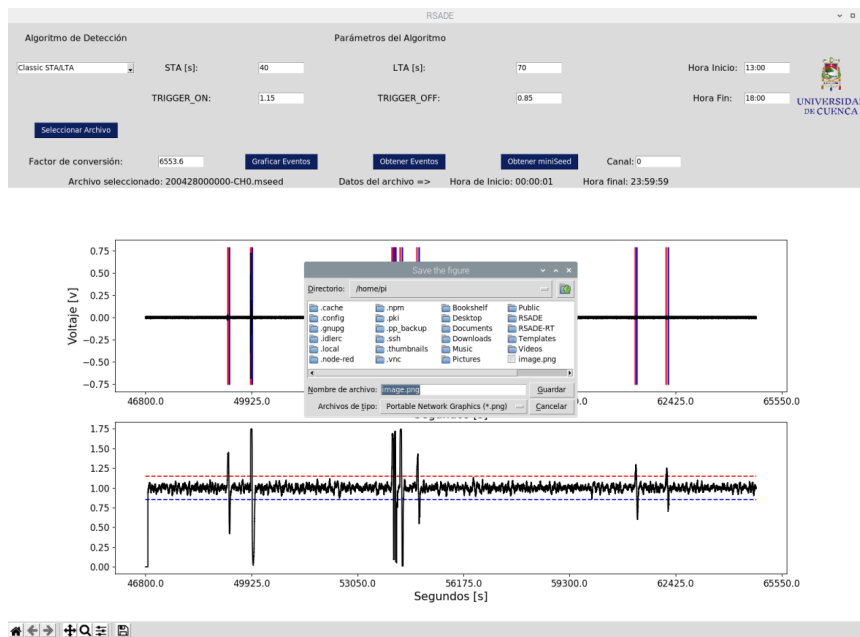


Figura A.4.5: Guardar gráfica de eventos

Otra opción que permite la herramienta, es exportar los eventos obtenidos. Se presiona en el botón **Obtener eventos**. Esto abre una ventana donde se puede elegir la ubicación donde se va a guardar el archivo y el nombre y formato del mismo (como se presenta en la Figura A.4.6).

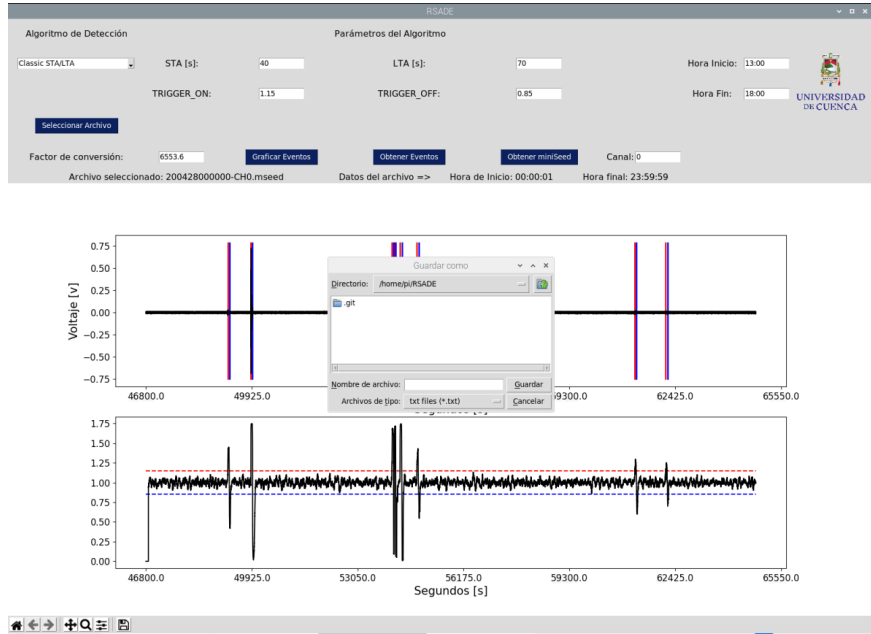


Figura A.4.6: Exportar eventos

Una vez que se guardan los eventos, se presenta un mensaje que nos pregunta si deseamos generar archivos miniSeed de los eventos obtenidos, como se muestra en la Figura A.4.7. Esta opción lo que hace es generar un archivo miniSeed por cada evento obtenido. Esto facilita luego el análisis individual de cada evento.

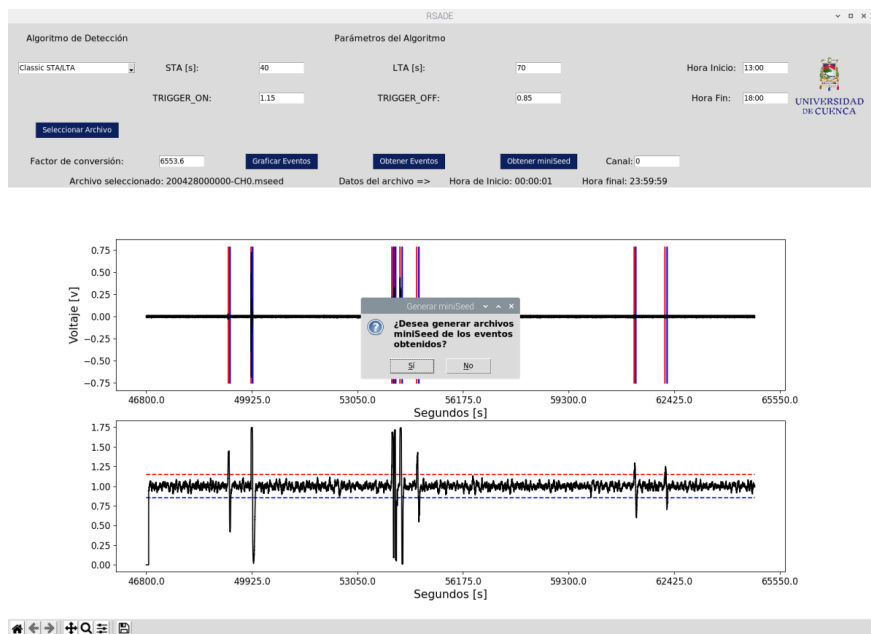


Figura A.4.7: Opciones para generar archivos miniSeed

Si confirmamos esta opción, procede a generar los archivos, al finalizar nos presenta un mensaje que nos confirma que los archivos se guardaron correctamente, esto se observa en la Figura A.4.8

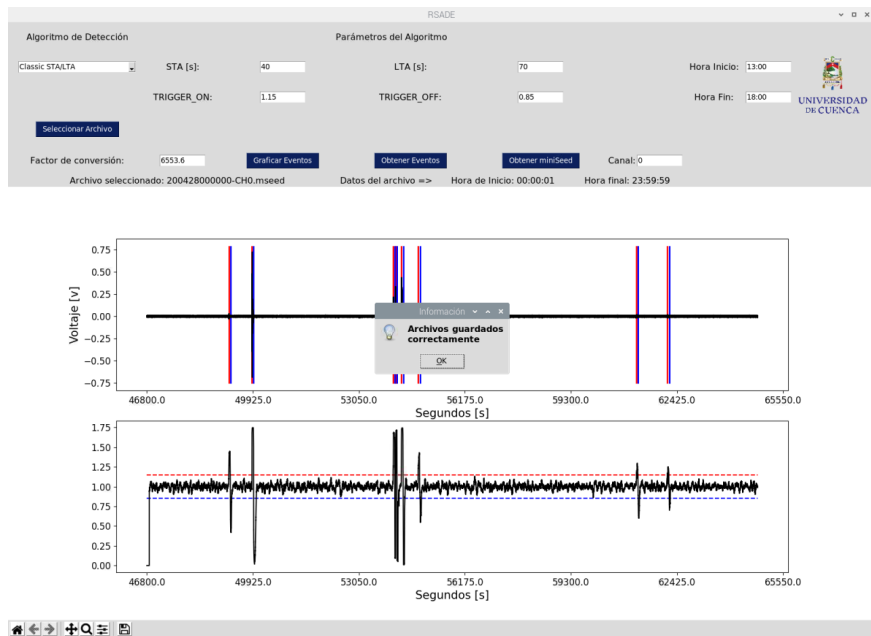


Figura A.4.8: Mensaje de confirmación para generar archivos miniSeed

La otra opción, es extraer un archivo miniSeed con los datos del tiempo graficado. Cuando se hace clic en el botón **Obtener miniSeed** se presenta el mensaje que se observa en la Figura A.4.9

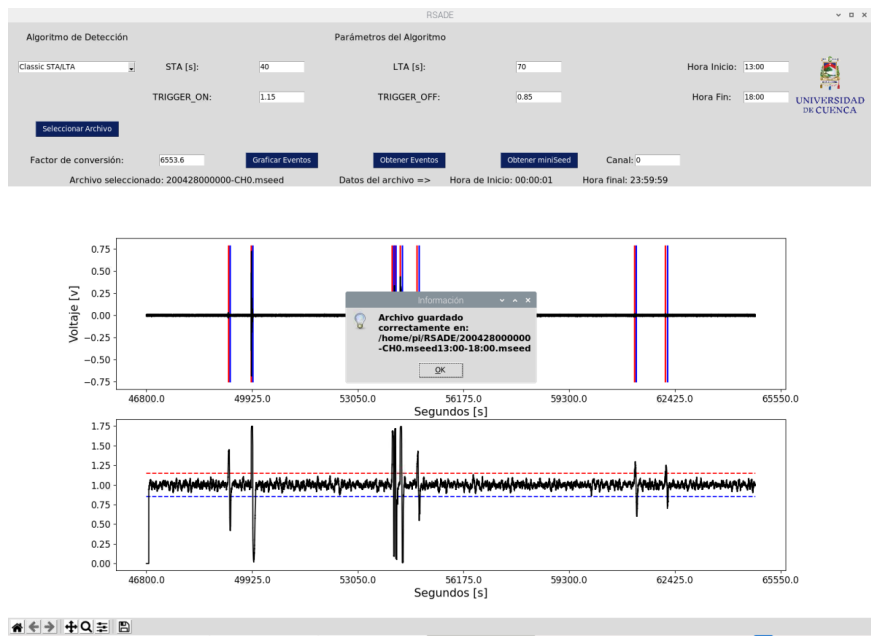


Figura A.4.9: Mensaje de confirmación luego de extraer un archivo miniSeed

Para los otros métodos se tienen las mismas funciones, excepto para el método AR-AIC ya que aquí no es posible exportar los eventos obtenidos, esto se observa en la Figura A.4.10. Para el método Baer- and Kradolf-picker, se presenta el resultado en la Figura A.4.11.

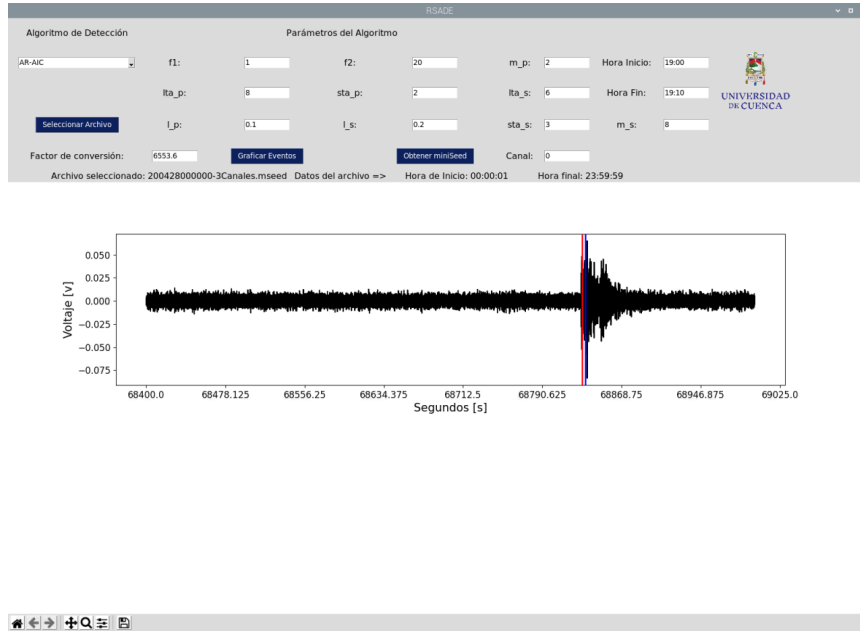


Figura A.4.10: Resultado del algoritmo AR-AIC

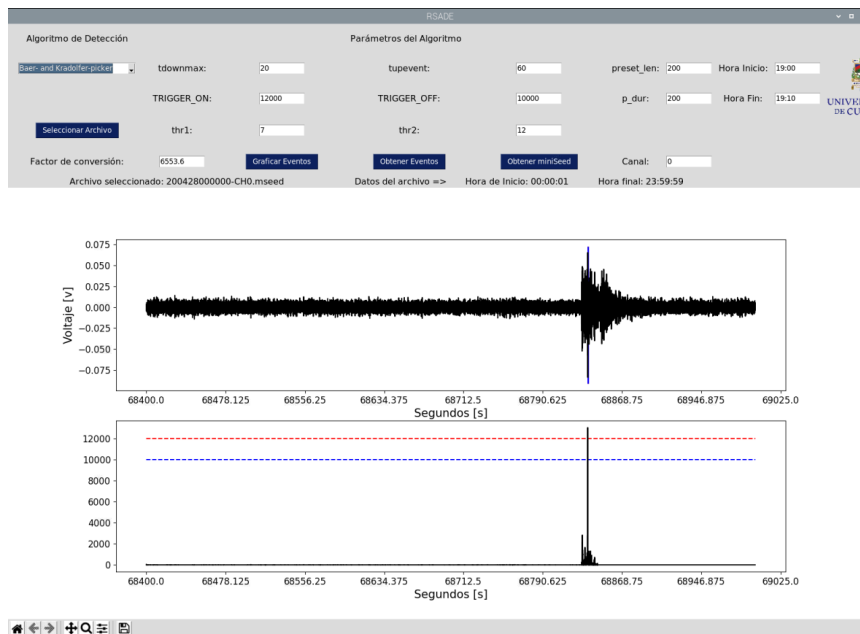


Figura A.4.11: Resultado del algoritmo Baer and Kradolfer picker

Apéndice B

Manual de usuario de la herramienta en tiempo real

B.1. Introducción

El presente manual pretende ayudar a utilizar de manera correcta la herramienta desarrollada para la detección de eventos sísmicos en tiempo real. Para el desarrollo del software se utilizó Python 3 y el algoritmo Classic STA/LTA. Dicho algoritmo requiere diferentes parámetros que son obligatorios para poder obtener resultados y se detallan a continuación.

B.2. Primeros pasos

B.2.1. Prerrequisitos

Se necesitan las siguientes librerías para el funcionamiento de la herramienta:

- ObsPy
- tkinter
- matplotlib
- numpy
- socketio

B.2.2. Descarga de la herramienta

La herramienta se encuentra en el repositorio de GitHub [64] y es de código abierto. Para descargarla se puede usar git:

```
https://github.com/alvaroarmijos/RSADE-RT.git
```

B.2.3. Compilación

Inicialmente se debe ejecutar el servidor. Para esto usamos el siguiente comando:

```
python3 server.py
```

El servidor lo que hace es leer un archivo en formato *miniSeed* y envía los datos cada 2 segundos al cliente que se conecta mediante sockets.

Para ejecutar la herramienta de detección en tiempo real desde la terminal se usa el siguiente comando:

```
python3 classic_stalta.py
```

B.2.4. Recomendación

La herramienta inicialmente toma el tamaño completo de pantalla, por lo que si la pantalla que se tiene conectada a la Raspberry no esta ajustada, no se puede observar bien todos los elementos de la herramienta, por lo que se recomienda ajustar el tamaño de la siguiente manera:

Preferencias/Configuracion de Raspberry Pi/Display/Set Resolution/ Aquí elegir la resolución adecuada a su pantalla

B.3. Classic STA/LTA

Los parámetros que se solicita ingresar para este método se muestran en la Figura B.3.1. Inicialmente se cargan valores predeterminados que fueron los mejores resultados obtenidos con este algoritmo en la Sección 5.1. Cabe recalcar que el valor de NSTA y NLTA deben ingresarse en segundos. En cuanto al factor de conversión, si no se desea se le puede dejar en el valor de 1 o en caso contrario agregar el valor indicado para el mismo.

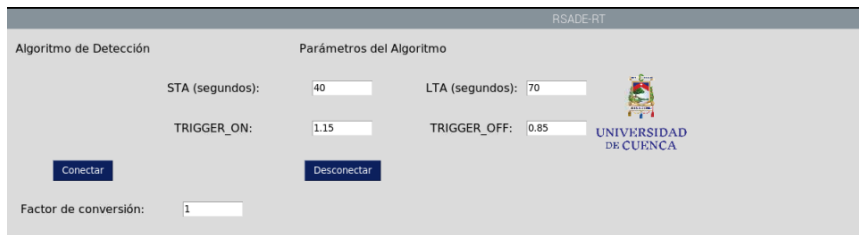


Figura B.3.1: Interfaz gráfica para el método Classic STA/LTA

B.4. Visualización y presentación de resultados

Después de introducir todos los parámetros necesarios, se tiene una interfaz como se observa en la Figura B.4.1. Para conectarse al servidor y empezar a recibir los eventos en tiempo real, se debe presionar el botón **Conectar**.

Si la conexión al servidor se realiza correctamente, en la interfaz se muestra un indicador de color verde (Figura B.4.2). Inicialmente la herramienta no grafica los datos recibimos instantáneamente, sino que espera hasta acumular en el *buffer* una cantidad de datos mayor o igual al valor ingresado de NLTA en número de muestras. Esto se debe a que para realizar el análisis el algoritmo Classic STA/LTA necesita que se cumpla esa condición para obtener resultados.

Cuando se tiene la suficiente cantidad de datos, la herramienta empieza a graficar los registros sísmicos como se muestra en la Figura B.4.3. De igual forma, ahora empieza a acumular datos hasta un valor del doble de NLTA en muestras. Una vez que completa este valor, mantiene siempre ese numero de muestras y se va actualizando con los nuevos datos que van llegando en tiempo real. En la Figura B.4.4 se puede observar que ya se completó el *buffer*.

Cuando la herramienta detecta un evento sísmico, muestra un indicador de color rojo (Figura B.4.5). De igual forma se muestra la hora del último evento detectado y la frecuencia de muestreo de la señal recibida.

Luego de que el evento pasa, el indicador se vuelve de color verde, como se observa en la Figura B.4.6 pero se mantiene la hora del último evento detectado.

Otra función que se tiene disponible es la posibilidad de guardar las gráficas obtenidas con la herramienta. En la Figura B.4.7 se muestra el proceso para guardar un evento de interés. Primero se presiona sobre el ícono de guardar en la barra de herramientas de la parte inferior, luego se selecciona la ubicación y el nombre del archivo.

Para desconectarse del servidor y dejar de recibir los eventos sísmicos, se debe presionar el botón **Desconectar**. Luego de esto se muestra un indicador de color rojo que nos indica que no se tiene conexión al servidor, como se observa en la Figura B.4.8

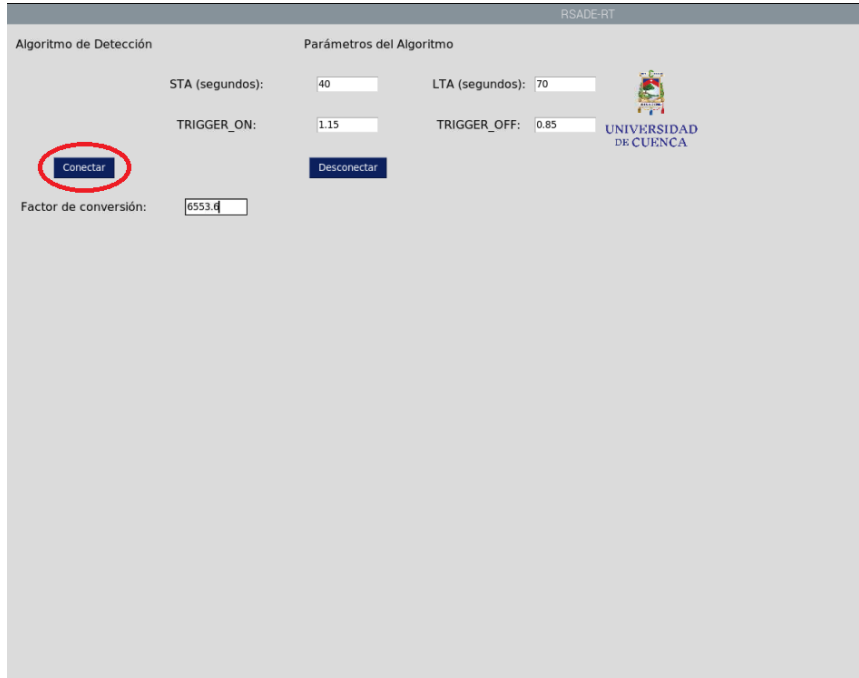


Figura B.4.1: Interfaz sin parámetros

Finalmente, durante el análisis la herramienta va guardando todos los eventos detectados (hora de inicio y hora de fin del evento sísmico) en el archivo "eventos.txt". Si al final del análisis se desea revisar todos los eventos solo se abre este archivo y se muestra la información de todos los eventos sísmicos detectados.

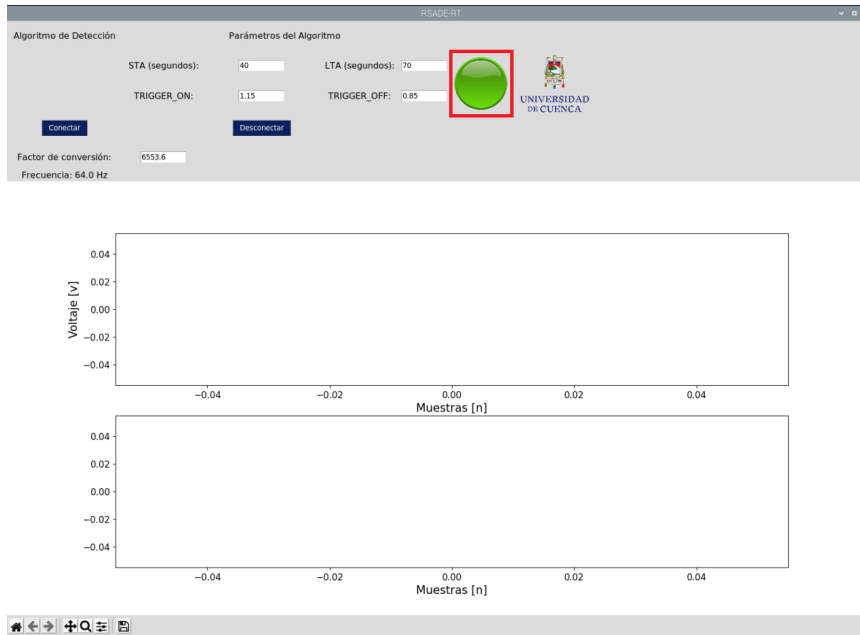


Figura B.4.2: Conexión al servidor

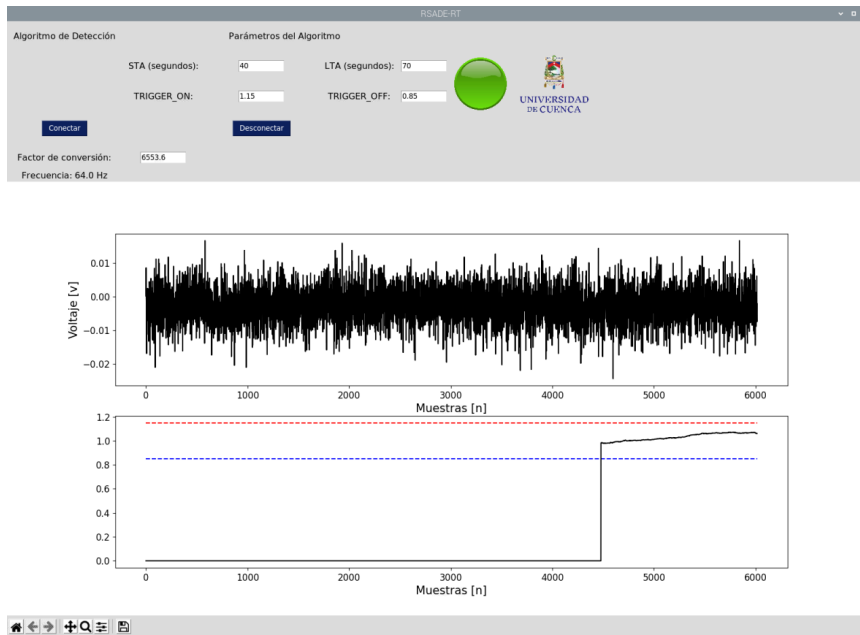


Figura B.4.3: Llegada de datos sísmicos

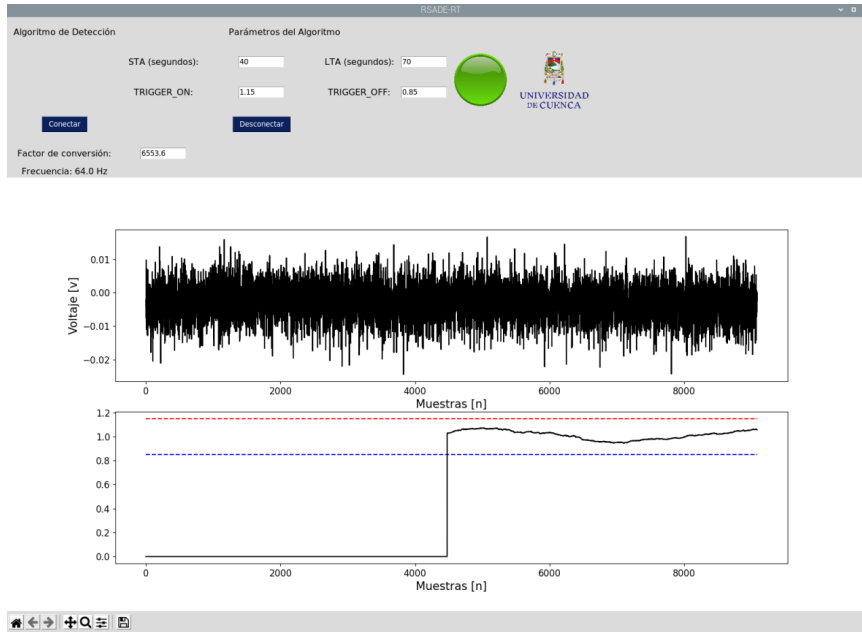


Figura B.4.4: Llegada datos sísmicos y buffer completo

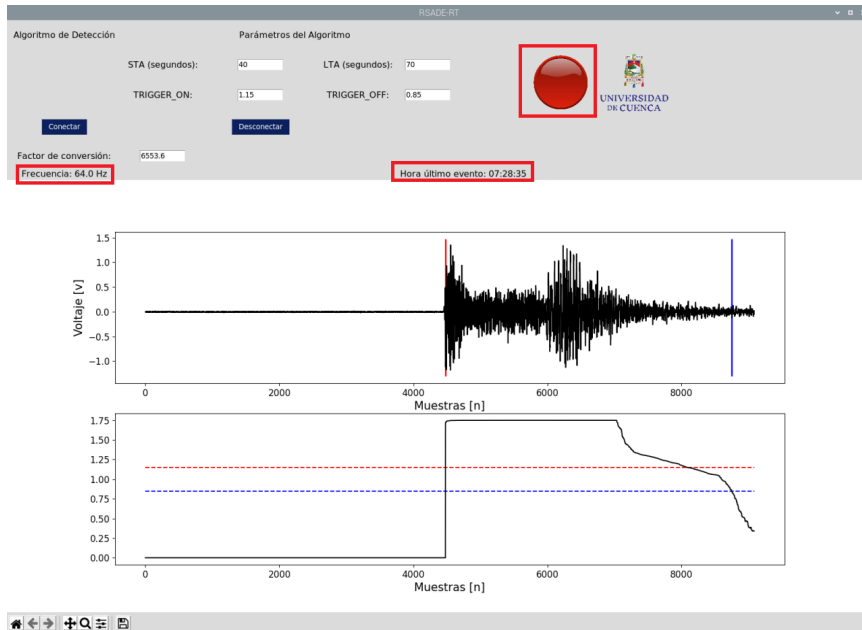


Figura B.4.5: Detección de eventos sísmicos

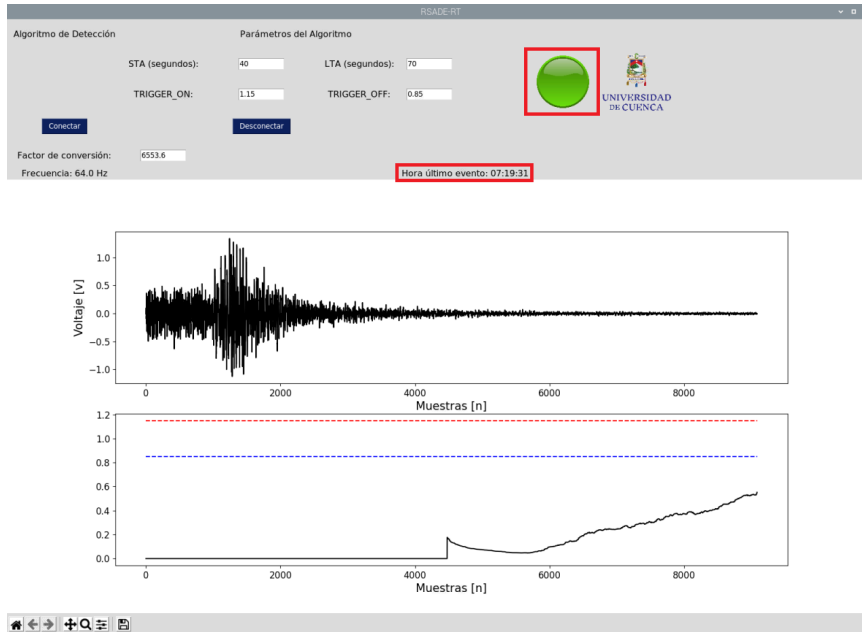


Figura B.4.6: Registros sísmicos luego de un evento

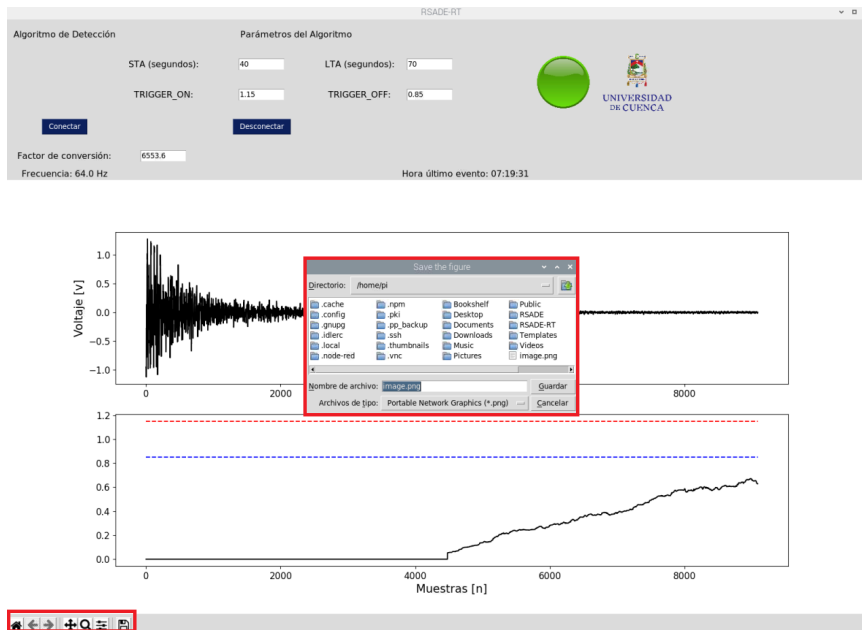


Figura B.4.7: Guardar gráficas

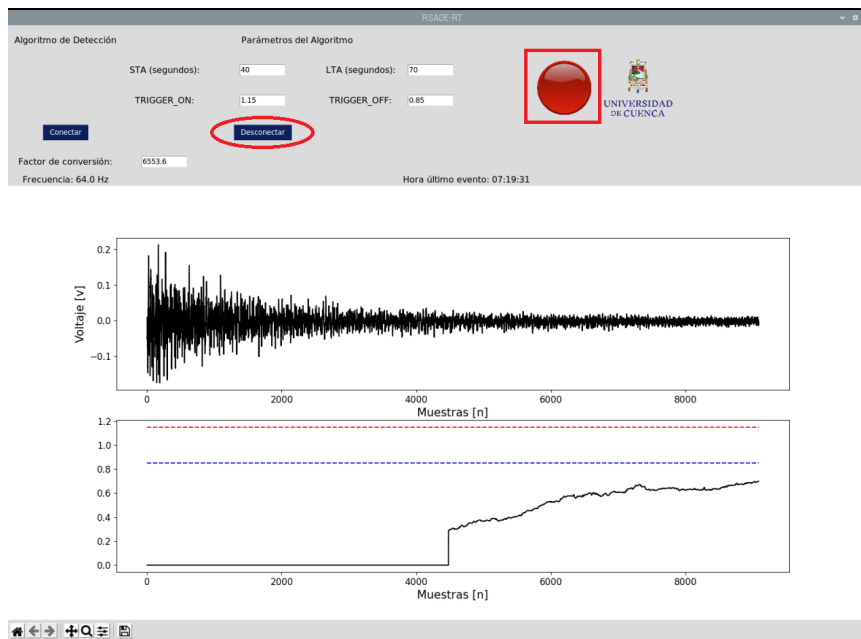


Figura B.4.8: Desconectarse del servidor