PROTOTIPO DE CIFRADOR DE FLUJO EN LÍNEA EN TIEMPO REAL MEDIANTE SALSA20 EN PLATAFORMAS ARDUINO UNO PARA TRANSMISIÓN DE AUDIO

Universidad
Tecnológica
de Pereira

DAVID MARTINEZ VALDES

BRYAN ESTEFAN MORENO DIAZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
PEREIRA
2016

PROTOTIPO DE CIFRADOR DE FLUJO EN LÍNEA EN TIEMPO REAL MEDIANTE SALSA20 EN PLATAFORMAS ARDUINO UNO PARA TRANSMISIÓN DE AUDIO

DAVID MARTINEZ VALDES

BRYAN ESTEFAN MORENO DIAZ

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACION

Director
ANA MARIA DE LAS MERCEDES LOPEZ ECHEVERRY

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERIA EN SISTEMAS Y COMPUTACION
PEREIRA
2016

1 Tabla de contenido

Tabla de ilustraciones

ÍNDICE DE TABLAS

INTRODUCCIÓN

El manejo de información ha sido siempre una necesidad a través de la historia de la humanidad, pero solo fue recientemente que la cantidad de información producida por los humanos rebasó límites que cualquiera hubiera imaginado, esto se debió al vertiginoso avance de los sistemas de información y plataformas computacionales, lo que dificultó el manejo de dicha información y obligó a los científicos e intelectuales de diversos campos a buscar métodos para optimizar y mejorar las herramientas existentes.

La electrónica fue la que propició tal desarrollo al incrementar las capacidades de procesamiento, almacenamiento y transmisión de información, y diariamente se producen incrementos de estas capacidades Este crecimiento en las características de los equipos computacionales ayudaron al desarrollo de incontables campos del conocimiento, pero también se hicieron evidentes muchos inconvenientes de la era de la información, siendo la seguridad uno de los más apremiantes. Para dar solución a estos problemas la optimización y adaptación de algoritmos a nuevos sistemas computacionales se ha convertido en una de las opciones más viables y que más partido saca de los constantes avances tecnológicos ya que mediante pequeños cambios conceptuales y lógicos, se aumenta la velocidad por varios factores, lo que representa una ventaja ya que no son necesarias modificaciones en el hardware, reduciendo costos y aumentando la seguridad.

Es por eso que en el ejercicio profesional como Ingenieros se debe propiciar el desarrollo e investigación en el uso de las capacidades que estos sistemas nos brindan para proveer soluciones oportunas y eficientes a los problemas de la sociedad.

1. GENERALIDADES

1.1 DEFINICIÓN DEL PROBLEMA

Existe una escasez de métodos de encriptación portátiles que sean de fácil implementación y que se ajusten a diferentes entornos como el empresarial, académico o doméstico. Se hace necesario encontrar una solución que permita transmitir la información y que solo sea legible para su destinatario, y pueda proveerse un alto porcentaje de confianza al usuario. Inicialmente esta solución debería estar enmarcada para un entorno académico, hasta que pruebas suficientes hayan sido realizadas y pueda considerarse un desarrollo comercial y profundo en la industria. No se quiere abarcar en este punto toda la información susceptible de ser transmitida, sino solo audio y la transmisión de esta inicialmente se realizará por medio de cables y conexiones seriales.

1.2 JUSTIFICACIÓN

Las plataformas de hardware libre que permitan cifrar y enviar una comunicación de forma transparente al usuario no son comunes en el mercado y pueden convertirse en una opción viable para proveer seguridad. En el ámbito de plataformas privativas, el dispositivo más prominente lo produce la compañía Motorola y permite cifrado y transmisión de información inalámbrico denominado "CRYPTR 2"[1], que usa encriptación AES[2] y ofrece un rendimiento de 20mb con algunas características de QoS[3]. La mayor desventaja de estos sistemas es su alto costo e inflexibilidad para posibles personalizaciones o situaciones particulares que siempre se han de presentar en diferentes contextos.

[1] "CRYPTR 2 Broadband IP Encryption Unit - Motorola Solutions." 11 Mar. 2015 <http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Project+25+(P25)+Systems/Encryption/CryptR>

[2] Daemen, J. "The Design of Rijndael - Joan Daemen." 2001. <http://jda.noekeon.org/JDA_VRI_Rijndael_2002.pdf>

[3] "What is QoS (Quality of Service)? - Definition from WhatIs.com." 2010. 24 Apr. 2015 <http://searchunifiedcommunications.techtarget.com/definition/QoS-Quality-of-Service>

La privacidad de las comunicaciones es una preocupación latente en esta sociedad de la información, donde las personas están interconectados a través del internet por medio de infinidad de dispositivos y cualquier persona con un mínimo conocimiento técnico y una cantidad moderada de recursos informáticos puede capturar los datos que circulan por los medios de transmisión físicos y aéreos usados por los proveedores de servicio de internet y demás redes ajenas a nuestro control. Este trabajo y proyecto se realiza con el propósito de presentar una plataforma propietaria que pueda mejorar la seguridad inicialmente de transmisiones de audio y tal vez en un futuro de cualquier dato en tiempo real mediante encriptación con el algoritmo Salsa20.

Debido a la creciente necesidad de seguridad de la información y datos, y que estas soluciones, particularmente el cifrado de datos, actualmente son proveídas por empresas que involucran software privativo en sus dispositivos, no se tiene la certeza que dada una circunstancia estas medidas de seguridad no serán rotas y los datos vulnerados. Como lo recalca James Ball en su artículo "Secret US cybersecurity report: encryption vital to protect data"[4] uno de los problemas más grandes al proteger negocios y ciudadanos del espionaje, sabotajes y crimen que anualmente le cuestan a la economía global más de 400 millones de dólares, es la clara falta de equilibrio entre el desarrollo de las capacidades ofensivas y defensivas, ya que la adaptación de encriptación y otras tecnologías es más lenta de lo esperado, esto evidencia una clara falencia en seguridad a pesar de los grandes esfuerzos de los gigantes tecnológicos como Google y Apple. En el ámbito colombiano el panorama es precario para muchas empresas, según la firma de seguridad informática estadounidense FireEye, el 98% de las empresas nacionales[5] son objetos de ataques informáticos permanentes, el hecho de que estas empresas tengan presencia en medios de comunicación e internet las hace vulnerables a ataques y generalmente no se percatan de los ataques hasta mucho tiempo después.

[4] "Secret US cybersecurity report: encryption vital to protect ..." 2015. 11 Mar. 2015 <http://www.theguardian.com/us-news/2015/jan/15/-sp-secret-us-cybersecurity-report-encryption-protect-data-cameron-paris-attacks>

[5] "98% de empresas colombianas son víctimas de ataques ..." 2014. 11 Mar. 2015 <http://www.vanguardia.com/actualidad/colombia/250540-98-de-empresas-colombianas-son-victimas-de-ataques-informaticos>

Adicionalmente, en la Universidad Tecnológica de Pereira, son pocos los trabajos de grado enfocados a la criptografía, y el uso de hardware libre es una innovación con respecto al hardware y software usado en los laboratorios prácticos del programa. Este proyecto podría dar lugar a una investigación posterior sobre medios de transmisión encriptados en microcontroladores.

Entre las soluciones posibles para este problema, se plantea la implementación de un sistema de cifrado en tiempo real en plataformas arduino, las cuales son placas con microcontroladores que usan las ideas y conceptos de hardware y software libre.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Implementar el prototipo de un sistema con plataformas Arduino Uno para cifrar, descifrar y transmitir audio cifrado con el algoritmo Salsa20 mediante conexión serial en tiempo real.

1.3.2 OBJETIVOS ESPECÍFICOS

- Estudiar la viabilidad de hardware libre para encriptación en tiempo real
- Definir el proceso de digitalización de una señal análoga de audio mediante arduino
- Especificar la implementación de cifrado y descifrado de los algoritmos planteados mediante arduino
- Medir la calidad de servicio, eficiencia y confiabilidad proveída por el sistema
- Elaborar manuales de usuario y técnicos

1.4 ALCANCE

El trabajo de grado abarcará la implementación de un sistema de transmisión de audio encriptado mediante salsa20, la transmisión consta de un emisor y un receptor los cuales estarán en placas arduino con sus respectivos componentes electrónicos definidos en los recursos tecnológicos

disponibles, el código de programación podrá contener códigos de terceros propiamente acreditados y enunciados en el informe final; la implementación no estará quemada en una baquela, solo en la protoboard. El proyecto generaría un informe que detalle el desarrollo de este, inconvenientes encontrados y conclusiones pertinentes, con el formato adecuado definido en el estándar ICONTEC.

2 MARCO REFERENCIAL

2.1 MARCO TEÓRICO

A causa del incremento en la producción de información y la capacidad para transmitir ésta a través de la red pública más grande jamás creada, internet, la importancia de la criptografía ha crecido debido a su gran aporte a la seguridad. La historia de la criptografía data de la antigüedad, el objetivo principal en sus inicios era la confidencialidad del mensaje (encriptación) y consistía la conversión de los mensajes a un texto incomprensible para los espías y cualquier otro al que no estuviera dirigido el mensaje. Antes de la era moderna, la criptografía y encriptación eran sinónimos, actualmente se hace una más clara diferenciación debido a que ahora la criptografía abarca un abanico más amplio de disciplinas, como las matemáticas, las ciencias de la computación y la ingeniería eléctrica, y se han incluido técnicas como la revisión de la integridad del mensaje, identificación del transmisor/receptor, firmas digitales, pruebas interactivas, computación segura entre otras.

La criptografía ha tenido un papel importante en la historia humana, esto se puede ver en el impacto que tuvo en la segunda guerra mundial. En este evento histórico el criptoanálisis permitió a los aliados conocer de antemano muchas de las comunicaciones cifradas de los alemanes y así atacar o protegerse más eficientemente.

Con el desarrollo de computadores mucho más potentes se hizo posible el desarrollo de cifrados más complejos, ahora es posible la encriptación de cualquier tipo de dato que sea representable en un formato binario, siendo esto nuevo e interesante y suplantando lentamente a la encriptación lingüística. Sin embargo, que se incremente la complejidad de estos cifrados no significa que consuman muchos más recursos, siendo este uno de sus puntos fuertes, ya que un cifrado de buena

calidad usa pocos recursos de CPU y memoria, y requiere considerables esfuerzos computacionales para ser roto.

Entre las muchas áreas de estudio en que la criptografía ha sido dividida, la criptografía de llave simétrica es la que se utiliza en este proyecto, debido que Salsa20, el cifrado seleccionado para desarrollar el proyecto, es un cifrado de flujo basado en criptografía de llave simétrica. El método de cifrado de flujo por su simplicidad de implementación en hardware y su gran ventaja en aplicaciones donde vienen paquetes de datos en longitudes de tamaño desconocidas es ideal para casos como conexiones inalámbricas seguras, o en este caso, transmisiones de audio.

Para la implementación del cifrado de audio en línea en tiempo real mediante salsa20 se planea el uso de arduino, el cual es una plataforma de hardware que permite la fácil manipulación, programación y uso de los microcontroladores, esta plataforma de hardware libre introducida al mercado en 2005 y basada en variados microcontroladores de 8 y 32 bits provee un set de pines de entrada y salida análoga y digital que pueden ser utilizados como interfaz para la comunicación con diversas boards de extensión y circuitos y el microcontrolador. Para programar los microcontroladores la plataforma arduino provee un IDE basado en el proyecto Processing que incluye soporte para los lenguajes C y C++.Debido a que la eficiencia y simplicidad en esta implementación es muy importante, el hardware Arduino opera con un microcontrolador que no posee tanta capacidad de procesamiento ni memoria como un computador portátil o de escritorio actual, y precisamente para esto están destinados. Mientras que un procesador es como un microcomputador que solo se encarga de traer instrucciones de la memoria y ejecutarlas y depende completamente de muchos otros periféricos de hardware y software, en contraste los microcontroladores se han diseñado para ser más diversos en sus áreas de aplicación y mucho más compactos, optimizados para tareas más específicas, estos operan ejecutando un programa que está almacenado permanentemente en su memoria e interactúa con el exterior con líneas de entrada y salida que dispone. Estos microcontroladores poseen mecanismos de seguridad internos que operan sobre sus memorias las cuales son generalmente FLASH o EEPROM, estos mecanismos proveen un alto nivel de confianza en que la llave criptográfica del algoritmo no será vulnerada por métodos de manipulación de hardware como ataques mediante análisis de potencia, microscopia, manipulación de voltaje o de luz y radiación.

2.2 MARCO CONCEPTUAL

Electrónica

La electrónica es la ciencia del manejo de la electricidad. La historia de esta es relativamente reciente pero debido al rápido crecimiento y la diversificación y evolución de sus tres principales componentes [6](el tubo de vacío, el transistor y el circuito integrado) ha tomado un protagonismo al ayudar al avance de la ciencia, tecnología y la comunicación de la humanidad.

La electrónica tiene diferentes ramificaciones como la digital y la análoga, y viene en multitud de presentaciones, en todos los productos electrónicos se combina un poco de cada rama y usualmente los diseñadores recurren a diferentes diseños y combinaciones para optimizar estos sistemas. Es por eso que el uso de esta en desarrollos prácticos que involucran variables reales y sistemas físicos se hace esencial, dada la facilidad en el desarrollo y la integración de hardware debido a la masificación en el consumo y producción de diferentes módulos para diversas necesidades.

Arduino

La masificación de los componentes electrónicos, entre ellos los circuitos integrados hizo que la adquisición de estos sistemas costará poco dinero, sin embargo, para el uso de estos todavía se requieren altos conocimientos de electrónica, electricidad, matemáticas, y muchos otros campos de la ciencia. Es por eso que se hizo necesario el uso de plataformas electrónicas las cuales facilitaran el uso de estos componentes.

Arduino es entonces una plataforma electrónica que ofrece la facilidad de tener en un dispositivo de tamaño reducido, capacidades de salida/entrada de información análoga y digital, regulación de voltaje, frecuencias y esencialmente una comunicación más fácil entre periféricos, con la ventaja de una fácil programación a alto nivel del microcontrolador que es el cerebro de la plataforma.

---

[6] "Electronics - History - Science Encyclopedia - JRank." 2007. 24 Apr. 2015 <http://science.jrank.org/pages/2376/Electronics-History.html>

Otra de las grandes ventajas de arduino, es su amplia red de colaboradores alrededor del mundo, que aportan diariamente librerías y código implementado con buenas prácticas para que el resto de la comunidad lo usen con la iniciativa de código abierto[7], es además una de las plataformas electrónicas más usadas por entusiastas de la filosofía DIY (Do It Yourself), esta y muchas ventajas más hacen de arduino la plataforma electrónica para hardware libre idónea para proyecto.

Microcontroladores

Los microcontroladores [8] son pequeñas computadoras en un pequeño circuito integrado que tienen un procesador, memoria y periféricos de entrada y salida programables. Estos dispositivos están diseñados para aplicaciones embebidas y pueden funcionar a bajas frecuencias y requerimientos de voltaje, o servir roles donde el desempeño es crítico y necesitan de un alto voltaje para funcionar.

Existen muchos tipos de microcontroladores de diversos fabricantes y diferentes arquitecturas, el arduino uno funciona con la marca Atmel, específicamente el Atmega328 PU[9]. El uso de este tipo de circuitos integrados es ideal para las aplicaciones donde se requiere un buen desempeño y bajo costo, ya que tienen la capacidad de ser puestos en producción en ambientes industriales o domésticos con relativa facilidad. En el caso particular, se buscará un prototipo haciendo uso de la placa arduino, y diferentes componentes electrónicos, que serán finalmente manejada por este microcontrolador.

Encriptación

[7] "Open Source Initiative." 2003. 24 Apr. 2015 <http://opensource.org/>
[8] "How Microcontrollers Work - HowStuffWorks - Electronics." 2003. 24 Apr. 2015 <http://electronics.howstuffworks.com/microcontroller.htm>
[9] "Datasheet - Atmel Corporation." 2013. 24 Apr. 2015 <http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf>

La encriptación [10] es la conversión de datos electrónicos a otra forma, llamada texto cifrado, el objetivo principal es proteger la confidencialidad de estos datos que pueden estar almacenados en computadores o transmitidos mediante algún medio. Esto se hace con algoritmos de encriptación que juegan un rol vital en la era de la información actual y siguen 3 principios básicos que son la autenticación, la integridad, y el no repudio.

La encriptación funciona mediante un algoritmo de encriptación, una llave de encriptación y texto plano que es como se le llaman a los datos que son elegibles para el humano, este proceso convierte el texto plano, a texto cifrado, que sólo puede ser llevado a su forma de texto plano de nuevo con el uso de la llave de encriptación correcta. El proceso de descifrado es básicamente revertir los pasos sobre el texto cifrado con el orden en el que las llaves de encriptación fueron aplicadas. En la actualidad los algoritmos de encriptación se dividen en asimétricos y simétricos.

Los algoritmos asimétricos[11] también conocidos como criptografía de llave pública, usan dos claves, una privada y una pública las cuales están relacionadas matemáticamente, en este algoritmo la llave opuesta de la usada para encriptar el mensaje es usada para descifrar.

Los algoritmos simétricos[12] también conocidos como criptografía de llave privada, usan una llave que es usada para encriptar y descifrar el texto, pero el transmisor debe compartir la llave con el receptor antes de que este pueda realizar el proceso de descifrar, la ventaja de estos algoritmos es que son mucho más rápidos que los asimétricos, pero hacen uso de un algoritmo asimétrico para compartir la llave privada.

---

[10] "What is encryption? - SearchSecurity - TechTarget." 2011. 24 Apr. 2015 <http://searchsecurity.techtarget.com/definition/encryption>
[11] "What is asymmetric cryptography - SearchSecurity." 2011. 24 Abr. 2015 <http://searchsecurity.techtarget.com/definition/asymmetric-cryptography>
[12] "What is secret key algorithm (symmetric algorithm ..." 2011. 24 Abr. 2015 <http://searchsecurity.techtarget.com/definition/secret-key-algorithm>

Estos últimos algoritmos simétricos o de llave privada o secreta generalmente caen dentro de las siguientes categorías, cifrados de bloque[13], o cifrados de flujo[14].

Los cifrados de bloque encriptan el texto plano en bloques de un tamaño fijo a la vez, produciendo un bloque de texto cifrado del mismo tamaño, el mejor ejemplo de un algoritmo de este tipo es el AES, estándar actual de los Estados Unidos para proteger para proteger sus archivos confidenciales, este algoritmo acepta bloques de 64 bits y llaves de 56 bits, para producir bloques de texto cifrado de 64 bits.

Los cifrados de flujo encriptan secuencias de bits, uno a la vez, en este método la llave criptográfica y el algoritmo son aplicados a cada dato binario en un flujo de datos, este método es poco usado debido a que si un atacante cambia un bit dentro del texto cifrado puede cambiar un valor del texto plano y todavía podría tener sentido, representando un riesgo a la seguridad.

Salsa20

El algoritmo de encriptación Salsa20[15] es un algoritmo de cifrado de datos que usa el modo CTR[16] para volver un cifrado de bloque, en un cifrado de flujo. Este consiste de una función llamada Snuffle 2005 que usa el núcleo salsa20 para encriptar datos, fue publicada por Daniel J. Bernstein[17] en 2005. El núcleo de esta función opera como una función hash con una entrada de 64(8 bits u (2^8-1)) bytes y una salida de 64 bytes. Este núcleo opera en un bloque de 64 bytes de texto plano haciendo hash con la llave una vez y haciendo xor con el resultado y el texto plano.

Una función hash[18] es una función que mapea valores a enteros pequeños tratando de que estos se produzcan de una forma pseudoaleatoria, incluso si los valores ingresados son iguales o parecidos. Y una función XOR es una operación lógica entre valores binarios (0 o 1).

---

[13] "What is block cipher? - SearchSecurity - TechTarget." 2011. 24 Abr. 2015 <http://searchsecurity.techtarget.com/definition/block-cipher>

[14] "What is stream cipher? - SearchSecurity - TechTarget." 2011. 24 Abr. 2015 <http://searchsecurity.techtarget.com/definition/stream-cipher>

[15] Bernstein, DJ. "Salsa20 specification." 2005. <http://cr.yp.to/snuffle/spec.pdf>

[16] "CTR Mode - Crypto++ Wiki." 2007. 24 Apr. 2015 <http://www.cryptopp.com/wiki/CTR_Mode>

[17] "D. J. Bernstein." 24 Apr. 2015 <http://cr.yp.to/djb.html>

[18] "Hash Functions - HMC Computer Science." 2007. 24 Apr. 2015 <http://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html>

3 ESTADO DEL ARTE

A continuación, se presentan algunos de los trabajos más relevantes publicados en revistas científicas sobre temas sobre los cuales se desarrolló el prototipo como encriptación, microcontroladores, arduino, tiempo real, y salsa20 durante el año 2015 y parte del 2016.

En la universidad de Alicante en el laboratorio de control automatizado y robótica han desarrollado proyectos integrando el uso de arduino[19] con implementos especializados encontrados en laboratorios universitarios de ingeniería, mostrando una rigurosidad en el aprendizaje y la posibilidad de incluir esta plataforma de hardware libre en un ambiente académico para ayudar en el desarrollo de habilidades diversas en estudiantes de ingeniería eléctrica e ingeniería de computación. Mediante el desarrollo de experimentos tan diversos como controles de temperaturas para impresoras 3D, programación de robots humanoides y seguidores de línea, se busca integrar las habilidades de lógica de programación con el manejo y calibración de sensores y actuadores de diversos tipos. Esta experiencia apoya la idea de desarrollar trabajos de grado orientados a la implementación de soluciones de seguridad basadas en hardware libre por su bajo costo y con un estudio riguroso de su desempeño y viabilidad. En el trabajo experimental de una comunicación segura basada en caos [20]se muestra la utilidad del arduino no solo para trabajos de medición y control, sino de procesos que requieren el uso de operaciones matemáticas y algoritmos. Se hace uso de un generador de señales para obtener una señal sinusoidal que se muestra en la entrada análoga del arduino, convirtiéndola en digital con una resolución de 10 bits, posteriormente se generan valores en un mapa logístico que servirán para encriptar el mensaje que se quiere enviar, y finalmente se sigue el proceso inverso en el lado del receptor para obtener de nuevo la señal enviada. En este trabajo se puede ver el desarrollo e implementación de un algoritmo para la comunicación segura con muestreo y transmisión de señales análogas, adicionalmente aunque la encriptación usa operaciones que no requieren un alto grado de complejidad y tiempo de procesamiento, logra ser segura.

[19] Candelas, F. A., García, G. J., Puente, S., Pomares, J., Jara, C. A., Pérez, J., … Torres, F. (2015). Experiences on using Arduino for laboratory experiments of Automatic Control and Robotics.
[20] Zapateiro, M., Hoz, D., Acho, L., & Vidal, Y. (2015). An Experimental Realization of a Chaos-Based Secure Communication Using Arduino Microcontrollers

La placa arduino se diseñó con el propósito de que la interacción con periféricos como sensores y actuadores se pudiera realizar fácilmente, por esta razón existen muchas librerías soportadas por el equipo oficial, y por colaboradores entusiastas a través de internet, en el artículo Implementación de un monitoreo en tiempo real usando arduino y android [21] se lleva esta funcionalidad a otro nivel, ya que se integra la medición y monitoreo en tiempo real apoyado en android, mediante el desarrollo de una aplicación móvil que se comunica directamente con la placa arduino para obtener información inmediata sobre el estado de la fábrica. Esta comunicación se realiza mediante bluetooth, el Smartphone se encarga de establecer la conexión mediante la aplicación para posteriormente acceder a 6 ítems que abordan diversos factores que se ven en una fábrica, estos ítems son: Producción, seguridad, calidad, eficiencia, moral y análisis. Cada ítem puede ser configurado para ajustarse a las necesidades particulares de la fábrica, y es calculado internamente por el arduino y enviado a la aplicación en tiempo real.

La versatilidad del hardware libre abre las posibilidades para ampliar el desarrollo de la encriptación en tiempo real, sin embargo, su limitada memoria y la velocidad del procesador hacen que la optimización sea una tarea prioritaria. Se propone una librería para desarrollar aplicaciones embebidas en tiempo real en C [22] la cual busca afrontar el problema para las aplicaciones embebidas de próxima generación mediante el desarrollo de un framework contenido en una librería que facilita y oculta los detalles de implementación de aplicaciones en tiempo real, la librería se llama ECP-C y busca mejorar 6 áreas : manejo de hilos, intercambio de recursos, manejo de memoria, eventos externos o interrupciones, señalización asíncrona, y acceso a memoria. En la librería se implementan 3 perfiles orientados a hardware con diferentes capacidades de procesamiento, micro, mini y general, estos perfiles se enfocan en elegir cuidadosamente las características que podrían tener un impacto mayor en sistemas con menor capacidad de procesamiento, siguiendo esta idea de diseño, el perfil micro no contiene funcionalidades para la des asignación de memoria ya que estos sistemas no realizan estas operaciones usualmente, sino que la asignación de memoria es permanente, en los dos perfiles restantes se ven estas decisiones de diseño orientadas a la optimización de las implementaciones en tiempo real. Se realizó una evaluación empírica para

[21] Watve, O. J. (2015). Implementation of real time factory information system using arduino and android, 1343–1347

[22] Basanta-Val, P., & García-Valls, M. (2015). A library for developing real-time and embedded applications in C. Journal of Systems Architecture, 61(5-6), 239–255.

medir el sobrecosto total de la inclusión de esta librería sobre 2 prototipos diferentes corriendo sobre microcontroladores ATMega328 (Microcontrolador usado en la placa arduino) , una corriendo directamente sobre el microcontrolador en una implementación desnuda, y la otra sobre un sistema operativo como intermediario llamado ChibiOS presentando resultados prometedores sobre la factibilidad de la implementación en términos de desempeño y huella al analizar el sobrecosto producido por la librería.

El desarrollo de este tipo de librerías permite una optimización que permita mitigar los efectos de recursos limitados, y amplía el horizonte de las aplicaciones embebidas, en el desarrollo de un sistema embebido robusto para la autenticación basado en la huella y encriptación caótica[23], se implementó un sistema orientado al acceso lógico y físico en entornos que requieren seguridad, usando la información biométrica de un individuo previamente almacenada en un microcontrolador de 32 bits con velocidad de reloj de 48 MHz, con un algoritmo de encriptación basado en caos con una clave de 128 bits. La encriptación de la información biométrica se realiza con un mapa logístico, que se genera con unas condiciones iniciales muy susceptibles a cambios y le dan robustez al proceso de encriptación. Este sistema tiene una capacidad máxima de 370 usuarios registrados y se logró una tasa de falso reconocimiento que es una medida para medir cuántas veces se le da acceso a un individuo porque se emparejo adecuadamente su huella con la almacenada de 10^6 (uno en un millón) y una tasa de falso rechazo que es una medida para medir cuántas veces se rechaza a alguien que legítimamente está registrado de 10^2 (uno en cien), este sistema puede considerarse un sistema experto ya que puede realizar autenticación con alto índice de seguridad a un bajo costo y alto desempeño. Se verificó la seguridad de esta propuesta mediante un análisis completo en un nivel estadístico confirmando las altas capacidades de seguridad de este esquema.

En el estudio de sistemas con capacidades de hardware limitadas se miden aspectos como vida de batería, memoria, latencia, ancho de comunicación, etc., estos aspectos se tienen en cuenta a la hora de diseñar sistemas embebidos y a menudo se escogen soluciones minimalistas que se ajusten

[23] Murillo-Escobar, M. A., Cruz-Hernández, C., Abundiz-Pérez, F., & López-Gutiérrez, R. M. (2015). A robust embedded biometric authentication system based on fingerprint and chaotic encryption. Expert Systems with Applications, 42(21), 8198–8211.

a estas capacidades, un estudio exhaustivo de soluciones modernas en criptografía simétrica[24] se encarga de hacer una recopilación para entornos con recursos limitados, permitiendo tener un panorama completo de las necesidades de cada algoritmo ya que esta elección afecta dinámicamente el tiempo de vida y desempeño en los aspectos anteriormente mencionados de un dispositivo. Se evalúa la taxonomía de soluciones criptográficas simétricas como cifradas de clave simétrica, cifrados de involución, de peso ligero, de flujo, redes de sustitución y permutación y redes de feistel. Posteriormente se abarcan más de 60 algoritmos divididos en las categorías anteriores con datos en implementaciones en fpgas. En la categoría de cifrados de flujo se muestra el desempeño de Salsa20/12, y el análisis muestra que este algoritmo presenta su lado fuerte en la velocidad de encriptación, siendo su lado débil la confianza, lo que lo hace un candidato ideal para su implementación en hardware con capacidades limitadas, ya que permite un mayor nivel de respuesta y velocidad que otros algoritmos.

Salsa 20 pertenece a la familia de cifrados ARX donde las 3 operaciones en las que se basan son de adición módulo, rotación y XOR. Se propone el análisis de un ataque de análisis de poder en una implementación de Salsa20[25]. Este tipo de ataque se conoce como DPA (por sus significado en inglés Differential power analysis) y es una forma de ataque a los canales laterales (del inglés side-channel attack), donde se busca información por medio de la implementación física del algoritmo criptográfico, es decir, se miden variaciones en el uso de energía del procesador y se compara con diferentes etapas del proceso de encriptación. En este trabajo se usó el modelo de peso de Hamming que permite modelar el consumo de energía para la ejecución de una instrucción en un microcontrolador de forma precisa. Ya que salsa20 permite usar llaves de 128 y de 256 bits, se usó una llave de 256 para una mayor seguridad, esta llave se divide en 8 vectores k0-k7, se encontró mediante el ataque que no todas las palabras de la llave tienen la misma vulnerabilidad, sin embargo, en el proceso de dos pasos donde primero se determinan ecuaciones entre las palabras y posteriormente se atacan, se pudo descifrar completamente la llave, lo que representa un riesgo si se considera un ambiente en la vida real donde los sistemas embebidos son

[24]Kong, J. H., Ang, L.-M., & Seng, K. P. (2015). A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments. Journal of Network and Computer Applications, 49, 15–50.
[25]Mazumdar, B., Ali, S. S., & Sinanoglu, O. (2015). Power analysis attacks on ARX: An application to Salsa20. In 2015 IEEE 21st International On-Line Testing Symposium (IOLTS) (pp. 40–43).

susceptibles de ser obtenidos y analizados físicamente, por eso se hace imperativo desarrollar contramedidas contra este tipo de ataques. La familia de cifrados ARX tiene una alternativa conocida como cifrado de fase, y se diferencian en las operaciones básicas que hacen para lograr la encriptación del texto plano, se realizó un estudio comparativo para diferenciar claramente la seguridad y desempeño (Huo & Gong, 2015). En el estudio se encontró que la encriptación de fase en la capa física puede resistir ataques de análisis de trafico al compararse con la encriptación XOR, y que en términos de desempeño, al tener un tamaño menor en el vector generado de llave puede ahorrar energía y hacer que los ataques de canales laterales sean más difíciles de realizar, sin embargo, la implementación en hardware puede incurrir en costos adicionales porque requiere un módulo de multiplicación, situación en la que la encriptación XOR es más eficiente en términos de uso de recursos en hardware.

En el ámbito del internet de las cosas, donde se busca que sistemas muy pequeños puedan detectar y enviar información multimedia eficientemente, se propone un modelo para reducir la cantidad de datos que se deben enviar por medio de dispositivos con acceso limitado a redes[26], en este trabajo se aborda un método de reducción de tamaño mediante el uso del mosaico de voronoi, una técnica que transforma una imagen en un vector y usa solo el 1% de volumen original. Para el análisis comparativo se implementaron 3 modelos, uno con compresión usando JPEG-2000, uno de encriptación usando Salsa20, y uno de compresión y encriptación mediante JPEG-AES, y se midieron 2 factores, tiempo promedio end-to-end y consumo total de energía. Los resultados arrojaron una diferencia dramática entre las implementaciones propuestas y el modelo de los autores, logrando un acercamiento muy eficiente en sistemas de sensores multimedia para redes inalámbricas donde la resolución de las imágenes y videos no es de gran importancia.

[26] Mostefaoui, A., Noura, H., & Fawaz, Z. (2015). An integrated multimedia data reduction and content confidentiality approach for limited networked devices. Ad Hoc Networks, 32, 81–97.

4 DESARROLLO

4.1 VIABILIDAD DEL HARDWARE LIBRE EN LA ENCRIPTACIÓN EN TIEMPO REAL

A continuación, se enuncian algunos artículos encontrados donde se emplea el hardware libre Arduino y se tocan temas sensibles para este proyecto de grado. Los temas de interés en los cuales se realizó la búsqueda fueron los siguientes: tiempo real en arduino, encriptación en arduino, transmisión en arduino

Diseño e implementación de un sistema inalámbrico en tiempo real de domótica basado en Arduino-UNO[27] muestra el proceso de diseño e implementación de un sistema para controlar diferentes tipos de electrodomésticos en tiempo real a través de microcontroladores basado en la arquitectura arduino-uno. En este sistema se implementaron dos tipos de funcionamiento, manual y automático. En el modo manual el usuario puede verificar el estado actual de un electrodoméstico en particular y puede manipular su funcionamiento, en el modo automático el sistema monitorea de manera automática los electrodomésticos usando diferentes sensores que están conectados a dichos elementos. La comunicación entre el sistema y el usuario se hace a través de una red inalámbrica conectada a internet lo cual permite el acceso al sistema desde cualquier parte donde se pueda tener una conexión. La conclusión mostrada indica que los controladores funcionan correctamente usando la arquitectura arduino-uno en los dos modos de funcionamiento y demuestra que la comunicación en tiempo real con estos dispositivos es factible[28]

[27] Putra, L., & Kanigoro, B. (2015). Design and Implementation of Web Based Home Electrical Appliance Monitoring, Diagnosing, and Controlling System. *Procedia Computer Science*, *59*, 34–44.
[28] Bader M. O. A, Design and Implementation of a Reliable Wireless Real-Time Home Automation System Based on Arduino Uno Single-Board Microcontroller. Julio 2014 < http://researchpub.org/journal/jac/number/vol3-no3/vol3-no3-3.pdf >

Un sistema de criptografía RSA minúsculo basado en un microcontrolador arduino útil para redes pequeñas[29] se realiza la implementación de un sistema criptográfico llamado RSA[30], este es un algoritmo de encriptación donde tanto el receptor como el emisor usan la misma llave para decodificar el mensaje. En la implementación se usó un dispositivo Arduino Mega2560R3 además de una pantalla lcd y un teclado básico para digitar el mensaje, al probar el método de encriptación en la medición de cantidad de bytes contra tiempo, se demuestra que a medida que aumenta el tamaño del dato el tiempo de encriptación es mayor, esto debido a las características limitadas de procesamiento del dispositivo. Se obtiene que el algoritmo de encriptación funciona de manera correcta usando el dispositivo arduino y demostrando que sus características son suficientes para este tipo de procesos en determinado tamaño de datos, así mismo muestra su versatilidad al soportar periféricos de entrada y salida diversos.

Diseño de aplicación móvil para la comunicación inalámbrica de señales audiovisuales[31] Este trabajo es la realización de una aplicación que mediante un módulo arduino conectado por Ethernet a un router permite la transmisión de archivos audiovisuales apoyado en Android. Básicamente es un sistema de arquitectura cliente servidor donde el dispositivo Android actúa como cliente y el módulo arduino como servidor. El usuario hace una petición a través del dispositivo Android, esta petición va al servidor; en este caso el módulo arduino es quien envía los datos por partes. Al enviar una parte el dispositivo espera una señal de confirmación para verificar que el dato llegó y además que este no sea un dato corrupto, si el dato no llega o lo hace de manera errónea el fragmento del archivo es enviado de nuevo, en caso contrario se envía el siguiente fragmento hasta completar el proceso

[29] Qasem Abu A.,A tiny RSA Cryptosystem based on arduino Microcontroller useful for a Small Scale Networks.

[30] "Qué es RSA?." 2007. 2016 de Feb 19. {En línea}: <https://seguinfo.wordpress.com/2007/09/14/%C2%BFque-es-rsa/ >

[31] Alberto Esteban Pérez "Diseño de aplicación móvil para la comunicación inalámbrica de señales de audiovisuales junio 2013"
<http://upcommons.upc.edu/bitstream/handle/2099.1/21895/alberto.esteban.perez_90828.pdf?sequence=1>

El modulo necesario para la trasmisión de datos es una ETHERNET SHIELD este dispositivo adicional provee de conectividad a la placa arduino-UNO además de tener una ranura de tarjeta SD que da la posibilidad de expandir la memoria básica de la plataforma de hardware

Características ETHERNET SHIELD

Chip	**Conectividad**	**Tipo de trasmisión**	**Conexión por socket**	**Almacenamiento**
Wiznet W5100	Permite conexión a internet	UDP - TCP	soporta 4 conexiones simultaneas	dispone una ranura para SD

En las pruebas realizadas en el sistema para medir los tiempos de trasmisión entre el cliente y el servidor se realizó un promedio con tres tipos de archivos (imagen, texto, audio) y se obtuvieron los siguientes resultados

Tipo de archivo	**Tamaño del archivo**	**Tiempo de transferencias**	**Velocidad**
imagen	432KB	25.987 segundos	16.62KBps
texto	1.73KB	0.254 segundos	6.81KBps
audio	1.57MB	90.437 segundos	17.26KBps

La importancia de las pruebas radica en el tamaño del archivo y las velocidades para cada uno, la implementación obtiene unos tiempos que se podrían considerar de baja velocidad con dispositivos modernos, sin embargo, tomando en consideración las capacidades de desempeño del microcontrolador demuestra que es posible el envío de tales archivos.

Es importante resaltar que la calidad del audio es parte importante e influye en el peso final, y se debe realizar una elección entre desempeño y calidad. Las limitaciones de acuerdo a los resultados de este trabajo se pueden observar en archivos de tamaño mayor a 1 Mb.

Una realización experimental de la comunicación segura basada en el caos usando microcontroladores Arduino-UNO[32]. Se realiza una comunicación segura entro dos puntos usando microcontroladores de la plataforma arduino-UNO; tanto en el emisor como en el receptor el tipo de dato a trasmitir es una señal, el procedimiento se compone recepción, encriptación, envío y descifrado de la señal.

En el transmisor la placa Arduino recibe el mensaje m (□) a través de uno de sus puertos de entrada analógica. Se muestra el mensaje de entrada analógica, □ (□), y convierte la señal m en digital mediante muestreo. Esta señal es entonces cifrada mediante el uso de un mapa logístico y un simple Delta modulador. Después, una señal clave, □ (□), se genera con el fin de descifrar el mensaje en el receptor, para mayor seguridad esta clave se cifra de nuevo usando un segundo mapa logístico. Como resultado, el transmisor Arduino genera tres salidas: la primera es el mensaje cifrado, □□ (□), la segunda es la señal clave cifrada, □□ (□), y el tercero es una señal de llave auxiliar, □1 (□), que se utiliza para fines de descifrado. Posteriormente tres canales por cable se utilizan para enviar el mensaje cifrado y los mensajes clave.

En el receptor se toman las señales □□ (□), □□ (□), y □1 (□) para descifrar la señal modulada-Delta antes se convierte en su forma analógica. La salida es una señal digital, □□ (□), que corresponde a la señal de □□ (□). El delta demodulador es el segundo bloque en el receptor. Consiste en un integrado, un filtro, y algunos amplificadores para recuperar el mensaje original. Su salida es una señal de □□ (□) que se aproxima a la □ señal original (□).

El método de encriptación usado fue el de mapas logísticos, estos son un modelo matemático originalmente creado para predecir el crecimiento de una población dadas unas condiciones iniciales; se considera un sistema caótico ya que cualquier variación en la variable inicial puede causar un gran cambio en el sistema.

[32] Zapateiro, M., Hoz, D., Acho, L., & Vidal, Y. (2015). An Experimental Realization of a Chaos-Based Secure Communication Using Arduino Microcontrollers - ProQuest. Retrieved January 18, 2016

Está definido por la siguiente ecuación: □□ + 1 = □□□ (1-□□), 0≤□□ ≤ 1,

Para la encriptación se generan dos valores □1 (□) y □2 (□) mediante mapas logísticos. Estos tienen diferentes condiciones iniciales; es decir, □1 (0) = □/2 (0). En primer lugar, el mensaje se codifica con un valor verdadero o falso que se asigna en función de la □1 valor (□) del primer mapa caótico como puede verse en la parte 1 de la Figura 1, donde □□ (□) es el mensaje encriptado y □ (□) es la clave necesaria para recuperar □□ (□). Con el fin de aumentar la seguridad del sistema, la clave, □ (□), se cifra aún más siguiendo el mismo esquema. Esto asignándole un valor verdadero o falso que depende el segundo valor del mapa caótico, □2 (□), como se muestra en el la parte 2 de la Figura 1 donde □1 (□) y □2 (□) son señales auxiliares que son utilizadas para cifrar y descifrar la señal de clave □ (□). La clave es entonces finalmente cifrada mediante la aplicación de La función XOR con los □1 Variables (□) y □2 (□) para producir lo que se muestra en la parte 3 en la Figura 1. Las señales □□ (□), □□ (□), y □1 (□) se envían al receptor a través de salidas digitales D2, D4, D7. En el receptor, las señales □□ (□), □□ (□) y □1 (□) va reciben en las entradas de Arduino D2, D4, D7.

```
Part 1
(1) if x1(k) > 0.5 then
(2)     me(k) = mb(k)
(3)     s(k) = true
(4) else
(5)     me(k) = !mb(k)          ▷Symbol ! means boolean negation
(6)     s(k) = false
(7) end if
Part 2
(8) if x2(k) < 0.1 then
(9)     s1(k) = !s(k)           ▷Symbol ! means boolean negation
(10)    s2(k) = true
(11) else
(12)    s1(k) = s(k)
(13)    s2(k) = false
(14) end if
Part 3
(15) se(k) = (!s1(k) AND s2(k)) OR (s1(k) AND !s2(k))
```

Ilustración 1. Algoritmo de cifrado de una señal

Al enviar una señal sinusoidal de 125 Hz con una amplitud de 5V se puede observar la señal recuperada por el osciloscopio en color amarillo en la Figura 2 dista de la señal original, representada en color azul.

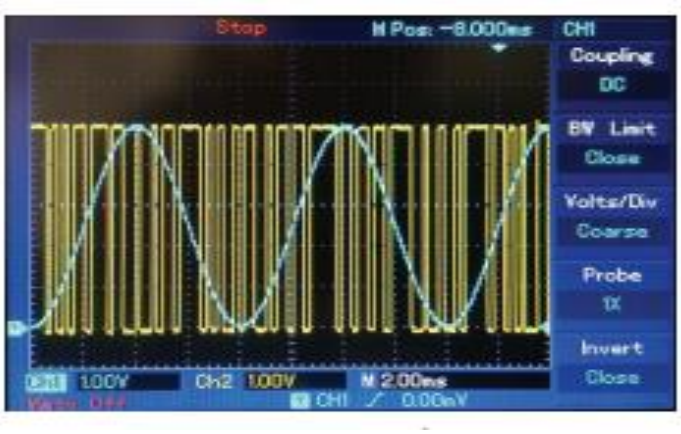


Ilustración 2. Señal cifrada y original

Gracias a los artículos anteriormente analizados es posible validar la capacidad de los dispositivos arduino en las tareas de trasmisión de datos, procesamiento y versatilidad a la hora de implementar algoritmos, cuando se realizan este tipo de procesos es necesario tener siempre presente las características tanto de hardware y de software, debido a las características técnicas de arduino como la velocidad de procesamiento y la memoria interna pueden limitar los tamaños de la información que se desea procesar. Sin embargo, debido a su lenguaje de programación basado en C y los múltiples periféricos tanto de entrada como de salida convierten a este elemento en una opción viable para gran número de aplicaciones.

4.2 DEFINIR EL PROCESO DE DIGITALIZACIÓN DE UNA SEÑAL ANÁLOGA DE AUDIO MEDIANTE ARDUINO

Previo a la digitalización de la señal, se realiza la captura de la onda sonora a través de un micrófono dinámico unidireccional que provee una salida monofónica, es decir, con un único canal de salida.

En la práctica esto se representa mediante dos cables, el que transporta la información de la señal, y el otro que es un punto de referencia conocido como neutro o tierra. Para el transporte de la señal se utilizan cables UTP (por sus siglas en inglés Unshielded Twisted Pair), ya que son de fácil acceso y manipulación; este tipo de cables son los que usualmente transportan las señales del internet.

El proceso de digitalización de la señal se lleva a cabo en tres etapas, dos de las cuales corresponden a un proceso electrónico que incluye la amplificación de la señal y su desplazamiento (Ilustración 3),este último se conoce comúnmente como “DC offset”. La tercera etapa es el código necesario en el arduino para la captura de la señal análoga.

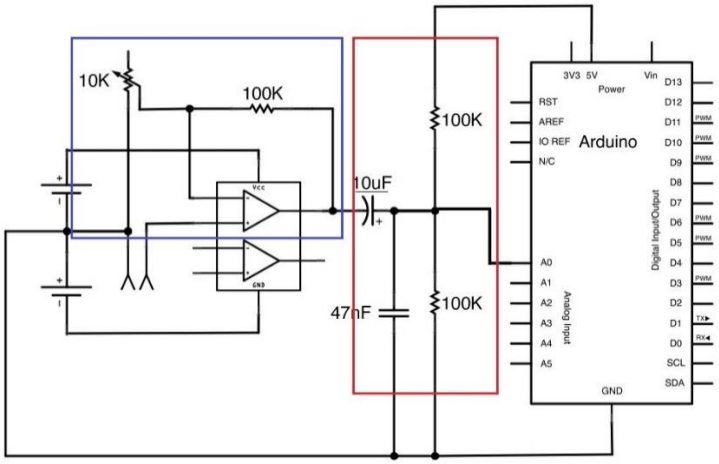


Ilustración 3. Digitalización de una señal

En la amplificación de la señal, la onda sinusoidal que provee el micrófono tiene una amplitud inicial cercana a 200 mV, esto significa que la onda oscila entre -200 mV y 200mV, con centro en 0. Puesto que las entradas análogas del arduino tienen una sensibilidad de 0 a 5V, es necesario aumentar la amplitud de la señal original aproximadamente 1000 veces; para lograrlo se conecta la entrada que contiene la información a un amplificador operacional que en este caso corresponde a un TL028 (Ilustración 4). De esta manera se obtiene en la salida una señal que varía entre -2V y 2V. Para suplir el costo energético que el sistema acarrea se usan dos baterías de 9V en serie, logrando los 18V que requiere el amplificador, cabe aclarar que este amplificador tiene 2 circuitos

internos, pero solo se usa 1, lo cual abre la posibilidad implementar un canal adicional y obtener una señal estéreo (2 canales).

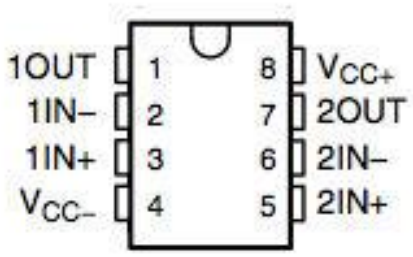


Ilustración 4. Amplificador

Adicionalmente esta porción del circuito utiliza un potenciómetro lineal de 10k Ohm conectado como una resistencia variable con el propósito de ajustar la ganancia, es decir, la cantidad que el amplificador operacional amplifica la señal, ya que puede darse el caso de que la señal que proviene del micrófono sea mucho mayor a 200 mV en cuyo caso después de realizar el proceso en el circuito la salida sea mucho mayor a 5V y el rango de sensibilidad del arduino no permita capturarla, resultando en un recorte de los puntos máximos y mínimos de esta. El potenciómetro puede utilizarse para aumentar la resistencia entre los dos puntos de la señal de salida y reducir la amplitud, lo que devolvería a valores normales la entrada análoga y permitiría su muestreo adecuado mediante el arduino.

En el desplazamiento del centro de la señal, conocido como DC offset, el objetivo es que esta oscile con un centro en 2.5 V con límites en 0 y 5 V para que se quede en el rango aceptable de la entrada análoga del arduino. Este tiene 2 componentes principales, un divisor de voltaje y un condensador. El divisor de voltaje se logra mediante 2 resistencias de 100k Ohm conectadas en serie a la salida de 5V del arduino y la tierra, ya que las dos resistencias son de igual valor, el voltaje en la unión es igual a 2.5 V. Esta unión se une a la salida del amplificador operacional mediante un condensador de 10uF, como el voltaje del lado del amplificador en el condensador aumenta y disminuye, esto

causa que el condensador se cargue momentáneamente y rechace del lado de la unión de 2.5 V. Esto causa que el voltaje en la unión oscile arriba y abajo, centrado en 2.5 V, logrando el objetivo inicial.

Después de esta etapa concluye el tratamiento electrónico que se le debe dar a la señal, ahora es necesario capturar la señal en el arduino, para hacer esto se deben tener algunas consideraciones con respecto a la señal, la calidad de esta y la capacidad de procesamiento del arduino.

En el software incluido en el arduino existe una librería llamada ADC (por sus siglas en inglés Analog to Digital Conversion) la cual se usa para convertir un voltaje análogo que continuamente varía dentro de un rango conocido, a un valor digital. El valor análogo a menudo representa una medida real en el mundo y estos valores se producen ya sea por la presión del aire o la temperatura o velocidad que son inherentemente análogas. El funcionamiento interno del ADC escapa a nuestro rango de estudio para más información referirse a 'What is an ADC'[33] sin embargo es necesario comprender su uso para elegir adecuadamente ciertos aspectos que son susceptibles de configuración y afectan el desempeño.

Los microcontroladores Atmega328, los cuales son la unidad de procesamiento central del Arduino Uno, están diseñados con una arquitectura conocida como AVR, la cual fue una de las primeras en su tipo en usar memoria tipo flash. El ADC de estos dispositivos tiene una velocidad recomendada de entre 50 kHz y 200 kHz para una resolución de 10 bits. Esto significa una lectura de 50,000 a 200,000 ciclos cada segundo, al aumentar más la velocidad, la resolución empezará a degradarse. Existe un método para controlar el reloj del ADC, este es necesario ya que la velocidad del reloj del procesador es muy alta, 16 MHz, o 16'000.000 de ciclos cada segundo, y algunos componentes no pueden alcanzar tales velocidades; se define el termino prescaler o predivisor como un divisor de frecuencia usando un factor múltiplo de 2 para reducir la velocidad del reloj del procesador, y poder hacer uso de esta señal se usa una división entera o prescaler para reducir esta velocidad a los valores que requeridos.

[33] "What is an ADC" {En línea}. Fecha[29 de Enero 2016] [Disponible en:]<http://www.microsmart.co.za/technical/2014/03/01/advanced-arduino-adc>

Se proveen los siguientes valores por defecto: of 2, 4, 8, 16, 32, 64 & 12, así que los valores que se pueden lograr con el prescaler se muestran en la Ilustración 5. De acuerdo a estos valores, es posible usar el valor de 128 y lograr un reloj de 125 kHz. Para lograr una velocidad de reloj mayor en el ADC se podría reducir la velocidad del núcleo central, en cuyo caso un núcleo a 12 MHz con un prescaler de 64 lograra una velocidad de reloj de aproximadamente 187 KHz. Estos son los tipos de intercambios que se deben realizar cuando se diseña un sistema.

Velocidad del reloj	**Factor divisor**	**Velocidad resultante**
16 MHz	2	8 MHz
16 MHz	4	4 MHz
16 MHz	8	2 MHz
16 MHz	16	1 MHz
16 MHz	32	500 KHz
16 MHz	64	250 KHz
16 MHz	128	125 KHz

Una conversación normal en el ADC toma alrededor de 13 ciclos del reloj, así que se debe factorizar hacia abajo por 13 para descubrir cuántas muestras podremos lograr cada segundo. Con una velocidad de reloj para el ADC de 125 kHz se divide por 13 para encontrar el número de muestras, en cuyo caso es una velocidad de muestreo de 9600 Hz, este es el límite físico con este prescaler, y no es posible conseguir mejores velocidades con una resolución de 10 bits. Cuando se habla de audio hay una variedad de combinaciones a considerar, para el caso particular no se requiere alta fidelidad, así que se optó por una tasa de muestreo de 8Khz, común en líneas telefónicas con una profundidad de 10 bits, lo cual otorga una tasa de transmisión de 80 kbps.

Para la etapa final se hace uso de la función analogRead, la cual es nativa del software arduino, esta función se encarga de convertir un valor análogo en una entrada en determinado momento en el tiempo y devolver un valor digital con resolución de 10 bits. Se tienen 5 entradas análogas, en el prototipo se usa la entrada A0 sin ninguna razón en particular. El arduino cuenta con 3 tipos de memoria: la memoria flash de 32KB se utiliza para almacenar el sketch o código fuente que ejecutara el arduino y solo sirve a este propósito, la sram de 2 KB se utiliza para almacenar y manipular las variables en el momento de ejecución y finalmente la eeprom de 1 KB que puede

utilizarse para almacenar variables a largo plazo en el microcontrolador del arduino uno. Puesto que el espacio es limitado y la cantidad de información que se genera por segundo es significativa al compararse con el tamaño de almacenamiento del arduino uno se optó por ampliarlo con un módulo de microSD cuya utilidad es ampliar nuestra capacidad de almacenamiento, en el caso particular a 4GB mediante una memoria microSD.

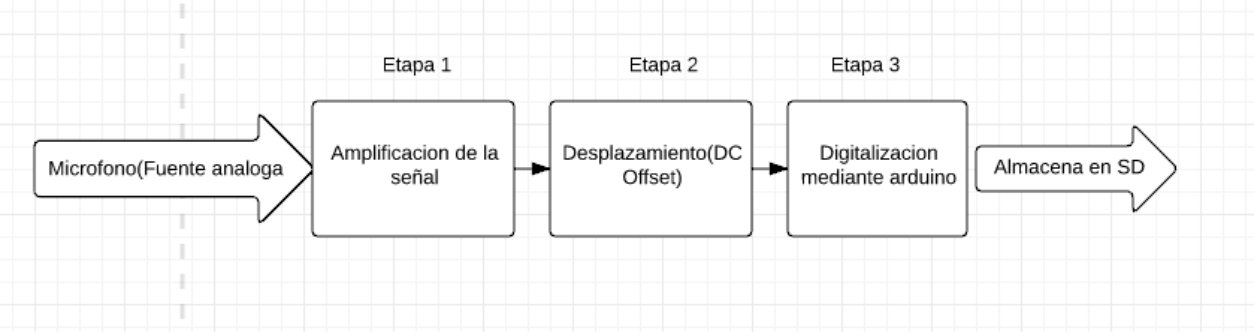


Ilustración 5. Proceso captura de audio

4.3 IMPLEMENTACIÓN DE CIFRADO Y DESCIFRADO DE LOS ALGORITMOS PLANTEADOS MEDIANTE ARDUINO

El núcleo de salsa20 es una función hash con 64 bytes de entrada y 64 bytes de salida[34] diseñada para encriptar eficientemente, basada en 3 operaciones simples que se realizan 320 veces en sucesión y la cual carga muy pocos estados en los bloques. Las 3 operaciones elegidas por el diseñador de la función hash son la adición de 32 bits, XOR de 32 bits y rotación de distancia constante de 32 bits.

Estas operaciones se realizan sobre lo que el autor define una palabra, que son expresiones de 4 bytes (32 bits). La suma de dos palabras se hace en módulo 2^32, es decir cuando la suma sobrepasa el máximo de 2^32-1, el número se envuelve alrededor de sí mismo, como los relojes basados en sistemas de 12 horas, se define como +. El XOR de dos palabras es la suma de estas con el acarreo suprimido, se define como $\oplus$. La operación final es una rotación de distancia constante a la

[34]Bernstein, DJ. "Salsa20 specification." 2005. <http://cr.yp.to/snuffle/spec.pdf>

izquierda, también conocida como rotación circular, esta operación tiene como propósito difundir los cambios de los bits más significativos, a los menos significativos y está definida por el símbolo "<<< " por ejemplo, aplicar repetidamente rotaciones circulares a la siguiente expresión (a, b, c, d) sucesivamente

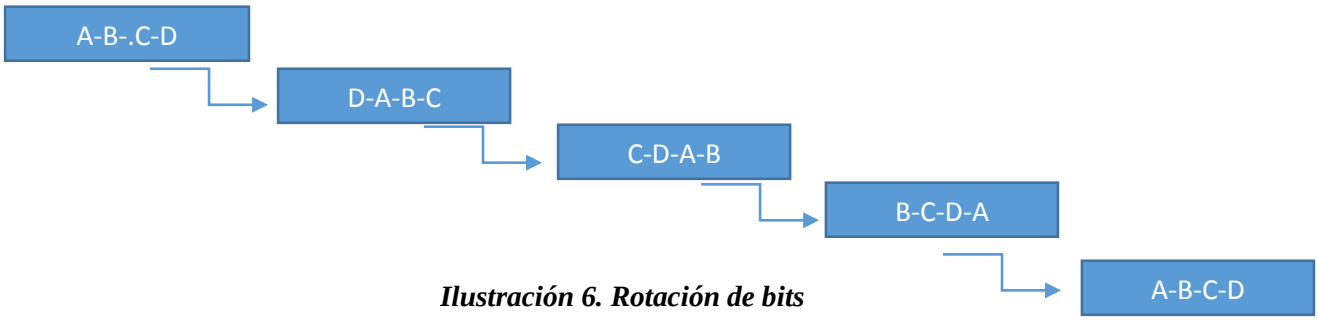


Ilustración 6. Rotación de bits

Aplicando la rotación de bits se va iterando hasta llegar a la expresión original

La función quarterround

Salidas y entradas

Esta función recibe una entrada de 4 palabras y aplica una transformación dando como resultado una salida de 4 palabras.

Definición de la función

Si y = (y0, y1, y2, y3) entonces *quarterround* (y) = (z0, z1, z2, z3) donde
z1 = y1 $\oplus$((y0 + y3) <<< 7),
z2 = y2 $\oplus$((z1 + y 0) <<< 9),
z3 = y3 $\oplus$((z2 + z1) <<< 13),
z0 = y0 $\oplus$((z3 + z2) <<< 18).

Se puede observar que la función suma palabras, luego hace una rotación de un valor constante, y finalmente la palabra resultante se resuelve como un XOR con otra palabra. A continuación, se muestran 2 ejemplos de los resultados de la función aplicados a una entrada, los ejemplos se muestran en una notación 0x, y hexadecimal, ya que sería muy extenso tener los valores en binario.

$$\begin{aligned}&\text{quarterround}(\texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000001})\\ &\qquad = (\texttt{0x00048044}, \texttt{0x00000080}, \texttt{0x00010000}, \texttt{0x20100001}).\\ &\text{quarterround}(\texttt{0xe7e8c006}, \texttt{0xc4f9417d}, \texttt{0x6479b4b2}, \texttt{0x68c67137})\\ &\qquad = (\texttt{0xe876d72b}, \texttt{0x9361dfd5}, \texttt{0xf1460244}, \texttt{0x948541a3}).\end{aligned}$$

Ilustración 7. Función quarterround

La función rowround

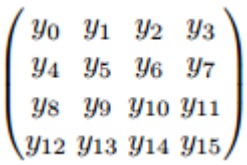
Entradas y salidas

Esta función tiene como entrada 16 palabras y aplica una transformación que da como resultado una secuencia de 16 palabras

Definición de la función

Si y = (y0, y1, y2, y3,. . ., y15) entonces *rowround* (y) = (z0, z1, z2, z3, . . ., z15) donde
(z0, z1, z2, z3) = *Quaterround* (y0, y1, y2, y3),
(z5, z6, z7, z4) = *Quaterround* (y5, y6, y7, y4),
(z10, z11, z8, z9) = *Quaterround* (y10, y11, y8, y9),
(z15, z12, z13, z14) = *Quaterround* (y15, y12, y13, y14).

Se puede visualizar la entrada de esta función como una matriz cuadrada de 4 x 4



$$\begin{pmatrix} y_0 & y_1 & y_2 & y_3 \\ y_4 & y_5 & y_6 & y_7 \\ y_8 & y_9 & y_{10} & y_{11} \\ y_{12} & y_{13} & y_{14} & y_{15} \end{pmatrix}$$

Ilustración 8. Matriz 4x4

Esta función modifica las filas de la matriz en paralelo al proveer una permutación a cada fila por medio de la función *Quaterround.* A continuación, se muestra un ejemplo de la transformación que aplica la función *rowround, se* utiliza la misma notación 0x, para mayor facilidad de lectura.

$$\begin{aligned}
&\text{rowround}(\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000})\\
&\quad = (\texttt{0x08008145}, \texttt{0x00000080}, \texttt{0x00010200}, \texttt{0x20500000},\\
&\qquad \texttt{0x20100001}, \texttt{0x00048044}, \texttt{0x00000080}, \texttt{0x00010000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00002000}, \texttt{0x80040000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000200}, \texttt{0x00402000}, \texttt{0x88000100}).\\
&\text{rowround}(\texttt{0x08521bd6}, \texttt{0x1fe88837}, \texttt{0xbb2aa576}, \texttt{0x3aa26365},\\
&\qquad \texttt{0xc54c6a5b}, \texttt{0x2fc74c2f}, \texttt{0x6dd39cc3}, \texttt{0xda0a64f6},\\
&\qquad \texttt{0x90a2f23d}, \texttt{0x067f95a6}, \texttt{0x06b35f61}, \texttt{0x41e4732e},\\
&\qquad \texttt{0xe859c100}, \texttt{0xea4d84b7}, \texttt{0x0f619bff}, \texttt{0xbc6e965a})\\
&\quad = (\texttt{0xa890d39d}, \texttt{0x65d71596}, \texttt{0xe9487daa}, \texttt{0xc8ca6a86},\\
&\qquad \texttt{0x949d2192}, \texttt{0x764b7754}, \texttt{0xe408d9b9}, \texttt{0x7a41b4d1},\\
&\qquad \texttt{0x3402e183}, \texttt{0x3c3af432}, \texttt{0x50669f96}, \texttt{0xd89ef0a8},\\
&\qquad \texttt{0x0040ede5}, \texttt{0xb545fbce}, \texttt{0xd257ed4f}, \texttt{0x1818882d}).
\end{aligned}$$

Ilustración 9. Función rowround

La función columnround

Entradas y salidas

Esta función tiene como entrada 16 palabras y aplica una transformación que da como resultado una secuencia de 16 palabras

Definición de la función

Si x = (x0, x1, x2, x3, . . ., x15) entonces columnround(x) = (y0, y1, y2, y3, . . ., y15) donde
(y0, y4, y8, y12) = *Quaterround* (x0, x4, x8, x12),
(y5, y9, y13, y1) = *Quaterround* (x5, x9, x13, x1),
(y10, y14, y2, y6) = *Quaterround* (x10, x14, x2, x6),
(y15, y3, y7, y11) = *Quaterround* (x15, x3, x7, x11).

Al igual que la función rowround se puede visualizar la entrada como una matriz cuadrada 4x4

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix}$$

Ilustración 10. Matriz 4x4

La función es la traspuesta de la función *rowround*: modifica las columnas de la matriz en paralelo al proveer una permutación a cada columna a través de la función *Quaterround*. A continuación, se muestra un ejemplo de la transformación que aplica la función *columnround*, se utiliza la misma notación 0x, para mayor facilidad de lectura.

$$
\begin{aligned}
&\text{columnround}(\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000})\\
&= (\texttt{0x10090288}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000101}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00020401}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x40a04001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}).\\
&\text{columnround}(\texttt{0x08521bd6}, \texttt{0x1fe88837}, \texttt{0xbb2aa576}, \texttt{0x3aa26365},\\
&\qquad \texttt{0xc54c6a5b}, \texttt{0x2fc74c2f}, \texttt{0x6dd39cc3}, \texttt{0xda0a64f6},\\
&\qquad \texttt{0x90a2f23d}, \texttt{0x067f95a6}, \texttt{0x06b35f61}, \texttt{0x41e4732e},\\
&\qquad \texttt{0xe859c100}, \texttt{0xea4d84b7}, \texttt{0x0f619bff}, \texttt{0xbc6e965a})\\
&= (\texttt{0x8c9d190a}, \texttt{0xce8e4c90}, \texttt{0x1ef8e9d3}, \texttt{0x1326a71a},\\
&\qquad \texttt{0x90a20123}, \texttt{0xead3c4f3}, \texttt{0x63a091a0}, \texttt{0xf0708d69},\\
&\qquad \texttt{0x789b010c}, \texttt{0xd195a681}, \texttt{0xeb7d5504}, \texttt{0xa774135c},\\
&\qquad \texttt{0x481c2027}, \texttt{0x53a8e4b5}, \texttt{0x4c1f89c5}, \texttt{0x3f78c9c8}).
\end{aligned}
$$

Ilustración 11. Función columround

La función doubleround

Entradas y salidas

Esta función tiene como entrada 16 palabras y aplica una transformación que da como resultado una secuencia de 16 palabras

Definición de la función

La función doubleround aplica una columnround seguida por una rowround: doubleround(x) = rowround (columnround(x)). SE puede visualizar como modificar las columnas de la entrada en

paralelo, y luego modificar las filas en paralelo, cada palabra en la entrada se modifica 2 veces. A continuación se muestra un ejemplo de la transformación que aplica la función *doubleround*, se utiliza la misma notación 0x, para mayor facilidad de lectura.

$$\begin{aligned}
&\text{doubleround}(\texttt{0x00000001}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000},\\
&\qquad \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000}, \texttt{0x00000000})\\
&= (\texttt{0x8186a22d}, \texttt{0x0040a284}, \texttt{0x82479210}, \texttt{0x06929051},\\
&\qquad \texttt{0x08000090}, \texttt{0x02402200}, \texttt{0x00004000}, \texttt{0x00800000},\\
&\qquad \texttt{0x00010200}, \texttt{0x20400000}, \texttt{0x08008104}, \texttt{0x00000000},\\
&\qquad \texttt{0x20500000}, \texttt{0xa0000040}, \texttt{0x0008180a}, \texttt{0x612a8020}).\\
&\text{doubleround}(\texttt{0xde501066}, \texttt{0x6f9eb8f7}, \texttt{0xe4fbbd9b}, \texttt{0x454e3f57},\\
&\qquad \texttt{0xb75540d3}, \texttt{0x43e93a4c}, \texttt{0x3a6f2aa0}, \texttt{0x726d6b36},\\
&\qquad \texttt{0x9243f484}, \texttt{0x9145d1e8}, \texttt{0x4fa9d247}, \texttt{0xdc8dee11},\\
&\qquad \texttt{0x054bf545}, \texttt{0x254dd653}, \texttt{0xd9421b6d}, \texttt{0x67b276c1})\\
&= (\texttt{0xccaaf672}, \texttt{0x23d960f7}, \texttt{0x9153e63a}, \texttt{0xcd9a60d0},\\
&\qquad \texttt{0x50440492}, \texttt{0xf07cad19}, \texttt{0xae344aa0}, \texttt{0xdf4cfdfc},\\
&\qquad \texttt{0xca531c29}, \texttt{0x8e7943db}, \texttt{0xac1680cd}, \texttt{0xd503ca00},\\
&\qquad \texttt{0xa74b2ad6}, \texttt{0xbc331c5c}, \texttt{0x1dda24c7}, \texttt{0xee928277}).
\end{aligned}$$

Ilustración 12. Función doubleround

La función littleendian

Entradas y salidas

Si la entrada es una secuencia de 4 bytes entonces la salida resultante es una palabra.

Definición de la función

SI b = (b0, b1, b2, b3) entonces littleendian (b) = b0 + 2^8.b1 + 2^16.b2 + 2^24.b3.

El propósito principal de la función es tener estandarizadas las palabras para el uso de la función salsa20 en formato little endian. A continuación, se presentan unos ejemplos.

$$\begin{aligned} \text{littleendian}(0,0,0,0) &= \texttt{0x00000000}. \\ \text{littleendian}(86,75,30,9) &= \texttt{0x091e4b56}. \\ \text{littleendian}(255,255,255,250) &= \texttt{0xfaffffff}. \end{aligned}$$

Ilustración 13. Función littleendian

La función hash de Salsa20

Entradas y salidas

SI la entrada es de 64 bytes entonces Salsa20(x) produce un resultado de 64 bytes.

Definición de la función

Salsa20(x) = x + doubleround^10(x), donde cada secuencia de 4 bytes es vista como una palabra en formato little endian.

La transformación que aplica esta función a la entrada x comienza con la conversión de la entrada x de 64 bytes, a una cadena de 16 posiciones con una palabra en cada posición mediante el uso de la función *littleendian.* Después de obtener esta cadena, se aplica la función *doubleround* 10 veces para obtener una transformada z de 16 posiciones, posterior a la realización de las transformaciones se obtiene una salida de 64 bytes nuevamente concatenando la inversa de la función littleendian con la suma de la entrada x y la transformada z en la misma posición como se muestra a continuación.

$$
\begin{aligned}
&\text{littleendian}^{-1}(z_0 + x_0),\\
&\text{littleendian}^{-1}(z_1 + x_1),\\
&\text{littleendian}^{-1}(z_2 + x_2),\\
&\vdots\\
&\text{littleendian}^{-1}(z_{15} + x_{15}).
\end{aligned}
$$

Ilustración 14. Función littleendian invertida

Esta salida es el resultado de Salsa20(x). A continuación, se muestra un ejemplo

$$
\begin{aligned}
\text{Salsa20}(\ & 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)\\
= (\ & 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,\\
& 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).\\
\text{Salsa20}(&211,159, 13,115, 76, 55, 82,183, 3,117,222, 37,191,187,234,136,\\
&49,237,179, 48, 1,106,178,219,175,199,166, 48, 86, 16,179,207,\\
&31,240, 32, 63, 15, 83, 93,161,116,147, 48,113,238, 55,204, 36,\\
&79,201,235, 79, 3, 81,156, 47,203, 26,244,243, 88,118,104, 54)\\
= (&109, 42,178,168,156,240,248,238,168,196,190,203, 26,110,170,154,\\
&29, 29,150, 26,150, 30,235,249,190,163,251, 48, 69,144, 51, 57,\\
&118, 40,152,157,180, 57, 27, 94,107, 42,236, 35, 27,111,114,114,\\
&219,236,232,135,111,155,110, 18, 24,232, 95,158,179, 19, 48,202).
\end{aligned}
$$

Ilustración 15. Función salsa20

La función de encriptación Salsa20

Entradas y salidas

Donde k es una secuencia de 32 bytes o 64 bytes. Donde v es una secuencia de 8 bytes y m es una secuencia de L-bytes donde L ∈ {0, 1,. . ., 2^70}.La función de encriptación Salsa20 de m con un nonce (llave única) v bajo una llave k, se denota como Salsa20k (v) ⊕m, es una secuencia de L-bytes. Normalmente k es una llave secreta preferiblemente de 32 bytes; v es un nonce o un número único de mensaje; m es un mensaje en texto plano; y Salsa20k (v) ⊕m es un mensaje cifrado. O m puede ser un mensaje cifrado, en cuyo caso Salsa20k (v) ⊕m es el texto plano original.

Gracias a la interacción de las funciones descritas anteriormente es posible la encriptación de este algoritmo, para mostrar una relación más detallada del mismo en el siguiente diagrama se puede ver el flujo de los datos a través del proceso de encriptado

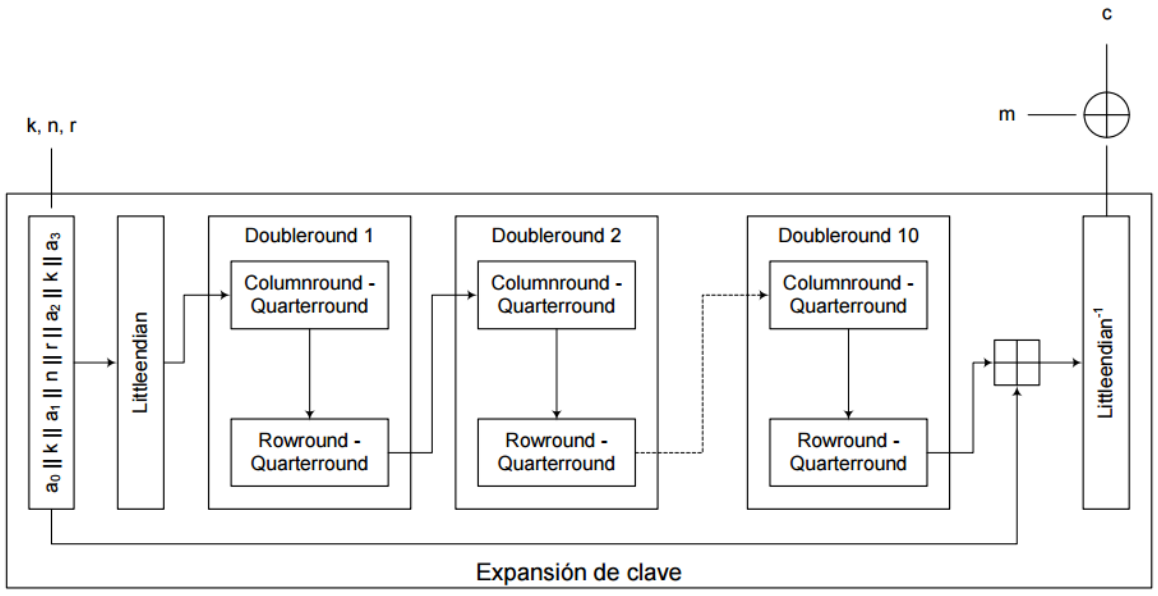


Ilustración 16. Secuencia de los datos en salsa20

Proceso de cifrado y descifrado

Anteriormente se enunciaron y explicaron matemáticamente las funciones usadas por el algoritmo además de una gráfica con la secuencia del flujo de datos en el proceso de cifrado, con la idea de dar una explicación más detallada en cada paso a continuación se muestran los pasos que realiza dicho algoritmo.

Fase 1

La primera fase consiste en extender el tamaño de la clave con la función hash aquí se toma como entrada 128 o 256 bits y se da como salida siempre 512 bits que es el tamaño con el que trabaja el algoritmo en sus demás procesos aquí se manejan los siguientes parámetros

K = clave de 128 o 256 bits

r = contador de 64 bits

n = nonce (cadena aleatoria de un solo uso)

-a_{0-3} = cuatro constantes de 32 bits cada una

Todas estas variables son intercaladas en un solo arreglo de 512 bits debido a que el tamaño de K puede variar este ordenamiento y los valores que toman algunas de las constantes (a) varia

	K=128 bits	K= 256bits
$\boldsymbol{a0}$	$\mathbf{0x65787061}$	$\mathbf{0x65787061}$
$\boldsymbol{a}1$	$\mathbf{0x6E642031}$	$\mathbf{0x6E64203}3$
$\boldsymbol{a}2$	$\mathbf{0x362D6279}$	$\mathbf{0x32}2\boldsymbol{D6279}$
$\boldsymbol{a}3$	$\mathbf{0x7465206B}$	$\mathbf{0x7465206B}$

Concatenación	$e = \boldsymbol{a0}\ \vert\boldsymbol{K}\vert\boldsymbol{a1}\ \vert\boldsymbol{N}\vert\boldsymbol{R}\vert\boldsymbol{a2}\ \vert\boldsymbol{K}\vert\boldsymbol{a3}$	$e = \boldsymbol{a0}\vert\boldsymbol{K0}\vert\boldsymbol{a1}\ \vert\boldsymbol{N}\vert\boldsymbol{R}\vert\boldsymbol{a2}\vert\boldsymbol{K1}\vert\boldsymbol{a3}$

Fase 2

Esta fase consiste en dividir la concatenación de 512 bits en 16 palabras de 32 bits cada una:

$e = \boldsymbol{w0}\ \vert\boldsymbol{w1}\ \vert\boldsymbol{w2}\vert\ldots\ \vert\boldsymbol{w}15\vert$

Donde cada *wi* está formada por 4 bytes;

$\boldsymbol{wi} = \boldsymbol{bi0}\vert\boldsymbol{bi1}\vert\boldsymbol{bi2}\vert\boldsymbol{bi}3$

Después de dividir la concatenación en palabras cada palabra es procesada de manera independiente y se le realiza una transposición inversa de bits usando la función Littleendian

Littleendian $(\boldsymbol{wi}) = \boldsymbol{bi0} + \boldsymbol{2\ 8bi1} + \boldsymbol{2\ 16bi2} + \boldsymbol{2\ 24bi3} \Rightarrow \boldsymbol{bi3}\vert\boldsymbol{bi2}\vert\boldsymbol{bi1}\vert\ \boldsymbol{bi0}$

Fase 3

El resultado de la función Littleendian entra a la función doubleround esta función da como resultado una cadena de 512 bits que son las 16 palabras de 32 bits estas palabras entran a 10 iteraciones de la función doubleround

$$(\boldsymbol{x0}, \boldsymbol{x1}, \boldsymbol{x2}, \boldsymbol{x15}) = \text{Doubleround}\boldsymbol{10}\ (\boldsymbol{w0'}, ', \boldsymbol{w2'}, \ldots, \boldsymbol{w'15})$$

Como se mencionó en el desglose de las funciones del algoritmo la función doubleround tiene 2 funciones internas (rowround, columnround) una aplicada a filas y la otra a columnas debido a que este una matriz 4x4

$$W' = \begin{bmatrix} w'_0 & w'_1 & w'_2 & w'_3 \\ w'_4 & w'_5 & w'_6 & w'_7 \\ w'_8 & w'_9 & w'_{10} & w'_{11} \\ w'_{12} & w'_{13} & w'_{14} & w'_{15} \end{bmatrix}$$

Ilustración 17. Matriz 4x4 doubleround

Estas dos funciones rowround y columnround a su vez aplican cada una otra operación más que es la Quaterround

El orden en que las operaciones son ejecutadas es el siguiente: primero se aplica la función columnround, la cual a su vez aplica la Quarterround, la función de esta es tomar las columnas de la matriz y convertirlas en la matriz U

$$(\boldsymbol{u0}, \boldsymbol{u4}, \boldsymbol{u8}, \boldsymbol{u12}) = \text{Quarterround } (\boldsymbol{w'0}, \boldsymbol{w'4}, \boldsymbol{w8'}, \boldsymbol{w'12})$$
$$(\boldsymbol{u1}, \boldsymbol{u5}, \boldsymbol{u9}, \boldsymbol{u13}) = \text{Quarterround } (\boldsymbol{w'1}, \boldsymbol{w'5}, \boldsymbol{w9'}, \boldsymbol{w'13})$$
$$(\boldsymbol{u2}, \boldsymbol{u6}, \boldsymbol{u10}, \boldsymbol{u14}) = \text{Quarterround } (\boldsymbol{w'2}, \boldsymbol{w'6}, \boldsymbol{w10'}, \boldsymbol{w'14})$$
$$(\boldsymbol{u3}, \boldsymbol{u7}, \boldsymbol{u11}, \boldsymbol{u15}) = \text{Quarterround } (\boldsymbol{w'3}, \boldsymbol{w'7}, \boldsymbol{w11'}, \boldsymbol{w'15})$$

$$U = \begin{bmatrix} U_0 & U_1 & U_2 & U_3 \\ U_4 & U_5 & U_6 & U_7 \\ U_8 & U_9 & U_{10} & U_{11} \\ U_{12} & U_{13} & U_{14} & U_{15} \end{bmatrix}$$

Ilustración 18. Matriz quarterround

Ahora se aplica la función rowround, la cual toma los renglones de U para también aplicarles Quarterround.

$$(\boldsymbol{u'0}, \boldsymbol{u'1}, \boldsymbol{u'2}, \boldsymbol{u'3}) = \text{Quarterround } (\boldsymbol{u0}, \boldsymbol{u1}, \boldsymbol{u2}, \boldsymbol{u3})$$
$$(\boldsymbol{u'4}, \boldsymbol{u'5}, \boldsymbol{u'6}, \boldsymbol{u'7}) = \text{Quarterround } (\boldsymbol{u4}, \boldsymbol{u5}, \boldsymbol{u6}, \boldsymbol{u7})$$
$$(\boldsymbol{u'8}, \boldsymbol{u'9}, \boldsymbol{u'10}, \boldsymbol{u'11}) = \text{Quarterround } (\boldsymbol{u8}, \boldsymbol{u9}, \boldsymbol{u10}, \boldsymbol{u11})$$
$$(\boldsymbol{u'12}, \boldsymbol{u'13}, \boldsymbol{u'14}, \boldsymbol{u'15}) = \text{Quarterround } (\boldsymbol{u12}, \boldsymbol{u13}, \boldsymbol{u14}, \boldsymbol{u15})$$

La función Quarterround es donde se realiza la mezcla de bits para crear el identificador único hash y se realiza las operaciones del XOR suma módulo 32 y corrimiento de bits

$$(\boldsymbol{u'0}, \boldsymbol{u'1}, \boldsymbol{u'2}, \boldsymbol{u'3}) = \text{Quarterround } (\boldsymbol{u0}, \boldsymbol{u1}, \boldsymbol{u2}, \boldsymbol{u3})$$
$$(\boldsymbol{u'4}, \boldsymbol{u'5}, \boldsymbol{u'6}, \boldsymbol{u'7}) = \text{Quarterround } (\boldsymbol{u4}, \boldsymbol{u5}, \boldsymbol{u6}, \boldsymbol{u7})$$
$$(\boldsymbol{u'8}, \boldsymbol{u'9}, \boldsymbol{u'10}, \boldsymbol{u'11}) = \text{Quarterround } (\boldsymbol{u8}, \boldsymbol{u9}, \boldsymbol{u10}, \boldsymbol{u11})$$
$$(\boldsymbol{u'12}, \boldsymbol{u'13}, \boldsymbol{u'14}, \boldsymbol{u'15}) = \text{Quarterround } (\boldsymbol{u12}, \boldsymbol{u13}, \boldsymbol{u14}, \boldsymbol{u15})$$

$$a' = a \oplus ((c' \boxplus d') \lll 18)$$
$$b' = b \oplus ((a \boxplus d) \lll 7)$$
$$c' = c \oplus ((a \boxplus b') \lll 9)$$
$$d' = d \oplus ((b' \boxplus c') \lll 13)$$

Y de esta manera finaliza en proceso de cifrado, debido a que la salsa 20 es un algoritmo de flujo el proceso de descifrado se realiza de la misma manera en sentido inverso, para revertirlo se aplica XOR entre en texto cifrado y la secuencia S que es la clave hash generada.

Para la implementación se utilizó la librería del usuario de Github alexwebr, almacenada en el repositorio[35] de la página web Github, esta librería es una implementación en c enfocada en la legibilidad y brevedad recoge la lógica de el algoritmo en 11 funciones, no provee expansión de llave, autenticación de mensajes y funcionalidad digest. La implementación no está diseñada para ser óptima, sin embargo se recogen algunos resultados en la descripción de la librería que agrupan velocidades en diferentes sistemas. Para su adaptación al IDE de arduino se debieron incluir las librerías stdint.h y stddef.h que agrupan un conjunto de definiciones de tipo usadas en el algoritmo, y que ayudan a delimitar la cantidad de bytes que debe asignar el programa a cada tipo de variable, se debió modificar en algunas funciones la especificación de longitud estática para variables de tipo array, ya que no era reconocido por el compilador. Las funciones se tienen en el maestro y esclavo, ya que los dos utilizan procesos de cifrado y descifrado, adicionalmente se creó otra función que sirve para encapsular el proceso, y que recibe dos archivos, uno de fuente y otro destino.

[35] "An implementation of the Salsa20 stream cipher in C99" 2015. 17 Feb de 2016 <https://github.com/alexwebr/salsa20>

5 ANALISIS

5.1 ANALISIS DE CIFRADO

Para realizar el análisis de cifrado se realizaron siete pruebas con distintos tamaños de datos como se ve en la tabla 1, cada prueba se realizó un total de diez veces y se sacó un promedio para medir la velocidad de cifrado y descifrado. Se realizó un análisis estadístico simple comprendido de desviación estándar muestral, rango, regresión cuadrática y el indicador definido para la prueba tamaño del archivo sobre velocidad total, disponible en el anexo Plan de pruebas.

Tabla 1 cifrado tamaño y tiempo

Cantidad de bytes(KB)	Tiempo(s)	Desviación estándar muestral(s)
4.34	0.0840232	0.006908
19	17.30807431	0.012704
49.6	44.89599224	0.047735
100	91.36240154	0.291047
150	139.0434738	0.052654
300	283.1738403	0.162231
500	469.8419372	0.268859

En general se nota un incremento en la desviación estándar al aumentar la cantidad de datos, con algunos picos significativos en las pruebas de 100 y de 500 KB, sin embargo, en ninguno de los casos la desviación sobrepasa 1 segundo, lo que demuestra la estabilidad y uniformidad del dispositivo para realizar cálculos aritméticos.

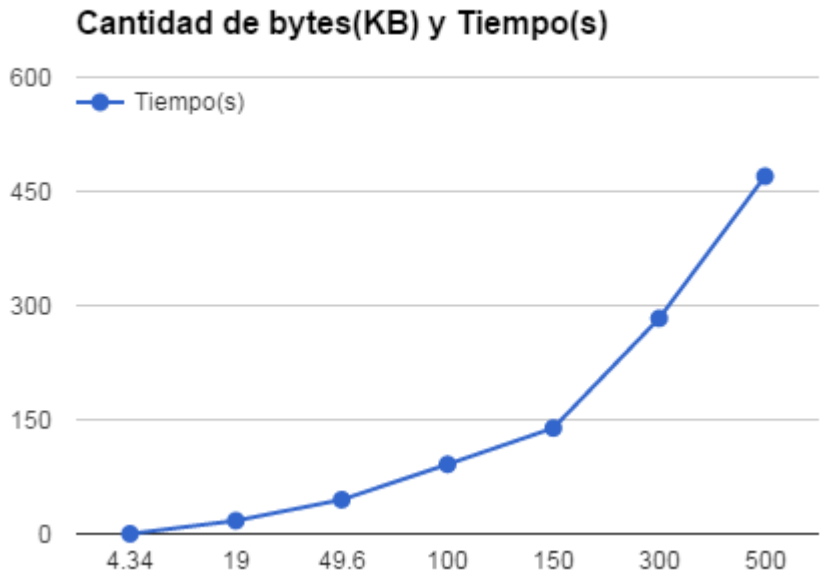


Ilustración 19 Cantidad de bytes vs tiempo

Cuando se representa gráficamente el comportamiento de los datos (Ilustración 19) se observa un crecimiento curvilíneo del tiempo que tarda el dispositivo en cifrar y descifrar una cantidad determinada de datos, esto se debe al sobrecosto que representa para el dispositivo manejar un volumen de datos grande, debido a su limitada memoria y a que tiene que hacer uso del módulo SD, y que cada acceso e instrucción representa un costo general. Se observó que el dispositivo arduino presenta inestabilidad cuando se cargan más de 450 bytes simultáneamente en un arreglo. Debido a que la función Salsa20 tiene una entrada de 64 bytes y una salida de 64 bytes, se optó por cargar 64 bytes de la memoria SD.

Se realizó una regresión cuadrática para modelar el comportamiento de los datos mediante una ecuación (Ilustración 20), se puede afirmar que este comportamiento cuadrático es el que mejor define la evolución de los datos, y que para archivos de datos más grandes seguirá esta tendencia, generando tiempos que no son óptimos para el uso en tiempo real.

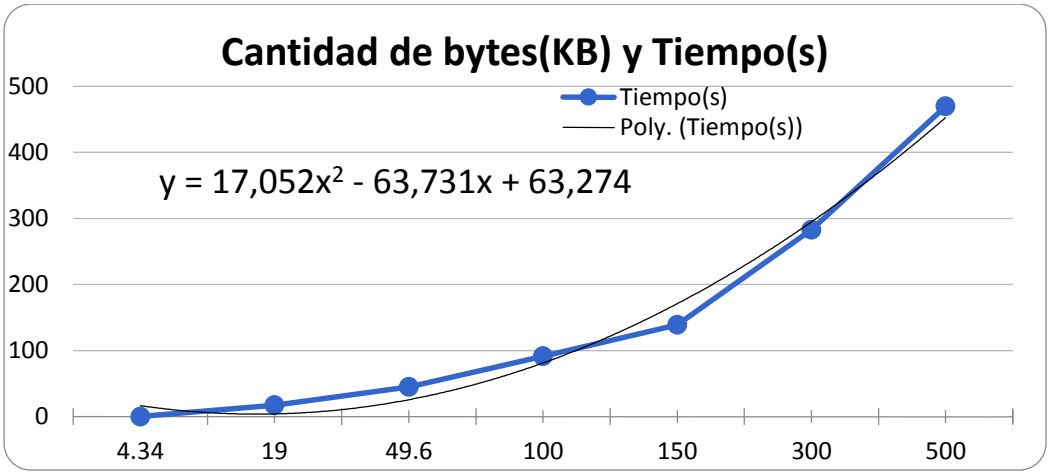


Ilustración 20 Regresión cuadrática

Se observa en la Ilustración 21 un gráfico comparativo del comportamiento del tamaño de los datos y el tiempo de cifrado, además se representa la desviación estándar que muestra el pico ocasionado por el incremento en el tamaño de los datos, aunque se muestre este pico en la desviación es un valor muy pequeño por ende el comportamiento del arduino es consistente en los tamaños de los datos en las diferentes pruebas.

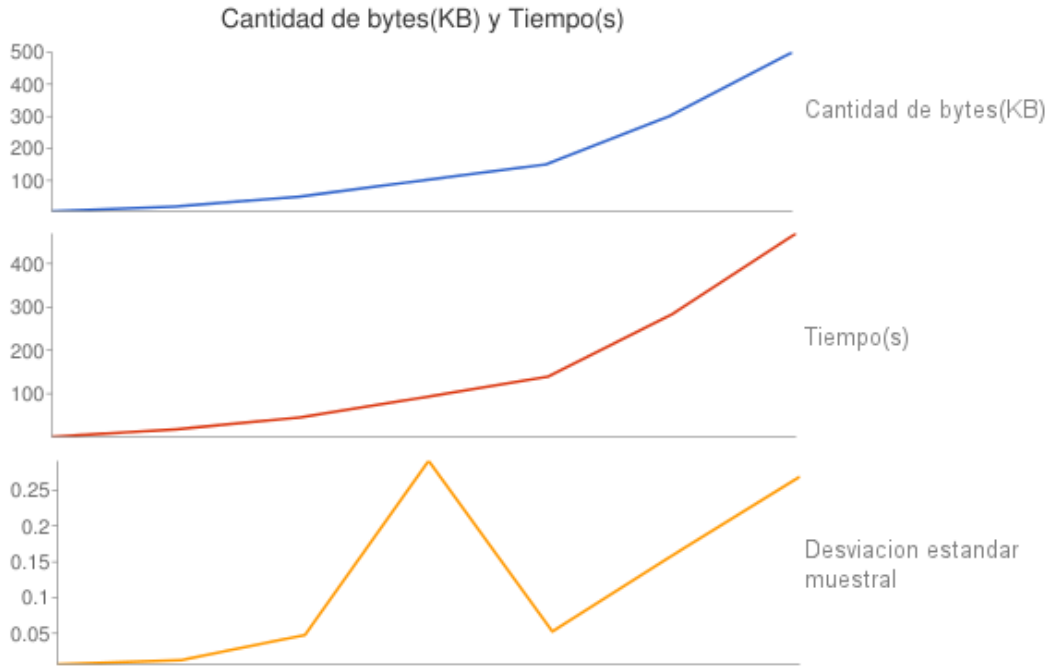


Ilustración 21 comparación tiempo, tamaño y desviación

5.1.1 SOBRECOSTO MEMORIA SD

Se realizó un análisis del sobrecosto que produce en el sistema escribir y leer de la memoria SD en el proceso de cifrado y descifrado mediante Salsa20. Se llevaron a cabo dos tipos de pruebas, la primera con una cadena de caracteres almacenada en la memoria RAM del dispositivo arduino y la segunda con archivos de texto con la misma cantidad de caracteres en la tarjeta SD. Debido a que la plataforma utiliza 2KB de memoria para las variables globales, y una parte se requiere para la ejecución del algoritmo Salsa20 y el funcionamiento general, la cantidad de caracteres máxima que se permitió almacenar en el dispositivo fue de 450 bytes. Los resultados de las pruebas se observan en la Tabla 2.

Tabla 2 comparación SD y RAM

Cantidad de bytes(B)	Sin memoria SD-Cifrado y Descifrado	Con memoria SD-Cifrado y Descifrado
	Tiempo(s)	Tiempo(s)
100	0.0245212	0.1577156
200	0.0488936	0.2707424
300	0.061374	0.3222
400	0.085672	0.4384568
450	0.097812	0.55190202

El análisis gráfico de los datos se observa en la Ilustración 22. En promedio, la proporción de tiempo extra que toma el dispositivo para realizar la encriptación es de 5.59 veces, esto se debe a que la librería y el dispositivo SD generan un consumo de recursos del dispositivo porque cada 64 bytes se debe leer y escribir información en archivos diferentes. Una alternativa a esto, es dependiendo del medio de transmisión que se quiera usar y un análisis de diseño, utilizar módulos especializados que incluyan capacidad de almacenamiento con mayor velocidad de transferencia y de lectura/escritura, lo que permitirá optimizar los recursos del Arduino, un ejemplo y opción viable

son los módulos Ethernet[36] el cual soporta una conexión RJ-45, viene con un slot para la tarjeta SD y soporta hasta 4 sockets simultáneamente, otra opción a considerar son los módulos bluetooth los cuales transmiten a velocidades muy altas en un rango útil para aplicaciones locales, estos también dejan libre al arduino de la lógica de transmisión y almacenamiento de información.

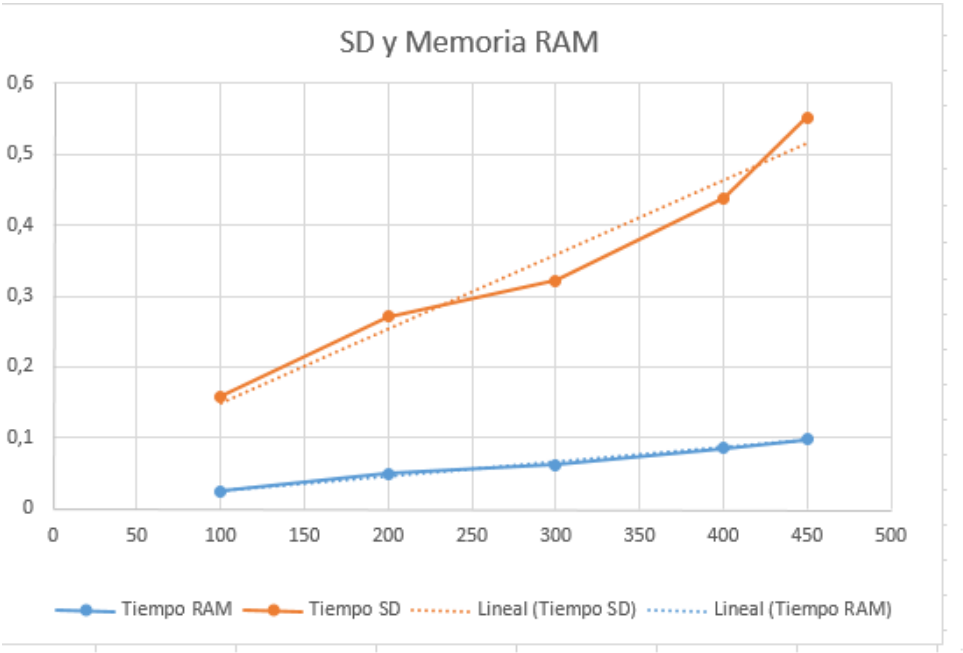


Ilustración 22 comparación SD y RAM

5.2 ANALISIS DE TRANSMISION

Para realizar las pruebas de transmisión se conectaron los dos dispositivos arduino por comunicación serial I2C, se realizaron un total de 4 pruebas de un total de 7 debido a que la velocidad de transmisión era muy baja. Los resultados de las pruebas contenidos en la Tabla 2 muestran los valores obtenidos para cada prueba. Se realizó un análisis estadístico simple comprendido de desviación estándar muestral, rango, regresión cuadrática. Los datos completos de las tablas son incluidos en los anexos.

[36] "Ethernet shield" 17 Feb 2016. <https://www.arduino.cc/en/Main/ArduinoEthernetShield>

Tabla 3 transmisión bytes y tiempo

Cantidad de bytes(KB)	Tiempo(s)	Desviación estándar muestral
5	97,93329309	0,182520
19,5	342,6533722	16,777466
50	900,6657653	3,827713
100	1953,723083	246,552281

Al representar los datos se observa que a medida que el tamaño de los datos aumenta, el tiempo de transmisión se eleva, esto se nota especialmente en el archivo de 100kb donde se presenta un aumento de tiempo realmente considerable. Esto ocurre debido a la sincronización de los dispositivos, el dispositivo maestro envía la estructura a una velocidad muy alta, es necesario que el esclavo restrinja la velocidad mediante una variable que le indica si puede o no enviar. Esto ocasiona retrasos en la comunicación debido a las esperas que tiene que ejecutar el maestro. También se adiciona un sobrecosto debido a las escrituras y lecturas en la SD.

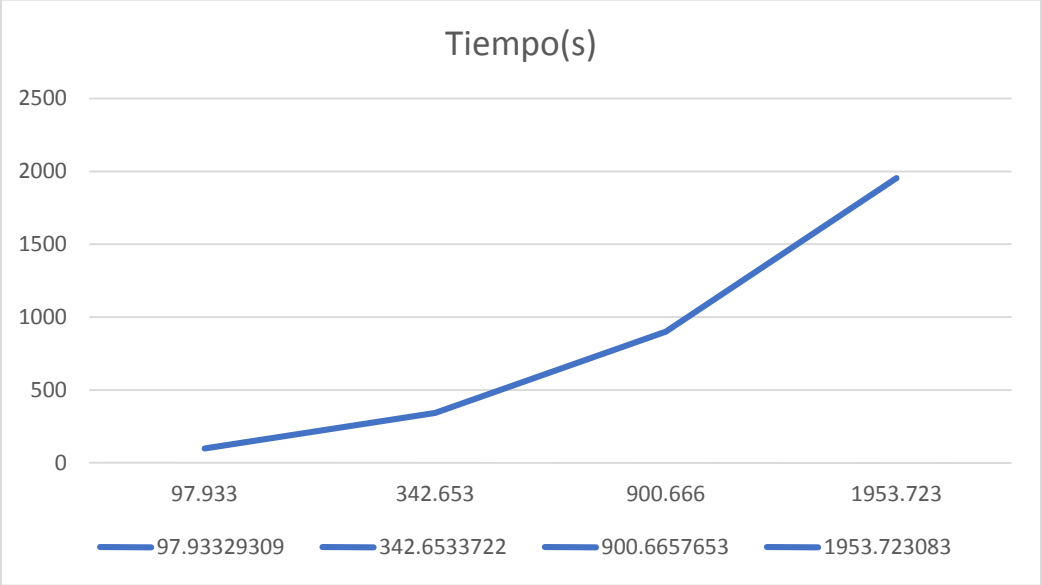


Ilustración 23 grafica de transmisión bytes y tiempo

Al agregar una línea de tendencia se observa una ecuación cuadrática, se puede inferir que a mayor tamaño del archivo mayor serán los tiempos de transmisión, lo que vuelve poco práctico el medio y método de transmisión.

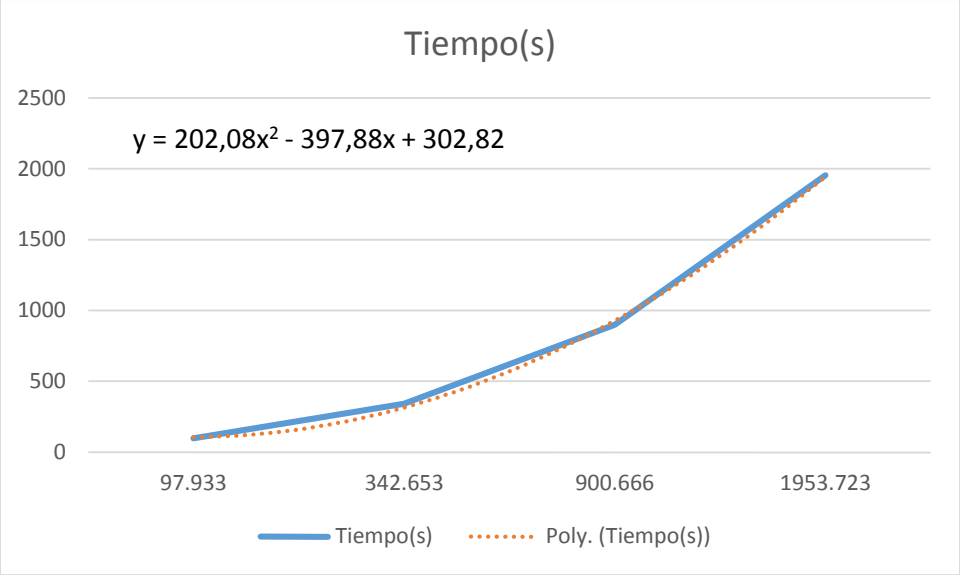


Ilustración 24 transmisión línea tendencia y ecuación

En la ilustración 24 se observa la comparación de tiempo, tamaño y desviación muestral. Se observa un cambio marcado en la desviación muestral al final debido a que comparativamente los tiempos de los archivos grandes y pequeños difieren significativamente en duración.

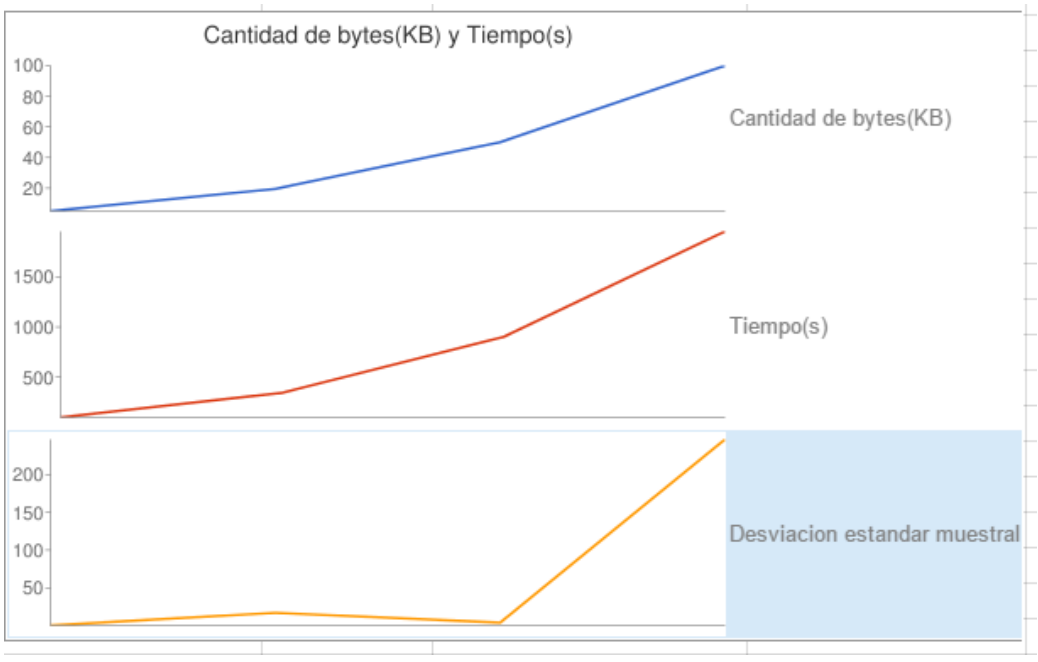


Ilustración 25 comparación tamaño tiempo y desviación

6 CONCLUSIONES Y RECOMENDACIONES

El dispositivo Arduino uno con sus limitadas capacidades de memoria y procesamiento se desempeñó bien en algunos de los procesos realizados en la implementación como la captura de información mediante el micrófono y cifrado de estos datos. Mediante el análisis de las pruebas de cifrado y descifrado se observó que para archivos de tamaño mayor a 300 KB, el tiempo puede volverse un inconveniente si se considera que aspectos como la velocidad son necesarios si se quiere un sistema que opere bajo condiciones de tiempo real, sin embargo mediante optimización del algoritmo y mejor manejo de la memoria del dispositivo, es posible mejorar el tiempo para tamaños de información no mayores a 500KB para los cuales se considera que es posible encontrar alternativas, ya sea en la línea de productos arduino, u otros dispositivos con mayor velocidad de procesador, almacenamiento cache y RAM.

Con respecto a la transmisión, el método implementado probó ser muy poco eficiente con respecto a velocidad, alcanzando velocidades de apenas 50 Bytes/s, esto se debió a la necesidad de la sincronización mediante un pin digital en el cual el esclavo indica al maestro cuando se había leído y copiado correctamente a la memoria y podía copiar los nuevos datos en la estructura.

Adicionalmente la necesidad de la memoria SD para ampliar la limitada capacidad de almacenamiento del dispositivo arduino imprimió una latencia en todos los procesos de por lo menos 5 veces el tiempo regular. Sin embargo la confiabilidad y relación entre paquetes enviados y paquetes recibidos fue de 100%, debido a lo riguroso y costoso en tiempo del método implementado.

Es evidente la necesidad del uso de un protocolo soportado y regulado por una norma internacional, que pueda ser adaptado eficientemente a dispositivos de capacidades limitadas como microcontroladores y operar sobre los pines digitales o análogos de dichos dispositivos.

En general el prototipo cumplió su funcionalidad: capturar sonido, cifrarlo enviarlo a través de un medio serial y descifrarlo en el otro extremo, con un desempeño aceptable, pero no puede considerarse óptimo para aplicaciones en tiempo real.

Sin embargo, este prototipo deja abiertas posibilidades para la adaptación de otros métodos, módulos e implementaciones que permitirían en el futuro crear un sistema robusto y veloz de acuerdo a las capacidades de los microcontroladores.

Como recomendación se sugiere la adición de un módulo Ethernet para próximos prototipos/desarrollos y la implementación del código necesario para usar este método de transmisión, ya que siendo un sistema de cifrado enfocado a la seguridad de información, este medio de transporte de datos es uno de los más idóneos para obtener el mayor uso en una red.

La optimización del código Salsa20 y la paralelización de tareas usando otros dispositivos arduino, o controladores adicionales podrían ayudar a mejorar los tiempos de cifrado y descifrado del dispositivo, permitiendo un acercamiento más realista a la meta de tiempo real, de igual manera es esencial buscar otros métodos de almacenamiento que no adicionen tanto sobrecosto a las operaciones que debe realizar el dispositivo para archivos mayores a 500 KB, ya sea diferentes a las tarjetas SD o tarjetas SD con mayor velocidad de lectura y escritura.

7 BIBLIOGRAFIA

Basanta-Val, P., & García-Valls, M. (2015). A library for developing real-time and embedded applications in C. *Journal of Systems Architecture*, *61*(5-6), 239–255. http://doi.org/10.1016/j.sysarc.2015.03.003

Candelas, F. A., García, G. J., Puente, S., Pomares, J., Jara, C. A., Pérez, J., … Torres, F. (2015). Experiences on using Arduino for laboratory experiments of Automatic Control and Robotics. *IFAC-PapersOnLine*, *48*(29), 105–110. http://doi.org/10.1016/j.ifacol.2015.11.221

Huo, F., & Gong, G. (2015). XOR Encryption Versus Phase Encryption, an In-Depth Analysis. *IEEE Transactions on Electromagnetic Compatibility*, *57*(4), 1–9. http://doi.org/10.1109/TEMC.2015.2390229

Kong, J. H., Ang, L.-M., & Seng, K. P. (2015). A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments. *Journal of Network and Computer Applications*, *49*, 15–50. http://doi.org/10.1016/j.jnca.2014.09.006

Mazumdar, B., Ali, S. S., & Sinanoglu, O. (2015). Power analysis attacks on ARX: An application to Salsa20. In *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)* (pp. 40–43). IEEE. http://doi.org/10.1109/IOLTS.2015.7229828

Mostefaoui, A., Noura, H., & Fawaz, Z. (2015). An integrated multimedia data reduction and content confidentiality approach for limited networked devices. *Ad Hoc Networks*, *32*, 81–97. http://doi.org/10.1016/j.adhoc.2015.01.007

Murillo-Escobar, M. A., Cruz-Hernández, C., Abundiz-Pérez, F., & López-Gutiérrez, R. M. (2015). A robust embedded biometric authentication system based on fingerprint and chaotic encryption. *Expert Systems with Applications*, *42*(21), 8198–8211. http://doi.org/10.1016/j.eswa.2015.06.035

Putra, L., & Kanigoro, B. (2015). Design and Implementation of Web Based Home Electrical Appliance Monitoring, Diagnosing, and Controlling System. *Procedia Computer Science*, *59*, 34–44. http://doi.org/10.1016/j.procs.2015.07.335

Watve, O. J. (2015). Implementation of real time factory information system using arduino and android, 1343–1347.

Zapateiro, M., Hoz, D., Acho, L., & Vidal, Y. (2015). An Experimental Realization of a Chaos-Based Secure Communication Using Arduino Microcontrollers - ProQuest. Retrieved January 18, 2016, from http://search.proquest.com.ezproxy.utp.edu.co/docview/1709690349/8AE3B4C0120C483APQ/2?ac

countid=45809

Web

* 98% de empresas colombianas son víctimas de ataques ..." {En línea}. Fecha [11 de Marzo de 2015][Disponible en:]http://www.vanguardia.com/actualidad/colombia/250540-98-de-empresas-colombianas-son-victimas-de-ataques-informaticos

* "Secret US cybersecurity report: encryption vital to protect ..." {En línea}. Fecha [11 de Marzo de 2015][Disponible en:]<http://www.theguardian.com/us-news/2015/jan/15/-sp-secret-us-cybersecurity-report-encryption-protect-data-cameron-paris-attacks>

* "CRYPTR 2 Broadband IP Encryption Unit - Motorola Solutions." {En línea}. Fecha [11 de Marzo 2015][Disponibleen:]<http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Project+25+(P25)+Systems/Encryption/CryptR>

* Daemen, J. "The Design of Rijndael - Joan Daemen." {En línea}. Fecha [11 de Marzo de 2015]. [Disponible en:]<http://jda.noekeon.org/JDA_VRI_Rijndael_2002.pdf>

* Ball,James.Secret US cybersecurity report: encryption vital to protect private data.{En línea}.16 de enero del 2015.{4 de marzo del 2015}. Disponible en: http://www.theguardian.com/us-news/2015/jan/15/-sp-secret-us-cybersecurity-report-encryption-protect-data-cameron-paris-attacks

* "What is QoS (Quality of Service)?" {En línea}. Fecha [24 de Abril 2015] [Disponible en:]<http://searchunifiedcommunications.techtarget.com/definition/QoS-Quality-of-Service>

* "Electronics - History - Science Encyclopedia - JRank." {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://science.jrank.org/pages/2376/Electronics-History.html>

* "Open Source Initiative."{En línea}.Fecha [24 Abril 2015][Disponible en:]<http://opensource.org/>

* "What is encryption?."{En línea}.Fecha [24 Abril 2015][Disponible en:]<http://searchsecurity.techtarget.com/definition/encryption>

* "What is asymmetric cryptography" {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://searchsecurity.techtarget.com/definition/asymmetric-cryptography>

* "What is secret key algorithm (symmetric algorithm)" {En línea}. Fecha [24 Abril 2015][Disponible en:]<http://searchsecurity.techtarget.com/definition/secret-key-algorithm>

* "What is block cipher?" {En línea}. Fecha [24 de Abril 2015] [Disponible en:]<http://searchsecurity.techtarget.com/definition/block-cipher>

* "What is stream cipher?" {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://searchsecurity.techtarget.com/definition/stream-cipher>

* "Datasheet - Atmel Corporation." {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf>

* Bernstein, DJ. "Salsa20 specification." {En línea}.Fecha [24 de Abril 2015][Disponible en:] <http://cr.yp.to/snuffle/spec.pdf>

* "CTR Mode" {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://www.cryptopp.com/wiki/CTR_Mode>

* "D. J. Bernstein." {En línea}.Fecha [24 de Abril 2015][Disponible en:]<http://cr.yp.to/djb.html>

* "Hash Functions"{En línea}. Fecha [24 de Abril 2015][Disponible en:]<http://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html>

* “What is an ADC” {En línea}. Fecha [29 de Enero 2016] [Disponible en:]<http://www.microsmart.co.za/technical/2014/03/01/advanced-arduino-adc>

7. ANEXOS

7.1 Manual de Usuario

7.2 Manual Técnico

7.3 Plan de Pruebas