FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD Y MONITOREO DE UN VEHÍCULO

AUTOR

JAVIER ALBERTO VILLAGÓMEZ ÁLVAREZ

AÑO

2019

FACULTAD DE INGENIERÍAS Y CIENCIAS APLICADAS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD Y MONITOREO DE UN VEHÍCULO

"Trabajo De Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Redes y Telecomunicaciones"

Profesor Guía
Ph.D. Nathaly Verónica Orozco Garzón

Autor
Javier Alberto Villagómez Álvarez

Año
2019

DECLARACIÓN PROFESOR GUÍA

Declaro haber dirigido el trabajo Diseño e implementación de un sistema de seguridad y monitoreo de un vehículo. A través de reuniones periódicas con el estudiante Javier Alberto Villagómez Álvarez, en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

_____

Nathaly Verónica Orozco Garzón
Doctora en Ingeniería Eléctrica en el Área de Telecomunicaciones y Telemática
CI. 1720938586

DECLARACIÓN PROFESOR CORRECTOR

Declaro haber revisado este trabajo Diseño e implementación de un sistema de seguridad y monitoreo de un vehículo, de Javier Alberto Villagómez Álvarez, en el semestre 201920, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

_____

Henry Ramiro Carvajal Mora
Doctor en Ingeniería Eléctrica en el Área de Telecomunicaciones y Telemática
CI. 1721327862

DECLARACIÓN DE AUDITORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

_____

Javier Alberto Villagómez Álvarez

CI. 1711284164

AGRADECIMIENTOS

Agradezco a mi papá y a mi mamá por su amor y apoyo incondicional. A mi hermano Fernando por su apoyo constante. A mi novia por su apoyo incondicional, fuerza y amor. Agradezco a mi tutora Nathaly por su apoyo y orientación brindados.

DEDICATORIA

Este trabajo está dedicado a mi familia, especialmente a mi papá y a mi novia Giovanna. Quienes con su apoyo y amor han hecho posible que culmine mi carrera a pesar de todos los obstáculos.

RESUMEN

Los índices de inseguridad en el Ecuador se han incrementado considerablemente según reportes presentados en (Bravo, 2018). Los delincuentes están usando controles de alarmas universales que encienden en zonas donde hay varios vehículos y se roban los que se activan. En estos reportes, desde enero hasta junio del 2017 se calcularon 640 robos de automotores en la capital. En el mismo período del 2018 fueron 677 robos. A escala nacional fueron 2 239 carros sustraídos en el 2017 y en el 2018 fueron 2 184.

En el presente proyecto de titulación se diseña e implementa un sistema de seguridad para monitorear un vehículo mientras el propietario no está. La implementación se basa en un modelo a escala. El diseño de este prototipo fue optimizado para lograr una solución eficaz e instantánea de adquisición de imágenes del perímetro del auto desde cámaras integradas, para luego ser enviadas a una aplicación móvil en el teléfono celular del propietario del vehículo alertando de un incidente en su auto.

El diseño del prototipo consta de tres partes: modulo vehicular, servidor de servicios y aplicación móvil. El módulo vehicular se encarga de captar imágenes del perímetro del auto, al accionarse un acelerómetro, para luego enviarlas a un servidor a través de un módem de cuarta generación (4G). El servidor de servicios se encarga de recibir esas imágenes y enviarlas simultáneamente a la aplicación móvil y a un correo de emergencia para alertar al propietario que el sistema ha sido activado y tenga evidencia del perímetro del vehículo. Finalmente, la aplicación móvil se encarga de controlar la activación del sistema, configurar el contacto de emergencia y recibir las imágenes enviadas por el servidor.

El prototipo implementado logra un tiempo de respuesta de 35 segundos desde que se activa el sistema hasta que el propietario recibe las fotos en el teléfono y llega el correo electrónico con las imágenes. Este tiempo de respuesta es

aceptable para un prototipo. Sin embargo, el sistema podría ser mejorado aún más siguiendo las recomendaciones indicadas al final de este trabajo. Finalmente, el sistema podría incorporar nuevas prestaciones en futuras iteraciones.

ABSTRACT

Crime rate in Ecuador has increased considerably according to reports from (Bravo, 2018). Criminals are using universal car remotes to trigger alarms in an area and steal the cars activated. In these reports, from January to June 2017, 640 car robberies were calculated in Quito. In the same period of 2018, there were 677 robberies. On a national scale, there were 2239 cars stolen in 2017 and 2184 in 2018.

In this final project, a security system is designed and created to monitor a vehicle while the owner is not present. The prototype is based on a scale model. The design of this prototype was optimized to achieve an effective and real-time solution of image acquisition from the perimeter of the car through integrated cameras, to then be sent to a mobile application on the cell phone of the car owner alerting of an incident with his car.

The design of the prototype consists mainly of three components: vehicle module, backend server and mobile app. The vehicle module is in charge of taking photos of the perimeter of the car when the accelerometer is triggered, and then send them to a server through a fourth generation (4G) módem. The backend server is responsible for receiving these images and sending them simultaneously to the mobile app and an emergency e-mail alerting the owner that the system has been triggered and has evidence of the perimeter of the vehicle. Finally, the mobile application is responsible for controlling the activation of the system, configuring the emergency contact and receiving the images sent by the server.

The prototype achieved a response time of 35 seconds starting at the triggering of the accelerometer and finishing with the owner receiving the potos in his cellphone and the mail with the potos arrives. This response time is acceptable for a prototype. However, the system could be upgraded further by following the recommendations at the end of this paper. Finally, the system could incorporate new features in future iterations.

ÍNDICE

1. Introducción

1.1 Antecedentes

Los índices de inseguridad en el Ecuador se han incrementado considerablemente según reportes presentados en (Bravo, 2018). Los delincuentes están usando controles de alarmas universales que encienden varios vehículos de forma simultánea y se roban los que se activan. En estos reportes, desde enero hasta junio del 2017 se calcularon 640 robos de automotores en la capital. En el mismo período del 2018 fueron 677 robos. A escala nacional fueron 2 239 carros sustraídos en el 2017 y en el 2018 fueron 2 184.

De acuerdo a una campaña en (Guayaquil, 2007) que recoge 100 casos en la ciudad de Guayaquil a lo largo de 20 días de mayo del 2007, 6 casos incluyen autos estacionados. Esto solo representa accidentes denunciados, muchos de estos casos no son denunciados porque la persona solo se entera que le ha pasado algo a su carro cuando sale del lugar en el que está y nadie sabe que pasó. De la misma manera, en (Agencia Nacional de Tránsito, 2018) se reportaron 31 accidentes de tránsito a nivel nacional por vehículos mal estacionados.

Debido a estos antecedentes, los sistemas de seguridad en vehículos han ido evolucionando considerablemente, tenemos desde alarmas, sistemas de bloqueo o incluso sistemas de rastreo satelital. Sin embargo, la inseguridad ha ido creciendo alarmantemente, por este motivo, los dueños de los vehículos buscan nuevas formas de proteger su inversión. Así también, en la actualidad se desarrollan varias aplicaciones móviles que solventan parcialmente esta problemática. Por ejemplo, rastreo de vehículos usando el sistema de posicionamiento global (GPS). No obstante, estos dispositivos son muy costosos en algunos casos. En este sentido, a las personas les gustaría poder vigilar constantemente su vehículo en todo momento de manera que el costo del sistema no sea elevado.

Por esta razón, en el presente trabajo se propone un sistema de seguridad y monitoreo para vehículos que puede ser implementado paralelamente con los

sistemas existentes. Este sistema consiste en alertar al dueño del auto a través de una aplicación móvil y un sistema de cámaras, las cuales se implementan en el vehículo. A través de este sistema se enviarán fotos del perímetro y habitáculo del vehículo al celular una vez que se detecte un choque o si se activa la alarma. Esta propuesta es novedosa ya que el usuario puede observar en tiempo real lo que le sucede con su vehículo.

1.2 Justificación

En la literatura existen proyectos que utilizan tecnologías similares a la propuesta como aquella presentada en (Acosta & Herrera, 2009), que utiliza radio bases celulares y tecnología Global System for Mobile (GSM) de segunda generación (2G) para localizar el auto y apagar el motor, en (Reyes Huertas & Huertas Vitery, 2014) se presenta un enfoque similar al anterior pero también abre y cierra seguros de puertas. Por último, en (Chamorro, 2013) el trabajo es similar a los mencionados con el adicional de que prende o apaga el vehículo usando una secuencia de marcación por tonos de números del teléfono. Estos proyectos proveen soluciones de ubicación, control de seguros, control de encendido, etc. Sin embargo, ninguno de los proyectos mencionados muestra los alrededores del vehículo en tiempo real, por lo que, si el carro es dañado mientras el dueño no está, el mismo no se enterará que sucedió hasta que regrese.

Los sistemas de seguridad existentes proveen seguridad a los vehículos, pero al ser violentados, estos sistemas no notifican al dueño del automotor por lo que este solo se entera del daño cuando regresa al auto. Consecuentemente, estas propuestas no son tan eficientes cuando de tiempo real se habla. La solución propuesta en este trabajo está pensada para que los dueños de los vehículos sepan en tiempo real si algo le pasa a su auto, incluso con fotos como evidencia de lo ocurrido. P esto permitiría incluso poder tomar medidas judiciales de ser posible y resguardar el automotor de mejor manera. Este proyecto implica una mejora a la seguridad y calidad de vida de los dueños de vehículos. Además, provee un avance tecnológico contribuyendo con la mejora constante en sistemas de seguridad vehicular existentes.

1.3 Objetivos

- **Objetivo General**

Diseñar e implementar un sistema de seguridad y monitoreo de un vehículo usando un módulo instalado en el vehículo que incluya una plataforma de microcontrolador y un complemento 4G.

- **Objetivos específicos**

  - Describir los conceptos técnicos básicos que se usarán en el transcurso de proyecto.
  - Diseñar el prototipo empleando los módulos de microcontrolador, 4G y cámaras.
  - Diseñar la aplicación móvil empleando un lenguaje de programación para Android.
  - Implementar la aplicación móvil junto con el módulo vehicular, incorporarlo sobre el ambiente de pruebas y realizar las conexiones necesarias para su funcionamiento.
  - Analizar las tecnologías de seguridad vehicular existentes y compararlas con el prototipo propuesto.
  - Analizar los resultados obtenidos de la implementación del sistema.

2. Marco Teórico

2.1 Sistemas de Seguridad Vehicular

Los sistemas de seguridad vehicular se definen como mecanismos de resguardo de un vehículo contra robos o daños mediante la disuasión, alerta al dueño o monitoreo constante. Existen varios sistemas alrededor del mundo, los más usados en Ecuador son: sistema de rastreo de vehículo por GPS, bastón traba volante / traba palanca, Goodlock, corta corriente, llaves codificadas, alarmas clásicas, láminas de seguridad en las ventanas y bloqueador de capó (Lucio, 2016).

El sistema de rastreo por GPS utiliza un módulo en el vehículo que envía la información de ubicación del mismo a un servidor de la empresa que provee el servicio usando la red celular. Con esto informar en tiempo real el paradero del

vehículo a la empresa proveedora y esta informa al propietario. El bastón traba volante o traba palanca es un candado con llave o combinación que inmoviliza la palanca de cambios o el volante para que el auto no pueda ser robado. Goodlock es un sistema que se instala en el vehículo que obliga a realizar una serie de acciones específicas para poder prender el vehículo, estas pueden ser: girar el volante, accionar las direccionales, prender el radio, aplastar el freno, etc. El módulo corta corriente es un sistema que permite cortar el sistema eléctrico del auto para que se detenga desde una aplicación móvil instalada en el celular del dueño. Las llaves codificadas son dispositivos que poseen un código único de verificación que el vehículo comprueba para encender el motor, accionar los seguros, activar o desactivar la alarma, abrir el baúl, etc. Esto se lo hace a través de ondas de radio de corto alcance o *NFC* (por sus siglas en inglés *Near Field Communication)* y antenas pasivas en los controles remotos de los autos. Las alarmas clásicas son dispositivos disuasivos que se instalan en el auto conectadas a las puertas y tienen sensores de movimiento que, al activarse, emiten una alerta de sonido para avisar que se ha abierto una puerta o el auto ha sido golpeado. Las láminas de seguridad en las ventanas son un recubrimiento plástico que se pone en los vidrios del auto para evitar que, en caso de un siniestro o un asalto, los pedazos de vidrio no caigan sobre los ocupantes del auto provocando heridas. El bloqueador de capó es un candado que asegura el capó del auto contra ladrones que quieran robar el auto (Lucio, 2016). Estos sistemas se los analizará en comparación con la solución propuesta en este proyecto más adelante.

Además, se han desarrollado sistemas que combinan el rastreo por GPS, control de encendido, control de alarma, control de seguros desde aplicaciones en el teléfono celular. Todos estos sistemas están pensados para evitar robos parciales o totales de los vehículos. Sin embargo, cuando un auto está parqueado, el mismo está expuesto a robos, golpes de vehículos cercanos y choques por parte de conductores irresponsables.

Actualmente, los dueños de los vehículos solo pueden depender de un seguro contra accidentes y robos, una alarma y ciertas precauciones al momento de parquear su vehículo para evitar que le pase algo. Por esta razón, se ha

propuesto una solución que complemente los sistemas existentes haciendo uso de tecnologías accesibles al público como son la red celular, los teléfonos celulares inteligentes y el internet que se elaborará en los siguientes capítulos del presente proyecto.

2.2 Internet de las Cosas

El internet de las cosas (IoT, del inglés Internet of Things) es el principio de la interconexión de cualquier dispositivo a un interruptor de encendido y apagado en el internet o a otro dispositivo. Esto incluye celulares, refrigeradoras, aires acondicionados, puertas de garaje, lámparas, lavadoras de ropa, cafeteras, etc. El IoT es una red de dispositivos conectados que puede incluir personas usando telecomunicaciones y permite una interacción dinámica entre ellas (Morgan, 2014).

Un dispositivo IoT tiene cuatro componentes principales: objetivo, comunicación, manipulación de datos y desarrollo. El “objetivo” es en donde se va a usar el dispositivo y está relacionado con la implementación del *hardware*. La “comunicación” es como se transmitirá la información recibida desde el dispositivo IoT. La “manipulación de datos” considera toda la información que el dispositivo IoT genera o recibe de múltiples sensores conectados como acelerómetros, sensores de temperatura, sensores ultrasónicos, sensores de movimiento, sensores infrarrojos, sensores de nivel de líquidos, sensores de humedad, sensores de humo, entre otros. Finalmente, el componente de desarrollo analiza las características y funcionalidades que deben ser programadas en el dispositivo para su funcionamiento (Caivano, Cassano, Lanzilotti, & Piccinno, 2018).

Estas aristas de los dispositivos IoT implican consideraciones sobre que equipos se usará como microcomputadoras, microcontroladores, sensores, módulos de comunicación, etc. Si se usará una red celular de cuarta generación (4G), o si se usará una conexión cableada de internet, o *Wireless Fidelity* (Wi-Fi), o *bluetooth, o NFC*, o identificación por radiofrecuencia (por sus siglas en inglés RFID).

2.3 Tecnologías Móviles

“Cuando vemos la evolución de la red de comunicación móvil, la G de generación inalámbrica móvil generalmente nos indica que ha habido un cambio en la naturaleza del sistema, la velocidad, la tecnología y la frecuencia. Cada generación tiene algunos estándares, capacidades técnicas y nuevas características que la diferencian de la anterior” (Universidad Internacional de Valencia, 2018).

2.3.1 Tecnologías móviles 1G a 3G

La primera generación (1G) de comunicaciones móviles usaba Sistema Avanzado de Teléfono Móvil (AMPS por sus siglas en inglés) como estándar analógico, con velocidades de hasta 2,4 kbps, en las frecuencias de 800-900 MHz y usando acceso múltiple por división de frecuencia (FDMA). La segunda generación (2G) tenía como estándares digitales GSM y CDMA (por sus siglas en inglés *Code-Division Multiple Access*), con velocidades de hasta 64 kbps, en las frecuencias 850-1900 MHz (GSM) y 825-849 MHz (CDMA), usando TDMA (Acceso Múltiple por División de Tiempo), FDMA y CDMA como métodos de acceso múltiple, respectivamente. Además, la segunda generación tuvo un avance intermedio que fue 2,5G para proveer servicios de datos con los estándares GPRS (Servicio General de Paquetes de Radio) y EDGE (Enhanced Data Rates for GSM Evolution), con velocidades de hasta 384 kbps, en las frecuencias 850-1900 MHz, usando modulaciones GMSK y 8-PSK. La tercera generación (3G) tiene como estándar a UMTS (Servicios de Telecomunicaciones Móviles Universales) que utiliza CDMA de banda ancha (WCDMA), con velocidades de hasta 2 Mbps, en las frecuencias 800 MHz-2.5GHz. Sus evoluciones son con estándares HSPA, HSPA+, dando velocidades de hasta 168 Mbps (Universidad Internacional de Valencia, 2018).

2.3.2 Tecnología móvil 4G

La cuarta generación (4G) utiliza el estándar de evolución a largo plazo (del inglés *Long Term Evolution*, *LTE*) que permite aumentar la capacidad y la velocidad de las redes de datos inalámbricas. LTE se usa a menudo para referirse a tecnologías de banda ancha inalámbrica o de redes móviles que

consideran altas velocidades de transmisión, con menor latencia (Technopedia.com, 2019).

LTE es uno de los estándares del 3GPP, que es un acrónimo de *3rd Generation Partnership Project*, que opera bajo un nombre registrado por el *European Telecommunications Standards Institute* (3GPP.org, 2019).

Las características de LTE incluyen una arquitectura de red plana completamente IP, calidad de servicio de extremo a extremo, tasas de descarga más altas que se acercan a los 300 Mbps en movimiento y 1 Gbps cuando los usuarios están inmóviles, y velocidades de carga de 75 Mbps, capacidad de las celdas expandidas para acomodar hasta 200 usuarios activos en función de las condiciones de tráfico e interferencia (Technopedia.com, 2019; Universidad Internacional de Valencia, 2018). Esto contribuye al uso de dispositivos de IoT que usan 4G para su comunicación. ya que la tecnología está diseñada para altas velocidades y alto movimiento. Por esta razón, este proyecto aprovecha dichas prestaciones para incorporar en el prototipo un módem 4G que se conecta por *Universal Serial Bus* (USB), proporcionando así una baja latencia, una alta velocidad de transmisión y una alta tolerancia a movimiento de una radio base a otra.

LTE tiene la capacidad de soportar una alta demanda de conectividad de nuevos dispositivos de consumo adaptados a nuevas aplicaciones móviles (Technopedia.com, 2019).

LTE maneja los estándares LTE-TDD (División de Tiempo Dúplex), LTE-FDD (División de Frecuencia Dúplex). Tiene una amplia gama de frecuencias de operación dependiendo de la región del mundo que son 700 MHz, 750 MHz, 800 MHz, 850 MHz, 900 MHz, 1800 MHz, 1900 MHz,1700/2100 MHz, 2300 MHz, 2500 MHz y 2600 MHz. Además, puede usar un ancho de banda entre 1,25 y 20 MHz. LTE utiliza acceso múltiple por división de frecuencia ortogonal (OFDMA) (3GPP, 2019). Los terminales móviles LTE transmiten con potencia de hasta 200mW y proporciona alrededor de 10ms de latencia que es mucho más bajo que las generaciones anteriores que están por encima de los 150ms. (CableFree, 2019; Joshi, Colombi, Thors, Larsson, & Tornevik, 2017; Kavanagh, 2018)

2.4 Placas de Microcomputadoras

Son computadoras diseñadas para funcionar en un circuito integrado único que contiene un microprocesador, un microcontrolador, memoria RAM, almacenamiento, conectores de entrada y salida entre otras cosas. Estos dispositivos permiten implementar aplicaciones de internet de las cosas por su tamaño y costo reducido. Considerando estos aspectos, el presente proyecto de titulación utiliza una de dichas placas para implementar un módulo vehicular. Estas placas a pesar de ser muy pequeñas, tienen gran capacidad y versatilidad ya que permiten instalar un sistema operativo completo(Technopedia.com, 2018).

Las placas más populares en dispositivos de internet de las cosas son Raspberry Pi, Arduino, Aaeon UP Squared y la nueva placa Nvidia Jetson. La placa Raspberry Pi 3B+ es la más icónica de todas ya que, por su accesibilidad, bajo costo y versatilidad, permite la implementación de muchas aplicaciones de IoT. Esta placa se verá en más detalle en el capítulo 3. Las placas Arduino son placas de microcontrolador, confiables pero limitadas en comparación a las otras en cuanto a capacidad de procesamiento ya que no poseen un microprocesador sino un microcontrolador, lo cual limita las capacidades de la placa. Las placas Aaeon UP Squared y Nvidia Jetson son superiores a las anteriores ya que son microcomputadoras, la última posee un GPU que puede procesar gran cantidad de información gráfica. La placa Aaeon se describe con más detalle adelante. Sin embargo, la placa Nvidia tiene entre 2 y 4 veces las capacidades de la Aaeon dependiendo del modelo en cuanto a procesamiento gráfico por la presencia del GPU de 128 núcleos. Todas las placas mencionadas tienen arquitectura similar, las diferencias se reducen a la cantidad de memoria RAM, almacenamiento, velocidad de procesador/controlador, numero de conectores de entrada y salida, y si tiene procesador gráfico. La placa Arduino no permite instalar un sistema operativo y está diseñada para tareas más mecánicas por lo que fue descartada como opción de este proyecto en etapa de diseño (Nvidia, 2019).

2.5 Python

Es un lenguaje de programación creado por *Guido van Rossum* en el año 1991. Generalmente usado para desarrollo web en servidores de aplicación, desarrollo de programas, matemáticas y rutinas de sistema. Actualmente, se está usando la versión 3 de dicho lenguaje.

Es un lenguaje multiplataforma que tiene sintaxis bastante simple y permite programar en menor número de líneas de código que otros lenguajes. Python funciona en un sistema interprete lo que significa que el código puede ser ejecutado inmediatamente después de escrito, lo cual hace que se puede hacer un prototipo muy rápido. Se puede programar de forma estructurada, orientada a objetos o en funciones (w3schools, 2019).

Con Python se puede programar en un simple editor de texto, sin embargo, también es posible programar en un ambiente de desarrollo integrado como Thonny, Pycharm, Netbeans o Eclipse, que son particularmente útiles para manejar grandes cantidades de archivos. Para este proyecto se programó directamente en un ambiente Python disponible para Ubuntu.

Entre las ventajas que tiene Python está la facilidad de lectura ya que tiene mucha similitud con el idioma ingles con influencia de matemáticas. Otra ventaja es que separa un comando de otro solo con saltos de línea a diferencia de otros lenguajes que usan punto y coma o paréntesis, esto reduce la probabilidad de errores de escritura al momento de programar. Además, utiliza tabulaciones para definir profundidad de bucles, funciones y clases mientras otros lenguajes usan llaves (w3schools, 2019). También se debe tomar en cuenta que es de código abierto por lo que no requiere un alto costo de licenciamiento. Así, tiene varias librerías que reducen el tiempo de programación y tiene conectividad con bases de datos (net-informations.com, 2019).

Por otro lado, Python tiene desventajas como mayores velocidades de ejecución frente a otros lenguajes como C o C++. No es un buen lenguaje para aplicaciones móviles, es considerado débil por lo que muy pocas aplicaciones son hechas en Python. Otra desventaja es que utiliza gran cantidad de memoria RAM por la flexibilidad de tipos de datos. El acceso a bases de datos es limitado comparado con otras tecnologías más populares, la capa de acceso es primitiva. Por último,

debido a que es un lenguaje escrito dinámicamente, requiere mayor cantidad de pruebas de ejecución ya que muchos errores no aparecen hasta que el programa se ejecuta varias veces seguidas (net-informations.com, 2019). En este trabajo se evidenció que después de dos o tres ejecuciones consecutivas del código aparecían errores que no ocurrían en la primera ejecución.

2.6 Transmisor-Receptor Asíncrono Universal

El Transmisor-Receptor Asíncrono Universal (UART por sus siglas en inglés) es un circuito físico en un microcontrolador. La función principal de un *UART* es trasmitir y recibir información de forma serial, esto generalmente se hace por conexiones directas, pero también se lo puede hacer de forma inalámbrica.

El transmisor envía directamente al receptor del otro dispositivo y viceversa. La información es enviada desde el bus de datos de forma paralela al *UART* transmisor, luego es transformado en un paquete de datos con bits de inicio, fin y paridad. El paquete de datos es transmitido de forma serial al *UART* receptor a una determinada velocidad en baudios. El *UART* receptor quita los bits de inicio, fin y paridad, convierte los datos seriales en paralelo y los transfiere al bus de datos.

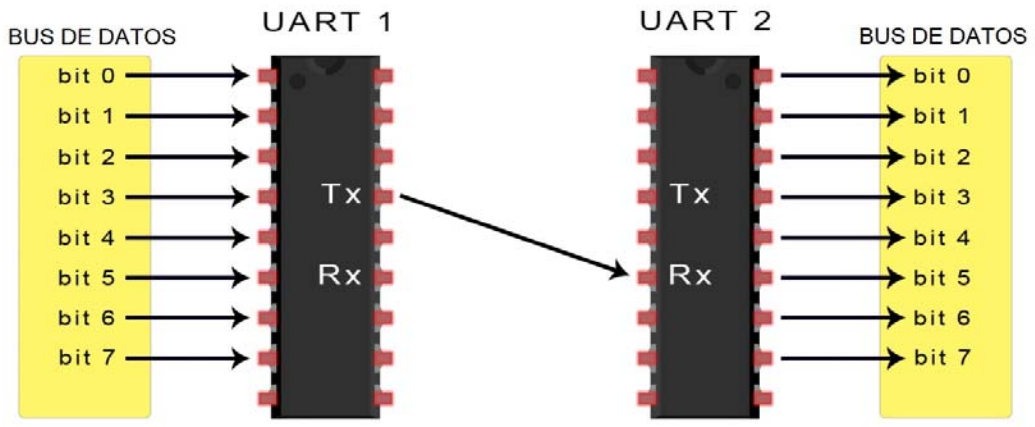


Figura 1. Transmisión tipo UART

Tomado de: (Circuit Basics, 2016)

Este tipo de transmisión tienes sus ventajas como es la conexión simple (dos cables o vía inalámbrica), no es necesaria señal de reloj, tiene un bit de paridad para evitar errores, la estructura del paquete puede ser cambiada siempre que se mantenga consistencia, y sobre todo hay gran cantidad de documentación. Por otro lado, las desventajas son que tiene limitación del tamaño del paquete de datos a 9 bits, no soporta múltiples sistemas maestro-esclavo, y los niveles de baudios de cada *UART* solo pueden diferir en 10% entre ellos (Circuit Basics, 2016).

2.7 Acelerómetros

“Es un dispositivo electromecánico que nos sirve para medir las fuerzas de aceleración, ya sea estática o dinámica. Son útiles para detectar las vibraciones y el movimiento en los sistemas. Son comúnmente usados para activar sistemas de protección en vehículos, o computadoras, para registrar la actividad física en las personas, para aplicaciones de orientación en los dispositivos móviles, etc.” (electronicaestudio, 2019)

“Existen acelerómetros de 1, 2, y 3 ejes, las conexiones básicas que se requieren para la operación son la alimentación y las líneas de comunicación interactúan a través de un conversor análogo-digital o *pulse-wide modulation* (pwm). El acelerómetro con interfaz analógica envía un voltaje proporcional a la aceleración en cada uno de sus ejes, estos son más baratos que los digitales y más fáciles de usar” (electronicaestudio, 2019).

“Un acelerómetro con una interfaz digital se comunica a través de los protocolos de comunicación de SPI o I2C que es lo óptimo para este proyecto. Además, es menos susceptible al ruido que los acelerómetros con interfaz analógica. Los acelerómetros con salida PWM tienen una salida de onda cuadrada con un periodo conocido, pero un ciclo de trabajo que varía con el cambio en la aceleración” (electronicaestudio, 2019).

2.8 Servidores de Servicios

Existen tres tipos básicos de servidores en el internet: servidores web, servidores de aplicación o servicios, y servidores de bases de datos.

Los servidores web son todos los servidores a los que las personas se conectan cada vez que desean acceder una página web como Facebook, Google o Wikipedia. Son servidores que se presentan al usuario y donde se almacenan las páginas web públicas o privadas que conforman el internet visible. Estos servidores se implementan montando el servicio de dominio con una plataforma de servidor web.

Los servidores de bases de datos son servidores que solo almacenan y administran bases de datos usando un motor de base de datos como Oracle o SQL server generalmente las bases que manejan son bases de aplicaciones que están funcionando en otros servidores. Estos servidores generalmente suelen ser conjuntos de servidores redundantes para evitar perdida de información en caso de alguna falla.

Los servidores de aplicación o servicios son servidores que ejecutan funciones específicas de forma transparente para el usuario final. Por esta razón, se los suele llamar servidores de *backend* junto con los de base de datos. Son indispensables en implementaciones de IoT, ya que son un acceso en el internet al que los dispositivos pueden conectarse para cumplir su función. Estos tres tipos de servidores muchas veces están combinados entre sí de distintas maneras y muchas veces son varios servidores de cada tipo para evitar fallas, es decir, existe redundancia.

Las ventajas de tener los servidores descentralizados y visibles al internet es que si falla uno no se corta el servicio. Además, son accesibles desde cualquier parte del mundo a través de internet, siempre están funcionando, son administrados de forma centralizada y fácil. No obstante, algunas de estas características pueden también representar una desventaja debido a altos costos de implementación.

2.9 PHP

PHP es un acrónimo recursivo de preprocesador de hipertexto y es un lenguaje de programación de código abierto que está especialmente diseñado para desarrollo de páginas web y puede ser embebido en las mismas para cumplir diversas funciones en servidores de *backend* de manera que sea el punto de enlace entre dispositivos de *IoT* (php.net, 2019).

PHP tiene la característica de que todo el código se ejecuta en el servidor generando así un código html (lenguaje marcado de hipertexto) que es enviado al cliente. Por lo tanto, el cliente recibe los resultados pero desconoce las operaciones ejecutadas por el servidor. Esto representa una gran ventaja de seguridad en entorno de desarrollo web. Adicionalmente, PHP es extremadamente simple de usar y provee muchas prestaciones avanzadas para programadores profesionales (php.net, 2019).

PHP permite realizar varias tareas, pero hay tres áreas en las que el lenguaje es más usado, estas áreas son: Rutinas del lado de servidor, rutinas de línea de comando y aplicaciones de escritorio (The PHP Group, 2019).

Las rutinas implementadas en el servidor constituyen las mayores aplicaciones en PHP. En esta área el PHP se enlaza con la página web, el navegador, la base de datos y hace todas las transacciones de forma invisible para el usuario y muestra los resultados a través de páginas web. Las rutinas de línea de comando son servicios que se ejecutan sin servidor web o navegador, solo se necesita un ambiente de ejecución para que se ejecute la rutina, como sucede en este proyecto. Las aplicaciones de escritorio no son el fuerte de PHP, pero se puede lograr muchas cosas en el lado de cliente debido a que es multiplataforma (The PHP Group, 2019).

2.10 Aplicaciones Móviles

Son programas que permiten realizar una o varias funciones de interacción entre el usuario y una aplicación. Se instalan en dispositivos móviles como teléfonos celulares inteligentes o *tablets*. Actualmente, existen dos grandes sistemas operativos móviles en el mercado: Android y iOS. Para Android se puede programar en Kotlin, JAVA o C++ usando el kit de desarrollo de aplicación Android (SDK) dentro de Android Studio, IntelliJ IDEA, Netbeans IDE o AppInventor. Para iOS se programa en una variación de C o en Swift usando los IDEs Xcode o Swift.

2.11 Ambientes Integrados de Desarrollo Android

Los ambientes integrados de desarrollo (IDE) permiten programar aplicaciones compatibles con el sistema operativo Android de Google. Existen muchos IDEs

Android, sin embargo, los más populares son Android Studio, IntelliJ, NetBeans, Eclipse, AppInventor, etc. Todos los IDEs para Android tienen tiempos similares de desarrollo excepto AppInventor que tiene tiempos más bajos, debido a que usa programación basada en bloques. AppInventor fue creado por el Instituto Tecnológico de Massachusets y es un IDE web que no requiere muchos recursos (López Villegas, 2014; Massachusetts Institute of Technology, 2019).

3. Diseño del Prototipo

En el presente capítulo se describirá el diseño del prototipo del sistema de seguridad y monitoreo propuesto, el cual como se observa en la figura 2, se estructura de la siguiente manera: un módulo vehicular, el cual permite detectar movimiento gracias a la presencia de un acelerómetro y enviar fotos instantáneas del perímetro del auto al propietario, un servidor de servicios en la nube que permite enviar y recibir información entre el módulo vehicular y la aplicación móvil, y una aplicación móvil que nos permite controlar el sistema remotamente y recibir las fotos instantáneas tomadas por las cámaras conectadas al módulo. Para un mejor entendimiento, cada componente será debidamente descrito a continuación.

Figura 2. Diagrama General del Prototipo

3.1 Módulo Vehicular

El módulo vehicular consta de un dispositivo que va ubicado dentro del auto. Como se puede observar en la figura 3, el diagrama del módulo vehicular cuenta con: una placa *Aaeon Up Squared* conectada a un acelerómetro/giroscopio MPU6050 en los pines 2-5 de la placa para transmitir la información del acelerómetro; un módem 4G USB, el cual permite transmitir las fotos al servidor de aplicación con alta velocidad; un *hub* USB en el cual se conectan las cuatro cámaras web que permiten tomar las fotos del perímetro del auto; dos LEDs (uno verde para indicar si la placa está prendida y otro rojo para indicar si el sistema está activado o en *standby*), un pulsador de respaldo para encender y apagar el sistema; un transformador DC-DC (asegura la entrada de 5V a la placa), una batería de 12V; un *switch* de encendido de la placa y un conector para cargar la batería.

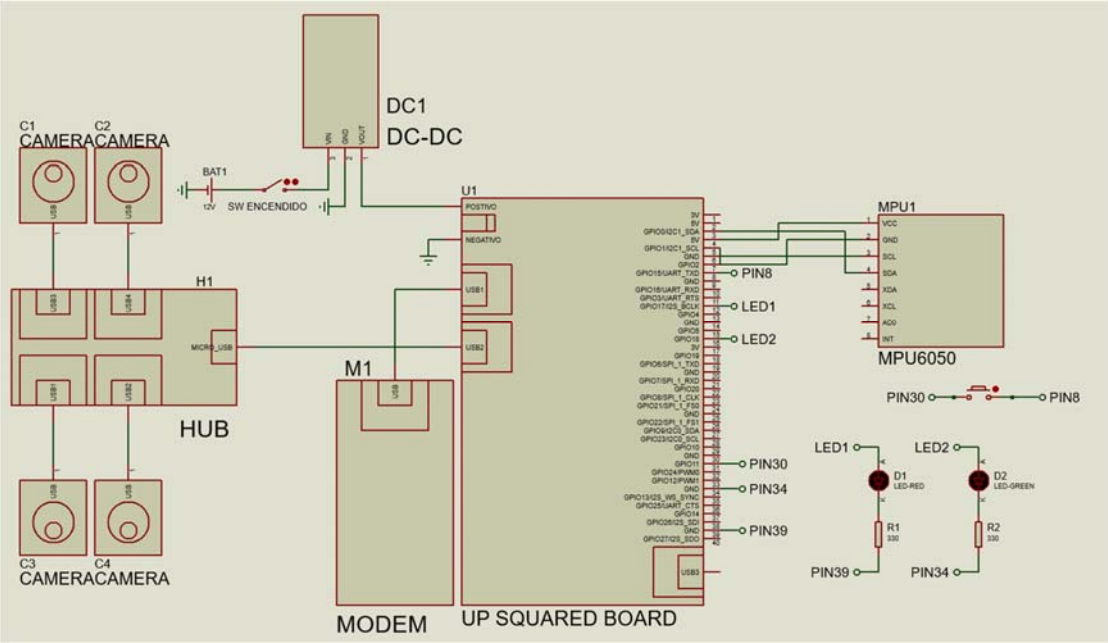


Figura 3 Diagrama del Módulo Vehicular

Inicialmente, se optó por realizar pruebas con un Arduino MEGA, pero al realizar un test con una cámara, la placa dejó de operar con lo que se descartó como opción. En segundo lugar, se conectó una cámara a un Raspberry Pi 3 B+ y se determinó que la placa no contaba con las especificaciones necesarias para manejar el nivel de información requerido. Por tal motivo, existió un

sobrecalentamiento y el tiempo de respuesta fue demasiado alto. Considerando estas pruebas fallidas, se realizaron pruebas con otros dispositivos como la placa Aaeon Up Squared con la cual se obtuvieron buenos resultados. Específicamente, esta placa cuenta con un microprocesador y un microcontrolador que puede manejar toda la información que genera el sistema de una manera adecuada. Genera un nivel tolerable de calor y tiene un tiempo de respuesta mejor que la interfaz Raspberry. Las especificaciones técnicas de estos dispositivos serán explicadas con mayor detalle en el siguiente capítulo.

Por otro lado, para la parte de comunicación se planteó usar un shield 4G (Libelium blog, 2016) pero este componente no resultó óptimo para este diseño ya que, como podemos observar en el datasheet del Anexo 5 (Invensense, 2013), ocupa pines de la placa usados por el acelerómetro MPU6050. Además, la transmisión es demasiado lenta ya que usa un tipo de transmisión UART (Libelium, 2016). Por otra parte, concentra demasiado el calor generado por la placa produciendo un sobrecalentamiento de todo el sistema lo que hace que la placa deje de funcionar. Por estos motivos, se realizaron pruebas de funcionamiento con un módem inalámbrico con un módulo Wi-Fi para lograr la conexión a internet del módulo vehicular. Sin embargo, este dispositivo tampoco resultó eficiente debido a que incrementaba los tiempos de respuesta del sistema, lo cual no es óptimo ni eficiente para el prototipo. Por este motivo, se realizaron pruebas con un módem USB 4G Huawei el cual, luego de realizar pruebas, se observó que satisface todas las necesidades del proyecto, brindando así una alta velocidad de transmisión y baja latencia para poder transmitir las fotos e información al servidor de aplicación.

Después de realizar las primeras pruebas, se consideró necesario agregar dos LEDs indicadores. Uno verde para saber si la placa está encendida y otro rojo para poder verificar si el sistema esta encendido o apagado. Se incluyó este último led debido a que si no existe una pantalla conectada a la placa Aaeon no se puede saber si la misma está encendida lo que causa incertidumbre en el usuario. Luego de fallas en la comunicación entre la aplicación móvil y el modulo vehicular, al prender o apagar el sistema, por cuestiones de cobertura celular, se determinó que era necesario añadir un pulsador de respaldo para el sistema en

la parte posterior izquierda del auto y un switch de encendido de la placa en la parte posterior derecha del carro.

De la misma forma, dado que se implementó el módulo en un auto a escala, fue necesario incorporar una fuente de poder para que el sistema no dependa de una toma de electricidad y conservar su movilidad. En consecuencia, se puso una batería de 6V y 4Ah para proveer al sistema de energía, pero en las pruebas realizadas esta fuente demostró ser insuficiente debido a que la placa no recibía suficiente voltaje. Por lo tanto, se optó finalmente por una batería de 12V y 5Ah conectada a un transformador DC-DC para proporcionar los 5V que necesita la placa. Con esta batería se obtuvieron buenos resultados, pero no ideales, ya que la batería dura media hora y permite hacer pruebas o una demostración del funcionamiento del prototipo lo cual es una deficiencia del prototipo. Como se observa en la figura 4, esta batería requiere ser recargada después de uso del sistema por lo que se incorporó como se puede ver en la Figura 5 un conector para un adaptador de carga en el costado izquierdo del modelo a escala para dicha tarea.

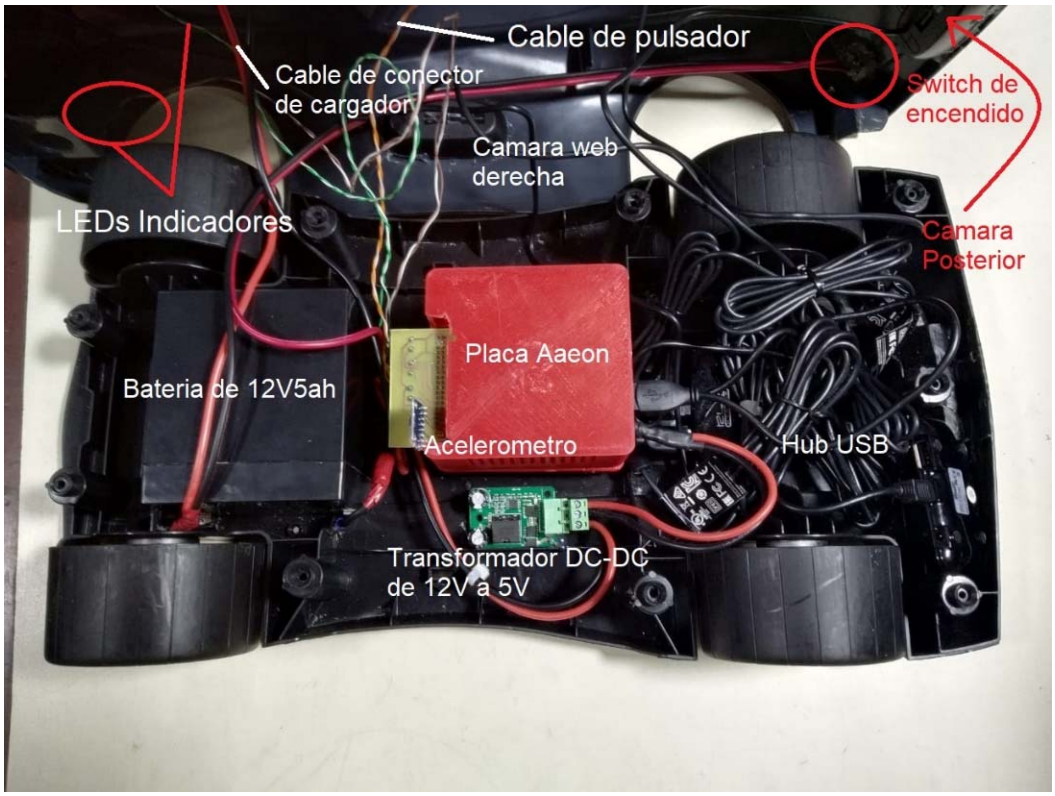


Figura 4 Diagrama de Prototipo Vehicular

Figura 5 Conector de Carga de Batería

3.2 Servidor de Servicios

Para el diseño del servidor en la nube, se consideró inicialmente el diagrama del sistema mostrado en la figura 6. Este diagrama consta de dos componentes, un repositorio temporal para las fotos y una cuenta en un servicio llamado Twilio, el cual se encarga de recibir las instrucciones del módulo vehicular, adjuntar las fotos del repositorio, enviarlas al Whatsapp del propietario y al correo de emergencia. Desafortunadamente, esta arquitectura de funcionamiento creó múltiples puntos de posibles fallos debido a que implica demasiados enlaces o rutas de transmisión. Específicamente, estos tramos son: entre el carro y el repositorio, entre el repositorio y Twilio, entre el carro y Twilio, entre Twilio y el usuario. Todas esas rutas de transmisión y distintos servicios aumentan el tiempo de respuesta, genera una conexión inestable. De manera particular, si la conexión falla, todo el sistema deja de funcionar. Además, con esta configuración, se obtuvieron altos tiempos de respuesta del sistema desde su activación hasta la recepción de las fotos en el teléfono del propietario o el correo de emergencia, es decir, una alta latencia. El tiempo de respuesta fue de entre 4 y 5 minutos, lo cual hacía deficiente al prototipo y muy poco óptimo, ya que para el objetivo planteado, se requiere que la información llegue al usuario en tiempo real de ser posible.

Figura 6 Estructura Inicial del Sistema Propuesto

Por consiguiente, se eliminó el servicio Twilio y se incorporó sus funciones a un servidor unificado de servicios y repositorio como se observa en la figura 7, con el fin de reducir tiempos de respuesta y puntos de fallo debido a la reducción de tramos de trasmisión y servicios involucrados. Para esto se creó una rutina en PHP, a la cual el modulo vehicular se conecta directamente. Dicho servidor de servicios unificados envía las fotos directamente a la aplicación móvil y al correo de emergencia. Con esta configuración, se optó por no enviar las fotos a través de Whatsapp debido a que se analizó que no todas las personas tienen este programa en sus teléfonos celulares. Adicionalmente, debido a que en el mismo servidor unificado se almacenan las fotos, se redujo los puntos de fallo (puntos en el sistema donde podría fallar la comunicación por latencia o errores y el sistema deje de funcionar) de cuatro a dos, y el envío de las fotos al correo de emergencia es una redundancia del sistema para asegurar que las fotos llegan al dueño. De esta manera, el sistema fue optimizado y el tiempo de respuesta se redujo notablemente a 45 segundos en promedio.

Figura 7 Diagrama final del Sistema Propuesto

3.3 Aplicación Móvil

Para el desarrollo de la aplicación móvil, inicialmente se planteó utilizar Android Studio. Sin embargo, dadas las demoras, cambios de diseño en hardware, errores en el prototipo del módulo vehicular y largos tiempos de desarrollo en Android Studio, se optó por realizar la aplicación móvil en App Inventor que proporciona las mismas funcionalidades, pero el tiempo de desarrollo es mucho menor. En este sentido, el proceso de implementación se torna más eficiente.

En la aplicación, inicialmente se solicitará el correo electrónico del contacto de emergencia, que puede ser cambiado en cualquier momento en la pantalla de configuración. Además, en la aplicación se puede encender y apagar el sistema desde la pantalla principal en cualquier parte del mundo y recibe las fotos directamente en la pantalla de fotos generando una notificación en el teléfono del propietario. Después de visualizar las fotos, estas pueden ser borradas para no causar confusión al usuario.

La aplicación cuenta con 5 pantallas en su interface gráfica. Cabe indicar que la aplicación es escalable y, por lo tanto, estas pantallas se pueden aumentar en número o en botones según su necesidad. A continuación, se detallan dichas pantallas. En primer lugar, como se puede observar en la figura 8, la aplicación

tiene una pantalla de inicio donde se puede ver una imagen del sistema y el botón que llevará al usuario a la pantalla principal.

Figura 8 Pantalla de Inicio

En la figura 9, se observa la pantalla principal de la aplicación. A través de esta interfaz se puede acceder a un menú que contiene: ayuda, configuración en el cual podrá ingresar el correo de emergencia, y a opciones de encender o apagar el sistema. Con un mensaje de estado del sistema y en esta pantalla, se podrán verificar las fotos enviadas desde el vehículo.

Figura 9 Pantalla Principal

Posterior a esto, se puede observar que en la figura 10 se solicita el correo electrónico del contacto de emergencia. Esta pantalla de configuración permite ingresar o cambiar la dirección de correo electrónico, así como también permite guardar la configuración y regresar a la pantalla anterior.

Figura 10 Pantalla de Configuración

En la figura 11 se observa que la aplicación cuenta con una pantalla de ayuda donde se proporciona indicaciones e información de cómo configurar y operar la aplicación de una forma rápida y sencilla.

Figura 11 Pantalla de ayuda

Finalmente, en la figura 12 se muestra que la aplicación permite revisar las fotos recibidas en la pantalla denominada fotos, las cuales se despliegan con indicadores de a qué cámara corresponden en el prototipo.

Figura 12 Pantalla de Fotos

4. Implementación del Prototipo

En este capítulo se detalla la implementación de cada parte del prototipo y se describen los obstáculos que surgieron durante el desarrollo del mismo. Para un

mejor entendimiento, la implementación se dividirá en tres partes: el módulo vehicular, el servidor de aplicación y la aplicación móvil.

4.1 Implementación del Módulo Vehicular

En la sección 2.1, se mencionó que el módulo en primera instancia sería implementado con un Arduino MEGA o un Raspberry Pi 3 B+. Sin embargo, al realizar pruebas con el Arduino MEGA, la placa dejaba de operar cuando se usaba junto con una cámara Logitech C270. Por otra parte, al realizar pruebas con la cámara conectada al Raspberry, el prototipo se demoraba aproximadamente 40 segundos solo para capturar y guardar una foto, antes de enviarla. Estos resultados no eran aceptables ni eficientes de acuerdo a los objetivos del proyecto, por lo que se buscó una alternativa más eficiente. Se realizaron pruebas con una placa Aaeon Up Squared. Con este dispositivo, el tiempo de captura y almacenamiento de una foto fue aproximadamente 4 segundos, por lo que se decidió optar por la placa Aaeon. Las principales especificaciones técnicas entre las placas consideradas se observan en la Tabla 1.

Como se puede observar en la tabla 1 y en los Anexos 3 y 4, la placa Aaeon es muy superior a la placa Raspberry Pi en muchos aspectos: posee el doble de RAM (Ubuntu usa 2GB), mayores opciones de conectividad, mayor cantidad de pines de acceso y además tiene un procesador gráfico. El procesador del Raspberry, aunque tiene un cortex quad-core de 1.4Ghz, no logra el mismo desempeño que el procesador Celeron Duo Core de 2.4Ghz de la placa Aaeon. Todas estas diferencias significaron un cambio fuerte en el diseño del prototipo, pero un cambio necesario ya que el proyecto depende de un tiempo de respuesta lo más cercano al tiempo real, de manera que se logre mantener un equilibrio entre costo y beneficio.

Tabla 1.

Comparación entre Placas Consideradas

Representación gráfica de las placas / **Parámetros**			
Modelo de Placa	**Arduino MEGA 2560 Rev. 3**	**Raspberry Pi 3B+**	**Aaeon UP Squared**
Procesador / Microcontroladora	ATmega2560 @ 16 MHz	Broadcom BCM2837B0, Cortex-A53 quad-core 64-bit SoC @ 1,4GHz	Intel Celeron Duo Core N3350 2,4 GHz
Memoria	8 KB	1 GB LPDDR2 SDRAM	2 GB LPDDR4 SDRAM
Conectividad	• 1x USB-B port	• 2,4 GHz and 5 GHz 802.11.b/g/n/ac Wireless, Bluetooth 4.2 • 4 × USB 2.0 ports	• 2x RJ-45 • 2x USB 2.0 ports • 4x USB 3.0 ports • 1x mPCI-e • 1x M.2 2230 • 1x SATA
Acceso	Digital I/O Pins 54 Analog Input Pins 16	GPIO header de 40 pines	40 pin GP-bus enabled by Altera MAX 10 FPGA + 60 pin EXHAT
Almacenamiento	256 KB	Tarjeta microSD	32 GB eMMC
Gráficos	N/D	N/D	Intel HD Graphic 505

Adaptados de: (Aaeon, 2018; Arduino, 2019; Raspberry Pi Foundation, 2016)

Una vez decidida la placa a utilizar para el procesamiento de la información, se utilizaron las cámaras web Logitech C270 que tienen interfaz USB y tienen una resolución de 720 pixeles.

Posteriormente, se procedió a realizar un análisis sobre el módulo 4G para el prototipo. Se planteó usar el *shield* 4G + GPS para *Arduino / Raspberry* LE910. Desafortunadamente, después de revisar más en detalle las especificaciones del *shield*, se encontró que el tipo de transmisión es UART. UART por el tipo de transmisión, genera una velocidad de transmisión máxima en módems de 230 kbps que no es aceptable para la cantidad de información que será transmitida (ARC Electronics, 2018). Adicionalmente, otro inconveniente fue que no había disponibilidad de pines suficiente para el *shield*. Específicamente no existían pines suficientes lo que limitaba la conexión del acelerómetro/giroscopio MPU6050. Por otro lado, si se conectaba el *shield* sobre la placa *Aaeon*, hubiera generado demasiado calor para el modulo vehicular, provocando una temperatura de operación insostenible para todo el sistema y haciéndolo ineficiente. Finalmente, no era compatible con la placa *Aaeon* utilizada.

En consecuencia, se decidió usar un módem 4G que funcione por USB. En primer lugar, se realizaron pruebas con un módem inalámbrico Huawei modelo E5573. Este equipo usa dos puertos USB. Específicamente, uno era para cargar el equipo y otro para un *dongle* wifi para la placa. Esto aumento el tiempo de respuesta de todo el sistema por la conexión wifi de la placa y, por lo tanto, no es óptimo para el proyecto por el uso de USBs. Por último, se realizaron pruebas con un módem Huawei E3372 4G que demostró ser adecuado y eficiente porque solo usa un puerto USB y proporciona velocidad de hasta 150 Mbps. A continuación, en la Tabla 2 se presenta una comparación con las especificaciones técnicas de cada módem. El prototipo requiere, por el tamaño de las imágenes a transmitir y el tiempo de respuesta del sistema, una velocidad superior a los 21 Mbps proporcionada por 3G. Por estas razones, es necesaria la velocidad de hasta 150 Mbps que brinda 4G. En la tabla se puede evidenciar que 4G solo dan los últimos dos módems, a pesar de todos ser compatibles con las bandas de frecuencia de las operadoras celulares ecuatorianas (850 MHz y 1900 MHz). Por esta razón, se escogió el módem Huawei 3372.

Tabla 2.

Tabla Comparativa de los módems considerados

Representación Gráfica / Parámetros				
Marca / Modelo	**Huawei / E173**	**Alcatel / X602A**	**Huawei / E5573**	**Huawei / E3372**
Factor	USB 2.0	USB 2.0	Wi-Fi	USB 2.0
Conectividad	EDGE, GPRS, GSM, HSDPA, HSUPA, UMTS	GSM, UMTS	GSM, GPRS, EDGE, UMTS, HSPA, HSPA+, LTE	GSM, EDGE, HSUPA, UMTS, LTE
Bandas	GSM 850/900/1800 /1900 UMTS 900/2100 MHz	GSM 850/900/1800 /1900 UMTS 850/1900 MHz	UMTS 850/900/2100 MHz LTE 850/900/1800 /2100/2600 MHz	GSM 850/900/1800 /1900 UMTS 900/2100 LTE 700/850/1700 /1900/2100/2600 MHz
Velocidad maxima	7,2 Mbps	21 Mbps	150 Mbps	150 Mbps

Adaptados de: (Alcatel, 2017; CBS Interactive, 2019; Huawei, 2018, 2019)

Por otro lado, la implementación del prototipo se lo realizó en un auto de plástico de 50cm de largo el cual se muestra en la Figura 14

Figura 13 Foto del Auto de Plástico Usado Para Implementar El Prototipo.

El auto en cuestión se escogió por la cantidad de espacio interior para colocar los dispositivos. La placa Aaeon se ubicó en el centro del auto con un case impreso en 3D, un acelerómetro MPU6050, conectado los pines 1-4 del mismo a los pines 2-5 de la placa, la batería de 12V-5Ah en la parte frontal del auto, dos hubs USB en la parte trasera para conectar las cámaras, teclado y mouse para configuración de la placa, los distintos modelos usados del módem 4G, el switch de encendido de la placa y pulsador de respaldo de encendido del sistema instalados en la parte trasera y conectados a los pines de la placa, LEDs indicadores instalados en la parte frontal del carro a manera de faros y conectados a los pines de la placa.

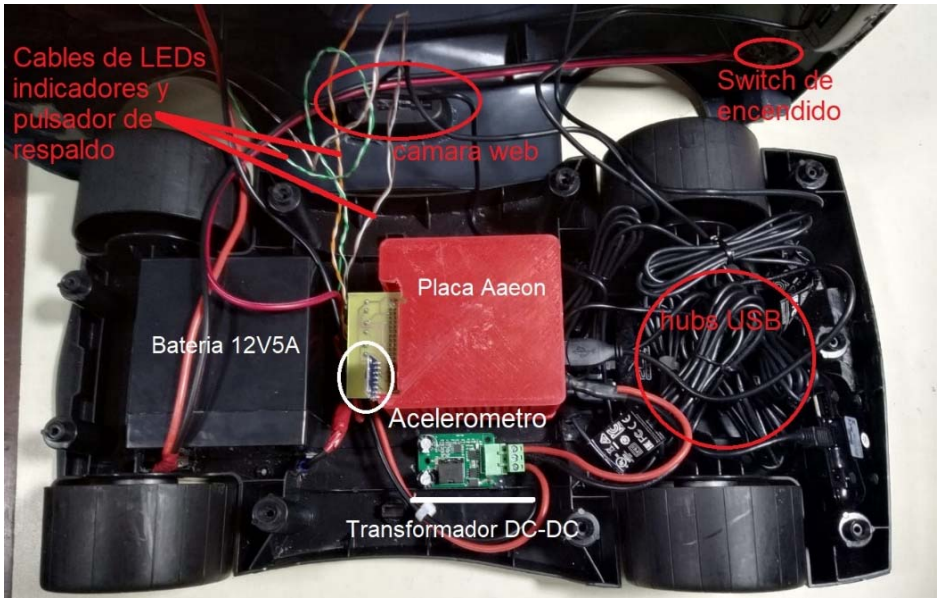


Figura 14 Prototipo Implementado

Para que el prototipo realice la captura, almacenamiento y envío de las imágenes por internet a la aplicación móvil, es necesario utilizar un sistema operativo que ejecute una aplicación de forma automática, por lo que se procedió a instalar Ubuntu con un programa realizado en Python. De esta manera, el sistema operativo ejecutará la aplicación al iniciarse y el sistema vehicular funcionará en el instante que se encienda la placa de forma automática. Para esto, se descargó el ISO de Ubuntu 18.04.2 de la página de Ubuntu y un programa llamado Etcher que permite convertir un flash drive USB en un dispositivo de arranque con la ISO de Ubuntu. Luego de instalar de manera normal en la placa Aaeon el sistema operativo Ubuntu, se procedió a ejecutar los siguientes comandos en el terminal para actualizar Ubuntu, instalar la versión para la placa UP Squared de Ubuntu, borrar las partes genéricas de la instalación, instalar el firmware actualizado de la placa, instalar las librerías extras de la placa y habilitar el uso de los pines de la placa:

- $ sudo apt update #actualiza lista de repositorio
- $ sudo apt full-upgrade –y #instala las actualizaciones
- $ sudo reboot #reinicia ubuntu
- $ sudo add-apt-repository ppa:ubilinux/up #añade el repositorio de la placa UP Squared
- $ sudo apt update #actualizala lista de repositorio
- $ sudo apt-get autoremove --purge 'linux-.*generic' #borra el kernel genérico
- $ sudo apt-get intall linux-image-generic-hwe-18.04-upboard #instala el kernel propio de la placa UP Squared
- $ sudo reboot #reinicia ubuntu
- $ uname –srv #muestra la versión de kernel
- $ sudo apt install firmware-ampak-ap5214a #instala el firmware para wifi y bluetooth desde PPA en ubuntu
- $ sudo apt install upboard-extras #instala librerías que permiten uso de GPIO, PWM, SPI y I2C de la placa
- $ sudo usermod -a -G gpio upsquared #se da permiso de uso al usuario de GPIO

- $ sudo usermod -a -G leds upsquared #se da permiso de uso al usuario de LEDs
- $ sudo usermod -a -G spi upsquared #se da permiso de uso al usuario de SPI
- $ sudo usermod -a -G i2c upsquared #se da permiso de uso al usuario de I2C
- $ sudo usermod -a -G dialout upsquared #se da permiso de uso al usuario de UART
- $ sudo reboot reinicia ubuntu

Con estos comandos, ubuntu está listo para funcionar en la placa Aaeon, a continuación, se usó los siguientes comandos para identificar los pines de la placa en el Ubuntu. Esta información será utilizada más adelante en el programa de Python para poder manejar los puertos ya que la numeración física no es igual a la numeración lógica de los pines.

- $ ls /sys/bus/pci/devices/*/i2c_designware.0/ | grep i2c
- $ ls /sys/bus/pci/devices/*/i2c_designware.1/ | grep i2c

La instalación de Python y sus complementos se realizó a través del terminal con los siguientes comandos:

- $ sudo apt-get install python-pip python-dev nginx #instala Python en ubuntu
- $ sudo add-apt-repository ppa:mraa/mraa #añade el repositorio mraa
- $ sudo apt-get update #actualiza la lista de repositorios
- $ sudo apt-get install libmraa2 libmraa-dev libmraa-java python-mraa node-mraa mraa-tools #instala las librerías Libmraa que permiten acceder a los pines de la placa desde python
- $ sudo apt-get install mraa-tools mraa-examples libmraa1 libmraa-dev libupm-dev libupm1 upm-examples #instala las librerías Libmraa que permiten acceder a los pines de la placa desde python

- $ sudo apt-get install python-mraa python3-mraa node-mraa libmraa-java #instala las librerías Libmraa que permiten acceder a los pines de la placa desde python
- $ sudo apt-get update #actualiza la lista de repositorios
- $ sudo pip install email_to #instala la librería que permite usar email en python
- $ pip install email_to #instala la librería que permite usar email en python
- $ sudo add-apt-repository ppa:thopiekar/pygame #añade el repositorio de la librería pygame que captura imágenes de camaras
- $ sudo apt-get update #actualiza la lista de repositorios
- $ sudo apt-get install python3-pygame #instala la librería que permite el acceso a las camaras y la captura de imágenes
- $ pip3 install pgzero #instala la librería que permite el acceso a las camaras y la captura de imágenes

A continuación, se procedió a realizar el programa en Python, cuyo código fuente se observa en el Anexo 1.

Terminado el programa se necesita configurar el sistema operativo para que la aplicación se inicie de forma automática al iniciar Ubuntu. Esto se lo hace haciendo una pequeña configuración del Cron tab de Ubuntu. A continuación, se presentan los pasos a seguir:

1. En el programa de Python se coloca el siguiente comando para comenzar. #!/usr/bin/python
2. Se crea un archivo bash en gedit (iniciarp.sh), en donde se indica que abrirá al archivo Python, con la dirección del programa:
   - #!/bin/bash
   - cd /home/uphost/Escritorio
   - python3 pruebareboot.py
3. Se coloca en el terminal el comando “sudo crontab -e” para abrir el cron tab

4. Se coloca @reboot /home/uphost/iniciarp.sh en el archivo del cron tab para que el programa se inicie de forma automática.

(Revsky01, 2018)

La figura 15 presenta el diagrama de flujo del programa en Python en el cual, en el diagrama de la izquierda, se hace la verificación de estado del sistema si esta encendido o no al momento de iniciar la placa. En el diagrama de la derecha está el proceso que sigue el sistema cuando se enciende el sistema. Luego se guarda el correo de contacto de emergencia recibido de la aplicación móvil, se lee los valores del acelerómetro. Si el sistema detecta un movimiento se captura las fotos desde las cámaras, caso contrario se continúa leyendo los valores del acelerómetro. Si el sistema captura las fotos de las cámaras, se las guarda en memoria, se las envía al servidor y el servidor las envía a la aplicación móvil y el correo. Finalmente, al terminar el proceso se vuelve a leer los valores del acelerómetro.

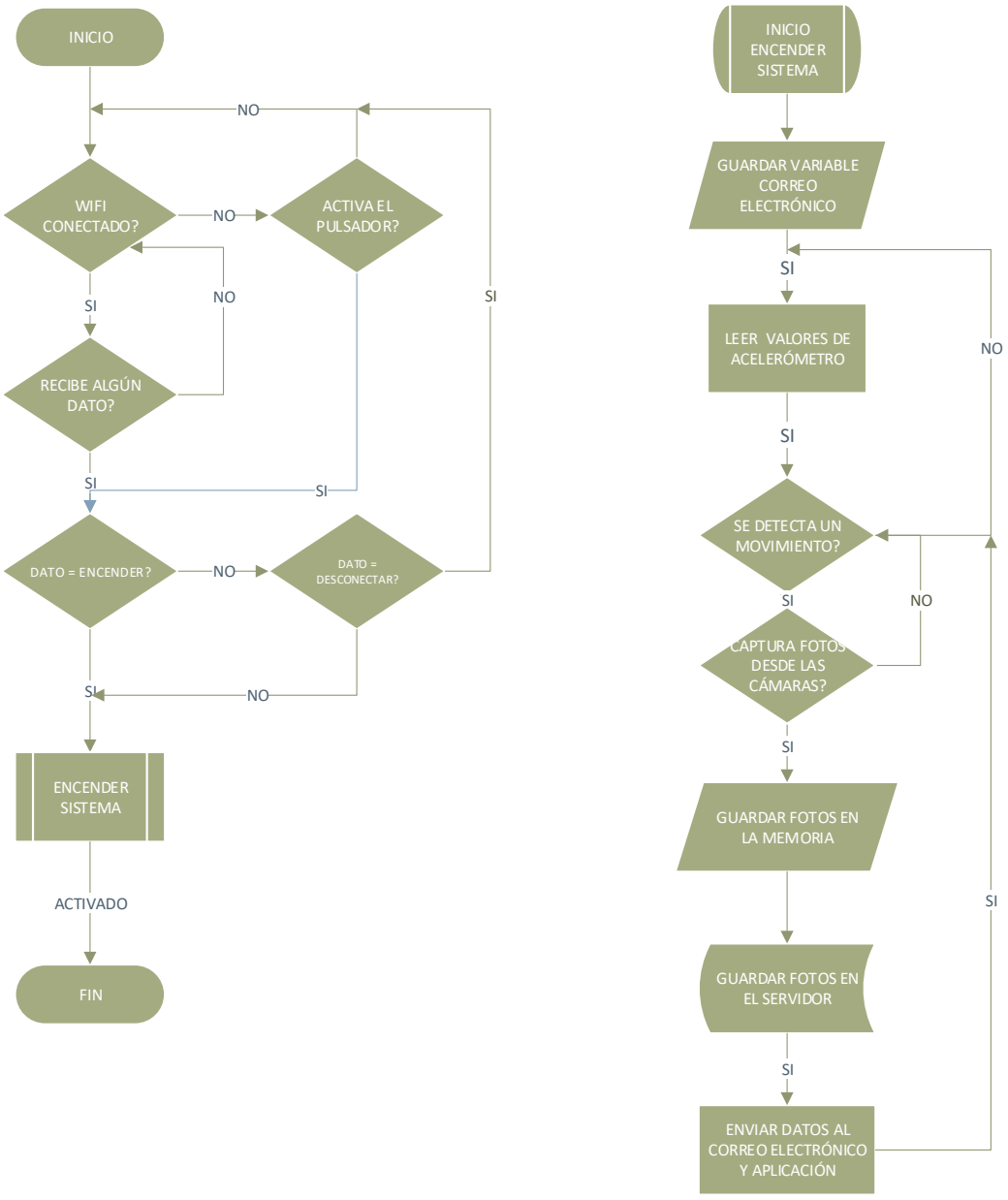


Figura 15 Diagrama de flujo del Programa en Python

4.2 Implementación del Servidor de Servicios

El sistema, por el funcionamiento del módulo vehicular y la conexión a una aplicación móvil a través del internet, necesita un punto fijo en la nube donde las dos partes puedan conectarse. Esto implica que es necesario tener un servidor de servicios transparente para el usuario, con una IP pública fija a la que pueda conectarse desde cualquier parte del mundo tanto la aplicación móvil, como el

modulo vehicular. Por este motivo, se implementó un programa en PHP, el cual está ubicado en un servidor público que provee servicios de *hosting* para este tipo de proyectos.

En la figura 16 se muestra que para poder acceder al programa PHP del servidor se ingresa en Administrador de archivos.

Figura 16 Administrador de Archivos del Servidor

Una vez en Administrador de archivos, se abre la carpeta public_html y creamos una carpeta “CarroTesis”. Los archivos de public_html pueden ser vistos colocando el link como se ve subrayado en rojo en la figura 16 mediante permisos dados. Esta carpeta tendrá el programa en PHP y las imágenes guardadas.

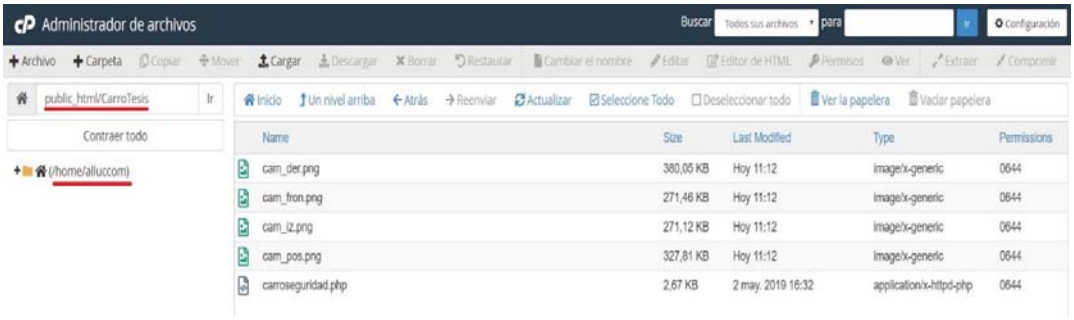


Figura 17 Directorio del Administrador de Archivos del Servidor

En la figura 18 se muestra que al dar clic en cargar se sube el archivo del programa en PHP.

Figura 18 Carga del Programa en PHP

Una vez cargado el programa en PHP, se da clic en Permisos (ver en la Figura 19) con lo cual se otorga permisos de lectura. Esto se puede realizar desde cualquier dispositivo. Si no se realiza este procedimiento no se conectarán la tarjeta AEEON ni la aplicación móvil.

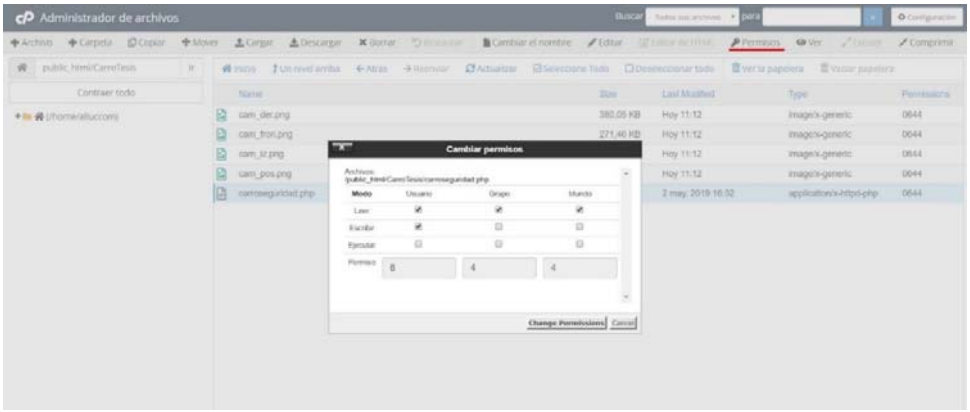


Figura 19 Modificación de Permisos del Servidor

4.2.1 Programa PHP

El programa en PHP se realizó con el IDE Sublime Text. El servidor de PHP recibe el estado del sistema desde el vehiculo. Asimismo, el servidor recibe el correo del contacto de emergencia y estado de la aplicación móvil. Se crea una base de datos en el servidor para guardar si se encuentra activado o desactivado el sistema, y notifica a la aplicación y a la tarjeta AEEON. Además, guarda el dato del estado de la alarma y se envía el correo electrónico de emergencia al módulo vehicular.

Se conecta a la base de datos mediante el usuario y contraseña:

Host: localhost

Usuario: alluccom_tesisca

Contraseña: tesiscarro

Base de datos: alluccom_CarroTesis

Los datos de la aplicación y de la tarjeta se reciben con funciones *GET* (se envía la información de forma visible) y *POST* (se envía la información de forma no visible) como se puede ver en la Figura 20. En el caso de la aplicación se recibe mediante el método GET y en el caso de la tarjeta se recibe mediante el método POST.

```php
<?php
#Conexion al servidor, con usuario y contraseña
$link=mysql_connect("localhost","alluccom_tesisca","tesiscarro") or die (mysql_error());
mysql_set_charset("utf8");
mysql_select_db("alluccom_CarroTesis") or die(mysql_error());

#Variables enviadas desde aplicacion
$estadoapp = $_GET['estadoapp'];
$correo =$_GET ['correo'];
$boton=$_GET['boton'];
$apagar=$_GET['apagar'];

#Variables enviadas desde python
$dato=$_POST['dato'];
$dato1=$_POST['dato1'];
$aviso=$_POST['aviso'];
```

Figura 20 Recepción y envío de datos

El programa de Python recibe la imagen desde cada cámara y a cada imagen se le asigna un nombre. Luego, se envía mediante PHP esa información y se guardan las imágenes en el servidor con el nombre recibido como se puede observar a continuación en la Figura 21.

```php
#Si recibe mediante el método post el nombre de la imagen se guarda en una variable
if (isset($_POST["nombre"])){
    $nombre_foto=$_POST["nombre"];
}
#Si recibe mediante el método POST la imagen
if(isset($_POST["foto"])){
    #Guarda la imagen en la carpeta de servidor creada.
    $bytes = file_put_contents($nombre_foto, base64_decode($_POST["foto"]));
    echo json_encode($bytes);
}
```

Figura 21 Recepción de las Imágenes y Almacenamiento de las mismas

En la figura 22 se puede observar la forma en la que se obtiene en la variable dato “enviar” el último ID guardado. Despues de obtener el ultimo ID, se obtienen los datos del estado del sistema, el correo y se envian a la tarjeta AEEON.

```
#Si envio mediante POST enviar
if ($dato == "enviar"){
    #Busco el ultimo dato guardado
    $result = mysql_query("SELECT MAX(ID) FROM Datos", $link);
    #si encuentro el ultimo valor guardado
    if ($row = mysql_fetch_array($result)){
        #Guardo en la variable id
        $id = trim($row[0]);
        print ($id);
    }
    #Con el id verifico el ultimo dato guardado
    $resultado2 = mysql_query("SELECT * FROM `Datos` WHERE `ID` LIKE '$id'", $link);
    #Si encuentro los datos
    if ($row = mysql_fetch_array($resultado2)){
        #Envio los datos de correo y estado de la aplicacion
        printf("&%s&%s&", $row[1], $row[2]);
    }

}
```

Figura 22 Verificación de la Base de Datos

Como se puede evidenciar en la Figura 23, si se obtiene la palabra “recibe” en la variable dato1, se guardarán los datos que envía la tarjeta en la base de datos. Por otro lado, si se recibe desde python la palabra “encendido”, se guarda la activación de la alarma en la base de datos.

```
#Envia el estado del sistema desde la tarjeta AEEON, guarda en la base de datos
if ($dato1=="recibe"){
    $correo1 = $_POST['correo'];
    $estado =$_POST['estado'];
    $query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES (NULL,'$correo1','$estado')";
    $result = mysql_query($query);

}

#Envio de alarma encendida
if ($aviso=="encendido"){
    $alarma=$_POST['alarma'];
    print ($alarma);
    $query = "INSERT INTO `Sistema` (`ID`, `Alarma`,`Fecha`) VALUES (NULL,'$alarma', CURRENT_TIMESTAMP)";
    $result = mysql_query($query);

}
```

Figura 23 Activación del Sistema

En la figura 24 se muestra que si se recibe en el servidor el comando “activado”, desde la aplicación o se aplasta el pulsador, se agrega en la base de datos el estado del sistema, se guarda el correo de emergencia en la base de datos y se envía a la aplicación y a la placa “sistema activado”. En caso que se reciba el comando “desactivado” o se aplaste el pulsador, se agrega en la base de datos

el estado del sistema de “desactivado” y se envía a la aplicación móvil y al módulo vehicular la actualización.

```
#Si activo el pulsador o pulso el botón de encender, entonces se guarda el estado de activacion
if($estadoapp == "activado"){
    #Se guarda el correo registrado y el estado de la activacion/desactivacion
    $query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES (NULL,'$correo','activado')";
    $result = mysql_query($query);
    echo "Sistema Activado";
}
#Si activo el pulsador o pulso el botón de desconectar, entonces se guarda el estado de desactivacion
if($estadoapp == "desactivado"){
    $query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES (NULL,'$correo','desactivado')";
    $result = mysql_query($query);
    echo "Sistema Desactivado";
}
```

Figura 24 Control de la Activación del Sistema

En la figura 25 se muestra que si el comando “recibir” se obtiene desde la aplicación en la variable boton, entonces el programa verifica en la base de datos el estado del sistema que tiene guardado con el estado recibido desde la aplicación, si existe una diferencia entonces actualiza la base de datos.

```
if ($boton == "recibir"){
    $result = mysql_query("SELECT MAX(ID) FROM Datos", $link);
    if ($row = mysql_fetch_array($result)){
        $id = trim($row[0]);
    }
    $resultado2 = mysql_query("SELECT * FROM `Datos` WHERE `ID` LIKE '$id'", $link);
    if ($row = mysql_fetch_array($resultado2)){
        print($row[2]);
    }
}
```

Figura 25 Verificación del Estado del Sistema

Por otra parte, en la figura 26 se evidencia que si se recibe en la variable $apagar desde la aplicación móvil el estado denominado “apagado”, entonces se envía a la base de datos el comando para apagar el sistema, para luego cerrar la conexión a la base de datos.

```
if ($apagar=="apagado"){
    $query = "INSERT INTO `Sistema` (`ID`, `Alarma`,`Fecha`) VALUES (NULL,'0', CURRENT_TIMESTAMP)";
    $result = mysql_query($query);
}

mysql_close($link);

?>
```

Figura 26 Apagado del Sistema

4.2.2 Base de Datos

En esta subsección describimos el proceso para crear la base de datos utilizada. Para esto, se coloca en Base de datos MySQL como se muestra en la Figura 27. Para poder abrir esa base de datos se da clic en PHPMyAdmin.

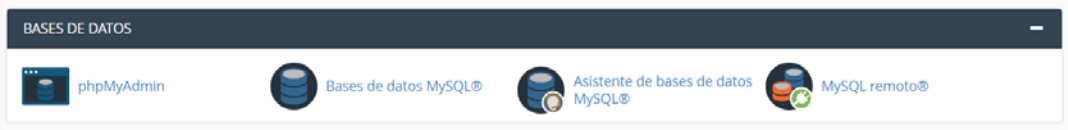


Figura 27 Panel de control de Servidor

En la figura 28 se muestra una parte del proceso para la creación de la base de datos. Así, observe que se coloca el nombre de la base de datos en el campo nueva bases de datos. Debido a que se usó un *hosting* externo, se mantiene el resto de los datos sin cambiar.

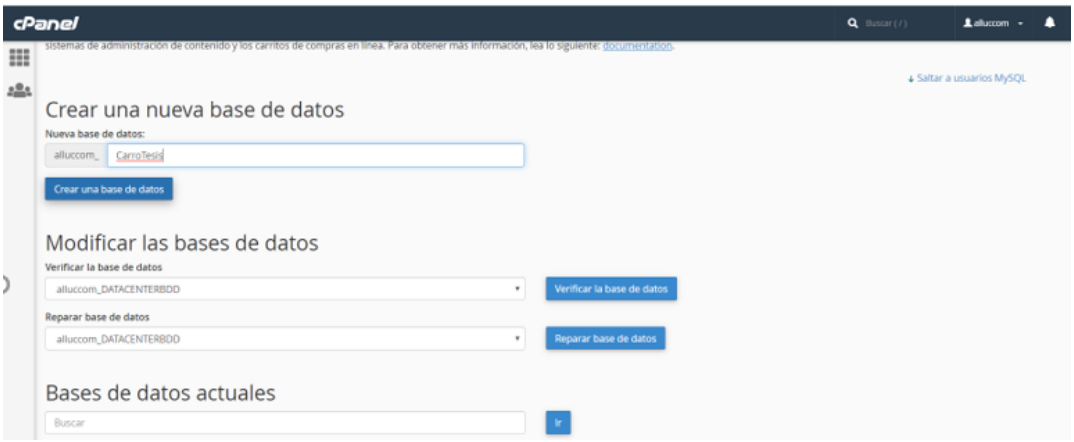


Figura 28 Creación de la Base de Datos

Una vez creada la base de datos, se crea el usuario de administración y se registra en la base de datos. Este proceso se muestra en la figura 29 donde el formulario solicita el nombre de usuario y el nombre de la base de datos.

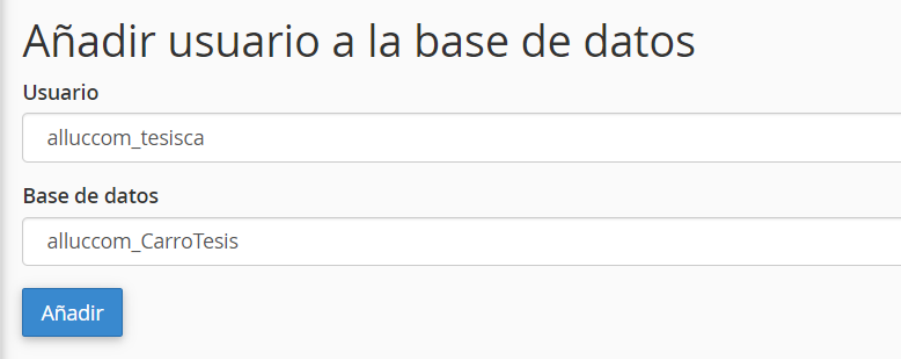


Figura 29 Registro del Usuario en la Base de Datos

Como se puede observar en las figuras 30, 31 y 32, en la base de datos se crean dos tablas. La primera tabla se denomina Datos, en la cual se registraran los estados de la aplicación a través de los campos: ID (tipo de dato entero de 11 digitos autoincremental), correo (tipo de dato char variable de 100 caracteres) y estadoapp (habilitado/deshabilitado) (tipo de dato char variable de 11 caracteres). La segunda tabla se denomina Sistema, en la cual se registran las fotos tomadas por las cámaras del prototipo con los campos ID (tipo de dato entero de 11 digitos autoincremental), alarma (tipo de dato char variable de 11 caracteres) y fecha (tipo de dato timestamp).

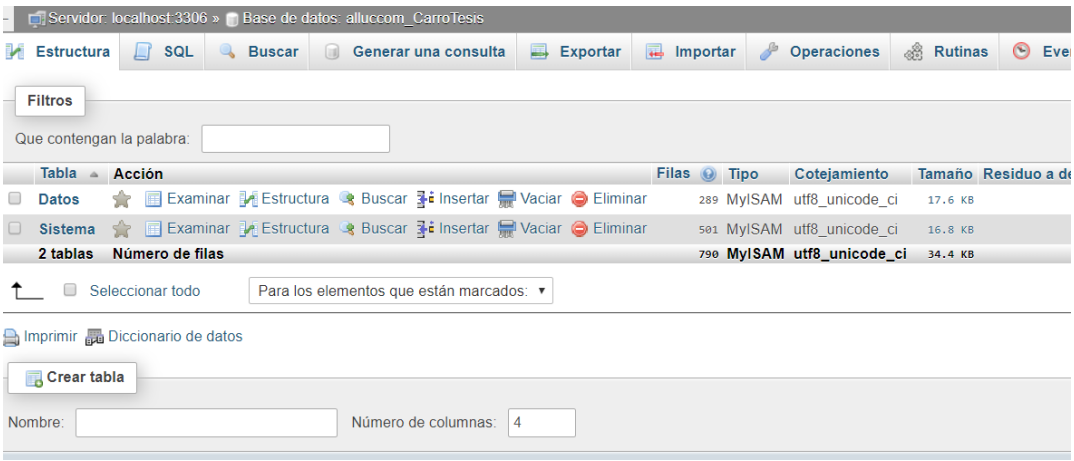


Figura 30 Creación de las Tablas

Figura 31 Campos de Tabla Datos

Figura 32 Campos de Tabla Sistema

El servidor de servicios cumple una función importante que es enlazar la aplicación móvil con el módulo vehicular (ver figura 7). Las funciones específicas que cumple el servidor son: recibe la señal de activación y el correo electrónico de emergencia de la aplicación móvil, envía el comando de activación y el correo electrónico de emergencia al módulo vehicular, en caso de activación de la alarma, el servidor recibe una notificación y las fotos tomadas por el prototipo, envia la notificación y las fotos a la aplicación móvil, recibe la señal de desactivación del sistema desde la aplicación móvil y envía el comando de desactivación al módulo vehicular.

En la figura 33 podemos observar un diagrama de flujo del servido de servicios, el cual inicia y verifica la recepción de datos. Para esto, monitorea la variable Dato. Luego, si el Dato es Encender, entonces procede a guardar en la variable “correo” el correo electrónico de emergencia, para luego enviar la información al módulo vehicular. Si el Dato es Imagen, guarda las imágenes en el servidor y posteriormente las envía a la aplicación móvil. Finalmente, si el Dato es Desconectar, envía el comando desconectar al módulo vehicular.

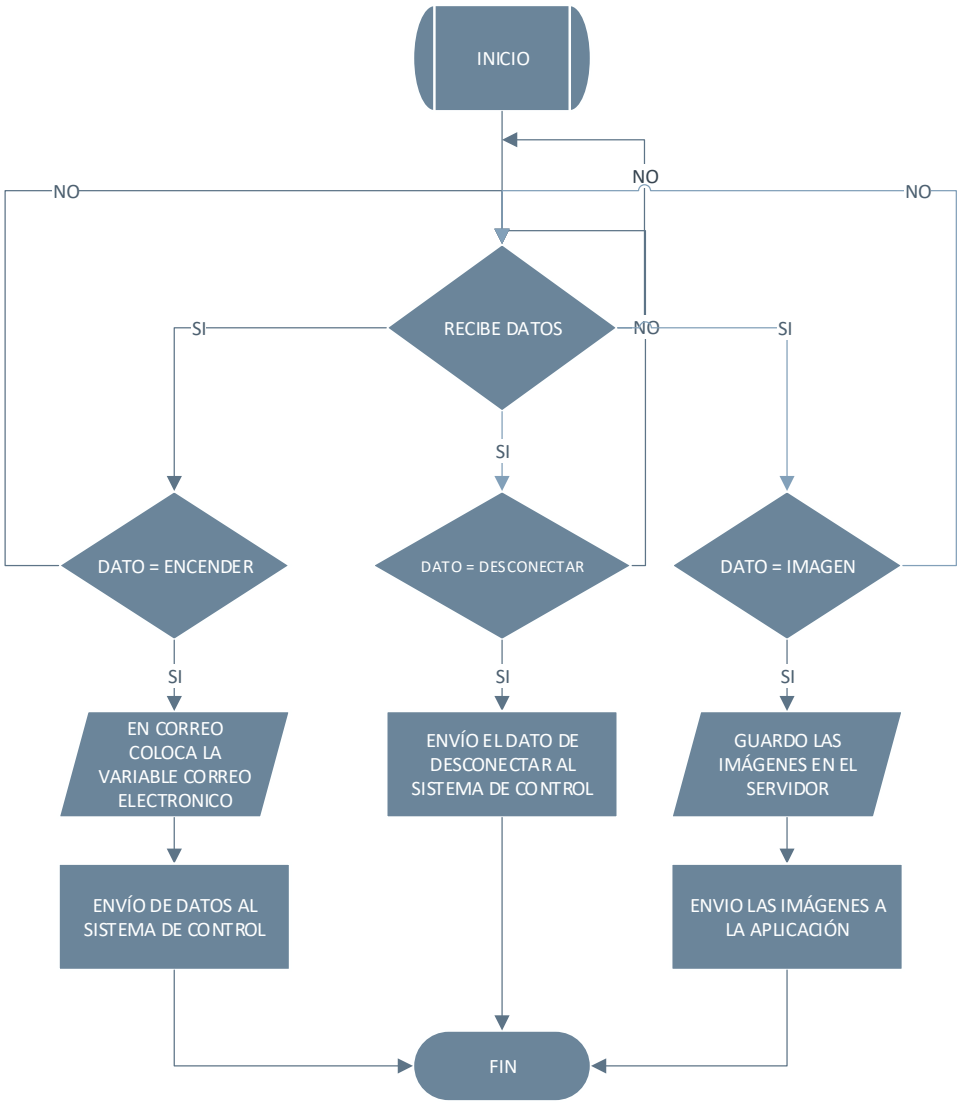


Figura 33 Diagrama de flujo del Servidor de Servicios

4.3 Desarrollo de la aplicación móvil

Como se mencionó en la sección 2.3, la aplicación móvil fue desarrollada en *AppInventor*, con 5 pantallas diferentes.

En la figura 34 podemos ver el código de la pantalla de inicio, que es la pantalla de presentación de la aplicación (ver figura 8) y solo posee un botón que pasa a la pantalla principal (*Screen* 2). En la figura 35, se puede observar el código de la pantalla de ayuda que posee un botón de regreso a la pantalla principal (*Screen* 2). Esto se debe a que esta pantalla solo posee información de operación de la aplicación como se pudo ver en la figura 11.

Figura 34 Pantalla de Inicio

Figura 35 Pantalla de Ayuda

En la figura 36, se puede ver el código de la pantalla de fotos, dónde primero está programado un botón que permite regresar a la pantalla principal, luego está una función que se inicia cuando la pantalla fotos hace visible 4 campos a través de los cuales se conecta con el servidor de servicios y descarga las fotos guardadas. A continuación, está un botón que borra las imágenes guardadas en el servidor. En la figura 37 se observa el código de la pantalla de configuración que posee una función que pone a todas las variables en 0. Otra función que está asociada a un botón para regresar a la pantalla principal. Una tercera función asociada al botón de guardar, que registra los valores ingresados en un base de datos local hasta poder enviarlos al servidor de aplicación. Finalmente, una función que se ejecuta al cargar la pantalla de configuración que llama a los valores guardados en la base de datos local.

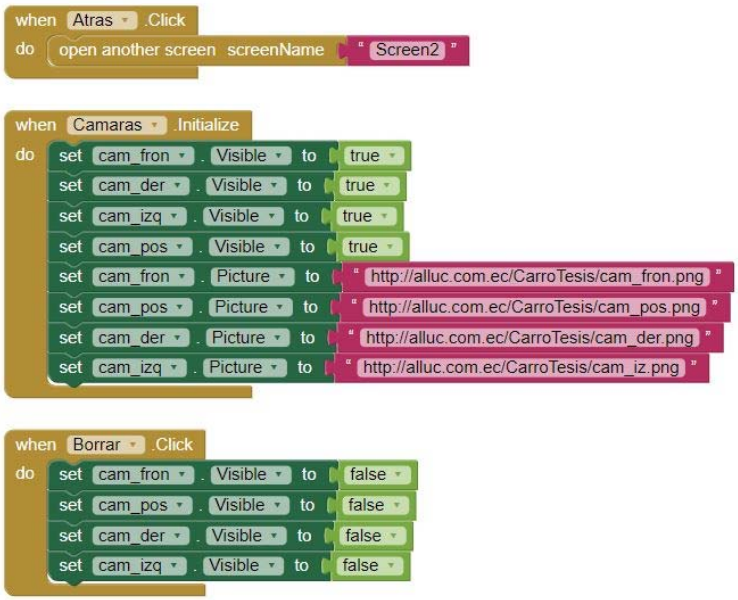


Figura 36 Pantalla de Fotos

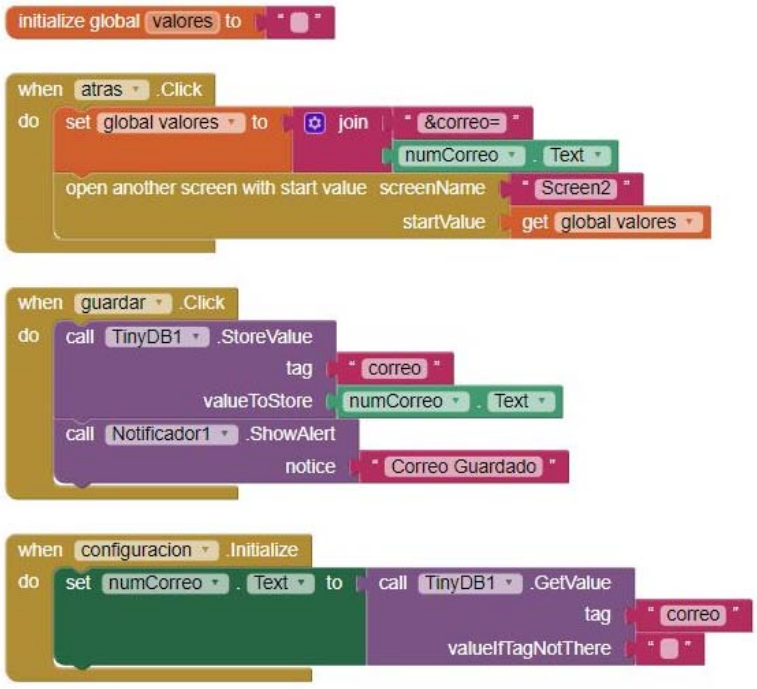


Figura 37 Pantalla de Configuración

En la figura 38 se muestra una parte del código de la pantalla principal. Esta parte del código inicia con la declaración de variables en cero. Luego, se muestra

un proceso que cuando se inicia la pantalla principal obtiene el valor del correo electrónico de emergencia guardado en la base de datos. A continuación, se puede observar una función llamada “envio” donde se coloca la trama de datos que se enviará al servidor usando el método *GET*.

En la Figura 39, se puede observar el código asociado a los botones de encender el sistema, desactivar el sistema y el proceso de recepción de datos desde el servidor. En el botón de encender y desactivar se guarda el estado en el que está el sistema en dos variables “envio” y “valores”, para luego llamar al proceso “envio”. La variable “envio” se usa para enviar los datos al servidor usando el proceso “envio”, y la variable “valores” se usa en la aplicación para verificación posterior del estado del sistema. A continuación de dichas funciones, se muestra el bloque de procesos “*web1*”, el cual se encarga de recibir los datos desde el servidor. En este Bloque de procesos se toman los dos primeros caracteres de la trama de datos recibida y se guarda en una variable “estado”. La variable “estado” puede tomar 4 valores para indicar los estados del sistema. Si el valor de la variable “estado” es igual a “de”, el sistema esta desactivado y el mensaje en la aplicación muestra “sistema desactivado”, oculta el botón desactivar y muestra el botón activar. Si la variable “estado” es igual a “ac” entonces el sistema está activado, el mensaje en la aplicación muestra “sistema activado”, oculta el botón activar y muestra el botón desactivar. De la misma manera, si la variable “estado” es igual a “1&”, la alarma ha sido activada y la aplicación emite la notificación de que la alarma ha sido activada. Si la variable “estado” es igual a “0&” la alarma no está activa y se guarda el estado en la base de datos local.

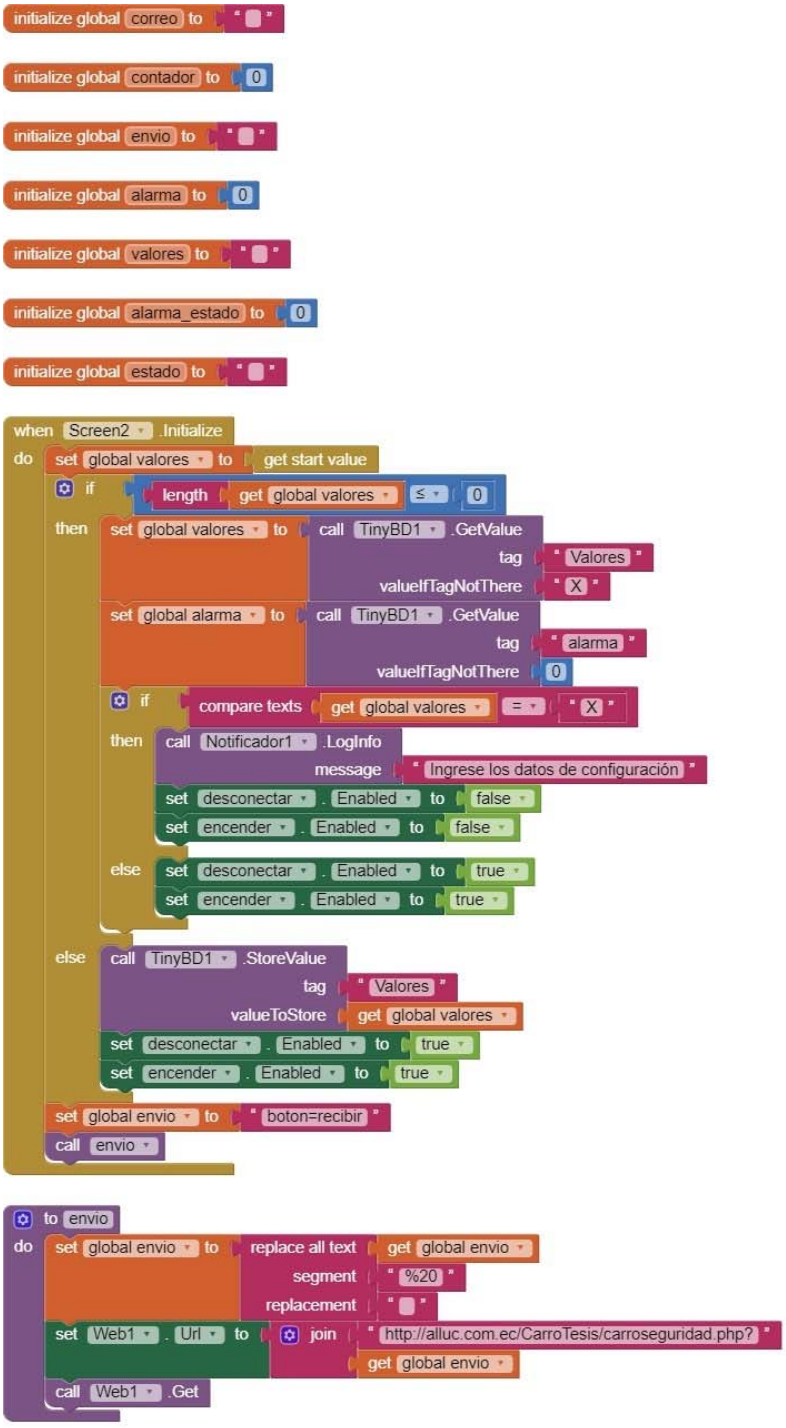


Figura 38 Pantalla Principal Parte 1 de 3

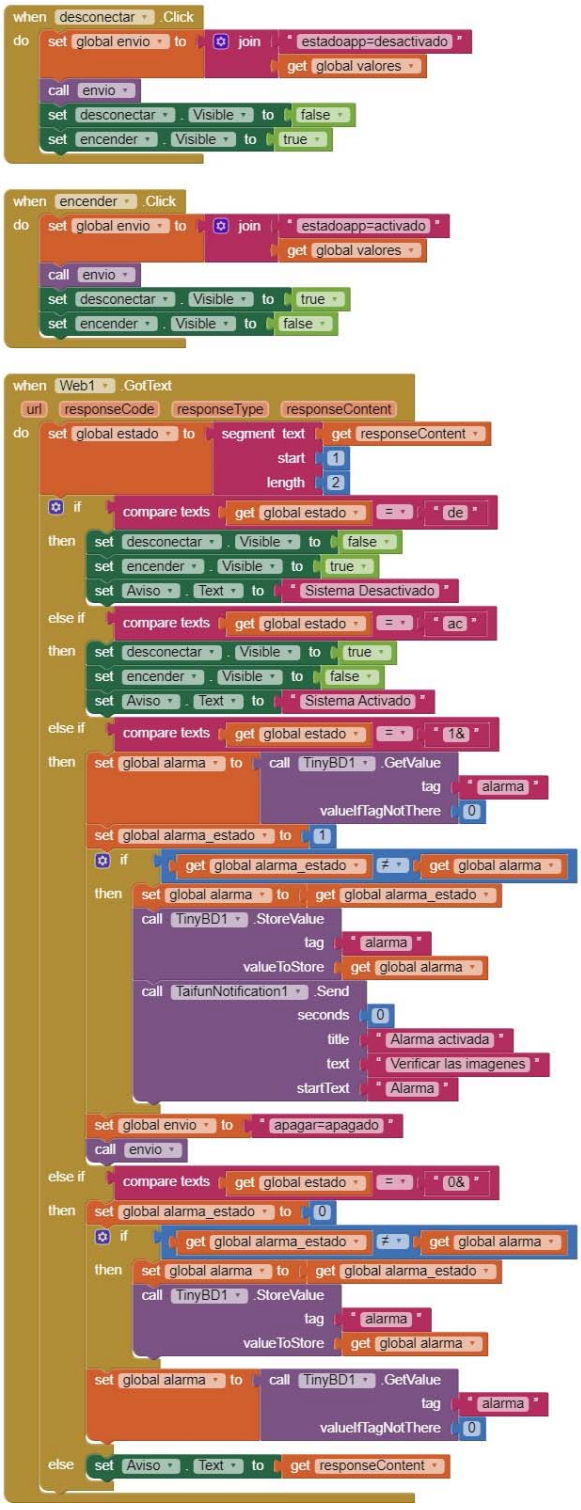


Figura 39 Pantalla Principal Parte 2 de 3

En la figura 40, podemos observar el código asociado al botón imágenes, el botón ayuda, el botón configuración, una función de reloj y una función en caso de error. El botón imágenes permite abrir la pantalla Cámaras, donde se encuentran las imágenes recibidas desde el servidor y se emite una notificación de que se está abriendo dicha pantalla. El botón ayuda y el botón configuración redireccionan la aplicación a las respectivas pantallas. A continuación, se encuentra una función de reloj que servirá como contador para verificar el estado de la aplicación y el estado de la alarma en caso de cualquier cambio. Finalmente, se puede observar la función de error, la cual verifica si la aplicación emite un error número 1101 que significa que no existe conexión a internet o que no hay conexión con el resto del sistema.

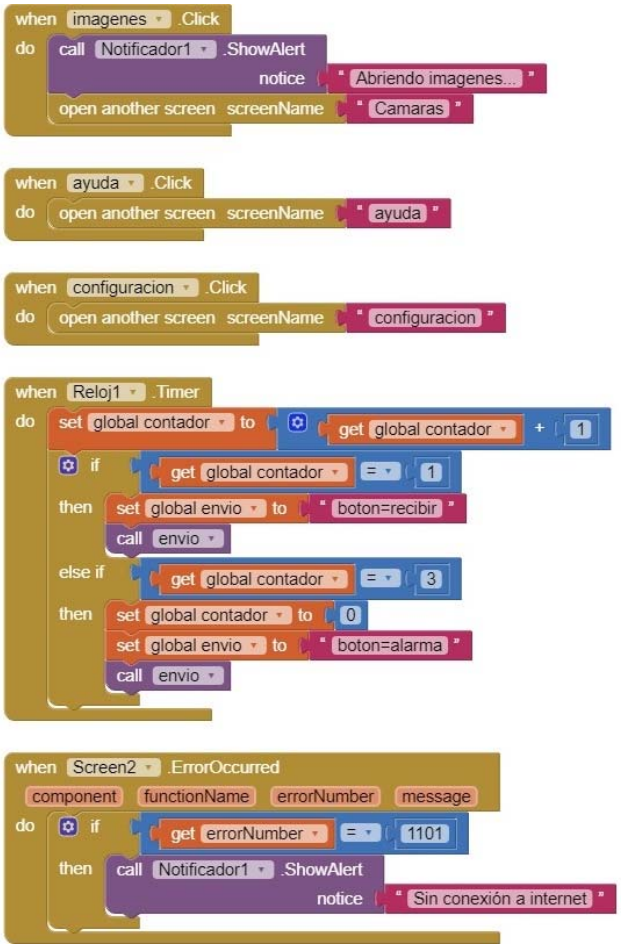


Figura 40 Pantalla Principal Parte 3 de 3

Finalmente, en la figura 41 se muestra el diagrama de flujo relacionado con el funcionamiento de la aplicación móvil. El cual inicia si se pulsa el botón de inicio. Luego, espera a que se pulsen los otros botones de la pantalla principal, caso contrario espera a que se pulse el botón de inicio.

A continuación, si se pulsa el botón de configuración, se puede ingresar el correo de emergencia. De ser ese el caso, se puede guardar en una base de datos interna, caso contrario se pulsa el botón atrás y se regresa a la pantalla principal. Si pulsa el botón ayuda, muestra los pasos a seguir de la aplicación.

De la misma forma, si se pulsa el botón de encender sistema, se verifica si existe un correo electrónico de emergencia. Si se ingresó el correo electrónico y existe conexión a internet, verifica si el sistema está encendido o apagado y cambia de estado a “encendido”. Si no está registrado el correo entonces espera a que ingrese el correo de emergencia. Si no se tiene conexión a internet, espera a que se reestablezca la conexión. Si se activa el sistema, se envía las fotos a la aplicación, se envía la notificación de alarma y se muestra las fotos recibidas del servidor. Si el sistema está desconectado se apaga el sistema.

Por otro lado, si se pulsa el botón fotografías, se redirecciona a la pantalla de cámaras para mostrar las fotos recibidas. Si se pulsa borrar, se borran las fotos del servidor. Luego se pulsa el botón de regresar para volver a la pantalla principal.

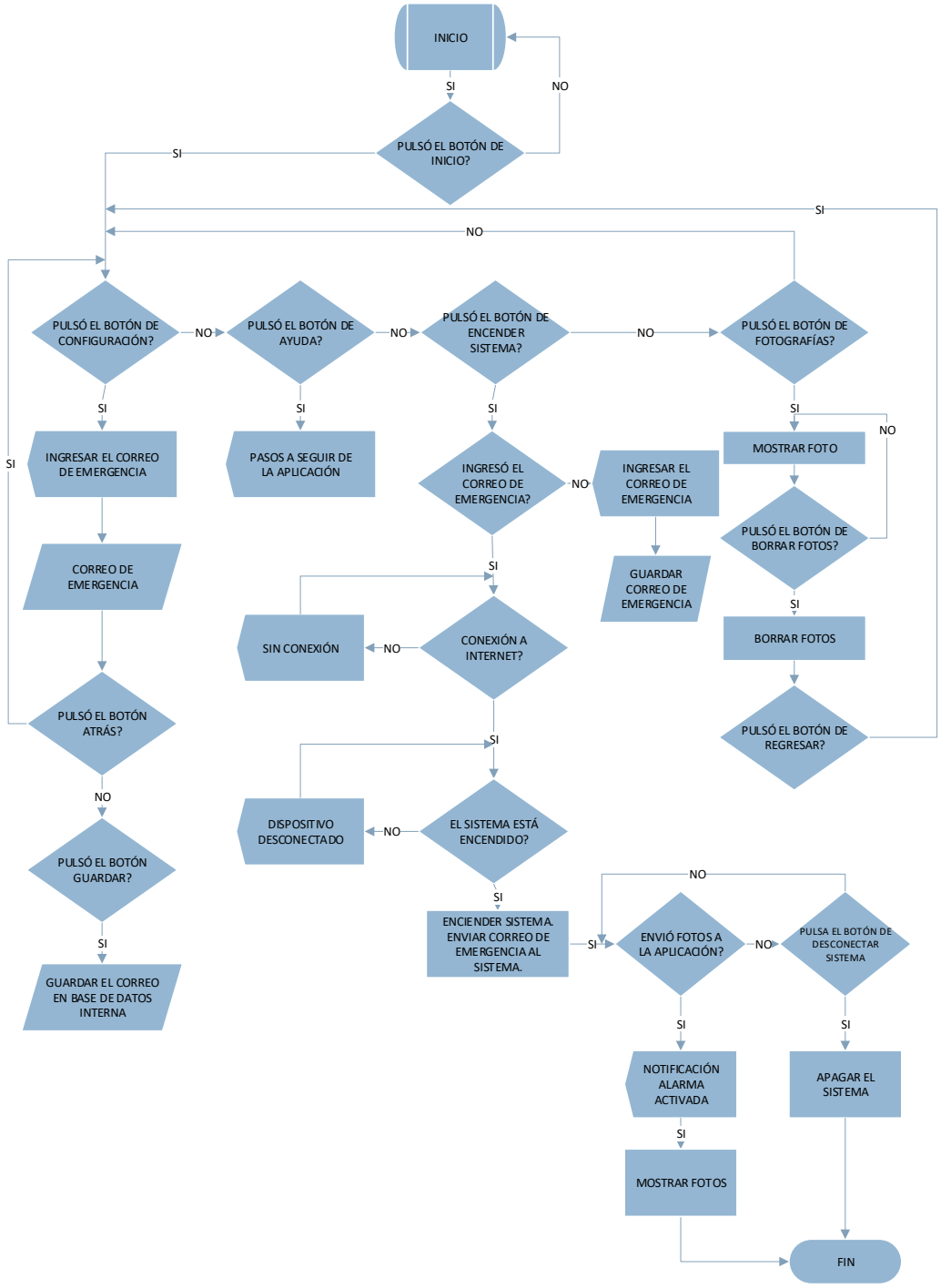


Figura 41 Diagrama de Flujo de la Aplicación Móvil

5. Análisis de Resultados

En este capítulo se muestran los resultados obtenidos a través de diferentes pruebas a las que fue sometido el prototipo desarrollado.

5.1 Análisis del Tiempo de Respuesta

Se realizaron pruebas con el modelo a escala para verificar el tiempo de respuesta del sistema desde que se activa el acelerómetro hasta que se reciben las imágenes en la aplicación móvil y en el correo de emergencia. Los resultados obtenidos se muestran en la Tabla 3.

En la Tabla 3 se muestra en la izquierda las filas de los tiempos de respuesta de referencia del sistema con las velocidades 3G, con internet inalámbrico y el número de pruebas realizadas con el modem 4G. En la parte superior de la tabla se muestra las columnas de tiempos de respuesta desde que se activa el sistema hasta que el dueño del auto recibe en su celular la notificación, desde que se activa el sistema hasta que el propietario del vehículo recibe las fotos en su celular y desde que se activa el sistema hasta que el dueño del auto recibe en su e-mail el correo electrónico con las fotos adjuntas.

Tabla 3

Tiempos de Respuesta del Sistema

	Tiempo de Notificación (seg)	**Tiempo de recepción de fotos (seg)**	**Tiempo de recepción de mail (seg)**
Tiempo de Referencia con velocidad WiFi (100 Mbps)	5	56	56
Tiempo de Referencia de velocidad 3G	63	228	300
Prueba 1 (con 4G)	5	35	35

Prueba 2 (con 4G)	6	36	36
Prueba 3 (con 4G)	5	35	35

Durante las pruebas realizadas también se monitoreó la llegada de las imágenes desde el prototipo hasta la aplicación móvil. Los resultados se muestran en la Figura 42. Específicamente, estas imágenes se muestran en la siguiente secuencia: cámara frontal, cámara posterior, cámara derecha y cámara izquierda. Las imágenes tienen una resolución de 720 pixeles. Además, las imágenes capturadas muestran el entorno del prototipo a color.

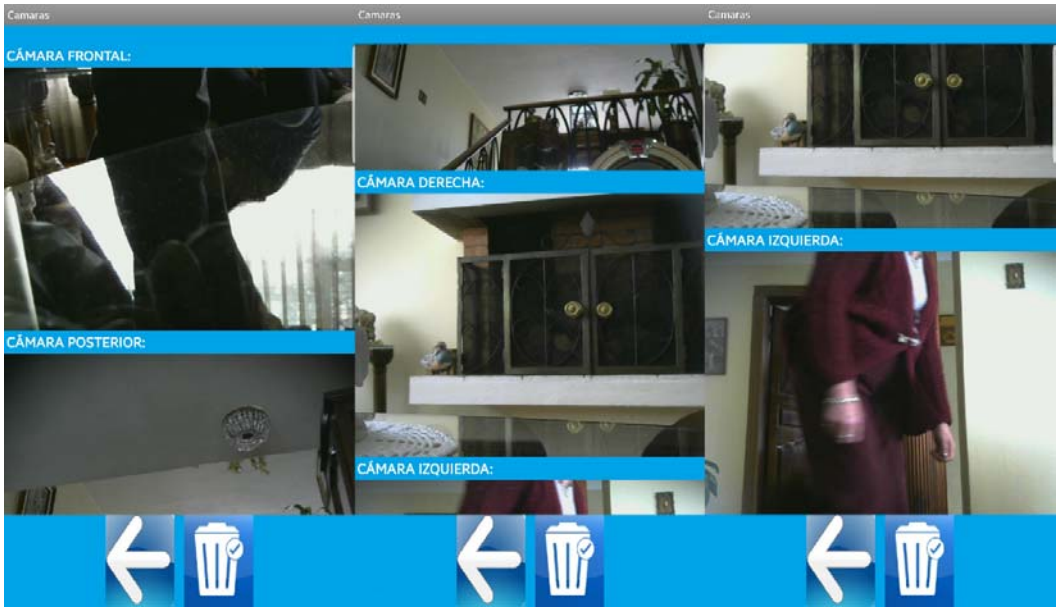


Figura 42 Prueba de Recepción de fotos

5.2 Análisis del Consumo de Datos

Adicionalmente, las pruebas del prototipo incluyeron verificar la llegada del correo electrónico a la dirección del correo de emergencia. Las imágenes tomadas por el prototipo llegaron como archivos adjuntos, en formato .png. El correo recibido tiene además el mensaje “La alarma del carro ha sido activada, verifique las imágenes adjuntas”. El tamaño total de los archivos adjuntos es 1,27 MB entre las 4 fotos. Lo antes mencionado se muestra en las Figuras 43 y 44.

Figura 43 Correo Electrónico de Prueba con Fotos Adjuntas

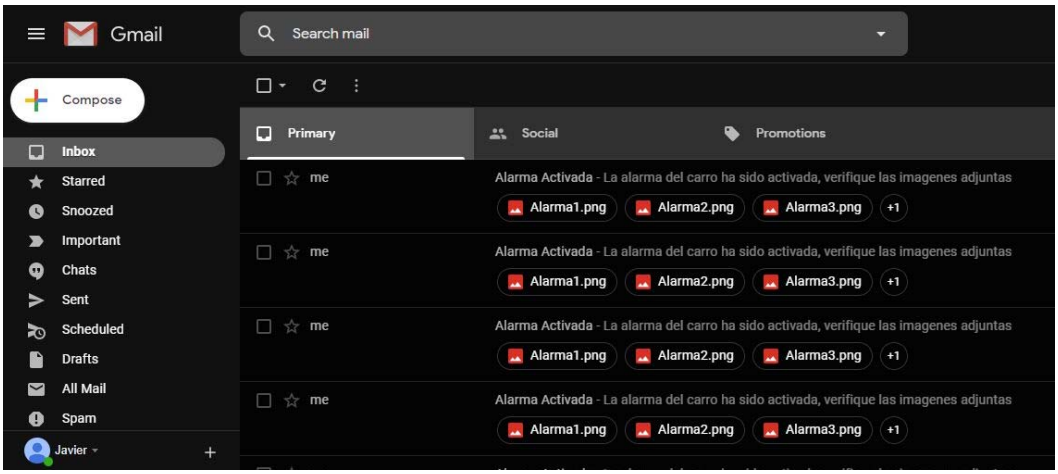


Figura 44 Buzón de Entrada de Dirección de Correo de Emergencia Destinada para Pruebas

Por otro lado, las pruebas del prototipo también proporcionaron información sobre el consumo de datos del sistema, específicamente los datos utilizados por la aplicación de correo electrónico. Como se puede observar en la Figura 45, se descargó el correo electrónico con sus archivos adjuntos y se pudo evidenciar que el correo tiene un tamaño de 1,75 MB. Esto representa un tamaño aceptable ya que las fotos están en formato .png, que posee un alto nivel de compresión. Desde el punto de vista del prototipo, que utiliza un plan de datos celular limitado, se utiliza alrededor de 1,27MB. Este valor de las imágenes adjuntas es un

aproximado en base a una de las pruebas ya que, dependiendo del contraste y nivel de color de cada foto, el tamaño de cada foto puede variar entre los 200 KB y 360 KB.

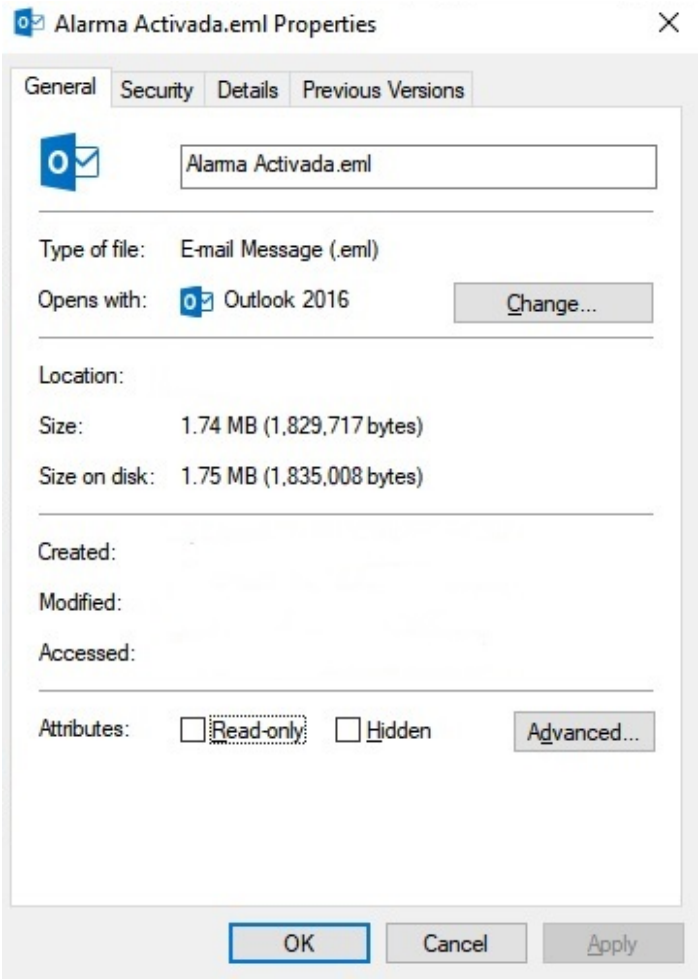


Figura 45 Tamaño total del Correo electrónico de Prueba

5.3 Análisis de Consumo de Energía

Durante las pruebas realizadas con el prototipo, se monitoreo el consumo de energía del mismo. La batería tiene 12 V y 5 Ah, la misma está conectada a un transformador DC-DC que recibe esos 12 V y los transforma en 5 V y 5 A de corriente para dar un total de 25 W. El prototipo puede funcionar con la batería por 20-30 minutos en una sola carga dependiendo del número de veces que se active el sistema.

Sin embargo, como se mencionó en la sección 2.1, la batería necesita ser recargada por lo que se instaló un puerto de carga en el modelo a escala que también funciona como un conector a energía fija. En este conector se inserta un adaptador de corriente que va conectado a la pared y entrega 12V y 2A. De esta manera, la batería se carga y el prototipo puede funcionar al mismo tiempo con energía eléctrica pública.

La microcomputadora requiere 5V y entre 4 A y 6 A para funcionar dependiendo del número de dispositivos conectados a la misma. Por esta razón es que la batería de 12 V y 5 Ah resulta insuficiente para un uso prolongado del sistema. Se considera que para una implementación real el módulo vehicular será conectado al alternador del vehículo y tendrá una batería de 12 V con un mayor amperaje hora para poder funcionar de manera prolongada en caso de haber una pérdida de energía del vehículo.

5.4 Comparación de Tecnologías de Seguridad Vehicular

Los sistemas de seguridad vehicular más populares en el mercado como son el sistema de rastreo por GPS, las llaves codificadas y las alarmas clásicas. Estos sistemas han probado su efectividad para la función para la que fueron diseñados. Entre las funciones de estos sistemas tenemos: Informar en tiempo real el paradero del vehículo a su dueño, en el caso del sistema de rastreo por GPS. Verificar el código de las llaves del propietario como es el caso de las llaves codificadas. Emitir una alerta de sonido para avisar que se ha abierto una puerta o el auto ha sido golpeado como es el caso de la alarma básica.

Sin embargo, a pesar de las funciones ya mencionadas que proveen estos sistemas, no proveen algo que a los dueños de los vehículos les gustaría tener, un set de cámaras que les permita saber qué pasa con su vehículo en tiempo real mientras no están.

5.5 Consideraciones para una Implementación Real

Para una implementación real es necesario tener en cuenta varias consideraciones respecto del módulo vehicular, ya que la configuración del modelo a escala es óptima, pero puede resultar insuficiente en un vehículo real.

Para empezar, el acelerómetro debe ser calibrado de acuerdo a las necesidades de cada cliente o de cada vehículo, ya que la misma configuración no será adecuada para todos los vehículos y para todas las circunstancias. Esta configuración se la realiza en el código fuente del programa en Python, en el Anexo 1, bajo el comentario “#Division del valor obtenido por el factor de escala de sensibilidad”. En dicha sección, se cambia el factor de escala de sensibilidad con lo cual se logra un sistema más o menos sensible a fluctuaciones en la estabilidad del acelerómetro.

Por otro lado, es necesario analizar en donde se ubicarán las cámaras dependiendo del vehículo, requerimientos del cliente, posibilidades mecánicas y eléctricas del auto. La ubicación de las cámaras y módulo podría afectar la interfaz usada ya que el cable USB tiene una distancia máxima de 3 metros. En este aspecto, también es importante considerar donde irá ubicado el módulo vehicular para causar la menor intromisión posible al vehículo y al mismo tiempo lograr los mejores resultados.

Adicionalmente, para una implementación real es necesario integrar el módulo vehicular dentro de un solo componente, incluyendo una batería de polímero de litio para proporcionarle mayor seguridad, resistencia, eficiencia y confiabilidad. De igual manera, al integrar todos los elementos en un solo componente también se reducirán costos, por lo que harán al sistema más accesible y rentable para producción en masa.

Finalmente, el prototipo propuesto hace uso de tecnologías y componentes de código y arquitectura abierta. Esto permite que el sistema completo sea muy flexible y escalable para realizar cambios o mejoras en software y hardware de manera rápida y fácil. En la aplicación móvil se pueden realizar cambios en la interfaz o incorporar más funcionalidades a través de actualizaciones en Google Play. En el servidor de servicios se pueden integrar nuevas prestaciones o modificar las existentes modificando el código PHP. Por último, en el módulo vehicular se puede aumentar el número de cámaras, usar una microcomputadora más avanzada o incorporar otros componentes que proporcionen nuevas o mejores funcionalidades al módulo manteniendo o reduciendo el tiempo de respuesta del sistema.

6. Conclusiones y Recomendaciones

6.1 Conclusiones

Este proyecto hace uso optimizado y eficiente de la microcomputadora *Aaeon UP Squared*, cámaras web integradas a una plataforma vehicular a escala, la red celular de alta velocidad 4G, servidores de aplicación PHP en la nube y aplicaciones móviles para Android desarrolladas en *AppInventor*. El uso optimizado de todos los componentes mencionados permite tener un producto mínimo viable para una futura implementación en un entorno real.

El prototipo propuesto obtuvo buen desempeño basado en las pruebas realizadas y el mismo fue optimizado para que tenga el menor costo posible. Esta optimización se logró Analizando varias opciones de implementación en hardware y software. Así, se propuso un producto con un costo accesible.

El prototipo implementado permite a los propietarios tener una vigilancia constante de sus vehículos debido a la rápida respuesta dada por el módulo vehicular, la cobertura celular de alta velocidad 4G y la aplicación móvil que notifica inmediatamente al propietario de cualquier evento con su vehículo. Logrando así una solución viable y eficiente al problema planteado.

Las tecnologías de seguridad existentes como el GPS, candados, alarmas, llaves codificadas y sistemas de bloqueo son eficaces, sin embargo, no satisfacen la problemática que este proyecto soluciona al proveer una visión periférica del auto en caso de algún incidente con el mismo.

Este sistema es revolucionario, ya que además de solucionar una necesidad existente, tiene infinitas posibilidades de escalabilidad y flexibilidad frente a posibles amenazas no consideradas.

Los tiempos de respuesta medios de 35 segundos logrados en este proyecto pueden ser considerados como aceptables teniendo en cuenta que se trata de un prototipo. Sin embargo, la cobertura celular y las especificaciones de la placa del módulo vehicular fueron un factor importante para poder conseguir dicho tiempo de respuesta.

Los niveles de consumo de datos en el prototipo planteado son reducidos y representan un consumo mínimo tomando en cuenta la probabilidad que el sistema se active en el periodo de un mes. Esto debido a que si se implementa

en un entorno real sería necesario un plan de datos con una operadora celular y esto requería una suscripción mensual al servicio que tienen un numero de MB por mes.

La batería contenida en este proyecto no provee una cantidad de energía para uso prolongado del prototipo. Sin embargo, el módulo vehicular debe funcionar con la batería de un auto real que dispone de mayor energía y un cargador eficiente como es el alternador del vehículo.

6.2 Recomendaciones

Se recomienda aumentar la memoria RAM de la placa a 4GB y usar la versión con procesador *Pentium Quad-Core* de 2.5 GHz, para poder reducir más el tiempo de procesamiento de las imágenes en el módulo vehicular o agregar funcionalidades al prototipo.

El sistema funciona actualmente con una aplicación para sistema operativo Android de acuerdo al alcance de este proyecto. Sin embargo, podría implementarse en iOS de Apple en futuras iteraciones, ya que el servidor de aplicación tiene la flexibilidad de funcionar con los dos sistemas operativos móviles.

Si se hace una implementación masiva, es recomendable incorporar en la placa el módulo 4G, el acelerómetro, mayor número de puertos USB, y de ser posible el transformador de DC-DC, para lograr una placa unificada de fábrica a menor costo y más eficiente.

Se debe tomar en cuenta el utilizar una batería de polímero de litio, que permita mayor cantidad de energía a un menor tamaño de la batería, aunque esto implique un mayor costo, para garantizar el funcionamiento prolongado del módulo vehicular ante una falla del sistema eléctrico del auto. Caso contrario funcionaría con la batería del vehículo.

En futuras implementaciones se recomienda aprovechar de la nueva tecnología 5G para reducir el tiempo de respuesta del sistema o incluso incorporar video en lugar de únicamente imágenes. Además, al tener una velocidad de transmisión de datos mayor, se podrían incorporar otras funcionalidades sin sacrificar tiempo de respuesta.

Debe analizarse de una manera especializada la ubicación de las cámaras en una implementación real, para lograr un mayor campo de visión con un menor riesgo de daño por agentes externos al vehículo.

En una implementación real debe evaluarse si la interfaz USB sigue siendo la mejor opción para las cámaras, ya que dicha conexión tiene una distancia máxima de tres metros.

Si se desea incorporar mayor cantidad de cámaras, se recomienda tomar en cuenta un mejor procesamiento gráfico por parte de la placa. Además de reevaluar las especificaciones de la placa usada para garantizar los tiempos de respuesta del sistema.

Referencias

3GPP.org. (2019). *About 3GPP*. Recuperado el 16 de junio de 2019, de 3GPP.org website: https://www.3gpp.org/about-3gpp

3GPP. (2019). LTE. Recuperado el 12 de julio de 2019, de 3GPP.org website: https://www.3gpp.org/technologies/keywords-acronyms/98-lte

Aaeon. (2018). *UP Square Datasheetv5*. Recuperado el 5 de febrero de 2019, de Aaeon Europe website: www.up-board.org/

Acosta, S., & Herrera, F. (2009). DISEÑO E IMPLEMENTACIÓN DE UNA CENTRAL DE MONITOREO PARA SEGURIDAD ELECTRÓNICA Y LOCALIZACIÓN DE VEHÍCULOS MEDIANTE RADIO BASES. Recuperado el 26 de julio de 2019, de http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/1099/T-ESPE-025967.pdf?sequence=1&isAllowed=y

Agencia Nacional de Tránsito. (2018). SINIESTROS POR PROVINCIA A NIVEL NACIONAL- DICIEMBRE 2018. Recuperado el 26 de julio de 2019, de https://www.ant.gob.ec/index.php/descargable/file/6096-siniestros-diciembre-2018

Alcatel. (2017). 3G+. Recuperado el 22 de mayo de 2019, de Alcatel website: http://www.alcatel-mobile.com/ec/products/detail/X602#.XPfk4oj0mUl

ARC Electronics. (2018). *Configuring Your Modem For Optimum Performance*. Recuperado el 12 de junio de 2019, de arcelect.com website: https://arcelect.com/Configuring-Your-Modem-For-Optimum-Performance.htm

Arduino. (2019). ARDUINO MEGA 2560 REV3. Recuperado el 5 de febrero de 2019, de store.arduino.cc website: https://store.arduino.cc/usa/mega-2560-r3

Bravo, D. (2018). Tres nuevas formas de robos de vehículos se utilizan en Quito. Recuperado el 16 de junio de 2019, de El Comercio website: https://www.elcomercio.com/actualidad/nuevasformas-robos-vehiculos-quito-seguridad.html

CableFree. (2019). *LTE Network Latency compared with 2G, 3G & WiFi*. Recuperado el 16 de junio de 2019, de cablefree.net website: https://www.cablefree.net/wirelesstechnology/4glte/lte-network-latency/

Caivano, D., Cassano, F., Lanzilotti, R., & Piccinno, A. (2018). *Towards an IoT model for the assessment of smart devices*. *Proceedings of the 2018 International Conference on Advanced Visual Interfaces - AVI '18*, 1–3. Recuperado el 26 de julio de 2019, de *ACM Library*: https://doi.org/10.1145/3206505.3206587

CBS Interactive. (2019). *Huawei E173 - wireless cellular modem Specs*. Recuperado el 22 de mayo de 2019, de cnet.com website: https://www.cnet.com/products/huawei-e173-wireless-cellular-modem/

Chamorro, J. E. (2013). Diseño e implementación de sistemas de seguridad para vehículos mediante GPS y telefonía celular (SMS Y DTMF). Recuperado el 6 de mayo de 2019, de http://bibdigital.epn.edu.ec/handle/15000/6458

Circuit Basics. (2016). *BASICS OF UART COMMUNICATION*. Recuperado el 12 de junio de 2019, de circuitbasics.com website: http://www.circuitbasics.com/basics-uart-communication/

Electronic wings. (2019). *MPU6050 (Accelerometer+Gyroscope) Interfacing with Raspberry Pi*. Recuperado el 7 de junio de 2019, de ElectronicWings.com website: https://www.electronicwings.com/raspberry-pi/mpu6050-accelerometergyroscope-interfacing-with-raspberry-pi

electronicaestudio. (2019). Acelerómetro. Recuperado el 16 de junio de 2019, de electronicaestudio.com website: https://www.electronicaestudio.com/tienda/sensores/acelerometro-y-gyro-6-ejes-mpu-6050/

Guayaquil, E. G. (2007). En promedio, diez casos de accidentes ocurren a diario. Recuperado el 16 de junio de 2019, de El Universo.com website: https://www.eluniverso.com/2007/05/20/0001/18/DD1B71B87AE548DA8FBD3DD2A522A117.html

Huawei. (2018). HUAWEI E5573. Recuperado el 22 de mayo de 2019, de

consumer.huawei.com website: https://consumer.huawei.com/ar/mobile-broadband/e5573/specs/

Huawei. (2019). HUAWEI 4G Dongle E3372. Recuperado el 22 de mayo de 2019, de consumer.huawei.com website: https://consumer.huawei.com/en/mobile-broadband/e3372/specs/

Invensense. (2013). *MPU-6050 Datasheet (PDF) - TDK Electronics.* Recuperado el 7 de junio de 2019, de Alldatasheet.com website: http://pdf1.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html

Joshi, P., Colombi, D., Thors, B., Larsson, L.-E., & Tornevik, C. (2017). *Output Power Levels of 4G User Equipment and Implications on Realistic RF EMF Exposure Assessments*. Recuperado el 7 de junio de 2019, de *IEEE Access*, *5*, 4545–4550. https://doi.org/10.1109/ACCESS.2017.2682422

Kavanagh, S. (2018). *5G vs 4G: No Contest*. Recuperado el 16 de junio de 2019, de 5g.co.uk website: https://5g.co.uk/guides/4g-versus-5g-what-will-the-next-generation-bring/

Libelium. (2016). *4G + GPS Shield for Arduino and Raspberry Pi Tutorial (LTE / WCDMA / HSPA+ / 3G / GPRS)*. Recuperado el 6 de mayo de 2019, de Cooking Hacks website: https://www.cooking-hacks.com/documentation/tutorials/4g-gps-lte-wcdma-hspa-3g-gprs-shield-arduino-raspberry-pi-waspmote-tutorial/

Libelium blog. (2016). *Choosing the right cellular module for Arduino and Raspberry Pi: 4G / 3G / GPRS / GSM*. Recuperado el 5 de mayo de 2019, de Cooking Hacks website: https://www.cooking-hacks.com/blog/choosing-the-right-cellular-module-for-arduino-and-raspberry-pi-4g-3g-gprs-gsm/

López Villegas, D. (2014). IDE: Entornos Integrados de Desarrollo para Android. Recuperado el 12 de julio de 2019, de Academia Android website: https://academiaandroid.com/ide-entornos-integrados-de-desarrollo-para-android/

Lucio, M. A. (2016). 5 mejores dispositivos antirrobo. Recuperado el 16 de junio de 2019, de ecuador.seguros123.com website: https://ecuador.seguros123.com/5-mejores-dispositivos-antirrobo/

Massachusetts Institute of Technology. (2019). *Hour of Code with MIT App Inventor*. Recuperado el 12 de julio de 2019, de http://appinventor.mit.edu website: http://appinventor.mit.edu/explore/hour-of-code.html

Morgan, J. (2014). *A Simple Explanation Of "The Internet Of Things".* Recuperado el 5 de febrero de 2019, de Forbes website: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#6665227f1d09

net-informations.com. (2019). *What are the drawbacks of Python?* Recuperado el 5 de abril de 2019, de net-informations.com website: http://net-informations.com/python/iq/disadvantages.htm

Nvidia. (2019). Nvidia Jetson. Recuperado el 12 de junio de 2019, de developer.nvidia.com website: https://developer.nvidia.com/embedded/develop/hardware

php.net. (2019). *What is PHP?* Recuperado el 5 de abril de 2019, de php.net website: https://www.php.net/manual/en/intro-whatis.php

Raspberry Pi Foundation. (2016). Raspberry Pi 3 Model B. Recuperado el 6 de mayo de 2019, de Raspberry Pi Website website: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Revsky01. (2018). ¿Cómo se hace en linux para ejecutar un archivo python al arrancar la máquina? Recuperado el 10 de junio de 2019, de Stackoverflow.com website: https://es.stackoverflow.com/questions/150343/cómo-se-hace-en-linux-para-ejecutar-un-archivo-python-al-arrancar-la-máquina

Reyes Huertas, J. D., & Huertas Vitery, M. F. (2014). Diseño e Implementacion de un Modulo Alarma Para el Monitoreo y Control del Vehiculo a Traves del Sistema GSM y GPS. Recuperado el 26 de julio de 2019, de http://repositorio.espe.edu.ec/handle/21000/9270

Technopedia.com. (2018). *Single-Board Computer* (SBC). Recuperado el 16 de junio de 2019, de technopedia.com website: https://www.techopedia.com/definition/9266/single-board-computer-sbc

Technopedia.com. (2019). *Long Term Evolution* (LTE). Recuperado el 16 de junio de 2019, de technopedia.com website: https://www.techopedia.com/definition/8149/long-term-evolution-lte

The PHP Group. (2019). *What can PHP do?* Recuperado el 5 de abril de 2019, de php.net website: https://www.php.net/manual/en/intro-whatcando.php

TR4NSDUC7OR. (2014). Tutorial de Arduino y MPU-6050. Recuperado el 7 de junio de 2019, de robologs.net website: https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

Universidad Internacional de Valencia. (2018). Evolución de la red de comunicación móvil, del 1G al 5G. Recuperado el 16 de junio de 2019, de Universidad Internacional de Valencia website: https://www.universidadviu.com/evolucion-la-red-comunicacion-movil-del-1g-al-5g/

up-community. (2017). Pinout UP2. Recuperado el 12 de junio de 2019, de wiki.up-community.org website: https://wiki.up-community.org/Pinout_UP2

up-community Ubuntu. (2017). Ubuntu. Recuperado el 12 de junio de 2019, de wiki.up-community.org website: https://wiki.up-community.org/Ubuntu

w3schools. (2019). *Python Introduction*. Recuperado el 5 de abril de 2019, de w3schools.com website: https://www.w3schools.com/python/python_intro.asp

ANEXOS

Anexo 1. Código Fuente en Python del Módulo Vehicular

```
#Instrucción para inicio automático de la aplicación al iniciar ubuntu
#!/usr/bin/python
#libreria de hora
import time
#Verificacion de conexion a Wifi
import requests
#Libreria para codificar la imagen
import base64
#Librerias para enviar los datos mediante PHP
from urllib.parse import urlencode
from urllib.request import Request,urlopen
#Libreria de la camara
import pygame
import pygame.camera
#Libreria para fecha y hora
from datetime import datetime
#Librerias para enviar los datos mediante PHP
from urllib.parse import urlencode
from urllib.request import Request,urlopen
#Libreria del Acelerometro
import smbus                    #import SMBus module of I2C
#Librerias para el correo electronico
import smtplib
import mimetypes
#Importamos los modulos necesarios para el correo
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
from email.mime.text import MIMEText
#Libreria de entrada y salida de la AAEON
import mraa
```

```
#Inicializacion de la camara
pygame.init()
pygame.camera.init()
camara="/dev/video"

#Pines de entrada y salida
#Pulsador
pulsador = mraa.Gpio(8)
pulsador.dir(mraa.DIR_IN)
#Led Rojo de indicación de estado de sistema, si titila esta en standby, si esta solo rojo esta encendido
led = mraa.Gpio(12)
led.dir(mraa.DIR_OUT)
led.write(0)
#Led Verde de indicación de estado de la placa, si encendida o apagada
ledi = mraa.Gpio(16)
ledi.dir(mraa.DIR_OUT)
ledi.write(1)
time.sleep(1)

#Direcciones de registro necesarios del acelerometro MPU6050
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47
```

#Variables utilizadas

```
encendido=0
aux_correo=""
aux_estadoapp=""
correo ="tesisjv2019@gmail.com"
estadoapp=""
valor_anterior=0
valor_actual=0
valor_actuals=""
aux_resp=0
cont = 0
e_led=1
alarma_anterior=0
alarma_actual=1
estado="desactivado"
contador=0
```

###FUNCIONES

#Funcion para inicializar el MPU

```
def MPU_Init():
        bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)          #Escribe
en el registro de frecuencia de muestreo
        bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)          #Escribe
en el registro de administracion de energia
        bus.write_byte_data(Device_Address, CONFIG, 0)              #Escribe en
el registro de configuracion
        bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)        #Escribe
en el registro de la configuracion del giroscopio
        bus.write_byte_data(Device_Address, INT_ENABLE, 1)          #Escritura
para interrumpir el registro de habilitacion
```

```
#Funcion para leer la direccion del MPU
def read_raw_data(addr):
        #Los valores del acelerometro y giroscopio son de 16 bits
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)
        value = ((high << 8) | low)        #Se concatena el valor mayor y menor
        #Para obtener el valor de la señal del MPU6050
        if(value > 32768):
                value = value - 65536
        return value

#Funcion para obtener los valores del acelerometro
def acelerometro (cont, correo):
        global file
        global msg

        #Lectura de los valores del acelerometro obtenidos
        acc_x = read_raw_data(ACCEL_XOUT_H)
        acc_y = read_raw_data(ACCEL_YOUT_H)
        acc_z = read_raw_data(ACCEL_ZOUT_H)
        #Division del valor obtenido por el factor de escala de sensibilidad
        Ax = acc_x/16384.0
        Ay = acc_y/16384.0
        Az = acc_z/16384.0
        print ("\tAx=%.2f g" %Ax, "\tAy=%.2f g" %Ay, "\tAz=%.2f g" %Az)
        if (Ax > 1.03 or Ax < 1.00) and (Ay > 0.07 or Ay < 0.03) and (Az < -0.25 or
Az > -0.20):
                alarma_actual=1
                print ("Alarma activada")
                tomar_foto(cont, correo)
#Funcion para tomar las fotos
def tomar_foto(cont, correo):
```

```
camara="/dev/video"
#Se crea un objeto multipart el cual contenido que se enviara
msg =MIMEMultipart()
msg['From']= "tesisjv2019@gmail.com"
msg['To']=correo
msg['Subject']="Alarma Activada"
msg.attach(MIMEText("La alarma del carro ha sido activada, verifique las imagenes adjuntas"))

while cont < 4:
    camara = camara+str(cont)
    if cont == 0:
        nom="cam_iz"
    elif cont == 1:
        nom="cam_pos"
    elif cont == 2:
        nom="cam_fron"
    elif cont == 3:
        nom="cam_der"
    cam = pygame.camera.Camera(camara,(640,480))
    time.sleep(1)
    cam.start()
    time.sleep(1)
    print("Tomando Foto...")
    captura_imagen = cam.get_image()
    print ("Foto Tomada")
    #Obtiene la fecha y hora
    fecha=str(datetime.now())
    fecha_foto=fecha[0:4]+fecha[5:7]+fecha[8:10]+"_"+fecha[11:13]+f
echa[14:16]+fecha[17:19]
    #Coloca el nombre a la imagen
    nombre_foto= nom+fecha_foto
```

```
        file="/home/uphost/Escritorio/fotos/"+nombre_foto+".png"
        pygame.image.save(captura_imagen,file)
        cam.stop()
        print ("Foto Guardada")
        #Verificacion conexion
        conexion=verificacionConexion()
        #print (conexion)
        if conexion:
        if cont == 0:
                print ("envio de datos")
                url = 'http://alluc.com.ec/CarroTesis/carroseguridad.PHP'
                alarma_actual=1
                datos_enviar = {'aviso':"encendido",'alarma':alarma_actual}
                peticion = Request(url, urlencode(datos_enviar).encode())
                respuesta = urlopen(peticion).read().decode()
        #Envio de imagenes al servidor
        with open (file, "rb") as imageFile: imagen_en_base64 =
base64.b64encode(imageFile.read()).decode('utf-8')
        url = 'http://alluc.com.ec/CarroTesis/carroseguridad.PHP'
        nombre_camara=nom+".png"
        datos_enviar = {'foto':imagen_en_base64,
'nombre':nombre_camara}
        print ("Enviando foto....")
        peticion = Request(url, urlencode(datos_enviar).encode())
        respuesta = urlopen(peticion).read().decode()
        print ("Guardada con éxito")
        enviar_correo(file, msg,correo,cont)

#Variable de inicializacion
cont = cont+1
camara="/dev/video"
```

```
if cont == 4:
        cont = 0
def enviar_correo(file, msg,correo, cont):
        print ("Correo enviando")
        #Envio de imagenes al correo
        file1 = open(file,"rb")
        attach_image = MIMEImage(file1.read())
        attach_image.add_header('Content-Disposition','attachment;
filename="Alarma"')
        msg.attach(attach_image)
        #Autenticamos
        mailServer = smtplib.SMTP('smtp.gmail.com', 587)
        mailServer.ehlo()
        mailServer.starttls()
        mailServer.ehlo()
        mailServer.login("tesisjv2019@gmail.com", "tesiscarro2019")
        #Envio del correo
        if (cont ==3):
                mailServer.sendmail("tesisjv2019@gmail.com", correo,
        msg.as_string())
        #Se cierra la conexion
        mailServer.close()

def verificacionConexion():
        url='http://www.google.com/'
        timeout = 3
        try:
                _=requests.get(url,timeout=timeout)
                #print("Conectado a Internet")
                return True
        except requests.ConnectionError:
                #print("No conectado a Internet")
```

```
                return False

def activacion_led (e_led):
        if e_led ==1:
                led.write(1)
                time.sleep(0.5)
                led.write(0)
                time.sleep(0.5)
        if e_led == 2:
                led.write(1)

def wifiActivado():
        global correo
        global estadoapp
        global valor_actuals
        global aux_resp
        global aux_correo
        global aux_estadoapp
        global valor_anterior
        global encendido
        aux_resp=0

        url = 'http://alluc.com.ec/CarroTesis/carroseguridad.PHP'
        datos_enviar = {'dato':"enviar"}
        peticion = Request(url, urlencode(datos_enviar).encode())
        respuesta = urlopen(peticion).read().decode()
        #print (len(respuesta))
        #print (respuesta)
        for i in range (len(respuesta)):
                if respuesta[i:i+1]=='&':
                        aux_resp= aux_resp+1
                if aux_resp == 0:
```

```
                valor_actuals =valor_actuals +respuesta[i:i+1]
                valor_actual=int(valor_actuals)
            if aux_resp == 1:
                if respuesta[i:i+1]!="&":
                    aux_correo=aux_correo+respuesta[i:i+1]
            if aux_resp == 2:
                if respuesta[i:i+1]!="&":
                    aux_estadoapp=aux_estadoapp+respuesta[i:i+1]
            activacionPulso()
        if valor_actual > valor_anterior:
            correo=aux_correo
            estadoapp=aux_estadoapp
            valor_anterior = valor_actual
            if estadoapp == "activado":
                encendido = 1
            elif estadoapp == "desactivado":
                encendido = 0
        #time.sleep(3)
        aux_resp=""
        aux_correo=""
        aux_estadoapp=""
        valor_actuals=""

def activacionPulso():
        global encendido
        global enc_led
        global correo

        estado_pulso = pulsador.read()
        #print (estado_pulso)
        if estado_pulso == 1:
            #print ("pulso")
```

```
                if encendido == 1:
                        encendido = 0
                        enc_led = 1
                        estado="desactivado"
                elif encendido == 0:
                        encendido = 1
                        enc_led = 2
                        estado="activado"
                time.sleep(1.5)
                conexion=verificacionConexion()
                #print (conexion)
                if conexion:
                        time.sleep(0.2)
                        url = 'http://alluc.com.ec/CarroTesis/carroseguridad.PHP'
                        datos_enviar = {'dato1':"recibe",'correo':correo,
'estado':estado}
                        peticion = Request(url, urlencode(datos_enviar).encode())
                        respuesta = urlopen(peticion).read().decode()
                else:
                        time.sleep(1)

##Iniciar acelerometro
bus = smbus.SMBus(5)    # i2c 1 en la Aeeon
Device_Address = 0x68   # Direccion de la direccion del MPU6050
MPU_Init()              #Inicializacion MPU6050


while True:
        global encendido
        global correo
        global estadoapp
```

```
        activacionPulso()
        activacion_led(e_led)

        if encendido == 1:
                e_led = 2
                acelerometro(cont, correo)
        elif encendido == 0:
                e_led = 1
        contador = contador+1
        #print (contador)
        if (contador==5):
                conexion = verificacionConexion()
                contador=0
                #print (conexion)
                if conexion:
                        #print ("Encendido")
                        wifiActivado()
```

(Electronic wings, 2019; TR4NSDUC7OR, 2014; up-community, 2017; up-community Ubuntu, 2017)

Anexo 2. Código fuente en PHP del Servidor de Aplicación.

```
<?PHP
$link=mysql_connect("localhost","alluccom_tesisca","tesiscarro") or die
(mysql_error());
mysql_set_charset("utf8");
mysql_select_db("alluccom_CarroTesis") or die(mysql_error());

$estadoapp = $_GET['estadoapp'];
$correo =$_GET ['correo'];
$boton=$_GET['boton'];
$dato=$_POST['dato'];
$dato1=$_POST['dato1'];
$aviso=$_POST['aviso'];
$apagar=$_GET['apagar'];

if (isset($_POST["nombre"])){
        $nombre_foto=$_POST["nombre"];
}
if(isset($_POST["foto"])){
        $bytes = file_put_contents($nombre_foto,
base64_decode($_POST["foto"]));
        echo json_encode($bytes);
}

if($estadoapp == "activado"){
        $query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES
(NULL,'$correo','activado')";
        $result = mysql_query($query);
        echo "Sistema Activado";
}

if($estadoapp == "desactivado"){
```

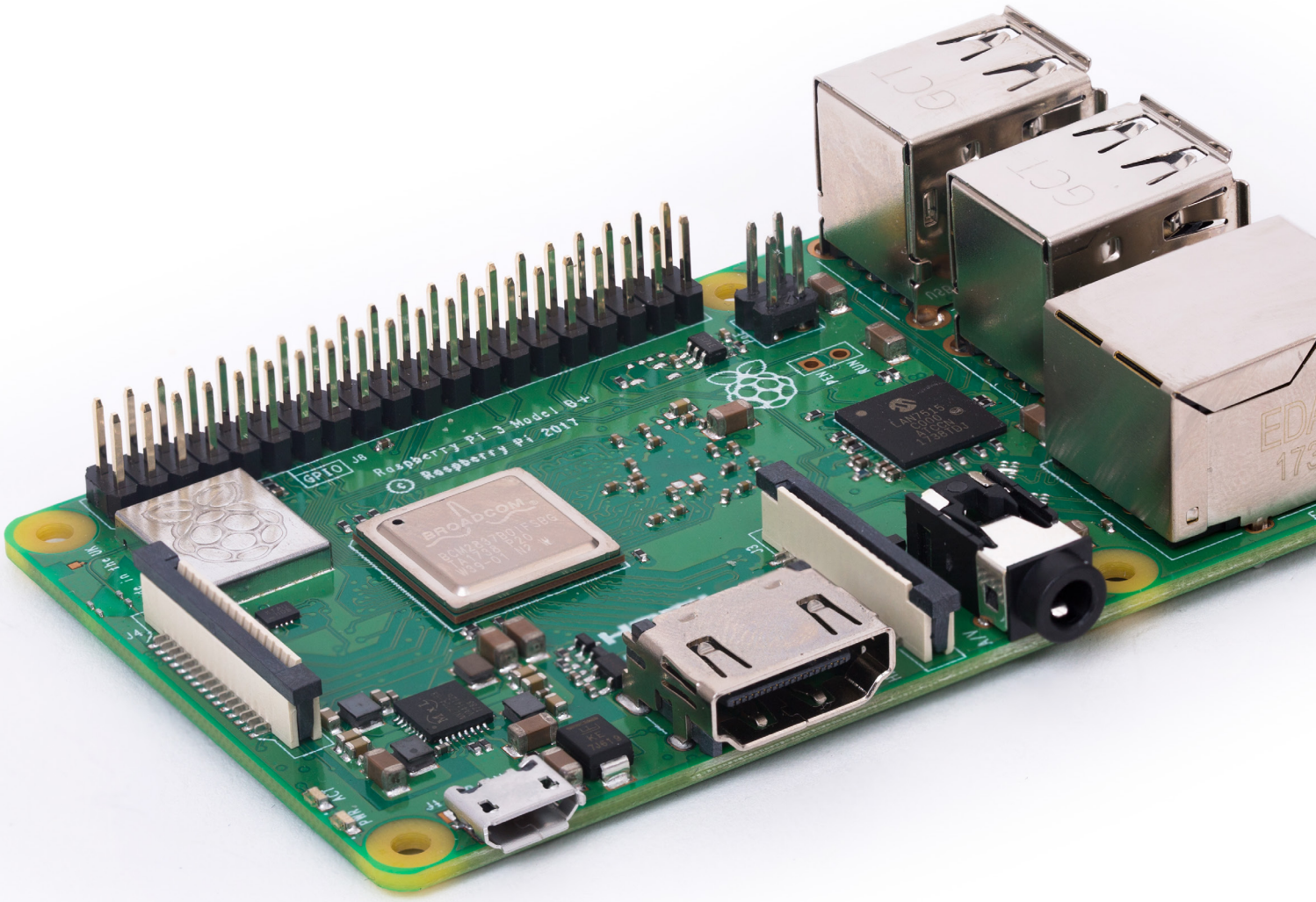
```
	$query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES
(NULL,'$correo','desactivado')";
	$result = mysql_query($query);
	echo "Sistema Desactivado";
}

if ($dato == "enviar"){
	$result = mysql_query("SELECT MAX(ID) FROM Datos", $link);
	if ($row = mysql_fetch_array($result)){
		$id = trim($row[0]);
		print ($id);
	}
	$resultado2 = mysql_query("SELECT * FROM `Datos` WHERE `ID`
LIKE '$id'", $link);
		if ($row = mysql_fetch_array($resultado2)){
		printf("&%s&%s&", $row[1], $row[2]);
		}
}
if ($boton == "recibir"){
	$result = mysql_query("SELECT MAX(ID) FROM Datos", $link);
	if ($row = mysql_fetch_array($result)){
		$id = trim($row[0]);
	}
	$resultado2 = mysql_query("SELECT * FROM `Datos` WHERE `ID`
LIKE '$id'", $link);
		if ($row = mysql_fetch_array($resultado2)){
		print($row[2]);
		}
}

if ($dato1=="recibe"){
	$correo1 = $_POST['correo'];
```

```
	$estado =$_POST['estado'];
	$query = "INSERT INTO `Datos` (`ID`, `correo`, `estadoapp`) VALUES
(NULL,'$correo1','$estado')";
	$result = mysql_query($query);
}

if ($aviso=="encendido"){
	$alarma=$_POST['alarma'];
	print ($alarma);
	$query = "INSERT INTO `Sistema` (`ID`, `Alarma`,`Fecha`) VALUES
(NULL,'$alarma', CURRENT_TIMESTAMP)";
	$result = mysql_query($query);
}

if ($boton=="alarma"){
	$result = mysql_query("SELECT MAX(ID) FROM Sistema", $link);
	if ($row = mysql_fetch_array($result)){
		$id = trim($row[0]);
	}
	$resultado2 = mysql_query("SELECT * FROM `Sistema` WHERE `ID`
LIKE '$id'", $link);
		if ($row = mysql_fetch_array($resultado2)){
		printf("%s&",$row[1]);
		}
}

if ($apagar=="apagado"){
	$query = "INSERT INTO `Sistema` (`ID`, `Alarma`,`Fecha`) VALUES
(NULL,'0', CURRENT_TIMESTAMP)";
	$result = mysql_query($query);
}
mysql_close($link);
```

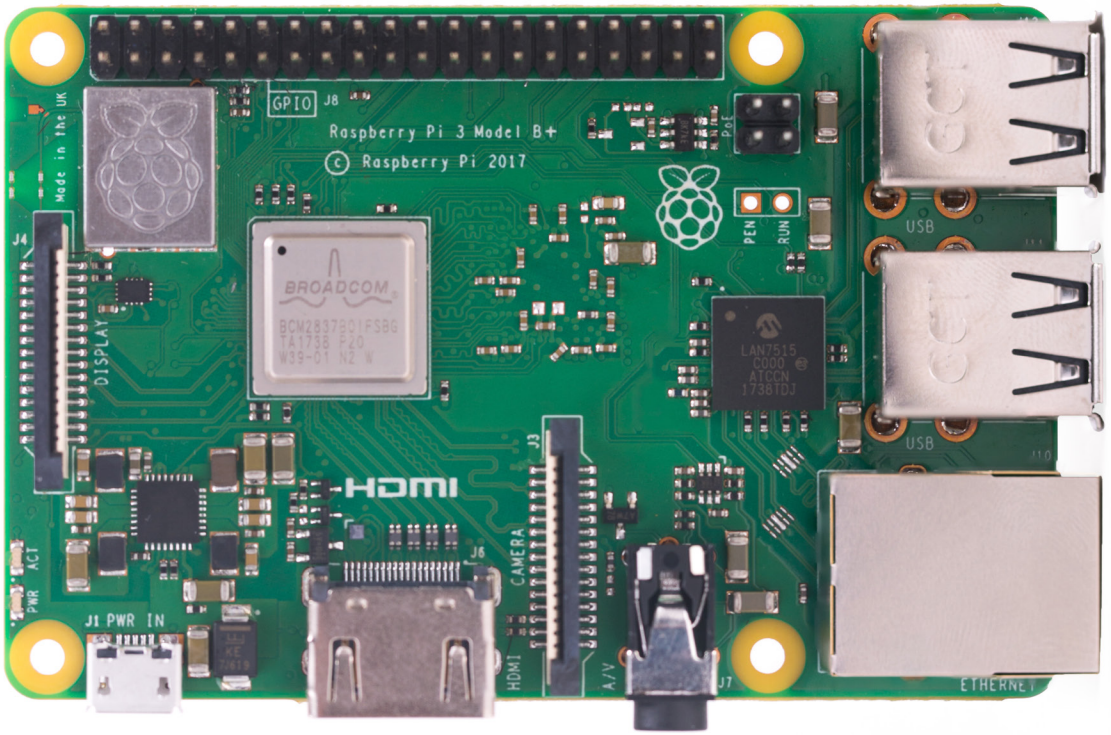
?>

Anexo 2. Datasheet de Raspberry Pi 3 Modelo B+ (Raspberry Pi Foundation, 2016)

Raspberry Pi 3 Model B+



Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:
Broadcom BCM2837B0, Cortex-A53
64-bit SoC @ 1.4 GHz

Memory:
1GB LPDDR2 SDRAM

Connectivity:
- 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- 4 × USB 2.0 ports

Access:
Extended 40-pin GPIO header

Video & sound:
- 1 × full size HDMI
- MIPI DSI display port
- MIPI CSI camera port
- 4 pole stereo output and composite video port

Multimedia:
H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics

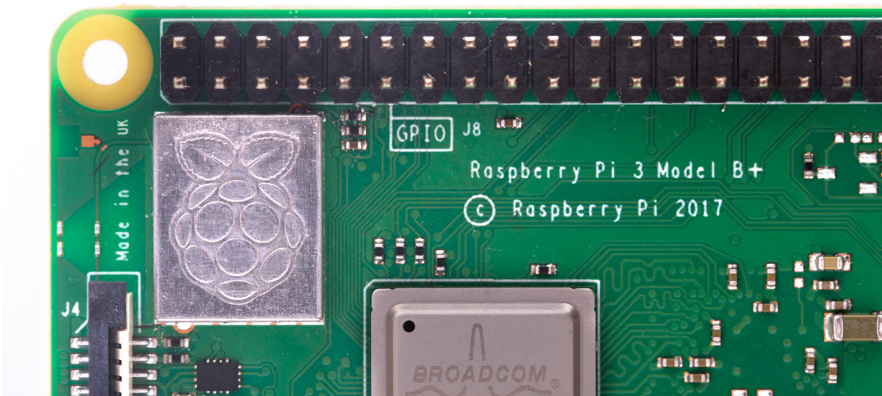
SD card support:
Micro SD format for loading operating system and data storage

Input power:
- 5 V/2.5 A DC via micro USB connector
- 5 V DC via GPIO header
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)

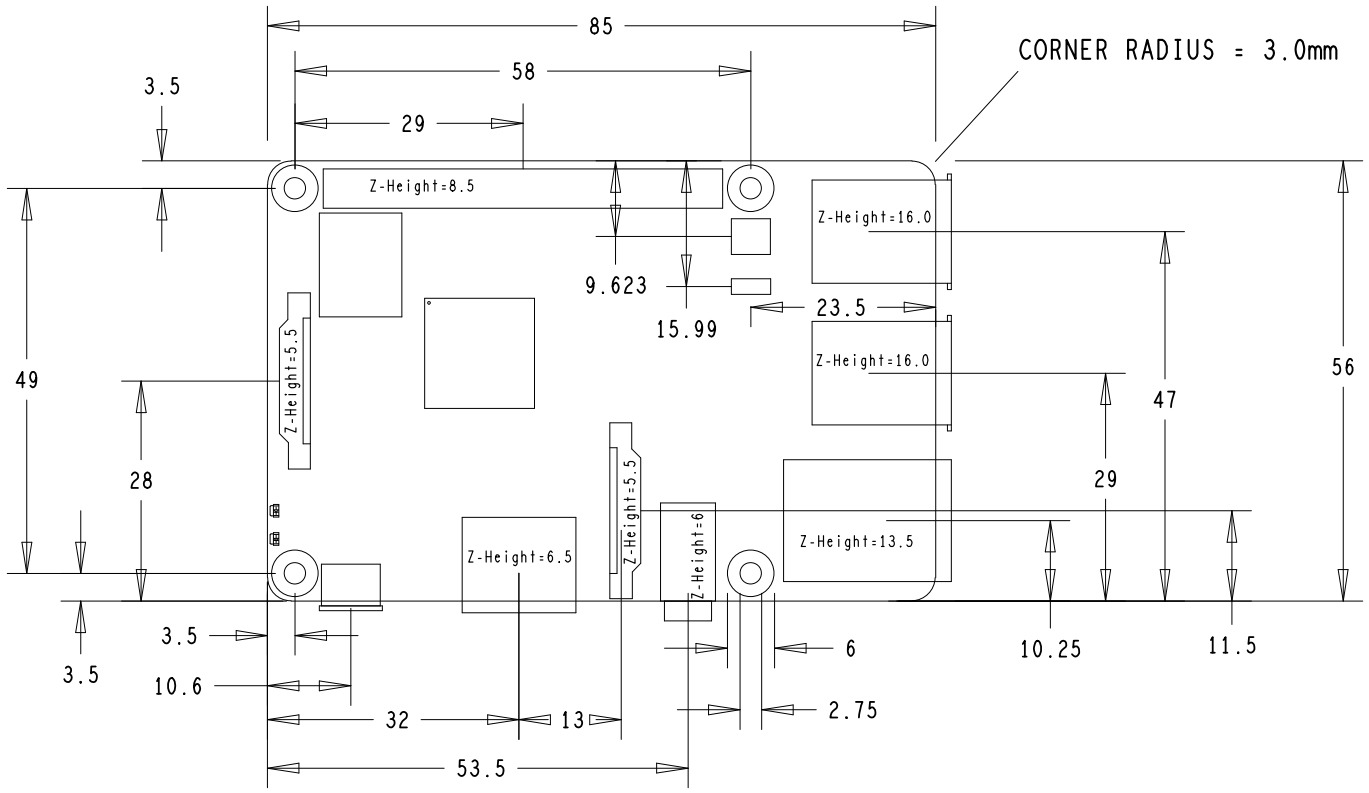
Environment:
Operating temperature, 0–50 °C

Compliance:
For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+

Production lifetime:
The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5 V/2.5 A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

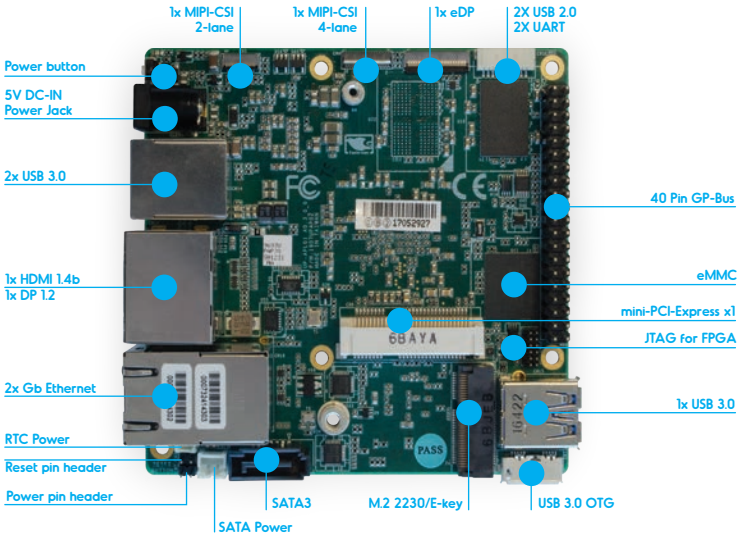
Safety instructions

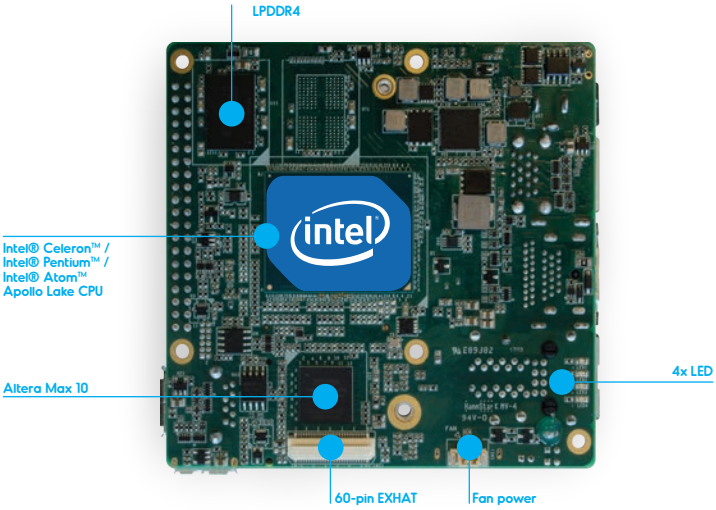
To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

Anexo 3. Datasheet de Placa Aaeon UP Squared. (Aaeon, 2018)

UP² specification





UP² (UP Squared) is world's fastest maker board with the high performance and low power consumption features of **Intel® Celeron™, Pentium™ and Atom™ Processors (codename Apollo Lake).**

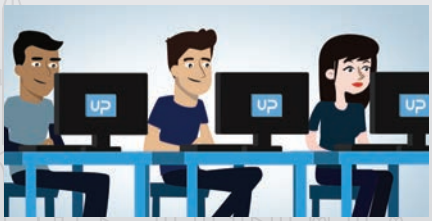
The internal GPU is the new **Intel Gen 9 HD with 12 / 18 Execution Units,** supporting **4K Codec Decode and Encode** for HEVC⁴, H.264 and VP8.
Thanks to the Vector Units Image Processing Unit and Precision Timing Management to synchronize CPU with I/O, improved determinism (cache QoS, Intel Virtualization Technology), all the graphic processing is effortless to UP² (UP Squared).

UP² (UP Squared) comes with **2GB/4GB/8GB LPDDR4** and **32GB/64GB/128GB eMMC**. A **40-pin GP-bus** provides the freedom for makers to build up their module. Additionally, there is a **60-pin EXHAT** for embedded applications. This allows for the exploration of more possibilities. The expansion capabilities of UP² (UP Squared) goes much further than this. Native **mini-PCI-e, M.2 2230 and SATA3** are all built in on the board. What more could one desire?

The board supports **Windows 10, Windows IoT Core, Ubilinux, Ubuntu, Yocto and Android Marshmallow**. It's really UP to you to decide which operating system is best for your application. Now, all you need is an UP² (UP Squared) to begin your project!

UP - Applications

Drones



Education

Robotics

Media Center

Internet of Things

Home Automation

UP² - Specifications

SOC
Intel® Celeron™ N3350 (up to 2.4 GHz)
Intel® Pentium™ N4200 (up to 2.5 GHz)
Intel® Atom ™ E3940 (up to 1.8Ghz)

Graphics
Intel® Gen 9 HD, supporting 4K Codec Decode and Encode for HEVC4, H.264, VP8

Video & Audio
HDMI 1.4b x1 4K @ 30 hz + DP 1.2 4K @ 60 hz
I2S audio port

Camera interface
MIPI-CSI2 2-lane (2MP) + MIPI-CSI2 4-lane (8MP)

Display interface
eDP

Power
5V DC-in @ 4A-6A

Operating humidity
0% ~ 90% relative humidity, non-condensing

Operating Tempature
32-140°F / 0~60°C

Altera MAX 10 FPGA
2KLE --Celeron/ Pentium
4KLE -- ATOM

Memory
2GB (single channel) LPDDR4
4GB/8GB (dual channel) LPDDR4

Storage Capacity
32 GB / 64 GB / 128 GB eMMC

USB
3x UB3.0 (Type A) + 1x USB 3.0 OTG (Micro B)
2x USB2.0+2 X UART (Tx/Rx) debug port (pin header)

Ethernet
2x Gb Ethernet (full speed, Realtek 8111G) RJ-45

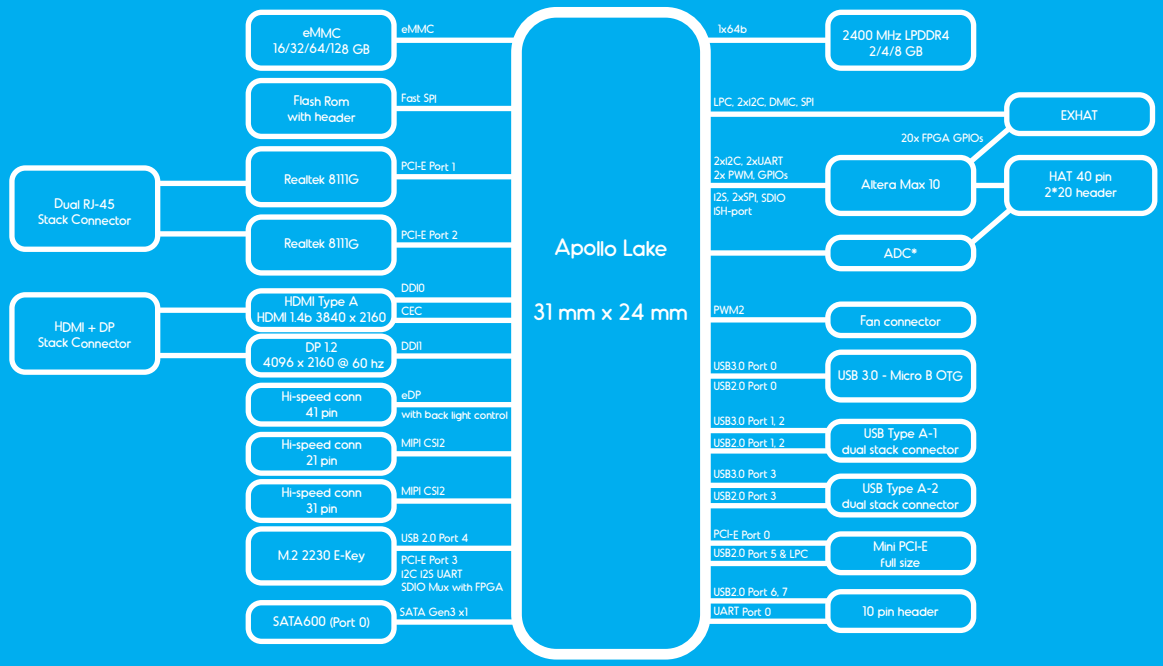
RTC
Yes

Expansion
40 pin General Purpose bus + 4-channel 12-bit A/D converter (500 ksps to 1 Msps)
60 pin EXHAT
1xmini-PCIe (full-size, auto switch to m-SATA)
M.2 2230, SATA3

Compatible Operating system
Microsoft Windows 10 (full), Windows IOT Core, Linux (ubilinux, Ubuntu, Yocto), Android Marshmallow

Dimensions
3.37" x 3.54" / 85.60 mm × 90 mm

Certificate
CE/FCC Class A, RoHS compliant, REACH



UP² - Pinout

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39

1 3V3	2 5V	3 GPIO0/ I2C1_SDA	4 5V	5 GPIO1/ I2C1_SCL
6 Ground	7 GPIO2/ ADC*_in1	8 GPIO15/ UART_TXD	9 Ground	10 GPIO16/ UART_RXD
11 GPIO3/ UART_RTS/ SPI_2_FS1*/ ADC*_in2	12 GPIO17/ I2S_BCLK/ SPI_2_FS0*	13 GPIO4/ ADC*_in3	14 Ground	15 GPIO5/ ADC**_in4
16 GPIO18	17 3V3	18 GPIO19	19 GPIO6/ SPI_1_TXD	20 Ground
21 GPIO7/ SPI_1_RXD	22 GPIO20	23 GPIO8/ SPI_1_CLK	24 GPIO21/ SPI_1_FS0	25 Ground
26 GPIO22/ SPI_1_FS1	27 GPIO9/ I2C0_SDA	28 GPIO23/ I2C0_SCL	29 GPIO10	30 Ground
31 GPIO11	32 GPIO24/ PWM0	33 GPIO12/ PWM1	34 Ground	35 GPIO13/ I2S_WS_SYNC/ SPI_2_RXD*
36 GPIO25/ UART_CTS/ SPI_2_FS2*	37 GPIO14	38 GPIO26/ I2S_SDI/ SPI_2_TXD*	39 Ground	40 GPIO27/ I2S_SDO/ SPI_2_CLK*

* 2nd SPI and ADC will be available only with E3940 SoC

Part number :

UPS-APLC2-A10-0232	Intel® Celeron™ N3350 - 2 GB + 32 GB eMMC
UPS-APLC2-A10-0432	Intel® Celeron™ N3350 - 4 GB + 32 GB eMMC
UPS-APLX5-A10-0432	Intel® ATOM™ x5-E3940 - 4GB DDR4 + 32GB eMMC
UPS-APLP4-A10-0432	Intel® Pentium™ N4200 - 4GB DDR4 + 32GB eMMC
UPS-APLP4-A10-0864	Intel® Pentium™ N4200 - 8GB DDR4 + 64GB eMMC
UPS-APLP4-A10-08128	Intel® Pentium™ N4200 - 8GB DDR4 + 128GB eMMC

www.up-board.org/ www.up-shop.org /www.up-community.org

Anexo 3. Datasheet de Acelerómetro MPU6050. (Invensense, 2013)

InvenSense Inc.
1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A.
Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104
Website: www.invensense.com

Document Number: RM-MPU-6000A-00
Revision: 4.2
Release Date: 08/19/2013

MPU-6000 and MPU-6050
Register Map and Descriptions
Revision 4.2

CONTENTS

1 Revision History

Revision Date	Revision	Description
11/29/2010	1.0	Initial Release
04/20/2011	1.1	Updated register map and descriptions to reflect enhanced register functionality.
05/19/2011	2.0	Updates for Rev C silicon: Edits for readability (section 2.1) Edits for changes in functionality (section 3, 4.4, 4.6, 4.7, 4.8, 4.21, 4.22, 4.23, 4.37)
10/07/2011	3.0	Updates for Rev D silicon: Updated accelerometer sensitivity specifications (sections 4.6, 4.8, 4.10, 4.23)
10/24/2011	3.1	Edits for clarity
11/14/2011	3.2	Updated reset value for register 107 (section 3) Updated register 27 with gyro self-test bits (section 4.4) Provided gyro self-test instructions and register bits (section 4.4) Provided accel self-test instructions (section 4.5)
3/9/2012	4.0	Updated register map to include Self-Test registers (section 3) Added description of Self-Test registers (section 4.1) Revised temperature register section (section 4.19) Corrections in registers 107 and 108 (section 4.30)
2/11/2013	4.1	Added reset clarification for SPI interface (section 4.3)
8/19/2013	4.2	Updated sections 6, 7, 8, 10

2 Purpose and Scope

This document provides preliminary information regarding the register map and descriptions for the Motion Processing Units™ MPU-6000™ and MPU-6050™, collectively called the MPU-60X0™ or MPU™.

The MPU devices provide the world's first integrated 6-axis motion processor solution that eliminates the package-level gyroscope and accelerometer cross-axis misalignment associated with discrete solutions. The devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor™ (DMP™) capable of processing complex 9-axis sensor fusion algorithms using the field-proven and proprietary MotionFusion™ engine.

The MPU-6000 and MPU-6050's integrated 9-axis MotionFusion algorithms access external magnetometers or other sensors through an auxiliary master I^2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. The devices are offered in the same 4x4x0.9 mm QFN footprint and pinout as the current MPU-3000™ family of integrated 3-axis gyroscopes, providing a simple upgrade path and facilitating placement on already space constrained circuit boards.

For precision tracking of both fast and slow motions, the MPU-60X0 features a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps). The parts also have a user-programmable accelerometer full-scale range of ±2*g*, ±4*g*, ±8*g*, and ±16*g*.

The MPU-6000 family is comprised of two parts, the MPU-6000 and MPU-6050. These parts are identical to each other with two exceptions. The MPU-6050 supports I^2C communications at up to 400kHz and has a VLOGIC pin that defines its interface voltage levels; the MPU-6000 supports SPI at up to 20MHz in addition to I^2C, and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part.

For more detailed information for the MPU-60X0 devices, please refer to the "MPU-6000 and MPU-6050 Product Specification".

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Note: Register Names ending in _H and _L contain the high and low bytes, respectively, of an internal register value.

In the detailed register tables that follow, register names are in capital letters, while register values are in capital letters and italicized. For example, the ACCEL_XOUT_H register (Register 59) contains the 8 most significant bits, *ACCEL_XOUT*[15:8], of the 16-bit X-Axis accelerometer measurement, *ACCEL_XOUT*.

The reset value is 0x00 for all registers other than the registers below.

- Register 107: 0x40.
- Register 117: 0x68.

4 Register Descriptions

This section describes the function and contents of each register within the MPU-60X0.

Note: The device will come up in sleep mode upon power-up.

4.1 Registers 13 to 16 – Self Test Registers SELF_TEST_X, SELF_TEST_Y, SELF_TEST_Z, and SELF_TEST_A

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	

Description:

These registers are used for gyroscope and accelerometer self-tests that permit the user to test the mechanical and electrical portions of the gyroscope and the accelerometer. The following sections describe the self-test process.

1. Gyroscope Hardware Self-Test: Relative Method

Gyroscope self-test permits users to test the mechanical and electrical portions of the gyroscope. Code for operating self-test is included within the MotionApps™ software provided by InvenSense. Please refer to the next section (*Obtaining the Gyroscope Factory Trim (FT) Value*) if not using MotionApps software.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation will move the sensor's proof masses over a distance equivalent to a pre-defined Coriolis force. This proof mass displacement results in a change in the sensor output, which is reflected in the output signal. The output signal is used to observe the self-test response.

The self-test response (STR) is defined as follows:

$$SelfTest\ Response =$$

$$Gyroscope\ Output\ with\ Self\text{-}Test\ Enabled - Gyroscope\ Output\ with\ Self\text{-}Test\ Disabled$$

This self test-response is used to determine whether the part has passed or failed self-test by finding the change from factory trim of the self-test response as follows:

$$Change\ from\ Factory\ Trim\ of\ the\ Self\text{-}Test\ Response(\%) = \frac{(STR - FT)}{FT}$$

where,

$$FT = Factory\ trim\ value\ of\ selftest\ response, available\ via\ MotionApps\ software$$

This change from factory trim of the self-test response must be within the limits provided in the MPU-6000/MPU-6050 Product Specification document for the part to pass self-test. Otherwise, the part is deemed to have failed self-test.

Obtaining the Gyroscope Factory Trim (FT) Value

If InvenSense MotionApps software is not used, the procedure detailed below should be followed to obtain the Factory trim value of the self test response (FT) mentioned above. For the specific registers mentioned below, please refer to registers 13-15.

The Factory trim value of the self test response (FT) is calculated as shown below. FT[Xg], FT[Yg], and FT[Zg] refer to the factory trim (FT) values for the gyroscope X, Y, and Z axes, respectively. XG_TEST is the decimal version of XG_TEST[4-0], YG_TEST is the decimal version of YG_TEST[4-0], and ZG_TEST is the decimal version of ZG_TEST[4-0].

When performing self test for the gyroscope, the full-scale range should be set to ±250dps.

$$\begin{cases} \text{FT [Xg]} = 25 * 131 * 1.046^{(XG\_TEST-1)} & \text{if XG\_TEST} \neq 0 \\ \text{FT [Xg]} = 0 & \text{if XG\_TEST = 0} \end{cases}$$

$$\begin{cases} \text{FT [Yg]} = -25 * 131 * 1.046^{(YG\_TEST-1)} & \text{if YG\_TEST} \neq 0 \\ \text{FT [Yg]} = 0 & \text{if YG\_TEST = 0} \end{cases}$$

$$\begin{cases} \text{FT [Zg]} = 25 * 131 * 1.046^{(ZG\_TEST-1)} & \text{if ZG\_TEST} \neq 0 \\ \text{FT [Zg]} = 0 & \text{if ZG\_TEST = 0} \end{cases}$$

2. Accelerometer Hardware Self-Test: Relative Method

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. Code for operating self-test is included within the MotionApps software provided by InvenSense. Please refer to the next section (titled Obtaining the Accelerometer Factory Trim (FT) Value) if not using MotionApps software.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response (STR) is defined as follows:

$$\begin{aligned} &SelfTest\ Response \\ &\quad = Accelerometer\ Output\ with\ Self\text{-}Test\ Enabled \\ &\quad - Accelerometer\ Output\ with\ Self\text{-}Test\ Disabled \end{aligned}$$

This self test-response is used to determine whether the part has passed or failed self-test by finding the change from factory trim of the self-test response as follows:

$$Change\ from\ Factory\ Trim\ of\ the\ Self\text{-}Test\ Response(\%) = \frac{(STR - FT)}{FT}$$

where,

$$FT = Factory\ trim\ value\ of\ selftest\ response, available\ via\ MotionApps\ software$$

This change from factory trim of the self-test response must be within the limits provided in the MPU-6000/MPU-6050 Product Specification document for the part to pass self-test. Otherwise, the part is deemed to have failed self-test.

Obtaining the Accelerometer Factory Trim (FT) Value

If InvenSense MotionApps software is not used, the procedure detailed below should be followed to obtain the Factory trim value of the self test response (FT) mentioned above. For the specific registers mentioned below, please refer to registers 13-16.

The Factory trim value of the self test response (FT) is calculated as shown below. FT[Xa], FT[Ya], and FT[Za] refer to the factory trim (FT) values for the accelerometer X, Y, and Z axes, respectively. In the equations below, the factory trim values for the accel should be in decimal format, and they are determined by concatenating the upper accelerometer self test bits (bits 4-2) with the lower accelerometer self test bits (bits 1-0).

When performing accelerometer self test, the full-scale range should be set to ±8*g*.

$$\begin{cases} \text{FT[Xa]} = 4096 * 0.34 * \frac{0.92}{0.34}^{\left(\frac{XA\_TEST-1}{2^5-2}\right)} & \text{if XA\_TEST} \neq 0. \\ \text{FT[Xa]} = 0 & \text{if XA\_TEST = 0.} \end{cases}$$

$$\begin{cases} \text{FT[Ya]} = 4096 * 0.34 * \frac{0.92}{0.34}^{\left(\frac{YA\_TEST-1}{2^5-2}\right)} & \text{if YA\_TEST} \neq 0. \\ \text{FT[Ya]} = 0 & \text{if YA\_TEST = 0.} \end{cases}$$

$$\begin{cases} \text{FT[Za]} = 4096 * 0.34 * \frac{0.92}{0.34}^{\left(\frac{ZA\_TEST-1}{2^5-2}\right)} & \text{if ZA\_TEST} \neq 0. \\ \text{FT[Za]} = 0 & \text{if ZA\_TEST = 0.} \end{cases}$$

Parameters:

XA_TEST	5-bit unsigned value. FT[Xa] is determined by using this value as explained above.
XG_TEST	5-bit unsigned value. FT[Xg] is determined by using this value as explained above.
YA_TEST	5-bit unsigned value. FT[Ya] is determined by using this value as explained above.
YG_TEST	5-bit unsigned value. FT[Yg] is determined by using this value as explained above.
ZA_TEST	5-bit unsigned value. FT[Za] is determined by using this value as explained above.
ZG_TEST	5-bit unsigned value. FT[Zg] is determined by using this value as explained above.

4.2 Register 25 – Sample Rate Divider SMPRT_DIV

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

Description:

This register specifies the divider from the gyroscope output rate used to generate the Sample Rate for the MPU-60X0.

The sensor register output, FIFO output, and DMP sampling are all based on the Sample Rate.

The Sample Rate is generated by dividing the gyroscope output rate by *SMPLRT_DIV*:

Sample Rate = Gyroscope Output Rate / (1 + *SMPLRT_DIV*)

where Gyroscope Output Rate = 8kHz when the DLPF is disabled (*DLPF_CFG* = 0 or 7), and 1kHz when the DLPF is enabled (see Register 26).

Note: The accelerometer output rate is 1kHz. This means that for a Sample Rate greater than 1kHz, the same accelerometer sample may be output to the FIFO, DMP, and sensor registers more than once.

For a diagram of the gyroscope and accelerometer signal paths, see Section 8 of the MPU-6000/MPU-6050 Product Specification document.

Parameters:

SMPLRT_DIV	8-bit unsigned value. The Sample Rate is determined by dividing the gyroscope output rate by this value.

4.3 Register 26 – Configuration
CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Description:

This register configures the external Frame Synchronization (FSYNC) pin sampling and the Digital Low Pass Filter (DLPF) setting for both the gyroscopes and accelerometers.

An external signal connected to the FSYNC pin can be sampled by configuring *EXT_SYNC_SET*.

Signal changes to the FSYNC pin are latched so that short strobes may be captured. The latched FSYNC signal will be sampled at the Sampling Rate, as defined in register 25. After sampling, the latch will reset to the current FSYNC signal state.

The sampled value will be reported in place of the least significant bit in a sensor data register determined by the value of *EXT_SYNC_SET* according to the following table.

EXT_SYNC_SET	FSYNC Bit Location
0	Input disabled
1	TEMP_OUT_L[0]
2	GYRO_XOUT_L[0]
3	GYRO_YOUT_L[0]
4	GYRO_ZOUT_L[0]
5	ACCEL_XOUT_L[0]
6	ACCEL_YOUT_L[0]
7	ACCEL_ZOUT_L[0]

The DLPF is configured by *DLPF_CFG*. The accelerometer and gyroscope are filtered according to the value of *DLPF_CFG* as shown in the table below.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Bit 7 and bit 6 are reserved.

Parameters:

EXT_SYNC_SET 3-bit unsigned value. Configures the FSYNC pin sampling.
DLPF_CFG 3-bit unsigned value. Configures the DLPF setting.

4.4 Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

Description:

This register is used to trigger gyroscope self-test and configure the gyroscopes' full scale range.

Gyroscope self-test permits users to test the mechanical and electrical portions of the gyroscope. The self-test for each gyroscope axis can be activated by controlling the *XG_ST*, *YG_ST*, and *ZG_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation will move the sensor's proof masses over a distance equivalent to a pre-defined Coriolis force. This proof mass displacement results in a change in the sensor output, which is reflected in the output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each gyroscope axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

FS_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

Bits 2 through 0 are reserved.

Parameters:

XG_ST	Setting this bit causes the X axis gyroscope to perform self test.
YG_ST	Setting this bit causes the Y axis gyroscope to perform self test.
ZG_ST	Setting this bit causes the Z axis gyroscope to perform self test.
FS_SEL	2-bit unsigned value. Selects the full scale range of gyroscopes.

4.5 Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Description:

This register is used to trigger accelerometer self test and configure the accelerometer full scale range. This register also configures the Digital High Pass Filter (DHPF).

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. The self-test for each accelerometer axis can be activated by controlling the *XA_ST*, *YA_ST*, and *ZA_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each accelerometer axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

AFS_SEL selects the full scale range of the accelerometer outputs according to the following table.

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

Parameters:

XA_ST	When set to 1, the X- Axis accelerometer performs self test.
YA_ST	When set to 1, the Y- Axis accelerometer performs self test.
ZA_ST	When set to 1, the Z- Axis accelerometer performs self test.
AFS_SEL	2-bit unsigned value. Selects the full scale range of accelerometers.

4.6 Register 35 – FIFO Enable
FIFO_EN

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23	35	TEMP_ FIFO_EN	XG_ FIFO_EN	YG_ FIFO_EN	ZG_ FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN

Description:

This register determines which sensor measurements are loaded into the FIFO buffer.

Data stored inside the sensor data registers (Registers 59 to 96) will be loaded into the FIFO buffer if a sensor's respective FIFO_EN bit is set to 1 in this register.

When a sensor's FIFO_EN bit is enabled in this register, data from the sensor data registers will be loaded into the FIFO buffer. The sensors are sampled at the Sample Rate as defined in Register 25. For further information regarding sensor data registers, please refer to Registers 59 to 96

When an external Slave's corresponding FIFO_EN bit (*SLVx_FIFO_EN,* where x=0, 1, or 2) is set to 1, the data stored in its corresponding data registers (EXT_SENS_DATA registers, Registers 73 to 96) will be written into the FIFO buffer at the Sample Rate. EXT_SENS_DATA register association with I^2C Slaves is determined by the I2C_SLVx_CTRL registers (where x=0, 1, or 2; Registers 39, 42, and 45). For information regarding EXT_SENS_DATA registers, please refer to Registers 73 to 96.

Note that the corresponding FIFO_EN bit (*SLV3_FIFO_EN)* is found in I2C_MST_CTRL (Register 36). Also note that Slave 4 behaves in a different manner compared to Slaves 0-3. Please refer to Registers 49 to 53 for further information regarding Slave 4 usage.

Parameters:

TEMP_FIFO_EN	When set to 1, this bit enables TEMP_OUT_H and TEMP_OUT_L (Registers 65 and 66) to be written into the FIFO buffer.
XG_ FIFO_EN	When set to 1, this bit enables GYRO_XOUT_H and GYRO_XOUT_L (Registers 67 and 68) to be written into the FIFO buffer.
YG_ FIFO_EN	When set to 1, this bit enables GYRO_YOUT_H and GYRO_YOUT_L (Registers 69 and 70) to be written into the FIFO buffer.
ZG_ FIFO_EN	When set to 1, this bit enables GYRO_ZOUT_H and GYRO_ZOUT_L (Registers 71 and 72) to be written into the FIFO buffer.
ACCEL_ FIFO_EN	When set to 1, this bit enables ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, and ACCEL_ZOUT_L (Registers 59 to 64) to be written into the FIFO buffer.

SLV2_ FIFO_EN When set to 1, this bit enables EXT_SENS_DATA registers (Registers 73 to 96) associated with Slave 2 to be written into the FIFO buffer.

SLV1_ FIFO_EN When set to 1, this bit enables EXT_SENS_DATA registers (Registers 73 to 96) associated with Slave 1 to be written into the FIFO buffer.

SLV0_ FIFO_EN When set to 1, this bit enables EXT_SENS_DATA registers (Registers 73 to 96) associated with Slave 0 to be written into the FIFO buffer.

Note: For further information regarding the association of EXT_SENS_DATA registers to particular slave devices, please refer to Registers 73 to 96.

4.7 Register 36 – I²C Master Control I2C_MST_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24	36	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			

Description:

This register configures the auxiliary I²C bus for single-master or multi-master control. In addition, the register is used to delay the Data Ready interrupt, and also enables the writing of Slave 3 data into the FIFO buffer. The register also configures the auxiliary I²C Master's transition from one slave read to the next, as well as the MPU-60X0's 8MHz internal clock.

Multi-master capability allows multiple I²C masters to operate on the same bus. In circuits where multi-master capability is required, set *MULT_MST_EN* to 1. This will increase current drawn by approximately 30µA.

In circuits where multi-master capability is required, the state of the I²C bus must always be monitored by each separate I²C Master. Before an I²C Master can assume arbitration of the bus, it must first confirm that no other I²C Master has arbitration of the bus. When *MULT_MST_EN* is set to 1, the MPU-60X0's bus arbitration detection logic is turned on, enabling it to detect when the bus is available.

When the *WAIT_FOR_ES* bit is set to 1, the Data Ready interrupt will be delayed until External Sensor data from the Slave Devices are loaded into the EXT_SENS_DATA registers. This is used to ensure that both the internal sensor data (i.e. from gyro and accel) and external sensor data have been loaded to their respective data registers (i.e. the data is synced) when the Data Ready interrupt is triggered.

When the Slave 3 FIFO enable bit (*SLV_3_FIFO_EN*) is set to 1, Slave 3 sensor measurement data will be loaded into the FIFO buffer each time. EXT_SENS_DATA register association with I²C Slaves is determined by I2C_SLV3_CTRL (Register 48).

For further information regarding EXT_SENS_DATA registers, please refer to Registers 73 to 96.

The corresponding FIFO_EN bits for Slave 0, Slave 1, and Slave 2 can be found in Register 35.

The *I2C_MST_P_NSR* bit configures the I²C Master's transition from one slave read to the next slave read. If the bit equals 0, there will be a restart between reads. If the bit equals 1, there will be a stop followed by a start of the following read. When a write transaction follows a read transaction, the stop followed by a start of the successive write will be always used.

I2C_MST_CLK is a 4 bit unsigned value which configures a divider on the MPU-60X0 internal 8MHz clock. It sets the I^2C master clock speed according to the following table:

I2C_MST_CLK	I^2C Master Clock Speed	8MHz Clock Divider
0	348 kHz	23
1	333 kHz	24
2	320 kHz	25
3	308 kHz	26
4	296 kHz	27
5	286 kHz	28
6	276 kHz	29
7	267 kHz	30
8	258 kHz	31
9	500 kHz	16
10	471 kHz	17
11	444 kHz	18
12	421 kHz	19
13	400 kHz	20
14	381 kHz	21
15	364 kHz	22

Parameters:

MUL_MST_EN When set to 1, this bit enables multi-master capability.

WAIT_FOR_ES When set to 1, this bit delays the Data Ready interrupt until External Sensor data from the Slave devices have been loaded into the EXT_SENS_DATA registers.

SLV3_FIFO_EN When set to 1, this bit enables EXT_SENS_DATA registers associated with Slave 3 to be written into the FIFO. The corresponding bits for Slaves 0-2 can be found in Register 35.

I2C_MST_P_NSR Controls the I^2C Master's transition from one slave read to the next slave read.
When this bit equals 0, there is a restart between reads.
When this bit equals 1, there is a stop and start marking the beginning of the next read.

When a write follows a read, a stop and start is always enforced.

I2C_MST_CLK 4 bit unsigned value. Configures the I^2C master clock speed divider.

Note: For further information regarding the association of EXT_SENS_DATA registers to particular slave devices, please refer to Registers 73 to 96.

4.8 Registers 37 to 39 – I²C Slave 0 Control I2C_SLV0_ADDR, I2C_SLV0_REG, and I2C_SLV0_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
25	37	I2C_SLV0_RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_EN	I2C_SLV0_BYTE_SW	I2C_SLV0_REG_DIS	I2C_SLV0_GRP	I2C_SLV0_LEN[3:0]			

Description:

These registers configure the data transfer sequence for Slave 0. Slaves 1, 2, and 3 also behave in a similar manner to Slave 0. However, Slave 4's characteristics differ greatly from those of Slaves 0-3. For further information regarding Slave 4, please refer to registers 49 to 53.

I²C slave data transactions between the MPU-60X0 and Slave 0 are set as either read or write operations by the *I2C_SLV0_RW* bit. When this bit is 1, the transfer is a read operation. When the bit is 0, the transfer is a write operation.

I2C_SLV0_ADDR is used to specify the I²C slave address of Slave 0.

Data transfer starts at an internal register within Slave 0. This address of this register is specified by *I2C_SLV0_REG*.

The number of bytes transferred is specified by *I2C_SLV0_LEN.* When more than 1 byte is transferred (*I2C_SLV0_LEN* > 1), data is read from (written to) sequential addresses starting from *I2C_SLV0_REG*.

In read mode, the result of the read is placed in the lowest available EXT_SENS_DATA register. For further information regarding the allocation of read results, please refer to the EXT_SENS_DATA register description (Registers 73 – 96).

In write mode, the contents of *I2C_SLV0_DO* (Register 99) will be written to the slave device.

I2C_SLV0_EN enables Slave 0 for I²C data transaction. A data transaction is performed only if more than zero bytes are to be transferred (*I2C_SLV0_LEN* > 0) between an enabled slave device (*I2C_SLV0_EN* = 1).

I2C_SLV0_BYTE_SW configures byte swapping of word pairs. When byte swapping is enabled, the high and low bytes of a word pair are swapped. Please refer to *I2C_SLV0_GRP* for the pairing convention of the word pairs. When this bit is cleared to 0, bytes transferred to and from Slave 0 will be written to EXT_SENS_DATA registers in the order they were transferred.

When *I2C_SLV0_REG_DIS* is set to 1, the transaction will read or write data only. When cleared to 0, the transaction will write a register address prior to reading or writing data. This bit should equal 0 when specifying the register address within the Slave device to/from which the ensuing data transaction will take place.

I2C_SLV0_GRP specifies the grouping order of word pairs received from registers. When cleared to 0, bytes from register addresses 0 and 1, 2 and 3, etc (even, then odd register addresses) are paired to form a word. When set to 1, bytes from register addresses are paired 1 and 2, 3 and 4, etc. (odd, then even register addresses) are paired to form a word.

I^2C data transactions are performed at the Sample Rate, as defined in Register 25. The user is responsible for ensuring that I^2C data transactions to and from each enabled Slave can be completed within a single period of the Sample Rate.

The I^2C slave access rate can be reduced relative to the Sample Rate. This reduced access rate is determined by *I2C_MST_DLY* (Register 52). Whether a slave's access rate is reduced relative to the Sample Rate is determined by I2C_MST_DELAY_CTRL (Register 103).

The processing order for the slaves is fixed. The sequence followed for processing the slaves is Slave 0, Slave 1, Slave 2, Slave 3 and Slave 4. If a particular Slave is disabled it will be skipped.

Each slave can either be accessed at the sample rate or at a reduced sample rate. In a case where some slaves are accessed at the Sample Rate and some slaves are accessed at the reduced rate, the sequence of accessing the slaves (Slave 0 to Slave 4) is still followed. However, the reduced rate slaves will be skipped if their access rate dictates that they should not be accessed during that particular cycle. For further information regarding the reduced access rate, please refer to Register 52. Whether a slave is accessed at the Sample Rate or at the reduced rate is determined by the Delay Enable bits in Register 103.

Parameters:

I2C_SLV0_RW	When set to 1, this bit configures the data transfer as a read operation. When cleared to 0, this bit configures the data transfer as a write operation.
I2C_SLV0_ADDR	7-bit I^2C address of Slave 0.
I2C_SLV0_REG	8-bit address of the Slave 0 register to/from which data transfer starts.
I2C_SLV0_EN	When set to 1, this bit enables Slave 0 for data transfer operations. When cleared to 0, this bit disables Slave 0 from data transfer operations.
I2C_SLV0_BYTE_SW	When set to 1, this bit enables byte swapping. When byte swapping is enabled, the high and low bytes of a word pair are swapped. Please refer to *I2C_SLV0_GRP* for the pairing convention of the word pairs. When cleared to 0, bytes transferred to and from Slave 0 will be written to EXT_SENS_DATA registers in the order they were transferred.
I2C_SLV0_REG_DIS	When set to 1, the transaction will read or write data only. When cleared to 0, the transaction will write a register address prior to reading or writing data.
I2C_SLV0_GRP	1-bit value specifying the grouping order of word pairs received from registers. When cleared to 0, bytes from register addresses 0 and 1, 2 and 3, etc (even, then odd register addresses) are paired to form a word. When set to 1, bytes from register addresses are paired 1 and 2, 3 and 4, etc. (odd, then even register addresses) are paired to form a word.
I2C_SLV0_LEN	4-bit unsigned value. Specifies the number of bytes transferred to and from Slave 0. Clearing this bit to 0 is equivalent to disabling the register by writing 0 to *I2C_SLV0_EN*.

Byte Swapping Example

The following example demonstrates byte swapping for *I2C_SLV0_BYTE_SW* = 1, *I2C_SLV0_GRP* = 0, *I2C_SLV0_REG* = 0x01, and *I2C_SLV0_LEN* = 0x4:

1. The first byte, read from Slave 0 register 0x01, will be stored at EXT_SENS_DATA_00. Because *I2C_SLV0_GRP* = 0, bytes from even, then odd register addresses will be paired together as word pairs. Since the read operation started from an odd register address instead of an even address, only one byte is read.
2. The second and third bytes will be swapped, since *I2C_SLV0_BYTE_SW* = 1 and *I2C_SLV0_REG[0]* = 1. The data read from 0x02 will be stored at EXT_SENS_DATA_02, and the data read from 0x03 will be stored at EXT_SENS_DATA_01.
3. The last byte, read from address 0x04, will be stored at EXT_SENS_DATA_03. Because there is only one byte remaining in the read operation, byte swapping will not occur.

Slave Access Example

Slave 0 is accessed at the Sample Rate, while Slave 1 is accessed at half the Sample Rate. The other slaves are disabled. In the first cycle, both Slave 0 and Slave 1 will be accessed. However, in the second cycle, only Slave 0 will be accessed. In the third cycle, both Slave 0 and Slave 1 will be accessed. In the fourth cycle, only Slave 0 will be accessed. This pattern continues.

4.9 Registers 40 to 42 – I²C Slave 1 Control I2C_SLV1_ADDR, I2C_SLV1_REG, and I2C_SLV1_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
28	40	I2C_SLV1_RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_EN	I2C_SLV1_BYTE_SW	I2C_SLV1_REG_DIS	I2C_SLV1_GRP	I2C_SLV1_LEN[3:0]			

Description:

These registers describe the data transfer sequence for Slave 1. Their functions correspond to those described for the Slave 0 registers (Registers 37 to 39).

4.10 Registers 43 to 45 – I²C Slave 2 Control I2C_SLV2_ADDR, I2C_SLV2_REG, and I2C_SLV2_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B	43	I2C_SLV2_RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_EN	I2C_SLV2_BYTE_SW	I2C_SLV2_REG_DIS	I2C_SLV2_GRP	I2C_SLV2_LEN[3:0]			

Description:

These registers describe the data transfer sequence for Slave 2. Their functions correspond to those described for the Slave 0 registers (Registers 37 to 39).

4.11 Registers 46 to 48 – I²C Slave 3 Control I2C_SLV3_ADDR, I2C_SLV3_REG, and I2C_SLV3_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E	46	I2C_SLV3_RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_EN	I2C_SLV3_BYTE_SW	I2C_SLV3_REG_DIS	I2C_SLV3_GRP	I2C_SLV3_LEN[3:0]			

Description:

These registers describe the data transfer sequence for Slave 3. Their functions correspond to those described for the Slave 0 registers (Registers 37 to 39).

4.12 Registers 49 to 53 – I²C Slave 4 Control
I2C_SLV4_ADDR, I2C_SLV4_REG, I2C_SLV4_DO, I2C_SLV4_CTRL, and I2C_SLV4_DI

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
31	49	I2C_SLV4_RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_EN	I2C_SLV4_INT_EN	I2C_SLV4_REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI[7:0]							

Description:

These registers describe the data transfer sequence for Slave 4. The characteristics of Slave 4 differ greatly from those of Slaves 0-3. For further information regarding the characteristics of Slaves 0-3, please refer to Registers 37 to 48.

I²C slave data transactions between the MPU-60X0 and Slave 4 are set as either read or write operations by the *I2C_SLV4_RW* bit. When this bit is 1, the transfer is a read operation. When the bit is 0, the transfer is a write operation.

I2C_SLV4_ADDR is used to specify the I²C slave address of Slave 4.

Data transfer starts at an internal register within Slave 4. This register address is specified by *I2C_SLV4_REG*.

In read mode, the result of the read will be available in *I2C_SLV4_DI*. In write mode, the contents of *I2C_SLV4_DO* will be written into the slave device.

A data transaction is performed only if the *I2C_SLV4_EN* bit is set to 1. The data transaction should be enabled once its parameters are configured in the _ADDR and _REG registers. For write, the _DO register is also required. *I2C_SLV4_EN* will be cleared after the transaction is performed once.

An interrupt is triggered at the completion of a Slave 4 data transaction if the interrupt is enabled . The status of this interrupt can be observed in Register 54.

When *I2C_SLV4_REG_DIS* is set to 1, the transaction will read or write data instead of writing a register address. This bit should equal 0 when specifying the register address within the Slave device to/from which the ensuing data transaction will take place.

I2C_MST_DLY configures the reduced access rate of I²C slaves relative to the Sample Rate. When a slave's access rate is decreased relative to the Sample Rate, the slave is accessed every

1 / (1 + *I2C_MST_DLY*) samples

This base Sample Rate in turn is determined by *SMPLRT_DIV* (register 25) and *DLPF_CFG* (register 26). Whether a slave's access rate is reduced relative to the Sample Rate is determined by I2C_MST_DELAY_CTRL (register 103).

For further information regarding the Sample Rate, please refer to register 25.

Slave 4 transactions are performed after Slave 0, 1, 2 and 3 transactions have been completed. Thus the maximum rate for Slave 4 transactions is determined by the Sample Rate as defined in Register 25.

Parameters:

I2C_SLV4_RW	When set to 1, this bit configures the data transfer as a read operation. When cleared to 0, this bit configures the data transfer as a write operation.
I2C_SLV4_ADDR	7-bit I^2C address for Slave 4.
I2C_SLV4_REG	8-bit address of the Slave 4 register to/from which data transfer starts.
I2C_SLV4_DO	This register stores the data to be written into the Slave 4. If *I2C_SLV4_RW* is set 1 (set to read), this register has no effect.
I2C_SLV4_EN	When set to 1, this bit enables Slave 4 for data transfer operations.
	When cleared to 0, this bit disables Slave 4 from data transfer operations.
I2C_SLV4_INT_EN	When set to 1, this bit enables the generation of an interrupt signal upon completion of a Slave 4 transaction.
	When cleared to 0, this bit disables the generation of an interrupt signal upon completion of a Slave 4 transaction.
	The interrupt status can be observed in Register 54.
I2C_SLV4_REG_DIS	When set to 1, the transaction will read or write data. When cleared to 0, the transaction will read or write a register address.
I2C_MST_DLY	Configures the decreased access rate of slave devices relative to the Sample Rate.
I2C_SLV4_DI	This register stores the data read from Slave 4.
	This field is populated after a read transaction.

4.13 Register 54 – I²C Master Status I2C_MST_STATUS

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
36	54	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK

Description:

This register shows the status of the interrupt generating signals in the I²C Master within the MPU-60X0. This register also communicates the status of the FSYNC interrupt to the host processor.

Reading this register will clear all the status bits in the register.

Parameters:

PASS_THROUGH This bit reflects the status of the FSYNC interrupt from an external device into the MPU-60X0. This is used as a way to pass an external interrupt through the MPU-60X0 to the host application processor. When set to 1, this bit will cause an interrupt if *FSYNC_INT_EN* is asserted in INT_PIN_CFG (Register 55).

I2C_SLV4_DONE Automatically sets to 1 when a Slave 4 transaction has completed. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted and if the *SLV_4_DONE_INT* bit is asserted in the I2C_SLV4_CTRL register (Register 52).

I2C_LOST_ARB This bit automatically sets to 1 when the I²C Master has lost arbitration of the auxiliary I²C bus (an error condition). This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

I2C_SLV4_NACK This bit automatically sets to 1 when the I²C Master receives a NACK in a transaction with Slave 4. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

I2C_SLV3_NACK This bit automatically sets to 1 when the I²C Master receives a NACK in a transaction with Slave 3. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

I2C_SLV2_NACK This bit automatically sets to 1 when the I²C Master receives a NACK in a transaction with Slave 2. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

I2C_SLV1_NACK This bit automatically sets to 1 when the I²C Master receives a NACK in a transaction with Slave 1. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

I2C_SLV0_NACK This bit automatically sets to 1 when the I²C Master receives a NACK in a transaction with Slave 0. This triggers an interrupt if the *I2C_MST_INT_EN* bit in the INT_ENABLE register (Register 56) is asserted.

4.14 Register 55 – INT Pin / Bypass Enable Configuration
INT_PIN_CFG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
37	55	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC_ INT_EN	I2C _BYPASS _EN	-

Description:

This register configures the behavior of the interrupt signals at the INT pins. This register is also used to enable the FSYNC Pin to be used as an interrupt to the host application processor, as well as to enable Bypass Mode on the I^2C Master. This bit also enables the clock output.

FSYNC_INT_EN enables the FSYNC pin to be used as an interrupt to the host application processor. A transition to the active level specified in *FSYNC_INT_LEVEL* will trigger an interrupt. The status of this interrupt is read from the *PASS_THROUGH* bit in the I^2C Master Status Register (Register 54).

When *I2C_BYPASS_EN* is equal to 1 and *I2C_MST_EN* (Register 106 bit[5]) is equal to 0, the host application processor will be able to directly access the auxiliary I^2C bus of the MPU-60X0. When this bit is equal to 0, the host application processor will not be able to directly access the auxiliary I^2C bus of the MPU-60X0 regardless of the state of *I2C_MST_EN.*

For further information regarding Bypass Mode, please refer to Section 7.11 and 7.13 of the MPU-6000/MPU-6050 Product Specification document.

Parameters:

INT_LEVEL
When this bit is equal to 0, the logic level for the INT pin is active high.
When this bit is equal to 1, the logic level for the INT pin is active low.

INT_OPEN
When this bit is equal to 0, the INT pin is configured as push-pull.
When this bit is equal to 1, the INT pin is configured as open drain.

LATCH_INT_EN
When this bit is equal to 0, the INT pin emits a 50us long pulse.
When this bit is equal to 1, the INT pin is held high until the interrupt is cleared.

INT_RD_CLEAR
When this bit is equal to 0, interrupt status bits are cleared only by reading INT_STATUS (Register 58)

When this bit is equal to 1, interrupt status bits are cleared on any read operation.

FSYNC_INT_LEVEL
When this bit is equal to 0, the logic level for the FSYNC pin (when used as an interrupt to the host processor) is active high.
When this bit is equal to 1, the logic level for the FSYNC pin (when used as an interrupt to the host processor) is active low.

FSYNC_INT_EN
When equal to 0, this bit disables the FSYNC pin from causing an interrupt to the host processor.

When equal to 1, this bit enables the FSYNC pin to be used as an interrupt to the host processor.

I2C_BYPASS_EN When this bit is equal to 1 and *I2C_MST_EN* (Register 106 bit[5]) is equal to 0, the host application processor will be able to directly access the auxiliary I^2C bus of the MPU-60X0.

When this bit is equal to 0, the host application processor will not be able to directly access the auxiliary I^2C bus of the MPU-60X0 regardless of the state of *I2C_MST_EN (*Register 106 bit[5]).

4.15 Register 56 – Interrupt Enable
INT_ENABLE

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
38	56		-		FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN

Description:

This register enables interrupt generation by interrupt sources.

For information regarding the interrupt status for each interrupt generation source, please refer to Register 58. Further information regarding I^2C Master interrupt generation can be found in Register 54.

Bits 2 and 1 are reserved.

Parameters:

FIFO_OFLOW_EN When set to 1, this bit enables a FIFO buffer overflow to generate an interrupt.

I2C_MST_INT_EN When set to 1, this bit enables any of the I^2C Master interrupt sources to generate an interrupt.

DATA_RDY_EN When set to 1, this bit enables the Data Ready interrupt, which occurs each time a write operation to all of the sensor registers has been completed.

4.16 Register 58 – Interrupt Status
INT_STATUS

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3A	58	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Description:

This register shows the interrupt status of each interrupt generation source. Each bit will clear after the register is read.

For information regarding the corresponding interrupt enable bits, please refer to Register 56.

For a list of I^2C Master interrupts, please refer to Register 54.

Bits 2 and 1 are reserved.

Parameters:

FIFO_OFLOW_INT This bit automatically sets to 1 when a FIFO buffer overflow interrupt has been generated.

The bit clears to 0 after the register has been read.

I2C_MST_INT This bit automatically sets to 1 when an I^2C Master interrupt has been generated. For a list of I^2C Master interrupts, please refer to Register 54.

The bit clears to 0 after the register has been read.

DATA_RDY_INT This bit automatically sets to 1 when a Data Ready interrupt is generated.

The bit clears to 0 after the register has been read.

4.17 Registers 59 to 64 – Accelerometer Measurements ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, and ACCEL_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Description:

These registers store the most recent accelerometer measurements.

Accelerometer measurements are written to these registers at the Sample Rate as defined in Register 25.

The accelerometer measurement registers, along with the temperature measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the accelerometer sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit accelerometer measurement has a full scale defined in *ACCEL_FS* (Register 28). For each full scale setting, the accelerometers' sensitivity per LSB in *ACCEL_xOUT* is shown in the table below.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Parameters:

ACCEL_XOUT 16-bit 2's complement value.

Stores the most recent X axis accelerometer measurement.

ACCEL_YOUT 16-bit 2's complement value.

Stores the most recent Y axis accelerometer measurement.

ACCEL_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis accelerometer measurement.

4.18 Registers 65 and 66 – Temperature Measurement TEMP_OUT_H and TEMP_OUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
41	65	TEMP_OUT[15:8]							
42	66	TEMP_OUT[7:0]							

Description:

These registers store the most recent temperature sensor measurement.

Temperature measurements are written to these registers at the Sample Rate as defined in Register 25.

These temperature measurement registers, along with the accelerometer measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the temperature sensor's internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

The scale factor and offset for the temperature sensor are found in the Electrical Specifications table (Section 6.4 of the MPU-6000/MPU-6050 Product Specification document).

The temperature in degrees C for a given register value may be computed as:

Temperature in degrees C = (TEMP_OUT Register Value as a signed quantity)/340 + 36.53

Please note that the math in the above equation is in decimal.

Parameters:

TEMP_OUT 16-bit signed value.

Stores the most recent temperature sensor measurement.

4.19 Registers 67 to 72 – Gyroscope Measurements GYRO_XOUT_H, GYRO_XOUT_L, GYRO_YOUT_H, GYRO_YOUT_L, GYRO_ZOUT_H, and GYRO_ZOUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

Description:

These registers store the most recent gyroscope measurements.

Gyroscope measurements are written to these registers at the Sample Rate as defined in Register 25.

These gyroscope measurement registers, along with the accelerometer measurement registers, temperature measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the gyroscope sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit gyroscope measurement has a full scale defined in *FS_SEL* (Register 27). For each full scale setting, the gyroscopes' sensitivity per LSB in *GYRO_xOUT* is shown in the table below:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

Parameters:

GYRO_XOUT 16-bit 2's complement value.

Stores the most recent X axis gyroscope measurement.

GYRO_YOUT 16-bit 2's complement value.

Stores the most recent Y axis gyroscope measurement.

GYRO_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis gyroscope measurement.

4.20 Registers 73 to 96 – External Sensor Data
EXT_SENS_DATA_00 through EXT_SENS_DATA_23

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
49	73	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23[7:0]							

Description:

These registers store data read from external sensors by the Slave 0, 1, 2, and 3 on the auxiliary I^2C interface. Data read by Slave 4 is stored in I2C_SLV4_DI (Register 53).

External sensor data is written to these registers at the Sample Rate as defined in Register 25. This access rate can be reduced by using the Slave Delay Enable registers (Register 103).

External sensor data registers, along with the gyroscope measurement registers, accelerometer measurement registers, and temperature measurement registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the external sensors' internal register set is always updated at the Sample Rate (or the reduced access rate) whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Data is placed in these external sensor data registers according to I2C_SLV0_CTRL, I2C_SLV1_CTRL, I2C_SLV2_CTRL, and I2C_SLV3_CTRL (Registers 39, 42, 45, and 48). When more than zero bytes are read (*I2C_SLVx_LEN* > 0) from an enabled slave (*I2C_SLVx_EN* = 1), the slave is read at the Sample Rate (as defined in Register 25) or delayed rate (if specified in Register 52 and 103). During each Sample cycle, slave reads are performed in order of Slave number. If all slaves are enabled with more than zero bytes to be read, the order will be Slave 0, followed by Slave 1, Slave 2, and Slave 3.

Each enabled slave will have EXT_SENS_DATA registers associated with it by number of bytes read (*I2C_SLVx_LEN)* in order of slave number, starting from EXT_SENS_DATA_00. Note that this means enabling or disabling a slave may change the higher numbered slaves' associated registers. Furthermore, if fewer total bytes are being read from the external sensors as a result of such a change, then the data remaining in the registers which no longer have an associated slave device (i.e. high numbered registers) will remain in these previously allocated registers unless reset.

If the sum of the read lengths of all SLVx transactions exceed the number of available EXT_SENS_DATA registers , the excess bytes will be dropped. There are 24 EXT_SENS_DATA registers and hence the total read lengths between all the slaves cannot be greater than 24 or some bytes will be lost.

Note: Slave 4's behavior is distinct from that of Slaves 0-3. For further information regarding the characteristics of Slave 4, please refer to Registers 49 to 53.

Example:

Suppose that Slave 0 is enabled with 4 bytes to be read (*I2C_SLV0_EN* = 1 and *I2C_SLV0_LEN* = 4) while Slave 1 is enabled with 2 bytes to be read, (*I2C_SLV1_EN*=1 and *I2C_SLV1_LEN* = 2). In such a situation, EXT_SENS_DATA _00 through _03 will be associated with Slave 0, while EXT_SENS_DATA _04 and 05 will be associated with Slave 1.

If Slave 2 is enabled as well, registers starting from EXT_SENS_DATA_06 will be allocated to Slave 2.

If Slave 2 is disabled while Slave 3 is enabled in this same situation, then registers starting from EXT_SENS_DATA_06 will be allocated to Slave 3 instead.

Register Allocation for Dynamic Disable vs. Normal Disable

If a slave is disabled at any time, the space initially allocated to the slave in the EXT_SENS_DATA register, will remain associated with that slave. This is to avoid dynamic adjustment of the register allocation.

The allocation of the EXT_SENS_DATA registers is recomputed only when (1) all slaves are disabled, or (2) the *I2C_MST_RST* bit is set (Register 106).

This above is also true if one of the slaves gets NACKed and stops functioning.

4.21 Register 99 – I²C Slave 0 Data Out I2C_SLV0_DO

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
63	99	I2C_SLV0_DO[7:0]							

Description:

This register holds the output data written into Slave 0 when Slave 0 is set to write mode.

For further information regarding Slave 0 control, please refer to Registers 37 to 39.

Parameters:

I2C_SLV0_DO 8 bit unsigned value that is written into Slave 0 when Slave 0 is set to write mode.

4.22 Register 100 – I²C Slave 1 Data Out I2C_SLV1_DO

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
64	100	I2C_SLV1_DO[7:0]							

Description:

This register holds the output data written into Slave 1 when Slave 1 is set to write mode.

For further information regarding Slave 1 control, please refer to Registers 40 to 42.

Parameters:

I2C_SLV1_DO 8 bit unsigned value that is written into Slave 1 when Slave 1 is set to write mode.

4.23 Register 101 – I²C Slave 2 Data Out I2C_SLV2_DO

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
65	101	I2C_SLV2_DO[7:0]							

Description:

This register holds the output data written into Slave 2 when Slave 2 is set to write mode.

For further information regarding Slave 2 control, please refer to Registers 43 to 45.

Parameters:

I2C_SLV2_DO 8 bit unsigned value that is written into Slave 2 when Slave 2 is set to write mode.

4.24 Register 102 – I²C Slave 3 Data Out I2C_SLV3_DO

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
66	102	I2C_SLV3_DO[7:0]							

Description:

This register holds the output data written into Slave 3 when Slave 3 is set to write mode.

For further information regarding Slave 3 control, please refer to Registers 46 to 48.

Parameters:

I2C_SLV3_DO 8 bit unsigned value that is written into Slave 3 when Slave 3 is set to write mode.

4.25 Register 103 – I²C Master Delay Control I2C_MST_DELAY_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN

Description:

This register is used to specify the timing of external sensor data shadowing. The register is also used to decrease the access rate of slave devices relative to the Sample Rate.

When *DELAY_ES_SHADOW* is set to 1, shadowing of external sensor data is delayed until all data has been received.

When *I2C_SLV4_DLY_EN, I2C_SLV3_DLY_EN, I2C_SLV2_DLY_EN, I2C_SLV1_DLY_EN,* and *I2C_SLV0_DLY_EN* are enabled, the rate of access for the corresponding slave devices is reduced.

When a slave's access rate is decreased relative to the Sample Rate, the slave is accessed every

1 / (1 + *I2C_MST_DLY*) samples.

This base Sample Rate in turn is determined by *SMPLRT_DIV* (register 25) and *DLPF_CFG* (register 26).

For further information regarding *I2C_MST_DLY*, please refer to register 52.

For further information regarding the Sample Rate, please refer to register 25.

Bits 6 and 5 are reserved.

Parameters:

DELAY_ES_SHADOW	When set, delays shadowing of external sensor data until all data has been received.
I2C_SLV4_DLY_EN	When enabled, slave 4 will only be accessed at a decreased rate.
I2C_SLV3_DLY_EN	When enabled, slave 3 will only be accessed at a decreased rate.
I2C_SLV2_DLY_EN	When enabled, slave 2 will only be accessed at a decreased rate.
I2C_SLV1_DLY_EN	When enabled, slave 1 will only be accessed at a decreased rate.
I2C_SLV0_DLY_EN	When enabled, slave 0 will only be accessed at a decreased rate.

4.26 Register 104 – Signal Path Reset
SIGNAL_PATH_RESET

Type: Write Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
68	104	-	-	-	-	-	GYRO _RESET	ACCEL _RESET	TEMP _RESET

Description:

This register is used to reset the analog and digital signal paths of the gyroscope, accelerometer, and temperature sensors.

The reset will revert the signal path analog to digital converters and filters to their power up configurations.

Note: This register does not clear the sensor registers. The reset initializes the serial interface as well.

Bits 7 to 3 are reserved.

Parameters:

GYRO_RESET When set to 1, this bit resets the gyroscope analog and digital signal paths.

ACCEL_RESET When set to 1, this bit resets the accelerometer analog and digital signal paths.

TEMP_RESET When set to 1, this bit resets the temperature sensor analog and digital signal paths.

4.27 Register 106 – User Control
USER_CTRL

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6A	106	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET

Description:

This register allows the user to enable and disable the FIFO buffer, I^2C Master Mode**,** and primary I^2C interface. The FIFO buffer, I^2C Master, sensor signal paths and sensor registers can also be reset using this register.

When *I2C_MST_EN* is set to 1, I^2C Master Mode is enabled. In this mode, the MPU-60X0 acts as the I^2C Master to the external sensor slave devices on the auxiliary I^2C bus. When this bit is cleared to 0, the auxiliary I^2C bus lines (AUX_DA and AUX_CL) are logically driven by the primary I^2C bus (SDA and SCL). This is a precondition to enabling Bypass Mode. For further information regarding Bypass Mode, please refer to Register 55.

MPU-6000: The primary SPI interface will be enabled in place of the disabled primary I^2C interface when *I2C_IF_DIS* is set to 1.

MPU-6050: Always write 0 to *I2C_IF_DIS*.

When the reset bits (*FIFO_RESET, I2C_MST_RESET, and SIG_COND_RESET*) are set to 1, these reset bits will trigger a reset and then clear to 0.

Bits 7 and 3 are reserved.

Parameters:

FIFO_EN — When set to 1, this bit enables FIFO operations.

When this bit is cleared to 0, the FIFO buffer is disabled. The FIFO buffer cannot be written to or read from while disabled.

The FIFO buffer's state does not change unless the MPU-60X0 is power cycled.

I2C_MST_EN — When set to 1, this bit enables I^2C Master Mode.

When this bit is cleared to 0, the auxiliary I^2C bus lines (AUX_DA and AUX_CL) are logically driven by the primary I^2C bus (SDA and SCL).

I2C_IF_DIS — MPU-6000: When set to 1, this bit disables the primary I^2C interface and enables the SPI interface instead.

MPU-6050: Always write this bit as zero.

FIFO_RESET — This bit resets the FIFO buffer *w*hen set to 1 while *FIFO_EN* equals 0. This bit automatically clears to 0 after the reset has been triggered.

I2C_MST_RESET — This bit resets the I^2C Master when set to 1 while *I2C_MST_EN* equals 0. This bit automatically clears to 0 after the reset has been triggered.

SIG_COND_RESET When set to 1, this bit resets the signal paths for all sensors (gyroscopes, accelerometers, and temperature sensor). This operation will also clear the sensor registers. This bit automatically clears to 0 after the reset has been triggered.

When resetting only the signal path (and not the sensor registers), please use Register 104, SIGNAL_PATH_RESET.

4.28 Register 107 – Power Management 1
PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Description:

This register allows the user to configure the power mode and clock source. It also provides a bit for resetting the entire device, and a bit for disabling the temperature sensor.

By setting *SLEEP* to 1, the MPU-60X0 can be put into low power sleep mode. When *CYCLE* is set to 1 while *SLEEP* is disabled, the MPU-60X0 will be put into Cycle Mode. In Cycle Mode, the device cycles between sleep mode and waking up to take a single sample of data from accelerometer at a rate determined by *LP_WAKE_CTRL* (register 108). To configure the wake frequency, use *LP_WAKE_CTRL* within the Power Management 2 register (Register 108).

An internal 8MHz oscillator, gyroscope based clock, or external sources can be selected as the MPU-60X0 clock source. When the internal 8 MHz oscillator or an external source is chosen as the clock source, the MPU-60X0 can operate in low power modes with the gyroscopes disabled.

Upon power up, the MPU-60X0 clock source defaults to the internal oscillator. However, it is highly recommended that the device be configured to use one of the gyroscopes (or an external clock source) as the clock reference for improved stability. The clock source can be selected according to the following table.

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Stops the clock and keeps the timing generator in reset

For further information regarding the MPU-60X0 clock source, please refer to the MPU-6000/MPU-6050 Product Specification document.

Bit 4 is reserved.

Parameters:

DEVICE_RESET	When set to 1, this bit resets all internal registers to their default values. The bit automatically clears to 0 once the reset is done. The default values for each register can be found in Section 3.
SLEEP	When set to 1, this bit puts the MPU-60X0 into sleep mode.
CYCLE	When this bit is set to 1 and *SLEEP* is disabled, the MPU-60X0 will cycle between sleep mode and waking up to take a single sample of data from active sensors at a rate determined by *LP_WAKE_CTRL* (register 108).
TEMP_DIS	When set to 1, this bit disables the temperature sensor.
CLKSEL	3-bit unsigned value. Specifies the clock source of the device.

Note:

When using SPI interface, user should use DEVICE_RESET (register 107) as well as SIGNAL_PATH_RESET (register 104) to ensure the reset is performed properly. The sequence used should be:

1. Set DEVICE_RESET = 1 (register PWR_MGMT_1)
2. Wait 100ms
3. Set GYRO_RESET = ACCEL_RESET = TEMP_RESET = 1 (register SIGNAL_PATH_RESET)
4. Wait 100ms

4.29 Register 108 – Power Management 2
PWR_MGMT_2

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6C	108	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG

Description:

This register allows the user to configure the frequency of wake-ups in Accelerometer Only Low Power Mode. This register also allows the user to put individual axes of the accelerometer and gyroscope into standby mode.

The MPU-60X0 can be put into Accelerometer Only Low Power Mode using the following steps:

(i) Set CYCLE bit to 1
(ii) Set SLEEP bit to 0
(iii) Set TEMP_DIS bit to 1
(iv) Set STBY_XG, STBY_YG, STBY_ZG bits to 1

All of the above bits can be found in Power Management 1 register (Register 107).

In this mode, the device will power off all devices except for the primary I^2C interface, waking only the accelerometer at fixed intervals to take a single measurement. The frequency of wake-ups can be configured with *LP_WAKE_CTRL* as shown below.

LP_WAKE_CTRL	Wake-up Frequency
0	1.25 Hz
1	5 Hz
2	20 Hz
3	40 Hz

For further information regarding the MPU-6050's power modes, please refer to Register 107.

The user can put individual accelerometer and gyroscopes axes into standby mode by using this register. If the device is using a gyroscope axis as the clock source and this axis is put into standby mode, the clock source will automatically be changed to the internal 8MHz oscillator.

Parameters:

LP_WAKE_CTRL 2-bit unsigned value.
Specifies the frequency of wake-ups during Accelerometer Only Low Power Mode.

STBY_XA When set to 1, this bit puts the X axis accelerometer into standby mode.

STBY_YA When set to 1, this bit puts the Y axis accelerometer into standby mode.

STBY_ZA When set to 1, this bit puts the Z axis accelerometer into standby mode.

STBY_XG When set to 1, this bit puts the X axis gyroscope into standby mode.

STBY_YG When set to 1, this bit puts the Y axis gyroscope into standby mode.

STBY_ZG When set to 1, this bit puts the Z axis gyroscope into standby mode.

4.30 Register 114 and 115 – FIFO Count Registers FIFO_COUNT_H and FIFO_COUNT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
72	114	FIFO_COUNT[15:8]							
73	115	FIFO_COUNT[7:0]							

Description:

These registers keep track of the number of samples currently in the FIFO buffer.

These registers shadow the FIFO Count value. Both registers are loaded with the current sample count when FIFO_COUNT_H (Register 72) is read.

Note: Reading only FIFO_COUNT_L will not update the registers to the current sample count. FIFO_COUNT_H must be accessed first to update the contents of both these registers.

FIFO_COUNT should always be read in high-low order in order to guarantee that the most current FIFO Count value is read.

Parameters:

FIFO_COUNT 16-bit unsigned value. Indicates the number of bytes stored in the FIFO buffer. This number is in turn the number of bytes that can be read from the FIFO buffer and it is directly proportional to the number of samples available given the set of sensor data bound to be stored in the FIFO (register 35 and 36).

4.31 Register 116 – FIFO Read Write FIFO_R_W

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
74	116	FIFO_DATA[7:0]							

Description:

This register is used to read and write data from the FIFO buffer.

Data is written to the FIFO in order of register number (from lowest to highest). If all the FIFO enable flags (see below) are enabled and all External Sensor Data registers (Registers 73 to 96) are associated with a Slave device, the contents of registers 59 through 96 will be written in order at the Sample Rate.

The contents of the sensor data registers (Registers 59 to 96) are written into the FIFO buffer when their corresponding FIFO enable flags are set to 1 in FIFO_EN (Register 35). An additional flag for the sensor data registers associated with I^2C Slave 3 can be found in I2C_MST_CTRL (Register 36).

If the FIFO buffer has overflowed, the status bit *FIFO_OFLOW_INT* is automatically set to 1. This bit is located in INT_STATUS (Register 58). When the FIFO buffer has overflowed, the oldest data will be lost and new data will be written to the FIFO.

If the FIFO buffer is empty, reading this register will return the last byte that was previously read from the FIFO until new data is available. The user should check *FIFO_COUNT* to ensure that the FIFO buffer is not read when empty.

Parameters:

FIFO_DATA 8-bit data transferred to and from the FIFO buffer.

4.32 Register 117 – Who Am I
WHO_AM_I

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
75	117	-	WHO_AM_I[6:1]						-

Description:

This register is used to verify the identity of the device. The contents of *WHO_AM_I* are the upper 6 bits of the MPU-60X0's 7-bit I^2C address. The least significant bit of the MPU-60X0's I^2C address is determined by the value of the AD0 pin. The value of the AD0 pin is not reflected in this register.

The default value of the register is 0x68.

Bits 0 and 7 are reserved. (Hard coded to 0)

Parameters:

WHO_AM_I Contains the 6-bit I^2C address of the MPU-60X0.

The Power-On-Reset value of Bit6:Bit1 is 110 100.