

# Desarrollo de comunicación HRP con antenas RFID Clou/Hopeland

Arturo Hernandez Calleja<sup>1\*</sup>

## Resumen

La identificación de etiquetas por Radio-Frecuencia, se utiliza actualmente en Bolivia en dos sistemas importantes, el B-SISA de la ANH, y el SOAT, administrado actualmente por Univida, usándose en ambos casos etiquetas que trabajan en UHF bajo el estándar ISO18000-6C EPC Class 1 Gen 2 y pudiéndose utilizar antenas que trabajen entre 915 a 930MHz. En este caso se estudió el protocolo HRP para antenas Clou/Hopeland.

## Palabras Clave

RFID — UHF — ISO18000-6

<sup>1</sup> Profesional de Investigación, Area de Investigación, Unidad de Innovación Investigación y Desarrollo, AGETIC

\*Contacto: ahernandez@agetic.gob.bo

## Índice

Introducción	1
1 Objetivo	1
2 Antecedentes	1
3 Justificación	2
4 Marco teórico	2
5 Metodología	4
6 Resultados	8
7 Conclusiones	8
Referencias	8

## Introducción

La tecnología RFID utiliza campos electromagnéticos para identificar y rastrear etiquetas (tags) adheridas a objetos. Estas etiquetas contienen un circuito electrónico y una antena por la cual se energiza y se comunica permitiendo el intercambio de información de forma inalámbrica <sup>1</sup>.

Existen diversos estándares y frecuencias para esta tecnología, dependiendo del uso y la distancia que se requiere, entre ellos algunos son:

- 120 kHz identificación simple de 32 bits hasta 10[cm]
- 134 kHz similar al anterior, pero utilizado para identificación de mascotas
- 13.56 MHz entre 10[cm] a 1[m] utilizado sobre todo con tarjetas inteligentes (Smartcards) (ISO 15693, ISO 14443) y otros (Mifare, iCLASS) así como ISO28560 para bibliotecas.

<sup>1</sup> Existen también etiquetas activas que poseen alimentación propia mediante una batería, permitiendo mayor alcance que una etiqueta pasiva

- 860 a 960MHz utilizado para identificación a mayor distancia (hasta 12[m]) (ISO18000-6) para manejo de almacenes e identificación vehicular.

## 1. Objetivo

Desarrollar un módulo de comunicación con antenas RFID Clou/Hopeland y probar la lectura de vehículos mediante sus etiquetas B-SISA y SOAT.

## 2. Antecedentes

La ANH como institución responsable de regular, controlar, supervisar y fiscalizar las actividades relacionadas con hidrocarburos <sup>2</sup> tiene como obligación la implementación de un sistema de Información de Comercialización de combustibles, a través de (...) el colocado de etiquetas de auto-identificación en todo vehículo automotor que circule en territorio nacional. <sup>3</sup> el cual se fue aprobado y reglamentado como Sistema de Información y Comercialización de Combustibles (B-SISA) el 2015 <sup>4</sup>.

Figura 1. Etiqueta B-SISA



Fuente: ANH

<sup>2</sup> Constitución política del Estado, Artículo 365

<sup>3</sup> Ley 264, Ley del Sistema Nacional de Seguridad ciudadana "Para una vida segura", Artículo 49

<sup>4</sup> RA-ANH-DJ 0102/2015 y RAN-ANH-UN 0012/2015

El Decreto Supremo N°2920 de 28 de septiembre de 2016, establece que el Seguro Obligatorio de Accidentes de Tránsito - SOAT será administrado y comercializado por la Entidad Pública de Seguros, y por la Resolución Administrativa AP-S/DJ/DS/N° 1517-2016 de 17 de octubre de 2016, se habilitó a la Compañía Seguros y Reaseguros Personales Univida S.A. en su condición de Entidad Pública de Seguros, para que asuma la administración y comercialización del SOAT desde la Gestión 2017.

**Figura 2.** Etiqueta SOAT



Fuente: APS

### 3. Justificación

Actualmente existen antenas RFID UHF que pueden leer la información de identificación de un tag (EPC y TID) como ser los FX7500 o FX9600 de Zebra(Motorola) mediante el protocolo LLRP, pero no tiene una solución integral, es decir, que se requiere un controlador y aparte las antenas de comunicación. Otras soluciones Integrales como el ZK-RFID-101 tiene un controlador limitado, que sólo permite leer 4 bytes del EPC. Clou/Hopeland tiene antenas integradas con capacidad de leer todo el tag y tiene un protocolo documentado denominado HRP que se puede implementar en cualquier lenguaje de programación<sup>5</sup>

### 4. Marco teórico

Algunos de los conceptos clave para entender esta tecnología y poder aplicarla son: el formato de las etiquetas y la comunicación HRP.

#### Identificación en UHF (860 a 960MHz)

Estas etiquetas se definieron originalmente por EPCGlobal como el protocolo de identificación por radiofrecuencia de

Código de Producto Electrónico (EPC) clase 1, 2da generación [1], para trabajar en 860 a 960MHz<sup>6</sup> este protocolo fue aprobado como parte de la ISO18000-6 para manejo de ítems por radiofrecuencia.

En sí el protocolo define un sistema compuesto por interrogadores (llamados también lectores – *Readers*) y las etiquetas o *Tags*. Un lector transmite información a la etiqueta al modular señal de Radio Frecuencia entre 860 a 960MHz. La etiqueta recibe información y energía de dicha señal modulada, lo que indica que este sistema utiliza una etiqueta pasiva. Para recibir información de la etiqueta, el lector envía una señal de onda continua (CW), donde la etiqueta responde modulando el coeficiente de reflexión de su antena, retro-dispersando información al lector. La comunicación se realiza en semidúplex con inicio en el lector, es decir que se alterna la comunicación entre el lector y las etiquetas, siempre iniciado por el lector.

#### Áreas de memoria de una etiqueta EPC

Estas etiquetas tienen definidas 4 áreas de memoria:

**Memoria reservada** (64 bits) donde se almacena las contraseñas de acceso y de bloqueo ambos de 32 bits, en una etiqueta nueva, ambas contraseñas son 00000000

**Memoria EPC** ( $\geq 128$  bits) es donde se encuentra el código de producto electrónico (EPC). Los primeros 16 bits son para la verificación de redundancia cíclica (CRC) los siguientes 16 bits son de control de protocolo (PCW), que indican algunas características como ser el tamaño y el uso del EPC, y los siguientes bits corresponden al EPC como tal (normalmente se tiene un EPC de 96 bits) Cabe mencionar que en una etiqueta nueva el EPC es reescribible. Opcionalmente en este espacio de memoria se implementa el Control de protocolo Extendido (XPC) que define el tipo de bloqueo que tiene la etiqueta.

**Memoria TID** ( $\geq 96$  bits). Corresponde al identificador de etiqueta (Tag), este es un código de 96 bits (32 bits de identificación del fabricante/modelo y 64 bits de identificador único) que es inalterable y único en el mundo (opcionalmente puede tener otros 96 bits adicionales para configuración del dispositivo)

**Memoria de Usuario** dependiendo de la etiqueta, hasta 512 bits, por defecto con acceso libre.

Las contraseñas de acceso, permite limitar la escritura e incluso la lectura de determinadas áreas, mientras que la contraseña de bloqueo (kill password) permite “destruir” el tag (bloqueando el tag e inutilizado completamente)<sup>7</sup> y cabe mencionar que los tags de B-SISA y SOAT tienen sus respectivas contraseñas de acceso y bloqueo

<sup>6</sup>Banda ISM según la UIT, pero restringida parcialmente en otros países como Bolivia, donde sólo la banda de 915 a 930 se puede utilizar en aplicaciones de redes privadas [2] BOL19

<sup>7</sup>algunas etiquetas, permite recomisionar el tag, mandando el comando de bloqueo 2 veces consecutivas lo que permite eliminar la información de usuario. Pero una etiqueta ya bloqueada no se puede recomisionar.

<sup>5</sup>Ya existen librerías para .NET y JAVA del propio distribuidor. pero con una implementación limitada que dificulta la lectura simultánea del SOAT y B-SISA

## Protocolo HRP

Las antenas Clou/Hopeland implementan un protocolo para sus lectores denominado Hopeland Reader Protocol (HRP) que permite interactuar entre un controlador (ordenador PC o similar) y el lector mediante una interfaz serial (RS232 – RS485 – USB) o Ethernet (TCP).

**Cuadro 1.** Trama HRP

HEAD	PCW	ADR	LENGTH	DATA	CRC
------	-----	-----	--------	------	-----

Fuente: *The HRP Protocol* [3]

La trama de datos se envía en formato big-endian (byte más significativo primero) y consiste en: un byte de cabecera consistente en 0xAA, 2 bytes de palabra de control de protocolo (PCW) donde se tiene los siguientes bits:

**Cuadro 2.** Palabra de Control

Bits	Definición	Descripción
15-14	Reservado	Mantener en 0
13	RS485	Utiliza ADR
12	Origen	0: indica un mensaje originado en el controlador o como respuesta 1: mensaje iniciado por el lector
11-8	Tipo	0: Error o advertencia 1: mensaje de Configuración 2: mensaje de Operación 3: mensaje de bitacora 4: mensaje de actualización 5: mensaje de prueba
7-0	Identificador	depende del tipo de mensaje

Fuente: *The HRP Protocol* [3]

El bit 13 indica si la comunicación se realiza por 485 que indica que existe y se utiliza la dirección de 8 bits ADR. Si este bit es 0 (para comunicación TCP) el byte ADR no existe. El bit 12 de Origen normalmente es 0 para cualquier comando emitido por el PC controlador y su respectiva respuesta del lector. Sólo cuando se inicia un proceso de lectura continua, el lector emite tramas automáticamente con cada tag detectado y con este bit en 1.

El Tipo e Identificador de mensajes (denominados MT y MID) permite diferenciar los distintos Comandos que se verán más adelante.

La Longitud (LENGTH) de los datos es un número de 16 bits <sup>8</sup> que indica el tamaño del siguiente campo de Datos. Finalmente viene la Verificación de redundancia cíclica que es otro número de 16 bits, que adopta el algoritmo CRC-16/UMTS (polinomio 0x8005 con inicio 0x0000) sobre todo el paquete exceptuando la cabecera 0xAA.

Por ejemplo el comando de reinicio de dispositivo corresponde al tipo MT=0x1 MID=0x0F y no tiene ningún parámetro por lo que genera una trama AA 01 0F 00 00 94 CF. Este es el único comando que no genera una respuesta del

lector, sino más bien reinicia el equipo cortando la comunicación.

## Parámetros

La sección de datos contienen los parámetros de cada comando, y se divide en dos tipos: parámetros obligatorios (M - mandatorio), que deben ser enviados en el orden indicado, y parámetros opcionales que se deben enviar con un identificador de parámetro (PID) por delante. También en ambos casos existe parámetros de longitud variable, donde se antecede al parámetro con dos bytes de tamaño de parámetro.

Por ejemplo uno de los comandos más importantes es el de lectura de etiquetas.

**Cuadro 3.** Lectura de tags MT = 0x02 MID = 0x10

Nombre	PID	tipo	descripción
Antenas	(M)	U8	máscara de antenas
modo	(M)	U8	0: un solo barrido 1: modo continuo
Filtro	0x01	U8 var	Byte 0: área (1:EPC,2:TID) Byte 1 y 2: inicio Byte 3: longitud Byte 4 +: dato
Leer TID	0x02	U8 x2	Byte 0: modo de lectura Byte 1: tamaño a leer
Usuario	0x03	U8 x3	Byte 0 + 1: inicio Byte 2: tamaño a leer
Reservado	0x04	U8 x3	Byte 0 + 1: inicio Byte 2: tamaño a leer
Contraseña	0x05	U32	para autenticación
Datos EPC	0x09	U8 x3	Byte 0 + 1: inicio Byte 2: tamaño a leer

Fuente: *The HRP Protocol* [3]

Tiene dos parámetros obligatorios, máscara de Antenas, donde cada bit habilita una antena correspondiente, por ejemplo el valor 0x05 o 0b0101 habilita la antena 1 y 3 para la lectura. el otro parámetro obligatorio, es el modo de lectura, donde 0 corresponde a un solo barrido (y el lector responderá con todos los tags que encuentre o con fin de lectura) o barrido continuo, donde el lector enviará una respuesta cada vez que detecte un tag, hasta que se le envíe el comando de parada (Stop)

El Filtro es un comando variable, (se debe anteponer el tamaño del subpaquete generado) se compone de 4 campos, el primer byte es el área a filtrar, se puede filtrar por EPC, TID o memoria de usuario. los siguientes 16 bits corresponde a la dirección de inicio (en bits) para la comparación, el siguiente byte es la longitud en bits a comparar, y a continuación los datos que se desea utilizar de filtro. Por ejemplo si quisiéramos filtrar la lectura para leer solamente los tags con EPC <sup>9</sup> que

<sup>8</sup> Actualmente el límite del protocolo es de 1024 bytes

<sup>9</sup> recordemos que la parte útil del EPC empieza en el bit 32 o 0x20

empiecen con 010190c00100 se debe enviar como parámetro 0x01, 00 0A 01 00 20 30 01 01 90 c0 01 00.

Los siguientes parámetros corresponden a lectura de ciertas áreas de memoria, para leer el TID se debe enviar dos bytes, el primer byte indica al lector si hará una lectura hasta un máximo de N words o si forzará la lectura de N words, el segundo byte es el tamaño (N) en words (palabras de 16 bits), normalmente el TID es el 6 words = 96 bits por tanto para leer el TID se debe enviar como parámetro 0x02, 00 06

Para leer el área reservada, el área de usuario o el área del EPC, se envía 3 bytes adicionales, Los primeros dos corresponden a la dirección de inicio de lectura (en 16 bits) y el tercer byte corresponde al tamaño en words que se desea leer.

Luego de enviar el comando, el lector responde inmediatamente con un acuse con el mismo MID=0x10 y un solo argumento obligatorio, el cual es 0 si se aceptó el comand, 1 si se activó una antena inexistente, 2 si hubo error en el parámetro del filtro, 3 si hubo error en el parámetro del TID, 4 si hubo error en el parámetro de usuario, 5 para el parámetro de memoria reservada, y 6 para un error en los otros parámetros.

Si se detecta un tag (ya sea en modo simple o en modo continuo) el lector enviará una trama de respuesta:

**Cuadro 4.** Tag leído MT = 0x12 MID =0x00

Nombre	PID	tipo	descripcion
EPC	(M)	var	EPC leído
PCW	(M)	U16	palabra de control
Antena	(M)	U8	origen
RSSI	0x01	U8	Intensidad de señal
Resultado	0x02	U8	0: lectura exitosa
			1: sin respuesta
			2: error CRC
			3: area bloqueada
			4:desborde
			5: error de acceso
			6:otro error de tag
			7: error del lector
TID	0x03	var	dato del TID
Usuario	0x04	var	dato de área usuario
Reservado	0x05	var	dato de area de usuario
Datos EPC	0x0C	var	dato del área EPC

Fuente: *The HRP Protocol* [3]

Para una lectura simple el lector enviará solamente el EPC detectado y su PCW, al ser de tamaño variable el EPC se devuelve con dos bytes adicionales del tamaño del EPC, también se envía la antena que realizó la lectura y como primer parámetro opcional la intensidad de la señal de lectura <sup>10</sup>.

El Resultado nos indica si hubo algún problema al leer el resto de datos del tag, si es 0 los siguientes parámetros contienen la información leída si ésta hubiese sido solicitada previamente, al ser todos los campos de datos variables, se

anteponen de 2 bytes del tamaño correspondiente.

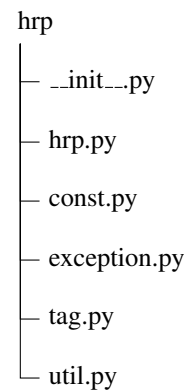
Ya sea si fue un simple barrido, o luego de finalizar el modo continuo, el lector envía una trama de finalización con MT=0x12 y MID=0x01 con un solo argumento que es 0: si la lectura de barrido simple finalizó correctamente, 1, si se recibió el comando de parada (stop) o 2, si hubo algún error para interrumpir la lectura. Por ejemplo una lectura simple finaliza con la trama AA 12 01 00 01 00 15 70.

El comando de parada no tiene argumento y corresponde a MT=0x2 MID=0xFF AA 02 FF 00 00 A4 0F y su correspondiente respuesta tiene un argumento que es 0 si el comando fue aceptado o 1 si hay un error en el sistema, normalmente la respuesta es 0: AA 02 FF 00 01 00 0  
↪ A D8.

## 5. Metodología

La implementación se realizó en Python para facilitar las pruebas rápidas y aprovechar la experiencia en manejo de sockets[4].

**Figura 3.** Estructura del módulo



Fuente: Elaboración propia

se implementó un módulo llamado `hrp`, donde existen varios archivos:

**const.py** contiene las definiciones de constantes para el protocolo HRP, como ser los tipos de mensajes y los identificadores de comandos.

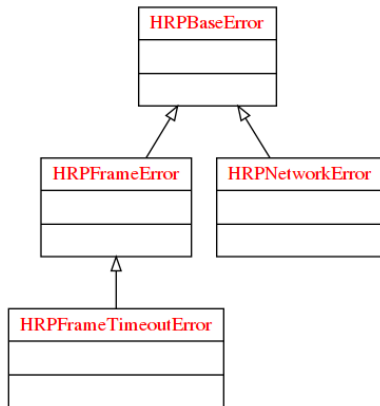
**exception.py** define las excepciones de éste módulo: la excepción base, `HRPBaseError` es para capturar cualquier error del módulo, `HRPNetworkError` indica si hubo un error en la comunicación del socket (pérdida de comunicación por ejemplo), `HRPFrameError` indica si hubo un error en la trama (respuesta incorrecta o no esperada), finalmente `HRPFrameTimeoutError` indica que no se recibió la respuesta a un comando a tiempo.

**util.py** contiene algunas funciones de bitácora, la función `_ord()` para compatibilidad entre python2 y python3, y la función `crc16(data)`, para el cálculo de la verificación de redundancia <sup>11</sup>.

<sup>10</sup>Lo que permite en otras aplicaciones determinar la posición de un tag a partir de la intensidad de lectura en diversas antenas

<sup>11</sup>actualmente la implementación sólo genera esta verificación para enviar datos al lector, aún no implementa una verificación de la integridad de la

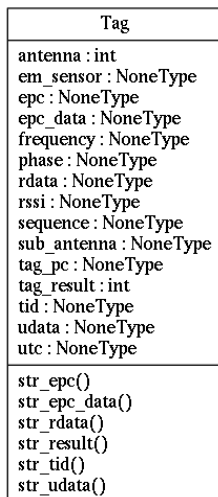
Figura 4. hrp.exception



Fuente: Elaboración propia

**tag.py** define la clase Tag, y las clases para los parámetros de filtro o lectura de datos.

Figura 5. hrp.Tag



Fuente: Elaboración propia

Tiene como argumentos para la creación de un tag: la antena origen (antenna) y su RSSI (rssi), el EPC (epc), el PCW (tag\_pc) y el resultado (tag\_result) de la lectura, el resto de los argumentos son opcionales (tid, udata, rdata, epc.data, etc) y son poblados dependiendo el tipo de lectura que se realizó. Para la representación en texto (**str()**) dispone métodos de apoyo para cada tipo de argumento.

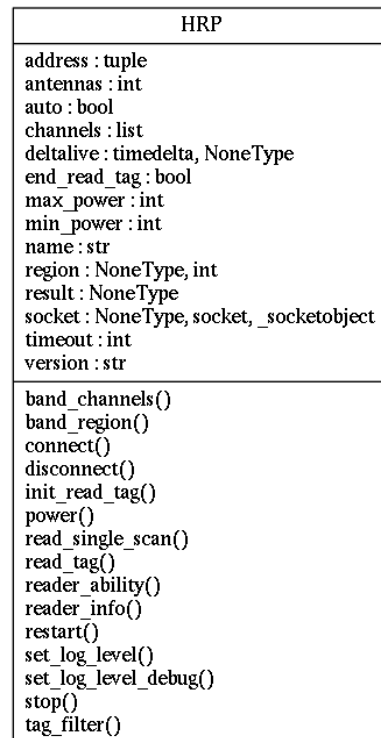
**hrp.py** define la clase HRP que se encarga de realizar todas las comunicaciones correspondientes con el lector

la inicialización de la clase tiene como argumentos: la (ip) o host del lector, (port) o el puerto de comunicación (por defecto 9090), (ommit\_ping) para omitir realizar un ping antes de iniciar a comunicación <sup>12</sup>, (timeout) que indica el tiempo

trama, ya que al utilizar TCP no es muy necesario

<sup>12</sup>Se agregó esta opción para permitir la implementación en contenedores que no posean el comando ping, o en S.O. que bloqueen el uso de este

Figura 6. hrp.HRP



Fuente: Elaboración propia

de espera al realizar la comunicación TCP (por defecto espera 10s antes de emitir un error de HRPFrameTimeoutError).

Para iniciar la comunicación con el lector se utiliza el método `connect()` el cual verifica la conexión ejecutando un ping, y luego crea el socket. A continuación, lee la información del lector `reader_info()` que puebla las propiedades de `version`, `name` y `deltalive`, con la versión del firmware, el nombre del lector y el tiempo de encendido del mismo. Luego obtiene la información de potencias `reader_ability()`, que puebla las propiedades de `min.power`, `max.power`, `antennas`, con las potencias mínimas y máximas soportadas, la máscara de antenas disponibles y retorna la lista de bandas y protocolos disponibles. Y luego obtiene la región actual con `band_region()` poblando a propiedad `region`, y los canales habilitados con `band_channels()` y poblando `channels`, lo cual permite verificar el uso correcto del ancho de banda y el espectro permitido en Bolivia. Finalmente lee la potencia actual de cada antena con el método `power()`.

Existe dos métodos para realizar la lectura de etiquetas, `read_single_scan()` y `read_tag()`, ambos métodos tienen los mismos argumentos, (antenna) para indicar la máscara de antenas <sup>13</sup>, (match) es un argumento del tipo `MatchParameter()` para indicar el filtro a aplicar en la lectura <sup>14</sup>, el parámetro

comando

<sup>13</sup>Donde cada bit corresponde a que antena estará habilitada para el escaneo

<sup>14</sup> `MatchParameter` tiene tres argumentos (area, start, content) donde área es `MATCH.EPC` o `MATCH.TID` para indicar el área a aplicar el filtro, start es el bit de inicio para la comparación y content es el contenido en cadena



(tid) es un argumento del tipo `TidReadParameter()` que indica el tipo y tamaño de lectura del TID, los parámetros (udata, rdata, edata) son parámetros tipo `TagAddress()` que indican el inicio y el tamaño en palabras para leer las áreas de memoria de usuario, Reservado y EPC respectivamente <sup>15</sup>, en caso de requerirse, se puede dar el parámetro de (password) para indicar la contraseña de acceso.

La diferencia entre los métodos `read.single_scan()` y `read.tag()` es que `read.single_scan` realiza un solo barrido con la antena, y devuelve una lista [Tag] con todos las etiquetas que han sido detectadas, esta lista estará vacía si no se detectó ninguna etiqueta. el método `read.tag()` retorna un generador <sup>16</sup> que se utiliza para obtener interactivamente instancias de Tag cuando son detectadas etiquetas, y None si no detecta ninguna durante timeout. Para finalizar el generador se debe colocar la propiedad `end_read_tag` en True <sup>17</sup>.

Para finalizar la comunicación con el lector se utiliza el método `disconnect()`, el cual permite detener cualquier lectura en espera, y cierra el socket TCP.

## Ejemplos

Se ha preparado tres ejemplos simples, que se encuentran en la carpeta `examples/`. El ejemplo básico, utiliza el generador `read.tag()`

---

```
from hrp import HRP, const, exception

try:
    conn = HRP('192.168.1.116', 9090)
    print ("Connectando")
    conn.connect()
    print ("presione Ctrl+C para terminar")

    counter = 0
    for tag in conn.read.tag(
        antenas=const.ANTENNA_1|const.ANTENNA_2):
        if tag is None:
            print ("Time out, {}".format(counter))
            counter += 1
            if counter > 10:
                conn.end_read_tag = True
        else:
            print (tag) #tag valido
except Exception as e:
    print ("Process terminate : {}".format(e))
finally:
    print ("Desconectando, bye!")
    conn.disconnect()
```

---

de bytes a comparar, como limitante de la implementación, solo se pueden comparar bytes enteros

<sup>15</sup>Permitiendo este parámetro edata, leer los datos adicionales del área EPC, que se utiliza para leer la información de los Tags SOAT

<sup>16</sup>Un tipo de función de Python, que permite devolver un resultado dinámicamente, para un consumo secuencial [5]

<sup>17</sup>O generar una interrupción de teclado o finalizar el sistema

## Filtrado de B-SISA y SOAT

Luego de realizar pruebas se procedió a realizar lecturas esporádicas de vehículos en tránsito. posteriormente, la ANH confirmó la siguiente estructura del EPC de las etiquetas B-SISA

**Cuadro 5.** EPC del B-SISA

Header	01
Filter	01
Partition	98
Company	C00100
Class	XXXXXX
Serial	YYYYYY
Fuente: ANH	

la Clase corresponde al número de la etiqueta (codificado en mod10 para el código de barras) pero se puede notar que el inicio para el B-SISA siempre será 010198C00100.

Una forma simple de leer solamente etiquetas de B-SISA es:

---

```
import codecs
from hrp import HRP, const, exception
from hrp.tag import TidReadParameter, TagAddress
from hrp.tag import MatchParameter

BSISA_START = codecs.decode("010198C00100", "hex")

try:
    conn = HRP('192.168.1.116', 9090)
    print ("Connectando")
    conn.connect()
    print ("presione Ctrl+C para terminar")

    counter = 0
    for tag in conn.read.tag(
        antenas=const.ANTENNA_1|const.ANTENNA_2,
        match=MatchParameter(const.MATCH_EPC,
                               0x20, BSISA_START),
        tid=TidReadParameter(0, 6)):
        if tag is None:
            print ("Time out, {}".format(counter))
            counter += 1
            if counter > 10:
                conn.end_read_tag = True
        else:
            print (tag) #tag valido
except Exception as e:
    print ("Process terminate : {}".format(e))
finally:
    print ("Desconectando, bye!")
    conn.disconnect()
```

---

En este caso, utilizamos el Filtro `MatchParameter` para indicar que sólo lea EPC empezando con la secuencia B-SISA, además de leer el TID de dicha etiqueta.

Para el caso del SOAT, las pruebas mostraron que el EPC

que devuelve es único y corresponde a 00000000 00000024 00000000, esta etiqueta es un caso particular, puesto que el EPC reportado es de 96 bits, pero los datos que corresponden al código de barras se encuentran también en el área de memoria EPC, por lo que debemos leer datos adicionales de esa área.

---

```
import codecs
from hrp import HRP, const, exception
from hrp.tag import TidReadParameter, TagAddress
from hrp.tag import MatchParameter

SOAT_START = codecs.decode("0000000000000024",
                             'hex')

try:
    conn = HRP('192.168.1.116', 9090)
    print ("Connectando")
    conn.connect()
    print ("presione Ctrl+C para terminar")

    counter = 0
    for tag in conn.read_tag(
        antenas=const.ANTENNA_1|const.ANTENNA_2,
        match=MatchParameter(const.MATCH_EPC,
                               0x20, SOAT_START),
        tid=TidReadParameter(0, 6),
        edata=TagAddress(0x08, 2)):
        if tag is None:
            print ("Time out, {}".format(counter))
            counter += 1
            if counter > 10:
                conn.end_read_tag = True
        else:
            print (tag) #tag valido
except Exception as e:
    print ("Process terminate : {}".format(e))
finally:
    print ("Desconectando, bye!")
    conn.disconnect()
```

---

en este caso, el Tag leído tendrá un campo adicional epc.data con los 4 bytes que corresponden al código de barras de la etiqueta. al intentar leer campos adicionales del EPC\_data, impide la correcta lectura de B-SISA pudiéndose solamente obtener el EPC del mismo<sup>18</sup>.

Si bien ambos tags sólo permiten su autoidentificación (corroborar la lectura con el código de barras) ninguno aprovecha el espacio de memoria de datos, para almacenar datos como ser la placa u otros datos del vehículo, el SOAT tiene planificado implementar dicha funcionalidad. para ello sólo se requerirá leer los campos de datos del tag:

---

```
...
for tag in conn.read_tag(
```

---

<sup>18</sup>Ya que responde con un tag\_result=6 (Invalid data) y no devuelve el valor del TID

```
antenas=const.ANTENNA_1|const.ANTENNA_2,
match=MatchParameter(const.MATCH_EPC,
                      0x20, SOAT_START),
tid=TidReadParameter(0, 6),
udata=TagAddress(0x28, 44),
edata=TagAddress(0x08, 2)):

...
```

---

El campo de datos del SOAT tendrá dos espacios, un área codificada que contendrá la placa y otros datos del vehículo, y otra área con datos como color, modelo y año del vehículo.

---

```
...
if tag.udata:
    #quitar espacios
    data = str(tag.udata).strip()
    #dividir espacio codificado
    cdata = data.split('|')
    pdata = base64.b64decode(cdata[0])
    #placa vehiculo
    placa = pdata.split('>')[0]
```

---

Si bien el despliegue de dichos campos aún no está en el parque automotor<sup>19</sup>.

### Lectura de B-SISA y SOAT

Para leer ambos tipos de etiquetas al mismo tiempo, se optó por una lectura más secuencial, leyendo cualquier tag, y forzando una segunda lectura en caso de encontrar un SOAT.

---

```
import codecs
from hrp import HRP, const, exception
from hrp.tag import TidReadParameter, TagAddress
from hrp.tag import MatchParameter
```

---

```
BSISA_START = codecs.decode("010198C00100", "hex")
SOAT_START = codecs.decode("0000000000000024",
                             'hex')
```

```
try:
    conn = HRP('192.168.1.116', 9090)
    print ("Connectando")
    conn.connect()
    print ("presione Ctrl+C para terminar")

    counter = 0
    while True:
        tags = conn.read_single_scan(
            antenas=const.ANTENNA_1|const.ANTENNA_2,
            tid=TidReadParameter(0, 6)):
        if not tags:
            print ("Time out, {}".format(counter))
            continue
        for tag in tags:
            if tag.epc[:len(SOAT_START)] ==
                ↳ SOAT_START:
```

---

<sup>19</sup>Sólo en las etiquetas de prueba provistas por Quipus

```

soat = conn.read_single_scan(
    antennas=const.ANTENNA_1 |
        const.ANTENNA_2,
    match=MatchParameter(
        const.MATCH_TID,
        0x0, tag.tid),
    tid=TidReadParameter(0, 6),
    edata=TagAddress(0x08, 2))
if soat and soat[0].tag_result == 0:
    tag = soat[0] # reemplazar
    print (tag) #tag valido
except Exception as e:
    print ("Process terminate : {}".format(e))
finally:
    print ("Desconectando, bye!")
    conn.disconnect()

```

## Pruebas de Campo

para facilitar las pruebas de campo se desarrolló una aplicación web mediante FLASK y SOCKET.IO generando una pequeña interfaz web y generando un websocket cada vez que se detecta un tag válido, mostrándose en tiempo real en la interfaz web. También se aprovechó para almacenar los datos registrados en una base de datos PostgresQL

### Figura 7. Captura Tags


Captura de Tags

Fecha Inicial

2018-12-27

Fecha Final

2018-12-27

 VER

Reportes de Marcado.

<div>Registrado</div> <div>94001058</div>	2018-12-31 13 44 04.813985	OTRO	94001058	detalle
	2018-12-31 13 43 50.255113	B-SISA	068020328958	detalle
	2018-12-27 17 40 38.266940	B-SISA	068006671710	detalle
	2018-12-27 17 40 18.617622	SOAT	"0613156"	detalle
	2018-12-27 17 40 18.047836	B-SISA	0680 13431231	detalle
	2018-12-27 17 40 14.127804	B-SISA	0680 13267817	detalle
	2018-12-27 17 40 13.687984	B-SISA	068008043918	detalle
	2018-12-27 17 40 6.850595	B-SISA	0680 10496989	detalle
	2018-12-27 17 40 6.811096	SOAT	"0057567"	detalle
	2018-12-27 17 40 6.811096	SOAT	"0973513"	detalle
	2018-12-27 17 40 06.948683	B-SISA	0680080277862	detalle
		B-SISA	0680 13802704	detalle
		B-SISA	"0903936"	detalle
	2018-12-27 17 40 00.728246	B-SISA	068010417244	detalle

Fuente: Elaboración Propia

**Figura 8. Captura Tags: Detalle**

Fecha Inicial:	
Reportes de	
2018-12-31 13:4	Fecha: 2018-12-31 13:44:19 834652
2018-12-31 13:4	Antena : 1
2018-12-31 13:4	BD ID : 8366
2018-12-31 13:4	COD : *0434335*
2018-12-27 17:4	CAP : 434335
2018-12-27 17:4	EPC: 0000000000000002400000000
2018-12-27 17:4	TID: e280b0a02000000001228804c
2018-12-27 17:4	Placa: no aplica
2018-12-27 17:4	Servicio: no aplica
2018-12-27 17:4	
2018-12-27 17:40 06 850959	B-SISA
2018-12-27 17:40 06 818096	SOAT
2018-12-27 17:40 05 048893	SOAT
2018-12-27 17:40 05 088046	B-SISA
2018-12-27 17:40 02 189421	B-SISA
2018-12-27 17:40 01 709776	SOAT

Fuente: Elaboración Propia

## 6. Resultados

Con el sistema funcionando, se realizó una prueba en la calle Ecuador esquina Pedro Salazar, registrando los vehículos de subida y comparando con una grabación de los vehículos para ver la ubicación de las etiquetas.

De 102 vehículos que circularon en un lapso de 15 minutos, sólo se indentificó 57 lecturas de SOAT (56%) y 72 lecturas de B-SISA (70%) pero tras un análisis del video tomado, se identificó sólo 88 vehículos con el SOAT visible e incluso 67 con el SOAT instalado en una posición óptima, en cuyos casos, el porcentaje sube a 65% y 91% respectivamente. para el B-SISA, los vehículos con etiquetas visibles fueron 85 y 51 con el B-SISA bien ubicado, alcanzándose porcentajes de éxito de 82% y 96%. combinando ambos tipos de etiquetas, se tiene que: al intentar leer ambos SOAT y B-SISA al mismo tiempo, solamente el 37% de los 102 se pudo leer ambas etiquetas, pero si éstas estaban ubicadas en una posición óptima (debajo del retrovisor en medio del parabrisas) el alcance es de 85%. En cambio si se intenta leer al menos una de las dos etiquetas, se percibe al menos un 89% de los 102 vehiculos, y si las etiquetas están en posición óptima, un 100% de detección.

## 7. Conclusiones

Luego de realizar algunas otras pruebas con resultados similares, se puede concluir que esta antena con este protocolo, puede ser utilizado para leer las etiquetas del B-SISA y/o el SOAT (incluso ambas al mismo tiempo), pero que las etiquetas no se encuentran instaladas de forma óptima en el parque automotor, ya que en varios casos se encuentran en los extremos del parabrisas e incluso en algunos casos los vehículos carecen de una o ambas etiquetas.

## Referencias

- [1] EPCGlobal. *EPC Radio Frequency Identity Protocols*, volume Class 1 Generation 2 UHF RFID. EPCGlobal Inc, 2008.
- [2] Viceministerio de Telecomunicaciones. *Plan Nacional de Frecuencias*. Ministerio de Obras públicas, 2012.
- [3] Hopeland. *Data Communication Protocol*. Number 1.4. Shenzhen Hopeland Technologies Ltd, 2016.
- [4] Nathan Jennings. Socket programming in python. *Real Python*, 2018.
- [5] Radu Raicea. How and why you should use python generators. *freeCodeCamp*, 2017.