Control de activos mediante el uso de tecnologías RFID y Raspberry Pi

Ing. Ramón Omar Parra Guerrero, Ing. Julián Alejandro Galindo Carlton, M.C. Fredy Alberto Hernández Aguirre, M.C. Jesús Manuel Tarín Fontes, M.C. Héctor Martín Córdova Córdova
Tecnológico Nacional de México, Instituto Tecnológico de Hermosillo,
División de Estudios de Posgrado e Investigación, Departamento de Ingeniería Eléctrica y Electrónica
Avenida Tecnológico S/N, C.P. 83170, Col. El Sahuaro; Hermosillo, Sonora, México,
Teléfono (6622) 2606500
Correos electrónicos: ramon@rparra.me, julian_ro21@hotmail.com, faha.singapur@gmail.com, jesustarin55@hotmail.com, hemacoco@gmail.com

Resumen: Actualmente, una realidad a la que se enfrentan muchas empresas en la gestión eficiente, inteligente, y segura de sus activos, es la demora y errores humanos en la generación de inventarios, la seguridad deficiente en bodegas, y la logística al momento de ubicar determinados elementos de sus activos. Además, el robo y pérdida de mobiliario y equipo electrónico representa un gasto significativo para las empresas. Para contrarrestar esta necesidad existen distintas soluciones que involucran personal de seguridad y diversos métodos de registro y revisión; estrategias que no solo han resultado ser poco efectivas, sino que además representan un costo a las empresas.

En respuesta a esta problemática se propone un sistema que automatice la generación de inventarios, proporcione seguridad y facilite la ubicación de los activos de las empresas, mediante la integración de las tecnologías de identificación por radiofrecuencia (RFID, *Radio Frequency Identification*) y Raspberry Pi para el desarrollo de un software que permita la concentración de la información recopilada para la administración y gestión de los activos de las empresas, impactando en la toma de decisiones relacionadas con los inventarios, localización, seguridad y logística de los mismos.

El objetivo del presente artículo es dar a conocer los resultados de la investigación realizada durante el desarrollo de la plataforma para el control de activos, presentando las principales características y procedimientos de lectura-escritura de datos en la tarjeta RFID utilizada, por medio de la tarjeta lectora PN532 RFID *Reader* controlada por una plataforma Arduino uno. Además, se analiza la arquitectura utilizada para la administración de la base de datos de los activos a través de un servidor desarrollado en Raspberry Pi, y el diseño de una interfaz gráfica para visualización de los datos por el usuario. **Palabras clave:** RFID, Raspberry, *tag* (etiqueta), Arduino, interfaz de usuario.

1. Introducción

El presente artículo trata acerca del uso de la plataforma Arduino Uno y las tecnologías de identificación por radiofrecuencia (RFID) en el desarrollo de un sistema para la identificación y gestión de activos a través de un servidor operado con Raspberry Pi.

El objetivo de esta oportunidad de innovación tecnológica para el control de activos mediante el uso de tecnologías RFID, es brindar herramientas a los administradores y colaboradores de una empresa para maximizar el rendimiento de sus recursos impactando en la reducción de los costos por inventarios, mediante la implementación de un sistema que concentre una base de datos con la información crítica de cada activo (nombre, área, fecha de alta o baja, ubicación, etc.) y la muestre en una plataforma inteligente para facilitar un control más preciso y robusto sobre cualquier activo etiquetado.

En la sección dos se describen los trabajos más relevantes encontrados durante la revisión de literatura relacionada con la administración de activos mediante el uso de tecnologías RFID y Raspberry. En la sección tres se presenta el desarrollo de la plataforma para el control de activos a partir de *tags* (etiquetas) RFID de alta frecuencia (*High Frequency*) y la tarjeta lectora PN532 RFID *Reader*, incluyendo los procedimientos y los códigos principales en Arduino Uno para formatear, escribir y leer las *tags* colocadas en los activos. Además, también se describen los modos de comunicación del servidor Raspberry con Arduino y la base de datos. En la sección cuatro se mencionan los resultados esperados a través del desarrollo de una Interfaz gráfica para el usuario donde se muestran los datos relevantes de cada activo cuando es detectado a través de su *tag* correspondiente. Y por último, la sección cinco se dedica para la descripción de las conclusiones.

2. Trabajos relacionados

Durante la revisión de literatura se encontraron algunos trabajos relacionados con el tema de investigación.

En [1], Po Yang et al. resentan los trabajos realizados para una distribución óptima de tarjetas RFID en la localización de activos. Shaojie Tang et al [2] en sus trabajos de investigación proponen un protocolo para una multilectura de tarjetas RFID con la plataforma Arduino. Por su parte, Sourish Behera et al [3] proponen un nuevo modelo para un sistema de localización en tiempo real (RTLS: *Real Time Location System*) enfocado en un seguimiento de la ubicación exacta de los objetos a través de un sistema GPS (*Global Positioning System*) y DGPS (*Differential Global Positioning System*), usando tarjetas RFID activas colocadas en los objetos y otros dispositivos co-lectores para el monitoreo, rastreo y cálculo de la posición exacta del objeto.

Cabe destacar que actualmente se pueden emplear sistemas de control basados en RFID, sin embargo, el problema de su alto costo limita su adquisición para diversas empresas, además la complejidad en su arquitectura dificulta su uso por operadores poco experimentados en estas tecnologías.

3. Desarrollo

La estructura del sistema está conformada por dos etapas representadas en el diagrama a bloques de la figura 1. Donde cada bloque representa un dispositivo por el cual se envían los datos del *tag* RFID.

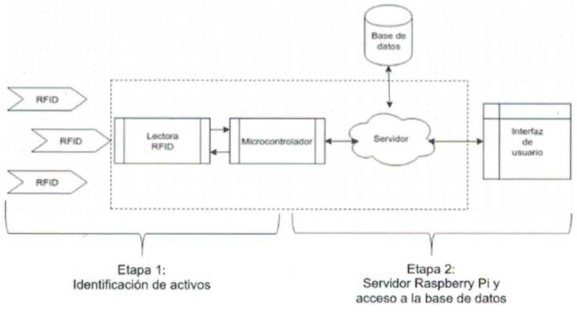

Figura 1. Diagrama a bloques del control de activos con RFID y Raspberry Pi.

3.1. Características y pruebas realizadas con la plataforma Arduino y la tarjeta PN532 RFID/NFC *Reader*.

El objetivo principal de la primera etapa consiste en las pruebas realizadas para formatear, escribir y leer los datos en las *tags* colocadas en los activos de alguna empresa. Esta etapa comprende la conexión y el desarrollo del código correspondiente en Arduino Uno para la configuración y puesta en marcha de la unidad lectora PN532 RFID *Reader* a través del protocolo de comunicación SPI. Es necesario conectar el buffer 74HC4550 entre la unidad lectora (5V) y el Arduino (3.3V) para el acoplamiento de niveles de voltaje. En la figura 2 se muestra le interconexión de estos elementos.

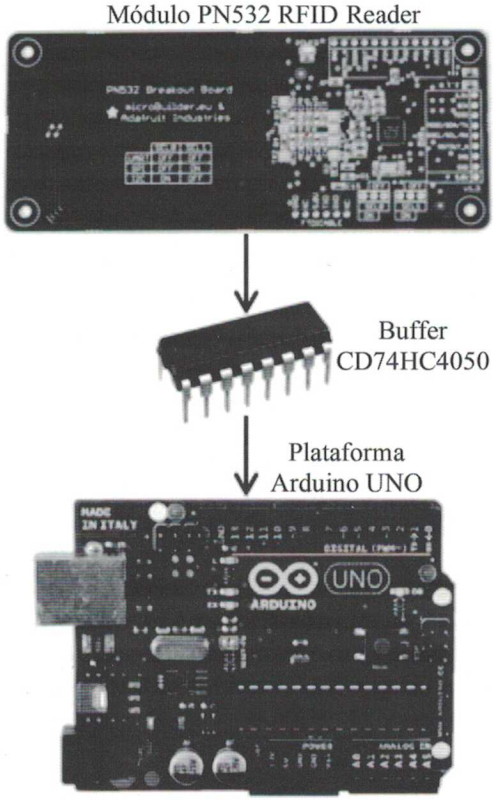

Figura 2. Interconexión de la unidad lectora RFID y Arduino Uno para la identificación de activos.

3.1.1. Tarjeta de identificación por radiofrecuencia

La identificación por radiofrecuencia (RFID) es una tecnología que permite detectar automáticamente un objeto único gracias a una onda emitida por la *tag* (etiqueta), incorporada en el mismo objeto, que transmite por radiofrecuencia sus datos de identificación.

Este dispositivo de identificación es del tipo *Contactless Smart Card IC* (*Integrated Circuit*), contiene una antena y una memoria EEPROM, y comúnmente se

les conoce como *transponder* (*Transmitter-Response*) o *tag* (etiqueta) y pueden adherirse al objeto. La figura 3 muestra un diagrama de la arquitectura RFID utilizada.

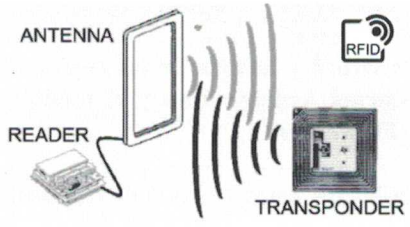

Figura 3. Tecnología para la identificación por radiofrecuencia (RFID) [4].

El modo de funcionamiento de los sistemas RFID es muy sencillo: la *tag* adherida al objeto genera una señal de radiofrecuencia que contiene los datos de identificación, esta señal puede ser captada por un lector RFID utilizado para leer la información y pasarla en formato digital a la aplicación específica. En la figura 4 se muestra la arquitectura de una tag RFID.

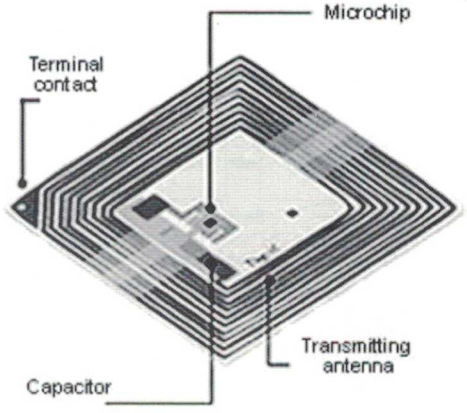

Figura 4. Arquitectura de la *tag* RFID pasiva [5].

La *tag* utilizada para identificar los activos por RFID es de la familia MIFARE *Classic* EV1, figura 5, con las siguientes características [6]:

- Bajo costo, 2,50 USD por tarjeta.
- Tipo pasiva (no requiere batería).
- La señal de radiofrecuencia emitida por la unidad lectora RFID induce una pequeña corriente eléctrica suficiente para operar el circuito integrado CMOS en la *tag*.
- La frecuencia de operación es de 13.56 MHz.
- Cumple con la norma ISO/IEC 14443A.
- Identificación única (UID) de 7 bytes.
- Memoria EEPROM de 1KB.
- Distancia típica de lectura-escritura: 10cm.
- Protocolo RFID de bajo nivel.
- La condición de encendido será solo dentro del rango de la lectora RFID, y solo puede ser escrita o leída dentro del rango de alcance.

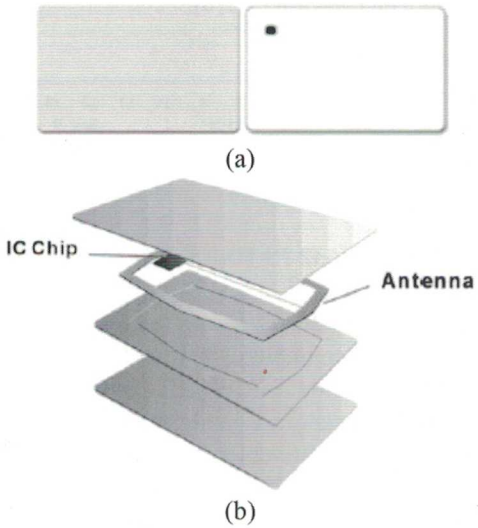

Figura 5. Tag MIFARE *Classic* EV1 [6].

Las condiciones de acceso a la memoria ROM de la *tag* son libremente programables, es una memoria EEPROM de 1KB organizada en 16 sectores de 4 bloques (un total de 64 bytes por sector), ver figura 6.

```
Seems to be a Mifare Classic card (4 byte UID)
-----------------------Sector 0-----------------------
Block 0  21 01 2E 0F 01 08 04 00 01 98 0A F9 B2 4F 6C 1D
Block 1  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 3  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
-----------------------Sector 1-----------------------
Block 4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 5  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 7  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
-----------------------Sector 2-----------------------
Block 8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 11 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
-----------------------Sector 3-----------------------
Block 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 15 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF
```

Figura 6. Organización de la memoria EEPROM de la tag MIFARE *Classic* Ev1 [7], [8].

La memoria EEPROM en la *tag* utilizada tiene características básicas de seguridad con accesos configurables tanto de lectura como de escritura con dos

diferentes llaves de autenticidad para condicionar el acceso a cada sector, constan de 6 bytes cada una y se guardan en el tercer bloque de cada sector (*trailer block*), ver figura 7. El bloque 0 del sector 0 es un sector de solo lectura y contiene la información de manufactura de cada *tag* [7], [8].

```
Sector Trailer Bytes
-----------------------------------------------------------------
0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
[       Key A       ]  [Access Bits]  [       Key B       ]
```

Figura 7. Formato de la llave de autenticidad [7], [8].

3.1.2. Módulo PN532 RFID/NFC *Reader*

El módulo PN532 RFID *Reader* es un sistema de transmisión para comunicación basado en el microcontrolador 80C51 de ATMEL. El transmisor utiliza un destacado concepto de modulación y demodulación integrado para diferentes métodos de comunicación sin contacto a 13.56MHz. La parte del transmisor interno permite el uso de una antena diseñada para comunicarse con los *transponder* MIFARE e ISO/IEC 14443A sin ningún circuito adicional. En la figura 8 se muestra el diagrama a bloques del módulo Adafruit PN532 RFID/NFC utilizado para la escritura y lectura de datos en las *tags* adheridas en los activos para su identificación.

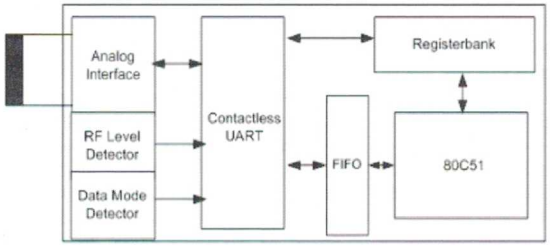

Figura 8. Diagrama a bloques del módulo PN532 RFID/NFC *Reader* [9].

Las principales características del módulo PN532 RFID/NFC *Reader* son [8]:

- Tecnología NFC: *Near Field Communications.*
- Núcleo basado en microcontrolador 80C51 con 40KB de memoria ROM y 1KB de RAM.
- Puede utilizar los protocolos de comunicación SPI, I2C y serial UART.
- Cuenta con un *framing* y detector de error.
- Detector de niveles de RF integrado.
- Detector de modos de información.
- Distancia promedio de comunicación: 10 cm, dependiendo del tamaño de la antena.
- Interrupción flexible (pin IRQ).
- El voltaje de alimentación es de 2.7V a 5.7V.

3.1.3. Escritura y lectura de las *tags* a través de la plataforma Arduino Uno y el módulo PN532 RFID/NFC *Reader*

Para escribir los datos en las *tags* (etiquetas) RFID para la identificación y validación de la información de los activos en una empresa, se utiliza la plataforma Arduino Uno. El código *middleware* desarrollado en esta plataforma tiene dos propósitos fundamentales (refiérase a la figura 9):

- Formatear las *tags* nuevas. Comunicarse con el módulo PN532 *Reader* a través del protocolo SPI para la escritura del formato correspondiente y de los datos para identificación del activo.
- Lectura de las *tags*. Enviar los datos leídos de la *tag* en el activo en turno al servidor (Raspberry Pi) por comunicación USB (simulando un puerto serial).

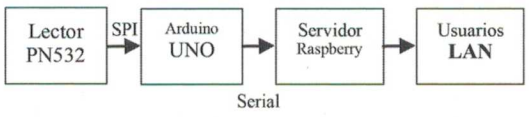

Figura 9. Comunicación Arduino con la tarjeta lectora PN532 y el servidor Raspberry.

Arduino es un sistema multiplataforma (Windows, Macintosh OS X y GNU/Linux) de bajo costo para desarrollo de prototipos electrónicos de código abierto (*Open Source*), basado en los microcontroladores de ATMEL ATMEGA8 y ATMEGA168, se programa en un lenguaje similar al C++ en un entorno simple y flexible, puede comunicarse fácilmente con una computadora a través del protocolo del puerto serial y cuenta con una enorme comunidad de usuarios.

El protocolo utilizado para la comunicación entre el Arduino Uno y el circuito PN532 *Reader* es el SPI (*Serial Peripheral Interface*) ya que es un protocolo de datos síncrono y puede ser configurado para comunicarse con otro microcontrolador o con uno más dispositivos periférico de manera rápida en cortas distancias. El código middleware desarrollado en la plataforma Arduino incluye las siguientes librerías que facilitan la comunicación SPI:

- SPI.h, define las funciones de la interfaz SPI, librería muy utilizada en dispositivos AVR32.
- Adafruit_PN532.h, para la configuración y control del módulo PN532 RFID/NFC Reader.

3.1.4 Formato de una *tag* (etiqueta) RFID nueva

Cuando se va a utilizar por primera vez una *tag* MIFARE *Classic* EV1 es necesario formatearla para agregar una llave de autenticidad (contraseña), y el programa guardado en Arduino Uno pueda realizar la escritura de los datos de identificación del activo. El código utilizado para dar formato a las *tags* nuevas se puede bajar libremente de Internet [10]. Es importante colocar la *tag* sobre la tarjeta PN532 RFID/NFC *Reader* para realizar las actividades de formato o escritura de datos y no moverlas durante todo el tiempo que dure este proceso (ver figura 10).

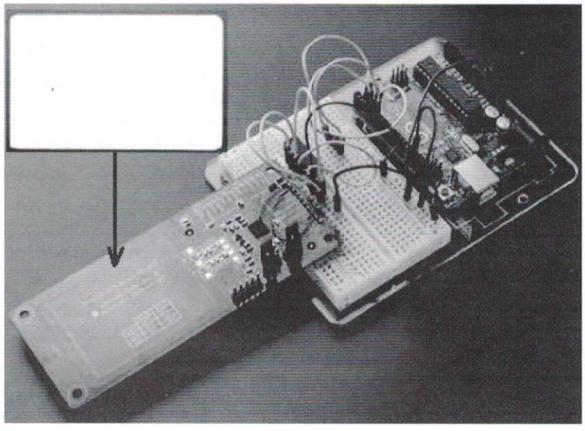

Figura 10. La tarjeta tag debe colocarse sobre la tarjeta PN532 *Reader* para formatear o escribir datos.

Las partes claves del programa en Arduino Uno para el formato de una *tag* son (figura 11):

a) Se define una llave para la tarjeta como un arreglo de bytes.
b) Después de comprobar que se trata de una tarjeta que se puede formatear, se llama a la función mifareclassic_FormatNDEF() para formatearla, si la función tiene éxito regresa un valor verdadero..
c)

```
uint8_t keya[6] = { 0xFF, 0xFF, 0xFF, 0xFF,
   0xFF, 0xFF };
```

(a)

```
success = nfc.mifareclassic_FormatNDEF();
if (!success)
{ Serial.println("Unable to format the card
for NDEF");
 return;}
Serial.println("Card has been formatted for
NDEF data using MAD1");
```

(b)

Figura 11. Partes claves del programa en Arduino Uno para el formato de una *tag*.

3.1.5. Escritura de datos en la *tag* (etiqueta) RFID

Para dar de alta los nuevos activos es necesario escribir sus datos de identificación e información relevante en la memoria EEPROM de la *tag* RFID. Esto se debe realizar inmediatamente después de finalizar el proceso de formato por cuestiones de seguridad. Dentro de cada *tag* se escriben los datos correspondientes a los campos que contiene la base de datos que concentra la información de todos los componentes del sistema, se asigna un campo por sector en la distribución de memoria EEPROM.

Las partes claves del programa en Arduino Uno para la escritura de datos en una *tag* son (figura 12):

a) Crear un arreglo de bytes con el mensaje.
b) Autenticar un bloque usando la llave con la que se formateó.
c) Escribir la información.

```
uint8_t success; // Flag para revisar errores
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  //
Buffer para guardar el UID
uint8_t uidLength;
// ...
uint8_t keya[6] = { 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF };
const char* mensaje = "Hola mundo!";
// ...
success =
nfc.mifareclassic_AuthenticateBlock(uid,
uidLength, 4, 0, keya);
if (!success)
{ Serial.println("Authentication failed.");
 return; }
```

(a)

```
success =
nfc.mifareclassic_AuthenticateBlock(uid,
uidLength, 4, 0, keya);
if (!success)
{ Serial.println("Authentication failed.");
 return; }
```

(b)

```
success = nfc.mifareclassic_WriteDataBlock(4,
mensaje);
if (success)
{ Serial.println("NDEF URI Record written to
sector 1");}
```

(c)

Figura 12. Partes claves del programa en Arduino uno para la escritura de datos en una *tag*.

3.1.6. Lectura de los datos escritos en la *tag* (etiqueta) RFID

Después de finalizado el proceso de formato y escritura de una *tag*, el programa retorna a su función estándar que es la lectura de los datos escritos en la memoria EEPROM de la *tag*.

Cada vez que se acerca un activo (10 cm) a la tarjeta lectora PN532, se detecta la *tag* correspondiente (adherida al activo) y el programa en el Arduino Uno inicia el proceso de autenticidad con la misma llave que se formateó y escribió, pero esta vez la información se lee y se envía por el puerto serial, como se muestra en la figura 13 que presenta la parte clave del programa en Arduino Uno para este proceso dentro de una función *loop()*.

```
success = nfc.mifareclassic_ReadDataBlock(4,
data);
if (success)
{
  for (int i = 0; i < 16; i++) {
    lectura[i] += (char)data[i];}
  Serial.println(lectura); // Envío a la RPi
  delay(1000);
}
```

Figura 13. Parte clave del programa en Arduino Uno para la lectura de datos en una tag.

3.2. Características y pruebas realizadas con la plataforma de red para el acceso a la base de datos basado en tecnología Raspberry.

El sistema se diseñó para poder correr en diferentes arquitecturas, pues dado que el servidor corre exclusivamente software de código abierto, esto se pudo hacer de una manera sencilla. Las dos arquitecturas utilizadas para el desarrollo del código del servidor y la base de datos fueron:

- La arquitectura x86. Es la misma que las computadoras de escritorio, y se utilizó como sistema operativo la versión 14.04 LTS de Ubuntu Linux, y la plataforma Node.JS versión 0.12.7.
- Para la arquitectura ARM se usó una Raspberry Pi modelo B+, ver figura 14, con el sistema operativo Raspbian Wheezy y la misma versión de Node JS.

En la sección 3.1.3 se presentó el diagrama a bloques (figura 9) de la interconexión de la unidad lectora y el servidor Raspberry con Arduino Uno. Cuando se lleva a cabo el proceso de lectura para la autenticidad e identificación de un activo a través de su *tag*, el programa en el Arduino Uno envía el identificador correspondiente al servidor Raspberry a través del protocolo de comunicación serial. El programa en la Raspberry permite sincronizar el identificador de cada activo con la base de datos almacenada en la misma Raspberry. Para crear la base de datos se utilizó SQLite 3, por ser el más sencillo y ligero de todos, además de tener una alta compatibilidad con el software utilizado en la programación web.

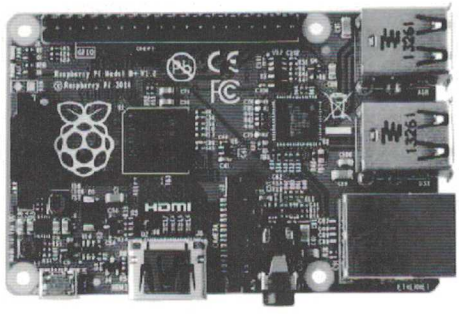

Figura 14. Raspberry Pi modelo B+

Las funciones de la Raspberry como servidor son:

- Monitoreo constante del Arduino (espera que la tarjeta lectora PN532 detecte un activo a través de su *tag*).
- Revisar si el ID de la *tag* concuerda con un ID dentro de la base de datos (si concuerda, decir si

va de entrada o de salida).
- Alojar la página de estadísticas y configuración.

Al mismo tiempo que se escribe en una *tag*, un programa dentro del servidor donde se encuentra conectado el Arduino está leyendo los datos escritos y los almacena dentro de una base de datos con los campos asignados, que pueden ser por ejemplo el número de identificación, nombre del activo, departamento, uso, etc. Para poder realizarlo fue necesario desarrollar una aplicación que siempre estuviera ejecutándose en el sistema (comúnmente denominadas "demonios"), escrita en Javascript con la ayuda del framework Express JS. Este tipo de software se caracteriza porque trabaja siempre en segundo plano y toma decisiones sin necesidad que el usuario interactúe, esto hace que el sistema sea casi autónomo, basta que la tarjeta lectora PN532 identifique una *tag* para que el software tome las decisiones apropiadas, como la de identificar de qué activo se trata, y si este mismo activo va de salida o de entrada.

Además de encargarse de insertar nuevas entradas a la base de datos cada que el usuario escribe en un *tag*, la aplicación "demonio" se encarga también de actualizar los datos cada vez que se lee un valor, como los son la frecuencia de uso del activo, su estado (afuera o dentro del almacén) y la fecha en que se actualizaron estos mismos datos.

Las aplicaciones Web utilizadas se desarrollaron con la plataforma Node.JS, es un entorno de ejecución basado en el mismo motor que usa el navegador de Google, usa el lenguaje de programación Javascript, pero está hecho para aplicaciones en servidores.

Las partes claves en la programación para que el Arduino Uno se comunicara con la Raspberry son:

a) Al conectar un Arduino a un sistema con Linux se crea un dispositivo serial localizado en: /dev/ttyACM0.
b) La aplicación del servidor trata de abrirlo usando la librería SerialPort para NodeJS.
c) Ya que está conectado se crea un listener dentro de la función connectArd() que escucha un evento que se dispara cada vez que el Arduino escribe en el puerto serial.
d) La variable asset_id se envía a la función actualizarActivo() para comunicarse con la base de datos.

Las partes claves en el desarrollo del código para que la Raspberry se comunique con la base de datos son:

a) La base de datos generada es un archivo db.rfid.sqlite que se encuentra en el directorio del mismo proyecto, esto debido a que el motor de la base de datos es SQLite y sólo ocupa de un archivo para poder trabajar.
b) Para hacer el modelo de la base de datos se usó un ORM (Object-Relational Mapping) llamado Sequelize, de acuerdo con el código desarrollado en Node.JS.
c) Para interactuar con la base de datos se desarrolló la aplicación para que tuviera una API tipo RESTful, así cuando al servidor le llegue un método POST con el identificador del activo, éste se actualiza. Otra ventaja es que si cuenta con una API, lo hace compatible con otros sistemas web si se requiere integrar en un futuro. En la figura 15 se muestra el código de una de las rutas del servidor.
d) El POST se realiza desde la misma aplicación con la función de actualizarActivo(id).

4. Resultados

Se puso a disposición del usuario final una interfaz en forma de una página web, a la cual se puede acceder desde cualquier punto de la red de área local a través de la dirección IP asignada al servidor desde un navegador. Dentro de la interfaz, el operador puede ver las estadísticas del sistema, además de poder escribir en nuevos *tags* (y a su vez en la base de datos); o actualizar y borrar entradas ya existentes.

```
router.post('/:id', function(req, res) {
  models.Assets.findAll({
    where: { id: req.params.id }
  }).then(function(data) {
    status = data[0]['dataValues']['status'];
    var freq =
data[0]['dataValues']['frequency'] + 1;
    if (status == 'in')
       { models.Assets.update(
        { frequency: freq,
          status: 'out' },
        { where: { id: req.params.id }}
      ).success(function () { res.send('In ->
Out'); })
      .error(function() {
res.send('Error')}); }
    else { models.Assets.update(
        { status: 'in' },
        { where: { id: req.params.id } }
      ).success(function () { res.send('Out -
> In'); })
      .error(function() {
res.send('Error')}); }
  });
});
```

Figura 15. Código para una de las rutas del servidor.

La aplicación web se desarrolló en el entorno de diseño llamado Bootstrap y se integró a la aplicación "demonio". En esta se pueden visualizar el contenido de la base de datos y su estado, este entorno grafico se aprecia en la figura 16.

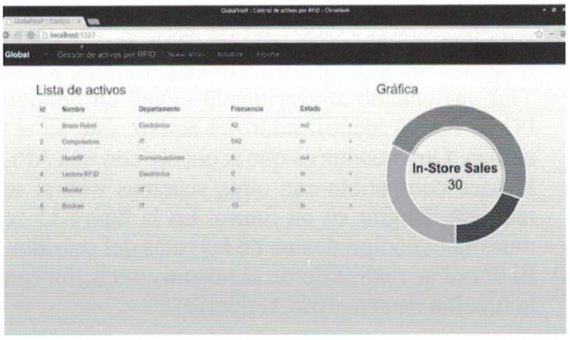

Figura 16. Interfaz de usuario para la base de datos

La gestión de la aplicación consiste en dar de alta las *tags* en el sistema (ver figura 17). Cuando el usuario hace clic sobre el enlace "Nuevo activo", aparece una ventana donde puede ingresar el nombre y el departamento de un nuevo activo. Además, esta ventana le avisa al operador que tiene que colocar una *tag* nueva sobre la lectora para que puedan sincronizarse.

Una vez registrado el nuevo activo, y sincronizado con la *tag*, aparecerá automáticamente una nueva entrada en la tabla. El número de identificación se asignará automáticamente, se asumirá que el activo se encuentra dentro del almacén y la frecuencia de uso comenzará en 0. La única forma de editar una entrada es eliminándola (con la cruz en la última columna) y volver a grabar otro *tag*, ya que ambos están sincronizados.

Figura 17. Interfaz de usuario para la gestión de activos.

Además, la interfaz de usuario muestra en una gráfica los cinco activos más usados (ver figura 18).

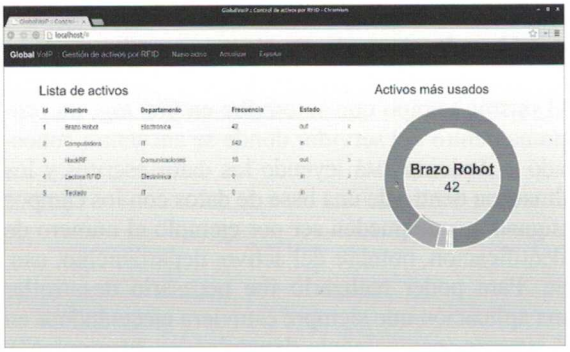

Figura 18.Interfaz gráfica.

5. Conclusiones

Es factible integrar las tecnologías RFID y Raspberry controladas con una plataforma Arduino para desarrollar un sistema que permita un control eficiente de los activos en una empresa, impactando en la reducción de los gastos por inventarios. Su costo aproximado es de 100 dólares, lo que lo hace competitivo frente a otros productos similares en la industria de las telecomunicaciones. Además, el protocolo de comunicación es muy transparente para el operador de los activos en una empresa.

6. Referencias

[1] Po Yang; Wen Yan Wu; Mansour Moniri; Claude C. Chibelushi "Efficient Object Localization Sparsely Distributed Passive RFID Tgs", IEEE Transactions on Industrial Electronics, 2013, Volume: 60.

[2] Shaojie Tang; Jing Yuan; Xiang-Yang Li; Guihai Chen; Yunhao Liu; Jizhong Zhao, "RASPberry: A stable reader activation schedulling protocol in multi-reader RFID systems", Network Protocols, 2009, ICNP 2009, IEEE conference publications.

[3] Sourish Behera; Chandan Maity, "Active RFID tag in Real Time Location System", Systems, signals and Devices, 2008. IEEE SSD 2008 International Multi-conference on.

[4] University of Oklahoma, "Radio Frequency IDentification (RFID) for Birds", http://animalmigration.org/RFID/index.htm, (visitada el 15 de junio de 20015).

[5] Rob Shakir, "Musings on life and network architecture", http://rob.sh/post/8, (visitada el 15 de junio de 20015).

[6] Smart Card World, MIFARE Classic MF1 ICS50 White PVC Cards, http://www.smartcardworld.com/mifare1k.htm?gclid=COes_K2qk8sCFQUMaQodxxML0w, (visitada el 10 de enero de 20015).

[7] Adafruit PN532 NFC/RFID controller shield for Arduino, https://www.adafruit.com/products/789, (visitada el 15 de enero de 2015).

[8] Adafruit PN532 RFID/NFC Breakout and Shield, https://learn.adafruit.com/downloads/pdf/adafruit-pn532-rfid-nfc.pdf (visitada el 15 de enero de 2015).

[9] PN532/C1 NFC Controller short form data sheet, https://www.adafruit.com/datasheets/pn532ds.pdf, (visitada el 15 de enero de **2015).**

[10] Adafruit-PN532 mifareclassic_formatndef.pde, https://github.com/adafruit/Adafruit-PN532/blob/master/examples/mifareclassic_formatndef/mifareclassic_formatndef.pde, (visitada el 18 de enero de 2015).