Implementación de Sistema de Seguridad y Monitoreo de Acceso con dispositivos RFID

Jorge Gabriel Cruz Guzñay [(1)], Washington A. Velásquez Vargas [(2)]

Facultad de Ingeniería en Electricidad y Computación, Escuela Superior Politécnica del Litoral, Km. 30.5 Vía Perimetral, Guayaquil, Ecuador
jgcruz@espol.edu.ec[(1)], wavelasq@espol.edu.ec[(2)]

Resumen. RFID es una tecnología que usa ondas electromagnéticas para almacenamiento y recuperación de datos, el cual posee una gran gama de usos. El proyecto implementa un sistema de seguridad usando dispositivos de fabricante Arduino, los cuales por ser elementos open source permiten hacer uso de un lector de tarjetas RFID con su propio dispositivo datalogger, de esta manera se dará acceso a los usuarios que posean tarjetas autorizadas. Posteriormente cuando el administrador del sistema de seguridad desee monitorear los accesos de los usuarios que ingresaron al sistema, lo podrá realizar mediante una aplicación programada en Android que se comunicara con el datalogger vía mensajes bluetooth. Una vez conectado al datalogger, el administrador podrá ver los últimos accesos o hacer una limpieza del banco de registros.

Abstract. RFID is a technology that have a large capacity for development and many ways to use. With this project it will implement a system of security using Arduino devices, these are open source elements and this will let to use a RFID card reader with its own data logger device, in this way the users with authorized card will access to the system and access will be recorded. Later when the administrator of the system wants to check access of users to system, it will do with Android programming application that will communicate with data logger via bluetooth messages. Once the application will connects to data logger, the administrator could check recent access or clean register.

Palabras Clave: Tags RFID, Open Source, Arduino, Android, seguridad, cuentas de usuario.

1 Introducción

A principios de 1920 ya se hablaba de los primeros indicios de tecnología RFID pero sus usos eran dudosos, ya en las últimas décadas se ha vuelto más popular gracias a la reducción de costos. Básicamente esta tecnología se aprovecha de los beneficios de los dispositivos pasivos que usan las ondas electromagnéticas para transmitir información y energía. De esta manera usando elementos llamados TAGS, se puede almacenar y recuperar información de manera segura. Por otro lado, el desarrollo basado en software y hardware open source se está volviendo más rentable y eficiente cada vez. Por ello el uso de software Android y hardware prototyping como Arduino, se vuelve algo más útil al momento de querer implementar nuevos sistema y tener un pleno control de los mismos.

1.1 Tecnología RFID

La tecnología RFID [1] básicamente usa las ondas electromagnéticas para transmitir información entre un módulo lector y un dispositivo con integrado pasivo. El integrado pasivo más la antena de recepción o también llamadas TAGS usan las portadoras de las ondas electromagnéticas enviadas para cargarse y poder trabajar. Las señales enviadas por el modulo lector poseen modulación de PAM. Las tags RFID últimamente se han vuelto bastante populares por su fácil implementación y bajo costo en manipulación y mantenimiento. La capacidad de almacenamiento varía de acuerdo al tipo de integrado pasivo que se use. Por ejemplo las TAGS Mifare S70 usadas en este proyecto soportan aproximadamente 4kB de almacenamiento de datos hexadecimales organizados por sectores. Hay que añadir que estos dispositivos tienen capacidad para proteger con dos contraseñas de 6 bytes cada sector, y además la posibilidad de poder cifrar los datos hexadecimales contenidos en los sectores.

1.2 Software Android

Actualmente el sistema Android es el más usado en cuanto a sistemas operativos para dispositivos inteligente. Como se observa en la figura 1, el sistema Android es basado en software open-source comenzando con un kernel basado en Linux y una máquina virtual reducida basada en Java llamada Dalvik que se usa para ejecutar las diferentes aplicaciones y servicios que posee [2].

Figura 1: Framework Android

1.3 Hardware Arduino

Arduino [3] es una herramienta creada para censar y tener un mayor control con el mundo real mucho más de lo que haría un simple ordenador. Arduino es una plataforma física de computación de código abierto basado en una placa con un simple microcontrolador y un ambiente de desarrollo para escribir código en la placa.

Las placas de Arduino son capaces de manejar un gran cantidad de elementos como motores, simular servidores web, dataloggers, conectares con dispositivos a puertos seriales, entre otros. Además maneja entradas tanto digitales como analógicas y la capacidad de almacenar una cantidad de data considerable gracias a la EEPROM del microcontrolador que posee.

Se escogió usar elementos Arduino por su fácil manipulación, costo relativamente bajo y por su capacidad de soportar variados protocolos de comunicación.

2 Descripción del Proyecto

El proyecto se basa en implementar un sistema de seguridad completo que incluye la implementación física de los dispositivos de control más un software o aplicación de administración. La implementación física se lo realizo usando elementos open source como lo son las placas de fabricante Arduino. A esto se añadió los módulos de comunicación RFID y bluetooth de modelos MFRC522 [4] y HC-05[5] respectivamente. La placa Arduino modelo UNO, servirá como dispositivo de procesamiento, almacenamiento y comunicación. Se lo programo de tal manera que sea capaz de leer constantemente TAGS RFID y almacene la información y al mismo tiempo esté atento a comandos recibidos por un puerto serial simulado conectado a un módulo bluetooth.

Con la aplicación en Android que se diseñó, se es capaz de leer los datos almacenados en la EEPROM de la tarjeta Arduino y a su vez eliminar o limpiar ese registro. Todo eso se realiza mediante comandos enviados por una conexión bluetooth. La aplicación en Android se la denomino SecureSystem. Hay que tomar en cuenta que para instalar y hacer uso de la aplicación se deben dar los permisos necesarios como es el acceso al dispositivo bluetooth integrado en el dispositivo inteligente.

3 Modelo de Negocio

En la figura 1 se muestra el funcionamiento del sistema completo y el cual consta de 5 partes:

1) **Tags RFID:** estos dispositivos fueron previamente programados para contener información relevante del usuario de la tarjeta. Se añadió un código propio de cada usuario y un identificador de acceso al área respectiva.

2) **Módulo MFRC522:** este módulo es específicamente para lectura de varios tipos de tags RFID como los modelos S70 y S50. Es un módulo programable para transmitir es diferentes frecuencias y aparte cuanta con una memoria EEPROM de poca capacidad destinada para actividades específicas.
3) **Placa Arduino Uno y Módulo Bluetooth HC-05:** la placa Arduino Uno contiene un microprocesador ATMega328 el cual mediante un IDE de programación se puede cargar programas ligeros escritos en C o C++. Con el IDE se programara la placa para primero realizar la comunicación con las lecturas de módulo RFID y el módulo bluetooth. Segundo se programara que procese la información obtenida de los tags RFID y almacene la información pertinente para que luego cuando se reciban comandos de la comunicación bluetooth, este realice las operaciones pertinentes como enviar o borrar información almacenada.
4) **Dispositivo Inteligente con Bluetooth:** es necesario poseer un dispositivo inteligente con capacidad de comunicación bluetooth. Este dispositivo deberá tener instalado sistema Android con al menos el API 8 o también conocido como Froyo. Para el desarrollo del proyecto se utilizó como dispositivo inteligente al celular HTC Wildfire.
5) **Aplicación Android - SecureSystem:** una vez que se poseen todas las características mencionadas en el literal 4, se instala la aplicación SecureSystem desarrollada previamente y con el cual mediante mensajes vía bluetooth se controlara los dispositivos anteriormente mencionados.

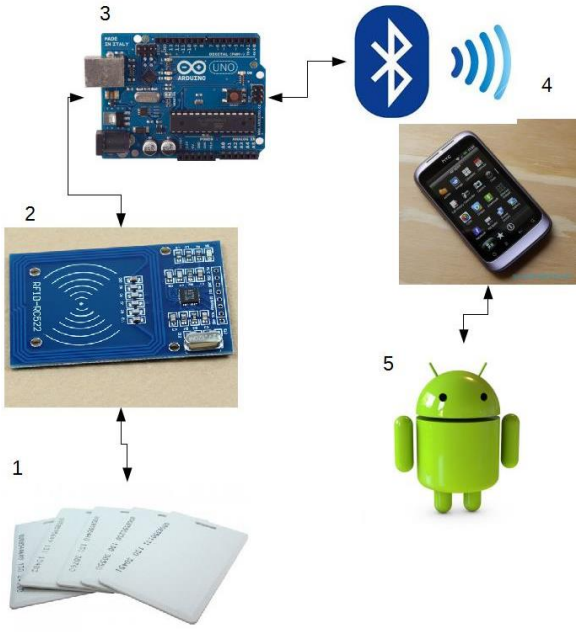

Figura 2: Esquema de Funcionamiento

4 Descripción de las Activitys de la aplicación SecureSystem

La aplicación consta básicamente de 3 Activities: menú principal, menú bluetooth y menú para administración del sistema de seguridad.

4.1 Menú Principal

En este menú podemos acceder a los dos submenús que son: Menú Bluetooth y Menú de Datos.

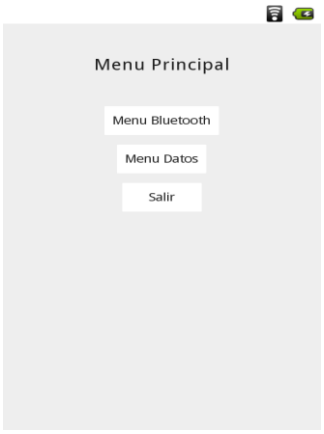

Figura 3: Menú Principal

4.2 Menú Bluetooth

En este menú se pueden escoger las opciones referentes a la conectividad bluetooth: nos permite encender o apagar la comunicación bluetooth, escanear y aparear nuevos dispositivos y conectarnos con dispositivos ya apareados.

Figura 4: Menú Bluetooth

4.3 Menú Datos

Se puede entrar a este menú una vez se haya realizado la conectividad vía bluetooth con el dispositivo propio del sistema de seguridad. En esta activity se puede enviar los comandos básicos para revisar los accesos de las tags RFID o a su vez hacer un borrado de los datos almacenados en la EEPROM de la placa Arduino.

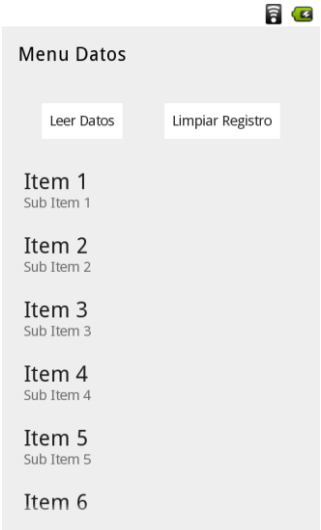

Figura 5: Menú Datos

5 Desarrollo del Sistema

5.1 Desarrollo del Código en plataforma Arduino

El sistema como se mencionó consta de un lector de tarjetas RFID y un módulo bluetooth.

El lector de tarjetas RFID tiene un interfaz de comunicación SPI lo cual nos da una ventaja para leer los datos de forma serial y realizar diversas configuraciones en la lectura. Para optimizar las operaciones de comunicación con el lector, se utilizó una librería externa diseñada para trabajar específicamente con el lector MFRC522 [6]. La librería fue implementada en el código fuente usado en la placa Arduino logrando con esto manejar la lectura de tarjetas.

```
#include <EEPROM.h>
#include <MFRC522.h>
#include <SPI.h>

MFRC522 mfrc522(10, 9);   // Create MFRC522 instance.
MFRC522::MIFARE_Key key;
```

Figura 6: Librería MFRC522

Para el proyecto se desarrolló dos programas, el primer programa "*writeToBlock.ino*" se basó en acceder a cierto sector de memoria del tag RFID para poder introducir un código de identificación del usuario de dicha tarjeta. Como se observa en la figura 7, se muestra un arreglo con el código de identificación del usuario en formato hexadecimal, posterior a esa declaración como se observa en la figura 8, se procede a escribir los datos en el tag manejando los posibles errores durante la escritura. Básicamente la utilidad de este programa y configurar los tags RFID que serán entregados a los usuarios del sistema.

```
byte dataBlock[]    = {
    0x30, 0x39, 0x39, 0x39, //
    0x39, 0x39, 0x39, 0x35, //
    0x35, 0x35, 0x20, 0x45, //
    0x4E, 0x55, 0x20, 0x20  //
};
```

Figura 7: Código Identificación de Usuario de 16 bytes

```
// Write data to the block
Serial.print("Writing data into block "); Serial.print(blockAddr);
Serial.println(" ...");
dump_byte_array2(dataBlock, 16); Serial.println();
status = mfrc522.MIFARE_Write(blockAddr, dataBlock, 16);
if (status != MFRC522::STATUS_OK) {
    Serial.print("MIFARE_Write() failed");
}
else
  Serial.println("Write OK");
```

Figura 8: Escritura del Dato en el Tag RFID

El segundo programa *"lectorTarjetaBluetooth.ino"* realiza las acciones de control de los tags RFID y almacenamiento en memoria EEPROM (datalogger). Como se observa en la función *loop* de la figura 9, se describen las acciones más relevantes que son:

- leerDatosBluetooth(): se encarga de reconocer los comandos que serán enviados a través del módulo bluetooth y tomar las acciones correspondientes.
- MRFC522.PICC_IsNewCardPresent(): verifica que no se lea más de una vez la misma tarjeta mientras siga en el rango del lector. Eso quiere que la función evita que se dupliquen los datos almacenados cuando el microcontrolador vuelva a ejecutar la función *loop*().
- MFRC522.PICC_ReadCardSerial(): verifica que se pueda leer el tag RFID.
- leerTarjeta(): esta función realiza en si las operaciones de autorización al sistema y registro de los accesos exitosos.

```
void loop() {
  if(bluetooth.available())
    leerDatosBluetooth();
  else if ( ! mfrc522.PICC_IsNewCardPresent())
      return;
  else if ( ! mfrc522.PICC_ReadCardSerial())
       return;
  else
      leerTarjeta();
}
```

Figura 9: Lectura de entrada Bluetooth y tag RFID

En la figura 10 y 11 se pueden observar la implementación del código de lectura de los tags RFID y almacenamiento en memoria EEPROM respectivamente. Siendo en ambos casos la variable buffer que contenga los datos obtenidos del tag RFID.

```
// Read data from the block
Serial.print("Reading data from block "); Serial.print(blockAddr);
Serial.println(" ...");
status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
if (status != MFRC522::STATUS_OK) {
    Serial.print("MIFARE_Read() failed");
}
```

Figura 10: Lectura del Tag RFID

```
guardarEEPROM(buffer);
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCryptol();
```

Figura 11: Almacenamiento de Datos y Desvinculación del TAG

5.2 Desarrollo del código en plataforma Android

En [6] se tiene un ejemplo del código fuente que puede usarse como esqueleto para desarrollar una aplicación en Android Studio que permita comunicación Bluetooth. Desgraciadamente como se sabe Android genera nuevos APIs con librerías actualizadas, estas no necesariamente van a ser soportadas por los APIs más antiguos como Android 2.2 o también conocido Froyo. En la figura 12 se ve el código utilizado para crear un socket para comunicación Bluetooth, el cual puede ser usado en cualquier versión de Android y garantizar la estabilidad de la comunicación.

```
try {
    this.device = this.bluetoothAdapter.getRemoteDevice(this.address);
    //this.blueSocket=this.device.createRfcommSocketToServiceRecord(BTMODULEUUID);

    Method m;
    m = this.device.getClass().getMethod("createRfcommSocket", new Class[]{int.class});
    this.blueSocket = (BluetoothSocket)m.invoke(this.device, Integer.valueOf(1));
}
```

Figura 12: Socket de comunicación bluetooth Android <=2.2

Otro punto importante es el manejo de los recursos en el Sistema Android, ya que leer un dato continuamente del socket y mantener activa la Activity consumira el mayor porcentaje de procesamiento del CPU. Por eso se opto por la utilizacion de hilos con la clase Handler y manejar todas las entradas del socket con este. En la figura 13 y 14 se pueden observar la implementacion del hilo y la forma en como obtiene los datos el hilo, respectivamente.

```
this.bluetoothIn=new HandlerPrueba();
this.mConnectedThread = new ConnectedThread(this.blueSocket);
this.mConnectedThread.start();
```

Figura 13: Hilo para lectura de datos bluetooth

```
private class HandlerPrueba extends Handler
{
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {                                     //if message is what we want
            String readMessage = (String) msg.obj;                          // msg.arg1 = bytes from connect thread
            recDataString.append(readMessage);                                  //keep appending to string until ~
            int endOfLineIndex = recDataString.indexOf("+");                    // determine the end-of-line
            if (endOfLineIndex > 0) {                                           // make sure there data before ~
                String dataInPrint = recDataString.substring(0, endOfLineIndex);    // extract string
                compareDataBase(dataInPrint);
                recDataString.delete(0, recDataString.length());                    //clear all string data
            }
        }
    }
}
```

Figura 14: Datos de la comunicación Bluetooth

Finalmente, una aplicación diferente de la interfaz OnItemClickListener es darle dos acciones diferentes cuando se presiona un botón A o un botón B asociados a una misma ListView. Esto se logró almacenando que botón se presionó en la variable *botonPresionado* y con este dato al momento de llamarse al listener, este sabrá que acción tomar.

```
private int botonPresionado = 0;
this.listaDispositivos = (ListView) findViewById(R.id.ListView2);
this.listaDispositivos.setAdapter(this.arrayAdapter);
this.listaDispositivos.setOnItemClickListener(this.listenerClick);
```

Figura 15: Dos acciones en OnItemClickListener

7 Conclusiones

Con el desarrollo de este proyecto se demostró uno de los múltiples usos que se le puede dar a los dispositivos tags RFID. Además se puedo observar la gran capacidad de almacenamiento considerando su estructura física y las seguridades en cuanto a contraseñas y cifrados que se pueden ayudar añadir a la información.

Otro de los puntos importantes que se puede mencionar es la libertad de desarrollo que nos dan tanto las placas Arduino como el sistema Android. Las placas Arduino con sus librerías basadas en C, C++ y Assembly nos permiten añadir libremente diversas funcionalidades en cuanto a shields que deseen añadirse, procesamiento de la información o funcionalidades diversas según la necesidad.

El sistema Android nos provee la libertad de hacer uso de todo el hardware de los dispositivos inteligentes y hacer aplicaciones bastantes amigables con el usuario, a su vez funcionales con respecto a lo que se desee desarrollar: procesamiento, interfaces gráficas, bases de datos, conectividad con el mundo exterior.

8 Anexo

8.1 Implementación física

En las figuras 16, 17 y 18 se puede observar el hardware implementado y listo para usarse. La conexión del módulo Bluetooth se realizó utilizando una interfaz serial definida por software que nos provee la librería SoftwareSerial del mismo IDE Arduino [8]. Por otro lado, la conexión del lector RFID se realizó mediante un puerto SPI incluido en la placa Arduino UNO [9] y que se activa haciendo uso de la librería SPI del IDE Arduino.

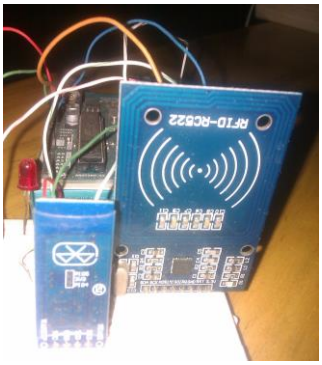

Figura 16: Modulo Bluetooth (HC-05) y Lector RFID (RFID-RC522)

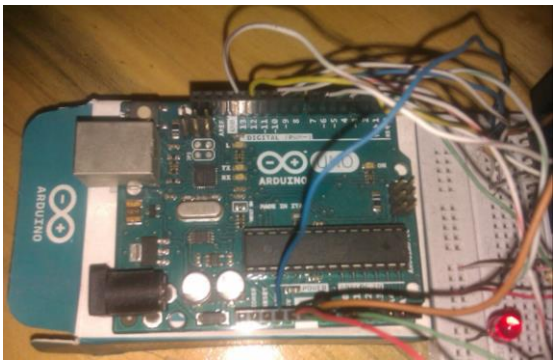

Figura 17: Placa Arduino UNO

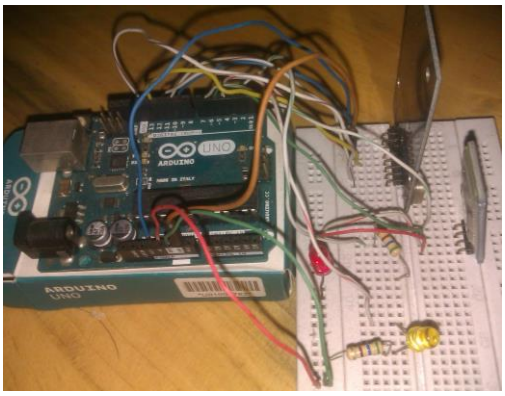

Figura 18: Hardware Implementado

Referencias

1. Wikipedia, “RFID”, Junio 2015 [Online]. Disponible: https://es.wikipedia.org/wiki/RFID
2. Wikipedia, “Android (operating System)”, Junio 2015 [Online]. Disponible: https://en.wikipedia.org/wiki/Android_%28operating_system%29
3. Arduino, “What is Arduino?”, Junio 2015 [Online]. Disponible: http://www.arduino.cc/en/Guide/Introduction
4. NXP Semiconductors N.V., “MFRC522”, Junio 2015 [Online]. Disponible: http://www.nxp.com/documents/data_sheet/MFRC522.pdf
5. Guangzhou HC Information Technology Co, “HC Serial Bluetooth Products – User Instructional Manual”, Junio 2015 [Online]. Disponible: http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf

6. Miki Balboa, GitHub, “Arduino RFID Library for MFRC522”, Agosto 2015 [Online]. Disponible: https://github.com/miguelbalboa/rfid
7. Wingoodharry, WordPress, “Android Send/Receive data with Arduino using Bluetooth”, Agosto 2015 [Online]. Disponible: https://wingoodharry.wordpress.com/2014/04/15/android-sendreceive-data-with-arduino-using-bluetooth-part-2/
8. Arduino, “SoftwareSerial Library”, Agosto 2015 [Online]. Disponible: https://www.arduino.cc/en/Reference/SoftwareSerial/
9. Grant Gibson, “How to get started with the Mifare MF522-AN and Arduino”, Agosto 2015 [Online]. Disponible: http://www.grantgibson.co.uk/2012/04/how-to-get-started-with-the-mifare-mf522-an-and-arduino/