A
Project Report
On

RFID – BASED SECURITY SYSTEM

By

- **AJINKYA BEDEKAR (U101116FCS183) (S1)**
- **BIREN SHARMA (U101116FCS246) (S4)**
- **SHANTANU BAHUGUNA (U101116FCS248) (S5)**
- **YOGESH SHARMA (U101116FCS247) (S3)**

DEPARTMENT OF DIGITAL LOGIC AND CIRCUIT (EL 101)

NIIT UNIVERSITY, NH – 8, DELHI – JAIPUR HIGHWAY, NEEMRANA, RAJASTHAN – 301705

[2017 – 2018]

A
Project Report
On

RFID – BASED SECURITY SYSTEM

In partial fulfillment of requirements for the course of

Digital Logic and Circuit (EL 101)

In the Degree of

Bachelor of Technology (Computer Science and Engineering)

Submitted By:
1. Ajinkya Bedekar
2. Biren Sharma
3. Shantanu Bahuguna
4. Yogesh Sharma

Under the Guidance of
Mr. Vikas Upadhyay
Mr. Narendra Bisht

DEPARTMENT OF DIGITAL LOGIC AND CIRCUIT (EL 101)

NIIT UNIVERSITY, NH – 8, DELHI – JAIPUR HIGHWAY, NEEMRANA, RAJASTHAN – 301705

CERTIFICATE

This is to certify that the project entitled "RFID Based Security System" has been carried out by the team under my guidance in partial fulfillment of the Digital Logic and Circuit (EL 101) in NU during the academic year 2017–2018 (Semester–3).

Team:

- Ajinkya Bedekar
- Biren Sharma
- Shantanu Bahuguna
- Yogesh Sharma

Date: November 23, 2017

Place: Neemrana

Guide
(Mr. Narendra Bisht)

Head, EC Department
(Mr. Vikas Upadhyay)

Principal
(Professor V S Rao)

External guide
(Mr. Arghya Guchhait)

ACKNOWLEDGEMENT

This is an opportunity to humbly express our thankfulness to all those people concerned with our project entitled "Voice controlled automation & Security System".

At the time of complication of this project, we would like to thank everyone who made a great effort to make this possible. Our project wouldn't exist without help of our project leaders. Trying to cover next to impossible then also we have bought all facts in the application with the help of our guides.

I express very sincere thanks to our college faculties for their guidance throughout preparation of project. Their valuable guidance has proved to be a key to our success In overcoming challenges we faced during this project. At last, we would like to thank all the members related to this project for their kind co-operation and help during the project. Again as the law of nature "No One Is PERFECT", I will be very glad to know my mistakes and also if any suggestions are there.

Ajinkya Bedekar

Biren Sharma

Shantanu Bahuguna

Yogesh Sharma

RFID BASED SECURITY SYSTEM

Submitted By:

Ajinkya Bedekar (U101116FCS183)
Biren Sharma (U101116FCS246)
Shantanu Bahuguna (U101116FCS248)
Yogesh Sharma (U101116FCS247)

ABSTRACT

The security system is basically an embedded one. Embedded stands for hardware controlled by software. Here, the software using a microcontroller controls all the hardware components. The microcontroller plays an important role in the system.

The main objective of the system is to uniquely identify and to make security for a person. This requires a unique product, which has the capability of distinguishing different person. This is possible by the new emerging technology RFID (Radio Frequency Identification). The main parts of an RFID system are RFID tag (with unique ID number) and RFID reader (for reading the RFID tag). In this system, RFID tag and RFID reader used are operating at 125KHz. The microcontroller internal memory is used for storing the details. The PC can be used for restoring all the details of security made.

This report provides a clear picture of hardware and software used in the system. It also provides an overall view with detailed discussion of the operation of the system.

Introduction

Introduction:

Most educational institutions administrators are concerned about student security. The conventional method allowing access to students inside a college/educational campus is by showing photo I-Card to security guard is very time consuming and insecure, hence inefficient.

Radio Frequency Identification (RFID) based security system is one of the solutions to address this problem. This system can be used to allow access for student in school, college, and university. It also can be used to take attendance for workers in working places. Its ability to uniquely identify each person based on their RFID tag type of ID card make the process of allowing security access easier, faster and secure as compared to conventional method.

Student or workers only need to place their ID card on the reader and they will be allowed to enter the campus. And if any invalid card is shown then the buzzer is turned on.

Feasibility

2.1 Financial feasibility:

The resources used in this project are quite feasible financially.

Components list:

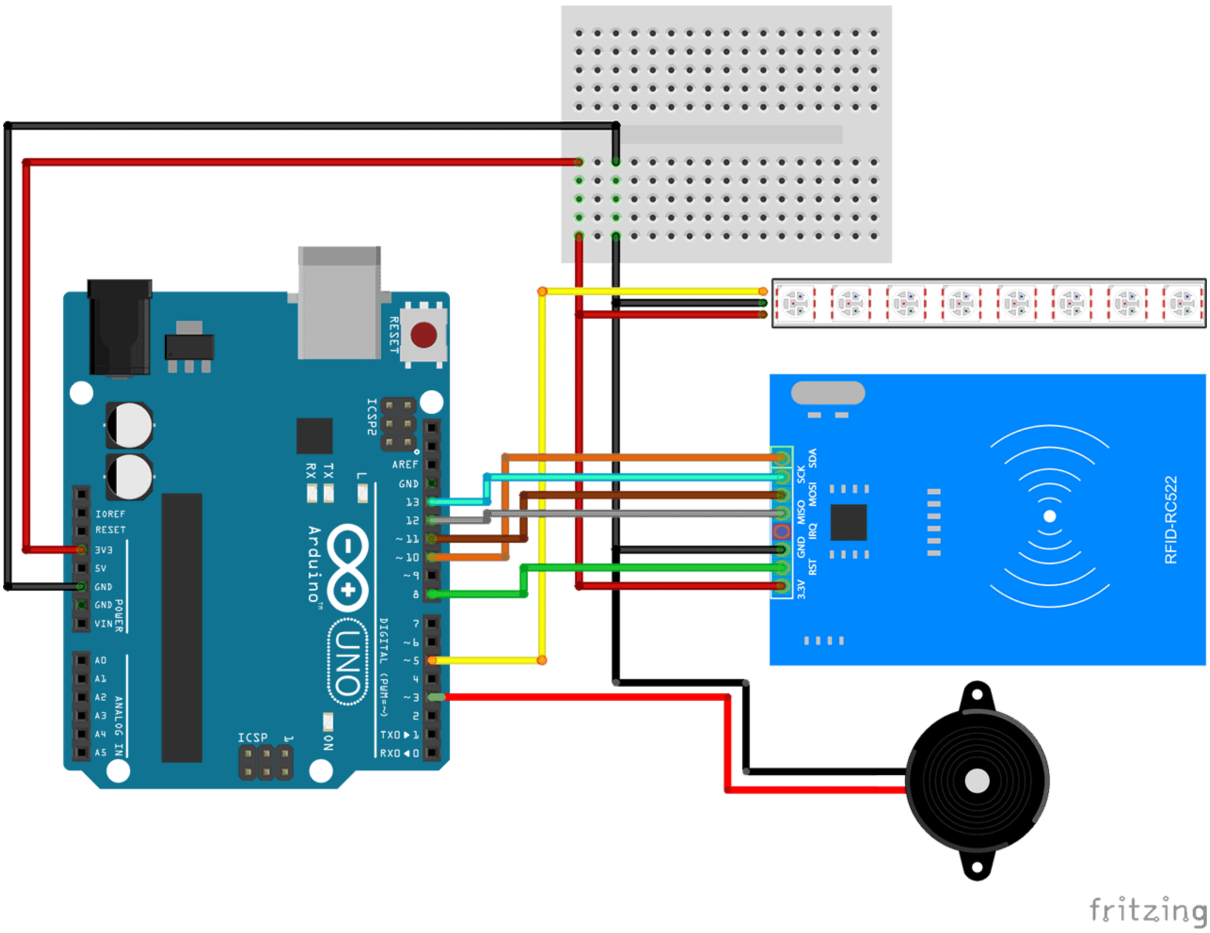
- Resisters
- LED
- RFID Module
- Arduino
- Servo Motor
- Wooden Door
- Hinge
- Latch
- RFID Tag

The list of components given above shows that all the components are cheap and feasible. The company will not have any problem in using this simple project circuit.

Block Diagram of The Project

Construction

Block Diagram:-



History of RFID:

In a very interesting article, the San Jose Mercury News tells us about Charles Walton, the man behind the radio frequency identification technology (RFID). Since his first patent about it in 1973, Walton, now 83 years old, collected about $3 million from royalties coming from his patents. Unfortunately for him, his latest patent about RFID expired in the mid-1990s. So he will not make any money from the billions of RFID tags that will appear in the years to come. But he continues to invent and his latest patent about a proximity card with incorporated PIN code protection was granted in June 2004.

What is RFID.

RFID is short for Radio Frequency Identification. Generally, a RFID system consists of 2 parts. A Reader, and one or more Transponders, also known as Tags. RFID systems evolved from barcode labels as a means to automatically identify and track products and people. You will be generally familiar with RFID systems as seen in:

Access Control

RFID Readers placed at entrances that require a person to pass their proximity card (RF tag) to be read before the access can be made.

Contact less Payment Systems

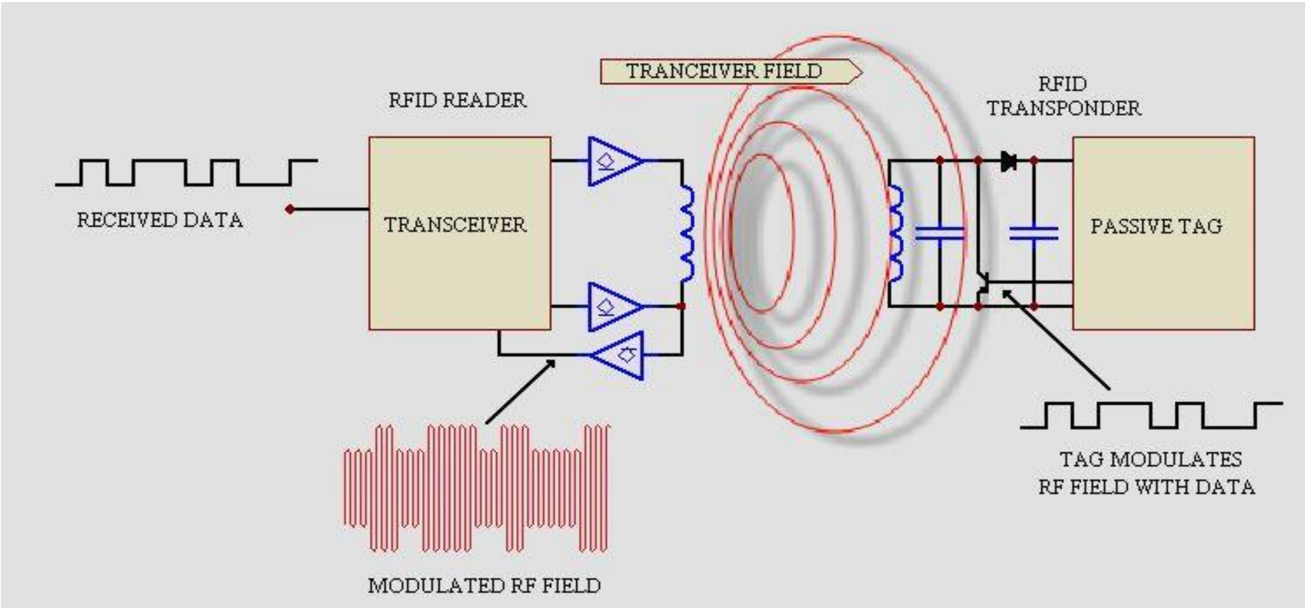
RFID tags used to carry payment information. RFIDs are particular suited to electronic Toll collection system. Tags attached to vehicles, or carried by people transmit payment information to a fix reader attached to a Toll station. Payments are then routinely deducted from a users account, or information is changed directly on the RFID tag.

Product Tracking and Inventory Control

RFID systems are commonly used to track and record the movement of ordinary items such as library books, clothes, factory pallets, electrical goods and numerous items.

How do RFIDs work

Shown below is a typical RFID system. In every RFID system the transponder Tags contain information. This information can be as little as a single binary bit, or be a large array of bits representing such things as an identity code, personal medical information, or literally any type of information that can be stored in digital binary format.



Shown is a RFID transceiver that communicates with a passive Tag. Passive tags have no power source of their own and instead derive power from the incident electromagnetic field.

Commonly the heart of each tag is a microchip. When the Tag enters the generated RF field it can draw enough power from the field to access its internal memory and transmit its stored information.

When the transponder Tag draws power in this way the resultant interaction of the RF fields causes the voltage at the transceiver antenna to drop in value. This effect is utilized by the Tag to communicate its information to the reader. The Tag is able to control the amount of power drawn from the field and by doing so it can modulate the voltage sensed at the Transceiver according to the bit pattern it wishes to transmit.

<u>COMPONENTS OF RFID</u>

A basic RFID system consist of three components:

- An antenna or coil
- A transceiver (with decoder)
- A transponder (RF tag) electronically programmed with unique information

These are described below:

1. ANTENNA

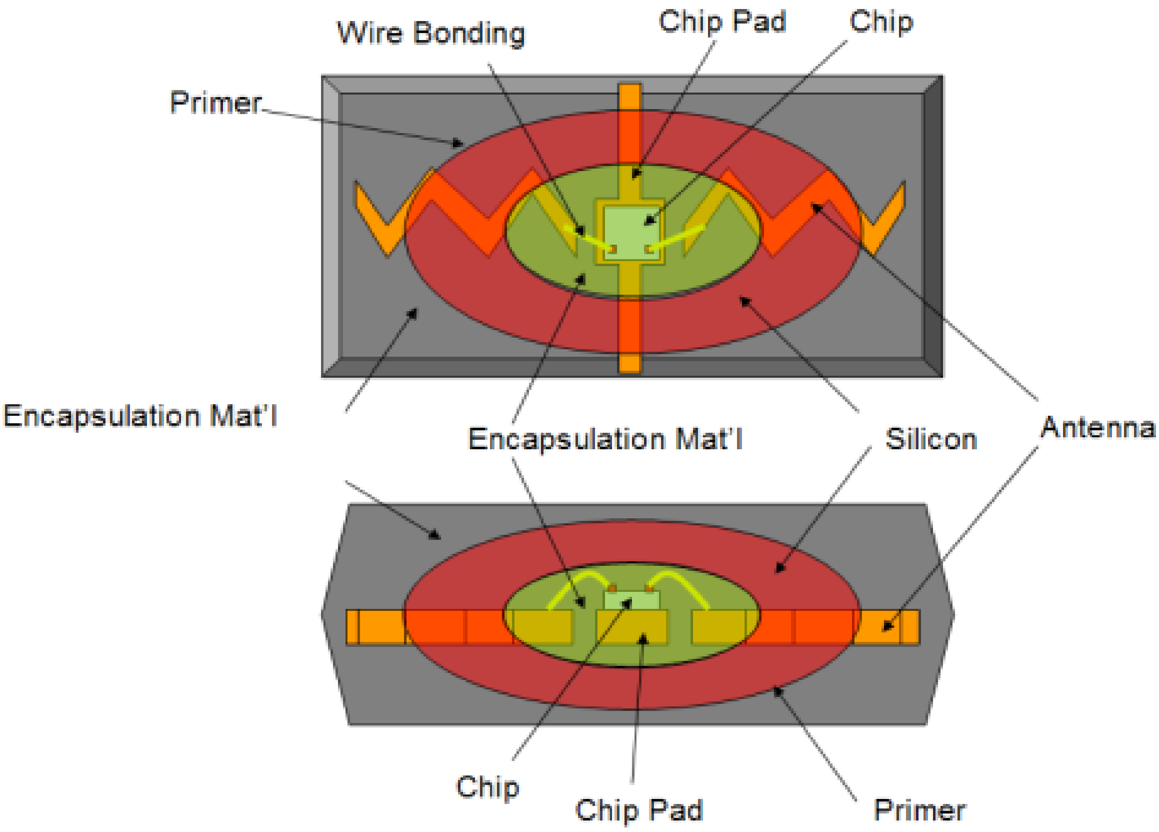
The antenna emits radio signals to activate the tag and read and write data to it. Antennas are the conduits between the tag and the transceiver, which controls the system's data acquisition and communication.

Antennas are available in a variety of shapes and sizes; they can be built into a door frame to receive tag data from persons or things passing through the door, or mounted on an interstate tollbooth to monitor traffic passing by on a freeway.

The electromagnetic field produced by an antenna can be constantly present when multiple tags are expected continually. If constant interrogation is not required, a sensor device can activate the field.

Often the antenna is packaged with the transceiver and decoder to become a reader (a.k.a. interrogator), which can be configured either as a handheld or a fixed-mount device. The reader emits radio waves in ranges of anywhere from one inch to 100 feet or more, depending upon its power output and the radio frequency used.

When an RFID tag passes through the electromagnetic zone, it detects the reader's activation signal. The reader decodes the data encoded in the tag's integrated circuit (silicon chip) and the data is passed to the host computer for processing.



2. TAGS (Transponders)

An RFID tag is comprised of a microchip containing identifying information and an antenna that transmits this data wirelessly to a reader. At its most basic, the chip will contain a serialized identifier, or license plate number, that uniquely identifies that item, similar to the way many bar codes are used today.

A key difference, however is that RFID tags have a higher data capacity than their bar code counterparts. This increases the options for the type of information that can be encoded on the tag, including the manufacturer, batch or lot number, weight, ownership, destination and history (such as the temperature range to which an item has been exposed).

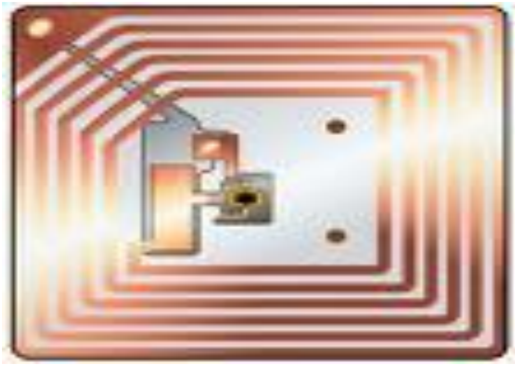
In fact, an unlimited list of other types of information can be stored on RFID tags, depending on application needs. An RFID tag can be placed on individual items, cases or pallets for identification purposes, as well as on fixed assets such as trailers, containers, totes, etc.

Tags come in a variety of types, with a variety of capabilities. Key variables include:"Read-only" versus "read-write"

There are three options in terms of how data can be encoded on tags: (1) Read-only tags contain data such as a serialized tracking number, which is pre-written onto them by the tag manufacturer or distributor. These are generally the least expensive tags because they cannot have any additional information included as they move throughout the supply chain.

Any updates to that information would have to be maintained in the application software that tracks SKU movement and activity. (2) "Write once" tags enable a user to write data to the tag one time in production or distribution processes. Again, this may include a serial number, but perhaps other data such as a lot or batch number. (3) Full "read-write" tags allow new data to be written to the tag as needed—and even written over the original data.

Examples for the latter capability might include the time and dateof ownership transfer or updating the repair history of a fixed asset. While these are the most costly of the three tag types and are not practical for tracking inexpensive items, future standards for electronic product codes (EPC) appear to be headed in this direction.



RFID TAGS

Data capacity

The amount of data storage on a tag can vary, ranging from 16 bits on the low end to as much as several thousand bits on the high end. Of course, the greater the storage capacity, the higher the price per tag.

Form factor

The tag and antenna structure can come in a variety of physical form factors and can either be self-contained or embedded as part of a traditional label structure (i.e., the tag is inside what looks like a regular bar code label—this is termed a 'Smart Label') companies must choose the appropriate form factors for the tag very carefully and should expect to use multiple form factors to suit the tagging needs of different physical products and units of measure.

For example, a pallet may have an RFID tag fitted only to an area of protected placement on the pallet itself. On the other hand, cartons on the pallet have RFID tags inside bar code labels that also provide operators human-readable information and a back-up should the tag fail or pass through non RFID-capable supply chain links.

Passive versus active

“Passive” tags have no battery and "broadcast" their data only when energized by a reader. That means they must be actively polled to send information. "Active" tags are capable of broadcasting their data using their own battery power.

In general, this means that the read ranges are much greater for active tags than they are for passive tags—perhaps a read range of 100 feet or more, versus 15 feet or less for most passive tags. The extra capability and read ranges of active tags, however, come with a cost; they are several times more expensive than passive tags.

Today, active tags are much more likely to be used for high-value items or fixed assets such as trailers, where the cost is minimal compared to item value, and very long read ranges are required. Most traditional supply chain applications, such as the RFID-based tracking and compliance programs emerging in the consumer goods retail chain, will use the less expensive passive tags.

Frequencies

Like all wireless communications, there are a variety of frequencies or spectra through which RFID tags can communicate with readers. Again, there are trade-offs among cost, performance and application requirements. For instance, low-frequency tags are cheaper than ultra high-frequency (UHF) tags, use less power and are better able to penetrate non-metallic substances.

They are ideal for scanning objects with high water content, such as fruit, at close range. UHF frequencies typically offer better range and can transfer data faster. But they use more power and are less likely to pass through some materials. UHF tags are typically best suited for use with or near wood, paper, cardboard or clothing products. Compared to low-frequency tags, UHF tags might be better for scanning boxes of goods as they pass through a bay door into a warehouse.

While the tag requirements for compliance mandates may be narrowly defined, it is likely that a variety of tag types will be required to solve specific operational issues. You will want to work with a company that is very knowledgeable in tag and reader technology to appropriately identify the right mix of RFID technology for your environment and applications.

EPC Tags

EPC refers to "electronic product code," an emerging specification for RFID tags, readers and business applications first developed at the Auto-ID Center at the Massachusetts Institute of Technology. This organization has provided significant intellectual leadership toward the use and application of RFID technology.

EPC represents a specific approach to item identification, including an emerging standard for the tags themselves, including both the data content of the tag and open wireless communication protocols. In a sense, the EPC movement is combining the data standards embodied in certain bar code specifications, such as the UPC or UCC-128 bar code standards, with the wireless data communication standards that have been developed by ANSI and other groups.

3. RF Transceiver:

The RF transceiver is the source of the RF energy used to activate and power the passive RFID tags. The RF transceiver may be enclosed in the same cabinet as the reader or it may be a separate piece of equipment. When provided as a separate piece of equipment, the transceiver is commonly referred to as an RF module.

The RF transceiver controls and modulates the radio frequencies that the antenna transmits and receives. The transceiver filters and amplifies the backscatter signal from a passive RFID tag.

Typical Applications for RFID

- Automatic Vehicle identification
- Inventory Management
- Work-in-Process
- Container/ Yard Management
- Document/Jewellery tracking
- Patient Monitoring

Code Uploaded

Arduino Code:

```
#include <RFID.h>
#include <Servo.h>
#include <SPI.h>


#define SS_PIN 10
#define RST_PIN 9
#define GREEN_LED 6
#define RED_LED 7
Servo myservo;
RFID rfid(SS_PIN, RST_PIN);

unsigned char reading_card[5]; //for reading card
unsigned char master[5] = {66, 107 ,2 ,13 ,38 };
unsigned char i;

void indication(int led);
void allow();
void denied();
int pos = 0;
void setup()
{
   myservo.attach(3);
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  myservo.write(20);
}

void loop()
{
    if (rfid.isCard())
    {
        if (rfid.readCardSerial())
        {
                /* Reading card */
                Serial.println(" ");
                Serial.println("Card found");
                Serial.println("Cardnumber:");
                for (i = 0; i < 5; i++)
                {
                  Serial.print(rfid.serNum[i]);
                  Serial.print(" ");
                  reading_card[i] = rfid.serNum[i];
                }
                Serial.println();
                //verification
```

```
                for (i = 0; i < 5; i++)
                {
                  if (reading_card[i]!=master[i])
                  {
                    break;
                  }
                }
                if (i == 5)
                {
                  allow();
                }
                else
                {
                  denied();
                }
         }
    }
    rfid.halt();
}

void allow()
{
  Serial.println("Access Granted!");
  indication(GREEN_LED);
  myservo.write(90);
  digitalWrite(GREEN_LED, HIGH);
  delay(2000);
  digitalWrite(GREEN_LED, LOW);
 myservo.write(20);
}
void denied()
{
  Serial.println("Access denied!");
  indication(RED_LED);
}
void indication(int led)
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);

}
```

RFID Module Library:

RFID.h

```
/* RFID.h - Library to use ARDUINO RFID MODULE KIT 13.56 MHZ WITH
TAGS SPI W AND R BY COOQROBOT.
 * Based on code Dr.Leong   ( WWW.B2CQSHOP.COM )
 * Created by Miguel Balboa (circuitito.com), Jan, 2012.
 */
#ifndef RFID_h
```

```
#define RFID_h

#include <Arduino.h>
#include <SPI.h>



/*****************************************************************
***********
 * Definitions

*****************************************************************
**********/
#define MAX_LEN 16   // Largo máximo de la matriz

//MF522 comando palabra
#define PCD_IDLE              0x00               // NO action; Y
cancelar el comando
#define PCD_AUTHENT           0x0E               // autenticación de
clave
#define PCD_RECEIVE           0x08               // recepción de
datos
#define PCD_TRANSMIT          0x04               // Enviar datos
#define PCD_TRANSCEIVE        0x0C               // Enviar y recibir
datos
#define PCD_RESETPHASE        0x0F               // reajustar
#define PCD_CALCCRC           0x03               // CRC calcular

//Mifare_One  Tarjeta Mifare_One comando palabra
#define PICC_REQIDL           0x26               // Área de la
antena no está tratando de entrar en el estado de reposo
#define PICC_REQALL           0x52               // Todas las cartas
para encontrar el área de la antena
#define PICC_ANTICOLL         0x93               // anti-colisión
#define PICC_SElECTTAG        0x93               // elección de
tarjeta
#define PICC_AUTHENT1A        0x60               // verificación key
A
#define PICC_AUTHENT1B        0x61               // verificación Key
B
#define PICC_READ             0x30               // leer bloque
#define PICC_WRITE            0xA0               // Escribir en el
bloque
#define PICC_DECREMENT        0xC0               // cargo
#define PICC_INCREMENT        0xC1               // recargar
#define PICC_RESTORE          0xC2               // Transferencia de
datos de bloque de buffer
#define PICC_TRANSFER         0xB0               // Guardar los
datos en el búfer
#define PICC_HALT             0x50               // inactividad

//MF522 Código de error de comunicación cuando regresó
#define MI_OK                 0
#define MI_NOTAGERR           1
#define MI_ERR                2
```

```
//------------------ MFRC522 registro---------------
//Page 0:Command and Status
#define     Reserved00          0x00
#define     CommandReg          0x01
#define     CommIEnReg          0x02
#define     DivlEnReg           0x03
#define     CommIrqReg          0x04
#define     DivIrqReg           0x05
#define     ErrorReg            0x06
#define     Status1Reg          0x07
#define     Status2Reg          0x08
#define     FIFODataReg         0x09
#define     FIFOLevelReg        0x0A
#define     WaterLevelReg       0x0B
#define     ControlReg          0x0C
#define     BitFramingReg       0x0D
#define     CollReg             0x0E
#define     Reserved01          0x0F
//Page 1:Command
#define     Reserved10          0x10
#define     ModeReg             0x11
#define     TxModeReg           0x12
#define     RxModeReg           0x13
#define     TxControlReg        0x14
#define     TxAutoReg           0x15
#define     TxSelReg            0x16
#define     RxSelReg            0x17
#define     RxThresholdReg      0x18
#define     DemodReg            0x19
#define     Reserved11          0x1A
#define     Reserved12          0x1B
#define     MifareReg           0x1C
#define     Reserved13          0x1D
#define     Reserved14          0x1E
#define     SerialSpeedReg      0x1F
//Page 2:CFG
#define     Reserved20          0x20
#define     CRCResultRegM       0x21
#define     CRCResultRegL       0x22
#define     Reserved21          0x23
#define     ModWidthReg         0x24
#define     Reserved22          0x25
#define     RFCfgReg            0x26
#define     GsNReg              0x27
#define     CWGsPReg          0x28
#define     ModGsPReg           0x29
#define     TModeReg            0x2A
#define     TPrescalerReg       0x2B
#define     TReloadRegH         0x2C
#define     TReloadRegL         0x2D
#define     TCounterValueRegH   0x2E
#define     TCounterValueRegL   0x2F
//Page 3:TestRegister
#define     Reserved30          0x30
#define     TestSel1Reg         0x31
#define     TestSel2Reg         0x32
```

```
#define     TestPinEnReg          0x33
#define     TestPinValueReg       0x34
#define     TestBusReg            0x35
#define     AutoTestReg           0x36
#define     VersionReg            0x37
#define     AnalogTestReg         0x38
#define     TestDAC1Reg           0x39
#define     TestDAC2Reg           0x3A
#define     TestADCReg            0x3B
#define     Reserved31            0x3C
#define     Reserved32            0x3D
#define     Reserved33            0x3E
#define     Reserved34                  0x3F
//-----------------------------------------------

class RFID
{
  public:
    RFID(int chipSelectPin, int NRSTPD);

      bool isCard();
      bool readCardSerial();

    void init();
      void reset();
      void writeMFRC522(unsigned char addr, unsigned char val);
      void antennaOn(void);
      unsigned char readMFRC522(unsigned char addr);
      void setBitMask(unsigned char reg, unsigned char mask);
      void clearBitMask(unsigned char reg, unsigned char mask);
      void calculateCRC(unsigned char *pIndata, unsigned char len,
unsigned char *pOutData);
      unsigned char MFRC522Request(unsigned char reqMode, unsigned
char *TagType);
      unsigned char MFRC522ToCard(unsigned char command, unsigned
char *sendData, unsigned char sendLen, unsigned char *backData,
unsigned int *backLen);
      unsigned char anticoll(unsigned char *serNum);
      unsigned char auth(unsigned char authMode, unsigned char
BlockAddr, unsigned char *Sectorkey, unsigned char *serNum);
      unsigned char read(unsigned char blockAddr, unsigned char
*recvData);
      unsigned char write(unsigned char blockAddr, unsigned char
*writeData);
      void halt();

      unsigned char serNum[5];       // Constante para guardar el
numero de serie leido.
      unsigned char AserNum[5];      // Constante para guardar el
numero d serie de la secion actual.

  private:
    int _chipSelectPin;
      int _NRSTPD;

};
```

```
#endif
```

RFID.cpp

```
/*
 * RFID.cpp - Library to use ARDUINO RFID MODULE KIT 13.56 MHZ WITH
TAGS SPI W AND R BY COOQROBOT.
 * Based on code Dr.Leong   ( WWW.B2CQSHOP.COM )
 * Created by Miguel Balboa, Jan, 2012.
 * Released into the public domain.
 */

/******************************************************************
***********
 * Includes

*******************************************************************
**********/
#include <Arduino.h>
#include <RFID.h>

/******************************************************************
***********
 * User API

*******************************************************************
**********/

/**
 * Construct RFID
 * int chipSelectPin RFID /ENABLE pin
 */
RFID::RFID(int chipSelectPin, int NRSTPD)
{
     _chipSelectPin = chipSelectPin;

  pinMode(_chipSelectPin,OUTPUT);              // Set digital as
OUTPUT to connect it to the RFID /ENABLE pin
  digitalWrite(_chipSelectPin, LOW);


  pinMode(NRSTPD,OUTPUT);                      // Set digital pin,
Not Reset and Power-down
  digitalWrite(NRSTPD, HIGH);
  _NRSTPD = NRSTPD;
}
/******************************************************************
***********
 * User API

*******************************************************************
**********/
```

```
 bool RFID::isCard()
 {
      unsigned char status;
      unsigned char str[MAX_LEN];

      status = MFRC522Request(PICC_REQIDL, str);
    if (status == MI_OK) {
           return true;
      } else {
           return false;
      }
 }

 bool RFID::readCardSerial(){

      unsigned char status;
      unsigned char str[MAX_LEN];

      // Anti-colisión, devuelva el número de serie de tarjeta de 4
bytes
      status = anticoll(str);
      memcpy(serNum, str, 5);

      if (status == MI_OK) {
           return true;
      } else {
           return false;
      }

 }

/******************************************************************
***********
 * Dr.Leong   ( WWW.B2CQSHOP.COM )

*******************************************************************
**********/

void RFID::init()
{
    digitalWrite(_NRSTPD,HIGH);

      reset();

      //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    writeMFRC522(TModeReg, 0x8D);           //Tauto=1; f(Timer) =
6.78MHz/TPreScaler
    writeMFRC522(TPrescalerReg, 0x3E);//TModeReg[3..0] +
TPrescalerReg
    writeMFRC522(TReloadRegL, 30);
    writeMFRC522(TReloadRegH, 0);

      writeMFRC522(TxAutoReg, 0x40);         //100%ASK
      writeMFRC522(ModeReg, 0x3D);           // CRC valor inicial de
0x6363
```

```
	//ClearBitMask(Status2Reg, 0x08);	//MFCrypto1On=0
	//writeMFRC522(RxSelReg, 0x86);	//RxWait =
RxSelReg[5..0]
	//writeMFRC522(RFCfgReg, 0x7F);	//RxGain = 48dB

	antennaOn();		//Abre  la antena


}
void RFID::reset()
{
	writeMFRC522(CommandReg, PCD_RESETPHASE);
}

void RFID::writeMFRC522(unsigned char addr, unsigned char val)
{
	digitalWrite(_chipSelectPin, LOW);

	//0XXXXXX0 formato de dirección
	SPI.transfer((addr<<1)&0x7E);
	SPI.transfer(val);

	digitalWrite(_chipSelectPin, HIGH);
}

void RFID::antennaOn(void)
{
	unsigned char temp;

	temp = readMFRC522(TxControlReg);
	if (!(temp & 0x03))
	{
		setBitMask(TxControlReg, 0x03);
	}
}

/*
 *  Read_MFRC522 Nombre de la función: Read_MFRC522
 *  Descripción: Desde el MFRC522 leer un byte de un registro de
datos
 *  Los parámetros de entrada: addr - la dirección de registro
 *  Valor de retorno: Devuelve un byte de datos de lectura
 */
unsigned char RFID::readMFRC522(unsigned char addr)
{
	unsigned char val;
	digitalWrite(_chipSelectPin, LOW);
	SPI.transfer(((addr<<1)&0x7E) | 0x80);
	val =SPI.transfer(0x00);
	digitalWrite(_chipSelectPin, HIGH);
	return val;
}

void RFID::setBitMask(unsigned char reg, unsigned char mask)
{
```

```
    unsigned char tmp;
    tmp = readMFRC522(reg);
    writeMFRC522(reg, tmp | mask);  // set bit mask
}

void RFID::clearBitMask(unsigned char reg, unsigned char mask)
{
    unsigned char tmp;
    tmp = readMFRC522(reg);
    writeMFRC522(reg, tmp & (~mask));  // clear bit mask
}

void RFID::calculateCRC(unsigned char *pIndata, unsigned char len,
unsigned char *pOutData)
{
    unsigned char i, n;

    clearBitMask(DivIrqReg, 0x04);              //CRCIrq = 0
    setBitMask(FIFOLevelReg, 0x80);             //Claro puntero
FIFO
    //Write_MFRC522(CommandReg, PCD_IDLE);

      //Escribir datos en el FIFO
    for (i=0; i<len; i++)
    {
           writeMFRC522(FIFODataReg, *(pIndata+i));
      }
    writeMFRC522(CommandReg, PCD_CALCCRC);

      // Esperar a la finalización de cálculo del CRC
    i = 0xFF;
    do
    {
        n = readMFRC522(DivIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x04));            //CRCIrq = 1

      //Lea el cálculo de CRC
    pOutData[0] = readMFRC522(CRCResultRegL);
    pOutData[1] = readMFRC522(CRCResultRegM);
}

unsigned char RFID::MFRC522ToCard(unsigned char command, unsigned
char *sendData, unsigned char sendLen, unsigned char *backData,
unsigned int *backLen)
{
    unsigned char status = MI_ERR;
    unsigned char irqEn = 0x00;
    unsigned char waitIRq = 0x00;
      unsigned char lastBits;
    unsigned char n;
    unsigned int i;

    switch (command)
    {
```

```
        case PCD_AUTHENT:       // Tarjetas de certificación cerca
           {
                 irqEn = 0x12;
                 waitIRq = 0x10;
                 break;
           }
           case PCD_TRANSCEIVE:  //La transmisión de datos FIFO
           {
                 irqEn = 0x77;
                 waitIRq = 0x30;
                 break;
           }
           default:
                 break;
    }

    writeMFRC522(CommIEnReg, irqEn|0x80);   //De solicitud de
interrupción
    clearBitMask(CommIrqReg, 0x80);              // Borrar todos los
bits de petición de interrupción
    setBitMask(FIFOLevelReg, 0x80);              //FlushBuffer=1,
FIFO de inicialización

     writeMFRC522(CommandReg, PCD_IDLE);   //NO action;Y cancelar
el comando

     //Escribir datos en el FIFO
    for (i=0; i<sendLen; i++)
    {
          writeMFRC522(FIFODataReg, sendData[i]);
     }

     //???? ejecutar el comando
     writeMFRC522(CommandReg, command);
    if (command == PCD_TRANSCEIVE)
    {
          setBitMask(BitFramingReg, 0x80);
     //StartSend=1,transmission of data starts
     }

     // A la espera de recibir datos para completar
     i = 2000;  //i????????,??M1???????25ms??? i De acuerdo con el
ajuste de frecuencia de reloj, el tiempo máximo de espera operación
M1 25ms tarjeta??
    do
    {
          //CommIrqReg[7..0]
          //Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq
TimerIRq
        n = readMFRC522(CommIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x01) && !(n&waitIRq));

    clearBitMask(BitFramingReg, 0x80);           //StartSend=0
```

```
    if (i != 0)
    {
        if(!(readMFRC522(ErrorReg) & 0x1B)) //BufferOvfl Collerr
CRCErr ProtecolErr
        {
            status = MI_OK;
            if (n & irqEn & 0x01)
            {
                    status = MI_NOTAGERR;            //??
                }

            if (command == PCD_TRANSCEIVE)
            {
                n = readMFRC522(FIFOLevelReg);
                lastBits = readMFRC522(ControlReg) & 0x07;
                if (lastBits)
                {
                        *backLen = (n-1)*8 + lastBits;
                    }
                else
                {
                        *backLen = n*8;
                    }

                if (n == 0)
                {
                        n = 1;
                    }
                if (n > MAX_LEN)
                {
                        n = MAX_LEN;
                    }

                    //??FIFO??????? Lea los datos recibidos en el
FIFO
                for (i=0; i<n; i++)
                {
                        backData[i] = readMFRC522(FIFODataReg);
                    }
            }
        }
        else
        {
                status = MI_ERR;
            }

    }

    //SetBitMask(ControlReg,0x80);           //timer stops
    //Write_MFRC522(CommandReg, PCD_IDLE);

    return status;
}


/*
```

```
 *  Nombre de la función: MFRC522_Request
 *  Descripción: Buscar las cartas, leer el número de tipo de
tarjeta
 *  Los parámetros de entrada: reqMode - encontrar el modo de
tarjeta,
 *                  Tagtype - Devuelve el tipo de tarjeta
 *                    0x4400 = Mifare_UltraLight
 *                    0x0400 = Mifare_One(S50)
 *                    0x0200 = Mifare_One(S70)
 *                    0x0800 = Mifare_Pro(X)
 *                    0x4403 = Mifare_DESFire
 *  Valor de retorno: el retorno exitoso MI_OK
 */
unsigned char  RFID::MFRC522Request(unsigned char reqMode, unsigned
char *TagType)
{
     unsigned char status;
     unsigned int backBits;                //   Recibió bits de
datos

     writeMFRC522(BitFramingReg, 0x07);          //TxLastBists =
BitFramingReg[2..0]   ???

     TagType[0] = reqMode;
     status = MFRC522ToCard(PCD_TRANSCEIVE, TagType, 1, TagType,
&backBits);

     if ((status != MI_OK) || (backBits != 0x10))
     {
          status = MI_ERR;
     }

     return status;
}

/**
 *  MFRC522Anticoll -> anticoll
 *  Anti-detección de colisiones, la lectura del número de serie de
la tarjeta de tarjeta
 *  @param serNum - devuelve el número de tarjeta 4 bytes de serie,
los primeros 5 bytes de bytes de paridad
 *  @return retorno exitoso MI_OK
 */
unsigned char RFID::anticoll(unsigned char *serNum)
{
    unsigned char status;
    unsigned char i;
     unsigned char serNumCheck=0;
    unsigned int unLen;


    //ClearBitMask(Status2Reg, 0x08);       //TempSensclear
    //ClearBitMask(CollReg,0x80);                //ValuesAfterColl
     writeMFRC522(BitFramingReg, 0x00);          //TxLastBists =
BitFramingReg[2..0]
```

```
    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522ToCard(PCD_TRANSCEIVE, serNum, 2, serNum,
&unLen);

    if (status == MI_OK)
     {
           //?????? Compruebe el número de serie de la tarjeta
           for (i=0; i<4; i++)
           {
                 serNumCheck ^= serNum[i];
           }
           if (serNumCheck != serNum[i])
           {
                 status = MI_ERR;
           }
    }

    //SetBitMask(CollReg, 0x80);       //ValuesAfterColl=1

    return status;
}

/*
 * MFRC522Auth -> auth
 * Verificar la contraseña de la tarjeta
 * Los parámetros de entrada: AuthMode - Modo de autenticación de
contraseña
                  0x60 = A 0x60 = validación KeyA
                  0x61 = B 0x61 = validación KeyB
             BlockAddr--  bloque de direcciones
             Sectorkey-- sector contraseña
             serNum--,4? Tarjeta de número de serie, 4 bytes
 * MI_OK Valor de retorno: el retorno exitoso MI_OK
 */
unsigned char RFID::auth(unsigned char authMode, unsigned char
BlockAddr, unsigned char *Sectorkey, unsigned char *serNum)
{
    unsigned char status;
    unsigned int recvBits;
    unsigned char i;
     unsigned char buff[12];

     //????+???+????+???? Verifique la dirección de comandos de
bloques del sector + + contraseña + número de la tarjeta de serie
    buff[0] = authMode;
    buff[1] = BlockAddr;
    for (i=0; i<6; i++)
    {
           buff[i+2] = *(Sectorkey+i);
     }
    for (i=0; i<4; i++)
    {
           buff[i+8] = *(serNum+i);
     }
    status = MFRC522ToCard(PCD_AUTHENT, buff, 12, buff, &recvBits);
```

```
    if ((status != MI_OK) || (!(readMFRC522(Status2Reg) & 0x08)))
    {
           status = MI_ERR;
      }

    return status;
}

/*
 * MFRC522Read -> read
 * Lectura de datos de bloque
 * Los parámetros de entrada: blockAddr - dirección del bloque;
recvData - leer un bloque de datos
 * MI_OK Valor de retorno: el retorno exitoso MI_OK
 */
unsigned char RFID::read(unsigned char blockAddr, unsigned char
*recvData)
{
    unsigned char status;
    unsigned int unLen;

    recvData[0] = PICC_READ;
    recvData[1] = blockAddr;
    calculateCRC(recvData,2, &recvData[2]);
    status = MFRC522ToCard(PCD_TRANSCEIVE, recvData, 4, recvData,
&unLen);

    if ((status != MI_OK) || (unLen != 0x90))
    {
        status = MI_ERR;
    }

    return status;
}

/*
 * MFRC522Write -> write
 * La escritura de datos de bloque
 * blockAddr - dirección del bloque; WriteData - para escribir 16
bytes del bloque de datos
 * Valor de retorno: el retorno exitoso MI_OK
 */
unsigned char RFID::write(unsigned char blockAddr, unsigned char
*writeData)
{
    unsigned char status;
    unsigned int recvBits;
    unsigned char i;
      unsigned char buff[18];

    buff[0] = PICC_WRITE;
    buff[1] = blockAddr;
    calculateCRC(buff, 2, &buff[2]);
    status = MFRC522ToCard(PCD_TRANSCEIVE, buff, 4, buff,
&recvBits);
```

```
    if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) !=
0x0A))
    {
           status = MI_ERR;
      }

    if (status == MI_OK)
    {
        for (i=0; i<16; i++)           //?FIFO?16Byte?? Datos a la
FIFO 16Byte escribir
        {
           buff[i] = *(writeData+i);
        }
        calculateCRC(buff, 16, &buff[16]);
        status = MFRC522ToCard(PCD_TRANSCEIVE, buff, 18, buff,
&recvBits);

           if ((status != MI_OK) || (recvBits != 4) || ((buff[0] &
0x0F) != 0x0A))
        {
                 status = MI_ERR;
           }
    }

    return status;
}


/*
 * MFRC522Halt -> halt
 * Cartas de Mando para dormir
 * Los parámetros de entrada: Ninguno
 * Valor devuelto: Ninguno
 */
void RFID::halt()
{
     unsigned char status;
    unsigned int unLen;
    unsigned char buff[4];

    buff[0] = PICC_HALT;
    buff[1] = 0;
    calculateCRC(buff, 2, &buff[2]);

    status = MFRC522ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);
}
```

Advantages and limitations

Advantages of RFID over bar coding:

1. No "line of sight" requirements: Bar code reads can sometimes be limited or problematic due to the need to have a direct "line of sight" between a scanner and a bar code. RFID tags can be read through materials without line of sight.

2. More automated reading: RFID tags can be read automatically when a tagged product comes past or near a reader, reducing the labor required to scan product and allowing more proactive, real-time tracking.

3. Improved read rates: RFID tags ultimately offer the promise of higher read rates than bar codes, especially in high-speed operations such as carton sortation.

4. Greater data capacity: RFID tags can be easily encoded with item details such as lot and batch, weight, etc.

5. "Write" capabilities: Because RFID tags can be rewritten with new data as supply chain activities are completed, tagged products carry updated information as they move throughout the supply chain.

Common Problems with RFID

Some common problems with RFID are reader collision and tag collision. Reader collision occurs when the signals from two or more readers overlap. The tag is unable to respond to simultaneous queries. Systems must be carefully set up to avoid this problem Tag collision occurs when many tags are present in a small area; but since the read time is very fast, it is easier for vendors to develop systems that ensure that tags respond one at a time. See Problems with RFID for more details.

Conclusion and Bibliography

CONCLUSION:

Thus, there are various applications of this project at different-different places. This project is also cheap and can be used on large scale. One more is by adding different types of controllers like ATMEL(AT89C51/52) etc. and also replacing latest RFID module and some compatible component we can use some different applications at low cost.

BIBLIOGRAPHY

- Circuit reference taken from *www.circuitstoday.com* and *www.electronicsforu.com*
- Reference from *www.technologystudent.com*
- Reference from the book