This is the current WORKING version of the RFID door Control. The only things missing at the moment are:

- a Password for the menu
- a DB De-fragger for when a tag not on the end is deleted. This should be simple to implement however.

This program uses the NewSoftSerial library, and the EXROM library.

Main Functions

```
#include <NewSoftSerial.h>
#include <EXROM.h>

NewSoftSerial rfid(2,3); //RX 2, TX 3 (from Arduino Perspective)

int redLED = 4;
int yellowLED = 5;
int greenLED = 6;

int lock = 7;

void setup()
{
  Serial.begin(9600);
  rfid.begin(9600);
  Serial.println("RFID Reader initialising");
  createHead();
  Serial.println("Please type 'menu' to enter the menu");
  pinMode(lock, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  digitalWrite(yellowLED, HIGH);
}

void loop()
{
  checkRFID();
  enterMenu();
}

void openDoor()
{
  digitalWrite(lock, HIGH);
  digitalWrite(greenLED, HIGH);
  Serial.println("Access Granted");
  delay(3000);
  digitalWrite(lock, LOW);
  digitalWrite(greenLED, LOW);
```

```
  rfid.flush();
}

void closeDoor()
{
  digitalWrite(redLED, HIGH);
  Serial.println("Access Denied");
  delay(1000);
  digitalWrite(redLED, LOW);
  rfid.flush();
}
```

Database functions

```
#define HEADER_P 255
#define HBYTE 250
#define TAGBYTE 251

struct Header
{
  byte hByte;
  int noTag;
}
header;

struct Tag
{
  byte tagByte;
  byte tagData[10];
}
tag;

void writeHead()              //writes a new Header
{
  header.hByte=HBYTE;
  header.noTag=0;
  EXROM.write(HEADER_P, header.hByte);
  EXROM.write(HEADER_P+1, header.noTag);
}

int readHead()              //reads the header if there is one present
{
  EXROM.read(HEADER_P, &header.hByte);
  if(header.hByte==HBYTE)
  {
    EXROM.read(HEADER_P+1, &header.noTag);
    return 1;
  }
```

```
  else
    return 0;
}

int updateHead()            //updates the header with total number of tags
{
  byte hCheck;
  EXROM.read(HEADER_P, &hCheck);
  if(hCheck==HBYTE)
  {
    EXROM.write(HEADER_P+1, header.noTag);
    return 1;
  }
  else
    return 0;
}

int createHead()
{
  Serial.println("Checking for Database");
  if(readHead())
  {
    Serial.println();
    Serial.println("Database present");
  }
  else
  {
    Serial.println();
    Serial.println("Database not present: Creating Database");
    writeHead();
  }
}

void printHead()
{
  Serial.print("HeadID: ");
  Serial.println(header.hByte, DEC);
  Serial.print("HeadData: ");
  Serial.print(header.noTag);
  Serial.println("\r\n");
}

int tagPointer(int tagID) //returns pointer for specified tagID
{
  return HEADER_P + sizeof(header) + (sizeof(tag)*tagID);
}

void writeTag(byte tagW[10], int tagID)
{
  tag.tagByte = TAGBYTE;
  EXROM.write(tagPointer(tagID), tag.tagByte);
```

```
  EXROM.write(tagPointer(tagID)+1, tagW, 10);
  readHead();
  header.noTag++;
  updateHead();
}

int readTag(int tagID)
{
  EXROM.read(tagPointer(tagID), &tag.tagByte);
  if(tag.tagByte==TAGBYTE)
  {
    EXROM.read(tagPointer(tagID)+1, tag.tagData, sizeof(tag.tagData));
    return 1;
  }
  else
    return 0;
}

void printTag(int tagID)
{
  if(readTag(tagID))
  {
    Serial.print("TagID: ");
    Serial.print(tagID);
    Serial.print(" Tag Data: ");
    for(int i=0;i<10;i++)
      Serial.print(tag.tagData[i]);
    Serial.println("\r\n");
  }
  else
  {
    Serial.print("No tag at ID ");
    Serial.print(tagID);
    Serial.println("\r\n");
  }
}

boolean compTag(byte tagA[10], byte tagB[10])
{
  boolean flag = false;
  for(int i=0;i<10;i++)
  {
    if (tagA[i]==tagB[i])
      flag = true;
    else
    {
      flag = false;
      break;
    }
```

```
  }
  return flag;
}

int findTag(byte tagF[10])
{
  int flag = -1;
  readHead();
  for(int i=0;i<header.noTag;i++)
  {
    readTag(i);
    if (compTag(tagF, tag.tagData))
    {
      flag = i;
      break;
    }
  }
  return flag;
}

int addTag(byte tagA[10])
{
  if(findTag(tagA)==-1)
  {
    readHead();
    for(int i=0; i <= header.noTag; i++)
    {
      if(readTag(i)==0)
      {
        writeTag(tagA, i);
        Serial.println("Adding Tag to DB");
        return 0;
      }
    }

  }
  else
    Serial.println("Tag already in Database");
}

void printDB()
{
  Serial.println("Printing DB");
  readHead();
  Serial.println(header.noTag);
    for(int i=0;i<header.noTag;i++)
      printTag(i);
  if(header.noTag == 0)
    Serial.println("No Tags in Database");
}
```

```
void delTag(int tagID)
{
  byte eTag[10] = {
    255,255,255,255,255,255,255,255,255,255                    };
  EXROM.write(tagPointer(tagID), eTag, 10);
  readHead();
  header.noTag--;
  updateHead();
}
```

Menu Functions

```
char menuInput[20];

void enterMenu()
{
  readString();
  if(!strcmp(menuInput, "menu"))
  {
    digitalWrite(redLED, HIGH);
    digitalWrite(greenLED, HIGH);
    mainMenuOptions();
    mainMenu();
    menuInput[0] = '\0';
    return;
  }
}

void readString()
{
  if(Serial.available() > 0)
  {
    delay(100);
    int serialNo = Serial.available();
    for(int i=0; i < serialNo; i++)
    {
      menuInput[i] = Serial.read();
    }
    menuInput[serialNo] = '\0';
    Serial.flush();
  }
}

void mainMenuOptions()
{
  Serial.println("Welcome to the Menu");
  Serial.println("Please choose an option");
  Serial.println("----------------------");
```

```
  Serial.println("1. Open Door");
  Serial.println("2. Print Tag Database");
  Serial.println("3. Add RFID Tag to Database");
  Serial.println("4. Remove RFID Tag from Database");
  Serial.println("5. Re-print this Menu");
  Serial.println("6. Exit Menu");
  Serial.println();
}


void mainMenu()
{
  while(1)
  {
    byte serialIn = 0;
    if(Serial.available())
    {
      serialIn = Serial.read();
      Serial.flush();
    }
    if(serialIn == 49) // 1
    {
      openDoor();
    }
    else if(serialIn == 50) // 2
    {
      printDB();
    }
    else if(serialIn == 51) // 3
    {
      addRFID();
    }
    else if(serialIn == 52) // 4
    {
      delRFID();
    }
    else if(serialIn == 53) // 5
    {
      mainMenuOptions();
    }
    else if(serialIn == 54) // 6
    {
      Serial.println("Exiting Menu...");
      digitalWrite(redLED, LOW);
      digitalWrite(greenLED, LOW);
      return;
    }
    else if(serialIn != 0)
    {
      Serial.println("That is not an option");
      Serial.flush();
```

```
    }
  }
}
```

RFID Functions

```
union tagdata
{
  byte tagin[16];
  struct tag
  {
    byte stx;
    byte data[10];
    byte cs[2];
    byte cr;
    byte lf;
    byte etx;
  }
  tag;
}
tagdata;

boolean readRFID()
{
  if(rfid.available())
  {
    delay(100);
    Serial.println("Reading RFID Tag");
    for(int i=0; i < sizeof(tagdata.tagin); i++)
    {
      tagdata.tagin[i] = rfid.read();
    }
    printRFID();
    rfid.flush();
    return 1;
  }
  else return 0;
}

void printRFID()
{
  for(int i=0; i < sizeof(tagdata.tag.data);i++)
  {
    Serial.print(tagdata.tag.data[i]);
  }
  Serial.println();
}
```

```
boolean checkRFID()
{
  if(readRFID())
  {
    int tagNo = findTag(tagdata.tag.data);
    if(tagNo != -1)
    {
      Serial.println("Tag in Database");
      openDoor();
      return true;
    }
    else
    {
      Serial.println("Tag not in Database");
      closeDoor();
    }
  }
  return false;
}

void addRFID()
{
  Serial.println("Please Scan RFID Tag to add.");
  while(1)
  {
    if(readRFID())
    {
      addTag(tagdata.tag.data);
      return;
    }
  }
}

void delRFID()
{
  Serial.println("Please Scan RFID Tag to delete.");
  while(1)
  {
    if(readRFID())
    {
      delTag(findTag(tagdata.tag.data));
      return;
    }
  }
}
```

Category:Madlab projects

From:
http://testwiki.hecatron.com/ - **Hacman DEMO ONLY**

Permanent link:
http://testwiki.hecatron.com/doku.php?id=old:hackspace:door-system:door-control:version-0-2

Last update: **2022/11/30 16:34**