

Kappa Eta Kappa Professional ECE & CS Fraternity

RFID Door Entry System User Guide

Author: Matthew From

Matfrom@gmail.com

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Name | Date | Explanation of Edits |
| 1 | Matthew From | 12/31/2013 | Initial Release |

Table of Contents

[Introduction](#h.navsytxielan)

[Components](#h.cf14253pc9y1)

[RFID System](#h.et1ahv57o7s8)

[Shield, Door Strike, and Keypad](#h.7kin9syajoux)

[Shield (or custom-made circuit board which sits on top of the arduino)](#h.y5i7hcgzjrr5)

[Door Strike](#h.ivjrpduuf75b)

[Keypad](#h.hk7u9eg13d0n)

[Operation guide](#h.4dooxqj1db05)

[Debugging](#h.xafkcksp8qxq)

[Appendix](#h.zgyngvw2k065)

[Arduino Code](#h.4qas15bg5ha3)

# Introduction

The RFID Door entry system is meant to make it easier for KHK members who do not live in the house to gain entry. It will also eliminate the danger of losing keys and having to change out locks when enough keys are lost.The system is powered by a 9v power supply, the logic is handled by an arduino diecimila, and the latching mechanism is a normally closed door strike. In the event of power outage or the arduino becomes unplugged, keys can still be used to gain entry.

# Components

## RFID System

The RFID system is composed of the RFID card (the UW-Madison Student card) and the RFID (in our case and HID Proxpoint Plus RFID reader).

UW-Madison student ID cards are 35-bit HID Corporate 1000 125kHz RFID Cards. They are encoded in the following format:

P P A A A A A A A A A A A A B B B B B B B B B B B B B B B B B B B B P

P = Parity Bits (0)

A = Vendor ID Code (Identifies UW-Madison as a Vendor)

B = Unique Student ID

The RFID Reader is a microcontroller in itself, which signifies its successful bootup by beeping twice. It has several wires coming out. Most are not used, except for power (5v), ground, and two data lines.

The lines output in the wiegand format, which means that the data lines signify 0 and 1, and are normally held at 5v. When a bit needs to be sent, the respective line is pulled low for a short amount of time. So when a card is swiped, the zero line is pulled low for a little bit, then it returns to 5v. Then the zero line is pulled low again for a little bit and returns to 5v again. Then the respective lines which corresponding to the company code and ID code are pulled low for a little bit and returned to 5v in order, then finally the zero line is pulled low again for a little bit and brought back to 5v.

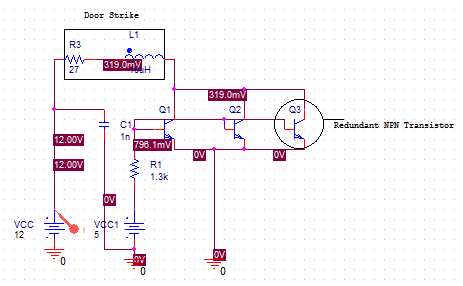
## Shield, Door Strike, and Keypad

These are all together because the shield interfaces the arduino to the shield and keypad.

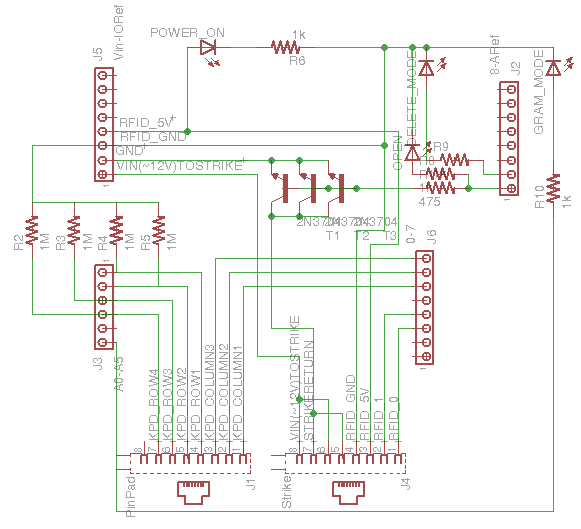
### Shield (or custom-made circuit board which sits on top of the arduino)

The primary purpose of the shield is to drive the electric door strike according to the 5v signal from the arduino. It also lights up LEDs to demonstrate power is flowing and when the door strike is activated.

The shield also provides routing for the wires coming in from the two RJ-45 female receptacles. The schematic (yes, I simulated it in pspice) :



The Schematic for the custom shield:



### Door Strike

The door strike is a Normally Open door strike. What this means is that the relay is open, and when the relay is open the strike is locked into place. Said another way, apply power to the door strike to open the door. When the power is removed, the door won’t open anymore. We purchased ours from Kawamall on ebay, but I’m sure there are other alternatives which will work just fine.

Specs---

Voltage: 12v

Amperage: 450mA

### Keypad

The keypads operates as simply an administrative function. Someone can enter the add code/delete code and the next card swiped (within a pre-defined time) will be added or removed from the database text file on the microsd card

**Arduino**

The arduino does all of the processing for the RFID system. It also has an ethernet shield (which the custom shield mentioned above plugs into) which interfaces the sd card to the arduino.

The arduino takes input from the RFID reader, Keypad, and SD card and sends output to the door strike and SD card. Most of the time the arduino will be idle waiting and someone to swipe a card. Rarely will the add people function be used, and I anticipate the delete function will never be used.

The SD card can also be manually removed from the shield and read by a computer to see the currently loaded student IDs. A big downfall about this method is that there is no way to see the names associated with the numbers.

The sd card shield also has a rj45 network port. This allows the possibility to network the arduino and have it communicate to a server for logs and easy updating of information. This would be amazing to implement, but we have not. If we do implement it, then we could one-up beta chapter :p

# Operation guide

There are three operating modes: normal, delete, and program. delete and program modes can be enabled or disabled by inputting a code into the keypad. (at the time of writing, “1924” will toggle program mode and “2013” will toggle delete mode. Two operation modes cannot be active at the same time.) The LEDs indicate what mode the Arduino is in:

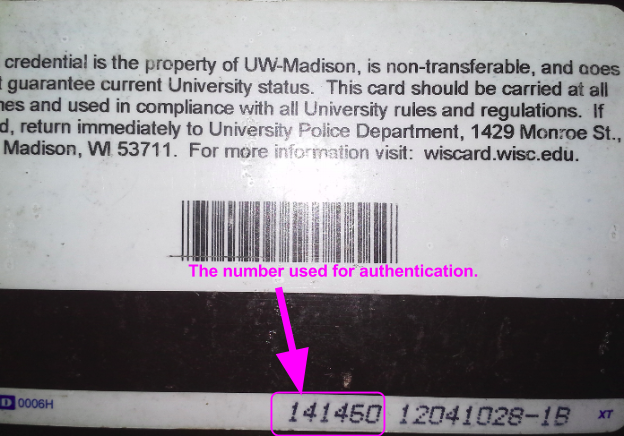
|  |  |
| --- | --- |
| LED | Indication |
| Green | The Arduino has power |
| Red | The door strike is powered |
| Yellow | Arduino is in Delete Mode |
| Purple | Arduino is in Program Mode |

Normal mode is the default operation mode. It is in this mode that the arduino will take input from the RFID reader and compare it to the stored values. If the value matches the stored value, then the door strike is opened for 5 seconds. In addition, the Arduino has a built-in timeout function. When it times out, all the input buffers are cleared and the arduino is returned to normal operation mode.

Program Mode is for adding the next scanned ID(s). This mode can be toggled by inputting “1924” into the keypad. The purple LED will light up in this mode. The Arduino will stay in this mode unless “1924” is entered into the keypad again or the arduinos’ inactivity timer times out.

Delete Mode is for removing the next scanned ID(s). This mode can be toggled by inputting “2013” into the keypad. The Yellow LED will light up while in this mode. The arduino will stay in this mode unless “2013” is entered into the keypad again or the arduinos’ inactivity timer times out.

NOTE: An alternative to using the program and delete modes to add/remove IDs is to remove the microSD card from the arduino and plugging it into a computer. On the card, the IDs are held within “IDfile.txt”. The numbers on every row correspond to one ID. The IDs are printed on the lower right hand corner of the back of the student IDs. Theoretically, an email could be sent out asking for people to send their ID numbers, and this would be a little easier than having everyone scan their ID.



# Debugging

The Arduino is setup to output information to the serial console. This can be accessed by opening up the arduino software, selecting the serial port where the Arduino is connected, and opening the serial monitor. A window will pop up, the Arduino will reset, and the ID count and ID array will print out. Every time an ID is scanned, it is printed out in the console. The keypad entries will also be printed out. If the inactivity timer times out, then a message will also print out.

If an ID is scanned in program mode or delete mode, then the ID will be printed out along with the updated array.

The Serial console can also send commands to the arduino. by sending an “o”, then the door strike is activated for its normal amount of time. Sending a “p” will toggle program mode. Sending a “d” will toggle delete mode. “o” simulates reading a valid ID. “p” and “d” simulate typing in the codes to the keypad.

# Appendix

## Arduino Code

// Bit Explanation: P P A A A A A A A A A A A A B B B B B B B B B B B B B B B B B B B B P

// Bit Numbers: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4

// Example: 1,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,1,0,1,0,0,1

// Stores an ID read from EEPROM

// 0,-,4,-,-,2,-,-,4,-,-,2,-,-,2,-,-,4,-,- =0,424,224

#include <EEPROM.h>// Needed to write to EEPROM storage

#include <SD.h>

char programModeCode[5] = "1924";

char deleteModeCode[5] = "2013";

//temporary

byte incomingByte;

uint16\_t timeKeeper;

#define MAX\_NUM\_IDS 100

//Arduino Hookups

int DELETE\_MODE\_LED = 9; // Yellow LED

int PROGRAM\_MODE\_LED = A5; // Purple LED

#define DOOR\_PIN 8 // Relay

#define SD\_PIN 4; //SD select

//for the keypad

#define COLUMN\_ONE 5

#define COLUMN\_TWO 6

#define COLUMN\_THREE 7

#define ROW\_ONE A0

#define ROW\_TWO A1

#define ROW\_THREE A2

#define ROW\_FOUR A3

char prevChar; //debouncing! woo!

char readChar;

char codeSequence[5];

short unsigned int codeLength = 0;

//bools

boolean noSD = true; // Initialize SD card not present flag to true

boolean programMode = false; // Initialize program mode to false

boolean deleteMode = false; // Initialize delete mode to false

boolean match = false; // initialize card match to false

unsigned long readCard; // Stores an ID read from the RFID reader

int lastNum=-1; // keeps track of the position of the last value in the array

unsigned long IDarray[MAX\_NUM\_IDS];

volatile int bitCount = 0;

// interrupt to read in zeros

void DATA0(void) {

if(bitCount >= 14 && bitCount <=33){

readCard=readCard << 1;

}

bitCount++;

}

// interrupt to read in a one

void DATA1(void) {

if(bitCount >= 14 && bitCount <=33){

readCard =(readCard << 1)+ 1;

}

bitCount++;

}

// on reset, this is run before the main loop

// this sets up the pins, serial communication, interrupts

// reads in from SD card all ids if there is a sd card

void setup()

{

pinMode(COLUMN\_ONE, OUTPUT);

pinMode(COLUMN\_TWO, OUTPUT);

pinMode(COLUMN\_THREE, OUTPUT);

pinMode(DOOR\_PIN, OUTPUT);

pinMode(10,OUTPUT);

pinMode(PROGRAM\_MODE\_LED, OUTPUT);

pinMode(DELETE\_MODE\_LED, OUTPUT);

digitalWrite(PROGRAM\_MODE\_LED, LOW);

digitalWrite(DELETE\_MODE\_LED, LOW);

Serial.begin(9600);

clearinterrupts();

attachInterrupt(0, DATA0, RISING);

attachInterrupt(1, DATA1, RISING);

delay(10);

// Serial.println("READER\_0001");

Serial.println("Trying to set-up SD card");

if(!SD.begin(4)){

Serial.println("Failed to set-up SD card");

noSD = true;

return;

}

else{

Serial.println("Successfully set-up SD card");

}

bitCount = 0;

noSD = false;

readFromSD(IDarray, &lastNum);

Serial.print("lastNum = ");

Serial.println(lastNum);

printArray(lastNum);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN LOOP!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loop(){

// if we have read in all bits we need from a UW card

if(bitCount==35){

timeKeeper = 0;

if(readCard!=0){

Serial.print("readCard = ");

Serial.println(readCard);

}

timeKeeper = 0;

// if we are adding this card

if ( programMode == true ) {

insertID(readCard, &lastNum);

printArray(lastNum);

} // if we are deleting this card

else if ( deleteMode == true ) {

removeID(readCard, &lastNum,validID(readCard,lastNum));

printArray(lastNum);

} // check if this is a valid card to open the door

else {

if ( validID(readCard, lastNum) >= 0 ) {

openDoor();

}

}

//clear the buffers.

bitCount = 0;

readCard = 0;

}

//keypad logic

readChar = scanKeypad();

if(readChar != 'r' && readChar != 'n') {

timeKeeper = 0;

if (codeLength <= 3){

codeSequence[codeLength] = readChar;

codeLength++;

}

else{

for(int i = 0; i < codeLength; i++){

codeSequence[i] = 0;

}

codeLength = 0;

codeSequence[codeLength] = readChar;

codeLength++;

}

if(codeLength == 4){

if(charCompare(codeSequence, programModeCode) && !deleteMode){

programMode = 1-programMode;

Serial.print("programMode = ");

Serial.println(programMode);

}

else if(charCompare(codeSequence, deleteModeCode) && !programMode){

deleteMode = 1-deleteMode;

Serial.print("deleteMode = ");

Serial.println((int)deleteMode);

}

}

Serial.print("keycode: ");

Serial.println(codeSequence);

}

//can send commands from serial console!

if (Serial.available() > 0) {

timeKeeper = 0;

// read the incoming byte:

incomingByte = Serial.read();

// say what you got:

//Serial.print("I received: ");

//Serial.println(incomingByte);

switch (incomingByte) {

case 'o':

openDoor();

break;

case 'p':

if(programMode == false && deleteMode == false){

Serial.println("programMode engaged");

programMode = true;

}

else{

Serial.println("programMode disengaged");

programMode = false;

}

break;

case 'd':

if(deleteMode == false && programMode == false){

Serial.println("deleteMode engaged");

deleteMode = true;

}

else{

Serial.println("deleteMode disengaged");

deleteMode = false;

}

break;

}

}

//resets all the buffers if there are no inputs for awhile

// this allows for correct functionality given a card that does not have 35 bits

if(timeKeeper < 1023){

timeKeeper++;

}

else {

timeKeeper = 0;

Serial.println("Timed Out. Resetting all the buffers.");

clearAllBuffers();

}

// this is SD card present checking. If a sd card is inserted then it will load

// the current ids onto it

if (noSD == true && SD.begin(4)) {

noSD = false;

writeToSD(lastNum);

}

if (!SD.begin(4)) {

noSD = true;

}

//for status LEDS

digitalWrite(PROGRAM\_MODE\_LED, programMode);

digitalWrite(DELETE\_MODE\_LED, (boolean)deleteMode);

delay(10);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*end of main loop, start of functions\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void clearAllBuffers(){

//for the keypad

codeLength = 0;

//for the RFID Reader

readCard = 0;

bitCount = 0;

//to return to normal mode

programMode = false;

deleteMode = false;

}

boolean charCompare(char firstChar[], char secondChar[]){

String firstString(firstChar);

String secondString(secondChar);

if(firstString == secondString){

return true;

}

return false;

}

// prints the contents of the id array

void printArray (int last){

for(int i=0;i<=last;i++){

Serial.print("IDarray: ");

Serial.print(i);

Serial.print(" = ");

Serial.println(IDarray[i]);

}

}

// this opens the door

void openDoor(){

digitalWrite(8, HIGH);

Serial.println("door has opened!!!");

delay(5000);

digitalWrite(8, LOW);

}

// inserts an id into to id array at the last position

// it updates the last number that is passed in

void insertID(unsigned long ID, int\* last) {

if (\*last > MAX\_NUM\_IDS - 1){ // if we have too many, dont add

return;

}

int valid = validID(ID,\*last);

if (validID(ID,\*last)>=0){

return;

}

\*last = \*last + 1;

Serial.print("last= ");

Serial.println(\*last);

IDarray[\*last]=ID;

writeToSD(\*last);

}

// checks if this is a valid id

// returns the position in the array

int validID(unsigned long ID, int last){

for (int i = 0; i <= last; i++){

if (ID==IDarray[i]) return i;

}

return -1;

}

//removes the id at the given position, updates the location of the last position

void removeID(unsigned long ID,int \*last,int pos){

if (pos<0) return;

for (int i =pos;i<\*last;i++){

IDarray[i]=IDarray[i+1];

}

(\*last)--;

writeToSD(\*last);

}

// overwrites all id's to a file

void writeToSD(int lastNum) {

File toWrite;

if (SD.exists("IDFile.txt")) {

SD.remove("IDFile.txt");

}

toWrite = SD.open("IDFile.txt",FILE\_WRITE);

for (int i =0; i <= lastNum; i++) {

toWrite.println(String(IDarray[i]));

}

toWrite.close();

}

// read all ids on a file

void readFromSD(long unsigned int IDArray[],int\* total) {

if (!SD.exists("IDFile.txt")) {

return;

}

int lastNum = \*total;

int count = 0;

IDArray[0] = 0;

char value;

File toRead = SD.open("IDFile.txt",FILE\_READ);

if(toRead){

while (toRead.available()) {

value = toRead.read();

if (((int)value >= 48) && ((int)value <= 57)) { //48 is ascii for 0, 57 is ascii for 9

int add = (int)value - 48;

IDArray[count] = IDArray[count] \* 10 + add;

}

if ((int)value == 10) { //10 is the ascii newline character

count++;

IDArray[count] = 0;

}

}

}

//clears the rest of the array

for (int i = count; i < lastNum; i++) {

IDArray[i] = 0;

}

\*total = count - 1; //because we don't care about the last return in the IDFile.

toRead.close();

}

//returns either the key being pressed, r for repeat, or n for no return

char scanKeypad(){

char returnChar = 'n'; //n for no return

digitalWrite(COLUMN\_ONE, HIGH);

if(analogRead(ROW\_ONE) > 1000){

returnChar = '1';

}

else if(analogRead(ROW\_TWO) > 1000){

returnChar = '4';

}

else if(analogRead(ROW\_THREE) > 1000){

returnChar = '7';

}

else if(analogRead(ROW\_FOUR) > 1000){

returnChar = '\*';

}

digitalWrite(COLUMN\_ONE, LOW);

digitalWrite(COLUMN\_TWO, HIGH);

if(analogRead(ROW\_ONE) > 1000){

returnChar = '2';

}

else if(analogRead(ROW\_TWO) > 1000){

returnChar = '5';

}

else if(analogRead(ROW\_THREE) > 1000){

returnChar = '8';

}

else if(analogRead(ROW\_FOUR) > 1000){

returnChar = '0';

}

digitalWrite(COLUMN\_TWO, LOW);

digitalWrite(COLUMN\_THREE, HIGH);

if(analogRead(ROW\_ONE) > 1000){

returnChar = '3';

}

else if(analogRead(ROW\_TWO) > 1000){

returnChar = '6';

}

else if(analogRead(ROW\_THREE) > 1000){

returnChar = '9';

}

else if(analogRead(ROW\_FOUR) > 1000){

returnChar = '#';

}

digitalWrite(COLUMN\_THREE, LOW);

if (returnChar == prevChar) {

return 'r'; //r for repeat

}

else {

prevChar = returnChar;

return returnChar;

}

}

// clears all interrupts

void clearinterrupts () {

// the interrupt in the Atmel processor mises out the first negitave pulse as the inputs are already high,

// so this gives a pulse to each reader input line to get the interrupts working properly.

// Then clear out the reader variables.

// The readers are open collector sitting normally at a one so this is OK

for(int i = 2; i<4; i++){

pinMode(i, OUTPUT);

digitalWrite(i, HIGH); // enable internal pull up causing a one

digitalWrite(i, LOW); // disable internal pull up causing zero and thus an interrupt

pinMode(i, INPUT);

digitalWrite(i, HIGH); // enable internal pull up

}

delay(10);

}