```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

#include <Servo.h>
Servo servo;
int servoPosition = 0;
int servoPin = 6;

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif
#define PIN 7
#define NUM_OF_PIXELS    4
Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_OF_PIXELS, PIN, NEO_GRB + NEO_KHZ800);

int previousPinReedSwitch = 0 ; // used to serial print open or closed
const int pinReedSwitch = 4;
const int pinLed    = 5;

bool chestClosed() {
  return digitalRead(pinReedSwitch) == HIGH;
}

#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10 // SDA
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
String cardId;

unsigned long standbyLastMillis = 0;
boolean fadeDirection = true;
int color = 0;

byte action;

#define ACTION_STANDBY 0
#define ACTION_SCANNING 1
#define ACTION_WAITINGFORNEXTSCAN 2
#define ACTION_WRONGSEQUENCE 3
#define ACTION_OPENING 4
#define ACTION_OPEN 5

#define MESSAGE_WELCOME   "Toon je sleutel!"
#define MESSAGE_WELCOME2  "en open de kist "
#define MESSAGE_CORRECT   "Goedzo,         "
#define MESSAGE_SCANNEXT  "Volgende sleutel"
#define MESSAGE_OPENING   "Bijna open...   "
#define MESSAGE_WRONG     "Sleutel is fout "
#define MESSAGE_TRYAGAIN  "probeer opnieuw "
#define MESSAGE_TIMEOUT   "Te lang gewacht!"
#define MESSAGE_OPEN      "Het is gelukt   "
#define MESSAGE_OPEN2     "gefeliciteerd!  "
#define MESSAGE_CHESTOPEN "De kist is open "
```

```
#define MESSAGE_EMPTY     "                "

String cardids[] = {"9083ee7b", "50e34527c", "1561ac2e", "07f3daad"};
String masterid = "fc29fa40";
int CARD_COUNT = 4;
int lastCardScanned = 0;
String lastCardIdScanned;
unsigned long CardScanLastMillis = 0;

void standbymodusLights() {
  if (millis() - standbyLastMillis > 10) {
    standbyLastMillis = millis();

    if (fadeDirection == true) {
      color = color + 1;
      if (color >= 60) {
        fadeDirection = false;
      }
    } else {
      color = color - 1;
      if (color <= 10) {
        fadeDirection = true;
      }
    }

    for (int i = 0; i < strip.numPixels(); i++) {
      strip.setPixelColor(i, strip.Color(color, color, color));
    }
    strip.show();
  }
}

void readCard() {
  if (mfrc522.PICC_IsNewCardPresent()) {
    if (mfrc522.PICC_ReadCardSerial()) {
      cardId = "";
      for (byte i = 0; i < mfrc522.uid.size; i++) {
        cardId += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
        cardId += String(mfrc522.uid.uidByte[i], HEX);
      }

      if (lastCardIdScanned != cardId) {
        lastCardIdScanned = cardId;
        if (lastCardIdScanned == masterid) {
          action = ACTION_OPENING;
        } else if (lastCardIdScanned == cardids[lastCardScanned]) {
          lastCardScanned++;
          if (lastCardScanned >= CARD_COUNT) { // all cards are scanned
            action = ACTION_OPENING;
            lcd.setCursor(0, 0);
            lcd.print(MESSAGE_OPENING);
            lcd.setCursor(0, 1);
            lcd.print(MESSAGE_EMPTY);
          } else { // correct card in this sequence
            action = ACTION_WAITINGFORNEXTSCAN;
            lcd.setCursor(0, 0);
```

```
          lcd.print(MESSAGE_CORRECT);
          lcd.setCursor(0, 1);
          lcd.print(MESSAGE_SCANNEXT);
          CardScanLastMillis = millis();
         }
        } else { // wrong card in this sequence
         lastCardScanned = 0;
         lastCardIdScanned = "";
         action = ACTION_WRONGSEQUENCE;
         lcd.setCursor(0, 0);
         lcd.print(MESSAGE_WRONG);
         lcd.setCursor(0, 1);
         lcd.print(MESSAGE_TRYAGAIN);
        }
       }
     }
   }
}

void statusLights() {
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, strip.Color(200, 0, 0));
  }
  strip.show();
  for (int i = 0; i < lastCardScanned; i++) {
    strip.setPixelColor(i, strip.Color(0, 200, 0));
  }
  strip.show();
}

void almostThereLights() {
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, strip.Color(0, 200, 0));
  }
  strip.show();
}

void wrongLights() {
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, strip.Color(200, 0, 0));
  }
  strip.show();
}

void theaterChaseRainbow(uint8_t wait) {
  for (int j=0; j < 256; j++) {
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {
        strip.setPixelColor(i+q, Wheel( (i+j) % 255));
      }
      strip.show();

      digitalWrite(pinLed, !digitalRead(pinReedSwitch));
      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {
```

```
        strip.setPixelColor(i+q, 0);

      }
    }
  }

        // Hack voor sluiten
      if (chestClosed()) {
        return;
       }

}

uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}

void startHunt() {
  digitalWrite(pinLed, LOW);
  closeChest();
  action = ACTION_STANDBY;
}

void setup() {
  Serial.begin(9600);

  pinMode(pinLed, OUTPUT);
  pinMode(pinReedSwitch, INPUT);

  SPI.begin();

  mfrc522.PCD_Init();

  lcd.init();
  lcd.backlight();

  strip.begin();
  strip.show();

  startHunt();
}

void closeChest() {
  lcd.setCursor(0,0);
  lcd.print(MESSAGE_WELCOME);
  lcd.setCursor(0,1);
  lcd.print(MESSAGE_WELCOME2);
```

```
  closeLock();
}

void closeLock() {
  servo.attach(servoPin);
  servo.write(0);
  delay(1500);
  servo.detach();
  servoPosition = 0;
}

void openLock() {
  servo.attach(servoPin);
  servo.write(50);
  delay(1500);
  servo.detach();
  servoPosition = 50;
}

void loop() {
  digitalWrite(pinLed, LOW);

  if(digitalRead(pinReedSwitch) != previousPinReedSwitch){
       if(digitalRead(pinReedSwitch) == 1) {
          Serial.println("Closed");
       }
       else if (digitalRead(pinReedSwitch) == 0) {
        Serial.println("Open");
       }
    previousPinReedSwitch = digitalRead(pinReedSwitch);
  }

  switch (action) {
    case ACTION_STANDBY:
      standbymodusLights();
      readCard();
    break;

    case ACTION_SCANNING:
      standbymodusLights();
    break;

    case ACTION_WAITINGFORNEXTSCAN:
      readCard();
      statusLights();
      if (millis() - CardScanLastMillis > 5000) {
        lcd.setCursor(0, 0);
        lcd.print(MESSAGE_TIMEOUT);
        lcd.setCursor(0, 1);
        lcd.print(MESSAGE_TRYAGAIN);
        action = ACTION_STANDBY;
        lastCardScanned = 0;
        lastCardIdScanned = "";
      }
    break;
```

```
    case ACTION_WRONGSEQUENCE:
      wrongLights();
      delay(1000);
      action = ACTION_STANDBY;
    break;

    case ACTION_OPENING:
      almostThereLights();
      delay(2000);
      lcd.setCursor(0, 0);
      lcd.print(MESSAGE_OPEN);
      lcd.setCursor(0, 1);
      lcd.print(MESSAGE_OPEN2);
      openLock();
      action = ACTION_OPEN;
    break;

    case ACTION_OPEN:
      theaterChaseRainbow(50);
      if(chestClosed()) {
        startHunt();
      }
      break;
  }
}
```