```
001 /*
002   Гр. 050541, Сёмин Даниил Николаевич
003   Проект: Микропроцессорное устройство идентификации проездных билетов
004 */
005 // Подключение библиотек
006 #include <SPI.h>
007 #include <MFRC522.h>
008 #include <Int64String.h>
009 #include "U8glib.h"
010 #include <EEPROM.h>  // Библиотека EEPROM для хранения ключей
011
012 U8GLIB_SH1106_128X64 u8g(U8G_I2C_OPT_NONE);  // I2C / TWI
013
014 // константы подключения контактов SS и RST
015 #define RST_PIN 9
016 #define SS_PIN 10
017 MFRC522 mfrc522(SS_PIN, RST_PIN);  // Инициализация MFRC522
018 MFRC522::MIFARE_Key key;           // Объект ключа
019 MFRC522::StatusCode status;        //Объект статуса
020 #define MAX_TAGS 3     // Максимальное количество хранимых меток - ключей
021 #define BUZZER_PIN 6      // Пин баззера
022 #define RED_LED_PIN 2     // Пин красного светодиода
023 #define GREEN_LED_PIN 3   // Пин зеленого светодиода
024 #define YELLOW_LED_PIN 4  // Пин желтого светодиода
025 #define BTN_PIN 8         // Пин кнопки
026 #define BTN_PIN_2 7       // Пин кнопки 2
027 #define BTN_PIN_3 5       // Пин кнопки 3
028 #define EE_START_ADDR 5   // Начальный адрес в EEPROM
029 #define EE_KEY 100        // Ключ EEPROM, для проверки на первое вкл.
030
031 #define DECLINE 0  // Отказ
032 #define SUCCESS 1  // Успешно
033 #define SAVED 2    // Новая метка записана
034 #define DELITED 3  // Метка удалена
035
036 //переменные хранящие состаяние кнопки
037 boolean lastButton = LOW;
038 boolean lastButton_2 = LOW;
039 boolean lastButton_3 = LOW;
040 boolean currentButton = LOW;
041 boolean ledOn = false;
042
043 uint8_t savedTags = 0;  // кол-во записанных меток
044
045 //для хранения номера метки в десятичном формате
046 uint32_t uidDec, uidDecTemp;
047 //для хранения записанных меток
048 short int f_find = 0;  //результат нахождения метки в памяти
049 uint32_t start = 0;
050 bool needClear = 0;
051
052 void setup() {
053   // Настраиваем пины
054   pinMode(BTN_PIN, INPUT);
055   pinMode(BUZZER_PIN, OUTPUT);
056   pinMode(RED_LED_PIN, OUTPUT);
057   pinMode(GREEN_LED_PIN, OUTPUT);
058   pinMode(YELLOW_LED_PIN, OUTPUT);
059
060   for (int i = 0; i < 3; i++) {
061     digitalWrite(RED_LED_PIN, HIGH);
062     delay(125);
063     digitalWrite(RED_LED_PIN, LOW);
064     delay(125);
065     digitalWrite(GREEN_LED_PIN, HIGH);
```

```
066     delay(125);
067     digitalWrite(GREEN_LED_PIN, LOW);
068     delay(125);
069     digitalWrite(YELLOW_LED_PIN, HIGH);
070     delay(125);
071     digitalWrite(YELLOW_LED_PIN, LOW);
072     delay(125);
073   }
074   Serial.begin(9600);          // инициализация последовательного порта
075   SPI.begin();                    // инициализация SPI
076   mfrc522.PCD_Init();             // инициализация MFRC522
077   for (byte i = 0; i < 6; i++) {  //Наполняем ключ
078     key.keyByte[i] = 0xff;        //Ключ по умолчанию 0xffffffffffff
079   }
080
081   // Полная очистка при включении при зажатой кнопке
082   start = millis(); //Отслеживание длительного удержания кнопки после включения
083   needClear = 0;          // Чистим флаг на стирание
084   // Инициализация EEPROM
085   if (needClear or EEPROM.read(EE_START_ADDR) != EE_KEY) {
086     for (uint16_t i = 0; i < EEPROM.length(); i++) EEPROM.write(i, 0x00);
087     EEPROM.write(EE_START_ADDR, EE_KEY);   // Пишем байт-ключ
088   } else {                                 // Обычное включение
089     savedTags = EEPROM.read(EE_START_ADDR + 1); //Читаем кол-во меток в памяти
090   }
091   digitalWrite(YELLOW_LED_PIN, LOW);
092 }
093
094 void loop() {
095     if (debounce(lastButton_3, BTN_PIN_3)) {
096       needClear = true;   // Ставим флаг стирания при нажатии кнопки 3
097       indicate(DELITED);             // Подаем сигнал удаления
098     }
099   if (needClear) {                   // при необходимости очистки ключей
100     for (uint16_t i = 0; i < EEPROM.length(); i++) EEPROM.write(i, 0x00);
101     EEPROM.write(EE_START_ADDR, EE_KEY);   // Пишем байт-ключ
102     needClear = false;
103   }
104   if (debounce(lastButton_2, BTN_PIN_2)){
105     savedTags = EEPROM.read(EE_START_ADDR + 1); //Читаем кол-во меток в памяти
106     Serial.println(savedTags);
107     PrintOnDisplay("TAGS in MEMARY", savedTags);
108   }
109
110   //выбор режима
111   currentButton = debounce(lastButton, BTN_PIN);
112   if (lastButton == LOW && currentButton == HIGH) {
113     ledOn = !ledOn;
114   }
115   lastButton = currentButton;
116   digitalWrite(YELLOW_LED_PIN, ledOn);
117
118   if (!mfrc522.PICC_IsNewCardPresent()) //если метка не поднесена-вернуться в l-p
119       return;
120     if (!mfrc522.PICC_ReadCardSerial()) //если метка не читается-вернуться в loop
121       return;
122
123    //получение доступа ко 2-му сектору
124     status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 7,
&key, &(mfrc522.uid));
125     if (status != MFRC522::STATUS_OK) {
126       Serial.println("\nErr доступа ко 2-му сектору");  // Выводим ошибку
127     }
128     // показать результат чтения UID и тип метки
```

```
129     Serial.println("  ");
130     Serial.print(F("Card UID:"));
131     buffer_HEX_print(mfrc522.uid.uidByte, mfrc522.uid.size);
132     Serial.println(" ");
133     uidDec = UidDecF(uidDec, uidDecTemp);  //получаем UID в десятичной сс
134     Serial.println(uidDec);                // Выводим UID метки в консоль
135     Serial.print(F("PICC type: "));
136     byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
137     Serial.println(mfrc522.PICC_GetTypeName(piccType));
138
139     //получение доступа ко 2-му сектору
140     status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 7,
&key, &(mfrc522.uid));
141     if (status != MFRC522::STATUS_OK) {                 // Если не окэй
142       Serial.println("\nErr доступа ко 2-му сектору");  // Выводим ошибку
143     }
144   //первый режим работы устройства
145   if (ledOn == false) {
146     if (foundTag(mfrc522.uid.uidByte, mfrc522.uid.size) >= 0) {
147       indicate(SUCCESS);   // Если нашли - подаем сигнал успеха
148       delay(1000);
149       f_find = 1;
150     } else {              // Метка не найдена
151       indicate(DECLINE);  // Выдаем отказ
152       delay(1000);
153       f_find = 0;
154     }
155     switch (f_find) {
156       case 1:
157         {
158           /* Чтение блока, указываем блок данных #4 */
159           uint8_t dataBlock[18];                    // Буфер
160           uint8_t size = sizeof(dataBlock);         // Размер буфера
161           status = mfrc522.MIFARE_Read(4, dataBlock, &size); // чит. б 4
162           if (status != MFRC522::STATUS_OK) {
163             Serial.println("Err чиения 4-го блока");
164             break;
165           }
166           /* Выводим счетчик метки */
167           if (dataBlock[0] > 0) {
168             dataBlock[0]--;
169             Serial.print("Осталось: ");
170             Serial.println(dataBlock[0]);
171             PrintOnDisplay("UID FOUDED:", dataBlock[0], uidDec);
172           } else {
173             Serial.print("Осталось: ");
174             Serial.println(dataBlock[0]);
175             PrintOnDisplay("UID FOUDED:", dataBlock[0], uidDec);
176             for (uint8_t i = 0; i < 2; i++) {
177               tone(BUZZER_PIN, 500);
178               delay(300);
179               noTone(BUZZER_PIN);
180               delay(100);
181             }
182           }
183           /* Запись блока, указываем блок данных #4 */
184           status = mfrc522.MIFARE_Write(4, dataBlock, 16); //Пишем массив
185           if (status != MFRC522::STATUS_OK) {
186             Serial.println("Err записи 4-го блока");
187             break;
188           }
189           break;
190         }
191       case 0:
```

```
192         {
193           Serial.println("UID not found");
194           PrintOnDisplay("UID NOT FOUDED", uidDec);
195           break;
196         }
197       default:
198         {
199           Serial.println("Err!");
200           PrintOnDisplay("ERROR", uidDec);
201           break;
202         }
203     }
204     f_find = 0;
205     //2й режим работы устройства
206   } else {
207     saveOrDeleteTag(mfrc522.uid.uidByte, mfrc522.uid.size, uidDec);  //
208   }
209   mfrc522.PICC_HaltA();  // Завершаем работу с меткой
210   mfrc522.PCD_StopCrypto1();
211 }
212
213 // Вывод результата чтения данных в HEX-виде
214 void buffer_HEX_print(byte *buffer, byte bufferSize) {
215   for (byte i = 0; i < bufferSize; i++) {
216     Serial.print(buffer[i] < 0x10 ? " 0" : " ");
217     Serial.print(buffer[i], HEX);
218   }
219 }
220
221 //функци сглаживания дребезга кнопки
222 boolean debounce(boolean last, int BTN_PIN_x) {
223   boolean current = digitalRead(BTN_PIN_x);
224   if (last != current) {
225     delay(10);
226     current = digitalRead(BTN_PIN_x);
227     return current;
228   }
229 }
230
231 // Устанавливаем состояние светодиодов
232 void ledSetup(bool state) {
233   if (state) {  // Зеленый
234     digitalWrite(GREEN_LED_PIN, HIGH);
235     digitalWrite(RED_LED_PIN, LOW);
236   } else {  // Красный
237     digitalWrite(GREEN_LED_PIN, LOW);
238     digitalWrite(RED_LED_PIN, HIGH);
239   }
240 }
241
242 // Звуковой сигнал + лед
243 void indicate(uint8_t signal) {
244   ledSetup(signal);  // Лед
245   switch (signal) {  // Выбираем сигнал
246     case DECLINE:
247       Serial.println("DECLINE");
248       for (uint8_t i = 0; i < 2; i++) {
249         tone(BUZZER_PIN, 100);
250         delay(300);
251         noTone(BUZZER_PIN);
252         delay(100);
253       }
254       return;
255     case SUCCESS:
```

```
256       Serial.println("SUCCESS");
257       tone(BUZZER_PIN, 890);
258       delay(330);
259       noTone(BUZZER_PIN);
260       return;
261     case SAVED:
262       Serial.println("SAVED");
263       for (uint8_t i = 0; i < 2; i++) {
264         tone(BUZZER_PIN, 890);
265         delay(330);
266         noTone(BUZZER_PIN);
267         delay(100);
268       }
269       return;
270     case DELITED:
271       Serial.println("DELITED");
272       for (uint8_t i = 0; i < 3; i++) {
273         tone(BUZZER_PIN, 890);
274         delay(330);
275         noTone(BUZZER_PIN);
276         delay(100);
277       }
278       return;
279   }
280 }
281
282 // Сравнение двух массивов известного размера
283 bool compareUIDs(uint8_t *in1, uint8_t *in2, uint8_t size) {
284   for (uint8_t i = 0; i < size; i++) { //Проходим по всем элементам
285     if (in1[i] != in2[i]) return false;//Если хоть один не сошелся-массивы не совп
286   }
287   return true;  // Все сошлись - массивы идентичны
288 }
289
290 // Поиск метки в EEPROM
291 int16_t foundTag(uint8_t *tag, uint8_t size) {
292   uint8_t buf[8];                               // Буфер метки
293   uint16_t address;                             // Адрес
294   for (uint8_t i = 0; i < savedTags; i++) {   // проходим по всем меткам
295     address = (i * 8) + EE_START_ADDR + 2; // Считаем адрес текущей метки
296     EEPROM.get(address, buf);                   // Читаем метку из памяти
297     if (compareUIDs(tag, buf, size)) return address;  // Сравниваем
298   }
299   return -1;  // Если не нашли - вернем минус 1
300 }
301
302 // Удаление или запись новой метки
303 void saveOrDeleteTag(uint8_t *tag, uint8_t size, uint32_t *decUid) {
304   int16_t tagAddr = foundTag(tag, size);        // Ищем метку в базе
305   uint16_t newTagAddr = (savedTags * 8) + EE_START_ADDR + 2;//Адрес кр метки
306   if (tagAddr >= 0) {                    // Если метка найдена - стираем
307     for (uint8_t i = 0; i < 8; i++) {
308       EEPROM.write(tagAddr + i, 0x00);  // Стираем байт старой метки
309       EEPROM.write(tagAddr + i, EEPROM.read((newTagAddr - 8) + i)); //На
     ее место пишем байт последней метки
310       EEPROM.write((newTagAddr - 8) + i, 0x00); //Удаляем байт последней метки
311     }
312     EEPROM.write(EE_START_ADDR + 1, --savedTags); //Уменьшаем кол-во меток
313     PrintOnDisplay("DELITED", decUid);
314     indicate(DELITED);
315   } else if (savedTags<MAX_TAGS){//метка не найдена-нужно записать, лимит не достигнут
316     for (uint16_t i = 0; i < size; i++) EEPROM.write(i + newTagAddr,
      tag[i]);  // Зная адрес пишем новую метку
317     EEPROM.write(EE_START_ADDR + 1, ++savedTags);// Увеличиваем кол-во меток
```

```
318     PrintOnDisplay("SAVED", decUid);
319     indicate(SAVED);  // Подаем сигнал
320   } else {            // лимит меток при попытке записи новой
321     PrintOnDisplay("DECLINE", decUid);
322     indicate(DECLINE);  // Выдаем отказ
323     ledSetup(DECLINE);
324   }
325 }
326
327 // вывод на экран
328 void PrintOnDisplay(char *text1, uint32_t DecUID) {
329   u8g.firstPage();
330   do {
331     u8g.setRot180();
332     u8g.setFont(u8g_font_unifont);
333     u8g.setPrintPos(1, 20);
334     u8g.print(DecUID);
335     u8g.setPrintPos(1, 45);
336     u8g.print(text1);
337   } while (u8g.nextPage());
338 }
339 void PrintOnDisplay(char *text1, uint8_t dataBlock, uint32_t DecUID) {
340   u8g.firstPage();
341   do {
342     u8g.setRot180();
343     u8g.setFont(u8g_font_unifont);
344     u8g.setPrintPos(1, 20);
345     u8g.print(DecUID);
346     u8g.setPrintPos(10, 40);
347     u8g.print(text1);
348     u8g.setPrintPos(45, 40);
349     u8g.print(dataBlock);
350   } while (u8g.nextPage());
351 }
352
353 //формирование UID в десятичной сс
354 uint32_t UidDecF(uint32_t decUid, uint32_t decUidTemp) {
355   decUid = 0;
356   for (byte i = 0; i < mfrc522.uid.size; i++) {
357     decUidTemp = mfrc522.uid.uidByte[i];
358     decUid = decUid * 256 + decUidTemp;
359   }
360   return decUid;
361 }
```