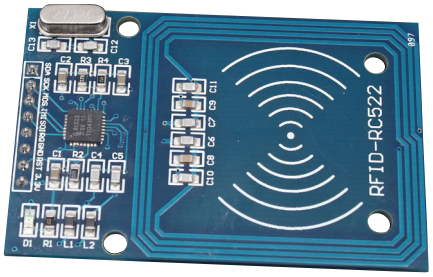
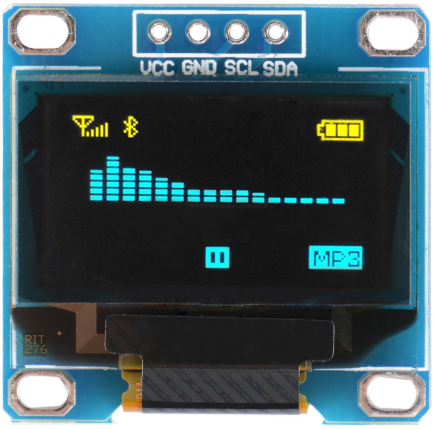
Dodatkowe informacje:
- Oryginalny opis kodu można znaleźć pod adresem ***http://bit.ly/2o6d1e2***.
- Projekt jest omówiony po angielsku, również w materiale wideo, dostępnym pod adresem ***http://bit.ly/2MSZjt3***.

Arduino z RFID



Znaczniki RFID to bardzo wygodna technologia do zbliżeniowego sterowania urządzeniami elektronicznymi. Świetnie sprawdzają się w systemach kontroli dostępu, ale mogą mieć i inne zastosowania. Korzystając z gotowych modułów, można ją z łatwością zaimplementować przy użyciu Arduino. W artykule opisano łatwy do wykonanie projekt, który demonstruje, jak połączyć ze sobą Arduino, moduł RFID i miniaturowy wyświetlacz OLED.



Na wstępie warto wyjaśnić, że pokazywany projekt daleki jest od kompletnego urządzenia. Nie ma obudowy i został wykonany na płytce prototypowej. Jego autor, Nick Koumaris ze Sparty, przygotował go w celach edukacyjnych, by pokazać jak łatwo jest wykorzystać RFID w praktyce. I faktycznie, zarówno od strony programowej, jak i z punktu widzenia połączeń, wdrożenie obsługi znaczników i wyświetlacza nie powinno nastręczać trudności.

Samych znaczników nie będziemy omawiać w niniejszym artykule. Autor w swoim projekcie wykorzystał moduły pracujące ze standardem MIFARE w paśmie 13,56 MHz i koncentruje się jedynie na odczycie unikalnego ID znaczników. W wersji rozszerzonej projektu można pokusić się o zapis i odczyt innych danych do pamięci tagu.

Użyte komponenty

Projekt został zbudowany z użyciem:

1. płytki Arduino Uno,
2. czytnika RFID, opartego o układ RC522,
3. wyświetlacza OLED o przekątnej 0,96”,
4. małej płytki prototypowej,
5. kilku przewodów.

Projekt, szczególnie jeśli wykonywany byłby z użyciem połączeń lutowanych i obudowy, można doposażyć o powerbank.

Autor zadowolił się komponentami dalekowschodnimi. Użył klonu Arduino i wyświetlacza, markowanych logiem firmy Geekcreit. Zaletą tego typu podzespołów jest cena, ale wadą – brak jakiejkolwiek gwarancji, że spełniają jakieś normy. Zazwyczaj dobrze sprawdzają się w warunkach domowych i w niezbyt złożonych projektach, ale potrafią sprawiać problemy w specyficznych kombinacjach. Nie sugerujemy też stawiać na nich swojej reputacji, zanim nie sprawdzi się samodzielnie, że dany dostawca trzyma jakość produkcji na stałym poziomie, jeśli chce się wykonać gotowy produkt, bazujący na tego typu podzespołach. Warto też się upewnić, że korzystanie z danego modułu nie wiąże się z łamaniem praw firm trzecich w danym regionie. Chińskie przepisy nie są w tym zakresie bardzo restrykcyjne, co oznacza że tamtejsi producenci często nie przejmują się takimi „błahostkami”. Tak czy inaczej, łączny koszt użytych podzespołów wyniósł niecałe 60 zł.

Zastosowany czytnik jest w stanie komunikować się ze znacznikami z odległości około 2 cm. Przysłonięcie go obudową z tworzywa sztucznego może nieznacznie zmniejszyć ten dystans, ale w praktyce konieczność większego przybliżenia znacznika do czytnika będzie wynikała z samej grubości obudowy. Standard MIFARE korzysta ze znaczników pasywnych, które zasilane są w trakcie odczytu, bezprzewodowo, za pomocą czytnika.

Użyty wyświetlacz jest dwukolorowy, przy czym jego górna sekcja może świecić na żółto, a reszta wyświetlacza na niebiesko. Rozdzielczość ekranu to 128×64 piksele. Można go z łatwością nabyć w wersji w pełni monochromatycznej, a sterowanie jedną czy drugą odmianą absolutnie się od siebie nie różni. Pobór prądu wyświetlacza mieści się w zakresie od 10 mA do 20 mA. Moduł ma sterownik, który komunikuje się przez interfejs I²C, a do tego dostępne są dla niego proste w użyciu biblioteki programistyczne. W efekcie, jest bardzo prosty do podłączenia z Arduino. Wystarczą dwa przewody sygnałowe i dwa zasilające.

Podłączenie elementów

Zarówno wyświetlacz, jak i czytnik RFID wymagają napięcia zasilania 3,3 V. Jest ono dostępne na płytce Arduino Uno. Należy przy tym upewnić się, przypadkiem nie podłącza się napięcia 5 V – czytnik nie jest przed nim zabezpieczony i prawdopodobnie ulegnie uszkodzeniu, gdy dostanie zbyt wysoki poziom napięcia.

Wyświetlacz dołączamy w następujący sposób:

- Pin VCC do napięcia 3,3 V Arduino.
- Pin GND do masy Arduino.
- Pin SCL do pinu analogowego 5.
- Pin SDA do pinu analogowego 4.

Czytnik RFID korzysta natomiast z interfejsu SPI, w efekcie czego ryzyko, że jego

Listing 1. Kod źródłowy programu projektu

```
/* Arduino RFID Tutorial v1.02
   Get the latest version of the code here:
   http://educ8s.tv/arduino-rfid-tutorial/  */
#include <MFRC522.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SPI.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#define SS_PIN 10
#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN); // Instancja klasy MFRC522
MFRC522::MIFARE_Key key;

int code[] = {69,141,8,136}; // Zapisany, poprawny Unikalny Identyfikator
int codeRead = 0;
String uidString;
void setup() {
  Serial.begin(9600);
  SPI.begin(); // Inicjalizacja SPI
  rfid.PCD_Init(); // Inicjalizacja MFRC522
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // Inicjlizacja z adresem I2C 0x3D (dla wyświetlacza 128x64)
  // Czyszczeznie bufora
  display.clearDisplay();
  display.display();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(10,0);
  display.print("RFID Lock");
  display.display();
}

void loop() {
  if(rfid.PICC_IsNewCardPresent()) readRFID();
  delay(100);
}

void readRFID()
{
  rfid.PICC_ReadCardSerial();
  Serial.print(F("\nPICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  Serial.println(rfid.PICC_GetTypeName(piccType));
  // Sprawdzenie czy znacznik jest zgodny z Classic MIFARE
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Twoj tag nie jest zgodny z MIFARE Classic."));
    return;
  }
  clearUID();
  Serial.println("Zeskanowany UID:");
  printDec(rfid.uid.uidByte, rfid.uid.size);
  uidString = String(rfid.uid.uidByte[0])+" "+String(rfid.uid.uidByte[1])+" "+String(rfid.uid.uidByte[2])+ " "+String(rfid.uid.uidByte[3]);
  printUID();
  int i = 0;
  boolean match = true;
  while(i<rfid.uid.size)
  {
    if(!(rfid.uid.uidByte[i] == code[i])) match = false;
    i++;
  }

  if(match)
  {
    Serial.println("\nZnacznik rozpoznany!");
    printUnlockMessage();
  }
    else
  {
    Serial.println("\nObcy znacznik");
  }
  // Wstrzymaj PICC
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}

void printDec(byte *buffer, byte bufferSize)
{
  for (byte i = 0; i < bufferSize; i++)
  {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], DEC);
  }
}

void clearUID()
{
  display.setTextColor(BLACK);
  display.setTextSize(1);
  display.setCursor(30,20);
  display.print(uidString);
  display.display();
}

void printUID()
{
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,20);
  display.print("UID: ");
  display.setCursor(30,20);
  display.print(uidString);
  display.display();
}

void printUnlockMessage()
{
  display.display();
```

Listing 1. cd.

```
  display.setTextColor(BLACK);
  display.setTextSize(2);
  display.setCursor(10,0);
  display.print("Zamek RFID");
  display.display();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(10,0);
  display.print("Zamek otwarty");
  display.display();
  delay(2000);
  display.setTextColor(BLACK);
  display.setTextSize(2);
  display.setCursor(10,0);
  display.print("Zamek otwarty");
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(10,0);
  display.print("Zamek RFID");
  display.display();
}
```

ewentualna błędna praca w jakikolwiek sposób zakłóci komunikację z wyświetlaczem jest zminimalizowane:

1. PIN RST czytnika podłączamy do cyfrowego pinu 9 Arduino.
2. MISO do cyfrowego pinu 12.
3. MOSI do cyfrowego pinu 11.
4. SCK do cyfrowego pinu 13.
5. SDA do cyfrowego pinu 10.
6. Wyprowadzenie IRQ pozostawiamy niepodłączone.

Kod programu

Przygotowany program jest nieskomplikowany, ale to dzięki temu, że korzysta z gotowych, dostępnych bezpłatnie bibliotek. Należy je więc najpierw zainstalować. W tym celu wybieramy z menu Sketch → Include Libraries → Manage libraries i wyszukujemy MFRC522 do obsługi czytnika RFID. Dodajemy też biblioteki do obsługi grafiki oraz wyświetlacza: Adafruit_GFX.h i Adafruit_SSD1306.h. W drugim z tych plików należy umieścić znak komentarza przed linią określającą domyślną rozdzielczość wyświetlacza, a usunąć go sprzed linii odpowiedniej dla wyświetlacza użytego w projekcie (tu 128×64). W praktyce ustawienia te powinny znajdować się w linijkach 69 i 70.

W kolejnych linijkach kodu, po załadowaniu bibliotek, dla czytelności kodu zdefiniowano piny oraz stworzono obiekty klas z ładowanych bibliotek. W linijce:

```
Int code[] = {69,141,8,136};
```

Zapisany został unikalny identyfikator użytego znacznika. Zaraz niżej inicjowane są zmienne globalne.

W funkcji inicjalizacyjnej setup() konfigurowane jest połączenie z czytnikiem RFID oraz praca wyświetlacza. Ekran jest czyszczony, a kolor znaków ustawiany jako pozytywny (biały), co oznacza że będą prezentowane przez zapalone piksele. Wskazywany jest rozmiar znaków oraz pozycja kursora. Następnie do wyświetlacza wysyłany jest tekst „RFID Lock" do wyświetlenia.

W głównej pętli programu dokonywana jest próba odczytu karty RFID i jeśli różni się ona od karty poprzednio odczytywanej, to wykonywana jest funkcja obsługująca znacznik. Pętla działa z opóźnieniem 100 ms na cykl.

Po zeskanowaniu znacznika, moduł RFID odpytywany jest o rodzaj zeskanowanego tagu – obsługiwane są tylko tagi MIFARE Classic. Jeśli typ się zgadza, wyświetlacz jest czyszczony, na następnie prezentowany jest na nim unikalny identyfikator zeskanowanej karty. W kolejnym kroku wszystkie bajty znacznika są kolejne porównywane ze wzorcem i jeśli pasują do wzorca, wywoływana jest funkcja wyświetlająca informacje o tym, że znacznik został poprawnie rozpoznany. W przeciwnym wypadku system komunikuje, że znacznik jest nieznany.

Uzupełnieniem kodu są oddzielne funkcje, zawierające wyświetlenie poszczególnych informacji na ekranie za pomocą pojedynczych linijek kodu. Korzystają z tych samych funkcji, jakie zostały użyte na początku do wyświetlenia komunikatu powitalnego, tuż po uruchomieniu projektu.

Podsumowanie

Zaprezentowany projekt to raczej wstęp do bardziej kompletnej aplikacji RFID z Arduino. Pokazuje, jak zacząć, ale nie obejmuje żadnego realnego mechanizmu otwierania zamka, ani wykonywania innych akcji – choćby czysto elektronicznych – adekwatnych do zeskanowanego tagu. Udowadnia, że obsługa znaczników MIFARE (a i innych też) może być bardzo prosta oraz niedroga. Projekt można by było rozbudować o obudowę lub wysyłanie komend do jakiegoś aktuatora, by całość faktycznie reprezentowała zamek.

Programiści o silniejszych zapędach w kierunku optymalizacji kodu zwrócą też pewnie uwagę, że pętle porównującą kolejne bajty unikalnego identyfikatora można byłoby przerywać w momencie wykrycia pierwszej niezgodności. A w wypadku, gdyby praktyka pokazywała, że nabyty zestaw znaczników różni się tylko ostatnimi bajtami, porównywanie można by było rozpocząć od końca.

Marcin Karbowniczek, EP