

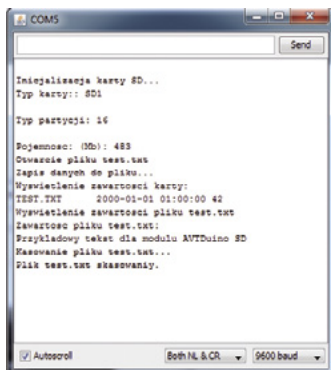
Obsługa modułu AVTDuino SD

W listopadowym numerze *Elektroniki Praktycznej* opisaliśmy moduł AVTDuino SD (AVT1649). Umożliwia on dołączenie karty pamięci SD do zestawu Arduino i zastosowanie jej do zapamiętywania danych. W artykule opisano programową obsługę modułu AVTDuino SD dla systemu Arduino UNO. Taka karta przyda się zwłaszcza w urządzeniach multimedialnych lub do akwizycji danych, w których niedroga pamięć o dużej pojemności jest atutem nie do przecenienia.

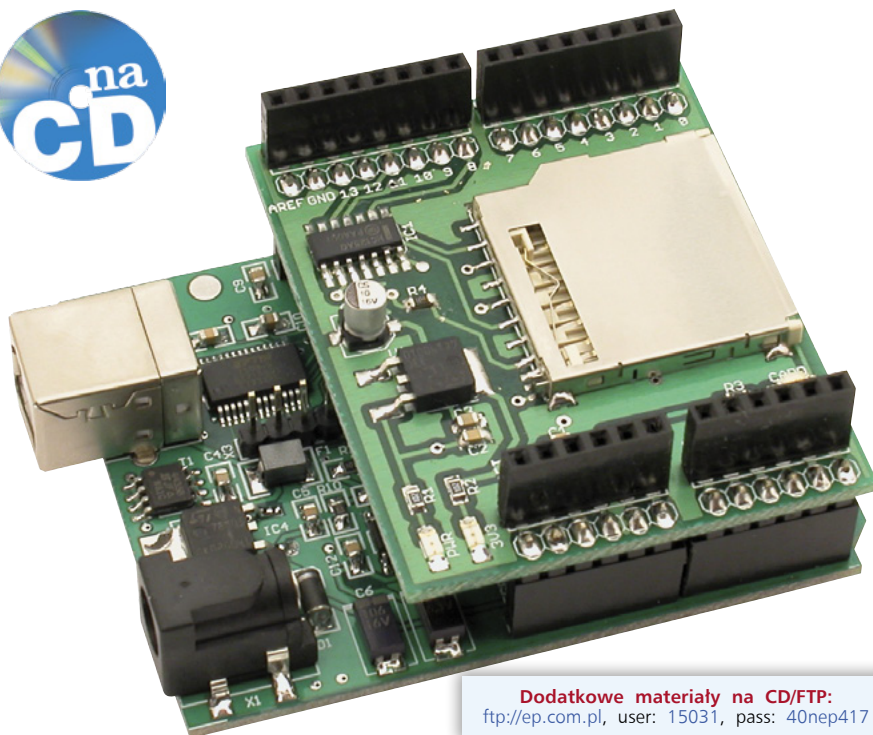
Moduł umożliwia obsługę kart SD jak i SDHC z systemem plików FAT16 lub FAT32. Te systemy plików są kompatybilne ze stosowanymi w komputerach PC, więc kartę zapisaną przez Arduino można odczytywać np. za pomocą Windows i vice versa. Jedynym wymaganiem jest zastosowanie w układzie lub zestawie Arduino UNO mikrokontrolera o dużej pamięci np. ATmega328.

Przykładowy program opisany w artykule odczyta i prześle przez RS232 parametry karty SD i utworzy przykładowy plik tekstowy zawierający pewne dane. Następnie te dane zostaną odczytane i przesłane przez interfejs szeregowy, a utworzony wcześniej plik zostanie usunięty.

Do obsługi karty SD jest przeznaczona biblioteka *SD.h*, w której są dostępne są komendy do obsługi karty SD. Zostały one podzielone na grupy *SD* i *FILE*. Jak łatwo domyślić się, pierwsza grupa komend jest związana z obsługą karty SD, a druga z obsługą systemu plików.



Rysunek 1. Komunikaty pojawiające się w trakcie pracy programu



Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 15031, pass: 40nep417

Wszystkie dostępne komendy do obsługi kart SD można znaleźć w opisie tej biblioteki na stronie Arduino. W przykładowym programie pokazanym na listingu 1 wykorzystano tylko niektóre z pokazanych komend przeznaczonych do obsługi karty SD. Program odczytuje parametry karty SD, tworzy, zapisuje, odczytuje i usuwa plik tekstowy.

W pierwszej kolejności są tworzone zmienne pomocnicze wymagane do obsługi karty SD. W programie, w procedurze konfiguracyjnej, oprócz nastawy prędkości interfejsu RS232 znajduje się komenda *SD.begin(8)* inicjująca kartę SD. Parametrem wywołania jest numer linii *CS* interfejsu karty. Za pomocą instrukcji *if (!card.init(SPI_HALF_SPEED, chipSelect))* jest sprawdzany stan inicjalizacji. Jeśli nie przebiegła ona poprawnie, wyświetlany jest odpowiedni komunikat. W procedurze głównej programu, za pomocą komendy *card.type()*, jest odczytywany typ karty. Funkcja zwraca wartości *SD1*, *SD2*, *SDHC* lub *nieznany*. Są one przesyłane za pomocą interfejsu RS232 do terminala uruchomionego na komputerze PC. Dalej, za pomocą instrukcji *volume.fatType()* jest odczytywany i wysyłany rodzaj systemu plików, dla którego jest sformatowana karta SD. Komendy:

```
volumeSize = volume.  
blocksPerCluster();
```

Grupa komend SD do obsługi karty pamięci

begin() – inicjalizacja karty SD
exists() – sprawdzenie czy istnieje plik lub katalog o podanej nazwie
mkdir() – utworzenie katalogu
open() – utworzenie lub otwarcie pliku
remove() – usunięcie pliku
rmdir() – usunięcie katalogu

Grupa komend FILE do obsługi plików

available() – liczba bajtów dostępnych w pliku
close() – zamknięcie pliku
print() – zapis danych w pliku
println() – zapis danych w pliku ze znakami końca i nowej linii
size() – wielkość pliku
read() – odczyt pojedynczego bajtu danych
write() – zapis pojedynczego bajtu danych

```
volumeSize *= volume.  
clusterCount();  
volumeSize *= 512;  
volumeSize /= 1024;  
volumeSize /= 1024;  
Serial.print("Pojemność: (Mb):  
");  
Serial.println(volumeSize);  
służą do odczytania i obliczenia pojemności karty SD umieszczonej w złączu modułu.
```

Teraz w programie przykładowym z użyciem instrukcji *myFile=SD.open("test.txt", FILE_WRITE)* jest tworzony plik o nazwie *test.txt*. Zmienna *myFile* jest odnośnikiem do zastosowanej karty SD. Pierwszym argumentem komendy *SD.open* jest nazwa pliku, natomiast drugim metoda jego otwarcia. W przykładzie jest to *FILE_WRITE*, która two-

Listing 1. Przykład programu obsługi modułu AVTduino SD

```

/*
Przykład obsługi modułu AVTduino SD.
W pierwszej kolejności jest inicjowana karta SD oraz są wyświetlane informacje
o jej typie i pojemności. Na karcie SD jest otwierany plik test.txt, w którym
zostaje zapamiętany przykładowy tekst. Następnie zawartość utworzonego pliku
jest odczytywana i wysyłana do komputera PC, a plik zostaje usunięty z karty
SD. Komunikaty o operacjach wykonywane na karcie SD są wysyłane w za pomocą
RS232.
*/

#include <SD.h> //Biblioteka obsługi karty SD
Sd2Card card; //zmienna pomocnicze do obsługi karty
SdVolume volume;
SdFile root;
File myFile; //zmienna potrzebna dla utworzenia pliku
const int chipSelect = 8; //wyprowadzenie aktywujące kartę SD

void setup() //procedura konfiguracyjna
{
  Serial.begin(9600); //ustawienie prędkości interfejsu RS232
  Serial.print("\nInicjalizacja karty SD..."); //komunikat
  SD.begin(8); //inicjacja SD
  pinMode(10, OUTPUT);
  //jeśli inicjalizacja nie przebiegła prawidłowo to
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("Brak karty"); //wyświetlenie komunikatu
    while(1); //pętla pusta
  }
}

void loop(void) { //procedura główna programu
  Serial.print("\nTyp karty: "); //komunikat
  switch(card.type()) { //sprawdzenie typu karty
    case SD_CARD_TYPE_SD1: //jest typ SD1
      Serial.println("SD1"); //komunikat
      break;
    case SD_CARD_TYPE_SD2: //jesli typ SD2
      Serial.println("SD2"); //komunikat
      break;
    case SD_CARD_TYPE_SDHC: //jesli typ SDHC
      Serial.println("SDHC"); //komunikat
      break;
    default: //w przeciwnym razie
      Serial.println("Nieznany"); //typ karty nieznan
  }

  if (!volume.init(card)) { //sprawdzenie typu partycji karty
    Serial.println("Typ partycji nieznan"); //komunikat
    while(1); //pętla pusta
  }

  uint32_t volumesize; //zmienna
  Serial.print("\nTyp partycji: "); //wyświetlenie typu partycji
  Serial.println(volume.fatType(), DEC);
  Serial.println(); //pusta linia
  volumesize = volume.blocksPerCluster(); //obliczenie pojemności karty SD
  volumesize *= volume.clusterCount();
  volumesize *= 512;
  volumesize /= 1024;
  volumesize /= 1024;
  Serial.print("Pojemność: (Mb): "); //wyświetlenie pojemności
  Serial.println(volumesize);
  Serial.println("Otwarcie pliku test.txt"); //komunikat
  myFile = SD.open("test.txt", FILE_WRITE); //utworzenie test.txt
  //plik utworzony, więc zapis komunikatu
  if (myFile) {
    Serial.println("Zapis danych do pliku...");
    myFile.println("Przykładowy tekst dla modułu AVTduino SD");
    myFile.close(); //zamknięcie pliku
  }
  else { //w przeciwnym razie wyświetlenie
    Serial.println("Błąd otwarcia pliku test.txt"); //komunikatu
  }
  Serial.println("Wyświetlenie zawartosci karty:"); //komunikat
  root.openRoot(volume); //wyświetlenie zawartosci karty
  root.ls(LS_R | LS_DATE | LS_SIZE);
  Serial.println("Wyświetlenie zawartosci pliku test.txt"); //komunikat
  myFile = SD.open("test.txt"); //otwarcie pliku
  if (myFile) { //jeśli plik otwarty to
    Serial.println("Zawartosc pliku test.txt:");
    while (myFile.available()) { //odczyt jego zawartosci
      Serial.write(myFile.read()); //odczyt
    }
    myFile.close(); //zamknięcie pliku
  }
  else { //w przeciwnym razie wyświetlenie
    Serial.println("Błąd otwarcia pliku test.txt"); //komunikatu
  }
  Serial.println("Kasowanie pliku test.txt..."); //komunikat
  SD.remove("test.txt"); //usunięcie test.txt
  if (SD.exists("test.txt")) { //plik usunięty?
    Serial.println("Błąd usunięcia pliku test.txt."); //komunikat
  }
  else { //w przeciwnym razie
    Serial.println("Plik test.txt skasowany."); //komunikat
  }
  while(1); //pusta petla
}

```

rzy plik lub otwiera go do zapisu. Dostępny jest również tryb `FILE_READ` – otwarcie pliku tylko do odczytu.

Zapis przykładowych danych do utworzonego pliku `test.txt` odbywa się z użyciem komendy `myFile.println()`. Po zapisaniu danych plik zostaje zamknięty za pomocą instrukcji `myFile.close()`. Komendy:

```

root.openRoot(volume);
root.ls(LS_R | LS_DATE | LS_SIZE);

```

służą do odczytania i wysłania do terminala (za pomocą interfejsu RS232) wykazu plików znajdujących się na karcie SD. W dalszej części programu, plik `test.txt` jest otwierany do odczytu, a jego zawartość jest przesyłana przez RS232 za pomocą poleceń:

```

if (myFile) {
  Serial.println("Zawartosc pliku
test.txt:");
  while (myFile.available()) {
    Serial.write(myFile.read());
  }
}

```

Komenda `myFile.available()` wskazuje ile pozostało do odczytania danych z pliku które są odczytywane z wykorzystaniem komendy `myFile.read()`. Po odczycie plik zostaje zamknięty z wykorzystaniem komendy `myFile.close()`. Po odczycie plik `test.txt` zostaje trwale usunięty z karty z użyciem komendy `SD.remove("test.txt")`; Argumentem jej wywołania jest nazwa pliku. Instrukcja `SD.exists("test.txt")` służy do upewnienia się, że plik testowy na pewno został usunięty. Informacja o usunięciu lub istnieniu pliku `test.txt` jest przesyłana do komputera PC.

Komunikaty przesyłane przez program opisywany w tym przykładzie pokazano na **rysunku 1**.

Podsumowanie

Przykład dla modułu AVTduino SD będącego uzupełnieniem Arduino UNO pokazuje sposób programowej obsługi operacji na karcie SD. Jak można zauważyć, oprogramowanie Arduino zwalnia użytkownika z konieczności implementacji skomplikowanych funkcji obsługi plików, a program zapisujący i odczytujący dane nie jest skomplikowany. Dzięki temu nie dosyć, że Arduino może zapamiętywać ogromne ilości danych, to jeszcze można je bez trudu przenosić w celu obróbki czy archiwizacji na nośnik komputera PC.

Zaprezentowany przykład programu można dowolnie modyfikować, tak aby dostosować go specyfiki aplikacji. Moduł AVTduino SD będzie można z powodzeniem zastosować w wielu budowanych urządzeniach, w których jest wymagane trwałe zapamiętywanie dużych plików.

Marcin Wiązania, EP