```
// this example will play a track and then
// every five seconds play another track
//
// it expects the sd card to contain these three mp3 files
// but doesn't care whats in them
//
// sd:/mp3/0001.mp3
// sd:/mp3/0002.mp3
// sd:/mp3/0003.mp3

#include <SoftwareSerial.h>
#include <DFMiniMp3.h>
// https://github.com/Makuna/DFMiniMp3


  #include <Wire.h>
  #include <PN532_I2C.h>
  #include <PN532.h>
  #include <NfcAdapter.h>

  PN532_I2C pn532i2c(Wire);
  PN532 nfc(pn532i2c);

const uint8_t uid_ok1[] = { 0x4, 0xB1, 0x63, 0xDA, 0x31, 0x5B, 0x80 };  //UID SPT Subway 1
const uint8_t uid_ok2[] = { 0x4, 0x42, 0x55, 0x72, 0x62, 0x57, 0x80 };  //UID ciondolo viola
const uint8_t uid_ok3[] = { 0xF6, 0x99, 0xBD, 0xAC,0,0,0 };  //UID ciondolo rosso
const uint8_t uid_ok4[] = { 0x4, 0x4D, 0x23, 0x2A, 0x52, 0x5D, 0x85 };  //UID GTT1
const uint8_t uid_ok5[] = { 0x4, 0x4B, 0x23, 0x2A, 0x52, 0x5D, 0x85 };  //UID GTT2
const uint8_t uid_ok6[] = { 0x4, 0x47, 0x3D, 0xDA, 0x64, 0x5D, 0x81 };  //UID GTT3
const uint8_t uid_ok7[] = { 0x5, 0x77, 0x7C, 0x6A, 0x99, 0x54, 0xE9 };  //UID GTT4
const uint8_t uid_ok8[] = { 0x5, 0x7D, 0x8C, 0xB9, 0x59, 0x54, 0xE9 };  //UID GTT5


const uint8_t uid_ok9[] = { 0x5, 0x76, 0x23, 0x1A, 0xE9, 0x54, 0xE9 };  //UID SPT Subway 2

const uint8_t uid_ok10[] = { 0x5, 0x7D, 0x62, 0x7B, 0x5A, 0x54, 0xE9 };  //UID SPT Subway 3


const uint8_t uid_ok11[] = { 0x5, 0x72, 0x8E, 0x19, 0x7A, 0x54, 0xE9 };  //UID SPT Subway 4


const uint8_t uid_ok12[] = { 0x5, 0x73, 0x7B, 0x9B, 0xE9, 0x54, 0xE9 };  //UID SPT Subway 5

int volume=15;

const int secondi=60;

// implement a notification class,
// its member methods will get called
//
class Mp3Notify
{
public:
  static void OnError(uint16_t errorCode)
  {
    // see DfMp3_Error for code meaning
    Serial.println();
    Serial.print("Com Error ");
    Serial.println(errorCode);
  }

  static void OnPlayFinished(uint16_t globalTrack)
  {
    Serial.println();
    Serial.print("Play finished for #");
    Serial.println(globalTrack);
  }

  static void OnCardOnline(uint16_t code)
  {
    Serial.println();
    Serial.print("Card online ");
```

```
    Serial.println(code);
  }

  static void OnCardInserted(uint16_t code)
  {
    Serial.println();
    Serial.print("Card inserted ");
    Serial.println(code);
  }

  static void OnCardRemoved(uint16_t code)
  {
    Serial.println();
    Serial.print("Card removed ");
    Serial.println(code);
  }
};

// instance a DFMiniMp3 object,
// defined with the above notification class and the hardware serial class
//
//DFMiniMp3<HardwareSerial, Mp3Notify> mp3(Serial1);

// Some arduino boards only have one hardware serial port, so a software serial port is needed instead.
// comment out the above definition and uncomment these lines
SoftwareSerial secondarySerial(10, 11); // RX, TX
DFMiniMp3<SoftwareSerial, Mp3Notify> mp3(secondarySerial);

void setup()
{
  Serial.begin(115200);

  Serial.println("initializing...");

  mp3.begin();

  uint16_t volume = mp3.getVolume();
  Serial.print("volume ");
  Serial.println(volume);
  mp3.setVolume(volume);

  uint16_t count = mp3.getTotalTrackCount();
  Serial.print("files ");
  Serial.println(count);

  Serial.println("starting...");


//PN532 Setup

  nfc.begin();

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Didn't find PN53x board");
    while (1); // halt
  }
/*
  // Got ok data, print it out!
  Serial.print("Found chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print("Firmware ver. "); Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);
*/
  // Set the max number of retry attempts to read from a card
  // This prevents us from waiting forever for a card, which is
  // the default behaviour of the PN532.
  nfc.setPassiveActivationRetries(0xFF);

  // configure board to read RFID tags
  nfc.SAMConfig();

  Serial.println("Waiting for an ISO14443A card");
```

```
}

void waitMilliseconds(uint16_t msWait)
{
  uint32_t start = millis();

  while ((millis() - start) < msWait)
  {
    // calling mp3.loop() periodically allows for notifications
    // to be handled without interrupts
    mp3.loop();
    delay(1);
  }
}

void loop()
{
int test=1;
  boolean success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Buffer to store the returned UID
  uint8_t uidLength;                        // Length of the UID (4 or 7 bytes depending on ISO14443A
card type)

  // Wait for an ISO14443A type cards (Mifare, etc.).  When one is found
  // 'uid' will be populated with the UID, and uidLength will indicate
  // if the uid is 4 bytes (Mifare Classic) or 7 bytes (Mifare Ultralight)
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);

  if (success) {
    Serial.println("Found a card!");
    //Serial.print("UID Length: ");Serial.print(uidLength, DEC);Serial.println(" bytes");
    Serial.print("UID Value: ");
    for (uint8_t i=0; i < uidLength; i++)
    {
      Serial.print(" 0x");Serial.print(uid[i], HEX);
    }
    Serial.println("");
    // Wait 1 second before continuing
   // delay(1000);
    if(uidLength==7){

      test=memcmp(uid,uid_ok1,sizeof(uid));
        if (test==0) {
          Serial.println("Carta SPT Subway");
 Serial.println("track 1");
 mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(1);  // sd:/mp3/0001.mp3
        }

      test=memcmp(uid,uid_ok4,sizeof(uid));
        if (test==0) {
          Serial.println("Carta GTT1");
 Serial.println("track 2");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(2);  // sd:/mp3/0001.mp3
        }

       test=memcmp(uid,uid_ok5,sizeof(uid));
        if (test==0) {
          Serial.println("Carta GTT2");
 Serial.println("track 3");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(3);  // sd:/mp3/0001.mp3
        }

       test=memcmp(uid,uid_ok6,sizeof(uid));
        if (test==0) {
          Serial.println("Carta GTT3");
```

```
 Serial.println("track 3");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(6);  // sd:/mp3/0001.mp3
          }

//-----------------------------------------------
       test=memcmp(uid,uid_ok7,sizeof(uid));
        if (test==0) {
          Serial.println("Carta GTT4");
 Serial.println("track 3");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(5);  // sd:/mp3/0001.mp3
          }

//-----------------------------------------------
       test=memcmp(uid,uid_ok9,sizeof(uid));
        if (test==0) {
          Serial.println("SPT Subway 2");
 Serial.println("track 100");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(100);  // sd:/mp3/0001.mp3
          }
//-----------------------------------------------
       test=memcmp(uid,uid_ok10,sizeof(uid));
        if (test==0) {
          Serial.println("SPT Subway 3");
 Serial.println("track 101");
  mp3.playMp3FolderTrack(10);
 delay(500);
  mp3.playMp3FolderTrack(101);  // sd:/mp3/0001.mp3
          }

//-----------------------------------------------

      test=memcmp(uid,uid_ok2,sizeof(uid));
        if (test==0) {
          Serial.println("Ciondolo Viola");

        }

//-----------------------------------------------
      test=memcmp(uid,uid_ok8,sizeof(uid));
        if (test==0) {
          Serial.println("Carta GTT4");
            Serial.println("Stop");
            mp3.playMp3FolderTrack(10);
             delay(500);
  mp3.stop(); // stop
         }
//-----------------------------------------------


//-----------------------------------------------
      test=memcmp(uid,uid_ok11,sizeof(uid));
        if (test==0) {
          Serial.println("SPT Subway 4");

  if (volume<21){
    Serial.println("volume+");
  mp3.setVolume(++volume);}
         }
//-----------------------------------------------


//-----------------------------------------------
      test=memcmp(uid,uid_ok12,sizeof(uid));
        if (test==0) {
          Serial.println("SPT Subway 5");
    if (volume>9){
```

```
      Serial.println("volume-");
  mp3.setVolume(--volume);}
        }
//-------------------------------------------------




    } //End Test uid_lenght 7

    if(uidLength==4){

/*
      test=memcmp(uid,uid_ok3,sizeof(uid));
        if (test==0) {
          Serial.println("Ciondolo Rosso");
            Serial.println("track 4");
  mp3.playMp3FolderTrack(4); // sd:/mp3/0002.mp3
        }
*/

      test=memcmp(uid,uid_ok3,sizeof(uid));
        if (test==0) {
          Serial.println("Ciondolo Rosso");
            Serial.println("Stop");
  mp3.stop(); // stop
        }

     } //End Test uid_lenght 7
   Serial.println("");
  }
  else
  {
    // PN532 probably timed out waiting for a card
   // Serial.println("Timed out waiting for a card");
  }
  delay(1000);
  mp3.loop(); //controlla il lettore per messaggi
    delay(1);




/*
  Serial.println("track 1");
  mp3.playMp3FolderTrack(1);  // sd:/mp3/0001.mp3

  waitMilliseconds(secondi*1000);

  Serial.println("track 2");
  mp3.playMp3FolderTrack(2); // sd:/mp3/0002.mp3

  waitMilliseconds(secondi*1000);

mp3.setVolume(0);
waitMilliseconds(5*1000);
mp3.setVolume(15);


  Serial.println("track 3");
  mp3.playMp3FolderTrack(3); // sd:/mp3/0003.mp3

  waitMilliseconds(5000);
  */
}

/* NOTE
https://github.com/Makuna/DFMiniMp3/blob/master/src/DFMiniMp3.h

DfMp3_Error
    // from device
    DfMp3_Error_Busy = 1,
```

```
    DfMp3_Error_Sleeping,
    DfMp3_Error_SerialWrongStack,
    DfMp3_Error_CheckSumNotMatch,
    DfMp3_Error_FileIndexOut,
    DfMp3_Error_FileMismatch,
    DfMp3_Error_Advertise,
    // from library
    DfMp3_Error_PacketSize = 0x81,
    DfMp3_Error_PacketHeader,
    DfMp3_Error_PacketChecksum,
    DfMp3_Error_General = 0xff

*/
```