Arduino Serial MP3 Player YX5300 chip

In this post I will explain how to play MP3 with arduino and Catalex YX5300 board:

This module uses an SD card to store the songs in MP3 or WAV format. The card must be formatted to FAT32 or FAT16.
It is recommended to save the audios in numbered folders, even if you only have one folder, the most comfortable way (for example, if you want to play all the songs in a folder in a loop) is to follow the following structure:

```
.
├── 01
│   ├── 001-blue-piano.mp3
│   └── 002-gettin-ready.mp3
├── 02
│   ├── 001-ukulele-in-my-heart.mp3
│   ├── 002-inspiring-innovation.mp3
│   └── 003-thanks-for-the-memories.mp3
└── 03
    ├── 001-a-hero-is-born.mp3
    └── 002-lullaby.mp3
```

For the wiring connection if we use Arduino Uno or Nano it will be the following:

Serial MP3 Player -> ARDUINO

VCC -> 5V or 3.3V
GND -> GND
TX-> D5
RX -> D6

If we use Arduino Mega, the following connection that Serial3 uses should be used (in the code a couple of lines will also have to be commented / uncommented):

Serial MP3 Player -> ARDUINO

VCC -> 5V or 3.3V
GND -> GND

TX- > 15 RX3
RX -> 14 TX3
We must also connect to the output of the Jack 3.5 some speakers / headphones.
Then we can upload the following code or download it from Github :

```
/**********************************************************/
// Demo for the Serial MP3 Player Catalex (YX5300 chip)
// Hardware: Serial MP3 Player *1
// Board:  Arduino UNO
// http://www.dx.com/p/uart-control-serial-mp3-music-player-module-for-arduino-avr-arm-pic-blue-silver-342439#.VfHyobPh5z0
//
//
//


// Uncomment SoftwareSerial for Arduino Uno or Nano.

#include <SoftwareSerial.h>

#define ARDUINO_RX 5  //should connect to TX of the Serial MP3 Player module
#define ARDUINO_TX 6  //connect to RX of the module

SoftwareSerial mp3(ARDUINO_RX, ARDUINO_TX);
//#define mp3 Serial3    // Connect the MP3 Serial Player to the Arduino MEGA Serial3 (14 TX3 -> RX, 15 RX3 -> TX)

static int8_t Send_buf[8] = {0}; // Buffer for Send commands.  // BETTER LOCALLY
static uint8_t ansbuf[10] = {0}; // Buffer for the answers.    // BETTER LOCALLY

String mp3Answer;           // Answer from the MP3.

boolean autoResume = true;

/************ Command byte **************************/
#define CMD_NEXT_SONG     0X01  // Play next song.
#define CMD_PREV_SONG     0X02  // Play previous song.
#define CMD_PLAY_W_INDEX  0X03
#define CMD_VOLUME_UP     0X04
#define CMD_VOLUME_DOWN   0X05
#define CMD_SET_VOLUME    0X06

#define CMD_SNG_CYCL_PLAY 0X08  // Single Cycle Play.
#define CMD_SEL_DEV       0X09
#define CMD_SLEEP_MODE    0X0A
#define CMD_WAKE_UP       0X0B
#define CMD_RESET         0X0C
#define CMD_PLAY          0X0D
#define CMD_PAUSE         0X0E
#define CMD_PLAY_FOLDER_FILE 0X0F

#define CMD_STOP_PLAY     0X16
#define CMD_FOLDER_CYCLE  0X17
#define CMD_SHUFFLE_PLAY  0x18 //
#define CMD_SET_SNGL_CYCL 0X19 // Set single cycle.

#define CMD_SET_DAC 0X1A
#define DAC_ON  0X00
#define DAC_OFF 0X01

#define CMD_PLAY_W_VOL    0X22
```

```
#define CMD_PLAYING_N     0x4C
#define CMD_QUERY_STATUS      0x42
#define CMD_QUERY_VOLUME      0x43
#define CMD_QUERY_FLDR_TRACKS 0x4e
#define CMD_QUERY_TOT_TRACKS  0x48
#define CMD_QUERY_FLDR_COUNT  0x4f

/************ Opitons *************************/
#define DEV_TF          0X02


/*********************************************************************/

void setup()
{
  Serial.begin(9600);
  mp3.begin(9600);
  delay(500);

  sendCommand(CMD_SEL_DEV, DEV_TF);
  delay(500);
}


void loop()
{
  char c = ' ';

  // If there a char on Serial call sendMP3Command to sendCommand
  if ( Serial.available() )
  {
    c = Serial.read();
    sendMP3Command(c);
  }

  // Check for the answer.
  if (mp3.available())
  {
    Serial.println(decodeMP3Answer());
  }
  delay(100);
}


/*********************************************************************************/
/*Function sendMP3Command: seek for a 'c' command and send it to MP3  */
/*Parameter: c. Code for the MP3 Command, 'h' for
help.                                                                  */
/*Return:  void                                          */

void sendMP3Command(char c) {
  switch (c) {
    case '?':
    case 'h':
      Serial.println("HELP  ");
      Serial.println(" p = Play");
      Serial.println(" P = Pause");
      Serial.println(" > = Next");
      Serial.println(" < = Previous");
      Serial.println(" + = Volume UP");
      Serial.println(" - = Volume DOWN");
      Serial.println(" c = Query current file");
      Serial.println(" q = Query status");
      Serial.println(" v = Query volume");
      Serial.println(" x = Query folder count");
```

```
    Serial.println(" t = Query total file count");
    Serial.println(" 1 = Play folder 1");
    Serial.println(" 2 = Play folder 2");
    Serial.println(" 3 = Play folder 3");
    Serial.println(" 4 = Play folder 4");
    Serial.println(" 5 = Play folder 5");
    Serial.println(" S = Sleep");
    Serial.println(" W = Wake up");
    Serial.println(" r = Reset");
    break;


  case 'p':
    Serial.println("Play ");
    sendCommand(CMD_PLAY, 0);
    break;

  case 'P':
    Serial.println("Pause");
    sendCommand(CMD_PAUSE, 0);
    break;


  case '>':
    Serial.println("Next");
    sendCommand(CMD_NEXT_SONG, 0);
    sendCommand(CMD_PLAYING_N, 0x0000); // ask for the number of file is playing
    break;


  case '<':
    Serial.println("Previous");
    sendCommand(CMD_PREV_SONG, 0);
    sendCommand(CMD_PLAYING_N, 0x0000); // ask for the number of file is playing
    break;

  case '+':
    Serial.println("Volume Up");
    sendCommand(CMD_VOLUME_UP, 0);
    break;

  case '-':
    Serial.println("Volume Down");
    sendCommand(CMD_VOLUME_DOWN, 0);
    break;

  case 'c':
    Serial.println("Query current file");
    sendCommand(CMD_PLAYING_N, 0);
    break;

  case 'q':
    Serial.println("Query status");
    sendCommand(CMD_QUERY_STATUS, 0);
    break;

  case 'v':
    Serial.println("Query volume");
    sendCommand(CMD_QUERY_VOLUME, 0);
    break;

  case 'x':
    Serial.println("Query folder count");
    sendCommand(CMD_QUERY_FLDR_COUNT, 0);
    break;
```

```
187
188    case 't':
189      Serial.println("Query total file count");
190      sendCommand(CMD_QUERY_TOT_TRACKS, 0);
191      break;
192
193    case '1':
194      Serial.println("Play folder 1");
195      sendCommand(CMD_FOLDER_CYCLE, 0x0101);
196      break;
197
198    case '2':
199      Serial.println("Play folder 2");
200      sendCommand(CMD_FOLDER_CYCLE, 0x0201);
201      break;
202
203    case '3':
204      Serial.println("Play folder 3");
205      sendCommand(CMD_FOLDER_CYCLE, 0x0301);
206      break;
207
208    case '4':
209      Serial.println("Play folder 4");
210      sendCommand(CMD_FOLDER_CYCLE, 0x0401);
211      break;
212
213    case '5':
214      Serial.println("Play folder 5");
215      sendCommand(CMD_FOLDER_CYCLE, 0x0501);
216      break;
217
218    case 'S':
219      Serial.println("Sleep");
220      sendCommand(CMD_SLEEP_MODE, 0x00);
221      break;
222
223    case 'W':
224      Serial.println("Wake up");
225      sendCommand(CMD_WAKE_UP, 0x00);
226      break;
227
228    case 'r':
229      Serial.println("Reset");
230      sendCommand(CMD_RESET, 0x00);
231      break;
232   }
233 }
234
235
236
237 /*********************************************************************************/
238 /*Function decodeMP3Answer: Decode MP3 answer.                          */
239 /*Parameter:-void                                                      */
240 /*Return: The                                    */
241
242 String decodeMP3Answer() {
243   String decodedMP3Answer = "";
244
245   decodedMP3Answer += sanswer();
246
247   switch (ansbuf[3]) {
248     case 0x3A:
249       decodedMP3Answer += " -> Memory card inserted.";
250       break;
251
```

```
252     case 0x3D:
253       decodedMP3Answer += " -> Completed play num " + String(ansbuf[6], DEC);
254       break;
255
256     case 0x40:
257       decodedMP3Answer += " -> Error";
258       break;
259
260     case 0x41:
261       decodedMP3Answer += " -> Data recived correctly. ";
262       break;
263
264     case 0x42:
265       decodedMP3Answer += " -> Status playing: " + String(ansbuf[6], DEC);
266       break;
267
268     case 0x48:
269       decodedMP3Answer += " -> File count: " + String(ansbuf[6], DEC);
270       break;
271
272     case 0x4C:
273       decodedMP3Answer += " -> Playing: " + String(ansbuf[6], DEC);
274       break;
275
276     case 0x4E:
277       decodedMP3Answer += " -> Folder file count: " + String(ansbuf[6], DEC);
278       break;
279
280     case 0x4F:
281       decodedMP3Answer += " -> Folder count: " + String(ansbuf[6], DEC);
282       break;
283   }
284
285   return decodedMP3Answer;
286 }
287
288
289
290
291
292
293 /********************************************************************************/
294 /*Function: Send command to the MP3                                             */
295 /*Parameter:-int8_t command                                                     */
296 /*Parameter:-int16_ dat  parameter for the command                              */
297
298 void sendCommand(int8_t command, int16_t dat)
299 {
300   delay(20);
301   Send_buf[0] = 0x7e;   //
302   Send_buf[1] = 0xff;   //
303   Send_buf[2] = 0x06;   // Len
304   Send_buf[3] = command;//
305   Send_buf[4] = 0x01;   // 0x00 NO, 0x01 feedback
306   Send_buf[5] = (int8_t)(dat >> 8);  //datah
307   Send_buf[6] = (int8_t)(dat);       //datal
308   Send_buf[7] = 0xef;   //
309   Serial.print("Sending: ");
310   for (uint8_t i = 0; i < 8; i++)
311   {
312     mp3.write(Send_buf[i]) ;
313     Serial.print(sbyte2hex(Send_buf[i]));
314   }
315   Serial.println();
316 }
```

```
317
318
319
320 /*********************************************************************************/
321 /*Function: sbyte2hex. Returns a byte data in HEX format.              */
322 /*Parameter:- uint8_t b. Byte to convert to HEX.                       */
323 /*Return: String                                                       */
324
325
326 String sbyte2hex(uint8_t b)
327 {
328   String shex;
329
330   shex = "0X";
331
332   if (b < 16) shex += "0";
333   shex += String(b, HEX);
334   shex += " ";
335   return shex;
336 }
337
338
339
340
341 /*********************************************************************************/
342 /*Function: sanswer. Returns a String answer from mp3 UART module.     */
343 /*Parameter:- uint8_t b. void.                                         */
344 /*Return: String. If the answer is well formated answer.               */
345
346 String sanswer(void)
347 {
348   uint8_t i = 0;
349   String mp3answer = "";
350
351   // Get only 10 Bytes
352   while (mp3.available() && (i < 10))
353   {
354     uint8_t b = mp3.read();
355     ansbuf[i] = b;
356     i++;
357
358     mp3answer += sbyte2hex(b);
359   }
360
361   // if the answer format is correct.
362   if ((ansbuf[0] == 0x7E) && (ansbuf[9] == 0xEF))
363   {
364     return mp3answer;
365   }

      return "???: " + mp3answer;
    }
```

If we are working with Arduino Mega, line 13 should be commented

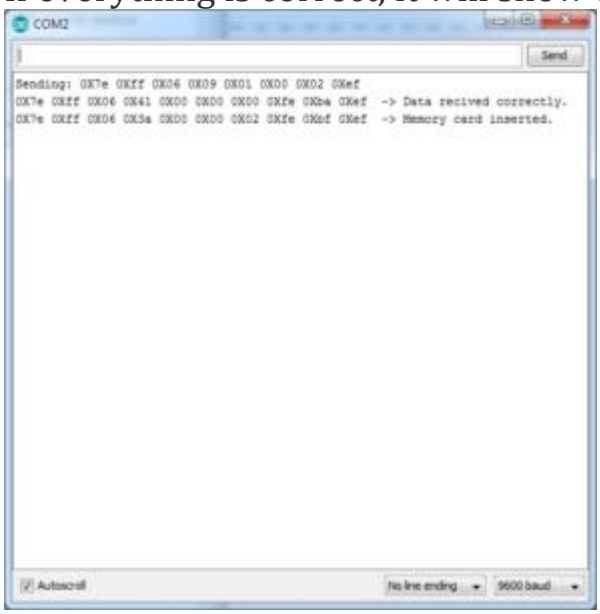
```
1 #include <SoftwareSerial.h>
```

Comment line 18 and uncomment line 19

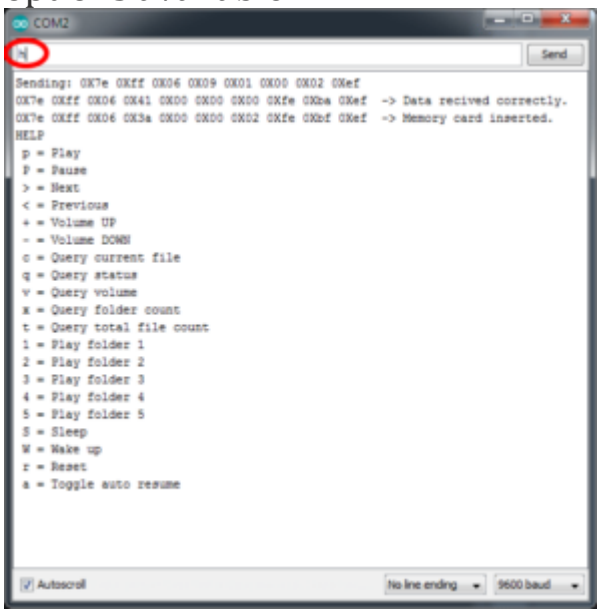
```
1 SoftwareSerial mp3(ARDUINO_RX, ARDUINO_TX);
2 //#define mp3 Serial3    // Connect the MP3 Serial Player to the Arduino MEGA Serial3 (14 TX3 -> RX, 15 RX3 ->
  TX)
```

Now we can manipulate the player through the serial monitor of the Arduino IDE itself. Once the code is uploaded to the Arduino, we open the serial monitor:

If everything is correct, it will show us the text:



If we enter the letter "h" it will show us the help menu with the different options available:



If we enter the letter “p” an audio will be played and if we enter “1” it will play all the songs in folder 01.

For more information on how the module works, you can go to my Github repository where you can find more documentation: https://github.com/cefaloide/ArduinoSerialMP3Player