Portable Music Player

by Constantin Traian-Alexandru

Introduction

The aim of this project is to build a practical portable music player that is compatible with common hardware and files (standard batteries, microSD cards and .mp3 music files, respectively). The ultimate goal of this would be to have a music player that can be built from scratch by anyone, and which is able to play music files without relying on cloud services.

Description

A DFPlayer Mini module will be used to read the .mp3 files inside the microSD card and will output the decoded audio. The DFPlayer will be used instead of a conventional SD card reader due to the difficulty in decoding .mp3 signals on an Arduino.

The Arduino will control the interfacing with the user (control buttons for input, an LCD display for output), as well as the functions of the DFPlayer (play/pause, track selection etc.).

The LCD Display will show information such as the current track. The buttons can be used to play/pause and navigate, as well as increase/decrease the volume. The audio is outputted to two stereo speakers (powered by an amplifier).

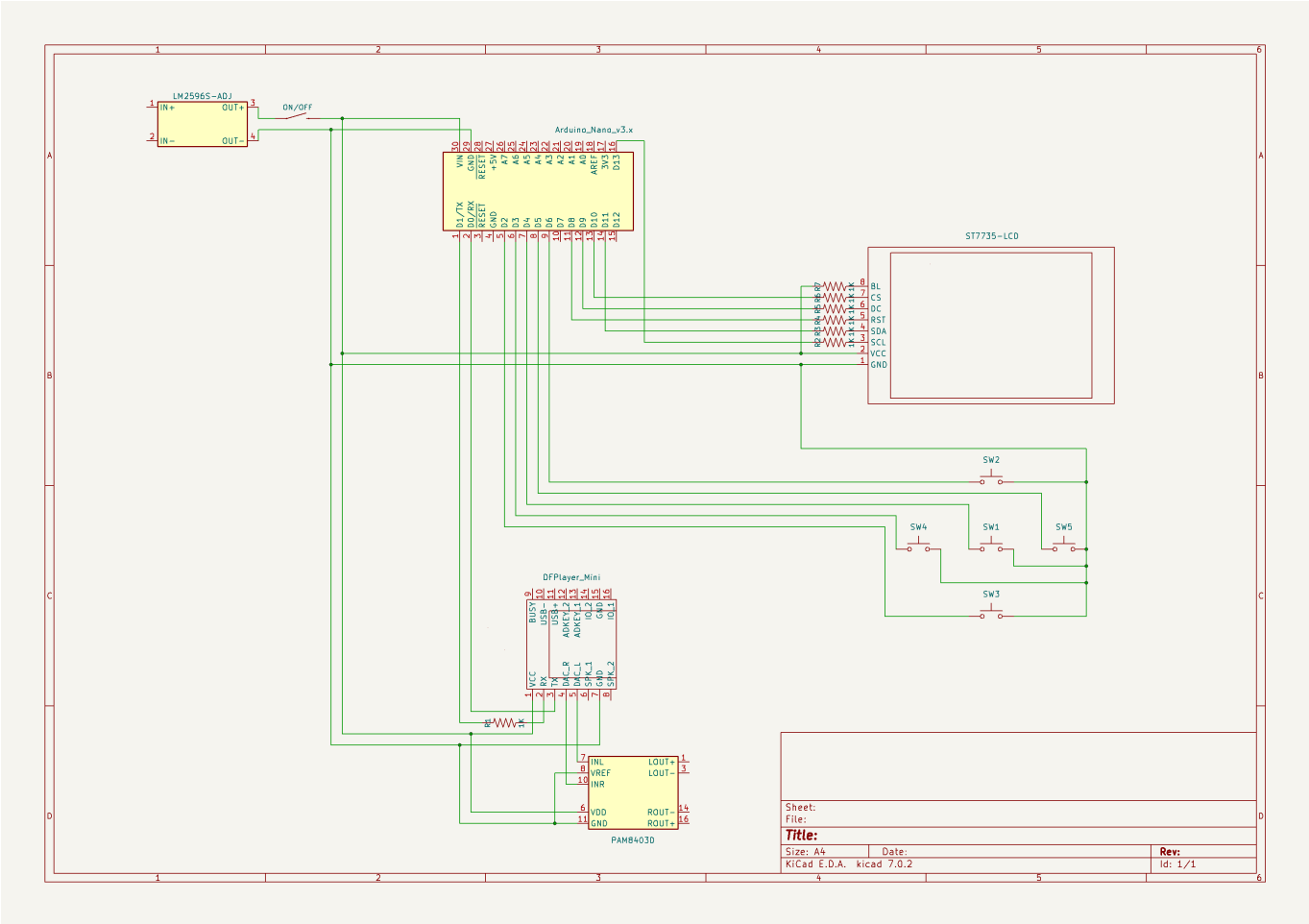
Hardware Design

The following components will be used in the project:

- Arduino Nano
- DFPlayer Mini
- ST7735 LCD Display
- PAM8403 Amplifier
- 2 x 1W 8Ω Speakers
- LM2596 Adjustable Voltage Regulator
- 9V Battery
- Various smaller components e.g. buttons, switches, resistors.

The LM2596 regulator has been chosen in order to allow for more current to power the many devices used in this project. Arduino's own AMS1117 regulator can provide up to 800mA output according to its datasheet, but the total consumption of the combined devices is only marginally below that parameter, so the LM2596 with its generous 3A maximum output current was used to ensure the proper functioning of the system.

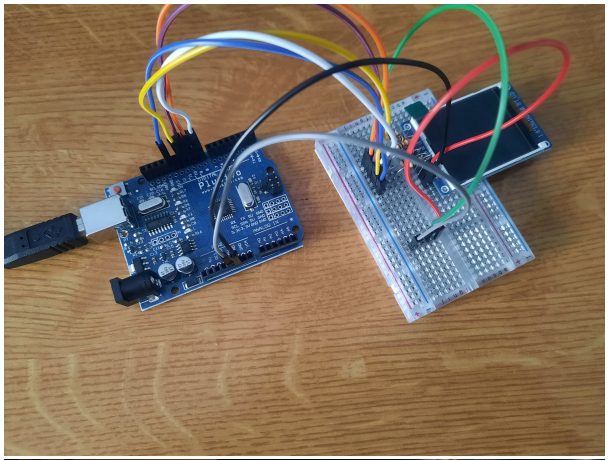
Schematic

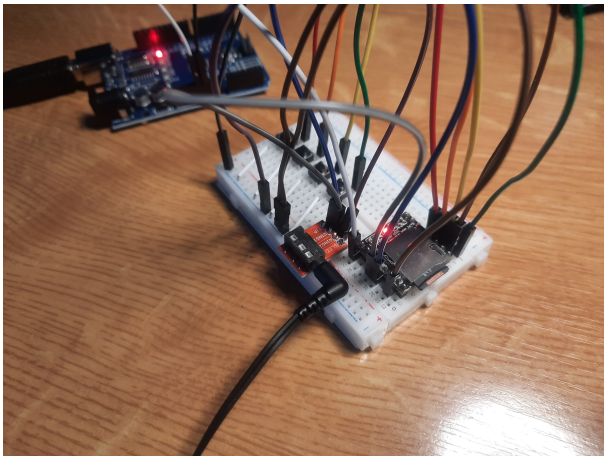


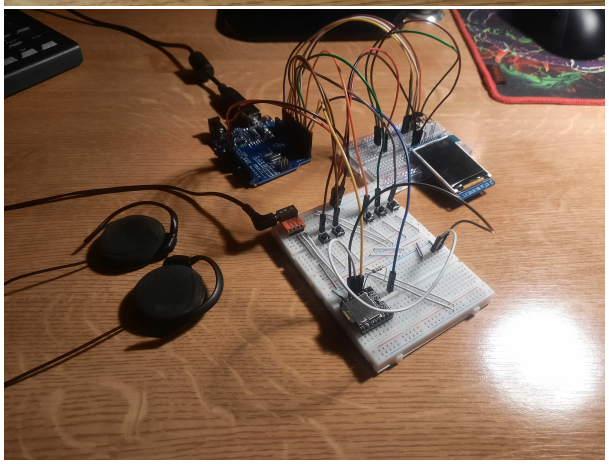
The schematic is representative of the final product, however the battery and the speakers have not been represented, and the connections to the button pins may not be identical. The pinout of the Arduino is visually that of a Nano, however the final product uses an Arduino Uno.

Prototyping

Breadboard test designs for the screen, DFPlayer, as well as testing the complete functionality using earphones:





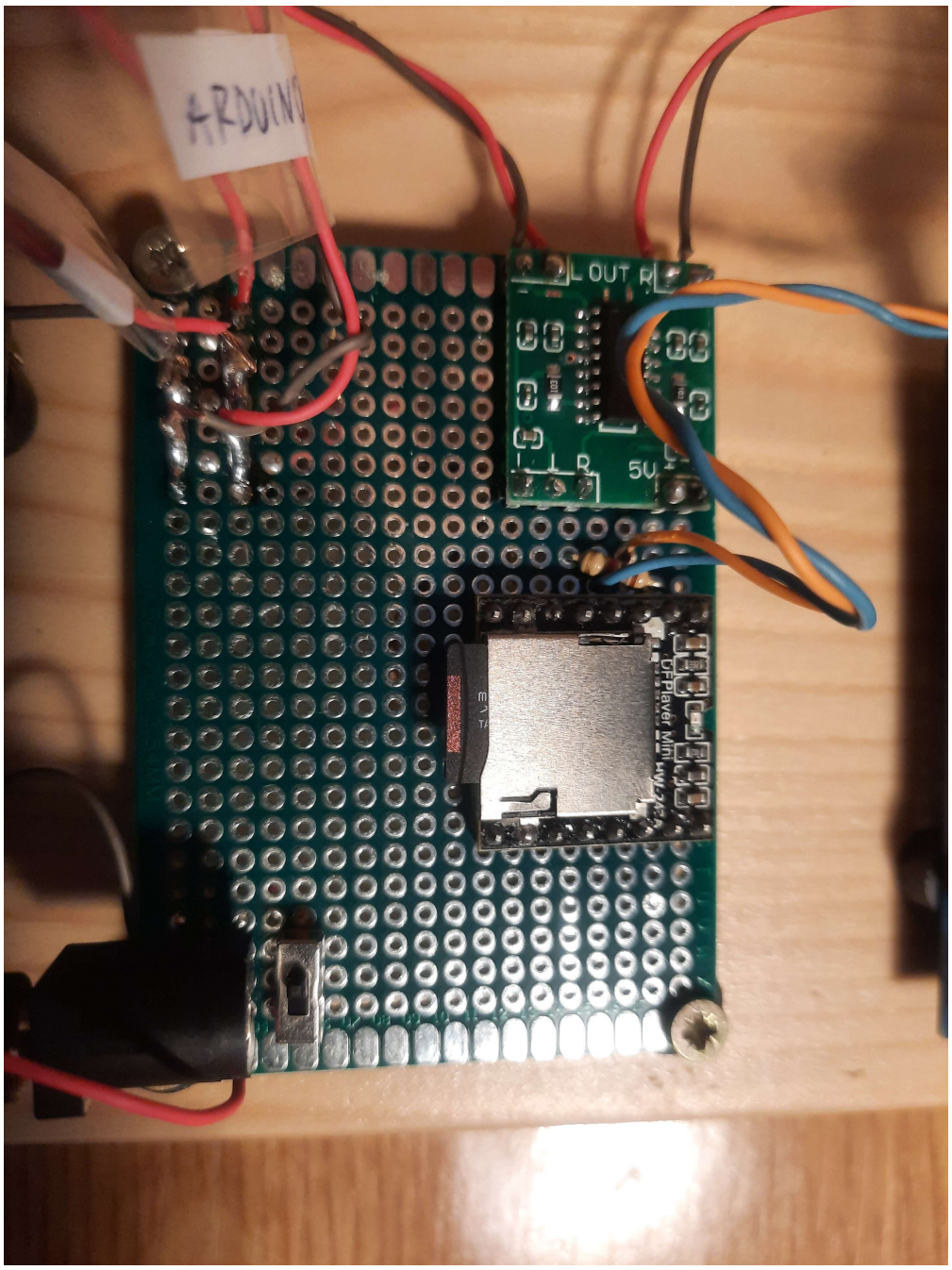


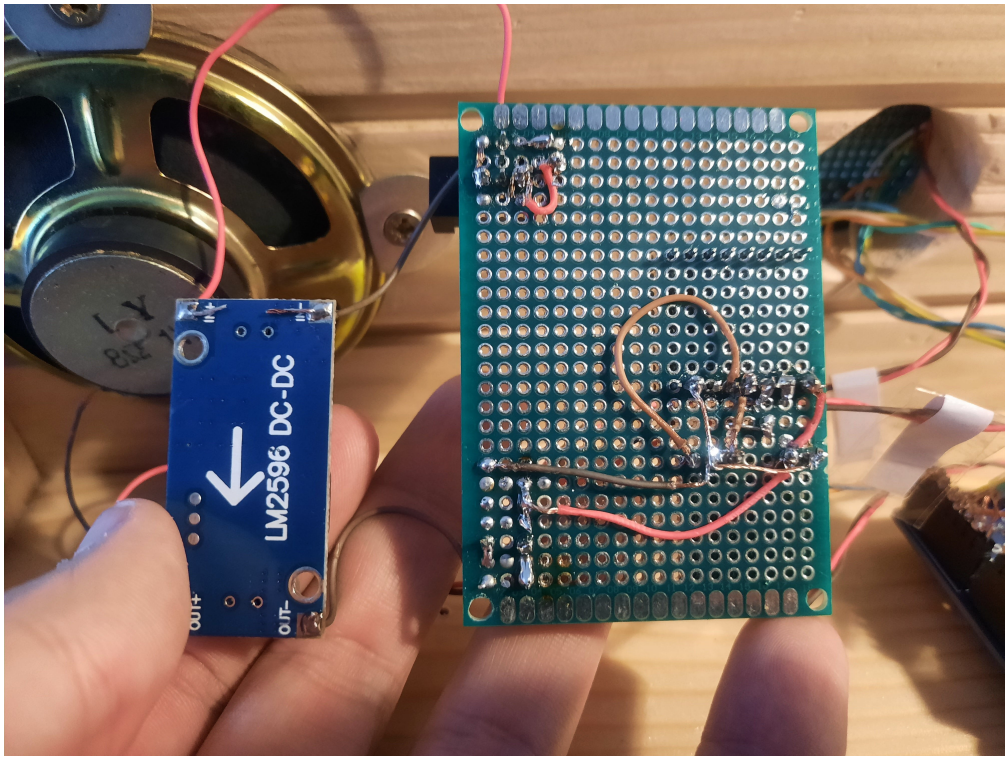
Soldered PCBs

While the breadboard prototype is entirely functional, because the project involves sound signals, noise is a significant concern. Noise interference has been observed to occur, primarily due to the many connectors the breadboard has, and therefore a soldered PCB design is more optimal.

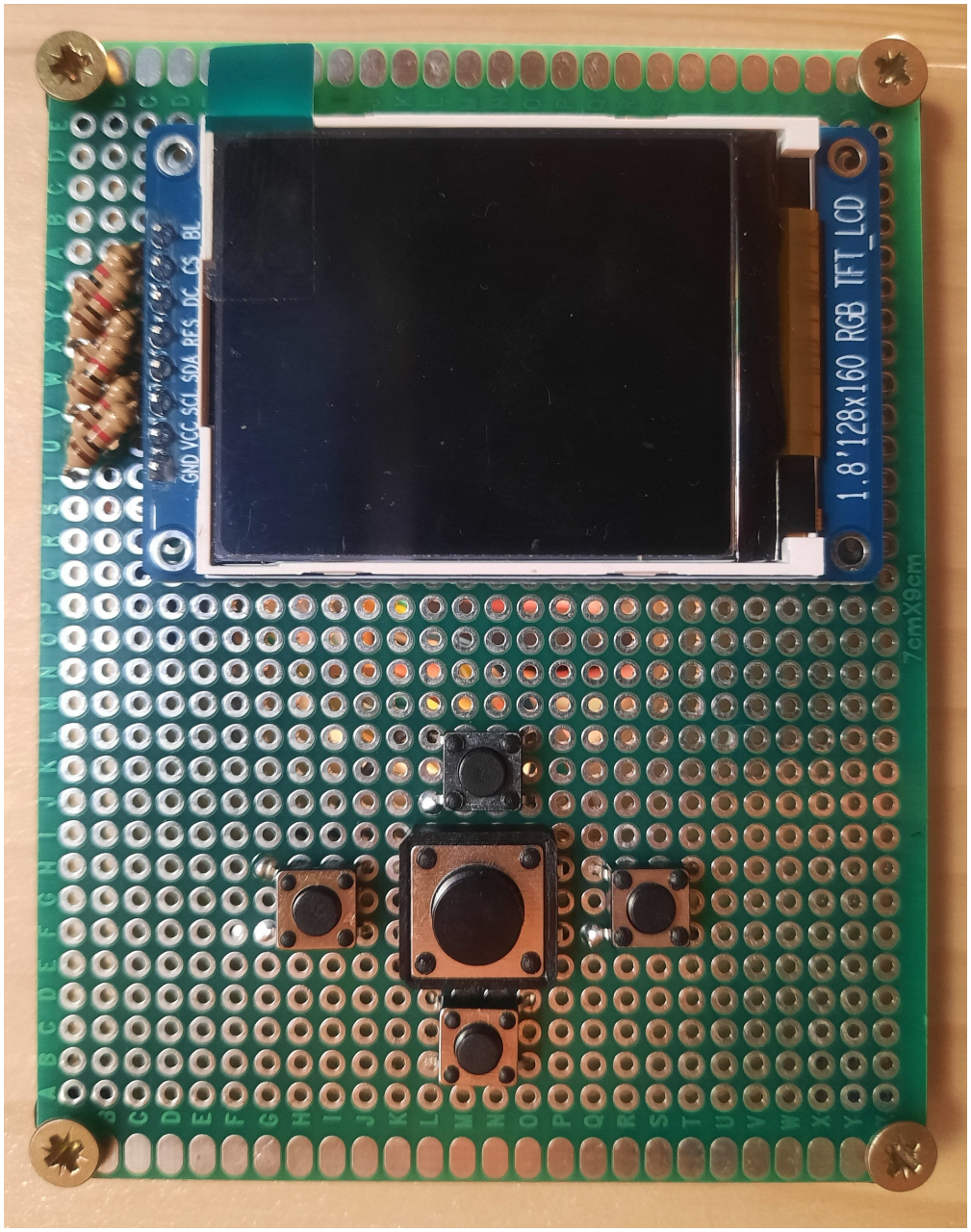
It is also recommended to use a 1KΩ resistor for the RX pin on the DFPlayer in order to avoid a poor sound quality.

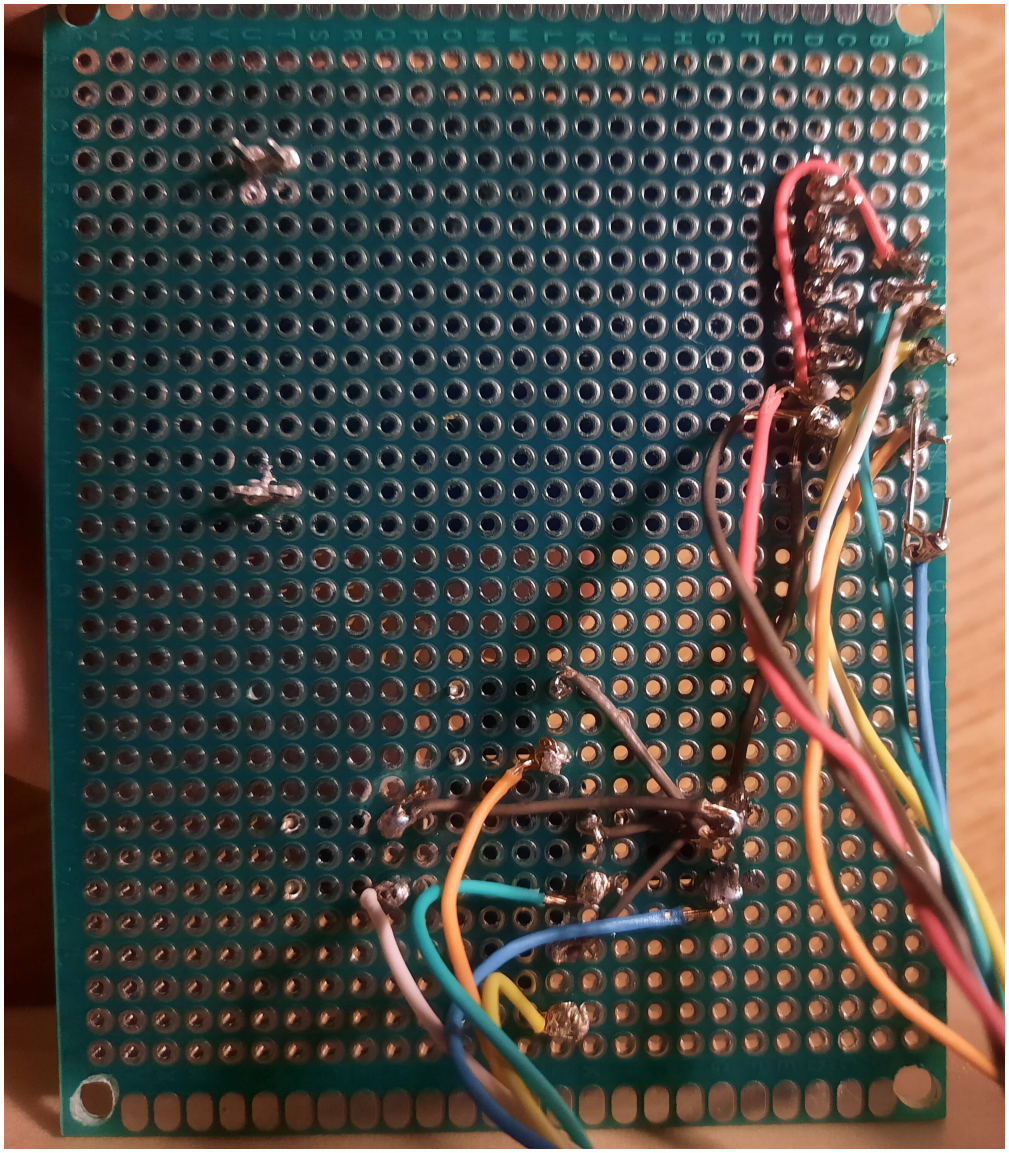
Sound & Power board: front and back


L OUT R
5V



Front Panel board: front and back


1.8'128x160 RGB TFT LCD
GND VCC SCL SDA RES DC CS BL
7cmX9cm



The front panel board is wired according to the schematic. The sound and power board has both the DFPlayer and the amplifier wired to it, in addition to the DC power jack 9V battery input and the 5V power rails where all the devices draw power from.

Software Design

The dedicated TX and RX Arduino pins have been used to communicate with the DFPlayer's. However, these pins are also used by the computer in order to program the Arduino, and as such they need to be disconnected while doing so.

The following libraries have been used in this project:

- Adafruit_GFX.h - The core Adafruit graphics library
- Adafruit_ST7735.h - The hardware-specific library for the ST7735 microcontroller
- SoftwareSerial.h - An USART interface for communicating with the DFPlayer

The file structure

Firstly, one major drawback of the DFPlayer in contrast to a normal SD card reader is the fact that it and only it has access to the contents of the micro SD card. The Arduino is able to give commands to the DFPlayer specifying what tracks should be played, but it will never have direct access to them.

Consequently, in order to allow the users to select the tracks they want to be played, a replica of a file system has been implemented, consisting of two multidimensional arrays: the first one, “filesystem”, stores the names of the songs (.mp3 files), as well as that of the album (file directory) in which they are located. The second array, “fileindex”, is necessary in order to account for the position of the tracks in the “filesystem” array, relative to the order in which they are read in the sequential “queue” of the DFPlayer.

The DFPlayer reads from the microSD card a folder structure of the following kind:

It is to be noted, however, that the files and folders need not be named to 01, 002 etc., as the DFPlayer orders them according to the date and time that they were written to the microSD card.

The DFPlayer is also able to read up to 255 files inside 99 folders - however, the small 2kB SRAM of the ATmega328p is not able to even initialize that large of a multidimensional array; the memory of the “file system” was restricted to holding 4 folders of 16 files.

The interface screens

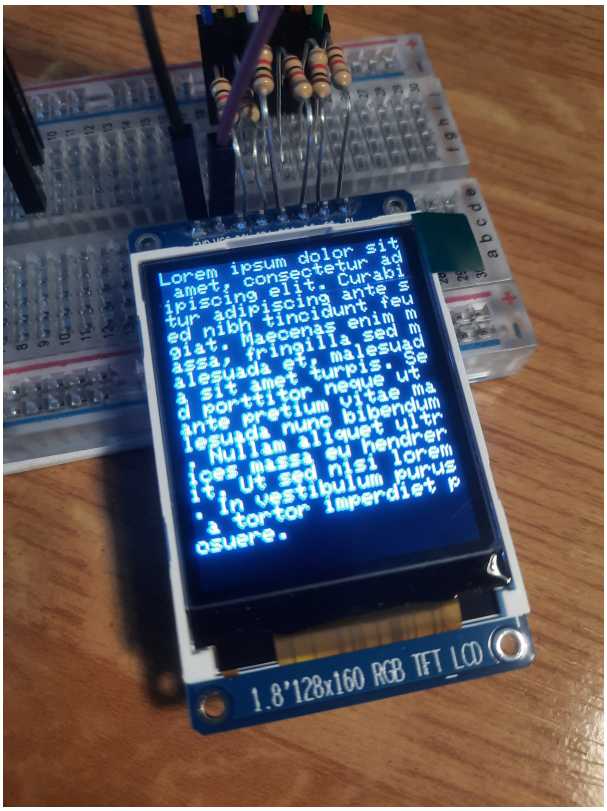
On the basis of such a file structure as discussed above, I have thought of implementing an interface to navigate through the albums and songs, as well as playing them. The result was a 3-screen system which allows the user to navigate through a list of albums, then a list of songs inside an album, and then allows for playing a song with basic music player controls (play/pause, next, volume up/down). In order to keep the project simplistic, the 5 buttons on the front panel each have specific functions when pressed, depending on the screen which is shown:

Navigation inside the screens is made by moving the orange cursor up and down, then selecting the desired list entry.

The screens can be scrolled up and down if the album has many songs that cannot be displayed on the screen, similarly to old Nokia UIs.

Offset bugfix

When initially tested, I discovered that my ST7735 LCD screen had a problem with the offset of the pixels. This can be attributed to the microcontroller of the display not being an original manufactured by Sitronix.



I managed to correct this problem by adding a manual offset inside the Adafruit library.

Inside Documents\Arduino\libraries\Adafruit_ST7735_and_ST7789_Library\Adafruit_ST77xx.cpp:

```
void Adafruit_ST77xx::commonInit(const uint8_t *cmdList) {

  _colstart = 2;
  _rowstart = 1;

  begin();

  if (cmdList) {
    displayInit(cmdList);
  }
}
```

Code summary

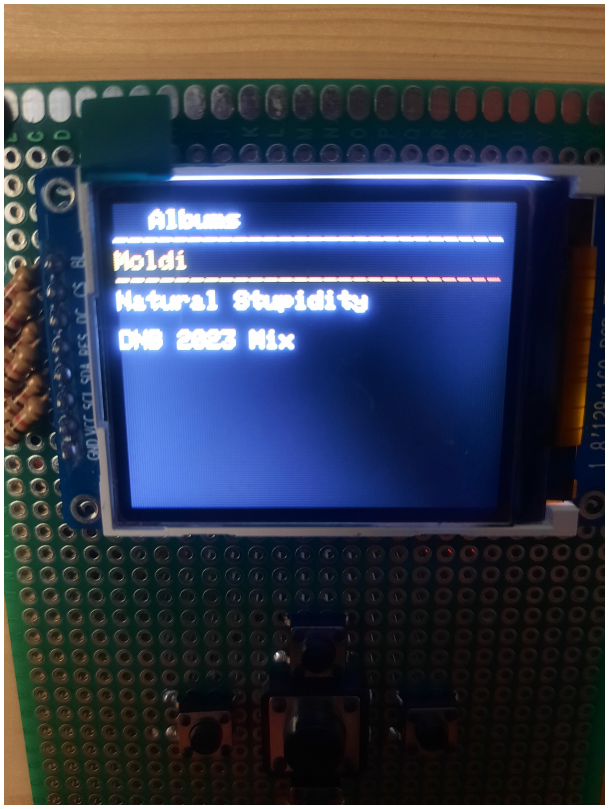
The code can be summarized as follows:

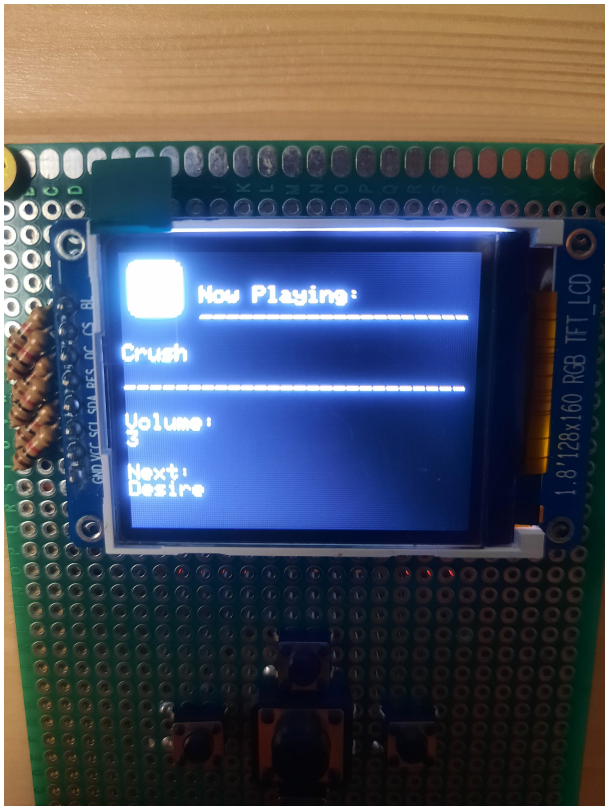
1. The initial definitions, configurations and variable initializations
2. In setup(): defining the songs in the file system
3. In setup(): starting the connections from the Arduino to the ST7735 screen and the DFPlayer

4. In loop(): multiple if statements that perform the functions assigned for the button presses depending on what is currently displayed on the screen
5. Outside of loop(): method definitions for functions that deal with binary commands to the DFPlayer and graphical commands to the screen (these functions are ultimately called by the button presses)

Resulting screens

These are the screens rendered by the software (Left to right: albums, songs, playing):



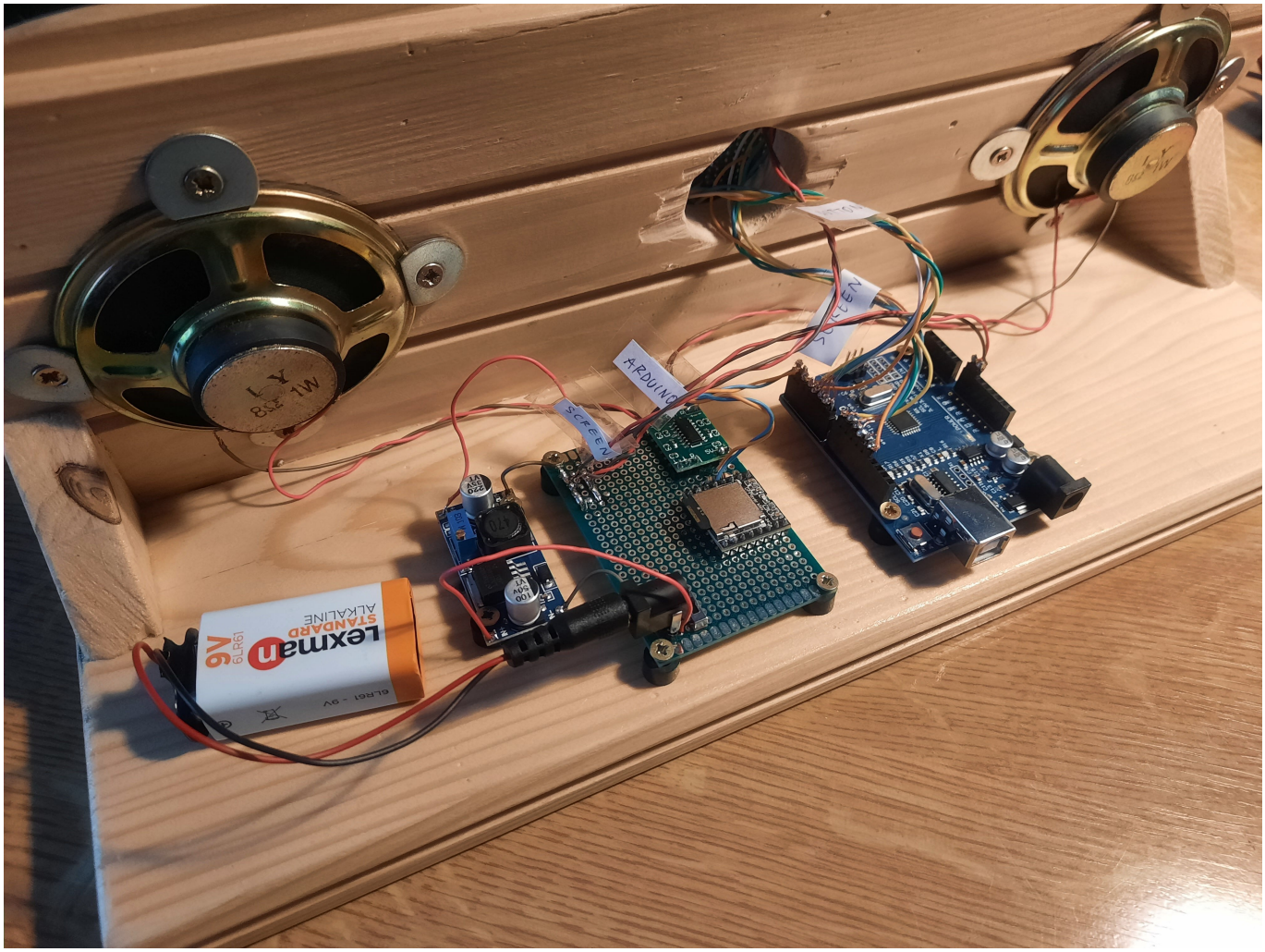


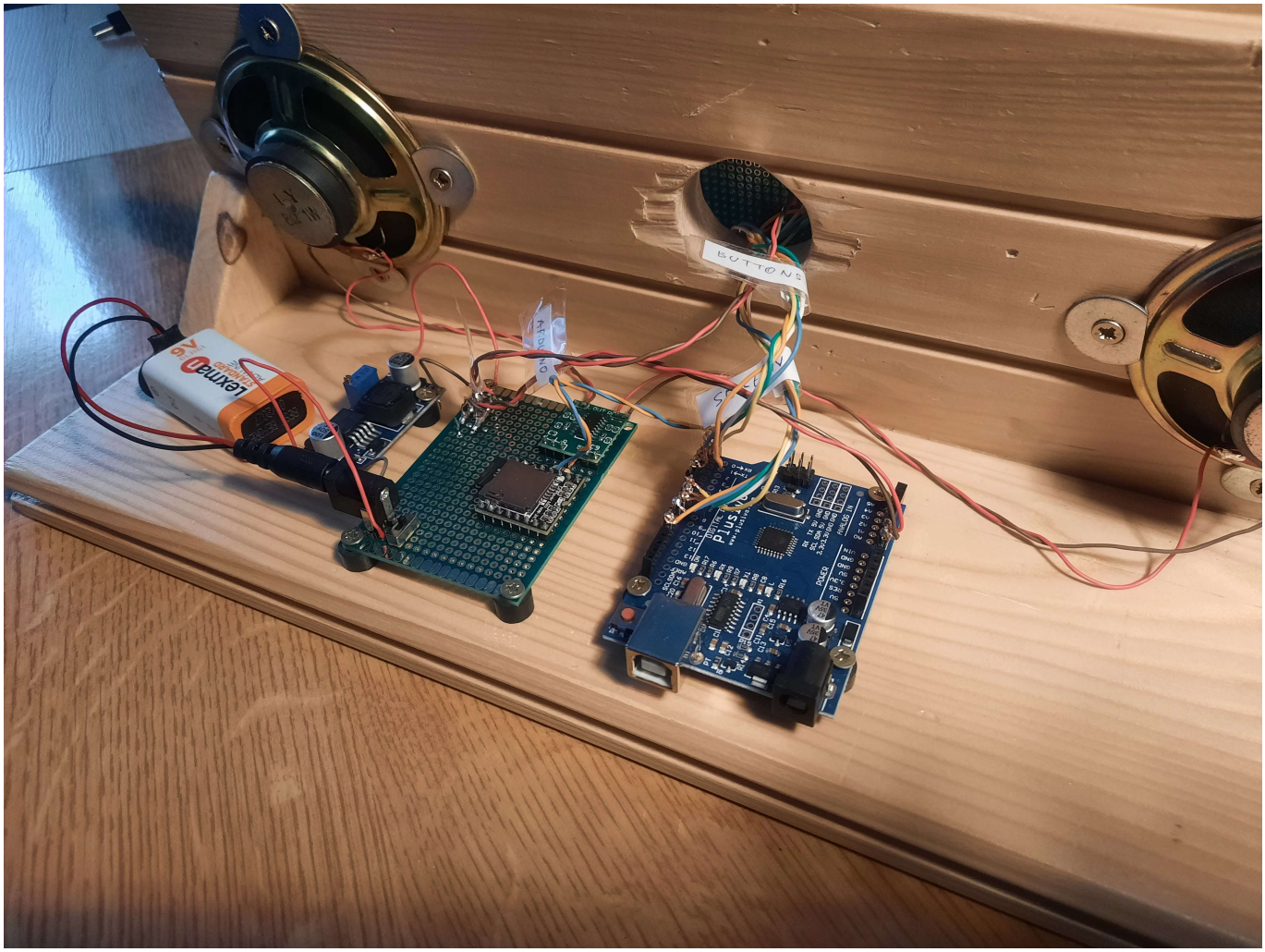
Results

Functionality demonstration video

Front view

Back view


ARDUINO
SCREEN
lexman
STANDARD
ALKALINE
6V
6LR61



Conclusions

There are many improvements that can potentially be made to the design, for example: the addition of an EEPROM memory module to expand the capacity of the files; an automated script for copying songs in sync from the PC to both the EEPROM and the microSD card.

However, I feel that the scope of the project has been fulfilled and that it demonstrates the capabilities of the Arduino in being used for the implementation of a Portable Music Player.

Download

Archive: pmp.zip

PDF: portable_music_player.pdf

Bibliography/Resources

Software used:

- Arduino IDE for programming
- Inkscape for the block diagram
- KiCad for the circuit schematic

Study materials:

- DFPlayer Mini wiki
- ATmega328P Datasheet
- Arduino mp3 player with similar functionality but different components
- About using the DFPlayer commands
- About using the EEPROM for file indexing

Export to PDF