Braille Printer

Introduction

This project implements a complete version of a braille printer. You just insert your sd card with the desired files, select the one you want, insert the paper, and the print is done!. I picked this project based on one question:

What can I build using scrap i have in my house for some reason?

After searching inside some boxes I found the main part of an old printer. (the head that moves on the x axis in order to put the ink down)
So this project was (almost) built without paying for anything.

General Description

Let's say, for whatever reason, that you want to send a letter to a blind person, or maybe you have someone blind in your circle/family. This project was thought as something very plug and play, that anyone can build at home with somewhat minimal materials. Someone with minimal coding experience can download this from github and make their own at home!

Main Workflow

An Arduino uno has a LCD screen (shield) on top of it. Let's call this arduino: arduino 1. This is the brain of the operation.
When the user want to print a .txt file, he inserts the sd card containing the file he wants in the sd card reader on the bottom of the LCD shield.

When powered on, the screen greets the user. After pressing the start button, the user can select the file he wants to print using the touch screen. After selecting the file, the second arduino comes into play. This is the muscle of the operation, so let's call it arduino 2.
When a file is selected, a custom communication protocol (over uart) is initiated. Arduino 1 sends the characters one by one, but only after arduino 2 acknowledges that his job (the printing) is done.

While the printing is happening, arduino 1 displays all the letters, numbers or special characters, alongside their braille representation, and a progress bar.
When both arduinos finish working, everything resets, allowing the user to print a second time.

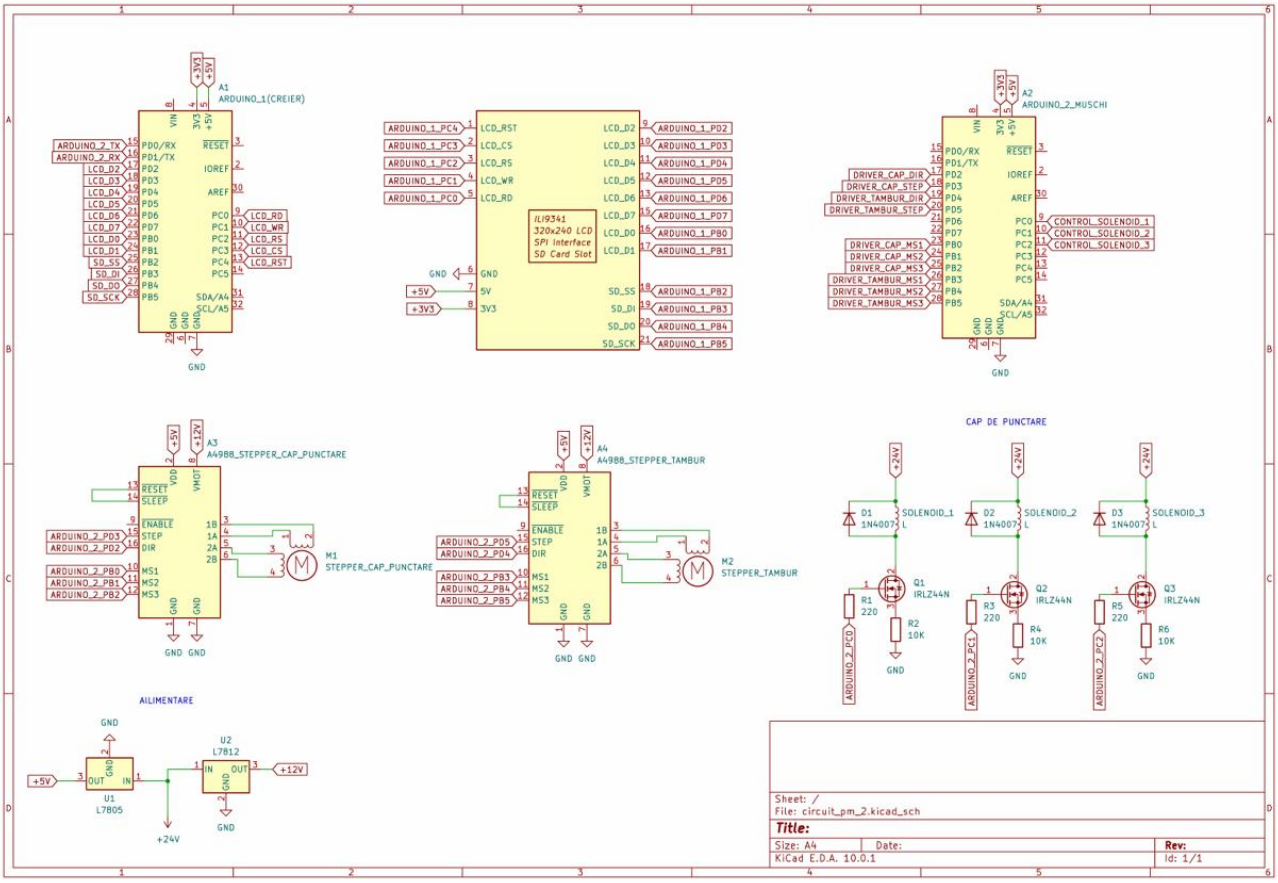
The paper will be printed in reverse, so it will start from the right. Each letter in the braille alphabet is built from two columns of 3 rows of dots, so 6 in total. The printer pushes only one column at a time (so maximum 3 dots at once). The head of the machine moves only on the x axis.

Hardware Design

For this project to work I had to use two arduino uno boards (for gpio reasons).
Here is a list of every component:

- Arduino uno (the brain) - let's call it 1
- Arduino uno (the consumer) - let's call it 2
- ILI9341 2.8' LCD with touchscreen
- SD-card reader (from the lcd shield)
- 2 stepper motors (bipolar, 12V, don't know the name)
- 2 A4988 drivers
- 3 push solenoids (built myself because they are small)
- 3 x Transistors
- 2 x Condenser: 100 nf
- One 24V power source
- n x Step-down Converter

Main Circuit



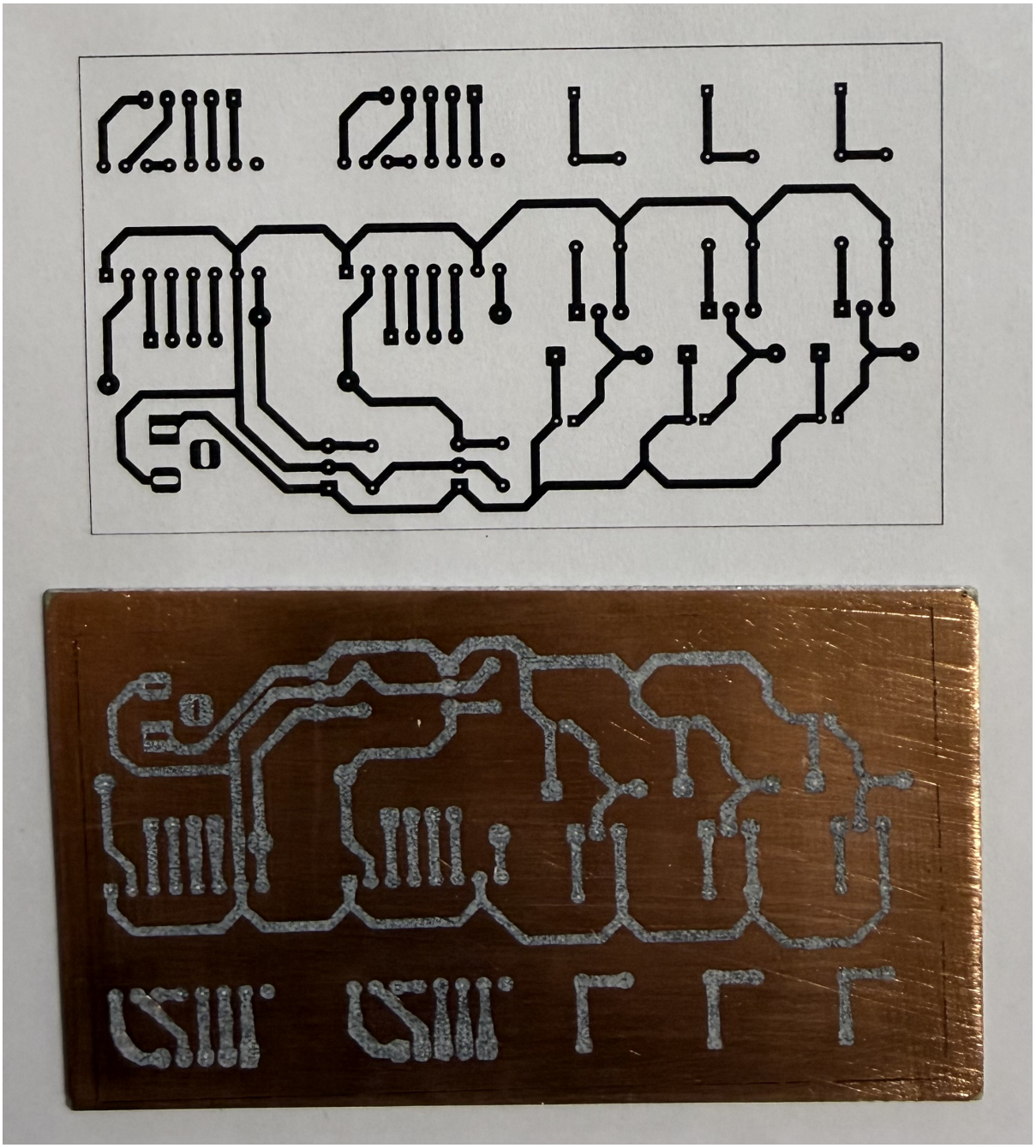
Perhaps some important information:

- Because i could not find the right solenoid push actuator, i had to make my own.
- The shaft that feeds the paper was recycled from an old printer
- The motor and gears that drive the head around were recycled from that same old printer

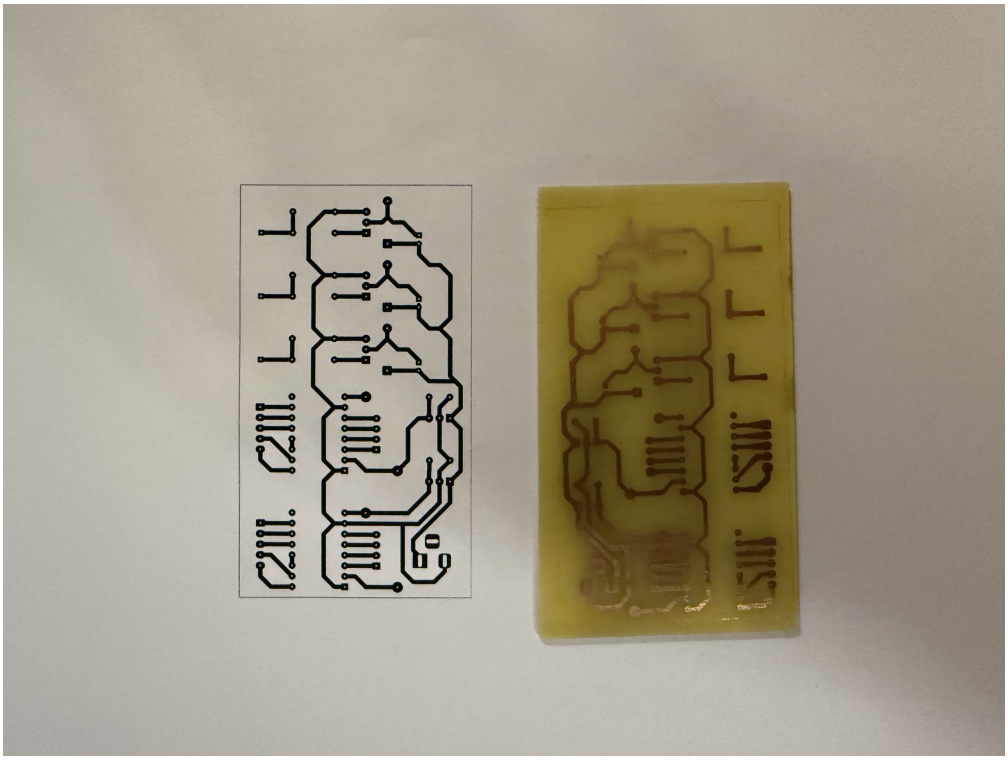
I wanted to go the extra mile when doing this project, and in the spirit of DIY, i had to make my own pcb. so i came up with this:

So this, alongside two arduino uno is the full circuit. The holes in the top right are meant for the a4988 modules. In the testing phase of this build i managed to burn some. The conclusion: every component that can burn has to be in header pins. So the pcb has a pot of header pins, besides the ones seen in the 3d render.
The pcb was routed by hand and adapted for the laser printer method (larger routes). The material i had at home was only one sided, so i had no possibility for vias. There are 4 in the pcb that had to be soldered with normal wires.

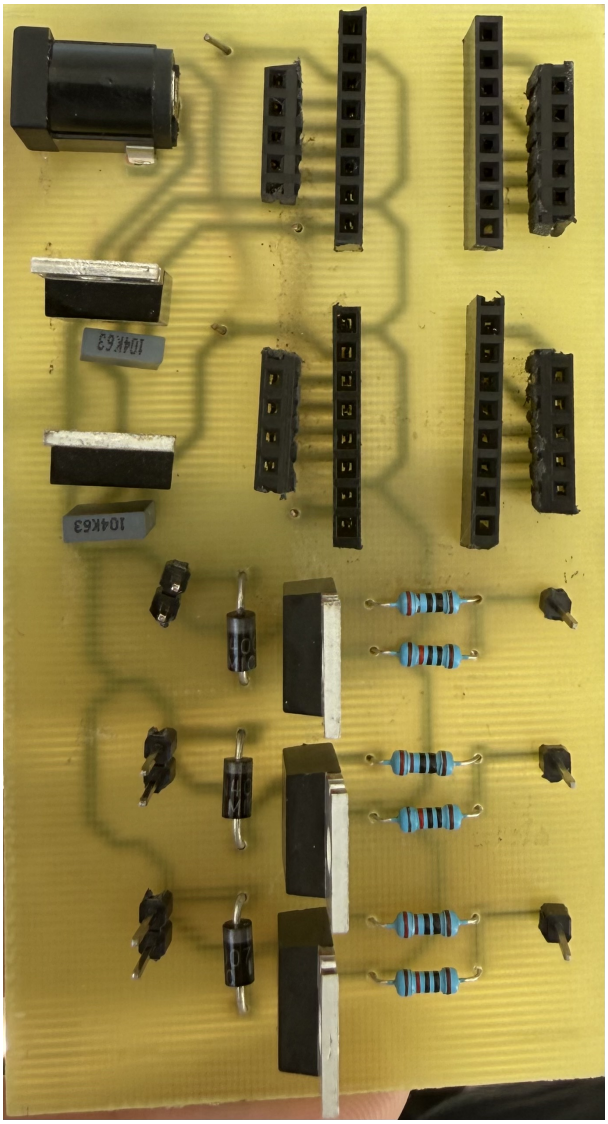
The process was really straight forward.
I printed the circuit on the board:



After this step, i submerged the printed board in FeCl3 for about 30 min, and it came out looking like this!



In the end i added all the components and ended up lookig like this! (drilling with 0.8 is a pain like i've never experienced before)



The holes are not in line with each other, but that is the best i could!
As every first time trying, i forgot to add the dc jack for the arduinos, but i added it later.

Rant about LM7812:

It was the first time trying this component. This is a voltage “stabilizer”. It takes a voltage between 12V and 25V and generates a steady 12 V on the output. It works by converting the surplus energy into heat.
The first time plugging the pcb in, the stepper motors started moving normally, but as time passed, they became weaker and weaker until coming to a full stop. When i picked up the pcb to see if anything got burnt, i picked up the pcb by the lm7812. It as hot. The data sheet says 150 deg hot (celsius).
After a lot of wasted time, i read in the data sheet that in case of high temperatures, the component enters a “protective state”, that shuts off the component.
In the end the solution was adding a radiator. Now it looks really goofy but works just fine.

== Making the push solenoid ==
I could not find a component with enough power to make the perfect point, so i had to make my own.
* The shaft is 3D printed

- Length: 2.75cm
- Internal diameter: 3mm
- External diameter: 8mm

* The wire used: 0.2 mm copper wire * It has 11 rows of 137 wire passes * Power source: 24V * Resistance 12 ohm

Because of this it tends to get REALLY hot after continued use. During the testing of the software, because of intense use, the nail that pushes the paper gets hot. This heat transfers to the inside of the shaft, that melts, making the solenoid unusable. So for the testing, it had to be done over multiple days.

The springs bring another challenge. It had to be soft enough to get compressed by the magnetic field, and springy enough to push back the nail. The height from which the nail drops is important as well. The higher it is, the harder the punch.

All the springs i had were too hard, so I had to make my own. The first iteration was made out of 0.2 mm copper wire. the second out of 0.3mm copper wire. the 0.2 one was perfect in strength, but it could not hold it's shape. It got compressed after a couple of prints. So i had to get 0.2 mm made from another material. Fortunately, 0.09 gauge guitar strings are almost 0.2mm, and the material is harder. These are perfect for this project. So i made my own springs using a drill.

There was a lot of iteration when building this design, a lot was not mentioned.

Software Design

The project was done in PlatformIO, using AVR as much as possible. It is as bare metal as it gets.

The code is separated in two main parts : arduino 1 (the brain) and arduino 2 (the muscle).

Arduino 1

This arduino holds most of the computational load. It has the purpose of interfacing with the user using a touchscreen lcd, and reading the files from the sd card. The communication with the second arduino is made using a custom usart protocol (we'll com back to this later).

Arduino 2

This arduino holds the code that moves stuff around. It communicates with the first arduino. After processing a single character, it waits foe the following and so on. You could say that the system is a producer-consumer situation.

Used imports

- avr/io.h
- util/delay.h

Self made imports

- init.h → init function for lcd and other stuff
- timers.h → the watchdog timer for resets and the timer for an animation
- touch_module.h → the adc implementation of the resistive touchscreen
- SimpleSd.h → a custom built sd card reader. I needed a really memory-small module, so I built my own
- LCD.h → library for the ui elements. It is built on the MCUFriend library
- usart.h → a self made usart library, containing the custom protocol
- stepper.h → library for controlling a generic stepper motor, with all functions included
- solenoid.h → abstraction layer over some port changes

USART Protocol

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

TODO

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

TODO

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF