Password manager

Rusu Dinu-Ștefan 1221EA

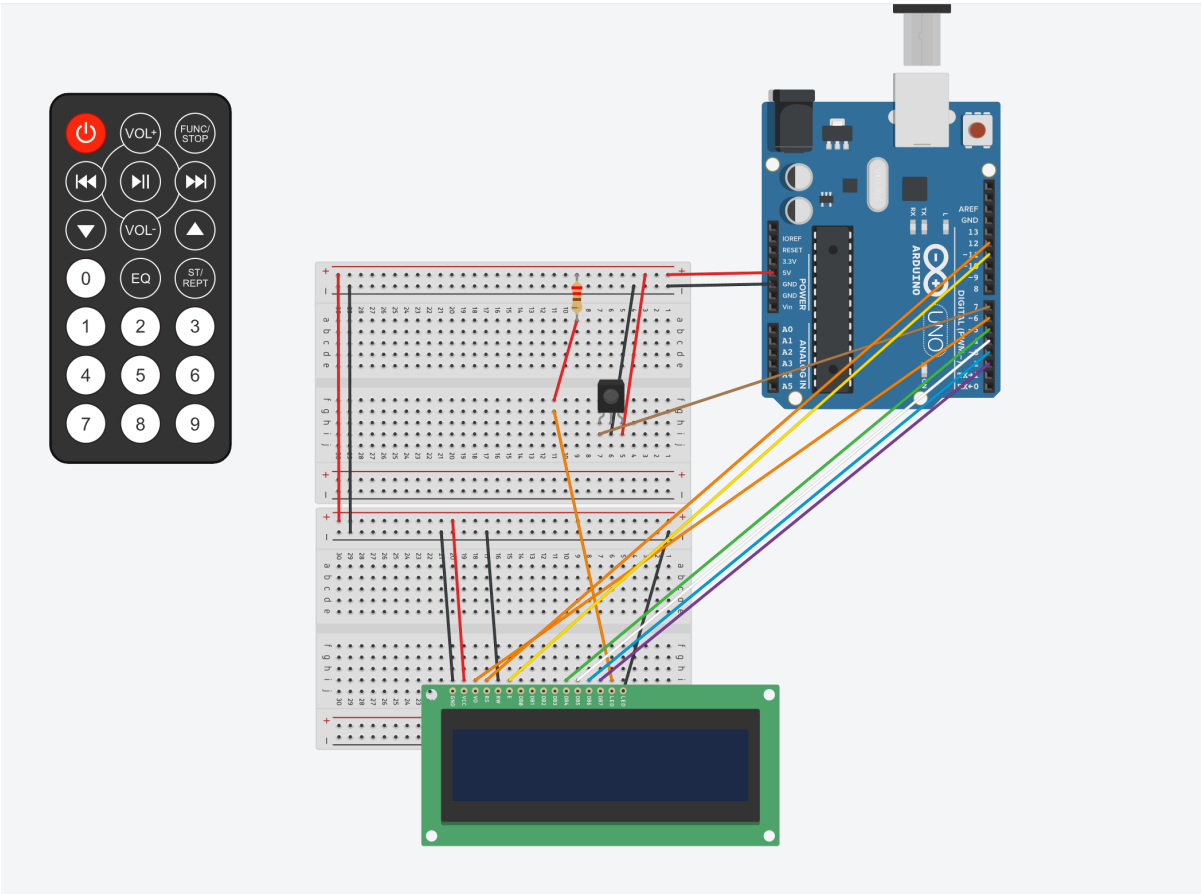
Introduction

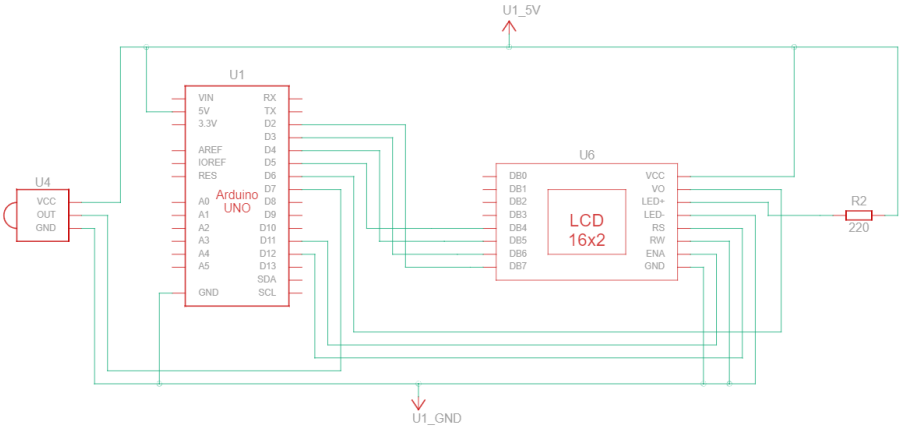
A password manager (similar to a physical crypto wallet), that can be interacted with using a remote. The user can save multiple passwords by using the menus of the wallet. On the LCD are displayed the saved passwords and the status. Passwords can be added, deleted and saved to the EEPROM. It is intended to be used with highly sensitive passwords such as banking passwords (card pins, etc) that should never touch the internet.

General description

Hardware Design

Parts list: Arduino UNO R3, Infrared receiver, remote control, Alphanumeric LCD, breadboard, resistor 1k, jumper wires





Software Design

Developed using CLion (+PlatformIO) and Arduino IDE (to upload the code to the Arduino). On boot, the wallet displays a seed, that is used in the companion app to retrieve the password. After the password is entered in the wallet, the user can scroll trough the passwords, edit or delete a password or save a new one. After performing edits on the passwords, they can be saved by pressing FOL+ while in ADD_PASSWORD mode (mode 4). This is done in order to prevent too many writes to the EEPROM (after 100000 writes, the EEPROM will become damaged, hence an 'autosave' feature is not provided).

Modularity and reusability were key while writing the code for the wallet, in order to allow for ease of expansion, and addition of new features.

In order to easily debug the code and to trim'spammy' logs in the case that not all of them are necessary, 3 logging levels exist (No logs, warning, warning & info), and 2 type of logs (Info & Warning). Logging is handled by:

```
void printDebugInfoMessage(const String &message)
```

and

```
void printDebugWarningMessage(const String &message)
```

In order to define the logging level, debuggingLevelEnabled from currentState must be changed to one of: NO_LOGS, WARNINGS, INFO.

To decode the signal from the remote, the

```
String decodeRemoteCode(uint32_t code)
```

function is used, that contains a switch that returns the letter that was pressed. I did not use a HashMap in order to save memory.

Each remote code is written as a define at the top of the program, in order to easily update the code in case that a different remote is used. Also, all other constants such as pins, the password cover, and the logging levels are defined in the same way.

The wallet has a custom built state management system, in order to not refresh the display every time loop() is ran. The state of the wallet is kept in a struct, and if any field in that struct is changed, a state changed is triggered, and the display is refreshed. The state is handled by:

```
void setState(int32_t state, int32_t debuggingEnabled, const String &display,
const String &displayRow2, const String &input = currentState.input, const
String &password = currentState.password, const String &secrets =
currentState.secrets)
```

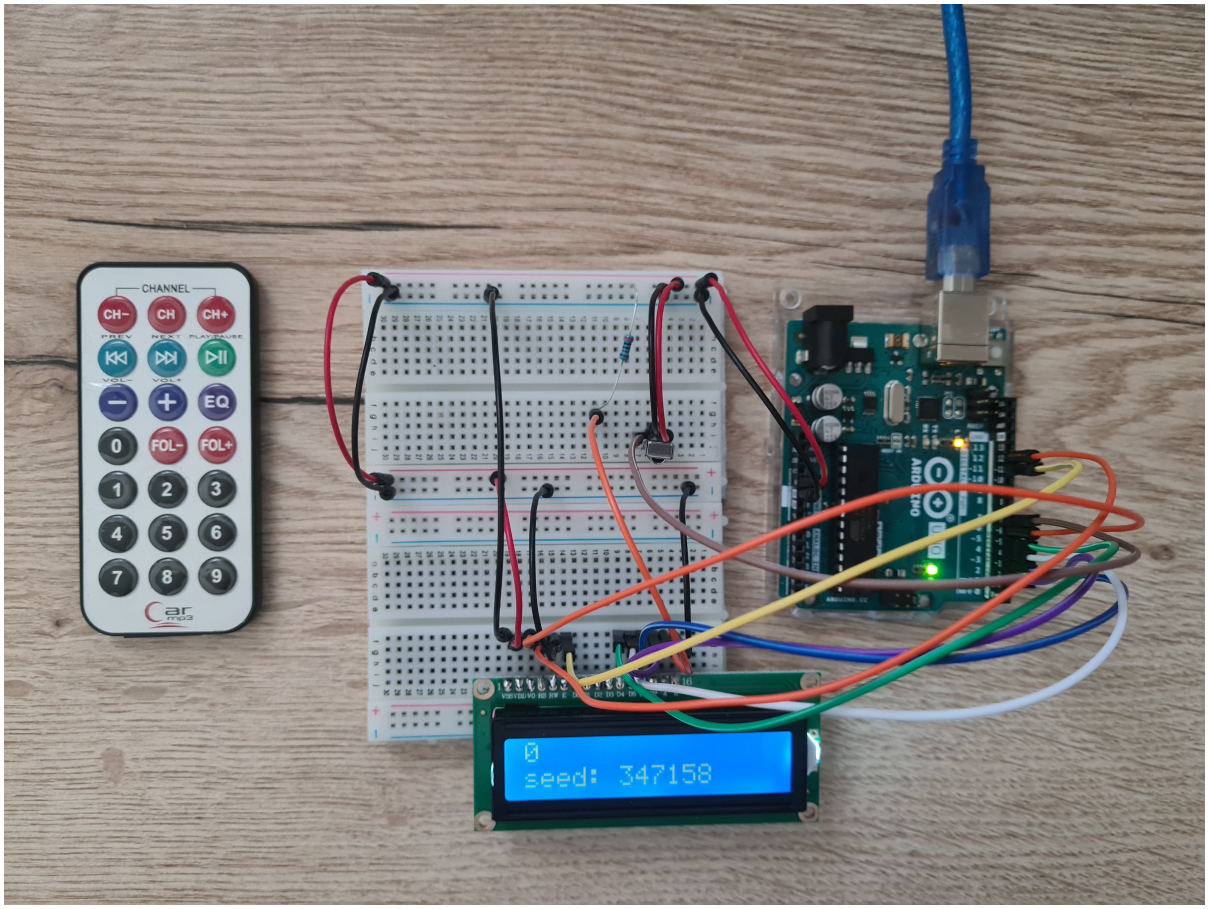
In order to write to the display, a custom function is used, that handles a few write modes, such as appending text and overwriting, and also the line on which the text is going to be written to. Writing to the display is handled by:

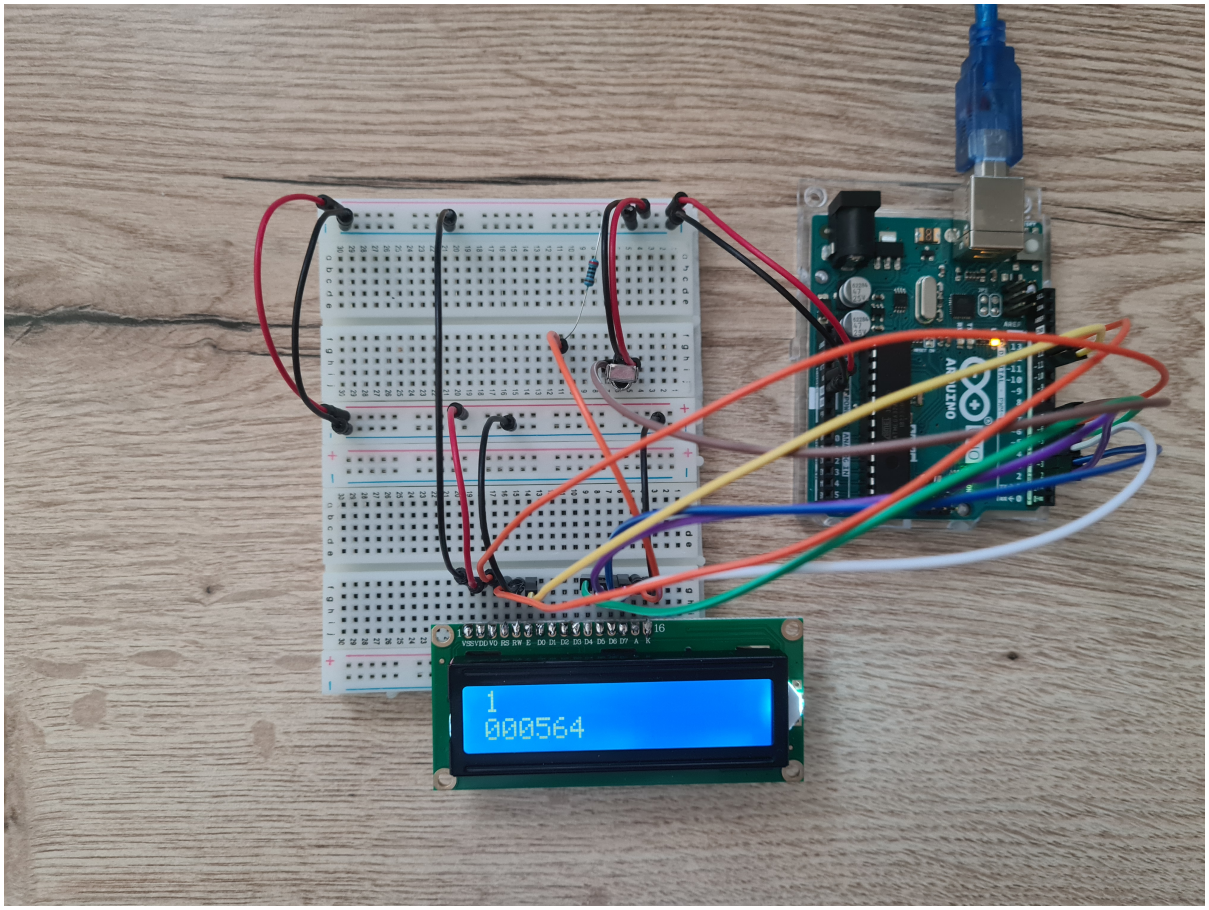
```
void writeToDisplay(const String &word, bool append = true, bool firstRow =
true, bool appendSpace = true)
```

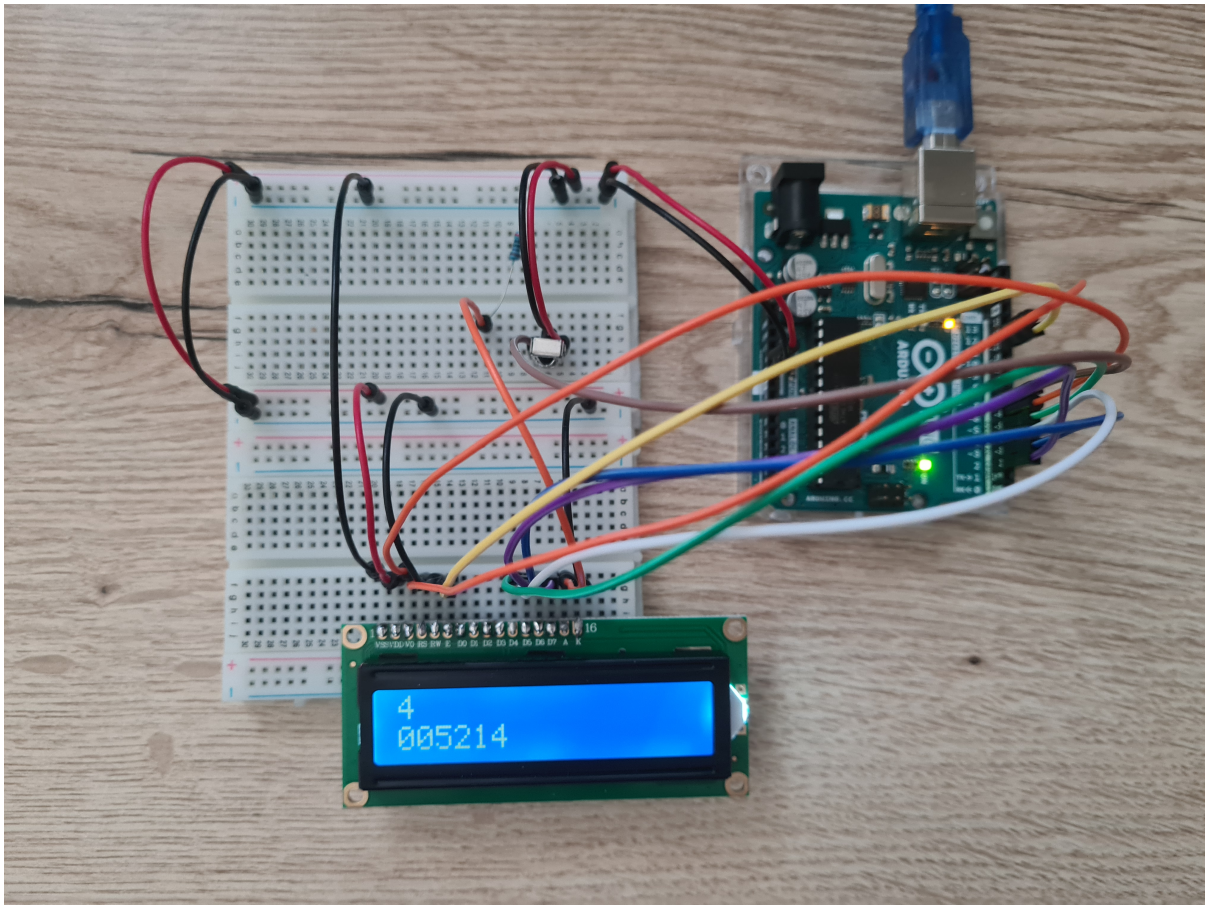
The companion app is a cross-platform app, written in Flutter (a framework for building cross-platform apps developed by Google, on top of Dart) that allows the user to get the wallet password and view quick tutorials about each mode of the wallet.

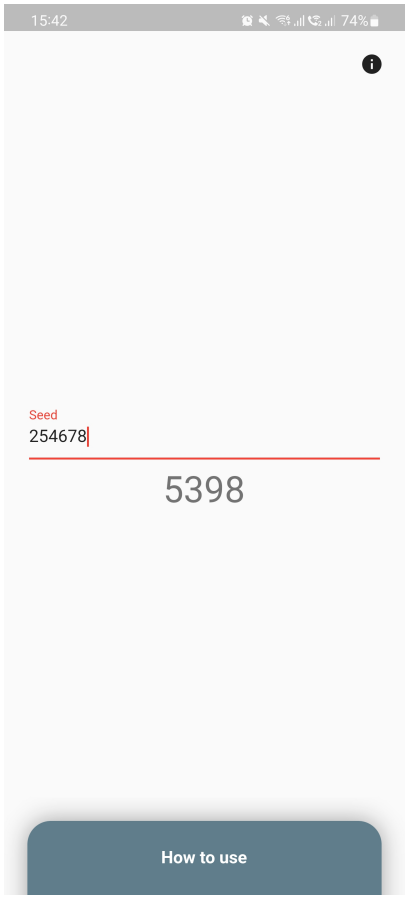
Results

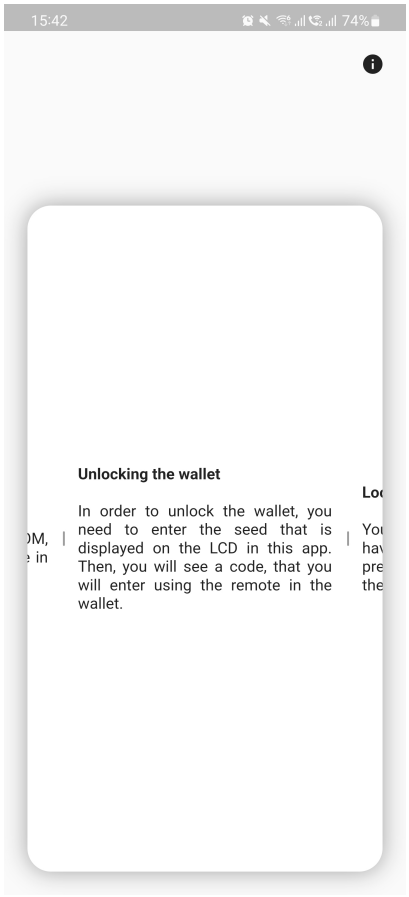
The device works as expected, passwords are saved and retrieved from EEPROM, there seem to be no bugs in the software, neither for the Arduino nor for the companion app.

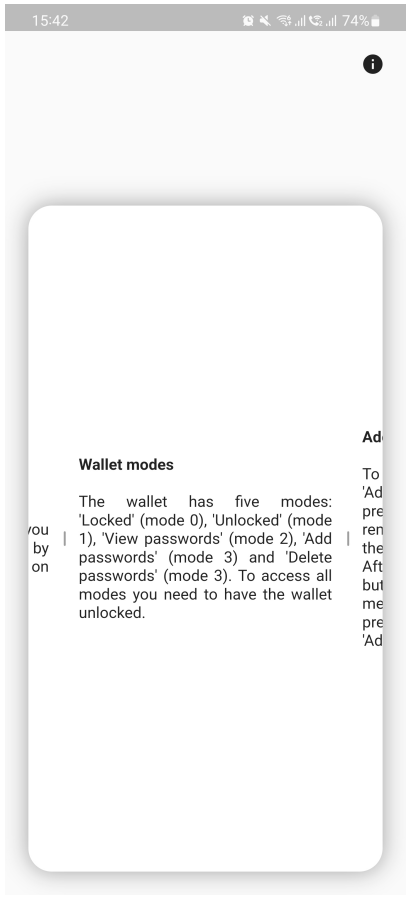


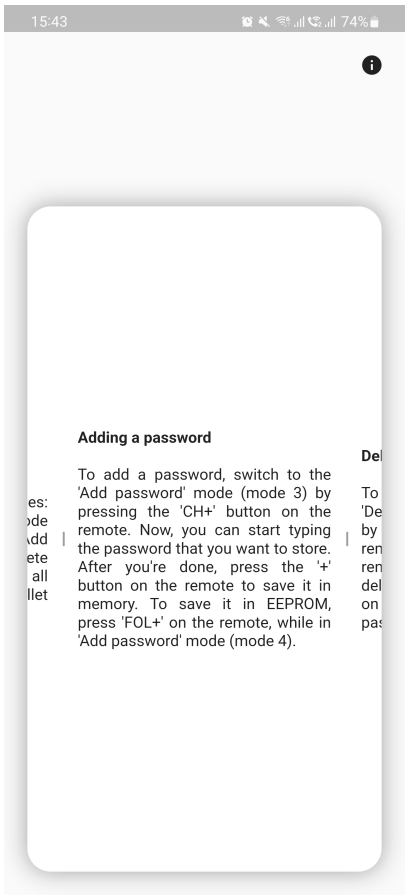

1
000564

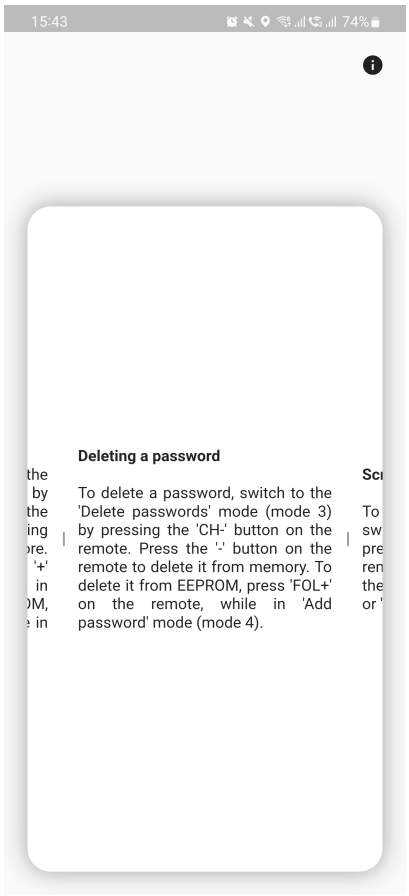

4
005214

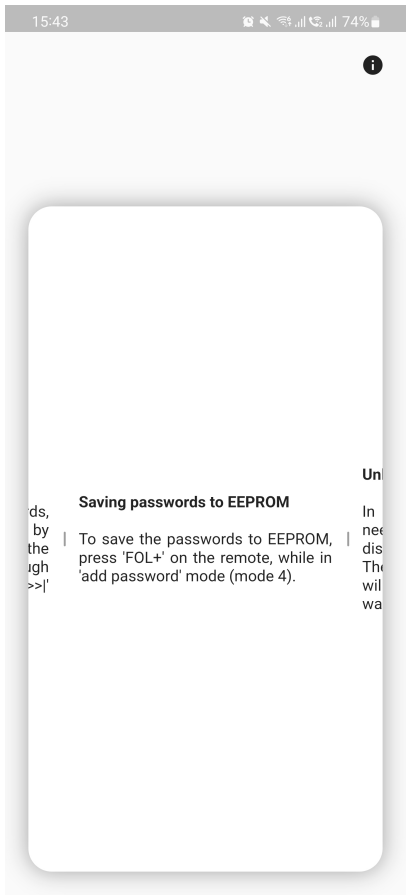

15:42
74%
Seed
254678
5398
How to use


15:42
74%
Unlocking the wallet
In order to unlock the wallet, you need to enter the seed that is displayed on the LCD in this app. Then, you will see a code, that you will enter using the remote in the wallet.


15:42
74%
Wallet modes
The wallet has five modes: 'Locked' (mode 0), 'Unlocked' (mode 1), 'View passwords' (mode 2), 'Add passwords' (mode 3) and 'Delete passwords' (mode 3). To access all modes you need to have the wallet unlocked.


15:43
74%
Adding a password
To add a password, switch to the 'Add password' mode (mode 3) by pressing the 'CH+' button on the remote. Now, you can start typing the password that you want to store. After you're done, press the '+' button on the remote to save it in memory. To save it in EEPROM, press 'FOL+' on the remote, while in 'Add password' mode (mode 4).


15:43
74%
Deleting a password
To delete a password, switch to the 'Delete passwords' mode (mode 3) by pressing the 'CH-' button on the remote. Press the '-' button on the remote to delete it from memory. To delete it from EEPROM, press 'FOL+' on the remote, while in 'Add password' mode (mode 4).


15:43
74%
Saving passwords to EEPROM
To save the passwords to EEPROM, press 'FOL+' on the remote, while in 'add password' mode (mode 4).

Conclusions

This was the first project of this type that I've made, it helped me better understand the labs, and how hardware works.

Download

You can find the code for the Arduino here https://github.com/xrusu/ma-lab and for the companion app here https://github.com/xrusu/ma-lab-companion, or you can download the archives from here:

Wallet code archive

Companion code archive

Log

20/05/2022: Added more details about 'Software' on wiki

09/05/2022: Wiki brush-up

08/05/2022: Code brush-up

07/05/2022: Started writing the software

22/04/2022: Wiki page

21/04/2022: Project selection

Bibliography/Resources

LCD Datasheet

IR Receiver Datasheet

Dart

Flutter

Export to PDF