Safe-critical Smart Toaster

Petr Kucera

May 24, 2024

1 Project description

1.1 Target and motivation

Do you have a regular toaster at home? What if you could turn it into a smart device that prepares warm toast for you at a precise time? So when you go to bed at night, you prepare everything, and in the morning, you have warm toast ready on your plate.

The goal of my project is to create a device that will allow controlling the toaster through a user interface. The device will enable delayed start.

1.2 Base architecture description

The device is based on STM32H747I-DISCO, and I use a relay to control the toaster's power.[1] Users control the device through a touchscreen display. The device run internal timers to count down the time for the toaster to start and how long it should run. To ensure safety, the microcontroller utilize both processors, which mutually check each other in all their activities.

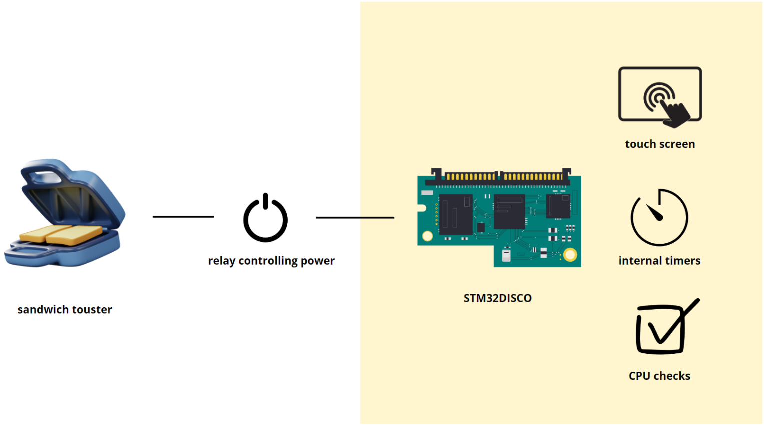

Figure 1: Ilustration schema of project

[1] For example, this one: https://dratek.cz/arduino/7522-rele-ssr-solid-state-ssr-40da-380v.html.

2 Achitecture

2.1 Wiring

The MCU controls relay using GPIO output pin and the pin is protected by 470 Ω resistor. The input control voltage of relay is 3-24 VDC and output voltage of relay is 24-240 VAC. For higher loads, it's recommended to add passive or active cooling to the relay. However, in our case, we won't utilize it because we'll assume that the toaster will only be used for short periods at a time.

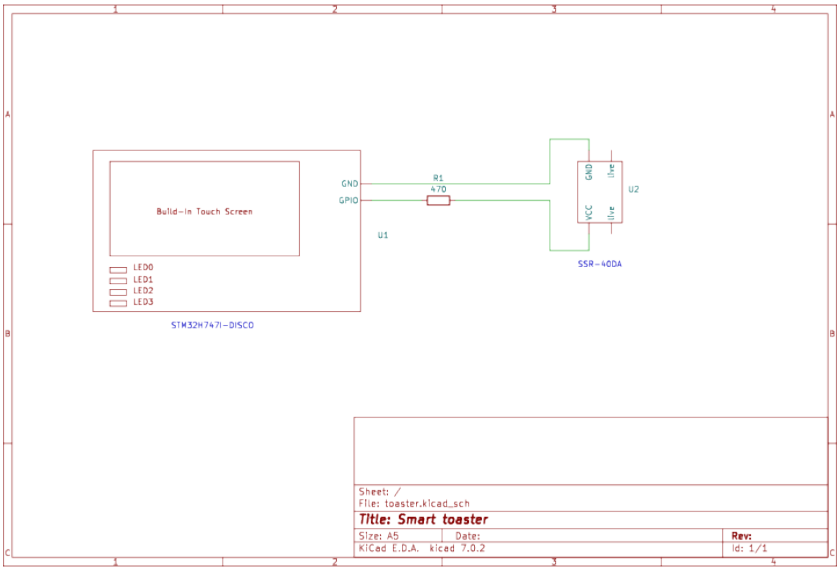

Figure 2: Schema of wiring MCU and Relay

2.2 User interface

The user interacts with the device using an LCD touchscreen display.

2.2.1 Controling

The display will contain one screen that will change according to the remaining time and the selected settings. On the screen you can:

- start or stop the toaster
- set a delayed start
- cancel a delayed start
- monitor the time until the toaster starts
- monitor how long it has been running

2.2.2 Mockup

The interface consists of four basic parts:

1. status message
2. progress bar
3. left control button
4. right control button

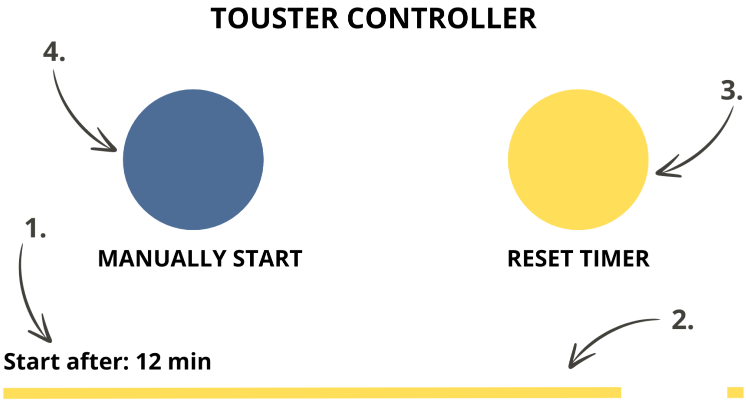

Figure 3: The UI proposal with parts

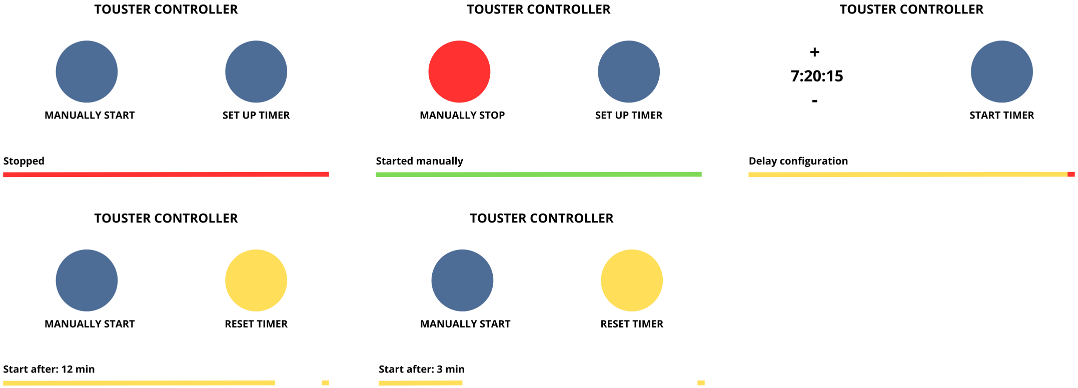

Figure 4: Display status

The control consists of several states. All of them can be viewed below (Figure: 4), where their diagrams are prepared. The user interface uses four status colors:

- **RED** - stop

- **YELLOW** - waiting or config
- **DARK BLUE** - start
- **GREEN** - process is running

2.3 Program diagram

The entire process is shown in the figure 5.

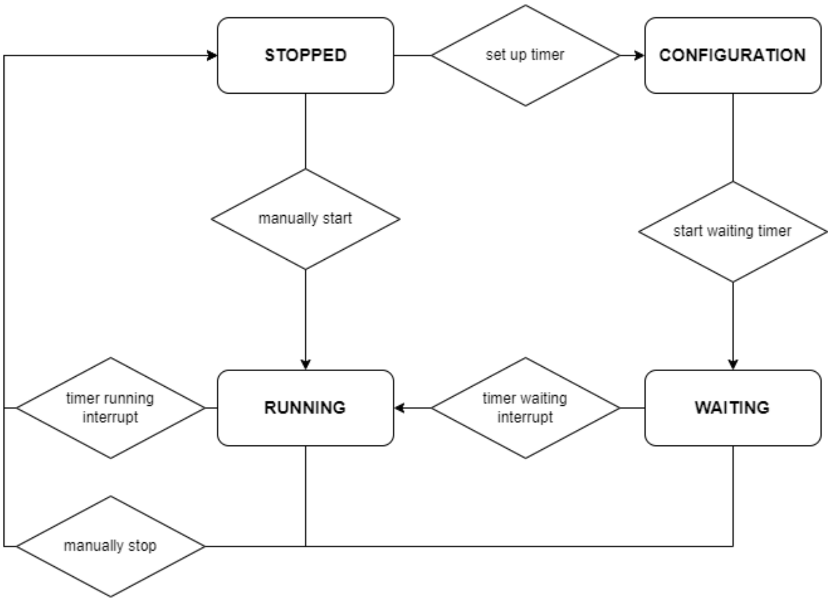

Figure 5: Status diagram

The device can be in 4 states:

1. **STOPPED** - default state, nothing is happening
2. **CONFIGURATION** - setting the countdown timer length, i.e., how long the toaster should be started
3. **WAITING** - the timer is counting down and waiting for the waiting timer interrupt
4. **RUNNING** - the toaster is on

The program utilizes 2 timers:

1. **waiting timer** - used for counting down the time until the toaster itself should start
2. **running timer** - terminates the toaster operation, for safety reasons, it cannot be modified by the user

3 Software

For implementation, I'm using tools from STM, specifically STM Cube and HAL libraries. The code is written in the C language. For versioning, I use a repository[2] that I use for versioning projects created within this subject.

4 Hadware requirements

For the project, primarily 2 basic components are needed:

- MCU (STM32H747I-DISCO) - microcontroller for device control and management
- Relay (SSR-40DA) - for switching and controlling the toaster's power

Additionally, a wire will be required to connect to the relay. I will likely use an extension cord that will be controlled by the relay to avoid damaging the toaster's power source.

[2] Subject repository https://github.com/petrkucerak/AVS.