# COMPANY AD SCREEN

You have been tasked with writing the application logic for the Ad Screen used by your company and finishing the driver initialization of the LCD. The company's ad screen prints out the advertiser's name and then their ad message.

# Sources

For this project you will use the Requirements Specification down below, a provided Wokwi sketch and the template codebase.

Link to template is here: https://github.com/lafftale1999/cpp_embedded_assignment

Link to the sketch is here: https://wokwi.com/projects/445497618841194497

# Requirements Specification

The following requirements are specified by the product manager.

## Functional Requirements

### 1. Program Logic

The program should always run while powered on. When powered on the program should:

1. Choose the next advertiser
2. Show the company name for 5 seconds
3. Show one of the company's ad messages for 10 seconds

### 2. Advertisement logic

**2.1.** Paid weighting: probability of the next selected advertiser co-relates to how much they've paid for their ads / the total ad sum.

**2.2.** Companies will have different amounts of ad messages. Make sure to rotate their ads independently and that all of their ads will have been shown at some point. This should be sequential and not randomly generated.

## 3. Proper initialization of hd44780

The team has already written a part of the driver for the company's LCD driver. You need to finish the initialization in the hd44780 constructor.

**lcd_driver.cpp**

```
hd44780::hd44780() {
    /* TASK
    Set up all the pins used for the program. Look at the wokwi
    sketch and compare to a pinout diagram for Arduino Uno R3.
    Use the definitions from lcd_driver.hpp
    */

    /* TASK
    Set the pins LCD_E and LCD_RS to low.
    */

    [ ... Initialization Sequence ...]
}
```

## 4. Create a blink effect

Some of the companies want to blink their ad messages, therefore we need to implement a blinking effect on the *hd44780*. Make use of the `#defines` from `lcd_driver.hpp` and finish the `blink_text()` method.

**lcd_driver.cpp**

```
void hd44780::blink_text(const char *text) {
    /* TASK
    Implement a blinking effect where the program first writes out the
text
    and then blinks the message.
    */
}
```

# Non-functional Requirements

Non-business logic requirements for the system.

### 1. No heap allocated memory

Because of the small RAM, the program should only use the stack.

### 2. No blocking delays

The program should always be able to call an ISR at any given time. Use the function from `millis.h` called `millis_wait_ms()`

# Deliverable Requirements

Requirements on how the codebase should be delivered.

### 1. Folder structure

The following folder structure should be upheld in this project.

```
ProjectName
├── Makefile
├── main.cpp
├── include
│    └──header files
└── src
     └──source code files
```

### 2. Code Separation

Build cohesive files where every file represents a specific functionality for the program.

# Advertisers

The following companies are paying customers that want to show their ads on the screen:

## 1. CoolCars LLC

- **Company name:** CoolCars LLC
- **Paid amount:** 5 000 SEK
- **Messages:**
    - *Drive me crazy!* `plain`
    - *30-day moneyback guarantee* `plain`

## 2. Pie People

- **Company name:** Pie People
- **Paid amount:** 2 500 SEK
- **Messages:**
    - *Making people float* `blink`
    - *They are hot and jammy* `plain`

## 3. Washomania

- **Company name:** Washomania
- **Paid amount:** 4 900 SEK
- **Messages:**
    - *Clean customers happy customers* `blink`

## 4. Holy Burgers

- **Company name:** Holy Burgers
- **Paid amount:** 1 750 SEK
- **Messages:**
    - *Free drinks included in menu* `plain`
    - *Burgermania! 2 for 59 SEK* `blink`
    - *Badabim... Bada BURGER!* `blink`

## 5. Gymbo James

- **Company name:** Gymbo James
- **Paid amount:** 6 700 SEK
- **Messages:**
    - *No more excuses! Only GAINS!* `blink`
    - *Pro Gymbono - 14 days free* `plain`

# Grading

## 1. Deliverable

- The complete program must be submitted via a private GitHub repository and shared with the examiner (@lafftale1999). There should be a .hex file ready to run the program available in the GitHub repository.
- The student must also perform a live demonstration in a small group, where the program is run and key features are shown.
- During the demo, the student must be able to explain key design and implementation decisions, as well as answer questions regarding the codebase.

## 2. Grading

This assignment can only be failed (IG) or passed (G). To pass:

- All functional and non-functional requirements are met to a level that demonstrates the intended behavior and stability of the system.
- The system runs without crashes.
- The student can clearly explain how the program works.
- The code should be well-structured, readable and logically divided per the specified deliverable requirements.

# Learning objectives

The following learning objectives of the course is handled in this assignment:

- Explain bit manipulation, operator overloading, exception handling, compilation, linking, and execution
- Describe Makefiles and tools, including their functionality
- Provide an overview of low-level and resource-conscious programming
- Develop applications in C++ at a specialized level while utilizing relevant functions, libraries, and templates
  Show testing, debugging and troubleshooting of applications in C++
- Apply and compare procedural and object-oriented programming techniques
- Handle and comprehend bit manipulation and operator overloading, manage exceptions
- Apply analysis and design in embedded system solutions using C++
- Communicate commitments and solutions in the development of applications in C++ at a professional level
- Independently develop and debug software applications in C++ based on given requirements
- Independently identify and address a problem and be able to choose the appropriate tools and methods to solve the issue at hand
- Independently drive development in C++ by selecting suitable tools and methods like CMakeFiles.