Project Report in ET095G: RANPW

Full Name: Nicolas Ferraresso **Date:** Octorber 4, 2023

Table of Contents

1. Introduction

The RANPW project aims to create a versatile, Tamagotchi-inspired tool enriched with a selection of games (LCatch, Pong, and LockY) and productivity applications (Cronometer and Termometer). This dynamic platform encourages high modularity, allowing individuals with basic C++ knowledge to develop and integrate their custom apps and games. RANPW operates on the mbed NXP LPC1768 microcontroller, coupled with the mbed application board.

User Interaction Components

- **LCD (C12832):** The 32x128 pixel LCD display serves as the primary visual interface for all interactions within the application.
- **Joystick:** The joystick plays a pivotal role in navigating through all application interfaces.
- **Potentiometers:** Both potentiometers enhance specific user interactions, with one notably enhancing gameplay in the LockY game.
- **Temperature Sensor (LM75BD):** The LM75BD temperature sensor is utilized in the "Termometer" application to capture environmental temperature data.
- **Accelerometer (MMA7660):** The MMA7660 accelerometer contributes significantly to user interactions, particularly in the Hell and Pong games.

2. Implementation Analysis

2.1 Hardware

2.1.1 LCD (C12832)

The C12832 LCD is controlled using a customized C12832.h library, ensuring compatibility with the project's software ecosystem. It is used to display game graphics, app interfaces, and visual feedback.

2.1.2 Joystick

The joystick is managed through APIs provided by mbed.h and is used for navigation, menu selection, and game control. It facilitates smooth user interaction.

2.1.3 Potentiometers

Potentiometers are utilized for game controls and interactions, offering analog input. mbed.h APIs are employed to handle these components, enhancing gameplay experiences.

2.1.4 Temperature Sensor (LM75BD)

The LM75BD temperature sensor is integrated into the "Termometer" application using the LM75B.h library, with slight modifications to ensure compatibility. It captures temperature data and presents it to users in a visually appealing manner.

2.1.5 Accelerometer (MMA7660)

The MMA7660 accelerometer is employed in the Hell and Pong games, capturing user input for game control. The MMA7660.h library is adapted to align with project requirements.

2.1.6 Local storage

Local storage is essential for saving user data and application state. DataSaver class handles data saving and loading operations, ensuring that critical data, such as high scores and Kraken's details, is persistently stored in the device's internal memory.

2.2 Software

2.2.1 Thread overview

At startup, RANPW initializes three distinct threads to manage various aspects of its functionality:

Thread 1: User iterface and interaction

This thread is responsible for overseeing the entire user interface and facilitating interaction with end-users. It dynamically creates and initializes instances of all apps and games while orchestrating their display on the LCD screen. Further insights into this thread's workings can be found in the Ranpw.h and Ranpw.cpp files.

Thread 2: Kraken's management

Thread 2 operates on a straightforward principle, employing basic rand() functions to introduce wait times of 1 to 5 minutes. During these intervals, it randomly decreases the pet's hunger by a value ranging from 1 to 5. Should the hunger level plummet to zero, the pet's life diminishes at varying rates —1, 2, or 3 points per minute. Conversely, if hunger surpasses 20, the pet gains 1 life point. Notably,

both life and hunger levels have a cap of 25. Detailed insights into this thread's functionality are available in the live() function within Kraken.cpp.

Thread 3: Data saving management

Thread 3 operates with a clear focus on data integrity. It consistently monitors the data stored in the members of the ConcurrentData structure. This structure is vital for ensuring safe data access across multiple threads. When disparities between the stored data and in-memory data are detected, Thread 3 takes action, overwriting the data and saving the respective file in the device's internal memory.

2.2.2 Class overview

In the process of realizing the RANPW project, I've meticulously crafted different classes to lay the foundation for its functionality and modularity. Each of these components plays a specific role in the project's architecture:

ConcurrentData

This essential serves as the backbone for managing concurrent data access across multiple threads. It ensures data integrity and consistency, preventing conflicts and race conditions.

DataSaver

The DataSaver class is responsible for handling data saving and data loading operations, ensuring that changes to critical data are persistently stored in the device's internal memory.

Ranpw

The Ranpw class serves as the central hub for user interface management. It oversees the creation and initialization of app and game instances, orchestrates their display on the LCD screen, and manages user input via various hardware interfaces.

Kraken

The Kraken class embodies the behavior and lifecycle of the virtual pet, monitoring key metrics such as hunger and life. It also simulates the pet's interactions with the user.

Bag

The Bag class manages the virtual inventory system, allowing users to collect and interact with food items within the application.

Cronometer

The Cronometer is a specialized application designed to provide users with a versatile stopwatch.

Termometer

This class allow users to monitor temperature levels in their environment. What sets this component apart is its ability to visualize temperature data through a graph.

LCatch

LCatch is one of the interactive games featured in RANPW. It provides an engaging experience where users can test their reflexes and coordination. What sets LCatch apart is its incremental speed feature, where the game progressively increases in difficulty as players successfully capture objects.

LockY

LockY is another game that offers users a unique challenge, in this game the user must, using the two potentiometers, align two lines with the holes of a third line

Pong

In Pong, player use the accelerometer as a unique input method to control the paddle in a classical Pong game.

Hell

Hell is a captivating minigame tucked within the RANPW project, offering a distinctive twist on the gaming experience. Unlike other games, Hell only surfaces when the Kraken's life reaches zero, triggering the Kraken's respawn process. Distinguished by its lack of a tutorial, Hell immerses players in a realm where they need to discover how to play by hiself.

3. Discussion

3.1 Optimization

Given the scope of this project, ensuring optimal resource utilization became a crucial aspect of development. To achieve this, I delved into various optimization techniques to maximize the efficiency of the system. Several key strategies were implemented:

- **Global Initialization:** To minimize memory usage, I opted to initialize certain variables and resources globally in the main function. This approach prevented them from being allocated in the .bss section and instead placed them in the stack, reducing the overall memory footprint.
- **Sensor Initialization:** I centralize sensor initialization within a dedicated class, initializing sensors only once and passing their pointers where needed throughout the project. This approach eliminates redundant sensor reinitialization, conserving more memory
- **Scoped Game Initialization:** To further optimize memory allocation, I strategically limited the scope of game initialization. Games are initialized only when required, ensuring they occupy memory space only when actively in use.
- **Pointer Management:** Ensuring the absence of dangling or unclean pointers was paramount.

3.2 Future plan

In the future we plan:

3.2.1 Secure data storage

Enhance data security by implementing minor encryption measures for all saved data. This safeguard prevents unauthorized user modifications and ensures data integrity.

3.2.2 Code refactoring

Optimize and streamline the codebase with the following refactorings:

Game as superclass

Consider redesigning all the game components to have a common superclass, promoting code reusability and consistency among different games.

App as superclass

Similarly, create a superclass for apps within the project to standardize their structure and behavior, making it easier to add new apps.

Doxygen documentation

3.2.3 Add more app

3.2.4 Music to evrything

Integrate music and sound effects into various aspects of the project to enhance the overall user experience and immersion.

Setting for music

Implement user settings that allow users to control music and sound effects, granting them the option to enable or disable audio as per their preference.

3.2.5 Hardware

3D printable case

Design and create a 3D-printable case for the project.

Custom PCB

Make a custom PCB to integrate all the need hardware in a more compact manner.