Documentation for the Embedded Challenge (Team 29)

Overview

The Embedded Distance Tracker is a sophisticated project meticulously crafted for the STM32F429I Discovery board. This system integrates seamlessly with a gyroscope sensor to capture angular velocity, converting it into linear velocity, and ultimately providing insights into the total distance traveled by a user over a precisely monitored 20-second interval. This detailed documentation aims to provide an exhaustive understanding of the code's structure, functionality, and usage.

Motivation

The primary motivation behind the Embedded Distance Tracker is to create an intelligent and versatile embedded system capable of tracking movement and providing valuable data related to speed and distance. This project serves as an excellent educational resource for individuals interested in embedded systems, sensor interfacing, and real-time data processing.

YouTube Video Tutorial

To facilitate an intuitive understanding of the project, a comprehensive video tutorial is available on YouTube. This visual guide covers the project's setup, operation, and key features. Watch the tutorial: The Embedded Gyrometer - "The Need For Speed" | Group 29 | RTES Fall 2023 | Video Presentation.

Hardware Requirements

To replicate this project, it is essential to gather the following hardware components:

- **STM32F429I Discovery board**: The central microcontroller platform providing the necessary processing power.
- **Gyroscope sensor**: Responsible for capturing angular velocity data.
- **LCD_DISCO_F429ZI display**: A display interface for real-time data visualization.
- **USB-to-Serial converter**: Facilitates debugging and communication with an external device.
- **Onboard LEDs (LED1 and LED2)**: Used for visual indicators during operation.

Dependencies

The Embedded Speed Tracker relies on several external dependencies and libraries:

- **mbed.h**: The mbed platform header file, providing essential functionality for the microcontroller.
- **math.h**: A standard C library that facilitates mathematical computations.
- **gyroscope.h**: A custom header file defining functions and configurations for gyroscope interfacing.
- **configuration.h**: A custom header file containing configuration settings and constants.
- **LCD_DISCO_F429ZI.h**: A custom header file for interfacing with the LCD display.

Global Variables

The code employs a set of global variables to manage and store crucial data:

- **angVelData[40][3]**: A two-dimensional array storing angular velocity data for the X, Y, and Z axes over a 20-second interval.
- **linVelData[40][3]**: Another two-dimensional array holding linear velocity data for the X, Y, and Z axes over the same time span.
- **Index**: A counter tracking the number of intervals processed during the system's operation.
- **totalDistanceCovered**: A variable that accumulates the total distance covered by the user.

Functions

The project incorporates a variety of functions to facilitate different aspects of its operation:

1) `Gyro_Init()`

- **Description**: This function initializes the gyroscope, configuring its settings and returning a unique identifier.
- **Returns**: An integer representing the gyroscope's identification.

2) `Gyro_Get_XYZ(float xyz[])`

- **Description**: Retrieves the X, Y, and Z axis values from the gyroscope and stores them in the provided array.
- **Parameters**: xyz - An array to store the gyroscope's axis values.

3) `setupGyro()`

- **Description**: Configures the Serial Peripheral Interface (SPI) communication for the gyroscope, setting data format and frequency.

4) `processGyroData()`

- **Description**: Processes the raw gyroscope data, calculates linear velocity, and stores the information in global arrays.

5) `initializeLCD()`

- **Description**: Initializes the LCD display with relevant project information, such as the team name and project title.

6) `displayTimeOnLCD()`

- **Description**: Dynamically updates the LCD display with the real-time count of elapsed seconds during system operation.

7) `displayDistanceOnLCD(double totalDistanceCovered)`

- **Description**: Displays the total distance traveled by the user on the LCD, considering calibration factors for accuracy.

8) `calculateDistance()`

- **Description**: Computes the linear velocity and total distance covered over the entire 20-second interval, incorporating calibration and scaling factors.

9) `main()`

- **Description**: The main function orchestrating the overall system operation. It initializes the gyroscope, sets up the LCD, processes gyroscope data, and continuously updates the LCD display with real-time information.

How to Use

To utilize the Embedded Speed Tracker effectively, follow these step-by-step instructions:

Hardware Setup: Connect the gyroscope, LCD display, and LEDs to the STM32F429I Discovery board as per the hardware requirements.
Code Deployment: Flash the provided code onto the STM32F429I Discovery board using an appropriate programming tool.
System Execution: Run the embedded system.
Real-time Monitoring: Observe the LCD display for real-time updates on elapsed time and LED1 blinking.
Post-20-Second Analysis: After 20 seconds, observe the LCD display for the total distance covered. LED1 remains ON, and LED2 turns ON.

Calibration and Customization

For optimal performance, consider the following:

- **Configuration Settings**: Adjust parameters in the configuration.h file to match your specific hardware specifications.
- **Calibration**: Calibration factors are implemented for enhanced accuracy. Adjust them if necessary.
- **Scaling**: Scaling factors are applied to achieve better accuracy. Modify them based on your requirements.

Additional Notes

- The project is designed with educational purposes in mind, providing insights into embedded systems, sensor interfacing, and real-time data processing.
- This comprehensive documentation aims to serve as a reference for users exploring the intricacies of the Embedded System Distance Tracker.

Contributors

The development of this project was made possible through the efforts of Group 29 members:

- dk4852
- tx701
- ka3210
- hs5472

License