1 Changelog

This section provides a summary of changes made to the Balloon project.

1.1 Version 1.0.0 (4/14/2025)

1.1.1 Features

- Added `main.cpp` in the `src/` folder for core functionality.
- Added `testing.cpp` in the `src/` folder for altitude data processing.
- Added `altitude_data.py` and `altitude_data.txt` in the `Altitude Testing/` folder for altitude testing and data storage.
- Added `Adafruit_Examples.cpp`, `Cpp_Example.cpp`, and `Prof_Cpp_Example.cpp` in the `Examples/` folder for example implementations.
- Added `README` files in the `include/` and `lib/` folders for documentation.
- Added `Meeting Notes/` and `Project Notes/` in the `Notes/` folder for project-related notes.
- Added `changelog.txt`, `todo.txt`, and `LaTeX/` folder in the `Documentation/` folder for project documentation.
- Added `.vscode/` folder with configuration files for Visual Studio Code.
- Added `platformio.ini` for PlatformIO project configuration.

1.1.2 Changes

- Reorganized files into a structured folder hierarchy for better project management.
- Modified `testing.cpp` to include functionality for reading and processing altitude data from `altitude_data.txt`.

1.1.3 Bug Fixes

- None.

2 `main.cpp` Overview

The `main.cpp` file is the core of the Balloon project, responsible for reading altitude data from the BMP280 sensor, processing it, and calculating key metrics such as velocity, acceleration, and jerk. It also controls the LED indicators based on the calculated jerk value.

2.1 Functionality

The `main.cpp` file performs the following tasks:

- Initializes the BMP280 sensor and configures it for oversampling and filtering.
- Reads altitude data from the BMP280 sensor at a 1 Hz sampling rate.
- Processes the altitude data to calculate velocity, acceleration, and jerk.
- Outputs the calculated jerk value to the serial monitor.
- Controls the state of two LEDs based on the jerk value:
  - Green LED: Indicates positive jerk.
  - Red LED: Indicates negative jerk.
  - Both LEDs off: Indicates zero jerk.

2.2 Key Calculations

The highlighted section of the code calculates velocity, acceleration, and jerk using the following equations:

2.2.1 Velocity

Velocity is calculated as the change in altitude over time (Δt):

$$v_1 = \frac{\text{altitude}_1 - \text{altitude}_0}{\Delta t}, \quad v_2 = \frac{\text{altitude}_2 - \text{altitude}_1}{\Delta t}, \quad v_3 = \frac{\text{altitude}_3 - \text{altitude}_2}{\Delta t}$$

2.2.2 Acceleration

Acceleration is calculated as the change in velocity over time (Δt):

$$a_1 = \frac{v_2 - v_1}{\Delta t}, \quad a_2 = \frac{v_3 - v_2}{\Delta t}$$

2.2.3 Jerk

Jerk is calculated as the change in acceleration over time (Δt):

$$\text{jerk} = \frac{a_2 - a_1}{\Delta t}$$

2.3 LED Behavior

The LEDs are controlled based on the calculated jerk value:

- **Positive jerk:** The green LED is turned on, and the red LED is turned off.
- **Negative jerk:** The red LED is turned on, and the green LED is turned off.
- **Zero jerk:** Both LEDs are turned off.

2.4 Example Output

The `main.cpp` file outputs the current altitude and jerk value to the serial monitor in the following format:

```
Altitude: 250.00 | Jerk: 0.50
```

This provides real-time feedback on the altitude and motion dynamics of the balloon.

2.5 Future Enhancements

Planned improvements for `main.cpp` include:

- Adding error handling for sensor failures.
- Supporting additional sensors for enhanced data collection.
- Optimizing the LED control logic for energy efficiency.

3 Balloon Project Report

3.1 Overview

The Balloon Project is a program designed to collect and analyze data from a weather balloon. It utilizes sensors to measure altitude and acceleration, calculates the rate of change of acceleration (jerk) in three axes (X, Y, Z), and controls a NeoPixel LED to visually represent the jerk values. The program also logs the collected data to a CSV file and outputs it to the Serial Monitor for further analysis.

3.2 Key Features

- Reads altitude data from a BMP280 barometric pressure sensor.
- Reads acceleration data from an LSM6DS33 accelerometer.
- Calculates jerk values for the X, Y, and Z axes.

- Controls a NeoPixel LED to display colors based on the direction and magnitude of jerk:
  - X-axis: Red (positive) / Blue (negative)
  - Y-axis: Green (positive) / Yellow (negative)
  - Z-axis: Purple (positive) / White (negative)
- Logs data to a CSV file on a QSPI flash filesystem.
- Outputs data to the Serial Monitor for real-time monitoring.

3.3 Code Functionality

1. **Setup:** The `setup()` function initializes the I2C bus, sensors (BMP280 and LSM6DS33), NeoPixel LED, and QSPI flash filesystem. It also prepares the CSV file for data logging and prints a header to the Serial Monitor.
2. **Main Loop:** The `loop()` function performs the following tasks:
   - Reads altitude and acceleration data.
   - Updates acceleration history arrays for each axis.
   - Ensures sufficient data points are collected before calculating jerk.
   - Computes jerk values using the `computeJerk()` function.
   - Controls the NeoPixel LED based on jerk values and a predefined threshold.
   - Logs data to the Serial Monitor and the CSV file.
3. **Helper Functions:**
   - `scanI2CDevices()`: Scans the I2C bus for connected devices and prints their addresses.
   - `getAltitude()`: Reads and returns the altitude from the BMP280 sensor.
   - `getAcceleration()`: Reads acceleration data from the LSM6DS33 sensor and updates global variables for the current acceleration values.
   - `updateAccelHistory()`: Updates the history arrays for acceleration values by shifting old values and adding new ones.
   - `computeJerk()`: Calculates the jerk values for each axis based on the acceleration history and time interval.

3.4 Data Logging

The program logs the following data to both the Serial Monitor and a CSV file:

- Time (seconds)
- Altitude (meters)
- Acceleration values for X, Y, and Z axes (m/s^2)
- Jerk values for X, Y, and Z axes (m/s^3)

3.5 Conclusion

This program provides a robust framework for collecting, analyzing, and visualizing data from a weather balloon. The use of sensors, real-time data logging, and visual feedback via the NeoPixel LED makes it a versatile tool for atmospheric data collection and analysis.