# Coding Standards

Team Oriented Project
Team D
Data Acquisition and Development Tool

Date:

**October 2024**

**Cooperation with Institute for Mechatronics and Medical Engineering**
**Technische Hochschule Ulm**
Piro, Neltje Emma
Moosbauer, Sebastian

## Why do we need coding standards?

The reason to create this document is to increase the quality of our development and testing process and to also maintain a good standard for the future development of the project

## Best Practices to follow:

### Code Structure and Readability

- **Consistent Naming Conventions:**
  All variables, functions, and classes must follow camelCase for variables and methods, and PascalCase for class names. Use meaningful names to reflect the purpose of the variable or function (e.g., getSensorData() rather than gsd()).
- **Constants:**
  Use all caps for constants with words separated by underscores (e.g., MAX_BUFFER_SIZE).
- **Commenting**: For every function provide comments to explain the purpose of complex logic and for sometimes better understanding, maybe if required write down some explanations for complex lines of codes.

  Use XML comments to provide metadata about public methods and classes. This is especially useful for future developers and helps in generating automated documentation

```
/// <summary>
/// Retrieves sensor data from the ESP32 device.
/// </summary>
/// <param name="sensorId">ID of the sensor to retrieve data from.</param>
/// <returns>A structured SensorData object.</returns>
0 references
public void GetSensorData(int sensorId)
{
    // Method implementation
}
```

- **Scrum Master Notification:**
  When creating a new class, notify the Scrum Master with details regarding its purpose and design. Ensure that the new class does not violate existing design patterns or SOLID principles followed in the project. Each new class should be carefully designed to maintain the integrity of the architecture and uphold the project's adherence to these principles
  - Eventually that new class needs to be added in our diagrams

**Synchronization**

- Scrum Master or the testers need to update the Class Diagram to maintain good synchronization of the development process so later for unit testing each function is checked properly.

**Testing and Reporting**

- Once a User Story is completed, the testing and integration team members are responsible for filling out a report provided by the Scrum Master. This report should include:
    - A summary of the development process.
    - Details of any bug fixes implemented.
    - A comprehensive testing report, including test cases, results, and any issues encountered.
    - The report will provide a clear overview of the User Story's progression and ensure that all relevant information is documented for future reference.