Development Plan

Cyclops Ride Assist: Real-time bicycle crash detection and blindspot monitoring.

Team 9
Aaron Li (lia79)
Amos Cheung (cheuny2)
Amos Yu (yua25)
Brian Le (leb7)
Manny Lemos (lemosm1)

Table of Contents

List of Tables

List of Figures

1. Revision History

Table 1: Revision History

Date	Developer(s)	Change
2022-09-22	Brian Le	Document created
2022-09-23	Brian Le	Detailed steps and tools
2022-10-20	Amos Yu	Improved formatting

2. Version Control

Team members are expected to use the private Github repository. Team members will also be expected to maintain repos containing clean and accurate code. Members are required to create new branches when developing different aspects of software before merging into the main branch once functionality has been verified and is stable. Changes are highly encouraged to be made through multiple commits with a new commit made for per module or per 50 lines of code. Pull Requests are to be processed with the validation of at least two other members within the team with discussions encouraged to be held within their respective topics.

3. Roles and Responsibilities

3.1. Aaron Li

- Responsible for the design of mounting and the physical components of Cyclops
- Ensuring the seamless integration of electrical and mechanical components

3.2. Amos Cheung

- Responsible for the design of the electrical components of Cyclops
- Ensure accurate reading of accelerometer and/or gyroscope

3.3. Amos Yu

- Responsible for crash detection module for Cyclops
- Ensuring that proper logging system is in place

3.4. Brian Le

- Responsible for blindspot detection module for Cyclops
- Ensuring proper vehicle and cyclist detection

3.5. Manny Lemos

- Responsible for sensor and camera module of Cyclops
- Ensuring proper communication between the crash detection and blindspot detection module for Cyclops

4. Process Workflow

Team members of Cyclops are expected to follow this general outline as a workflow:

1. Before beginning work on a new module, pull any new changes from the master branch.
2. Create a new branch to develop on if a branch doesn't already exist for this module.
3. Create a detailed plan of the structure of the module.
4. Implement the modules/functions.
5. Perform unit testing on the modules and its functions.
6. Push any changes to the current working branch.
7. Repeats steps 4 through 6 towards the completion of the module.
8. Merge new functionality to the Master branch after code review from 2 other team members.

5. Details on Steps to be Taken

The following are steps that the Cyclops team aim to complete:

1. Create CAD designs, using Inventor or SolidWorks.
2. Build base of Cyclops mount to support electrical and mechanical components.
3. Build gyroscopic system with relevant electrical components and Arduino environment for the crash detection system.
4. Gather feature set of vehicles to train models on vehicle recognition within the blindspot detection system.
5. Build blindspot detection system with relevant sensor and computer vision components with Python IDE.
6. Implement crash logging system for Cyclops.
7. Implement communication between the crash detection system, blindspot detection system and the crash logging system.
8. Assemble closed physical system.

6. Development Tools

The following are environments that team members will be using for the development and debugging of the system:

- Visual Studio Code: Utilized for computer vision and video log development.
- Arduino IDE: Will be used for the development of the crash detection system.
- Autodesk Inventor: Utilized for creating 3D models for the physical aspects of Cyclops including the mount and chassis.
- Github: The version control and issue tracking platform utilized.
- Bike: Tester bike will be used to simulate and test the functionalities of Cyclops in a real time environment.

7. Handling Changes

Team members will also be encouraged to follow the format within **CONTRIBUTING.md**:

1. Create an issue in the Github page.
2. Create a new branch if necessary with the name of the change.
3. Perform unit testings to verify that the issue is still prevalent.
4. Perform regression testing to ensure that there are no further issues appearing.
5. Merge to the Master branch.
6. Close the issue