Velomobile Control & Telemetry System Integration Plan

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Ver.** | **Description** | **Author** |
| March 13, 2010 | 1.0 | Initial Composition | Daniel Johnson |
| March 13, 2010 | 1.1 | QA | John Schmidt |

Table of Contents

[1. Introduction 4](#_Toc262129931)

[1.1 Purpose 4](#_Toc262129932)

[1.2 Terminology 4](#_Toc262129933)

[2. Development 4](#_Toc262129934)

[2.1 Embedded Control Unit 4](#_Toc262129935)

[2.1.1 Embedded Control Unit Testing 4](#_Toc262129936)

[2.2 Telemetry Web System 5](#_Toc262129937)

[2.2.1 Telemetry Web System Testing 5](#_Toc262129938)

# Introduction

## Purpose

This document is intended to explain the integration and deployment process we have planned for our system. It will cover the development, testing, and deployment process; also covered will be the use cases and requirements we will be delivering in our alpha release of the system.

## Terminology

IDE: Interactive Development Environment (Visual Studios is an example)

ECU: Embedded Control Unit (Arduino, Touch Shield Slide is examples)

# Development

This project has two primary development areas that we will be focusing on. The first is the ECU on the Velomobile. The ECU’s primary role is to control the power assist motor of the Velomobile. The second area is the web-system, which is used for storing and displaying telemetry data to web users. In this section we will decompose these two areas and the developing environment they are created in.

## Embedded Control Unit

The ECU is primarily developed in the IDE provided for the device. The IDE compiles and runs its own variant of C++ that is close enough that it allows members of the team to use Visual Studio to develop for the ECU. The ECU has two main components, the Arduino (small board with its own CPU) and the TouchShield Slide (a small touch screen). Both run off of the Arduino chipset and run in parallel to each other. On the ECU we will have a Bluetooth module allowing us access to the web to transmit telemetry data to the web-system, thus linking the two aspects of our project together.

### Embedded Control Unit Testing

Testing of the ECU integration happens in two phases. First testing is done on components internal to this piece of the system. We start components that can be tested individually, like our data calculations, or our GPS parser. Both will be tested using Visual Studio to verify their accuracy. After individual components are tested, we move on to components that rely on the ECU itself for operation or the system as a whole. These components will be tested by mimicking input from an actor on the system. We will then check the output to verify the systems functionality. For the alpha release we will fulfill the requirements listed below.

* 1. **Sensor Input**
     1. *[Critical]* The System shall read and store sensor data for revolutions.

Revolutions are a critical piece of data that should be use in almost all of our calculations.

* + 1. *[Critical]* The System shall read and store sensor data for battery levels.

Battery levels will be used to provide raw information to the user and in other calculations.

* + 1. *[Critical]* The System shall read and store sensor data for voltage levels.

Voltage levels will be used to provide raw information and in other calculations.

* + 1. *[Critical]* The System shall read and store sensor data for throttle position.

Taking some form of level reading from the throttle, we'll record a position for use in assist calculations.

* + 1. *[Critical]* The System shall read and store sensor data for torque levels.

Taking readings from a torque sensor, the system shall store the information for use later.

(While this is considered a critical requirement, it will not be tested. No torque sensor has been acquired at this time.)

* 1. **Calculations**
     1. *[Critical]* The System shall calculate velocity using sensor data.

Using the revolutions and a clock internal to the system we shall calculate the velocity in meters per seconds and miles per hour.

* + 1. *[Critical]* The System shall calculate revolutions per minute using sensor data.

Using the revolutions and a clock internal to the system we shall calculate the revolutions per minute.

* 1. **Power Assist Control**
     1. *[Critical]* The System shall apply power assist based on throttle.

Using the throttle assist we will be able to apply power assist to propel the Velomobile.

* 1. **Embedded Display**
     1. *[Critical]* The System's default display will include primary data, velocity, battery level, etc.

The system's default display should display only vital information to the user, allowing them to get readings like the velocity, battery level, estimated battery life, or other such primary information.

## Telemetry Web System

The web system is primarily developed using Visual Studio 2008. The web system is comprised of these programming languages: C#, ASP.NET, HTML, XML, SQL, and JavaScript. Some development was done using Microsoft SQL Server 2008 to implement our database.

### Telemetry Web System Testing

Testing of the telemetry web system is primarily done controlling input and monitoring output. Due to the nature of the system this is very easy to do, as we can fabricate an XML stream and send it to the web server, monitoring changes to the database and the acknowledgments that are sent back to the sender. Below we have outlined the requirements we will be fulfilling with our alpha release.

* 1. **Telemetry Web Front-End**
     1. *[Critical]* Display telemetry data in visual form.

We will have a web front end that will allow the viewing of telemetry data from devices in graphs, charts, etc.

* + 1. *[High]* Allow users to switch between devices they observe.

Since we are designing the web system to dynamically accept telemetry data from devices, users should be able to select which device's telemetry data they would like to view.

* 1. **Telemetry Web Back-End**
     1. *[Critical]* Record telemetry streams from devices.

When receiving data from devices, we will decode and store it in the database.