

# Project Plan

*Client*  
Vermeer

*Project Members*  
John Shelley - Web Master - Software Engineer  
Alec Johanson - Team Lead - Computer Engineer  
Ahmad Muaz Mohd Khairi - Communication - Electrical Engineer  
Koray Celik - Advisor  
Arun Somani - Advisor

*Android Bridge for Off-Highway Vehicles*

## Table of Contents:

[Problem Statement:](#)

[System Block Diagram:](#)

[System Description:](#)

[Operating Environment:](#)

[User Interface Description:](#)

[Functional Requirements:](#)

[Non-Functional Requirements:](#)

[Use Cases:](#)

[Start the 'CAN Application'](#)

[Description](#)

[Actors](#)

[Triggers](#)

[Basic Flow](#)

[Preconditions](#)

[Postconditions](#)

[View the Vehicles Outputted Display](#)

[Description](#)

[Actors](#)

[Triggers](#)

[Basic Flow](#)

[Preconditions](#)

[Postconditions](#)

[Send Commands to the Machine](#)

[Description](#)

[Actors](#)

[Triggers](#)

[Basic Flow](#)

[Preconditions](#)

[Postconditions](#)

[Deliverables:](#)

[V&V Plan \(Validation & Verification\):](#)

[Work plan:](#)

[Work Breakdown Structure:](#)

[Project Schedule:](#)

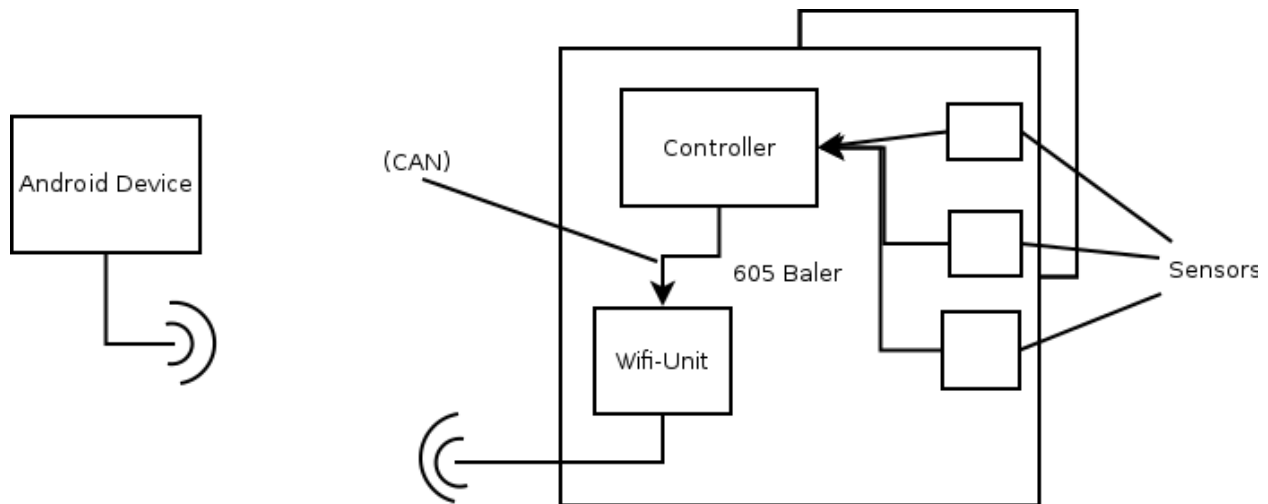
[Risks:](#)

[Appendix:](#)

## Problem Statement:

Modern off-highway/agricultural systems use outdoor rated LCD displays to implement user interfaces. These current solutions are either too expensive, or put users at risk being an unsafe distance from the machines. Vermeer wishes to replace these expensive display systems with a more inexpensive solution as well as provide a more flexible development environment. A J1939 CAN system is used on most Vermeer systems, but would like to be expandable to other protocols. Vermeer has expressed their wish of emphasizing research and development with solutions utilizing android systems due to their prior experience with this technology. The solution must be capable of both receiving and sending messages to the CAN bus from a safe distance.

## System Block Diagram:



## System Description:

The android bridge for off-highway equipment will take into account four major hardware items, and allow them to communicate with each other. The Android Tablet will communicate with a server operating on our microcontroller with CAN capabilities (PIC32MX), these will connect via an XBEE WIFI Router. After communicating with the Processor, the tablet sends and receives information to the processor relays it to the CAN Bus mounted on an Off-Highway Vehicle. Our end result is to have the Android tablet give and receive commands to and from the vehicle. This type of system allows for workers in dangerous environments to operate from remote distances, and allows for more flexibility when developing the software that controls the machines.

## Operating Environment:

As we are building a system for off-highway equipment, the system itself should be operable in high temperature and high pressure. The range of temperature that the system should withstand is about  $-25^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$  as the off-highway vehicle will be using a lot of power for its functionality, it will produce quite high amount of heat and pressure in addition to surrounding outdoor environment.

We need to have a network that is not interfered by other networks. This being that the data may have a variety of uses and Vermeer may plan to send the given information to a range of sources, either inside the isolated network, or outside of it. Also the network should have the capabilities of blocking out unwanted users, viruses, or malware.

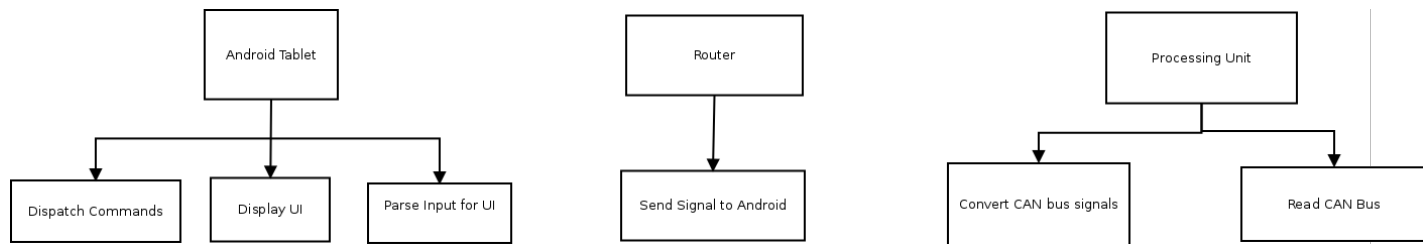
The system must be able to function in a high vibration environment due to the nature of the machines it will be mounted on. The level of this vibration is difficult to quantify and will need to be tested directly on the working machine if possible.

## User Interface Description:

Currently the interface will be hosted on an Android tablet. We have decided to go native instead of using a route such as html. This will allow for more native operations such as GPS, Motion Detection, and the Camera, not to mention overall CPU and GPU speeds. Because of this the interface will also have to be not only user friendly but also safe. The reason being is that our users will be in tough environments and if anything were to go wrong then we could potentially be at risk for causing harm to users. The interface will consist of buttons, toggles, switches, gauges, graphs, and charts for measuring.

*See Appendix A for an example UI design.*

## Modular Design:



The android tablet will receive and send signals through the WiFi network. It will also be responsible for parsing the input and generating the UI for the VT.

The router is responsible for sending signals wirelessly through the system.

The processing unit reads the CAN bus signal and sends it to the router in a format the router can send (SPI).

## Functional Requirements:

- stream > 10% CAN bus load at 250Kbps throughput
- Must be able to run outdoors
- Must be able to run in a high-vibration environment
- Must be able to communicate over a (large/safe) distance
- Must be able to run non-stop

## Non-Functional Requirements:

- Must be cheaper than current solution
- User friendly for the android user
- Extensible to other CAN protocols

## Use Cases:

### 1. Start the 'CAN Application'

- 1.1. Description
  - The user wishes to view the application.
- 1.2. Actors
  - User
- 1.3. Triggers
  - User determines to open the app.
- 1.4. Basic Flow
  - User turns the android device on
  - User finds the application icon in the app drawer
  - User launches the application by clicking on the icon
- 1.5. Preconditions
  - User must have the app installed
- 1.6. Postconditions
  - None

### 2. View the Vehicles Outputted Display

- 2.1. Description
  - The user wishes to start using the vehicle and hence view the outputted display.
- 2.2. Actors
  - User
  - Vehicle
- 2.3. Triggers
  - User chooses to operate the vehicle. User uses the given application.
- 2.4. Basic Flow
  - User must select the button to start the display
  - The display is then transmitted to the device via the wifi and vehicle connection.
- 2.5. Preconditions
  - User must have completed Use Case 1
  - User must be using the vehicles sensory input
- 2.6. Postconditions
  - Display is then transmitted to the device.

### 3. Send Commands to the Machine

#### 3.1. Description

- If a user wishes to send an instruction or command to the vehicle or machine.

#### 3.2. Actors

- User
- Vehicle

#### 3.3. Triggers

- User sends a command to the vehicle.

#### 3.4. Basic Flow

- User selects the proper button for sending the given outcome
- Android device send the command
- Vehicle receives the instruction via the wifi/can-bus interpreter.
- Vehicle acts to the given command
- Android receives a success/failure statement

#### 3.5. Preconditions

- User must have completed Use Case 1
- User must have completed Use Case 2
- The display must have a send command function

#### 3.6. Postconditions

- Display reacts to the commands return statement.

### Deliverables:

Deliverable	Date
Achieve equipment for tasks needed	15 February 2014
List of existing known industry options	20 February 2014
Project Plan v1	22 February 2014
A working CAN bus wifi bridge through android, that can observe and control Vermeer's off-highway equipment.	TBA

## V&V Plan (Validation & Verification):

We will use the steps below to validate and verify our product.

### 1) Reviews

- a) Requirement will be reviewed by key stakeholders (*See Glossary*).
- b) User Interface design will be reviewed by key stakeholders.
- c) The data from the microcontroller to be sent will be reviewed by key stakeholders
- d) Application prototype will be reviewed by key stakeholders.
- e) The client will provide beta testing feedback.

### 2) Testing

The test will be conducted with different kinds of tests:

#### a) **Monkey Testing.**

The system is tested with random input to see whether it can transmit and receive correctly. It will then produce a system that can be functioning for various situation.

#### b) **Stress Testing.**

The system is tested under a high temperature, high pressure and high vibration area.

#### c) **Endurance Testing.**

The system is tested for long period of time to ensure that frequent maintenance is not needed.

#### d) **Interference Testing.**

The system is tested with a lot of interference from outside such as other network interference to see how the data can be received without any error.

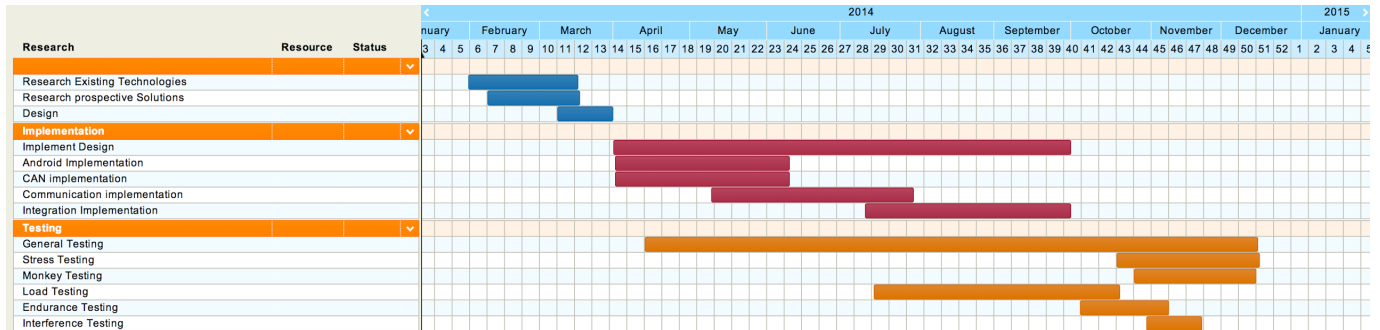
The acceptance criteria for the first release are as follows:

- The data that were received by the android is 95% correct.
- Zero major error in data receives from WiFi.
- The microcontroller can read the data from CAN with zero major error.



## Work plan:

### Work Breakdown Structure:



### Project Schedule:

#### Waterfall

A clear goal is set for us. Due to the time, course, and the wishes of Vermeer, Waterfall makes the most sense. Our MVP is very close to the final product, making an iterative approach less intuitive. We will use some agile methodologies in the sense that we will be periodically checking with Vermeer that we are on the right track, making adjustments as necessary.

Requirements - Given by Vermeer

Design - Includes needed research, To be completed by 04/01/2014

Implementation - To be completed by 10/15/2014

Verification - To be completed by 12/15/2014 based on the requirements given by Vermeer

Maintenance - Projected to be minimal, this is more of a prototype

## Risks:

The following risks are associated with our current project. Note that the probability is a percentage, criticality is out of 100, and risk factor is a product of probability and criticality.

Risk	Probability of occurrence	Criticality	Risk Factor	Mitigation Strategy
Application has critical bugs.	.25	75	18.75	Test as frequently as we can, integration tests included.
Hardware Compatibility with Code Base.	.50	80	40	Test early the software and hardware communication.
People on the team leave.	.10	25	2.5	Make sure documentation is updated and have multiple people on the same tasks.
Leadership is forsaken, and tasks not assigned	.20	70	14	Communicate early with leaders, and assign tasks for the future to have a future plan assigned

## Glossary:

[CAN](#) - Controller Area Network

[MVP](#) - Minimum Viable Product

[SPI](#) - Serial Peripheral Interface

[UI](#) - User Interface

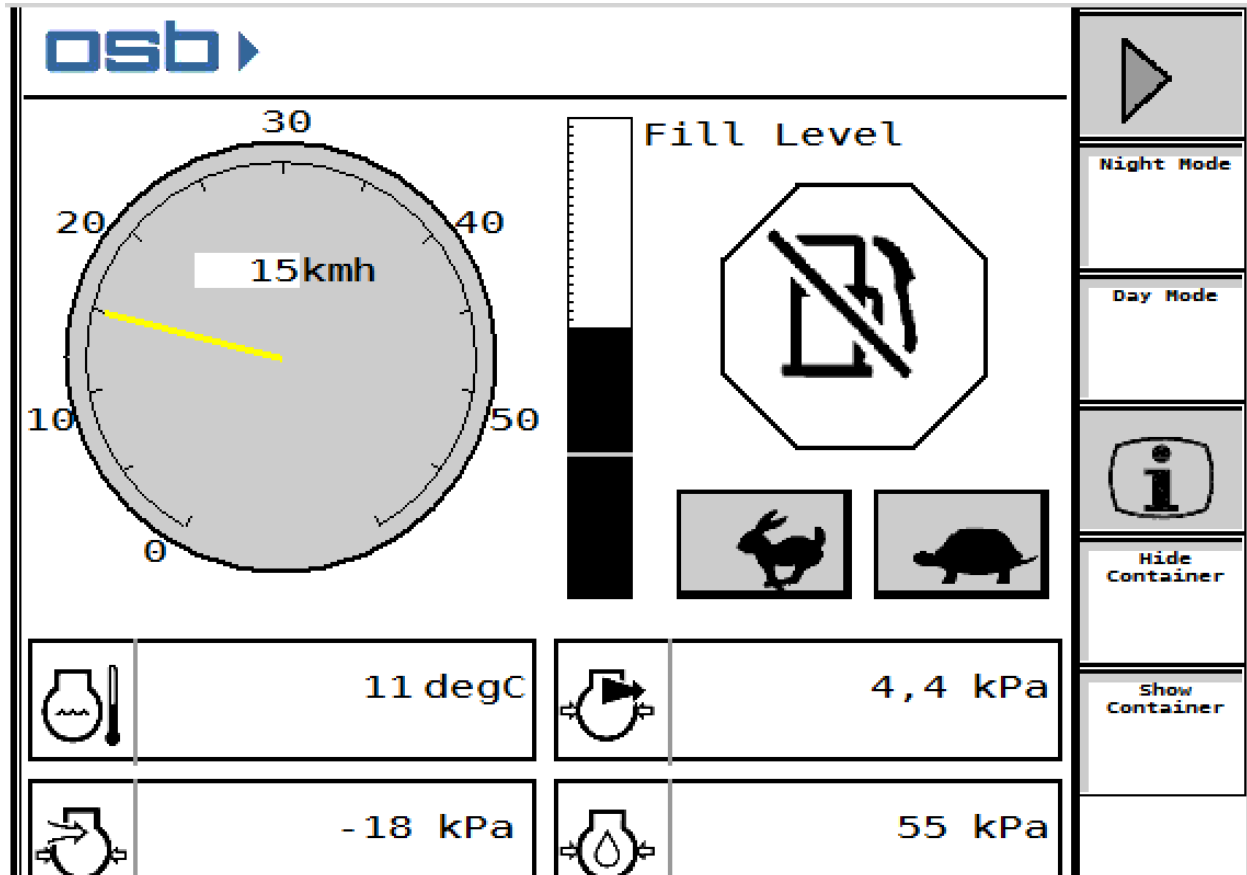
[VT](#) - Virtual Terminal


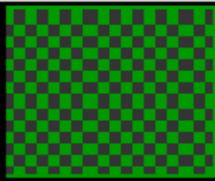



[Key Stakeholders](#) - Vermeer & Advisors

# Appendix:

A)

Two possible implementations:



	<b>ISO</b> <i>AgLib</i> ISOAgLib Tutorial	
	ISOAgLib Version: 0.0.0	
	Tutorial Version: 0.0.0	
-	Build Date:	
+	Time: 0 : 0 : 0	