Department of Computer Science and Engineering
The University of Texas at Arlington

Sliding Profiler

Detailed Design Specification

The Builders Group Ltd.

Team Members:
Chudamani Aryal
Lloyd Bond
Tyler Buchanan
Jefferson White

Late Updated: 15 January 2013 @ 11:10:00 AM
Copy Printed: 15 January 2013 @ 11:10:00 AM
Table of Contents

1 Sliding Profiler Introduction ................................................................. 6
   1.1 Document Overview ........................................................................... 6
   1.2 Product Overview ............................................................................ 6

2 Architectural Overview ........................................................................... 10
   2.1 Overview ......................................................................................... 10
   2.2 Module Decomposition .................................................................... 14
   2.3 Module Descriptions ....................................................................... 16

3 Design Overview .................................................................................... 21
   3.1 Introduction ..................................................................................... 21
   3.2 Physical Platform ........................................................................... 21
   3.3 Hardware/Electronics Processor component .................................. 21
   3.4 Programming Language selection .................................................. 21

4 Input Layer ............................................................................................ 23
   4.1 Design Overview ............................................................................. 23
   4.2 Sensors ........................................................................................... 24
   4.3 A/D Converter .................................................................................. 26
   4.4 UIC .................................................................................................. 27
   4.5 Data Handler .................................................................................... 28
   4.6 Net Comm ....................................................................................... 30
   4.7 Power Input ..................................................................................... 31

5 Processing Layer .................................................................................... 33
   5.1 Design Overview ............................................................................. 33
   5.2 Data Handler .................................................................................... 34
   5.3 Algorithm ......................................................................................... 36

6 Storage Layer .......................................................................................... 39
   6.1 Design Overview ............................................................................. 39
   6.2 Data Handler .................................................................................... 40
   6.3 Data Manager ................................................................................... 41
   6.4 Physical Storage ............................................................................... 43

7 Output Layer .......................................................................................... 45
   7.1 Design Overview ............................................................................. 45
   7.2 Data Handler .................................................................................... 46
   7.3 Hardware Out ................................................................................... 48
<table>
<thead>
<tr>
<th>Senior Design Documentation Library</th>
<th>Detailed Design Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4  Net Comm..........................................................</td>
<td>49</td>
</tr>
<tr>
<td>8  GUI Layer..............................................................</td>
<td>51</td>
</tr>
<tr>
<td>8.1 Design Overview ..................................................</td>
<td>51</td>
</tr>
<tr>
<td>8.2 Network ....................................................................</td>
<td>52</td>
</tr>
<tr>
<td>8.3 Librarian...............................................................</td>
<td>53</td>
</tr>
<tr>
<td>8.4 Storage .....................................................................</td>
<td>54</td>
</tr>
<tr>
<td>8.5 Display ......................................................................</td>
<td>54</td>
</tr>
<tr>
<td>9  Error Handling............................................................</td>
<td>56</td>
</tr>
<tr>
<td>9.1 Description..................................................................</td>
<td>56</td>
</tr>
<tr>
<td>9.2 Modules affected......................................................</td>
<td>56</td>
</tr>
<tr>
<td>9.3 Inputs / Outputs.......................................................</td>
<td>56</td>
</tr>
<tr>
<td>9.4 Data ..........................................................................</td>
<td>56</td>
</tr>
<tr>
<td>9.5 Processing / Pseudo Code ..........................................</td>
<td>56</td>
</tr>
<tr>
<td>10 Platform Design..........................................................</td>
<td>58</td>
</tr>
<tr>
<td>11 Hardware Design ........................................................</td>
<td>59</td>
</tr>
<tr>
<td>11.1 Introduction............................................................</td>
<td>59</td>
</tr>
<tr>
<td>11.2 Design Considerations .............................................</td>
<td>59</td>
</tr>
<tr>
<td>11.3 Hardware Design .....................................................</td>
<td>59</td>
</tr>
<tr>
<td>11.4 Printed Circuit Board Design ....................................</td>
<td>66</td>
</tr>
<tr>
<td>12 Hardware Components ..................................................</td>
<td>67</td>
</tr>
<tr>
<td>12.1 PIC Microcontroller ................................................</td>
<td>67</td>
</tr>
<tr>
<td>12.2 SD/MMC Card Reader ...............................................</td>
<td>67</td>
</tr>
<tr>
<td>12.3 Inclinometer ...........................................................</td>
<td>67</td>
</tr>
<tr>
<td>12.4 Encoder ....................................................................</td>
<td>68</td>
</tr>
<tr>
<td>12.5 Temperature Sensor ..................................................</td>
<td>68</td>
</tr>
<tr>
<td>12.6 Wi-Fi Network Interface Card ....................................</td>
<td>68</td>
</tr>
<tr>
<td>12.7 Ethernet Network Interface Card ................................</td>
<td>69</td>
</tr>
<tr>
<td>12.8 GPS Receiver ...........................................................</td>
<td>69</td>
</tr>
<tr>
<td>13 Traceability Matrices....................................................</td>
<td>70</td>
</tr>
<tr>
<td>14 Quality Assurance.......................................................</td>
<td>73</td>
</tr>
<tr>
<td>14.1 Test Plans and Procedures ........................................</td>
<td>73</td>
</tr>
<tr>
<td>14.2 Test Cases ..................................................................</td>
<td>75</td>
</tr>
<tr>
<td>15 Acceptance Plan ...........................................................</td>
<td>76</td>
</tr>
<tr>
<td>15.1 Overview....................................................................</td>
<td>76</td>
</tr>
<tr>
<td>15.2 Packaging and Installation ........................................</td>
<td>76</td>
</tr>
<tr>
<td>15.3 Acceptance Testing ...................................................</td>
<td>76</td>
</tr>
<tr>
<td>15.4 Acceptance Criteria ..................................................</td>
<td>76</td>
</tr>
</tbody>
</table>
List Of Figures

Figure 1 - Product Concept ........................................................................................................... 7
Figure 2 - Architectural Diagram ................................................................................................. 11
Figure 3 - Module Decomposition Chart ..................................................................................... 14
Figure 4 - Input Layer .................................................................................................................. 23
Figure 5 - Process Layer ............................................................................................................. 33
Figure 6 - Storage Layer ............................................................................................................. 39
Figure 7 - Output Layer .............................................................................................................. 45
Figure 8 - GUI Layer ................................................................................................................... 51
Figure 9 - Platform Design Top View ......................................................................................... 58
Figure 10 - Platform Design Side View ...................................................................................... 58
Figure 11 - Platform Design Front View .................................................................................... 58
Figure 12 - Power Subsystem Diagram ....................................................................................... 61
Figure 13 - Net Comm Subsystem Diagram ............................................................................... 62
Figure 14 - Physical Storage Subsystem Diagram ........................................................................ 63
Figure 15 - Sensor Subsystem Diagram ...................................................................................... 64
Figure 16 - Hardware Out Subsystem Diagram .......................................................................... 65
## List Of Tables

Table 1 - Architectural Data Elements................................................................. 11
Table 2 - Data Flows Within the System ............................................................... 15
Table 3 - Thermometer Inputs/Outputs ................................................................ 24
Table 4 - GPS Inputs/Outputs .............................................................................. 24
Table 5 - Inclinometer Inputs/Outputs ................................................................. 25
Table 6 - Distance Encoder Inputs/Outputs ......................................................... 25
Table 7 - A/D Converter Inputs/Outputs .............................................................. 26
Table 8 - UIC Inputs/Outputs .............................................................................. 27
Table 9 - Pack (Input Layer) Inputs/Outputs ...................................................... 28
Table 10 - Unpack (Input Layer) Inputs/Outputs ............................................... 30
Table 11 - Net Comm (Input Layer) Inputs/Outputs ........................................... 31
Table 12 - Unpack (Process Layer) Inputs/Outputs ............................................ 34
Table 13 - Pack (Process Layer) Inputs/Outputs ................................................. 35
Table 14 – Bump Detection Inputs/Outputs ...................................................... 36
Table 15 - Unpack (Storage) Inputs/Outputs ..................................................... 40
Table 16 - Pack (Storage) Inputs/Outputs .......................................................... 41
Table 17 - Session Handler Inputs/Outputs ....................................................... 42
Table 18 - Initialize Inputs/Outputs ..................................................................... 43
Table 19 - Physical Storage Inputs/Outputs ....................................................... 43
Table 20 - Unpack (Storage) Inputs/Outputs ...................................................... 46
Table 21 - Pack (Output Layer) Inputs/Outputs .................................................. 47
Table 22 - Bump Light Inputs/Outputs .............................................................. 48
Table 23 - Fan Direction Inputs/Outputs ............................................................ 48
Table 24 - Error Lights/Heartbeat Inputs/Outputs ............................................. 49
Table 25 - Traceability Matrix .......................................................................... 72
# Document Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>10/7/2012</td>
<td>First Draft</td>
<td>Submitted for Peer Review</td>
</tr>
<tr>
<td>1.0</td>
<td>10/08/2012</td>
<td>First Draft for Informal Review</td>
<td>Submitted to Mr. O’Dell for Review</td>
</tr>
<tr>
<td>2.0</td>
<td>10/22/2012</td>
<td>Baseline</td>
<td>Final Submission To Mr. O’Dell</td>
</tr>
</tbody>
</table>
1 Sliding Profiler Introduction

1.1 Document Overview
The Detailed Design Specifications will further detail based on the Architectural Design Specifications. In Architectural Design Specifications, the system was divided into layers and sub-systems. In Detailed Design specifications, the system will be further divided into modules. The modules provide details about the functionalities of each sub-systems. The module includes inputs, outputs, data required, processing, and the pseudo code. Also, this document provides the traceability matrices which provide relationship between the various modules and the requirements. Also, this document includes quality assurance chapter which covers the testing consideration. Finally, this document includes Acceptance Plan chapter which covers packing, installation, acceptance testing, and acceptance criteria.

1.2 Product Overview
The Sliding Profiler is a wet concrete bump detection product that aggregates data from many sensors to determine the probable smoothness of the freshly laid road. Before the concrete has dried, the operators will know if the concrete has met the state requirement for smoothness of concrete roads. The Sliding Profiler will use various sensors to determine if the freshly poured concrete has any bumps unfit for vehicular travel. Our expectations are the Sliding Profiler will follow a cement spreader detecting inclination and distance, and then notify concrete smoothers of detected bumps to be smoothed. The Sliding Profiler will benefit construction contractors who specifically specialize or intend to build concrete highway or road. The Sliding Profiler will need to be made available commercially.
1.2.1 Sliding Profiler Features/Components

The Sliding Profiler system will feature/use the following components:

1) Sled: - The sled portion of the profiler is the part of the product that glides across wet road surface. It provides a platform for which the profiler electronics and sensors can function and collect data.

2) Electronics Housing: - The electronic housing is responsible for providing a secure mounting platform for all the sensors and circuitry. It is also responsible for protecting the electronics from the elements.

3) Sensors: -
   - Inclinometer - The inclinometer is responsible for reacting to the angle of the platform and providing a voltage signal that the software system will use for operational data.
   - Distance Sensor - The distance sensor is responsible for translating the distance traveled into a voltage signal that the software system will use for operational data.
Temperature Sensor - The temperature sensor is responsible for reading the temperature inside of the electronics housing and providing a voltage signal that the software system will use for error checking.

4) Display: The display subsystem is responsible for providing the user with a visual indicator about the current state of the Sliding Profiler as well as alerts to detected bumps.

5) Communications: The communications subsystem is responsible for transmitting/receiving data to/from the client software. It is also responsible for communicating with the various hardware peripherals.

6) Processor: The processor subsystem is responsible for executing instructions from the Software system.

7) Storage: The storage subsystem is responsible for storing operational data on a non-volatile medium that will be read by the client software. It is also responsible for storing all calibration and configuration information for future use.

8) Cooling: The cooling subsystem is responsible for ventilating the electronics housing.

9) Power Distribution: The power distribution subsystem is responsible for providing the necessary and proper power to all subsystems requiring a separate power source. The power source is provided externally from road paver.

10) Client PC: The client PC provides the operating system for the software subsystems intended for use by a technician. See the Software system for detailed information.

1.2.2 Product Purpose
The Sliding Profiler will benefit construction contractors who specifically specialize or intend to build concrete highway or road. The Sliding Profiler will need to be made available commercially.

1.2.3 Product Scope
The Sliding Profiler is currently realized as a prototype used by TxDOT contractors. It is designed to slide behind a road paving machine to measure and indicate sharp bumps in the pavement that would cause poor ride quality. The scope of this effort is to replace and redesign the key components of this product.
## 1.2.4 Definitions and Terms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>Sliding Profiler</td>
</tr>
<tr>
<td>UIC</td>
<td>User Input Configuration</td>
</tr>
<tr>
<td>RAVE</td>
<td>Running Average</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>SRS</td>
<td>System Requirement Specifications</td>
</tr>
<tr>
<td>DDS</td>
<td>Detailed Design Specifications</td>
</tr>
<tr>
<td>ADS</td>
<td>Architectural Design Specifications</td>
</tr>
</tbody>
</table>
2 Architectural Overview

2.1 Overview

The system consists of five layers: Input Layer, Process Layer, Storage Layer, Output Layer and GUI Layer. The Input Layer gets all the inputs from the sensors as well as the end users. The Process Layer is responsible for processing all the input/raw data. Once the data has been processed, the data is sent to the Storage Layer and Output Layer. The Storage Layer is responsible for managing and storing raw/processed data. It is also responsible for making stored data readily accessible, if needed by the end users. The Output Layer is responsible for displaying the result through hardware lights and sending the results to network communications. The GUI Layer is responsible for displaying the results/data on PC. It is also responsible for getting inputs from the user, which are transferred back to Input Layer via network communications.
Figure 2 - Architectural Diagram

Table 1 - Architectural Data Elements

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GU 1</td>
<td>Formatted data (raw, processed) sent to GUI display</td>
</tr>
<tr>
<td>GU 2</td>
<td>Command for stored data retrieval</td>
</tr>
<tr>
<td>GU 3</td>
<td>Formatted data (algorithm variables) sent to input layer</td>
</tr>
<tr>
<td>GU 4</td>
<td>Data being sent to be displayed</td>
</tr>
<tr>
<td>GU 5</td>
<td>Transaction of data to be sent to the data manager</td>
</tr>
<tr>
<td>GU 6</td>
<td>Formatted data being sent to the DM to be stored on local storage</td>
</tr>
<tr>
<td>GU 7</td>
<td>Packaged data to be sent out to different layers</td>
</tr>
<tr>
<td>GU 8</td>
<td>Received data stored on PC file system</td>
</tr>
<tr>
<td>IN 1</td>
<td>Network communication between GUI and Input Layer</td>
</tr>
<tr>
<td>IN 2</td>
<td>Formatted data being sent to the DM for transmission to the Process layer</td>
</tr>
<tr>
<td>IN 3</td>
<td>Raw data sent to be transform data from analog to digital</td>
</tr>
</tbody>
</table>
2.1.1 Input Layer

This layer contains inputs from such areas as sensors, power, and UIC. Also, this layer contains inputs from users through GUI via networking. The sensors will be the main source of input as it contains information such as GPS, temperature, distance, and angle. Most of these inputs will then be packaged and sent to the Process layer. And some of the requests from the GUI like data retrieval will be sent to the Storage layer.

The purpose of input layer is to handle all input devices from all sources. The data collected will be stored, processed, or displayed from this layer to Storage and Processing Layer. The main goal is to maintain operations through collecting data and transmit data effectively. To summarize, this layer is responsible for collection, packaging, and sending data to its destination.

The input layer will depend upon all sensors, power input, and communication with other layers to perform correctly. If any part should fail then an error should occur and trigger an error code and then stop processing data under certain conditions.

2.1.2 Processing Layer

The processing layer provides the function of processing raw data to produce meaningful results. The layer primarily runs the bump detection algorithm.

The purpose of the processing layer is to process the incoming stream of sensor data in real time to detect when it indicates a bump based on the user input configuration that defines a bump. The processing layer shall be the sole component that analyzes the sensor data, and the only place that provides bump detection for the system. To summarize, the processing layer receives raw sensor data from the input layer, processes it, and sends the processed data (including bump detection information) to both the storage layer and the output layer.
The processing layer depends on the microcontroller to provide processing power. The function of the data processing will be determined by the bump detection algorithm provided by Dr. Walker.

2.1.3 Storage Layer

The storage layer is in charge of keeping and managing both the raw and processed data that the system acquires. It will maintain the physical storage of the data in an orderly manner, and provide this data on request.

The purpose of the storage layer is to provide a means to keep a history of raw and processed data that has been collected by the system so that it can be retrieved later. To summarize, the storage layer maintains the physical storage, include initializing the file system. The layer receives live packets of the raw data and processed data, and keeps them on the physical storage. When requested, the storage layer will provide the stored data to the output layer.

The storage layer will depend on some To-Be-Decided file format to manage the physical storage.

2.1.4 Output Layer

The output layer provides output regarding bump detection, error conditions, and normal operation to the user. It also controls networking with the GUI layer to provide information such as raw and processed data as output to the GUI.

The purpose of the output layer is to convey the output of the system to the user. It does this with the lights on the platform (bump detection light and heartbeat in Hardware Out subsystem) and also provides output to the GUI layer so it can be viewed. To summarize, the output layer receives packaged raw and processed data from the input layer and processing layer. It uses the processed data to run the lights on the platform appropriately. It also relays this data to the GUI layer over the network when connected. This data can be both live data and stored data.

The output layer depends on the implementation of the GUI/Platform network to provide a connection to the GUI for output.

2.1.5 GUI Layer

The GUI layer is the only layer of the system that runs independently of the physical platform. The GUI layer operates on a PC connected to the system over a network. Both raw and processed data can be displayed and analyzed through the GUI, providing an easy way for the user to check the results. It also allows the user to provide input to configure the device.

The purpose of the GUI layer is to provide an easy to use interface with the user, and display all relevant data from the system in an easy to follow manner. The GUI layer can view live data as it is streamed from the running system, or saved data from the system through either the network or removable storage. The GUI layer also allows the user to configure the device.

To summarize, the GUI layer gets the data of the system from the Output layer, and stores the data using the PC file system. This data is analyzed and displayed in the GUI to the user. The GUI also reads data from the Local Storage subsystem which allows the systems removable
storage to be connected. Finally, the GUI also provides the function for the user to configure the system (User Input Configuration) via easy to use GUI inputs.

The GUI layer is dependent on the Windows operating system to provide a storage system to hold data, a run time GUI application and the underlying network capabilities to allow a connection with the system.

2.2 Module Decomposition

2.2.1 Overview

The main purpose of DDS was to break down each sub system into modules. This section contains the high level definition of each module that we have in our product.

2.2.2 Module Decomposition Chart

![Module Decomposition Chart](image-url)
2.2.3 Data Flows within the System

Table 2 - Data Flows Within the System

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GU 1</td>
<td>Formatted data (raw, processed) sent to Net Comm. Via Networking</td>
</tr>
<tr>
<td>GU 2</td>
<td>Data sent for unpacking</td>
</tr>
<tr>
<td>GU 3</td>
<td>Unpacked data sent to Data Manager</td>
</tr>
<tr>
<td>GU 4</td>
<td>Data Manager sends data for file storage</td>
</tr>
<tr>
<td>GU 5</td>
<td>Data retrieved from file storage to data manager</td>
</tr>
<tr>
<td>GU 6</td>
<td>Data Manager sends data for external storage</td>
</tr>
<tr>
<td>GU 7</td>
<td>Data retrieved from external storage to data manager</td>
</tr>
<tr>
<td>GU 8</td>
<td>Data send to visual for display</td>
</tr>
<tr>
<td>GU 9</td>
<td>Inputs from GUI users to Net CMD</td>
</tr>
<tr>
<td>GU 10</td>
<td>Data send from Net CMD to Pack to be packed</td>
</tr>
<tr>
<td>GU 11</td>
<td>Packed data forwarded to Net Comm. To be forwarded to Input Layer</td>
</tr>
<tr>
<td>GU 12</td>
<td>Inputs from GUI users to Data Manager</td>
</tr>
<tr>
<td>IN 1</td>
<td>Network communication between GUI and Input Layer</td>
</tr>
<tr>
<td>IN 2</td>
<td>Data from Net Comm to Unpack for unpacking data</td>
</tr>
<tr>
<td>IN 3</td>
<td>Input configuration file communication line to be used in the process layer</td>
</tr>
<tr>
<td>IN 4</td>
<td>Input configuration file communication line to be used in the process layer</td>
</tr>
<tr>
<td>IN 5</td>
<td>Packaged data from GUI Layer</td>
</tr>
<tr>
<td>IN 6</td>
<td>Data from Inclinometer sent to A/D Converter</td>
</tr>
<tr>
<td>IN 7</td>
<td>Data from Encoder sent to A/D Converter</td>
</tr>
<tr>
<td>IN 8</td>
<td>Data from GPS sent to A/D Converter</td>
</tr>
<tr>
<td>IN 9</td>
<td>Data from Thermometer sent to A/D Converter</td>
</tr>
<tr>
<td>IN 10</td>
<td>Digital data converted from A/D to Pack for packing purpose</td>
</tr>
<tr>
<td>OU 1</td>
<td>Packed data sent from Storage Layer to Output Layer</td>
</tr>
<tr>
<td>OU 2</td>
<td>Packed data sent from Processing Layer to Output Layer</td>
</tr>
<tr>
<td>OU 3</td>
<td>Unpacked data sent to data divider</td>
</tr>
<tr>
<td>OU 4</td>
<td>Data sent to pack for packing purpose (which is to be send to GUI)</td>
</tr>
<tr>
<td>OU 5</td>
<td>Data send to Bit Ranger to bit formatting</td>
</tr>
<tr>
<td>OU 6</td>
<td>Data send to NIC which outputs as JSON packets</td>
</tr>
<tr>
<td>OU 7</td>
<td>Data send to indicate bump light</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>OU 8</td>
<td>Data send to indicate fan direction</td>
</tr>
<tr>
<td>OU 9</td>
<td>Data send to indicate error/heartbeat light</td>
</tr>
<tr>
<td>PR 1</td>
<td>Communication between Input and processing layer</td>
</tr>
<tr>
<td>PR 2</td>
<td>Unpacked data sent to algorithms for processing</td>
</tr>
<tr>
<td>PR 3</td>
<td>Processed data to be sent for packaging</td>
</tr>
<tr>
<td>ST 1</td>
<td>Communication between Processing and Storage layer</td>
</tr>
<tr>
<td>ST 2</td>
<td>Communication between Input and Storage layer</td>
</tr>
<tr>
<td>ST 3</td>
<td>Data send to Session handler for (initialize, read, write) purpose</td>
</tr>
<tr>
<td>ST 4</td>
<td>Data send to pack for packing purposed which is then send to Output Layer</td>
</tr>
<tr>
<td>ST 5</td>
<td>Session Handler to Initialize for initialization purpose</td>
</tr>
<tr>
<td>ST 6</td>
<td>Initialize to Physical storage to make sure that the file is initialized. If not initialize.</td>
</tr>
<tr>
<td>ST 7</td>
<td>Data from session handler to put data for requesting to write data</td>
</tr>
<tr>
<td>ST 8</td>
<td>Data from put data to physical storage for writing on file</td>
</tr>
<tr>
<td>ST 9</td>
<td>Data from session handler to get data for requesting to get data</td>
</tr>
<tr>
<td>ST 10</td>
<td>Data from get data to session handler for result to data request</td>
</tr>
<tr>
<td>ST 11</td>
<td>Get data asking data from the physical storage</td>
</tr>
<tr>
<td>ST 12</td>
<td>Physical storage giving data to get data</td>
</tr>
</tbody>
</table>

### 2.3 Module Descriptions

#### 2.3.1 Thermometer Module (Input Layer)

The thermometer module contains the internal sensor to measure the temperature of the inside of the hardware enclosure. This data is necessary to the system to identify when the system is too hot (overheating) or too cold, and to determine how to manage the system fans.

#### 2.3.2 GPS Module (Input Layer)

The GPS module contains the physical GPS of the system, which produces a GPS location of the device. This data is necessary to the system to track the location and progress of the product during its use.
2.3.3 Inclinometer Module (Input Layer)
The inclinometer module contains the inclinometer sensor which detects changes in inclination and provides it as input to the system. This data is necessary to use in the bump detection algorithm to determine bumps in the concrete.

2.3.4 Distance Encoder Module (Input Layer)
The distance encoder module contains the sensor that provides the rotation of the encoder sensor. The encoder will be connected to the distance wheel that turns as the platform slides along the concrete. So as the distance wheel turns the distance encoder changes rotation and indicates this as digital input to the system. This data is necessary to match with the inclination of the platform to provide for analysis.

2.3.5 A/D Converter (Input Layer)
The A/D converter module is responsible for converting signals from the input subsystem into digital formats to be used by the system. This includes analog to digital conversion as well as digital format conversion. The A/D converter will wait for a detected change on the distance encoder which indicates an encoder pulse. When this pulse occurs the module will send the necessary data from the sensors (in the format required) to the system.

2.3.6 UIC Module (Input Layer)
The UIC module receives and stores values to be used as variables in the bump detection variable. The values provided are the Bias Window Size, Running Average Window Size and the Threshold. These values are provided to the algorithm as a struct during operation.

2.3.7 Data Handler Pack Module (Input Layer)
The pack module packs data that will be sent to another layer. This data handler packs the raw data from the A/D converter and the algorithm parameters from the UIC and sends them both to the process layer.

2.3.8 Data Handler Unpack Module (Input Layer)
The unpack module receives JSON packets from the Net Comm module and interprets what they contain. Depending on the type of data contained in the JSON packet, this module sends the data to either the UIC module or the CMD module. The data that will be sent to the CMD module will be requests for stored data.

2.3.9 Net Comm Module (Input Layer)
The Net Comm module implements the communication over the wireless network with the GUI PC. This Net Comm receives the TCP/IP packets from the GUI and translates them.

2.3.10 Power Input Module (Input Layer)
This module provides power to the system. It contains the power switch with which the user interacts with, and provides power to all necessary modules. For detailed description, refer to Power Design section.
2.3.11 Data Handler Pack Module (Processing Layer)
The Pack module packages the processed data with the corresponding raw data to be sent. Raw and processed data packets are sent to the storage layer and the output layer.

2.3.12 Data Handler Unpack Module (Processing Layer)
The unpack module receives intermittent AlgoParam and RawData structs from the Input layer. This module forwards these packets to the pack module, and also sends the required data to the Algorithm module for processing.

2.3.13 Bump Detection Module (Processing Layer)
The Bump Detection module implements the bump detection algorithm for the system. It receives the raw data and algorithm parameters and detects when a bump has occurred. When a bump is found, the bump detection bit of the data being output is set high.

2.3.14 Data Handler Unpack Module (Storage Layer)
The unpack module receives raw and processed data as a struct from the Process layer, and turns it into a bit field to be written on disk as one entry. It passes this bit field to the session handler. The unpack module also receives requests for stored data which it forwards to the session handler.

2.3.15 Data Handler Pack Module (Storage Layer)
The Pack module packs outgoing stored data in the JSON format to pass to the Output Layer. This module is only active after a request for stored data, when the system is outputting stored data.

2.3.16 Data Manager Session Handler Module (Storage Layer)
The session handler module keeps track of sessions of recorded data. A session refers to one use of the system (ie. from power on to power off). Sessions are kept track of in the stored data so that when data is requested, only data from the current session will be sent. The session handler keeps track of these sessions by adding one bit to each stored data entry. This bit is reversed whenever the system is turned on, so sequential data entries with similar session bits indicate that they are from the same session. The session handler handles the physical storage, putting data and getting requested stored data.

2.3.17 Data Manager Initialize Module (Storage Layer)
The initialize module implements the file system on the physical storage when given the command. The file system used will be FAT32. The file system initializes the physical storage, preparing it to be used by the system.

2.3.18 Physical Storage Module (Storage Layer)
The physical storage module contains the means for storing data in the system. The physical storage will be a removable SD card. The system will initialize the physical storage by
formatting in FAT32. Once formatted, data will be intermittently written to and read from the physical storage.

### 2.3.19 Data Handler Pack Module (Output Layer)

The pack module creates JSON packets that will be sent to the GUI layer, and hands them off to the Net Comm module to get it there. These JSON packets can be either live data or requested stored data.

### 2.3.20 Data Handler Unpack Module (Output Layer)

The unpack module receives RawAndProData from the Process and Output Layer. This module takes the data necessary to send to hardware out. It also sends the data to the pack module so it can be sent to the GUI.

### 2.3.21 Hardware Out Fan Direction Module (Output Layer)

The fan direction module contains the fan and hardware to control the fans to cool the system hardware. The module receives one byte that indicates how the fans should be operated.

### 2.3.22 Hardware Out Error/Heartbeat Light Module (Output Layer)

The Error Lights/Heartbeat module contains an 8 led array. These leds are used to indicate error condition codes as well as the heartbeat to the user(s). The heartbeat is indicated as one led moving progressively along the led array, which is controlled with the error code.

### 2.3.23 Hardware Out Bump Light Module (Output Layer)

The Bump Light is used to indicate to the user when a bump is detected in the data. The light is comprised of 1 led light. The light receives one bit that indicates when the led should be illuminated.

### 2.3.24 Net Comm. Bit Ranger Module (Output Layer)

The main purpose of this module is to convert the data into desirable format.

### 2.3.25 Net Comm. NIC Module (Output Layer)

The main purpose of this module is to covert the TCP/IP packets into SPI

### 2.3.26 Net Comm. Module (GUI Layer)

The Net Comm module is the line between TCP/IP communication and the GUI layer. It handles all communication in and out that requires being sent to the physical device from the user interface.

### 2.3.27 Display Inputs Module (GUI Layer)

Although having a graphical interface is nice and useful. Having the ability to interact with the interface and send personalized requirements to the device is even better. This module is for interacting with the user.
2.3.28 Display Visual Module (GUI Layer)
This module is in a sense the graphics card of the GUI layer. It is responsible for sending graphical information to the user’s computer screen. This is important because without this, the user will be unable to view anything in any form. This creates the locations for pixels which are needed to be displayed.

2.3.29 Librarian Pack Module (GUI Layer)
This module is for packing data in a format that can be read by any other part of the Device subsystems. Maintaining a proper global format allows the system to be built much faster by being able to expect where data can be divided and used within the format.

2.3.30 Librarian Unpack Module (GUI Layer)
This module is for unpacking data in a format that can be read by any other part of the Device subcomponents in the GUI layer. Maintaining a proper global format allows the system to be built much faster by being able to expect where data can be divided and used within the format.

2.3.31 Librarian Data Manager Module (GUI Layer)
Data Manager is responsible for receiving data and sending it where it needs to go. This is an important module as it controls the flow of information and decides where it must be placed.

2.3.32 Librarian Net Command Module (GUI Layer)
This module is responsible for giving the command to be sent over TCP/IP on the network card.

2.3.33 File Storage Module (GUI Layer)
This module is to prepare the data for being stored on the system in a file. This gives the user the ability to have an alternative way to save his data.

2.3.34 External Storage Module (GUI Layer)
Being an alternative to file storage, an SD card is also available to be saved to. This allows the user mobility.
3 Design Overview

3.1 Introduction

The design of the Sliding Profiler encompasses physical, hardware and software components. Software components include binary instructions running on the device hardware as well as a client PC application. In general, the physical platform, hardware electronics and the embedded software are sufficient for the stand-alone operation of the device. In order to provide configuration, real time data streaming and historical data retrieval capabilities, the client PC application is necessary.

3.2 Physical Platform

The physical platform is composed of a sled, hardware mounts and paver mounts. TBGL will focus our efforts on the sled and hardware mounts. Instead of re-creating a paver mount, the sled will provide the mounting brackets necessary to interface with the existing paver mount.

3.3 Hardware/Electronics Processor component

TBGL has selected the PIC24EP512GP806 MCU from MICROCHIP.com and will build all circuits necessary to power and program the MCU. This processor provides 53 MB memory, 512 Kbytes of program memory, a temperature range of -45 to 125 degrees Celsius, 64 pins with 53 as I/O, 2 separate A/D converters each with 24 channels, 4UART, 4SPI, 2 I2C, and comes in a TQFP form factor.

3.4 Programming Language selection

The embedded software will be written in C with MICROCHIP’s tool chain designed for their embedded MCU. The client PC application will be written in Java to increase compatibility across pc platforms.

There are three scenarios for connectivity between the embedded device and the client PC application. These circumstances, and the design consideration for them, are explained as follows:

3.4.1 General Connectivity

We will use the RN-171 802.11g wireless chip and the ENC28j60 wired network chip to provide network interfaces. These interfaces will be placed into infrastructure mode and connect to a wired network or a wireless network with a broadcasting SSID. To interface with the device, client PC’s will join that network. All communications will occur in this manner. At this point, no consideration is given to security, though that may be a future upgrade.
3.4.2 Commands from PC to Embedded Device

The Client application will act as a server. The embedded device and the client pc will establish network sockets and maintain connectivity over the established network. The client pc will handle all streaming data from the embedded device once the network is established. The embedded device will periodically check incoming packets for commands such as data retrieval.

3.4.3 Real-time Data Retrieval

When operating, the device will provide real-time data by using the subnet broadcast address on a port specified during project implementation. The client PC application will listen for TCP/IP traffic on the specified port.
4 Input Layer

4.1 Design Overview

This layer contains inputs from such areas as sensors, power, and UIC. The sensors will be the main source of input as it contains the GPS, temperature, distance, and inclination. All these inputs will then be packaged and sent to the Process layer. This layer also receives user inputs from the GUI layer as requests for stored data, and updates to the UIC.

The main goal is to maintain operations through collecting data and transmit data effectively.
4.2 Sensors

4.2.1 Thermometer

4.2.1.1 Module Description
The thermometer module contains the internal sensor to measure the temperature of the inside of the hardware enclosure. This data is necessary to the system to identify when the system is too hot (overheating) or too cold, and to determine how to manage the system fans.

4.2.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermometer</td>
<td>A/D Converter</td>
<td>Analog value of internal temperature of the hardware enclosure.</td>
</tr>
</tbody>
</table>

4.2.1.3 Data
Internal temperature of the hardware enclosure – The output of this module is a 5V Analog signal.

4.2.1.4 Processing/Pseudo code
None.

4.2.2 GPS

4.2.2.1 Module Description
The GPS module contains the physical GPS of the system, which produces a GPS location of the device. This data is necessary to the system to track the location and progress of the product during its use.

4.2.2.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>A/D Converter</td>
<td>Digital signal containing GPS location.</td>
</tr>
</tbody>
</table>

4.2.2.3 Data
The output of this module is a digital signal of the GPS location using the NMEA format.
4.2.2.4 Processing/Pseudo code
None.

4.2.3 Inclinometer

4.2.3.1 Module Description
The inclinometer module contains the inclinometer sensor which detects changes in inclination and provides it as input to the system. This data is necessary to use in the bump detection algorithm to determine bumps in the concrete.

4.2.3.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inclinometer</td>
<td>A/D Converter</td>
<td>Analog value of the instantaneous change in inclination.</td>
</tr>
</tbody>
</table>

4.2.3.3 Data
This module outputs a 5V analog signal representing the instantaneous change in platform inclination.

4.2.3.4 Processing/Pseudo code
None.

4.2.4 Distance Encoder

4.2.4.1 Module Description
The distance encoder module contains the sensor that provides the rotation of the encoder sensor. The encoder will be connected to the distance wheel that turns as the platform slides along the concrete. So as the distance wheel turns the distance encoder changes rotation and indicates this as digital input to the system. This data is necessary to match with the inclination of the platform to provide for analysis.

4.2.4.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Encoder</td>
<td>A/D Converter</td>
<td>Digital signal of the encoder rotational position.</td>
</tr>
</tbody>
</table>
4.2.4.3 Data

Two square waves in quadrature with channel A leading channel B for clockwise shaft rotation indicates forwards and backwards pulses.

4.2.4.4 Processing/Pseudo code

None.

4.3 A/D Converter

4.3.1 A/D Converter

4.3.1.1 Module Description

The A/D converter module is responsible for converting signals from the sensor subsystem into digital formats to be used by the system. This includes analog to digital conversion as well as digital format conversion. The A/D converter will wait for a detected change on the distance encoder which indicates an encoder pulse. When this pulse occurs the module will send the necessary data from the sensors (in the format required) to the system.

4.3.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermometer</td>
<td>A/D Converter</td>
<td>Analog value of internal temperature of the hardware enclosure.</td>
</tr>
<tr>
<td>GPS</td>
<td>A/D Converter</td>
<td>Digital signal containing GPS location.</td>
</tr>
<tr>
<td>Inclinometer</td>
<td>A/D Converter</td>
<td>Analog value of the instantaneous change in inclination.</td>
</tr>
<tr>
<td>Distance Encoder</td>
<td>A/D Converter</td>
<td>Digital signal of the encoder rotational position.</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital temperature value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital GPS value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital change in inclination value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>One bit determining forward or backward pulse</td>
</tr>
</tbody>
</table>
4.3.1.3 Data
The data input to the module is determined by the sensors subsystem. The data output is as follows:

- Digital temperature value – one byte
- Digital GPS value – GPX format
- Digital change in inclination value – Double Word (32 bits)
- Encoder pulse bit – one bit (1 indicating forward pulse, 0 indicating backwards)

4.3.1.4 Processing/Pseudo code

<table>
<thead>
<tr>
<th>Read encoder value</th>
</tr>
</thead>
<tbody>
<tr>
<td>While encoder value not changed</td>
</tr>
<tr>
<td>Add current inclinometer value to inclinometer sum</td>
</tr>
<tr>
<td>If new encoder value &gt; previous value</td>
</tr>
<tr>
<td>AnalogToDigital(Temperature)</td>
</tr>
<tr>
<td>NMEAtoGPX(GPS)</td>
</tr>
<tr>
<td>Send(Temperature, GPS, Inclinometer Sum, Forward Pulse)</td>
</tr>
<tr>
<td>Set inclinometer sum = 0</td>
</tr>
<tr>
<td>If new encoder value &lt; previous value</td>
</tr>
<tr>
<td>AnalogToDigital(Temperature)</td>
</tr>
<tr>
<td>NMEAtoGPX(GPS)</td>
</tr>
<tr>
<td>Send(Temperature, GPS, Inclinometer Sum, Backward Pulse)</td>
</tr>
<tr>
<td>Set inclinometer sum = 0</td>
</tr>
</tbody>
</table>

4.4 UIC

4.4.1 UIC

4.4.1.1 Module Description
The UIC module receives and stores values to be used as variables in the bump detection algorithm. The values provided are the Bias Window Size, Running Average Window Size and the Threshold. These values are provided to the algorithm as a struct during operation.

4.4.1.2 Inputs/Outputs

Table 8 - UIC Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
</table>

015 January 2013 @ 11:10:00 AM 27 Team TBGL
### 4.4.1.3 Data
The data received and output is defined as:
- Bias window size – word (16 bits)
- Running average window size – word (16 bits)
- Threshold – word (16 bits)

### 4.4.1.4 Processing/Pseudo code
Default values for Bias, Ravg and Thresh during start
When data received, update Bias, Ravg and Thresh
When data requested, send Bias, Ravg and Thresh

### 4.5 Data Handler

#### 4.5.1 Pack

#### 4.5.1.1 Module Description
The pack module packs data that will be sent to another layer. This data handler packs the raw data from the A/D converter and the algorithm parameters from the UIC and sends them both to the process layer.

#### 4.5.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital temperature value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital GPS value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>Digital change in inclination value</td>
</tr>
<tr>
<td>A/D Converter</td>
<td>Pack</td>
<td>One bit determining forward or backward pulse</td>
</tr>
</tbody>
</table>
4.5.1.3 Data

The data received is defined as:

- Bias window size – word (16 bits)
- Running average window size – word (16 bits)
- Threshold – word (16 bits)
- Digital temperature value – one byte
- Digital GPS value – GPX format
- Digital change in inclination value – Double Word (32 bits)
- Encoder pulse bit – one bit (1 indicating forward pulse, 0 indicating backwards)
- Request for stored data – 1 byte

The data output is defined as:

- AlgoParam struct, containing Bias, Ravg and Threshold – 3 words (48 bits)
- RawData struct, containing Temp, GPX, Inclination Sum and encoder pulse
- Request for stored data – 1 byte

4.5.1.4 Processing/Pseudo code

None.

4.5.2 Unpack

4.5.2.1 Module Description

The unpack module receives JSON packets from the Net Comm module and interprets what they contain. Depending on the type of data contained in the JSON packet, this module sends the data to either the UIC module or the CMD module. The data that will be sent to the CMD module will be requests for stored data.
4.5.2.2 Inputs/Outputs

Table 10 - Unpack (Input Layer) Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Comm</td>
<td>Unpack</td>
<td>JSON packet from GUI</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>Request for Stored Data</td>
</tr>
<tr>
<td>Unpack</td>
<td>UIC</td>
<td>Bias window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>UIC</td>
<td>Running average window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>UIC</td>
<td>Threshold</td>
</tr>
</tbody>
</table>

4.5.2.3 Data

The data received is defined as:

- JSON packet from GUI – opcode (1 byte) can be either request for stored data or updates to UIC. Data (3 words) is either null or new UIC data.

The data output is defined as:

- Request for stored data – 1 byte
- Bias window size – word (16 bits)
- Running average window size – word (16 bits)
- Threshold – word (16 bits)

4.5.2.4 Processing/Pseudo code

Deserialize(JSON)
If opcode = request for stored data
    Send request to Pack
If opcode = update to UIC
    SendtoUIC(Bias, Ravg, Threshold)

4.6 Net Comm

4.6.1 Net Comm

4.6.1.1 Module Description

The Net Comm module implements the communication over the wireless network with the GUI PC. This Net Comm receives the TCP/IP packets from the GUI and translates them to JSON packets.
4.6.1.2 Inputs/Outputs

Table 11 - Net Comm (Input Layer) Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Comm (GUI)</td>
<td>Net Comm (Input)</td>
<td>TCP/IP packet</td>
</tr>
<tr>
<td>Net Comm (Input)</td>
<td>Unpack</td>
<td>JSON packet from GUI</td>
</tr>
</tbody>
</table>

4.6.1.3 Data
The data received is defined as:
- TCP/IP packet – wireless packet sent over network.

The data output is defined as:
- JSON packet from GUI – opcode (1 byte) can be either request for stored data or updates to UIC. Data (3 words) is either null or new UIC data.

4.6.1.4 Processing/Pseudo code

Read TCP/IP packets sent from GUI Net Comm.
NIC changes to SPI
Bit Banger converts it to JSON packets

4.7 Power Input

4.7.1 Power Input

4.7.1.1 Module Description
This module provides power to the system. It contains the power switch with which the user interacts with, and provides power to all necessary components. For detailed description, refer to Power Design section.

4.7.1.2 Inputs/Outputs

4.7.1.3 Data
There are no data flows in the Power Input module.

4.7.1.4 Processing/Pseudo code

Loop
If PowerSwitch = ON
    Provide power to system
Else
| Block power to system |
5 Processing Layer

5.1 Design Overview

The processing layer provides the function of processing raw data to produce meaningful results. The layer primarily runs the bump detection algorithm. The processing layer receives raw sensor data from the input layer, processes it, and sends the processed data (including bump detection information) to both the storage layer and the output layer.

![Diagram of the Processing Layer]

**Figure 5 - Process Layer**
5.2 Data Handler

5.2.1 Unpack

5.2.1.1 Module Description

The unpack module receives intermittent AlgoParam and RawData structs from the Input layer. This module forwards these packets to the pack module, and also sends the required data to the Bump Detection module for processing.

5.2.1.2 Inputs/Outputs

Table 12 - Unpack (Process Layer) Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pack (Input)</td>
<td>Unpack (Process)</td>
<td>Raw Data struct: containing Temp, GPS, Inclination Sum and encoder pulse</td>
</tr>
<tr>
<td>Pack (Input)</td>
<td>Unpack (Process)</td>
<td>Algo Param struct: containing Bias, Ravg and Threshold</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>Raw Data struct: containing Temp, GPS, Inclination Sum and encoder pulse</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>Algo Param struct: containing Bias, Ravg and Threshold</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Digital change in inclination value</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Bias window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Running average window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Threshold</td>
</tr>
</tbody>
</table>

5.2.1.3 Data

The data received is defined as:

- RawData struct, containing Temp, GPS, Inclination Sum and encoder pulse
- AlgoParam struct, containing Bias, Ravg and Threshold – 3 words (48 bits)

The data output is defined as:

- RawData struct, containing Temp, GPS, Inclination Sum and encoder pulse
- AlgoParam struct, containing Bias, Ravg and Threshold – 3 words (48 bits)
- Digital change in inclination value – Double Word (32 bits)
• Bias window size – word (16 bits)
• Running average window size – word (16 bits)
• Threshold – word (16 bits)

5.2.1.4 Processing/Pseudo code

```plaintext
Receive AlgoParam
Receive RawData
SendToPack(AlgoParam, RawData)
```

5.2.2 Pack

5.2.2.1 Module Description
The Pack module packages the processed data with the corresponding raw data to be sent. Raw and processed data packets are sent to the storage layer and the output layer.

5.2.2.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>Raw Data struct: containing Temp, GPS, Inclination Sum and encoder pulse</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>Algo Param struct: containing Bias, Ravg and Threshold</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Bump detect bit</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Raw variable</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Profile variable</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Diff variable</td>
</tr>
<tr>
<td>Pack</td>
<td>Unpack (Storage)</td>
<td>RawAndProData struct</td>
</tr>
<tr>
<td>Pack</td>
<td>Unpack (Output)</td>
<td>RawAndProData struct</td>
</tr>
</tbody>
</table>

5.2.2.3 Data
The data received is defined as:

• AlgoParam struct, containing Bias, Ravg and Threshold – 3 words (48 bits)
- RawData struct, containing Temp, GPS, Inclination Sum and encoder pulse
- Bump detect bit – 1 bit
- Raw variable – QWORD (64 bits)
- Profile variable – QWORD (64 bits)
- Diff variable – QWORD (64 bits)

The data output is defined as:
- RawAndProData struct, containing Raw Data struct, AlgoParam struct, bump detect bit, Raw variable, Profile variable, Diff variable

5.2.2.4 Processing/Pseudo code

Receive AlgoParam
Receive RawData
Receive Processed Data
Create RawAndProData(AlgoParam, RawData, ProData)
SendToOutput(RawAndProData)
SendToStorage(RawAndProData)

5.3 Algorithm

5.3.1 Bump Detection

5.3.1.1 Module Description

The Bump Detection module implements the bump detection algorithm for the system. It receives the raw data and algorithm parameters and detects when a bump has occurred. When a bump is found, the bump detection bit of the data being output is set high.

5.3.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Digital change in inclination value</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Bias window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Running average window size</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Detection</td>
<td>Threshold</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Bump detect bit</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>Pack</td>
<td>Raw variable</td>
</tr>
</tbody>
</table>
5.3.1.3 Data

The data received is defined as:

- Digital change in inclination value – Double Word (32 bits)
- Bias window size – word (16 bits)
- Running average window size – word (16 bits)
- Threshold – word (16 bits)

The data output is defined as:

- Bump detect bit – 1 bit
- Raw variable – QWORD (64 bits)
- Profile variable – QWORD (64 bits)
- Diff variable – QWORD (64 bits)

5.3.1.4 Processing/Pseudo code

1. Algorithm
   1. Distance
      1. Sum of the change in X's
   2. Profile (Height)
      1. Sum of the Y's
      2. Sum of change in X's * sin(of the angles)
   3. Bias
      1. B is the window of Y values
      2. Sum all the Y's in the window
      3. Then divide that sum by the window
   4. AdjProfile
      1. Current Y minus the current bias
   5. Running Average
      1. R is a moving average window of length
      2. Sum the Y values in the sliding window
         1. Divide that sum by R
   6. Adjrunningaverage
      1. R is a moving average window of length
      2. Sum the adjprofile values
         1. Divide that sum by R
7. Difference
   1. For your current position
   2. Take the difference of AdjProfile and Adjrunning average
   3. Then take the absolute value of this result

2. The algorithm uses 150 mils
   1. This is the difference allowed in the Y direction
   2. 150mils, 150/1000=0.15 inches

3. What is Mils
   1. 1in equals 1000mils
   2. 1mil equals 1/1000 of an inch

4. UIC values needed for DDS
   1. Y threshold (in mils)
   2. Bias window length (in feet)
   3. R window length (in feet)
6 Storage Layer

6.1 Design Overview

The storage layer manages the physical storage onboard the system, reading and writing raw and processed data. It organizes the file system on the physical storage, keeps track of sessions and stores the data. The storage layer allows the user to get historic data from the system, so that they don’t have to analyze it immediately.

![Figure 6 - Storage Layer](image-url)
6.2 Data Handler

6.2.1 Unpack

6.2.1.1 Module Description
The unpack module receives raw and processed data as a struct from the Process layer, and turns it into a bit field to be written on disk as one entry. It passes this bit field to the session handler. The unpack module also receives requests for stored data which it forwards to the session handler.

6.2.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pack (Process)</td>
<td>Unpack (Storage)</td>
<td>RawAndProData struct</td>
</tr>
<tr>
<td>Pack (Input)</td>
<td>Unpack</td>
<td>Request for Stored Data</td>
</tr>
<tr>
<td>Unpack</td>
<td>Session Handler</td>
<td>RawAndProData bit field</td>
</tr>
<tr>
<td>Unpack</td>
<td>Session Handler</td>
<td>Request for Stored Data</td>
</tr>
</tbody>
</table>

6.2.1.3 Data
The data received is defined as:

- RawAndProData struct, containing Raw Data struct, AlgoParam struct, bump detect bit, Raw variable, Profile variable, Diff variable
- Request for stored data – 1 byte

The data output is defined as:

- RawAndProData bit field
- Request for stored data – 1 byte

6.2.1.4 Processing/Pseudo code

Convert RawAndProData struct to bit field from Pack(Process)
Unpack Request for Stored Data from Pack(Input)
Send to Session Handler
6.2.2  Pack

6.2.2.1  Module Description
The Pack module packs outgoing stored data in the JSON format to pass to the Output Layer. This module is only active after a request for stored data, when the system is outputing stored data.

6.2.2.2  Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Handler</td>
<td>Pack</td>
<td>RawAndProData bit field</td>
</tr>
<tr>
<td>Pack</td>
<td>Unpack (Output)</td>
<td>JSON containing RawAndProData</td>
</tr>
</tbody>
</table>

6.2.2.3  Data
The data received is defined as:
- RawAndProData bit field

The data output is defined as:
- JSON containing one RawAndProData packet

6.2.2.4  Processing/Pseudo code
Receive RawAndProData bit field from Session Handler
Pack into JSON
Send to Unpack Output

6.3  Data Manager

6.3.1  Session Handler

6.3.1.1  Module Description
The session handler module keeps track of sessions of recorded data. A session refers to one use of the system (ie. from power on to power off). Sessions are kept track of in the stored data so that when data is requested, only data from the current session will be sent. The session handler keeps track of these sessions by adding one bit to each stored data entry. This bit is reversed whenever the system is turned on, so sequential data entries with similar session bits indicate that they are from the same session. The session handler handles the physical storage, putting data and getting requested stored data. If the physical storage is not initialized, the session handler sends the command to do so.
6.3.1.2 Inputs/Outputs

Table 17 - Session Handler Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Session Handler</td>
<td>Request for stored data</td>
</tr>
<tr>
<td>Unpack</td>
<td>Session Handler</td>
<td>RawAndProData bit field</td>
</tr>
<tr>
<td>Physical Storage</td>
<td>Session Handler</td>
<td>RawAndProData with session bit</td>
</tr>
<tr>
<td>Session Handler</td>
<td>Initialize</td>
<td>Initialize Command</td>
</tr>
<tr>
<td>Session Handler</td>
<td>Physical Storage</td>
<td>RawAndProData with session bit</td>
</tr>
</tbody>
</table>

6.3.1.3 Data

The data received is defined as:
- Request for stored data – 1 byte
- RawAndProData bit field
- RawAndProData with session bit

The data output is defined as:
- Initialize Command – 1 byte
- RawAndProData with session bit

6.3.1.4 Processing/Pseudo code

When Power ON
Reverse session bit
If physical storage not initialized
    Send Command to initialize
Loop
    If RawAndProData received
        Write RawAndProData to File Counter location on disk
        Increment File Counter
    If Request for stored data received
        Reverse through physical storage until session bit changes
        While at file location before File Counter
        Get data at location
6.3.2 Initialize

6.3.2.1 Module Description
The initialize module implements the file system on the physical storage when given the command. The file system used will be FAT32. The file system initializes the physical storage, preparing it to be used by the system.

6.3.2.2 Inputs/Outputs

Table 18 - Initialize Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Handler</td>
<td>Initialize</td>
<td>Initialize Command</td>
</tr>
<tr>
<td>Initialize</td>
<td>Physical Storage</td>
<td>FAT32</td>
</tr>
</tbody>
</table>

6.3.2.3 Data
The data received is defined as:
- Initialize Command – 1 byte

The data output is defined as:
- FAT32 – Formatting of FAT32 file system

6.3.2.4 Processing/Pseudo code

If received a command to initialize from Session Handler

Initialize to Physical Storage

6.4 Physical Storage

6.4.1 Physical Storage

6.4.1.1 Module Description
The physical storage module contains the means for storing data in the system. The physical storage will be a removable SD card. The system will initialize the physical storage by formatting in FAT32. Once formatted, data will be intermittently written to and read from the physical storage.

6.4.1.2 Inputs/Outputs

Table 19 - Physical Storage Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
</table>
### Initialize
- **Physical Storage**
- **FAT32**

### Session Handler
- **Physical Storage**
- **RawAndProData with session bit**

### Physical Storage
- **Session Handler**
- **RawAndProData with session bit**

#### 6.4.1.3 Data
The data received is defined as:
- FAT32 – Formatting of FAT32 file system
- RawAndProData with session bit

The data output is defined as:
- RawAndProData with session bit

#### 6.4.1.4 Processing/Pseudo code
```plaintext
If received a command from initialize
  Initialize
If received a command to write from session handler
  Write to the Physical Storage
If received a command to read from session handler
  Read from the Physical Storage and send to Session Handler
```
7 Output Layer

7.1 Design Overview

The output layer provides output regarding bump detection, error conditions, and normal operation to the user. It also controls networking with the GUI layer to provide information such as raw and processed data as output to the GUI.

Figure 7 - Output Layer
7.2 Data Handler

7.2.1 Unpack

7.2.1.1 Module Description
The unpack module receives RawAndProData from the Process and Output Layer. This module takes the data necessary to send to hardware out. It also sends the data to the pack module so it can be sent to the GUI.

7.2.1.2 Inputs/Outputs

Table 20 - Unpack (Storage) Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pack (Process)</td>
<td>Unpack</td>
<td>RawAndProData struct</td>
</tr>
<tr>
<td>Pack (Storage)</td>
<td>Unpack</td>
<td>JSON containing RawAndProData</td>
</tr>
<tr>
<td>Unpack</td>
<td>Bump Light</td>
<td>Bump bit</td>
</tr>
<tr>
<td>Unpack</td>
<td>Error Lights</td>
<td>Error Code</td>
</tr>
<tr>
<td>Unpack</td>
<td>Fan Direction</td>
<td>Fan direction</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>RawAndProData struct</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>JSON containing RawAndProData</td>
</tr>
</tbody>
</table>

7.2.1.3 Data
The data received is defined as:

- RawAndProData struct, containing Raw Data struct, AlgoParam struct, bump detect bit, Raw variable, Profile variable, Diff variable
- JSON containing one RawAndProData packet

The data output is defined as:

- Bump bit – 1 bit
- Error Code – 8 bit error code
- Fan direction – 8 bits
- RawAndProData struct, containing Raw Data struct, AlgoParam struct, bump detect bit, Raw variable, Profile variable, Diff variable
- JSON containing one RawAndProData packet
### 7.2.1.4 Processing/Pseudo code

<table>
<thead>
<tr>
<th>Condition</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>If JSON received from Storage</td>
<td>Send JSON to Pack</td>
</tr>
<tr>
<td>If RawAndProData received from Process</td>
<td>SendToBumpLight(RawAndProData.BumpBit) SendToErrorLights(RawAndProData.ErrorCode) SendToFanDirection(RawAndProData.FanDirection) SendToPack(RawAndProData)</td>
</tr>
</tbody>
</table>

### 7.2.2 Pack

#### 7.2.2.1 Module Description

The pack module creates JSON packets that will be sent to the GUI layer, and hands them off to the Net Comm module to get it there. These JSON packets can be either live data or requested stored data.

#### 7.2.2.2 Inputs/Outputs

**Table 21 - Pack (Output Layer) Inputs/Outputs**

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>RawAndProData struct</td>
</tr>
<tr>
<td>Unpack</td>
<td>Pack</td>
<td>JSON containing RawAndProData</td>
</tr>
<tr>
<td>Pack</td>
<td>Net Comm</td>
<td>JSON containing RawAndProData</td>
</tr>
</tbody>
</table>

#### 7.2.2.3 Data

The data received is defined as:

- RawAndProData struct, containing Raw Data struct, AlgoParam struct, bump detect bit, Raw variable, Profile variable, Diff variable
- JSON containing one RawAndProData packet

The data output is defined as:

- Bump bit – 1 bit
- JSON containing one RawAndProData packet
7.2.2.4 Processing/Pseudo code

<table>
<thead>
<tr>
<th>If RawAndProData received</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create JSON containing RawAndProData</td>
</tr>
<tr>
<td>SendToNetComm(JSON)</td>
</tr>
<tr>
<td>If JSON received</td>
</tr>
<tr>
<td>SendToNetComm(JSON)</td>
</tr>
</tbody>
</table>

7.3 Hardware Out

7.3.1 Bump Light

7.3.1.1 Module Description

The Bump Light is used to indicate to the user when a bump is detected in the data. The light is comprised of 1 led light. The light receives one bit that indicates when the led should be illuminated.

7.3.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Bump Light</td>
<td>Bump bit</td>
</tr>
</tbody>
</table>

7.3.1.3 Data

The data received is defined as:

- Bump bit – 1 bit

7.3.1.4 Processing/Pseudo code

None.

7.3.2 Fan Direction

7.3.2.1 Module Description

The fan direction module contains the fan and hardware to control the fans to cool the system hardware. The module receives one byte that indicates how the fans should be operated.

7.3.2.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Fan Direction</td>
<td>Fan direction</td>
</tr>
</tbody>
</table>
7.3.2.3 Data
The data received is defined as:

- Fan direction – 8 bits

7.3.2.4 Processing/Pseudo code

<table>
<thead>
<tr>
<th>If too hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>One blows in. One blows out</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If too cold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both blows out</td>
</tr>
</tbody>
</table>

7.3.3 Error Lights/Heartbeat

7.3.3.1 Module Description
The Error Lights/Heartbeat module contains an 8 led array. These leds are used to indicate error condition codes as well as the heartbeat to the user(s). The heartbeat is indicated as one led moving progressively along the led array, which is controlled with the error code.

7.3.3.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>Error Lights</td>
<td>Error Code</td>
</tr>
</tbody>
</table>

7.3.3.3 Data
The data received is defined as:

- Error Code – 8 bit error code (0-255 combinations of error)

7.3.3.4 Processing/Pseudo code

<table>
<thead>
<tr>
<th>If error bit is set high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicate Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Else</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicate Normal</td>
</tr>
</tbody>
</table>

7.4 Net Comm

7.4.1 Net Comm
This receives JSON packets from pack module and coverts them into TCP/IP packets to be send to the GUI.
7.4.1.1 Module Description

7.4.1.2 Inputs/Outputs

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pack</td>
<td>Net Comm (Output)</td>
<td>JSON containing RawAndProData</td>
</tr>
<tr>
<td>Net Comm (Output)</td>
<td>Net Comm (GUI)</td>
<td>TCP/IP packet</td>
</tr>
</tbody>
</table>

7.4.1.3 Data

The data received is defined as:
- JSON containing one RawAndProData packet

The data output is defined as:
- TCP/IP packet – wireless packet sent over network.

7.4.1.4 Processing/Pseudo code

- Get JSON Packets from pack module
- Bit Banger through SPI sends to NIC
- NIC converts to TCP/IP packets to be send to GUI Net Comm
8 GUI Layer

8.1 Design Overview

The GUI layer is the only layer of the system that runs independently of the physical platform. The GUI layer operates on a PC connected to the system over a network. Both raw and processed data can be displayed and analyzed through the GUI, providing an easy way for the user to check the results. It also allows the user to provide input to configure the device.

![GUI Layer Diagram](image)
8.2 Network

8.2.1 Pack

8.2.1.1 Module Description
This module is for packing data in a format that can be read by any other part of the Device subsystems. Maintaining a proper global format allows the system to be built much faster by being able to expect where data can be divided and used within the format.

8.2.1.2 Inputs/Outputs
Pack has one input which is from Net CMD module as a struct. The only output from this module is to the Net Comm. This data is set in the form of a JSON packet.

8.2.1.3 Data
This data contains information from the user for the algorithm parameters that must be stored on the device in the input layer therefore requiring the data to be sent via TCP/IP which must first be packed.

8.2.2 Unpack

8.2.2.1 Module Description
This module is for unpacking data in a format that can be read by any other part of the Device subcomponents in the GUI layer. Maintaining a proper global format allows the system to be built much faster by being able to expect where data can be divided and used within the format.

8.2.2.2 Inputs/Outputs
Unpack has one input and one output. The output data line is in the format of a JSON packet which is then delivered to the Data Manager module. The input is from the Net Comm which is in the form of a JSON packet due to being released from TCP/IP data packet.

8.2.2.3 Data
The Data that is being delivered is raw and processed data as well as algorithm parameters which is being streamed in live from the device to the GUI.

8.2.3 Net Comm.

8.2.3.1 Module Description
The Net Comm module is the line between TCP/IP communication and the GUI layer. It handles all communication in and out that requires being sent to the physical device from the user interface.
8.2.3.2 Inputs/Outputs

This module has one data line in from TCP/IP and two data lines out which are to the unpack module and new algorithm parameters to the input layer on the device.

8.2.3.3 Data

The data that is being received is raw and processed data through TCP/IP. The data being sent is raw and processed data to be sent to be unpacked and new algorithm parameters which are sent to the input layer via TCP/IP.

8.3 Librarian

8.3.1 Data Manager

8.3.1.1 Module Description

Data Manager is responsible for receiving data and sending it where it needs to go. This is an important module as it controls the flow of information and decides where it must be placed.

8.3.1.2 Inputs/Outputs

The data manager receives three data lines in the format of a JSON packet from an SD card, loaded from a file, or being unpacked through TCP/IP. Two data lines are then processed and sent to either file storage or the visual display. The final data line being delivered is to the external storage.

8.3.1.3 Data

The data that’s being received is a JSON packet which is formatted in such a way that can be read easily by different modules because it is standardized throughout the system. Raw and processed data will be delivered through multiple JSON packets and then processed and sent to wherever requested.

8.3.2 Net CMD

8.3.2.1 Module Description

This module is responsible for giving the command to be sent over TCP/IP on the network card.

8.3.2.2 Inputs/Outputs

This module has one input from the Input module and one output which is the pack module.

8.3.2.3 Data

The data being received and sent is in the form of a struct. All information is only received and sent with the command to the Net Comm module on where to be sent.
8.4   Storage

8.4.1   File Storage

8.4.1.1   Module Description
This module is to prepare the data for being stored on the system in a file. This gives the user the ability to have an alternative way to save his data.

8.4.1.2   Inputs/Outputs
The input for file storage is from the data manager while the data that is output is then sent to the system for storage.

8.4.1.3   Data
Both input and output is raw and processed data including algorithm parameters. Both data lines are structs.

8.4.2   Ex. Storage

8.4.2.1   Module Description
Being an alternative to file storage, an SD card is also available to be saved to. This allows the user mobility.

8.4.2.2   Inputs/Outputs
The input for external storage is from the data manager while the data that is output is then sent to the SD card for storage.

8.4.2.3   Data
Both input and output is raw and processed data including algorithm parameters. Both data lines are structs.

8.5   Display

8.5.1   Visual

8.5.1.1   Module Description
This module is in a sense the graphics card of the GUI layer. It is responsible for sending graphical information to the user’s computer screen. This is important because without this, the user will be unable to view anything in any form. This creates the locations for pixels which are needed to be displayed.

8.5.1.2   Inputs/Outputs
With one input from the data manager and one output to the pc monitor, the module is very basic and straightforward.
8.5.1.3 Data

Raw and processed data in the format of a struct is being received to this component. The data is then processed into a VGA format which can then be displayed to the user.

8.5.2 Input

8.5.2.1 Module Description

Although having a graphical interface is nice and useful. Having the ability to interact with the interface and send personalized requirements to the device is even better. This module is for interacting with the user.

8.5.2.2 Inputs/Outputs

Currently only one data is being received from the visual display and is then sent to the Net CMD for output.

8.5.2.3 Data

On the input side we have information such as, Bias, Running average, CMD, and Threshold. All information being received is formatted to a struct and is then forwarded to the Net Cmd module in the same fashion.
9 Error Handling

9.1 Description
Error handling will be the sole means of handling critical and normal errors. If the module is unable to handle the error itself, then it must hand off to the Error Handler. The device will work on a trap / interrupt system for error handling.

9.2 Modules affected
- A/D Converter (Input)
- UIC (Input)
- NetComm (Input/Output)
- DataManager (Storage Layer)

9.3 Inputs / Outputs
None.

9.4 Data
Uses Global Error Variable - Byte

9.5 Processing / Pseudo Code
Start Module
   Discover Error
   Attempt to handle error
   If Fail
      start
         Update Global Error variable with corresponding error code(byte)
         Call Trap function
      end
   end
End Module
Trap Function
   While error byte code true
   Start
       Check error byte code
       Display error byte code at module Hardware Out
       Attempt to handle error and clear error
   End
End Trap Function
10 Platform Design

Figure 9 - Platform Design Top View

Figure 10 - Platform Design Side View

Figure 11 - Platform Design Front View
11 Hardware Design

11.1 Introduction
The Hardware system describes the physical electronic devices as well as the supporting circuitry to allow them to function. It also describes the system for with the embedded software (described later) is stored and run on. It is responsible for providing all communication connections between devices as well as necessary power and ground connections for every device on board.

11.2 Design Considerations
The following is a list of considerations in the design:

1. There will be a +12v DC source provided from the job site.
2. The voltage source will provide sufficient current to run the device.
3. The PIC24EP512GP806 requires a voltage source of 3.0 – 3.6 volts with a max of 320 mA of current.
4. The electronics being designed to support the system should be neat and fit inside the electronics housing.
5. The electronics inside the housing should have a high heat and cold tolerances to ensure continued operation with one exhaust fan and one variable direction fan for intake and exhaust mode mounted to the housing.

11.3 Hardware Design
The hardware is a series of circuits that are dependent upon each other to provide their respective resources to the system. The Hardware system is designed in such a way that each subsystem is designed independently but when joined together, they provide support for the Software system that will be running on them. It is important to note that all components not implemented on the printed circuit board (PCB) will be connected to the PCB via screw terminals designated on the schematics as J#.

11.3.1 Electronics Housing
The electronics housing that has been selected is a weather proof plastic box that will be mounted to the plate in Figure 11. It measures 7.5” x 14” x 4” excluding the lid. We have selected this box due to its popular use by the sponsor.
11.3.2 Sliding Profiler Electronics

All electronic components in this system have been chosen because of their specific purpose in this system. At the center of our components will be the PIC24EP512GP806 MCU. All conversion from analogue, data serialization/deserialization, network communications, storage communications, and the bump detection algorithm will be processed, handled, and calculated on this MCU.

11.3.2.1 Physical Storage Subsystem

This subsystem will consist of an SD/MMC card reader that communicates over the SPI bus that is native to the MCU. It will be formatted by the DataManager with FAT32 to ensure the interoperability of the SD/MMC card with the client PC as well.

11.3.2.2 Sensor Subsystem

This subsystem will consist of several A/D converters that will provide necessary communications from the sensors to the MCU. It will be responsible for converting the voltage signals into digital values so that the MCU can convert the values to their respective measurements.

11.3.2.3 NetComm Subsystem

This subsystem will contain the RN-171 802.11g Wireless network chip and an Ethernet chip that will provide us with a medium to communicate directly with the client. It will be configured as an infrastructure device. We will use TCP/IP for reliable data communication and streaming of the bump detection data.

11.3.2.4 Algorithm Subsystem

The MCU is a PIC24EP512GP806 from MicroChip. This MCU has a 16bit modified Harvard architecture with an enhanced instruction set. The MCU operates natively at 7.37MHz. However, we will use a 40MHZ external clock for normal operation.
11.3.3 Power Subsystem

The power management subsystem is designed to take in a 12v unregulated DC source and convert that voltage to 3.3V, +12V, +15V, and -15V regulated DC. This is accomplished with voltage regulators and a DC-DC converter. All power will be controlled by a master switch mounted on the housing.

![Power Subsystem Diagram](image-url)
11.3.4 Net Comm (Input Layer/Output Layer)

The Net Com subsystem consists of either a wired Ethernet controller (ENC28j60) or a Wi-Fi network controller (RN-171). Raw and processed data will be streamed across the network for the GUI.

Figure 13 - Net Comm Subsystem Diagram
11.3.5 Physical Storage

The Physical Storage subsystem consists of a SD/MMC memory card reader (PRT-11362). Raw and processed data will be streamed to the memory storage.

![Physical Storage Subsystem Diagram](image)

**Figure 14 - Physical Storage Subsystem Diagram**

11.3.6 Sensor Subsystem

The sensor subsystem consists of 5 sensors. Temperature sensor (SEN-10988) 2ea, an Accu-Coder 15-T/H series distance encoder, GPS receiver (GPS-09838) and a Columbia Research Labs ±5º SI-701BHP inclinometer.
Figure 15 - Sensor Subsystem Diagram
11.3.7 Hardware Out Subsystem

The hardware out subsystem will be a series of 8 LED’s and one bright beacon light, a fan, and a bi-directionally controlled fan. The LED’s will be used to indicate various error conditions to the user. The design of this subsystem is that the LED’s connect to the 8 pins of the MCU in digital output mode. They are directly controlled by the MCU. The beacon light requires a higher current than the MCU can supply so there will be a relay used that is controlled by the MCU to flash the beacon light. The bi-directionally controlled fan will use a single pole double throw relay to switch current direction.

![Hardware Out Subsystem Diagram](image)

Figure 16 - Hardware Out Subsystem Diagram
11.4 Printed Circuit Board Design

The printed circuit board all the circuitry of the schematics above that is not explicitly stated as not included on the PCB. This PCB will include the entire electronic circuitry for the necessary operation of the sliding profiler

<Will be updated in 2-3 days>

Figure ## – PCB Design for Daughterboard
12 Hardware Components

12.1 PIC Microcontroller

12.1.1 Purpose
The purpose of the PIC Microcontroller is to facilitate data retrieval, bump detection calculations, and bump/error display.

12.1.2 Specifications
Operates between 3.0V to 3.6V, -40ºC to +125ºC, DC to 60MIPS. Uses SPI, UART, timers, hardware interrupts, hardware divide, 32bit multiply, 512KB of program memory, 24 analog channels, ICSP.

12.1.3 Inputs/Outputs
64 pin count with 53 pints input and output.

12.2 SD/MMC Card Reader

12.2.1 Purpose
The purpose of the SD/MMC Card Reader is to perform interim storage while the device is not in communication range of the client PC.

12.2.2 Specifications
50 MB/sec data transfer rate, Correction of memory-field errors, Minimum 2Gbyte of storage, -20ºC to +85ºC, card insertion/eject force: 1.40Kgf Max and 2.7V to 3.6V operation.

12.2.3 Inputs/Outputs
Input: SPI
Output: SPI

12.3 Inclinometer

12.3.1 Purpose
The purpose of the inclinometer is to track minor changes to angle of attack. With the summation of these minor angle changes we can further calculate the total degree change between distance intervals.
12.3.2 Specifications

12.3.3 Inputs/Outputs
Input: +12VDC, -12VDC
Output: GND, 0 to +5VDC

12.4 Encoder

12.4.1 Purpose
The purpose of the encoder is to track distance traveled for bump detection calculations.

12.4.2 Specifications
Accuracy within 1 arc-minute (.017°), 64 pulses per rotation, operation between -40°C to +85°C, and Two square waves in quadrature with channel A leading B for clockwise shaft rotation.

12.4.3 Inputs/Outputs
Input: +12 VDC, -12VDC
Output: GND, +5VDC

12.5 Temperature Sensor

12.5.1 Purpose
There will be two temperature sensors on the device. The purpose is to sense a potential overheat condition. The purpose of two sensors is to avoid a faulty sensor situation.

12.5.2 Specifications
A temperature sensor must sense at least -40°C to +125°C, +/- 2°C. Must work on 3.3v DC power.

12.5.3 Inputs/Outputs
Input: +VDD IN
Output: VOUT, GND

12.6 Wi-Fi Network Interface Card

12.6.1 Purpose
The purpose of the network interface card is to provide wireless data streaming to the client pc over a LAN network.
12.6.2 Specifications
Must transmit wirelessly as 2.4GHz IEEE 802.11b/g. Needs 3.3v DC Power, support Infrastructure networks, and communicates with the MCU over UART/SPI bus. Operating bounds of 3.0V to 3.6V, -40ºC to +85ºC, +/-2 ºC.

12.6.3 Inputs/Outputs
Input: SPI,GPIO,UART,VDD IN, VDD BATT
Output: 2.4 GHZ TX/RX

12.7 Ethernet Network Interface Card

12.7.1 Purpose
The purpose of the network interface card is to provide wired data streaming to the client pc over a LAN network.

12.7.2 Specifications
10Base-T, Automatic retransmits on collisions. Operating bounds of 3.0V to 3.6V, -40ºC to +85ºC, +/-2 ºC.

12.7.3 Inputs/Outputs
Input: SPI
Output: SPI

12.8 GPS Receiver

12.8.1 Purpose
Provide location tracking via the GPS coordinates system. The bump detection data will be location stamped by the gps data. This will provide bump location tracking after at a later time and date.

12.8.2 Specifications
20 channel receiver, Navigation Update Rate: 1 Hz, Input power range: 3.0 to 6.0 V, Cold start time: 36 s
Hot start Time: 1 s

12.8.3 Inputs/Outputs
Input: UART
Output: UART
13 Traceability Matrices

Traceability Matrices follow on the next page. It shows the module level of requirement mapping. The purpose of this module level requirement mapping is to give an overview of the requirements specified in System Requirement Specification that are intended to be satisfied based on the Detailed Design Specification’s modules.

The work division specified by Table below accurately depicts the modules required for each specific requirement and only the necessary modules used to achieve/meet the goal of the requirement. By specifying the required modules for each requirement we can use this as a means to determine if any requirements are specified to use unnecessary module to achieve/meet the requirement goal. The distribution of work among modules keeps the complexity within scope for the project.
## Requirements

### Power
- Power on/off easily
- Not operate after a critical error
- Visually indicate error conditions

### Indicate
- Indicate when it is operating normally
- Generate operational data from the raw data of the sensors
- Transfer operational data to the client application in real-time

### Measure
- Measure the travel distance of the platform
- Measure the internal temperature of the electronics housing
- Store operational data

### Display
- Display operational data
- Read stored data

### Configuration
- The client application can configure the PIC microcontroller
- The configuration process is abstracted from the end user
- Able to operate in a stand-alone mode

### User-friendly
- User-friendly client application
- Visually indicate a bump using a beacon light
- Use GPS to track the bump location

### GUI
- Unpack
- Pack
- Net Comm.
- Data Manager
- Net CMD
- File Storage
- Ex. Storage
- Visual
- Input

### Output
- Unpack
- Pack
- Data Divider
- Fan direction
- Error Lights
- Bit Banger
<table>
<thead>
<tr>
<th>NIC</th>
<th>Process</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unpack</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Pack</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Bump Detection</td>
<td>● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Storage</td>
<td>● ● ● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Unpack</td>
<td>●</td>
<td>● ● ● ●</td>
</tr>
<tr>
<td>Pack</td>
<td>●</td>
<td>● ● ● ●</td>
</tr>
<tr>
<td>Sessions Handler</td>
<td>● ●</td>
<td>● ● ●</td>
</tr>
<tr>
<td>Initializer</td>
<td>●</td>
<td>● ● ● ●</td>
</tr>
<tr>
<td>Physical Storage</td>
<td>● ●</td>
<td>● ● ●</td>
</tr>
</tbody>
</table>

Table 25 - Traceability Matrix
14 Quality Assurance

14.1 Test Plans and Procedures

Each layer will first be tested independently by performing the unit testing. Once it passes the unit testing, each component will be tested. Then, it will be tested with other layers and subsystems upon integration which will be done through integration testing. Integration testing will ensure that each layer interacts with each other as expected. Finally, the system verification testing will be conducted to ensure all the customer requirements are met and the system is robust.

Other ways of testing the quality and completeness of the design are requirement mapping and test cases. The DDS was tested for design completeness by static testing against the requirements. As shown in requirement mapping, each requirement is matched against a set of design components. Rows that have no corresponding design components were requirements that do not have any consideration in the design. Some of the test cases will be provided in this chapter. However, complete and thorough test cases will only be provided only in System Test Plan.

14.1.1 Module/Unit Test

14.1.2 Component Testing

Sled Testing

We all perform some lab testing with sandbox. If we don’t have any trail marks on sand, we will assume that the sled will be light enough to float. However, this may not be guaranteed. Actual testing on the road construction has to be performed to know whether the platform floats on top of freshly poured concrete or not.

Electronic Housing Testing

We will visually inspect and make sure that no wires are hanging outside and also make sure that the housing is light and water proof.

Sensor Testing
1. Temperature Check- The sensor will be calibrated; then we will vary the surrounding temperature with an external thermometer to verify that the sensor is functioning properly by a voltage variation tests.
2. Angle Check- The sensor will be calibrated then using a flat tiltable surface, we will tilt the sensor to known angles and confirm the given response via voltage variation tests.
3. Distance Check- The sensor has a turning gear. We will turn the gear to a given distance and verify the response signal given by the sensor via voltage variation.

**Display Testing**

When the device is turned on, the normal heart beat light should be turned on. When we give some bump events, then beacon light for bump detection should be turned on. When we pass some bad value, then the error light should be turned on. If all the expected output takes place, then we can verify that the display component is working properly.

**Communication Testing**

We will mount the wireless interface and access web server externally.

**Processor Testing**

Power on and connect.

**Storage Testing**

Store a file and verify its existence.

**Cooling Testing**

We will apply the correct power voltages and check to verify the operation of the fan.

**Power Distribution Testing**

We will connect an outside equivalent power source and verify output by multimeter.

**Client PC Testing**

Not to be tested.

1. Integration Testing

**Input Layer Testing**

When running on normal condition, the input layer should collect data and convert them to digital data, if necessary. And then, send those data to Storage and Processing Layer.

**Processing Layer Testing**

The processing layer should perform the bump detection algorithm on those data and send those processed data to Storage Layer and Output Layer.
Storage Layer Testing
The storage layer should store both raw and processed data. Also, this layer should be able to handle data retrieval requests.

Output Layer Testing
The output layer should be displaying proper events on the hardware display. Also, this layer is responsible for communicating and transferring raw and processed data to the GUI.

GUI Layer Testing
The GUI layer should be receiving raw and processed data from the output layer. And this layer should display those data digitally or graphically. The GUI layer should also handle user requests. Also, this should save raw and processed data in file storage.

14.1.4 System Verification Testing
The entire system will be tested as whole to ensure that it satisfies all high level requirements and all requirements listed in the Acceptance Criteria of the SRS. It will be tested to ensure that all interaction between layers is correct, and all playback functions are present and work correctly.

14.2 Test Cases
15 Acceptance Plan

15.1 Overview
This section discusses the acceptance criteria that must be met by the Sliding Profiler system to be considered minimally complete. These criterions are critical and must be fulfilled in order for the end product to be accepted by the customers and the stakeholders.

15.2 Packaging and Installation
The Sliding Profiler shall arrive completely assembled with a Plug-n-Play mentality. All the electronics components shall be enclosed with the electronic housing and shall be mounted to the platform. Also, the platform shall be designed in a way that it can be attached to the back of road paver. The software for GUI application shall be customer installable. Also, the software can be made available from any media source.

15.3 Acceptance Testing
The Sliding Profiler system acceptance testing shall be priority to ensure that product meets all the requirements to be accepted. The acceptance criteria for the product shall go by a critical testing process from the unit testing to the system testing which will be run successfully over the sophisticated test cases. The details of this testing will be provided in the System Test Plan document.

15.4 Acceptance Criteria
Acceptance criteria are the requirements that must be completed for the project to be accepted as complete. These requirements include the top priorities of the project which are derived from the critical requirements and would affect the functionality of the product if not taken into consideration.

The Sliding Profiler System must meet the following requirements agreed upon by all the stakeholders involved in the project:

- The device shall not operate after a critical error
- The device shall be able to measure the travel distance of the platform
- The device shall be able to measure the inclination of the platform from a calibration point
- The device shall generate operational data from the raw data of the sensors
- The device shall detect bumps in concrete pavement while the product is floating on the freshly poured concrete surface
- The device shall be able to operate in extreme temperatures
- The platform shall attach to the road paver
- The client application shall calibrate sensors
- The client application shall configure the PIC microcontroller
- The configuration process shall be abstracted from the end-user
- The product shall be able to operate in a stand-alone mode
- The product shall visually indicate a bump using a beacon light
- The product shall be light enough to float on top of the concrete pavement
- The device power source shall be provided externally
- The slider shall not accumulate cement
- The product shall be easy to clean