

1. REPORT NUMBER CA20-3182	3. GOVERNMENT ASSOCIATION NUMBER N/A	3. RECIPIENT'S CATALOG NUMBER N/A
4. TITLE AND SUBTITLE Hand-Held Diagnostic Controller for ITS Field Maintenance Phase II		5. REPORT DATE October 11, 2019
		6. PERFORMING ORGANIZATION CODE AHMCT Research Center
7. AUTHOR Stephen M. Donecker, Travis B. Swanston, Kin S. Yen, & Ty A. Lasky		8. PERFORMING ORGANIZATION REPORT NO. UCD-ARR-18-08-31-02
9. PERFORMING ORGANIZATION NAME AND ADDRESS AHMCT Research Center UCD Dept. of Mechanical & Aerospace Engineering Davis, California 95616-5294		10. WORK UNIT NUMBER N/A
		11. CONTRACT OR GRANT NUMBER IA65A0560 Task 3182
13. SPONSORING AGENCY AND ADDRESS California Department of Transportation P.O. Box 942873, MS #83 Sacramento, CA 94273-0001		13. TYPE OF REPORT AND PERIOD COVERED Final Report April 2017 – Sept. 2019
		14. SPONSORING AGENCY CODE Caltrans
15. SUPPLEMENTARY NOTES N/A		
16. ABSTRACT The Caltrans field element network includes several hundred changeable message signs (CMS), generically referred to as dynamic message signs (DMS), and many more closed-circuit TV (CCTV) cameras. The number of these elements motivates development of diagnostic tools that are handheld, quick, inexpensive, and straightforward for maintenance staff. The AHMCT Research Center worked with the Caltrans Division of Research, Innovation and System Information (DRISI) to develop a solution to this need. The solution is composed of off-the-shelf smartphones and tablets, wireless interfaces, and two apps for field element communications, diagnostics, and control. One app addresses DMS field elements, while the other addresses CCTV field elements. This report presents the overall Handheld Diagnostic Controller (HHDC) concept, design, and implementation details, including architecture, protocols, and configurations for a few representative districts. The details include discussion of DMS and CCTV app software and use cases, as well as the ruggedized HHDC hardware kit. The report also presents the evaluation and selection of a console app for inclusion with the HHDC in order to support diagnostics and control of other field elements.		
17. KEY WORDS Traffic operations, Highway maintenance, Intelligent Transportation Systems (ITS)	18. DISTRIBUTION STATEMENT No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.	
19. SECURITY CLASSIFICATION (of this report) Unclassified	20. NUMBER OF PAGES 230	21. COST OF REPORT CHARGED N/A

Reproduction of completed page authorized

DISCLAIMER

The research reported herein was performed by the Advanced Highway Maintenance and Construction Technology (AHMCT) Research Center, within the Department of Mechanical and Aerospace Engineering at the University of California – Davis, for the Division of Research, Innovation and System Information (DRISI) at the California Department of Transportation. AHMCT and DRISI work collaboratively to complete valuable research for the California Department of Transportation.

This document is disseminated in the interest of information exchange. The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California or the Federal Highway Administration. This publication does not constitute a standard, specification or regulation. This report does not constitute an endorsement by the Department of any product described herein.

The contents of this report do not necessarily reflect the official views or policies of the University of California. This report does not constitute an endorsement by the University of California of any product described herein.

For individuals with sensory disabilities, this document is available in alternate formats. For information, call (916) 654-8899, TTY 711, or write to California Department of Transportation, Division of Research, Innovation and System Information, MS-83, P.O. Box 942873, Sacramento, CA 94273-0001.



Advanced Highway Maintenance and Construction Technology Research Center

Department of Mechanical and Aerospace Engineering
University of California at Davis

Hand-Held Diagnostic Controller for ITS Field Maintenance Phase II

Stephen M. Donecker, Travis B. Swanston, Kin S. Yen, &
Ty A. Lasky: Principal Investigator

Report Number: CA20-3182

AHMCT Research Report: UCD-ARR-18-08-31-02

Final Report of Contract: IA65A0560 Task 3182

October 11, 2019

California Department of Transportation

Division of Research, Innovation and System Information

Executive Summary

The California Department of Transportation (Caltrans) has a wide variety of traffic operations field element devices that must be managed and maintained. This includes closed-circuit TV cameras (CCTV), changeable message signs (CMS), vehicle detection systems (VDS), road weather information systems (RWIS), etc. To maintain and configure these devices, personnel currently use laptops, which presents a large end user burden. Caltrans Division of Maintenance and Division of Research and Innovation have identified the need for a ruggedized handheld diagnostic controller (HHDC) to manage and maintain field element devices. Caltrans needs an HHDC kit including apps for CCTV and dynamic message signs (DMS, generic CMS term) diagnostics and control.

Research Objectives and Methodology

The primary purpose of this research was to complete the HHDC apps per the requirements listed in the appendices and support testing and updates for the HHDC software and hardware kit. The research included two primary software development iterations, each followed by detailed testing by Caltrans, along with one shorter final software iteration to address remaining issues. This research developed the HHDC software to maintain field element devices, including testing of this software on CMS and CCTV field elements. The HHDC uses ubiquitous commercial off-the-shelf (COTS) hardware. The HHDC is programmed to interface with CCTV and CMS hardware, using typical protocols for each, per the specifications in the appendices. The HHDC includes a COTS console application.

Results and Recommendations

Key contributions of this research project included:

- Development and testing of the HHDC CMS app
- Development and testing of the HHDC CCTV app
- Testing of the HHDC console app
- Development and testing of the HHDC hardware kit
- Enabling of HHDC for two key Caltrans ITS field elements
- Proof of the benefits of the HHDC approach for system diagnostics and control

The researchers recommend deployment of the current HHDC with developed app(s). In addition, the researchers recommend development of additional apps for ITS field elements including, but not limited to Vehicle Detection Systems (VDS), Road Weather Information Systems, microwave VDS, and ramp metering systems.

Table of Contents

Executive Summary	i
Research Objectives and Methodology	i
Results and Recommendations	i
Table of Contents	iii
List of Figures	vi
List of Tables	x
List of Acronyms and Abbreviations	xi
Acknowledgments	xiii
Chapter 1: Introduction	1
Problem	1
Objectives	1
Research Methodology	2
Overview of Research Results and Benefits	2
Chapter 2: Handheld Diagnostic Controller Concept	3
Motivating Example	5
Common HHDC Implementation Issues	6
HHDC Development Issue Tracker	8
HHDC Development Workflow	10
HHDC Software Architecture	10
Communications endpoints and parameters	15
Common Data Storage	16
General Testing Approach	17
Common Software Problems	18
Chapter 3: Handheld Diagnostic Controller Design DMS App	19
DMS App Overview	19
DMS App Design	20
DMS App Implementation	30
DMS Software Problems	37
SignView Issues	37
NTCIP Issues	37
DMS App Screen Shots	39
File Locations	98
Configurations	99
File Format	99
Sign configuration object parameters	101

Sign communication configuration object parameters _____	101
TCP endpoint object parameters _____	101
ComPort endpoint object parameters _____	102
Chapter 4: Handheld Diagnostic Controller Design CCTV App _____	103
CCTV App Overview _____	103
CCTV App Design _____	103
CCTV App Implementation _____	105
CCTV Software Problems _____	109
Camera/Encoder Video Protocol Issues _____	109
PTZ Protocol Issues _____	109
Drivers _____	109
Pelco-D _____	109
Cohu-I _____	110
ONVIF Profile S _____	110
CCTV App Screen Shots _____	111
File Locations _____	135
Configurations _____	135
File Format _____	135
Camera configuration object parameters _____	137
Camera communication configuration object parameters _____	137
TCP endpoint object parameters _____	138
ComPort endpoint object parameters _____	138
Chapter 5: HHDC System Kit Hardware _____	140
Equipment _____	141
Chapter 6: HHDC Console Software _____	151
Overview _____	151
Comparison _____	151
Final Selection _____	152
Chapter 7: Conclusions and Future Research _____	156
References _____	158
Appendix A: Glossary of Common Terms and Acronyms for Requirements ____	159
Appendix B: Handheld Diagnostic Controller Kit System Requirements _____	162
Appendix C: Handheld Diagnostic Controller DMS System Requirements ____	168
Appendix D: Handheld Diagnostic Controller CCTV System Requirements ____	174
Appendix E: Handheld Diagnostic Controller Console System Requirements _	181
Appendix F: HHDC Hardware Documentation _____	183
Appendix G: HHDC Hierarchical Listing of Source Code Files _____	185
Appendix H: HHDC Cable Assemblies _____	210

Appendix I: HHDC Kit Contents	212
-------------------------------	-----

List of Figures

Figure 2.1: DMS app conceptual overview	4
Figure 2.2: CCTV app conceptual overview	5
Figure 2.3: Mobile OS market share	7
Figure 2.4: Android version market share for May 7, 2018	8
Figure 2.5: Generic Trello-based issue tracker board	9
Figure 2.6: Creating an issue ticket	10
Figure 2.7: Marking an issue resolved	11
Figure 2.8: HHDC software development workflow	12
Figure 2.9: Architecture for the common Logger component	13
Figure 2.10: Architecture for the common DevComm (Device Communications) component	13
Figure 2.11: Architecture for the DMS-specific SignView DevComm plug-in	14
Figure 2.12: Architecture for the common NTCIP DevComm plug-in	14
Figure 2.13: Architecture for the common Airconsole DevComm plug-in	15
Figure 2.14: Model-View-ViewModel (MVVM) software pattern	15
Figure 3.1: DMS fragment graph	20
Figure 3.2: Create sign form mockup	21
Figure 3.3: Sign list mockup	22
Figure 3.4: Signs preview mockup	23
Figure 3.5: Sign overview mockup including edit and delete capability	24
Figure 3.6: Edit sign form mockup	25
Figure 3.7: Create communication form mockup	26
Figure 3.8: Communication overview mockup including edit and delete capability	27
Figure 3.9: Edit communication form mockup	28
Figure 3.10: TCP fragment mockup	29
Figure 3.11: ComPort fragment mockup	30
Figure 3.12: HHDC DMS software architecture	31
Figure 3.13 DMS configuration state architecture and transitions	32
Figure 3.14: District configuration DMS type A	34
Figure 3.15: District configuration DMS type B	35
Figure 3.16: District configuration DMS type C	35
Figure 3.17: Typical DMS use cases	36
Figure 3.18: DMS use case requiring field element network visibility	37
Figure 3.19: DMS startup screen	39
Figure 3.20: DMS main menu	40
Figure 3.21: DMS create sign configuration	41
Figure 3.22: DMS create sign configuration name field error message	42
Figure 3.23: DMS create sign configuration model options	43
Figure 3.24: DMS create sign creation location error message	44

Figure 3.25: DMS create sign communication with default values	45
Figure 3.26: DMS create sign communication name settings	46
Figure 3.27: DMS create sign communication port setting	47
Figure 3.28: DMS create sign communication data bits options	48
Figure 3.29: DMS create sign communication parity options	49
Figure 3.30: DMS create sign communication stop bits options	50
Figure 3.31: DMS create sign communication flow control options	51
Figure 3.32: DMS sign configuration with a serial communication configuration	52
Figure 3.33: DMS read-only sign communication option to edit	53
Figure 3.34: DMS read-only sign configuration with option to delete	54
Figure 3.35: DMS edit sign communication	55
Figure 3.36: DMS 170 sign configuration	56
Figure 3.37: DMS sign configuration with option to edit	57
Figure 3.38: DMS edit sign configuration	58
Figure 3.39: DMS edit sign configuration with option to delete	59
Figure 3.40: DMS 2070 create sign configuration	60
Figure 3.41: DMS create sign communication protocol options	61
Figure 3.42: DMS create sign communication endpoint type options	62
Figure 3.43: DMS 2070 sign communication configuration	63
Figure 3.44: DMS 2070 sign configuration with a communications configuration	64
Figure 3.45: DMS with 170 and 2070 sign configurations	65
Figure 3.46: Selection of the 170 sign configuration	66
Figure 3.47: DMS sign configured with the 170 sign and serial communications configurations	67
Figure 3.48: DMS sign in draft message mode	68
Figure 3.49: DMS sign draft mode font options	69
Figure 3.50: DMS sign draft mode displaying multi-font rendering	70
Figure 3.51: DMS sign sending draft configuration to the 170 controller	71
Figure 3.52: DMS sign displaying the content currently stored on the 170 controller	72
Figure 3.53: Selection of the 2070 sign configuration	73
Figure 3.54: DMS sign displaying the content currently stored on the 2070 controller	74
Figure 3.55: Sample DMS camera image in lab test	75
Figure 3.56: DMS camera image gallery	76
Figure 3.57: DMS camera gallery with image selected	77
Figure 3.58: DMS image annotation color options	78
Figure 3.59: DMS camera image annotation line width selection	79
Figure 3.60: Annotated DMS camera image	80
Figure 3.61: DMS camera image gallery including annotated and unmodified original images	81
Figure 3.62: DMS settings default configuration	82
Figure 3.63: DMS settings with show multi-string enabled	83
Figure 3.64: DMS sign with show multi-string enabled	84

Figure 3.65: DMS settings with show event log enabled	85
Figure 3.66: DMS settings event log level configuration	86
Figure 3.67: DMS sign with show event log enabled	87
Figure 3.68: DMS settings log level configuration	88
Figure 3.69: DMS sign following a detail request	89
Figure 3.70: DMS log showing the output following a detail request	90
Figure 3.71: DMS log begin new test annotation	91
Figure 3.72: DMS log displaying begin new test annotation	92
Figure 3.73: DMS sign draft new test message	93
Figure 3.74: DMS sign sending new test message to controller	94
Figure 3.75: DMS sign displaying the content currently stored on the controller	95
Figure 3.76: DMS log displaying results following the new test	96
Figure 3.77: DMS log end new test annotation	97
Figure 3.78: DMS log displaying the end new test annotation	98
Figure 4.1: CCTV fragment graph	104
Figure 4.2: District configuration CCTV type A	106
Figure 4.3: District configuration CCTV type B	106
Figure 4.4: District configuration CCTV type C	107
Figure 4.5: Typical CCTV use cases	108
Figure 4.6: CCTV use case requiring field element network visibility	108
Figure 4.7: CCTV startup screen	111
Figure 4.8: CCTV main menu	112
Figure 4.9: CCTV create camera configuration	113
Figure 4.10: CCTV create sign configuration with example values	114
Figure 4.11: CCTV create sign communication with default values	115
Figure 4.12: CCTV create camera communication protocol options	116
Figure 4.13: CCTV create camera communication endpoint type options	117
Figure 4.14: CCTV create camera communication configuration with example values	118
Figure 4.15: CCTV camera configuration with an Onvif over Tcp communication configuration	119
Figure 4.16: CCTV list of configured cameras	120
Figure 4.17: CCTV create camera configuration with example values	121
Figure 4.18: CCTV create camera communication configuration with example values	122
Figure 4.19: CCTV camera configuration with a Cohu I over Tcp communication configuration	123
Figure 4.20: CCTV create a second camera communication configuration with example values	124
Figure 4.21: CCTV camera configuration with multiple communication configurations	125
Figure 4.22: CCTV list of configured cameras	126
Figure 4.23: CCTV camera displaying streaming content	127
Figure 4.24: CCTV camera with PTZ controls enabled	128

Figure 4.25: CCTV camera with presets enabled	129
Figure 4.26: CCTV camera showing zoom out state	130
Figure 4.27: CCTV camera showing zoom in state	131
Figure 4.28: CCTV camera showing pan left state	132
Figure 4.29: CCTV camera showing pan left tilt down state	133
Figure 4.30: CCTV camera showing pan right tilt down state	134
Figure 5.1: General connectivity and role of the HHDC kit components	140
Figure 5.2: HHDC kit in carrying case	141
Figure 5.3: HHDC kit with tablet and monitor removed	142
Figure 5.4: HHDC kit full contents	143
Figure 5.5: HHDC kit tablet	144
Figure 5.6: HHDC kit Airconsole	145
Figure 5.7: HHDC Airconsole connected to Ethernet	146
Figure 5.8: HHDC Airconsole connected to USB serial	147
Figure 5.9: HHDC kit USB serial RS-232 to 170-C2 connector cable	148
Figure 5.10: HHDC kit USB serial RS-422 to D2 CCTV RJ45 connector	149
Figure 5.11: HHDC kit USB serial RS-422 to D6 CCTV DE-9 connector	150
Figure 6.1: Console app Command-Line View (courtesy of Sonelli Ltd.)	153
Figure 6.2: Console app Keyboard View (courtesy of Sonelli Ltd.)	153
Figure 6.3: Console app Connection View (courtesy of Sonelli Ltd.)	154
Figure 6.4: Console app Configuration View (courtesy of Sonelli Ltd.)	155

List of Tables

Table 2.1: HHDC communications endpoints	16
Table 2.2: Raw TCP endpoint parameters	16
Table 2.3: ComPort endpoint parameters	17
Table 3.1: Supported CMS and AVMS signs	33
Table 3.2: Supported DMS controllers	33
Table 3.3: Supported sign protocols	34
Table 4.1: Supported CCTV video protocols	105
Table 4.2: Supported CCTV PTZ protocols	105
Table 5.1: HHDC kit detailed hardware list	140
Table 6.1: Console app feature matrix	152
Table F.1: HHDC component images	183
Table H.1: Airconsole cable assembly pinout	210
Table H.2: USB RS-232 to 170 C2 cable assembly pinout	210
Table H.3: USB RS-422 to D2 CCTV cable assembly pinout	211
Table H.4: USB RS-422 to D6 CCTV cable assembly pinout	211
Table I.1: HHDC kit contents	212

List of Acronyms and Abbreviations

Acronym	Definition
AHMCT	Advanced Highway Maintenance and Construction Technology Research Center
API	Application Programming Interface
ATMS	Advanced Transportation Management System
AVMS	Advanced Variable Message Sign
BNC	Bayonet Neill–Councilman
Caltrans	California Department of Transportation
CCTV	Closed-Circuit TV
CLI	Command-Line Interface
CMS	Changeable Message Sign
COTS	Commercial Off-The-Shelf
DMS	Dynamic Message Sign
DOT	Department of Transportation
DRISI	Caltrans Division of Research, Innovation and System Information
HHDC	Handheld Diagnostic Controller
HTML	Hypertext Markup Language
IRIS	Intelligent Roadway Information System
IT	Information Technology
ITS	Intelligent Transportation Systems
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
MAC	Media Access Control
MJPEG	Motion JPEG

Acronym	Definition
MPEG	Moving Picture Experts Group
MVVM	Model-View-ViewModel
NTCIP	National Transportation Communications for ITS (Intelligent Transportation Systems) Protocol
NTSC	National Television System Committee
OS	Operating System
PM	Project Manager
PTZ	Pan-Tilt-Zoom
RFC	Request for Comments
RJ	Registered Jack
RS	Recommended Standard
RTS	Request to Send
RWIS	Road Weather Information System
Rx	Receive
SOCCS	Satellite Operations Center Command System
SSH	Secure Shell
TAG	Technical Advisory Group
TCP	Transmission Control Protocol
TMC	Transportation Management Center
Tx	Transmit
USB	Universal Serial Bus
VDS	Vehicle Detection Systems

Acknowledgments

The authors thank the California Department of Transportation (Caltrans) for their support, in particular Jeremiah Pearce, Chief, Office of ITS Engineering and Support, and Sean Campbell, Senior Engineer with the Division of Research, Innovation and System Information. The authors acknowledge the dedicated efforts of the AHMCT team who have made this work possible.

Chapter 1:

Introduction

Problem

The California Department of Transportation (Caltrans) has a wide variety of traffic operations field element devices that must be managed and maintained. This includes closed-circuit TV cameras (CCTV), changeable message signs (CMS), vehicle detection systems (VDS), road weather information systems (RWIS), etc. To maintain and configure these devices, personnel currently use laptops, which presents a large end user burden. Caltrans Division of Maintenance and Division of Research, Innovation and System Information (DRISI) have identified the need for a ruggedized handheld diagnostic controller (HHDC) to manage and maintain field element devices.

Under a previous research project, the Advanced Highway Maintenance and Construction Technology (AHMCT) Research Center worked with Caltrans to specify and develop a HHDC to manage and maintain field element devices [1]. This research developed the specifications for the hardware kit, and the CCTV, CMS, and console apps for Android tablets. A glossary of common HHDC-related terms and the HHDC specifications are included in the current report's Appendices A-E.

AHMCT worked on the previous prototype for this HHDC system. The hardware kit, including multiple instances, was completely developed. In addition, a commercial off-the-shelf (COTS) console app was identified and tested; this app meets Caltrans needs and the developed requirements. The custom software for the CMS and CCTV apps was initiated, and the CCTV prototype app was far along at the conclusion of the prior research. However, neither the CMS nor the CCTV apps were fully implemented and tested in that research, due in part to higher than anticipated complexity.

Caltrans' need for these two apps remains, and motivated the current research. Along with completing the development of the apps, it is essential that Caltrans is able to test the hardware kits, the two apps, the commercial console app, and provide feedback so that AHMCT can finalize the HHDC. This work must be performed in order for Caltrans to realize the benefits of the HHDC.

Objectives

The primary purpose of this research was to complete the HHDC apps per the requirements listed in the appendices and support testing and updates for the HHDC software and hardware kit. The research included two primary software

development iterations, each followed by detailed testing by Caltrans, along with one shorter final software iteration to address remaining issues. This research developed the HHDC software to maintain field element devices, including testing of this software on CMS and CCTV field elements. The HHDC uses ubiquitous COTS hardware. The HHDC is programmed to interface with CCTV and CMS hardware, using typical protocols for each, per the specifications in the appendices. The HHDC includes a COTS console application.

Research Methodology

A technical advisory group (TAG) was established early in the project, and regular meetings were held with the PM and/or the TAG. AHMCT and the Project Manager (PM) worked collaboratively during the research effort to best guide the research effort.

The work completed the development of the HHDC, supported testing of HHDC use in Caltrans operations, and revised the HHDC based on Caltrans feedback. The goal was to bring the HHDC to the point where it meets the requirements presented in the appendices, and it can be widely used by Caltrans and other agencies.

To achieve the project objectives, the basic research task structure consisted of iterative HHDC development followed by support for HHDC alpha testing, followed by feedback to the researchers, and another iteration cycle. The efforts, results, and system were documented throughout the research project, culminating in this final report.

Overview of Research Results and Benefits

The HHDC will give Caltrans field personnel a ruggedized, robust system to deal with the specific task of maintaining and configuring traffic operations field elements. This will enable employees to get their work done faster, safer, and with dramatically reduced chances for loss of potentially expensive equipment.

The key deliverables of this project include:

- HHDC development kits
- HHDC CMS diagnostic controller app
- HHDC CCTV diagnostic controller app
- HHDC CMS diagnostic controller app source code
- HHDC CCTV diagnostic controller app source code
- HHDC documentation
- Documentation of any written feedback received from alpha testing
- Documentation of any written feedback received from beta testing

Chapter 2:

Handheld Diagnostic Controller Concept

Caltrans maintenance staff must routinely configure, and diagnose field element hardware in order to keep the transportation system operating at peak efficiency. They currently use multiple tools and expensive cumbersome equipment to maintain each field element type. The vision of this research is to use one hardware platform populated with custom apps to configure and diagnose a wide range of field element devices. The diagnostic and configuration system is based on readily-available commercial off-the-shelf hardware, specifically Android smartphones and tablets, meeting the cost and acquisition needs of the Department of Transportation (DOT) and portability and ease of use demands of maintenance and traffic operations personnel.

AHMCT has developed a suite of applications for the handheld diagnostic controller configuration and diagnostics tool, as well as any needed supplemental hardware, e.g. cabling, to interface to the field element devices. The system is integrated into a kit that is well-suited to carry in a field maintenance vehicle, and allows quick and intuitive configuration and diagnostics of field element devices. The core of the system is an Android smartphone or tablet. This could be the field maintenance personnel's regular smartphone, or it could be part of the kit, depending on DOT operational preferences.

The HHDC is based on two primary feature-rich intuitive apps. Both applications are able to access their relevant field elements remotely as well as locally. The first app is for Dynamic Message Sign (DMS) configuration and diagnostics. Here, DMS is used generically to imply CMS or Advanced Variable Message Sign (AVMS), both of which Caltrans currently uses. It interfaces to DMS controllers in the field through both network and serial interfaces, and allows for full configuration and diagnosis of the DMS system. The app fully implements all features of the DMS SignView protocol, including all deployed versions, supporting accelerated diagnostic operations of the panel, controller, and firmware.

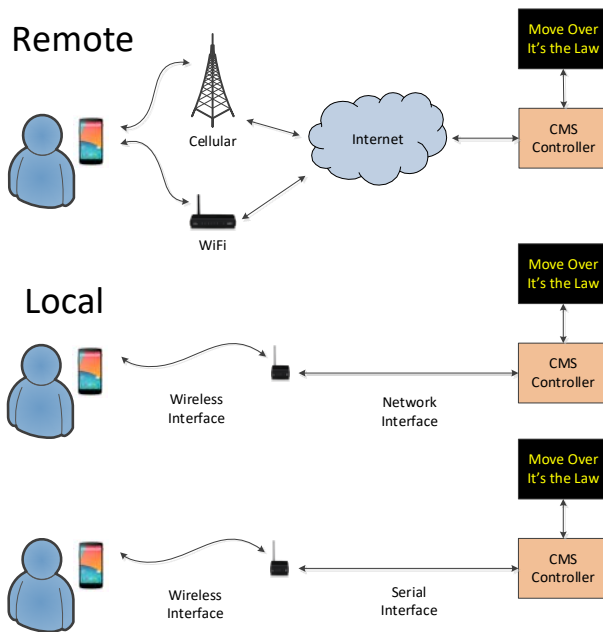


Figure 2.1: DMS app conceptual overview

The second app is for CCTV configuration and diagnostics. It interfaces to CCTV cameras, pan-tilt-zoom (PTZ) hardware, and video encoders in the field through both network and serial interfaces, and provides all aspects of CCTV system configuration and diagnostics. The application currently supports camera/encoder Moving Picture Experts Group (MPEG) MPEG4,¹ and H.264² video protocols, and Pelco-D,³ Cohu-I,⁴ and ONVIF Profile S⁵ PTZ control protocols. Besides having the ability to fully configure all aspects of the CCTV system, it provides advanced features to diagnose CCTV hardware and firmware from typical operational functionality all the way down to verification of key PTZ control protocol command and response.

AHMCT also evaluated COTS terminal console apps for inclusion in the HHDC software suite. We assessed the available features vs. anticipated DOT needs. In conjunction with Caltrans, we selected the most appropriate terminal console app, and included it in the suite of apps for the handheld diagnostic controller.

¹ [MPEG4 \(https://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm\)](https://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm)

² [H.264 \(https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC#External_links\)](https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC#External_links)

³ The author of the Pelco Protocol D document is [Eric Hamilton \(ehamilton@pelco.com\)](mailto:ehamilton@pelco.com), who can be contacted for protocol document requests and questions.

⁴ [CohuHD Protocols \(https://www.cohuhd.com/Support/Software-Downloads\)](https://www.cohuhd.com/Support/Software-Downloads)

This page requires an account to download relevant document(s).

⁵ [ONVIF Profile S \(https://www.onvif.org/profiles/profile-s/\)](https://www.onvif.org/profiles/profile-s/)

The terminal console provides a general command-line interface for interacting with a wide range of field element hardware.

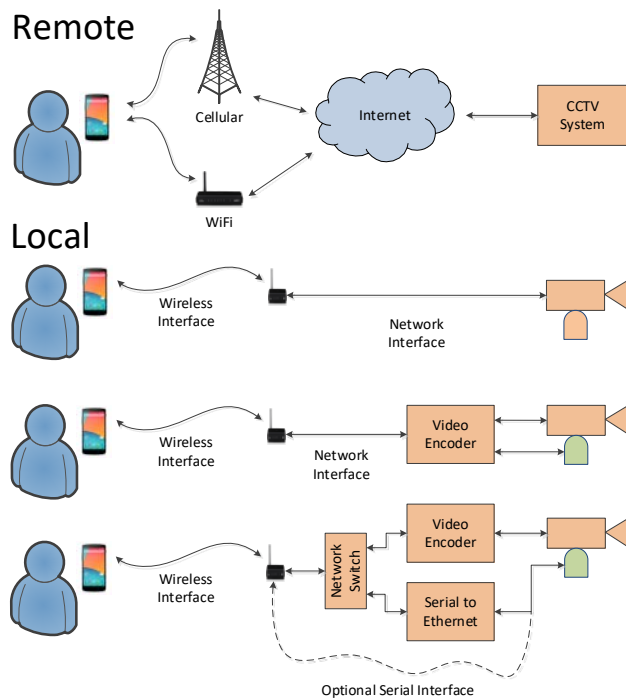


Figure 2.2: CCTV app conceptual overview

The HHDC kit provides a compact, unified system based on commodity smartphones and tablets. The software apps provide device-specific configuration and diagnostic capabilities, as well as general-purpose command-line interaction. This tool will greatly enhance the capabilities of DOT field maintenance personnel.

The availability of this system can eliminate the need for custom, costly, and cumbersome hardware typically used for current field element device setup, diagnostics, and maintenance. This can reduce costs for the DOT in both equipment and labor, through simplified and increased portability of the hardware that field maintenance personnel carry, enhanced and reduced setup and diagnostic efforts, and reduced trips and time at field element locations. This would lead to not only reduced equipment and personnel costs, but improved field element up-time, enhanced system operations, and reduction in fuel use and greenhouse emissions. The detailed HHDC hardware and software design is presented in the remainder of this report.

Motivating Example

Consider the case for a DMS, where the Transportation Management Center (TMC) cannot connect to the sign. For this scenario, and similarly for a non-responding CCTV camera, what do you do?

Typically, a TMC will begin by attempting to verify loss of connectivity using alternative TMC tools, such as the Advanced Transportation Management System (ATMS), the Intelligent Roadway Information System (IRIS), the Satellite Operations Center Command System (SOCCS), etc. With these and similar tools, the TMC can verify the status of communications network. For example, can they contact nearby field elements of the same or different type (DMS, camera, etc.)? Alternatively, can they connect to on-site networking equipment or embedded computers?

Depending on the outcome of these diagnostic tests, the TMC has two primary options: call Information Technology (IT), or send field diagnostics personnel to the sign. The HHDC is a useful tool for the second option.

In the pre-HHDC situation, the TMC would often try to determine whether there is someone nearby already in the field, and whether they have what they need in their vehicle. The pre-HHDC diagnostic toolkit included laptops, batteries, power supplies, and a somewhat *ad hoc* set of non-specific software, e.g. terminal emulation software. The software and the OS were typically locked down (i.e. difficult to configure on-site), and hardware was not organized into a kit. The miscellaneous hardware often included bulky cables and scattered adapters.

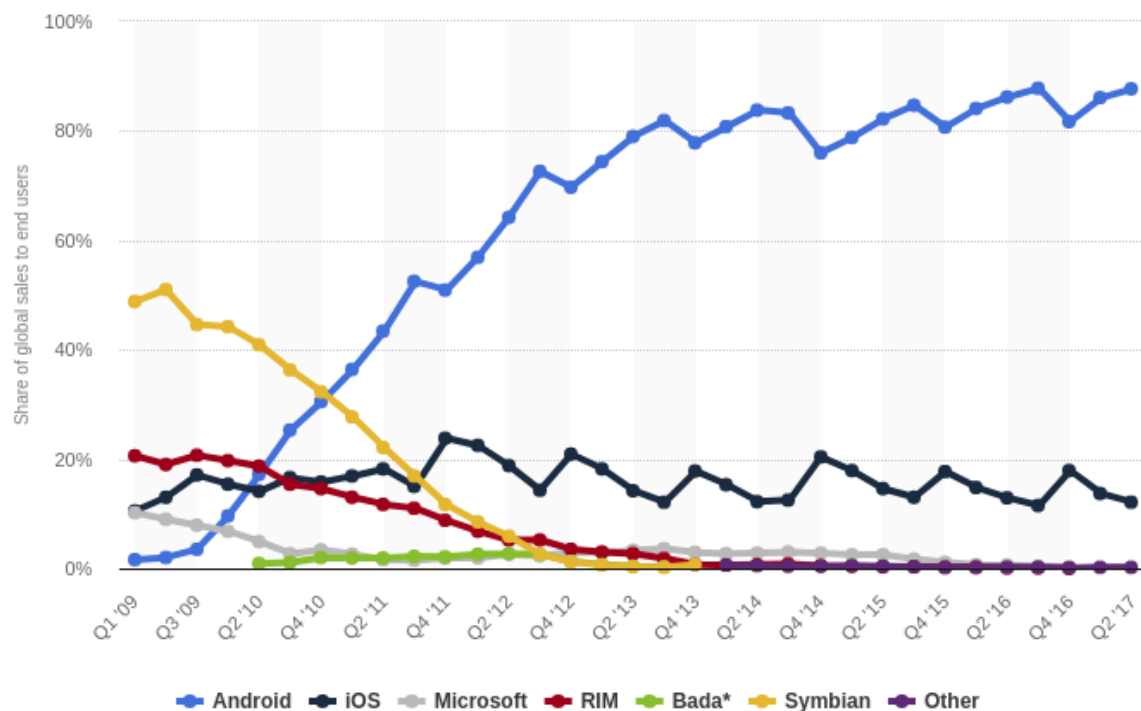
The HHDC concept introduces a well-organized kit packaged in a ruggedized case, coupled with smartphone or tablet apps specifically designed to support diagnosis for key field element hardware. This currently includes DMS and CCTV cameras. The kit includes all needed cabling and adapters, as well as a tablet and necessary power supplies and chargers. The concept could also be extremely useful in reduced form, wherein a technician has key adapters, a smartphone, and the apps, rather than a complete kit. Through availability of the full or reduced HHDC, Caltrans would be able to more quickly and accurately diagnose and repair its field elements, leading to more effective traffic management.

Common HHDC Implementation Issues

Detailed design and implementation of the specific apps will be addressed in subsequent chapters. Here, common implementation issues are presented.

Once the decision was made to implement the HHDC using smartphone and/or tablet hardware, the key decision was operating system (OS). The primary contender, based on market share, was Android, with Apple iOS second and Microsoft Windows a distant third. Recent market share is shown in Figure 2.3. While market share alone is not a sufficient criteria, numerous other criteria also pointed to the use of Android. The Android Application Programming Interface (API) is well-documented and powerful. It also supports distribution and updating through effective online means. For these and

additional reasons, Android was selected as the development platform for the HHDC.



© Statista 2018

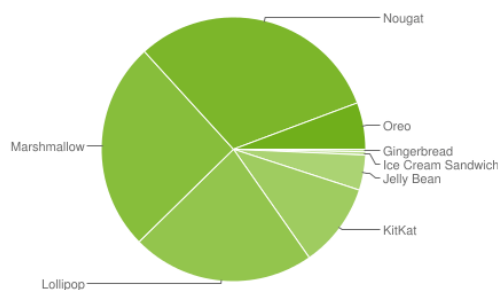
Additional Information
Worldwide; Gartner

Source
Gartner

Figure 2.3: Mobile OS market share

Once the decision to base the HHDC on Android was made, it was then necessary to decide which version(s) of Android to support. Maintaining legacy support for older versions is useful, but incurs added cost in terms of development effort, code complexity, and potential loss of advanced features. To make this decision, the researchers looked at the version market share within Android. Figure 2.4 provides a snapshot of this market share as of May 7, 2018. From this data as well as feature and development considerations, the researchers decided to support Android version 6.0 (Marshmallow) and higher. At the time this data was collected, this represented approximately 62% of the Android market share. This share would only have increased since then with the release of Android 9.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%



Data collected during a 7-day period ending on May 7, 2018.
Any versions with less than 0.1% distribution are not shown.

Figure 2.4: Android version market share for May 7, 2018

HHDC Development Issue Tracker

Throughout the HHDC development, an online issue tracker was used to keep track of bugs, enhancements, feature requests, and other issues. The tracker is based on the online web / app service Trello,⁶ which supports boards, lists, and cards to track a variety of activities and information. A generic Trello issue tracking board is shown in Figure 2.5. Here, there are several lists for creating and tracking the status of individual issues. The first list, called “Open,” is for newly created issues which have not yet had work started. The “In Progress” list is for issues which are currently being worked on by one or more developers. The “Resolved” list is record of issues which a developer has fixed. The “Closed” list is for issues that have then been confirmed as fixed by the original reporter.

⁶ [Trello \(https://trello.com\)](https://trello.com)

Finally, the “Reopened” list is for issues which were previously resolved and/or closed, but for some reason need to have further work performed. As issues are created, begin work, and are resolved, closed, or reopened, the ticket for the issue is moved to the appropriate list. This provides high visibility and clarity to reporters, developers, and managers for the state of the outstanding and completed software issues.

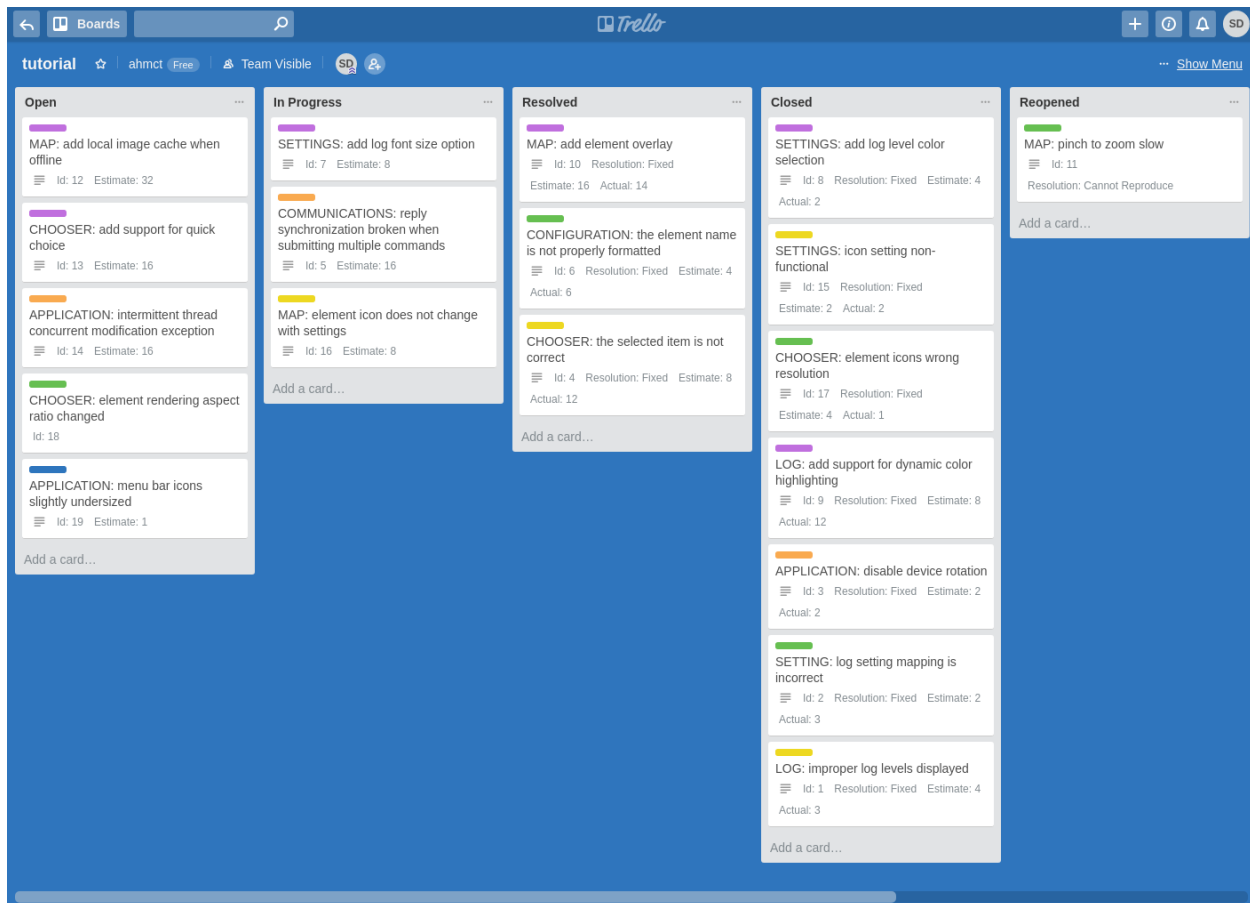


Figure 2.5: Generic Trello-based issue tracker board

Figure 2.6 shows the interface for creating a ticket, which would then appear in the “Open” list. Here, the reporter is creating an issue for the “LOG” component of the software. The specific incident title is “improper log levels displayed.” The reporter selects from a list of available labels to classify the issue. Here, the issue tag is “Bug – Major.” The ticket includes a field for the reporter to provide a detailed description of the issue. It also contains developer fields for design and notes related to the issue. For further description of the Trello ticket interface, see Trello’s site.⁷ Figure 2.7 shows the developer marking the ticket as

⁷ [Trello \(https://trello.com\)](https://trello.com)

“Fixed,” i.e. updating the issue resolution. At this point, the ticket is moved into the “Resolved” board.

HHDC Development Workflow

Figure 2.8 provides an overview of the HHDC software development workflow as implemented in this project. This includes the use of unit tests and logging to catch issues early in the development process. It also includes the use of the Git open source distributed version control system,⁸ which was used to track and manage revisions to the HHDC software repository.

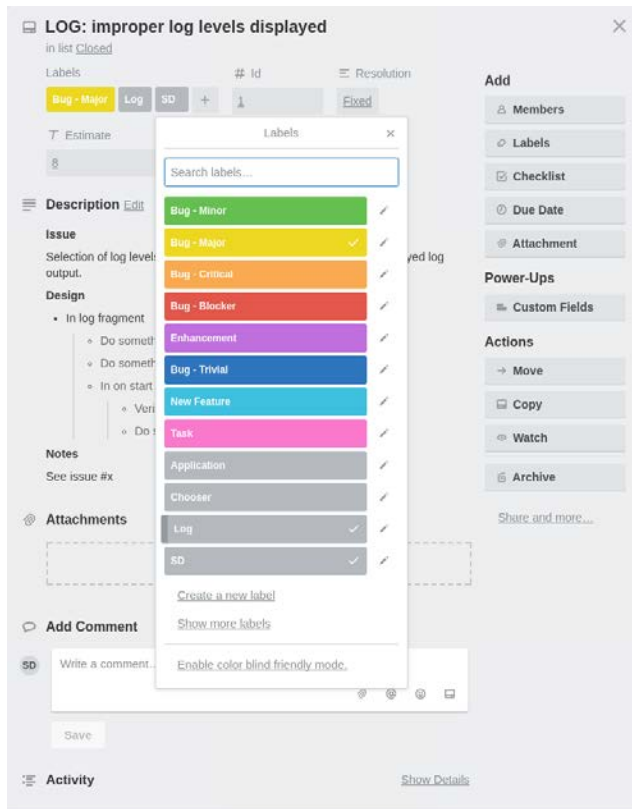


Figure 2.6: Creating an issue ticket

HHDC Software Architecture

The architecture for the common logger component is shown in Figure 2.9, while the common device communications architecture is shown in Figure 2.10. The DMS-specific SignView DevComm plug-in architecture is shown in Figure 2.11, while the architectures for the common DevComm NTCIP (National Transportation Communications for ITS (Intelligent Transportation Systems)

⁸ [Git \(https://git-scm.com/\)](https://git-scm.com/)

Protocol) and Airconsole plug-ins are shown in Figure 2.12 and Figure 2.13, respectively.

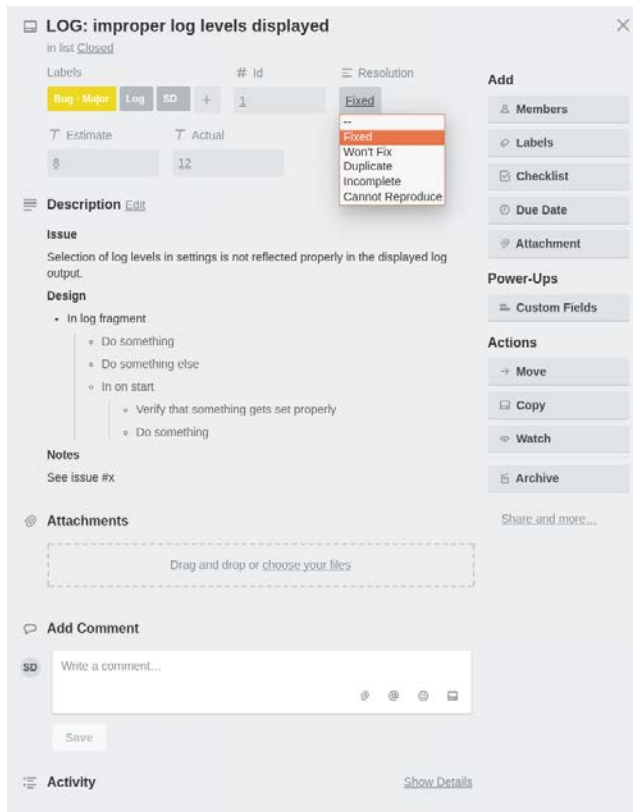


Figure 2.7: Marking an issue resolved

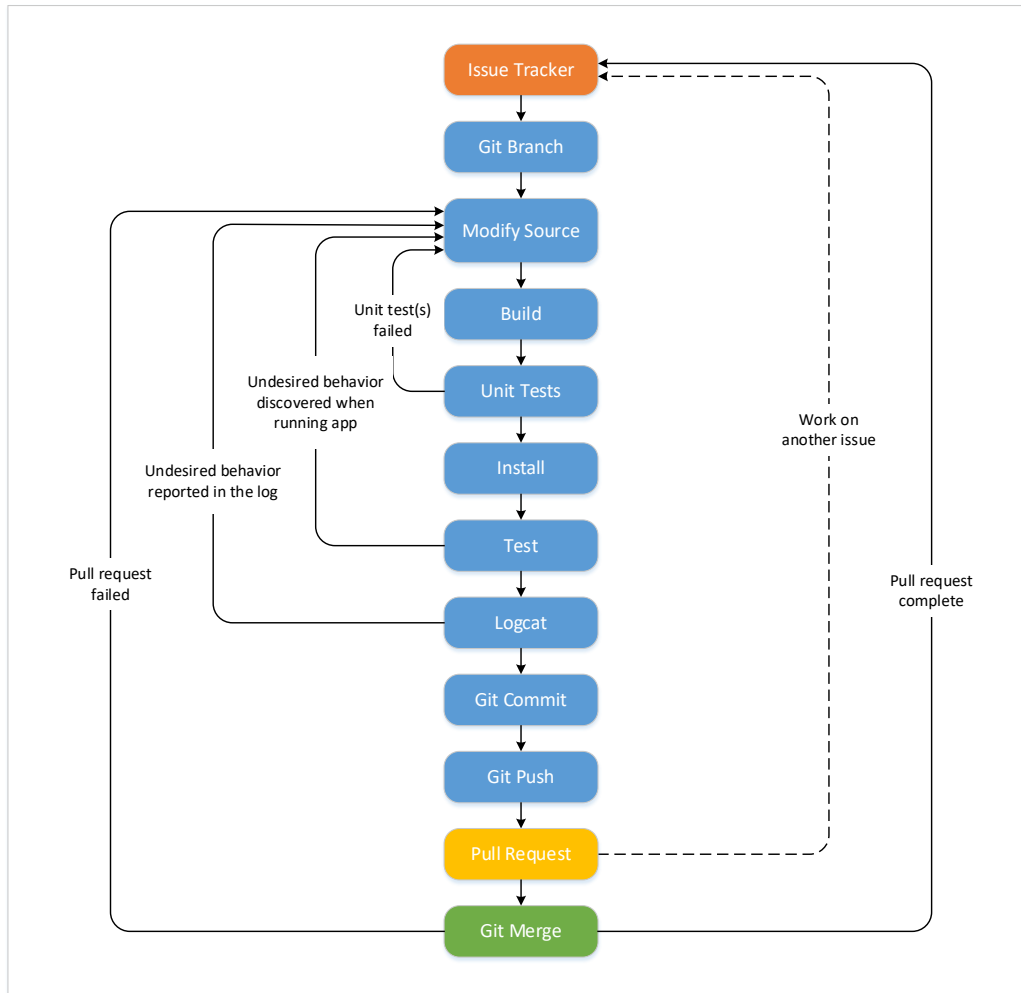


Figure 2.8: HHDC software development workflow

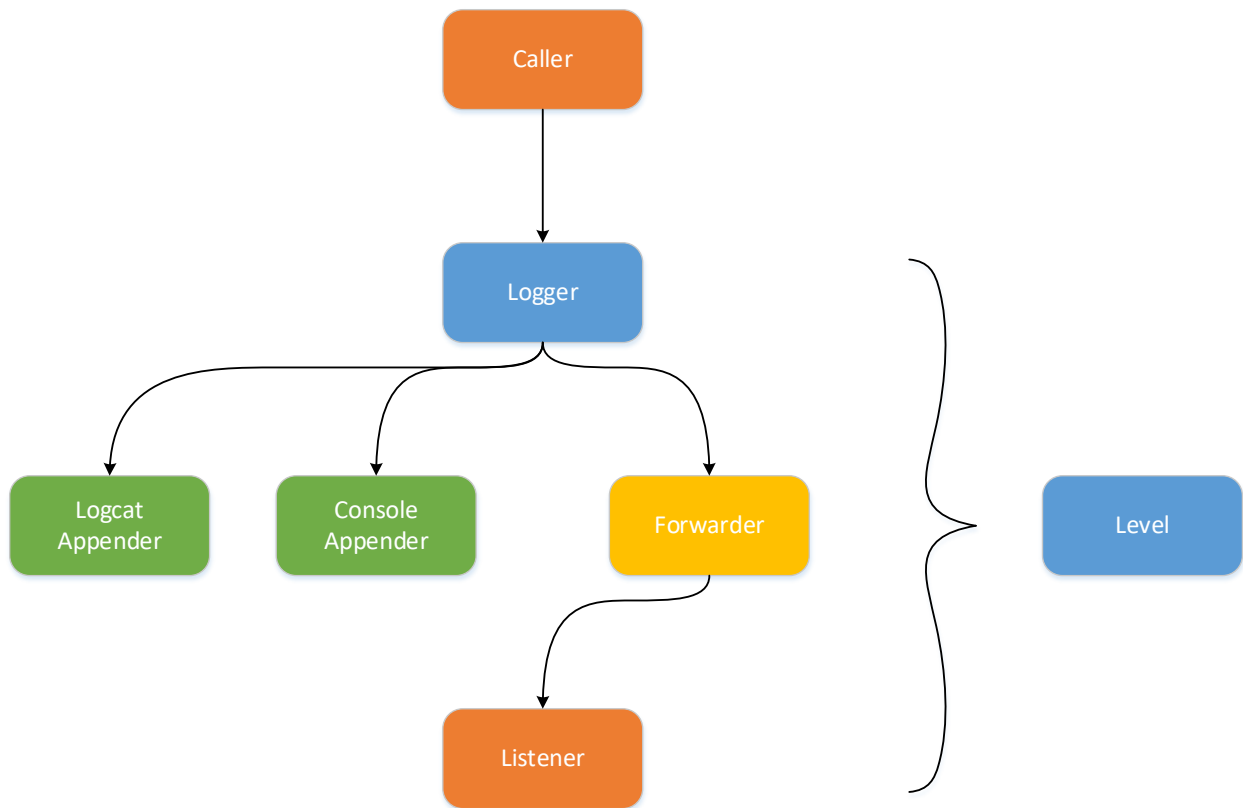


Figure 2.9: Architecture for the common Logger component

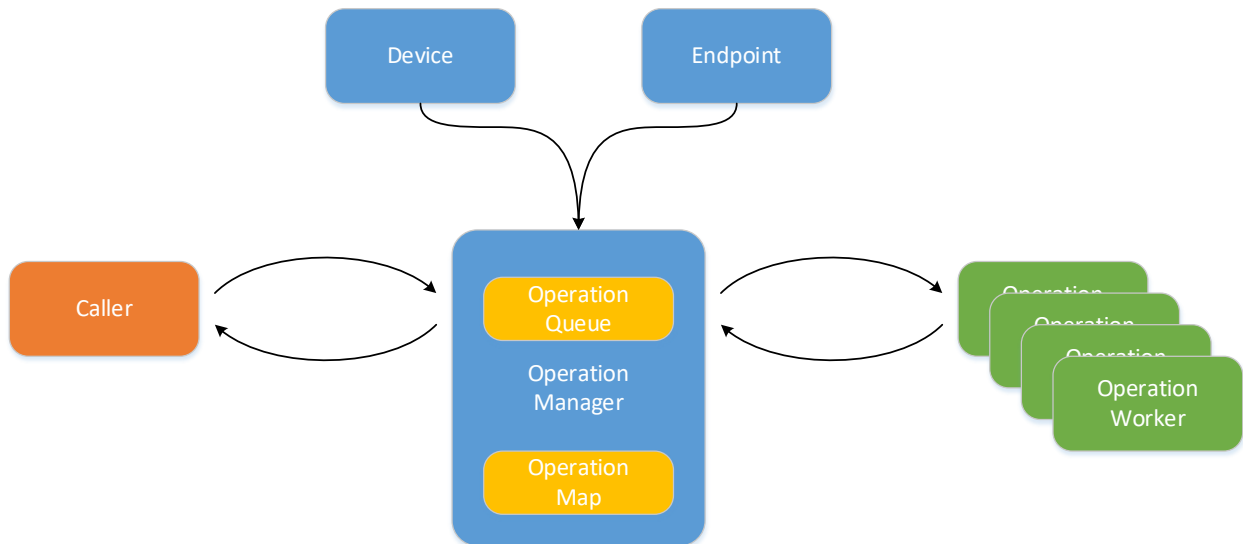


Figure 2.10: Architecture for the common DevComm (Device Communications) component

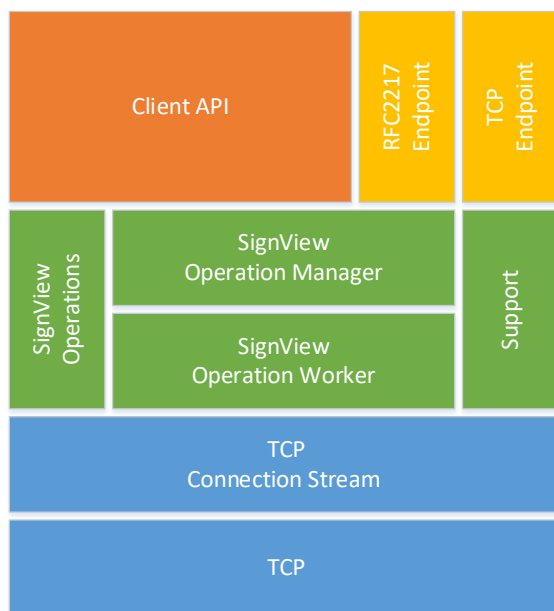


Figure 2.11: Architecture for the DMS-specific SignView DevComm plug-in

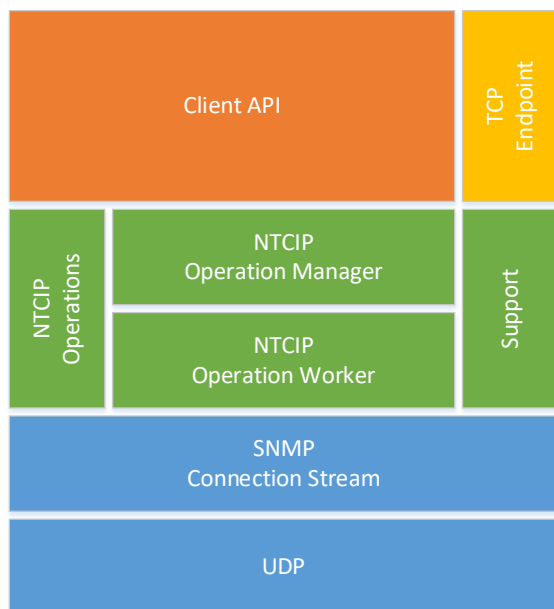


Figure 2.12: Architecture for the common NTCIP DevComm plug-in

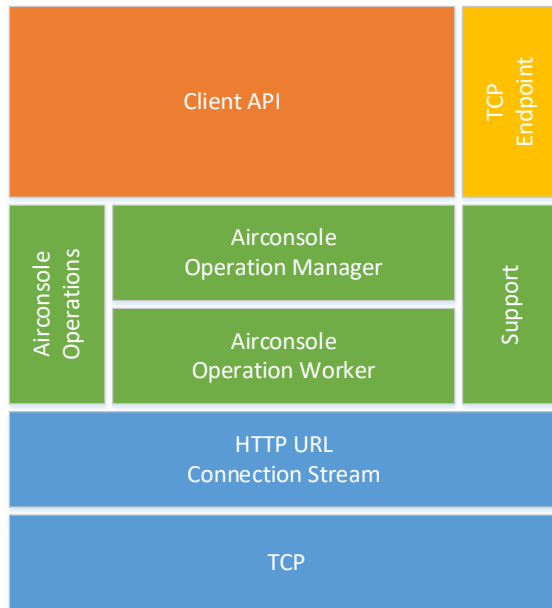


Figure 2.13: Architecture for the common Airconsole DevComm plug-in

The HHDC uses the software architecture pattern [2] Model-View-ViewModel (MVVM), as illustrated in Figure 2.14. The individual components are defined as follows:

- **View** is the actual user interface in the app (Activity, Fragment, or Custom View)
- **ViewModel** uses observable data to notify the view about data/state changes, passes events to the model, and converts model data to presentation-friendly data/state
- **Model** is the combination of data classes and service implementations

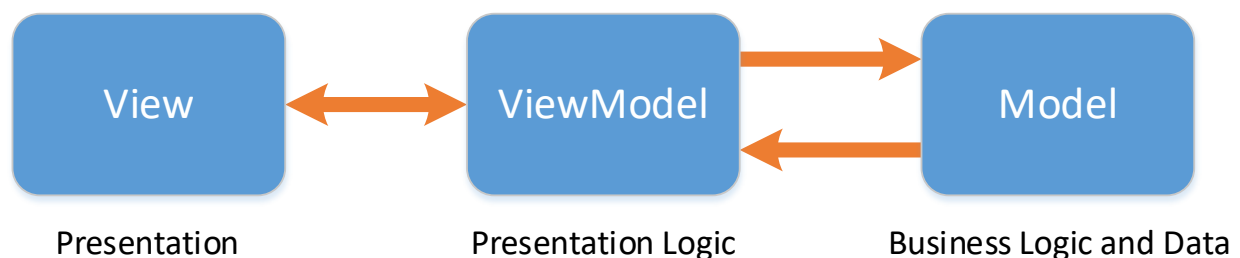


Figure 2.14: Model-View-ViewModel (MVVM) software pattern

Communications endpoints and parameters

The DMS, CCTV, and Console apps interface to their respective field element hardware through similar common communications endpoints, as shown in Table 2.1. The raw Transmission Control Protocol (TCP) endpoint parameters are

provided in Table 2.2, and the parameters for Request for Comments (RFC 2217 endpoint are provided in Table 2.3.

Table 2.1: HHDC communications endpoints

Endpoint	Note
Raw TCP endpoint	Typical host, port, timeout parameters used to establish a data stream over a TCP connection
Telnet endpoint	Additional timeout parameter, over TCP, used to establish a data stream over a Telnet connection
ComPort endpoint	Additional serial parameters, over Telnet, used to establish a data stream over an RFC 2217 connection

Table 2.2: Raw TCP endpoint parameters

Parameter	Parameter
Host	Read Timeout
Port	Connection Timeout
Idle Close Timeout	Socket Timeout

Common Data Storage

Log files are stored as follows:

- Text files, .log
- Typical Format
 - Timestamp
 - Level
 - ClassName
 - MethodName
 - Message
- EXTERNAL_FILES/logs/dms_<timestamp>.log

- Written on log export

Images are stored as follows:

- Joint Photographic Experts Group (JPEG) files, .jpg
- EXTERNAL_FILES/pictures/<timestamp>.jpg
- Written on image capture and/or annotation

Table 2.3: ComPort endpoint parameters

Parameter	Parameter
Host	Data Bits
Port	Parity
Idle Close Timeout	Stop Bits
Read Timeout	Flow Control
Connection Timeout	Request to Send (RTS)
Socket Timeout	Purge Receive (Rx)
Acknowledge Timeout	Purge Transmit (Tx)
Baud	

General Testing Approach

Testing of the software modules and apps followed a typical pattern in most cases:

- Run tests and monitor logs for results
- Android Profiler
 - CPU Profiler – inspect CPU usage, thread activity in real-time, and record method traces
 - Memory Profiler – identify memory leaks and memory churn that could lead to performance issues
- Manual Tests
 - Views
- Automated Unit Tests
 - ViewModels

- Models
- DevComm
- DevComm Plug-ins

Common Software Problems

Certain problems were encountered in the software common to both apps. These issues were resolved, and are presented here.

Airconsole Provisioning Issues

- All units were factory-programmed with the same Ethernet Media Access Control (MAC) address
- Shipped firmware version did not support routed mode
- Configuration:
 - Hardware lacked strict API
 - Hypertext Markup Language (HTML) driver
 - Blocking outbound traffic

Chapter 3:

Handheld Diagnostic Controller Design

DMS App

This chapter discusses design and implementation of the HHDC DMS app software. The analogous CCTV app design and implementation is presented in Chapter 4. The requirements for the HHDC DMS app are provided in Appendix C.

DMS App Overview

The HHDC DMS app supports DMS configuration and diagnostics. It interfaces to DMS controllers in the field through both network and serial interfaces, and supports full DMS configuration and diagnostics. The app supports all features of the Caltrans-specific DMS SignView protocol, as well as text, and algorithms to accelerate diagnostic operations of the panel, controller, and firmware. The DMS app also supports NTCIP communications with Caltrans' next-generation AVMS.

The DMS app can access Caltrans DMS (a.k.a. CMS) systems remotely over the network, or directly at the DMS field location. Local connection from the HHDC to the DMS components is supported by Wi-Fi-to-network, Universal Serial Bus (USB)-to-network, Wi-Fi-to-serial, or USB-to-serial adapters and associated software. Remote network connection is typically over standard tablet Wi-Fi or cellular connection.

Using these various wired and wireless connection methods, and supported by the DMS app and HHDC kit contents, it is possible to connect to the DMS controllers over network and/or serial connections for diagnostic and configuration purposes. Some common Caltrans use cases are:

1. Wired/wireless network connection to a DMS field element cabinet
2. Wired/wireless network connection to a terminal server
3. Wired/wireless serial connection to a DMS controller

The DMS app is standards-based wherever possible, and support standards commonly used by Caltrans and other DOTs. The app support local and remote diagnosis and configuration of DMS controllers and signs. For Caltrans-specific use, the app supports the DMS SignView protocol. The app also supports NTCIP communications. While NTCIP is a widely used standard, implementations typically vary. For this research, NTCIP support is provided specifically for

in camera and image gallery to allow a user to take and annotate pictures of a DMS for illustration of sign functionality and status. Finally, the DMS app supports logging for error handling and debugging. The sign and camera function design mockups will be discussed in this section.

The mockup shows a smartphone screen with a 'Create sign' form. The form has a title bar with a close button (X) and a checkmark. Below the title bar, there are three text input fields labeled 'Name', 'Model', and 'Location'. Below these fields is a section titled 'Communications'. This section contains two groups of input fields. The first group is preceded by a yellow circle with the letter 'F' and contains three input fields labeled 'First name', 'Protocol', and 'Endpoint'. The second group is preceded by a green circle with the letter 'S' and contains three input fields labeled 'Second name', 'Protocol', and 'Endpoint'. Below these groups are three vertical dots indicating more options. At the bottom right of the form is a circular button with a plus sign (+).

Figure 3.2: Create sign form mockup

Figure 3.2 shows the mockup for the DMS sign creation form. This form allows the user to set the sign name, designate the sign model (e.g. AVMS 700), and set the sign location. The form also allows the user to select the communications

type (name, protocol, and endpoint). The list of communications types includes a random statically-colored circle icon including a letter corresponding to the first letter of the communications type name, for easier identification. The user can create a new communications channel by clicking on the plus (+) icon in the lower right. To cancel the current sign creation, the user would click the 'X' icon in the upper left. Finally, to save the created sign, the user would click the checkmark (✓) icon in the upper right. In this and similar mockups, the three large dots represent repetition of elements.

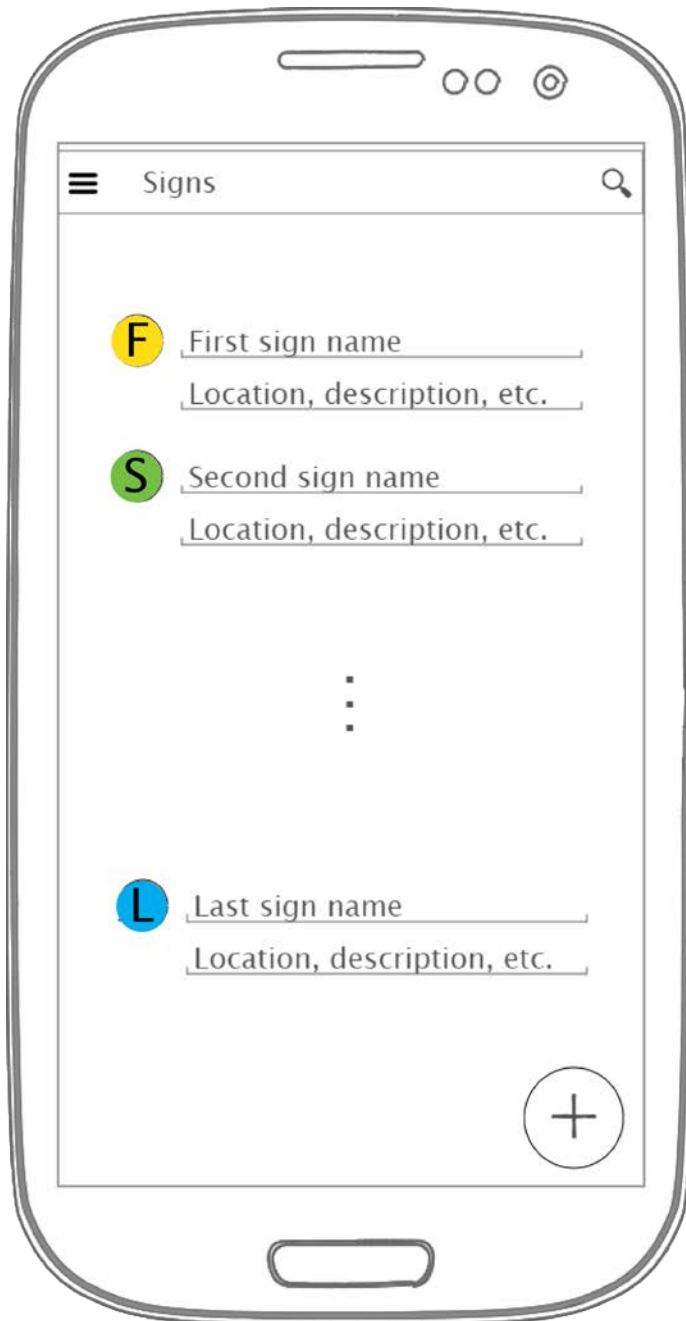


Figure 3.3: Sign list mockup

Figure 3.3 shows the mockup for the DMS sign list. This screen shows a list of the currently configured DMS signs. Each sign entry includes the sign name and an auxiliary field which provides the sign's location, description, etc. Each sign entry in the list includes a random statically-colored circle icon including a letter corresponding to the first letter of the sign name, for easier identification. The user can click on any sign entry to view that sign's details. The user can create a new sign by clicking on the plus (+) icon in the lower right. The user can search for a specific sign by name by clicking the search icon (magnifying glass) in the upper right.



Figure 3.4: Signs preview mockup

Figure 3.4 shows the mockup for the graphical DMS signs preview. This screen shows a list of the currently configured DMS signs, including an image of the last detail request for each sign. Each sign entry includes the sign name and location, along with the connection type (network or serial). The user can create a new sign by clicking on the plus (+) icon in the lower right. The user can search for a specific sign by name by clicking the search icon (magnifying glass) near the upper right. Finally, the user can sort the signs by name and also refresh the sign previews via the menu (three small dots) in the upper right.

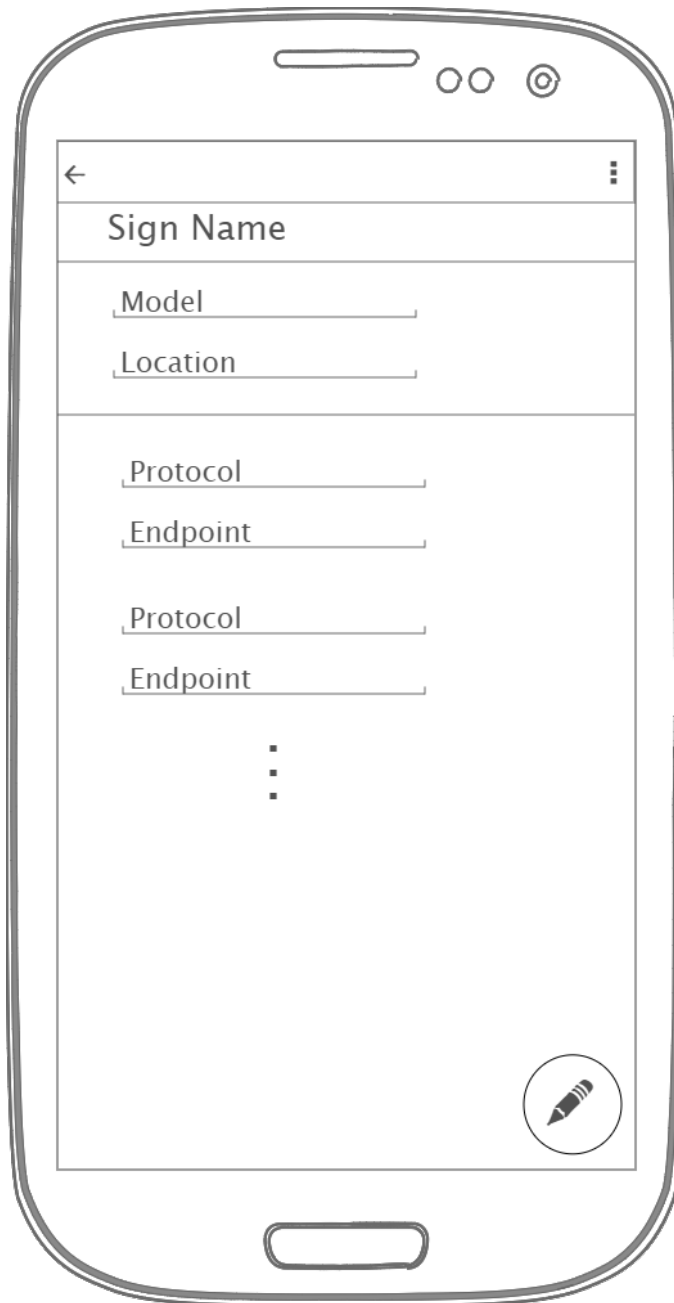


Figure 3.5: Sign overview mockup including edit and delete capability

Figure 3.5 shows the mockup for the DMS sign overview. This screen shows the sign name for the selected sign, along with the model and location. The form also shows the list of protocols and endpoints for the selected sign. The user can edit the sign by clicking on the pencil (✎) icon in the lower right. The user can delete the sign via the menu (three small dots) in the upper right. Finally, the user can return to the previous screen using the back-arrow (←) in the upper left.

× Edit sign ✓ ⋮

Name

Model

Location

Communications

F First name

Protocol

Endpoint

S Second name

Protocol

Endpoint

⋮

+

Figure 3.6: Edit sign form mockup

Figure 3.6 shows the mockup for the DMS sign editing screen. This screen allows the user to modify the sign name, model, and location. The form also allows the user to select the communications types (name, protocol, and endpoint). The list of communications types includes a random statically-colored circle icon including a letter corresponding to the first letter of the communications type name, for easier identification. The user can create a new communications channel by clicking on the plus (+) icon in the lower right. To cancel the current sign edit, the user would click the 'X' icon in the upper left. To save the created sign, the user would click the checkmark (✓) icon near the upper right. Finally, the user can delete the sign via the menu (three small dots) in the upper right.

The mockup shows a mobile phone interface for creating a communication. At the top, there's a status bar with three icons. Below it is a modal window titled "Create communication" with a close button (X) on the left and a checkmark icon on the right. Inside the modal, there are two input fields: "Name" and "Protocol" (which has a dropdown arrow). Below these fields, the text "TCP Fragment or RFC2217 Fragment" is displayed. The phone's home button is visible at the bottom.

Figure 3.7: Create communication form mockup

Figure 3.7 shows the mockup for the DMS communications creation form. This form allows the user to set the communications name, and select the protocol from a drop-down list. Depending on the protocol selected, either a TCP or Request for Comments (RFC) 2217 (Telnet COM Port Control Option)⁹ fragment will appear in the large area at the bottom of the screen. These two fragments are described below. To cancel the current communications creation, the user would click the 'X' icon in the upper left. Finally, to save the created communications, the user would click the checkmark (✓) icon in the upper right.

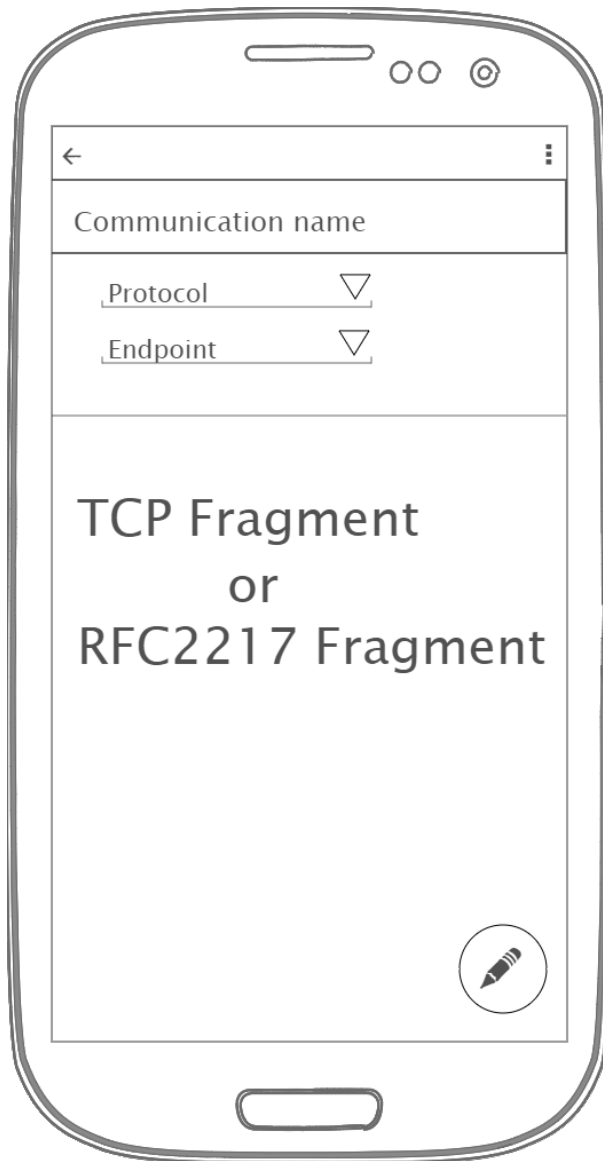
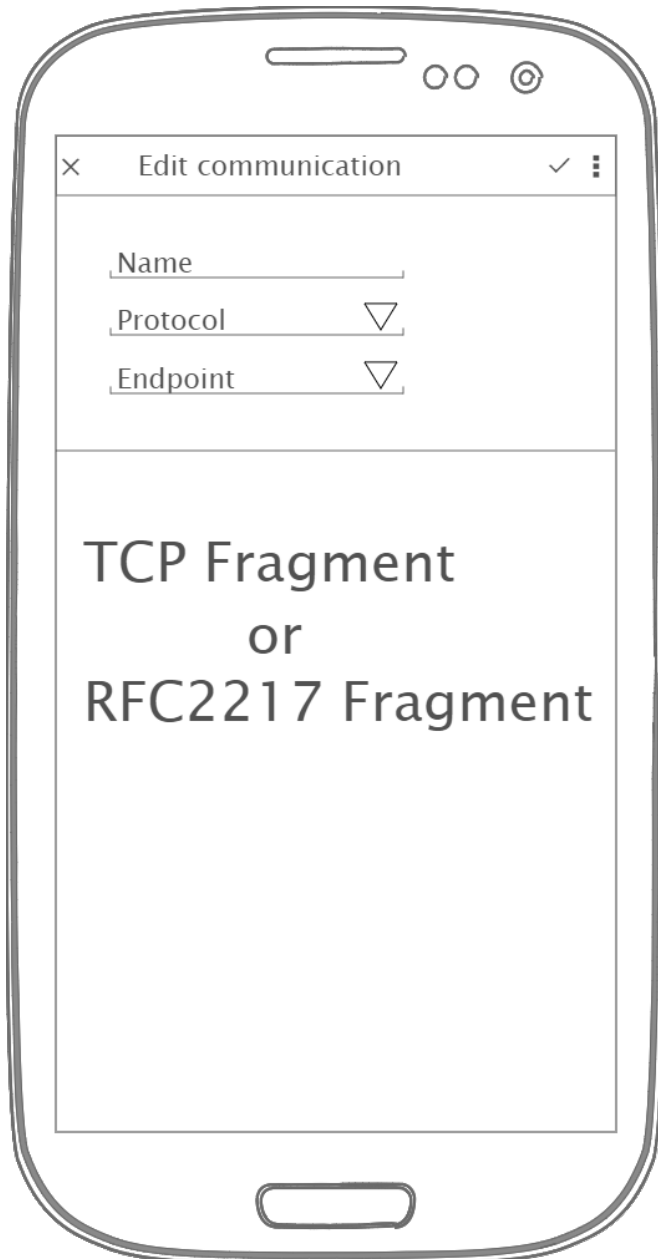


Figure 3.8: Communication overview mockup including edit and delete capability

⁹ <https://tools.ietf.org/html/rfc2217>

Figure 3.8 shows the mockup for the DMS communications details. This screen shows the communications name for the selected communication, along with the selected protocol and endpoint. Depending on the protocol selected, either a TCP or ComPort fragment will appear in the large area at the bottom of the screen. These two fragments are described below. The user can edit the protocol details by clicking on the pencil (✎) icon in the lower right. The user can delete the communications via the menu (three small dots) in the upper right. Finally, the user can return to the previous screen using the back-arrow (←) in the upper left.

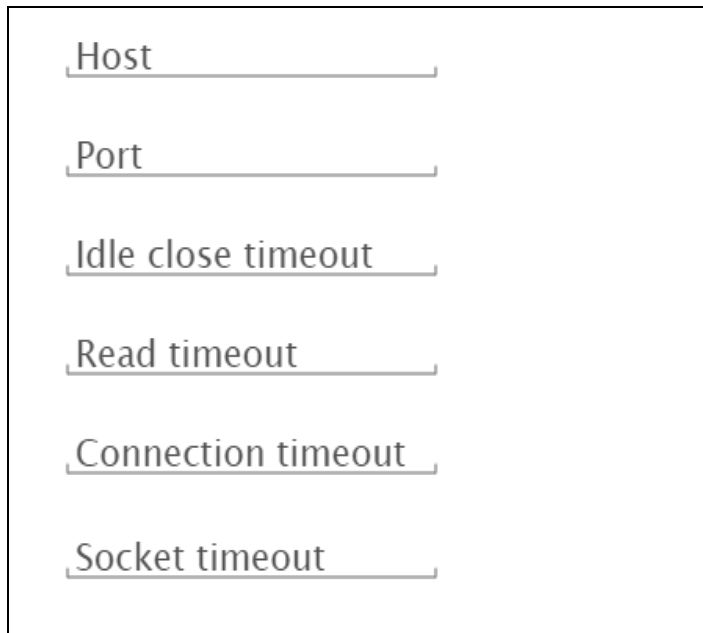


The mockup shows a smartphone screen with a rounded top and a home button at the bottom. The screen displays a form titled "Edit communication" with a close button (X) on the left and a checkmark and menu icon (three dots) on the right. The form contains three input fields: "Name" (text), "Protocol" (dropdown menu), and "Endpoint" (dropdown menu). Below these fields is a large text area containing the text "TCP Fragment or RFC2217 Fragment".

Figure 3.9: Edit communication form mockup

Figure 3.9 shows the mockup for the DMS communications editing screen. This screen allows the user to modify the communications name, protocol, and endpoint. Depending on the protocol selected, either a TCP or ComPort fragment will appear in the large area at the bottom of the screen. These two fragments are described below. To cancel the current communications edit, the user would click the 'X' icon in the upper left. To save the created communications, the user would click the checkmark (✓) icon near the upper right. Finally, the user can delete the communications via the menu (three small dots) in the upper right.

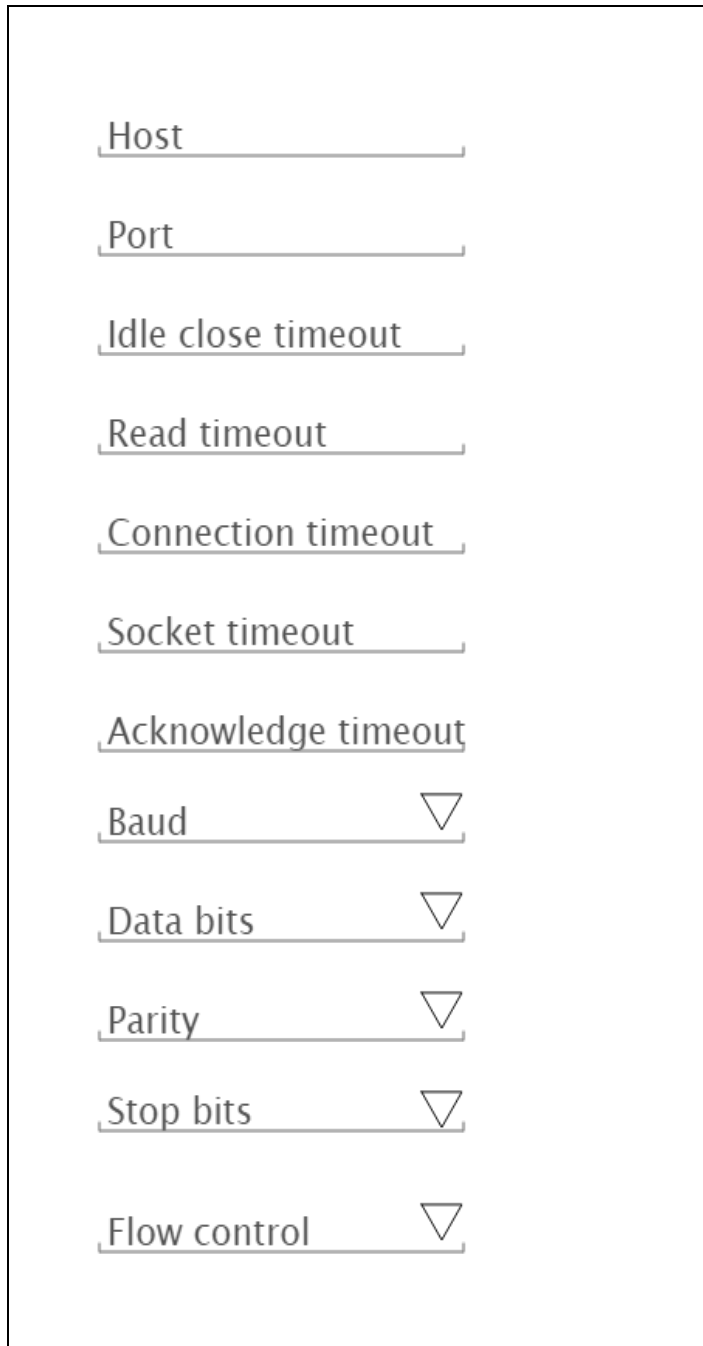
Figure 3.10 shows the mockup for the DMS TCP fragment. Depending on the selected protocol, this fragment appears in the screens for communications creation, communications overview, and communications edit. This screen allows the user to modify the TCP-specific communications parameters, including host, port, idle close timeout, read timeout, connection timeout, and socket timeout.



The mockup shows a vertical list of six input fields, each with a label and a text entry area. The labels are: Host, Port, Idle close timeout, Read timeout, Connection timeout, and Socket timeout. Each label is positioned to the left of its corresponding input field, which is a horizontal line with rounded ends.

Figure 3.10: TCP fragment mockup

Figure 3.11 shows the mockup for the DMS ComPort fragment. Depending on the selected protocol, this fragment appears in the screens for communications creation, communications overview, and communications edit. This screen allows the user to modify the ComPort-specific communications parameters, including host, port, idle close timeout, read timeout, connection timeout, socket timeout, acknowledge timeout, and serial parameters. The serial parameters include baud rate, data bits, parity, stop bits, and flow control. The fragment also supports setting Request To Send (RTS), Receive (Rx) purge, and Transmit (Tx) purge.



Host

Port

Idle close timeout

Read timeout

Connection timeout

Socket timeout

Acknowledge timeout

Baud ▾

Data bits ▾

Parity ▾

Stop bits ▾

Flow control ▾

Figure 3.11: ComPort fragment mockup

DMS App Implementation

The HHDC DMS app is dedicated to DMS (CMS and AVMS) diagnostics and control. The DMS software architecture is shown in Figure 3.12

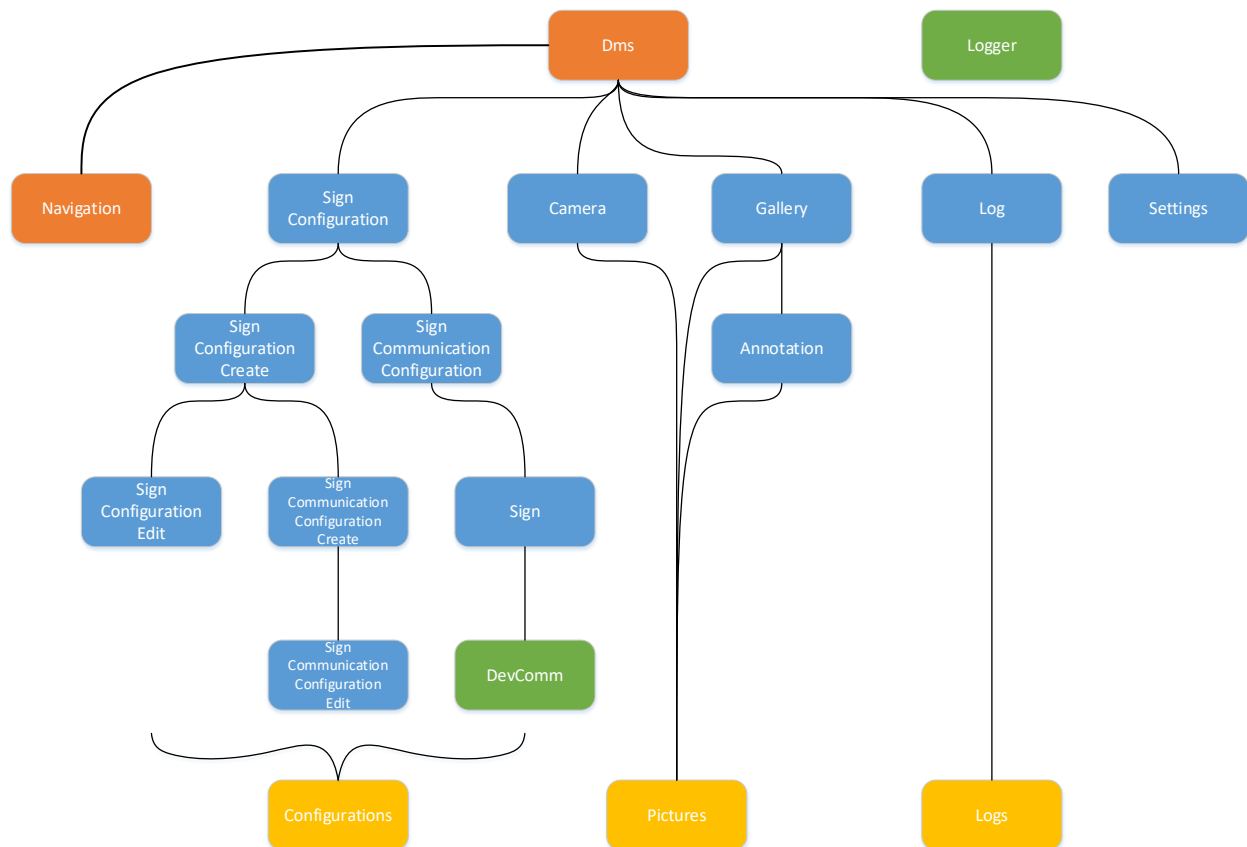


Figure 3.12: HHDC DMS software architecture

Figure 3.13 shows a detailed design of the configuration states and transitions between the states. Additionally shown, is the detailed definitions of the node state, transition, action, and state change. Design of this graph/architecture was required to insure robust implementation of the various (non-editable, create, edit, selectable) sign and communication configurations throughout the app. This design and strict adherence drove lower level design to ensure defect free implementation.

The HHDC software supports a wide range of Caltrans DMS with various display matrix dimensions, including traditional CMS and the newer AVMS models, as shown in Table 5.1.

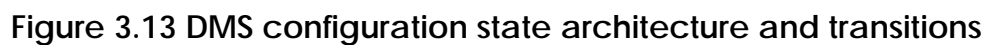


Table 3.1: Supported CMS and AVMS signs

Sign type	Display matrix
CMS 500	96 x 25
CMS 510	96 x 25
CMS 520	48 x 25
AVMS 710	105 x 27
AVMS 720	95 x 27
AVMS 730	55 x 27

The HHDC software also supports a wide range of controllers and communications modes, as shown in Table 3.2.

Table 3.2: Supported DMS controllers

Controller	Communications modes
170E	RS-232 (C2 connector) Ethernet
2070E	RS-232/485 Ethernet
AVMS controller	Ethernet
General	Serial (RS-232/422/485) Ethernet

The HHDC supports two primary sign protocols, i.e. those currently in use by Caltrans (Table 3.3). As noted, SignView is a Caltrans-specific protocol / standard. The HHDC supports all known deployed versions of 170/2070/AVMS SignView implementations. The SignView implementations are highly fragmented by device type and version. The HHDC implementation is based on the Caltrans SignView protocol standard plus support for implementation variances. It is hardware and version agnostic. NTCIP is a national standard for DMS and other field elements. It is supported on the AVMS models.

Table 3.3: Supported sign protocols

Protocol	Note
SignView	Caltrans-specific protocol / standard
NTCIP core	1203 v03 standard

The HHDC app can handle various DMS configurations. Several configurations exist in various Caltrans districts, as illustrated in Figures 3.14 – 3.16.

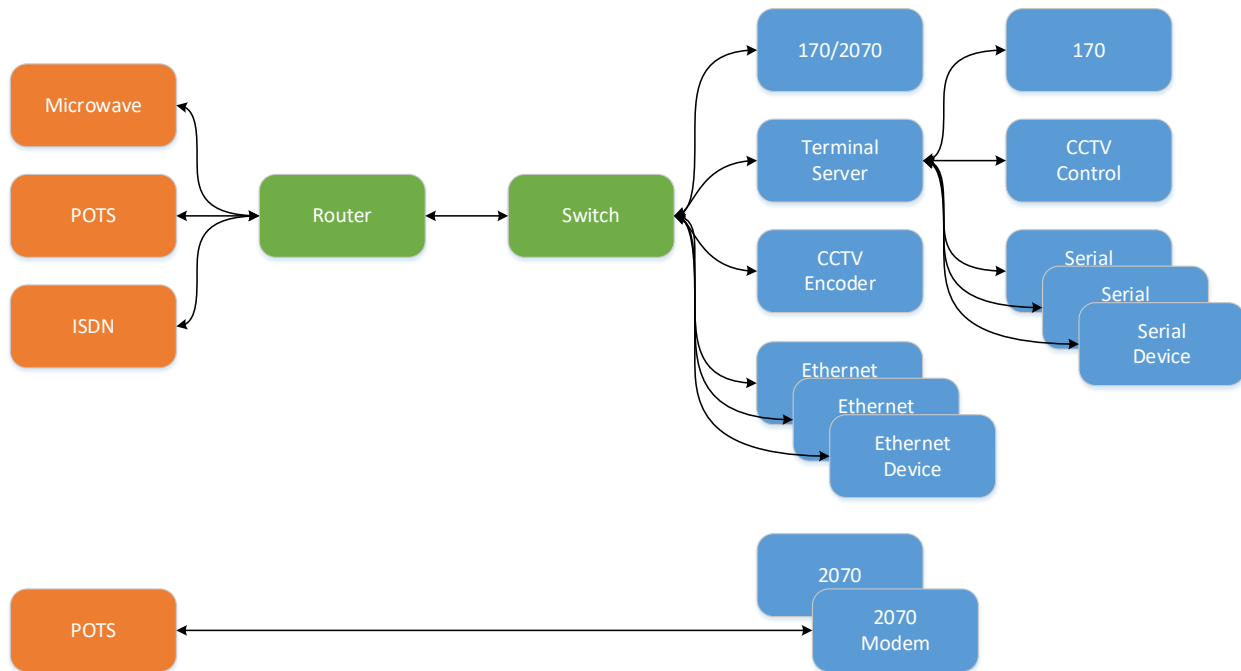


Figure 3.14: District configuration DMS type A

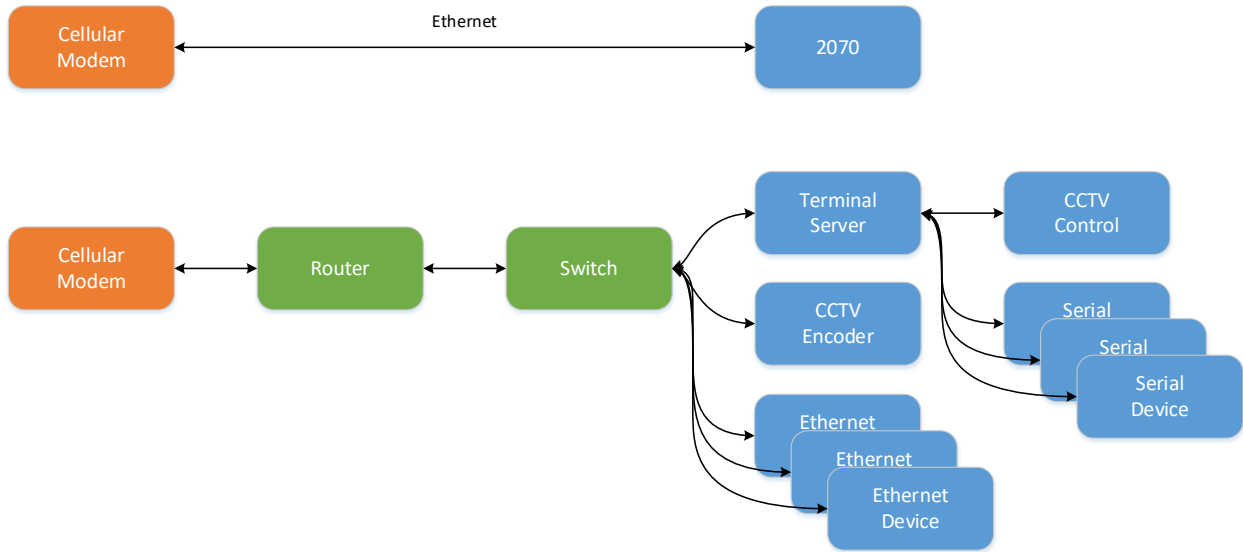


Figure 3.15: District configuration DMS type B

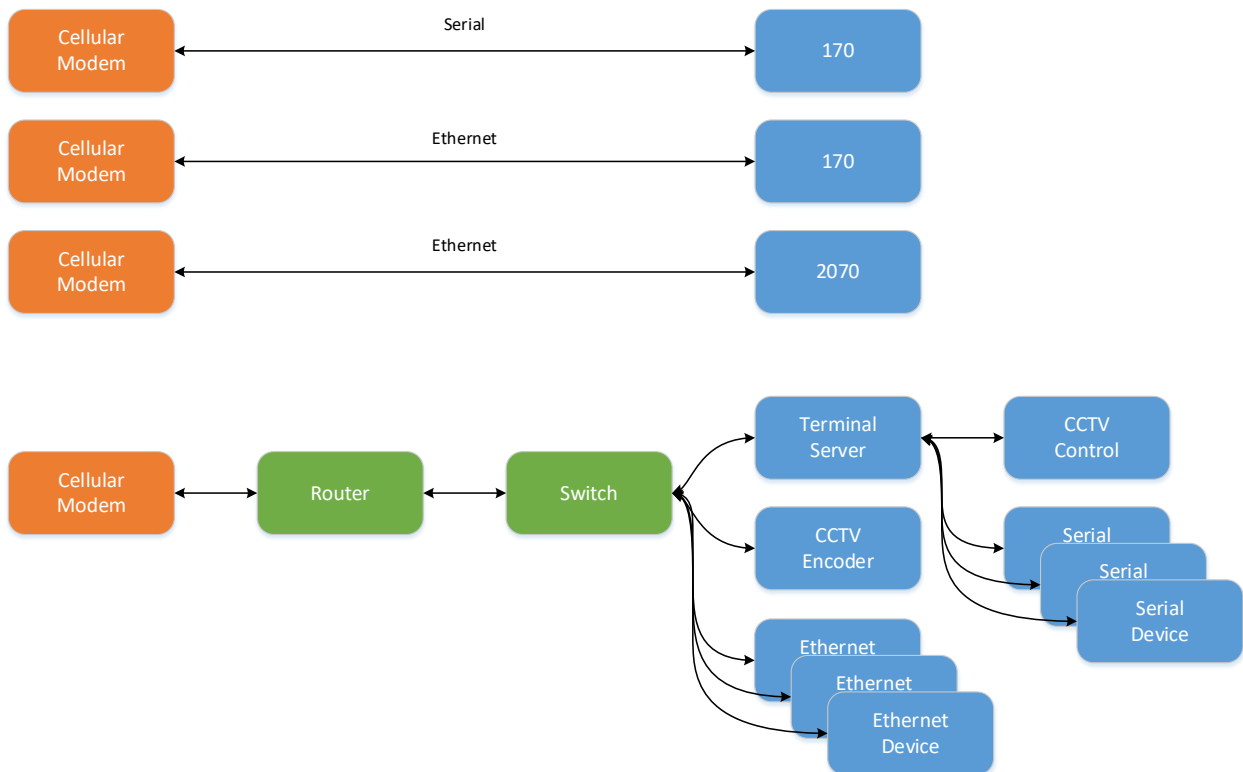


Figure 3.16: District configuration DMS type C

Sign configurations are stored as follows:

- JavaScript Object Notation (JSON) .json files
- Primarily (de)serialization of classes:
 - SignConfiguration

- SignCommunicationConfiguration
- TcpEndpoint
- ComPortEndpoint
- EXTERNAL_FILES/configurations/dms_signs.json
- Updated after every modification

Use cases are illustrated in Figure 3.17 (typical) and Figure 3.18 (requiring field element network visibility).

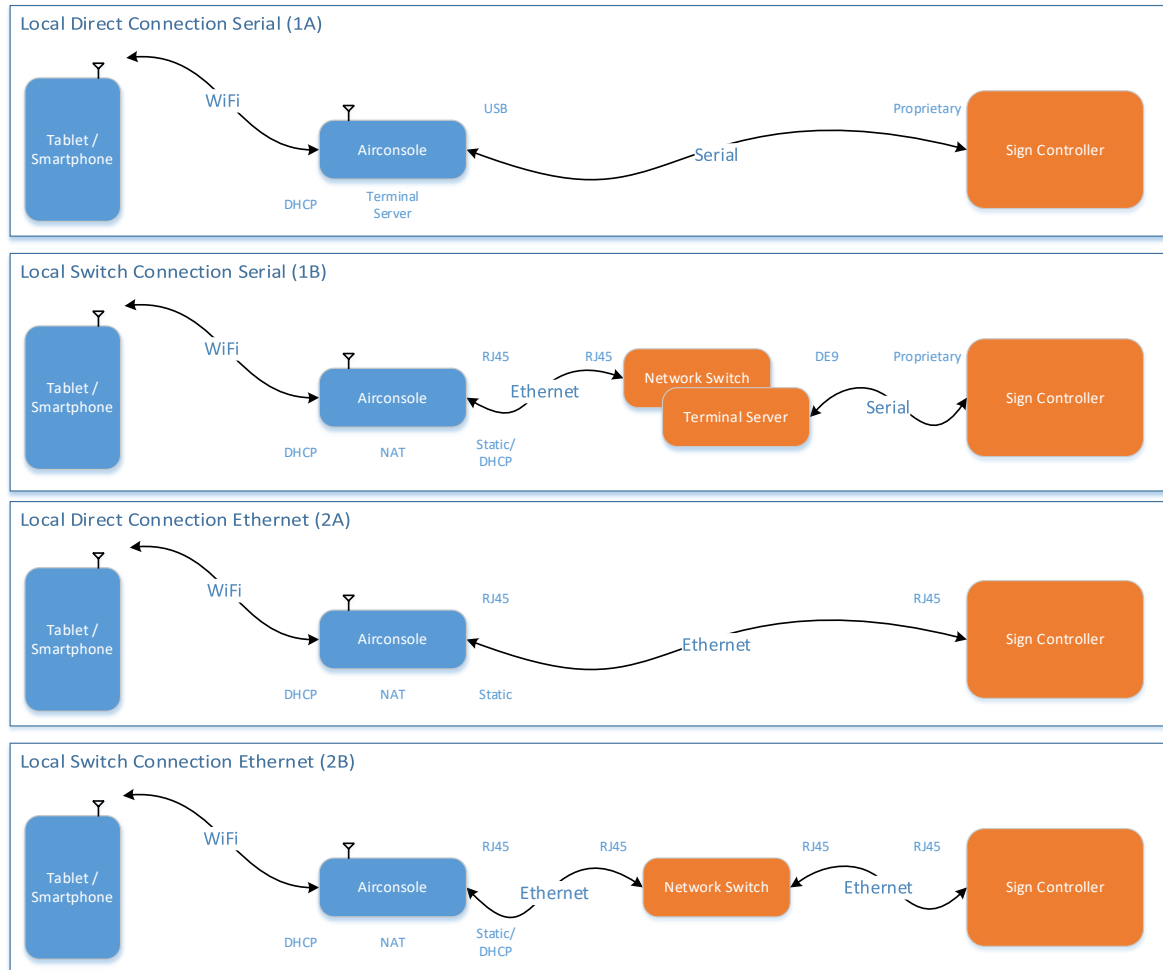


Figure 3.17: Typical DMS use cases

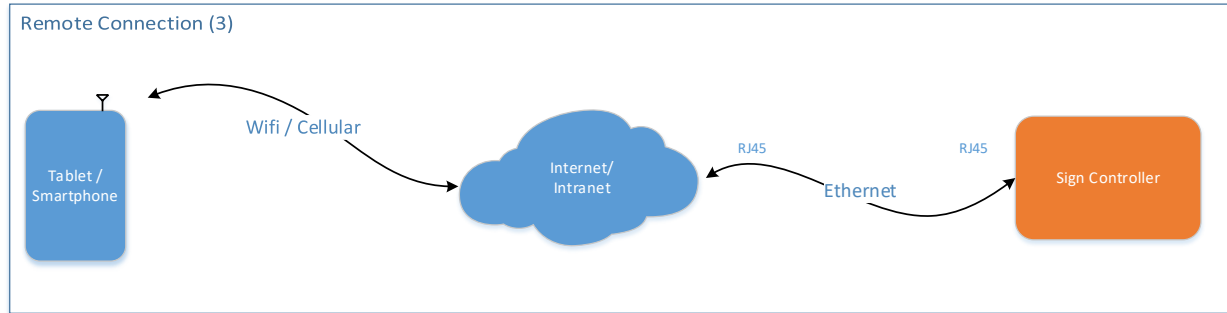


Figure 3.18: DMS use case requiring field element network visibility

DMS Software Problems

SignView Issues

- General:
 - Some field controllers incorrectly implement protocol
 - Some field controllers do not implement entire protocol
 - Omissions/errors/ambiguities in protocol specifications
- Specific:
 - Request Detail TMC Blank Wrong Bytes Remaining
 - Request Detail TMC Blank 0-Byte Payload
 - Request Detail TMC Blank 7-Byte Payload
 - Reply Clipped Tag
 - Request Detail Allow Field Blank
 - Request Detail Field Blank Wrong Bytes Remaining
 - Request Detail Health Check
 - Default to Display TMC Blank
 - Allow Reply Pad

NTCIP Issues

- Vendor:
 - Buggy implementations (e.g. crash if greater than one request per connection)
 - Incomplete implementations
- Specification Ambiguities:

- Line/page/message scopes for some MULTI attributes
- Page on/off time edge cases (i.e. [ptx] followed by [ptoy])
- Minimum spacing between line/page justification regions
- Pixel rounding in center-justification regions
- Minimum character spacing in full-justification regions
- Text alignment when multiple fonts on same line

DMS App Screen Shots

This section contains the screen shots for the DMS app, which is based on the design mockups of the previous section.

Figure 3.19 shows the primary startup screen for the DMS app. Typically, after opening the app, the end user will immediately select a predefined sign configuration to begin testing and/or operating a sign controller. Clicking on the add button will begin creation of a new sign configuration.

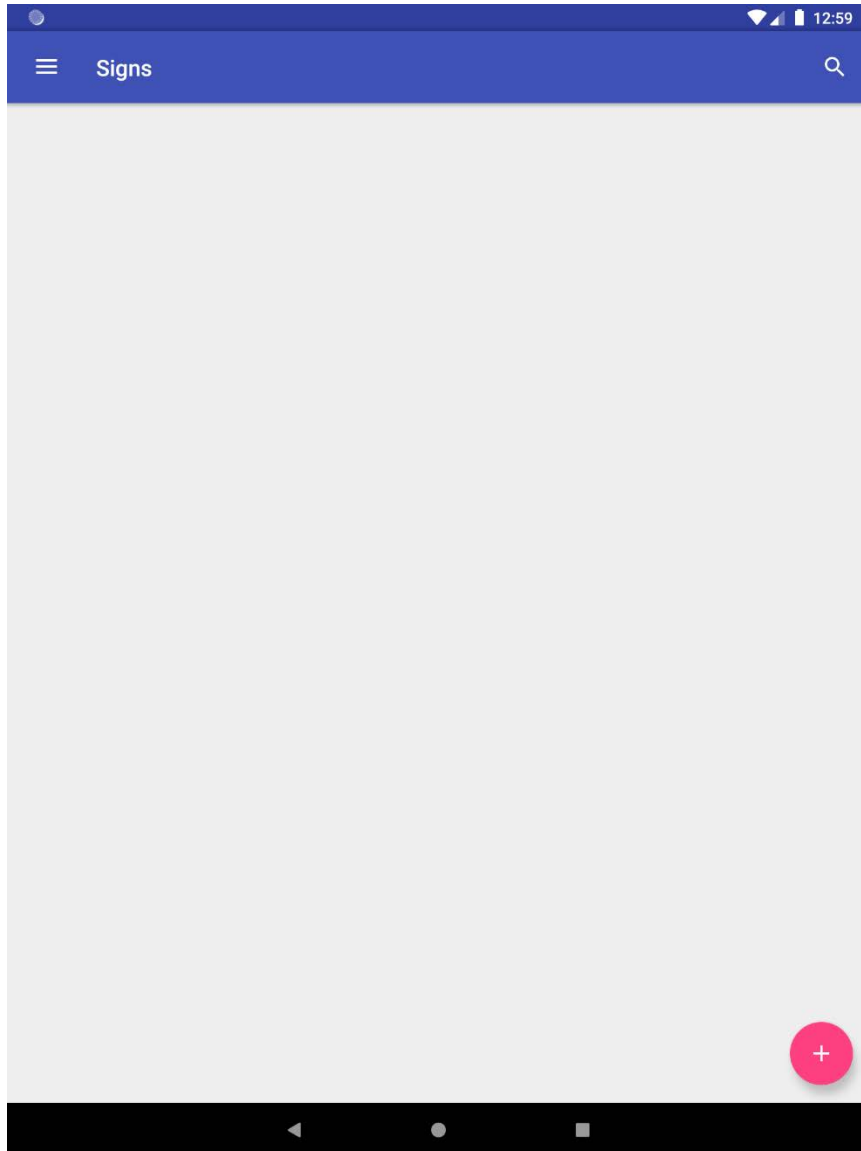


Figure 3.19: DMS startup screen

Figure 3.20 shows the primary menu of the DMS app. The menu is opened using one of two methods. The first is by swiping from the left edge of the screen to the right, and the second is by clicking on the 3-line icon in the upper left corner of all other screens (e.g. Figure 3.21). The menu is closed by either selecting a menu item, or swiping left. Using this menu, the user can access functionality for signs, app configuration, DMS app built-in camera and gallery, a log, and app settings.

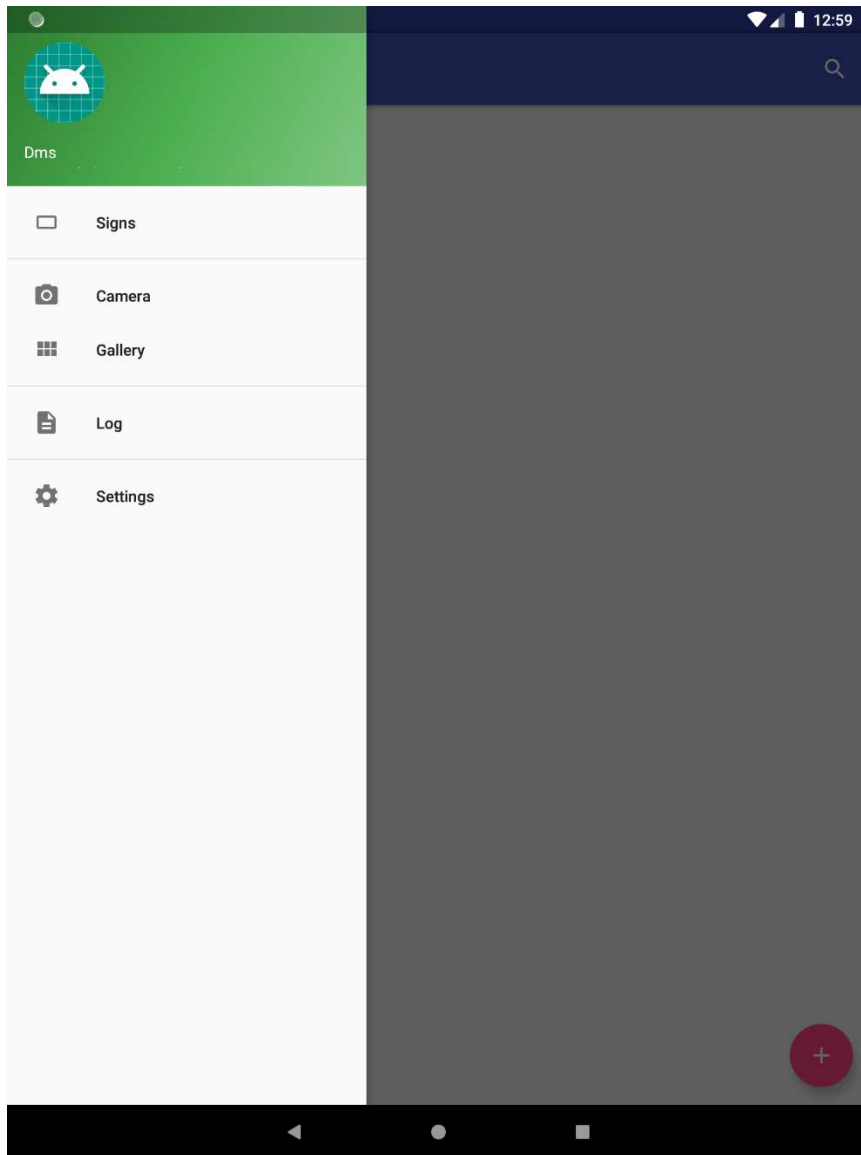


Figure 3.20: DMS main menu

Figure 3.21 shows the default state of a new create sign configuration screen. The sign configuration is composed of a sign name, model, location, and one or more sign communication methods. Here, the user is configuring a model 500 sign with a 170 controller running SignView communicating over a serial connection.

The screenshot displays the 'Create Sign Configuration' interface. At the top, a blue header bar contains a menu icon, the title 'Create Sign Configuration', and action icons (cancel and confirm). The main content area is divided into sections. The first section contains three input fields: 'Name' (with the value 'name'), 'Model' (a dropdown menu currently showing '500'), and 'Location' (with the value 'location'). Below these is a large, empty light gray area labeled 'Communications'. A red circular button with a white plus sign is positioned in the bottom right corner of the 'Communications' section, indicating where to add new communication methods. The bottom of the screen features a standard Android navigation bar.

Figure 3.21: DMS create sign configuration

Figure 3.22 shows that the majority of the various configuration screens are composed of intuitive text entry fields and drop-down selection boxes. When entering values in free-form fields, the entry will be continuously validated. When appropriate, a meaningful error message will be displayed below the associated field to prompt the user for corrective action. In this case, the name field must not be empty.

The screenshot displays a mobile application interface for creating a sign configuration. At the top, a blue header bar contains a hamburger menu icon, the title 'Create Sign Configuration', and 'X' and checkmark icons. Below the header, the 'Name' field is highlighted in red, with a red error message 'The name can not be empty' displayed underneath it. The 'Model' field is a dropdown menu currently showing '500'. The 'Location' field contains the text 'location'. Below these fields is a section titled 'Communications'. A pink circular button with a white plus sign is located on the right side of the screen. At the bottom, a virtual keyboard is visible, showing the text 'thanks | | we' in the input area and a list of letters and symbols on the keys.

Figure 3.22: DMS create sign configuration name field error message

Figure 3.23 shows the sign configuration selectable model options. Drop-down selection boxes do not include entry validation or error messages. A sign configuration must have a defined model number, which defaults to the prepopulated value of 500.

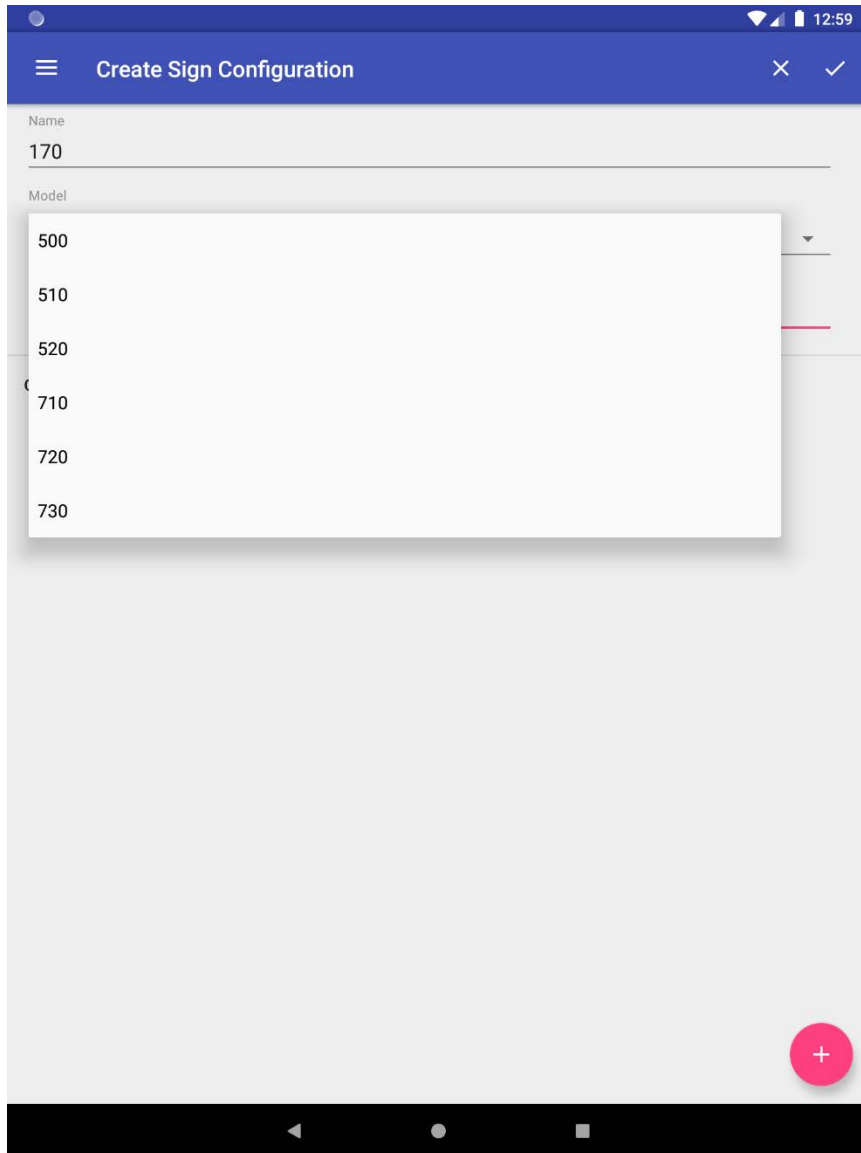


Figure 3.23: DMS create sign configuration model options

Figure 3.24 shows via the highlighted error message that a sign configuration must have a defined location. Clicking on the add button will begin creation of a new sign communication configuration associated with this sign.

The screenshot displays a mobile application interface titled "Create Sign Configuration". At the top, there is a blue header bar with a menu icon on the left, the title "Create Sign Configuration" in the center, and close and checkmark icons on the right. The status bar at the very top shows the time as 12:59 and various system icons. Below the header, the form contains three input fields: "Name" with the value "170", "Model" with the value "500", and "Location" which is currently empty. The "Location" field is highlighted with a red border, and a red error message "The location can not be empty" is displayed directly below it. Underneath the form fields is a section titled "Communications" which is currently empty. At the bottom right of the form area, there is a prominent pink circular button with a white plus sign. The entire interface is set against a light gray background, and the bottom of the screen shows the standard Android navigation bar.

Figure 3.24: DMS create sign creation location error message

Figure 3.25 shows the default sign communication configuration. All sign communication configurations are composed of a name, protocol, endpoint type, drop, and specific additional parameters depending on the selected endpoint type.

The screenshot shows a mobile application interface for creating a sign communication. The title bar is blue with a hamburger menu icon on the left, the text 'Create Sign Communication' in the center, and a close (X) and confirm (checkmark) icon on the right. The status bar at the top right shows signal strength, battery level, and the time 12:59. The form consists of several input fields with labels above them: 'Name' (Ethernet via Airconsole), 'Protocol' (Signview), 'Endpoint type' (ComPort), 'Drop' (81), 'Host' (192.168.10.1), 'Port' (3696), 'Idle Close Timeout' (0), 'Read Timeout' (16000), 'Socket Connection Timeout' (20000), 'Socket Read Timeout' (20000), 'Acknowledgement Timeout' (8000), 'Baud' (1200), and 'Data bits' (8). The bottom of the screen shows the Android navigation bar with back, home, and recent apps icons.

Field	Value
Name	Ethernet via Airconsole
Protocol	Signview
Endpoint type	ComPort
Drop	81
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	8

Figure 3.25: DMS create sign communication with default values

Figure 3.26 shows the communication configuration name being set serial via Airconsole in our case. The intention of the name field is to descriptively and uniquely identify one associated communication configuration from another. A sign configuration can contain multiple associated communication configurations. A single sign may be accessed over the internet via wireless or cellular internet connection, locally over Ethernet via the Airconsole, or locally over Serial via the Airconsole. A single sign configuration can contain multiple communication configurations for different use cases.

Create Sign Communication

Name
Serial via Airconsole

Protocol
Signview

Endpoint type
ComPort

Drop
81

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Figure 3.26: DMS create sign communication name settings

Figure 3.27 shows the communication configuration protocol (NTCIP, SignView), endpoint type (TCP, Telnet, ComPort), and drop being set. In this case, the default protocol SignView and default endpoint type ComPort are appropriate. The drop of the example 170 is set to 80. The remaining sign communication configuration parameters are dependent on the endpoint type selection. In all cases this includes a host, port, idle close timeout, read timeout, socket connection timeout, and socket read timeout. The ComPort case includes additional parameters. See below in the configurations section for parameter details.

Create Sign Communication

Name
Serial via Airconsole

Protocol
Signview

Endpoint type
ComPort

Drop
80

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Figure 3.27: DMS create sign communication port setting

Figure 3.28 shows the additional configuration parameters acknowledgement timeout, baud, data bits, parity, stop bits, flow control, rts, purge rx, and purge tx. See below in the configurations section for parameter details. The selectable data bits options are 5, 6, 7, and 8.

The screenshot displays the 'Create Sign Communication' interface. It features a blue header bar with a menu icon, the title 'Create Sign Communication', and close/confirm buttons. Below the header, various configuration parameters are listed with their current values: Host (192.168.10.1), Port (3696), Idle Close Timeout (0), Read Timeout (16000), Socket Connection Timeout (20000), Socket Read Timeout (20000), Acknowledgement Timeout (8000), Baud (1200), Data bits (5), Flow control (none), Rts, Purge Rx, and Purge Tx. The Data bits dropdown menu is open, showing the options 5, 6, 7, and 8. The Rts, Purge Rx, and Purge Tx settings are represented by red toggle switches.

Parameter	Value
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	5
Flow control	none
Rts	On
Purge Rx	On
Purge Tx	On

Figure 3.28: DMS create sign communication data bits options

Figure 3.29 shows the selectable parity options of none, odd, even, mark, and space.

The screenshot shows a mobile application interface for creating sign communication. The title bar is blue with a hamburger menu icon on the left and close/cancel and check/confirm icons on the right. The form fields are as follows:

Field	Value
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	8
Parity	none
Purge rx	<input type="checkbox"/>
Purge Tx	<input type="checkbox"/>

Figure 3.29: DMS create sign communication parity options

Figure 3.30 shows the selectable stop bits options of 1, 2, and 1.5.

Create Sign Communication

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Parity
none

Stop bits
1
2
1.5

Purge Rx ☒

Purge Tx ☒

Figure 3.30: DMS create sign communication stop bits options

Figure 3.31 shows the selectable flow control options of none, xon/xoff, and rts/cts.

Create Sign Communication

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Parity
none

Stop bits
1

Flow control
none
XON/XOFF
RTS/CTS

Three red toggle switches are visible on the right side of the Flow control section, all of which are currently turned on.

Figure 3.31: DMS create sign communication flow control options

Figure 3.32 shows the completed sign and associated serial communication configuration. Clicking on the check mark icon will save this configuration, while clicking on the x icon will discard this configuration and return to the signs screen. Clicking on the add icon will add another communication configuration. Clicking on the existing communication configuration will open the configuration in read-only mode with menu option to delete, or clickable option to edit.


Create Sign Configuration

Name
170

Model
500

Location
ucdavis

Communications

-  Serial via Airconsole
Signview
ComPort

+

Figure 3.32: DMS sign configuration with a serial communication configuration

Figure 3.33 shows an existing selected sign communication configuration. Clicking on the pencil edit icon will open the edit sign communication screen and allow for modification of this communication configuration. Clicking on the left-arrow icon will go back to the previous screen.

The screenshot displays a mobile application interface for configuring sign communication. The title bar is blue with a hamburger menu icon on the left, the text 'Sign Communication' in the center, and a back arrow and three-dot menu icon on the right. The status bar at the top shows signal strength, Wi-Fi, battery, and the time 6:21. The main content area has a light gray background and is titled 'Serial via Airconsole'. It contains a series of form fields, each with a label and a value: Protocol (Signview), Endpoint type (ComPort), Drop (80), Host (192.168.10.1), Port (3696), Idle Close Timeout (0), Read Timeout (16000), Socket Connection Timeout (20000), Socket Read Timeout (20000), Acknowledgement Timeout (8000), Baud (1200), Data bits (8), and Parity. A pink circular button with a white pencil icon is located in the bottom right corner of the form area. The bottom of the screen shows the Android navigation bar.

Field	Value
Protocol	Signview
Endpoint type	ComPort
Drop	80
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	8
Parity	

Figure 3.33: DMS read-only sign communication option to edit

Figure 3.34 shows the “more options” menu item delete. Clicking on delete will remove this sign communication configuration from its associated sign configuration and return to the previous screen.

Serial via Airconsole

Protocol
Signview

Endpoint type
ComPort

Drop
80

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Parity

Figure 3.34: DMS read-only sign configuration with option to delete

Figure 3.35 shows the editable currently selected communication configuration. Clicking on the x icon will cancel any edits, and on the check mark icon will save any edits and return to the sign communication read-only screen. Clicking on the more options delete menu item will completely discard this communication configuration and return to the sign configuration edit screen.

Edit Sign Communication

Name
Serial via Airconsole

Protocol
Signview

Endpoint type
ComPort

Drop
80

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Figure 3.35: DMS edit sign communication

Figure 3.36 shows the completed selectable 170 sign configuration with at least one associated communication configuration.

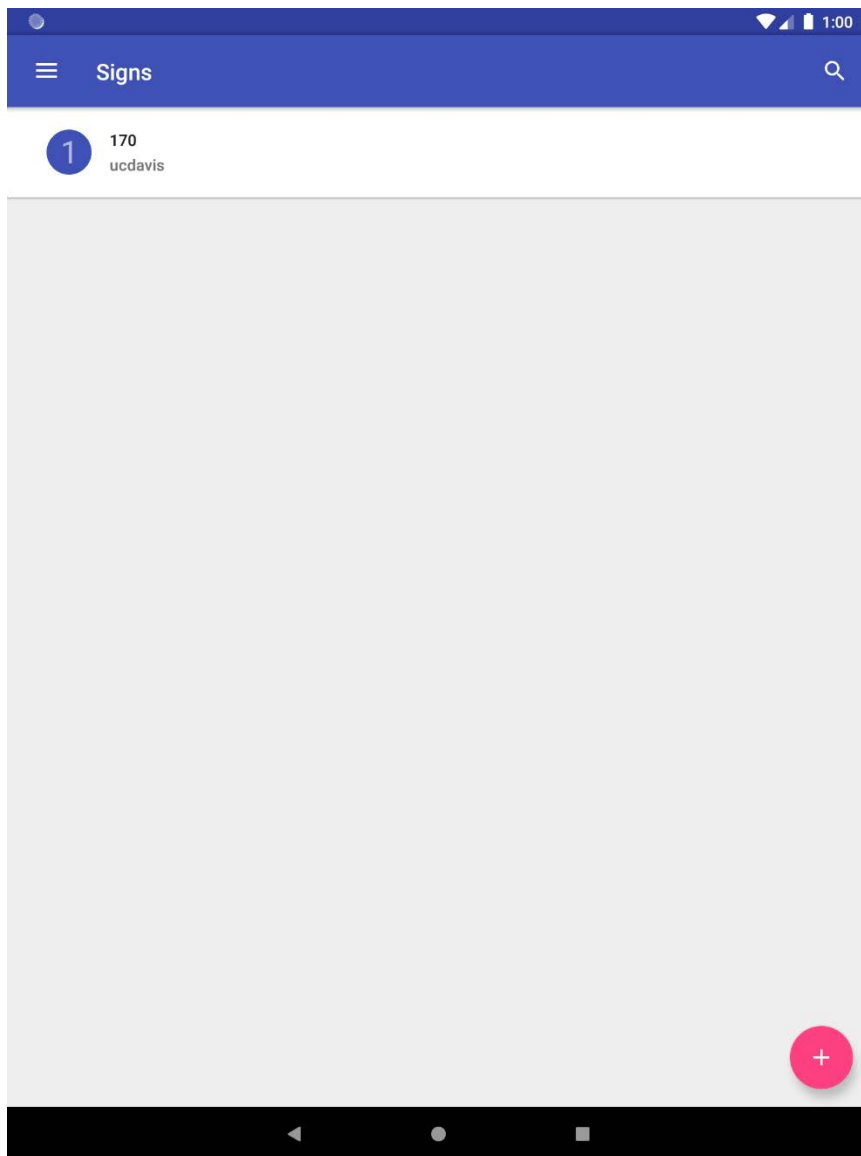


Figure 3.36: DMS 170 sign configuration

Figure 3.37 shows the read-only sign configuration when the 170 sign is selected from the signs screen shown above in Figure 3.35. Clicking on the pencil icon will open the edit sign configuration screen with this configuration.

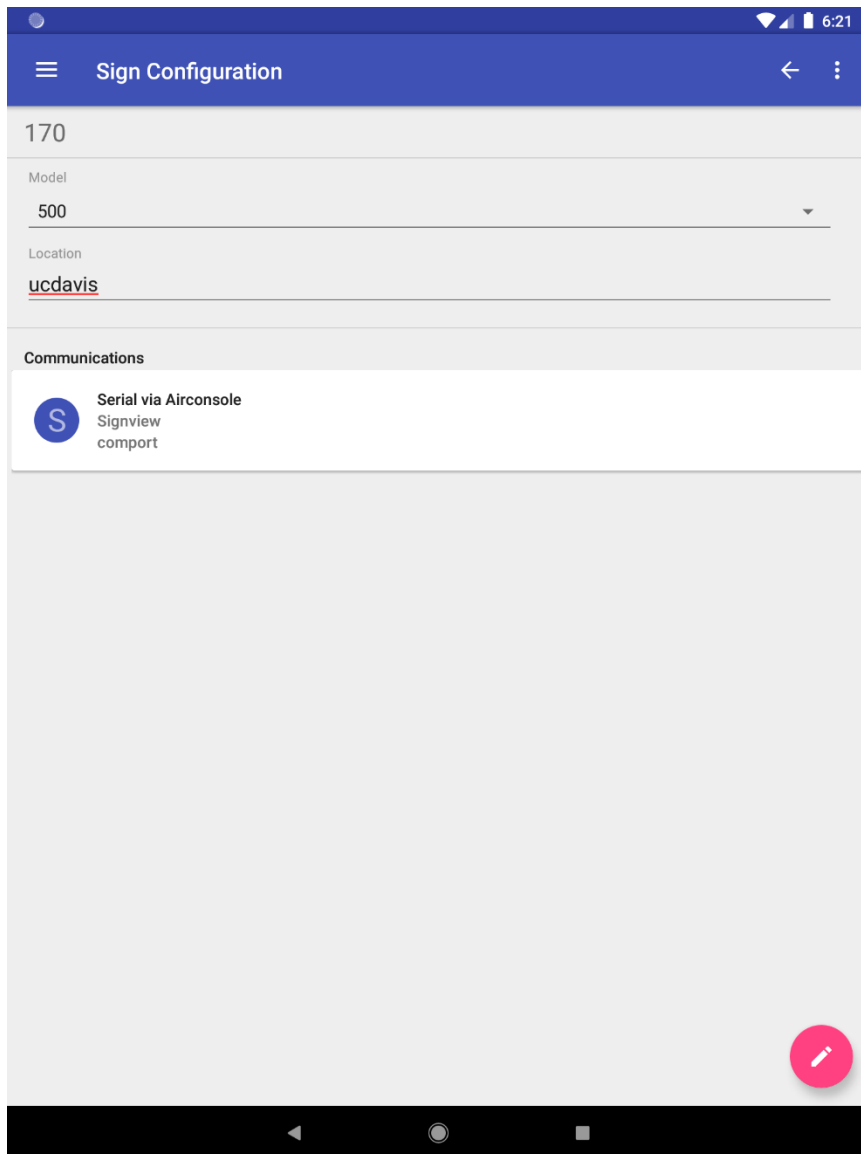


Figure 3.37: DMS sign configuration with option to edit

Figure 3.38 shows the editable currently selected sign configuration. Clicking on the x icon will cancel any edits, and on the check mark icon will save any edits and return to the sign configuration read-only screen. Clicking on the add icon will open the create communication configuration screen.


Edit Sign Configuration

Name
170

Model
500

Location
ucdavis

Communications

-  Serial via Airconsole
Signview
comport

+

Figure 3.38: DMS edit sign configuration

Figure 3.39 shows the editable currently selected sign configuration. Clicking on the delete menu item will completely discard this sign configuration and associated communication configuration and return to the sign configurations screen.

Edit Sign Configuration Delete

Name
170

Model
500

Location
ucdavis

Communications

- Serial via Airconsole
Signview
comport

+

Figure 3.39: DMS edit sign configuration with option to delete

Figure 3.40 shows the creation of a model 500 sign with a 2070 controller.

The screenshot shows a mobile application interface for creating a sign configuration. The title bar is blue with a hamburger menu icon on the left, the text 'Create Sign Configuration' in the center, and close and checkmark icons on the right. The status bar at the top shows signal strength, Wi-Fi, battery, and the time 1:01. The form has three input fields: 'Name' with the value '2070', 'Model' with a dropdown menu showing '500', and 'Location' with the value 'ucdavis'. Below these fields is a section titled 'Communications'. At the bottom right of the form area is a pink circular button with a white plus sign. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Figure 3.40: DMS 2070 create sign configuration

Figure 3.41 shows the selectable protocol options of SignView and NTCIP. The 2070 controller is running SignView and as such SignView is selected for the protocol option.

Create Sign Communication

Name
Ethernet via Airconsole

Protocol
Signview
Ntcip

Drop
81

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Figure 3.41: DMS create sign communication protocol options

Figure 3.42 shows the selectable endpoint type options of TCP, Telnet, and ComPort. The 2070 controller is interfaced over Ethernet and as such TCP is selected for the endpoint type.

Create Sign Communication

Name
Ethernet via Airconsole

Protocol
Signview

Endpoint type
Tcp
Telnet
ComPort

Host
192.168.10.1

Port
3696

Idle Close Timeout
0

Read Timeout
16000

Socket Connection Timeout
20000

Socket Read Timeout
20000

Acknowledgement Timeout
8000

Baud
1200

Data bits
8

Figure 3.42: DMS create sign communication endpoint type options

Figure 3.43 shows the configured parameters to communicate to the SignView 2070 over TCP via the Airconsole adapter.

The screenshot displays a mobile application interface for configuring a sign communication. The title bar at the top is blue and contains a hamburger menu icon, the text 'Create Sign Communication', and close and checkmark icons. The status bar at the very top shows signal strength, battery level, and the time 1:01. The configuration form consists of several labeled input fields:

- Name:** Ethernet via Airconsole
- Protocol:** Signview (selected from a dropdown menu)
- Endpoint type:** Tcp (selected from a dropdown menu)
- Drop:** 81
- Host:** 192.168.11.101
- Port:** 771
- Idle Close Timeout:** 0
- Read Timeout:** 16000
- Socket Connection Timeout:** 20000
- Socket Read Timeout:** 20000

The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Figure 3.43: DMS 2070 sign communication configuration

Figure 3.44 shows the configured 2070 sign with a single associated TCP via Airconsole communications configuration.

The screenshot displays a mobile application interface for creating a sign configuration. The title bar at the top is blue and contains a hamburger menu icon, the text 'Create Sign Configuration', and close/confirm icons. The main form area has a light gray background and contains three input fields: 'Name' with the value '2070', 'Model' with the value '500', and 'Location' with the value 'ucdavis'. Below these fields is a section titled 'Communications' which contains a single configuration entry. This entry is represented by a blue circle with a white 'E' icon, followed by the text 'Ethernet via Airconsole', 'Signview', and 'Tcp'. At the bottom right of the form area is a pink circular button with a white plus sign. The bottom of the screen shows a black Android navigation bar.

Figure 3.44: DMS 2070 sign configuration with a communications configuration

Figure 3.45 shows completed configurations for a 170 sign communicating over a serial connection, and a 2070 sign communicating over Ethernet.

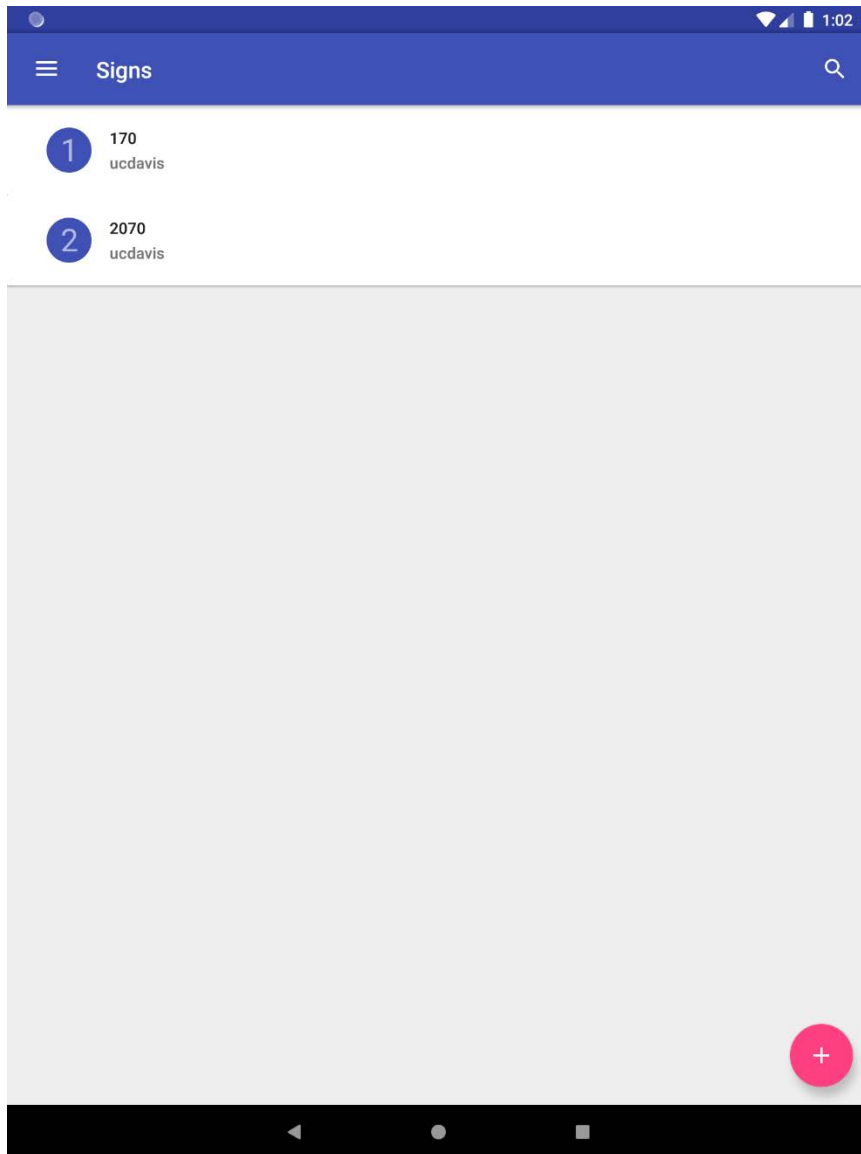


Figure 3.45: DMS with 170 and 2070 sign configurations

Figure 3.46 shows the configured 170 sign configuration when selected from the signs screen.

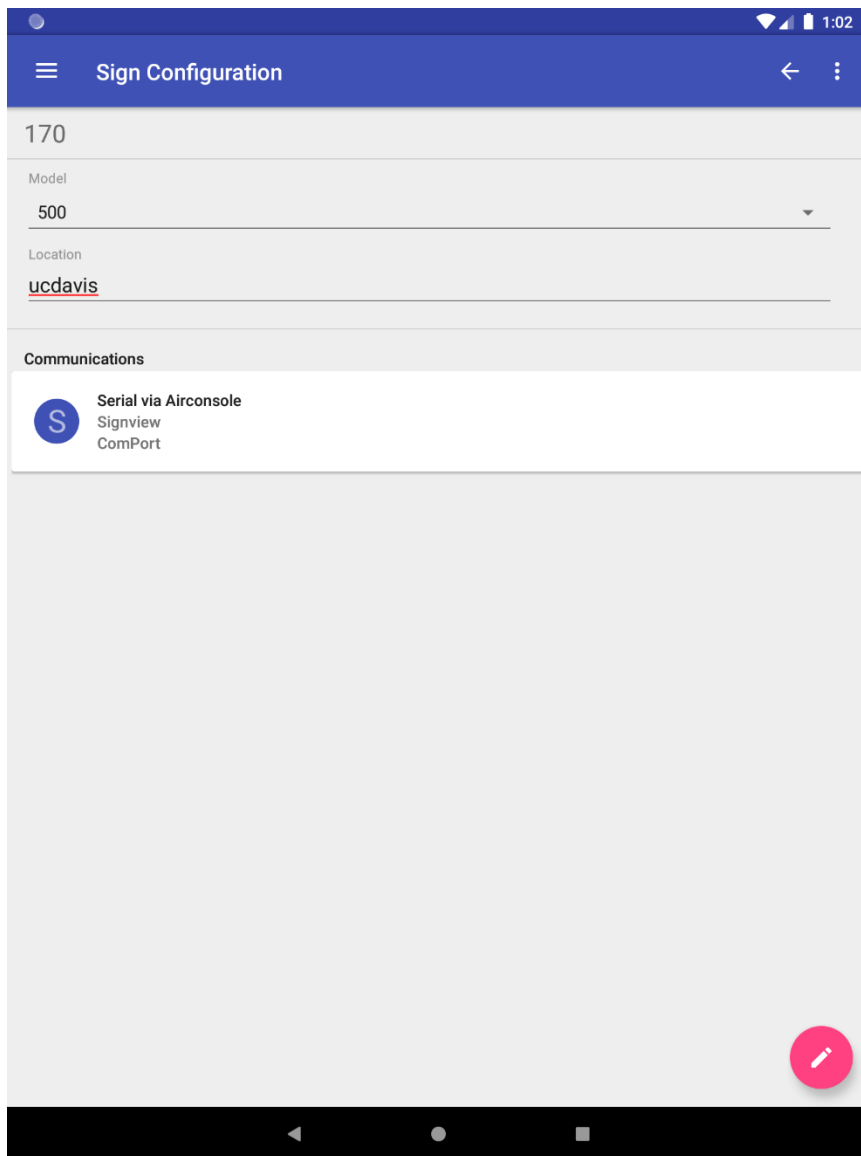


Figure 3.46: Selection of the 170 sign configuration

Figure 3.47 shows result of clicking on a communication configuration from a read-only sign configuration screen as seen in Figure 3.45. The basic sign screen is composed of a sign rendering and a message builder. The image view is composed of a draft view and a current view. The message builder supports two pages, three lines per page, with selectable fonts per line.

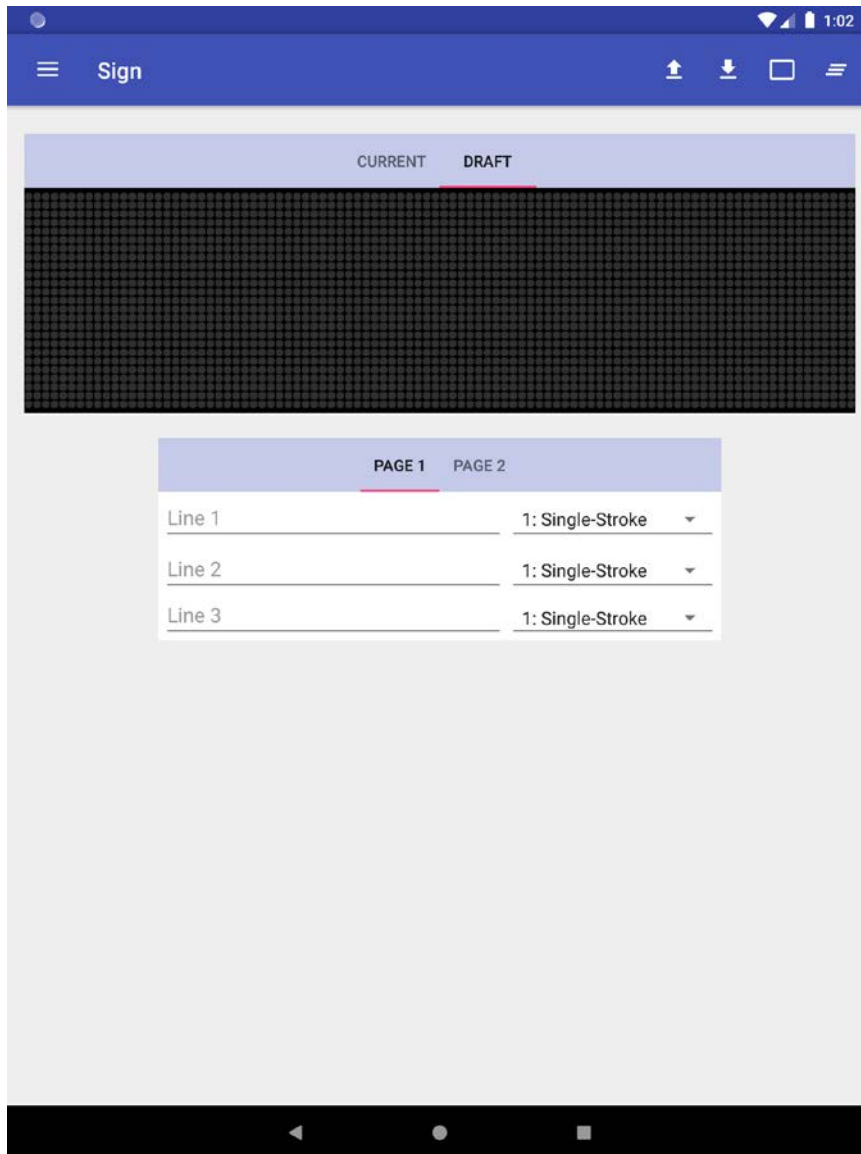


Figure 3.47: DMS sign configured with the 170 sign and serial communications configurations

Figure 3.48 shows the message builder contents rendered in real-time and displayed in the draft image tab. The draft and current tabs are clickable and swipeable.

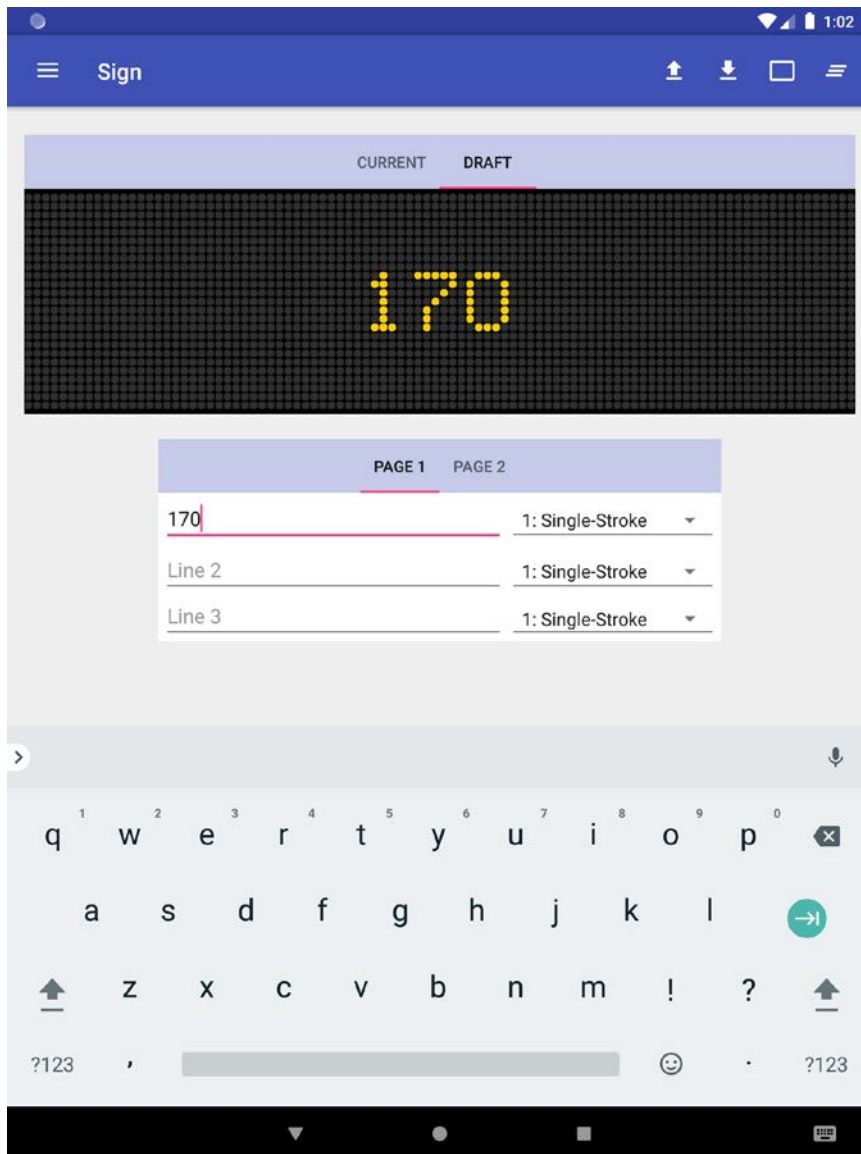


Figure 3.48: DMS sign in draft message mode

Figure 3.49 shows the message continuing to be entered with an additional line of text and the selection of an alternate font.

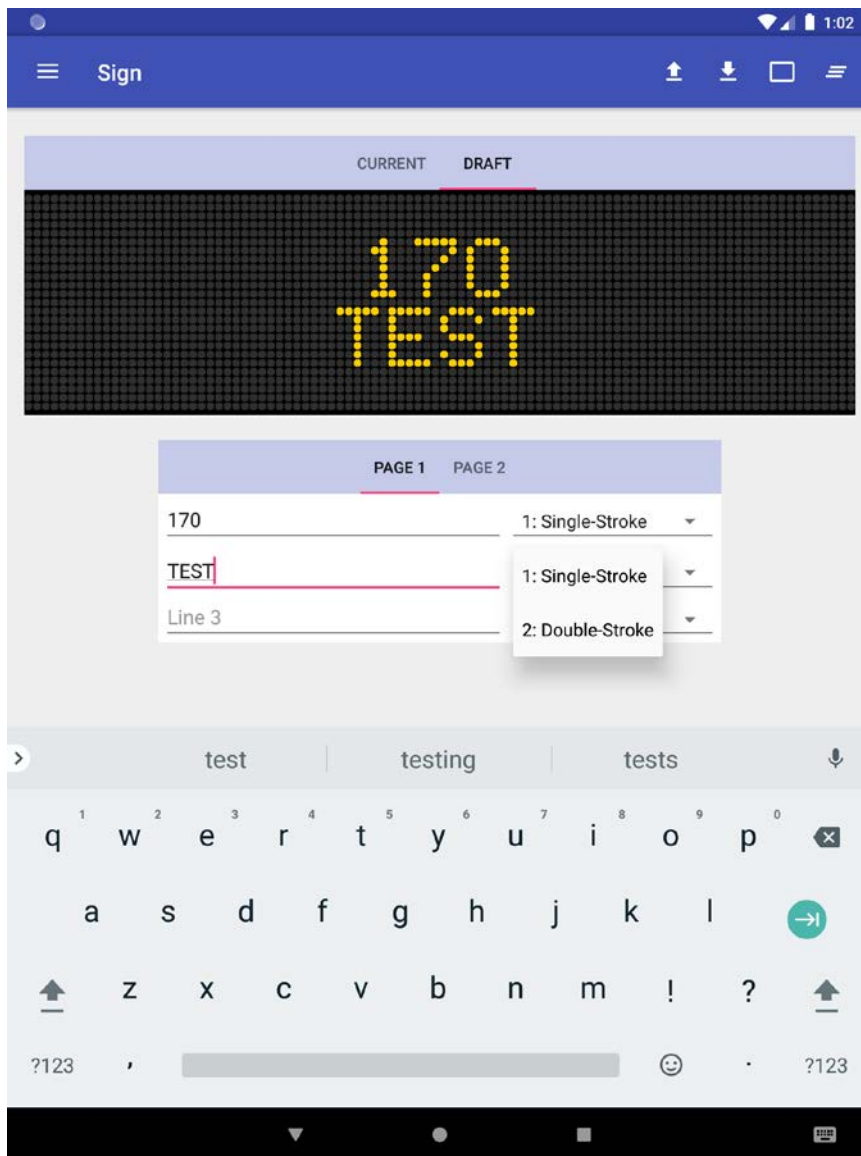


Figure 3.49: DMS sign draft mode font options

Figure 3.50 shows the immediate rendering of the draft message following the selection of the double-stroke font in line two. When the draft message and draft image are complete the contents can be sent to the sign. The up-arrow icon sends the current draft message to the sign and will appear on the sign exactly as rendered on screen.

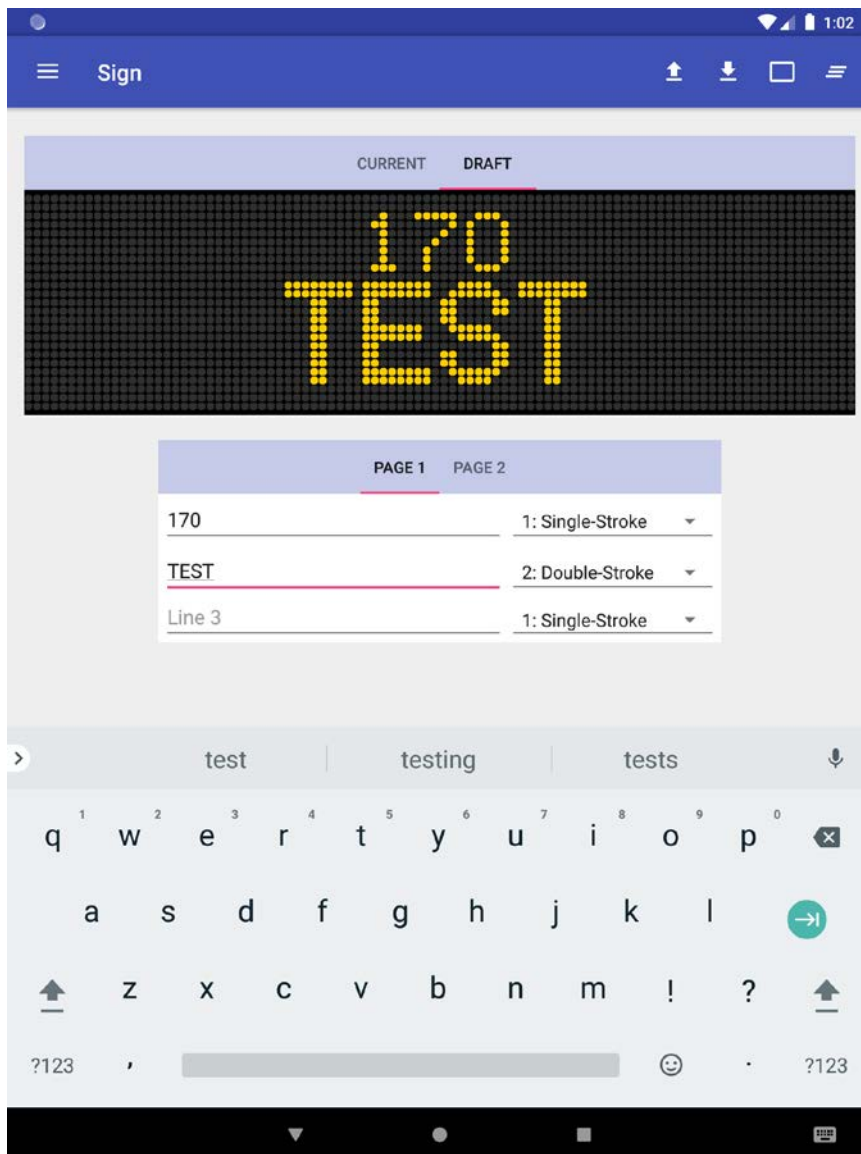


Figure 3.50: DMS sign draft mode displaying multi-font rendering

Figure 3.51 shows the send message operation in progress. During this process the message is sent to the sign, and then the contents of the sign are requested for verification purposes.

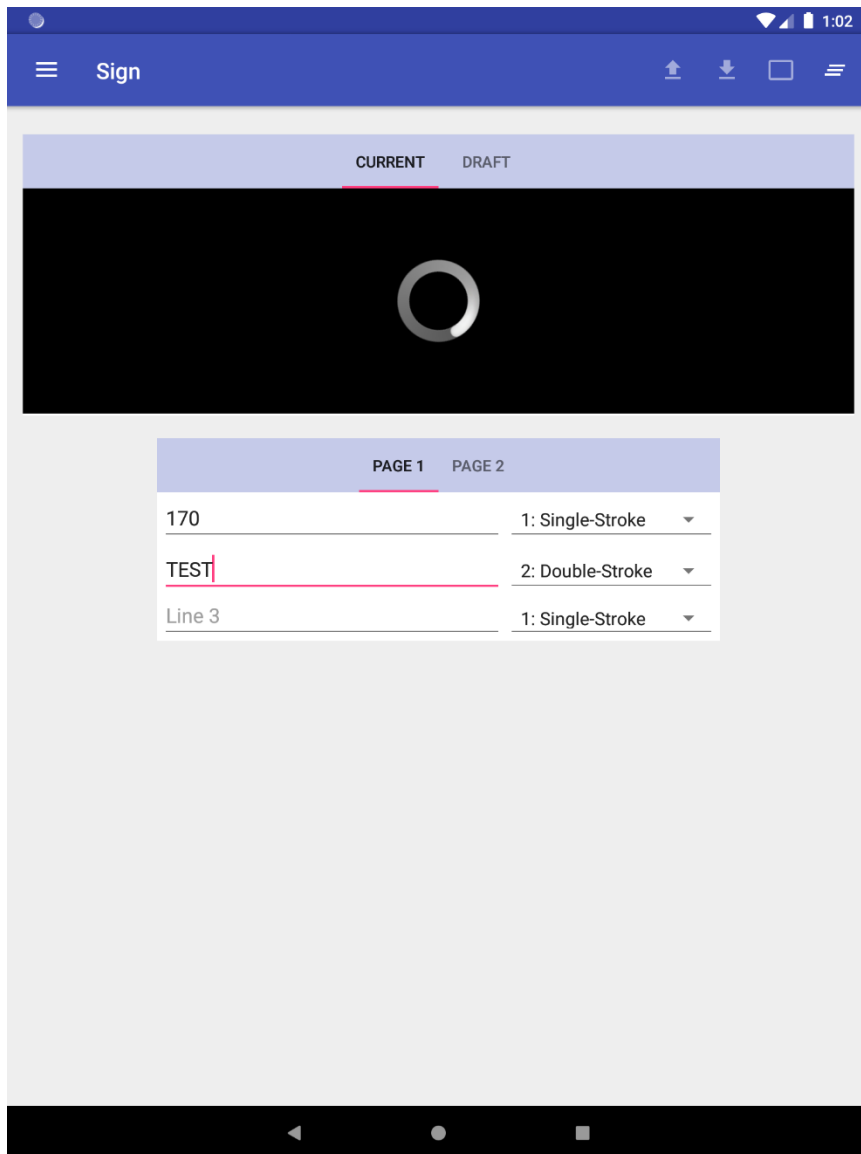


Figure 3.51: DMS sign sending draft configuration to the 170 controller

Figure 3.52 shows the verified current image displayed on the 170 sign. Clicking on the down-arrow icon will request the posted sign image and display it in the current image tab. Clicking on the empty sign icon will send a blank sign command, and clicking on the three offset horizontal lines icon will clear the message builder.

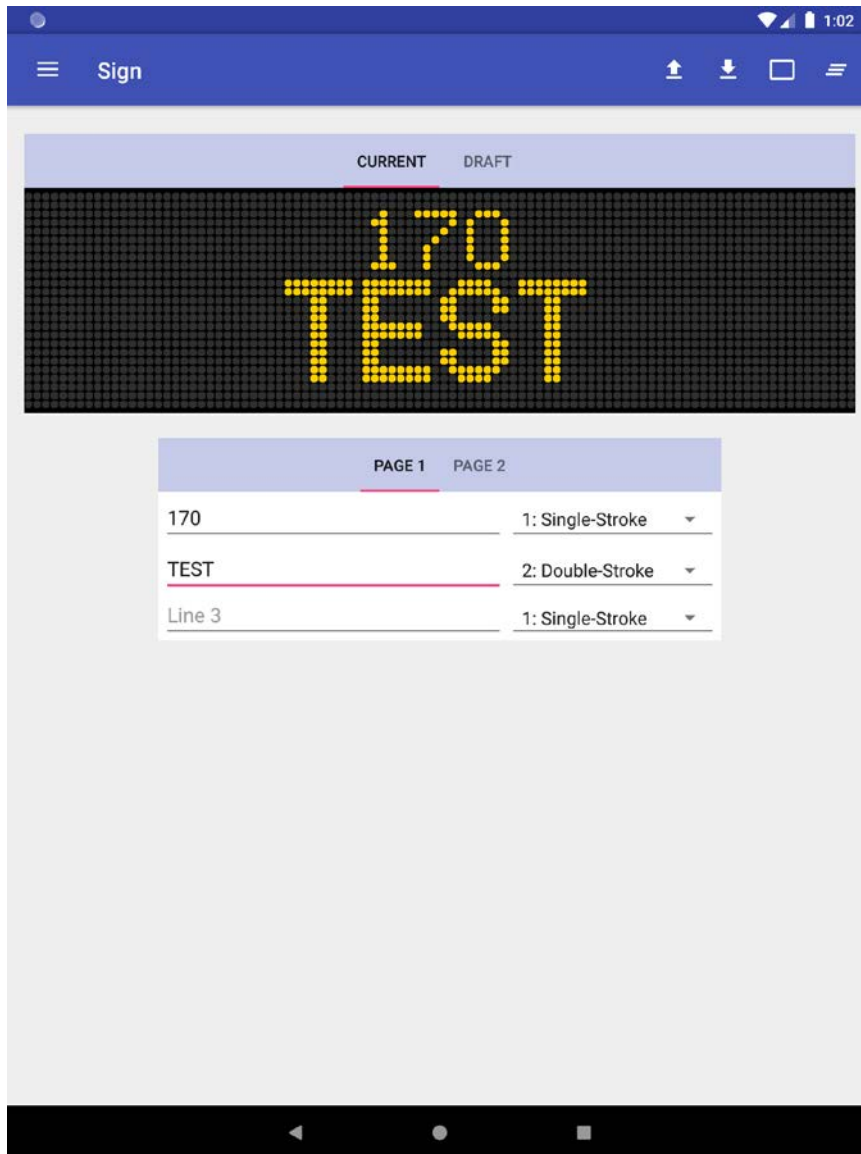


Figure 3.52: DMS sign displaying the content currently stored on the 170 controller

Figure 3.53 shows the configured 2070 sign configuration when selected from the signs screen.

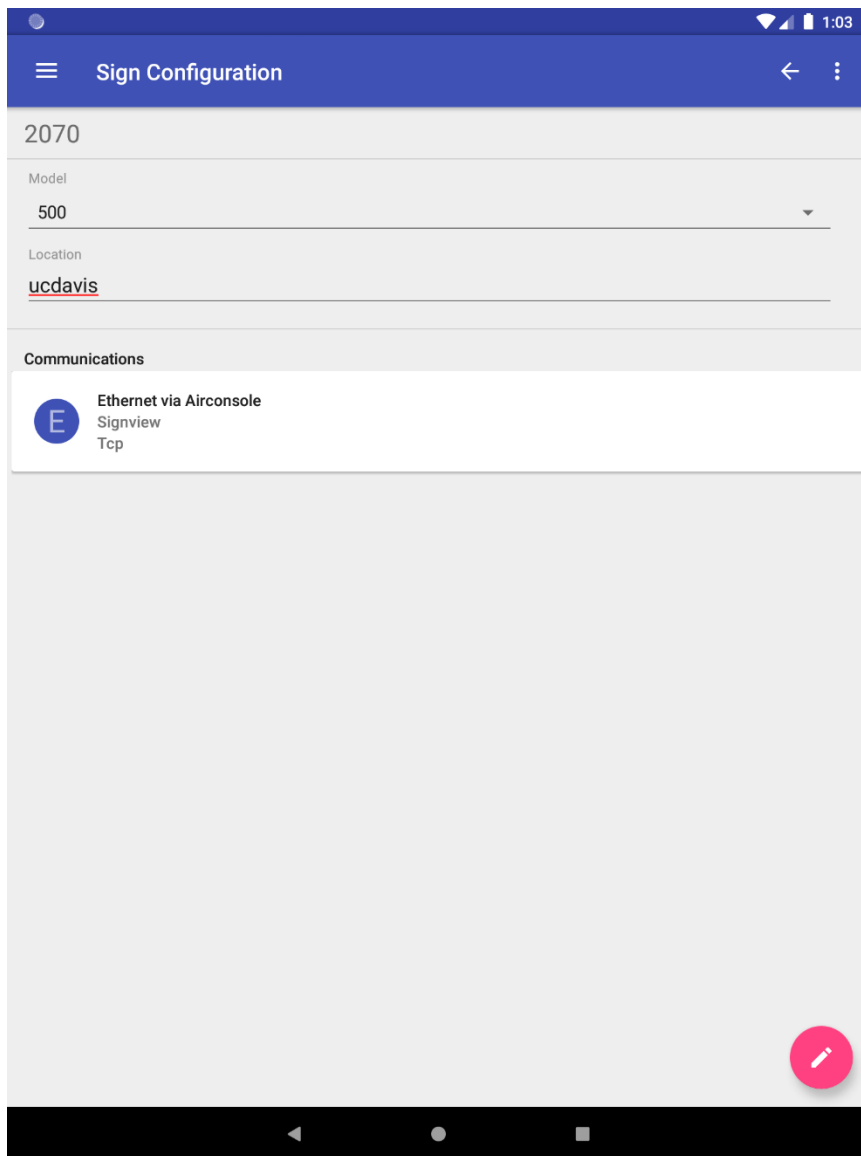


Figure 3.53: Selection of the 2070 sign configuration

Figure 3.54 shows the verified current image displayed on the 2070 sign.

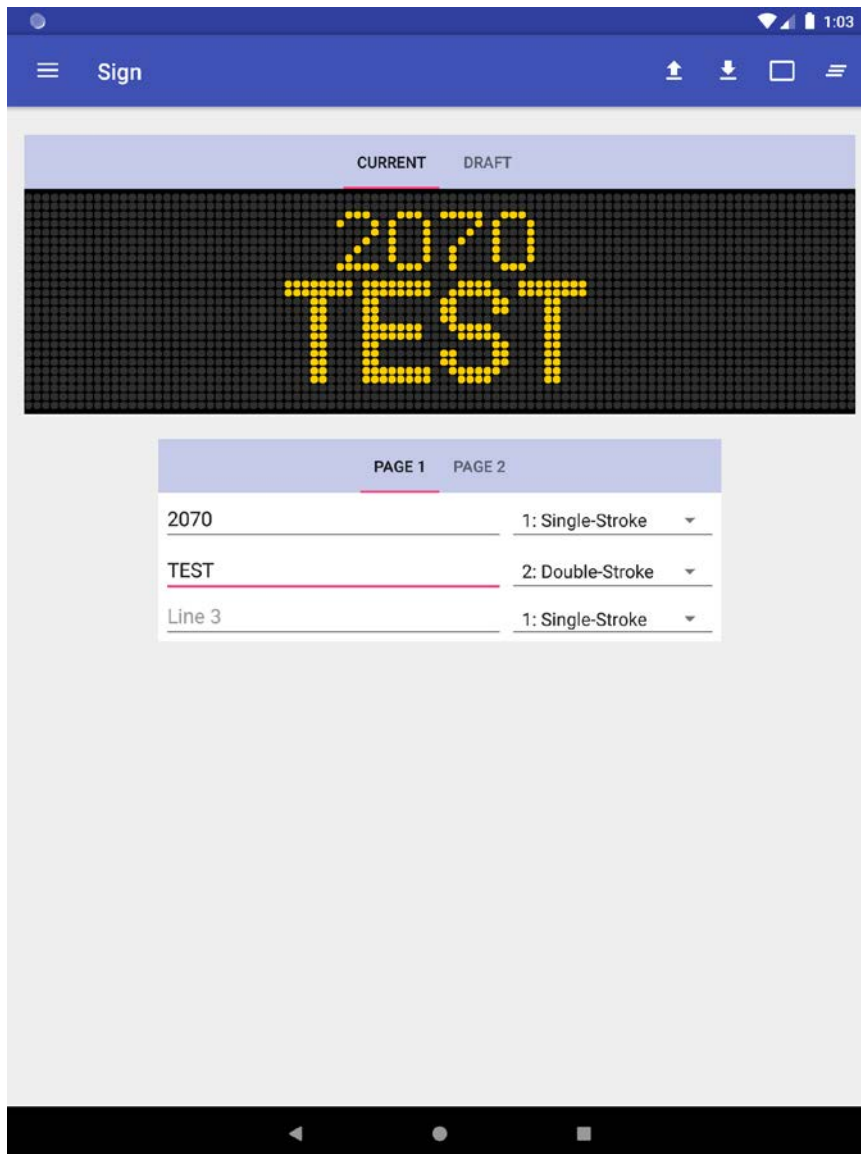


Figure 3.54: DMS sign displaying the content currently stored on the 2070 controller

Figure 3.55 shows an image taken by the built-in camera functionality. The camera screen is displayed when selected from the main menu.

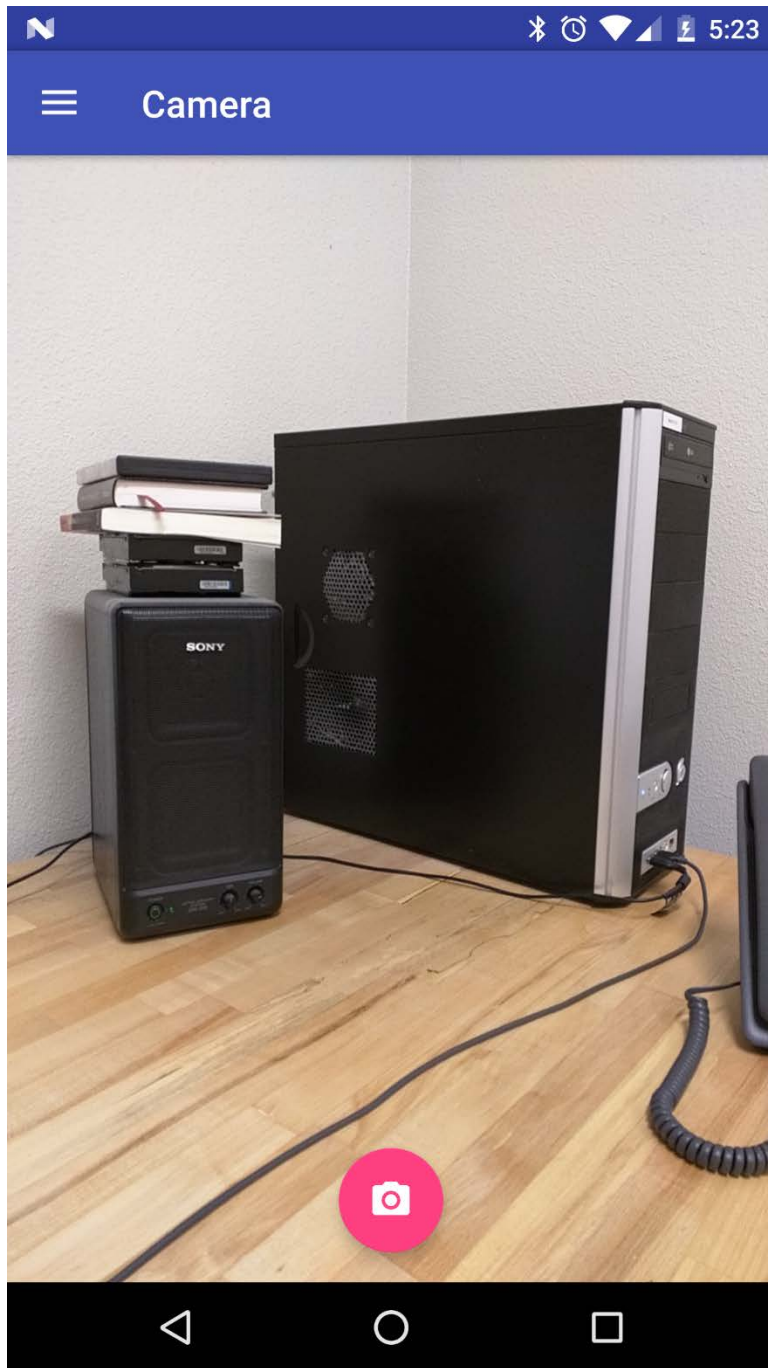


Figure 3.55: Sample DMS camera image in lab test

Figure 3.56 shows the gallery composed of camera images. The gallery is displayed when selected from the main menu.

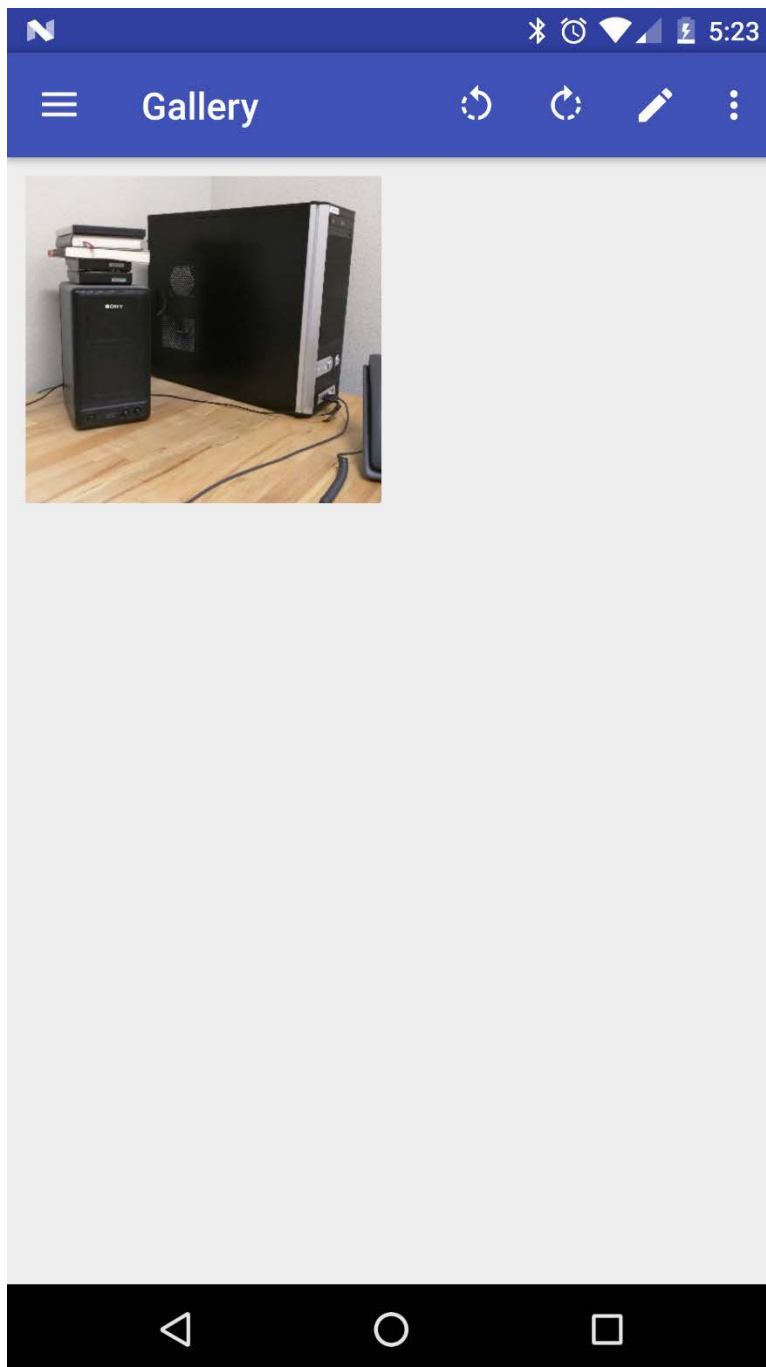


Figure 3.56: DMS camera image gallery

Figure 3.57 shows a selected gallery image. Selection of one or more images is performed by long-pressing on images of interest. Once selected, the various menu items can be used to perform operations. Clicking on the appropriate circular-arrow icons will rotate the image left or right by 90 degrees. Clicking on the more options delete menu item will remove all selected images. Clicking on the pencil icon will open the annotation screen.

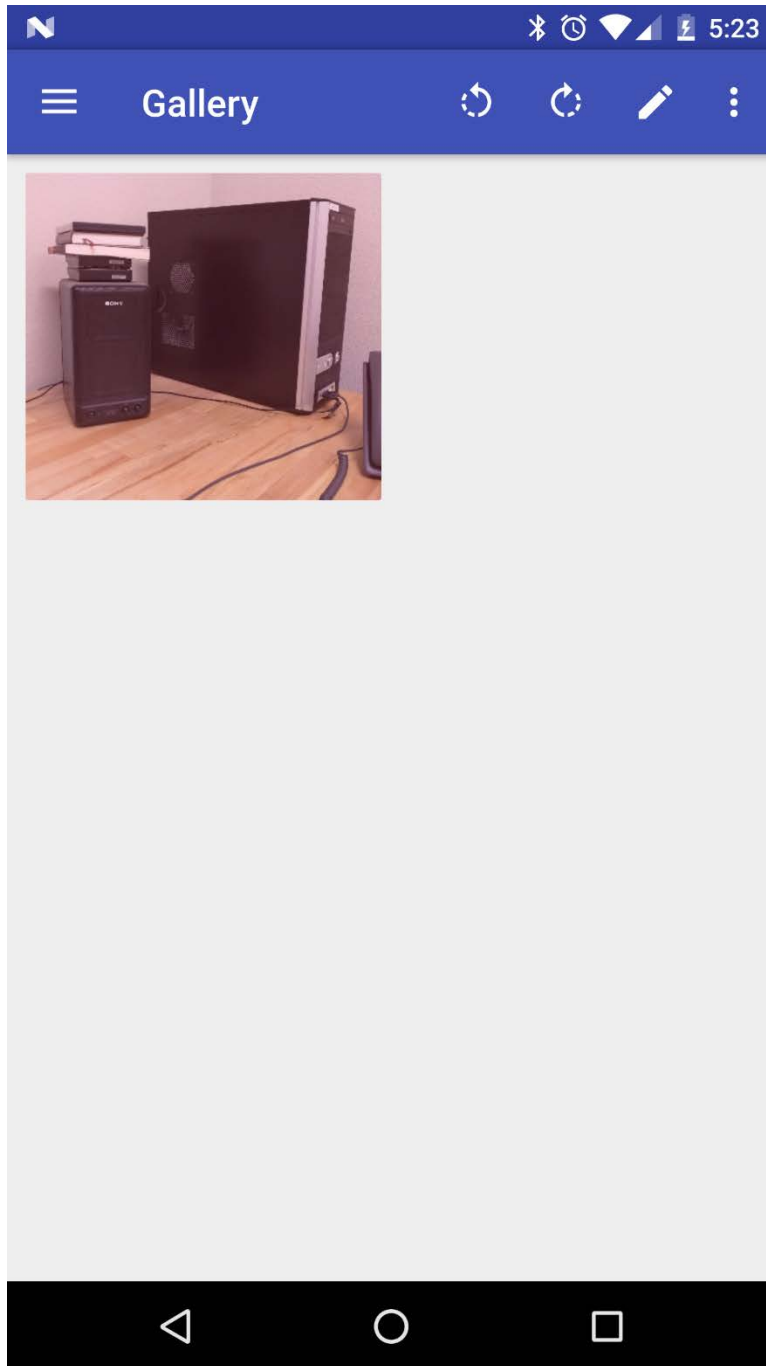


Figure 3.57: DMS camera gallery with image selected

Figure 3.58 shows the annotation screen with three primary modes: draw, text, and zoom/crop. Clicking on the pencil icon will set the draw mode, on the A icon will set the insert text mode, and on the arrows icon will set the zoom/crop mode. In the draw mode, freestyle drawings can be made over the top of the selected image. The more option menu items includes color, line width, font size, undo, and delete. Clicking on the color menu option allows for the selection of the drawing or text color.

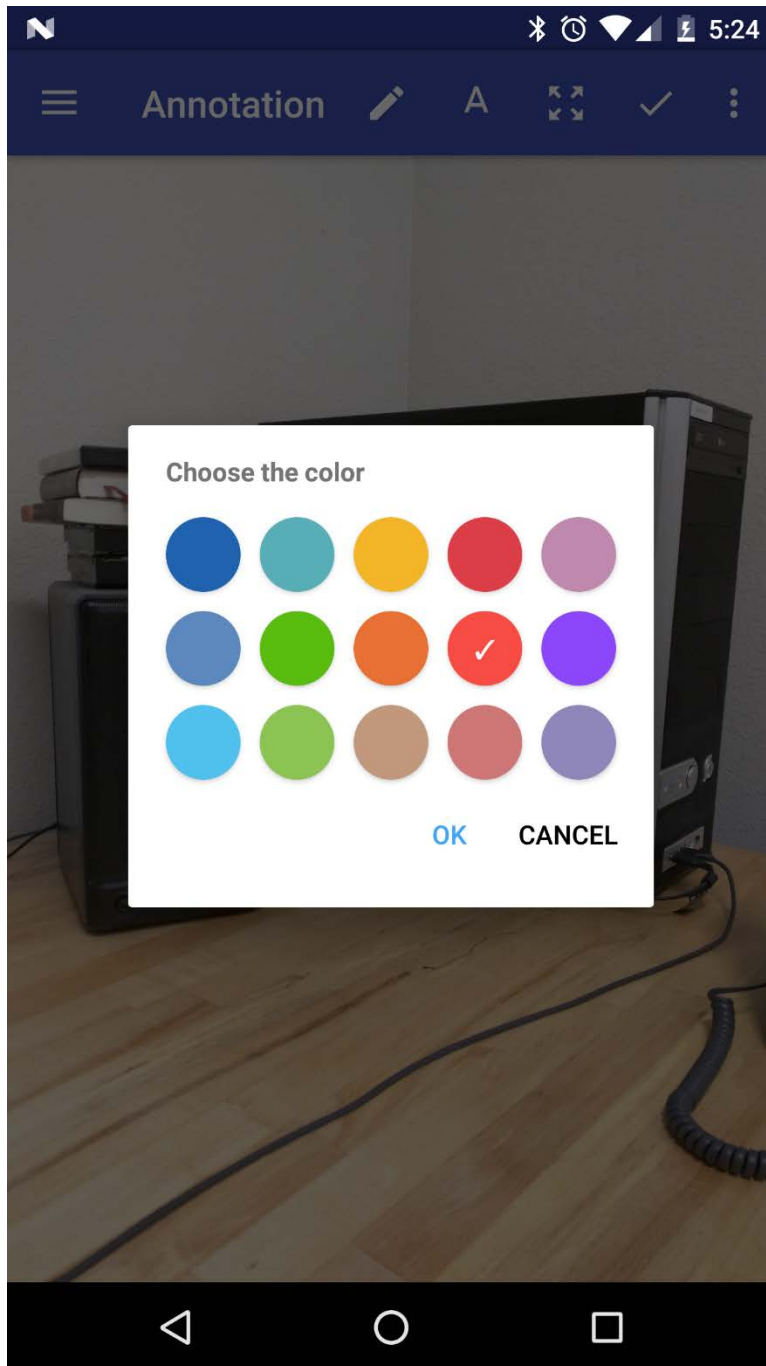


Figure 3.58: DMS image annotation color options

Figure 3.59 shows the result of clicking on the line width menu item.

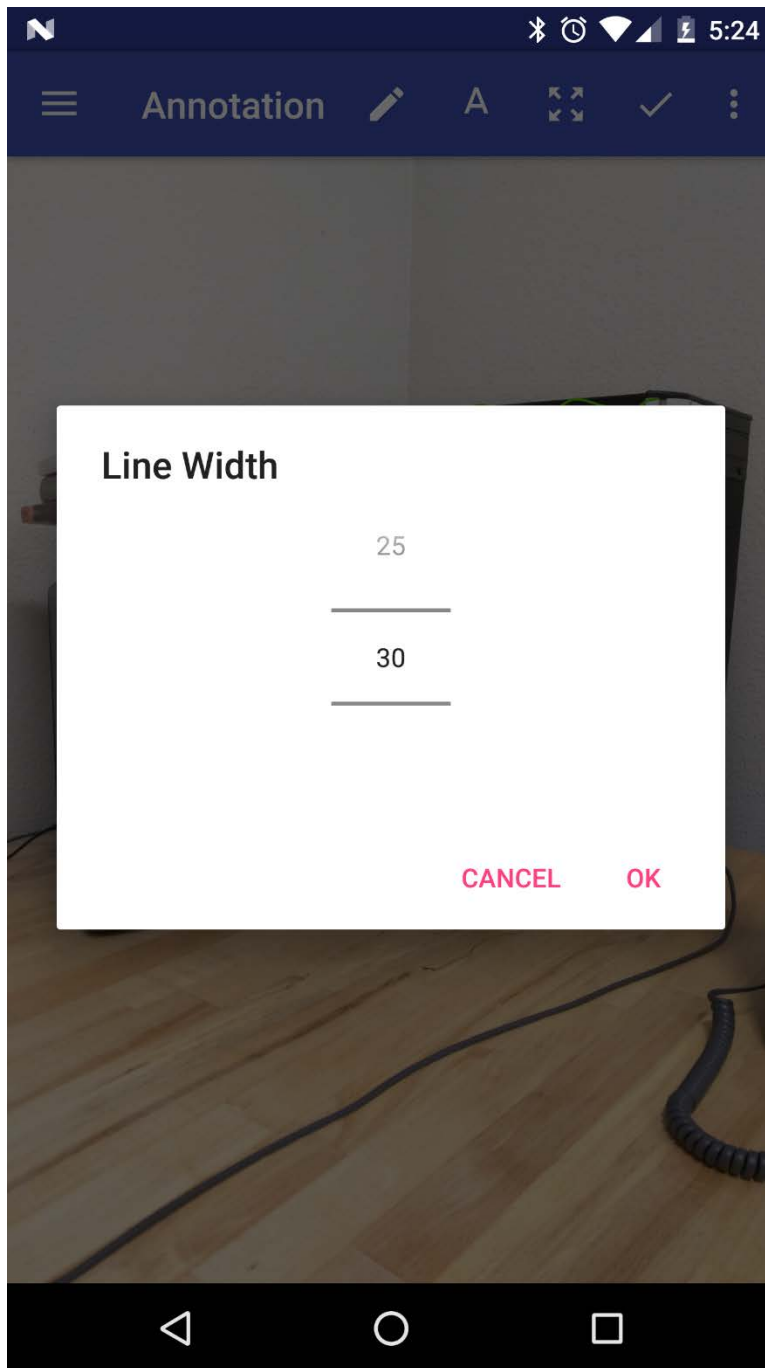


Figure 3.59: DMS camera image annotation line width selection

Figure 3.60 shows the result of several drawings of various colors and line widths, as well as insertion of text. Selection of the checkmark icon will save the annotated image in the gallery alongside the original unannotated image.

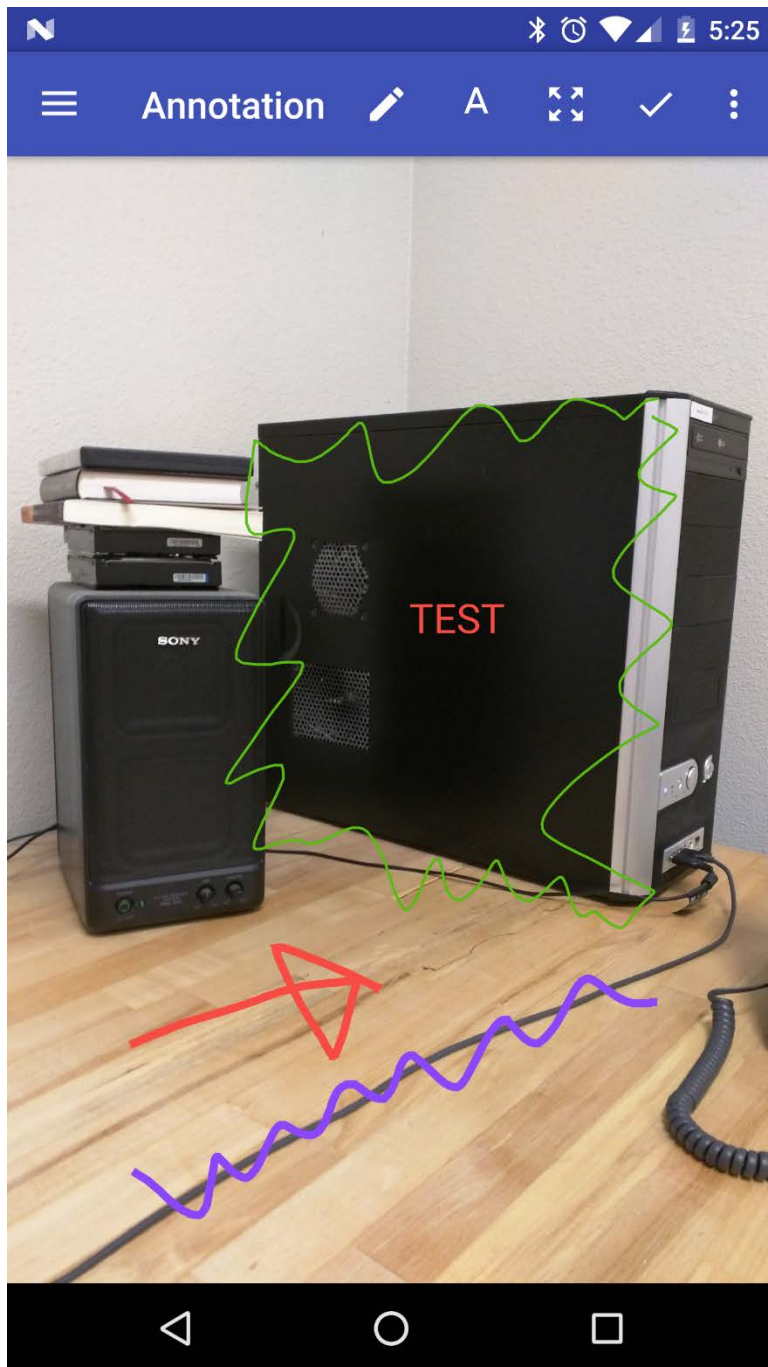


Figure 3.60: Annotated DMS camera image

Figure 3.61 shows the original and annotated images.

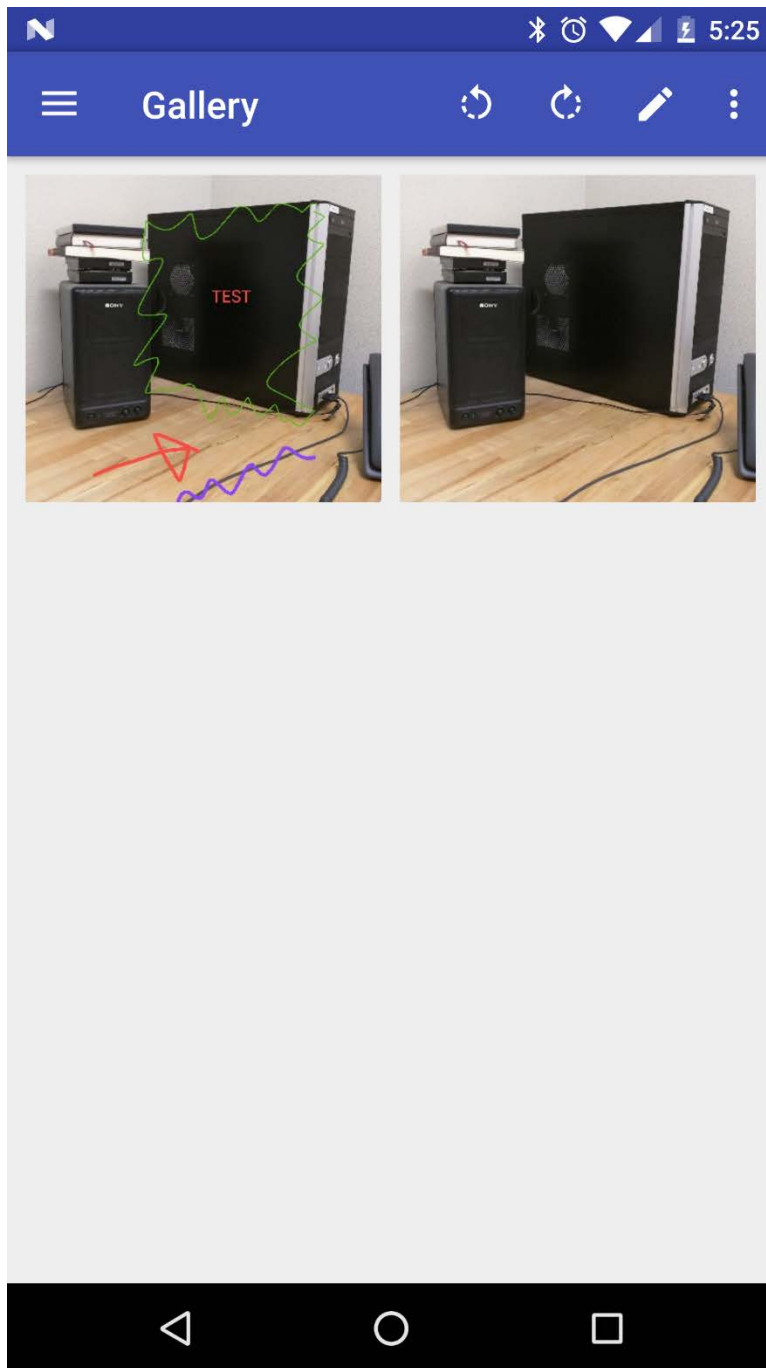


Figure 3.61: DMS camera image gallery including annotated and unmodified original images

Figure 3.62 shows the various app settings currently set to default values. The pixel rendering supports circle and square options. The input filters support converting all text to uppercase, and the option of displaying and selecting a substitute character for invalid message input characters.

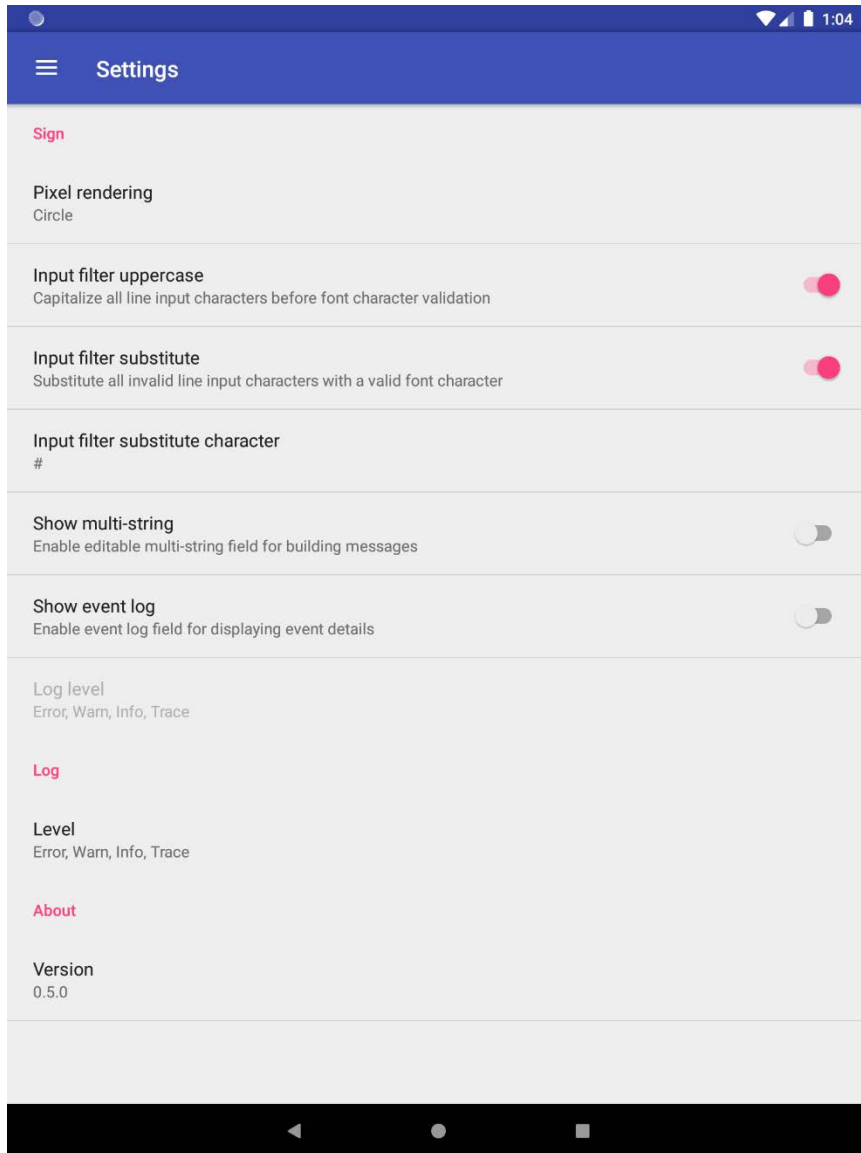


Figure 3.62: DMS settings default configuration

Figure 3.63 shows enabling the editable multi-string field for building messages.

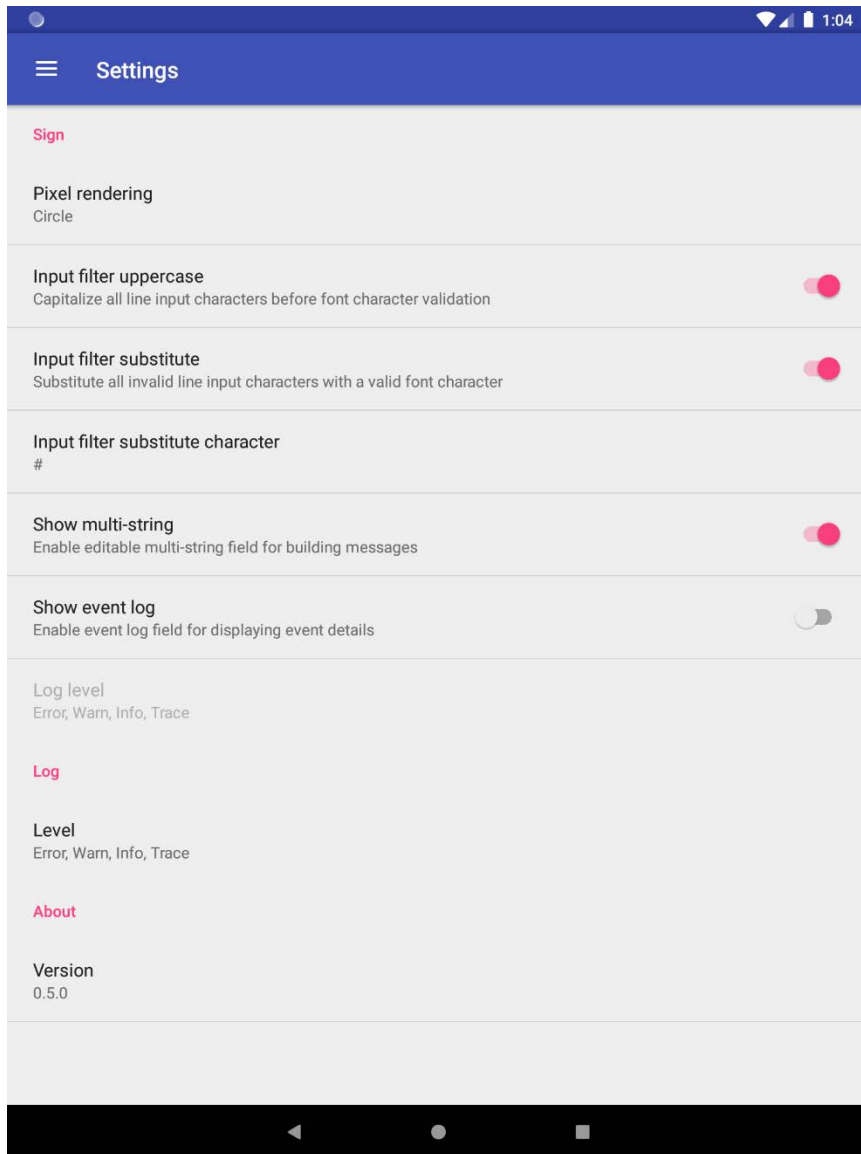


Figure 3.63: DMS settings with show multi-string enabled

Figure 3.64 shows the enabled multi-string field with the standards representation of the text from the message builder. Multi-string is the primary input to the NTCIP signs and is include for direct support of that standard. However even in that case the multi-string is built directly from the message builder, and is not necessary to operate a standard two-page three-line sign. However, the NTCIP standard supports a rich set of multi-string standards and the ability to manipulate the input is provided through this option.

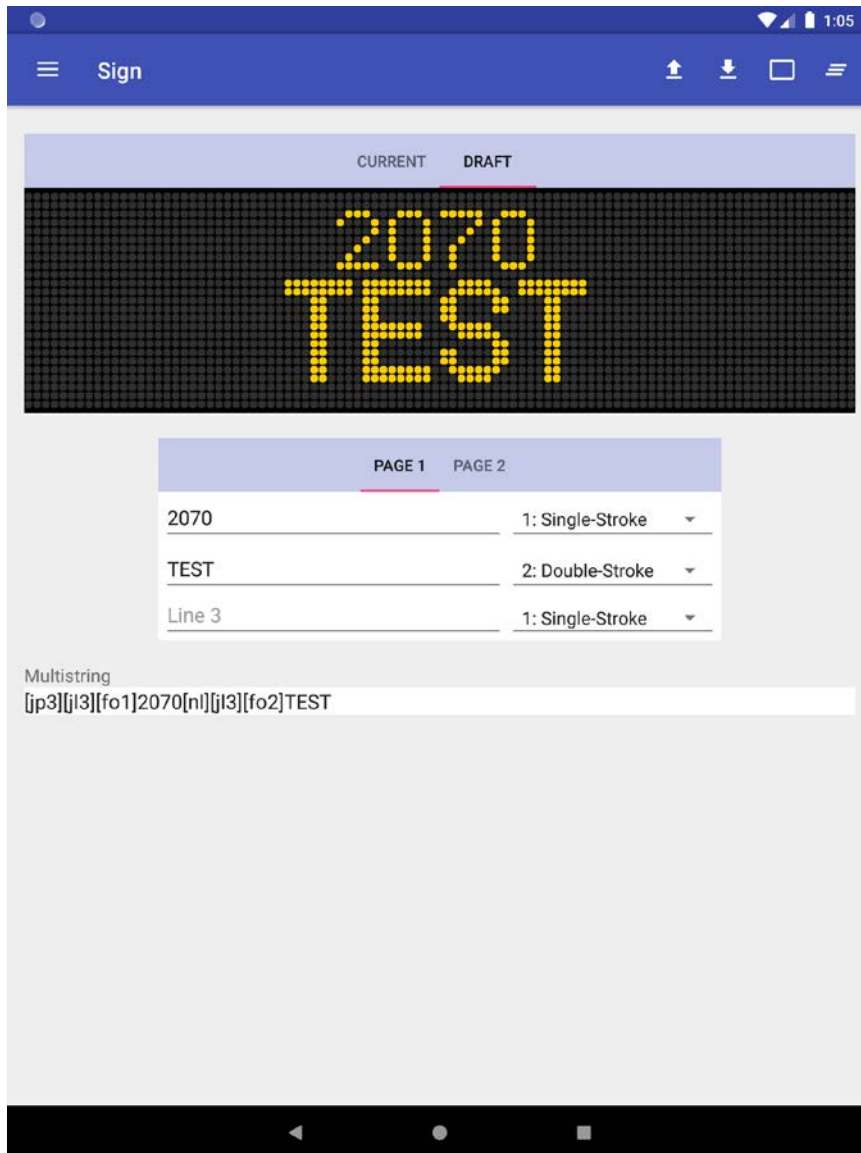


Figure 3.64: DMS sign with show multi-string enabled

Figure 3.65 shows enabling the event log field for displaying event details in the sign screen.

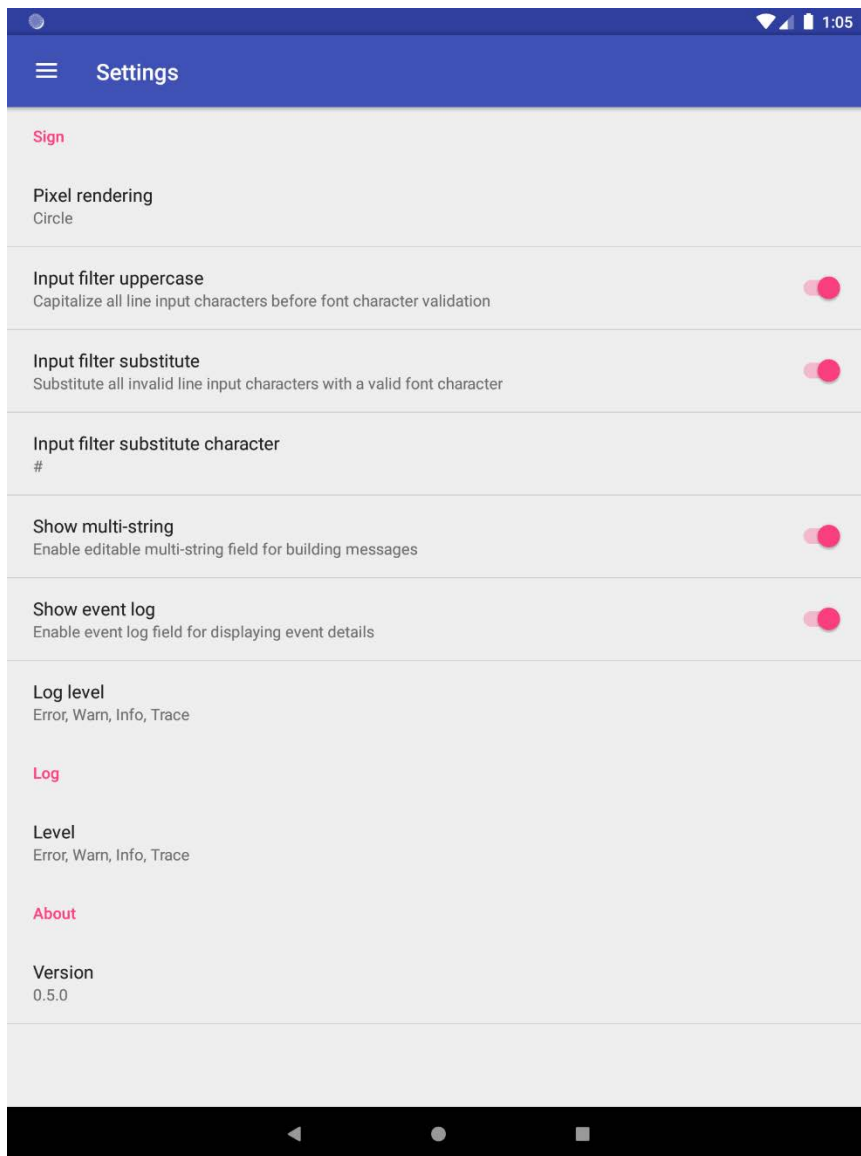


Figure 3.65: DMS settings with show event log enabled

Figure 3.66 shows the log level selections for messages to include in the sign event log.

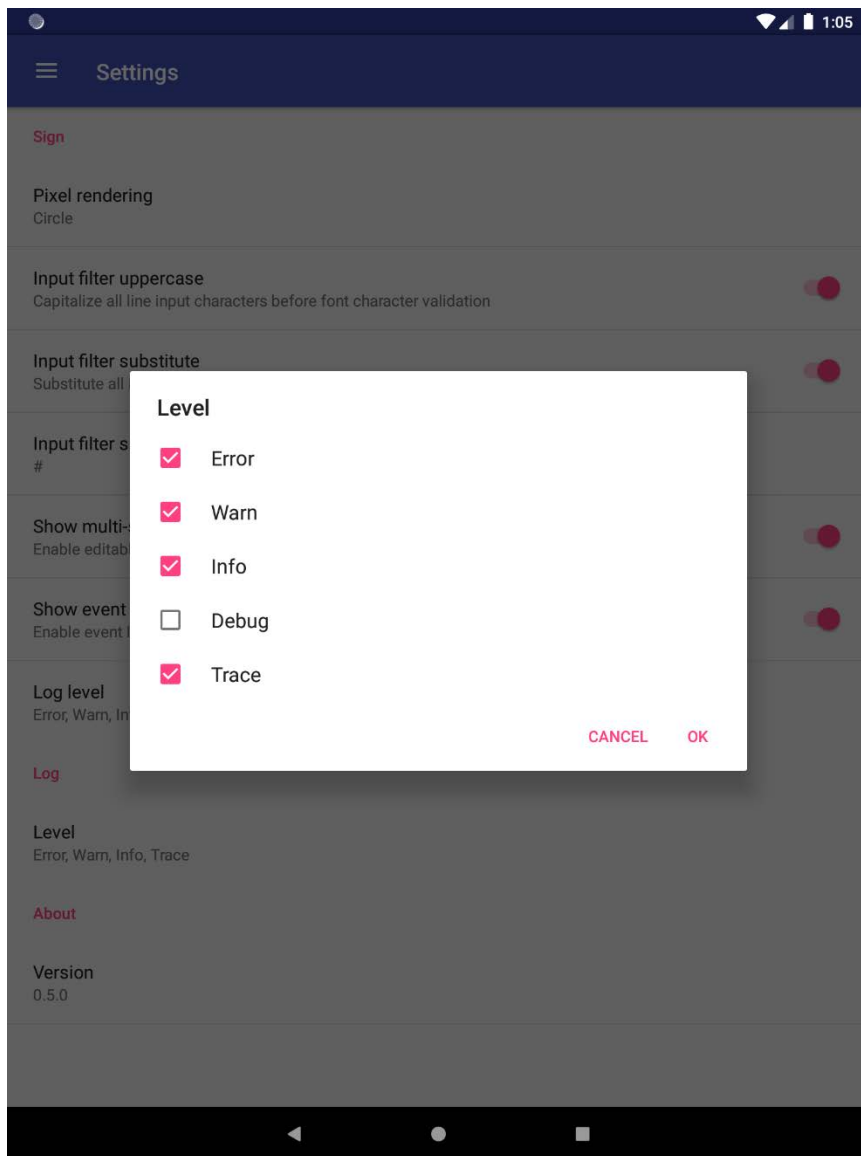


Figure 3.66: DMS settings event log level configuration

Figure 3.67 shows the event log displaying the details of the message communications between the software and the on-site sign controller. For technical end users the event log levels could be set to include the trace level as shown. However, for operations users this event log would be configured to only display errors, warnings, and high-level information.

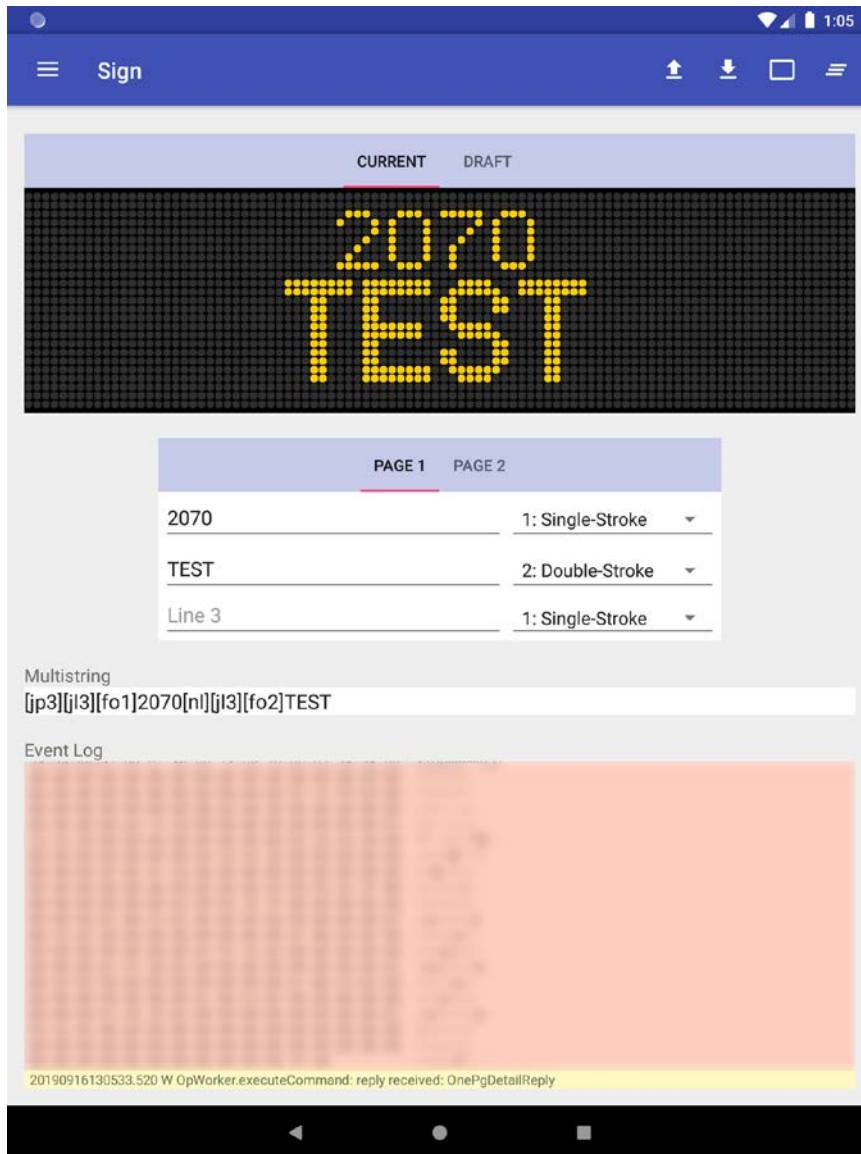


Figure 3.67: DMS sign with show event log enabled

Figure 3.68 shows the log level selection options. Typically, all but the debug levels will be selected for technical level controller communications debugging.

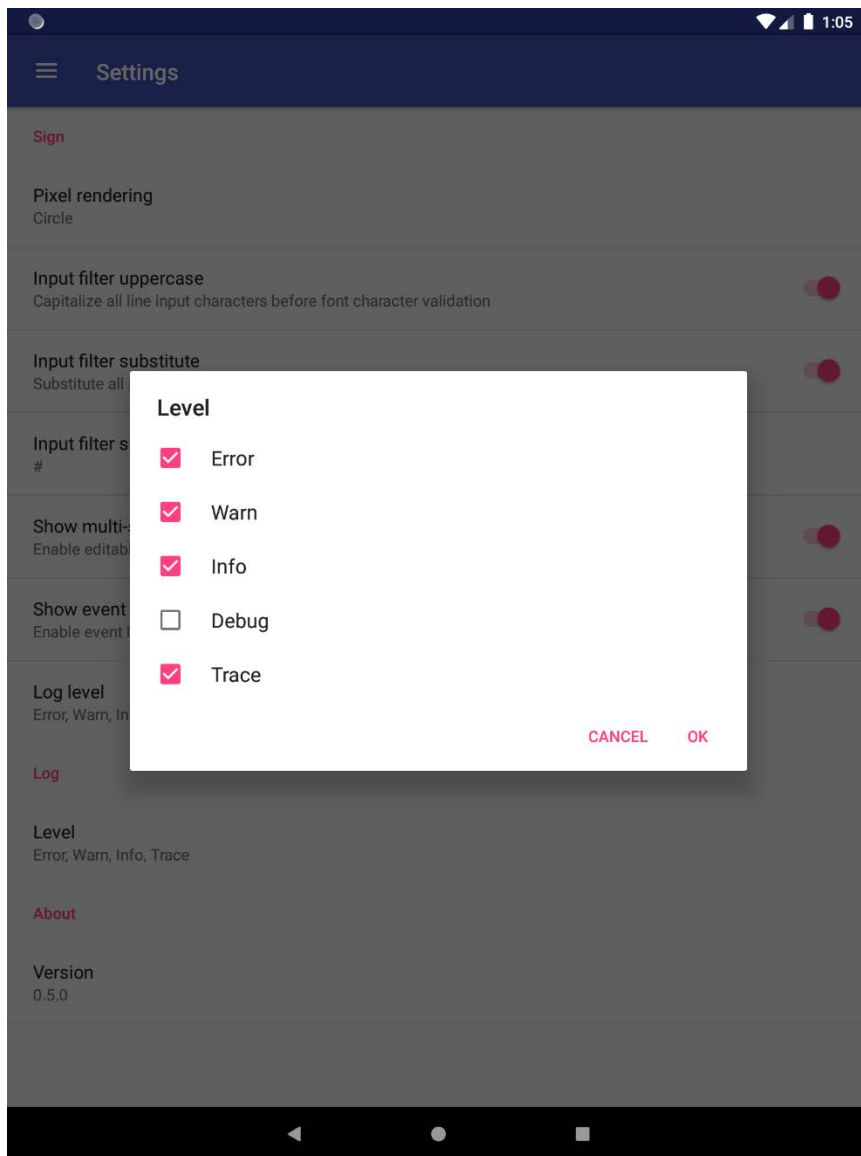


Figure 3.68: DMS settings log level configuration

Figure 3.69 shows the 2070 sign following a detail request.

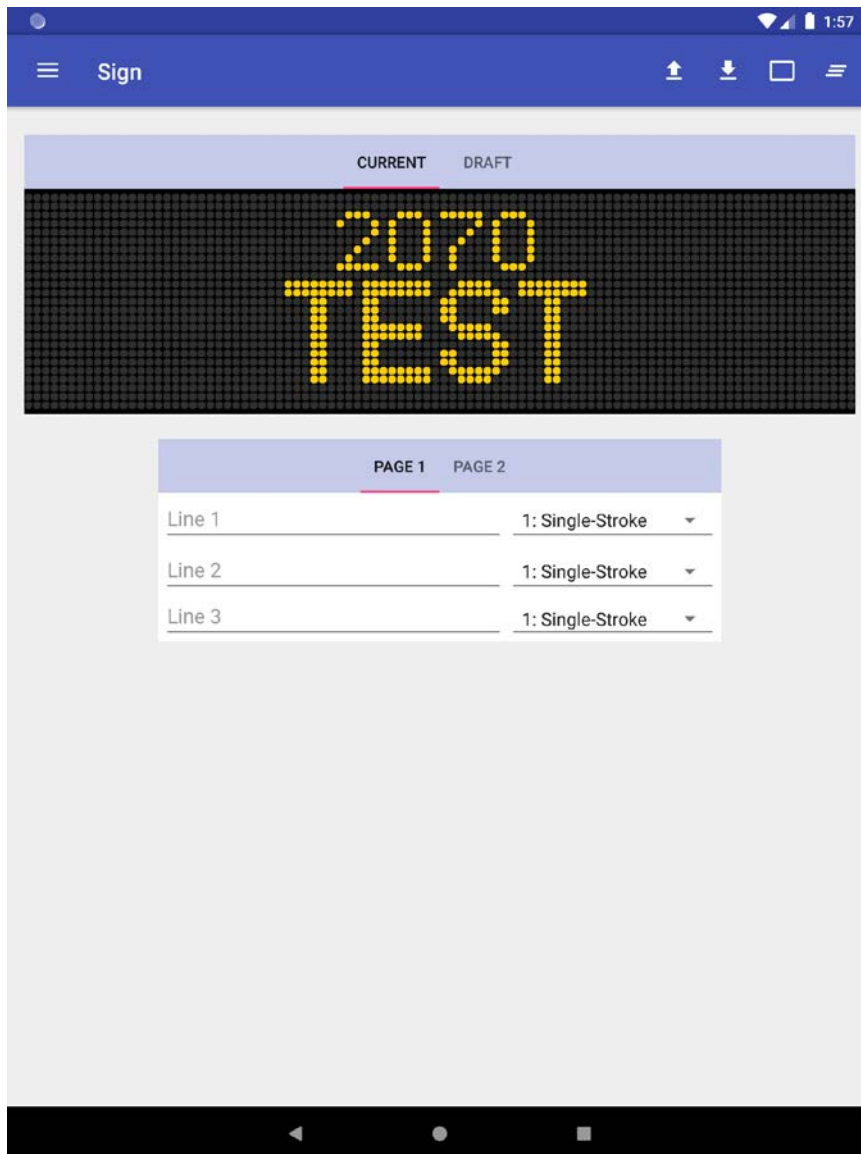


Figure 3.69: DMS sign following a detail request

Figure 3.70 shows the log configured with error, warn, info, and trace level options. The log levels are color coded. The log entries are currently configured to begin with a timestamp, followed by the first letter of the log level, class, method, and message. The trace level log entries are typically followed by byte level send and receive data. The log shown is the result of a detail request from the 2070 controller. The log view is scrollable and the newest entries are posted at the bottom.

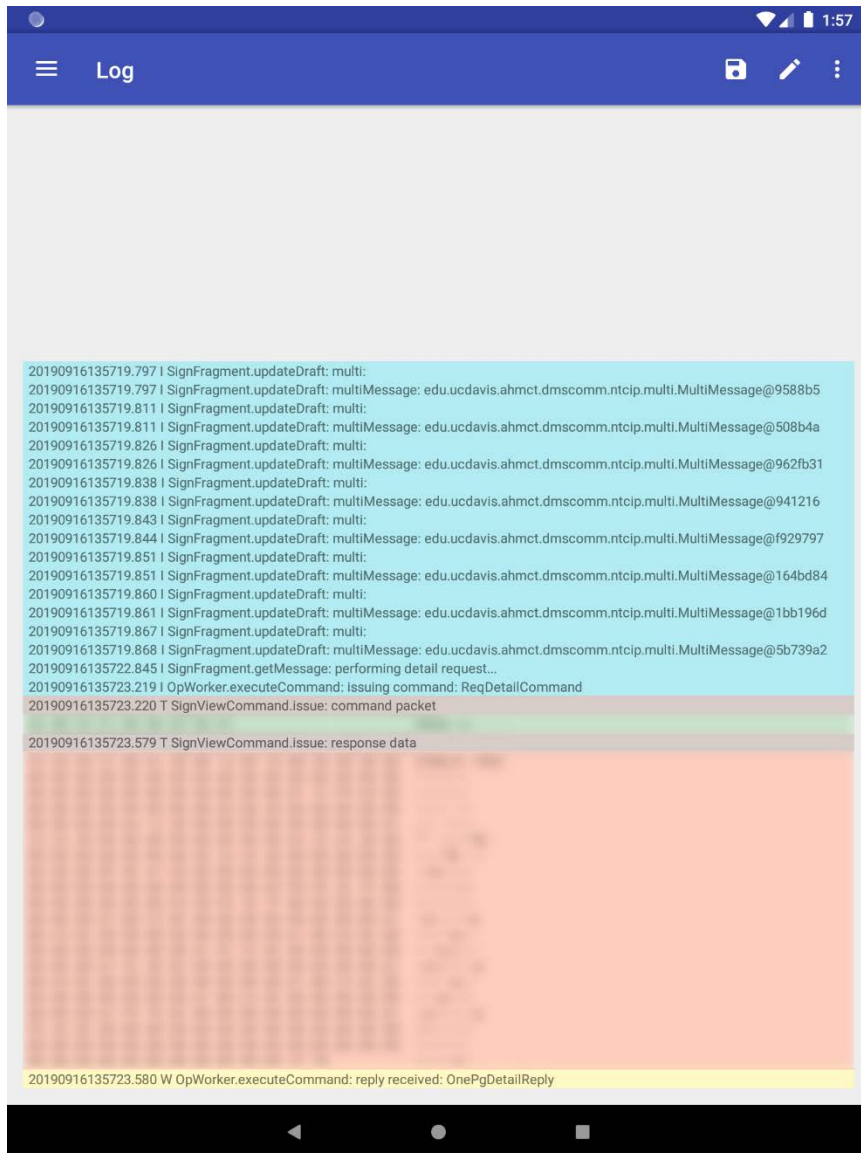


Figure 3.70: DMS log showing the output following a detail request

Figure 3.71 shows the resultant text annotation dialog by clicking on the pencil icon. Any text entered in this dialog will be entered into the existing log with the current timestamp.

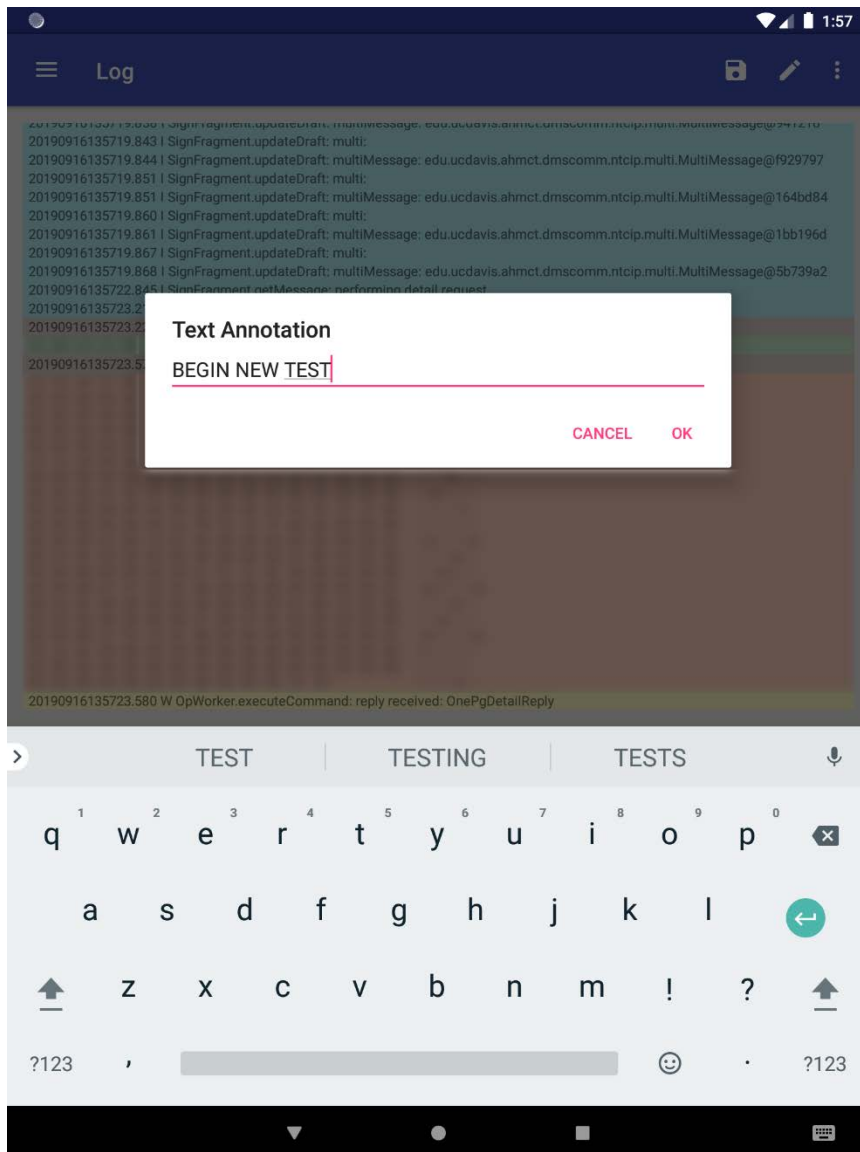


Figure 3.71: DMS log begin new test annotation

Figure 3.72 shows the log displaying the “BEGIN NEW TEST” annotation at the bottom of the log.

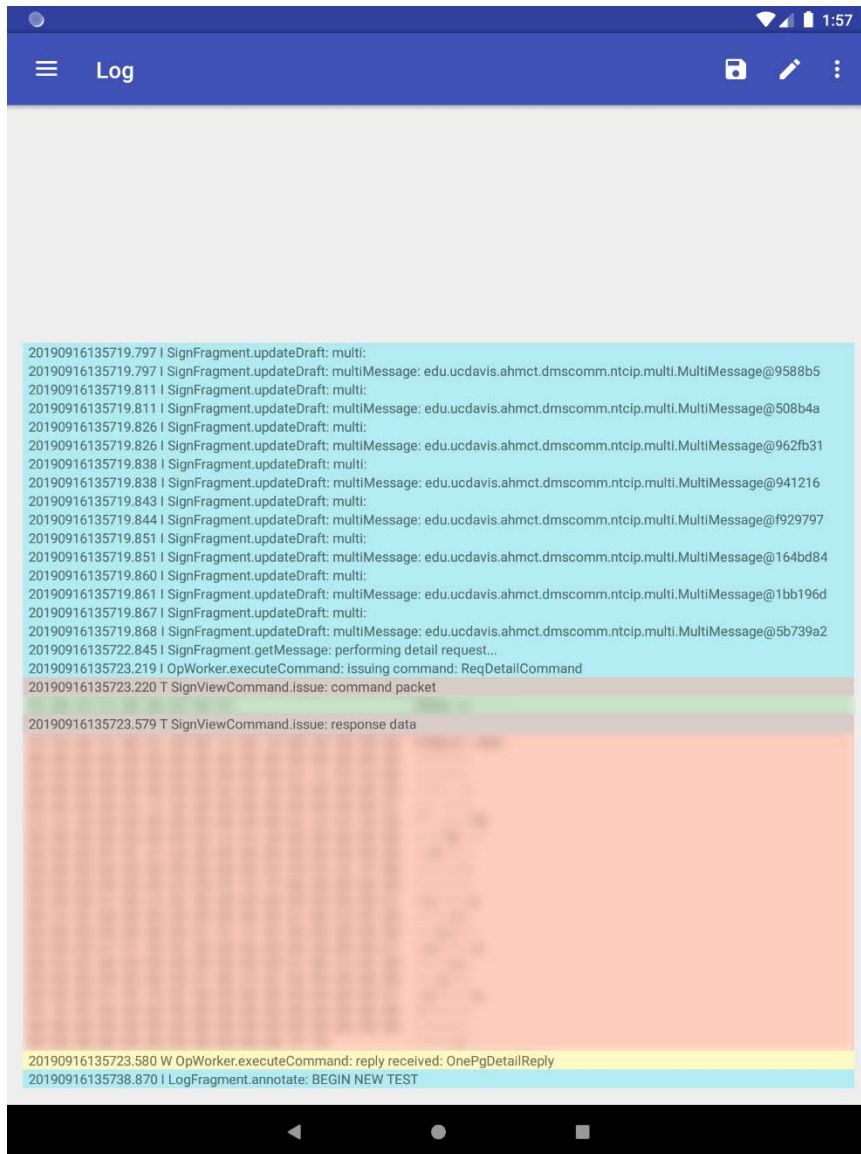


Figure 3.72: DMS log displaying begin new test annotation

Figure 3.73 shows building of the “NEW TEST” message.

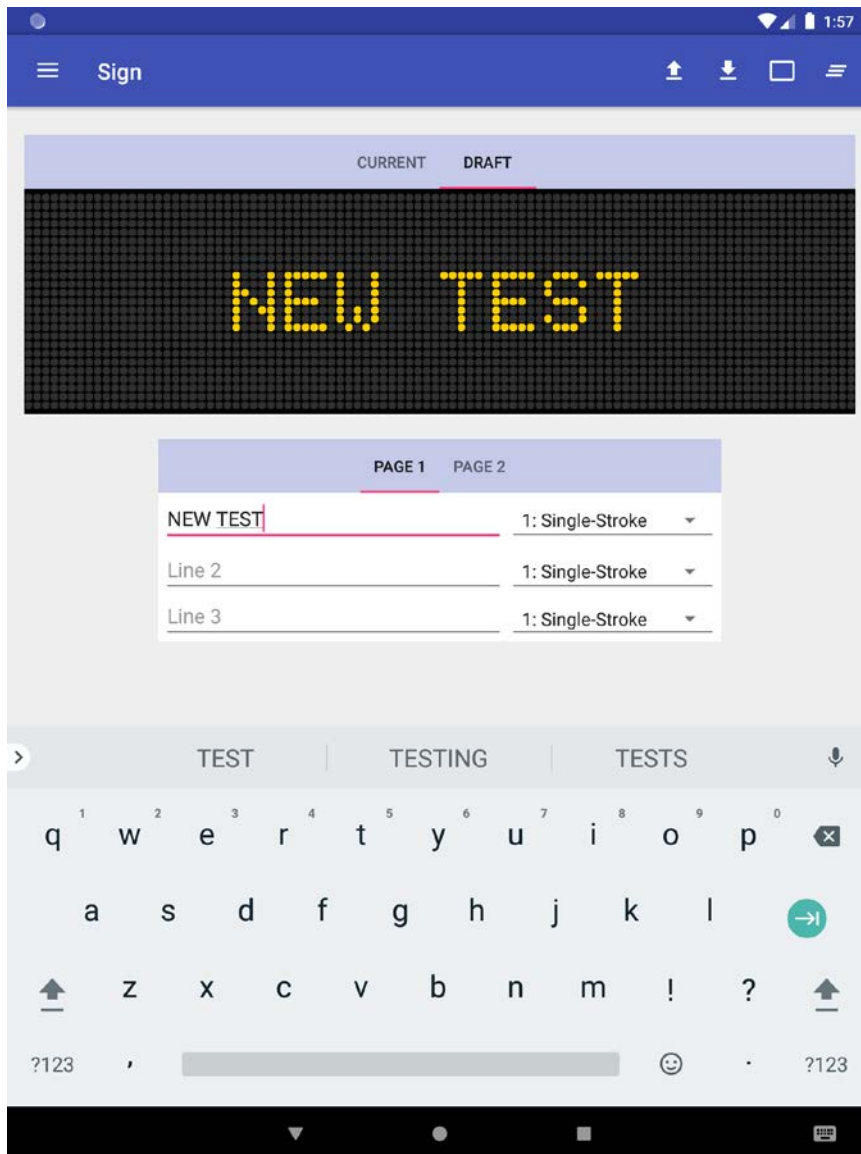


Figure 3.73: DMS sign draft new test message

Figure 3.74 shows the result of clicking on the up-arrow icon sending the new test message to the sign and awaiting a detail request for verification. All communications during this transaction are to be logged.

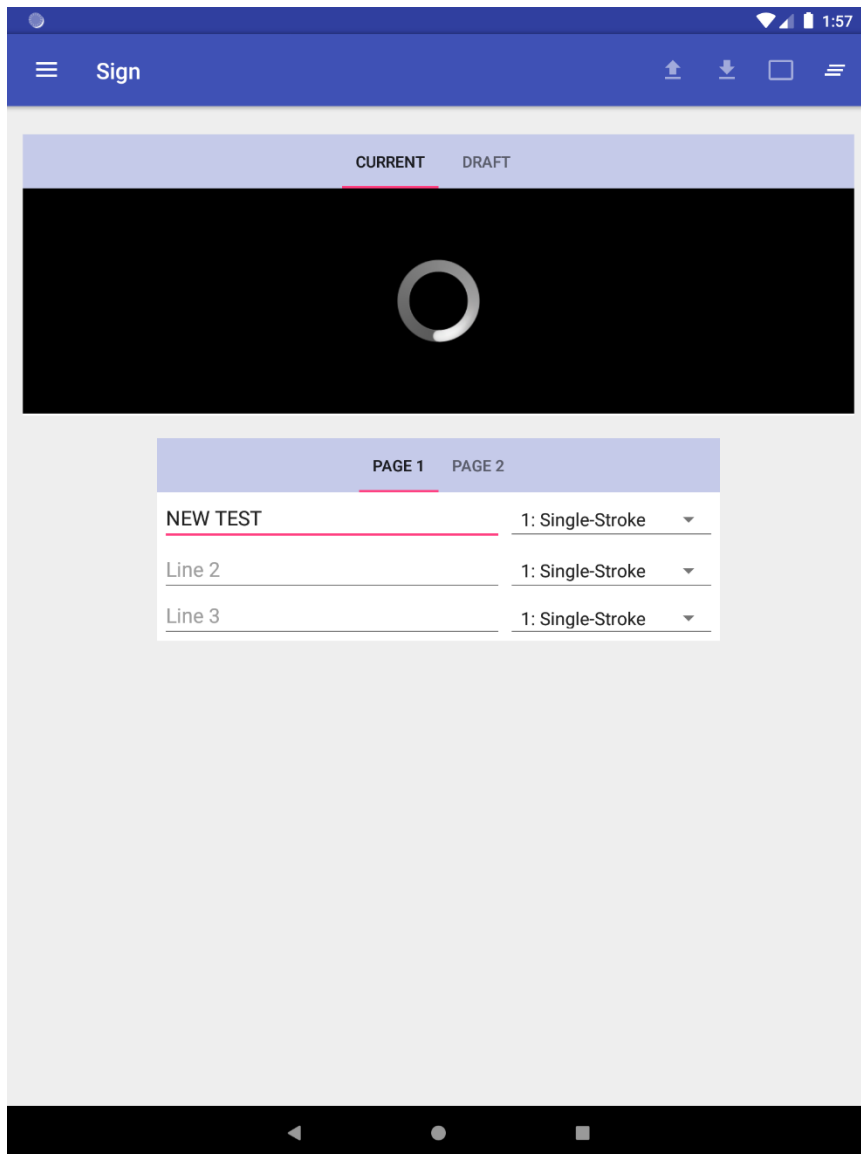


Figure 3.74: DMS sign sending new test message to controller

Figure 3.75 shows the currently display message on the sign.

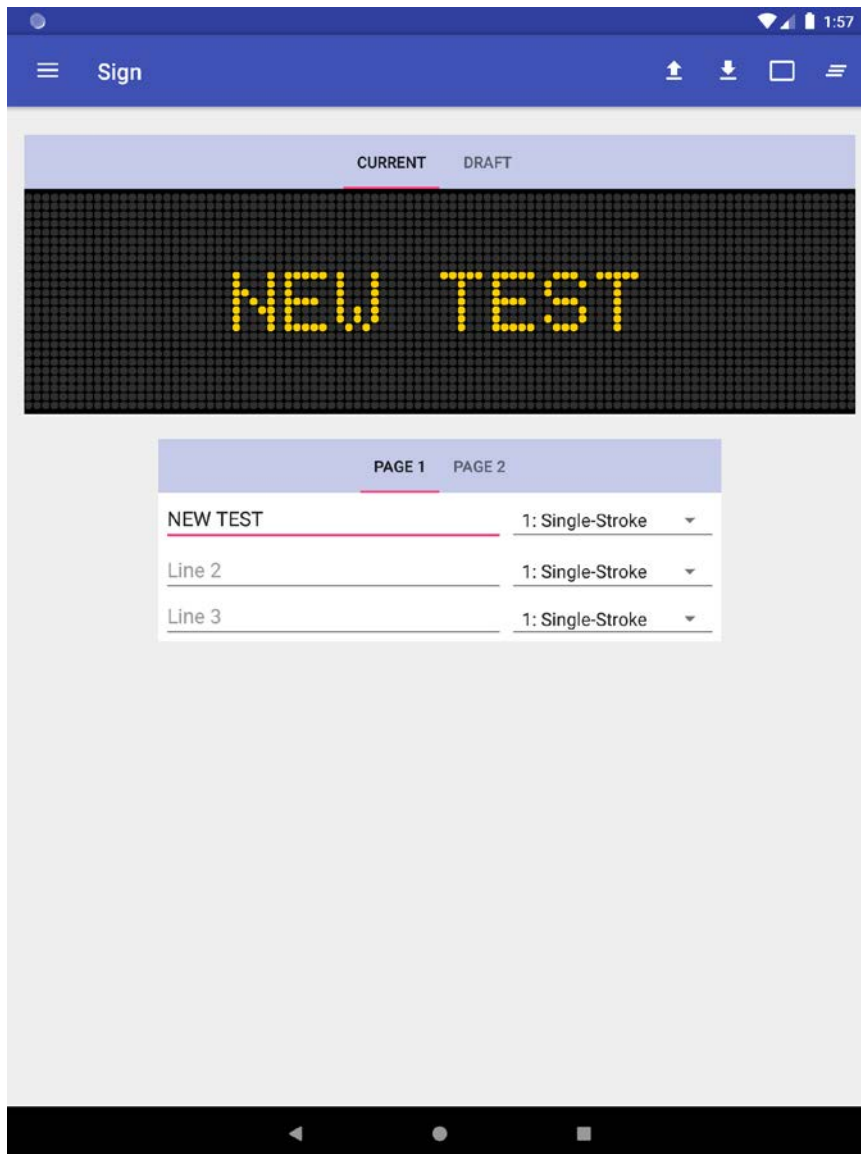


Figure 3.75: DMS sign displaying the content currently stored on the controller

Figure 3.76 shows the log entries due to the new test transactions.



Figure 3.76: DMS log displaying results following the new test

Figure 3.77 shows the log annotation to mark the “END NEW TEST”.

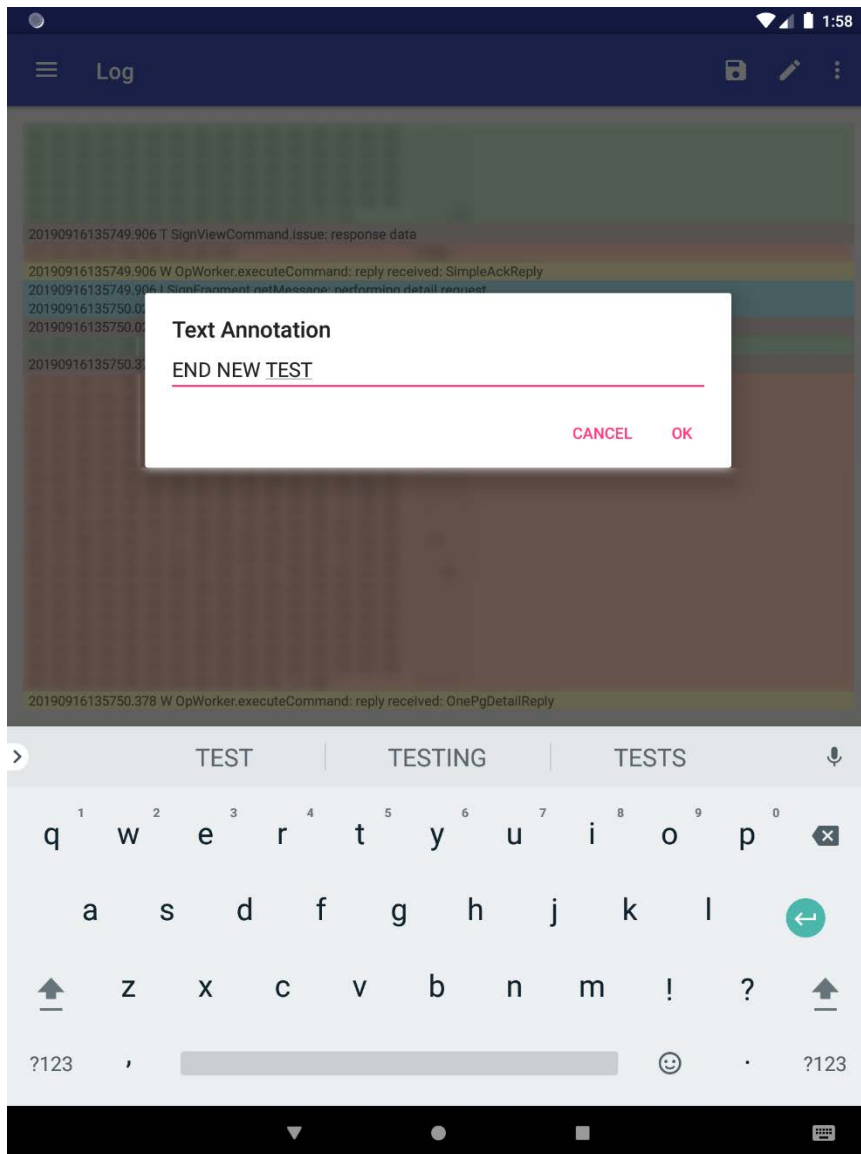


Figure 3.77: DMS log end new test annotation

Figure 3.78 shows the end of the new test annotation.



Figure 3.78: DMS log displaying the end new test annotation

File Locations

/sdcard/Android/data/edu.ucdavis.ahmct.dms/files/configurations

/sdcard/Android/data/edu.ucdavis.ahmct.dms/files/logs

/sdcard/Android/data/edu.ucdavis.ahmct.dms/files/pictures

Configurations

File Format

```
[
  {
    "communicationConfigurations":[
      {
        "drop":80,
        "endpoint":{
          "type":"COMPORT",
          "baud":1200,
          "dataBits":"EIGHT",
          "flowControl":"NONE",
          "parity":"NONE",
          "purgeRx":true,
          "purgeTx":true,
          "rts":true,
          "stopBits":"ONE",
          "acknowledgementTimeout":8000,
          "host":"192.168.10.1",
          "port":3696,
          "socketConnectionTimeout":20000,
          "socketReadTimeout":20000,
          "idleCloseTimeout":0,
          "readTimeout":16000,
          "name":"ComPort"
        },
        "name":"Serial via Airconsole",
        "protocol":"SIGNVIEW"
      }
    ],
  },
],
```

```

    "location":"ucdavis",
    "model":"MODEL_500",
    "name":"170"
  },
  {
    "communicationConfigurations":[
      {
        "drop":81,
        "endpoint":{
          "type":"TCP",
          "host":"192.168.11.101",
          "port":771,
          "socketConnectionTimeout":20000,
          "socketReadTimeout":20000,
          "idleCloseTimeout":0,
          "readTimeout":16000,
          "name":"Tcp"
        },
        "name":"Ethernet via Airconsole",
        "protocol":"SIGNVIEW"
      }
    ],
    "location":"ucdavis",
    "model":"MODEL_500",
    "name":"2070"
  }
]

```


Sign configuration object parameters

1. name - the name or description of the sign, not null
2. model - the sign model, not null, valid values {MODEL_500, MODEL_510, MODEL_520, MODEL_710, MODEL_720, MODEL_730}
3. location - the location of the sign, not null
4. communicationConfigurations - a list of sign communication objects, not null

Sign communication configuration object parameters

1. name - the name or description of the communication configuration, not null
2. protocol - the protocol of the communication configuration, not null, valid values {SIGNVIEW, NTCIP}
3. endpoint - the endpoint type of the communication configuration, not null, valid values {TCP, COMPORT}
4. drop - the drop number of the communication configuration, valid values [0, inf)

TCP endpoint object parameters

1. name - the endpoint name or description (somewhat duplicative of the communication configuration name and likely to be deprecated), not null, not empty
2. idleCloseTimeout - the idle close timeout of the operations manager in (ms) where a value of 0 implies a closure after each operation, valid values [0, inf)
3. readTimeout - the read timeout of the operations manager in (ms), valid values [0, inf)
4. host - the hostname or ip address, not null, not empty
5. port - the port number, valid values [1, 65535]
6. socketConnectionTimeout - the socket connection timeout in (ms) where a value of 0 implies none, valid values [0, inf)
7. socketReadTimeout - the socket read timeout in (ms) where a value of 0 implies none, valid values [0, inf)

ComPort endpoint object parameters

1. name - the endpoint name or description (somewhat duplicative of the communication configuration name and likely to be deprecated), not null, not empty
2. idleCloseTimeout - the idle close timeout of the operations manager in (ms) where a value of 0 implies a closure after each operation, valid values [0, inf)
3. readTimeout - the read timeout of the operations manager in (ms), valid values [0, inf)
4. host - the hostname or ip address, not null, not empty
5. port - the port number, valid values [1, 65535]
6. socketConnectionTimeout - the socket connection timeout in (ms) where a value of 0 implies none, valid values [0, inf)
7. socketReadTimeout - the socket read timeout in (ms) where a value of 0 implies none, valid values [0, inf)
8. acknowledgementTimeout - the option negotiation acknowledgement timeout in (ms), valid values [0, inf)
9. baud - the baud rate to request at initialization, valid values [1, inf)
10. dataBits - the data bits to request at initialization, not null, valid values {FIVE, SIX, SEVEN, EIGHT}
11. parity - the parity to request at initialization, not null, valid values {NONE, ODD, EVEN, MARK, SPACE}
12. stopBits - the stop bits to request at initialization, not null, valid values {ONE, TWO, ONE_POINT_FIVE}
13. flowControl - the flow control to request at initialization, not null, valid values {NONE, XONXOFF, RTSCTS}
14. rts - whether to request that the RTS be set on at initialization, valid values {true, false}
15. purgeRx - whether to request that the access server receive buffer be purged at initialization, valid values {true, false}
16. purgeTx - whether to request that the access server transmit buffer be purged at initialization, valid values {true, false}

Chapter 4:

Handheld Diagnostic Controller Design

CCTV App

This chapter discusses design and implementation of the HHDC CCTV app software. The analogous DMS app design and implementation is presented in Chapter 3. The requirements for the HHDC CCTV app are provided in Appendix D.

CCTV App Overview

The HHDC DMS app supports CCTV configuration and diagnostics. It interfaces to CCTV controllers in the field through both network and serial interfaces, and supports full CCTV configuration and diagnostics.

The CCTV app can access Caltrans CCTV systems remotely over the network, or directly at the CCTV field location. Local connection from the HHDC to the CCTV components is supported by Wi-Fi-to-network, Universal Serial Bus (USB)-to-network, Wi-Fi-to-serial, or USB-to-serial adapters and associated software. Remote network connection is typically over standard tablet Wi-Fi or cellular connection.

Using these various wired and wireless connection methods, and supported by the CCTV app and HHDC kit contents, it is possible to connect to the CCTV cameras over network and/or serial connections for diagnostic and configuration purposes. Some common Caltrans use cases are:

1. Wired/wireless network connection to a CCTV field element cabinet
2. Wired/wireless serial connection to a CCTV camera

The CCTV app is standards-based wherever possible, and support standards commonly used by Caltrans and other DOTs. The app support local and remote diagnosis and configuration of CCTV cameras. For Caltrans-specific use, the app supports the Pelco-D, Cohu-I, and ONVIF (Profile S) protocols. The app uses modular drivers to implement camera protocols. The CCTV app supports TCP/IP communications over Wi-Fi, cellular, and USB. It also supports serial communications over Wi-Fi and USB.

CCTV App Design

The CCTV app was designed for Android (version 4.1 and later). It is written in Java, and leverages the Android API. The app provides function-specific

screens which are referenced herein. It supports well-known smartphone and tablet gestures, including tap, swipe, double-tap-slide, and pinch. This section documents the overall CCTV app design.

The primary CCTV functions, as illustrated in the fragment graph of Figure 4.1, are camera configuration and communication configuration, including creating, editing, viewing, and selection, camera streaming, control, and settings, and logging for error handling and debugging.

CCTV Fragment Architecture

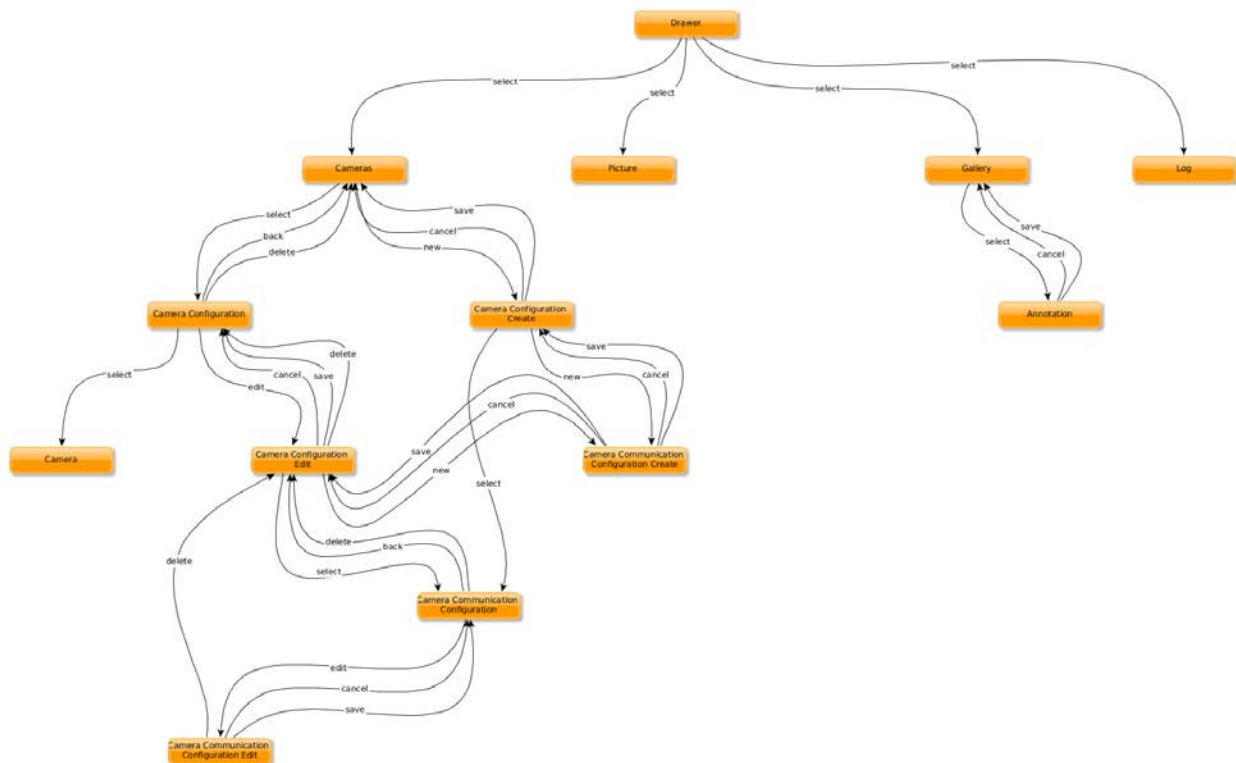


Figure 4.1: CCTV fragment graph

The overall configuration design is nearly identical to the DMS configuration design described in Chapter 3, with the exception of camera-specific options (models, protocols, etc.). The Pictures (i.e. tablet camera), Gallery, and Log functions are identical in every way to the DMS app.

The primary difference between the DMS app and the CCTV app are the primary operation screens (i.e. camera vs. sign). The camera screen was designed as a full-screen streaming viewer overlaid with standard PTZ controls. Other common controls (presets, focus, brightness, iris, etc.) are accessed through menu items.

CCTV App Implementation

The HHDC CCTV app supports common video protocols shown in Table 4.1. The supported PTZ protocols are provided in Table 4.2.

Table 4.1: Supported CCTV video protocols

Protocol
MPEG-4
H.264

Table 4.2: Supported CCTV PTZ protocols

Protocol	Version
Pelco-D	
Cohu-I	
ONVIF (Profile S)	2.2

The HHDC app can handle various CCTV configurations. Several configurations exist in various Caltrans districts, as illustrated in Figures 4.2 – 4.4.

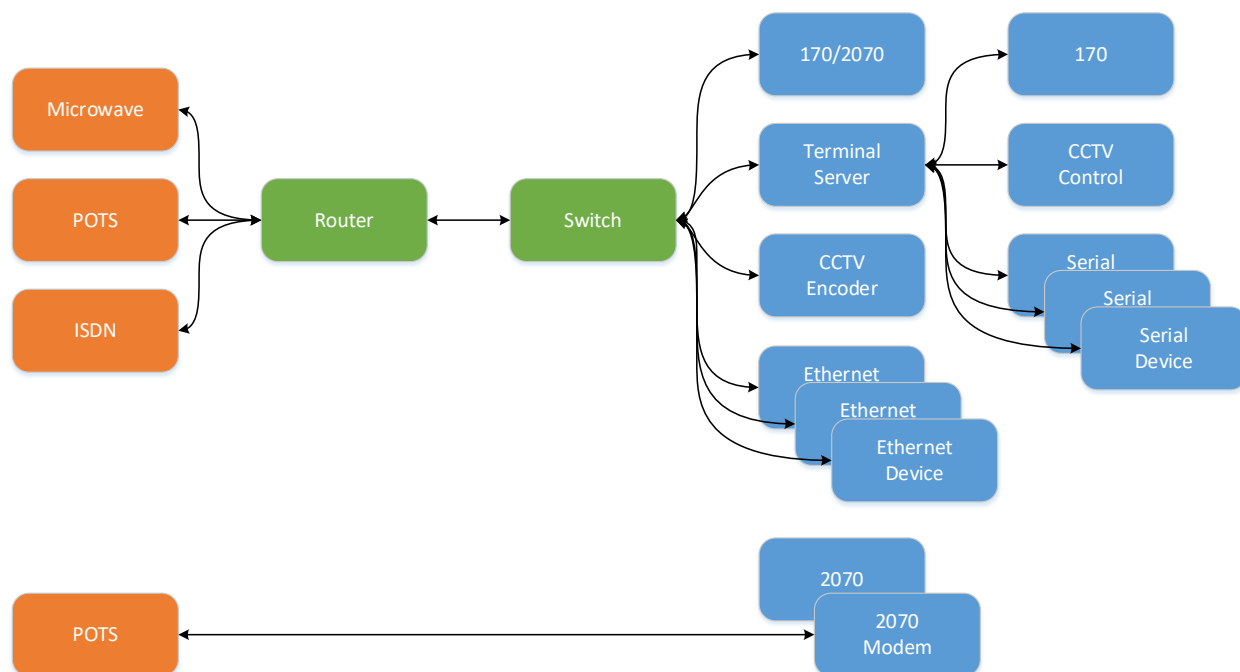


Figure 4.2: District configuration CCTV type A

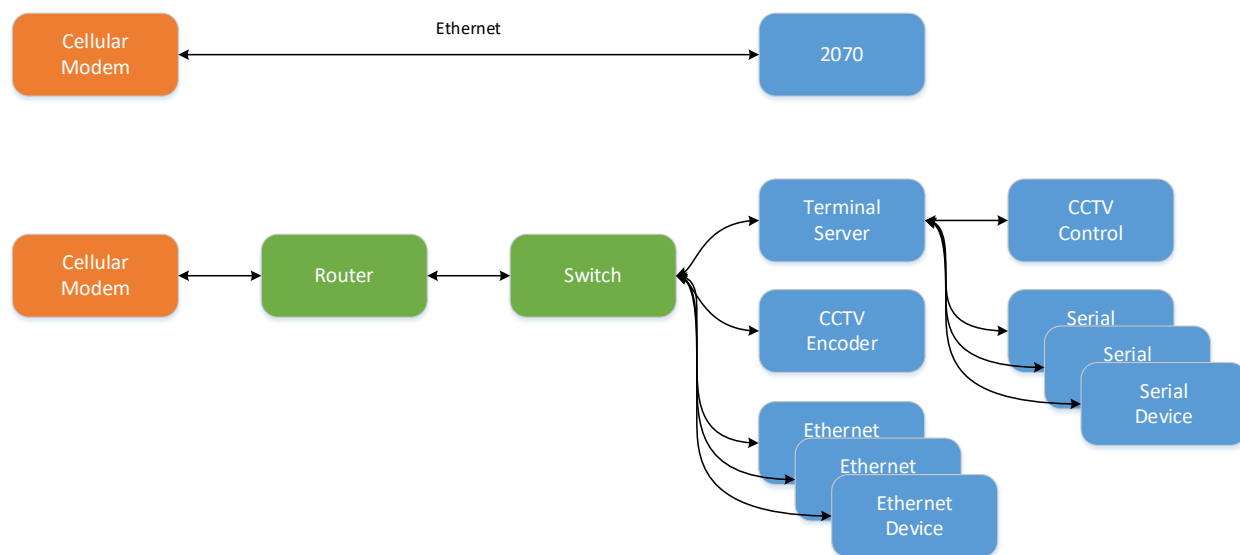


Figure 4.3: District configuration CCTV type B

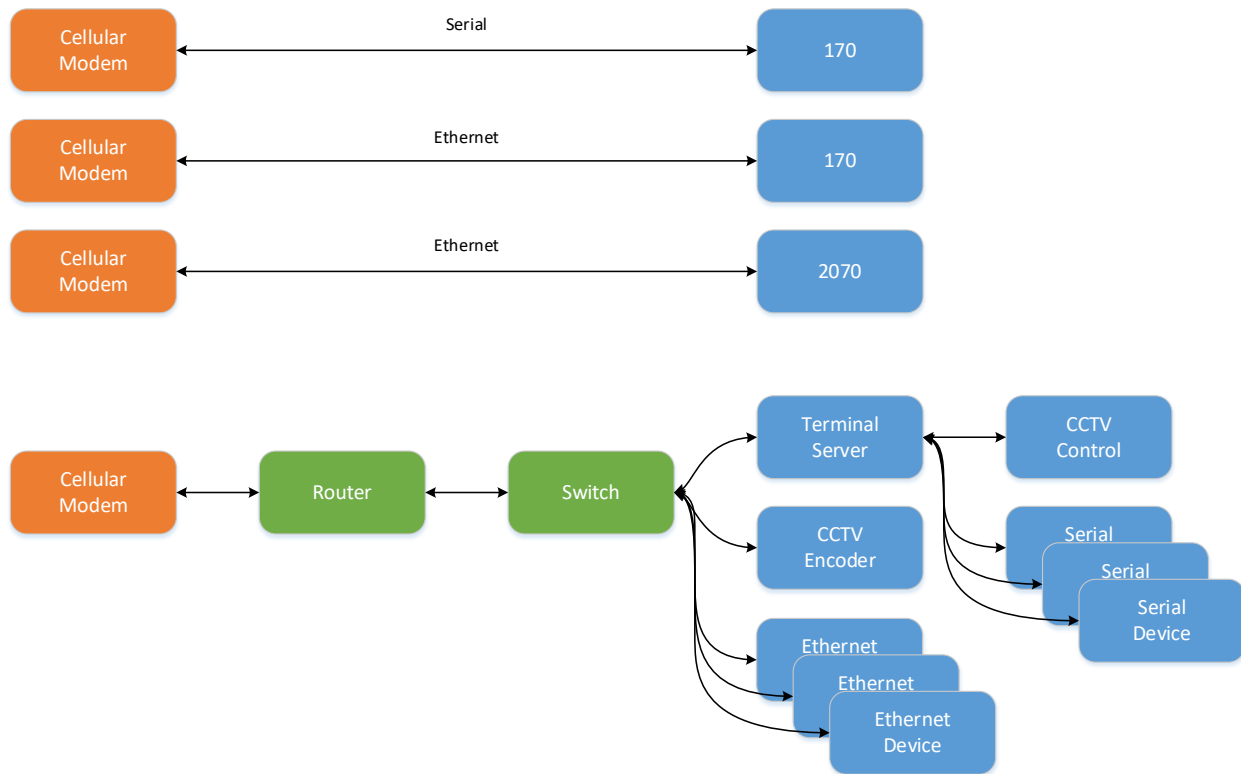


Figure 4.4: District configuration CCTV type C

Camera configurations are stored as follows:

- JavaScript Object Notation (JSON) .json files
- Primarily (de)serialization of classes:
 - CameraConfiguration
 - CameraCommunicationConfiguration
 - TcpEndpoint
 - ComPortEndpoint
- EXTERNAL_FILES/configurations/cctv_cameras.json
- Updated after every modification

Use cases are illustrated in Figure 4.5 (typical) and Figure 4.6 (requiring field element network visibility).

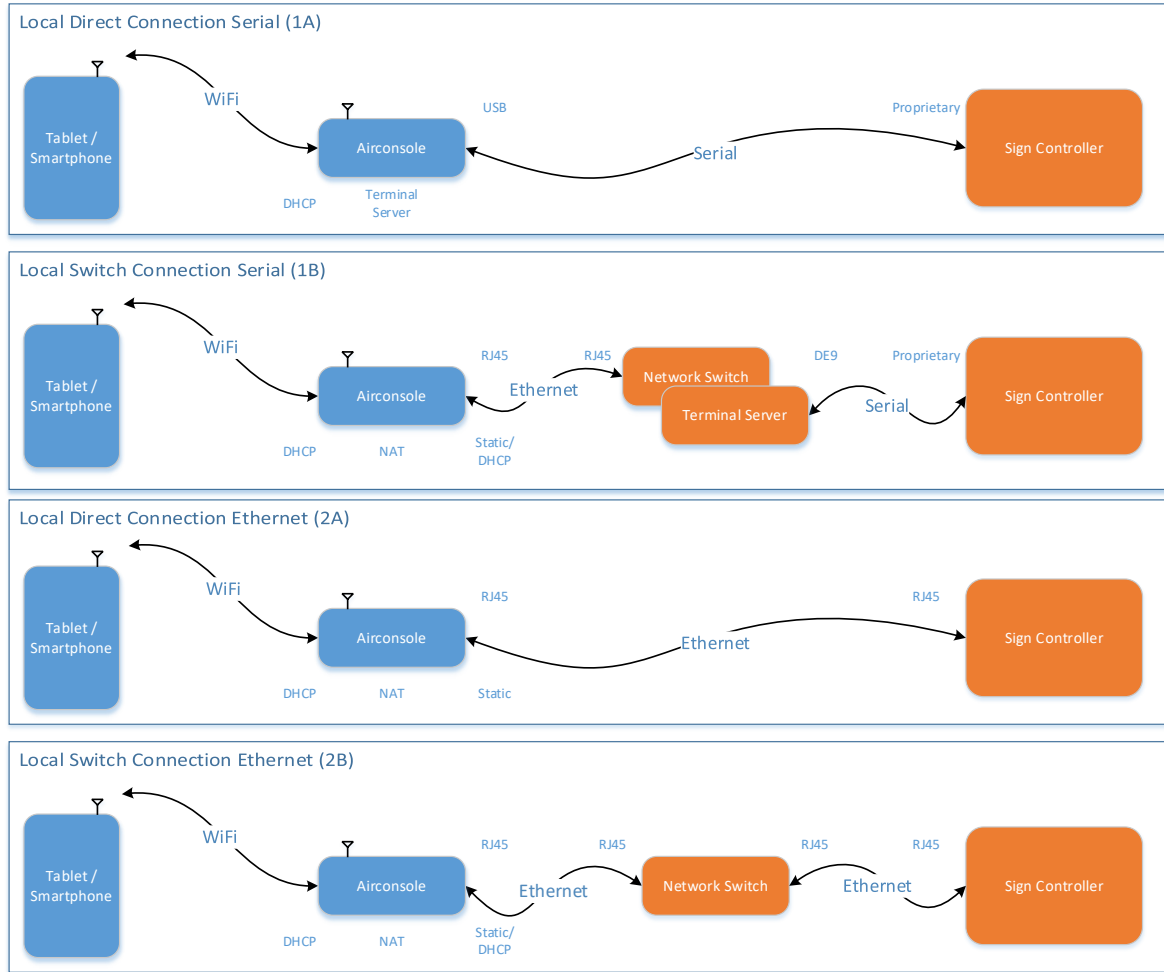


Figure 4.5: Typical CCTV use cases

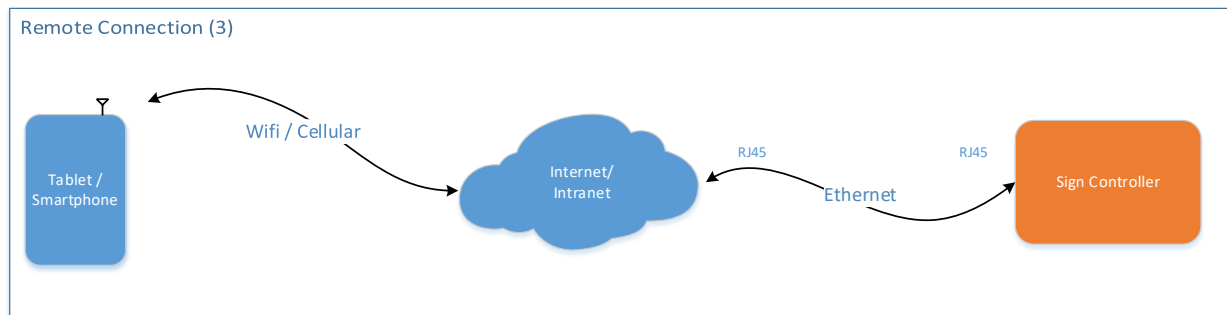


Figure 4.6: CCTV use case requiring field element network visibility

CCTV Software Problems

Camera/Encoder Video Protocol Issues

MJPEG Issues

Motion JPEG is not natively supported by the Android platform. Initial implementations of the drivers and implementation methodologies available at the time were non-performant. However, as of this writing, there appear to be new driver candidates for implementation.

MPEG4 Issues

No issues to report with lab testing.

H.264 Issues

No issues to report with lab testing.

PTZ Protocol Issues

Pelco-D Issues

No issues to report with lab testing, although the specification was not completely implemented for a variety of reasons, not limited to access to a narrow range of test cameras, and many reported/documented specification vs. camera implementation inconsistencies.

Cohu-I Issues

No issues to report with lab testing, although the specification was not completely implemented, for reasons noted in the Pelco-D section.

ONVIF Issues

ONVIF Profile S is a large specification. The entire mandatory client specification was developed and much of the conditional/optional specification. However, much of the breadth of the complete test coverage is not possible due to conditional/optional server implementation requirements and/or available verification cameras.

Drivers

Pelco-D

All widely-used specifications were implemented and tested.

Cohu-I

All widely-used specifications were implemented and tested.

ONVIF Profile S

All mandatory and much of the conditional/optional client specifications were implemented.

Primary Implemented Features

1. User authentication (Digest Authentication)
2. Query services and capabilities
3. Media transport (RTP/RTSP/HTTP/TCP)
4. Video streaming (MPEG4, H.264)
5. Video encoder configuration
6. PTZ move, presets, home, and configuration

CCTV App Screen Shots

This section contains the screen shots for the CCTV app, which is based on the design mockups of the previous section.

Figure 4.7 shows the primary startup screen for the CCTV app. Typically, after opening the app, the end user will immediately select a predefined camera configuration to begin testing and/or operating a camera. Clicking on the add button will begin creation of a new camera configuration.

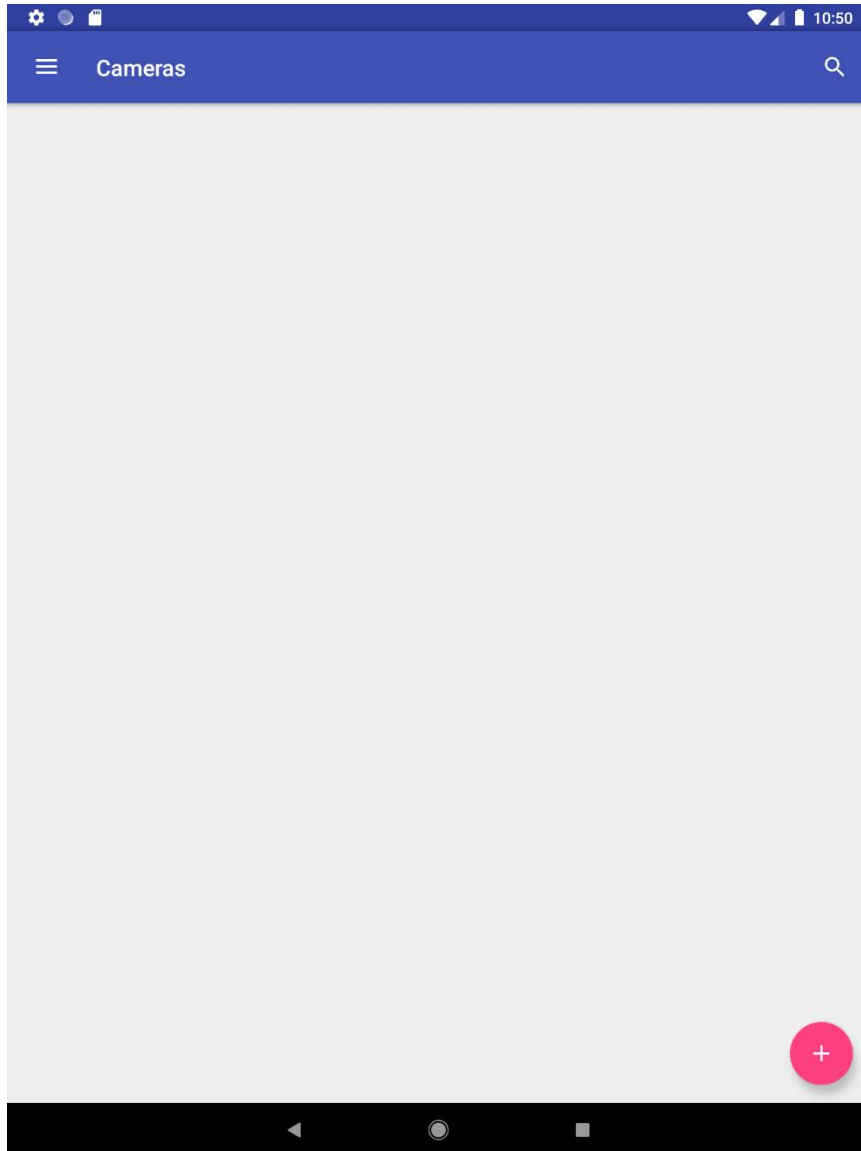


Figure 4.7: CCTV startup screen

Figure 4.8 shows the primary menu of the CCTV app. The menu is opened using one of two methods. The first is by swiping from the left edge of the screen to the right, and the second is by clicking on the 3-line icon in the upper left corner of all other screens (e.g. Figure 3.9). The menu is closed by either selecting a menu item, or swiping left. Using this menu, the user can access functionality for cameras, app configuration, CCTV app built-in camera and gallery, a log, and app settings.

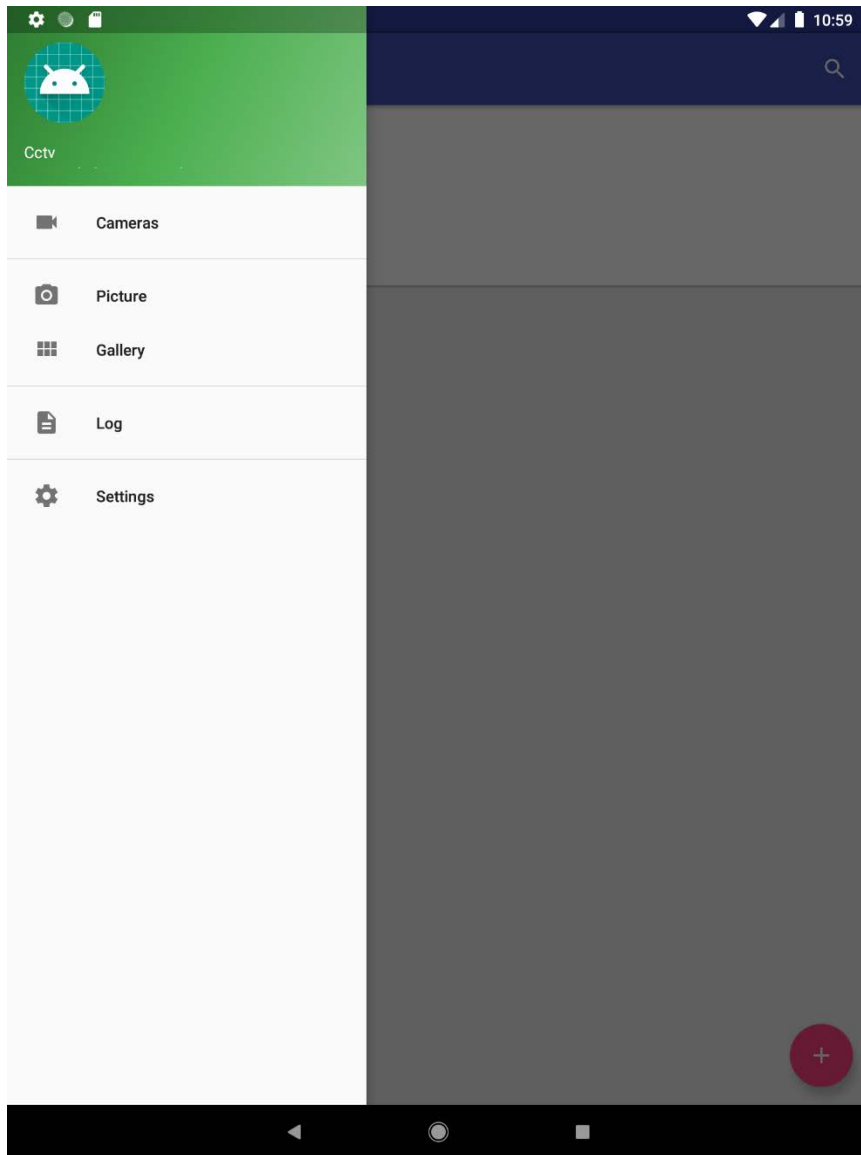


Figure 4.8: CCTV main menu

Figure 4.9 shows the default state of a new create camera configuration screen. The camera configuration is composed of a camera name, location, and one or more camera communication methods.

The screenshot displays a mobile application interface for creating a camera configuration. The top section is a blue header bar containing a hamburger menu icon on the left, the title "Create Camera Configuration" in the center, and "X" and checkmark icons on the right. Below the header, the form is divided into three sections. The first section, labeled "Name", contains a text input field with the placeholder text "name". The second section, labeled "Location", contains a text input field with the placeholder text "location". The third section, labeled "Communications", is a large, empty rectangular area. In the bottom right corner of this section, there is a red circular button with a white plus sign. The bottom of the screen features a black Android navigation bar with standard back, home, and recent apps icons.

Figure 4.9: CCTV create camera configuration

Figure 4.10 shows an example camera configuration, here for creation of camera 1.

The screenshot shows a mobile application interface for creating a camera configuration. The title bar is blue and contains the text "Create Camera Configuration". Below the title bar, there are two input fields: "Name" with the value "camera 1" and "Location" with the value "ucdavis". The "Location" field has a red underline. Below these fields is a large empty area labeled "Communications". At the bottom, there is a keyboard with the text "ucdavis" entered in the search bar. A red circular button with a white plus sign is located on the right side of the "Communications" area.

Figure 4.10: CCTV create sign configuration with example values

Figure 4.11 shows the default create camera communication configuration. All camera communication configurations are composed of a name, protocol, endpoint type, drop, depending on protocol, and specific additional parameters depending on the selected endpoint type.

The screenshot shows a mobile application interface for configuring camera communication. The title bar is blue and contains a hamburger menu icon, the text 'Create Camera Communication', and close and confirm icons. The form below has a light gray background and contains the following fields:

- Name:** Ethernet via Airconsole
- Protocol:** Cohu I (dropdown menu)
- Endpoint type:** ComPort (dropdown menu)
- Drop:** 0
- Host:** 192.168.10.1
- Port:** 3696
- Idle Close Timeout:** 0
- Read Timeout:** 16000
- Socket Connection Timeout:** 20000
- Socket Read Timeout:** 20000
- Acknowledgement Timeout:** 8000
- Baud:** 1200
- Data bits:** 8 (dropdown menu)

The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a circle, and a square.

Figure 4.11: CCTV create sign communication with default values

Figure 4.12 shows the camera communication protocol options composed of Cohu I, Pelco D, and Onvif.

The screenshot displays a mobile application interface for creating camera communication settings. The title bar is blue with a hamburger menu icon, the text 'Create Camera Communication', and a close/cancel icon. The status bar at the top shows system icons and the time 10:50. The form fields are as follows:

Field Label	Value
Name	Ethernet via Airconsole
Protocol	Cohu I (selected from dropdown)
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	8

Figure 4.12: CCTV create camera communication protocol options

Figure 4.13 shows the camera communication endpoint type options composed of Tcp, Telnet, and ComPort.

The screenshot displays a mobile application interface for creating camera communication settings. The title bar is blue with a hamburger menu icon on the left and close/cancel and check/confirm icons on the right. The form fields are as follows:

Field Label	Value
Name	Ethernet via Airconsole
Protocol	Cohu I
Endpoint type	Tcp, Telnet, ComPort (dropdown menu open)
Host	192.168.10.1
Port	3696
Idle Close Timeout	0
Read Timeout	16000
Socket Connection Timeout	20000
Socket Read Timeout	20000
Acknowledgement Timeout	8000
Baud	1200
Data bits	8

Figure 4.13: CCTV create camera communication endpoint type options

Figure 4.14 shows an example camera communication configuration. In this example, we are creating a communication configuration for camera 1 set to communicate Onvif over TCP.

The screenshot shows a mobile application interface for creating a camera communication configuration. The title bar is blue with a hamburger menu icon on the left and close/cancel and confirm/submit icons on the right. The main form area is light gray and contains the following fields:

- Name:** Ethernet via Airconsole
- Protocol:** Onvif (selected from a dropdown menu)
- Endpoint type:** Tcp (selected from a dropdown menu)
- Drop:** 0
- Host:** 192.168.11.101
- Port:** 771
- Idle Close Timeout:** 0
- Read Timeout:** 16000
- Socket Connection Timeout:** 20000
- Socket Read Timeout:** 20000

The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a home circle, and a recent apps square.

Figure 4.14: CCTV create camera communication configuration with example values

Figure 4.15 shows the completed camera and associated communication configuration.

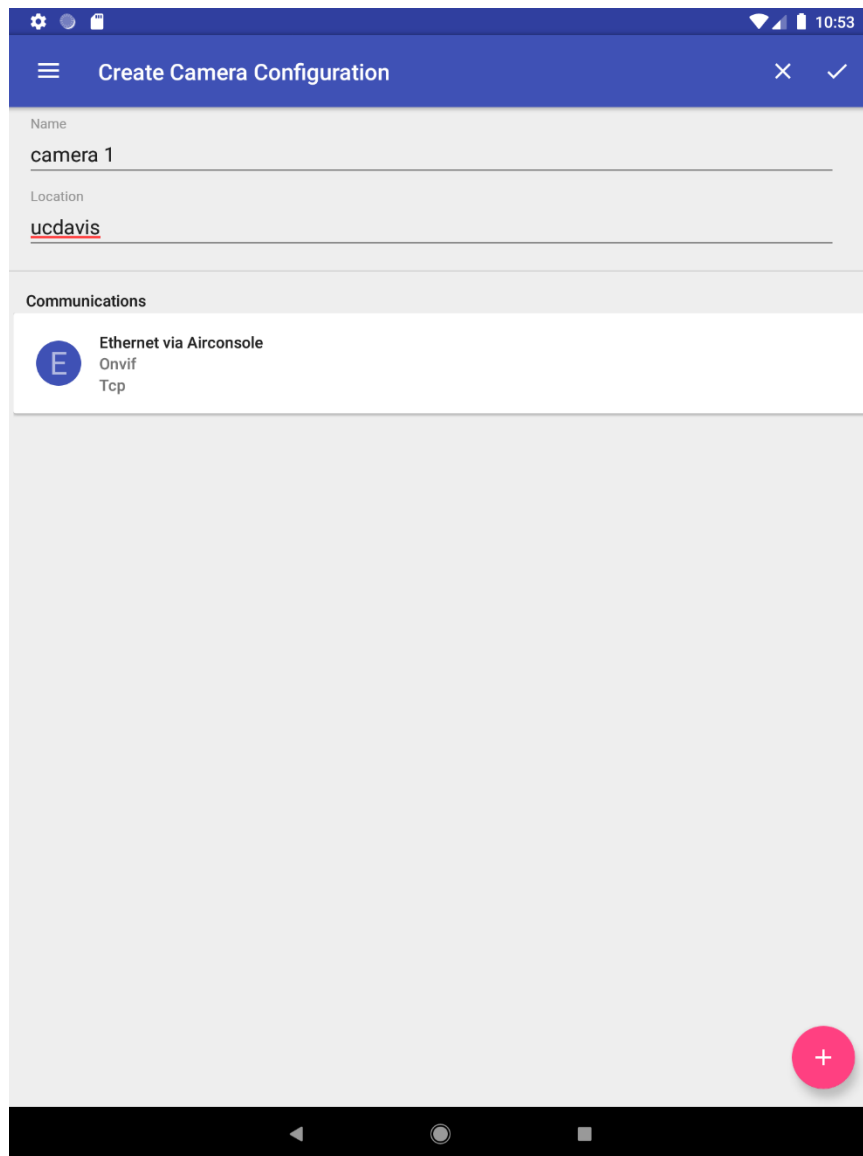


Figure 4.15: CCTV camera configuration with an Onvif over Tcp communication configuration

Figure 4.16 shows our example camera configuration, camera 1, available for use.

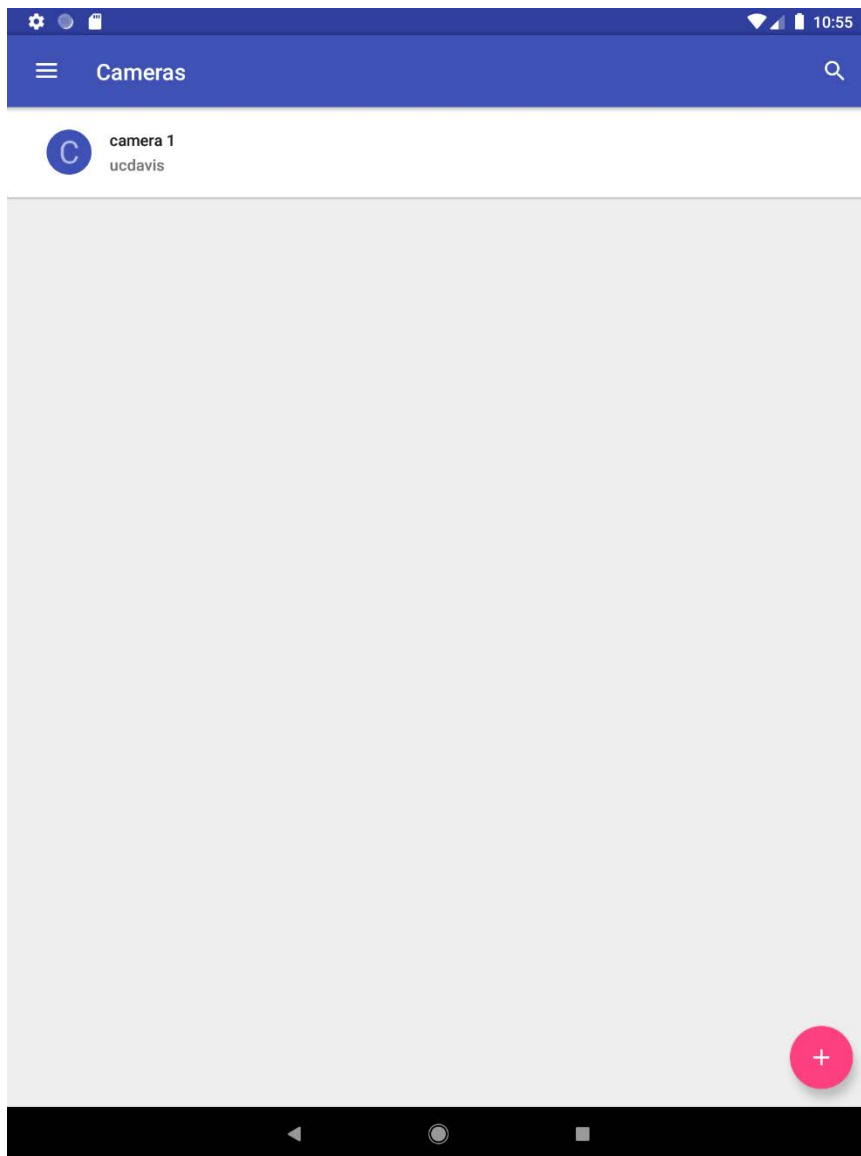


Figure 4.16: CCTV list of configured cameras

Figure 4.17 shows an example camera configuration, here for creation of camera 2.

The screenshot displays a mobile application interface for creating a camera configuration. The top section is a blue header bar containing a hamburger menu icon, the text "Create Camera Configuration", and "X" and checkmark icons. Below the header, the form is divided into three sections: "Name" with the input "camera 2", "Location" with the input "ucdavis", and "Communications" which is currently empty. A pink circular button with a white plus sign is positioned at the bottom right of the form area. The bottom of the screen shows the standard Android navigation bar.

Figure 4.17: CCTV create camera configuration with example values

Figure 4.18 shows an example camera communication configuration, here creating a communication configuration for camera 2 set to communicate Cohu I over TCP.

The screenshot displays a mobile application interface for configuring camera communication. The title bar at the top is blue and contains a menu icon, the text 'Create Camera Communication', and close/confirm icons. The main form area is light gray and contains the following fields:

- Name:** Ethernet via Airconsole
- Protocol:** Cohu I
- Endpoint type:** Tcp
- Drop:** 0
- Host:** 192.168.11.101
- Port:** 771
- Idle Close Timeout:** 0
- Read Timeout:** 16000
- Socket Connection Timeout:** 20000
- Socket Read Timeout:** 20000

The bottom of the screen shows a black navigation bar with standard Android icons (back, home, recent apps).

Figure 4.18: CCTV create camera communication configuration with example values

Figure 4.19 shows the completed camera and associated communication configuration.

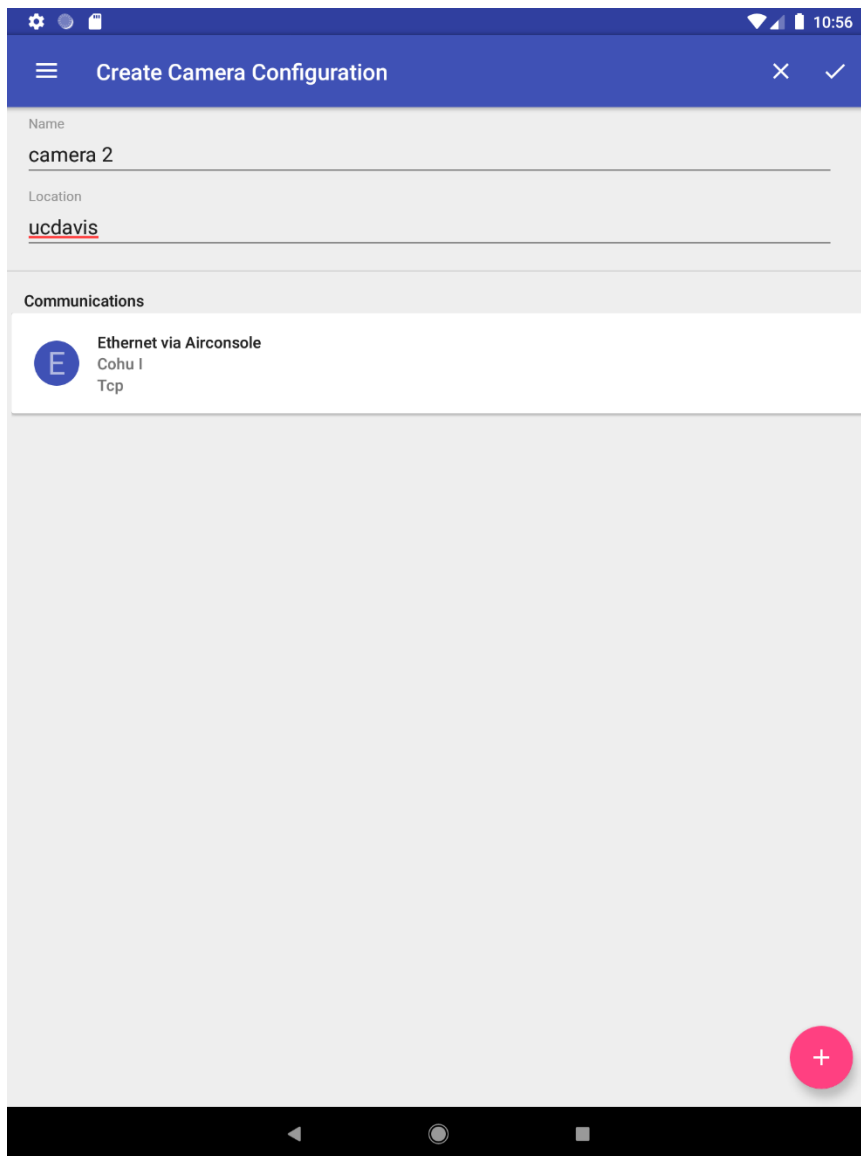


Figure 4.19: CCTV camera configuration with a Cohu I over Tcp communication configuration

Figure 4.20 shows an example camera communication configuration, here creating a second communication configuration for camera 2 set to communicate Pelco D over serial.

Create Camera Communication

Name: Serial via Airconsole

Protocol: Pelco D

Endpoint type: ComPort

Drop: 0

Host: 192.168.10.1

Port: 3696

Idle Close Timeout: 0

Read Timeout: 16000

Socket Connection Timeout: 20000

Socket Read Timeout: 20000

Acknowledgement Timeout: 8000

Baud: 1200

Data bits: 8

Figure 4.20: CCTV create a second camera communication configuration with example values

Figure 4.21 shows the completed camera and associated communication configurations. This example shows that it is possible to have more than one communication configuration per camera. This is useful in cases where it is necessary to exercise multiple camera protocols or connect over several interfaces (i.e. local network, remote network, and local serial).

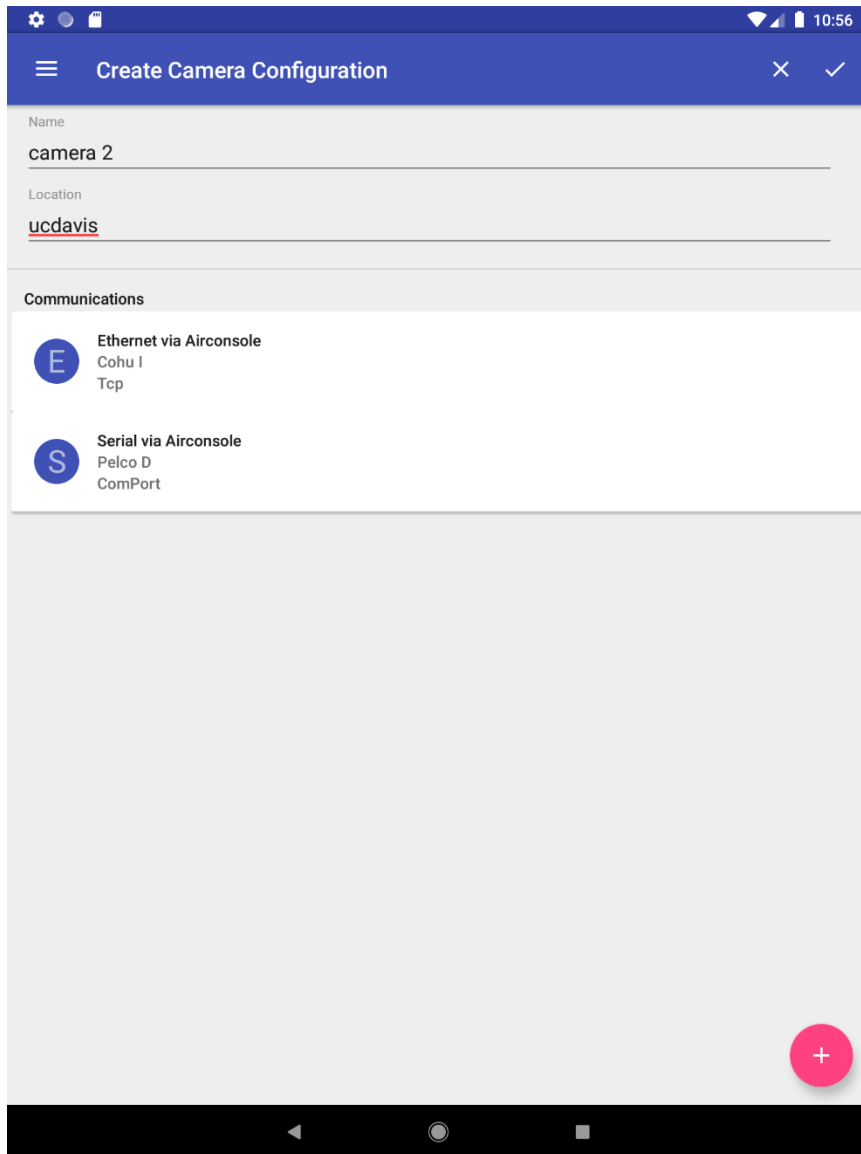


Figure 4.21: CCTV camera configuration with multiple communication configurations

Figure 4.22 shows example camera configurations, camera 1 and camera 2, available for use.

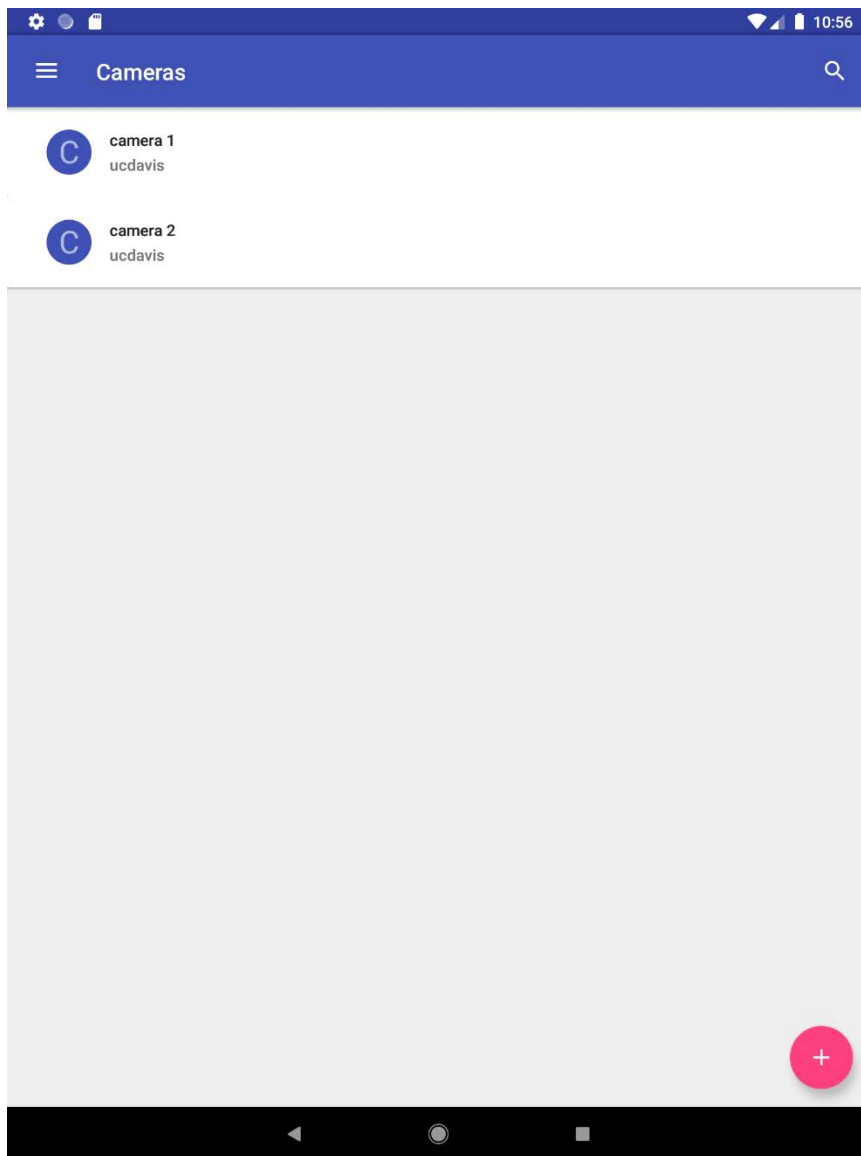


Figure 4.22: CCTV list of configured cameras

Figure 4.23 shows the results of clicking on a communication configuration from a read-only camera configuration screen as seen in Figure 4.22. The basic screen shows the streaming content from the camera, with menu options supporting, from left to right, PTZ control, presets, brightness, focus, overlays, and screenshots.



Figure 4.23: CCTV camera displaying streaming content

Figure 4.24 shows the results of clicking on the PTZ menu icon and enabling on-screen control. Control of the pan/tilt is accomplished by using a finger within the control circle like a joystick. The control supports movement in any direction, and the speed is a function of distance from the center.

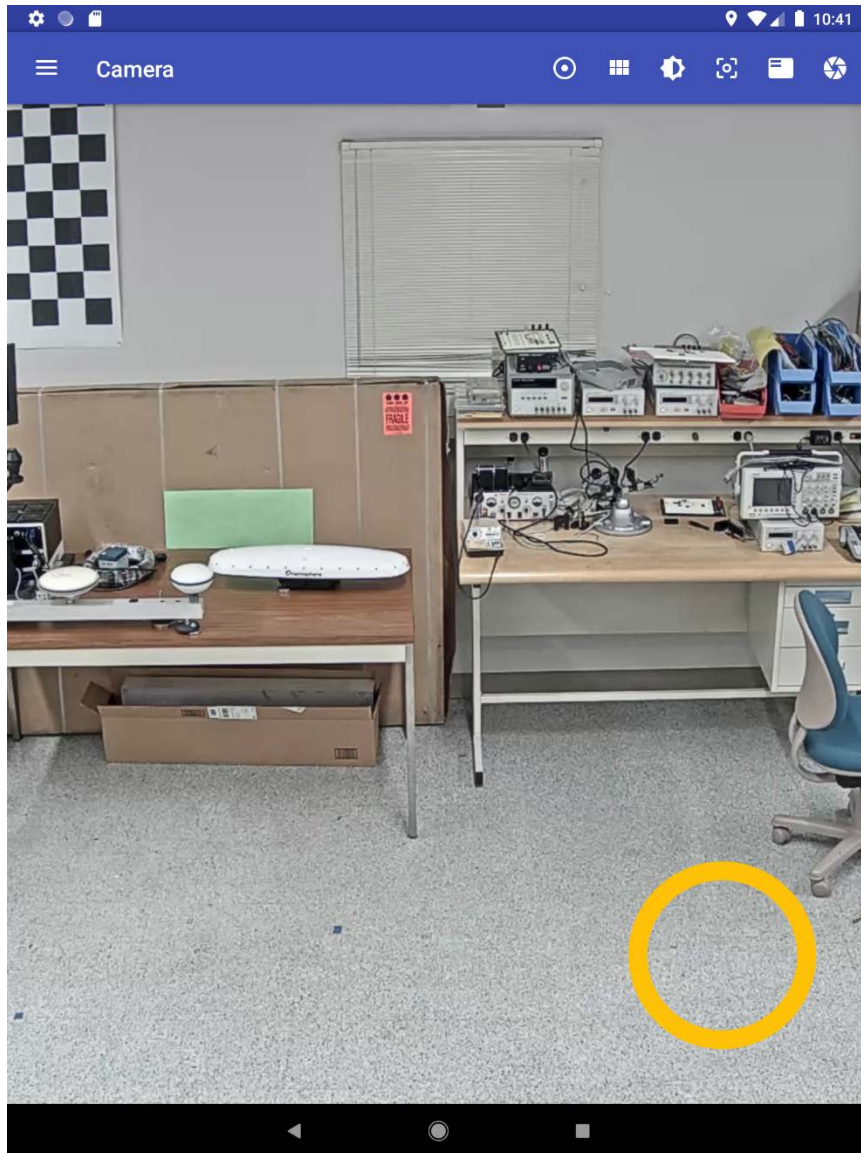


Figure 4.24: CCTV camera with PTZ controls enabled

Figure 4.25 shows the results of clicking on the presets menu icon and enabling the on-screen buttons. Setting a preset is accomplished by long-pressing a selected button, while recalling a preset location is accomplished by a normal short button click.

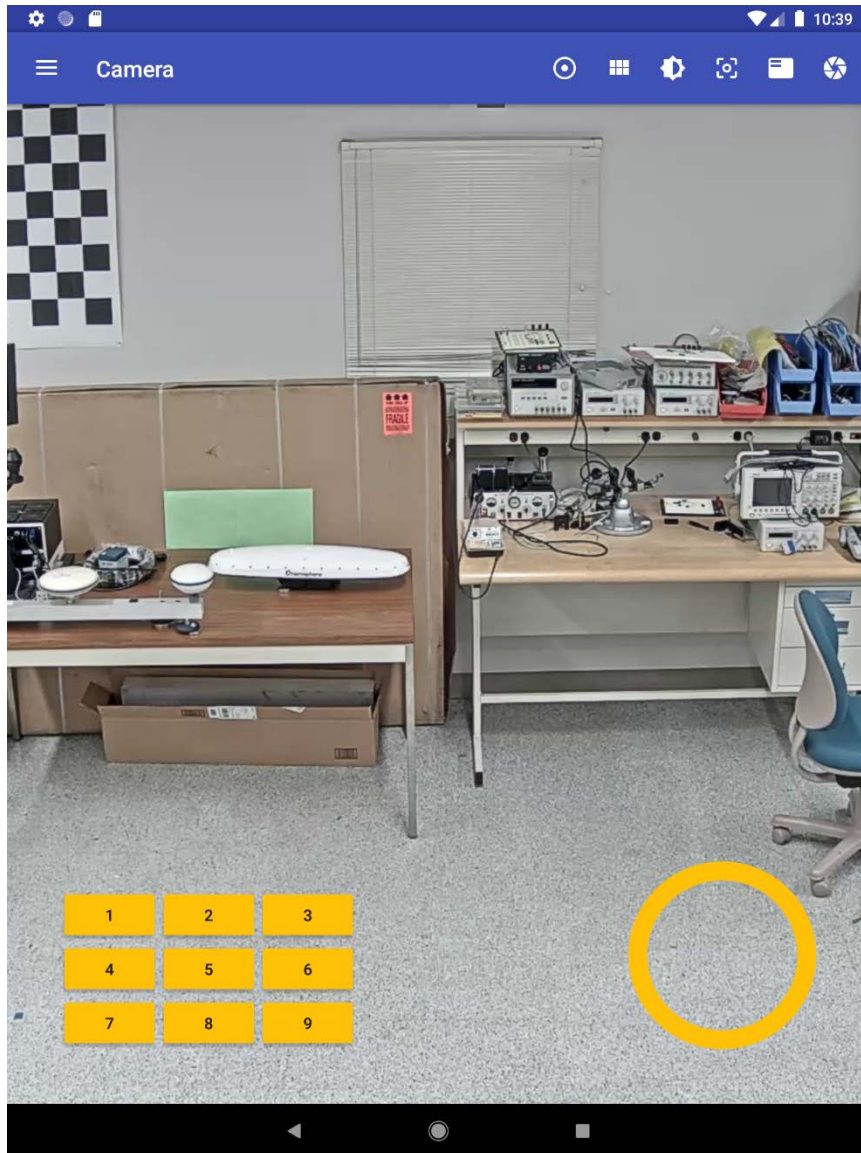


Figure 4.25: CCTV camera with presets enabled

Figure 4.26 shows the camera view, as seen in Figure 4.25, in a zoomed out state. Zoom out is accomplished by using the PTZ control pinch-to-zoom gestures. Bringing two fingers together inside the control will zoom out the camera view as a function of finger spacing.

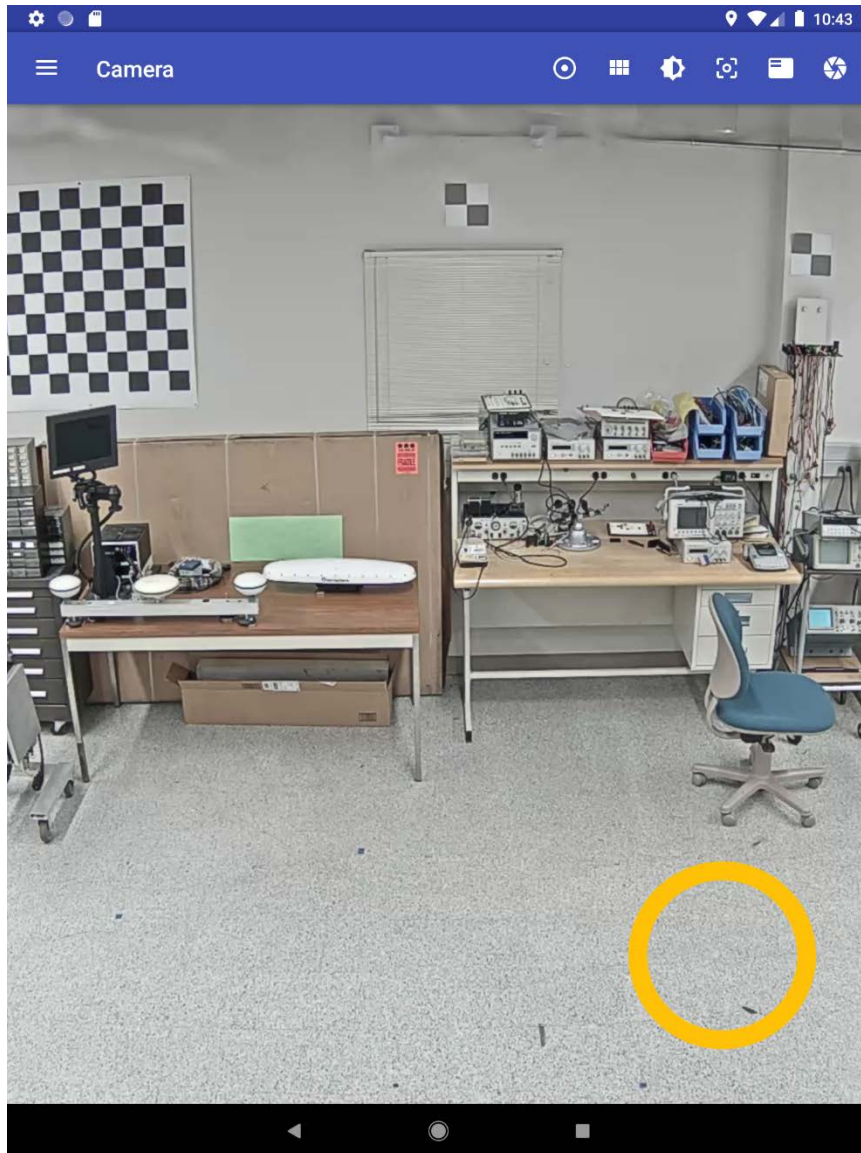


Figure 4.26: CCTV camera showing zoom out state

Figure 4.27 shows the camera view, as seen in Figure 4.25, in a zoomed in state. Zoom in is also accomplished by using the PTZ control pinch-to-zoom gestures. Separating two fingers apart inside the control will zoom in the camera view as a function of finger spacing.



Figure 4.27: CCTV camera showing zoom in state

Figure 4.28 shows the camera view, as seen in Figure 4.25, in a panned left state. This was accomplished by placing a finger within the control somewhere between the center and the left edge. Note that the distance from the center of the control affects the speed of the camera motion.



Figure 4.28: CCTV camera showing pan left state

Figure 4.29 shows the camera view, as seen in Figure 4.25, in a panned left and tilted down state. This was accomplished by placing a finger within the control somewhere between the center and the lower left edge.



Figure 4.29: CCTV camera showing pan left tilt down state

Figure 4.30 shows the camera view, as seen in Figure 4.25, in a panned right and tilted down state. This was accomplished by placing a finger within the control somewhere between the center and the lower right edge.

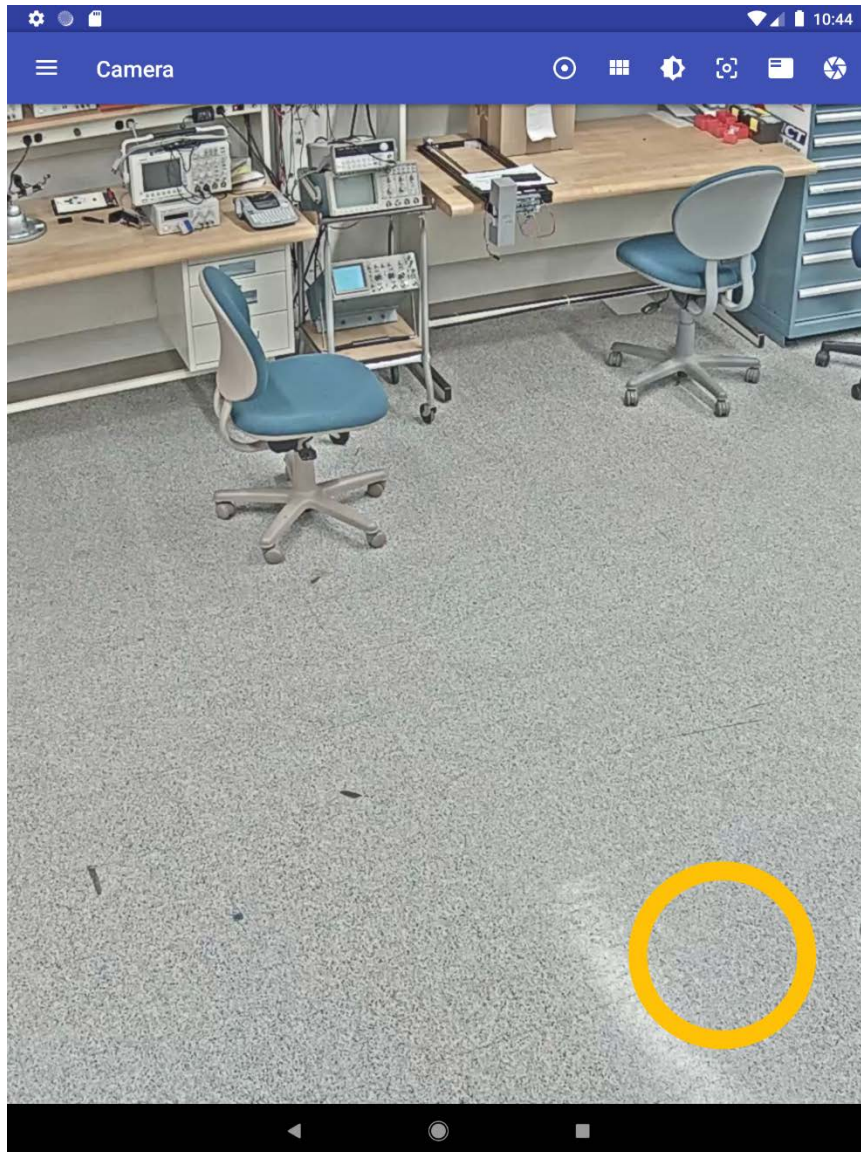


Figure 4.30: CCTV camera showing pan right tilt down state

File Locations

/sdcard/Android/data/edu.ucdavis.ahmct.cctv/files/configurations

/sdcard/Android/data/edu.ucdavis.ahmct.cctv/files/logs

/sdcard/Android/data/edu.ucdavis.ahmct.cctv/files/pictures

Configurations

File Format

```
[
  {
    "communicationConfigurations":[
      {
        "drop":0,
        "endpoint":{
          "type":"TCP",
          "host":"192.168.11.101",
          "port":771,
          "socketConnectionTimeout":20000,
          "socketReadTimeout":20000,
          "idleCloseTimeout":0,
          "readTimeout":16000,
          "name":"Tcp"
        },
        "name":"Ethernet via Airconsole",
        "protocol":"ONVIF"
      }
    ],
    "location":"ucdavis",
    "name":"camera 1"
  },
  {
    "communicationConfigurations":[
```

```

{
  "drop":0,
  "endpoint":{
    "type":"TCP",
    "host":"192.168.11.101",
    "port":771,
    "socketConnectionTimeout":20000,
    "socketReadTimeout":20000,
    "idleCloseTimeout":0,
    "readTimeout":16000,
    "name":"Tcp"
  },
  "name":"Ethernet via Airconsole",
  "protocol":"COHUI"
},
{
  "drop":0,
  "endpoint":{
    "type":"COMPORT",
    "baud":1200,
    "dataBits":"EIGHT",
    "flowControl":"NONE",
    "parity":"NONE",
    "purgeRx":true,
    "purgeTx":true,
    "rts":true,
    "stopBits":"ONE",
    "acknowledgementTimeout":8000,
    "host":"192.168.10.1",
    "port":3696,
    "socketConnectionTimeout":20000,

```

```

        "socketReadTimeout":20000,
        "idleCloseTimeout":0,
        "readTimeout":16000,
        "name":"ComPort"
    },
    "name":"Serial via Airconsole",
    "protocol":"PELCOD"
}
],
"location":"ucdavis",
"name":"camera 2"
}
]

```

Camera configuration object parameters

1. name - the name or description of the sign, not null
2. location - the location of the sign, not null
3. communicationConfigurations - a list of sign communication objects, not null

Camera communication configuration object parameters

1. name - the name or description of the communication configuration, not null
2. protocol - the protocol of the communication configuration, not null, valid values {COHUI, PELCOD, ONVIF}
3. endpoint - the endpoint type of the communication configuration, not null, valid values {TCP, COMPORT}
4. drop - the drop number of the communication configuration, valid values [0, inf)

TCP endpoint object parameters

1. name - the endpoint name or description (somewhat duplicative of the communication configuration name and likely to be deprecated), not null, not empty
2. idleCloseTimeout - the idle close timeout of the operations manager in (ms) where a value of 0 implies a closure after each operation, valid values [0, inf)
3. readTimeout - the read timeout of the operations manager in (ms), valid values [0, inf)
4. host - the hostname or ip address, not null, not empty
5. port - the port number, valid values [1, 65535]
6. socketConnectionTimeout - the socket connection timeout in (ms) where a value of 0 implies none, valid values [0, inf)
7. socketReadTimeout - the socket read timeout in (ms) where a value of 0 implies none, valid values [0, inf)

ComPort endpoint object parameters

1. name - the endpoint name or description (somewhat duplicative of the communication configuration name and likely to be deprecated), not null, not empty
2. idleCloseTimeout - the idle close timeout of the operations manager in (ms) where a value of 0 implies a closure after each operation, valid values [0, inf)
3. readTimeout - the read timeout of the operations manager in (ms), valid values [0, inf)
4. host - the hostname or ip address, not null, not empty
5. port - the port number, valid values [1, 65535]
6. socketConnectionTimeout - the socket connection timeout in (ms) where a value of 0 implies none, valid values [0, inf)
7. socketReadTimeout - the socket read timeout in (ms) where a value of 0 implies none, valid values [0, inf)
8. acknowledgementTimeout - the option negotiation acknowledgement timeout in (ms), valid values [0, inf)
9. baud - the baud rate to request at initialization, valid values [1, inf)

- 10.dataBits - the data bits to request at initialization, not null, valid values {FIVE, SIX, SEVEN, EIGHT}
- 11.parity - the parity to request at initialization, not null, valid values {NONE, ODD, EVEN, MARK, SPACE}
- 12.stopBits - the stop bits to request at initialization, not null, valid values {ONE, TWO, ONE_POINT_FIVE}
- 13.flowControl - the flow control to request at initialization, not null, valid values {NONE, XONXOFF, RTSCTS}
- 14.rts - whether to request that the RTS be set on at initialization, valid values {true, false}
- 15.purgeRx - whether to request that the access server receive buffer be purged at initialization, valid values {true, false}
- 16.purgeTx - whether to request that the access server transmit buffer be purged at initialization, valid values {true, false}

Chapter 5:

HHDC System Kit Hardware

The HHDC hardware is implemented as a kit which is packaged in a ruggedized case for easy field deployment and use. The primary kit components include a tablet, a Wi-Fi to serial/Ethernet device, USB to Recommended Standard (RS)-232 serial cables, USB to RS-422/485 serial cables, and an Ethernet cable. The general connectivity and role of the HHDC kit components is illustrated in Figure 5.1. Detailed discussions for the main components follow. The HHDC kit detailed hardware is listed in Table 5.1.

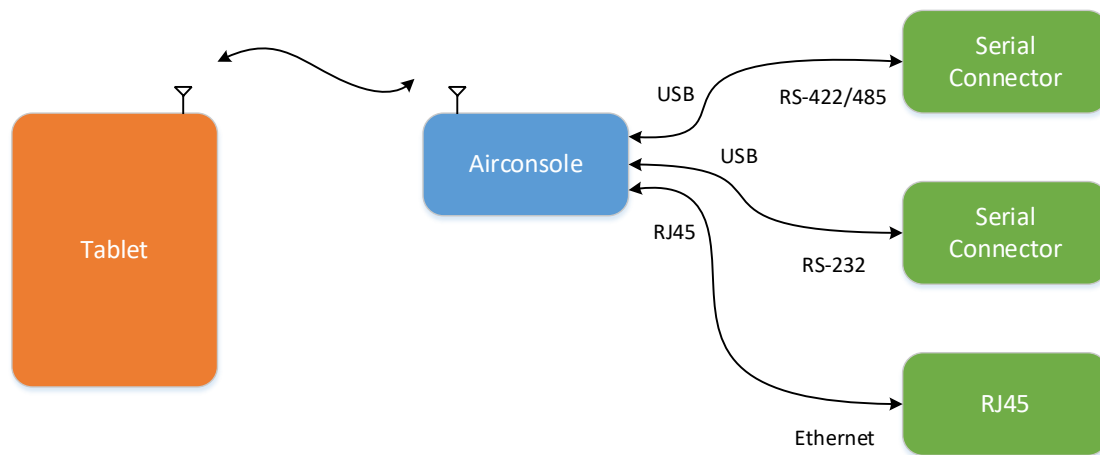


Figure 5.1: General connectivity and role of the HHDC kit components

Table 5.1: HHDC kit detailed hardware list

Component	Component
Pelican case	Registered Jack (RJ)45 to DB9F straight-through Adapter
Tablet	RJ45 to DB9F cross-over adapter
Airconsole XL	Ethernet cable
USB 170 C2 (RS-232) cable	Cross-over Ethernet adapter
USB Serial (RS-232/RJ45) cable	Chargers
USB to DE9M RS-232 cable	USB D2 CCTV (RS-422/RJ45) cable

Component	Component
DB9M to DB25F RS-232 adapter	USB D6 CCTV (RS-422/DB9M) cable
DB9M to DB25M RS-232 adapter	Bayonet Neill–Concelman (BNC) Cable
DB9M to DB9M adapter	National Television System Committee (NTSC) display
RS-232 to RS422 adapter	

Equipment

Figure 5.2 shows the complete HHDC kit when first opened. The Samsung tablet is used for running the developed applications, and the NTSC monitor is used for analog camera verification purposes.



Figure 5.2: HHDC kit in carrying case

Figure 5.3 shows the HHDC kit with the first layer of equipment removed. Below the tablet are two boxes of cables, chargers, and adapters. Below the monitor are the monitor charger, network cabling, and the interface cable to a standard 170 controller.



Figure 5.3: HHDC kit with tablet and monitor removed

Figure 5.4 shows the HHDC kit with all layers of equipment removed. This figure shows the two cable and adapter storage box contents.



Figure 5.4: HHDC kit full contents

Figure 5.5 shows the HHDC kit Samsung tablet with high brightness screen running the DMS application.



Figure 5.5: HHDC kit tablet

Figure 5.6 shows the HHDC kit Airconsole adapter used as the primary communications channel between tablet and field equipment cabinets. The Airconsole provides a Wi-Fi interface for the tablet to connect and Ethernet, USB serial, and Bluetooth interfaces.



Figure 5.6: HHDC kit Airconsole

Figure 5.7 shows the HHDC kit Airconsole connected to an Ethernet connection for direct interfacing with network-capable equipment hardware.



Figure 5.7: HHDC Airconsole connected to Ethernet

Figure 5.8 shows the HHDC kit Airconsole connected to a USB serial cable. Figures 5.9-5.11 show supported cables. The kit includes both RS-232 and RS-422 USB serial cables interfaced to various interfacing connectors.



Figure 5.8: HHDC Airconsole connected to USB serial

Figure 5.9 shows the HHDC kit USB serial RS-232 to 170-C2 connector cable used to connect the Airconsole directly to a standard 170 with serial C2 port.

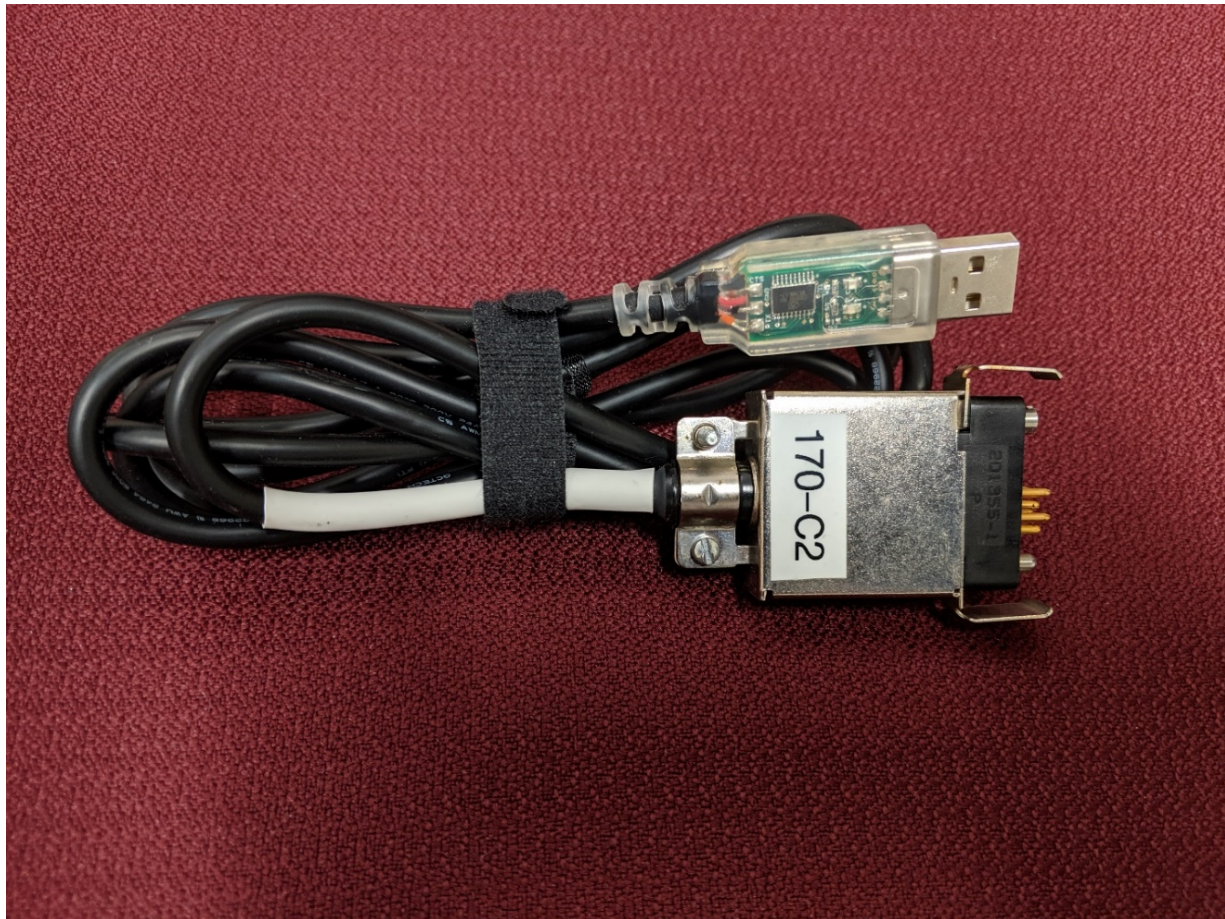


Figure 5.9: HHDC kit USB serial RS-232 to 170-C2 connector cable

Figure 5.10 shows the HHDC kit USB serial RS-422 to D2 CCTV RJ45 connector cable used to connect the Airconsole directly to a standard D2 camera PTZ controller.



Figure 5.10: HHDC kit USB serial RS-422 to D2 CCTV RJ45 connector

Figure 5.11 shows the HHDC kit USB serial RS-422 to D6 CCTV DE-9 connector cable used to connect the Airconsole directly to a standard D6 camera PTZ controller.



Figure 5.11: HHDC kit USB serial RS-422 to D6 CCTV DE-9 connector

Chapter 6:

HHDC Console Software

This chapter discusses the selection of the appropriate Console app for inclusion in the HHDC. The requirements for this app are provided in Appendix E.

Overview

As part of this research, AHMCT evaluated various COTS console apps for inclusion in the HHDC app suite. AHMCT assessed available features vs. anticipated Caltrans needs. In conjunction with Caltrans, AHMCT selected the most appropriate serial console app, and included it in the HHDC app suite. The Console app provides a general command-line interface (CLI) for interfacing with a wide range of field element hardware. The Console app can be used to connect to devices, send commands via CLI, and display resulting text-based field element response in the console screen.

The Console app can access general Caltrans field elements remotely over the network, or directly at the field location. Local connection from the HHDC to the field element is supported by Wi-Fi-to-network, USB-to-network, or Wi-Fi-to-serial adapters. Remote network connection is typically over standard tablet Wi-Fi or cellular connection. The selected Console app supports Secure Shell (SSH) and Telnet through the local CLI with multiple command history, copy and paste, macros, external keyboard support, and connection management, among other powerful features.

Comparison

Six popular and capable console apps were evaluated against the requirements of Appendix E. The evaluation results are presented in Table 5.1. Based on the requirements and a head-to-head comparison, the final app selected for inclusion in the HHDC is JuiceSSH Pro. The app is installed on each of the HHDC kit tablets. For this evaluation, features were deemed much more important than cost. The app is low-cost in the context of HHDC, and in the general app realm.

Table 6.1: Console app feature matrix

		Console Apps					
		JuiceSSH Pro	Better Terminal Emulator	Terminal IDE	Terminal Emulator	ConnectBot	SerialBot
Features	Android 5.0 Compatible	Yes	Yes	No	Yes	Yes	Yes
	Local Shell	Yes	Yes	Yes	Yes	Yes	Yes
	SSH	Yes	Yes	Yes	No	Yes	Yes
	Mosh	Yes	No	No	No	No	No
	Telnet	Yes	Yes	Yes	No	Yes	Yes
	Command History	Yes	Yes	Yes	Yes	Yes	Yes
	Copy and Paste	Yes	Yes	Yes	No	Yes	Yes
	Connection Management	Yes	?	Yes	No	No	No
	Macros	Yes	Yes	Yes	Yes	No	No
	Extensible	Yes	?	Yes	Yes	No	No
	External Keyboards	Yes	Yes	Yes	Yes	Yes	Yes
	Network (Wi-Fi / USB)	Yes	Yes	Yes	Yes	Yes	Yes
	Serial over Network	Yes	Yes	Yes	No	Yes	Yes
	Cost	\$6.89	\$3.99	Free	Free	Free	Free

Final Selection

The final selection for the HHDC Console app is the COTS app JuiceSSH Pro, available on the Google Play Store. The selection was based on the requirements of Appendix E, as well as more subjective criteria agreed upon between AHMCT and Caltrans. Several screen shots for the Console app views are shown in Figures 6.1 – 6.4. While this app is currently the best match for requirements and subjective criteria, apps are regularly being upgraded, and new apps are introduced often. It is in Caltrans' interest to monitor the state-of-practice in Android console apps, and consider replacing the existing Console app, or adding additional choices for the user. On the other hand, familiarity with a specific Console app's interface and workflow is quite important, and should not be discounted when considering a switch.

Figure 6.1 shows the Console app's Command-Line View. This is the standard CLI for interacting with devices.

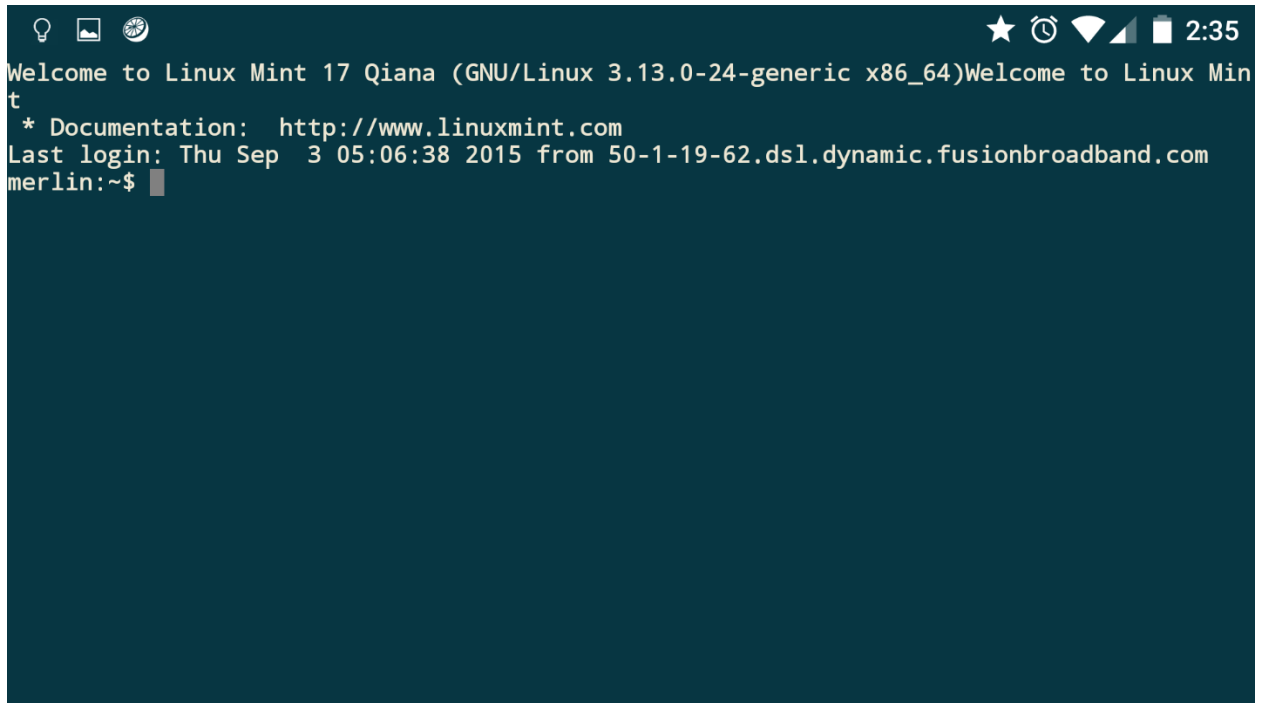


Figure 6.1: Console app Command-Line View (courtesy of Sonelli Ltd.)

Figure 6.2 shows the Console app's Keyboard View. This is the standard Android popup keyboard allowing the user to type commands into the CLI.

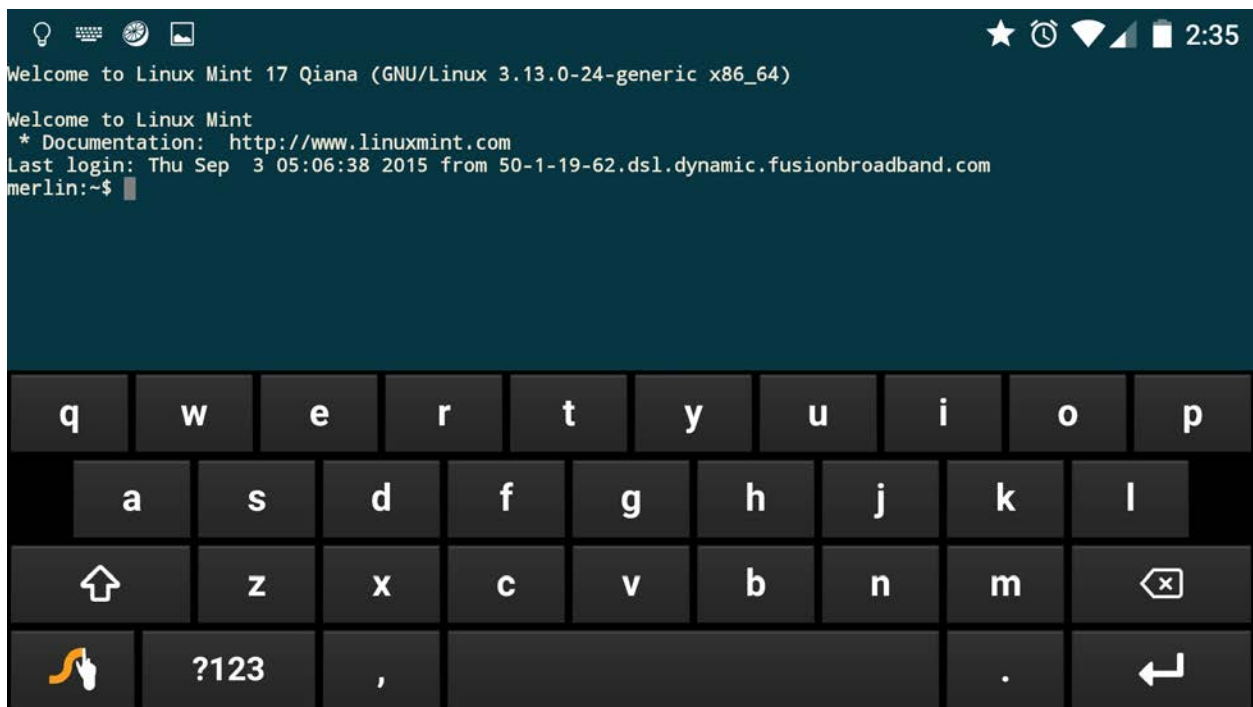


Figure 6.2: Console app Keyboard View (courtesy of Sonelli Ltd.)

Figure 6.3 shows the Console app's Connection View. In this view, the user can check connection status with the given ITS field element. In this figure, one connection has been created for a host nicknamed Merlin.

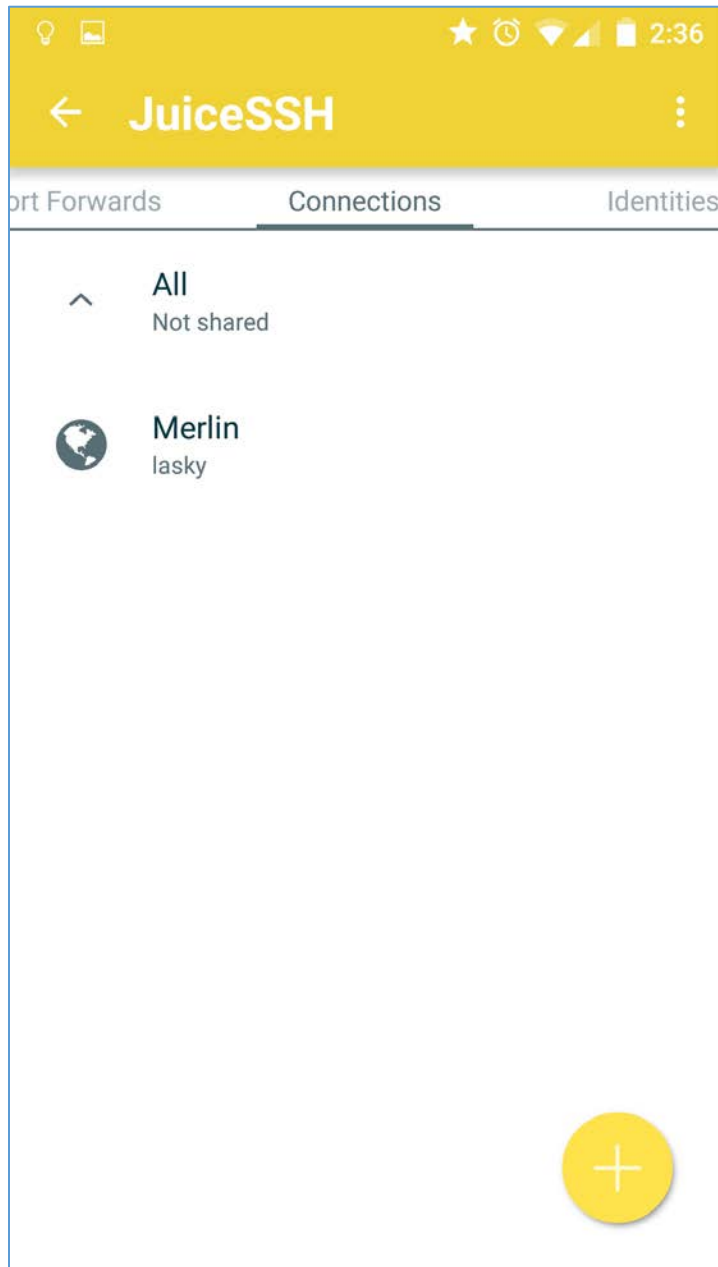


Figure 6.3: Console app Connection View (courtesy of Sonelli Ltd.)

Figure 6.4 shows the Console app's Configuration View. This allows the user to change configuration options for connecting to an ITS field element. In this figure Merlin's basic settings are being shown, and can be changed. Key settings include nickname, type (e.g. SSH or Telnet), host address, user identity (if any), and port (default SSH port is 22).

Update Connection

BASIC SETTINGS

Nickname:

Type:

Address:

Identity:

ADVANCED SETTINGS

Port:

Connect Via:

Run Snippet:

Backspace:

GROUPS

ADD TO GROUP

Figure 6.4: Console app Configuration View (courtesy of Sonelli Ltd.)

Chapter 7:

Conclusions and Future Research

Key contributions of this research project include:

- Development and testing of the HHDC DMS app
- Development and testing of the HHDC CCTV app
- Selection and testing of the HHDC console app
- Development and testing of the HHDC hardware kit
- Enabling of HHDC for two key Caltrans ITS field elements, DMS and CCTV
- Proof of the benefits of the HHDC approach for system diagnostics and control

The original work plan included two rounds of testing, i.e. alpha and beta testing. This was expected to occur in Districts 2, 3, 4, 6, and 10. Due to delays in the development cycle, this testing was curtailed. AHMCT performed internal testing of the apps and the kit. Additional limited testing was performed by the Caltrans PM.

The HHDC development yielded numerous lessons learned. The software development, demonstration, and delivery took substantially longer than originally estimated. The researchers and Caltrans project management have discussed this issue, and have identified key issues for future development. The most significant issue was a mindset which focused on handling edge case scenarios, i.e. situations that could arise on a relatively rare basis and which would adversely affect HHDC performance or use. The researchers and Caltrans management agree that it will in the future be better to focus on providing core functionality early and on schedule. In this way, a system with perhaps 80% functionality could be tested and demonstrated. This reduces the risk of long development delays. It would also provide the researchers and Caltrans with a working system to test, facilitating further debugging and development, as well as test-based refinement of functionality, features, and user interface. This is a fundamental principle for software engineering, yet there does seem to be a need to reinforce it, as the urge for perfection is strong in typical researchers, engineers, and software developers.

The likelihood of HHDC deployment in its current specific implementation is now low. Based on organizational and technical aspects, the Android operating system was selected for the HHDC implementation. At the time of this decision, this was the clear choice. However, since this decision, Caltrans policy now

dictates that all smartphone and tablet devices will be Apple iOS-based. While this is the current policy, due to procurement timing, Caltrans has also fielded a large number of Microsoft Windows-based ToughBooks for Maintenance field use. The net result is that there is currently no place within Caltrans in the foreseeable future for a system implemented on Android.

Future work may include:

- Support for further HHDC testing at the district user level
- Design and implementation of similar HHDC apps for additional ITS field elements including, but not limited to:
 - VDS
 - RWIS
 - Microwave VDS
 - Ramp metering systems
- Porting existing development to iOS

Given the above note regarding Caltrans and Android, it is feasible for Caltrans to continue testing the HHDC to assess the functional benefit. It may also allow a more detailed assessment of the HHDC user interface. However, for any full-scale deployment of the HHDC concept, the system will need to be ported (iOS) or completely rewritten (Windows) for a platform currently supported within Caltrans.

References

- [1] Stephen Donecker, Travis Swanston, Kin Yen, Bahram Ravani, and Ty A. Lasky, "A Handheld Terminal for Field Elements," AHMCT Research Center, UCD-ARR-15-09-30-03, Sep. 2015.
- [2] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

Appendix A:

Glossary of Common Terms and Acronyms for Requirements

Term or Acronym	Definition
3G	3 rd generation of mobile telecommunications technology
4G	4 th generation of mobile telecommunications technology
4G	4 th generation of mobile telecommunications technology
AC	Alternating Current
ADDCO	A portable changeable message sign control protocol
Android	A mobile operating system based on the Linux kernel developed by Google
App	A mobile OS software application
AVMS	Advanced Variable Message Sign
BNC	A quick connect/disconnect radio frequency connector (Bayonet Neill–Concelman) used for coaxial cable
CCTV	Closed Circuit Television
CMS	Changeable Message Sign
Cohu	A PTZ control protocol
DC	Direct Current
DE-9	A D-subminiature 9-pin connector commonly used for serial communications
DMS	Dynamic Message Sign
Field Element	An Intelligent Transportation System sensor or display device
Fps	Frames Per Second
Gesture	Touch patterns used to provide input to mobile applications
H.264	MPEG-4 Part 10, Advanced Video Coding
HHDC	Handheld Diagnostic Controller
IEEE	Institute of Electrical and Electronics Engineers

Term or Acronym	Definition
IEEE 803.11a	54 Mbps max Wi-Fi at 5.8 GHz
IEEE 803.11ac	1300 Mbps max MIMO Wi-Fi using 3.4 GHz and 5.8 GHz
IEEE 803.11b	11 Mbps max Wi-Fi at 3.4 GHz
IEEE 803.11g	54 Mbps max Wi-Fi at 3.4 GHz
IEEE 803.11n	600 Mbps max MIMO Wi-Fi using 3.4 GHz and 5.8 GHz
IP Camera	Internet Protocol camera
ITS	Intelligent Transportation System
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LTE	Long-Term Evolution, an implementation of 4G
Micro-USB	A USB Micro-B type plug for connecting to a USB On-The-Go device
MIMO	Multiple Input Multiple Output
Motion JPEG	A video compression format in which each frame of digital video is compressed separately as a JPEG image
MPEG-4	A video encoding standard (Moving Picture Experts Group)
NTCIP	National Transportation Communications for Intelligent Transportation Systems Protocol
NTSC	National Television System Committee
OS	Operating System
OTG	On-The-Go
Pelco D	A PTZ control protocol
POE	Power Over Ethernet
PTZ	Pan Tilt Zoom
RCA	A connector designed by the Radio Corporation of America
RJ45	A 8-position 8-contact connector used for Ethernet over twisted pair cable
RoHS	Reduction of Hazardous Substances
RS	Recommended Standard
RS-232	A serial communication standard using singled-ended signaling over a short distance connection
RS-422	A serial communication standard using differential signaling over a long distance connection
RS-485	A serial communication standard similar to RS-422 with the addition of support for true multi-point connections

Term or Acronym	Definition
SignView	California DOT proprietary DMS/CMS control protocol
Sony Visca	A PTZ control protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
Wi-Fi	A wireless local area network based on IEEE 803.11 standards
Wi-Fi Access Point	A device that allows wireless devices based on IEEE 803.11 standards to connect to a wired network

Appendix B: Handheld Diagnostic Controller Kit System Requirements

Version 1.0
September 30, 2015

AHMCT Research Center
University of California - Davis

The following are the system requirement for the Kit portion of the Handheld Diagnostic Controller system.

1. Kit

- 1.1. The kit shall be used to diagnose and configure CCTV field elements
- 1.2. The kit shall be used to diagnose and configure DMS field elements
- 1.3. The kit shall be used to diagnose and configure generic field elements with a serial interface
- 1.4. The kit shall support use locally at the field element
- 1.5. The kit shall support local diagnosis and configuration of the CCTV field element as a whole
- 1.6. The kit shall support local diagnosis and configuration of the DMS field element as a whole
- 1.7. The kit shall support local diagnosis and configuration of the CCTV field element sub-systems individually
- 1.8. The kit shall support local diagnosis and configuration of the DMS field element sub-systems individually
- 1.9. The kit shall support use remotely from field element
- 1.10. The kit shall support use remotely inside field element VPN network
- 1.11. The kit may support use remotely outside field element VPN network
- 1.12. The kit shall support remote diagnosis and configuration of the CCTV field element as a whole
- 1.13. The kit shall support remote diagnosis and configuration of the DMS field element as a whole
- 1.14. The kit shall be composed of a case housing various hardware, software, and cabling
- 1.15. The kit shall include a single storage case
- 1.16. The kit shall include a tablet
- 1.17. The kit shall include all necessary interface cabling
- 1.18. The kit shall include a Wi-Fi access point
- 1.19. The kit may include a terminal server
- 1.20. The kit may include a video encoder
- 1.21. The kit shall include a video monitor
- 1.22. The kit shall include software applications
- 1.23. The kit shall be portable by a single person
- 1.24. The kit shall weigh less than 20 lb
- 1.25. The kit shall be easily deployed
- 1.26. The kit shall be easily stored in a vehicle

- 1.27. The kit shall have an operational temperate range of 0° to 70° C
- 1.28. The kit may have an operational temperature range of -20° to 70° C
- 1.29. The kit shall have a storage temperature range of -40° to 70° C

2. Storage Case

- 2.1. The storage case shall be portable
- 2.2. The storage case shall have a form factor similar to a briefcase
- 2.3. The storage case shall weigh less than less than 10 lb
- 2.4. The storage case shall have an interior volume of at least 0.5 cubic feet
- 2.5. The storage case shall be watertight
- 2.6. The storage case shall be dustproof
- 2.7. The storage case shall be crushproof
- 2.8. The storage case shall be impact resistant
- 2.9. The storage case shall be abrasion resistant
- 2.10. The storage case shall have form-fitting compartments to hold contents
- 2.11. The storage case shall be foam filled
- 2.12. The storage case shall allow end user customization of foam
- 2.13. The storage case shall be storable over -40° to 70° C

3. Tablet

- 3.1. The tablet shall be portable
- 3.2. The tablet shall be lightweight
- 3.3. The tablet shall be high resolution
- 3.4. The tablet shall include a display greater than 7 inches diagonal
- 3.5. The tablet shall be operable in typical outdoor lighting conditions
- 3.6. The tablet shall be drop resistant
- 3.7. The tablet shall be impact resistant
- 3.8. The tablet may use a protective enclosure
- 3.9. The tablet shall include Wi-Fi
- 3.10. The tablet shall support 3.4 GHz Wi-Fi band
- 3.11. The tablet may supports 5.8 GHz Wi-Fi band
- 3.12. The tablet shall support 803.11b/g/n wireless
- 3.13. The tablet may support 803.11a wireless
- 3.14. The tablet may support 803.11ac wireless

- 3.15. The tablet shall have a runtime of at least 9 hours between charges
- 3.16. The tablet shall support the Android operating system
- 3.17. The tablet shall have a minimum Android OS version of 4.1
- 3.18. The tablet shall have USB networking compiled into the OS
- 3.19. The tablet shall be chargeable using a vehicle DC power outlet
- 3.20. The tablet shall be chargeable using an AC power outlet
- 3.21. The tablet shall include a camera
- 3.22. The tablet shall include a Micro-USB 3.0 port
- 3.23. The tablet shall be operable from 0° to 50° C
- 3.24. The tablet may be operable from -30° to 50° C for limited periods of time
- 3.25. The tablet shall be storable from -40° to 70° C

4. Cabling

- 4.1. The cabling shall include one Micro-USB B to USB A Female USB adapter pigtail
- 4.2. The cabling shall include one USB A Male to RJ45 Ethernet adapter pigtail
- 4.3. The cabling may include one USB A Male to RS232 DB-9 Female serial cable at least 3 ft long
- 4.4. The cabling may include one RS232 DE-9 Male to RS422/RS485 DE-9 Male adapter
- 4.5. The cabling may include one USB A Male to RS422/485 DE-9 Female serial cable at least 3 ft long
- 4.6. The cabling shall include one BNC Male to BNC Male coaxial cable at least 6 ft long
- 4.7. The cabling may include one DE-9 Male to 170-C2 Male serial cable at least 6 ft long
- 4.8. The cabling shall include one Ethernet cable at least 6 ft long
- 4.9. The cabling shall include one DE-9 gender changer
- 4.10. The cabling shall be RoHS compliant
- 4.11. The cabling shall be operable from -30° to 50°C
- 4.12. The cabling shall be storable from -40° to 70°C

5. Wi-Fi Access Point

- 5.1. The access point shall support 3.4 GHz band
- 5.2. The access point may support 5.8 GHz band
- 5.3. The access point shall support 803.11b/g/n wireless
- 5.4. The access point may support 803.11a
- 5.5. The access point may support 803.11ac
- 5.6. The access point shall be easy to configure

- 5.7. The access point shall have a minimum of one Ethernet port
- 5.8. The access point shall support RS232
- 5.9. The access point may support RS422/RS485
- 5.10. The access point may be powered by an AC-to-DC converter
- 5.11. The access point shall be operable from -30° to 50° C
- 5.12. The access point shall be storable from -40° to 70° C
- 5.13. The access point shall support wireless-to-wired LAN bridging
- 6. Terminal Server
 - 6.1. The terminal server shall have a minimum of one serial port
 - 6.2. The terminal server serial port shall be DB-9 Male
 - 6.3. The terminal server shall support RS232
 - 6.4. The terminal server shall support RS422/RS485
 - 6.5. The terminal server shall be easily configurable
 - 6.6. The terminal server may be powered by an AC-to-DC converter
 - 6.7. The terminal server shall be operable from 0° to 50° C
 - 6.8. The terminal server may be operable from -30° to 50° C
 - 6.9. The terminal server shall be storable from -40° to 70° C
- 7. Video Encoder
 - 7.1. The video encoder shall be easy to configure
 - 7.2. The video encoder shall have a minimum of one analog input
 - 7.3. The video encoder shall have a BNC Female input connector
 - 7.4. The video encoder shall support NTSC resolutions 720x480 to 176x120
 - 7.5. The video encoder shall support a minimum frame rate of 30 fps in all resolutions
 - 7.6. The video encoder shall support H.264 video compression
 - 7.7. The video encoder shall support Motion JPEG video compression
 - 7.8. The video encoder shall support PTZ of analog cameras
 - 7.9. The video encoder may support RS422/RS485
 - 7.10. The video encoder may support POE
 - 7.11. The video encoder may be powered by an AC-to-DC converter
 - 7.12. The video encoder shall be operable from -30° to 50° C
 - 7.13. The video encoder shall be storable from -40° to 70° C

8. Video Monitor

- 8.1. The video monitor shall support analog video input
- 8.2. The video monitor may support NTSC resolutions 720x480 to 176x120
- 8.3. The video monitor shall be compact
- 8.4. The video monitor shall be usable in typical outdoor lighting conditions
- 8.5. The video monitor shall have a BNC Female input connector
- 8.6. The video monitor may have an RCA Female input connector
- 8.7. The video monitor may be battery powered
- 8.8. The video monitor may be powered by an AC-to-DC converter
- 8.9. The video monitor shall be operable from 0° to 50° C
- 8.10. The video monitor may be operable from -30° to 50° C
- 8.11. The video monitor shall be storable from -40° to 70° C

9. Software

- 9.1. The software shall be pre-installed on the tablet
- 9.2. The software shall include a CCTV application
- 9.3. The software shall include a DMS application
- 9.4. The software shall include a Serial Console application
- 9.5. The software may include a VPN application

Appendix C: Handheld Diagnostic Controller DMS System Requirements

Version 1.0
September 30, 2015

AHMCT Research Center
University of California - Davis

The following are the system requirement for the DMS portion of the Handheld Diagnostic Controller system.

1. Use

- 1.1. The app shall be used to diagnose and configure DMS field elements
- 1.2. The app shall support use locally at the field element
- 1.3. The app shall support local diagnosis and configuration of the DMS field element as a whole
- 1.4. The app shall support local diagnosis and configuration of the DMS field element sub-systems individually
- 1.5. The app shall support use remotely from field element
- 1.6. The app shall support use remotely inside field element VPN network
- 1.7. The app may support use remotely outside field element VPN network
- 1.8. The app shall support remote diagnosis and configuration of the DMS field element as a whole
- 1.9. The app shall support local diagnosis of DMS field element in a wireless fashion
- 1.10. The app shall support local diagnosis of DMS field element in a wired fashion
- 1.11. The app shall be intuitive to use
- 1.12. The app shall be easy to configure

2. Operating System

- 2.1. The app shall run on the Android OS
- 2.2. The app shall be compatible with Android versions 4.1 and up
- 2.3. The app shall run on Android tablets
- 2.4. The app shall run on Android phones

3. Protocols

- 3.1. The app shall implement the SignView control protocol
- 3.2. The app shall implement the NTCIP control protocol as implemented in the Caltrans AVMS
- 3.3. The app may implement the ADDCO control protocol
- 3.4. The app shall use modular drivers to implement control protocols

4. Communications

- 4.1. The app shall support TCP/IP communications over Wi-Fi
- 4.2. The app shall support TCP/IP communications over 3G/4G/LTE cellular networks
- 4.3. The app shall support TCP/IP communications over USB
- 4.4. The app shall support Serial communications over USB

5. User Interface

- 5.1. The user interface shall include a DMS view
- 5.2. The user interface shall include a DMS configuration view
- 5.3. The user interface shall include a log view
- 5.4. The user interface shall include a diagnostics view
- 5.5. The user interface shall include a photo view
- 5.6. The user interface shall include an intuitive mechanism for switching between views
- 5.7. The user interface shall support tap gestures
- 5.8. The user interface shall support swipe gestures

6. DMS View

- 6.1. The DMS view shall display the state of the currently selected sign
- 6.2. The DMS view may have an icon to select a sign and configuration
- 6.3. The DMS view may have a spinner to select a sign and configuration
- 6.4. The DMS view shall have an icon to take a snapshot of the current view
- 6.5. The DMS view may support the use of swipe gestures to control the view of multiple sign pages
- 6.6. The DMS view shall have a library of messages
- 6.7. The DMS view additional options icon shall support the import of library messages
- 6.8. The DMS view shall have the ability to enter custom messages
- 6.9. The DMS view may have the ability to set brightness
- 6.10. The DMS view may have the ability to set color
- 6.11. The DMS view may have the ability to set timing between pages
- 6.12. The DMS view may have the ability to get current fonts
- 6.13. The DMS view may have the ability to set current fonts
- 6.14. The DMS view shall support getting the sign's current state
- 6.15. The DMS view shall support setting the sign's state
- 6.16. The DMS view shall display no sign selected state
- 6.17. The DMS view shall display updating sign state

7. DMS Configuration View

- 7.1. The sign view shall support the addition and selection of signs
- 7.2. The sign view shall include an add sign icon
- 7.3. The sign view shall include a search for sign icon

- 7.4. The sign view may include an additional options menu
- 7.5. The sign view additional options menu may support sign list display by criteria
- 7.6. The sign view additional options menu shall support deletion of a sign
- 7.7. The sign view additional options menu shall support the import/export of signs
- 7.8. The sign view shall display a list of currently defined signs
- 7.9. The sign view shall support the definition of a tile next to the sign names
- 7.10. The sign view may support the automatic definition of the tile next to the sign names
- 7.11. The sign view shall support the use of swipe gestures to view a list of signs
- 7.12. The sign view shall support the use of tap gestures to select a sign in the list
- 7.13. The sign view shall open the sign configurations view when a sign is selected
- 7.14. The sign view add sign icon shall open the sign configuration view
- 7.15. The sign configurations view shall include an edit configuration icon
- 7.16. The sign configurations view shall include an add configuration icon
- 7.17. The sign configurations view shall include an additional options icon
- 7.18. The sign configurations view additional options menu may support deletion of a sign configuration
- 7.19. The sign configurations view additional options menu may support deletion of all sign configurations
- 7.20. The sign configurations view shall include an edit sign configuration icon
- 7.21. The sign configurations view shall display a list of currently defined sign configurations
- 7.22. The sign configurations view shall support the use of swipe gestures to view a list of sign configurations
- 7.23. The sign configurations view shall support the use of tap gestures to select a sign configuration
- 7.24. The sign configurations view may open the DMS view when a sign configuration is selected
- 7.25. The sign configurations view shall open the sign configuration view when the edit icon is selected
- 7.26. The sign configuration view shall include an additional options icon
- 7.27. The sign configuration view additional options menu shall support discarding current changes
- 7.28. The sign configuration view additional options menu may support deletion of the current configuration
- 7.29. The sign configuration view shall include a settable configuration name

- 7.30. The sign configuration view shall include a settable IP address
- 7.31. The sign configuration view shall include a settable port
- 7.32. The sign configuration view shall include a settable serial configuration
- 7.33. The sign configuration view shall include a settable protocol

8. Log View

- 8.1. The log view shall support all application logging
- 8.2. The log view shall include a spinner for selecting log level filtering
- 8.3. The log view shall include an additional options icon
- 8.4. The log view additional options menu shall support enabling logging to file
- 8.5. The log view additional options menu shall support setting file logging level
- 8.6. The log view additional options menu shall support setting maximum log file size
- 8.7. The log view additional options menu shall support setting the number of rotating log files
- 8.8. The log view shall display a detailed timestamp
- 8.9. The log view shall display a log level
- 8.10. The log view shall display the module generating the log entry
- 8.11. The log view shall display the log entry
- 8.12. The log view shall support the use of swipe gestures to view the log

9. Diagnostics View

- 9.1. The diagnostics view shall support advanced diagnostics of signs
- 9.2. The diagnostics view shall include a sign and configuration selection icon
- 9.3. The diagnostics view shall include a snapshot icon for capturing the currently displayed view
- 9.4. The diagnostics view shall include a camera icon for capturing the currently displayed test results on the sign
- 9.5. The diagnostics view shall include an additional options icon
- 9.6. The diagnostics view additional options icon shall support importing diagnostic routines
- 9.7. The diagnostics view shall include a spinner containing all diagnostic routines
- 9.8. The diagnostics view shall include a begin routine button

10. Diagnostic Routines

- 10.1. The routine library may include a full on pattern
- 10.2. The routine library shall include a full off pattern
- 10.3. The routine library may include a flashing full on-off pattern
- 10.4. The routine library shall include a checkerboard pattern

- 10.5. The routine library may include a color bar pattern
- 10.6. The routine library shall include all symbols in the selected font set
- 10.7. The routine library shall exercise all pixels
- 10.8. The routine library shall exercise all states of pixels

11. Photo View

- 11.1. The photo view shall support viewing and managing app snapshots and pictures
- 11.2. The photo view shall support use of swipe gestures
- 11.3. The photo view shall support use of tap gestures
- 11.4. The photo view shall support deletion of a snapshot
- 11.5. The photo view shall support sorting pictures by timestamp
- 11.6. The photo view shall support sorting pictures by sign

Appendix D: Handheld Diagnostic Controller CCTV System Requirements

Version 1.0
September 30, 2015

AHMCT Research Center
University of California - Davis

The following are the system requirement for the CCTV portion of the Handheld Diagnostic Controller system.

1. Use

- 1.1. The app shall be used to diagnose and configure CCTV field elements
- 1.2. The app shall support use locally at the field element
- 1.3. The app shall support local diagnosis and configuration of the CCTV field element as a whole
- 1.4. The app shall support local diagnosis and configuration of the CCTV field element sub-systems individually
- 1.5. The app shall support use remotely from field element
- 1.6. The app shall support use remotely inside field element VPN network
- 1.7. The app may support use remotely outside field element VPN network
- 1.8. The app shall support remote diagnosis and configuration of the CCTV field element as a whole
- 1.9. The app shall support local diagnosis and configuration of the analog camera separately from the CCTV field element as a whole
- 1.10. The app shall support local diagnosis and configuration of the video encoder and camera as a unit separately from the CCTV field element as a whole
- 1.11. The app shall support local diagnosis and configuration of the PTZ controller separately from the CCTV field element as a whole
- 1.12. The app shall support local diagnosis and configuration of the PTZ controller and terminal server as a unit separately from the CCTV field element as a whole
- 1.13. The app shall support local diagnosis and configuration of digital cameras
- 1.14. The app shall support local diagnosis and configuration of analog cameras
- 1.15. The app shall support local diagnosis and configuration of cameras with integrated PTZ
- 1.16. The app shall support local diagnosis of CCTV field element sub-systems in a wireless fashion
- 1.17. The app shall support local diagnosis of CCTV field element sub-systems in a wired fashion
- 1.18. The app shall be intuitive to use
- 1.19. The app shall be easy to configure

2. Operating System

- 2.1. The app shall run on the Android OS
- 2.2. The app shall be compatible with Android versions 4.1 and up
- 2.3. The app shall run on Android tablets

- 2.4. The app shall run on Android phones
- 3. Protocols
 - 3.1. The app shall decode H.264 encoded video
 - 3.2. The app shall decode MPEG-4 encoded video
 - 3.3. The app shall decode Motion JPEG encoded video
 - 3.4. The app shall implement the Cohu-I series v6.8 PTZ control protocol
 - 3.5. The app shall implement the Pelco-D v5.3.7 PTZ control protocol
 - 3.6. The app shall implement the Sony Visca PTZ control protocol
 - 3.7. The app shall use modular drivers to implement PTZ protocols
 - 3.8. The app shall use modular drivers to implement video protocols
- 4. Communications
 - 4.1. The app shall support TCP/IP communications over Wi-Fi
 - 4.2. The app shall support TCP/IP communications over 3G/4G/LTE cellular networks
 - 4.3. The app shall support TCP/IP communications over USB
 - 4.4. The app shall support Serial communications over USB
- 5. User Interface
 - 5.1. The user interface shall include a video view
 - 5.2. The user interface shall include a camera configuration view
 - 5.3. The user interface shall include a log view
 - 5.4. The user interface shall include a diagnostics view
 - 5.5. The user interface shall include a photo view
 - 5.6. The user interface shall include an intuitive mechanism for switching between views
 - 5.7. The user interface shall support tap gestures
 - 5.8. The user interface may support double-tap gestures
 - 5.9. The user interface shall support swipe gestures
 - 5.10. The user interface shall support pinch gestures
 - 5.11. The user interface shall support double-tap-slide gestures
- 6. Video View
 - 6.1. The video view shall display the video output for the currently selected camera
 - 6.2. The video view may have an icon to select the various cameras from the video view
 - 6.3. The video view may have a spinner to select a camera and configuration from the video view
 - 6.4. The video view shall have an icon to take a snapshot of the current video view

- 6.5. The video view shall support the use of swipe gestures to control the pan-tilt of the camera
- 6.6. The video view shall support the use of pinch gestures to control the zoom of the camera
- 6.7. The video view shall support the use of double-tap-slide gestures to control the zoom of the camera
- 6.8. The video view may support the use of tap/double-tap gestures to select a point in the video view and pan-tilt to the appropriate location
- 6.9. The video view shall support the use of tap gestures in the center of the video view to view video in full screen mode
- 6.10. The video view may support the use of swipe gestures from the top of the video view to return to normal viewing size with icons
- 6.11. The video view may support the use of double-tap gestures in the center of the video view to return to normal viewing size with icons
- 6.12. The video view shall display no camera selected state
- 6.13. The video view shall display connecting to camera video stream state
- 6.14. The video view shall support an additional options icon
- 6.15. The settings shall support overlay support of camera configuration and performance
- 6.16. The video view shall support a camera control keypad
- 6.17. The camera control keypad shall be a semitransparent overlay of the video view
- 6.18. The camera control keypad may be selected from the video view by icon
- 6.19. The camera control keypad may be selected by the use of swipe gestures from the bottom of the video view
- 6.20. The camera control keypad shall include arrows to control the pan-tilt of the camera
- 6.21. The camera control keypad shall include buttons to control the zoom of the camera
- 6.22. The camera control keypad shall include buttons to set camera presets
- 6.23. The camera control keypad shall include buttons to clear camera presets
- 6.24. The camera control keypad shall include buttons to go to camera presets
- 6.25. The camera control keypad shall include buttons to control the focus of the camera
- 6.26. The camera control keypad shall include buttons to control the iris of the camera
- 6.27. The camera control keypad shall include a mechanism to set various camera video overlays
- 7. Camera Configuration View
 - 7.1. The camera view shall support the addition and selection of cameras
 - 7.2. The camera view shall include an add camera icon

- 7.3. The camera view shall include a search for camera icon
- 7.4. The camera view may include an additional options icon
- 7.5. The camera view additional options menu may support camera list display by criteria
- 7.6. The camera view additional options menu shall support deletion of a camera
- 7.7. The camera view additional options menu shall support the import/export of cameras
- 7.8. The camera view shall display a list of currently defined cameras
- 7.9. The camera view shall support the definition of a tile next to the camera names
- 7.10. The camera view may support the automatic definition of the tile next to the camera names
- 7.11. The camera view shall support the use of swipe gestures to view a list of cameras
- 7.12. The camera view shall support the use of tap gestures to select a camera in the list
- 7.13. The camera view shall open the camera configurations view when a camera is selected
- 7.14. The camera view add camera icon shall open the camera configuration view
- 7.15. The camera configurations view shall include an edit configuration icon
- 7.16. The camera configurations view shall include an add configuration icon
- 7.17. The camera configurations view shall include an additional options icon
- 7.18. The camera configurations view additional options menu may support deletion of a camera configuration
- 7.19. The camera configurations view additional options menu may support deletion of all camera configurations
- 7.20. The camera configurations view shall include an edit camera configuration icon
- 7.21. The camera configurations view shall display a list of currently defined camera configurations
- 7.22. The camera configurations view shall support the use of swipe gestures to view a list of camera configurations
- 7.23. The camera configurations view shall support the use of tap gestures to select a camera configuration
- 7.24. The camera configurations view may open the video view when a camera configuration is selected
- 7.25. The camera configurations view shall open the camera configuration view when the edit icon is selected
- 7.26. The camera configuration view shall include an additional options icon
- 7.27. The camera configuration view additional options menu shall support discarding current changes

- 7.28. The camera configuration view additional options menu may support deletion of the current configuration
- 7.29. The camera configuration view shall include a settable configuration name
- 7.30. The camera configuration view shall include a settable video URL
- 7.31. The camera configuration view shall include a settable video protocol
- 7.32. The camera configuration view shall include a settable PTZ IP address
- 7.33. The camera configuration view shall include a settable PTZ port
- 7.34. The camera configuration view shall include a settable PTZ serial configuration
- 7.35. The camera configuration view shall include a settable PTZ protocol
- 8. Log View
 - 8.1. The log view shall support all application logging
 - 8.2. The log view shall include a spinner for selecting log level filtering
 - 8.3. The log view shall include an additional options icon
 - 8.4. The log view additional options menu shall support enabling logging to file
 - 8.5. The log view additional options menu shall support setting file logging level
 - 8.6. The log view additional options menu shall support setting maximum log file size
 - 8.7. The log view additional options menu shall support setting the number of rotating log files
 - 8.8. The log view shall display a detailed timestamp
 - 8.9. The log view shall display a log level
 - 8.10. The log view shall display the module generating the log entry
 - 8.11. The log view shall display the log entry
 - 8.12. The log view shall support the use of swipe gestures to view the log
- 9. Diagnostics View
 - 9.1. The diagnostics view shall support advanced diagnostics of camera PTZ units
 - 9.2. The diagnostics view shall include a camera and configuration selection icon
 - 9.3. The diagnostics view shall include a snapshot icon for capturing the currently displayed view
 - 9.4. The diagnostics view shall include an additional options icon
 - 9.5. The diagnostics view shall include a spinner containing all PTZ protocol request commands
 - 9.6. The diagnostics view shall include a send request button
 - 9.7. The diagnostics view shall include a response text box
 - 9.8. The diagnostics view additional options menu shall support enabling response highlighting
 - 9.9. The diagnostics view shall support highlighting response green upon success

9.10. The diagnostics view shall support highlighting response red upon failure

9.11. The diagnostics view shall have a custom request option

10. Photo View

10.1. The photo view shall support viewing and managing app snapshots

10.2. The photo view shall support use of swipe gestures

10.3. The photo view shall support use of tap gestures

10.4. The photo view shall support deletion of a snapshot

10.5. The photo view shall support ordering pictures by timestamp

10.6. The photo view shall support ordering pictures by camera

Appendix E:

Handheld Diagnostic Controller Console System Requirements

Version 1.0
September 30, 2015

AHMCT Research Center
University of California - Davis













The following are the system requirement for the Console portion of the Handheld Diagnostic Controller system.

1. Use
 - 1.1. The app shall be used to diagnose and configure Telnet accessible field elements
 - 1.2. The app shall be used to diagnose and configure SSH accessible field elements
 - 1.3. The app shall support use locally at the field element
 - 1.4. The app shall support local diagnosis and configuration of field elements
 - 1.5. The app shall support local diagnosis and configuration of the field element sub-systems individually
 - 1.6. The app shall support use remotely from field element
 - 1.7. The app shall support use remotely inside field element VPN network
 - 1.8. The app may support use remotely outside field element VPN network
 - 1.9. The app shall support remote diagnosis and configuration of the field element as a whole
 - 1.10. The app shall support Telnet
 - 1.11. The app shall support SSH
2. Operating System
 - 2.1. The app shall run on the Android OS
 - 2.2. The app shall be compatible with Android versions 4.1 and up
 - 2.3. The app shall run on Android tablets
 - 2.4. The app shall run on Android phones
3. Communications
 - 3.1. The app shall support TCP/IP communications over Wi-Fi
 - 3.2. The app shall support TCP/IP communications over 3G/4G/LTE cellular networks
 - 3.3. The app shall support TCP/IP communications over USB
4. User Interface
 - 4.1. The user interface shall include a command line
 - 4.2. The user interface shall support last command history
 - 4.3. The user interface shall support multiple command history
 - 4.4. The user interface shall support command macros
 - 4.5. The user interface shall support copy and paste
 - 4.6. The user interface shall support connection management
 - 4.7. The user interface shall support external keyboards

Appendix F:

HHDC Hardware Documentation

Table F.1: HHDC component images

HHDC component images	
	
RS232 DE9M to DB25F	RS232 DE9M to DB25M
	
DE9M to DE9M Adapter	Nexus 9 AC Charger
	
Auto USB Charger	Ethernet Adapter Crossover
	
RS232 to RS422 Adapter	USB-Serial Cable Crossover (5.5 feet)
	
USB to MicroUSB Adapter	Airconsole
	
	Black RJ45 to DE9F Crossover

HHDC component images	
Beige RJ45 to DE9F Straight-through	
	
White USB to MicroUSB cable (2 feet)	Black USB to MicroUSB cable
	
6' Cat6 Ethernet Cable	BNC Cable
	
NTSC Display Charger	NTSC Display
	
DE9F to 170-C2 Cable	USB to RS232 DE9M

Appendix G:

HHDC Hierarchical Listing of Source Code Files

```
devcomm/src
├── main
│   ├── java
│   │   ├── edu
│   │   │   ├── ucdavis
│   │   │   │   ├── ahmct
│   │   │   │   │   ├── devcomm
│   │   │   │   │   │   ├── Command.java
│   │   │   │   │   │   ├── ConnType.java
│   │   │   │   │   │   ├── DevCommConfig.java
│   │   │   │   │   │   ├── Device.java
│   │   │   │   │   │   ├── Endpoint.java
│   │   │   │   │   │   ├── ErrorReply.java
│   │   │   │   │   │   ├── http
│   │   │   │   │   │   │   ├── HttpEndpoint.java
│   │   │   │   │   │   │   ├── HttpUtil.java
│   │   │   │   │   │   │   └── MiniBase64.java
│   │   │   │   │   │   ├── log
│   │   │   │   │   │   │   ├── DataReceived.java
│   │   │   │   │   │   │   └── DataTransmitted.java
│   │   │   │   │   │   ├── Op.java
│   │   │   │   │   │   ├── OpManager.java
│   │   │   │   │   │   ├── OpWorker.java
│   │   │   │   │   │   ├── ProtoException.java
│   │   │   │   │   │   ├── Reply.java
│   │   │   │   │   │   ├── ReplyListener.java
│   │   │   │   │   │   ├── stream
│   │   │   │   │   │   │   ├── BytePipe.java
│   │   │   │   │   │   │   ├── ByteQueue.java
│   │   │   │   │   │   │   ├── StreamConn.java
│   │   │   │   │   │   │   ├── StreamEndpoint.java
│   │   │   │   │   │   │   └── StreamOpManager.java
│   │   │   │   │   │   ├── tcp
│   │   │   │   │   │   │   ├── TcpEndpoint.java
│   │   │   │   │   │   │   └── TcpStreamConn.java
│   │   │   │   │   │   └── telnet
│   │   │   │   │   │       ├── comport
│   │   │   │   │   │       │   ├── ComPortConnManager.java
│   │   │   │   │   │       │   ├── ComPortConst.java
│   │   │   │   │   │       │   ├── ComPortEndpoint.java
│   │   │   │   │   │       │   ├── ComPortStreamConn.java
│   │   │   │   │   │       │   ├── DataBits.java
│   │   │   │   │   │       │   ├── FlowCtrl.java
│   │   │   │   │   │       │   ├── Parity.java
│   │   │   │   │   │       │   ├── StopBits.java
│   │   │   │   │   │       │   └── subneg
│   │   │   │   │   │       │       ├── ComPortCommand.java
│   │   │   │   │   │       │       ├── ComPortSubnegParams.java
│   │   │   │   │   │       │       ├── ControlValue.java
│   │   │   │   │   │       │       ├── DataSizeValue.java
│   │   │   │   │   │       │       ├── LineStateBit.java
│   │   │   │   │   │       │       ├── ModemStateBit.java
│   │   │   │   │   │       │       ├── ParityValue.java
│   │   │   │   │   │       │       ├── PurgeDataSubnegParams.java
│   │   │   │   │   │       │       ├── PurgeDataValue.java
│   │   │   │   │   │       │       ├── SetBaudRateSubnegParams.java
│   │   │   │   │   │       │       └── SetControlSubnegParams.java
```

```

├── SetDataSizeSubnegParams.java
├── SetParitySubnegParams.java
├── SetStopSizeSubnegParams.java
├── SignatureSubnegParams.java
├── StopSizeValue.java
├── GenericSubnegParams.java
├── OptionState.java
├── OptionStatement.java
├── SubnegParams.java
├── TelnetCommand.java
├── TelnetConnManager.java
├── TelnetConst.java
├── TelnetEndpoint.java
├── TelnetException.java
├── TelnetOption.java
├── TelnetStreamConn.java
├── TelnetSubneg.java
├── TelnetUtil.java
├── util
│   ├── HexDump.java
│   ├── TestUtil.java
│   └── Util.java
├── test
│   └── java
│       ├── BytePipeTest.java
│       ├── ByteQueueTest.java
│       ├── MiniBase64Test.java
│       ├── PurgeDataSubnegParamsTest.java
│       ├── SetBaudRateSubnegParamsTest.java
│       ├── SetControlSubnegParamsTest.java
│       ├── SetDataSizeSubnegParamsTest.java
│       ├── SetParitySubnegParamsTest.java
│       ├── SetStopSizeSubnegParamsTest.java
│       ├── SignatureSubnegParamsTest.java
│       ├── TelnetUtilTest.java
│       └── UtilTest.java

```

```

airconsolecomm/src/
├── main
│   └── java
│       ├── edu
│       │   └── ucdavis
│       │       └── ahmct
│       │           └── airconsolecomm
│       │               ├── AirconsoleDevice.java
│       │               ├── AirconsoleFirmwareVersion.java
│       │               ├── AirconsoleOpManager.java
│       │               ├── command
│       │               │   ├── AdminFactoryResetCommand.java
│       │               │   ├── AdminRestartCommand.java
│       │               │   ├── AirconsoleCommand.java
│       │               │   ├── CheckboxElement.java
│       │               │   ├── ConfigLanSetupCommand.java
│       │               │   ├── ConfigWirelessWlanConfigCommand.java
│       │               │   ├── ElementSet.java
│       │               │   ├── FormElement.java
│       │               │   ├── GetFirmwareVersionCommand.java
│       │               │   ├── Param.java
│       │               │   ├── SelectElement.java
│       │               │   ├── SubmitButtonElement.java
│       │               │   ├── TextElement.java
│       │               │   └── WaitForResponseCommand.java
│       │               └── reply
│       │                   ├── CmdIssuedReply.java
│       │                   ├── GetFirmwareVersionReply.java
│       │                   ├── IsNotUpReply.java
│       │                   └── IsUpReply.java

```

```

dmscomm/src
├── main
│   ├── java
│   │   ├── edu
│   │   │   ├── ucdavis
│   │   │   │   ├── ahmct
│   │   │   │   │   ├── dmscomm
│   │   │   │   │   │   ├── DmsDevice.java
│   │   │   │   │   │   ├── DmsFont.java
│   │   │   │   │   │   ├── DmsOpManager.java
│   │   │   │   │   │   ├── FontTableEntry.java
│   │   │   │   │   │   ├── FontTable.java
│   │   │   │   │   │   ├── ntcip
│   │   │   │   │   │   │   ├── DmsColorScheme.java
│   │   │   │   │   │   │   ├── DmsSignType.java
│   │   │   │   │   │   │   ├── JustificationLine.java
│   │   │   │   │   │   │   ├── JustificationPage.java
│   │   │   │   │   │   │   ├── multi
│   │   │   │   │   │   │   │   ├── MultiFont.java
│   │   │   │   │   │   │   │   ├── MultiLine.java
│   │   │   │   │   │   │   │   ├── MultiMessage.java
│   │   │   │   │   │   │   │   ├── MultiPage.java
│   │   │   │   │   │   │   │   ├── MultiParser.java
│   │   │   │   │   │   │   │   ├── MultiRender.java
│   │   │   │   │   │   │   │   ├── MultiSpan.java
│   │   │   │   │   │   │   │   ├── MultiText.java
│   │   │   │   │   │   │   │   ├── ParsingException.java
│   │   │   │   │   │   │   │   ├── ParsingState.java
│   │   │   │   │   │   │   │   └── NtcipDmsOpManager.java
│   │   │   │   │   │   ├── RasterImage.java
│   │   │   │   │   │   ├── RasterImageW1.java
│   │   │   │   │   │   ├── RenderException.java
│   │   │   │   │   │   ├── signview
│   │   │   │   │   │   │   ├── Action.java
│   │   │   │   │   │   │   ├── command
│   │   │   │   │   │   │   │   ├── DispBlankTmc6Command.java
│   │   │   │   │   │   │   │   ├── DispBlankTmcCommand.java
│   │   │   │   │   │   │   │   ├── DispOnePgCommand.java
│   │   │   │   │   │   │   │   ├── DispOnePgFlCommand.java
│   │   │   │   │   │   │   │   ├── DispTwoPgCommand.java
│   │   │   │   │   │   │   │   ├── ReqDetailCommand.java
│   │   │   │   │   │   │   │   └── SignViewCommand.java
│   │   │   │   │   │   │   ├── reply
│   │   │   │   │   │   │   │   ├── Blank0DetailReply.java
│   │   │   │   │   │   │   │   ├── Blank7DetailReply.java
│   │   │   │   │   │   │   │   ├── BlankDetailReply.java
│   │   │   │   │   │   │   │   ├── DetailReply.java
│   │   │   │   │   │   │   │   ├── HealthCheckDetailReply.java
│   │   │   │   │   │   │   │   ├── OnePgDetailReply.java
│   │   │   │   │   │   │   │   ├── OnePgFlDetailReply.java
│   │   │   │   │   │   │   │   ├── SignViewReply.java
│   │   │   │   │   │   │   │   ├── SimpleAckReply.java
│   │   │   │   │   │   │   │   └── TwoPgDetailReply.java
│   │   │   │   │   │   │   ├── SignModel.java
│   │   │   │   │   │   │   ├── SignViewConfig.java
│   │   │   │   │   │   │   ├── SignViewConst.java
│   │   │   │   │   │   │   ├── SignViewDetail.java
│   │   │   │   │   │   │   ├── SignViewDevice.java
│   │   │   │   │   │   │   ├── SignViewOpManager.java
│   │   │   │   │   │   │   ├── SignViewUtil.java
│   │   │   │   │   │   │   └── Status.java
│   │   │   │   │   │   └── util
│   │   │   │   │   │   │   └── CaltransUtil.java
│   │   │   │   │   │   ├── VmsDevice.java
│   │   │   │   │   │   └── VmsFullDevice.java
│   │   │   │   │   └── test
│   │   │   │   │   │   ├── java
│   │   │   │   │   │   │   ├── MultiParserTest.java
│   │   │   │   │   │   │   └── MultiRenderTest.java

```


└─ RasterImageTest.java
└─ SignViewTest.java

```

cctvcomm/src
├── main
│   ├── java
│   │   ├── edu
│   │   │   ├── ucdavis
│   │   │   │   ├── ahmct
│   │   │   │   │   ├── cctvcomm
│   │   │   │   │   │   ├── CctvDevice.java
│   │   │   │   │   │   ├── CctvOpManager.java
│   │   │   │   │   │   ├── cohui
│   │   │   │   │   │   │   ├── CohuIConst.java
│   │   │   │   │   │   │   ├── CohuIDevice.java
│   │   │   │   │   │   │   ├── CohuIDrop.java
│   │   │   │   │   │   │   ├── CohuIOpManager.java
│   │   │   │   │   │   │   ├── CohuIUtil.java
│   │   │   │   │   │   │   ├── command
│   │   │   │   │   │   │   │   ├── CohuICommand.java
│   │   │   │   │   │   │   │   ├── CohuIFocusCommand.java
│   │   │   │   │   │   │   │   ├── CohuIIrisCommand.java
│   │   │   │   │   │   │   │   ├── CohuIPanCommand.java
│   │   │   │   │   │   │   │   ├── CohuIPresetCommand.java
│   │   │   │   │   │   │   │   ├── CohuITiltCommand.java
│   │   │   │   │   │   │   │   ├── CohuIZoomCommand.java
│   │   │   │   │   │   │   ├── reply
│   │   │   │   │   │   │   │   ├── AckReply.java
│   │   │   │   │   │   │   │   ├── CohuIReply.java
│   │   │   │   │   │   │   │   └── NakReply.java
│   │   │   │   │   │   ├── FocusParams.java
│   │   │   │   │   │   ├── IrisParams.java
│   │   │   │   │   │   ├── PanParams.java
│   │   │   │   │   │   ├── pelcod
│   │   │   │   │   │   │   ├── command
│   │   │   │   │   │   │   │   ├── PelcoDCommand.java
│   │   │   │   │   │   │   │   ├── PelcoDFocusCommand.java
│   │   │   │   │   │   │   │   ├── PelcoDIrisCommand.java
│   │   │   │   │   │   │   │   ├── PelcoDPanCommand.java
│   │   │   │   │   │   │   │   ├── PelcoDPresetCommand.java
│   │   │   │   │   │   │   │   ├── PelcoDTiltCommand.java
│   │   │   │   │   │   │   │   └── PelcoDZoomCommand.java
│   │   │   │   │   │   │   ├── PelcoDConst.java
│   │   │   │   │   │   │   ├── PelcoDDevice.java
│   │   │   │   │   │   │   ├── PelcoDOpManager.java
│   │   │   │   │   │   │   ├── PelcoDUtil.java
│   │   │   │   │   │   │   ├── reply
│   │   │   │   │   │   │   │   ├── AckReply.java
│   │   │   │   │   │   │   │   ├── NoReply.java
│   │   │   │   │   │   │   │   └── PelcoDReply.java
│   │   │   │   │   │   ├── PresetParams.java
│   │   │   │   │   │   ├── TiltParams.java
│   │   │   │   │   │   └── ZoomParams.java
│   │   │   │   │   └── test
│   │   │   │   │   │   ├── java
│   │   │   │   │   │   │   ├── CohuIUtilTest.java
│   │   │   │   │   │   │   └── PelcoDUtilTest.java

```

191

```

    ├── SignConfigurationFragment.java
    ├── SignConfiguration.java
    ├── SignConfigurationNonEditableFragment.java
    ├── SignConfigurationsFragment.java
    ├── SignFragment.java
    ├── SignImageFragment.java
    ├── SignPageFragment.java
    ├── Storage.java
    ├── StringLevelComparator.java
    ├── TcpEndpointEditableFragment.java
    ├── TcpEndpointFragment.java
    ├── TcpEndpointNonEditableFragment.java
    ├── TextAnnotation.java
    ├── TextDialogFragment.java
    ├── validation
    │   ├── ClosedIntervalValidator.java
    │   ├── EndpointHostValidator.java
    │   ├── EndpointPortValidator.java
    │   ├── LeftBoundedIntervalValidator.java
    │   ├── NonNegativeValidator.java
    │   ├── NotEmptyValidator.java
    │   ├── PositiveValidator.java
    │   ├── RightBoundedIntervalValidator.java
    │   └── Validator.java
    └── res
        ├── drawable
        │   ├── ic_annotation_color.xml
        │   ├── ic_annotation_draw.xml
        │   ├── ic_annotation_line_width.xml
        │   ├── ic_annotation_move.xml
        │   ├── ic_annotation_save.xml
        │   ├── ic_annotation_text_size.xml
        │   ├── ic_annotation_text.xml
        │   ├── ic_annotation_undo.xml
        │   ├── ic_camera_snapshot.xml
        │   ├── ic_configuration_back.xml
        │   ├── ic_configuration_cancel.xml
        │   ├── ic_configuration_edit.xml
        │   ├── ic_configuration_icon.xml
        │   ├── ic_configuration_new.xml
        │   ├── ic_configuration_save.xml
        │   ├── ic_configuration_search.xml
        │   ├── ic_gallery_annotate.xml
        │   ├── ic_gallery_rotate_clockwise.xml
        │   ├── ic_gallery_rotate_counterclockwise.xml
        │   ├── ic_nav_camera.xml
        │   ├── ic_nav_configurations.xml
        │   ├── ic_nav_gallery.xml
        │   ├── ic_nav_log.xml
        │   ├── ic_nav_settings.xml
        │   ├── ic_nav_signs.xml
        │   ├── ic_sign_blank.xml
        │   ├── ic_sign_clear.xml
        │   ├── ic_sign_edit.xml
        │   ├── ic_sign_get_message.xml
        │   ├── ic_sign_set_message.xml
        │   └── side_nav_bar.xml
        ├── drawable-v21
        └── layout
            ├── activity_dms.xml
            ├── app_bar_dms.xml
            ├── content_dms.xml
            ├── fragment_annotation.xml
            ├── fragment_camera.xml
            ├── fragment_dialog_confirm.xml
            ├── fragment_dialog_number.xml
            ├── fragment_dialog_text.xml
            ├── fragment_gallery_image.xml
            └── fragment_gallery.xml

```

- fragment_log.xml
- fragment_sign_communication_configuration_card.xml
- fragment_sign_communication_configuration_com_port_endpoint.xml
- fragment_sign_communication_configuration_editable.xml
- fragment_sign_communication_configuration_non_editable.xml
- fragment_sign_communication_configuration_tcp_endpoint.xml
- fragment_sign_configuration_card.xml
- fragment_sign_configuration_editable.xml
- fragment_sign_configuration_non_editable.xml
- fragment_sign_configurations.xml
- fragment_sign_image.xml
- fragment_sign_page.xml
- fragment_sign.xml
- log_entry.xml
- nav_header_dms.xml
- layout-v21
- menu
 - activity_dms_drawer.xml
 - annotation.xml
 - dms.xml
 - gallery.xml
 - log.xml
 - sign_communication_configuration_create.xml
 - sign_communication_configuration_edit.xml
 - sign_communication_configuration_non_editable.xml
 - sign_configuration_create.xml
 - sign_configuration_edit.xml
 - sign_configuration_non_editable.xml
 - sign_configurations.xml
 - sign.xml
- mipmap-hdpi
 - ic_launcher.png
- mipmap-mdpi
 - ic_launcher.png
- mipmap-xhdpi
 - ic_launcher.png
- mipmap-xxhdpi
 - ic_launcher.png
- mipmap-xxxhdpi
 - ic_launcher.png
- values
 - arrays.xml
 - colors.xml
 - dims.xml
 - drawables.xml
 - strings.xml
 - styles.xml
 - template-dimens.xml
- values-v21
 - styles.xml
- values-w820dp
 - dims.xml
- xml
 - preferences.xml
- test
 - java
 - edu
 - ucdavis
 - ahmct
 - dms
 - ExampleUnitTest.java

```

cctv/app
├── build.gradle
├── libs
│   ├── airconsolecomm.jar
│   ├── devcomm.jar
│   ├── cctvcomm.jar
│   ├── gson-extras-2.8.2.jar
│   ├── ksoap2-android-assembly-3.6.2-jar-with-dependencies.jar
│   └── logger-0.1.0.jar
├── proguard-rules.pro
├── src
│   ├── androidTest
│   │   ├── java
│   │   │   ├── edu
│   │   │   │   ├── ucdavis
│   │   │   │   │   ├── ahmct
│   │   │   │   │   │   ├── cctv
│   │   │   │   │   │   │   ApplicationTest.java
│   │   └── main
│   │       ├── AndroidManifest.xml
│   │       ├── java
│   │       │   ├── edu
│   │       │   │   ├── ucdavis
│   │       │   │   │   ├── ahmct
│   │       │   │   │   │   ├── cctv
│   │       │   │   │   │   │   AnnotationFragment.java
│   │       │   │   │   │   │   AnnotationView.java
│   │       │   │   │   │   │   Camera2BasicFragment.java
│   │       │   │   │   │   │   ComPortEndpointEditableFragment.java
│   │       │   │   │   │   │   ComPortEndpointFragment.java
│   │       │   │   │   │   │   ComPortEndpointNonEditableFragment.java
│   │       │   │   │   │   │   ConfigurationFragment.java
│   │       │   │   │   │   │   ConfigurationViewModel.java
│   │       │   │   │   │   │   ConfirmDialogFragment.java
│   │       │   │   │   │   │   Constant.java
│   │       │   │   │   │   │   Cctv.java
│   │       │   │   │   │   │   CctvCommunicationConfiguration.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationAdapter.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationCreateFragment.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationEditFragment.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationEditableFragment.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationFragment.java
│   │       │   │   │   │   │   CctvCommunicationConfigurationNonEditableFragment.java
│   │       │   │   │   │   │   CctvConfiguration.java
│   │       │   │   │   │   │   CctvConfigurationAdapter.java
│   │       │   │   │   │   │   CctvConfigurationCard.java
│   │       │   │   │   │   │   CctvConfigurationCreateFragment.java
│   │       │   │   │   │   │   CctvConfigurationEditFragment.java
│   │       │   │   │   │   │   CctvConfigurationEditableFragment.java
│   │       │   │   │   │   │   CctvConfigurationFragment.java
│   │       │   │   │   │   │   CctvConfigurationNonEditableFragment.java
│   │       │   │   │   │   │   CctvConfigurationsFragment.java
│   │       │   │   │   │   │   CctvFragment.java
│   │       │   │   │   │   │   CctvImageFragment.java
│   │       │   │   │   │   │   ForwarderEventAdapter.java
│   │       │   │   │   │   │   GalleryFragment.java
│   │       │   │   │   │   │   ImageFileAdapter.java
│   │       │   │   │   │   │   ImageFileComparator.java
│   │       │   │   │   │   │   ImageFile.java
│   │       │   │   │   │   │   Image.java
│   │       │   │   │   │   │   Line.java
│   │       │   │   │   │   │   LogFragment.java
│   │       │   │   │   │   │   model
│   │       │   │   │   │   │   │   ComPortEndpoint.java
│   │       │   │   │   │   │   │   Endpoint.java
│   │       │   │   │   │   │   │   EndpointType.java
│   │       │   │   │   │   │   │   StreamEndpoint.java
│   │       │   │   │   │   │   │   TcpEndpoint.java
│   │       │   │   │   │   │   │   TelnetEndpoint.java

```

- NumberDialogFragment.java
- onvif
 - Algorithm.java
 - AuthenticateDigest.java
 - AuthenticationConstants.java
 - device
 - AddScopes.java
 - AnalyticsCapabilities.java
 - AnalyticsDeviceCapabilities.java
 - AnalyticsDeviceExtension.java
 - AttachmentData.java
 - AudioOutput.java
 - AudioSource.java
 - BacklightCompensation20.java
 - BacklightCompensation.java
 - BackupFile.java
 - BinaryData.java
 - CapabilitiesExtension2.java
 - CapabilitiesExtension.java
 - Capabilities.java
 - CertificateInformationExtension.java
 - CertificateInformation.java
 - Certificate.java
 - CertificateStatus.java
 - CertificateUsage.java
 - CertificateWithPrivateKey.java
 - Color.java
 - CreateUsers.java
 - Date.java
 - DateTime.java
 - DateTimeRange.java
 - DefoggingExtension.java
 - Defogging.java
 - DeleteCertificates.java
 - DeleteDot1XConfiguration.java
 - DeleteGeoLocation.java
 - DeleteUsers.java
 - DeviceBinding.java
 - DeviceCapabilitiesExtension.java
 - DeviceCapabilities.java
 - DeviceEntity.java
 - DeviceIOCapabilities.java
 - DeviceServiceCapabilities.java
 - DigitalInput.java
 - DisplayCapabilities.java
 - DNSInformationExtension.java
 - DNSInformation.java
 - Dot11AvailableNetworksExtension.java
 - Dot11AvailableNetworks.java
 - Dot11Capabilities.java
 - Dot11Configuration.java
 - Dot11PSKSetExtension.java
 - Dot11PSKSet.java
 - Dot11SecurityConfigurationExtension.java
 - Dot11SecurityConfiguration.java
 - Dot11Status.java
 - Dot1XConfigurationExtension.java
 - Dot1XConfiguration.java
 - Dot3Configuration.java
 - DynamicDNSInformationExtension.java
 - DynamicDNSInformation.java
 - EAPMethodConfiguration.java
 - EapMethodExtension.java
 - Enums.java
 - EventCapabilities.java
 - Exposure20.java
 - Exposure.java
 - ExtendedSoapSerializationEnvelope.java
 - FloatRange.java

- FocusConfiguration20Extension.java
- FocusConfiguration20.java
- FocusConfiguration.java
- GeoLocation.java
- GeoOrientation.java
- GetCACertificatesResponse.java
- GetCapabilities.java
- GetCertificatesResponse.java
- GetCertificatesStatusResponse.java
- GetDeviceInformationResponse.java
- GetDot11Capabilities.java
- GetDot1XConfigurationsResponse.java
- GetDPAddressesResponse.java
- GetEndpointReferenceResponse.java
- GetGeoLocationResponse.java
- GetNetworkInterfacesResponse.java
- GetNetworkProtocolsResponse.java
- GetRelayOutputsResponse.java
- GetScopesResponse.java
- GetServicesResponse.java
- GetStorageConfigurationsResponse.java
- GetSystemBackupResponse.java
- GetSystemUriResponse_Extension.java
- GetSystemUriResponse.java
- GetUsersResponse.java
- Helper.java
- HostnameInformationExtension.java
- HostnameInformation.java
- ImageStabilizationExtension.java
- ImageStabilization.java
- ImagingCapabilities.java
- ImagingSettings20.java
- ImagingSettingsExtension202.java
- ImagingSettingsExtension203.java
- ImagingSettingsExtension204.java
- ImagingSettingsExtension20.java
- ImagingSettingsExtension.java
- ImagingSettings.java
- Include.java
- IOCapabilitiesExtension2.java
- IOCapabilitiesExtension.java
- IOCapabilities.java
- IPAddressFilterExtension.java
- IPAddressFilter.java
- IPAddress.java
- IPv4Configuration.java
- IPv4NetworkInterface.java
- IPv4NetworkInterfaceSetConfiguration.java
- IPv6ConfigurationExtension.java
- IPv6Configuration.java
- IPv6NetworkInterface.java
- IPv6NetworkInterfaceSetConfiguration.java
- IrCutFilterAutoAdjustmentExtension.java
- IrCutFilterAutoAdjustment.java
- LayoutExtension.java
- Layout.java
- LoadCACertificates.java
- LoadCertificates.java
- LoadCertificateWithPrivateKey.java
- LocalLocation.java
- LocalOrientation.java
- LocationEntity.java
- MarshalGuid.java
- MediaCapabilitiesExtension.java
- MediaCapabilities.java
- MiscCapabilities.java
- NetworkCapabilities_1.java
- NetworkCapabilitiesExtension2.java
- NetworkCapabilitiesExtension.java

- NetworkCapabilities.java
- NetworkGateway.java
- NetworkHostExtension.java
- NetworkHost.java
- NetworkInterfaceConnectionSetting.java
- NetworkInterfaceExtension2.java
- NetworkInterfaceExtension.java
- NetworkInterfaceInfo.java
- NetworkInterface.java
- NetworkInterfaceLink.java
- NetworkInterfaceSetConfigurationExtension2.java
- NetworkInterfaceSetConfigurationExtension.java
- NetworkInterfaceSetConfiguration.java
- NetworkProtocolExtension.java
- NetworkProtocol.java
- NetworkZeroConfigurationExtension2.java
- NetworkZeroConfigurationExtension.java
- NetworkZeroConfiguration.java
- NoiseReduction.java
- NTPInformationExtension.java
- NTPInformation.java
- OnvifVersion.java
- OSDColor.java
- OSDConfigurationExtension.java
- OSDConfiguration.java
- OSDImgConfigurationExtension.java
- OSDImgConfiguration.java
- OSDPosConfigurationExtension.java
- OSDPosConfiguration.java
- OSDReference.java
- OSDTextConfigurationExtension.java
- OSDTextConfiguration.java
- PanelLayout.java
- PrefixedIPv4Address.java
- PrefixedIPv6Address.java
- ProfileCapabilities.java
- PTZCapabilities.java
- PTZNodeExtension2.java
- PTZNodeExtension.java
- PTZNode.java
- PTZPresetTourSupportedExtension.java
- PTZPresetTourSupported.java
- PTZSpacesExtension.java
- PTZSpaces.java
- RealTimeStreamingCapabilitiesExtension.java
- RealTimeStreamingCapabilities.java
- ReceiverCapabilities.java
- RecordingCapabilities.java
- Rectangle.java
- RelayOutput.java
- RelayOutputSettings.java
- RemoteUser.java
- RemoveScopes.java
- RemoveScopesResponse.java
- ReplayCapabilities.java
- RestoreSystem.java
- ScanAvailableDot11NetworksResponse.java
- Scope.java
- SearchCapabilities.java
- SecurityCapabilities_1.java
- SecurityCapabilitiesExtension2.java
- SecurityCapabilitiesExtension.java
- SecurityCapabilities.java
- Service_Capabilities.java
- Service.java
- SetCertificatesStatus.java
- SetDNS.java
- SetDPAddresses.java
- SetGeoLocation.java

- SetNetworkDefaultGateway.java
- SetNetworkProtocols.java
- SetNTP.java
- SetScopes.java
- SetUser.java
- Space1DDescription.java
- Space2DDescription.java
- StartFirmwareUpgradeResponse.java
- StartSystemRestoreResponse.java
- StorageConfigurationData_Extension.java
- StorageConfigurationData.java
- StorageConfiguration.java
- SupportInformation.java
- SystemCapabilities_1.java
- SystemCapabilitiesExtension2.java
- SystemCapabilitiesExtension.java
- SystemCapabilities.java
- SystemDateTimeExtension.java
- SystemDateTime.java
- SystemLog.java
- SystemLogUri.java
- SystemLogUriList.java
- Time.java
- TimeZone.java
- TLSConfiguration.java
- ToneCompensationExtension.java
- ToneCompensation.java
- UserCredential_Extension.java
- UserCredential.java
- UserExtension.java
- User.java
- Vector.java
- VideoOutputExtension.java
- VideoOutput.java
- VideoResolution.java
- VideoSourceExtension2.java
- VideoSourceExtension.java
- VideoSource.java
- WhiteBalance20Extension.java
- WhiteBalance20.java
- WhiteBalance.java
- WideDynamicRange20.java
- WideDynamicRange.java
- DigestAuthentication.java
- DigestChallengeException.java
- media
 - AACDecOptions.java
 - AnalyticsDeviceEngineConfigurationExtension.java
 - AnalyticsDeviceEngineConfiguration.java
 - AnalyticsEngineConfigurationExtension.java
 - AnalyticsEngineConfiguration.java
 - AnalyticsEngineControl.java
 - AnalyticsEngineInputInfoExtension.java
 - AnalyticsEngineInputInfo.java
 - AnalyticsEngineInput.java
 - AnalyticsEngine.java
 - AudioDecoderConfiguration.java
 - AudioDecoderConfigurationOptionsExtension.java
 - AudioDecoderConfigurationOptions.java
 - AudioEncoder2Configuration.java
 - AudioEncoderConfiguration.java
 - AudioEncoderConfigurationOption.java
 - AudioEncoderConfigurationOptions.java
 - AudioOutputConfiguration.java
 - AudioOutputConfigurationOptions.java
 - AudioOutput.java
 - AudioSourceConfiguration.java
 - AudioSourceConfigurationOptions.java
 - AudioSource.java

			AudioSourceOptionsExtension.java
			BacklightCompensation20.java
			BacklightCompensation.java
			Capabilities.java
			Color.java
			ColorOptions.java
			ColorspaceRange.java
			Config.java
			ConfigurationEntity.java
			CreateOSD.java
			CreateOSDResponse.java
			DefoggingExtension.java
			Defogging.java
			DeleteOSD.java
			DeviceEntity.java
			DigitalInput.java
			Dot11Configuration.java
			Dot11PSKSetExtension.java
			Dot11PSKSet.java
			Dot11SecurityConfigurationExtension.java
			Dot11SecurityConfiguration.java
			Dot3Configuration.java
			EFlip.java
			EngineConfiguration.java
			Enums.java
			EventFilter.java
			EventSubscription.java
			EventSubscription_SubscriptionPolicy.java
			Exposure20.java
			Exposure.java
			ExtendedSoapSerializationEnvelope.java
			FilterType.java
			FloatRange.java
			FocusConfiguration20Extension.java
			FocusConfiguration20.java
			FocusConfiguration.java
			G711DecOptions.java
			G726DecOptions.java
			GetAudioDecoderConfigurationsResponse.java
			GetAudioEncoderConfigurationsResponse.java
			GetAudioOutputConfigurationsResponse.java
			GetAudioOutputsResponse.java
			GetAudioSourceConfigurationsResponse.java
			GetAudioSourcesResponse.java
			GetCompatibleAudioDecoderConfigurationsResponse.java
			GetCompatibleAudioEncoderConfigurationsResponse.java
			GetCompatibleAudioOutputConfigurationsResponse.java
			GetCompatibleAudioSourceConfigurationsResponse.java
			GetCompatibleMetadataConfigurationsResponse.java
			GetCompatibleVideoAnalyticsConfigurationsResponse.java
			GetCompatibleVideoEncoderConfigurationsResponse.java
			GetCompatibleVideoSourceConfigurationsResponse.java
			GetGuaranteedNumberOfVideoEncoderInstancesResponse.java
			GetMetadataConfigurationsResponse.java
			GetOSD.java
			GetOSDOptions.java
			GetOSDOptionsResponse.java
			GetOSDResponse.java
			GetOSDsResponse.java
			GetProfilesResponse.java
			GetVideoAnalyticsConfigurationsResponse.java
			GetVideoEncoderConfigurationsResponse.java
			GetVideoSourceConfigurationsResponse.java
			GetVideoSourceModesResponse.java
			GetVideoSourcesResponse.java
			H264Configuration.java
			H264Options2.java
			H264Options.java
			Helper.java

				ImageStabilizationExtension.java
				ImageStabilization.java
				ImagingSettings20.java
				ImagingSettingsExtension202.java
				ImagingSettingsExtension203.java
				ImagingSettingsExtension204.java
				ImagingSettingsExtension20.java
				ImagingSettingsExtension.java
				ImagingSettings.java
				IntList.java
				IntRange.java
				IntRectangle.java
				IntRectangleRange.java
				IPAddress.java
				IPv4Configuration.java
				IPv4NetworkInterface.java
				IPv6ConfigurationExtension.java
				IPv6Configuration.java
				IPv6NetworkInterface.java
				IrCutFilterAutoAdjustmentExtension.java
				IrCutFilterAutoAdjustment.java
				ItemList_ElementItem.java
				ItemListExtension.java
				ItemList.java
				ItemList_SimpleItem.java
				JpegOptions2.java
				JpegOptions.java
				LayoutExtension.java
				Layout.java
				LensDescription.java
				LensOffset.java
				LensProjection.java
				MarshalGuid.java
				MaximumNumberOfOSDs.java
				MediaBinding.java
				MediaUri.java
				MetadataConfigurationExtension.java
				MetadataConfiguration.java
				MetadataConfigurationOptionsExtension2.java
				MetadataConfigurationOptionsExtension.java
				MetadataConfigurationOptions.java
				MetadataInputExtension.java
				MetadataInput.java
				Mpeg4Configuration.java
				Mpeg4Options2.java
				Mpeg4Options.java
				MulticastConfiguration.java
				NetworkInterfaceConnectionSetting.java
				NetworkInterfaceExtension2.java
				NetworkInterfaceExtension.java
				NetworkInterfaceInfo.java
				NetworkInterface.java
				NetworkInterfaceLink.java
				NoiseReduction.java
				OSDColor.java
				OSDColorOptionsExtension.java
				OSDColorOptions.java
				OSDConfigurationExtension.java
				OSDConfiguration.java
				OSDConfigurationOptionsExtension.java
				OSDConfigurationOptions.java
				OSDImgConfigurationExtension.java
				OSDImgConfiguration.java
				OSDImgOptionsExtension.java
				OSDImgOptions.java
				OSDPosConfigurationExtension.java
				OSDPosConfiguration.java
				OSDReference.java
				OSDTextConfigurationExtension.java

				OSDTextConfiguration.java
				OSDTextOptionsExtension.java
				OSDTextOptions.java
				PaneLayout.java
				PanTiltLimits.java
				PrefixedIPv4Address.java
				PrefixedIPv6Address.java
				ProfileCapabilities.java
				ProfileExtension2.java
				ProfileExtension.java
				Profile.java
				PTControlDirectionExtension.java
				PTControlDirection.java
				PTZConfigurationExtension2.java
				PTZConfigurationExtension.java
				PTZConfiguration.java
				PTZFilter.java
				PTZNodeExtension2.java
				PTZNodeExtension.java
				PTZNode.java
				PTZPresetTourSupportedExtension.java
				PTZPresetTourSupported.java
				PTZSpacesExtension.java
				PTZSpaces.java
				PTZSpeed.java
				PTZStatusFilterOptionsExtension.java
				PTZStatusFilterOptions.java
				Rectangle.java
				RelayOutput.java
				RelayOutputSettings.java
				Reverse.java
				RotateExtension.java
				Rotate.java
				RotateOptionsExtension.java
				RotateOptions.java
				RuleEngineConfigurationExtension.java
				RuleEngineConfiguration.java
				SceneOrientation.java
				SetOSD.java
				SourceIdentificationExtension.java
				SourceIdentification.java
				Space1DDescription.java
				Space2DDescription.java
				StreamingCapabilities.java
				StreamSetup.java
				ToneCompensationExtension.java
				ToneCompensation.java
				Transport.java
				Vector1D.java
				Vector2D.java
				Vector.java
				VideoAnalyticsConfiguration.java
				VideoEncoder2Configuration.java
				VideoEncoderConfiguration.java
				VideoEncoderConfigurationOptions.java
				VideoEncoderOptionsExtension2.java
				VideoEncoderOptionsExtension.java
				VideoOutputConfiguration.java
				VideoOutputExtension.java
				VideoOutput.java
				VideoRateControl2.java
				VideoRateControl.java
				VideoResolution2.java
				VideoResolution.java
				VideoSourceConfigurationExtension2.java
				VideoSourceConfigurationExtension.java
				VideoSourceConfiguration.java
				VideoSourceConfigurationOptionsExtension2.java
				VideoSourceConfigurationOptionsExtension.java

- VideoSourceConfigurationOptions.java
- VideoSourceExtension2.java
- VideoSourceExtension.java
- VideoSource.java
- VideoSourceModeExtension.java
- VideoSourceMode.java
- WhiteBalance20Extension.java
- WhiteBalance20.java
- WhiteBalance.java
- WideDynamicRange20.java
- WideDynamicRange.java
- ZoomLimits.java
- media2
 - AddConfiguration.java
 - AnalyticsDeviceEngineConfigurationExtension.java
 - AnalyticsDeviceEngineConfiguration.java
 - AnalyticsEngineConfigurationExtension.java
 - AnalyticsEngineConfiguration.java
 - AnalyticsEngineControl.java
 - AnalyticsEngineInputInfoExtension.java
 - AnalyticsEngineInputInfo.java
 - AnalyticsEngineInput.java
 - AnalyticsEngine.java
 - AudioDecoderConfiguration.java
 - AudioEncoder2Configuration.java
 - AudioEncoder2ConfigurationOptions.java
 - AudioEncoderConfiguration.java
 - AudioOutputConfiguration.java
 - AudioOutputConfigurationOptions.java
 - AudioOutput.java
 - AudioSourceConfiguration.java
 - AudioSourceConfigurationOptions.java
 - AudioSource.java
 - AudioSourceOptionsExtension.java
 - BacklightCompensation20.java
 - BacklightCompensation.java
 - Capabilities2.java
 - Color.java
 - ColorOptions.java
 - ColorspaceRange.java
 - Config.java
 - ConfigurationEntity.java
 - ConfigurationRef.java
 - ConfigurationSet.java
 - CreateProfile.java
 - DefoggingExtension.java
 - Defogging.java
 - DeviceEntity.java
 - DigitalInput.java
 - Dot11Configuration.java
 - Dot11PSKSetExtension.java
 - Dot11PSKSet.java
 - Dot11SecurityConfigurationExtension.java
 - Dot11SecurityConfiguration.java
 - Dot3Configuration.java
 - EFlip.java
 - EncoderInstanceInfo.java
 - EncoderInstance.java
 - EngineConfiguration.java
 - Enums.java
 - EventFilter.java
 - EventSubscription.java
 - EventSubscription_SubscriptionPolicy.java
 - Exposure20.java
 - Exposure.java
 - ExtendedSoapSerializationEnvelope.java
 - FilterType.java
 - FloatRange.java
 - FocusConfiguration20Extension.java

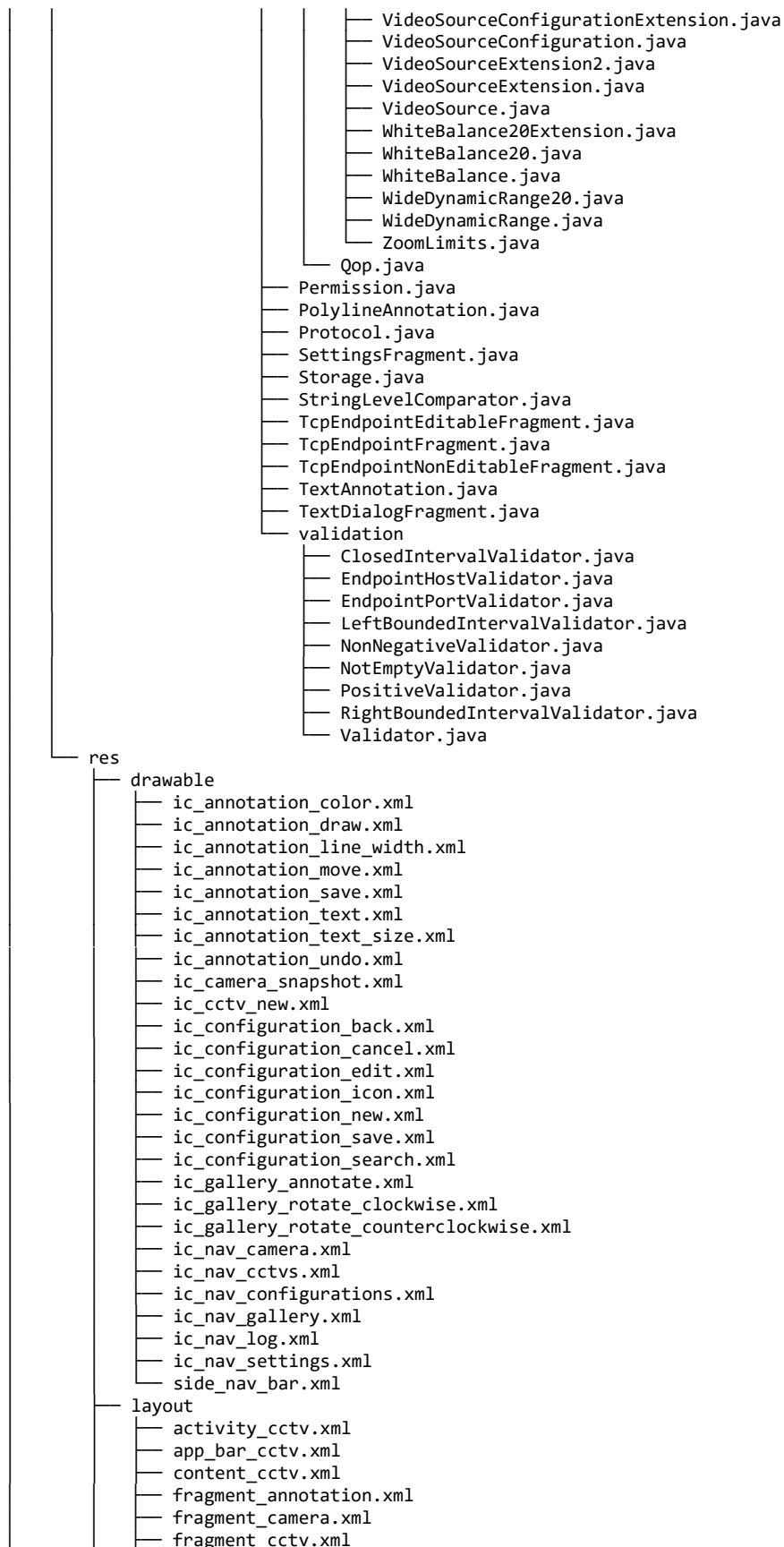
- FocusConfiguration20.java
- FocusConfiguration.java
- GetAnalyticsConfigurationsResponse.java
- GetAudioDecoderConfigurationOptionsResponse.java
- GetAudioDecoderConfigurationsResponse.java
- GetAudioEncoderConfigurationOptionsResponse.java
- GetAudioEncoderConfigurationsResponse.java
- GetAudioOutputConfigurationsResponse.java
- GetAudioSourceConfigurationsResponse.java
- GetMasksResponse.java
- GetMetadataConfigurationsResponse.java
- GetOSDsResponse.java
- GetProfiles.java
- GetProfilesResponse.java
- GetVideoEncoderConfigurationOptionsResponse.java
- GetVideoEncoderConfigurationsResponse.java
- GetVideoSourceConfigurationsResponse.java
- GetVideoSourceModesResponse.java
- H264Configuration.java
- Helper.java
- ImageStabilizationExtension.java
- ImageStabilization.java
- ImagingSettings20.java
- ImagingSettingsExtension202.java
- ImagingSettingsExtension203.java
- ImagingSettingsExtension204.java
- ImagingSettingsExtension20.java
- ImagingSettingsExtension.java
- ImagingSettings.java
- IntList.java
- IntRange.java
- IntRectangle.java
- IntRectangleRange.java
- IPAddress.java
- IPv4Configuration.java
- IPv4NetworkInterface.java
- IPv6ConfigurationExtension.java
- IPv6Configuration.java
- IPv6NetworkInterface.java
- IrCutFilterAutoAdjustmentExtension.java
- IrCutFilterAutoAdjustment.java
- ItemList_ElementItem.java
- ItemListExtension.java
- ItemList.java
- ItemList_SimpleItem.java
- LayoutExtension.java
- Layout.java
- LensDescription.java
- LensOffset.java
- LensProjection.java
- MarshalGuid.java
- Mask.java
- MaskOptions.java
- MaximumNumberOfOSDs.java
- Media2Binding.java
- MediaProfile.java
- MetadataConfigurationExtension.java
- MetadataConfiguration.java
- MetadataConfigurationOptionsExtension2.java
- MetadataConfigurationOptionsExtension.java
- MetadataConfigurationOptions.java
- MetadataInputExtension.java
- MetadataInput.java
- Mpeg4Configuration.java
- MulticastConfiguration.java
- NetworkInterfaceConnectionSetting.java
- NetworkInterfaceExtension2.java
- NetworkInterfaceExtension.java
- NetworkInterfaceInfo.java

- NetworkInterface.java
- NetworkInterfaceLink.java
- NoiseReduction.java
- OSDColor.java
- OSDColorOptionsExtension.java
- OSDColorOptions.java
- OSDConfigurationExtension.java
- OSDConfiguration.java
- OSDConfigurationOptionsExtension.java
- OSDConfigurationOptions.java
- OSDImgConfigurationExtension.java
- OSDImgConfiguration.java
- OSDImgOptionsExtension.java
- OSDImgOptions.java
- OSDPosConfigurationExtension.java
- OSDPosConfiguration.java
- OSDReference.java
- OSDTextConfigurationExtension.java
- OSDTextConfiguration.java
- OSDTextOptionsExtension.java
- OSDTextOptions.java
- PaneLayout.java
- PanTiltLimits.java
- Polygon.java
- PrefixedIPv4Address.java
- PrefixedIPv6Address.java
- ProfileCapabilities.java
- PTControlDirectionExtension.java
- PTControlDirection.java
- PTZConfigurationExtension2.java
- PTZConfigurationExtension.java
- PTZConfiguration.java
- PTZFilter.java
- PTZNodeExtension2.java
- PTZNodeExtension.java
- PTZNode.java
- PTZPresetTourSupportedExtension.java
- PTZPresetTourSupported.java
- PTZSpacesExtension.java
- PTZSpaces.java
- PTZSpeed.java
- PTZStatusFilterOptionsExtension.java
- PTZStatusFilterOptions.java
- Rectangle.java
- RelayOutput.java
- RelayOutputSettings.java
- RemoveConfiguration.java
- Reverse.java
- RotateExtension.java
- Rotate.java
- RotateOptionsExtension.java
- RotateOptions.java
- RuleEngineConfigurationExtension.java
- RuleEngineConfiguration.java
- SceneOrientation.java
- SourceIdentificationExtension.java
- SourceIdentification.java
- Space1DDescription.java
- Space2DDescription.java
- StreamingCapabilities.java
- ToneCompensationExtension.java
- ToneCompensation.java
- Vector1D.java
- Vector2D.java
- Vector.java
- VideoAnalyticsConfiguration.java
- VideoEncoder2Configuration.java
- VideoEncoder2ConfigurationOptions.java
- VideoEncoderConfiguration.java

- VideoOutputConfiguration.java
- VideoOutputExtension.java
- VideoOutput.java
- VideoRateControl2.java
- VideoRateControl.java
- VideoResolution2.java
- VideoResolution.java
- VideoSourceConfigurationExtension2.java
- VideoSourceConfigurationExtension.java
- VideoSourceConfiguration.java
- VideoSourceConfigurationOptionsExtension2.java
- VideoSourceConfigurationOptionsExtension.java
- VideoSourceConfigurationOptions.java
- VideoSourceExtension2.java
- VideoSourceExtension.java
- VideoSource.java
- VideoSourceMode.java
- WhiteBalance20Extension.java
- WhiteBalance20.java
- WhiteBalance.java
- WideDynamicRange20.java
- WideDynamicRange.java
- ZoomLimits.java
- ptz
 - AnalyticsDeviceEngineConfigurationExtension.java
 - AnalyticsDeviceEngineConfiguration.java
 - AnalyticsEngineConfigurationExtension.java
 - AnalyticsEngineConfiguration.java
 - AnalyticsEngineControl.java
 - AnalyticsEngineInputInfoExtension.java
 - AnalyticsEngineInputInfo.java
 - AnalyticsEngineInput.java
 - AnalyticsEngine.java
 - AudioDecoderConfiguration.java
 - AudioEncoder2Configuration.java
 - AudioEncoderConfiguration.java
 - AudioOutputConfiguration.java
 - AudioOutput.java
 - AudioSourceConfiguration.java
 - AudioSource.java
 - BacklightCompensation20.java
 - BacklightCompensation.java
 - Capabilities.java
 - Color.java
 - Config.java
 - ConfigurationEntity.java
 - DefoggingExtension.java
 - Defogging.java
 - DeviceEntity.java
 - DigitalInput.java
 - Dot11Configuration.java
 - Dot11PSKSetExtension.java
 - Dot11PSKSet.java
 - Dot11SecurityConfigurationExtension.java
 - Dot11SecurityConfiguration.java
 - Dot3Configuration.java
 - DurationRange.java
 - EFlip.java
 - EFlipOptionsExtension.java
 - EFlipOptions.java
 - EngineConfiguration.java
 - Enums.java
 - EventFilter.java
 - EventSubscription.java
 - EventSubscription_SubscriptionPolicy.java
 - Exposure20.java
 - Exposure.java
 - ExtendedSoapSerializationEnvelope.java
 - FilterType.java

- FloatRange.java
- FocusConfiguration20Extension.java
- FocusConfiguration20.java
- FocusConfiguration.java
- GeoLocation.java
- GetCompatibleConfigurationsResponse.java
- GetConfigurationsResponse.java
- GetNodesResponse.java
- GetPresetsResponse.java
- GetPresetToursResponse.java
- H264Configuration.java
- Helper.java
- ImageStabilizationExtension.java
- ImageStabilization.java
- ImagingSettings20.java
- ImagingSettingsExtension202.java
- ImagingSettingsExtension203.java
- ImagingSettingsExtension204.java
- ImagingSettingsExtension20.java
- ImagingSettingsExtension.java
- ImagingSettings.java
- IntRange.java
- IntRectangle.java
- IPAddress.java
- IPv4Configuration.java
- IPv4NetworkInterface.java
- IPv6ConfigurationExtension.java
- IPv6Configuration.java
- IPv6NetworkInterface.java
- IrCutFilterAutoAdjustmentExtension.java
- IrCutFilterAutoAdjustment.java
- ItemList_ElementItem.java
- ItemListExtension.java
- ItemList.java
- ItemList_SimpleItem.java
- LayoutExtension.java
- Layout.java
- LensDescription.java
- LensOffset.java
- LensProjection.java
- MarshalGuid.java
- MetadataConfigurationExtension.java
- MetadataConfiguration.java
- MetadataInputExtension.java
- MetadataInput.java
- Mpeg4Configuration.java
- MulticastConfiguration.java
- NetworkInterfaceConnectionSetting.java
- NetworkInterfaceExtension2.java
- NetworkInterfaceExtension.java
- NetworkInterfaceInfo.java
- NetworkInterface.java
- NetworkInterfaceLink.java
- NoiseReduction.java
- OSDColor.java
- OSDConfigurationExtension.java
- OSDConfiguration.java
- OSDImgConfigurationExtension.java
- OSDImgConfiguration.java
- OSDPosConfigurationExtension.java
- OSDPosConfiguration.java
- OSDReference.java
- OSDTextConfigurationExtension.java
- OSDTextConfiguration.java
- PaneLayout.java
- PanTiltLimits.java
- PrefixedIPv4Address.java
- PrefixedIPv6Address.java
- PresetTour.java

				PTControlDirectionExtension.java
				PTControlDirection.java
				PTControlDirectionOptionsExtension.java
				PTControlDirectionOptions.java
				PTZBinding.java
				PTZConfigurationExtension2.java
				PTZConfigurationExtension.java
				PTZConfiguration.java
				PTZConfigurationOptions2.java
				PTZConfigurationOptions.java
				PTZFilter.java
				PTZMoveStatus.java
				PTZNodeExtension2.java
				PTZNodeExtension.java
				PTZNode.java
				PTZPreset.java
				PTZPresetTourExtension.java
				PTZPresetTourOptions.java
				PTZPresetTourPresetDetail.java
				PTZPresetTourPresetDetailOptionsExtension.java
				PTZPresetTourPresetDetailOptions.java
				PTZPresetTourSpotExtension.java
				PTZPresetTourSpot.java
				PTZPresetTourSpotOptions.java
				PTZPresetTourStartingConditionExtension.java
				PTZPresetTourStartingCondition.java
				PTZPresetTourStartingConditionOptionsExtension.java
				PTZPresetTourStartingConditionOptions.java
				PTZPresetTourStatusExtension.java
				PTZPresetTourStatus.java
				PTZPresetTourSupportedExtension.java
				PTZPresetTourSupported.java
				PTZPresetTourTypeExtension.java
				PTZSpacesExtension.java
				PTZSpaces.java
				PTZSpeed.java
				PTZStatus.java
				PTZVector.java
				Rectangle.java
				RelayOutput.java
				RelayOutputSettings.java
				Reverse.java
				ReverseOptionsExtension.java
				ReverseOptions.java
				RotateExtension.java
				Rotate.java
				RuleEngineConfigurationExtension.java
				RuleEngineConfiguration.java
				SceneOrientation.java
				SourceIdentificationExtension.java
				SourceIdentification.java
				Space1DDescription.java
				Space2DDescription.java
				ToneCompensationExtension.java
				ToneCompensation.java
				Vector1D.java
				Vector2D.java
				Vector.java
				VideoAnalyticsConfiguration.java
				VideoEncoder2Configuration.java
				VideoEncoderConfiguration.java
				VideoOutputConfiguration.java
				VideoOutputExtension.java
				VideoOutput.java
				VideoRateControl2.java
				VideoRateControl.java
				VideoResolution2.java
				VideoResolution.java
				VideoSourceConfigurationExtension2.java



- fragment_cctv_communication_configuration_card.xml
- fragment_cctv_communication_configuration_com_port_endpoint.xml
- fragment_cctv_communication_configuration_editable.xml
- fragment_cctv_communication_configuration_non_editable.xml
- fragment_cctv_communication_configuration_tcp_endpoint.xml
- fragment_cctv_configuration_card.xml
- fragment_cctv_configuration_editable.xml
- fragment_cctv_configuration_non_editable.xml
- fragment_cctv_configurations.xml
- fragment_dialog_confirm.xml
- fragment_dialog_number.xml
- fragment_dialog_text.xml
- fragment_gallery.xml
- fragment_gallery_image.xml
- fragment_log.xml
- log_entry.xml
- nav_header_cctv.xml
- menu
 - activity_cctv_drawer.xml
 - annotation.xml
 - cctv.xml
 - cctv_communication_configuration_create.xml
 - cctv_communication_configuration_edit.xml
 - cctv_communication_configuration_non_editable.xml
 - cctv_configuration_create.xml
 - cctv_configuration_edit.xml
 - cctv_configuration_non_editable.xml
 - cctv_configurations.xml
 - gallery.xml
 - log.xml
- mipmap-hdpi
 - ic_launcher.png
- mipmap-mdpi
 - ic_launcher.png
- mipmap-xhdpi
 - ic_launcher.png
- mipmap-xxhdpi
 - ic_launcher.png
- mipmap-xxxhdpi
 - ic_launcher.png
- values
 - arrays.xml
 - colors.xml
 - dims.xml
 - drawables.xml
 - strings.xml
 - styles.xml
 - template-dimens.xml
- xml
 - preferences.xml
- test
 - java
 - edu
 - ucdavis
 - ahmct
 - cctv
 - onvif
 - AuthenticateDigestTest.java

Appendix H:

HHDC Cable Assemblies

Table H.1: Airconsole cable assembly pinout

RJ45	Beige DE9 Straight-through	Black DE9 Crossover
1 RTS	7 RTS	8 CTS
2 DTR	4 DTR	6 DSR
3 TXD	3 TXD	2 RXD
4 GND	5 GND	5 GND
5 GND	5 GND	5 GND
6 RXD	2 RXD	3 TXD
7 DSR	6 DSR	4 DTR
8 CTS	8 CTS	7 RTS
	9 RI	9 RI

Table H.2: USB RS-232 to 170 C2 cable assembly pinout

FTDI USB-RS232-WE-1800- BT_0.0 ¹⁰ Wire Color	170 C2 Connector Pin
Black / GND	N
Brown / CTS	Not Connected
Red / Power	Not Connected
Orange / TXD	L
Yellow / RXD	K
Green / RTS	H
Not Connected	J & M

¹⁰ [USB to RS232 Serial Converter Range of Cables Datasheet
\(https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS232_CABLES.pdf\)](https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS232_CABLES.pdf)

Table H.3: USB RS-422 to D2 CCTV cable assembly pinout

FTDI USB-RS422-WE-1800-BT¹¹ Wire Color	D2 RJ45 Socket Connector Pin
Black / GND	3 & Shield Drain
Brown / CTS+	Not Connected
Red / TXD-	4
Orange / TXD+	2
Yellow / RXD+	5
Green / RTS+	Not Connected
Blue / RTS-	Not Connected
White / RXD-	6
Grey / CTS-	Not Connected

Table H.4: USB RS-422 to D6 CCTV cable assembly pinout





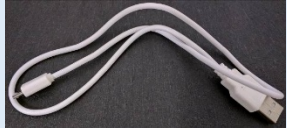

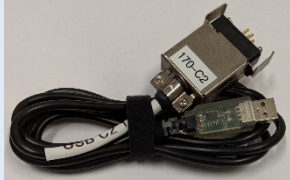
FTDI USB-RS422-WE-1800-BT Wire Color	D6 DE-9F Connector Pin
Black / GND	4 & 6
Brown / CTS+	Not Connected
Red / TXD-	8
Orange / TXD+	3
Yellow / RXD+	7
Green / RTS+	Not Connected
Blue / RTS-	Not Connected
White / RXD-	2
Grey / CTS-	Not Connected

¹¹ [Future Technology Devices International Ltd USB to RS422 Serial Converter Cable Datasheet \(https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS422_CABLES.pdf\)](https://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_USB_RS422_CABLES.pdf)

Appendix I:

HHDC Kit Contents

Table I.1: HHDC kit contents

Description	Manufacturer	Part Number	Image
Samsung Galaxy Tab S3	Samsung	SM-T820NZKAXAR	
Samsung USB Charger	Samsung	EP-TA20JWE	
Samsung USB to USB-C Cable	Samsung	Included with EP-TA20JWE	
Airconsole XL 2.0	Get Console	GCAC2-FB-1	
Airconsole USB to Micro USB Cable (2 ft)	Get Console	Included with GCAC2-FB-1	
Cat 6 Ethernet Cable (6ft)	Cable Matters	160001-BLU-1x5	
USB RS-232 to 170 C2 Cable	Custom	See HHDC cable assemblies document	

Description	Manufacturer	Part Number	Image
USB RS-422 to D2 CCTV Cable (optional)	Custom	See HHDC cable assemblies document	
USB RS-422 to D6 CCTV Cable (optional)	Custom	See HHDC cable assemblies document	
NTSC Display	ViewZ	VZ-56SM	
NTSC Display Charger	ViewZ	Included with VZ-56SM	
BNC Cable	Amphenol RF	115101-20-72.00	
USB-Serial Cable Crossover (5.5 ft)	Get Console	USB-RJ45-180	
Beige RJ45 to DE9F Straight-through	Get Console	ADA-RJ45F-DB9F-S	
Black RJ45 to DE9F Crossover	Get Console	ADA-RJ45F-DB9F-N	

Description	Manufacturer	Part Number	Image
RS232 to RS422 Adapter	Black Box	IC1474A-F	
RS232 DE9M to DB25F	SF Cable	31D1-26200	
RS232 DE9M to DB25M	SF Cable	31D1-26100	
DE9M to DE9M Adapter	Black Box	FA440-R2	
USB to RS232 DE9M	Plugable	USB to RS-232 DB9 Serial Adapter	
Ethernet Adapter Crossover	Star Tech	C6CROSSOVER	
Auto USB Charger	Cable Matters	401015-BLKx2	