ASWSC Phase 3

Migration of the Automated Safety Warning System Controller to the Caltrans Advanced Transportation Controller Platform

Phase 1 and Phase 2 Development at MSU (Part 2 of 6)

Jeremiah Pearce, Caltrans District 2

Jeff Worthington, Caltrans District 2

Doug Galarus, Montana Tech

Phase 1 and Phase 2 Development at MSU

(Part 2 of 6)

ASWSC Development Timeline

Phase 1
July 2006 – August 2008
Montana State University Western Transportation Institute

Phase 2
April 2010 – March 2012
Montana State University Western Transportation Institute

Phase 3
September 2018 – August 2020
Utah State University

Maintenance
July 2021 – March 31, 2023
Montana Technological University

Phase 1 and Phase 2 Development at MSU

Phase 1

July 2006 – August 2008

Montana State University Western Transportation Institute

Phase 1 Objectives

- Develop an Automated Safety Warning System Controller (ASWSC) that can be easily configured to acquire sensor data from existing, standard Roadside Weather Information Systems (RWIS), detection loops, Microwave Vehicle Detection Systems (MVDS) and video detection systems.

- The ASWSC must use best practice algorithms to analyze the sensor data and determine operating conditions (i.e., ice is present on pavement; a vehicle queue is present).

- The ASWSC must be easily configured to actuate a Changeable Message Sign (CMS), Extinguishable Message Sign (EMS) or flashing beacon.

Phase 1 Objectives (continued)

- The ASWSC must be able to be located at the roadside, in very remote locations, and meet all the environmental, power and communications requirements necessary for reliable operation.

- The ASWSC must also be able to operate as a stand-alone field system or be able to internetwork with a Transportation Management Center (TMC) or other automated systems.

General Requirements

Automated

- The ASWSC shall allow for automated data collection and application of best practice algorithms to analyze sensor data and to actuate related warning messages and signals.

Flexible

- The ASWSC system shall be designed for flexibility and extensibility, allowing for the integration and control of a variety of field elements.

Extensible

- The ASWSC elements shall be simple to add and configure into the system.

Standardized

- The ASWSC shall provide a single, open, multipurpose system, as opposed to many single purpose systems.

Self Contained

- The ASWSC form-factor shall be one piece, with a similar look and feel to a network router

General Requirements

The ASWSC shall include the necessary hardware and software interfaces to communicate with, control and acquire data from field elements.

The ASWSC shall include hardware and software interfaces for remote management.

The ASWSC shall be easy to setup for varied field site configurations.

The ASWSC shall facilitate user configurable and customizable alert scripts.

Field Elements

- Road Weather Information Systems (RWIS)
- Loop Detectors
- Microwave Vehicle Detection Systems (MVDS)
- Changeable Message Signs (CMS)
- Extinguishable Message Signs (EMS)
- Highway Advisory Radios (HAR)

Field Element Protocols

- RWIS: SNMP with NTCIP defined OIDs
- CMS: proprietary byte stream
- EMS, Flashing Beacon: HTTP (WebRelay)
- Loop Detector: Proprietary byte stream

System Concept

General purpose controller

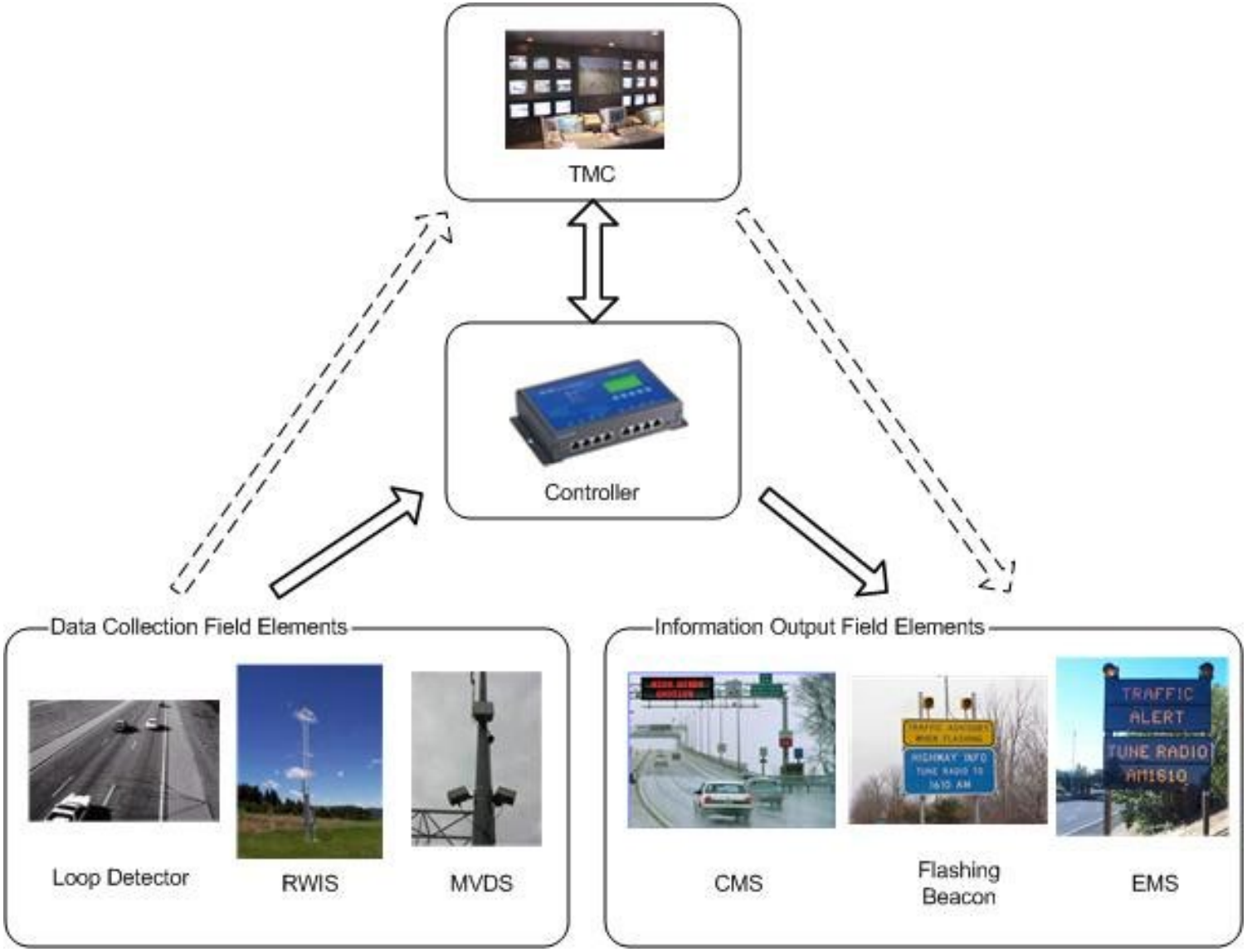
Uses existing sensors: RWIS, Loop Detector, MVDS

Uses existing displays: CMS, EMS, Flashing Beacon

User configurable by Field Engineers:

- Flexibility to specify devices for individual site configuration
- Users can write Alert Scripts to meet the needs of the site

Concept Summary



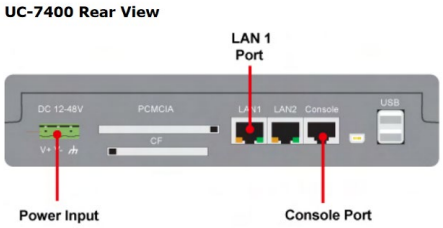
Advantages

- Located at site, short network length
- Frequent polling and evaluation of data
- 24/7/365
- Standard for all sites and all uses

Hardware Selection

MOXA UC-7420

Moxa Photo



- Intel XScale IXP-422/425, 266/533 MHz processor
- 128 MB RAM on-board, 32 MB flash
- 8 RS-232/422/485 serial ports
- Dual 10/100 Mbps Ethernet
- USB 2.0 host
- CompactFlash socket for storage expansion
- LCM display and keypad for HMI
- Ready-to-run Linux or WinCE 5.0 platform
- DIN-rail or wall-mount installation
- Robust, fanless design
- Environmental Limits:
  - 14 to 140°F
  - 5 – 95% RH
- ~$900
- 7.8 in. x 4.9 in. x 1.7 in.

Other Hardware Options

- Sun Jbox
  - VIA C3 (x86) processor
  - Choice of Solaris, Java JDS3, or Linux 2.6.10 Operating System
  - Full capabilities of a desktop system
  - Not hardened
  - $400
- Single Board Computers
  - Some assembly required
- Techsol gateway
  - < 2 watts
  - 32 bit RISC processor
  - Linux 2.6
- Axiomtek eBox
  - Wide range of x86 processors (Core 2 Duo, AMD LX800, Atom, Pentium M)
  - Wide temperature range, hardened

None of the above had a built in LCD or input buttons

Hardware Selected

Moxa UC-7420

- Software: Moxa modified version of MontaVista
  - Linux Kernel 2.4.18
  - Busybox
  - Bash
  - Apache

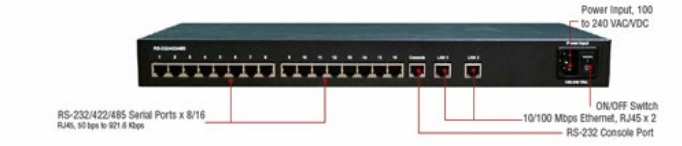
- Moxa supplied a cross-compilation toolchain to install on a desktop Linux distribution

- Python had to be built by us and installed on the device

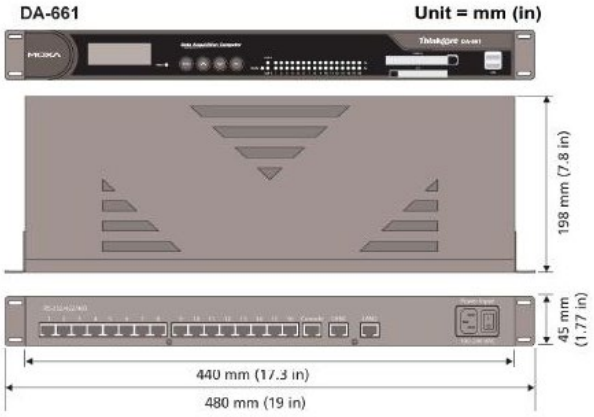
MOXA DA-661-16-LX

The Phase 2 ASWSC was deployed on the Moxa DA-661. The Moxa DA-661 was selected due to its compatibility with the UC-7420 used in Phase 1 and because it was a rack mount device consistent with Caltrans roadside network equipment mounting. shows the Moxa DA-661. A key difference between the UC-7420 used in Phase 1 and the DA-661 is the size of the display and number of front panel buttons.

Rear View



Dimensions



Moxa Photo

- Intel XScale IXP-425, 533 MHz processor
- 128 MB RAM on-board, 32 MB flash
- 16 RS-232/422/485 serial ports
- Dual 10/100 Mbps Ethernet
- USB 2.0
- CompactFlash socket for storage expansion
- LCM display and keypad for HMI
- Ready-to-run Linux or WinCE 5.0 platform
- Rack-mount installation
- Robust, fanless design
- Environmental Limits:
  - 14 to 140°F
  - 5 – 95% RH
- ~$1350

Operating System Selection

Embedded Linux

Example: MontaVista

Pros:

- Lightweight
- Better for a single use embedded system

Cons:

- Different deployment and development environments

Embedded Windows

Example: Windows CE, Windows Embedded Standard

Pros:

- More programming language options

Cons:

- Cost of software and development tools

Mainstream Linux

Examples: SUSE, Debian, RedHat

Pros:

- Full featured development environment
- All the tools Linux users are used to

Cons:

- Not as lightweight as embedded specific distributions
- More tools and features than necessary

Roll our own OS

Pros:

- Exactly what we need. No more, no less.
- Lower hardware requirements
- Can be done after initial development, after requirements are better understood

Cons:

- Very time and resource intensive

OS Selection

Linux

- Flexible
- “Free”
- Open
- Robust

Programming Options

C/C++

Pros:

- Native binary
- Speed and efficiency

Cons:

- Cross-compiling
- Less flexible after compilation

Java

Pros:

- Cross platform binary
- Almost as fast as C/C++

Cons:

- Wasn’t available on Moxa early on.
- Wasn’t open source.
- Requires compilation

Python

Pros:

- Doesn’t need to be compiled
- Cross-platform script
- Very flexible
- Broad range of modules

Cons:

- Mostly interpreted
- Less speed and efficiency

Perl

Pros:

- Doesn’t need to be compiled
- Cross-platform script
- Very flexible
- Broad range of modules

Cons:

- Interpreted
- Scales poorly
- Difficult to maintain
- Doesn’t handle OO well

Programming Selection

Python

- No need to develop an interpreter for alert logic scripts, they can be written in Python too
- Object oriented
- A lot of functionality supported in language
- Clean and readable syntax
- Native multithreading
- Conducive to rapid development and testing

Platform (Phase 1 and Phase 2)

Operating System: Embedded Linux – Montavista'

Programming / Scripting Language: Python

Hardware:

MOXA UC-7420

Moxa Photo

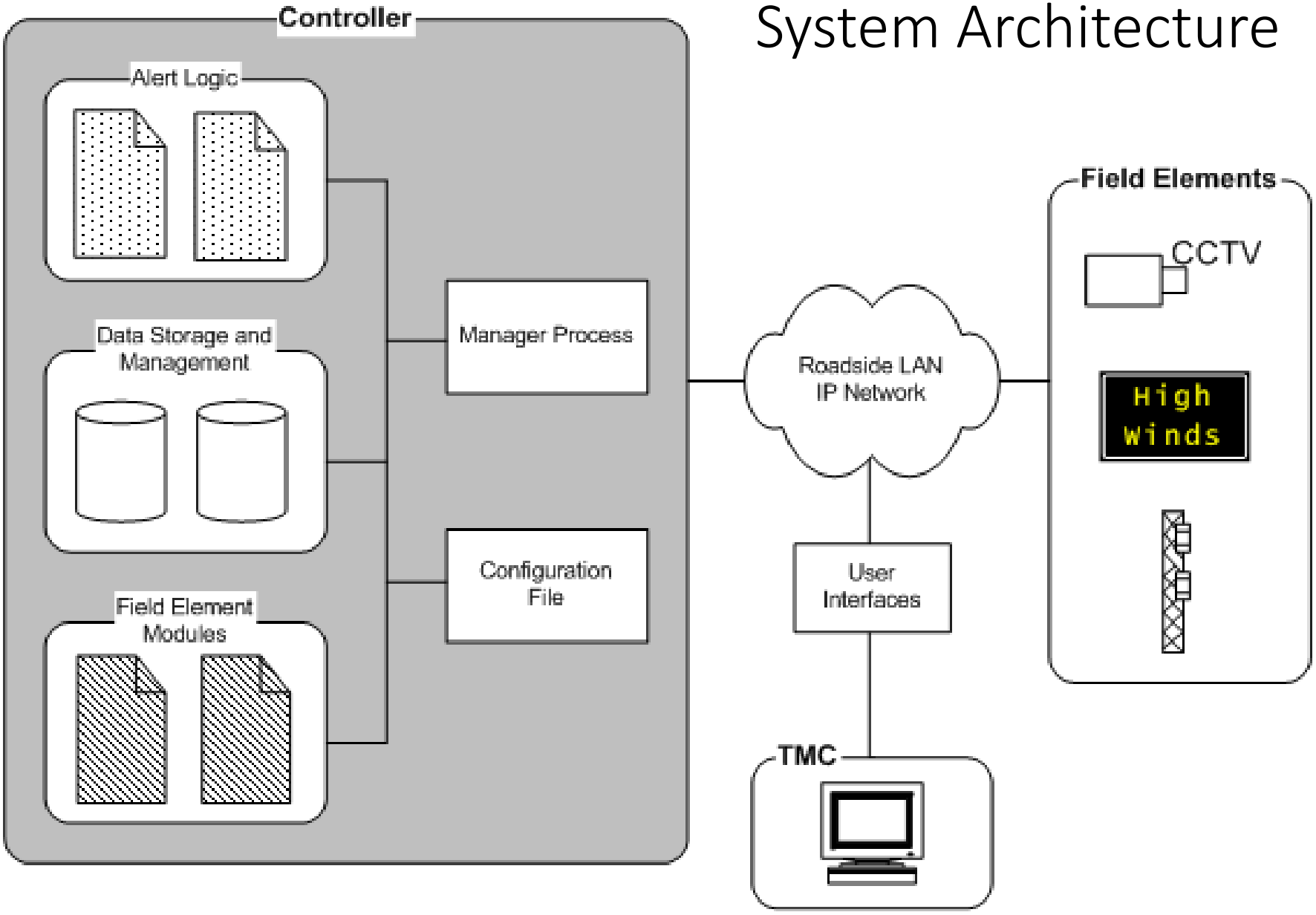
MOXA DA-661-16-LX

Moxa Photo

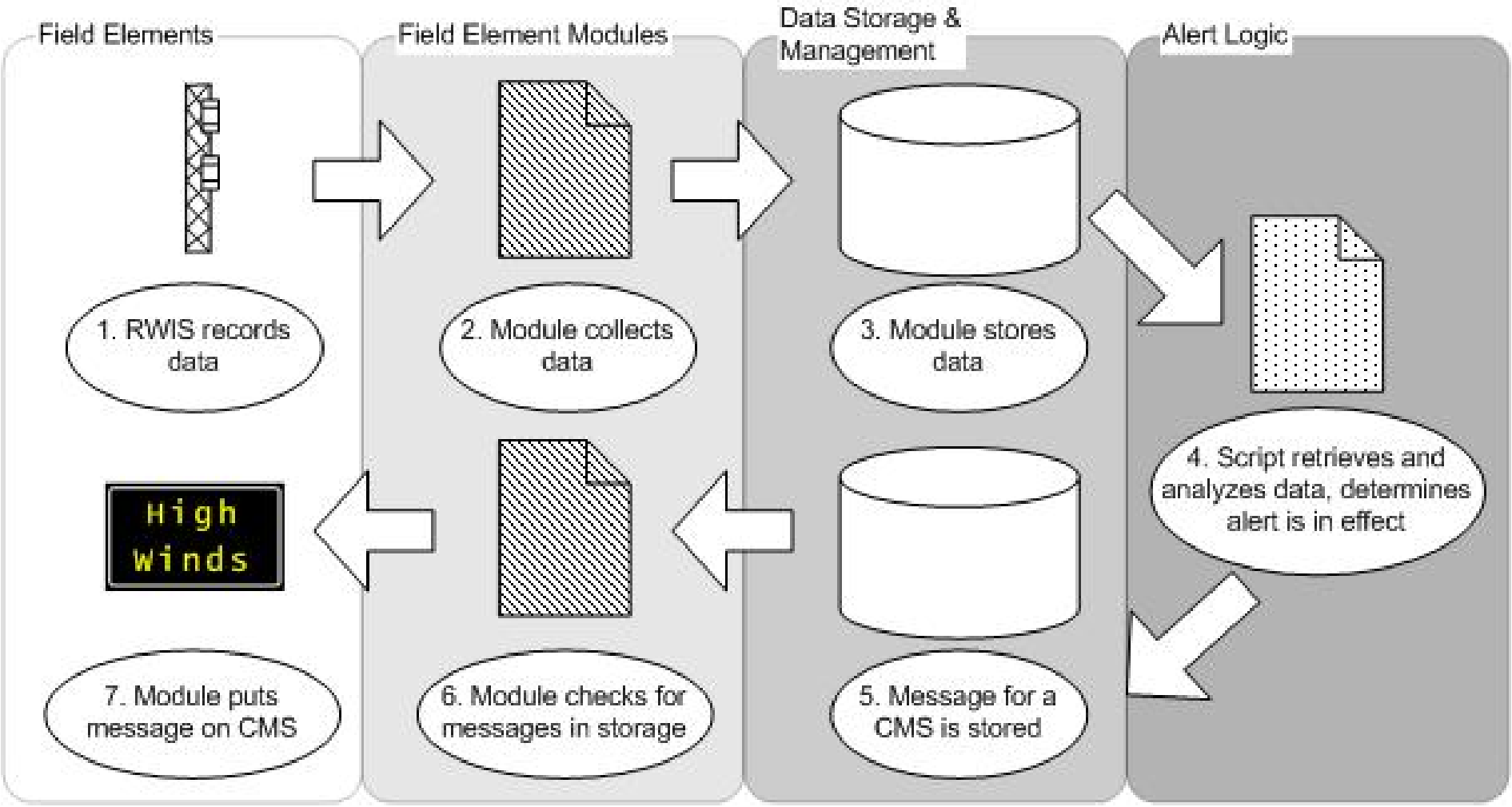
Software Design Considerations

- Not real-time, modules configured to run at set intervals
- Separate modules for each type of field element
- Modules read configuration to configure self for specific application
- Data stored in flat text files

System Architecture



System Data Flow



Manager Processes

Thread monitor

- One instance
- Manages application processes
- Also provides command line interface

Front panel interface

- One instance
- Provides management interface through front panel LCD and buttons

Modules

Data Module interface used by all modules to access data files

Where applicable, separate logic functions from communications functions

- CMS Module is generalized, uses CMS_SV170Model500 for protocol specific functions

Different types of modules

- Manager modules
- Field Element Input (RWIS, Loop Detector)
- Field Element Output (CMS)
- Alert Logic Scripts

Field Element Input Modules

RWIS

- Gets data from RWIS using NTCIP over SNMP
- Data gets parsed and is then stored in a file

Loop Detector

- Protocol is a byte stream over TCP/IP
- Data gets parsed and is then stored in a file

MVDS

- Device specific protocol is a byte stream over TCP/IP
- Data gets parsed and is then stored in a file

Field Element Output Modules

CMS

- Reads message queue from a file
- Determines what message, if any, should be sent
- Sends using a byte stream protocol over TCP/IP

EMS

- Either on or off
- Contact closure using a WebRelay device
- HTTP protocol

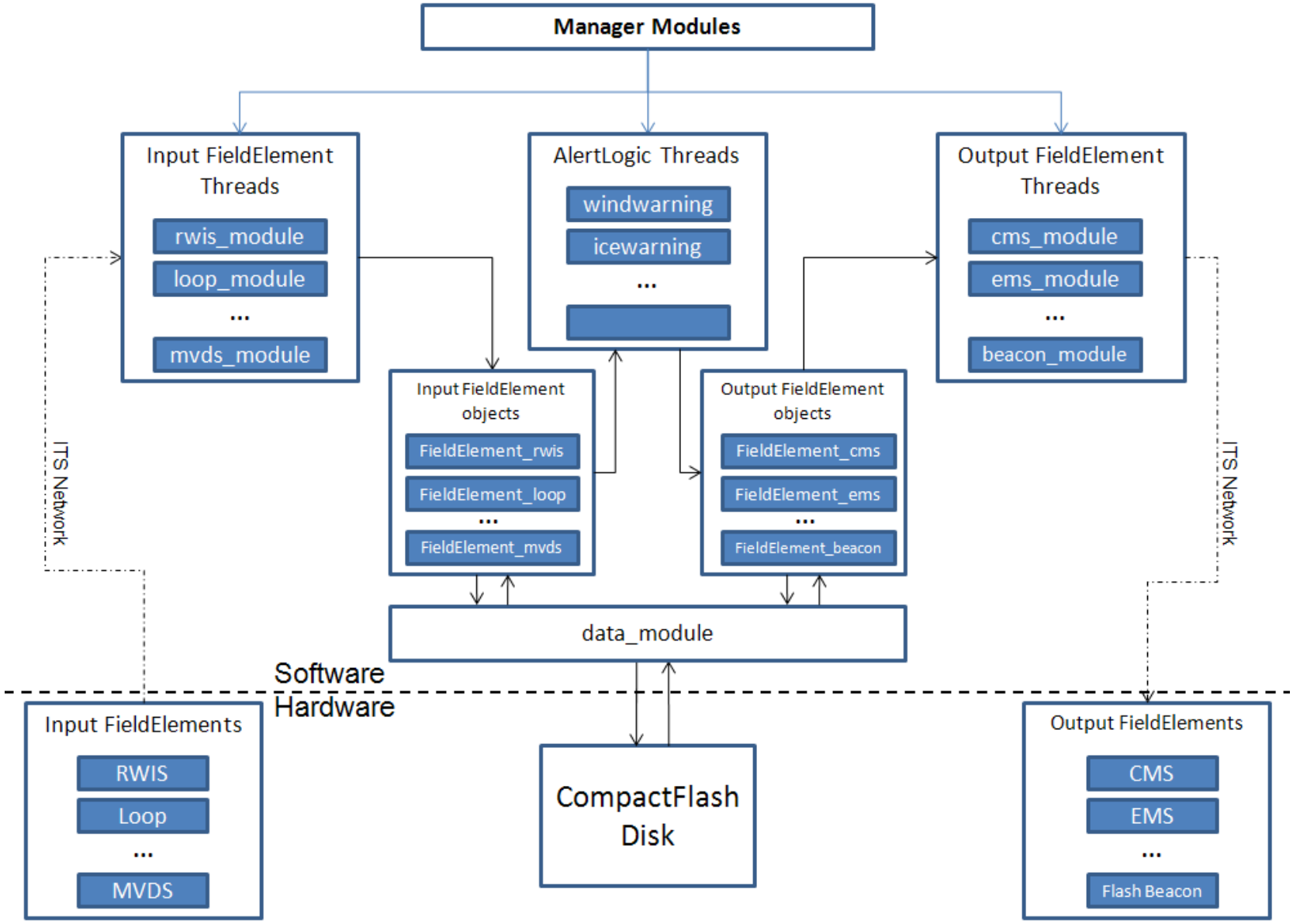
Flashing Beacon

- Similar to above

Alert Logic Scripts

Alert scripts retrieve data from input field element modules, perform logic, and write to output field element modules.

Software Architecture



Example Alert Logic Script: Ice Warning (First, some comments from Jeff …)

We have tweaked the ice warning script of the ASWSC a few times during the various project phases, mostly to address fringe conditions and to get good activations.

Events with heavy snow/ice were usually easier to determine conditions and had less questions of activation.

But the part of a winter event where the conditions are doing the first state change to a dangerous driving condition proved to be difficult to detect, especially with an older LX RPU, and the FP2000 pavement sensors.

-Jeff Worthington

Example Alert Logic Script: Ice Warning (First, some comments from Jeff …)

Scan cycles were slower, and the sensors could sometimes be fooled if there were too many cinders on the roadway, or an insulating layer of frozen snow formed over the FP2000 puck and skewed its ability to properly read the surface status.

The two NTCIP surface states of 'Trace Moisture' and 'Wet' were analyzed, and the Vaisala DST/DSC laser sensors proved especially useful in getting that 'Trace Moisture' condition, even in situations where dew would form based on humidity and dew points where the FP2000 had a much more difficult time.

-Jeff Worthington

Example Alert Logic Script: Ice Warning (First, some comments from Jeff …)

Why aren't these signs on?

I also recall tweaking the script at the Fredonyer LX RPUs to get better results after Ian (Turnbull) came to my office one day with a picture of snow on the roadway and said, "why aren't these signs on?".

-Jeff Worthington

Example Alert Logic Script: Ice Warning

```
# Name: Icy Road Alert Script
# Reference: Caltrans District 2 Icy Road Alert Parameters
# Date: 12-04-2008
# used at Spring Garden
# updated 25Aug2009 K. Beals
…
# updated 26Aug2013  J. Worthington
…
# updated 16Nov2016  J. Worthington
…
# updated 24Jun2020 B. Hoffman
…
# several edits made for WSF presentation 5Aug2021 D. Galarus
…
```

Example Alert Logic Script: Ice Warning continued …

```
TRACEMOISTURE = 4                #NTCIP essSurfaceStatus value for trace
moisture
WET = 5                          #NTCIP essSurfaceStatus value for wet
ICEWARN = 7                      #NTCIP essSurfaceStatus value for ice warning
ICEWATCH = 8                     #NTCIP essSurfaceStatus value for ice watch
SNOWWATCH = 10                   #NTCIP essSurfaceStatus value for snow watch
FROST = 13                       #NTCIP essSurfaceStatus value for frost
```

Example Alert Logic Script: Ice Warning

continued …

```
SurfaceIceWarning = False
SurfaceIceWatch = False
SurfaceFrost = False
SurfaceSnowWatch = False                    #Provided by Vaisala Laser
SensorSurfaceSnowInf = False                #Inferred Snow by conditions
conditionsSurfaceIceInf = False             #Inferred Ice by conditions
```

Example Alert Logic Script: Ice Warning

continued …

```
# Read in sensor values.
# Check for 'other' or 'error' condition, Status 1 or 2 respectively.
# Also check for invalid temps.
# Finally, log the condition.

SurfStat1 = RWIS.essSurfaceStatus1()
if SurfStat1 < 3:
    Log.system.info('Surface sensor #1 has invalid status')
    Log.system.info('Surface status for #1: %d' %(SurfStat1))

try:
    SurfTemp1 = RWIS.essSurfaceTemperature1() # this will raise an exception if data is invalid
    SurfaceSnowInfTemp1 = (SurfStat1 == WET and SurfTemp1 < 32.5)
    SurfaceIceInfTemp1 = (SurfStat1 == TRACEMOISTURE and SurfTemp1 < 31.6)
except:
    Log.system.info('Surface sensor #1 has invalid temp')
    SurfTemp1 = 212.18
    SurfaceSnowInfTemp1 = False
    SurfaceIceInfTemp1 = False
```

Example Alert Logic Script: Ice Warning

continued …

```
SurfStat2 = RWIS.essSurfaceStatus2()
if SurfStat2 < 3:
    Log.system.info('Surface sensor #2 has invalid status')
    Log.system.info('Surface status for #2: %d' %(SurfStat2))

try:
    SurfTemp2 = RWIS.essSurfaceTemperature2() # this will raise an exception if data is invalid
    SurfaceSnowInfTemp2 = (SurfStat2 == WET and SurfTemp2 < 32.5)
    SurfaceIceInfTemp2 = (SurfStat2 == TRACEMOISTURE and SurfTemp2 < 31.6)
except:
    Log.system.info('Surface sensor #1 has invalid temp')
    SurfTemp2 = 212.18
    SurfaceSnowInfTemp2 = False
    SurfaceIceInfTemp2 = False
```

Example Alert Logic Script: Ice Warning

continued …

```
# Now check the status of the sensors for any of the warning
conditions.

SurfaceIceWarning = SurfStat1 == ICEWARN or \
                    SurfStat2 == ICEWARN

SurfaceIceWatch = SurfStat1 == ICEWATCH or \
                  SurfStat2 == ICEWATCH

SurfaceSnowWatch = SurfStat1 == SNOWWATCH or \
                   SurfStat2 == SNOWWATCH

SurfaceFrost = SurfStat1 == FROST or \
               SurfStat2 == FROST
```

Example Alert Logic Script: Ice Warning

continued …

```
# Interpret a wet roadway at 32.4 F or less to have a chance of
# snow somewhere nearby.

SurfaceSnowInf = SurfaceSnowInfTemp1 or \
                 SurfaceSnowInfTemp2

# Interpret a moist roadway at 31.5F or less to have a chance of ice
# somewhere nearby.

SurfaceIceInf = SurfaceIceInfTemp1 or \
                SurfaceIceInfTemp2
```

Example Alert Logic Script: Ice Warning

continued …

```
# Now check for a warning condition
# If one is true, log the triggering condition
# and activate the CMS displays with the message.

if SurfaceIceWarning or \
   SurfaceIceWatch or \
   SurfaceSnowWatch or \
   SurfaceFrost or \
   SurfaceSnowInf or \
   SurfaceIceInf:
```

Example Alert Logic Script: Ice Warning

continued …

```
if SurfaceIceWarning:
        Log.CMS.info('Icy Road Warning: condition is Ice Warning')
        Log.CMS.info('Surface status: %d, %d' %(SurfStat1, SurfStat2))
    elif SurfaceIceWatch:
        Log.CMS.info('Icy Road Warning: condition is Ice Watch')
        Log.CMS.info('Surface status: %d, %d' %(SurfStat1, SurfStat2))
    elif SurfaceSnowWatch:
        Log.CMS.info('Icy Road Warning: condition is Snow Watch')
        Log.CMS.info('Surface status: %d, %d' %(SurfStat1, SurfStat2))
    elif SurfaceFrost:
        Log.CMS.info('Icy Road Warning: condition is Frost')
        Log.CMS.info('Surface status: %d, %d' %(SurfStat1, SurfStat2))
    elif SurfaceSnowInf:
        Log.CMS.info('Icy Road Warning: condition is Snow by inferred logic')
        Log.CMS.info('Surface temps: %3.1f, %3.1f' %(SurfTemp1, SurfTemp2))
    elif SurfaceIceInf:
        Log.CMS.info('Icy Road Warning: condition is Ice by inferred logic')
        Log.CMS.info('Surface temps: %3.1f, %3.1f' %(SurfTemp1, SurfTemp2))
    else:
        Log.CMS.info('Icy Road Warning: condition MISSING in icewarning.py')
```

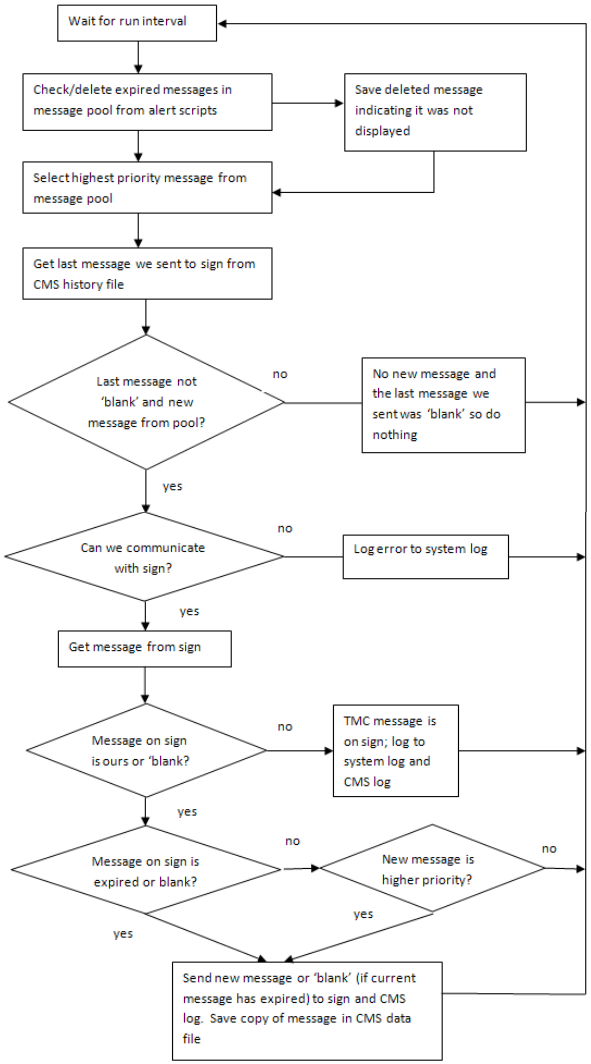
Example Alert Logic Script: Ice Warning

continued …

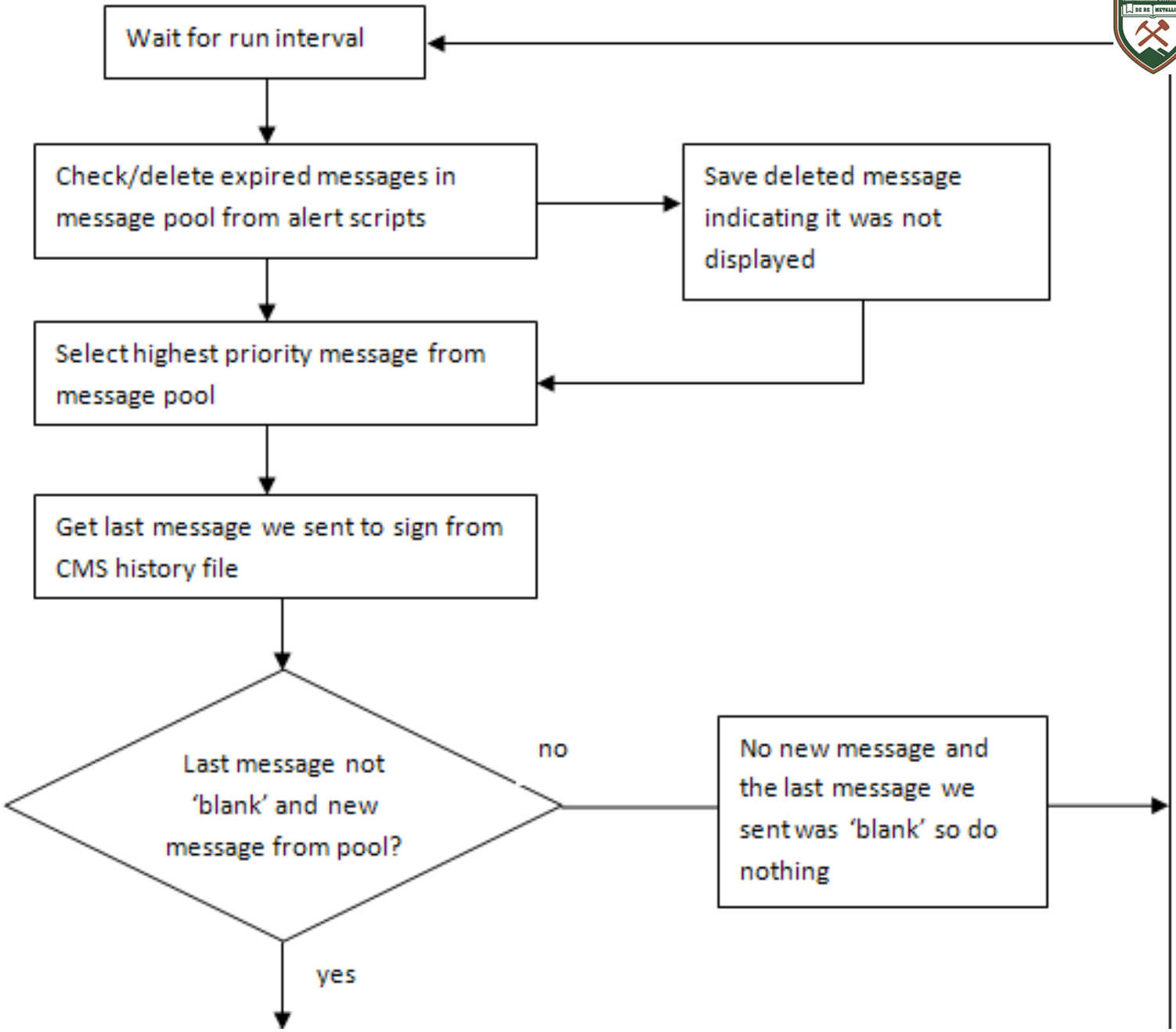
```
# Put the warning message in the sign queues
    CMSEast.AddMessageToQueue(Messages[icemessage])
    CMSWest.AddMessageToQueue(Messages[icemessage])

# Else if no warning conditions were found just log it.
else:
    Log.CMS.info('No Icy Road Warning')
```

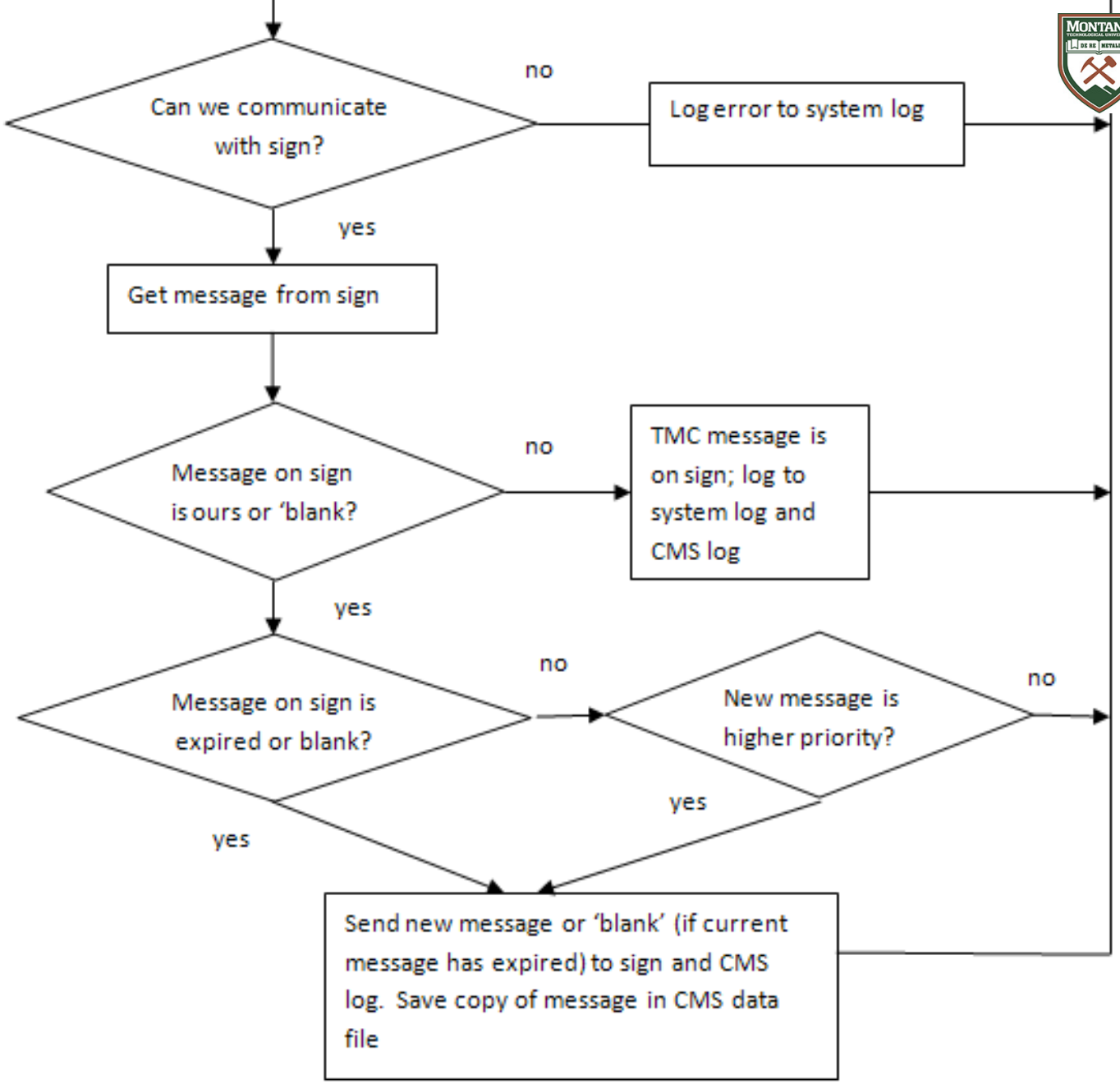
CMS Module Logic



CMS Module Logic



CMS Module Logic



Administration

Four levels of security:

- Operator
  - Monitor behavior and check for errors, but no changes permitted
- Supervisor
  - Edit threshold variables
  - start/stop/pause modules
- Technician
  - Modify alert scripts
  - Edit configuration files
  - Add/remove field elements
- Administrator
  - Root access, full control over the device

Phase 1 and Phase 2 Lab at WTI

Thanks to Caltrans and Ian Turnbull

WTI Photo

Simulates Caltrans field elements and communication via elements and TMC.

Equipment included:

- Two CMS Controllers (SignView 170, Model 500)
- Loop Detector (Model 222 GP5 in a DTS 170e Controller)
- SNI Servers
- Power Supplies
- Modems and Routers
- NTCIP exerciser software to simulate RWIS

WTI Systems Lab:

- Phone lines for dial up networking
- Communication tower on roof
- Private network separate from MSU/public network
- Connectivity to Systems Staff Offices, Labs and Rooftop

Testing/Evaluation

- In-house
  - Individual modules tested to verify communications, data parsing, etc.
  - Integration Testing
  - Dialed in to Fredonyer Pass RWIS to get real data from real RWIS

- Pilot Field Testing (More detail shown in Jeremiah's presentation.)
  - In lab at Caltrans D2
  - Field Site – Spring Garden

Evaluation

Technical Performance

- Log and data files examined to evaluate correctness

Reliability

- Long term testing in labs and at Spring Garden

Usability

- Survey sent to Ken Beals of Caltrans District 2 concerning ease of system setup and administration

Evaluation (continued)

Maintainability

- Autonomous system, goal to require little maintenance
- Tried to simplify as much as possible what maintenance tasks there were (scripts, config files, checking status and logfiles)

Security

- Minimize surface area
- Only ssh should be necessary, http is optional

Automated Safety Warning Controller Pilot Test

Controller was pilot tested at Spring Garden

Pilot test was limited to RWIS and CMS field elements

Pilot test was limited to Ice Warning alert script

Device was installed by Ken Beals of Caltrans District 2

- Installation of field elements
- Network configuration
- Alert scripts

Pilot Test Fixes and Changes

Added a sign test option to the front panel interface at Ken's request

CMS DisplayTime was off by a factor of 10

Python logging module had a bug if more than one rotate interval passes between messages being logged

CMS messages were logged when placed on sign *and* when deleted. A subtle timing issue would make the Controller think a different message was on the sign than there was.

There was an issue with archiving of the data files. Files are archived on read, but the ice warning script only reads from memory, so the RWIS file wasn't being archived.

General lessons learned

Python as a language?

- Good for rapid prototyping
- A lot of functionality built into the language

Flat files vs. SQLite

- SQLite is faster than just reading a flat file, but with some buffering techniques flat files become much faster

Buffering data in memory vs. reading files

- Managing buffers is complex
- Offers significant speed improvements

Phase 2

April 2010 – March 2012

Montana State University Western Transportation Institute

Phase 2 Goals

The outcome of Phase I of this project was a prototype hardware and software system that was tested in the field.

The outcome of Phase II would be a hardware and software system that has been pilot tested by multiple users in the field and has been prepared for wider deployment.

Additional research and development would be conducted to prepare for deployment including fault tolerance of system software, production of training materials and other documentation, and preparation of business case material to assist in deployment justification.

Phase 2 Tasks

Review Phase 1 Results

- System Review
  - Use results from Phase 1 to review system architecture and performance
- Evaluation Review
  - Identify gaps in Phase 1 functionality and items deferred to Phase 2
- Solicit input from Caltrans
  - Identify additional functionality or performance issues that need to be addressed
- Submit a review summary and recommendations document to Caltrans

Phase 2 Tasks

On-going Development

- Software will be re-factored to make it more robust and ready for production use
- Additional functionality will be added, and lab tested

Phase 2 Tasks

Lab Enhancement

- Equipment to support the additional functionality would be purchased by WTI or loaned by Caltrans to WTI for the project.

Phase 2 Tasks

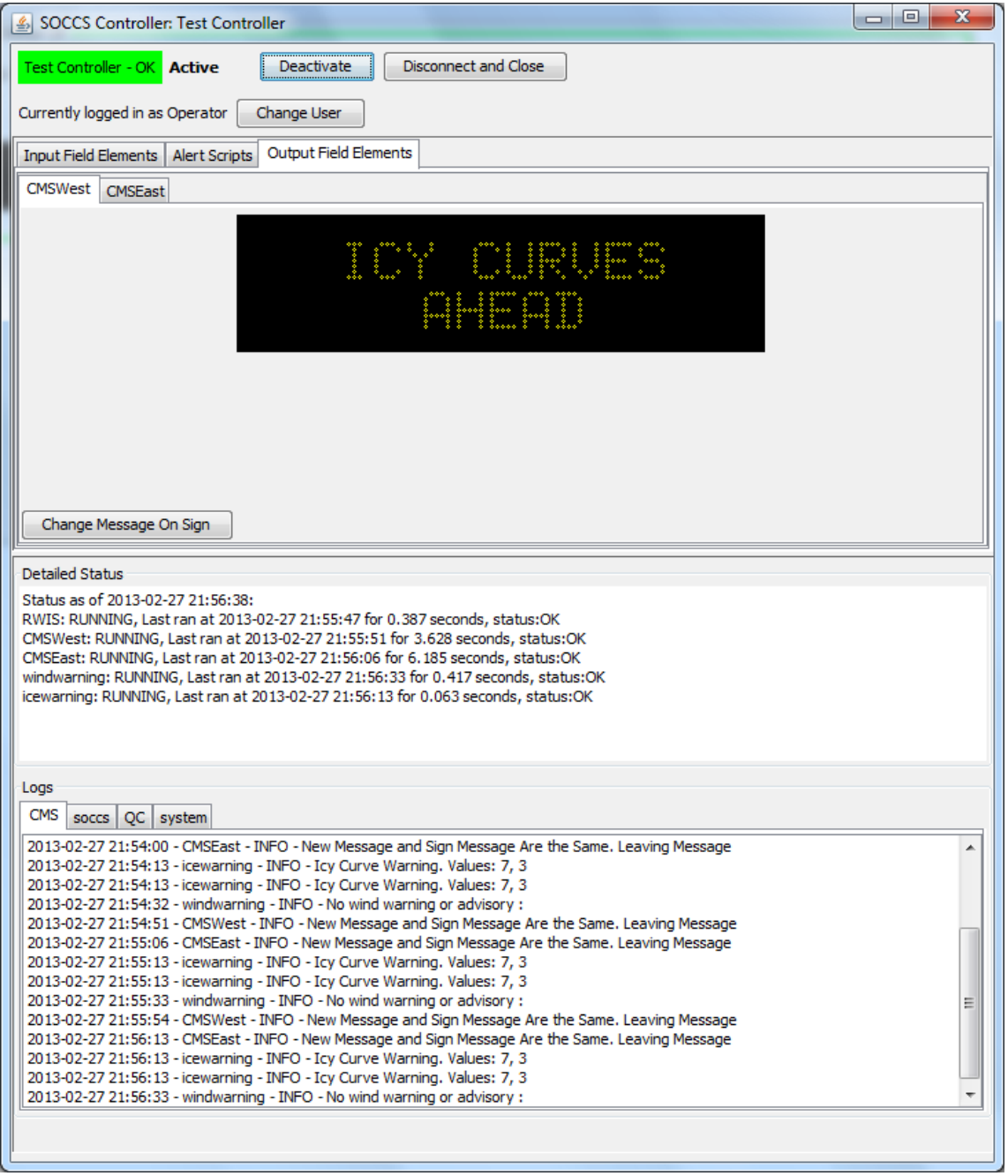
Add additional Field Element Modules

- Work with Caltrans to identify additional elements
  - Complete work started in Phase 1 on loop detector and RTMS
  - Wavetronix microwave vehicle detection
  - EIS X3 microwave vehicle detection
  - Video detection
  - HAR
  - EMS
  - Flashing Beacon

Phase 2 Tasks

Integration with SOCCS

- SOCCS Automated Controller will provide administration interface to TMC
- Implement hooks and protocols for SOCCS software to communicate with the device
- Work with Caltrans to develop a SOCCS interface



SOCCS ASWSC Java Applet

Small Print:

Phase 2 Tasks

Research implementing real-time processing

- Evaluate current response time
- Evaluate software architecture and applicability to real-time processing
- Increase poll rate or make event based

Research additional applications

- Ramp metering
- Other?

Revise the User Guide

Revise the System Management and Maintenance Guide

Phase 2 Tasks

Pilot testing

- Conduct outreach to identify prospective users from within Caltrans interested in participating in pilot testing
- Develop training materials and conduct training sessions to prepare crews for pilot testing
- Additional pilot test sites
    - Evaluate and install at potential test sites

Phase 2 Tasks

Evaluation

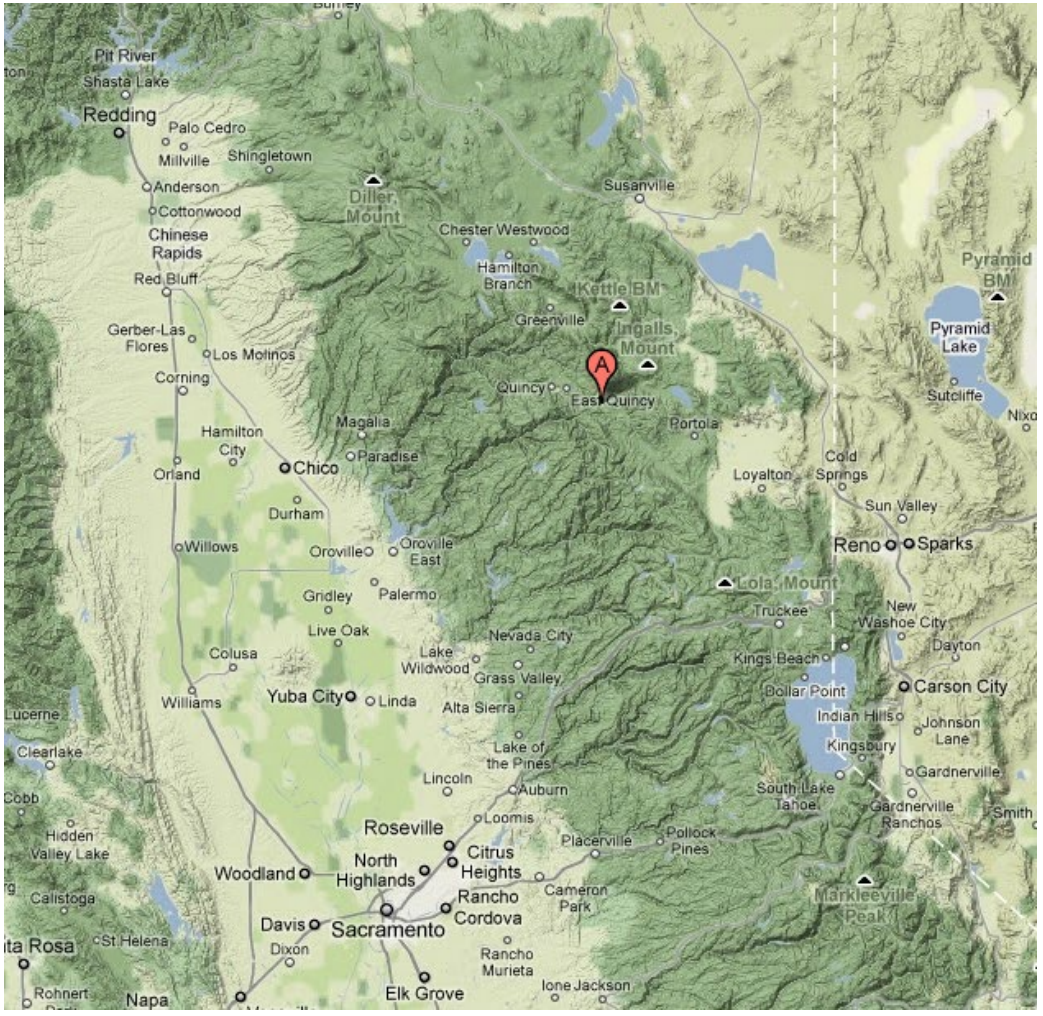
- Evaluation criteria will include, but not be limited to technical performance, reliability, usability, maintainability, and security
- An evaluation summary will be submitted to Caltrans summarizing methods and results

Evaluation

The Caltrans District 2 Spring Garden site, east of Quincy on SR-70, served as a pilot test site for the project since Phase 1. The Phase 1 system ran at Spring Garden since mid-August 2009. The Phase 2 system was subsequently installed at Spring Garden.

Phase 1 system data is presented here.

See Jeremiah's presentation for details.



Evaluation

Westbound CMS (photo by Ian Turnbull)

Eastbound CMS (photo by Ian Turnbull)

Evaluation

Pavement Surface Sensors

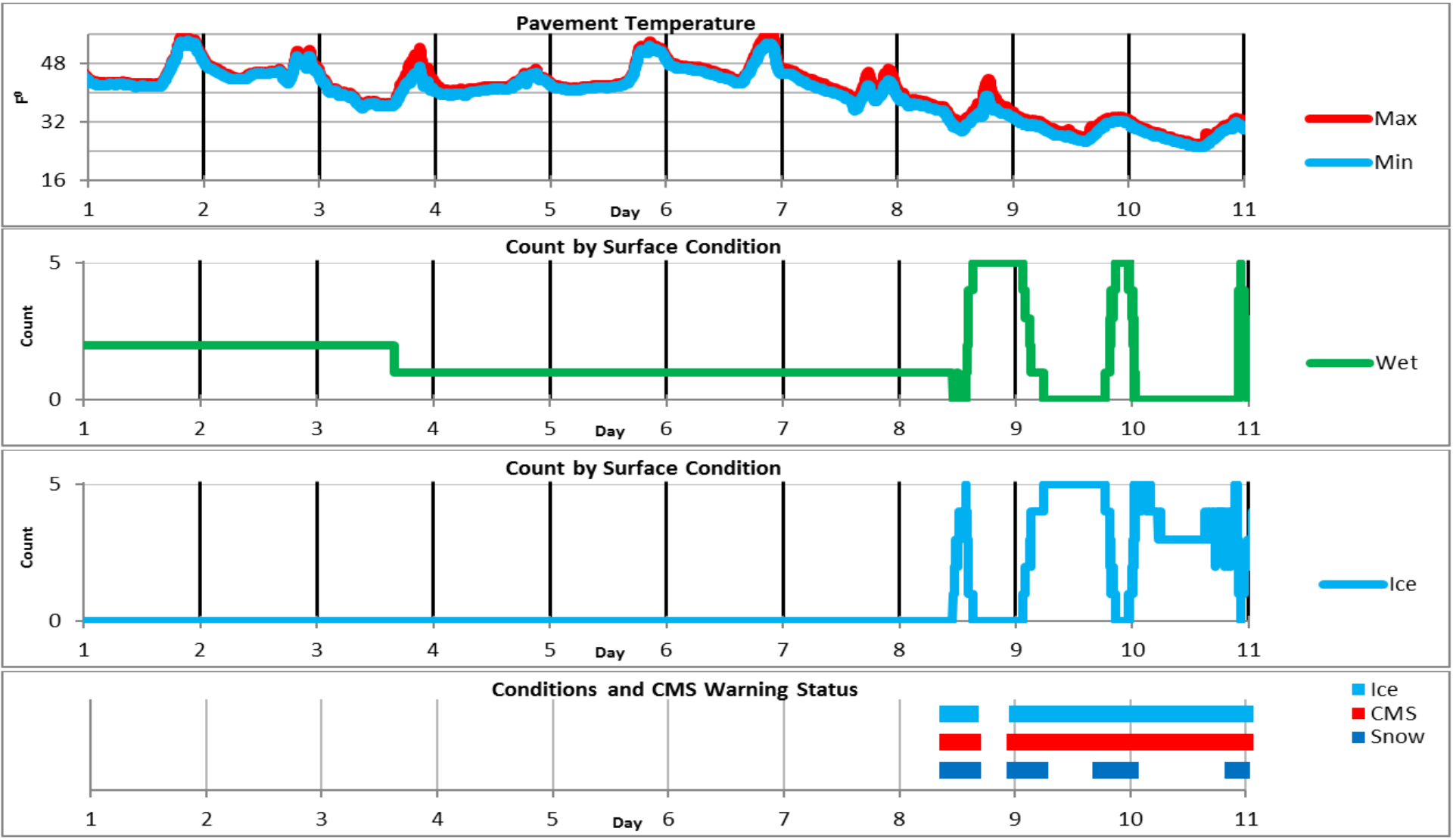
Pavement Surface Sensor

Evaluation

Spring Garden RWIS (photo by Ian Turnbull)

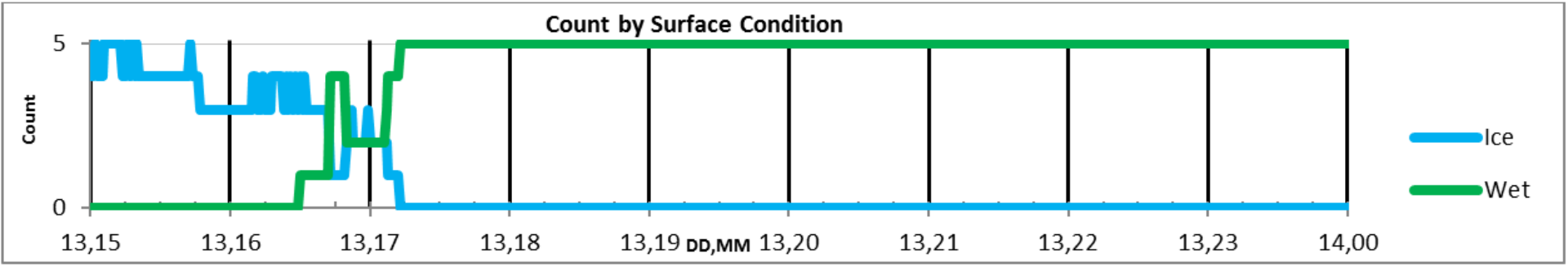
ASWSC in Spring Garden ITS Cabinet, as shown by Ian Turnbull

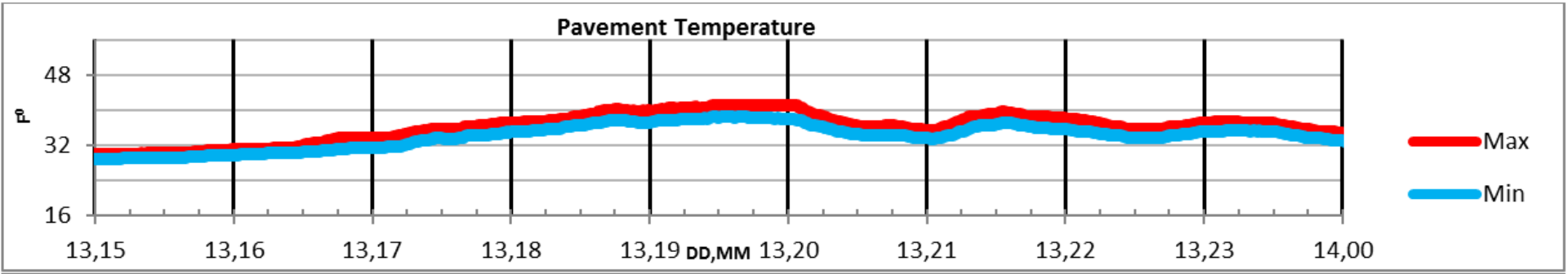
Evaluation

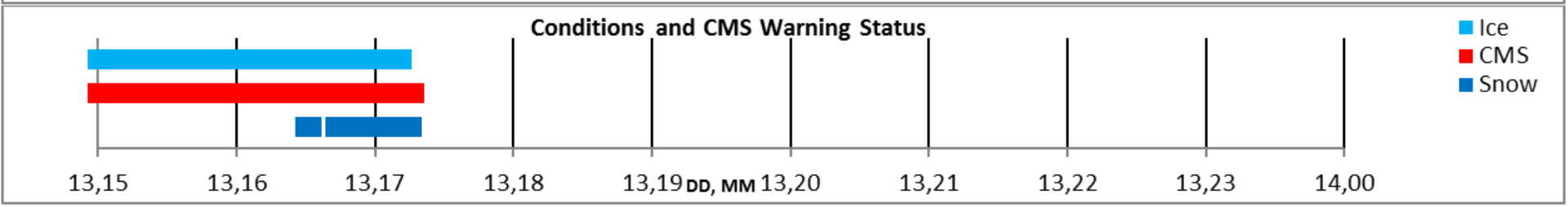

Pavement Temperature
48
32
16
F°
1
2
3
4
5
Day
6
7
8
9
10
11
Max
Min
Count by Surface Condition
5
0
Count
Wet
Count by Surface Condition
Ice
Conditions and CMS Warning Status
CMS
Snow

Evaluation


2012-12-13 15:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 16:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 17:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 18:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 12:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 20:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 21:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 22:00 GMT
SPRING GARDEN
PRESET 2
2012-12-13 23:00 GMT
SPRING GARDEN
PRESET 2


Count by Surface Condition
Count
5
0
13,15
13,16
13,17
13,18
13,19
DD,MM
13,20
13,21
13,22
13,23
14,00
Ice
Wet


Pavement Temperature
F°
48
32
16
13,15
13,16
13,17
13,18
13,19
DD,MM
13,20
13,21
13,22
13,23
14,00
Max
Min


Conditions and CMS Warning Status
13,15
13,16
13,17
13,18
13,19
DD, MM
13,20
13,21
13,22
13,23
14,00
Ice
CMS
Snow


2012-12-13 15:00 GMT
2012-12-13 16:00 GMT
2012-12-13 17:00 GMT
2012-12-13 18:00 GMT
2012-12-13 12:00 GMT
2012-12-13 20:00 GMT
2012-12-13 21:00 GMT
2012-12-13 22:00 GMT
2012-12-13 23:00 GMT

Phase 1 and 2 Testing and Evaluation at D2

See Jeremiah's presentation for more detail on Phase 1 and Phase 2 Testing and Evaluation at D2 …