Final Project

Fibonacci Series

✧ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

$$F_n = F_{n-1} + F_{n-2}$$

✧ Seed Values of:

$$F_0 = 0;\ F_1 = 1$$

✧ *Problem:*
Given n, Calculate F_n

✧ *Question:*
How long will this take to create a solution in Java?

Project Introduction

- ***Customer***: Port Authority of Allegheny County (PAAC)
- ***Project***: North Shore Extension
- ***Background***: Design and Implement a new Centralized Traffic Control (CTC) Center and Signaling System for Light Rail Transit system. The physical structure will be designed and built over the next 2 years. Out for bid is the control center, communications to and from the territory as well as the train and track controllers.
- ***Final Project Objective***: Develop fully operational demonstration of the control center, communications, train and track control system with simulator for the transit system for review by the PAAC procurement committee at the 15th class lecture.

Project Group Organization

- The groups will consist of five (5) or eight (8) students each
  - Same as lab groups.
- Each student shall be responsible for one module of the project described in the PAAC Demonstration Requirements.
- Each student will produce a set of deliverables for their module and a set of deliverables that represents the group work.

Project – Partitioned into Three Work Packages

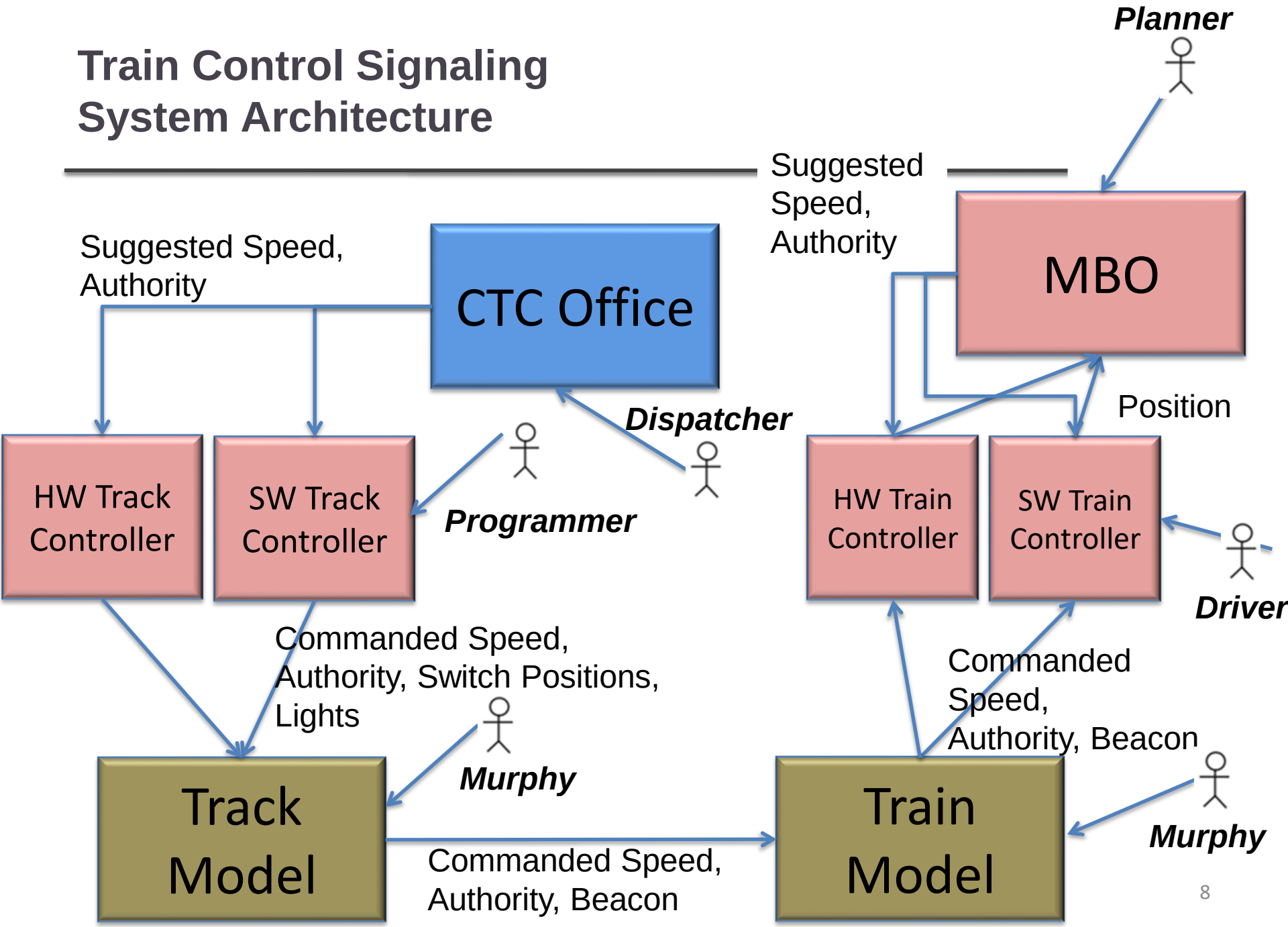
- ✧ *Iteration #1: UX Design Demonstration (5%)*
- ✧ *Work Package 1: Formulates the Question (10%)*
- ✧ *Work Package 2: Designs the Answer (10%)*
  - *Design Review (10%)*
- ✧ *Work Package 3: Implements the Answer (25%)*
  - *Iteration #2: System Connectivity Demonstration (5%)*
  - *Iteration #3: System Functional Demonstration (5%)*
  - *Iteration #4: Final System Demonstration*

PAAC Demonstration Modules

- ✧ Train Model
- ✧ Track Model
- ✧ Train Controller – Software
- ✧ Train Controller – Hardware
- ✧ Track Controller – Software
- ✧ Track Controller – Hardware
- ✧ CTC Office
- ✧ MBO Overlay (Software & Hardware)

Train Control Signaling System Architecture

Train Control Signaling System Architecture



Track Model

- **The system shall have a model of the transit system track layout.**
  - In the track layout consider grade and elevation.
  - Track model should be configurable, possibly stored in a database.
  - Allowable directions of travel, branching and speed limits should be specified in the track model.
  - Block size must be shown and configurable.
  - Need to show signals and switch machines.
  - Should have track circuits for presence detection.
  - Need to have a track layout input method.
  - Consider railway crossings.
  - Include stations for loading and unloaded people.
  - Could have power limitations
  - Track heater
  - Failure Modes: Broken Rail, Track Circuit failure, Power failure

Train Model

- ✧ The system shall have a model of a train that includes:
  - Models train movement using Newton's laws assuming the train is a rigid body point mass. Should have acceleration and velocity limits. Movement should account for the terrain of the track.
  - The train model should have a length, height, width, mass, crew count, and passenger count.
  - The train may consist of multiple cars.
  - Train inputs shall be authority (distance) from Wayside Controller, setpoint speed command, brake command, speed limit, acceleration limit, deceleration limit, route information system, temperature control, door open, door close, transponder input, track circuit input, light controller for tunnels, emergency brake from passenger.
  - Failure Modes: train engine failure, signal pickup failure, brake failure
  - Wheel slippage and power consumption are optional.

Train Controller

- This is “vital” controller that controls the train. (In the railroad and transit industry “vital” means “safety critical”.)
  - Regulates the speed of the train to the setpoint while not exceeding the speed limit or authority allowed by the system.
  - Uses track signal as input and decodes the information to determine speed limit and authority.
  - Takes as input the command setpoint from a Transit Operator.
  - Opens and closes doors at appropriate times
  - Turns on and off lights at the appropriate times
  - Announces stations and stops at the appropriate times
  - Monitors train for faults and acts upon faults in a safe manner

Track Controller

- This is a “vital” controller that controls the track.
  - The track controller received information from the office about authority and speed limits.(Authority is how far in distance the train is permitted to travel.)
  - Controls the switching of the track
  - Controls the railway crossing (lights and cross bar).
  - Detects broken rails
  - Detects the presence of trains
  - Reports back the state of the track, railway crossing, signals and trains to the central office.
  - This is a programmable unit that runs a PLC program written by the user. The program must be specifiable separately from the implementation of the track controller. Implementation must be diverse.

CTC Office

- From the office dispatchers control the transit system.
  - Schedule trains,
  - Route trains,
  - Close and opens track sections for maintenance,
  - Monitor trains.
  - Metrics should show throughput
  - Set authority
  - Display the current state of the entire transit system

Moving Block Overlay (MBO) - Controller

- Vital centralized controller that calculates the safe stopping distance of each train in real time. This distance is used to determine the safe authority of each train.
- Transmit safe Moving Block authority to each train
- Receive vital message from each train with position / location

MBO - Scheduler

- Given the track model and an interval for how often trains arrive at each station determine the train schedule for the system for a single day.
  - Accept input for throughput in 1 hour intervals.
  - Schedule shall be started at time input by user
  - Schedule shall be importable to the CTC
  - Schedule shall not violate any vital requirements
  - Scheduler shall have a startup mode that dispatches trains from the yard
  - Scheduler shall create schedule for train operators accounting for breaks and length of shift
  - Trains shall return to the yard for shift changes and breaks
  - Shifts shall be 8.5 hours long
  - Breaks shall be provided after 4 hours of driving for 30 minutes

General Requirements

- Has an automatic mode with preset scenarios to demo the system
- The system shall be capable of running at least 10 times faster than wall clock time and shall be able to pause the system.
- The final project must use at least one or more architectural and design patterns covered during the term.
- These must be identified in the architecture and design documentation.
- Identify any COTS components used in the system.
- Describe the vital aspects of the system and how it affected the architecture and design of the system.

Non-functional Requirements

- ✧ Executable on a Windows 10 Operating System
- ✧ Train and Track controller must have a vital architecture
- ✧ Each sub-system shall have a User Interface
- ✧ Each sub-system shall be submitted as an installable executable.
- ✧ The whole system shall be submitted as a runnable executable.

Information on Train Control Systems

- Sites you can learn about train control systems
  - http://www.rssi.org/index.html
  - http://www.apta.com/Pages/default.aspx
  - http://www.fra.dot.gov/
  - http://www.nrcma.org/ps.home.cfm?ID=155
- Suppliers
  - http://www.ansaldo-sts.com/en/company/our_companies/asts_usa.html
  - http://www.phwinc.com/home
  - http://www.railsim.com/

Train Control Systems

- ✧ All questions about the final project should be posted to the discussion board.
- ✧ I will post answers

Questions

Train Control Systems

Basic Architecture of Train Control

- ✧ Operation Control Center (OCC)
- ✧ Communications – Vital and Non-Vital
- ✧ Wayside Controller
- ✧ Track Circuits
- ✧ Train Controller
- ✧ Equipment – Switch Machine, Lights, Railway Crossing Gates and Lights

The Office

- Operation Control Center (OCC)
  - Sometimes called Centralized Traffic Control (CTC)
  - Or the Office
- Communications – non-vital in the USA
  - From Office to Wayside
  - From Wayside to Train
  - From Train to Wayside
  - From Wayside to Office
- Communications are vital in Europe

The Office

- ✧ Dispatchers monitor trains in whole system
- ✧ Usually each dispatcher has a portion of the territory
- ✧ Track is shown as blocks on the screen
- ✧ Dispatchers provide speed and authority to the train
- ✧ Track model held in a database
- ✧ Often see yards as a blackhole

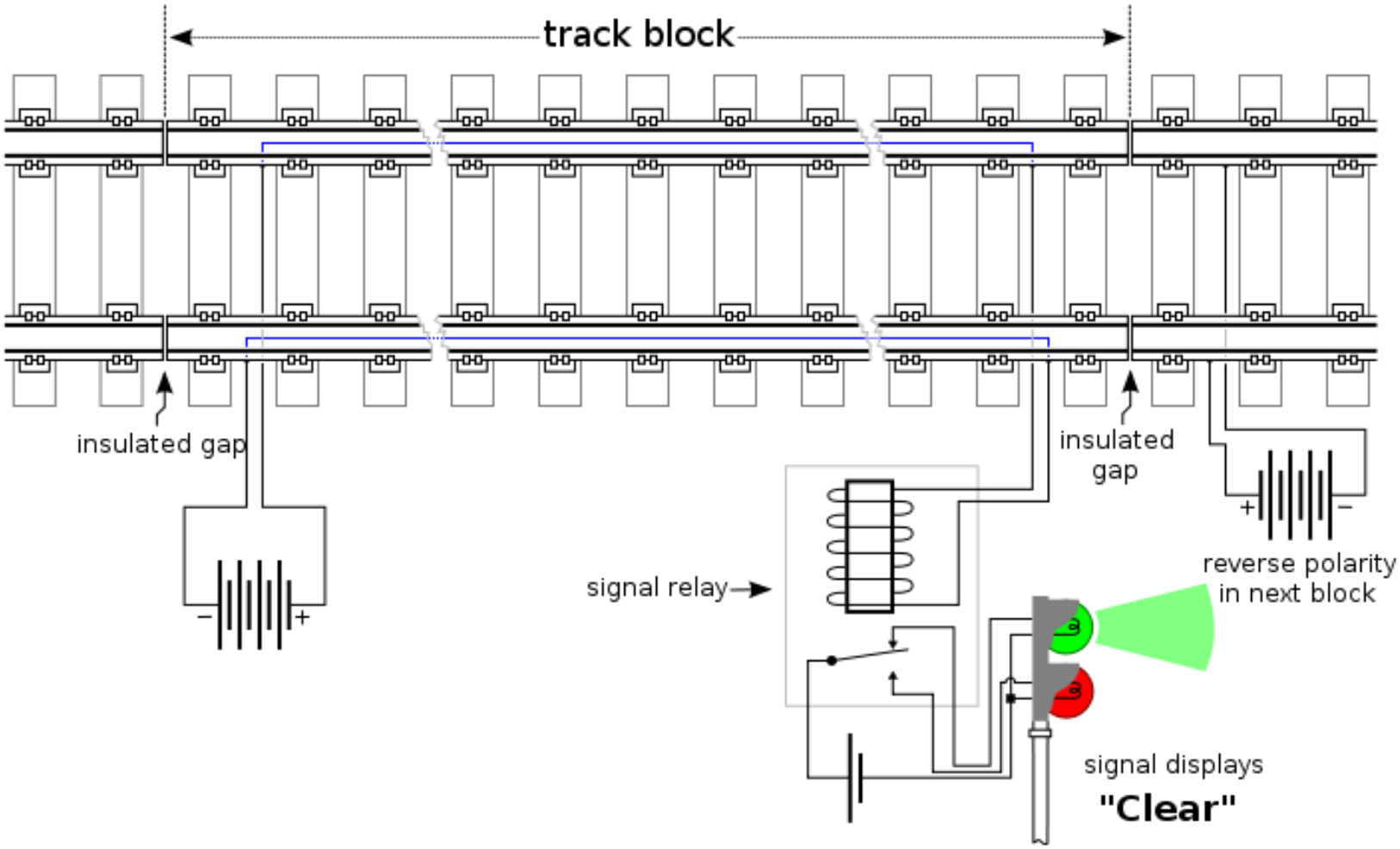
Amtrak's THORN Tower

CSX Central Office

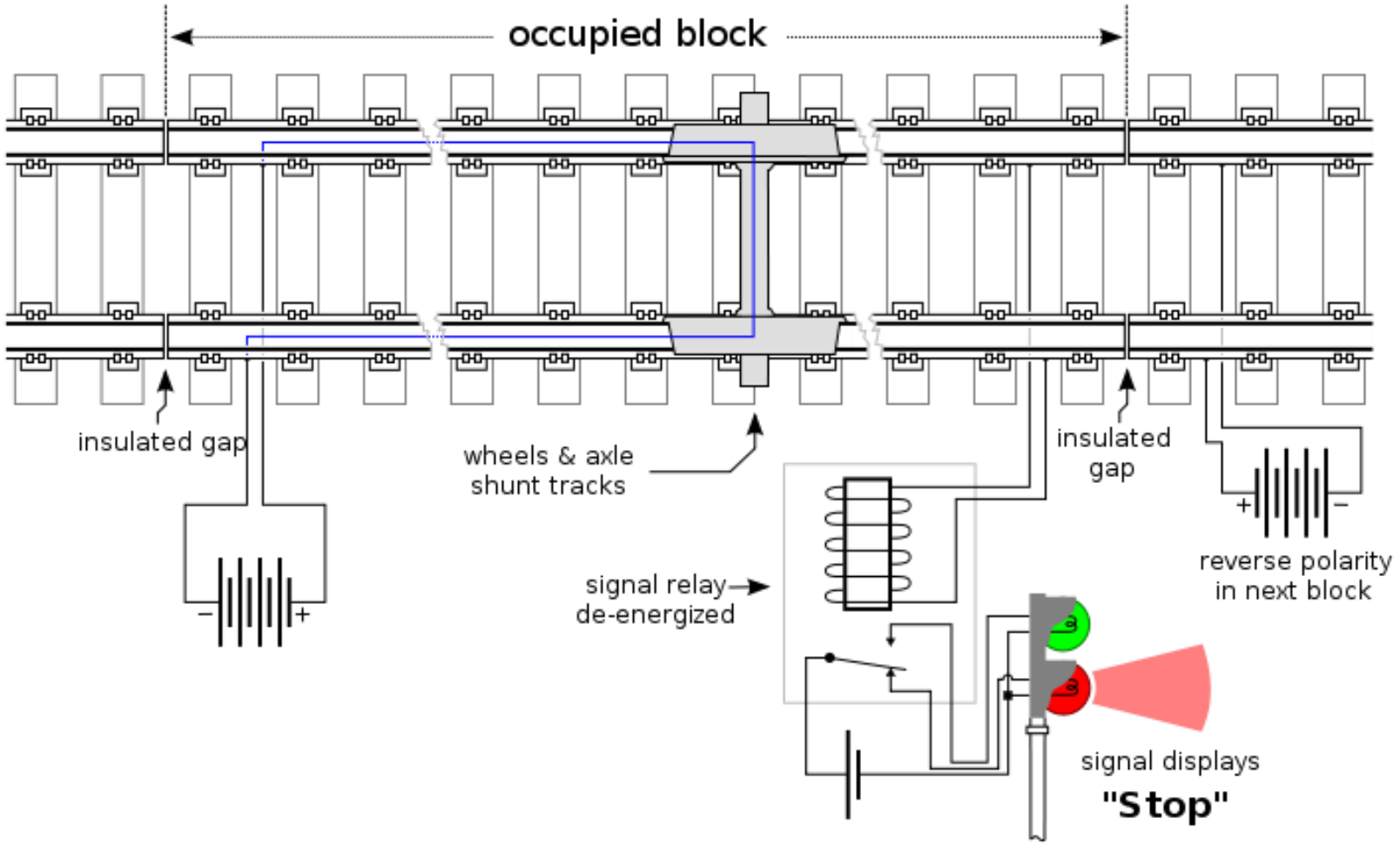
Track Circuits

- Send signal down the rail
  - AC or DC coded signals depending on age of the technology
  - Terminated by a bond – isolation transformer
  - Tracks are bonded at joints
  - If a return signal is received then no train is present
  - If a train is present, the wheels will short out the track circuit and the relay will not be energized – this detects a train.

DC Track Circuit – Unoccupied Block



DC Track Circuit – Occupied Block



Station at North Shore

Track Circuit Connection and Transponder

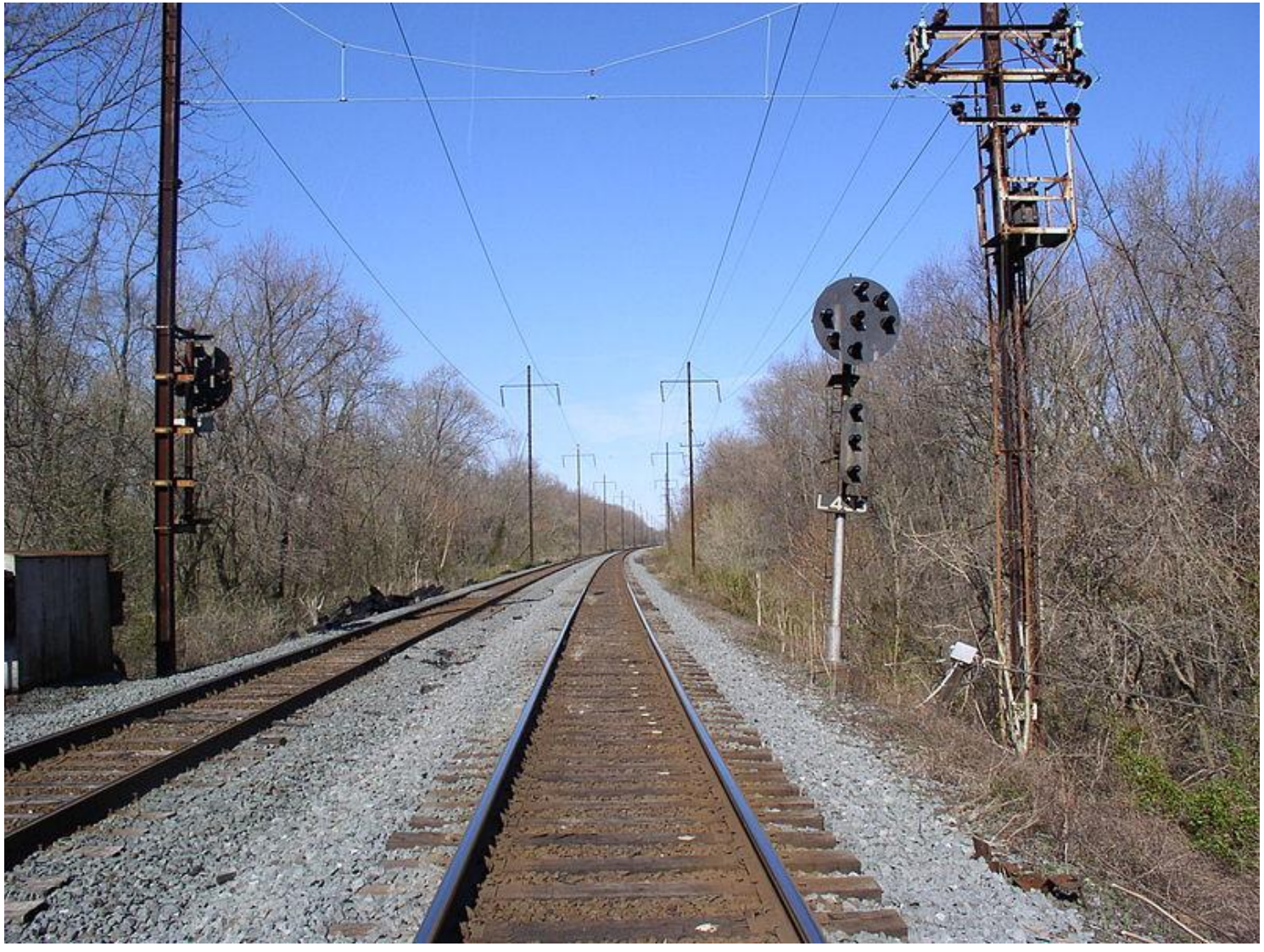
Wayside Controller

- Installed at the side of the track – called the wayside
- Collects status from the track circuits to determine occupancy
- Collects status from the switches to determine position
- Runs boolean logic to determine what is a safe action (which track, how far, and how fast) and sends this to the train though the track circuit
- Turns on lights for the train
- Moves Switches to change track
- Activates railway crossings – drops gate, turns on lights

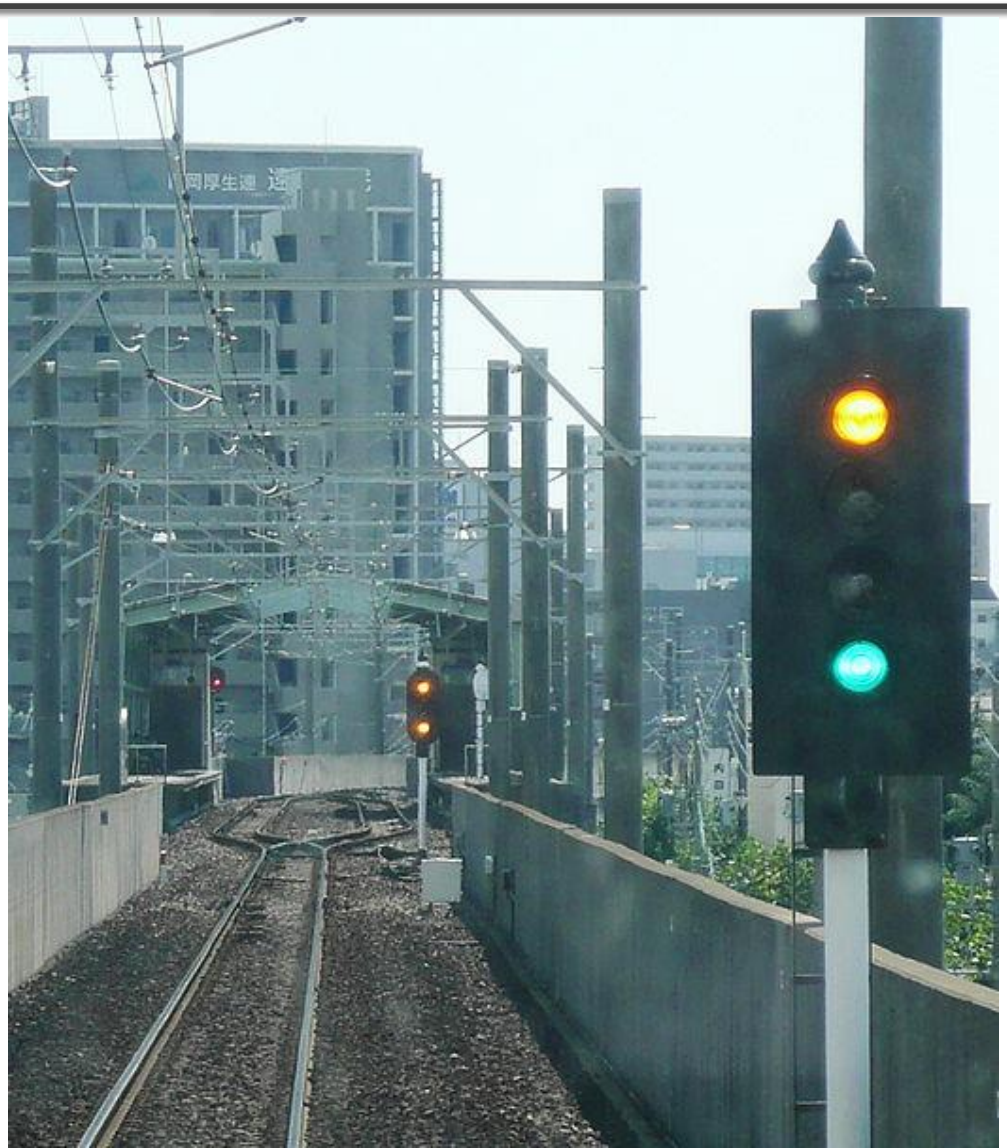
Italian Railway

Traffic Light

Traffic Light



Transit Lights



Switch Machine

Inductive System for Signaling on Rail

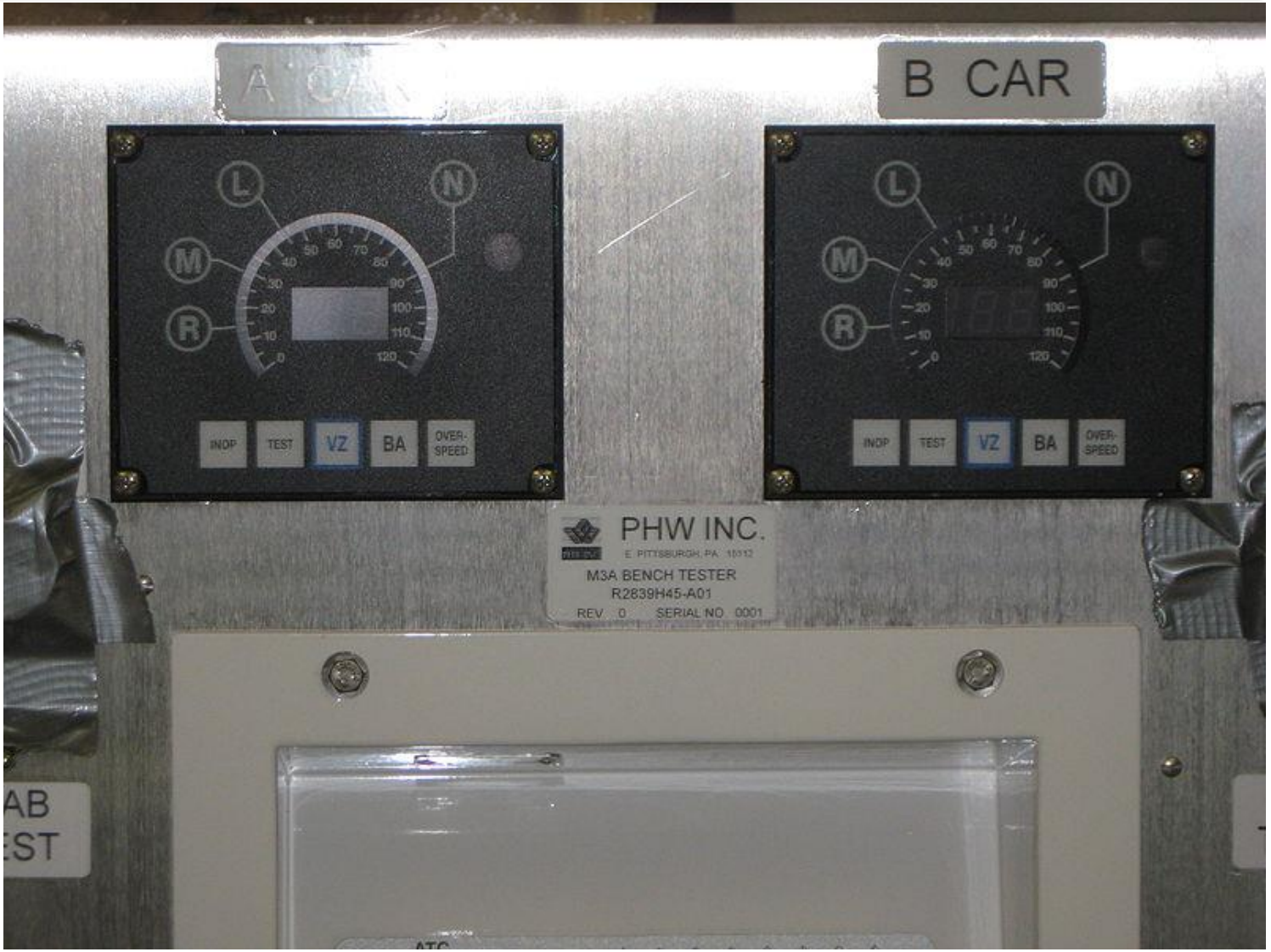
Train Controller

- ✧ Picks up signal from track circuit
- ✧ Decodes signal for Authority and Speed Limit
- ✧ Takes input from the Driver – power or speed
- ✧ Regulates the train to the correct speed
- ✧ Ensures train does not exceed speed limit
- ✧ Ensures train does not exceed authority
- ✧ Controls of vital and non-vital may be separated
- ✧ In transit, non-vital operations are operating doors, annunciation system, lights, air conditioning, route information, and advertisements

Los Angeles Greenline

Cab Pickup Coil

Metro North Cab Signaling Controller



Train Control

- ✧ Control center issues a Suggestion in USA
- ✧ Track receives Suggestion
  - Which track at what speed
- ✧ Track implements Suggestion if it is safe
- ✧ Train received Command from Track
- ✧ Train implements speed and authority if it is safe

Positive Train Control – Railroad not Transit

- A few years after a UP crashed two trains head-on government started to push PTC
- Began implementing new location device – PTC for dark territories
- PTC as an overlay in lighted territories
- Accelerometer + Gyro + GPS
- Still implementing new PTC today – progress is slow.

Go to Union Switch & Signal on Wikipedia for more information

Union Switch & Signal - Wikipedia, the free encyclopedia - Windows Internet Explorer

http://en.wikipedia.org/wiki/Union_Switch_%26_Signal

union switch and signal track circuit

File Edit View Favorites Tools Help

Favorites | Union Switch & ... | Log In | Page | Safety | Tools

Log in / create account

Article Talk | Read Edit View history | Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
日本語

Union Switch & Signal

From Wikipedia, the free encyclopedia

This article includes a list of references, related reading or external links, but **its sources remain unclear because it lacks inline citations**. Please improve this article by introducing more precise citations. *(October 2010)*

Union Switch and Signal (US&S) was a supplier of railway signaling equipment, systems and services in Pittsburgh, Pennsylvania. As of January 1, 2009, US&S is known as Ansaldo STS USA.

Contents [hide]

History [edit]

Union Switch & Signal

Industry	Rail transport
Fate	Merged with Ansaldo STS [1]
Predecessor (s)	Union Electric Signal, Interlocking Switch & Signal
Successor(s)	Ansaldo STS
Founded	1881 [2]
Founder(s)	George Westinghouse [2]
Defunct	2009 [1]
Headquarters	Pittsburgh, U.S.
Area served	Worldwide
Products	Railway signalling equipment, communication systems and services
Employees	500

Done | Internet | Protected Mode: On | 125%

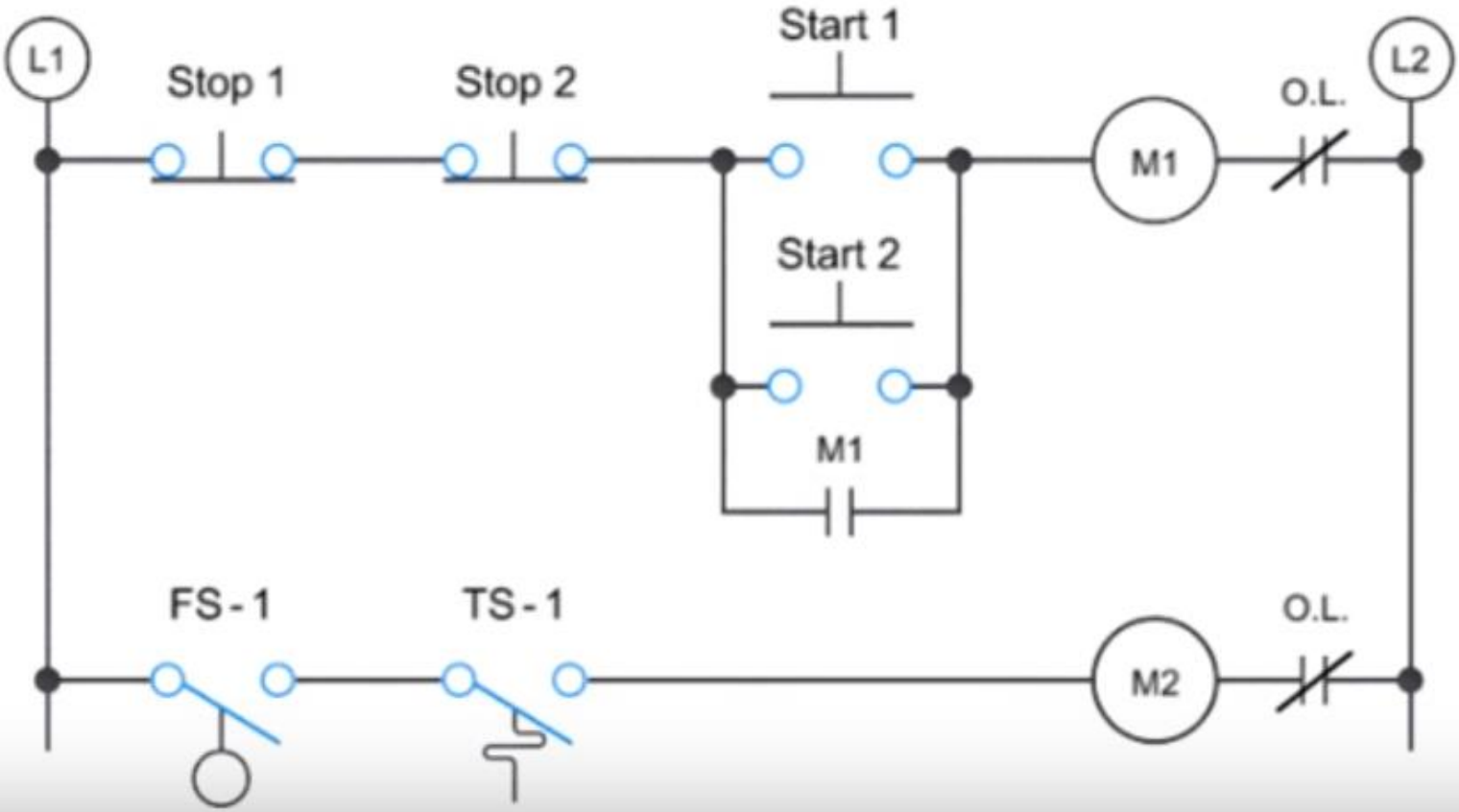
C:\Users\Joe\Deskto... | Train Control System... | Union Switch & Sign... | 9:49 PM

The Track Controller

The Track Controller

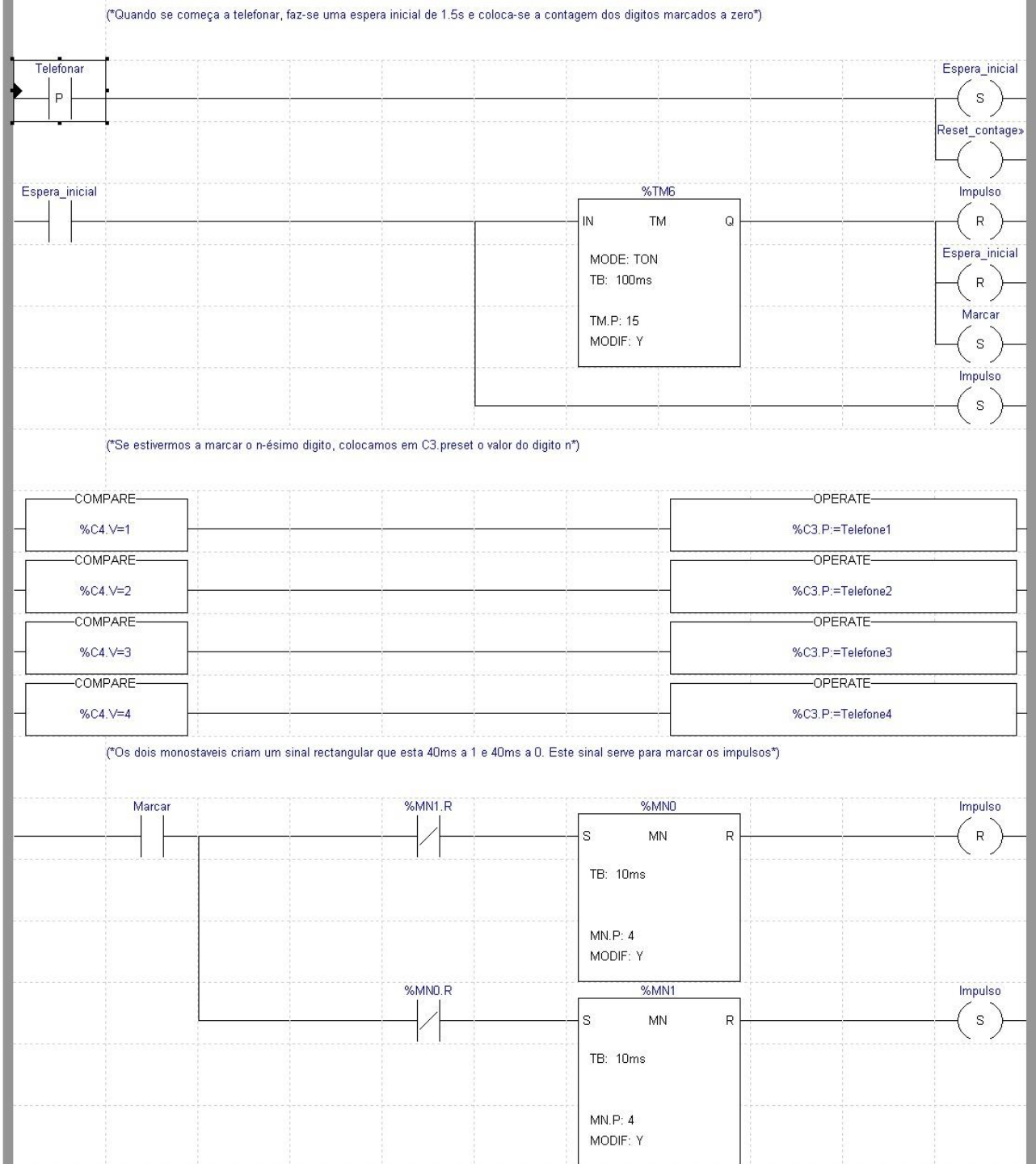
- The Track Controller is a vital computing platform
- The Track Controller is sometimes refer to as the Wayside Controller
- The Track Controller is a vital PLC
  - As such, the program it runs is boolean logic
  - Inputs to the logic are the track occupancies and switch positions
  - Outputs are switch positions, light colors, and authority
- The Track Controller also is a communications platform that passes receives information from the office and sends it through the track to the train.

PLC – Ladder Diagram



PLC – Function Block Diagram

- By Nuno Nogueira (w:User:Nmnogueira) - Own work, CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=4485176



The Train Model

The Engine Model

$$P = F\ v$$

- P = power
- F = force
- v = velocity

The Point Mass Model

$$F = m\,a$$

- ✧ *F* = force
- ✧ *m* = mass
- ✧ *a* = acceleration

Laplace Transform

✧ Remember from Basic Circuits

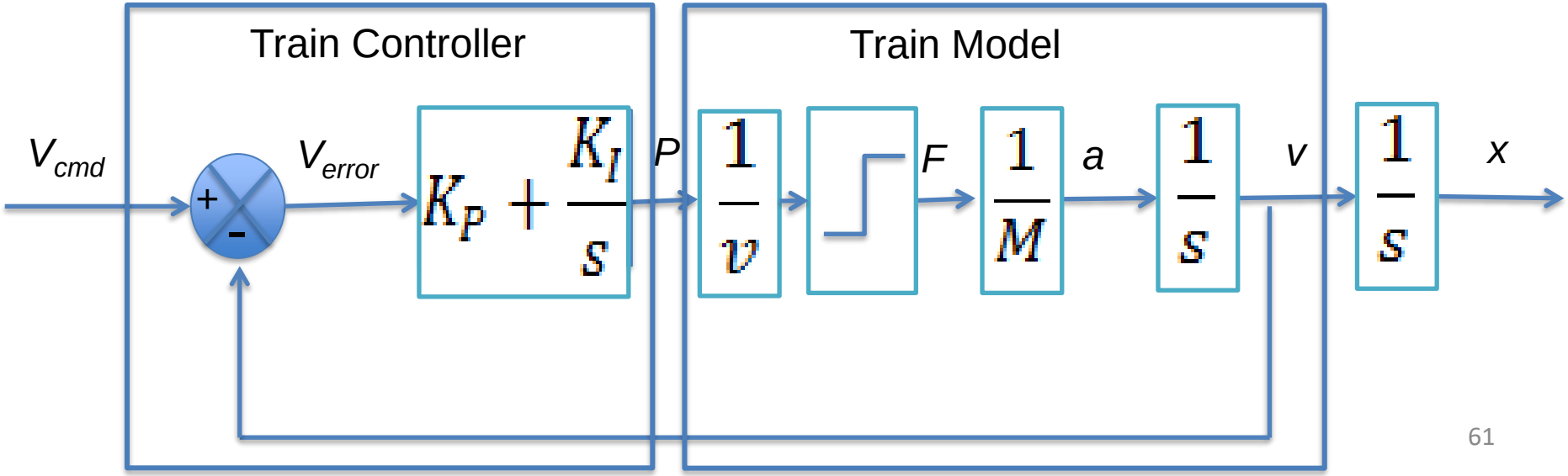
$$\mathfrak{L}[f(t)] = F(s) = \int_{t=0^-}^{t} f(t)e^{-st}dt$$

✧ So the Laplace Transform of an integral is

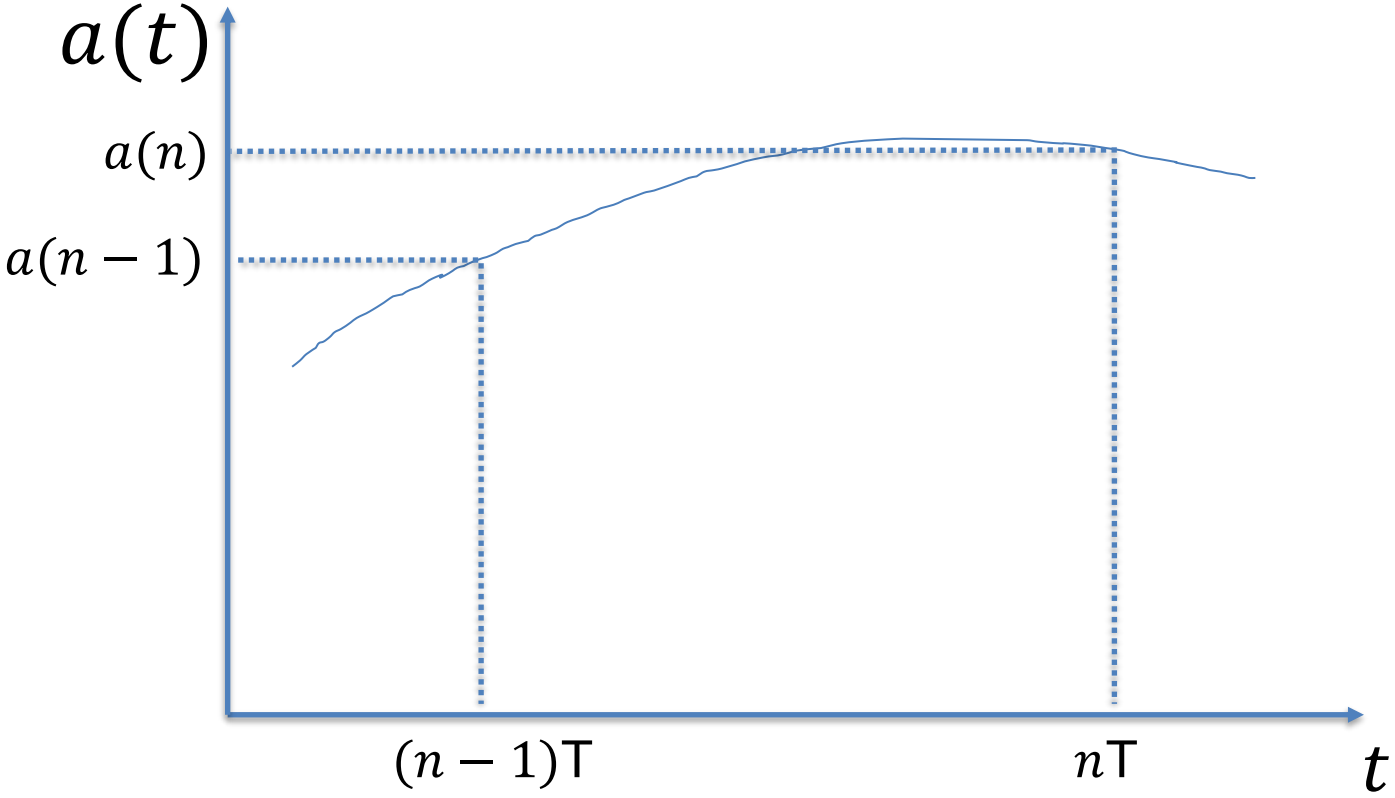
$$\mathfrak{L}\left[\int_{t=0^-}^{t} f(t)dt\right] = \frac{F(s)}{s}$$

The Train Model

- ✧ P = power
- ✧ M = mass
- ✧ V_{cmd} = Velocity command or the Setpoint from the driver
- ✧ V_{error} = Velocity error
- ✧ F = force
- ✧ a = acceleration
- ✧ *v* = velocity
- ✧ x = position



Derivation of Train Model



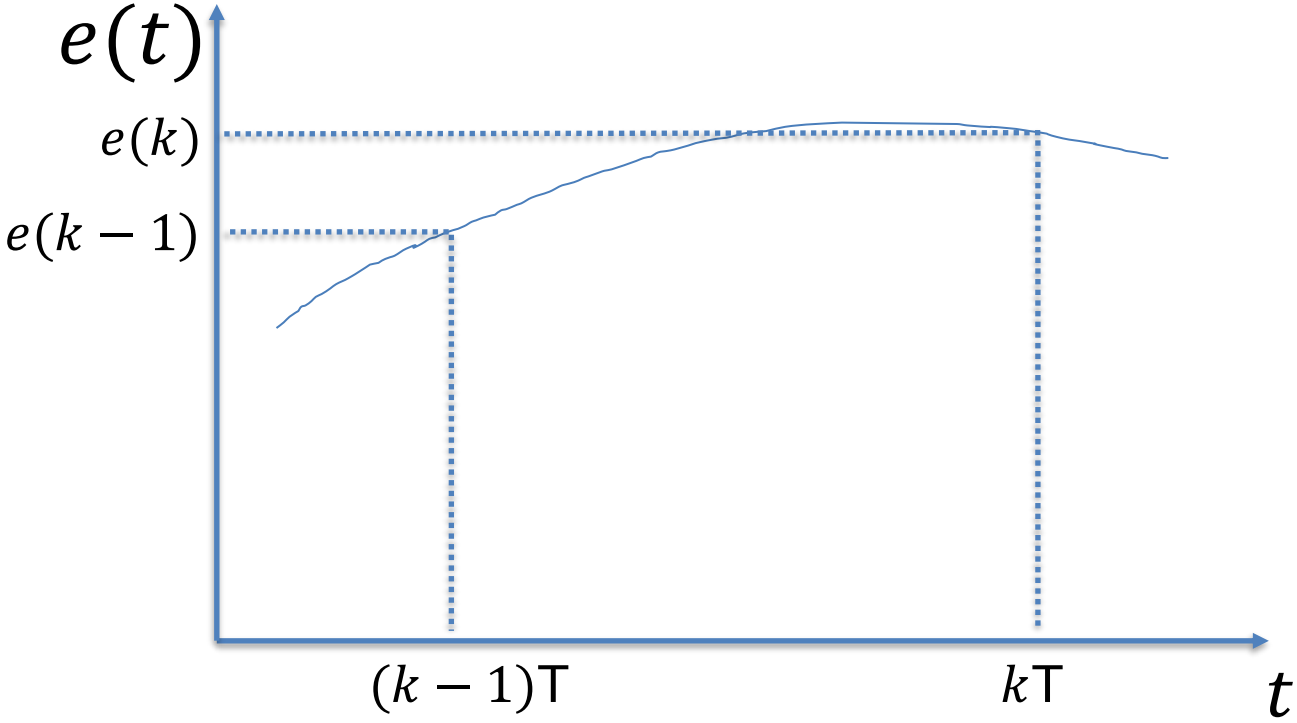
$$v_n = v_{n-1} + \frac{T}{2}(a_n + a_{n-1})$$

This Train Model is Incorrect

$$d = d_0 + v_0 t + \frac{1}{2} a t^2$$

https://youtu.be/HBxcalnSzIk

Derivation of Control Law



$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1})$$

The Control Law

- If $P^{cmd} < P^{max}$ then

$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1})$$

- If $P^{cmd} \geq P^{max}$ then

$$u_k = u_{k-1}$$

- Final calculation

$$P_k^{cmd} = K_p e_k + K_i u_k$$

- P^{cmd} = power command
- P^{max} = maximum power of engine
- T = sample period of train model
- e_k = kth sample of velocity error
- K_p = proportional gain
- K_I = integral gain

What Remains?

- You need to select K_p and K_I such that the system is stable.

Advice from Your Peers

From the Final Presentations of Teams in the Last 14 Classes

Biggest Lesson Learned

- Communication is key to working on a large group project

What would we do differently

- Start design early!
  - Understand the sub-systems and their input/output before trying to code

Lessons Learned

- Start early
- Integration takes MUCH longer than expected

If we were to do it again…

- Use an iterative development method, not waterfall
- Hard deadlines for the each module

After-Action Review (AAR)

Things We did Well:

- Met weekly (outside of lab)
- Broke up into “Black Team”/”Red Team”
- Shared Files on Box

Things We could Improve:

- Work schedule – should’ve used Gantt Software
- Limited automated testing

Takeaway:

We should have tested as we built functionality and started programming ~2 weeks sooner.

Biggest Lessons Learned

- Need To Get Modules Done Much Earlier
- Need a better way of holding teammates accountable for deadlines and standards
- Better time management

What We Would Do Next Time

- Start coding earlier
- Set hard deadlines for modules being done
- Ask more questions
- Track bugs in a better way
- Track time in a better way

Biggest Lesson Learned

We should not have put off code development and integration until the final two weeks just because we think we had a good understanding of the requirements and design.

What We Would Do Differently

- Make sure every team member learns how to use GitHub before the coding phase.
- Begin the code development and integration phases earlier.
- Have more full team participation during development and integration.

What Would You Do Differently?

- Kevin - Use a different UI building framework for java. I spent too much time writing Swing/AWT code
- Zach - I need to break up code more, breaking large classes into smaller ones. My small classes work, and my large ones tend to get cluttered and hard to debug.
- Alex - Get the test UI working first, then move to the simulation UI.
- Matt - Remove non display code from Train+Train Controller UI.
- Ryan - Use the Event Center. Integration went well for the Train Controller, but could have been smoother when writing independent UI.
- Team - More testing, both at the unit test level and the integration level. Preferably automated.

What to do differently for 2nd time

- Use a more agile approach
- Rapid Prototyping
- Focus more on design
- Do not meet for the sake of meeting, have a goal or purpose to every meeting
- Meet deadlines set by team

Resolutions for “Next Time”

- Start coding earlier, more strict schedule
- Strategic resource and personnel allocation
- More group meeting times
- Create descriptive interfaces and adhere to them strictly

Biggest Lesson Learned

- Besides the obvious of starting the coding part earlier, we realize now communication is the key to success. The smallest difference in expectations of other team members can lead to hours of refactoring.
- This includes communication with the client as well!

What We Would Do Differently

- We would try to figure out as early as possible what each member needed from the others so that everyone was on the same page.
- Write code little by little and test each chunk before moving forward
- We would definitely have a better understanding of what was desired from the documentation the second time around
- Try to actually stick to the coding standard
- Make more automatic tests earlier

What we would do differently

- Spend more time on the design of the system
- Start development earlier
  - Started development 4 days before the prototype demo

Common Advise

- Use an Agile Approach
- Start Coding Early (Prototyping)
- Spend More Time in Design
- Be Sure to Understand the Interfaces
- Communication is Key

Second Chance?

- Learn content managements systems (GitHub)
- Better Documentation
- Better Communication of Requirements
- GUI designer
- We estimated a short amount of time to implement this project which was around 30 hours per developer. We were wrong.

Things we would do differently:

-Communicate and express ideas more to each other
-Meet up as a group more often and try to integrate the whole system as much as possible
-Have set deadlines and meeting times

If we were to do it again…

- Use an iterative development method, not waterfall
- Hard deadlines for the each module
- Start early
- Integration takes MUCH longer than expected

Takeaway

Biggest lessons learned

Thou hast best check thyself, before thou wrecks thyself

What should be done differently

Group coordination is never at maximum unless everyone is in the same group. Were we to do the project again we would all attempt to meet more than once a week outside of class or lab, despite communication tools that exist today.

What would we do differently?

- Get minimum functional systems up and running ASAP
  - Some people finished modules earlier than others and had to wait around… Work on minimum functionality first and integrate with sub-optimal solutions
- Design sub-systems together.
  - Example: Track traversal methods worked for train movement but not easy Wayside access… this could have been avoided
- Use better systems-engineering integration techniques
  - Initial order of integration: All at once… FAILED
  - Successful order of integration: Train -> Train Controller -> Track -> Wayside Controller -> CTC -> MBO
  - Slowly build

Takeaways and Biggest Lessons Learned

- Continuous integration - Began integration week 4
- Onsite customer - Continuous feedback every few weeks
- Peer Programming - Live programming at meetings
- Small releases - Making sure each successive build worked (613 Commits)
- Sustainable pace - No stress towards end of project
- Common ownership - Everyone pushed to master (for better or worse)
- Refactoring - Code reviews and making changes as needed
- Incremental planning - Work packages, and design meetings each friday
- Communication is key - Slack, 3 weekly meetings, common “to do” list
- Utilize Backups - Needed to use backup configuration management on data corruption

- Global Unit Conversion - (should have) Operated on Metric and displayed Imperial
- Utilize Javadocs and Comments - Make code more readable by peers
- Failed to create proper documentation for issues when bugs arose