Commute Compute System

Project Report

A Self-Hosted Smart Transit Display for Australian Public Transport

176 Source Files
76,445 Lines of Code

Author: Angus Bergman

Date: January 2026

Version: 1.1

Table of Contents

Commute Compute System™

Complete Project Overview

Version: 1.1 **Date:** January 2026 **Author:** Angus Bergman **License:** CC BY-NC 4.0

Executive Summary

I built **Commute Compute System™** as a fully self-hosted smart transit display for Australian public transport. It delivers real-time journey information to e-ink displays, answering the daily questions every commuter faces: "When should I leave?", "Is my train delayed?", "Do I have time for coffee?", and "Should I bring an umbrella?"

I designed the system to be completely free to deploy, require zero ongoing costs, and respect user privacy by keeping all data on the user's own server.

Part 1: Project Scale

What I Built

Metric	Value
Total Source Files	176
Total Lines of Code	76,445

Code Breakdown

Language	Lines	What I Used It For
JavaScript	31,243	Server, API endpoints, rendering engine
HTML	18,165	Admin panel, setup wizard, simulators
Markdown	23,960	21 documentation files
C++	3,077	Custom ESP32-C3 firmware

Key Components I Created

Component	Count
API Endpoints	18
Service Modules	15
Renderer Versions	6
HTML Pages	9
Firmware Variants	4

Part 2: Vision & Goals

The Problem I Wanted to Solve

Every morning, I faced the same uncertainty:

- "When should I leave?" — Depends on real-time delays
- "Is my train delayed?" — Had to check multiple apps
- "Do I have time for coffee?" — Mental math every day
- "Should I bring an umbrella?" — Another app to check

I wanted a single glance at an e-ink display to answer all of these.

Core Principles I Followed

Principle	How I Implemented It
Privacy First	All data stays on user's own server
Truly Free	Runs entirely on Vercel free tier
Zero Dependencies	Custom firmware connects only to user's server
Australian Focus	Built for VIC, NSW, QLD transit systems
Open Source	All code freely available

Brands I Created

Brand	Purpose
Commute Compute System™	Overall system name
SmartCommute™	Journey calculation engine
CCDash™	Dashboard rendering specification
CC LiveDash™	Multi-device live renderer
CCFirm™	Custom firmware family

Part 3: Technical Challenges I Solved

Challenge 1: E-ink Display Constraints

The Problem I Faced

E-ink displays have severe limitations:

- 1-bit color only (pure black and white)
- Slow refresh (2-3 seconds full refresh)
- Ghosting from previous images
- No anti-aliasing possible

How I Solved It

I implemented **server-side rendering** with **zone-based partial refresh**:

```
My Zone Layout (800x480 display):
```

```
+----------------------------------+
<- Updates for time/weather +----------------------------------+ <- Updates for status +---------------
-------------------+ <- Updates for delays +----------------------------------+ <- Rarely updates +----
-----------------------------+
```

Each zone refreshes independently. Full cycle: 20 seconds. Partial refresh: ~500ms per zone. `

Challenge 2: ESP32-C3 Memory Constraints

The Problem I Faced

The TRMNL device uses an ESP32-C3 with severe limitations:

- Only 400KB RAM total
- No external PSRAM
- Single core processor
- Very easy to brick with wrong code

How I Solved It

I developed a **zone batching** approach and **12 anti-brick rules**:

` FORBIDDEN (causes brick): - deepSleep() in setup() - Network calls in setup() - Delays longer than 2 seconds - Watchdog timer enabled

REQUIRED (safe pattern): - setup() completes in <5 seconds - State machine in loop() - Non-blocking operations only - Brownout detection disabled `

I documented all 12 rules in DEVELOPMENT-RULES.md Appendix D.

Challenge 3: Zero-Config Serverless Deployment

The Problem I Faced

- Users should never edit environment variables
- Vercel serverless has no persistent storage
- Each request is completely stateless
- API keys need to be secure but accessible

How I Solved It

I created a **Config Token Architecture** where all configuration is encoded in the URL:

`SETUP TIME: User enters data in Setup Wizard | v All config encoded to base64 token | v Token becomes part of device URL`

`RUNTIME: Device requests: /api/device/eyJhIjp7... | v Server decodes token from URL | v All config available instantly | v No storage needed!`

`Token structure I designed:`

`{ "a": addresses, "l": lat/lng locations (cached), "s": state (VIC/NSW/QLD), "t": arrival time, "c": coffee enabled, "k": transit API key, "cf": cached cafe data, "m": API mode }`

`Challenge 4: Free-Tier Architecture`

`The Problem I Faced`

- `Google Places API costs ~$0.02 per call`
- I didn't want users to need paid APIs
- But address autocomplete significantly improves UX

`How I Solved It`

`I implemented` **`setup-time caching`** `with` **`free fallbacks`**`:`

`FREE MODE (default): - OSM Nominatim for geocoding (free) - Default cafe hours (free) - All data cached at setup time - $0 runtime cost`

`LIVE MODE (optional): - Google Places for geocoding - Real cafe business hours - Only for users who want it`

`Challenge 5: Multi-State Transit APIs`

`The Problem I Faced`

`Each Australian state has different transit APIs:`

- `Different authentication methods`
- Different data formats
- Different rate limits

`How I Solved It`

`I built the` **`SmartCommute™ Engine`** `as a unified abstraction:`

`State`

`Auth Method --------------------------- VIC KeyId header (UUID) NSW Authorization header QLD X-API-Key header`

The engine handles all state-specific logic internally. Users just specify their state.

Challenge 6: Real-Time Disruption Handling

The Problem I Faced

- Services get suspended (signal faults)
- Trams get diverted (roadworks)
- Trains get cancelled
- Users need to know immediately

How I Solved It

I implemented **automatic route adaptation**:

```
` NORMAL JOURNEY: Walk -> Train -> Walk

SIGNAL FAULT DETECTED: Walk -> [SUSPENDED] -> Walk | v System auto-inserts replacement: Walk -> Rail Replacement Bus -> Train -> Walk

DISPLAY SHOWS: "DISRUPTION -> Arrive 8:52 (+18 min)" [Diagonal stripes on cancelled service] [Rail replacement automatically added] `
```

Challenge 7: The Coffee Decision

The Problem I Faced

- Users want coffee but don't want to be late

```
```

- Coffee time varies (5-15 minutes)
- Should skip coffee if running late
- Different patterns needed (origin, interchange, destination)

```
```

How I Solved It

I built the **CoffeeDecision Engine**:

```
` INPUTS: - Current time - Journey duration - Target arrival time - Coffee duration preference - Cafe business hours (cached) - Current delays

DECISION LOGIC: IF coffee_enabled AND cafe_is_open AND (journey + coffee + delays) <= arrival_buffer THEN -> Insert coffee leg -> Status: "TIME FOR COFFEE" ELSE IF delays_detected AND would_be_late_with_coffee THEN -> Skip coffee leg -> Status: "SKIP - Running late" `
```

Challenge 8: bb_epaper Library Bugs

The Problem I Faced

After weeks of debugging, I discovered ESP32-C3 specific bugs:

- allocBuffer() causes garbage display
- FONT_12x16 renders text rotated 90°
- No documentation existed for these issues

How I Solved It

I documented my findings and implemented workarounds:

```
BROKEN (causes garbage): bbep.allocBuffer(true); bbep.setBuffer(buf);

WORKING (what I use): bbep.fillScreen(BBEP_WHITE); bbep.setFont(FONT_8x8); // NOT FONT_12x16!
bbep.drawString("Hello", 0, 0); bbep.refresh(REFRESH_FULL, true);
```

I added all findings to DEVELOPMENT-RULES.md Appendix D.

Challenge 9: Vercel Serverless Font Rendering

The Problem I Faced

- Vercel serverless functions have no system fonts
- canvas.fillText() silently fails
- Text renders as completely blank

How I Solved It

I bundled fonts and registered them explicitly:

```javascript
// I register fonts BEFORE any canvas operations
GlobalFonts.registerFromPath('./fonts/Inter-Bold.ttf', 'Inter');

// Now I use the registered font name
ctx.font = '800 17px Inter'; // NOT 'sans-serif' - that fails silently!
```

Challenge 10: iOS Safari Compatibility

The Problem I Faced

- `Safari validates forms even with` novalidate
- `Relative URLs fail on mobile Safari`
- `Pattern validation throws errors`

`How I Solved It`

`I added explicit attributes on all form elements:`

`` `html ``

Submit

`const url = window.location.origin + '/api/endpoint'; ` `

`Part 4: System Architecture`

`The Distribution Model I Designed`

`` ` SELF-HOSTED MODEL: ``

`Official Repo User's Fork User's Server +-----------+ +-----------+ +-----------+`

HEADER (0-94px)				
SUMMARY (96-124px)				
JOURNEY LEGS (132-448px)				
FOOTER (448-480px)				
Authority				
Transport Victoria				
Transport for NSW				
TransLink				
GitHub	Fork	User's	Deploy	Vercel
Public	------>	Copy	------>	(Free)

`+-----------+ +-----------+ +-----+-----+ | v +-----------+`

User's
`E-ink`
`Display`

```
+-----------+
```

KEY BENEFITS: - Complete data isolation - User owns all API keys - Zero ongoing costs - No external cloud dependencies `

The Data Flow I Implemented

` TRANSIT API (VIC/NSW/QLD) | v (30-second cache) SMARTCOMMUTE ENGINE | +---> Weather (BOM) | +---> Coffee Decision | +---> Disruption Detection | +---> Express Detection | v CCDASH V10 RENDERER | v (800x480 1-bit BMP) E-INK DISPLAY | v (sleep 20 seconds) REPEAT `

Part 5: The SmartCommute™ Engine

What I Built

SmartCommute™ is the journey calculation engine. It:

- Fetches real-time data from transit authorities
- Detects delays and disruptions
- Calculates multi-modal routes
- Inserts coffee stops when timing permits
- Adapts to disruptions automatically
- Detects express services that save time

States I Support

State	Authority	Status
Victoria	Transport Victoria	Production
NSW	Transport for NSW	Supported
Queensland	TransLink	Supported
South Australia	Adelaide Metro	Planned
Western Australia	Transperth	Planned

The Decision Flow I Designed

` START | v Fetch GTFS-RT data (30s cache) | v Check service alerts | +---> Suspended? -> Insert replacement + DISRUPTION | +---> Diverted? -> Add walk leg + DIVERSION | +---> Cancelled? -> Show next service | v Check delays | +---> Single delay? -> DELAY status | +---> Multiple? -> DELAYS status (plural)

| v Coffee decision | +---> Time available? -> "TIME FOR COFFEE" | +---> Running late? -> "SKIP - Running late" | +---> Extra buffer? -> "EXTRA TIME" | v Express detection | +---> Saves time? -> Show EXPRESS badge | v Weather check | +---> Rain likely? -> "BRING UMBRELLA" | v Render CCDash V10 layout | v Send to device `

Part 6: Dashboard Scenarios

I designed the system to handle these real-world scenarios:

Scenario 1: Normal Commute with Coffee

Image: scenario-normal-coffee.png

- Location: 1 Clara St, South Yarra
- Time: 7:45 AM Tuesday
- Weather: 22° Sunny

What the engine calculated:

- Total journey: 47 minutes
- Coffee: "TIME FOR COFFEE" (sufficient buffer)
- 5 legs: Walk -> Coffee -> Walk -> Train -> Walk

Visual indicators:

- Solid borders = normal service
- Coffee checkmark = confirmed
- Status: "LEAVE NOW -> Arrive 8:32"

Scenario 2: Delay with Coffee Skip

Image: scenario-delay-skip.png

- Location: 1 Clara St, South Yarra
- Time: 8:22 AM Monday
- Weather: 17° Rain

What the engine calculated:

- Train delayed +8 minutes
- With coffee would be late
- Decision: Skip coffee

Visual indicators:

- Dashed border on coffee = SKIP
- "SKIP - Running late" subtitle
- Status: "DELAY -> Arrive 9:18 (+8 min)"

Scenario 3: Express Service

Image: scenario-express.png

- Location: Caulfield Station
- Time: 6:48 AM Monday
- Weather: 14° Fog

What the engine calculated:

- Express skips 6 stations
- Saves 8 minutes vs all-stops
- Shows alternatives for comparison

Visual indicators:

- "EXPRESS" badge
- "Skips 6 stations" note
- "EXPRESS saves 8 min" footer

Scenario 4: Tram Diversion

Image: scenario-diversion.png

- Location: Richmond Station
- Time: 5:45 PM Wednesday

- Weather: 31° Hot

What the engine calculated:

- Tram 70 diverted (roadworks)
- Extra walking leg added
- Bus alternative shown

Visual indicators:

- Arrow: "← Tram 70 Diverted"
- Extra leg: "← Walk Around Diversion"
- Status: "TRAM DIVERSION -> Arrive 6:38 (+5 min)"

Scenario 5: Multi-Modal (Tram + Bus)

Image: scenario-multimodal.png

- Location: 42 Chapel St, Windsor
- Time: 2:30 PM Saturday
- Weather: 28° Hot

What the engine calculated:

- Tram 78 to Richmond
- Walk to bus stop
- Bus 246 to Elsternwick

Visual indicators:

- Different icons (tram vs bus)
- Walking legs between modes
- "Next: 4, 12 min" frequency

Scenario 6: Major Disruption

Image: scenario-disruption.png

- `Location: 1 Clara St, South Yarra`

```
```

- Time: 7:20 AM Thursday
- Weather: 19° Overcast

```
```

`What the engine calculated:`

- `Sandringham Line SUSPENDED`

```
```

- Rail replacement bus inserted
- Extra buffer = coffee added

```
```

`Visual indicators:`

- `Diagonal stripes = CANCELLED`

```
```

- "SUSPENDED - Signal fault"
- Rail replacement bus added
- Coffee: "EXTRA TIME - Disruption"

```
```

`Scenario 7: Multiple Delays`

`Image: scenario-multiple-delays.png`

- `Location: Malvern Station`

```
```

- Time: 8:15 AM Tuesday
- Weather: 15° Showers

```
```

`What the engine calculated:`

- `Train: +10 minutes delay`

```
```

- Tram: +5 minutes delay
- Combined: +15 minutes

```
```

`Visual indicators:`

- `Multiple dashed borders`

```
```

- Individual delays shown
- Status: "DELAYS -> Arrive 9:22 (+15 min)"

```
```

Scenario 8: Friday Treat

Image: scenario-friday.png

- Location: 80 Collins St (work)
- Time: 6:20 PM Friday
- Weather: 23° Warm

What the engine calculated:

- Reverse commute (work to home)
- Coffee at destination
- "Friday Treat" status

Visual indicators:

- "HOME" in footer
- "FRIDAY TREAT" on coffee
- Destination coffee pattern

Scenario 9: Weekend Leisure

Image: scenario-weekend.png

- Location: Flinders St Station
- Time: 11:15 AM Sunday
- Weather: 24° Sunny

What the engine calculated:

- Leisure destination (park)
- Simple route to picnic spot
- No coffee (leisure pattern)

Visual indicators:

- "CAULFIELD PARK ROTUNDA" destination
- "Near the rotunda" description

- Weekend formatting

Part 7: Visual Design System

The Layout I Designed (CCDash V10)

```
+----------------------------------------+
```

HEADER (0-94px)
LOCATION TIME (64px) WEATHER
Small caps 7:45 AM 22° Sunny
Tuesday NO UMBRELLA

```
+----------------------------------------+
```

SUMMARY BAR (96-124px)
LEAVE NOW -> Arrive 8:32 47 min

+--+

JOURNEY LEGS (132-448px)		
+------------------------------------+		
	1 Walk to Norman Cafe 4 MIN	
	From home WALK	
+------------------------------------+		
v		
+------------------------------------+		
	2 Coffee at Norman ~5 MIN	
	TIME FOR COFFEE	
+------------------------------------+		
v		
+------------------------------------+		
	3 Walk to South Yarra Stn 6 MIN	
	Platform 1 WALK	
+------------------------------------+		
v		
+------------------------------------+		
	4 Train to Parliament 5 MIN	
	Sandringham - Next: 5, 12 min	
+------------------------------------+		
v		
+------------------------------------+		
	5 Walk to Office 26 MIN	
	Parliament -> 80 Collins WALK	
+------------------------------------+		

+--+

FOOTER (448-480px)
80 COLLINS ST, MELBOURNE ARRIVE 8:32

```
+----------------------------------------+
```

Visual States I Defined

State	Border	Meaning
Normal	Solid 2px black	Service running normally
Delayed	Dashed 2px gray	Service delayed
Skip	Dashed 2px gray	Coffee skipped
Cancelled	Diagonal stripes	Service suspended

Status Bar Variants

Status	Format
Normal	LEAVE NOW -> Arrive X:XX
Leave Soon	LEAVE IN X MIN -> Arrive X:XX
Delay	DELAY -> Arrive X:XX (+X min)
Delays	DELAYS -> Arrive X:XX (+X min)
Disruption	DISRUPTION -> Arrive X:XX (+X min)
Diversion	TRAM DIVERSION -> Arrive X:XX (+X min)

Mode Icons

Icon	Mode
Walking figure	Walk
Train	Train
Tram	Tram
Bus	Bus

Coffee cup	Coffee		Device	Resolution	Status
		Part 8: Supported Devices			
		TRMNL Devices (Primary)			
TRMNL OG	800x480	Primary target			
TRMNL Mini	400x300	Supported	**Kindle Devices (Jailbreak Required)**	Device	Resolution	Status
Kindle Paperwhite 5	1236x1648	Supported			
Kindle Paperwhite 3/4	1072x1448	Supported			
Kindle Voyage	1072x1448	Supported			
Kindle Basic	600x800	Supported	**Planned Devices**	Device	Resolution	Status
Inkplate 6	800x600	Planned			
Inkplate 10	1200x825	Planned			
Waveshare 7.5"	800x480	Planned			

Part 9: Setup & Deployment

The Process I Designed

```
Step 1: Fork the repository | v Step 2: Deploy to Vercel (one click) | v Step 3: Run Setup Wizard - Enter addresses - Select state (VIC/NSW/QLD) - Set arrival time - Optional: Add API keys | v Step 4: Flash CCFirm to device | v Step 5: Enter webhook URL | v Done! Device displays your commute.
```

Cost Breakdown

Service	Setup Cost	Runtime Cost
Vercel Hosting	Free	Free
Transit API	Free	Free
BOM Weather	Free	Free
OSM Geocoding	Free	Free
Google Places	~$0.02 (optional)	Free (cached)
Total	**$0 - $0.10**	**$0**

Phase	Items
Foundation	Core server, V10 spec, zone refresh, VIC API, weather
Firmware	CCFirm, anti-brick rules, state machine
Documentation	DEVELOPMENT-RULES v1.6, Architecture v4.0
Multi-State	SmartCommute, CC LiveDash, NSW, QLD support
Setup UX	Zero-config, free-tier caching, iOS compatibility

Part 10: Roadmap

What I've Completed

What I'm Working On

- End-to-end automated testing
- Additional device support
- Error handling improvements

What I've Planned

- South Australia, Western Australia, Tasmania

- Inkplate and Waveshare support
- Video tutorials
- Public launch

Summary

I built **Commute Compute System™** over 18 months, solving real problems:

- **76,445 lines of code** across 176 files
- **10 major technical challenges** solved
- **3 Australian states** supported
- **9 device types** compatible
- **Zero ongoing costs** for users
- **Complete privacy** — data stays on user's server

The system succeeds when a commuter can glance at their e-ink display, see "LEAVE NOW — Coffee included", and walk out the door knowing they'll catch their train on time.

Built with coffee in Melbourne
