

Bitcoin and Weather Watchface

Software Architecture Documentation

Version 5.1.0

1. Project Overview

The Bitcoin and Weather watchface is a Pebble smartwatch application that displays real-time Bitcoin prices and weather information alongside traditional time and date. The application supports multiple Pebble watch models and integrates with external APIs for cryptocurrency pricing and weather data.

1.1 Key Features

- Real-time Bitcoin price tracking from Kraken exchange (CAD, USD, EUR)
- Weather data from Environment Canada or Open-Meteo APIs
- Historical Bitcoin price graph visualization (60 data points)
- Configurable display options (seconds ticker, health tracking)
- GPS-based automatic location detection
- Metric and imperial unit support
- Battery and Bluetooth status indicators

1.2 Supported Platforms

- Aplite (Pebble Classic)
- Basalt (Pebble Time)
- Chalk (Pebble Time Round)
- Diorite (Pebble 2)
- Emery (Pebble Time 2)

2. Architecture Overview

The application follows a three-tier architecture with clear separation between the watchface (C), JavaScript communication layer (PebbleKit JS), and external services.

Layer	Description
Watchface (C)	Native C code running on the watch. Handles UI rendering, user input, time updates, and persistent storage.
PebbleKit JS	JavaScript layer running on the connected phone.

Layer	Description
	Manages API calls, GPS location, and data formatting.
External APIs	Kraken (Bitcoin), Environment Canada / Open-Meteo (Weather), OpenStreetMap Nominatim (Geocoding)

3. Component Architecture

3.1 Watchface Layer (main.c)

The C watchface is organized into the following functional modules:

3.1.1 Display Components

- **Time Layer:** Large central display showing current time (12/24hr format)
- **Date Layer:** Shows day and date (e.g., 'Mon 19')
- **Bitcoin Layers:** Current price (bc_layer), 24h high (bcH_layer), 24h low (bcL_layer)
- **Graph Layer:** 60-point historical Bitcoin price visualization
- **Weather Layer:** Temperature, wind data, weather icons, forecast information
- **Trotteuse Layer:** Optional seconds ticker bar (configurable)
- **Status Indicators:** Battery level and Bluetooth connection

3.1.2 Data Structures

- **WeatherLayer:** Composite structure containing 5 temperature text layers, wind layer, 2 weather icon bitmap layers, and status indicators
- **Graph Data:** GPath structure with 120 points (60 points × 2 for path closure) storing normalized Bitcoin price history
- **Configuration:** Global variables for user preferences (exchange, weather service, units, display options)

3.1.3 Event Handlers

- **Tick Handler:** handle_minute_tick() - Updates time display and triggers periodic data fetches
- **Message Handler:** Receives Bitcoin and weather data from JavaScript layer
- **Battery Handler:** Monitors battery state and updates display
- **Bluetooth Handler:** Tracks connection status and triggers vibration alerts

3.1.4 Update Mechanisms

The watchface uses a configurable cadence system for data updates (default: 3 minutes). Updates are aligned to clock intervals (0, N, 2N, 3N minutes after the hour) to prevent duplicate fetches. A splash screen timer triggers the initial data fetch 3 seconds after launch.

3.2 PebbleKit JS Layer

3.2.1 Core Modules (index.js)

- **Location Services:** GPS acquisition, reverse geocoding via OpenStreetMap Nominatim, fallback to Quebec City coordinates
- **Bitcoin Fetcher:** fetch_BTC() - Retrieves prices from Kraken API, supports CAD/USD/EUR currencies
- **Weather Fetcher:** Dual API support (Environment Canada GeoMet / Open-Meteo), temperature unit conversion, weather code mapping
- **Configuration Manager:** Pebble Clay integration for user settings, AppMessage protocol for watch communication

3.2.2 Data Flow

Watch requests data → JS acquires GPS → Reverse geocode location → Fetch Bitcoin from Kraken → Fetch weather from chosen API → Format and send to watch via AppMessage → Watch updates display and stores in persistent storage

3.2.3 Configuration (config.js)

Pebble Clay configuration provides a web-based settings page with toggles, selects, and other controls. Settings include health display, temperature units, seconds ticker, Bitcoin exchange, weather service, and update interval.

4. Data Management

4.1 Persistent Storage

The watchface uses Pebble's persist API to store configuration and historical data between app launches. Storage includes user preferences (trotteuse, celsius, health, exchange, location, service, cadence), Bitcoin price data (current, high, low values plus 60-point graph history), weather data (cached for offline viewing), and location information.

4.2 Bitcoin Graph Algorithm

The push_point() function maintains a 60-point sliding window of Bitcoin prices. New values are added to the right, oldest values are removed from the left. Y-coordinates are normalized between btcL (low) and btcH (high) to fit the 23-pixel graph height. The graph uses a closed GPath structure for efficient rendering.

4.3 Message Protocol

Communication between C and JavaScript uses numbered message keys (0-18) defined in package.json. The protocol includes Bitcoin data (keys 0-2), weather observation (keys 3-9), forecast (keys 10-13), location (key 14), and configuration (keys for trotteuse, celsius, health, exchange, service, location, cadence, clearbtcdta).

Key	Name	Purpose
0-2	btcV, btcL, btcH	Current, low, high Bitcoin price

Key	Name	Purpose
3-9	Weather observation	Icon, temp, wind direction, gust, speed, wind chill, humidex
10-13	Weather forecast	Icon, high, low, period name
14	Location	3-character locality code
18	JS events	Status codes (1=ready, 2=timeout, 3-5=errors)

5. External API Integration

5.1 Kraken Cryptocurrency Exchange

- **Endpoint:** <https://api.kraken.com/0/public/Ticker>
- **Currency Pairs:** XBTCAD, XBTUSD, XBTEUR
- **Data Extracted:** 24h high, 24h low, current close price
- **Timeout:** 3 seconds

5.2 Environment Canada GeoMet API

- **Endpoint:** <https://api.weather.gc.ca/collections/citypageweather-realtime/items>
- **Query Method:** Bounding box around GPS coordinates (± 0.1 degrees)
- **Station Selection:** Closest station to GPS coordinates using distance calculation
- **Data Points:** Temperature, wind speed/direction/gusts, wind chill (if $\leq 0^{\circ}\text{C}$), weather icon code, forecast high/low
- **Timeout:** 10 seconds

5.3 Open-Meteo Weather API

- **Endpoint:** <https://api.open-meteo.com/v1/forecast>
- **Parameters:** GPS latitude/longitude, current weather, hourly forecast (12h), daily forecast
- **Weather Code Mapping:** WMO codes converted to Environment Canada icon codes for display consistency
- **Timeout:** 10 seconds

5.4 OpenStreetMap Nominatim

- **Endpoint:** <https://nominatim.openstreetmap.org/reverse>
- **Purpose:** Reverse geocoding GPS coordinates to city/locality names
- **Fallback Logic:** Locality \rightarrow County \rightarrow State/Province
- **Timeout:** 3 seconds

6. UI Layout and Positioning

6.1 Display Geometry

The watchface is designed for 144×168 pixel displays with three main regions: top (0-95px), middle (95-102px), and bottom (102-168px). The top region contains Bitcoin data and graph. The middle region shows the optional seconds ticker. The bottom region displays time, date, weather, and status indicators.

6.2 Dynamic Positioning

Component positions adjust based on configuration. When the trotteuse is enabled, the time layer moves up 5 pixels. Bitcoin layers shift vertically by 3 pixels when trotteuse is disabled. Graph positioning depends on both trotteuse state and platform (color vs monochrome).

6.3 Platform-Specific Rendering

The application uses conditional compilation (`#ifdef PBL_COLOR`) for platform-specific adjustments. Color platforms use yellow time display and slight horizontal offset for time layer. Monochrome platforms use white on black. Both maintain identical functionality with adjusted aesthetics.

7. Configuration and User Preferences

7.1 Available Settings

- **Health Display:** Enable/disable step count visualization
- **Temperature Units:** Celsius/Fahrenheit and km/h or mph
- **Seconds Ticker:** Show/hide trotteuse bar
- **Bitcoin Exchange:** Kraken CAD/USD/EUR
- **Weather Service:** Environment Canada or Open-Meteo
- **Update Interval:** 1 or 3 minutes
- **Clear Bitcoin Graph:** Reset historical price data

7.2 Configuration Workflow

Settings are managed through Pebble Clay web interface. User modifies settings in phone app → Clay sends AppMessage to watch → Watch handler updates global variables → Watch persists settings → Watch requests new data fetch → UI updates to reflect changes.

8. Error Handling and Resilience

8.1 Network Timeouts

All API requests implement timeout protection (3-10 seconds). On timeout, the watch displays cached data and appends an asterisk to the date to indicate staleness. Users receive status codes via message key 18.

8.2 GPS Fallback

When GPS acquisition fails, the system defaults to Quebec City coordinates (47.0, -71.2). This ensures weather data is always available even without location services.

8.3 Data Validation

JavaScript layer validates API responses before transmission. The C layer sanitizes float values before graph rendering to prevent out-of-bounds coordinates. Null checks protect against missing weather station data.

8.4 Vibration Alerts

Bluetooth disconnection triggers vibration alerts with night mode suppression (10pm-5am). Battery alerts use distinct vibration patterns (long pulses for critical, short pulses for low).

9. Performance Considerations

9.1 Battery Optimization

- Configurable update intervals (1 or 3 minutes) reduce API calls
- GPS timeout prevents excessive location searching
- Seconds ticker is optional to minimize second-unit tick subscriptions
- Cached weather data reduces network activity

9.2 Memory Management

- Static string buffers prevent dynamic allocation
- GPath reuse for graph rendering
- Conditional layer creation (trotteuse only when enabled)
- Bitmap resource management with proper cleanup

9.3 Rendering Efficiency

- Layer dirty marking only when data changes
- Transparent time layer to prevent background overwriting
- Optimized graph path with minimal points

10. Build and Deployment

10.1 Build System

The project uses Pebble SDK 3 with waf build system. Primary commands include 'pebble build' (compile for all platforms), 'pebble clean' (remove build artifacts), 'pebble install --emulator <platform>' (install to emulator), and 'pebble screenshot' (capture emulator display).

10.2 Dependencies

- Pebble SDK 3
- pebble-clay ^1.0.4 (configuration UI)

- Node.js (for PebbleKit JS)

10.3 Testing

Testing occurs on multiple emulators (aplite, basalt, chalk, diorite, emery) to verify cross-platform compatibility. Screenshot validation ensures visual correctness after changes. Log monitoring tracks API response times and error conditions.

11. Recent Enhancements

- Fixed time display overwriting issue with transparent layer implementation
- Added clear Bitcoin graph data feature
- Updated BTC graph border to 23-pixel tall vertical lines
- Implemented configurable update cadence (1 or 3 minutes)
- Enhanced Environment Canada API integration with closest station selection
- Added Open-Meteo as alternative weather service

12. Code Organization

The codebase uses folding markers ({{{ and }}}) for logical code organization. Major sections include Includes, Defines (Keys, Geometries, Colors), Globals (Layers, Variables), Utilities, Handlers (Time, Message, Configuration), and Graphics update procedures.

13. Future Considerations

- Support for additional cryptocurrency exchanges
- Extended historical data visualization (beyond 60 points)
- Customizable color themes for color platforms
- Multiple timezone support
- Enhanced health integration (heart rate, sleep data)
- Weather alerts and notifications

*Document Generated: 2025-10-19
Version: 5.1.0*