社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＩＴＳ－14　(11)
2003／9／9

ナビゲーションとの連携動作を可能とする車載端末向けテレマティクスプラットフォームの開発

田中　克明†　福田　善文†　上脇　正†
中野　正樹‡　里山　元章‡　中村　浩三‡‡

†株式会社　日立製作所　日立研究所　〒319-1292 茨城県日立市大みか町 7-1-1
‡株式会社　日立製作所　システム開発研究所　〒215-0013　神奈川県川崎市麻生区王禅寺 1099
‡‡株式会社　エイチ・シー・エックス　〒335-8511 埼玉県戸田市上戸田 50
E-mail:　†{tanakaka, fuk, kamiwaki}@gm.hrl.hitachi.co.jp,　‡{nakano, satoyama}@sdl.hitachi.co.jp,
‡‡k-nakamura@hcx.co.jp

あらまし　本論文では，ナビゲーションとの連携動作が可能な車載端末向けテレマティクスプラットフォームについて述べる．まず，テレマティクス車載端末に対する要求を整理する．次に，この整理の結果にもとづいて，動的機能拡張，複数アプリケーションによる端末資源の共有，ナビゲーションとの連携動作を可能とするプラットフォームの構成を考案した．さらに評価ボード上に実装することでその機能を確認した．また，評価実験を通して，本プラットフォームがアプリケーションの応答性において十分な性能を持つことを確認した．
キーワード　車載端末，テレマティクス，プラットフォーム，ナビゲーション，Java™，OSGi™

Telematics Terminal Platform which Enables Cooperation between Telematics Applications and Navigation Functions

Katsuaki TANAKA†　Yoshibumi FUKUDA†　Tadashi KAMIWAKI†
Masaki NAKANO‡　Motoaki SATOYAMA‡and　Kozo NAKAMURA‡‡

†Hitachi Research Laboratory, Hitachi,Ltd.　7-1-1 Omika-cho,　Hitachi-shi, Ibaraki, 319-1292 Japan
‡Systems Development Laboratory, Hitachi,Ltd.　1099 Ozenji, Asao-ku, Kawasaki-shi, Kanagawa, 215-0013 Japan
‡‡HCX Corp.　50 Kamitoda, Toda-shi, Saitama, 335-8511 Japan
E-mail:　†{tanakaka, fuk, kamiwaki}@gm.hrl.hitachi.co.jp,　‡{nakano, satoyama}@sdl.hitachi.co.jp
‡‡k-nakamura@hcx.co.jp

Abstract　This paper describes architecture and behavior of a telematics terminal platform we have developed. First, we analyzed requirements for the platform. Next, we created the architecture of the platform that enables dynamic extension of functions, sharing of terminal resources among multiple applications and operations coordinated with navigation functions based on the requirements. Furthermore, we implemented the platform on an evaluation board and it has been proven that this platform had sufficient performance in the response of applications.
Keyword　in-vehicle terminal, Telematics, platform, Navigation System, Java™, OSGi™.

1. はじめに

近年，ドライバー向けに自車周辺の POI(Point of Interest)情報や交通情報，エンターテインメント情報などを提供するテレマティクスサービスが注目されており，今後は従来のナビゲーション機能に加えてテレマティクスサービスに対応した車載端末の拡大が予測されている[1]．このような中，車載端末のソフトウェア実行環境に関する標準化の検討も始まっている[2]．

利用者とテレマティクスサービス提供者のそれぞれの立場から，テレマティクス車載端末に対する要求を下に述べる．

（１）利用者の要求

・新規サービスへの対応

テレマティクスの分野では，様々なサービス提供者

が早いサイクルで新規サービスを立ち上げるが，この新規サービスを受ける為には車載端末もこれに対応していく必要がある．しかし，現在の車載端末は自動車と一体化して販売される形態が多く，また車載端末のプログラムは固定的であるため，これを更新するには新たに CD-ROM 等を購入する必要があった．その為，多くのユーザは車を買い換えるまで，新たに提供されるテレマティクスサービスを利用できない．このような自動車と車載端末のライフサイクルの違いを解消し，購入後も機能拡張を行い，常に最新のサービスを受けたいという要求がある．

・複数サービスの連携

車載端末においては，複数のサービスを同時に提供することが要求される．例えば，POI 情報を閲覧中であっても，ナビゲーションは背後で動作しており，適切なタイミングで誘導案内をして欲しい．

（2）テレマティクスサービス提供者の要求

・ナビゲーションとの連携

車載端末においてはナビゲーションが主要なアプリケーションであり様々な情報を保持している．そこでテレマティクスアプリケーションの操作性やサービス品質の向上の為に，ナビゲーションの持つ情報や地図を活用したいという要求がある．

・オープンなプログラム実行環境

独自のプログラム実行環境を採用した場合，サービス提供の為のアプリケーション開発コストがかかり，サービス普及の障害となる．そのため，すでに普及の進んだオープンなプログラム実行環境が要求される．

以上を整理すると，テレマティクス車載端末に対する要求は，次の 4 つにまとめられる．

(a)オープンなアプリケーション実行環境

(b)新規サービスへの対応

(c)複数サービスの連携

(d)ナビゲーションとの連携

このような要求に対し，最近の車載端末では，要求(b)に対応しているものもあるが[3][4]，要求(c)(d)については考慮されていない．

そこで，報告者らは，これらの要求に対して Java™[5] 技術をベースとした車載端末プラットフォーム（Java™ –based Telematics Platform（以下 JTP と呼ぶ））を考案した．

JTP においては，要求(a)(b)に対して，オープン仕様のプログラム実行環境である Java™ と，Java™ プログラムのライフサイクルを管理する OSGi™[6]（Open Services Gateway Initiative）サービスプラットフォームを採用した．要求(c)に対しては，複数サービスの連携を可能とする Telematics アプリ実行環境を開発した．要求(d)に対してはナビゲーションの機能をアプリケーションに公開する Navi API(Application Program Interface)を定義することで，後から追加したアプリケーションでもナビゲーションとの連携を可能にした．

以下，本プラットフォームの構成及び動作について述べる．

2. 車載端末システムの概要

本研究において，評価ボード上で動作する車載端末システムのプロトタイプを作成した．このシステムを実現するに当たり，Java™ 及び OSGi™ の技術を応用したソフトウェア処理基盤が用いられている．これらの技術及びシステムの全体構成について以下に述べる．

2.1. Java™

Java™ は Sun Microsystems 社の提供するプログラミング環境であり，次のような特徴を持つ．

(1) Java™ VM(Virtual Machine)上でのプログラム実行

Java™ アプリケーションは計算機上に設けられた仮想的な実行環境である VM の上で動作する．どのような CPU や OS を持つ計算機においても，VM が設けられていればプログラムは動作するため，移植性が高まる．他のプラットフォームで開発されたアプリケーションを容易に持ち込むことが可能となる．

(2) セキュリティモデル

Java™ はサンドボックスと呼ばれる保護機構を持つ．Java™ アプリケーションは特定の端末資源以外を使用することがないため，アクセス違反によるセキュリティの侵害がない．また，アプリケーション間においても保護がなされるため，他のアプリケーションの処理領域を不正に侵害することがない．

(3) オープン仕様

Java™ において共通に使用する API が公開されているため，多くのサービス開発事業者が参入しやすくなっている．これにより，品質の高いサービスが多数提供されることが期待できる．

2.2. OSGi™ サービスプラットフォーム

OSGi™ は，家電，車載機器等を結ぶことによって構成されるネットワークをインターネットに接続するゲートウェイの標準化を進める団体である．OSGi™ サービスプラットフォームは，このゲートウェイのミドルソフトウェア仕様として，OSGi™ により定められた．

OSGi™ サービスプラットフォームは次のような特徴を持つ．

(1) アプリケーションの遠隔インストール

インターネット上に存在する Java™ アプリケーションを通信回線経由でダウンロードし，サービスプラ

ットフォーム上にインストールすることが可能である．同様に，サービスプラットフォーム上のアプリケーションを更新や削除する手法も用意されている．これにより，新たなサービスを提供するアプリケーションを，インターネットを通じて端末に追加することが可能となる．

(2) 標準化団体による採用

OSGi™ サービスプラットフォームは，車載機器の統合仕様を策定する標準化団体 AMI-C [7]により採用されている．また，インターネットを基礎とした ITS (Intelligent Transport Systems) 共通基盤の構築を行う団体であるインターネット ITS 協議会[8]も採用を検討している．これらにより，OSGi™ サービスプラットフォームをもとにしたアプリケーションのインストール手法の広範な普及が期待される．

2.3. JTP の全体構成

JTP では，動的機能拡張や複数サービスの連携を主目標としている．動的機能拡張では，アプリケーションをサーバよりダウンロードし実行する機能が必要となる．複数サービスの連携では，複数のアプリケーションが自分の持つ機能を API として公開する機能が必要となる．これら及び前節で説明したような特徴から，JTP では Java™ 及び OSGi™ の技術を基盤として用いた構成を検討した．JTP の全体構成を図 1に示す．

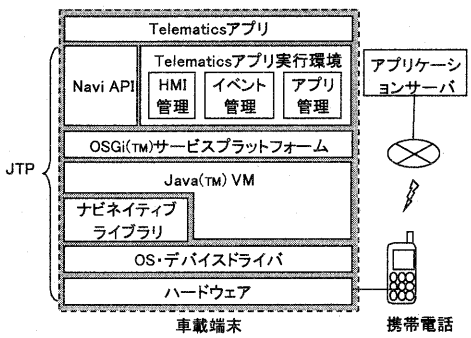

図 1　JTP の全体構成

図中の主なブロックについて以下に説明する．

2.3.1. ハードウェア

200MHz 動作の CPU 及びナビゲーション関連の機能を持つ ASIC を搭載する評価用ボードを用いる．記憶容量は RAM が 64MB，フラッシュメモリが 32MB である．このボードは USB インターフェイスを備えており，ここに携帯電話を接続して，ネットワーク接続されたアプリサーバとの通信を行う．

2.3.2. OS，ドライバ

OS にはリアルタイム OS を用い，USB ドライバをはじめとするドライバ類を用意する．

2.3.3. ナビネイティブライブラリ

地図表示や経路誘導などナビゲーションの機能を提供するライブラリで，Java™ VM 及び Navi API を通じてアプリケーションにより利用される．

2.3.4. Java™ VM

Java™ アプリケーションを動作させる環境を提供する．車載用途に合わせて，端末性能の限られたハードウェア上での動作を考慮し，次のような仕様のものを用いる:J2ME[9] (Java™2, Micro Edition)，CDC[10] (Connected Device Configuration)及び PBP[11](Personal Basis Profile)．

2.3.5. OSGi™ サービスプラットフォーム

ネットワーク経由によるアプリケーションのダウンロード，実行などを実現するためのサービスプラットフォームを提供する．

2.3.6. Navi API

Java™ で記述されたアプリケーションの発行した要求を解釈し，JNI (Java™ Native Interface) の機能を用いてナビネイティブライブラリへ要求の実行を指示する．詳細については4章にて説明する．

2.3.7. Telematics アプリ実行環境

OSGi™ サービスプラットフォームや Java™ VM を利用し，複数の車載端末用アプリケーションを実行させる環境を提供する．詳細については3章にて説明する．

2.4. Telematics アプリの実行

ここで，上述の JTP 上でアプリケーションがどのように実行されるかを説明する．なお，JTP 上で実行する車載端末用アプリケーションを，以下 Telematics アプリと呼ぶ．

Telematics アプリは，報告者らが定義した TApplication という Java™ インターフェイスを実装することにより，JTP 上で動作可能になる．Telematics アプリに関する情報は，このインターフェイスを通じて入出力される．JTP 上で実行する Telematics アプリの画面表示例を図 2に示す．

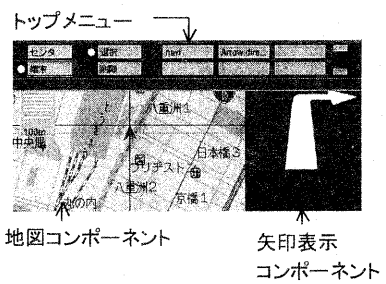

図 2　JTP 画面表示例

この例では，Telematics アプリのインストール状態

等を操作するトップメニューの他，地図コンポーネント及び矢印表示コンポーネントが表示されている．コンポーネントとは，アプリケーションの処理結果を表示するための部品である．コンポーネントは主に矩形の領域として画面上に存在し，アプリケーションは矩形の内部のみに描画することができる．画面上では複数のコンポーネントが存在可能であり，コンポーネントを並べたり重ね合わせることが可能である．なお，これらのコンポーネントを描画する画面をフレームと呼ぶ．

2.4.1. インストール

アプリサーバにあらかじめ配置しておいたTelematics アプリの位置（URL，Universal Resource Locator）を，トップメニューからの操作等により指定する．次に，Telematics アプリ実行環境のアプリ管理(3.1参照)が，OSGi™ サービスプラットフォームを利用して通信回線経由で車載端末上にダウンロードする．OSGi™ サービスプラットフォームはダウンロードしたアプリケーションを Java™ VM 上にインストールする．

2.4.2. 起動

Telematics アプリ実行環境のアプリ管理が，インストールした Telematics アプリの起動を行う．Telematics アプリは OSGi™ サービスプラットフォーム上で処理を開始する．

2.4.3. 表示部品登録

起動した Telematics アプリが画面表示を行う場合，生成したコンポーネントを車載端末の画面に相当するフレームに登録する．登録されたコンポーネントは，Telemtics アプリ実行環境の HMI 管理(3.3 参照)により，フレーム上のどの部分に表示すべきかが定められる．

2.4.4. イベントリスナ登録

車載端末利用者によるリモコン操作や，GPS の位置変化といった事象は，イベントとして JTP 上に反映される．これらの事象を Telematics アプリが利用したい場合は，イベントが発生した際に処理を行うイベントリスナを JTP に対して登録する．リモコン操作によるキー入力イベントは，Telematics アプリ実行環境のイベント管理(3.2 参照)によって制御される．

2.4.5. アプリケーション処理

Telematics アプリは，JTP に用意されているライブラリや，上記のイベントリスナ等を利用することによって処理を行う．

2.4.6. 結果表示

Telematics アプリは，処理を行った結果をコンポーネントへの描画などにより利用者へ提示する．

上記のような手順により，JTP 上において Telematics アプリは実行される．

3. Telematics アプリ実行環境

Telematics アプリ実行環境は，JTP 上で実行される複数の Telematics アプリの動作を検出し，使用される端末資源の管理を行う．これにより，Telematics アプリ間の連携を可能とする．

Telematics アプリ実行環境自身は，OSGi™ サービスプラットフォーム上で動作するアプリケーションである．また，管理する対象に応じて次の三つの部分に分けられている．

(1)アプリ管理
(2)イベント管理
(3) HMI 管理

上記のそれぞれについて以下に説明する．

3.1. アプリ管理

利用者に対して新規なサービスを提供するには，サービスを実行するアプリケーションを車載端末上に持ち込む必要がある．これを実現する方法の一つには，アプリケーションを記憶メディアに記録しておいて車載端末に挿入する方法が考えられる．しかしこの場合，メディアを持ち運ぶ煩雑さがあるとともに，メディアに記録されているのが常に最新のアプリケーションとは限らないという問題がある．もう一つの実現方法として，ネットワークを経由したアプリケーションのダウンロードが考えられる．この場合，コンテンツ開発者によって作成されたアプリケーションを配置するアプリサーバに接続することで最新のアプリケーションを取得可能である．また，移動することが前提となっている車両では，有線よりも無線の通信回線を経由する方が，移動を制約されることがなく効率的となる．JTP では，携帯電話を利用し，無線通信回線経由でアプリサーバより Telematics アプリをダウンロードする．

Telematics アプリをダウンロードする際，どこにどのようなものが存在するかを把握しておく必要がある．これらを利用者に提示することによって，どの Telematics アプリを使用するのか指定させるためである．これらの情報には，Telematics アプリの名称やダウンロード先 URL が含まれる．

ダウンロードした Telematics アプリは，車載端末上にインストールされ，ユーザの要求に応じて実行される．Telematics アプリの実行状況は，利用者の要求や，端末，車両の状態によって変化する可能性がある．例えば動画再生アプリケーションは，運転者の注意力を散漫にさせることがないよう，車両の走行中は実行を停止すべきである．

上記のような要求に対応するため，アプリ管理は図3に示す構成をとった．アプリ管理は，OSGi™ サービ

スプラットフォームに対する Telematics アプリのインストール，起動，停止，削除といった処理の状態を管理する．これらの状態が変化した場合，JTP の各部に対してイベントを発行し，対応する処理を実行させる．

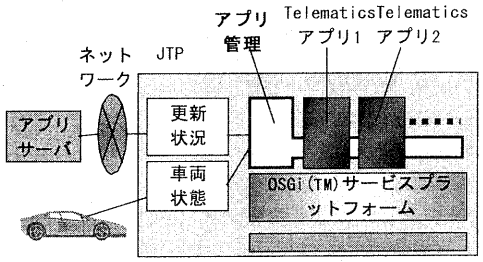

図 3　アプリ管理の構成

3.2. イベント管理

リモコンのような入力装置は通常 1 台の車載端末に対して 1 台備わっており，利用者はこれを使用することによりアプリケーションを操作できる．リモコンは上下左右といった方向キーや，広域表示，詳細表示などの機能キーを持つ．キー入力を行うことによってキーイベントが発行され，アプリケーションはこれを解釈して処理を行う．車載端末上で単一のアプリケーションが起動している場合は，キーはそのアプリケーションの内容に対応した動作をさせることができる．しかし複数のアプリケーションが同時に起動している場合，すべてのアプリケーションに対してキーイベントを発行すると利用者の予期しない動作を行ってしまう可能性がある．上記のような問題を解決するため，イベント管理は図 4に示す構成をとった．イベント管理では，キー入力等のイベントを，どれか一つの Telematics アプリが利用できるようイベントの出力先を管理する．この出力先を入力フォーカスと呼ぶ．入力フォーカスをいずれかの Telematics アプリに割り当てることにより，他の Telematics アプリへのイベント発行を抑制する．

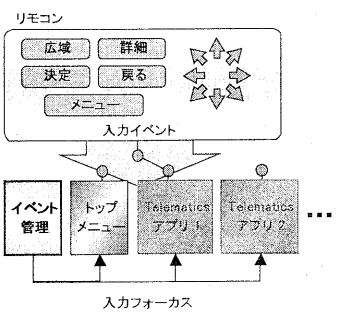

図 4　イベント管理の構成

3.3. HMI 管理

JTP では，利用者に新規のサービスを提供するに当たって，アプリサーバより Telematics アプリをダウンロードして起動する．この際，JTP 上で既に起動していたものと合わせて複数の Telematics アプリが動作することになる．それぞれの Telematics アプリは描画領域をフレーム上に確保し，内部での処理結果を，描画領域を通じて利用者に提示する．この際，Telematics アプリは種類によって多岐にわたる情報を利用者に提示する．これらの情報はそれぞれ，利用者にとってどれだけ重要であるかが異なってくる．利用者にとってより重要な情報は，優先的に表示する必要がある．また，この重要性は，車両や利用者の状態によって変化する可能性がある．

さらに，各 Telematics アプリの持つ描画領域は画面上の任意の場所に置くことができるため，互いに重なり合う場合が考えられる．重なっている場所ではどちらかの描画領域が表示され，他方は隠されてしまうことがある．この状態では，描画領域の重なった Telematics アプリ間で連携を行っても，結果を同時に表示することは困難である．これらの要求に対応するため，HMI 管理は図 5に示す構成をとった．HMI 管理では，それぞれの Telematics アプリがどの程度優先して表示されるべきかを表す表示優先度管理テーブルを持ち，その優先度にもとづいてフレーム上に描画領域の割り当てを行う．

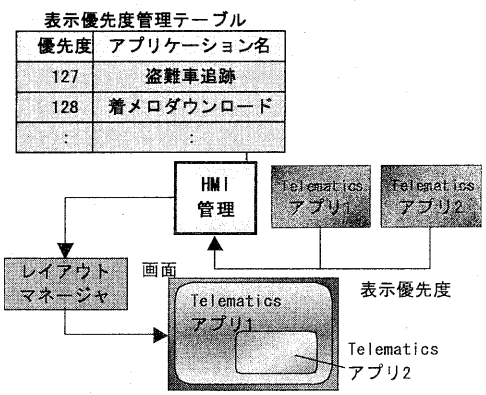

図 5　HMI 管理の構成

4. Navi API

Navi API は，Telematics アプリに対してナビゲーションの持つ機能を提供する．Navi API は Telematics アプリの発行した要求を解釈し，JNI を用いてナビネイティブライブラリへの指示を行う．これにより，車載端末購入後に追加した Telematics アプリであっても，

ナビゲーションの持つ様々な情報を活用することができ，Telematics アプリの操作性やサービス品質の向上につながる．Navi API の提供する主な機能を表 1に示す．

表 1　Navi API の提供する主な機能

分類	機能
地図表示	地図表示サイズ・位置変更
	指定緯度経度を中心した地図表示
経路操作	目的地・出発地・経由地の取得
	誘導ポイントまでの残距離取得.
	目的地までの残所要時間・残距離取得
自車	自車位置・速度・進行方向取得

5. Telematics アプリの実装

JTP の機能と性能を検証するために，図 6に示すようなキャラクタとの音声対話で操作し，交通情報や POI 情報などを提供する Telematics アプリを構築した[12]．

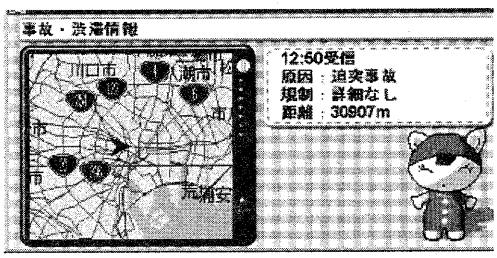

図 6 キャラクタアプリ HMI

この Telematics アプリでは，Navi API の提供する地図を画面左側にレイアウトし利用している．また Navi API で取得した自車位置や走行中の経路に基づいて交通情報や POI 情報を提供する Telematics アプリとナビゲーション機能の連携が実現できている．

また，Telematics アプリ実行環境の機能を使って追加・更新することができる．さらに，JTP 上で同時に２つの Telematics アプリを動作させ，イベント管理の入力フォーカスを切り替えることにより，操作対象を動的に切り替えることが可能である．

以上により，JTP の持つ，動的機能拡張，複数サービスの連携，ナビゲーションとの連携といった機能が確認できた．

6. 評価

6.1. プログラムサイズ

上述の Telematics アプリ実行環境と Navi API が JTP に与える影響を調べるために，記憶装置におけるプログラムの占有量を評価した．JTP 上では複数の Telematics アプリが動作するため，プログラムの占有量がプラットフォーム全体に影響を与える可能性があると考えられる．

プログラムの占有量として，補助記憶領域を占有するファイルサイズとメインメモリを占有するメモリサイズとを測定した．測定する箇所は，次の４つとする．

(1) アプリ管理，イベント管理，及び HMI 管理
　Telematics アプリ実行環境の中心となる部分．
(2) JTP ライブラリ
　Telematics アプリとのインターフェイスなどを構成する部分．
(3) トップメニュー
　Telematics アプリの操作をはじめとするメニュー機能．
(4)Navi API

測定結果を表 2 に示す．

表 2 プログラム占有量

	ファイルサイズ[kB]	メモリサイズ[kB]
アプリ管理,イベント管理,HMI管理	17.1	44.6
JTPライブラリ	16.8	126
トップメニュー	22.1	128.2
Navi API	90.2	231.9
合計	146.2	530.7

ファイルサイズは合計で 146.2KB であり，補助記憶として使用することが想定されるフラッシュメモリの容量 32MB に比べて約 0.44%となっている．占有量は 1%よりも小さくなっており，プラットフォームに与える影響は少ない．同様に，メモリサイズは合計 530.7KB であり，メインメモリとして使用する RAM の容量 64MB に比べて約 0.81%となっている．占有量は 1%よりも小さくなっており，プラットフォームに与える影響は少ない．

6.2. 起動時間の測定

OS が起動後，Java™VM が起動してから JTP が起動完了するまでの時間を測定した．測定結果を表 3に示す．

VM の起動開始から JTP の起動完了までには約 23 秒かかっており，起動時間の大幅な短縮が課題である．起動時間の大半は，Telematics アプリ実行環境と Navi API の初期化であり，ここでは，グラフィクスの初期化や地図の読込，経路誘導用データの初期化等を行っている．これらの中には，起動時に行う必要のない処理も含まれているため，そのような処理を起動後に行うように移動することにより，起動時間の短縮が可能

である.

表 3 JTP 起動時間測定結果

	[msec]
VM起動	240
OSGi™サービスプラットフォーム起動	950
プログラム情報の復帰	1140
Telematicsアプリ実行環境, Navi APIの初期化	20250
OSGi™サービスプラットフォームによるプログラム起動	730
合計	23310

6.3. イベント応答時間の測定

JTP 上で動作する Telematics アプリのキーイベント応答性を測定する. ここでは, VM がリモコンドライバからイベントを受信し, Telematics アプリから登録されたリスナーメソッドを呼び出すまでの VM 内の応答時間と, VM から呼ばれるリスナーメソッドの処理に要する Telematics アプリの応答時間とに分けて測定し, この和をキーイベント応答時間とした.

6.3.1. VM 内のイベント応答時間

表 4に, Telematics アプリが一万回のイベント通知を受けるまでにかかる時間を計測した結果を示す. 計測した時間は, VM がイベントを受け取り, 登録されたリスナーメソッドを呼び出すまでの応答時間を示しており, イベント発生から VM へ通知するまでの時間は含まれていない.

表 4 VM内のイベント応答時間

	10000回[msec]	1回[msec]
VM内のキーイベント応答時	5380	0.54

6.3.2. Telematics アプリの応答時間

JTP 上の Telematics アプリの応答時間として, 下にあげる二例のリスナーメソッドの所要時間を測定した.

①図 7(a)に示す地図のみが表示されている状態で, キーイベントを受信した時に図 7 (b)のプルダウンメニューが表示する.

②図 7 (a)に示すような地図のみが表示されている状態で, キーイベントを受信した時に図 7 (c)のオンマップメニューが表示する.

表 5に測定結果を示す. 6.3.1の結果とあわせると, キーイベントを受信してから, Telematics アプリが応答処理を完了するまでの時間は 23msec となり, 操作者はその応答時間の遅れが気にならない値である.

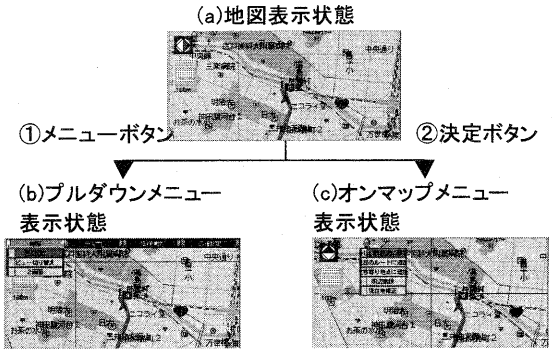

図 7 Telematics アプリ応答性測定例

表 5 Telematics アプリ応答時間

Telematicsアプリ応答時間	1000回[msec]	1回 [msec]
①	22940	23
②	22120	22

7. まとめ

本論文では, 動的機能拡張, 複数アプリケーションによる端末資源の共有, ナビゲーションとの連携動作を可能とする車載端末向けテレマティクスプラットフォームの構成を検討した. さらに評価ボード上に実装することで機能の確認をした. また, プログラムサイズ, 起動時間, イベント応答時間を測定し, プログラムサイズおよびイベント応答時間について十分な性能を持つことを確認した.

文 献

[1] “2002年度版世界In-Car-Computing市場の徹底研究”, (株) SRD ジャパン, 東京, 2002

[2] 和泉 順子他, “インターネットITSプロジェクトの概要”, 情報処理, Vol.43, No.4, pp369-375, Apr.2002.

[3] 三菱電機,“Java テクノロジー対応カーナビゲーションシステム”, 三菱電機技報, Vol.77,No.1,p.24,2003

[4] “G-BOOK” , https://g-book.com/pc/

[5] “The Source for Java™ Technorogy”, http://java.sun.com/

[6] “OSGi: Open Services Gateway Initiative”, http://www.osgi.org/

[7] “AMI-C”, http://www.ami-c.org/

[8] “Internet ITS” ,http://www.internetits.org/

[9] “Java™ 2 Platform, Micro Edition”,http://java.sun.com/j2me/

[10] “CONNECTED DEVICE CONFIGURATION (CDC) and the CVM Virtual Machine”, http://java.sun.com/products/cdc/

[11] “Personal Basis Profile”,
http://java.sun.com/products/personalbasis/
[12] えのきどいちろう, “次世代テレマティクスをめぐって” ひたち 2003.No.2 3月号, pp.12-15, Mar.2003.
http://www.hitachi.co.jp/Sp/hitachi/2003_03.html
