協調型自動運転のための組込み RTOS による 時空間グリッド予約システムの実装と評価

Implementation and Evaluation of a Spatio-temporal Grid Reservation System Using Embedded RTOS for Cooperative Automated Driving

岩井 駿人 [1]　梅田 寛斗 [1]　松本 翔汰 [1]　辰己 弘征 [2]　佐藤 健哉 [3]
Hayato Iwai　Hiroto Umeda　Shota Matsumoto　Hiroyuki Tatsumi　Kenya Sato

1. はじめに

人工知能やセンサ技術の進化により，自動運転技術は，交通事故の削減や渋滞緩和など，多くの社会課題を解決する手段として注目を集めている [1][2][3]．このような背景のもと，これまでの自動運転の研究では，車両に搭載されたセンサやカメラ，LiDAR，レーダ，GPS などを用いて周囲の環境を認識し，経路計画から運転操作までをすべて車両自身が行う「自律型自動運転（Autonomous Driving）」が中心的な研究対象となってきた [4]．自律型自動運転の利点は，外部との通信やインフラへの依存を最小限に抑え，車両自身が判断と制御を行うことで走行できる点である．また，実証実験や社会実装も目指されている [5]．しかしながら，自律型自動運転には技術的な課題が存在する．例えば，車両に搭載されているセンサによる認識範囲は限られており，見通しの悪い交差点や遮蔽物の裏側など，車両自身では認識できない情報に基づく判断が困難であり，誤認識による事故が発生する可能性がある．

このような背景により，近年では「協調型自動運転（Cooperative Automated Driving）」が新たなアプローチとして注目されている [6][7]．協調型自動運転とは，自車両のセンサ情報に加えて，他車両や道路インフラとリアルタイムに情報を共有し，連携して走行を行う方式である [8]．また，これらの情報を統合することにより，複数の車両や周辺の情報を考慮した，全車両の走行経路の最適化を行う走行調停ができるため，交差点の通過や合流などの複雑な場面でも，安全かつスムーズな走行が可能となる．

協調型自動運転を実現するための技術基盤としては，「V2X（Vehicle-to-Everything）通信」がある [9][10]．V2X 通信は，図 1 に示すように車両間で直接通信を行う V2V（Vehicle-to-Vehicle）通信，信号機などの道路設備と通信を行う V2I（Vehicle-to-Infrastructure）通信，ネットワークを介した V2N（Vehicle-to-Network）

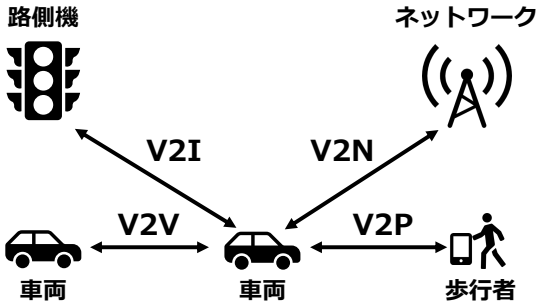

図 1: V2X 通信概要図

通信，および歩行者と車両との間で通信を行う V2P（Vehicle-to-Pedestrian）通信から構成されており，それぞれの通信手段を通して多様な情報を相互にやり取りすることができる．これにより，車両は自らの認識限界を補い，より広範囲かつ高精度な環境把握と意思決定を行うことが可能となる．

一方で，協調型自動運転の実現には新たな技術的課題も存在する．複数の車両が同時に通信を行い，意思決定と走行制御を協調させる必要があるため，通信量や計算量の増大の問題が発生する [11]．これらの課題を克服するには，通信・処理の効率性を保ちつつ，通信資源や計算資源を明示的に調整できるような仕組みが求められる．

本研究では，協調型自動運転における通信および処理負荷を軽減するために，プリエンプティブな優先度ベーススケジューリング方式の組込み RTOS (Real-Time Operating System) を利用して，複数の自動運転車両がリアルタイムに走行調停を行うシステムを構築する．RTOS とは，決められた時間内にタスクを確実に完了させることを保証するリアルタイム性を備えた OS であり，タスクの優先度に応じて即時に処理を切り替えることが可能である．本研究では，協調型自動運転において走行制御や通信など優先度の異なる処理を安全かつ確実に実行する必要があることから，プリエンプティブな優先度ベーススケジューリング方式を採用している．

RTOS 上に時空間グリッド予約システム [12][13][14]

[1] 同志社大学大学院理工学研究科情報工学専攻
[2] 名古屋大学大学院情報学研究科知能システム学専攻
[3] 同志社大学モビリティ研究センター

を実装し，各車両の走行制御を優先度に応じて動的に行うことで，即応性と安定性の両立を図る．また，計算量を最小限に抑えつつ，衝突回避や走行調停が確実に行えるよう設計する．

本論文の構成は以下のとおりである．第 2 章では，関連する技術として，µT-Kernel 3.0[15] および micro:bit[16][17][18] の概要と，時空間グリッド予約について述べる．第 3 章では，提案する制御方式の実装に関して，プロトコルの設計について説明する．第 4 章では，実験の内容と評価について説明し，第 5 章では，評価に対する考察と今後の課題を述べる．最後に第 6 章でまとめを行う．

2. 関連技術

2.1 µT-Kernel 3.0

µT-Kernel 3.0 は，IoT エッジノードのような小規模な組込みシステムに向けて設計された RTOS であり，軽量性と高いリアルタイム性を両立している．特徴の一つは，IEEE 2050-2018 [19] に準拠していることであり，これにより国際的な互換性と標準化を実現している．また，µT-Kernel 3.0 は，多様なプロセッサへの移植が可能で，特定の開発環境やツールに依存しない高い移植性を有している．その構成はモジュール化されており，必要な機能のみを選択して組み込むことができるため，リソースが限られた環境においても効率的な運用が可能である．構成要素としては，タスク管理や同期・通信，割込み管理などの基本機能を提供する µT-Kernel/OS，システムメモリ管理やデバイス管理，省電力制御などの拡張機能を担う µT-Kernel/SM（System Manager），およびデバッガとの連携を支援する µT-Kernel/DS（Debugger Support）の 3 つが中心となっている．さらに，µT-Kernel 3.0 にはサービスプロファイルと呼ばれる仕組みが導入されており，これにより実装の機能や特性が明確化されることで，ソフトウェアの互換性と移植性が一層向上している．

今回実装する時空間グリッド予約システムでは，サーバとの無線通信，ライントレース走行，複数車両の走行調停など全ての機能を µT-Kernel 3.0 上で実装する．

2.2 micro:bit と周辺機器

micro:bit は，イギリスの BBC が主導して開発された教育用のマイコンボードである．小型の基盤上に，25 個の LED ディスプレイ，2 つのボタン，各種センサ類が搭載されており，これらの機能を活用することで多様なプログラムが作成可能である．また，無線通信機能を備えており，無線によるデバイス間の通信を実現できる．さらに，基板下部に実装されているエッジコネクタを利用することで，外部の電子部品や拡張モジュールと接続し，外部機器との連携や制御を行うことが可能である．

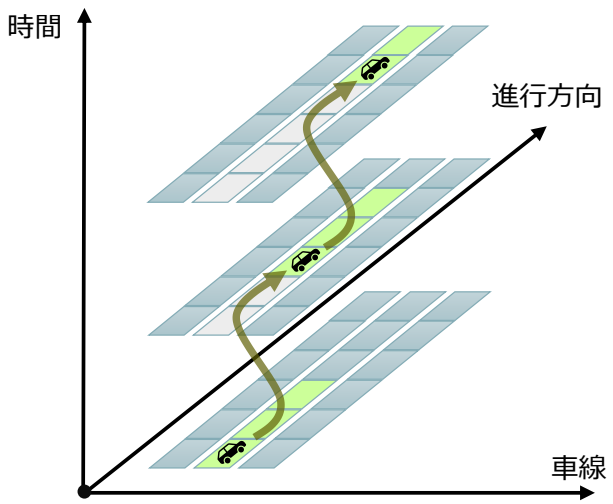

図 2: 時空間グリッド予約概要図

本実験では，micro:bit 上で µT-Kernel 3.0 を動作させ，車両の制御や無線通信の機能を実現する．

また実験用車両として，DFRobot 社が開発した micro:bit 用ロボットプラットフォームである micro:Maqueen Plus を用いる [20]．本機は，左右独立の駆動輪を備えた二輪駆動方式を採用し，前進・後退・旋回などの基本的な移動動作を制御できる．センサ類としては，底面に計 5 個の赤外線グレースケールセンサを搭載し，ライントレースや交差点検出などの複雑な走行制御に対応している．これにより，本実験では自律的な車両制御と道路環境の認識を実現することができる．

2.3 時空間グリッド予約

前述の V2V 通信における走行調停の通信量が増大するという問題点に対し，走行調停における通信量削減を目的として時空間グリッド予約が提案されている．時空間グリッド予約の概要を図 2 に示す．時空間グリッド予約とは，道路上の空間を一定の領域に分割し，時間情報と組み合わせることで，サーバが車両の現在位置および将来存在する位置をグリッドの占有状況により車両の走行を調停する手法である．

時空間グリッド予約の基本的なシステム構成を図 3 に示す．まず，車両は出発時刻，出発地，目的地をサーバに送信する．次に，サーバは出発地から目的地までの経路における時空間グリッドの空き状況をデータベースに問合わせ予約を行う．予約を希望する時空間グリッドが他車両により占有されている際はそのグリッドの直前で停止し，そのグリッドが解放されるまで待つようにグリッドを予約する．次に，予約結果をサーバに送信し，その結果をもとにサーバは車両の経路を組み立てる．最後にその結果を車両に送信する．

V2V 通信にかかる計算量は，車両台数 n に対し $O(n^2)$

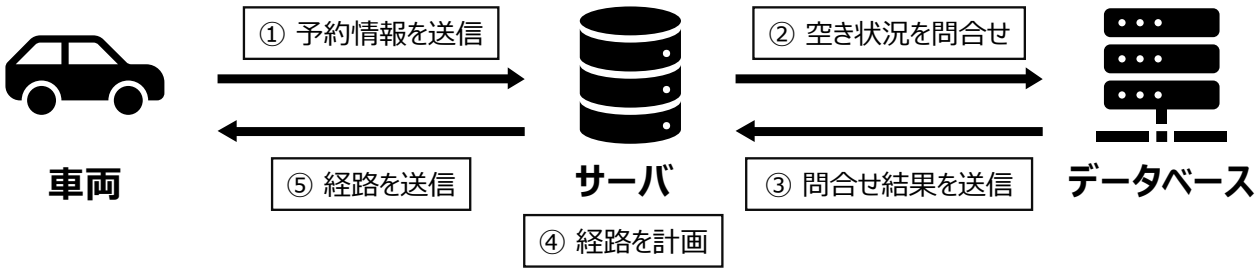

図 3: 時空間グリッド予約のシステム構成

になる一方，この手法では，車両はサーバとの通信しか行わないため通信にかかる計算量が $O(n)$ になり，通信量を大幅に削減することが可能となるといった利点がある．更なる利点として，事前に走行計画を確定可能な点が挙げられる．車両が，調停が必要な地点に接近したときに調停を行うのではなく，調停が必要な地点から遠い位置を走行する車両も調停に参加することができるため，V2V 通信による走行調停と比較し，広域での走行調停が可能となる．早い段階で車両の加減速や空いている車線への車線変更が可能となることにより，急減速や急停止が少ない安全で効率的な走行調停が行える．

3. 実装

3.1 概要

本システムの実装は，大きく 3 つの機能に分類される．車両の移動制御を担う走行機能，車両とサーバ間での情報の送受信を行う通信機能，および経路の割当てと調停を実現する予約機能である．

本システムは，サーバ 1 台と複数の車両クライアントから構成される．まず，車両は通信機能を用いてサーバに情報を送信し，サーバは予約機能により車両の経路を決定する．そして，再度通信機能を用いて経路を車両に送信し，車両は通信機能により経路に従い走行を行う．サーバは通信と予約，車両は走行と通信の各機能を用いて相互に連携することで，車両同士の衝突や予約の競合のない協調的な走行を実現している．

また，これらの 3 つの機能は µT-Kernel 3.0 上においてタスクとして実装されている．各タスクには優先度が設定され，状況に応じて効率的かつ安全に制御されるよう設計されている．具体的には，車両の安全な走行を確保するために，走行機能は高い優先度に設定されており，通信や予約に関する処理はそれよりも低い優先度に設定されている．さらに，走行機能の中でも，ステアリング操作や環境認識などの各タスクに対しても優先度を細かく設定することで，状況に応じた適切な処理の実行が可能となっている．

3.2 走行機能

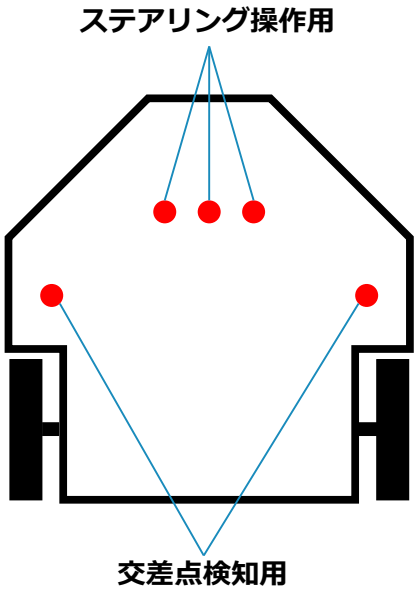

図 4: micro:Maqueen Plus の裏面センサの用途内訳

走行機能は，micro:Maqueen Plus のセンサを用いて，車両が自律的に道路上を走行するための制御を担う．具体的には，車両がセンサから取得した情報をもとに，直進や右左折，停止などの動作を行う．また，サーバから受信した経路指示に常に従い，進行方向や動作を切り替える．

前述のとおり micro:Maqueen Plus には，中央裏側に 3 つ，左右それぞれの裏側に 2 つずつ，計 5 つの赤外線グレースケールセンサが取り付けられている．そのため，それらを使い分けることでライントラッキングのほかに交差点検知など，高度な制御が可能となる．本実装では図 4 に示すように，中央裏側の 3 つのセンサをライン上を進むためのステアリング操作用に，左右それぞれのセンサを交差点検知および転回終了の検知用に割り当て，

自律的な走行を可能にした．

また，走行機能に関わる各種制御は，µT-Kernel 3.0 のタスクとして実装されており，最も高い優先度を持つ走行のためのステアリング操作をはじめとして，センサデータの取得，交差点検出の順に，あらかじめ定められた優先度に従ってタスクが実行される．

3.3 通信機能

通信機能は，車両とサーバ間の情報の送受信を担当し，走行調停や経路予約に必要なデータをリアルタイムでやり取りする．具体的には，車両は新しい目的地をサーバに送信し，サーバは時空間グリッドの占有状況を確認後，最適な経路を計算する．また，サーバから車両への経路指示通知もこの通信機能を通じて行われる．

通信は，micro:bit のマイコンである nRF52833 の無線通信モジュールを使用し，µT-Kernel 3.0 上で実装されたプロトコルに従ってデータの送受信が行われる．本システムでは，通信プロトコルおよび経路指示における時間の最小単位を 1 秒とし，すべての経路指示や予約情報はこの最小単位で管理される．このようにすることで，車両は常に最新の情報を基に走行制御を行うことができ，またサーバも各車両の状態をリアルタイムで把握することが可能となる．

本システムで行う通信フローと通信内容を示した図を図 5 に示す．本システムの通信プロトコルは，ID 取得通信と経路予約通信の 2 種類に限定し，構成を簡潔にした．実環境ではさらに多様な通信が必要となるが，本研究ではモデル化を重視し，将来的な拡張性を考慮した設計とした．まず，ID 取得通信では，車両がサーバに空のパケットを送信し，サーバから車両 ID と侵入可能時間が返される．これにより，新規車両がサーバ管轄エリアに入ることができるようになる．次に，経路予約通信では，車両が旅行開始時刻，旅行開始地点，旅行終了地点をサーバに送信し，サーバからは経路指示のリストが返送される．また，通信が正しく行われず，経路指示のリストが返送されない場合は，再度予約の要求を行う．経路指示は 1 バイトで進行方向（上位 4 ビット）と継続時間（下位 4 ビット）を表現し，複数の指示が配列として送信される．このプロトコルにより，車両はサーバから受け取った経路指示に従って走行を行う．

3.4 予約機能

予約機能は，サーバにおいて時空間グリッドの管理と経路予約を行うものであり，走行調停の中核をなす機能である．車両からの予約要求に基づき，サーバはデータベースに格納された時空間グリッドの情報をもとに，空きグリッドの検索や予約の確定を行う．また，経路探索には A*アルゴリズム [21] を用いており，最短経路を効率的に算出する．予約処理においても，時間の最小単位を 1 秒とし，すべてのグリッド占有情報や経路計画はこの最小単位で管理される．

時空間グリッドの実装では，限られたメモリ資源の中で最大限の情報を扱うため，多次元配列を用い，さらに循環バッファの仕組みを組み合わせて，効率的な管理を実現している．循環バッファとは，配列の先頭と末尾を連結するように管理するデータ構造であり，先頭要素から末尾要素まで使用した後は，再び先頭に戻ってデータを格納する．これにより，配列のサイズを固定したまま，時系列データなどを効率的に管理できる．本システムでは，時間軸方向に循環バッファを用いることで，将来の一定期間分のみを記録対象とし，過去のデータは自動的に破棄される設計としている．これにより，無駄なメモリ消費を抑えつつ，時系列情報を効率よく扱うことができる．

なお，本実装では，データベースにおけるトランザクション処理や排他制御といったリアルタイム性への考慮は行っていない．これは，モデルの簡潔化を重視し，実装構成を単純化することで走行調停アルゴリズムの検証を優先したためである．一方で，構成を単純化したことで，トランザクション競合時の応答時間について将来的には検討が必要となると考えられる．

4. 評価

4.1 概要

本システムの有効性を検証するため，第 3 章で示した機能を実装し，実験を行う．本システムは協調型自動運転における効率的な計算処理と走行調停の確実性を重視して設計しているため，これらの観点から，CPU 使用率と，実験継続時間の評価を行う．

4.2 実験環境

本実験では，図 6 のように格子状の道路環境を使用する．また，道路環境は 5×5 のグリッドに分割し，グリッドを矢印の方向にのみ移動可能であるという制約を設ける．グリッド座標系では，図 6 のように x 軸 y 軸を設定し，$(x, y) = (1, 1), (1, 3), (3, 1), (3, 3)$ の 4 箇所を通行不可領域として設定する．原点の横に車両の待機場所を用意し，各車両は待機場所から管轄範囲内に侵入する．また，1 グリッドは $30cm$ 四方である．

前述の通り，車両制御には，micro:bit を搭載したロボットカーである micro:Maqueen Plus を使用する．図 7 のように，ログを表示するための PC に接続した micro:bit を道路環境近くに固定設置し，サーバとする．

4.3 実験内容

車両はグリッドの $(0, 0)$ と $(4, 4)$ の位置を往復する走行パターンを設定する．各車両は目的地付近に到達する

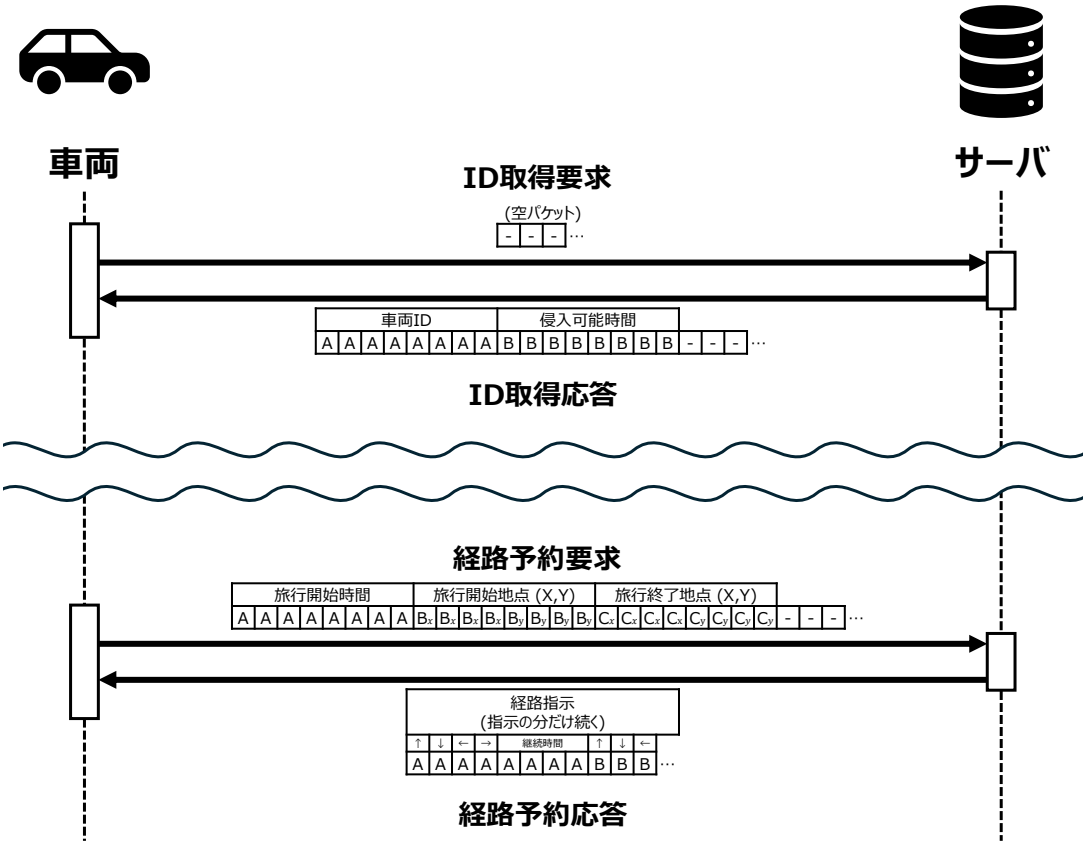

図 5: 車両とサーバの通信フローと通信内容

と新しい目的地をサーバに送信して経路予約を依頼する．サーバは計算した経路を予約を依頼した車両に送信する．1 回の実験時間は最大 3 分間とし，車両台数を 1 台から 3 台まで段階的に変化させて実験を実施する．

4.4 評価指標

本システムの有効性を定量的に評価するため，以下の 2 つの指標によって評価を行う．まず，µT-Kernel 3.0 上での CPU 使用率を評価する．CPU 使用率は，一定期間中に CPU がアクティブに動作している時間の割合であり，次の式で表される．

$$\text{CPU 使用率} = \frac{\sum_{i=1}^{n} T_i}{t_{\text{end}} - t_{\text{start}}}$$

ここで T_i は i 番目のタスクの実行時間，n はトレース対象のタスクの総数，t_{end} は最後のイベントが発生した時刻時刻，また t_{start} は最初にイベントが発生した時刻である．車両台数 1 台の条件下で，サーバおよび車両それぞれの CPU 使用率を測定する．

次にシステムの実験継続時間を評価する．実験継続時間は 3 分間の実験において車両同士の衝突やサーバのエラーなどが発生せず，システムが動作した時間である．車両台数を 2 台および 3 台に設定し，それぞれ 5 回ずつ実験を行うことで，車両数の増加がシステムの安定性に与える影響を評価する．

表 1: CPU 使用率測定結果

サーバ (%)	車両 (%)
0.20	0.90
0.20	0.88
0.20	0.90
0.20	0.89
0.20	0.88

4.5 実験結果

表 1 に CPU 使用率の測定結果を示す．車両台数 1 台の条件下で実施した 5 回の実験において，サーバの CPU 使用率は全実験を通じて 0.20%と一定の値を示した．車両の CPU 使用率は 0.88%から 0.90%の範囲で推移し，平均値は 0.89%であった．

表 2 に車両台数 2 台での実験継続時間を，表 3 に車両台数 3 台での実験継続時間を示す．車両台数 2 台での実験では，5 回中 4 回が正常終了し，成功率は 80%であった．異常終了した 1 回の実験では，車両同士の衝突が発生し，実験開始から 1 分で終了した．一方，車両台数 3

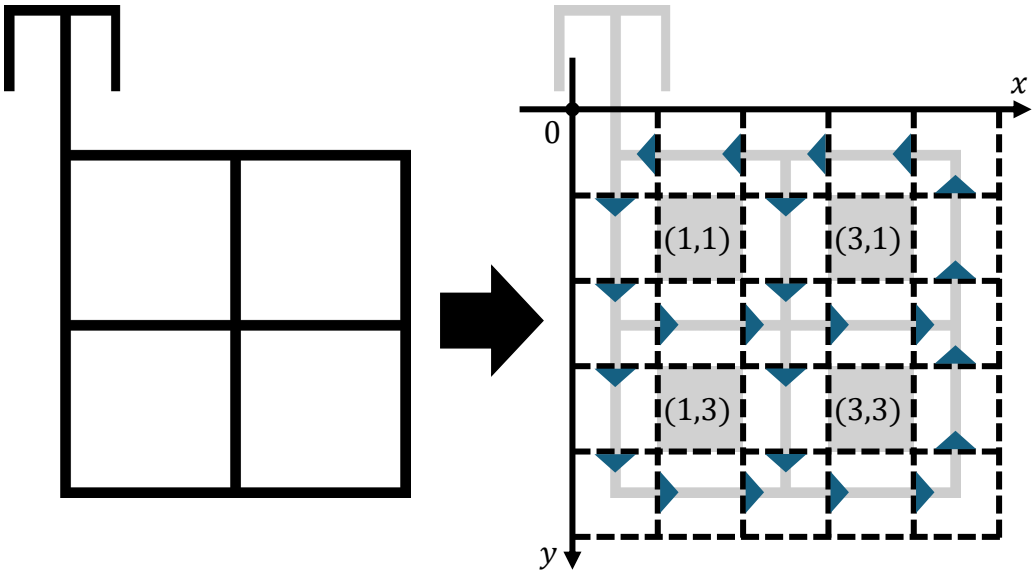

図 6: 実験に用いた道路環境とグリッド

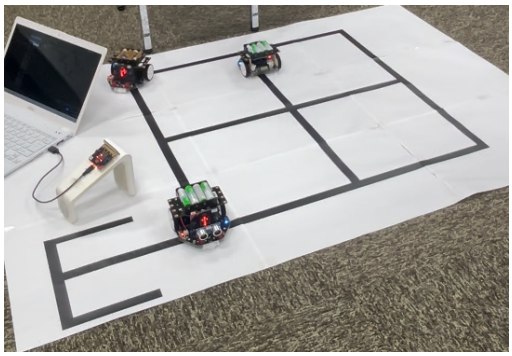

図 7: 車両 3 台での実験の様子

表 2: 車両台数 2 台での実験結果

実験継続時間 (分)	異常の原因
3	-
3	-
1	車両同士の衝突
3	-
3	-

台での実験では，5 回中 1 回のみが正常終了し，成功率は 20%にとどまった．異常終了の原因として，車両同士の衝突が 2 回，サーバの Hard Fault エラーが 2 回発生した．衝突事例の 1 回においては，初回の衝突後に後続車両が前の車両に追突するという連続的な衝突が観測された．

表 3: 車両台数 3 台での実験結果

実験継続時間 (分)	異常の原因
1	車両同士の衝突
1	Hard Fault エラー
3	-
1	Hard Fault エラー
1	車両同士の衝突

5. 考察

CPU 使用率の結果から，µT-Kernel 3.0 の効率的なタスク管理によって，時空間グリッドシステムが低い計算量で動作することが確認された．これは，システムに他の機能を追加する計算リソースの余裕があることを示しており，高い拡張性を有するシステムの構築が可能である．さらに，タスクの優先度を設定することで，安全性の担保が期待できる．一方，車両台数の増加に伴いシステムの安定性が低下することが確認された．車両台数 2 台では成功率 80%であったが，3 台では 20%まで低下した．原因は Hard Fault エラーと，車両同士の衝突である．

Hard Fault エラーは通常，メモリアクセス違反，スタックオーバーフロー，無効な命令実行，またはバスエラーなどが原因となる．今回の場合は，新たな予約要求に対して経路探索において有効な解を見つけることができずにメモリのアクセス違反を起こしたと考えられる．例えば，通信の遅れや制御の誤差が蓄積した際，次の出発地に到達する時刻に，その場所がすでに他車両により

予約されていれば，走行可能な経路は存在しないため解を見つけることができない．

車両同士の衝突については，時空間グリッド予約システムの想定する車両の動きと実際の車両の動きに誤差が生じることが主要な原因と考えられる．micro:Maqueen Plus の個体差による速度のばらつきや，ライントレースの精度のばらつき，また通信の遅延により予約された時刻通りにグリッドを通過できない場合がある．このようなタイミングの誤差が蓄積されると，複数の車両が同じグリッドに同時に侵入し，衝突が発生する．

車両同士の衝突等の問題は，制御や通信遅延，個体差などハードウェア起因の誤差を考慮していないことが主な要因である．そのため，システムの安定性を向上させるには, 動的な車両位置の把握に基づく予約の更新, キャンセル機能や，衝突リスクが高い場合に安全な停止や待避を行うフォールトトレラントな設計が求められる．また，RTOS の特徴であるタスク優先度制御を活用し，走行中に衝突リスクが検知された際には，通信や予約処理よりも高優先度で緊急回避行動を行う自律タスクへ即座に遷移させることで，安全性を確保できる．さらに，予約失敗時に安全にグリッド外へ退避するなど，異常時のエスケープ動作を担う補助機能を併設し，協調と自律を両立した設計へ拡張する必要がある．これにより，実環境での遅延や誤差にも耐性を持つ，より実用的な協調型走行システムが設計可能であると考えられる．

6. まとめ

本研究では，協調型自動運転における通信および処理負荷を軽減するために，RTOS 上に時空間グリッド予約システムを実装し，各車両の走行制御を優先度に応じて動的に行うことで，即応性と安定性の両立を図った．計算量と走行調停の確実性を評価するための評価実験において CPU 使用率と実験継続時間を測定した．その結果，サーバの CPU 使用率は 0.2%，車両の CPU 使用率は平均 0.89%と非常に低い値を示し，低い計算量で動作可能なことが確認された．一方，実験継続時間の評価では，車両台数 2 台で成功率 80%，3 台で 20%となり，車両台数の増加に伴いシステムの安定性が低下することが確認された．提案システムは計算量の面では有効性を示したが，安定性向上のためにハードウェアの特性を考慮した設計が必要である．

参考文献

[1] 菅沼直樹, 米陀佳祐, 柳瀬龍, 倉元昭季, 山下隆義, 藤吉弘亘, 目黒淳一.「②自動運転 (レベル 3, 4) に必要な認識技術等に関する研究」, *SIP* 成果報告書, 2022, vol.2022, no.1, p.120-128.

[2] 伊藤建.「MaaS・自動運転の経済産業省の取り組み状況について」, *IATSS Review*, 2025, vol.49, no.3, p.297-304.

[3] 小菅謙次.「地域公共交通の課題解決のための自動運転システム導入の可能性―経済的持続可能性の考察から―」, 国際公共経済研究, 2023, vol.34, p.94-104.

[4] 中川正夫.「自動運転に用いられるセンサ類の認識性能評価」, 電気学会誌, 2023, vol.143, no.8, p.521-524.

[5] 鎌田実.「自動運転技術・取組の最近の動向」, 学術の動向, 2022, vol.27, no.7, p.51-55.

[6] 小川伯文.「協調型自動運転のための通信方式の検討（概要）」, *SIP* 成果報告書, 2021, vol.2021, no.1, p.39-42.

[7] 神原滉一, 中里仁, 山田峻也, 高田広章, 渡辺陽介, 佐藤健哉, 塚田学.「協調型自動運転のためのネットワーク状態分析・可視化」, 映像情報メディア学会技術報告, 2023, vol.47, no.6, p.365-370.

[8] 小林雅文, 畑﨑由季子, 高柳雄一, 馬渕透, 川邉俊一.「インフラ協調型自動運転のための信号情報提供技術（V2I）の開発」, *SIP* 成果報告書, 2021, vol.2021, no.1, p.16-23.

[9] Kenya Sato. "Progress of automated driving based on cooperative ITS and the necessity for standardization," *ITS Standardization Activities of ISO/TC204*, 2024, p.5-9.

[10] Gaurav Kumar, Suresh Mikkili. "Critical review of vehicle-to-everything (V2X) topologies: Communication, power flow characteristics, challenges, and opportunities," *CPSS Transactions on Power Electronics and Applications*, 2024, vol.9, no.1, p.10-26.

[11] 木村聡, 布本剛史, 小川真人, 小西友明.「協調型自動運転のユースケースを実現する通信方式の検討」, *SIP* 成果報告書, 2022, vol.2022, no.1, p.85-93.

[12] 木村健太, 佐藤健哉.「協調型自動運転に向けた時空間グリッド予約に基づく走行調停手法の検討」, *ITS* シンポジウム論文集, 2019, vol.2019, no.1-A-05, p.1-6.

[13] Kotaro Yamamoto, Rui Teng. Kenya Sato. "Simulation evaluation of vehicle movement model using spatio-temporal grid reservation for automated valet parking," *IEEE Open Journal of Intelligent Transportation Systems*, 2023, vol.4, p.261-266.

[14] Hiroto Umeda, Kenya Sato. "A Collaborative Merging Method for Connected Vehicles Using Spatio-Temporal Grid Reservation," *2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC)*, 2025, p.1-2.

[15] トロンフォーラム. "µT-Kernel 3.0 仕様書". GitHub. `https://github.com/tron-forum/mtkernel_3`, (参照 2024 年 6 月).

[16] Jonny Austin, Howard Baker, Thomas Ball, James Devine, Joe Finney, Peli De Halleux, Steve Hodges, Michał Moskal, Gareth Stockdale. "The BBC micro: bit: from the UK to the world," *Communications of the ACM*, 2020, vol.63, no.3, p.62-69.

[17] BBC. "micro:bit 概要". micro:bit 公式サイト. `https://microbit.org/ja/get-started/features/overview/`, (参照 2025 年 7 月).

[18] IFTiny. "micro:bit データシート". iftiny: docs. `https://docs.iftiny.com/jp/education/microbit/stand-alone/info/datasheet/`, (参照 2025 年 7 月).

[19] IEEE. "IEEE Standard for a Real-Time Operating System (RTOS) for Small-Scale Embedded Systems," *IEEE Std 2050-2018*, 2018.

[20] DFRobot. "micro:Maqueen Plus." DFRobot. `https://www.dfrobot.com/blog-1523.html`, (参照 2025 年 6 月).

[21] Peter E. Hart, Nils J. Nilsson, Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, 1968, vol.4, no.2, p.100-107.