無線通信による共同作業ロボットの研究

システム科学技術学部　情報工学科
1年　福田　徹平
1年　棚橋　佑斗
1年　服部　聡史
指導教員　システム科学技術学部　情報工学科　准教授　石井　雅樹
助教　伊東　嗣功
教授　堂坂　浩二
助教　橋浦　康一郎

１．研究の動機と目的

昨今話題となっている「IoT｜モノのインターネット」をはじめとする情報技術は，今後ますます必要とされる事項であろう．私たちは情報工学科に属しており，大学生であるうちにそのような技術に積極的に関わることは将来のための重要な経験となると考えた．

本自主研究では，上記のような情報技術を利用した一組のシステムの計画から製作までをグループで行う．実際の製品開発と同様の手順をたどることで，将来に生かせる技術と経験を得ることが目的である．現代のデジタル社会では先に述べたようにIoTなどの技術が進歩し，ハードウェアとソフトウェアのより一層の連携が必要となる場面が数多くある．複雑なひとつのシステムを構築しようとしても自分一人だけの知識では目的を達成することは難しい．そのため，グループとして研究し，複数人でモノづくりを進める体験を自主研究という機会で行う価値は大きいと考えた．いずれ社会に出る際にも，人と意見を交え，協調する力は必要である．これらのような理由から，本自主研究を始めることにした．

２．研究方針

グループのメンバーが，それぞれの特に興味を持っている分野においてリーダーとなることで，互いに意見や知識を共有しながら研究を進める．

本研究で具体的に何をつくるかという議論の結果，「無線通信を介し，役割の異なる3台のロボットを使って*KAPLAを特定の形（図1）に組み立てる」ことを目標とした．ロボットの制御には*Arduino microと*TWELITEを使用する．

機体設計とセンサなどの位置設定を，主にハードウェア担当が行う．各ロボットの役割を明確にし，機体の安定した挙動を目指す．ロボットを動かすためのアプリケーション作成等はソフトウェア担当が行う．Arduinoと無線通信を用いて3台のロボットの円滑な連携を目指す．両担当はどちらが欠けてもシステム構築は望めないため，互いの意見交換や知識の提供は不可欠である．

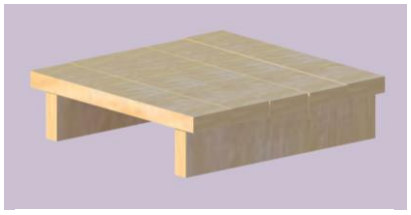

図１　KAPLA の組み立ての例

***KAPLA**　長辺を約117.4mmとし，各辺が1:3:15の直方体で積み木の一種．材にフランス海岸松を使用し，他に類を見ない高い加工精度に定評がある．

***Arduino**　マイコンを搭載したArduinoボードおよびArduino IDEから構成されるシステム．デジタル制御用システムの中でもシンプルで，消費電力が非常に小

さく，世界中で普及している．今回は機体サイズの制約上，小型版であるArduino Microを採用した．

***TWELITE**　IEEE802.15.4を標準プロトコルとした32bit無線マイコン．同規格のZigbeeと同じく小型で消費電力も小さいながら，双方向通信や長距離での安定した通信ができる．

３．研究内容の詳細

3.1　ハードウェア｜機体，センサ等

ハードウェア担当は，連携してKAPLAを組み立てる3台のロボットを製作する．機体設計には3次元モデリングソフトウェアであるGoogle SketchUpを利用した．機体材料は3Dプリンタで出力したもののほか，木材等も充てた．

ロボットを設計するにあたってまず必要なのが「ロボットを使用して最終的に実現したいこと」である．今回は，図1に示したようなユニットを前後と上方向に並べた構造をつくることを目的とした．ひとつのユニットをつくるためには，まず平行な2本の板を立てて設置し，その上に5枚の板を平らに置けばよい．このことを前提として，以下に3台のロボットを利用してKAPLAを積む工程を説明する．

はじめに，平行な2本の板を設置する専用ロボットRV-3が，図2(a)に示すように，設定したユニット数に必要な数だけ板を設置する．

RV-3は前後移動とKAPLA縦排出の2動作のみ可能である．前後移動にはDCモータを2軸4輪の駆動ユニットにつなぎ，KAPLAの押し出しにはひとつのサーボモータを使用した．

次に，同図(b)に示すように，RV-3が置いた板の上に，床板を1枚ずつ並べていく．この作業は床を置く専用のロボットRH-2が行う．

RH-2は前後移動とKAPLA横排出の2動作のみ可能である．こちらも前後移動にはDCモータの回転を4輪に伝え，押し出しはサーボモータを使用した．

上述の二つの工程を繰り返し，ユニットを組み上げていく．1階部分の組み立てが完了した後，上層階へ進む際は，同図(c)に示すように上下移動用のロボットEV-6を用いてRV-3とRH-2を任意の階層へ移動させる．

EV-6は左右移動とRV-3およびRH-2を載せるかごの上下移動のみ可能である．かごの上下にはステップ角1.8度のステッピングモータおよびリードスクリュ，左右移動にはDCモータをそれぞれ使用した．

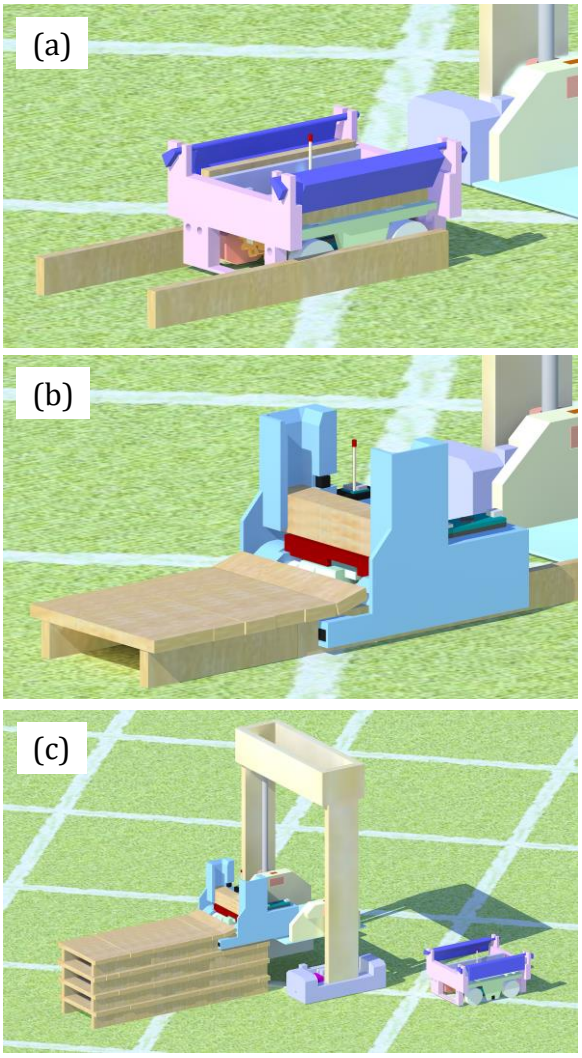

図 2　KAPLA の積み上げ工程

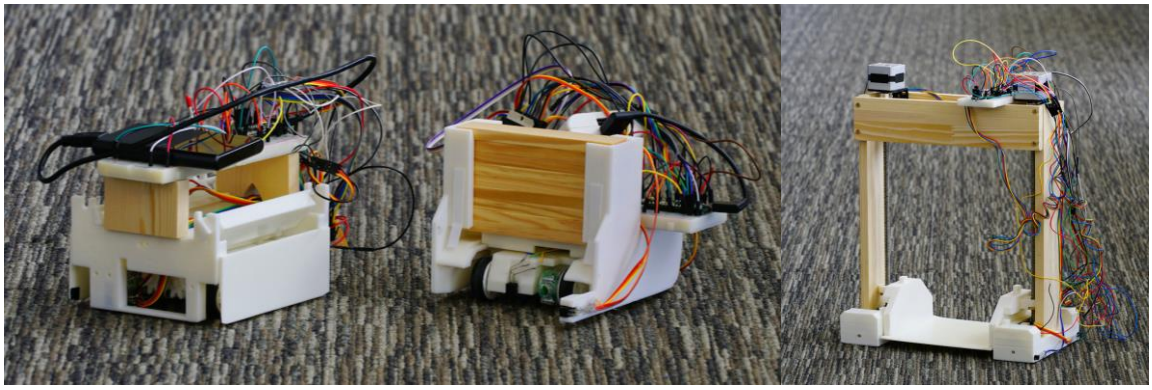

図 3　製作した 3 台のロボット（左から順に RV-3，RH-2，EV6）

以上のような手順でKAPLAの組み立てを進める．実際に製作した3台のロボットを図3に示す．前述の通り各ロボットは特定の動作しかできないため，これらを通信させてそれぞれの仕事を組み合わせる．KAPLAの落下位置の不具合など思い通りに動作しないことも多く，試作と調整を重ねつつ開発を続けた．

3.2　ソフトウェア｜動作プログラム

ソフトウェア担当は，3台のロボットを連携させるため各ロボットの動作に加え，相互通信システムを構築する．また，親機からインターネットへ接続し，サーバを経由して各種デバイスからのアクセスも可能にする．

プログラムの開発は主に2つの段階に分けて行った．1段階目はロボットに搭載されている多数のセンサを用いて独立した状態での動作，2段階目は各ロボットとの通信による連携した動作とした．1段階目については基本的なプログラムとなるため，ここでは2段階目の通信について説明する．

KAPLAを組み立てる動作を行うためには機体に多くのセンサを搭載する必要があり，TWELITEに接続できる上限を超えていた．そのため，ArduinoとTWELITEのシリアル通信を行い，Arduinoをセンサ・モータ類の制御に割り当て，TWELITEは通信のみの分担とした．TWELITEにはシリアル通信用のファームウェアを書き込み，バイナリ形式でデータを送受信した．この通信形式では1パケット80Byteで，最大640Byteまでのデータを送受信することができる．ヘッダやチェックサムに6Byte必要となるが，残りのデータ部については指定がないため，今回は表1に示す形式で親機から各ロボットへの命令を出した．

表1　通信のデータ形式

0xA5 0x5A 0x80 0x08 0x78	0x10 0x11 0xBB	0x01 0x1B 0xEE	0x2B	0xBB	0xA6
通信ヘッダ (5Byte)	特殊コマンド (3Byte)	移動コマンド (3Byte)	識別ID (1Byte)	動作モード (1Byte)	チェックサム (1Byte)

表1における特殊コマンドは各ロボットの持つ特定の動作用，移動コマンドはロボット下部のモータの制御用である．TWELITEには子機論理IDをつけることができるが，最大で(0x64)100台しか設定できない．そのため大量の接続が必要となるIoTシステムへの応用も考え，命令内の識別IDによる操作を行えるようにした．これにより今回のデータ形式を用いた場合，最大接続可能数は未使用の627Byte分を識別IDに割り当てることで，計算上160,768台分まで拡張可能である．動作モードは，自動でKAPLAを組み立てる動作，手動での操作を行うなどの切り替えに使用した．

実際の通信では，データが途中で破損することもあり，安定性が低くなってしまうため，チェックサムとは別でデータの破損/形式チェックとデータの再送リクエスト処理

などをロボット操作用プログラムとは別に作る必要があった．Arduino側ではシリアル通信時の受信バッファからのバイナリデータの取得が処理に追いつかないため，プログラム最適化など，かなりの改良を行った．

2段階目の通信による連携は，ハードウェアの設計の時点で動作が決定していたため，工程終了ごとに通信で命令を送ることで可能である．

今回は，親機と子機の単純な通信だけではなく，PC・スマートフォン等のデバイスからの操作を実現した．操作を行う流れとしては，はじめに，ブラウザ上で動作するJavaScriptのXML Http Requestを用いて非同期処理によってサーバへ一旦命令（データ）を送る．次に，親機と接続するコンピュータ上で，新たに開発したアプリケーションソフト｜BeeApp.exeがサーバ上のPHPにリクエストを送り，命令を取得する．最後に取得した命令（データ）を，コンピュータから親機，親機から子機へとシリアル通信によりバイナリデータとして送信する．

しかし，このように子機までの経路が長い場合，遅延が問題になってくる．今回，サーバにはMicrosoft Azureを利用した．また，自主研究の残された期間を考慮し，通信方式はCometとAjaxのいずれかに絞られたが，サーバ側の同時接続数に係る不具合を考慮し，Ajaxで実装した．実際に発生した遅延時間を図4に示す．

・サーバ平均応答時間：　25.29ms
・データ送信時間平均：　75.18ms
・データ取得間隔平均：　30.00ms
・推定平均遅延時間　：　155.76ms

この値は，Device > Server > BeeApp間の遅延時間であり，実際はこれに加えてTWELITE｜親機 -> TWELITE｜子機 -> Arduino の通信があるため，さらに遅延時間は長くなると考えられる．結果として多少遅延が発生したが，今回のシステムではリアルタイム性を必要としないため問題は無かった．

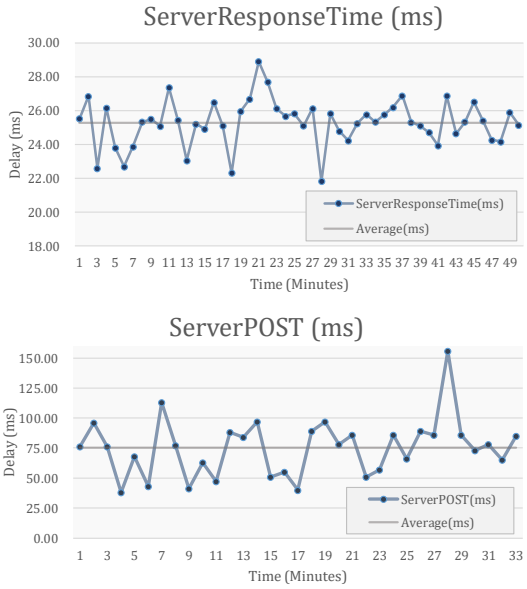

図 4　遅延時間

４．結論

機体とソフトウェアを新規に開発し，最終的にスマートフォンからの自動/手動操作も可能にした．ハードウェア，ソフトウェアともにまだ改良の余地があり，今後も手を加えていく．なお，今回の研究で使用した技術はIoTにも利用されているものである．複数のデバイスを同時に動かす点やスマートフォンからの操作が可能な点は，家庭用の電気製品や玩具などにも応用が可能であるため，研究で培った技術は今後有効に活用できると考えられる．

５．新たな課題

今回製作したロボットはデータ形式が一致すればすべての命令を実行してしまうため，だれでもアクセスできてしまう．実用的なシステムへ近づけるためには，セキュリティ面を改良しなければならない．また，通信を行う上で今回はサーバを経由させたが，すべてのロボットのデータが送信されるため消費リソースが大きかった．この課題の改善のためエッジコンピューティングを導入し，負荷を分散することで大規模な開発に対応させることも視野に入れていきたい．