論　文

携帯端末を用いた線追跡機械のPID姿勢制御手法に関する検討
ー タイ日工業大学とのコンテスト用マシン作成と制御方法ー

矢野　昌平[1]　・Prawet UEATRONGCHIT[2]

[1] 電気電子システム工学科（Department of Electrical Electronics System Engineering, Nagaoka National College of Technology）
[2] コンピュータエンジニアリング科（Department of Computer Engineering, Thai-Nichi Institute of Technology ）

Computer Vison Guidance of a Line Following Robot using Mobile Phone
- Description of How to Create Machines for Contest with Thai-Nichi Institute of Technology -

Shouhei YANO[1] and Prawet UEATRONGCHIT[2]

Abstract

The purpose of this report is to describe the implementation of a computer vision guided line following robot. The robots were utilized in the contests in 2013 and 2014 with students of Thai-Nichi institute of Technology and Nagaoka National College of Technology students in a student overseas dispatch training to Thailand. The robot reads the line on the course from the camera of mobile phone. The mobile phone installed Android will be applied to capture video, run the line following by communication via Bluetooth to the robot base. An Arduino microcontroller will speed drive the robot with PWM (Pulse Width Modulation) technology. An Android phone traces the line by control method. In this report, we describe the algorithms of the line trace which are included the PID (Proportional Integral Derivative) algorithm, the PID algorithm calculates the motor speeds such that the robot moves in a smooth fashion around line course, and how to the robot and the development environments of Arduino and Android on the PC.

Key Words: *Computer vision, Android, Arduino, oversea student training*

１．はじめに

近年，国際競争が激化し生産拠点を海外に移転する企業が年々増加している．上場企業の 7 割が海外生産を実施するなど，ものづくりの現場の多くは海外へと移転している．エンジニアには技術力以外に，英語力や国際化が求められている．将来のエンジニア育成を目指す長岡工業高等専門学校（以下，長岡高専と略す）においても，学生を海外に派遣し，現地学生と国際交流を行う学生海外派遣研修の取組みが行われている．その一つとして，2013 年からタイ日工業大学と長岡高専とで交互にコンテストを開催している．コンテストは 8 月下旬から 9 月に，タイにおいて開催される Android Following Line Robot (以後 AFR と記す)と，5 月に長岡高専で開催される Gr-SAKURA LED Music and Dance (以後 GLMD と記す)である．日本とタイの学生とで混合チームを作り，設定課題について共同で作品を制作し，他チームとレースや作品のパフォーマンスを競い合う．GLMD コンテストは 2014 年に，AFR コンテストは

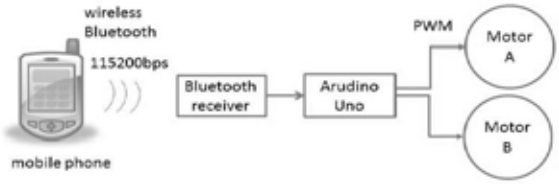

図-1 ライントレースロボットの構成図

図-2　AFR で使用するハードウェア

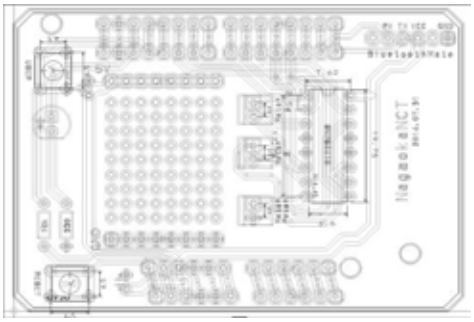

図-3 AFR 用シールド配線パターン

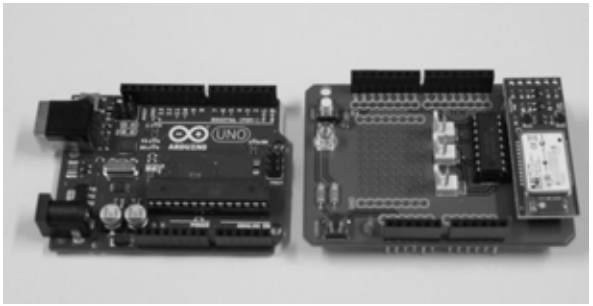

図-4 AFR シールド回路図

2013 年と 2014 年とで既に 2 回開催されている[1) 2)]．本論文においては，AFR コンテストで用いたライントレースロボットの開発方法と動作アルゴリズムの解説および検討を行った．

２．ハードウェア構成

ライントレースロボットの構成を図-1 に示す．ロボットは，携帯端末からの操縦指示を Bluetooth 通信によりモータ制御用マイコン(Arduino Uno)に送信する．マイコンには Bluetooth 通信モジュールが搭載されており，受信した信号に応じたモータ制御を行う．ロボットには左右のモータがついており，PWM 信号によりモータ回転数が制御され走行する．全ハードウェアは日本やタイはもちろん多くの国で入手可能である．AFR コンテストでは，事前に各国で部品を入手し作成したロボットを持込んだ．各チームはマシンやプログラムの調整，コースのテスト走行に多くの時間を費やすことができた．図-2 に主なハードウェアの写真を示す．

２．１　携帯端末

ライントレースロボットでは背面カメラを装備し Android が搭載されている携帯端末を用いる．携帯端末として KYOCERA 社製 DIGNO ISW11（図-2①）を使用した．Android OS は Ver.2.3.5 である．

２．２　Arduino Uno マイコンボード

Arduino は安価で簡単に使用できるように 2005 年にイタリアで開発された組込みボードである．基板上に AtmelAVR マイコンが搭載され，5V で駆動する．Arduino の統合開発環境を持つ PC より，クロスプラットフォームで開発でき USB 接続でプログラムをダウンロードすることでスタンドアローン動作が可能である．Arduino Uno (ATmega328) マイコンボード (図-2②) を使用した．

２．３　Arduino プロトシールド

プロトシールドは Arduino Uno の上に設置し，機能拡張用の電子回路を作成する基板である．本論文では，電子回路の安定性を高めるため，プリント回路パターン基板を起こした（図-2③）．図-3 に配線パターンと図-4 に作成したプロトシールド外観写真を示す．図-4 において左は Arduino である．図右は作成したシールドで，Arduino の上にピンコネクタ端子をもちいて設置可能である．基板上には Bluetooth モジュールや H-Bridge の他に LED やコンデンサ，抵抗，プッシュスイッチが配置される．図-5 に AFR 用シールド回路図を示す．基板を起こすことは時間と費用が掛かるため，数が少ない場合は市販されている SparkFun 製プロトシールドキット（SFE-DEV-0914）（図-2④）を購入し，キットの

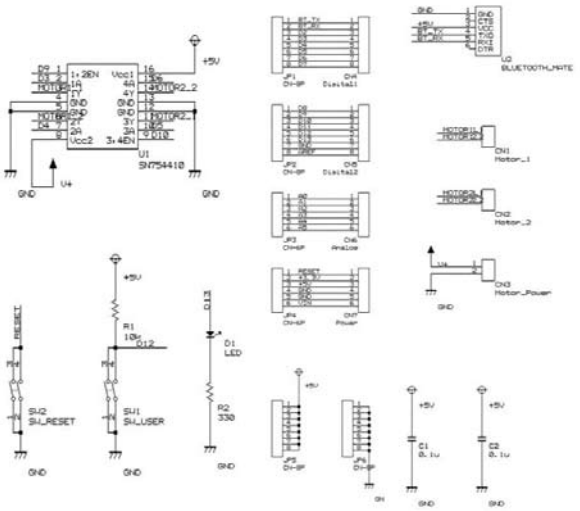

図-5 AFR シールド回路図

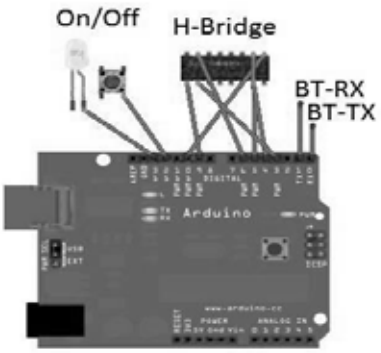

図-6 AFR シールド回路概念図

図-7 Bluetooth モジュール

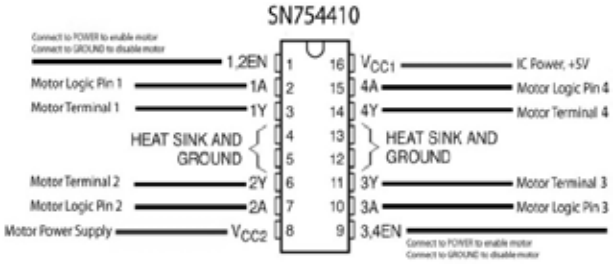

図-8 Hブリッジ IC SN754410 ピンアサイン

ユニバーサルエリアに電子回路を作成する方法が良い．図-6 に AFR シールド回路の概念図を示す．Bluetooth モジュール用に通信端子 (BT-RX, BT-TX) と電源 +5V，GND，H-Bridge 用に +5V，GND，MoterA+，A-，B+, B-そして SW，LED の各端子に合計 12 箇所の接続を行う．

２．４　Bluetooth モジュール

携帯端末との通信にはデジタル機器用近距離無線通信の「Bluetooth」を利用する．これは，携帯端末や周辺機器等がケーブルを使わずに，ワイヤレスで接続され機器間で音声やデータをやりとり可能とするものである．Bluetooth モジュールは，図-7 に示す sparkfun 社の BlueSMiRF WRL-12582 を使用した．

２．５　H-Bridge

モータに駆動 IC にはテキサス・インメンツ社製 H ブリッジ IC SN754410[3]を使用した．図-8 に SN754410 ピンアサインを示す．左右のモータは，図-7 中の 3-6pin と 11-13pin に接続を行う．モータ制御には 2-7pin と 10-15pin を用いる．

２．６　ツインモータギアーボックス

モータのギアボックスには，TAMIYA 社製 TWIN-MOTOR GEARBOX を使用した（図-2⑤）．2 つのモータのパワーを別々に伝えられ，組立て式であり，ギア比として標準 (58:1) と低速 (203:1) を選択できる．本論文では低速 (203:1) を利用した．

２．７　車体部品

車体部品には，TAMIYA 社製ユニバーサルプレートセット（図-2⑥）と，タイヤ部品には TAMIYA 社スリムタイヤセット（図-2⑦）を使用した．

３．Android アプリケーション開発環境

ライントレースロボットは携帯端末上で動作する Android アプリケーション（以後 Android アプリと記す）によって制御されている．Android アプリを開発するためには PC 上に開発環境を構築することが必要である．Android アプリは JAVA 言語で開発することができ，そのプログラム開発には 1）JDK (Java Development Kit)，2）eclipse，3）Android SDK を PC 上にインストールする必要がある．

３．１　JDK のインストール

JDK は，Java 言語を使ってプログラムを記述し，動作させる為に必要なソフトウェアのセットであり，オラクル社サイトからダウンロード可能である．

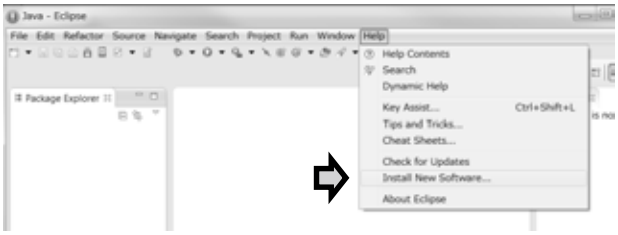

図-9　eclipse の plugin インストール

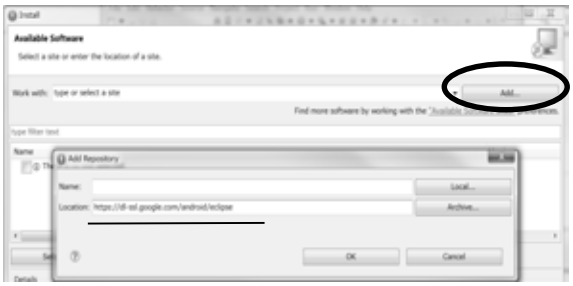

図-10　新しい Repository を Location に追加

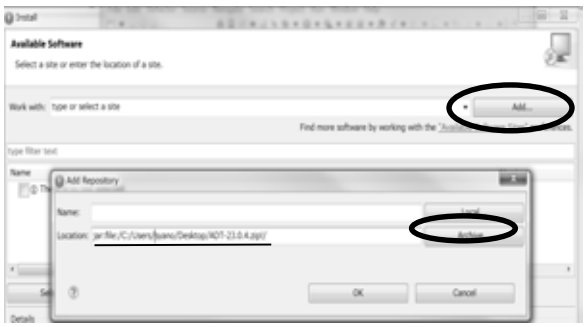

図-11　Plugin を直接指定する方法

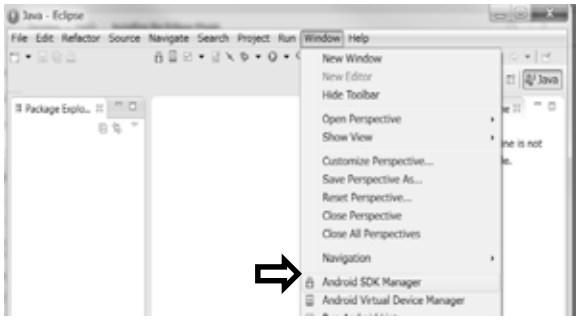

図-12 Android SDK Manager の起動

http://www.oracle.com/technetwork/java/javase/downloads/index.html．最新版 Java SE 8u20 (2014/10/3 現在) の JDK を選択し，Linux 版や Windows 版，そして 64bit 用(x64) と 32bit 用(x86) から使用している PC 環境に最適なものを選択しダウンロードする．ダウンロードを行う為には，「 Accept License Agreement」にチェックを入れライセンスに承諾する必要がある．本論文では Win 版 64bit をダウンロードし，完了後にインストール作業を行った．

３．２　eclipse のインストール

Eclipse は統合開発環境 (IDE: Integrated Development Environment) と呼ばれる java 開発ツールである．アンドロイド開発サイト http://developer.android.com/sdk/index.html より「Download Eclipse ADT」をクリックし，注意書きを読み「□ I have read and agree with the above terms and conditions」にチェックを入れ，adt-bundle-windows-x86_64-20140702.zip をダウンロードする．ファイル名の最後「20140702」は開発バージョンを示す．拡張子の「.zip」は圧縮ファイルであることを示す．アーカイブソフトを使って展開すると，展開先フォルダ中に実行ファイル eclipse.exe が作成される．

３．３　eclipse の ADT plugin のインストール

3.2 節でインストールした eclipse.exe を起動し，plugin のインストールを行う．メニューバーより [Help] → [Install New Software] を選択する（図-9）．インストールウィンドウ上の Work with:に新しい Repository を追加する為，「Add」ボタンをクリックし「https://dl-ssl.google.com/android/eclipse/」をロケーションに追加する（図-10）．

「Developer Tools」にチェックを入れ「Next」ボタンを押す．ダウンロードされるツールの一覧の表示画面で「Next」ボタンを押す．ライセンスを読みチェックを入れ「Finish」ボタンを押す．これにより plugin ソフトウェアのインストールが開始される．

プロクシーを使ったネットワークで行う場合は，ADT-plugin ファイルを先にダウンロードしておき，eclipse に登録する方法もある．その場合，https://dl.google.com/android/ADT-23.0.4.zip を任意のフォルダに保存し eclipse より[Help]→[Install New Software]を選択する（図-9）．インストールウィンドウ上の Work with:に新しいファイルを追加する為，「Add」ボタンをクリックし「Archive」ボタンを押し保存場所の ADT-23.0.4.zip を指定する（図-11）．

３．４　Android バージョンにあわせた API のインストール

携帯端末によって，稼動している Android OS のバージョンが異なる．アプリ開発のターゲットとなる携帯端末 OS バージョンを調べ必要な API をインストールする必要がある．携帯端末 OS バージョンは，携帯端末上で「設定」→「端末情報」→「Android バージョン」で知ることができる．次に Eclipse のメニューバーより [Window] → [Android SDK Manager] を選択する (図-12) .

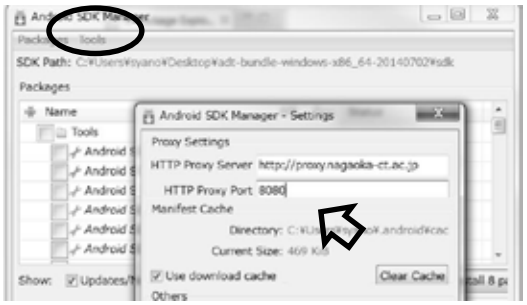

図-13 プロクシー設定

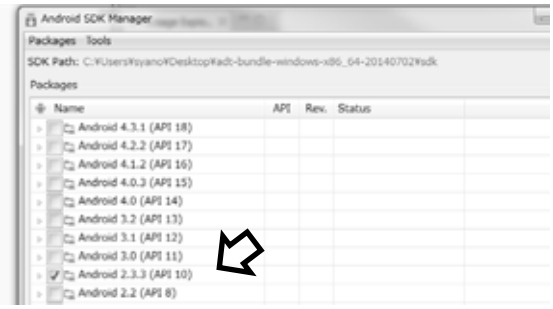

図-14　API の選択

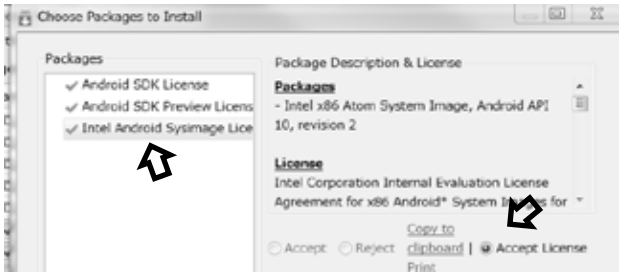

図-15　API のライセンス承諾

図-16　LineTracker プロジェクトのインポート

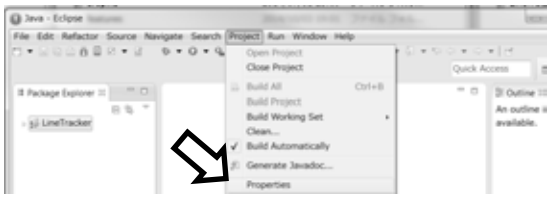

図-17　Properties の選択

SDK Manager を用いて API をダウンロードする為には，学内プロクシー設定が必要となる．Android SDK Manager のメニューバーより [Tools] → [Options] を選択し設定を行う（図-13）．次に OS バージョンに合った API をチェックする（図-14）．チェックした API 毎にライセンス承諾が必要となる．各々Packages ウインドウで選択し「Accept License」をクリックする（図-15）．

4．ソフトウェア構成

ライントレースロボットには 2 つのソフトウェアコンポーネントが必要である．Android アプリである LineTracker コンポーネンツと Arduino のプログラムである Amarino コンポーネンツである．

4．1　LineTracker の登録

LineTracker は PID 制御をモータのスピードを制御する Android アプリである．下記サイトより http://www.scs.ryerson.ca/mfiala/robotics_club/robot_projects/Radu_android_robot/LineTracker.zip ダウンロードし任意の場所に保存しておく．次に eclipse より LineTracker プロジェクトのインポートを行う．メニューバーより [File] → [Import…] → [General]→ [Existing Projects into Workspace]→[Next]と操作する．Select archive file: にて [Browse...] をクリックし LineTracker.zip を選び Finish を押す（図-16）．

インポートされた LineTracker には Arduino との通信で必要となる Amarino ライブラリが抜けている，サイト https://amarino.googlecode.com/files より AmarinoLibrary_v0_55.jar をダウンロードする．次に，プロジェクトに Amarino ライブラリを追加する．Package Explorer で LineTracker を選択，メニューバーより [Project] → [Properties] を選択する（図-17）．Properties 画面の左枠より [Java Build Path] → [Libraries タブ] → [AmarinoLibrary.jar] を選択し [Edit…] をクリックし AmarinoLibrary_v0_5.5.jar を選択する (図-18)．ビルドする OS のターゲットを設定する．[Project]→[Properties]→画面の左枠 [Android]とし，ビルドターゲットを選択する．本論文では[Android 2.3.3]とした（図-19）．

eclipse では，PC 上に仮想の携帯端末（以後，仮想端末と呼ぶ）を作成し仮想端末上で Android アプリの動作確認が可能である．メニューバーより [Window] → [Android Virtual Devices Manager] → [Create] → [AVD Name] を選択し，仮想端末に任意の名前を入れる．本論文では DIGNO とした．[Device: Nexus S (4.0”, 480x800: hdpi)] → [Target: Android 2.3.2-API Level 10] → [CPU/ABI: armeabi] → [Skin: No skin] とし仮想端末を作成する（図-20）．

４．２　仮想端末での動作確認

プロジェクトをビルドし，仮想端末上で動作確認を行う．メニューバーより [Run]→[Run Configurations…]→左画面より[Android Application]を選択しダブルクリック又は，New ボタンより[New Configuration]を作成する．Name: に識別名を記入し，[Android] タグより Project: に LineTracker を選択，[Target] タグより「DIGNO」を選択し[Run] を押す（**図-21**）．初期状態では，LineTracker の Android アプリは起動と同時に停止する．これは，仮想端末上で Bluetooth デバイス等の外部装置の動作を再現できず，通信エラーが発生する為である．仮想端末上で実行する場合は，通信処理部分のコメントアウトが必要である．Project Exproler より[LineTracker] → [src] → [line.tracker] → [lineTracker.java]のコードを表示し，21 行目の Amarino.connect 関数をコメントアウト（//を行頭に記入）する（**図-22**）．**図-23** に仮想端末上で LineTracker を起動した時の動作画面を示す．

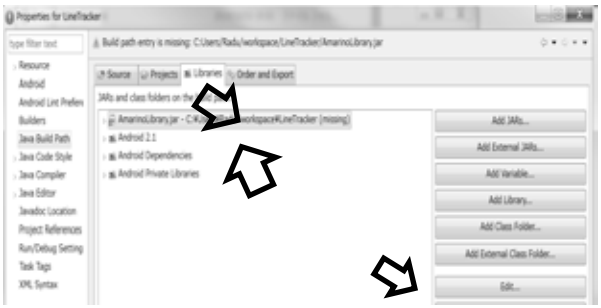

図-18 AmarinoLibrary_v0_5.5.jar の追加

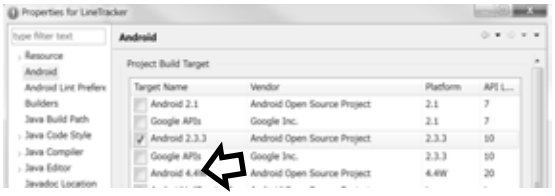

図-19 ビルドターゲットの選択

図-20　仮想端末の作成

図-21　Run Configurations 画面

４．３　LineTracker の構成

LineTracker は以下に示す①から④の 4 つの java ファイルから構成されており，各ファイルにはクラスのメンバーが定義されている．各ファイルと含まれるクラスおよびメソッドを以下に示す．

①LineTracker.java

•public class LineTracker extends Activity implements SensorEventListener

•protected void onCreate(Bundle savedInstanceState)

•protected void onStop()

•public void onAccuracyChanged(Sensor arg0, int arg1)

•public void onSensorChanged(SensorEvent event)

②ArduinoReceiver.java

•public class ArduinoReceiver extends BroadcastReceiver

•public void onReceive(Context context, Intent intent)

③Preview.java

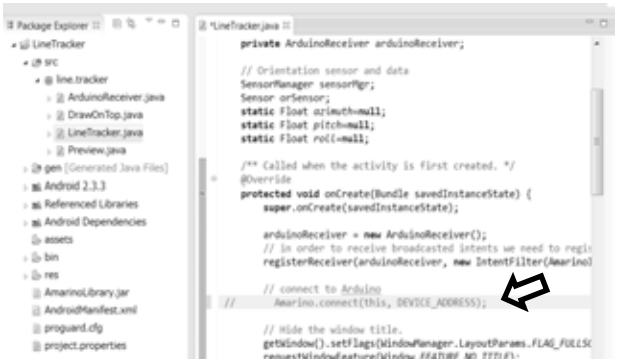

図-22　Amarino. Connect 関数のコメントアウト

図-23　仮想端末上での LineTracker 動作画面

•public class Preview extends SurfaceView implements SurfaceHolder.Callback

–Preview(Context context, DrawOnTop drawOnTop)

•public void surfaceCreated(SurfaceHolder holder)

•public void onPreviewFrame(byte[] data, Camera camera)

•public void surfaceDestroyed(SurfaceHolder holder)

•public void surfaceChanged(SurfaceHolder holder, int format, int w, int h)

④DrawOnTop.java

•public class DrawOnTop extends View

•public DrawOnTop(Context context)

•protected void onDraw(Canvas canvas)

•public int findTape(int[] rgb, int width, int height)

•public void grayscaleToBlackWhite(int[] rgb, byte[] yuv420sp, int width, int height)

LineTracker クラスは，Bluetooth デバイスの MAC アドレス取得・通信を行い，ライントレースアルゴリズムを呼び出す主幹となるクラスである．**図-24** に LineTracker クラスのダイアグラムを示す．Preview クラスはカメラの開始や携帯端末の画面の大きさを扱うクラスである．**図-25** に Preview クラスのダイアグラムを示す．

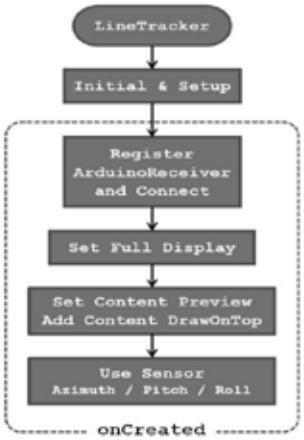

図-24　LineTracker クラスのダイアグラム

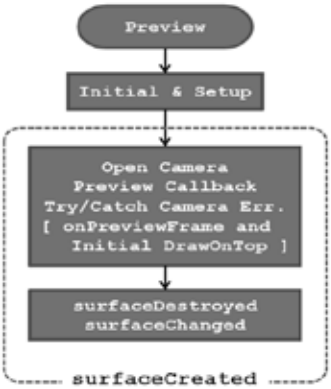

図-25　Preview クラスのダイアグラム

DrawOnTop クラスには，カメライメージを解析し，床に張られた黒テープ（以後，テープと呼ぶ）の位置を判定する FindTapeCenter メソッド．および，テープの中心と携帯画面の中心との差を計算し，PID 制御によりモータへの回転指令を出すライントレースアルゴリズムが含まれるクラスである．**図-26** に DrawOnTop クラスのダイアグラムを示す．FindTapeCenter メソッドは，白と黒に二値化されたカメライメージより，テープの中心を見つけ出す．**図-27** に FindTapeCenter の流れ図を示し，以下に処理手順について述べる．

1）カメライメージ画像を ImageWidth 方向に 50 ピクセルおきに pix 走査する．

2）pix 走査: ImageHeight 方向の黒色素子を探す．

3）ImageHeight 方向に minPixels 以上黒色の素子が連続した場合はテープ発見とする．

4）テープ発見とならなかった場合は ImageWidth 方向に 50 ピクセルずれ pix 走査を繰返す．テープを発見した場合は，ビット演算処理を行い発見されたテープを緑色表示する．

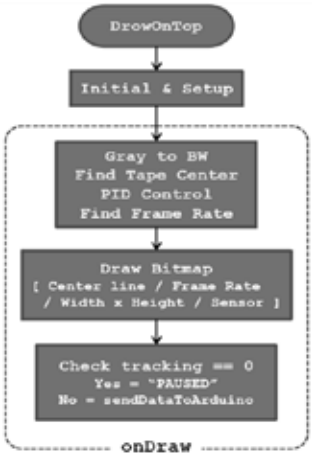

図-26　DrawOnTop クラスのダイアグラム

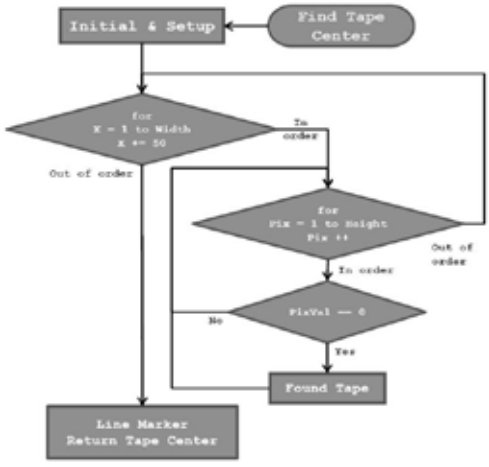

図-27 FindTapeCenter のフローチャート

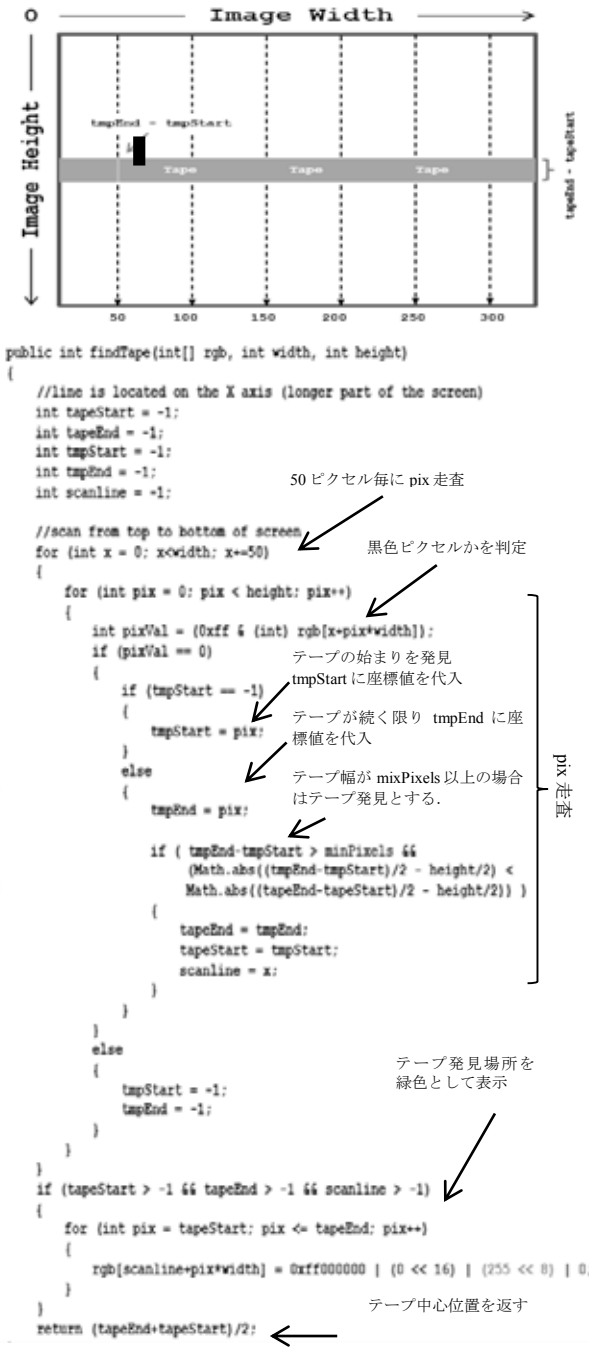

図-28　FindTapeCenter メソッド

図-28 に画面の走査方法とソースコードを示す.

４．４　携帯端末での動作確認

携帯端末において Android アプリを実行するためには，パソコンと携帯端末とをつなぐ USB ケーブル，および PC 側に携帯端末のドライバソフトがインストールされている必要がある．ドライバソフトは機種毎に異なる為，製造メーカのサイトよりダウンロードしインストールが必要である．機種によっては USB 接続ドライバの他，ABD(Android Debug Bridge)ドライバが必要な場合がある．また，携帯端末設定で開発者モードへの変更が必要となる．携帯端末にて，[設定]→[アプリケーション]にて[提供元不明のアプリ]にチェックする．[設定]→[アプリケーション]→[開発]にて[デバッグモード]，[スリープモードにしない]と[擬似ロケーションを許可]にチェックを入れる必要がある．携帯端末での実行は，メニューバーより [Run]→[Run Configurations] の [Target]タグより，ターゲットとなるデバイスとして[Active devices]を選択すればよい(**図-29**)．**図-30** に携帯端末上で LineTracker 起動した様子を示す.

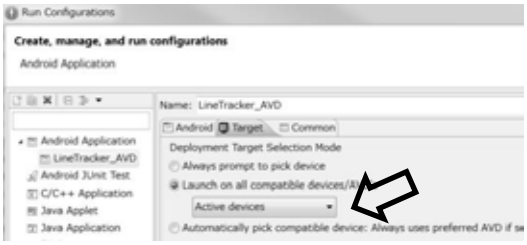

図-29　ターゲットデバイスの設定

図-30　携帯端末で LineTracker の起動

４．５　Arduino のソフトウェア

Arduino は，高価なロボット制御用デバイスの代わりとして開発されてきた．モータ制御のための PWM 出力，汎用性の高い I/O ピンを多く持つなど，ロボットの制御に適している．Arduino のプログラミングは C 言語で書かれている．PWM 制御や Bluetooth 通信のライブラリが提供されており，ライブラリ関数を使うことでコーディングの労力を節約できる．また，IC へのプログラムのダウンロードを含む統合開発環境が用意されている．統合開発環境の導入には，Arduino の開発サイト (http://arduino.cc/en/Main/Software) より Arduino IDE (最新版 Arduino 1.0.6 2014/09/30 現在)をダウンロードする．ダウンロード終了後にインストールを行う．

次に，Arduino コードを下記サイトより取得する．(http://www.scs.ryerson.ca/mfiala/robotics_club/robot_pr

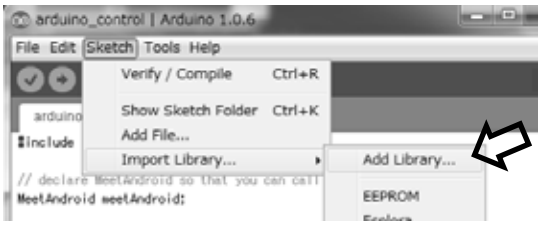

図-31 MeetAndroid ライブラリの追加

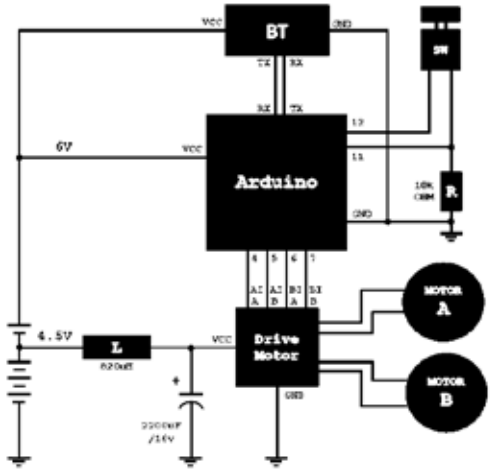

図-32 ライントレースロボット回路

ojects/Radu_android_robot/arduino_control.zip)

arduino_control.zip を解凍後， Arduino IDE を起動しメニューバーより [File] → [Open] arduino_control フォルダ内の [arduino_control.pde] を選択する．

arduino_control には Android 端末との Bluetooth 通信が用いられている，MeetAndroid ライブラリである MeetAndroid_4.zip を下記サイトより取得する．
http://amarino.googlecode.com/files/MeetAndroid_4.zip
Arduino IDE のメニューバーより [Sketch] → [Import Library…]→[Add Library…]とし，MeetAndroid_4.zip ファイルを選択する．これにより MeetAndroid ライブラリが追加される（図-31）．

メニューバーの下のチェックマークのボタンを押すことで，スケッチ（Arduino ではコードをスケッチと呼ぶ）のコンパイルとリンクがなされる．Arduino ではこの処理を「Verify」と呼ぶ．また，USB ケーブルで Arudino と PC を接続し，その隣にある右矢印のボタンを押すことでプログラムのダウンロードを行う．Arduino ではこの処理を「Upload」と呼ぶ．USB 接続は，PC 側でシリアル接続と認識されそのポート番号は PC 環境によって異なるため，メニューバーの [Tools] → [Serial Port] より適切なポートを選択する必要がある．

Bluetooth 通信に用いる通信速度として 119200bps を用いる．そのため，スケッチの 31 行目 Serial.begin の括弧内の数値を 57600 から 119200 へと書き直す必要がある．arduino_contorol では，図-32 に示す回路図を想定しプログラムされている．図-33 は完成したロボットの写真である．

図-33 完成したロボットの写真

５．PID 制御アルゴリズム

ライントレースロボットには PID 制御が用いられている．PID 制御は，フィードバック制御の一種であり，入力値$u(t)$の制御を出力値と目標値との偏差$e(t)$，その積分，および微分の 3 つの要素によって行う方法であり，式 (1) で定義される．

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (1)$$

PID 制御を離散時間で実現する為，DrawOnTop.java ファイル内の onDraw クラス内 (82 行目以降) にコードが記述されている．以下に PID 制御コードの各行を式 (2) から (6) として示す．

```
proportional = tapeCenter - mImageHeight/2;                    (2)
integral = integral + proportional;                            (3)
derivative = proportional - last_proportional;                 (4)
last_proportional = proportional;                              (5)
int lineError = (int) (proportional * Kp + integral * Ki +
derivative * Kd);                                              (6)
```

式(2)において，tapeCenter は FindTape メソッドがカメライメージよりライン認識を行ったテープ中心位置の横座標値を示す．mImageHeight は端末画面横幅を示し，2 で割ることにより画面中心座標と

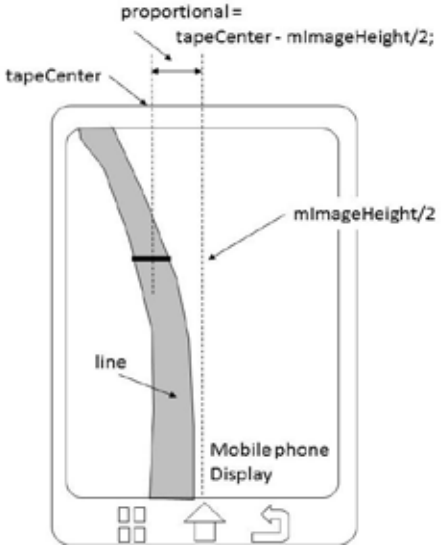

図-34　proportional 値の決定

なる．式(2)より，比例値 proportional には画面中心とテープ中心との差が代入される（**図-34**）．式(3)は積分値 integral を決定する．integral に，前回からの integral と現在の proportional の変数値を加算し代入すると積分を示す．式 (4)， (5) は微分値 derivative を決定する．現在の差を示す proportional と，前回の derivative の変数値を保持する last_ derivative との差を derivative に代入することで微分を示す．式 (5) において，現在の derivative の変数値を last_ derivative に代入し次のループの準備をする．

式 (6) において，proportional ， integral， derivative の各変数に比例ゲインを示す Kp，積分ゲインを示す Ki，微分ゲインを示す Kd との乗算を行い，総和を求めることで出力値を算出する．出力値は lineError に代入される．次に，lineError の変数値を用いたモータスピードの制御アルゴリズムについて述べる．LineError の変数値が 0 以下の場合は，**図-34** で示される様に携帯画面の中心より左側にテープのある可能性が高い．ロボットを左に回転させるため，左のモータを減速する．同様に lineError の変数値が 0 以上の場合は，右のモータを減速する．lineError の変数値が 0 であった場合は，ライン上をロボットが進んでいることを意味し，左右のモータスピードは最高速度の状態となる． ゲインを示す Kp，Ki，Kp の各変数は，DrawOnTop.java ファイル内の DrawOnTop クラス内 (32 行目以降) に int 型の変数として記述されている．使い勝手の問題から倍精度浮動少数型である double 型で Kp，Ki，Kp を宣言するよう修正を行う．各変数には初期ゲイン値として Kp=1，Ki＝2，Kd=0 が与えられている．PID 制御に最適なゲイン値を与える代数的な方法は存在しないため，試行錯誤的に最適値を探すものとなる．そこで，proportional の値に注目した状態推移手法を含め次の 3 手法を提案する．

手法1）proportional変数の値が一定値以下であった場合はPID制御アルゴリズムへの入力を0とする．

手法2）pix走査範囲を制限する．

手法3）携帯解像度を低くする．

手法2) と手法3) は携帯端末の処理速度を向上させる効果があり，提案手法の有効性については2014年コンテストにおいて確認を行った．

６．まとめ

携帯端末のカメラ映像を使って，コース上に書かれたラインをトレースするライントレースロボットを作成し，タイへの学生海外派遣研修においてタイ日工業大学の学生との2013年および2014年のコンテストに活用した．PC上にAndroidとArduinoの開発環境の構築について解説を行い，ロボットの作成方法と制御アルゴリズムについて解説と検討を行った．PID制御は制御方法として一般的であり広く用いられている．しかし離散信号でPID制御を実現し，ゲインを調整しその振る舞いを経験的に理解できる機会は少ない考えられる．本論文で作成したライトレースロボットは電子工学，アルゴリズム，制御工学の知識を深めるのに有用と考えられる．

謝辞：AFR,GLMDコンテストを開催・実施するにあたり，趣旨を理解し快く参加・協力して頂いた，学生をはじめ国際際交流推進センター長の吉野正信先生，中村将先生，皆川正寛先生，井山撤郎先生を始め，多くの方々のご協力を頂きました．本論文のプロジェクトに平成25,26年の長岡高専学長調裁量経費を頂きました．心より感謝申し上げます．

参考文献

1) S. Yano, Y. Tokoi, T. Kabasawa, M. Yamazaki, H. Miura, “Examination of ICT education for Embedded Software engineers”, Proceedings of the ISATE 2013, p-17, 2013.
2) M.Yamazaki, S.Yano and S.Nakamura “cooperative educational projects between NIT-Nagaoka and higher education institutions in ASIA”, Proceedings of the ISATE 2014, p-8, 2014.
3) http://www.ti.com/lit/ds/symlink/sn754410 (2014.10 参照)

（2014. 10. 6 受付）