

ArduinoIO（操作マニュアル）

九州工業大学大学院生命体工学研究科
我妻研究室
浦田竜行

1. ArduinoIO について

ArduinoIO は MATLAB/Simlink から Arduino の IO (Input/Output) を簡単に使うために開発されたライブラリである。ArduinoIO により、Arduino のプログラミングを行うことなく、MATLAB コマンドや Simlink から Arduino が使える。Simlink から使用する際も、Simlink モデルをコンパイル&ダウンロードして Arduino ボード上で実行するのではなく、Arduino は単なる IO ボードとして機能し、Simlink モデルの計算は PC 上で行われる。したがって、Simlink のすべての機能を使うことが可能である。また、Simlink が与えたサンプリング周期で動作するように、Simlink 上の時間と実際の時間を合わせるためのブロック「Real-Time Pacer」も用意されている。厳密なリアルタイム性には期待できないものの、サンプリング周波数が高くなければ問題なく動作する。

本マニュアルではインストール設定から装置の動作までの手順を説明する。なお、ArduinoIDE と MATLAB/Simlink 2012b が入っていることを前提に話を進める。ArduinoIDE についてはマニュアルフォルダに付属している。

2. ArduinoIO のインストール

ArduinoIO のインストールは、MATLAB/Simlink の設定と、ArduinoIDE へのライブラリのインストール、そして Arduino へのスケッチ転送の 3 ステップに分けられる。

2.1. ArduinoIO のダウンロード

マニュアルフォルダに付属している ArduinoIO の圧縮ファイルを確認する。次に ArduinoIO を解凍し、フォルダ ArduinoIO ごと ArduinoIDE のインストールの際に作成した Arduino フォルダの中に移動する。

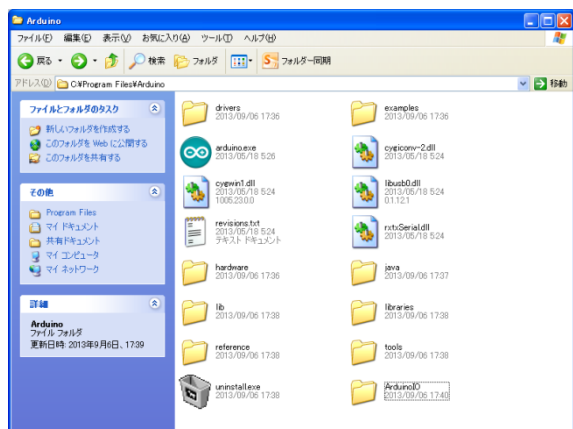


図1 Arduino フォルダ

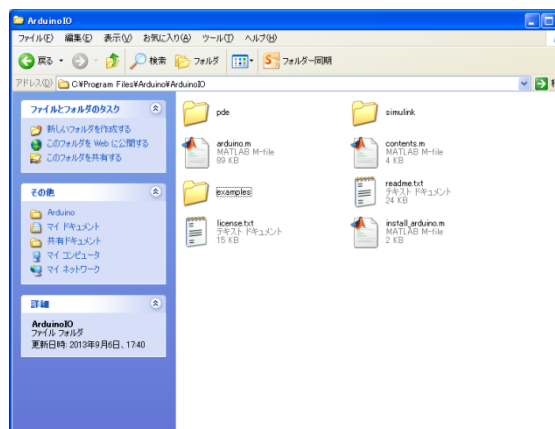


図2 ArduinoIO フォルダ

2.2. MATLAB/Simlink の設定

MATLAB/Simlink を起動し、ArduinoIO をインストールしたフォルダまで移動する。そして、install_arduino.m を実行する。MATLAB のコマンドウィンドウに以下のメッセージが表示されれば設定が完了する。

```
>> install_arduino  
  
Arduino folders added to the path  
  
Saved updated MATLAB path
```

ArduinoIO が正しくインストールされると、ArudinoIO を Simlink から使うためのブロックが追加されている。MATLAB/Simlink から「Simlink ライブラリブラウザ」を開き、ArduinoIO Library」が追加されていることを確認してほしい。

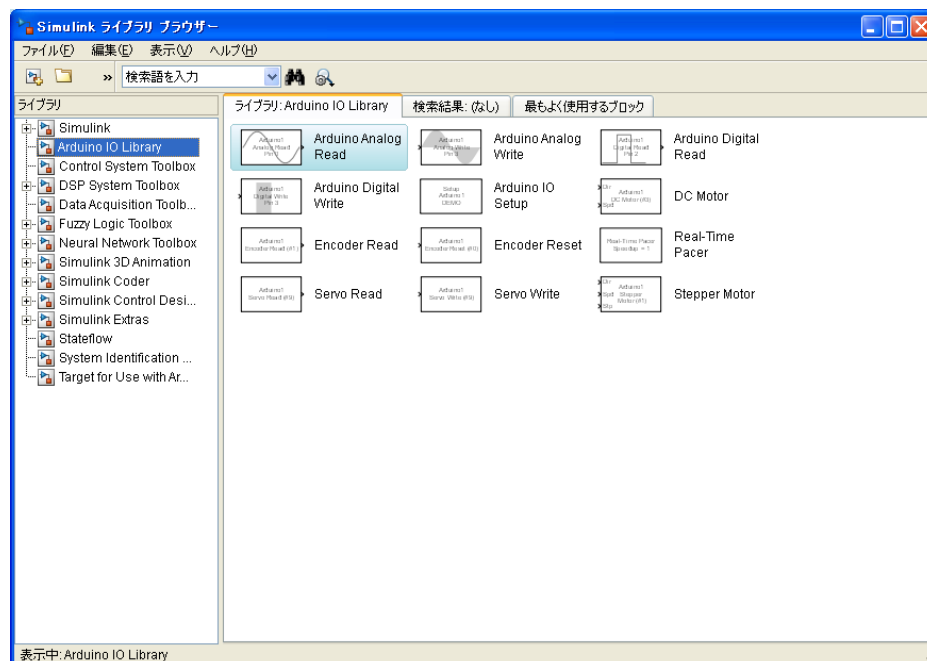


図3 Arduino IO Library

2.3. Arduino へのスケッチ

ArduinoIO では、MATLAB/Simlink と Arduino の入出力ポートとの通信のために、ArduinoIO に含まれるスケッチを Arduino へ転送しておく必要がある。この作業は最初の1回目に行うものである。（**RoTH** を実行した場合ここで転送するスケッチは消去されるので再度転送が必要になる）

Arduino が PC と USB ケーブルで接続されていることを確認し、ArduinoIDE を起動し「ファイル」→「開く」から次のスケッチを開く。

```
(Arduino のバージョン)¥ArduinoIO¥pde¥adiorrv¥adiorrv.pde
```

まず、ArduinoIDE の一番左の検証ボタン（チェックマーク）を押して正常にコンパイル

が出来ることを確認する。その後、マイコンボードに書き込むボタン（右矢印）を押して、スケッチを Arduino へ転送する。書き込みが正常終了すると ArduinoIDE のウィンドウの下に正常終了のメッセージと転送したスケッチサイズが表示される。

これで ArduinoIO のインストール&設定は全て完了である。

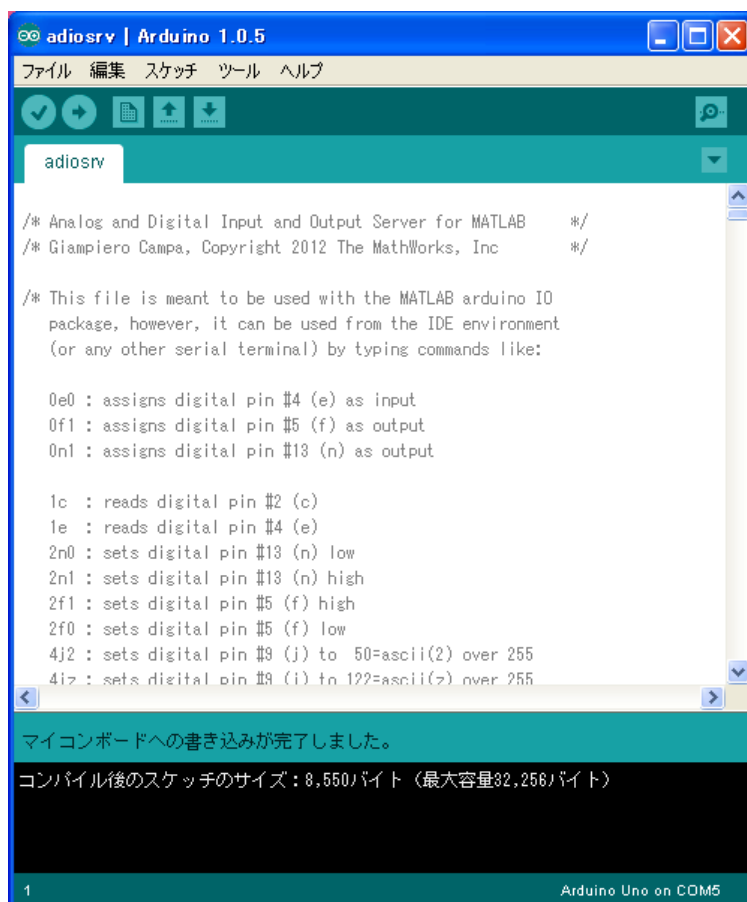


図4 書き込みが正常終了したときの ArduinoIDE のウィンドウ

3. 自作サーボモータの説明

3.1. はじめに

ロボットアームのように角度を目標角度へ正確に追従させる制御は、制御の中でも基本給の基本である。身近な所ではラジコンの RC サーボも角度制御系である。一般に目標角度へ追従させる制御はサーボ制御と呼ばれる。本章では、タミヤの High Power Gearbox に、角度検出のポテンシオメータを取り付けた実験装置を使ってサーボ制御を行う。

3.2. 自作サーボの回路図

今回使う実験装置の回路図を以下に示す。

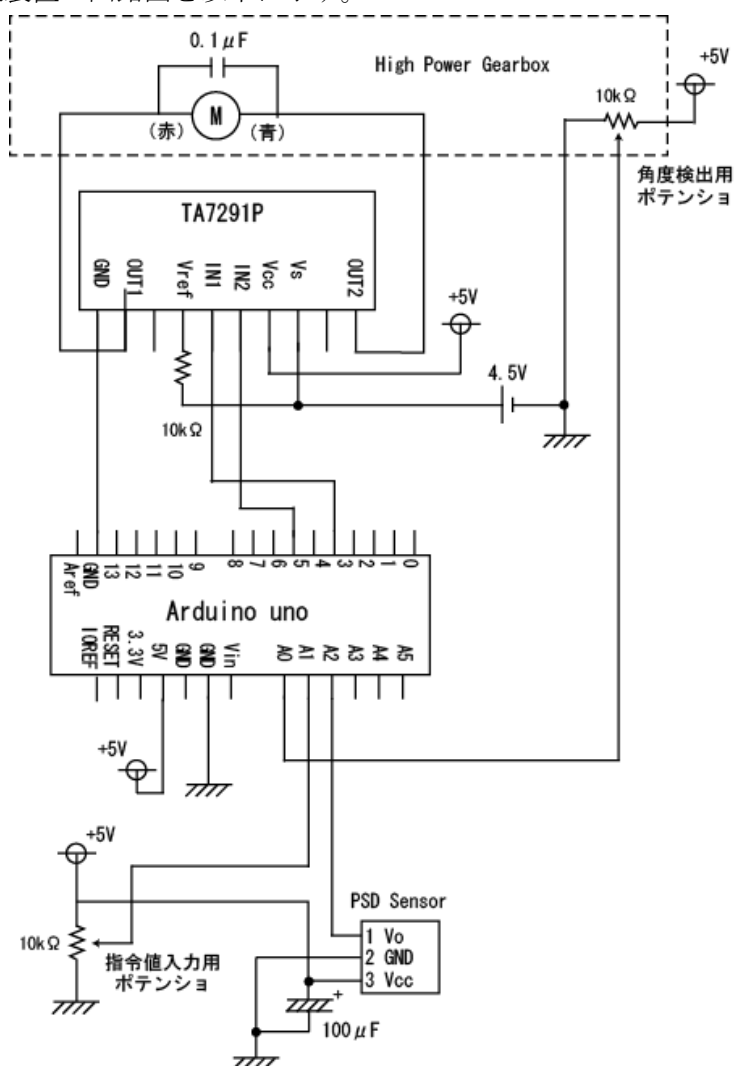


図5 実験装置の回路図

4. 自作サーボモータ実験装置の制御

4.1. はじめに

今回は実験装置の動作についての手順を説明する。

手順としては、

- i. モータのステップ応答実験
- ii. パラメータの導出
- iii. 検証実験
- iv. センサの特性測定
- v. 実験装置の動作

このような操作手順で説明を行う。

なお、これらの動作を開始する前に初期設定ファイルの `sim_init.m` で初期パラメータを設定し実行する必要がある。初期パラメータを変更する必要がある場合は `sim_param.mat` を起動するだけで良い。

4.2. モータのステップ応答実験

次の手順に従い実験を行う。

- i. まず、同定実験のための Simlink モデル `pos_id_step_sl.mdl` を開く。そして Scope を開いておく。この Scope は上段が出力応答、下段が制御入力である。また、simlink モデルの Arduino IO Setup のボックスをダブルクリックして開く。そうすると出力させるポートを選べるようになるので Arduino のポートに合わせる必要がある。
- ii. `pos_id_step.m` を開く。フィードバックゲインは 1 1 行目の `Kp_id` である。まず、デフォルト値で一度実験を行う。予備実験の場合ステップ応答の試行回数 `Ncyc` を少なくして（`Ncyc=2` 程度）で実験するとよい。`pos_id_step.m` を実行するとステップ応答実験が始まる。
- iii. `Kd_id` の値は応答が多少振動的になるように設定する。振動的ではない応答にすると、非線形摩擦の影響で出力応答が途中で止まってしまう。図 7 のような応答が得られれば良い。問題がなければ試行回数を `Ncyc=8` 程度にしてステップ応答実験を行う。
- iv. 実験が終わると、上段に 1 回目を除いた `Ncyc-1` 回分のステップ応答（ただし、正転側のステップ応答のみ）と下段にそれらの平均応答が表示される（図 8）。

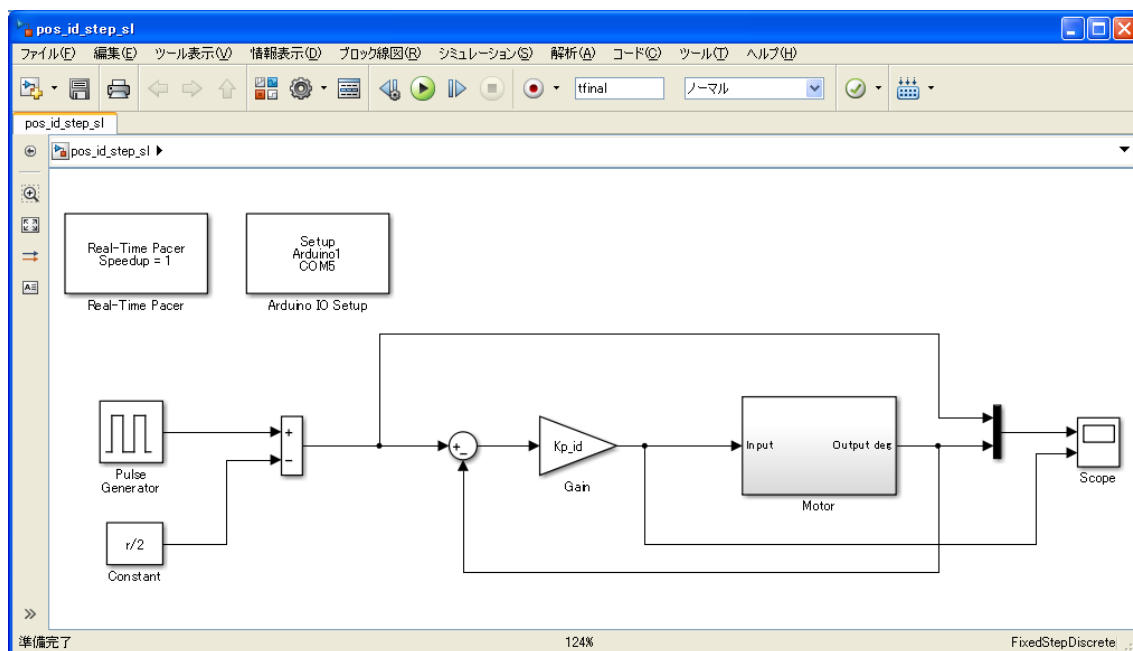


図6 ステップ応答実験のための Simlink モデル

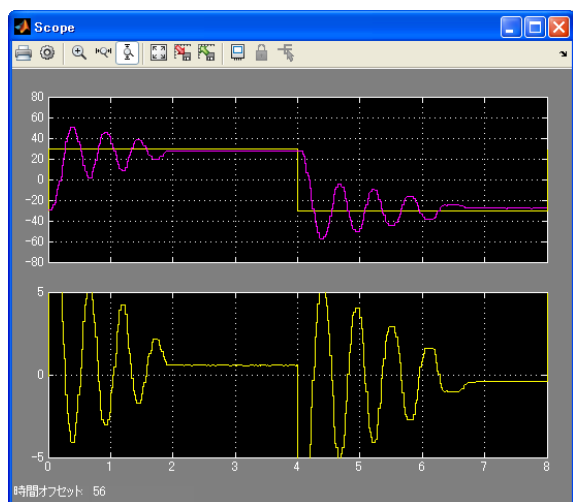


図7 実験中の波形

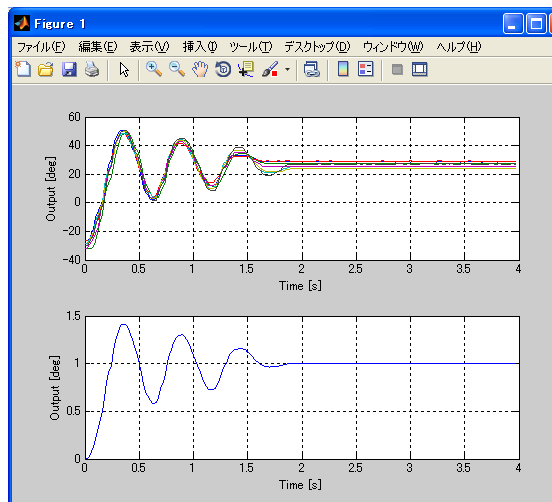


図8 8回の実験データと平均応答

4.3. パラメータの導出

得られたステップ応答から、制御対象のパラメータを求める。しかし本実験装置では、デジタル制御を行っており、サンプリング周期が 20ms で計測しているため制御対象のパラメータを求めると実際の値と少しずれてしまう。そこで離散化の影響も考慮した計算方法を利用して計算する。それらの計算を行うのが `myfunc.m` である。これは直接実行するのではなく `pos_id_step_fit.m` を実行してそこから呼び出すことで計算を行う。

このプログラム(pos_id_step_fit.m)を実行すると、0 秒から何秒の範囲において計算をするか聞いてくる。装置によって摩擦などの影響があるため試行錯誤を繰り返しながら実機データとモデルから計算される応答がよく一致する時間範囲を見つけてほしい。

pos_id_step_fit.m の実行結果の例を図 9 に示す。実機には非線形摩擦だけではなく多くの不確定要素を含んでいるため完全な一致は不可能である。ここではこの図に示す一部の一致が見られればよいものとする。図 9 に対するコマンドウィンドウ出力は以下の通りである。

```
>> pos_id_step_fit
```

```
Time length for fitting = 0.8
```

```
== Results ==
```

```
K = 64.411738
```

```
T = 0.146297
```

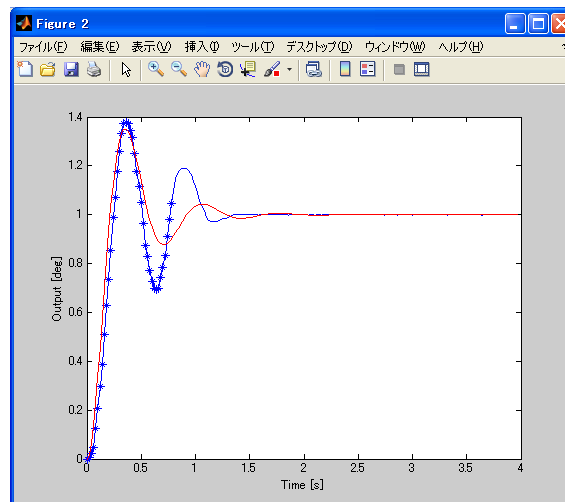


図9 実機の応答との比較の例

4.4. 検証実験

得られたモデルパラメータ T と K の妥当性を検証するため、実機とモデルの両方に同じフィードバック制御を行い両者の出力を比較する。そのため Simlnk モデル pos_pid_mbd_sl.mdl を開く。それでは pos_id_verify.m を開き、前に調べた K と T のパラメータを変更した値に書き換え実行する。実際には摩擦やモータに取り付けた装置の関係で振動は止まる時もあるがモデルの応答は振動を続ける。実機とモデル出力があまりにも一致しない場合には、同定実験をやり直してほしい。ただし、両者は完全に一致することはないので、ある程度の誤差は許容範囲内とする（図 11）。

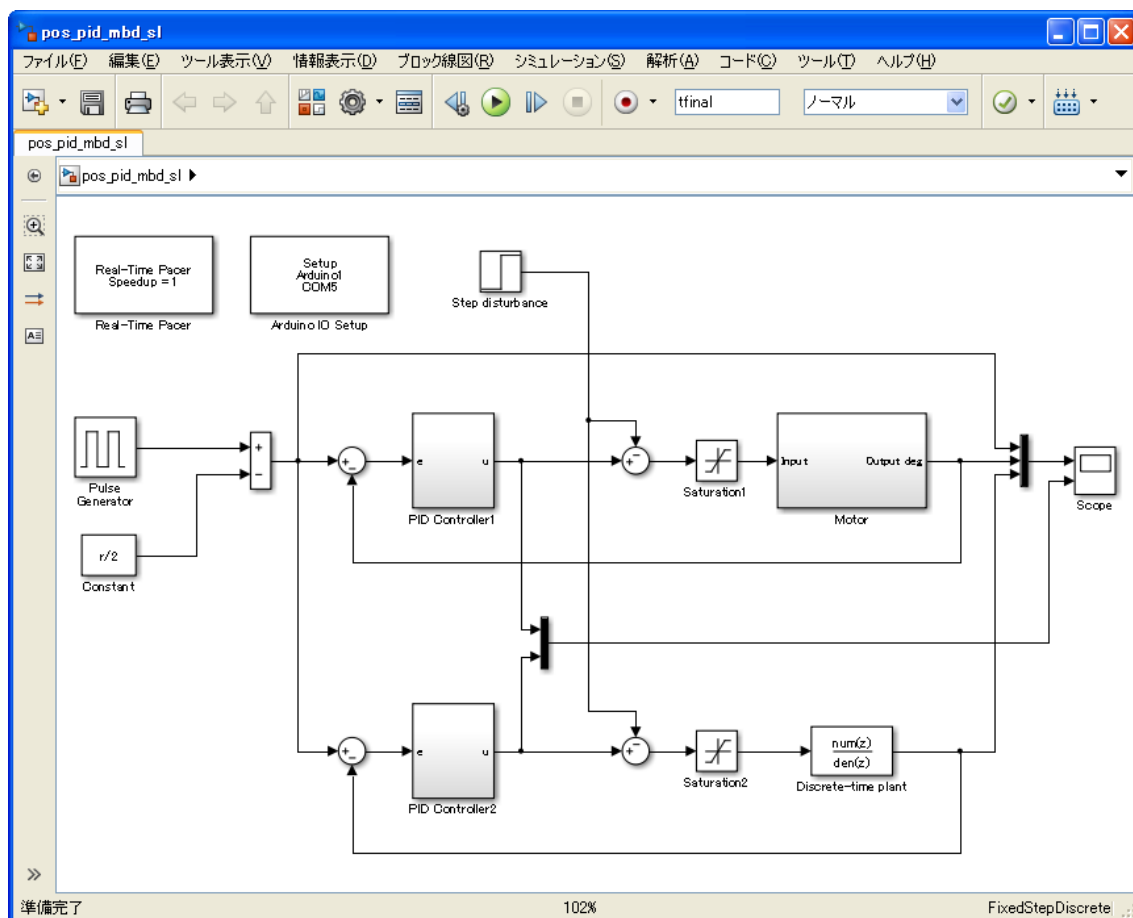


図10 pos_pid_mbd_sl

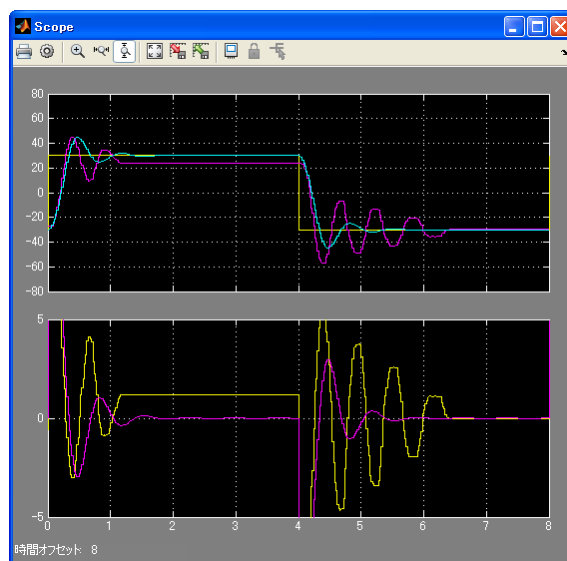


図11 検証実験の出力応答例

4.5. センサの特性測定

本実験装置で使う曲げセンサではある一定の特性を持っている。そこで 2 重構造の先端からモータに固定している部分までの距離と電圧の関係を実測し両者の関係を表す数式を求める。

電圧測定のためにpsd_test.mを実行する。その際、`a = arduino('COM5');`と書いているポートを現在使っているArduinoのポートに変更して実行してほしい。これは電圧を500回測定しその平均値を出すプログラムとなっている(図12)。またこの実測データをpsd_plot_bending.mに距離と電圧の関係をを入れていくことでプロットが可能となる(図13)。曲げセンサの特性が今回直線に近かったため近似直線を計算し、 $y=ax+b$ における係数の a 、 b を出力するようになっている。実行結果は以下のようになる。

```
>> psd_plot_bending
```

```
a = -85.075538
```

```
b = 99.726644
```

これによりセンサの特性を知ることが出来た。

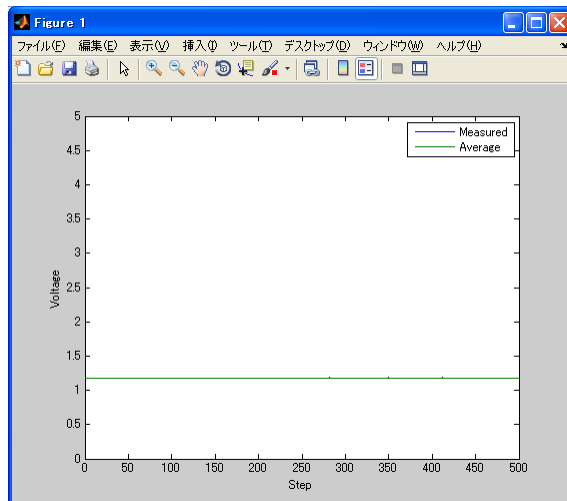


図12 500 回実験

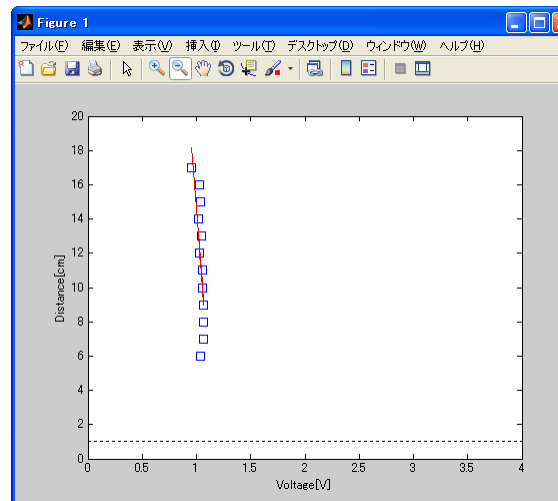


図13 近似直線

4.6. 実験装置の動作

自作サーボモータの制御するための `bb_pid_hg_bending.m` を開く。また波形を確認するための「position」と「Servo angle」を開いておく。各パラメータの定義は `bb_pid_hg_bending.m` で定義する。これを実行すると `bb_pid_hg_sl_bending.slx` が呼び出される(図 14)。K と T の値を同定実験で得られた値に変更する。また、センサの特性測定で得られた値の入力方法も説明する。まず、`bb_pid_hg_sl_bending.slx` の Beam のボックスをダブルクリックする(図 15)。

出てきた係数 a と b に計測した値を入力する。

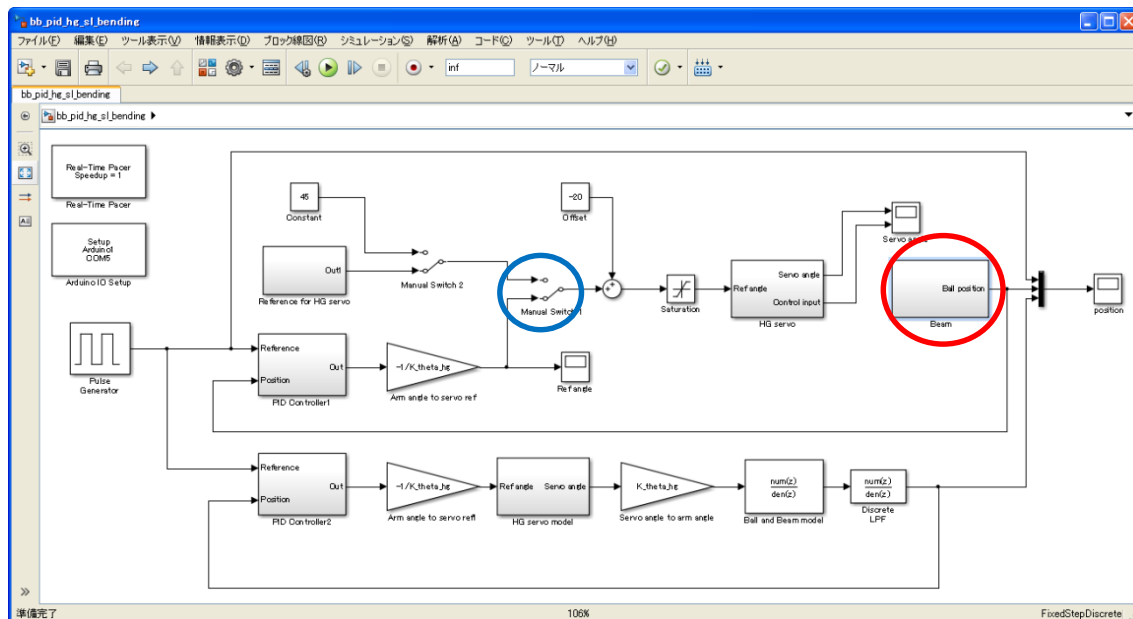


図14 `bb_pid_hg_sl_bending.slx`

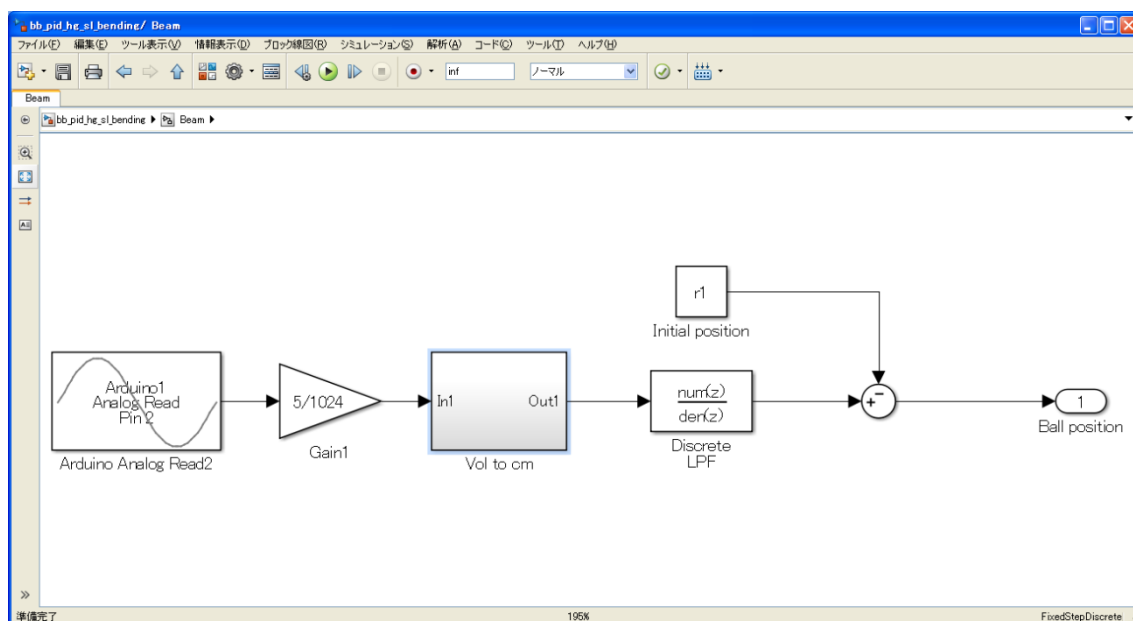


図15 beam

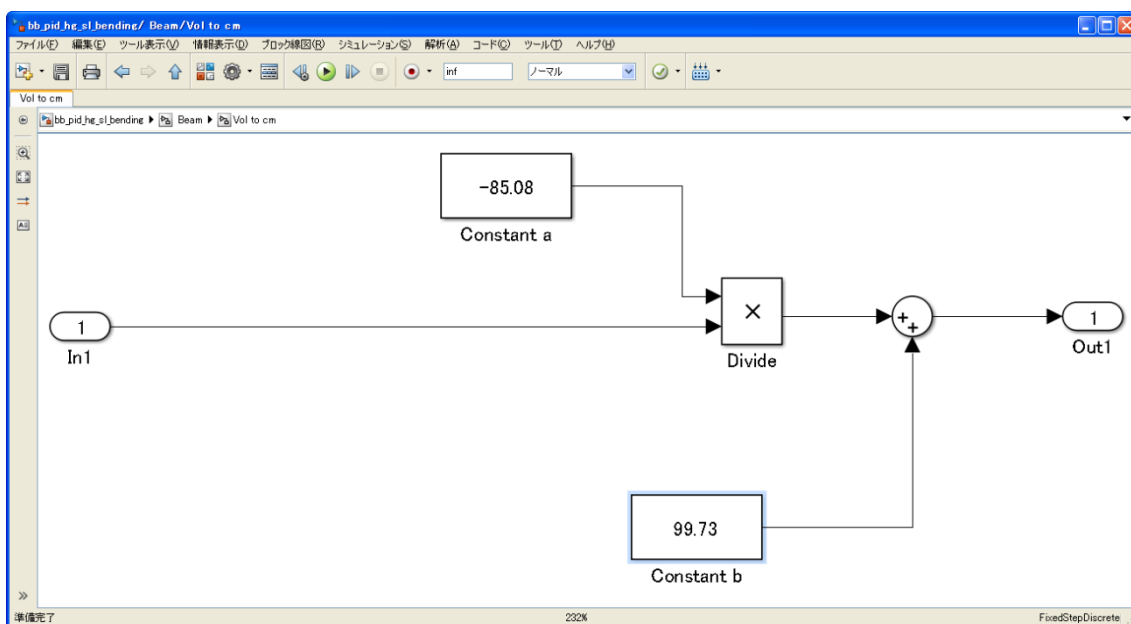


図16 Vol to cm

それらの設定変更を終え `bb_pid_hg_sl_bending.slx` の `Manual Switch1` を下に向けて実行すると実験装置が起動する。

終了時は、`simlink` の停止ボタンを押すことでプログラムを終了する。

5. 終わりに

実験装置の動作確認をもって ArduinoIO マニュアルを終了する。

bb_pid_hg_sl_bending.slx の offset の数値を変更するとモータの基準位置が、bb_pid_hg_bending.m の r1 の値を変更するとセンサの基準値が変わるので装置の挙動が変わってくる。

また、異なるセンサを使用する場合も、センサの特性が分かればフィードバックさせる状態の変更が可能となるため新たな挙動が得られると考えられる。

ここまでは ArduinoIO について解説を行ったが、この機能は PC に接続した状態でないと使用が出来ない。そこで Arduino に C コード生成し計算も行わせる機能である Run on Target Hardware についてのマニュアルも用意しているのでそちらも試してほしい。