Report on the Experiment

No. 2

Subject MATLAB/Simulink による RapidPrototyping

Date 2020. 09. 08

Weather 晴れ Temp 26.8 ℃ Wet 62.3 %

Class E4
Group 2
Chief
Partner 井上 隆治
重見 達也
宮崎 拓也
森 和哉

No 14
Name 小畠 一泰

Kure National College of Technology

1　目的

MATLAB/Simulink の Hardware Support Packages を使うと, programming を行うことなく MATLAB/Simulink から Arduino を動作させることができる. これを用いて rapid prototyping のさわりを体験することを目的とする.

2　実習

2.1　Digital I/O(Output)

図 1 の回路を Arduino UNO と solderless breadboard を用いて作成した.

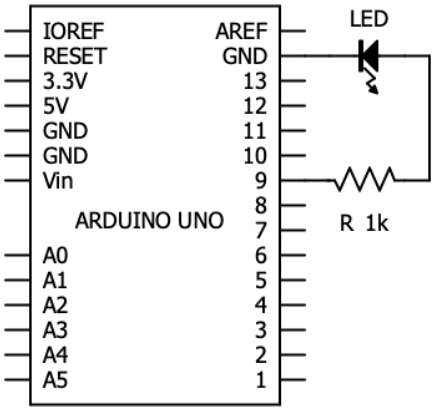

図 1: 回路図

Simulink を起動し, 図 2 のようにモデルを組み, 動作を確認した.

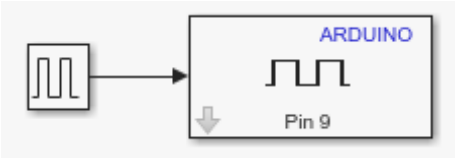

図 2: ブロック線図

次に図 3 のようにモデルを組み変えシミュレーションを実行したところ, 図 4 のようになった. なお Pulse Generator はパルスタイプ: サンプルベース, 周期: 100(サンプル数), パルス幅: 50(サンプル数), サンプル時間: 0.01 に変更した.

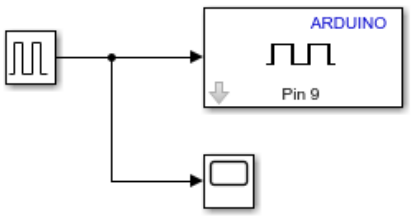

図 3: ブロック線図

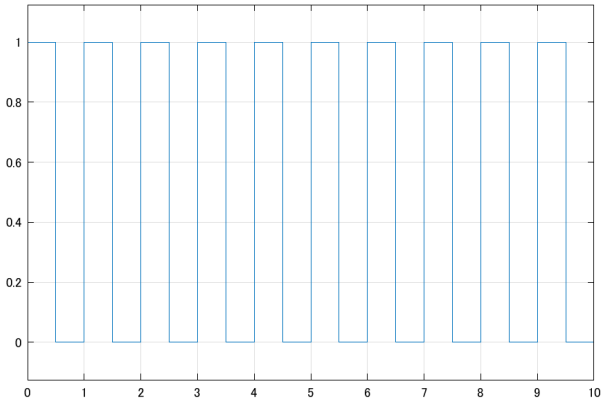

図 4: Scope

2.2 Digital I/O(Input)

図 5 の回路を Arduino UNO と solderless breadboard を用いて作成した.

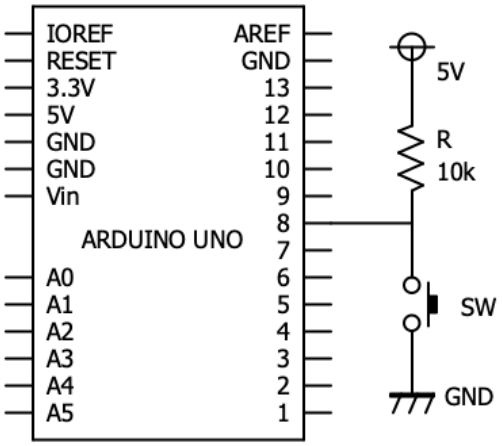

図 5: 回路図

図 6 のようにモデルを組み, シミュレーション中にスイッチを on/off 切り替えることで 図 7 のようになった. なお Digital Input ブロックの Sample Time は 0.02 に設定した.

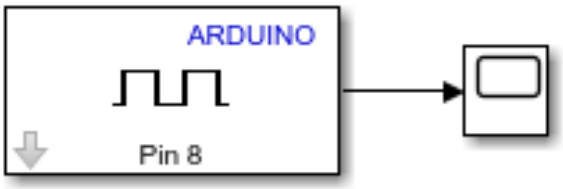

図 6: ブロック線図

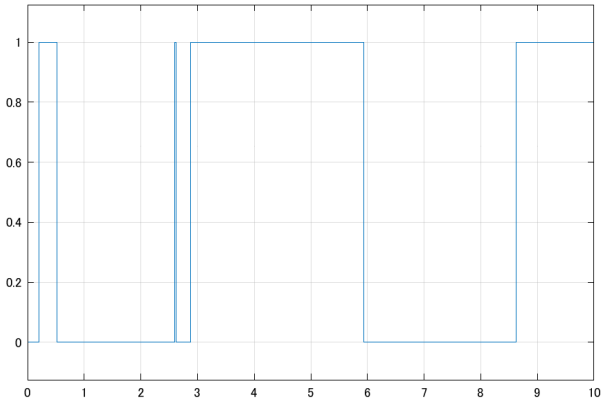

図 7: 実験結果

2.3 Digital I/O(Input/Output)

図 8 の回路を, Arduino UNO と breadboard を用いて作成した.

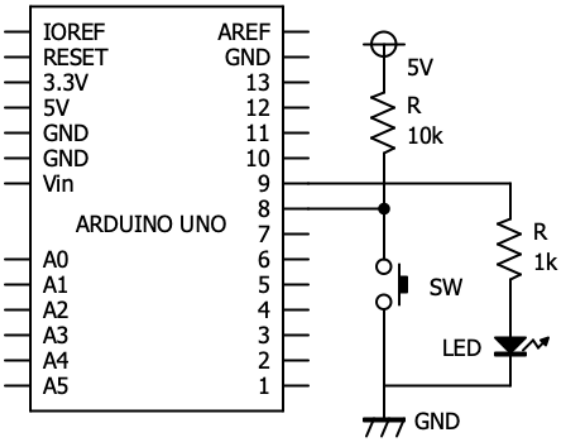

図 8: 回路図

図 9 のようにモデルを組み 2.2 のようにシミュレーションすると, 図 10 のような結果が得られた. なお Digital Input ブロックの Sample Time は 0.02 に設定した.

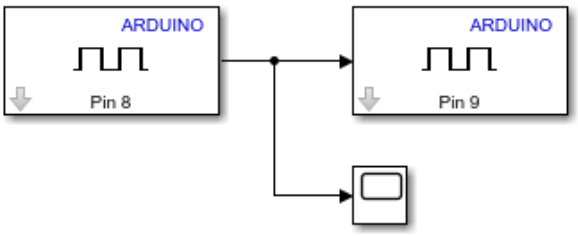

図 9: ブロック線図

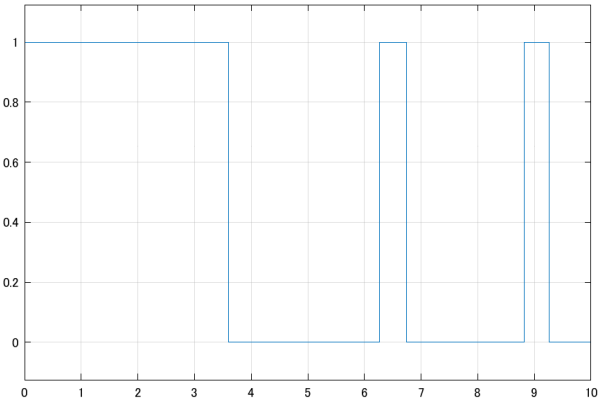

図 10: Scope

2.4 Analog I/O(Input)

図 11 の回路を Arduino UNO と soldedless breadboard を用いて作成した.

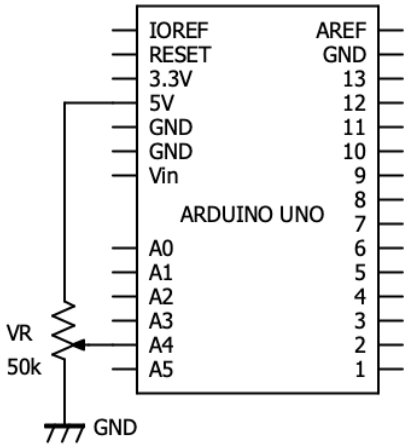

図 11: 回路図

図 12 のようにモデルを作成し, Analog Input ブロックの Sample Time を 0.02 ぐらいにすると, 図 13 のような結果が得られた.

図 12: ブロック線図

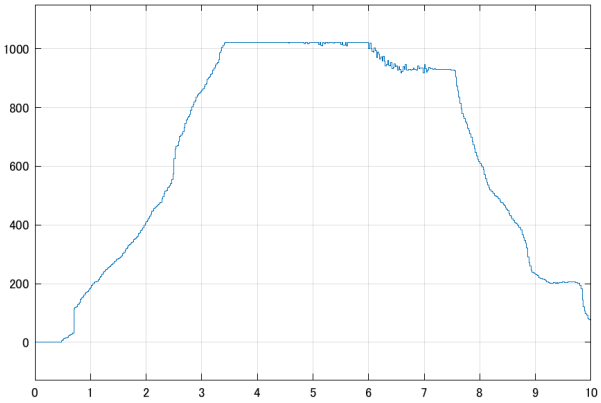

図 13: Scope

2.5 Analog I/O(Output)

図 1 の回路を利用しモデルを図 14 のようにすると, 図 15 のようになった. なお Sine Wave は正弦波タイプ: 時間ベース, 振幅: 1, バイアス: 0, 周波数: 1(rad/sec) とした. 負の値を取らないように, バイアスをかけると図 16 のようになった.

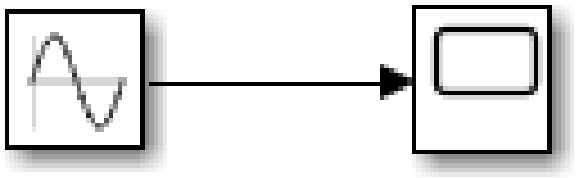

図 14: ブロック線図

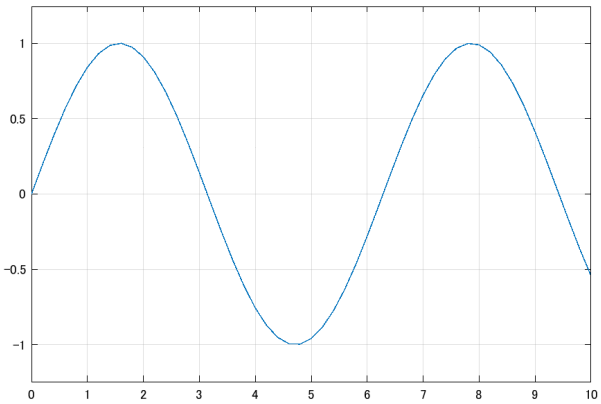

図 15: Scope

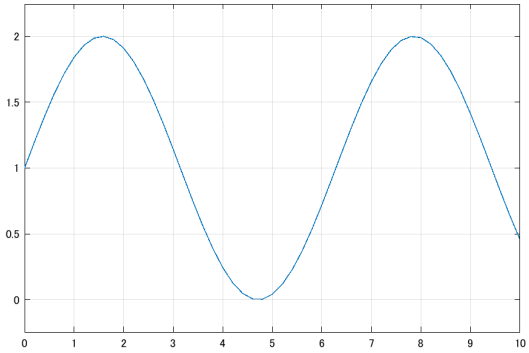

図 16: Scope(1V のバイアスあり)

Arduino の PWM 出力は, 0〜255 の入力値を 0~100 %のデューティ比に変換して出力するため,Gain ブロックを図 17 のように追加すると, 図 18 のようになった.

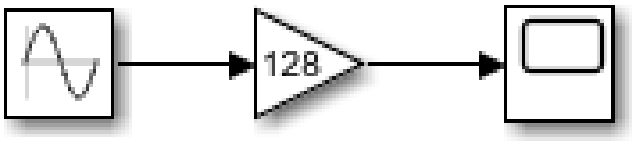

図 17: ブロック線図

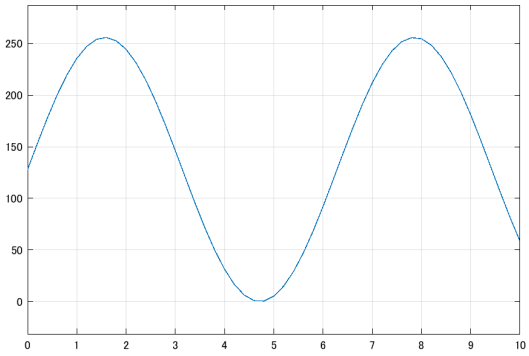

図 18: Scope(128 倍)

図 19 のようにモデルを組み替え, デプロイすると LED を任意の明るさに変えることができた.

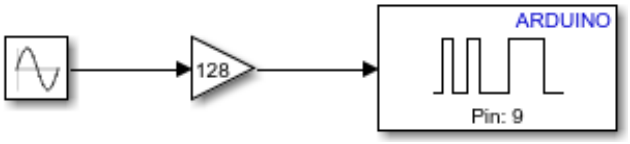

図 19: ブロック線図

2.6 Analog I/O(Input/Output)

図 20 の回路図を Arduino UNO と solderless breadboard を用いて作成した.

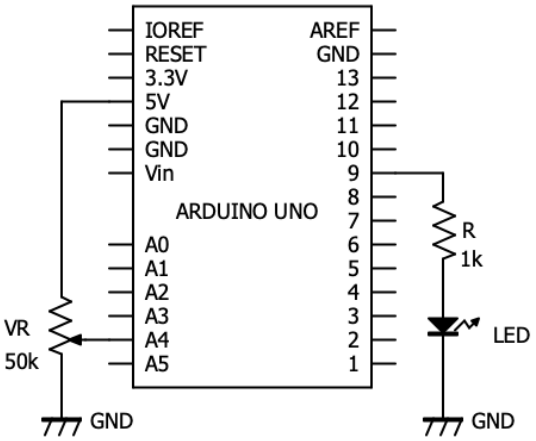

図 20: 回路図

モデルを図 21 のように作成し, デプロイして可変抵抗の抵抗値を変化させると LED の明暗が変わるように変化させた.

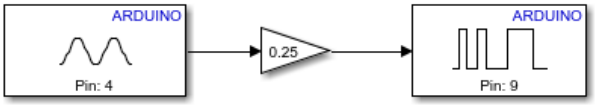

図 21: ブロック線図

2.7 RC servo

RC servo は, 角度指令を与えるだけで軸の角度制御を行なうことができる. 図 22 を Arduino UNO と solderless breadboard, Tower Pro を用いて作成した.

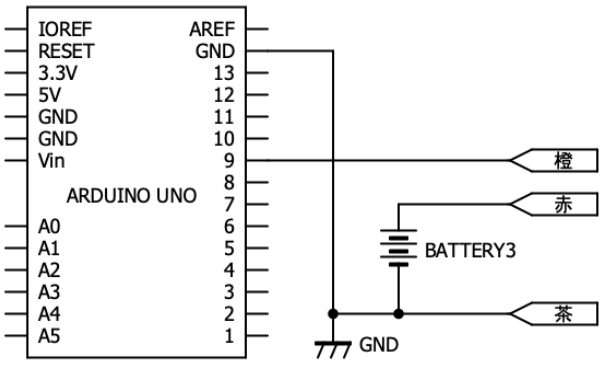

図 22: servo 回路

図 23 のようなモデルを作成した．このとき出力に設定されている Standard Servo Write は，0~180 までの値を入力すると，軸の角度がその値に追随する．

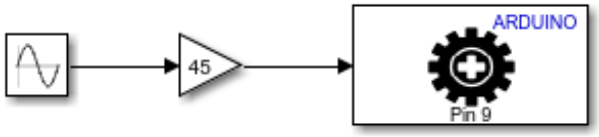

図 23: ブロック線図

3 課題

3.1 アナログ温度センサ LM61BIZ の読み込み

National Semiconductor 製の単一製電源動作温度センサ IC, LM61 を使用して温度を測定するために，図 24 の回路を作成した．

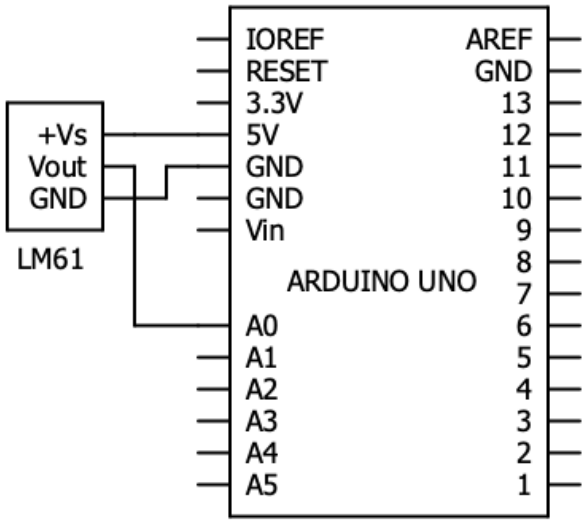

図 24: LM61 を用いた検温計

温度センサの出力電圧 V_{out} と摂氏温度 $T°C$ の関係は次式のようになる．

$$V_{out} = (+10mV/°C \times T°C) + 600mV$$

この IC は $1°C$ ごとに電圧が $10mV$ 上昇し，内部で $600mV$ のオフセットを持っている．

また，Arduino UNO で A/D 変換の結果は次式で表される．

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

$V_{IN} = V_{OUT}$, $V_{REF} = V_{CC} = 5000mV$ とすると摂氏温度 T は次式によって求められる．

$$T = \frac{\frac{5000}{1024} \times ADS - 600}{10} \ [\mathrm{°C}]$$

ここで $\frac{5000}{1024}$ は, $V_{CC} = 5000mV$ を 10 ビットで A/D 変換したときの 1LSB の電圧値を表している.

よって図 25 のようなモデルで表すことができる.

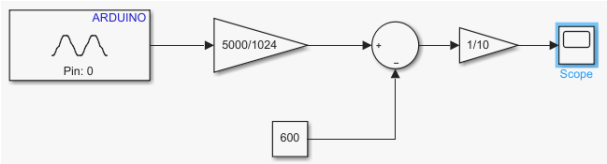

図 25: ブロック線図

以上のように回路を組み実行した結果, Scope は図 26 に出力された.

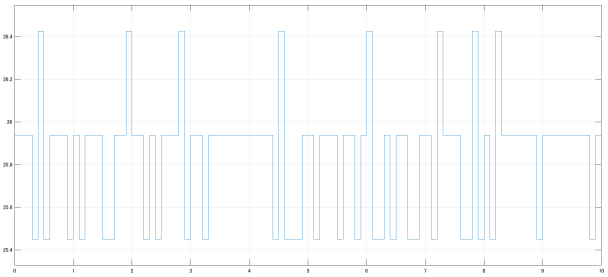

図 26: Scope

3.2　温度通知システムの作成

27°C 以上になると LED が点灯して知らせるシステムを作成せよ.

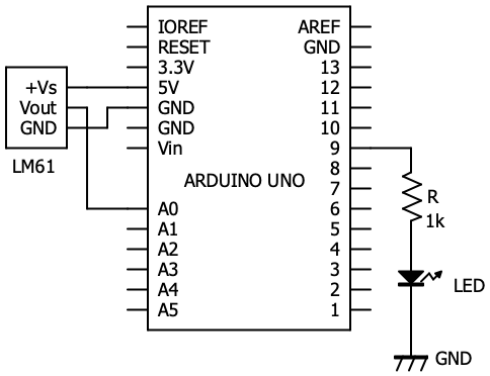

図 27: 回路図

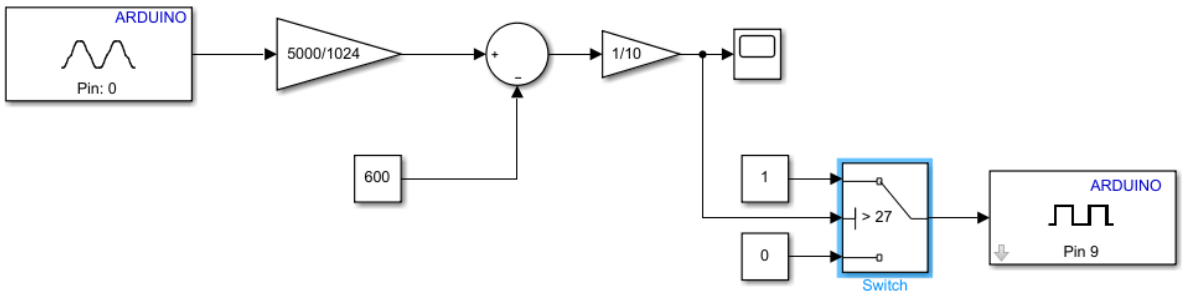

図 28: ブロック線図

図 27 のような回路を考え, 図 28 のようなモデルを作成した. これは閾値 (本実験では実験室の室温を考慮し 27°C とした.) を越えると 1 を, 下回る場合は 0 を出力する Swtich を利用し, それを Degital Output に渡すことで対応した. 実際に途中から手で IC を温めると図 29 のようになり, 閾値を超えた際に LED が点灯した.

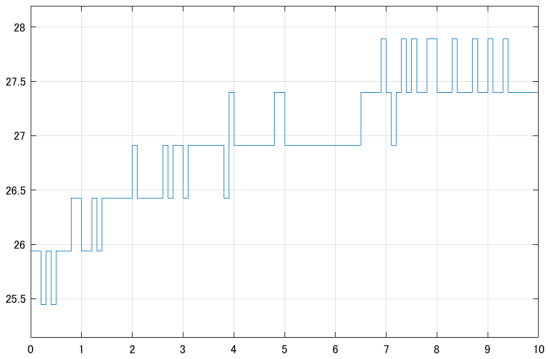

図 29: Scope

4 検討課題

1. Pulse Generator のパルスタイプ, 振幅, 周期, パルス幅, サンプル時間は, シミュレーション実行とデプロイ実行それぞれについて, どのような意味をもつと考えられるか.
   - パルスタイプ: 発生する矩形波のタイプを時間ベースかサンプルベースのどちらかで生成する計算手法. 時間ベースモードとサンプルベースモードの重要な違いは, 時間ベースモードではブロック出力がシミュレーション時間にもとづくのに対し, サンプルベースモードではブロック出力がシミュレーションの経過時間に関係なくシミュレーションの開始にのみ依存する.
   - 振幅: 信号の振幅を指定します.
   - 周期: パルスタイプが時間ベースの場合は秒単位で指定されたパルス周期. パルスタイプがサンプルベースの場合, 周期はサンプル時間の数として指定される.
   - パルス幅: 時間ベースの場合は信号がオンになっているパルス周期の割合 (%) として指定したデューティ比, サンプルベースの場合はサンプル時間の数として指定したデューティ比.
   - サンプル時間: このブロックのサンプル時間の長さ.

   またシミュレーション実行とデプロイ実行について大きな違いは見られなかった.
2. シミュレーション実行とデプロイ実行はそれぞれ, どのような動作を行っていると考えられるか.
   - シミュレーション実行: PC がプログラムを実行する.
   - デプロイ実行: コンパイルされたプログラムが Arduino に書き込まれプログラムを実行するためスタンドアローン化される. また, Simulink の入った PC は信号を送受信するだけ.

   したがって, シミュレーション実行は一度しか実行しないが, デプロイ実行は通常の Arduino 同様電源供給がある限り実行され続ける.
3. Sample Time が小さいときと大きいときでどのような違いがあるか. シミュレーション実行とデプロイ実行で試してみよ.
   - シミュレーション実行: Sample Time が小さいと実行時間が短くなり, 大きいと長くなった.
   - デプロイ実行: Sample Time が小さいと LED の明暗の差が細かく滑らかであり, 逆に大きいと激しく飛び飛びになった.

5 参考文献

- 平田光男, Arduino と MATLAB で制御系設計をはじめよう, TechShare(2012)
- 一定の間隔で矩形波パルスを生成 - Simulink - MathWorks 日本 (https://jp.mathworks.com/help/simulink/slref/pulsegenerator.html)