令和元年度　環境ロボティクス学科　卒業論文

ワイヤレス給電システムの最適化のための能率的な動作周波数スイープ

令和 2 年 (2020) 2 月 14 日

森田　光流

指導教員：穂高一条教授

目 次

1 緒言

1.1 背景

1.1.1 ワイヤレス給電の概要

ワイヤレス給電とは，コネクタや金属接点の接触を用いず無線で電力を供給，伝搬することが可能な給電方法である．非接触充電，非接触電力送電，無線給電などとも呼ばれている．従来の電気製品の給電方法の金属接点やコネクタを使用したものは，水がかかると水による感電やショートを起こす，配線による転倒などの安全面に関して問題点がある．しかしワイヤレス給電は金属接点が非接触なので前述に述べた安全面の問題点の解決するだけではなく，人が立ち入れない危険な場所にある機器や装置の遠隔操作ができるなどという利点が期待されており，現在盛んに研究されている．

1.1.2 ワイヤレス給電の特徴

ワイヤレス給電が無線で電力を供給するためには，送電側と受電側の 2 つのコイルを用いるため物理的な金属接点やコネクタが存在せず，また非金属のものであれば，コイル間に存在していても送電側と受電側の電力に影響を及ぼさない．また，送電コイルを表面に露出させる必要がないため，水による感電やショートする心配がないので，水場での使用が可能である．なお，ワイヤレス給電の給電方式には送電側と受電側との間発生する誘導磁束を利用した電磁誘導方式や，キャパシタを送電電極と受電電極として電力を送電する電解結合方式などがある [4][9] が，本研究ではその中の電磁共鳴方式で行う．

1.1.3 ワイヤレス給電の将来性

現在，ワイヤレス給電の特徴を生かして様々な場所に導入しようと 研究が行われている．前述に述べた通り金属接点が非接触なので，ワイヤレス給電化することにより物理的な金属接点が少なくなるので電源コードは最小限で済み余分な空間を作ることにより直接配線などの問題解決並びに自動化・効率化に大きく貢献している [10][1].

また，地球温暖化の影響により CO_2 の削減のため電気自動車開発並びに普及が望まれている．現在，電気自動車に使用されている接触式充電方式は雨天時にプラグを扱うと感電する可能性がある．それをワイヤレス給電に置き換えると自宅の駐車場に駐車するだけで充電が可能になり電の可能性が払拭される，充電の際に運転者が降車する必要がなくなり運転者の負担が大きく軽減されることが期待される．[7]

1.2 研究目的

ワイヤレス給電の電力供給の効率の改善には,
1. 最適な周波数調整
2. 結合操作
3. マルチループコイルと LC 回路を用いた適応マッチング
などがある [8]．本研究においては 1. に注目し，そのために最適動作周波数の探索をすることとした．1. を最適動作周波数を探索するには，受電電力と送電電力を使った電力効率を求める必要がある．しかし，電力効率が高いだけでは最適動作周波数が適切であるとはいえない．受電側と送電側の元々の電力が小さい場合も電力効率が高いということが考えられる．それではワイヤレス給電が最適に動作しているとは考えられない．したがって，最適動作周波数は高効率かつ大電力が出力される周波数と定義される．

また最適動作周波数を探索するためには多くの周波数を測定しなければいけないので測定時間をなるべく短くしたい．しかし，ワイヤレス給電に周波数を送った後，安定な電力が出力されるまで時間がかかる．したがって本研究は能率的な周波数スイープ設計し, ワイヤレス給電システムの最適動作周波数を能率的な周波数スイープによる測定，考察を目的にした．

2　本論文の構成

緒言

本研究の背景および，研究目的をこの章で述べる．

本研究での基礎理論と原理

ワイヤレス給電についての原理について，実験で扱う基礎理論についての説明をこの章で述べる．なお，実験で使われる素子についての説明もする．

3 本研究での基礎理論と実験原理

3.1 ワイヤレス給電の原理

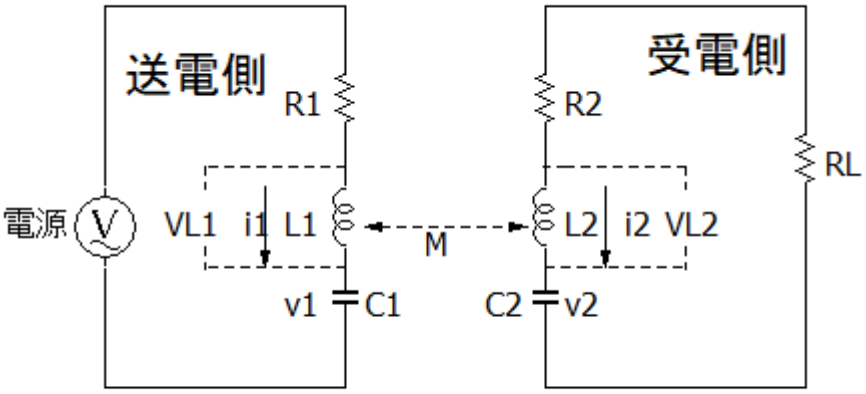

図 1: ワイヤレス給電回路図

本研究で扱うワイヤレス給電の回路は図 1 で示される．図 1 に示すように，ワイヤレス給電は送電側と受電側が電気的につながっていないことが分かる．送電側の交流電源の電流が時間的に変化することにより受電側の磁束が変化しファラデーの電磁誘導の法則より受電側に誘導起電力が生じ誘導電流を伝えるという原理である．また，図 1 の回路素子は以下の通りである．

- L1:送電側のコイルの自己インダクタンス．
- L2:受電側のコイルの自己インダクタンス．
- M:送電側と受電側のコイルの相互インダクタンス．
- R1:送電側コイルの内部抵抗
- R2:受電側コイルの内部抵抗
- C1:送電側のコンデンサの電気容量
- C2:受電側のコンデンサの電気容量
- RL:受電側の負荷抵抗
- 電源:送電側回路の交流電源

3.2 平均電力・効率

ワイヤレス給電の最適動作周波数を探索するためには平均電力と効率を求める必要があるが，平均電力を求めるにはワイヤレス給電の定常解を求める必要がある．

3.2.1 定常解の導出

状態方程式は，一般的に

$$\dot{x} = Ax + Bu \tag{1}$$

と表せられる．(ワイヤレス給電の状態方程式の求め方は後述の付録参照)

この式 (15) を解くと，

$$e^{-At}\dot{x}(t) - e^{-At}Ax(t) = e^{-At}Bu(t) \tag{2}$$

記号の都合により，t をτ にして求めると，

$$\frac{d}{d\tau}(e^{-A\tau}x(\tau)) = e^{-A\tau}Bu(\tau) \tag{3}$$

$$\left[e^{-A\tau}x(\tau)\right]_0^t = \int_0^t e^{-A\tau}Bu(\tau)d\tau \tag{4}$$

$$e^{-At}x(t) - x(0) = \int_0^t e^{-A\tau}Bu(\tau)d\tau$$

$$e^{-At}x(t) = x(0) + \int_0^t e^{-A\tau}Bu(\tau)d\tau \tag{5}$$

両辺に e^{At}をかけると，

$$x(t) = e^{-At}x(0) + e^{-At}\int_0^t e^{-A\tau}Bu(\tau)d\tau \tag{6}$$

となり，式 (6) は状態方程式の一般解である．平均電力と効率は状態方程式の周期的な解に基づくため，式を周期的な関数にしなければならない．したがって $x(t)$ が周期 T をもつ関数とすると，

$$x(t+T) = x(t) \tag{7}$$

となり，式 (7) に式 (6) を代入して整理すると，

$$x(0) = (I - e^{AT})^{-1}e^{A(T-t)}\int_0^T e^{-A\tau}Bu(t+\tau)d\tau - \int_0^t e^{-A\tau}Bu(\tau)d\tau \tag{8}$$

で求まる (但し，$det(1 - e^{AT} \neq 0)$ の時だけ)．式 (8) を式 (6) に代入し，周期解 $x_{ss}(t)$ を求めると，

$$x_{ss}(t) = e^{At}(I - e^{AT})^{-1}e^{A(T-t)} \quad \int_0^T e^{-A\tau}Bu(t+\tau)d\tau \quad - \int_0^t e^{-A\tau}Bu(\tau)d\tau + \int_0^t e^{-A\tau}Bu(\tau)d\tau$$

$$x_{ss}(t) = (I - e^{AT})^{-1}e^{At} \quad \int_0^T e^{-A\tau}Bu(t+\tau)d\tau \tag{9}$$

よって，周期 T 定常解は式 9 で求められる．

3.2.2 短形波入力の場合の定常解

短形波入力の定義は,

$$\begin{aligned} u(t) &= \begin{cases} u_0 & (0 \leqq t < Td) \\ 0 & (Td \leqq t < T) \end{cases} \\ \text{かつ} \quad u(t) &= u(t+T) \end{aligned} \tag{10}$$

であらわされる.T は 1 周期であり d はデューティー比である．そこから式 (9) に式 (13) を代入し範囲ごとに場合分けで積分し，$x_{sst}(t)$ を求める.

i. $0 \leqq t < Td$ の場合

$$\begin{aligned} \int_0^T e^{-Ap}But + pdp &= \int_0^{Td-t} e^{-Ap}Bu(t+p)dp + \int_{Td-t}^{T-t} e^{-Ap}Bu(t+p)dp + \int_{T-t}^{T} e^{-Ap}But + pdp \\ &= \int_0^{Td-t} e^{-Ap}Bu_0 dp + \int_{T-t}^{T} e^{-Ap}Bu_0 dp \\ &= -A^{-1}e^{-AT}(e^{A(t+T(1-d))} - e^{-At} - e^{AT} + I)Bu_0 \end{aligned} \tag{11}$$

ii. $Td \leqq t < Td$ の場合

$$\begin{aligned} \int_0^T e^{-Ap}Bu(t+p)dp &= \int_0^{T-t} e^{-Ap}Bu(t+p)dp + \int_{T-t}^{t+dt-t} e^{-Ap}Bu(t+p)dp + \int_{T+dt-t}^{T} e^{-Ap}Bu(t+p)dp \\ &= \int_{T-t}^{T+dt-t} e^{-Ap}Bu_0 dp \\ &= -A^{-1}e^{-AT}(e^{A(t-td)} - e^{At})Bu_0 \end{aligned} \tag{12}$$

となる．よって，式 (11) と式 (12) から $x_{sst}(t)$ は,

$$x_{sst}(t) = \begin{cases} -(I - e^{AT})^{-1}A^{-1}(e^{A(t+T(1-d))} - e^{At} - e^{AT} + I)Bu_o & (0 \leqq t < Td) \\ -(I - e^{A}T)^{)}(-1))(e^{A(t-Td)} - e^{At})Bu_0 & (Td \leqq t < T) \end{cases} \tag{13}$$

となる.

3.2.3 平均電力と電力効率の導出

ある時刻においての電力を瞬時電力というが，平均電力は時間範囲内でその瞬時電力を時刻ごと加算して平均したものである．ここでは計算での平均電力と電力効率を求める方法を述べる

ある時刻においての瞬間電力 p_t, 電圧 v，電流 i とおくと

$$p_t = vi \tag{14}$$

と表すことができる．瞬間電力が周期 T をもつ場合，平均電力は 0 から T まで積分し，そして T で割ることにより求められる．よって，

$$P = \frac{1}{T}\int_0^T p_t dt \tag{15}$$

となる．また，送電側と受電側の瞬時電力は次のように表される．

$$p_{t1} = ui_1 \tag{16}$$

$$p_{t2} = R_L i_2^2 \tag{17}$$

一次側と二次側の平均電力は式 (16) と式 (17) より

$$P_1 = \frac{1}{T}\int_0^T u(t)i_1(t)dt \tag{18}$$

$$P_2 = \frac{1}{T}\int_0^T R_L i_2^2 dt \tag{19}$$

と導くことができる．
このように正弦波のような連続の関数を持つ周期関数はそのまま式 (18) と式 (19) に代入することが可能だが，短形波の平均電力は連続ではないので，注意して求める必要がある．したがって短形波の平均電力を求めるには，式 (15）の状態方程式と式 (13) の短形波の定常解を使って解くと，

$$P_1 = \frac{1}{T}\int_0^{Td} ui_{1ss1}(t)dt \tag{20}$$

$$P_2 = \frac{1}{T}R_L\left(\int_0^{Td} i_{2ss1}^2 dt + \int_{Td}^T i_{2ss2}^2 dt\right) \tag{21}$$

となる．また i_{1ss1}, i_{2ss1} は $0 \leqq t < Td$ のときの i_1, i_2 で i_{2ss2} は $Td \leqq t < T$ のときの i_2 である．また，電力効率は 求めた P_1, P_2 より

$$\eta = \frac{P_2}{P_1} \tag{22}$$

と求まる．これにより，送電側と受電側の平均電力と効率を求められる．

3.3 実験における平均電力の求め方

前の小節にも述べた通り，最適動作周波数を求めるには送電側と受電側の平均電力と効率を求めなければならない．平均電力を求めるためには積分を使わなければいけない．そこで，以下の方法が挙げられる．

3.3.1 出力方法

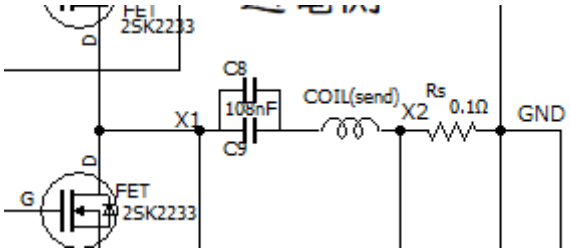

図 2: 送電側回路測定部分

図 (2) に示されているものは，送電側の回路図上にシャント抵抗 R_s を加えたものである．コイルに通る電流を i，シャント抵抗の抵抗は $R_s = 0.1\Omega$ なので，シャント抵抗間の電圧は，

$$v_s = 0.1i \tag{23}$$

となる．

電力を求めるにはシャント抵抗の電圧とコイル間電圧 V_L を乗算器でかけることにより求めることができる．実験で使用した乗算器は乗算した値が 1/10 倍出力される特性を持つから，オペアンプでシャント抵抗の電圧を 100 倍にすることが求められる．v_s を 100 倍された値は，

$$0.1i \times 100 = 10i \tag{24}$$

である．

4 実験内容

5 実験結果

6 考察

7 結論

8 今後の展望

9 謝辞

本研究の進行や本論文等の執筆にあたり，ご指導いただいた穂高一条教授に感謝の意を示すとともに深く御礼申し上げます．また本研究を進めるにあたり多大な助言・アドバイスをしてくださった自動制御研究室の先輩である白銀聡一郎さん，大畑貴弘さん，橋本菜生さん，松浦健斗さん，並びに共に研究した同期のメンバーも感謝の意を示すとともに深く御礼申し上げます．最後になりましたが，お世話になりました宮崎大学工学部環境ロボティクス学科の先生方，並びに大学関係各位の皆様に心より感謝し，ここに御礼申し上げます．

参考文献

[1] 松田一志 :”ワイヤレス給電システムのための電力測定回路の開発”，宮崎大学学士論文，平成 30 年度

[2] 中村裕馬 :”ワイヤレス給電のための送電側 100kHz プッシュプル回路”，宮崎大学学士論文，平成 30 年度

[3] ローム株式会社： ”ワイヤレス給電とは”ーエレクトロニクス豆知識, https://www.rohm.co.jp/electronics-basics/wireless-charging/wireless-charging_what1，最終アクセス：2020/1/20

[4] ローム株式会社:”ワイヤレス給電 (無線給電) 方式”ーエレクトロニクス豆知識, https://www.rohm.co.jp/electronics-basics/wireless-charging/wireless-charging_what2

[5] keicode.com-技術入門シリーズ:”Tkinter による GUI プログラミング”ー Python 入門,https://python.keicode.com/advanced/tkinter.php, 最終アクセス：2020/1/21

[6] 五位塚潤：”低周波数域駆動によるワイヤレス給電回路”， 宮崎大学学士論文, 平成 29 年度

[7] 白銀聡一郎：”ワイヤレス給電のための短形波電源装置の設計と開発”，宮崎大学学士論文，平成 29 年度

[8] 盛田穣文 : ”ワイヤレス給電システムの効率と電力の最適化について”, 宮崎大学修士論文， 平成 29 年度

[9] 伊東雅浩 : ”短径波入力による高効率ワイヤレス給電の制御について”, 宮崎大学修士論文， 平成 30 年度

[10] B&PLUS:”ワイヤレス給電の現状と未来”，https://www.b-plus-kk.jp/about/mechanism.html 最終アクセス：2020/1/23

10 付録

10.1 ワイヤレス給電の状態方程式導出

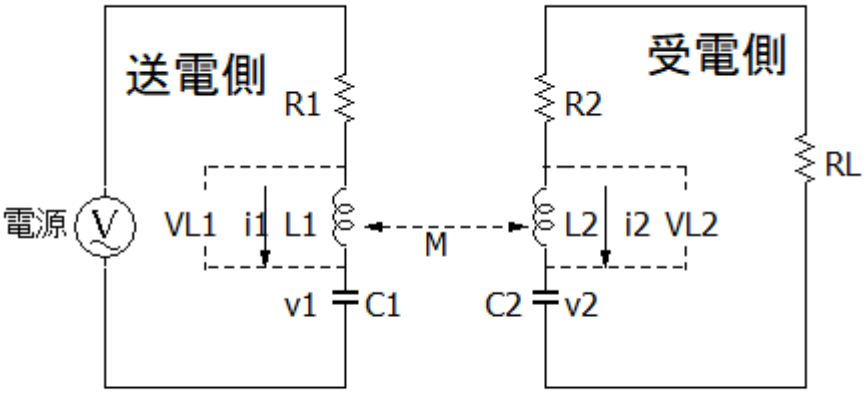

図 3: ワイヤレス給電回路図

図 (3) の定数と電源の値を u として与えると，オームの法則とキルヒホッフの法則より，

$$u = R_1 i_1 + v_1 + v_2 \tag{1}$$

$$(R_{L1} + R_2) i_2 + v_1 + v_2 = 0 \tag{2}$$

コンデンサーの電圧と電流の関係より，

$$C_1 \dot{v_1} = i_1 \tag{3}$$

$$C_2 \dot{v_2} = i_2 \tag{4}$$

V_{L_1} と V_{L_2} を相互インダクタンス M を含めた形にすると

$$V_{L1} = L_1 \dot{i_1} + M \dot{i_2} \tag{5}$$

$$V_{L2} = M \dot{i_1} + L_2 \dot{i_2} \tag{6}$$

式 (3) と式 (4) を変形すると，

$$\dot{v_1} = \frac{i_1}{C_1} \tag{7}$$

$$\dot{v_2} = \frac{i_2}{C_2} \tag{8}$$

それぞれ式 (1) に式 (5) を，式 (2) に式 (6) を代入して $\dot{i}$ で揃えると

$$\dot{i_1} = \frac{Mv_2 - L_2v_1 - R_1L_2i_1 + (R_L + R_2)Mi_2 + L_2u}{\Delta} \tag{9}$$

$$\dot{i_2} = \frac{Mv_1 - L_1v_2 + MR_1i_1 - (R_L + R_2)L_1i_2 + Mu}{\Delta} \tag{10}$$

但しΔは

$$\Delta = L_1L_2 - M^2$$

である．式 (9) と式 (10) を行列式でまとめると，

$$\begin{bmatrix} \dot{v_1} \\ \dot{v_2} \\ \dot{i_1} \\ \dot{i_2} \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} 0 & 0 & \frac{\Delta}{C_1} & 0 \\ 0 & 0 & 0 & \frac{\Delta}{C_2} \\ -L_2 & M & -R_1L_2 & (R_L + R_2)M \\ M & -L_1 & R_1 & -(R_L + R_2)L_1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ i_1 \\ i_2 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} 0 \\ 0 \\ L_2 \\ -M \end{bmatrix} u \tag{11}$$

となる．この行列は以下のように A, B と状態変数 x として表すと，

$$A = \frac{1}{\Delta} \begin{bmatrix} 0 & 0 & \frac{\Delta}{C_1} & 0 \\ 0 & 0 & 0 & \frac{\Delta}{C_2} \\ -L_2 & M & -R_1L_2 & (R_L + R_2)M \\ M & -L_1 & R_1 & -(R_L + R_2)L_1 \end{bmatrix} \tag{12}$$

$$B = \frac{1}{\Delta} \begin{bmatrix} 0 \\ 0 \\ L_2 \\ -M \end{bmatrix} \tag{13}$$

$$x = \begin{bmatrix} v_1 \\ v_2 \\ i_1 \\ i_2 \end{bmatrix} \tag{14}$$

$$\dot{x} = Ax + Bu \tag{15}$$

となり式 (15) はワイヤレス給電の状態方程式となる．

10.2 プログラムについて

前章に示された通り先行研究では周波数を変更する方法で，わざわざ Arduino のプログラムの一部を変更して再コンパイルさせ出力結果をシリアルモニターに表示させる方法であった．その面倒を省くため python を利用して周波数を arduino に送り，arduino の出力結果をデータに保存させる GUI を作成した．以下のプログラムは GUI で周波数を arduino へシリアル通信で送信して arduino に出力されたデータをシリアル通信で python 側に送る流れをするための，それぞれ GUI の python プログラムと送電側のマイ

コンの arduino プログラム，受電側の arduino プログラムである．なお，python のプログラムにおいて GUI を作成に tkinter を使用した．tkinter の使用方法並びに python の使い方については参考文献：[5] を参考にした．

ソースコード 1: GUI プログラム

```
import time
import statistics
import math
import serial
import collections
import csv
import tkinter as tk
import tkinter.filedialog as tkFileDialog
import tkinter.font as tkFont
import tkinter.ttk as ttk

x=0
L=[] #dataを保存
L1=[]
L2=[]
Lmsave=[]
Lmsend=[]
Lmreceive=[]
fre=0 #測定範囲の最小値
laf=0 #1目盛りの周波数
data=0 #測定範囲の最大値
ser1=0 #送電側のシリアル通信
ser2=0 #受電側のシリアル通信
t=1
tm=0



def maindef():
global x
global L
global ser
global fre
global data
global laf
global ser1
global ser2
global t
global tm
global L1
global L2
global Lmsave
global Lmsend
global Lmreceive

if x==0:
t=1#一応
elif x==1:
#次の周波数を入力
if float(fre) > float(laf):
x=3
else:

ser1.write('a'.encode('ascii')) # arduinoへ開始の合図を送る。
ser2.write('a'.encode('ascii'))
ser1.write(fre.encode('ascii'))
ser2.write(fre.encode('ascii'))
ser1.flush() # バッファ内の待ちデータを送りきる。
ser2.flush()
print("send:"+fre+"kHz")
L.append(fre+"kHz")
```

```
x=2
t=1
elif x==2:
#データ受け取りスイープ
line1 = ser1.readline().decode('ascii').rstrip()
line2 = ser2.readline().decode('ascii').rstrip()
L1.append(line1)
L2.append(line2)
print(fre+" "+line1+" "+line2+" ")
L.append(fre+" "+line1+" "+line2+" ")
if t>=int(tm):
math1=collections.Counter(L1).most_common()[0][0]
math2=collections.Counter(L2).most_common()[0][0]
Lmsave.append(fre+" "+math1+" "+math2)
Lmsend.append(fre+" "+math1)
Lmreceive.append(fre+" "+math2)
if float(fre)+float(data)*0.001 > float(laf) and float(laf) > float(fre):
fre=laf
else:
fre=str(round(float(fre) + float(data)*0.001,3))
x=1
L1=[]
L2=[]
else:
t=t+1

elif x==3:
#データを送らない、後始末
stop_data()
x=0
elif x==4:
#データ受け取り通常
line1 = ser1.readline().decode('ascii').rstrip()
line2 = ser2.readline().decode('ascii').rstrip()
print(fre+" "+line1+" "+line2+" ")
L.append(fre+" "+line1+" "+line2+" ")

root.after(10,maindef)

class Ser:
def __init__(self):
self.ser=None

def start_connect(self):
global ser1
global ser2
comport1='COM4' # arduino ideで調べてから。送電側
comport2='COM3' #受電側必ずcomportは送電側受電側異なるものを使用
tushinsokudo=57600 # arduinoのプログラムと一致させる。
timeout=5# エラーになったときのために。とりあえず5秒で戻ってくる。
ser1=self.ser
ser2=self.ser
ser1 = serial.Serial(comport1,tushinsokudo,timeout=timeout)
ser2 = serial.Serial(comport2,tushinsokudo,timeout=timeout)
time.sleep(2) # 1にするとダメ！短いほうがよい。各自試す。

def send_com(self):
global x
global data
global fre
global laf
global ser1
global ser2
global L
global tm
global L1
global L2
global Lmsave
```

```
global Lmsend
global Lmreceive
# v,u,sの文字列は、
#ぞれぞれv.get(),u.get(),s.get()で取り出す。
#下部send_entry内のTextvariableでデータ入力
data=v.get()
fre=u.get()
laf=s.get()
tm=v1.get()
if data.isdecimal()==True and fre.isdecimal()==True and laf.isdecimal()==True and tm.isdecimal()==True:
ser1.write('a'.encode('ascii')) # arduinoへ開始の合図を送る。
ser2.write('a'.encode('ascii'))
ser1.write(fre.encode('ascii'))
ser2.write(fre.encode('ascii'))#送電側と受電側の送るデータの量を合わせるため,
#あえて周波数を送る.送らなかった場合, 送電側と受電側の出力にずれが生じるから.
ser1.flush() # バッファ内の待ちデータを送りきる。
ser2.flush()
print("send incease_fre:"+data+" first_fre:"+fre+" last_fre:"+laf)
print("frequency transmission_ep receiving_ep")
L.append("increase_frequency:"+data+" first_frequency:"+fre+" last_frequency:"+laf)
L.append("frequency transmission_ep receiving_ep")
print("send:"+fre+"kHz")
L.append(fre+"kHz")
L1=[]
L2=[]
Lmsave=[]
Lmsend=[]
Lmreceive=[]


if int(data)==0:
x=4
else:
x=2
t=1
else:
print("error")
v.set("")
u.set("")
s.set("")
v1.set("")
def stop_com(self):
global x
x=3


def connect(self):
self.start_connect()
send_button.configure(state=tk.NORMAL)
stop_button.configure(state=tk.NORMAL)
send_entry.configure(state=tk.NORMAL)
defalt_entry.configure(state=tk.NORMAL)
saveas_button.configure(state=tk.NORMAL)
max_entry.configure(state=tk.NORMAL)
time_entry.configure(state=tk.NORMAL)
connect_button.configure(state=tk.DISABLED)
saveas_combo.configure(state=tk.NORMAL)

def saveas():
global L
global data
global Lmreceive
global Lmsave
global Lmsend
secomb=vc.get()
if secomb=='all':
save(L)
else:
```

```
198 if data=='0':
199 print("error!!")
200 else:
201 if secomb=='sweep:fre-send-receive':
202 save(Lmsave)
203 elif secomb=='sweep:fre-send':
204 save(Lmsend)
205 elif secomb=='sweep:fre-receive':
206 save(Lmreceive)
207
208 def save(a):
209 filename=tkFileDialog.asksaveasfilename(defaultextension=".csv",filetypes=[("csv","*.csv*")])
210 with open(filename,'w') as fout:
211 fout.write("\n".join(a))
212
213
214 #周波数をclock_genelaterに送る
215 #ストップするときの関数
216 def stop_data():
217 global ser1
218 global ser2
219 global fre
220 ser1.write('b'.encode('ascii')) # arduinoへ終了の合図を送る。
221 ser2.write('b'.encode('ascii'))
222 ser1.flush() # バッファ内の待ちデータを送りきる。
223 ser2.flush()
224 ser1
225 print("--stop--")
226 L.append("stop")
227 fre='0'
228 time.sleep(1)
229
230 root=tk.Tk()
231 font=tkFont.Font(size=24)
232 ser=Ser()
233 v=tk.StringVar() # tk.TK()の後に書く。
234 u=tk.StringVar()
235 s=tk.StringVar()
236 v1=tk.StringVar()
237 vc=tk.StringVar()
238
239 #ボタン入力
240 connect_button=tk.Button(root,text='connect',font=font,height=2,padx=20,command=ser.connect)
241 connect_button.grid(row=0,column=0)
242 send_button=tk.Button(root,text='send',font=font,height=2,padx=20,command=ser.send_com)
243 send_button.grid(row=0,column=1)
244 send_button.configure(state=tk.DISABLED)
245 stop_button=tk.Button(root,text='stop',font=font,height=2,padx=20,command=ser.stop_com)
246 stop_button.grid(row=0,column=2)
247 stop_button.configure(state=tk.DISABLED)
248 #entry
249 send_entry=tk.Entry(root,font=font,textvariable=v)
250 send_entry.grid(row=1,column=1,columnspan=2)
251 send_entry.configure(state=tk.DISABLED)
252 defalt_entry=tk.Entry(root,font=font,textvariable=u)
253 defalt_entry.grid(row=2,column=1,columnspan=2)
254 defalt_entry.configure(state=tk.DISABLED)
255 max_entry=tk.Entry(root,font=font,textvariable=s)
256 max_entry.grid(row=3,column=1,columnspan=2)
257 max_entry.configure(state=tk.DISABLED)
258 time_entry=tk.Entry(root,font=font,textvariable=v1)
259 time_entry.grid(row=4,column=1,columnspan=2)
260 time_entry.configure(state=tk.DISABLED)
261
262 #label
263 label1=tk.Label(root,font=font,text='increase_frequency')
264 label1.grid(row=1,column=0)
265 label1_Hz=tk.Label(root,font=font,text='Hz')
```

```
label1_Hz.grid(row=1,column=3)
label2=tk.Label(root,font=font,text='first_frequency')
label2.grid(row=2,column=0)
label2_Hz=tk.Label(root,font=font,text='kHz')
label2_Hz.grid(row=2,column=3)
label3=tk.Label(root,font=font,text='last_frequency')
label3.grid(row=3,column=0)
label3_Hz=tk.Label(root,font=font,text='kHz')
label3_Hz.grid(row=3,column=3)
label4_time=tk.Label(root,font=font,text='Measurement_Time')
label4_second=tk.Label(root,font=font,text='ds')
label4_time.grid(row=4,column=0)
label4_second.grid(row=4,column=3)

#セーブボタン
saveas_button=tk.Button(root,text='save',font=font,height=2,padx=20,command=saveas)
saveas_button.grid(row=0,column=3)
saveas_button.configure(state=tk.DISABLED)

#COMBOBOX
Comb=['all','sweep:fre-send-receive','sweep:fre-send','sweep:fre-receive']
saveas_combo=ttk.Combobox(root,values=Comb,textvariable=vc)
vc.set(Comb[0])
saveas_combo.grid(row=0,column=4)
saveas_combo.configure(state=tk.DISABLED)

root.after(100,maindef)
root.mainloop()
```

ソースコード 2: 送電側 arduino

```
    #include <si5351.h>
    #include <Wire.h>
    #include<MsTimer2.h>
    Si5351 si5351;

    unsigned long long freq = 5000000ULL;
      /*出力周波数50kHz(これをいじって周波数を変える)freq× 0.01=周波数Hz*/
    unsigned long long pll_freq = 705000000000ULL;
    /*PLL周波数(いじるな)*/

    String data;
    float data0 = 0;
    float f = 0;

    void setup() {

    Serial.begin(57600);
    MsTimer2::set(100, flash);

    bool i2c_found;
     /*I2C通信ができるかどうかブール値を入れる変数*/
    i2c_found = si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
     /*I2C通信を確認(ライブラリreadme参照)*/
    if (!i2c_found) {
    Serial.println("Error:I2C");
    }

    si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
    /*振動子負荷容量(使うモジュールが8pFなのでこれ)*/
    si5351.set_freq_manual(freq, pll_freq, SI5351_CLK0);
       /*出力周波数,PLL周波数,設定先出力ピン設定*/
    si5351.set_phase(SI5351_CLK0, 0);
     /*位相(今回特に意味はない)*/
    si5351.pll_reset(SI5351_PLLA);
     /*PLLをリセット（使う前に一回リセット）*/
    si5351.update_status();
```

```
37      /*si5351のステータスを読む(今回特に使っていない)*/
38
39      while (Serial.available() == 0);
40      }
41
42
43
44
45      void flash(void) {
46      int i = analogRead(0);
47      f = i * 5.0 / 1023.0;
48      Serial.println(f);
49      }
50
51      void loop() {
52      char aizu = Serial.read();
53      if (aizu == 'a') {
54      MsTimer2::stop();
55       //新しいduty比に変更されるまでflash関数を止める
56      aizu = 'c';
57      receive_duty_data();
58      MsTimer2::start();
59      }
60
61      else if (aizu == 'b') {
62      MsTimer2::stop();
63      si5351.set_freq(400000, SI5351_CLK0);
64        /*信号を止める*/
65          /*!!!!!set_freq(0)!!!これでは止まらない!!!!!*/
66      }
67
68      else if (aizu == 'c') {
69      //pass
70      }
71      }
72
73      void receive_duty_data() {
74      data = Serial.readString();
75      data0 = data.toFloat();
76      unsigned long long freq = data0 * 100000;
77       /*1=0.01Hzなので末尾に00をつける.入力単位をキロにしたいので末尾に10^3をつける. */
78      si5351.set_freq(freq, SI5351_CLK0);
79       /*周波数セット*/
80      si5351.pll_reset(SI5351_PLLA);
81       /*念のためPLLをリセット*/
82      si5351.update_status();
83      }
```

ソースコード 3: 受電側 arduino

```
1       #include<MsTimer2.h>
2
3       float f = 0;
4       String data;
5       float data0 = 0;
6
7       void setup() {
8       Serial.begin(57600);
9       MsTimer2::set(100,flash);
10      while(Serial.available() == 0);
11      }
12
13      void flash(void){
14      int i = analogRead(0);
15      f = i * 5.0 / 1023.0;
16      Serial.println(f);
17      }
```

```
18
19      void loop() {
20      char aizu = Serial.read();
21
22      if(aizu == 'a'){
23      MsTimer2::stop();
24      aizu = 'c';
25      receive_duty_data();
26      MsTimer2::start();
27      }
28
29      else if(aizu == 'b'){
30      MsTimer2::stop();
31      aizu='c';
32      }
33
34      }
35
36      void receive_duty_data() {
37
38      data = Serial.readString();
39      data0 = data.toFloat();
40      }
```