

令和元年度 環境ロボティクス学科 卒業論文

周波数スイープによるワイヤレス給電の 最適動作周波数の探索について

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	3
1.1	背景	3
1.2	研究目的	3
2	論文の構成	4
3	原理	5
4	実験内容	5
5	実験結果	5
6	考察	5
7	結論	5
8	今後の展望	5
9	謝辞	6
10	付録	7
10.1	プログラムについて	7

1 緒言

1.1 背景

ワイヤレス給電とは、導線や金属接点などを介さずに電力を送電、供給することが可能な給電方法である。これにより、従来の電源コードでの水によるショートや感電事故の回避、給電が困難な場所に機器がある場合でも給電することが可能、電子機器の利便性向上など利点が挙げられる。(参考文献:[1], [2], [3]) ワイヤレス給電の利用としては、スマートフォンやモバイルバッテリーを置くだけで充電することができる充電器や、医療福祉ではペースメーカーへの電力供給の利用、工業や産業においてロボットアームなどにも活用され、今後様々な分野で注目されている。

1.2 研究目的

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。本研究はワイヤレス給電の送電側、受電側の電力を周波数-電力の周波数スイープで測定し測定されたデータをもとに最適動作周波数を見つけ出しこれを評価並びに考察することを主目的とする。周波数スイープで出力可能なプログラムの開発並びに1周波数の測定時間の決定を事前に行う。

2 論文の構成

3 原理

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ご指導・助言して下さった自動制御研究室の先輩方並びに，共に研究した同期のメンバーに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，2018 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，2018 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：2019/1/20
- [4] 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：2019/1/21

10 付録

10.1 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：[4] を参考にした。

ソースコード 1: GUI プログラム

```
1  import time
2  import serial
3  import csv
4  import tkinter as tk
5  import tkinter.filedialog as tkFileDialog
6  import tkinter.font as tkFont
7
8  x=0
9  L=[] #dataを保存
10 fre=0 #測定範囲の最小値
11 laf=0 #1目盛りの周波数
12 data=0 #測定範囲の最大値
13 ser1=0 #送電側のシリアル通信
14 ser2=0 #受電側のシリアル通信
15 t=1
16 tm=0
17
18
19 def maindef():
20     global x
21     global L
22     global ser
23     global fre
24     global data
25     global laf
26     global ser1
27     global ser2
28     global t
29     global tm
30
31     if x==0:
32         t=1#一応
33     elif x==1:
34         #データ受け取り、次の周波数を入力
35         if float(fre) > float(laf):
36             x=3
37         else:
38
39             ser1.write('a'.encode('ascii')) # arduinoへ開始の合図を送る。
40             ser2.write('a'.encode('ascii'))
41             ser1.write(fre.encode('ascii'))
42             ser2.write(fre.encode('ascii'))
43             ser1.flush() # バッファ内の待ちデータを送りきる。
44             ser2.flush()
```

```
45     print("send:"+fre+"kHz")
46     L.append(fre+"kHz")
47     x=2
48     t=1
49     elif x==2:
50         line1 = ser1.readline().decode('ascii').rstrip()
51         line2 = ser2.readline().decode('ascii').rstrip()
52         line3 = str(round(float(line2) / float(line1),3))
53         print(fre+" "+line1+" "+line2+" "+line3)
54         L.append(fre+" "+line1+" "+line2+" "+line3)
55         if t>=int(tm)*10:
56             if float(fre)+float(data)*0.001 > float(laf) and float(laf) > float(fre):
57                 fre=laf
58             else:
59                 fre=str(round(float(fre) + float(data)*0.001,3))
60             x=1
61         else:
62             t=t+1
63
64     elif x==3:
65         #データを送らない、後始末
66         stop_data()
67         x=0
68     elif x==4:
69         line1 = ser1.readline().decode('ascii').rstrip()
70         line2 = ser2.readline().decode('ascii').rstrip()
71         line3 = str(round(float(line2) / float(line1),3))
72         print(fre+" "+line1+" "+line2+" ")
73         L.append(fre+" "+line1+" "+line2+" ")
74
75     root.after(10,maindef)
76
77     class Ser:
78     def __init__(self):
79         self.ser=None
80
81     def start_connect(self):
82         global ser1
83         global ser2
84         comport1='COM3' # arduino ideで調べてから。送電側
85         comport2='COM4' #受電側必ずcomportは送電側受電側異なるものを使用
86         tushinsokudo=57600 # arduinoのプログラムと一致させる。
87         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。
88         ser1=self.ser
89         ser2=self.ser
90         ser1 = serial.Serial(comport1,tushinsokudo,timeout=timeout)
91         ser2 = serial.Serial(comport2,tushinsokudo,timeout=timeout)
92         time.sleep(2) # 1にするとダメ！短いほうがよい。各自試す。
93
94     def send_com(self):
95         global x
96         global data
97         global fre
98         global laf
99         global ser1
100        global ser2
101        global L
102        global tm
103        # v,u,sの文字列は、
104        #それぞれv.get(),u.get(),s.get()で取り出す。
105        #下部send_entry内のTextvariableでデータ入力
106        data=v.get()
107        fre=u.get()
108        laf=s.get()
109        tm=v1.get()
110        if data.isdecimal()==True and fre.isdecimal()==True and laf.isdecimal()==True and tm.isdecimal()==True:
111            ser1.write('a'.encode('ascii')) # arduinoへ開始の合図を送る。
112            ser2.write('a'.encode('ascii'))
```



```
113     ser1.write(fre.encode('ascii'))
114     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、
115     #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。
116     ser1.flush() # バッファ内の待ちデータを送りきる。
117     ser2.flush()
118     print("send incease_fre:"+data+" first_fre:"+fre+" last_fre:"+laf)
119     print("frequency transmission_ep receiving_ep power_efficiency")
120     L.append("increase_frequency:"+data+" first_frequency:"+fre+" last_frequency:"+laf)
121     L.append("frequency transmission_ep receiving_ep power_efficiency")
122     print("send:"+fre+"kHz")
123     L.append(fre+"kHz")
124
125     if float(data)==0.0:
126         x=4
127     else:
128         x=2
129         t=1
130     else:
131         print("error")
132         v.set("")
133         u.set("")
134         s.set("")
135         v1.set("")
136     def stop_com(self):
137         global x
138         x=3
139
140
141     def connect(self):
142         self.start_connect()
143         send_button.configure(state=tk.NORMAL)
144         stop_button.configure(state=tk.NORMAL)
145         send_entry.configure(state=tk.NORMAL)
146         default_entry.configure(state=tk.NORMAL)
147         saveas_button.configure(state=tk.NORMAL)
148         max_entry.configure(state=tk.NORMAL)
149         time_entry.configure(state=tk.NORMAL)
150         connect_button.configure(state=tk.DISABLED)
151
152     def saveas():
153         global L
154         filename=tkFileDialog.asksaveasfilename(defaultextension=".csv",filetypes=[("csv","*.csv*")])
155
156         with open(filename,'w') as fout:
157             fout.write("\n".join(L))
158             #周波数をclock_genelaterに送る
159             #ストップするときの関数
160         def stop_data():
161             global ser1
162             global ser2
163             global fre
164             ser1.write('b'.encode('ascii')) # arduinoへ終了の合図を送る。
165             ser2.write('b'.encode('ascii'))
166             ser1.flush() # バッファ内の待ちデータを送りきる。
167             ser2.flush()
168             ser1
169             print("--stop--")
170             L.append("stop")
171             fre='0'
172             time.sleep(1)
173
174         root=tk.Tk()
175         font=tkFont.Font(size=24)
176         ser=Ser()
177         v=tk.StringVar() # tk.TK()の後に書く。
178         u=tk.StringVar()
179         s=tk.StringVar()
180         v1=tk.StringVar()
```

```

181 #ボタン入力
182 connect_button=tk.Button(root,text='connect',font=font,height=2,padx=20,command=ser.connect)
183 connect_button.grid(row=0,column=0)
184 send_button=tk.Button(root,text='send',font=font,height=2,padx=20,command=ser.send_com)
185 send_button.grid(row=0,column=1)
186 send_button.configure(state=tk.DISABLED)
187 stop_button=tk.Button(root,text='stop',font=font,height=2,padx=20,command=ser.stop_com)
188 stop_button.grid(row=0,column=2)
189 stop_button.configure(state=tk.DISABLED)
190 #entry
191 send_entry=tk.Entry(root,font=font,textvariable=v)
192 send_entry.grid(row=1,column=1,columnspan=2)
193 send_entry.configure(state=tk.DISABLED)
194 default_entry=tk.Entry(root,font=font,textvariable=u)
195 default_entry.grid(row=2,column=1,columnspan=2)
196 default_entry.configure(state=tk.DISABLED)
197 max_entry=tk.Entry(root,font=font,textvariable=s)
198 max_entry.grid(row=3,column=1,columnspan=2)
199 max_entry.configure(state=tk.DISABLED)
200 time_entry=tk.Entry(root,font=font,textvariable=v1)
201 time_entry.grid(row=4,column=1,columnspan=2)
202 time_entry.configure(state=tk.DISABLED)
203
204 #label
205 label1=tk.Label(root,font=font,text='increase_frequency')
206 label1.grid(row=1,column=0)
207 label1_Hz=tk.Label(root,font=font,text='Hz')
208 label1_Hz.grid(row=1,column=3)
209 label2=tk.Label(root,font=font,text='first_frequency')
210 label2.grid(row=2,column=0)
211 label2_Hz=tk.Label(root,font=font,text='kHz')
212 label2_Hz.grid(row=2,column=3)
213 label3=tk.Label(root,font=font,text='last_frequency')
214 label3.grid(row=3,column=0)
215 label3_Hz=tk.Label(root,font=font,text='kHz')
216 label3_Hz.grid(row=3,column=3)
217 label4_time=tk.Label(root,font=font,text='Measurement_Time')
218 label4_second=tk.Label(root,font=font,text='s')
219 label4_time.grid(row=4,column=0)
220 label4_second.grid(row=4,column=3)
221
222 #セーブボタン
223 saveas_button=tk.Button(root,text='save',font=font,height=2,padx=20,command=saveas)
224 saveas_button.grid(row=0,column=3)
225 saveas_button.configure(state=tk.DISABLED)
226
227 root.after(100,maindef)
228 root.mainloop()

```

ソースコード 2: 送電側 arduino

```

1 #include <si5351.h>
2 #include <Wire.h>
3 #include <MsTimer2.h>
4 Si5351 si5351;
5
6 unsigned long long freq = 50000000ULL;
7 /*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*/
8 unsigned long long pll_freq = 705000000000ULL;
9 /*PLL周波数(いじるな)*/
10
11 String data;
12 float data0 = 0;
13 float f = 0;
14
15 void setup() {
16

```

```
17 Serial.begin(57600);
18 MsTimer2::set(100, flash);
19
20 bool i2c_found;
21 /* I2C通信ができるかどうかブール値を入れる変数*/
22 i2c_found = si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
23 /* I2C通信を確認(ライブラリreadme参照)*/
24 if (!i2c_found) {
25     Serial.println("Error:I2C");
26 }
27
28 si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
29 /*振動子負荷容量(使うモジュールが8pFなのでこれ)*/
30 si5351.set_freq_manual(freq, pll_freq, SI5351_CLK0);
31 /*出力周波数,PLL周波数,設定先出力ピン設定*/
32 si5351.set_phase(SI5351_CLK0, 0);
33 /*位相(今回特に意味はない)*/
34 si5351.pll_reset(SI5351_PLLA);
35 /*PLLをリセット(使う前に一回リセット)*/
36 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   }
```