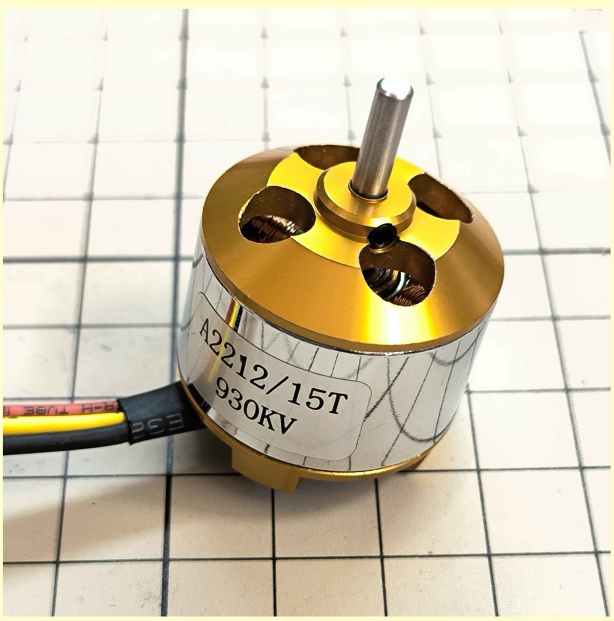
雑にQすBLDCモーター



作者:ΔV研

> 電動機（でんどうき、英: Electric motor）とは、電気エネルギーを力学的エネルギーに変換する電力機器、原動機の総称。モーター、電気モーターとも呼ばれる。
> — wikipedia, 電動機

まえがき

本書は, 制御工学や電子回路については知っていたり DC モータを触ったことはあるが BLDC を触ってみたい, という人に向けて最短経路でとりあえず BLDC モータを回すまでの流れを駆け抜ける本です. そのため本書をきっかけにモータについて楽しんで, より高度な内容を勉強してもっと高度な回し方をする勉強の足がかりになれば幸いです.

目次

第 1 章 前提知識 3
1.1 モータの分類, 使う同期モータとは . . . 3
1.2 永久磁石同期モータ (PMSM) の構造, モータモデル . . . 4
1.3 120 度通電制御 . . . 7
1.4 ベクトル制御 . . . 8
1.4.1 クラーク変換 (三相二相変換) . . . 8
1.4.2 パーク変換 . . . 9
1.4.3 三次高調波重畳 . . . 9
1.4.4 空間ベクトル変調 (SVM) . . . 10
1.5 ブロック図と dq 電流値制御 . . . 12
1.6 非干渉化制御 . . . 13
1.7 モータパラメータ測定 . . . 15
1.8 ホールセンサから電気角を推定する方法 . . . 15

第 2 章 実装編 17
2.1 回路図 . . . 17
2.2 物理的構成 . . . 19
2.3 dq 電圧制御 . . . 20
2.4 dq 電流制御 . . . 22
2.5 プログラム . . . 24

第 1 章

前提知識

1.1 モータの分類, 使う同期モータとは

"モータ", と言われて皆さんは何を想像するでしょうか.

タミヤのモータ?扇風機?鉄道趣味の人なら電車のモータなどを思い浮かべるでしょう. モータには様々な種類が存在し, その特徴と特性に応じて様々な使われ方をしています.

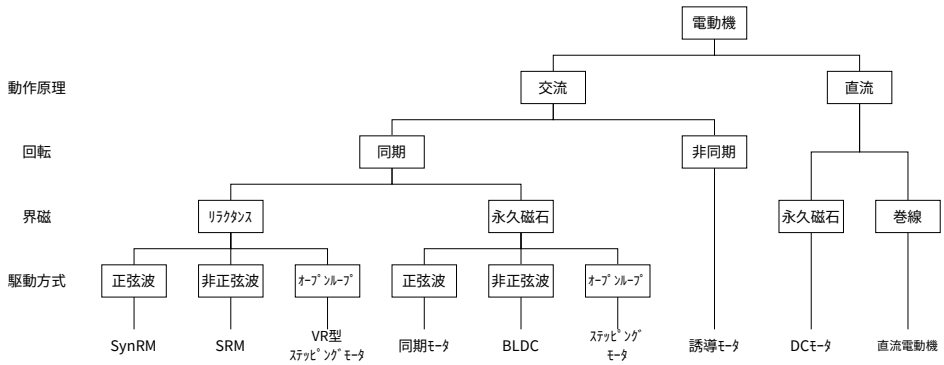


図 1.1: モータ (電動機) の分類

図 1.1 はモータの種類を大まかに分類した図です. 図 1.1 から更に細かい構造の違いで無数に種類が存在します. 今回はこの中から同期モータ,PMSM[*1]と呼ばれるモータを回します.

同期モータには原則位置センサが不可欠です. 基本的にはロータリーエンコーダやホールセンサなどにより位置を測定し, 回転角度を用いてモータの制御を行います.

[*1] Permanent Magnet Synchronous Motor; 永久磁石同期電動機

2.5 プログラム

完全版 (STM32CubeIDE のファイル群) は GitHub に公開しています.
https://github.com/rei512/C105/

プログラム 2.1: 実装したプログラム (抜粋)

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include <math.h>
#include <stdio.h>
#include <string.h>

/* Private define ------------------------------------------------------------*/
#define PI 3.141593f

#define Vdd 3.3f
#define PWM_RANGE 1000.0f

#define Ra 0.09f
#define La 0.12e-3f
#define Kt 2.2866e-3f
#define J 0.000001f

#define V_d 0.0f
#define V_q 0.0f

#define I_d 0.0f
#define I_q 0.5f

#define Wi 1000.0f
#define KIp (Wi * La)     // La
#define KIi (Wi * Ra / 1000.0f) // Ra

/* Private variables ---------------------------------------------------------*/
uint8_t buff[100];

uint16_t adc_dma[3];

float TIM_before;
float theta_e;
float vd, vq;
float id, iq;
float id_raw, iq_raw;
float U, V, W;
float iu, iv, iw;
float speed;

float id_integ, iq_integ;

int time;

/* Private user code ---------------------------------------------------------*/
// 指令値電圧をPWMのしきい値に変換
int convert_PWM(float voltage)
{
  int pwm = PWM_RANGE / 2.0f + PWM_RANGE / 2.0f * voltage * 2.0f / Vdd;
  if (pwm < 0)
  {
    pwm = 0;
  }
  else if (pwm > PWM_RANGE)
  {
    pwm = PWM_RANGE - 1;
  }
```

```
59    return pwm;
60  }
61
62  // ADCの入力を電流に変換
63  float convert_current(int adc)
64  {
65    return ((float)adc * 3.3f / 4096.0f - 1.66f) / 0.092f;
66  }
67
68  // 逆パーク変換
69  void dq_to_ab(float d, float q, float theta, float *a, float *b)
70  {
71    *a = d * cosf(theta) - q * sinf(theta);
72    *b = d * sinf(theta) + q * cosf(theta);
73  }
74
75  // uvw→dq変換
76  void uvw_to_dq(float u, float v, float w, float theta, float *d, float *q)
77  {
78    // クラーク変換
79    float alpha = u;
80    float beta = (u + 2.0f * v) / sqrtf(3.0f);
81
82    // パーク変換
83    *d = (alpha * cosf(theta) + beta * sinf(theta)) / sqrtf(3.0f / 2.0f);
84    *q = (-alpha * sinf(theta) + beta * cosf(theta)) / sqrtf(3.0f / 2.0f);
85  }
86
87  // 空間ベクトル変調
88  void ab_to_svm(float a, float b, float *u, float *v, float *w)
89  {
90    float v1, v2, v3;
91    int VecSector;
92
93    // 各成分の計算
94    v1 = b;
95    v2 = 0.5f * b + \sqrtf{3} / 2.0f * a;
96    v3 = v2 - v1;
97
98    // セクターの特定
99    VecSector = 3;
100   VecSector = (v2 >= 0) ? (VecSector - 1) : VecSector;
101   VecSector = (v3 > 0) ? (VecSector - 1) : VecSector;
102   VecSector = (v1 < 0) ? (7 - VecSector) : VecSector;
103
104   // セクター別の処理
105   switch (VecSector)
106   {
107   case 1:
108   case 4:
109     *u = v2;
110     *v = v1 - v3;
111     *w = -v2;
112     break;
113
114   case 2:
115   case 5:
116     *u = v2 + v3;
117     *v = v1;
118     *w = -v1;
119     break;
120
121   case 3:
122   case 6:
123     *u = v3;
124     *v = -v3;
125     *w = -v1 - v2;
126
127   default:
128     break;
129   }
```

```
130 }
131
132 // dq→uvw変換
133 void dq_to_uvw(float d, float q, float theta, float *u, float *v, float *w)
134 {
135   // 逆パーク変換
136   float alpha = d * cosf(theta) - q * sinf(theta);
137   float beta = d * sinf(theta) + q * cosf(theta);
138
139   ab_to_svm(a, b, u, v, w);
140 }
141
142 // 極零相殺
143 void pole_zero_cancel(float *vd, float *vq, float id, float iq)
144 {
145   *vd -= speed * iq * La;
146   *vq += speed * (id * La + Kt / 7.0f);
147 }
148
149 // PI制御器
150 float PI_Controller(float error, float *integral, float Kp, float Ki, float Limit)
151 {
152   // 出力の計算
153   float output = Kp * error + Ki * (*integral + error);
154
155   // アンチワインドアップ
156   if (output > Limit)
157     output = Limit;
158   else if (output < -Limit)
159     output = -Limit;
160   else
161     *integral += error;
162
163   return output;
164 }
165
166 // タイマ割り込みのコールバック関数
167 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
168 {
169   if (htim == &htim1) // PWM割り込み 21 kHz
170   {
171     // 電気角位置の計算
172     theta_e = TIM5->CNT / (float)TIM_before * 2.0f * PI + 1.25f;
173     if (theta_e < 0.0f)
174     {
175       theta_e += 2.0f * PI;
176     }
177     else if (theta_e > 2.0f * PI)
178     {
179       theta_e -= 2.0f * PI;
180     }
181
182     // ADCから電流取得
183     iu = convert_current(adc_dma[0]);
184     iv = convert_current(adc_dma[1]);
185     iw = convert_current(adc_dma[2]);
186
187     // dq軸電流の計算
188     uvw_to_dq(iu, iv, iw, theta_e, &id_raw, &iq_raw);
189
190     // 出力電圧の変更
191     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, convert_PWM(U));
192     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, convert_PWM(V));
193     __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, convert_PWM(W));
194
195     // 次のADCスタート
196     HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adc_dma, 3);
197   }
198   else if (htim == &htim2) // 制御部 1 kHz
199   {
200     // 計算したdq軸電流を取得
```

```
201     id = id_raw;
202     iq = iq_raw;
203
204     // PI制御器によるq軸電流指令値の決定
205     float iq_ref = PI_Controller(SPEED - speed, &speed_integ, KSp, KSi, 0.5f);
206
207     // PI制御器によるdq軸電圧指令値の決定
208     vd = PI_Controller(I_d - id, &id_integ, KIp, KIi, 2.0f);
209     vq = PI_Controller(I_q - iq, &iq_integ, KIp, KIi, 2.0f);
210
211     // 非干渉制御
212     pole_zero_cancel(&vd, &vq, id, iq);
213
214     // インバータが出力する電圧指令値の計算
215     dq_to_uvw(vd, vq, theta_e, &U, &V, &W);
216
217     // 速度の計算
218     speed = 1.0f / (TIM_before / 84e6f) * 2.0f * PI / 7.0f;
219
220     time++; // ms
221   }
222 }
223
224 // 外部割り込み(EXTI)のコールバック関数
225 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
226 {
227   // 前回の割り込みからの時間を取得
228   TIM_before = TIM5->CNT;
229   // タイマーのカウントをリセット
230   TIM5->CNT = 0;
231 }
232
233 int main(void)
234 {
235   setbuf(stdout, NULL);
236   HAL_TIM_Base_Start_IT(&htim1);
237   HAL_TIM_Base_Start_IT(&htim5);
238   HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
239   HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
240   HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
241   HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
242   HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
243   HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);
244
245   HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adc_dma, 3);
246
247   printf("Time[ms]\tV_d[V]\tV_q[V]\tI_d[A]\tI_q[A]\tSpeed[rad/s]\n");
248
249   // 初期位置決定のために2周させる
250   dq_to_uvw(0.0f, 2.0f, 0.0f, &U, &V, &W);
251   HAL_Delay(50);
252   dq_to_uvw(0.0f, 2.0f, 2.0f / 3.0f * PI, &U, &V, &W);
253   HAL_Delay(50);
254   dq_to_uvw(0.0f, 2.0f, 4.0f / 3.0f * PI, &U, &V, &W);
255   HAL_Delay(50);
256   dq_to_uvw(0.0f, 2.0f, 0.0f, &U, &V, &W);
257   HAL_Delay(50);
258   dq_to_uvw(0.0f, 2.0f, 2.0f / 3.0f * PI, &U, &V, &W);
259   HAL_Delay(50);
260   dq_to_uvw(0.0f, 2.0f, 4.0f / 3.0f * PI, &U, &V, &W);
261   HAL_Delay(50);
262   dq_to_uvw(0.0f, 2.0f, 0.0f, &U, &V, &W);
263   HAL_Delay(50);
264
265   // 制御部スタート
266   HAL_TIM_Base_Start_IT(&htim2);
267
268   while (1);
269 }
```

あとがき

やっとこさ終わりました. 色々とやり残した部分も多いですが, 一旦区切りたいと思います. 本当は位置センサレスベクトル制御まで行ければよかったのですが, 初心者向けとして書くには難しかったです. ここまで読んだそこの君!モータ制御界隈にはセンサレス制御という沼があるぞ!是非一緒に浸かろう!

あとは, 本書を読んだあとに興味を持ってくれたのであれば, 電気学会が同期モータのセンサレス制御について体系的にまとめた本が存在するので是非読んでみましょう (AC ドライブシステムのセンサレスベクトル制御 ― 電気学会・センサレスベクトル制御の整理に関する調査専門委員会)

また電機関連は各メーカーがネットで資料を大量に公開しているので読み漁るのも手ですヨ.

雑に回す BLDC モーター

2024 年 12 月 30 日　初版 1 刷発行　（コミックマーケット C105）

著者　：　Δ V 研，rei_512_

発行　：　Δ V 研

印刷所：　印刷通販プリントネット

連絡先：　[E-mail] contact@deltav-lab.org

　　：　[X] @DeltaV_Lab

ISDN278-4-871464-02-1

定価 500 円

AV LAB
ORBITAL VELOCITY

ISDN278-4-871464-02-1
C3054 ¥500E