Professional Drone, Hybrid Power Pack - Timebox 10

Team 2

29. maj 2019

Deltagere:

Stud. nr: 201602094	Navn: Søren Holm Korsgaard
Stud.nr.: 201607563	Navn: Jacob Gustafsson
Stud.nr.: 20084327	Navn: Simon Rasmussen
Stud.nr.: 201704483	Navn: Thomas Dueholm Jensen

Indhold

1 Strategy and planning

I denne timebox præsenteres resultater af test af ensretter samt PID-kontrol. Det er den afsluttende timebox for projekt 4 og der uddybes derfor ikke på videreudvikling.

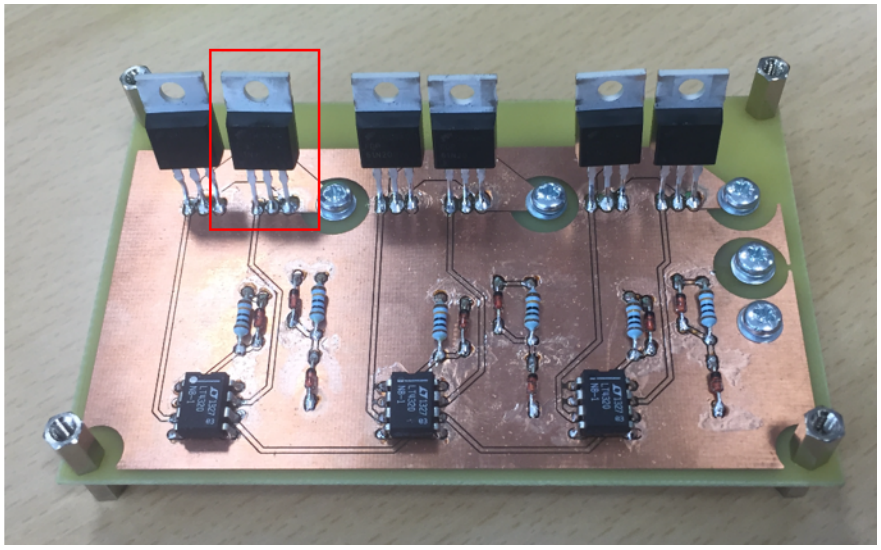
2 Aktiv ensretter (Thomas)

I timebox 9 konkluderede vi, at der skulle monteres nye MOSFETS med en lavere intern modstand og/eller påmonteres en heatsink på PCB printet Vi indkøbte derfor følgende MOSFETS

- $OptiMOS^{TM}5$ Power-transistor
  - Max drain strøm: 100 A.
  - Max drain-source spænding: 100 V.
  - $R_{DS(on)}$: 3.9 mΩ.
  - $R_{\theta JA}$: 62 °C/W

Disse MOSFET transistorer blev monteret på PCB printet, hvorefter samme testprocedure som tidligere blev gennemført.

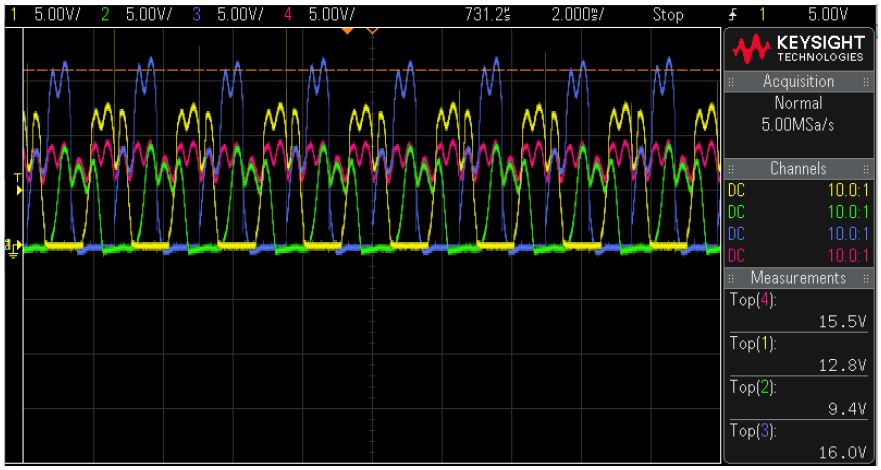
Testen resulterede dog i en defekt transistor.



Figur 1: Defekt transistor. Markeret med rød firkant.

Årsagen til defekten var på daværende tidpunkt uafklaret, hvorfor det blev besluttet at måle gate spændingerne på samtlige transistorer i forhold til stel, da vi havde en mistanke om, at der var ubalance i kredsløbet. Ubalance kunne muligvis være årsagen til defekten af den ene transistor.

Figur 2 viser målingen af gate spændingen på de tre transistorer, der håndterer den positive del af inputtet. Hvis kredsløbet fungerede korrekt, skulle der gerne være samme spændingsniveau på samtlige gates.



Figur 2: Gatespænding. Grøn: fase 1+, blå: fase 2+, gul: fase 3+, rød: V_{out}

Her ses, at gate spændingerne ikke har samme niveau. Peak værdier varierer mellem 9.4 V (fase 1), 12.8 V (fase 2) og 16.0 V (fase 3).

På baggrund af denne måling, og da vi ikke havde mere tid til rådighed, var vi desværre nødsaget til at stoppe udviklingen af den aktive ensretter.

3 Motorstyring (Simon)

På baggrund af tidligere PID-kode, se timebox 6 udarbejdes der nu et udkast til PID-kontrol software, som også benytter sig af tidligere udviklet kode til kontrol af servomotor samt aflæsning af motorens omdrejninger.

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"
#include "rpm-detect.h"
#include "servo_driver.h"

// Filtrering
double alpha = 0.2;
double measuredSpeed = 0;

// Koefficienter
double kp = 0.001;
double ki = 0.0514;
double kd = -1.4946;
double K1;
double K2;
double K3;

// Vægtning af koefficienter
double setpointWeight = 0.2;
double lowpassSpeed;
double setpointSpeed;

// Diverse
double output;
double throttleopen = 1.0;
float throttlePos;

int rpmch;
int MAX_RPM = 9000;

// Initialiering
```

```
37  double lastSetpointSpeed = 0;
38  double lastMeasuredSpeed = 0;
39  double lastLowpassSpeed = 0;
40  double lastOutput = 0;
41  double lastLastMeasuredSpeed = 0;
42
43  void velPID(int setpointSpeed, int measuredSpeed) {
44          lowpassSpeed = alpha * lastLowpassSpeed + (1 - alpha) * measuredSpeed;
45          K1 = kp * setpointWeight * (setpointSpeed - lastSetpointSpeed)
46                          + kp * (lastMeasuredSpeed - lowpassSpeed);
47          K2 = ki * (setpointSpeed - lowpassSpeed);
48          K3 = kd * (2 * lastMeasuredSpeed - lowpassSpeed - lastLastMeasuredSpeed);
49          output = lastOutput - K1 - K2 - K3;
50          if (output < 0) {
51                  output = 0;
52          }
53          throttlePos = output / MAX_RPM;
54          lastLowpassSpeed = lowpassSpeed;
55          lastLastMeasuredSpeed = lastMeasuredSpeed;
56          lastMeasuredSpeed = lowpassSpeed;
57          lastSetpointSpeed = setpointSpeed;
58          lastOutput = output;
59          angle_throttle(throttlePos);
60  }
61
62  int main(void) {
63          /* Init board hardware. */
64          BOARD_InitBootPins();
65          BOARD_InitBootClocks();
66          BOARD_InitBootPeripherals();
67          /* Init FSL debug console. */
68          BOARD_InitDebugConsole();
69          // init_read_rpm();
70          rpmch = 7000;
71          init_read_rpm();
72          init_pwm();
73          while (1) {
74                  int measure_rpm = get_rpm();
75                  velPID(5000, measure_rpm); // set rpm to 5000 as setpoint
76                  }
77          return 0;
78  }
```

Ved implementering af PID-koden sås desværre insufficient kontrol af omdrejningerne, idet kontrollen fremstod invers, sådan at ved øget omdrejningstal blev gasspjældet mere åbent. Der var afslutningsvis på semesteret desværre ikke tid til mere udvikling.