Bijlage C

[PID programma]

```
//  PID_v1 library geschreven door Brett Beauregard

#include <PID_v1.h>
#include <Servo.h>

Servo ESC;  //Maak een nieuwe servo object met naam "ESC"
const int buttonPin = 2;  //Definieer de pin van de Arduino voor de resetknop
int buttonStatus = 0;

double Setpoint; //De setpoint wordt uitgedrukt in een 10-bit getal
double Input; //De input is de sensorspanning van de F2 flow sensor
double Output; //De commando aan de ESC in een getal van 0 tot 40

double Kp=1, Ki=0.03, Kd=0.02;  //De 3 PID parameters die bepaalt hoe onze regelaar reageert

PID AirPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);  //Maak een nieuwe PID object met
de 3 parameters en de SP, Input, Output

void setup() {
  ESC.attach(9,1000,2000); //Link de ESC met pin 9, definieer de pulsbreedtes=1000µs tot
2000µs
  Serial.begin(9600); //Start seriele monitor met een snelheid van 9600 bits per seconde

   Setpoint = 200;
   //Turn the PID on
   AirPID.SetMode(AUTOMATIC); //Laat de PID zelf regelen
   AirPID.SetTunings(Kp, Ki, Kd); //Gebruik de drie parameters om de PID te tunen
   AirPID.SetOutputLimits(0, 40); //Zet de minimum en maximum commando's voor de ESC hier

}

void loop() {
  buttonStatus = digitalRead(buttonPin);
  int Air_sensorValue = analogRead(A2);
  Input = map(Air_sensorValue, 75, 470, 0, 255); //De gemeten digitale signaal omzetten naar 0
tot 255 signaal
  //PID calculation
  AirPID.Compute(); //PID voert berekeningen uit

  if (buttonStatus == HIGH) {  //Als resetknop ingedrukt is stopt de motor
```

```
    ESC.write(0);
  } else {
    analogWrite(9,Output);
    ESC.write(Output);    //Stuur de berekende Output als commando naar ESC
  }
  //Maak een grafiek
  Serial.print(40);
  Serial.print(' ');
  Serial.print(Input);
  Serial.print(' ');
  Serial.println(Output);
  Serial.print(' ');
  Serial.println(Setpoint);
}
```

[Fuzzy logic programma]

```
//Fuzzy library geschreven door A.J. Alves

#include <Fuzzy.h>
#include <Servo.h>

Servo ESC;
const int buttonPin = 2;
int buttonState = 0;

// Instantiating a Fuzzy object
Fuzzy *fuzzy = new Fuzzy();

void setup()
{
  ESC.attach(9,1000,2000);
  // Set the Serial output
  Serial.begin(9600);
  randomSeed(analogRead(0));

  // Instantiating a FuzzyInput object
  FuzzyInput *wind = new FuzzyInput(1);
  // Instantiating a FuzzySet object
  FuzzySet *geen_wind = new FuzzySet(25, 75, 75, 175);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(geen_wind);
  // Instantiating a FuzzySet object
  FuzzySet *weinig = new FuzzySet(75, 175, 175, 275);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(weinig);
  // Instantiating a FuzzySet object
  FuzzySet *gemiddeld = new FuzzySet(150, 275, 275, 395);
  // Including the FuzzySet into FuzzyInput
  wind->addFuzzySet(gemiddeld);
  //new set
  FuzzySet *veel = new FuzzySet(220, 370, 370, 500);
  wind->addFuzzySet(veel);
  //new set
  FuzzySet *maximum = new FuzzySet(370, 470, 470, 500);
  wind->addFuzzySet(maximum);
  // Including the FuzzyInput into Fuzzy
  fuzzy->addFuzzyInput(wind);
```

```
// Instantiating a FuzzyOutput objects
FuzzyOutput *ESC = new FuzzyOutput(1);
FuzzySet *minimum = new FuzzySet(-20, 0, 0, 8);
ESC->addFuzzySet(minimum);
FuzzySet *traag = new FuzzySet(3, 14, 14, 25);
ESC->addFuzzySet(traag);
FuzzySet *med = new FuzzySet(5, 20, 20, 35);
ESC->addFuzzySet(med);

FuzzySet *snel = new FuzzySet(20, 30, 30, 40);
ESC->addFuzzySet(snel);
// Including the FuzzyOutput into Fuzzy
FuzzySet *zeer_snel = new FuzzySet(30, 40, 40, 50);
ESC->addFuzzySet(zeer_snel);
fuzzy->addFuzzyOutput(ESC);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindGeen = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
ifWindGeen->joinSingle(geen_wind);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenESCZeerSnel = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenESCZeerSnel->addOutput(zeer_snel);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule01 = new FuzzyRule(1, ifWindGeen, thenESCZeerSnel);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule01);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindWeinig = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
ifWindWeinig->joinSingle(weinig);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenESCSnel = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenESCSnel->addOutput(snel);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule02 = new FuzzyRule(2, ifWindWeinig, thenESCSnel);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule02);

// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *ifWindGemiddeld = new FuzzyRuleAntecedent();
```

```
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
  ifWindGemiddeld->joinSingle(gemiddeld);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenESCMed = new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenESCMed->addOutput(med);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule03 = new FuzzyRule(3, ifWindGemiddeld, thenESCMed);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule03);

  FuzzyRuleAntecedent *ifWindVeel = new FuzzyRuleAntecedent();
  ifWindVeel->joinSingle(veel);
  FuzzyRuleConsequent *thenESCTraag = new FuzzyRuleConsequent();
  thenESCTraag->addOutput(traag);
  FuzzyRule *fuzzyRule04 = new FuzzyRule(4, ifWindVeel, thenESCTraag);
  fuzzy->addFuzzyRule(fuzzyRule04);

  FuzzyRuleAntecedent *ifWindMax = new FuzzyRuleAntecedent();
  ifWindMax->joinSingle(maximum);
  FuzzyRuleConsequent *thenESCMinimum = new FuzzyRuleConsequent();
  thenESCMinimum->addOutput(minimum);
  FuzzyRule *fuzzyRule05 = new FuzzyRule(5, ifWindMax, thenESCMinimum);
  fuzzy->addFuzzyRule(fuzzyRule05);
}

void loop()
{
  buttonState = digitalRead(buttonPin);
  int input = analogRead(A2);

  fuzzy->setInput(1, input);
  // Running the Fuzzification
  fuzzy->fuzzify();
  // Running the Defuzzification
  float output = fuzzy->defuzzify(1);

  if (buttonState == HIGH) {
    ESC.write(0);
  } else {
    analogWrite(9,output);
    ESC.write(output);    // Send the signal to the ESC
  }
  // Printing something
```

```
  // Serial.println("\n\n\nEntrance: ");
 // Serial.print("\t\t\tWind: ");
  Serial.print(input);
  Serial.print(' ');
  Serial.println(output);
  Serial.print(' ');
}
```