PROGRAM KENDALI KECEPATAN PUTAR BRUSHLESS DC MOTOR

Oleh :

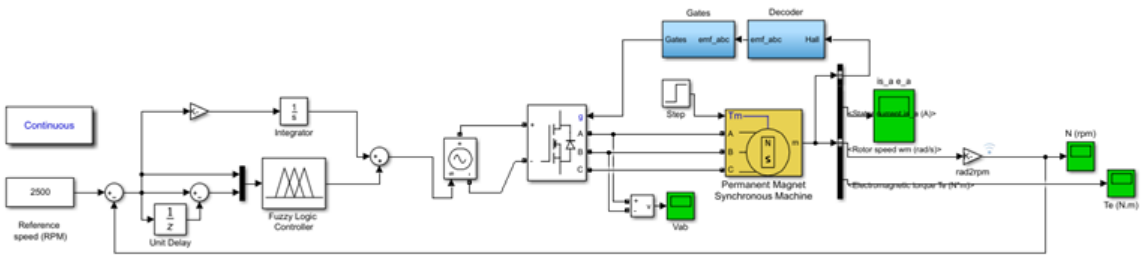
HARI PRASETIJO, S.T., M.T.
DARU TRI NUGROHO,S.T.,MT.
PRISWANTO,S.T.,M.Eng.

Pendahuluan

Dibandingkan motor DC, Motor Brushless DC (BLDC) memiliki keunggulan yaitu tidak menggunakan *brush* dan komutator melainkan menggunakan rangkaian elektronika sebagai kendali motor BLDC, efisiensi tinggi, memiliki umur pakai yang panjang, perawatan yang rendah, serta tingkat kebisingan yang rendah karena putarannya halus. Sistem pada motor BLDC menggunakan konsep rangkaian elektromagnetik yang dipadukan dengan elektromekanik, rangkaian elektronika, sistem sensor dan algoritma kendali.

Kendali kecepatan motor BLDC dapat menggunakan kontrol logika *fuzzy*. FLC *(Fuzzy Logic Control)* terbukti efektif untuk proses yang kompleks, non-linear dan proses yang tidak mungkin diterapkan dengan sistem kontrol berbasis model biasa.

Gambar 1 merupakan rangkaian simulasi sistem pengendalian kecepatan motor *brushlessi* DC dengan menghubungkan beberapa parameter blok yang ada pada *library* simulink menjadi sebuah rangkaian sistem pengendalian kecepatan motor BLDC.



Gambar 1. Rangkaian simulasi menggunakan FLC pada Simulink

Program pengambil keputusan berdasarkan kontrol Logika *Fuzzy* untuk menjalankan simulasi dilakukan menggunakan *software* arduino IDE. Program *Fuzzy Logic Control* yang digunakan beberapa proses, yaitu :

1. Inisialisasi

   Proses inisialisasi berisi pendeklarasian variabel, konfigurasi pin *hardware*, dan pemanggilan *library* yang digunakan untuk menjalankan tugas tertentu

2. Fuzzifikasi Data *Error*, *Delta Error*, serta data *Ouputt* PWM

   Merupakan proses yang digunakan untuk mengubah data yang terbaca oleh pembacaan kecepatan motor oleh modul *hall sensor* menjadi data *fuzzy.*

3. Kaidah aturan *(rule base)*

   Kaidah aturan *(rule base)* merupakan sekumpulan aturan yang dijadikan sebagai dasar pengambilan keputusan oleh sistem *fuzzy.*

4. Defuzzifikasi Data *Output* PWM *Fuzzy*

   Defuzzifikasi merupakan proses yang digunakan untuk mengubah nilai *output fuzzy* yang diperoleh berdasarkan kaidah aturan yang digunakan menjadi nilai tegas *(crisp).*

Berikut merupkan source code lengkap ke-4 proses tersebut :

Program Arduino Kendali Kecepatan Motor Menggunakan *Fuzzy Logic Control* (FLC)

```
#include <Fuzzy.h>

Fuzzy *fuzzy = new Fuzzy();
//FuzzyInput error
FuzzySet *eNB     = new FuzzySet(-2640, -1759, -1759, -880.1);
FuzzySet *eNS     = new FuzzySet(-1759, -880.1, -880.1, 0);
FuzzySet *eZ      = new FuzzySet(-880.1, 0, 0, 880.3);
FuzzySet *ePS     = new FuzzySet(0, 880.3, 880.3, 1759);
FuzzySet *ePB     = new FuzzySet(880.3, 1759, 1759, 2640);

//FuzzyInput derror
FuzzySet *deNB    = new FuzzySet(-2640, -1759, -1759, -880.1);
FuzzySet *deNS    = new FuzzySet(-1759, -880.1, -880.1, 0);
FuzzySet *deZ     = new FuzzySet(-880.1, 0, 0, 880.3);
FuzzySet *dePS    = new FuzzySet(0, 880.3, 880.3, 1759);
FuzzySet *dePB    = new FuzzySet(880.3, 1759, 1759, 2640);

//FuzzyOutput out
FuzzySet *outSL   = new FuzzySet(-102, -102, -51, -25.5);
FuzzySet *outL    = new FuzzySet(-51, -25.5, -12.75, 6.375);
FuzzySet *outS    = new FuzzySet(-6.375, 0, 0, 6.375);
FuzzySet *outC    = new FuzzySet(6.375, 12.75, 25.5, 51);
FuzzySet *outSC   = new FuzzySet(25.5, 51, 102, 102);

float revolutions = 0;
int rpm = 0;
int pinMotor = 9;
int kecepatan = 0;
long  startTime = 0;
long  elapsedTime;
int outputPWM;

int error, last_error = 0, derror, Setpoint;
unsigned long previousMillis = 0;
unsigned long currentMillis;

void setup()
{
  Serial.begin(115200);
  pinMode(2, INPUT_PULLUP);
  kecepatan = 1200; //2640
  Setpoint = kecepatan;
```

```
//FuzzyInput error
FuzzyInput *error = new FuzzyInput (1);

  error->addFuzzySet(eNB);
  error->addFuzzySet(eNS);
  error->addFuzzySet(eZ);
  error->addFuzzySet(ePS);
  error->addFuzzySet(ePB);
  fuzzy->addFuzzyInput(error);

//FuzzyInput derror
FuzzyInput *derror = new FuzzyInput (2);

  derror->addFuzzySet(deNB);
  derror->addFuzzySet(deNS);
  derror->addFuzzySet(deZ);
  derror->addFuzzySet(dePS);
  derror->addFuzzySet(dePB);
  fuzzy->addFuzzyInput(derror);

//FuzzyOutput out
FuzzyOutput *out = new FuzzyOutput (1);

  out->addFuzzySet(outSL);
  out->addFuzzySet(outL);
  out->addFuzzySet(outS);
  out->addFuzzySet(outC);
  out->addFuzzySet(outSC);
  fuzzy->addFuzzyOutput(out);

//--------RULE 1=IF eNB & deNB THEN outSL--------
  FuzzyRuleAntecedent *eNB_deNB = new FuzzyRuleAntecedent();
  eNB_deNB->joinWithAND(eNB, deNB);

  FuzzyRuleConsequent *out_outSL1 = new FuzzyRuleConsequent();
  out_outSL1->addOutput(outSL);

  FuzzyRule *fuzzyRule1 = new FuzzyRule(1, eNB_deNB,
out_outSL1);
  fuzzy->addFuzzyRule(fuzzyRule1);

//--------RULE 2=IF eNB & deNS THEN outSL--------
  FuzzyRuleAntecedent *eNB_deNS = new FuzzyRuleAntecedent();
  eNB_deNS->joinWithAND(eNB, deNS);

  FuzzyRuleConsequent *out_outSL2 = new FuzzyRuleConsequent();
  out_outSL2->addOutput(outSL);

  FuzzyRule *fuzzyRule2 = new FuzzyRule(2, eNB_deNS,
out_outSL2);
  fuzzy->addFuzzyRule(fuzzyRule2);
```

```
//--------RULE 3=IF eNB & deZ THEN outSL--------
  FuzzyRuleAntecedent *eNB_deZ = new FuzzyRuleAntecedent();
  eNB_deZ->joinWithAND(eNB, deZ);

  FuzzyRuleConsequent *out_outSL3 = new FuzzyRuleConsequent();
  out_outSL3->addOutput(outSL);

  FuzzyRule *fuzzyRule3 = new FuzzyRule(3, eNB_deZ,
out_outSL3);
  fuzzy->addFuzzyRule(fuzzyRule3);

//--------RULE 4=IF eNB & dePS THEN outSL--------
  FuzzyRuleAntecedent *eNB_dePS = new FuzzyRuleAntecedent();
  eNB_dePS->joinWithAND(eNB, dePS);

  FuzzyRuleConsequent *out_outSL4 = new FuzzyRuleConsequent();
  out_outSL4->addOutput(outSL);

  FuzzyRule *fuzzyRule4 = new FuzzyRule(4, eNB_dePS,
out_outSL4);
  fuzzy->addFuzzyRule(fuzzyRule4);

//--------RULE 5=IF eNB & dePB THEN outSL--------
  FuzzyRuleAntecedent *eNB_dePB = new FuzzyRuleAntecedent();
  eNB_dePB->joinWithAND(eNB, dePB);

  FuzzyRuleConsequent *out_outSL5 = new FuzzyRuleConsequent();
  out_outSL5->addOutput(outSL);

  FuzzyRule *fuzzyRule5 = new FuzzyRule(5, eNB_dePB,
out_outSL5);
  fuzzy->addFuzzyRule(fuzzyRule5);

//--------RULE 6=IF eNS & deNB THEN outL--------
  FuzzyRuleAntecedent *eNS_deNB = new FuzzyRuleAntecedent();
  eNS_deNB->joinWithAND(eNS, deNB);

  FuzzyRuleConsequent *out_outL6 = new FuzzyRuleConsequent();
  out_outL6->addOutput(outL);

  FuzzyRule *fuzzyRule6 = new FuzzyRule(6, eNS_deNB,
out_outL6);
  fuzzy->addFuzzyRule(fuzzyRule6);

//--------RULE 7=IF eNS & deNS THEN outL--------
  FuzzyRuleAntecedent *eNS_deNS = new FuzzyRuleAntecedent();
  eNS_deNS->joinWithAND(eNS, deNS);

  FuzzyRuleConsequent *out_outL7 = new FuzzyRuleConsequent();
  out_outL7->addOutput(outL);
```

```
  FuzzyRule *fuzzyRule7 = new FuzzyRule(7, eNS_deNS,
out_outL7);
  fuzzy->addFuzzyRule(fuzzyRule7);

//--------RULE 8=IF eNS & deZ THEN outL--------
  FuzzyRuleAntecedent *eNS_deZ = new FuzzyRuleAntecedent();
  eNS_deZ->joinWithAND(eNS, deZ);

  FuzzyRuleConsequent *out_outL8 = new FuzzyRuleConsequent();
  out_outL8->addOutput(outL);

  FuzzyRule *fuzzyRule8 = new FuzzyRule(8, eNS_deZ, out_outL8);
  fuzzy->addFuzzyRule(fuzzyRule8);

//--------RULE 9=IF eNS & dePS THEN outL--------
  FuzzyRuleAntecedent *eNS_dePS = new FuzzyRuleAntecedent();
  eNS_dePS->joinWithAND(eNS, dePS);

  FuzzyRuleConsequent *out_outL9 = new FuzzyRuleConsequent();
  out_outL9->addOutput(outL);

  FuzzyRule *fuzzyRule9 = new FuzzyRule(9, eNS_dePS,
out_outL9);
  fuzzy->addFuzzyRule(fuzzyRule9);

//--------RULE 10=IF eNS & dePB THEN outL--------
  FuzzyRuleAntecedent *eNS_dePB = new FuzzyRuleAntecedent();
  eNS_dePB->joinWithAND(eNS, dePB);

  FuzzyRuleConsequent *out_outL10 = new FuzzyRuleConsequent();
  out_outL10->addOutput(outL);

  FuzzyRule *fuzzyRule10 = new FuzzyRule(10, eNS_dePB,
out_outL10);
  fuzzy->addFuzzyRule(fuzzyRule10);

//--------RULE 11=IF eZ & deNB THEN outSL--------
  FuzzyRuleAntecedent *eZ_deNB = new FuzzyRuleAntecedent();
  eZ_deNB->joinWithAND(eZ, deNB);

  FuzzyRuleConsequent *out_outSL11 = new FuzzyRuleConsequent();
  out_outSL11->addOutput(outSL);

  FuzzyRule *fuzzyRule11 = new FuzzyRule(11, eZ_deNB,
out_outSL11);
  fuzzy->addFuzzyRule(fuzzyRule11);
```

```
//--------RULE 12=IF eZ & deNS THEN outL--------
  FuzzyRuleAntecedent *eZ_deNS = new FuzzyRuleAntecedent();
  eZ_deNS->joinWithAND(eZ, deNS);

  FuzzyRuleConsequent *out_outL12 = new FuzzyRuleConsequent();
  out_outL12->addOutput(outL);

  FuzzyRule *fuzzyRule12 = new FuzzyRule(12, eZ_deNS,
out_outL12);
  fuzzy->addFuzzyRule(fuzzyRule12);

//--------RULE 13=IF eZ & deZ THEN outS--------
  FuzzyRuleAntecedent *eZ_deZ = new FuzzyRuleAntecedent();
  eZ_deZ->joinWithAND(eZ, deZ);

  FuzzyRuleConsequent *out_outS13 = new FuzzyRuleConsequent();
  out_outS13->addOutput(outS);

  FuzzyRule *fuzzyRule13 = new FuzzyRule(13, eZ_deZ,
out_outS13);
  fuzzy->addFuzzyRule(fuzzyRule13);

//--------RULE 14=IF eZ & dePS THEN outC--------
  FuzzyRuleAntecedent *eZ_dePS = new FuzzyRuleAntecedent();
  eZ_dePS->joinWithAND(eZ, dePS);

  FuzzyRuleConsequent *out_outC14 = new FuzzyRuleConsequent();
  out_outC14->addOutput(outC);

  FuzzyRule *fuzzyRule14 = new FuzzyRule(14, eZ_dePS,
out_outC14);
  fuzzy->addFuzzyRule(fuzzyRule14);

//--------RULE 15=IF eZ & dePB THEN outSC--------
  FuzzyRuleAntecedent *eZ_dePB = new FuzzyRuleAntecedent();
  eZ_dePB->joinWithAND(eZ, dePB);

  FuzzyRuleConsequent *out_outSC15 = new FuzzyRuleConsequent();
  out_outSC15->addOutput(outSC);

  FuzzyRule *fuzzyRule15 = new FuzzyRule(15, eZ_dePB,
out_outSC15);
  fuzzy->addFuzzyRule(fuzzyRule15);

//--------RULE 16=IF ePS & deNB THEN outC--------
  FuzzyRuleAntecedent *ePS_deNB = new FuzzyRuleAntecedent();
  ePS_deNB->joinWithAND(ePS, deNB);

  FuzzyRuleConsequent *out_outC16 = new FuzzyRuleConsequent();
  out_outC16->addOutput(outC);
```

```
  FuzzyRule *fuzzyRule16 = new FuzzyRule(16, ePS_deNB,
out_outC16);
  fuzzy->addFuzzyRule(fuzzyRule16);

//--------RULE 17=IF ePS & deNS THEN outC--------
  FuzzyRuleAntecedent *ePS_deNS = new FuzzyRuleAntecedent();
  ePS_deNS->joinWithAND(ePS, deNS);

  FuzzyRuleConsequent *out_outC17 = new FuzzyRuleConsequent();
  out_outC17->addOutput(outC);

  FuzzyRule *fuzzyRule17 = new FuzzyRule(17, ePS_deNS,
out_outC17);
  fuzzy->addFuzzyRule(fuzzyRule17);

//--------RULE 18=IF ePS & deZ THEN outC--------
  FuzzyRuleAntecedent *ePS_deZ = new FuzzyRuleAntecedent();
  ePS_deZ->joinWithAND(ePS, deZ);

  FuzzyRuleConsequent *out_outC18 = new FuzzyRuleConsequent();
  out_outC18->addOutput(outC);

  FuzzyRule *fuzzyRule18 = new FuzzyRule(18, ePS_deZ,
out_outC18);
  fuzzy->addFuzzyRule(fuzzyRule18);

//--------RULE 19=IF ePS & dePS THEN outC--------
  FuzzyRuleAntecedent *ePS_dePS = new FuzzyRuleAntecedent();
  ePS_dePS->joinWithAND(ePS, dePS);

  FuzzyRuleConsequent *out_outC19 = new FuzzyRuleConsequent();
  out_outC19->addOutput(outC);

  FuzzyRule *fuzzyRule19 = new FuzzyRule(19, ePS_dePS,
out_outC19);
  fuzzy->addFuzzyRule(fuzzyRule19);

//--------RULE 20=IF ePS & dePB THEN outC--------
  FuzzyRuleAntecedent *ePS_dePB = new FuzzyRuleAntecedent();
  ePS_dePB->joinWithAND(ePS, dePB);

  FuzzyRuleConsequent *out_outC20 = new FuzzyRuleConsequent();
  out_outC20->addOutput(outC);

  FuzzyRule *fuzzyRule20 = new FuzzyRule(20, ePS_dePB,
out_outC20);
  fuzzy->addFuzzyRule(fuzzyRule20);

//--------RULE 21=IF ePB & deNB THEN outSC--------
  FuzzyRuleAntecedent *ePB_deNB = new FuzzyRuleAntecedent();
  ePB_deNB->joinWithAND(ePB, deNB);
```

```
  FuzzyRuleConsequent *out_outSC21 = new FuzzyRuleConsequent();
  out_outSC21->addOutput(outSC);

  FuzzyRule *fuzzyRule21 = new FuzzyRule(21, ePB_deNB,
out_outSC21);
  fuzzy->addFuzzyRule(fuzzyRule21);

//--------RULE 22=IF ePB & deNS THEN outSC--------
  FuzzyRuleAntecedent *ePB_deNS = new FuzzyRuleAntecedent();
  ePB_deNS->joinWithAND(ePB, deNS);

  FuzzyRuleConsequent *out_outSC22 = new FuzzyRuleConsequent();
  out_outSC22->addOutput(outSC);

  FuzzyRule *fuzzyRule22 = new FuzzyRule(22, ePB_deNS,
out_outSC22);
  fuzzy->addFuzzyRule(fuzzyRule22);

//--------RULE 23=IF ePB & deZ THEN outSC--------
  FuzzyRuleAntecedent *ePB_deZ = new FuzzyRuleAntecedent();
  ePB_deZ->joinWithAND(ePB, deZ);

  FuzzyRuleConsequent *out_outSC23 = new FuzzyRuleConsequent();
  out_outSC23->addOutput(outSC);

  FuzzyRule *fuzzyRule23 = new FuzzyRule(23, ePB_deZ,
out_outSC23);
  fuzzy->addFuzzyRule(fuzzyRule23);

//--------RULE 24=IF ePB & dePS THEN outSC--------
  FuzzyRuleAntecedent *ePB_dePS = new FuzzyRuleAntecedent();
  ePB_dePS->joinWithAND(ePB, dePS);

  FuzzyRuleConsequent *out_outSC24 = new FuzzyRuleConsequent();
  out_outSC24->addOutput(outSC);

  FuzzyRule *fuzzyRule24 = new FuzzyRule(24, ePB_dePS,
out_outSC24);
  fuzzy->addFuzzyRule(fuzzyRule24);

//--------RULE 25=IF ePB & dePB THEN outSC--------
  FuzzyRuleAntecedent *ePB_dePB = new FuzzyRuleAntecedent();
  ePB_dePB->joinWithAND(ePB, dePB);

  FuzzyRuleConsequent *out_outSC25 = new FuzzyRuleConsequent();
  out_outSC25->addOutput(outSC);

  FuzzyRule *fuzzyRule25 = new FuzzyRule(25, ePB_dePB,
out_outSC25);
  fuzzy->addFuzzyRule(fuzzyRule25);
}
```

```
void loop()
{
  revolutions = 0; rpm = 0;
  startTime = millis();
  attachInterrupt(digitalPinToInterrupt(2), interruptFunction,
RISING);
  delay(1000);
  detachInterrupt(2);
  elapsedTime = millis() - startTime;
  if (revolutions > 0)
  {
    rpm = (max(1, revolutions) * 60000) / elapsedTime;
//calculates rpm
  }

  int error = Setpoint - rpm;
  int derror = error - last_error;
  last_error = error;

  fuzzy->setInput(1, error);
  fuzzy->setInput(2, derror);
  fuzzy->fuzzify();

  float output = fuzzy->defuzzify(1);
  outputPWM = outputPWM + (int)output;
  if (outputPWM < 0)
    outputPWM = 0;
  analogWrite(pinMotor,outputPWM);
  Serial.println(rpm);
//  Serial.println(error);
//  Serial.println(derror);
//  Serial.println(output);
  delay(1000);
}
void interruptFunction() //interrupt service routine
{
  currentMillis = millis();
  if (currentMillis - previousMillis >= 18) {
    previousMillis = currentMillis;
    revolutions++;
  }
}
```