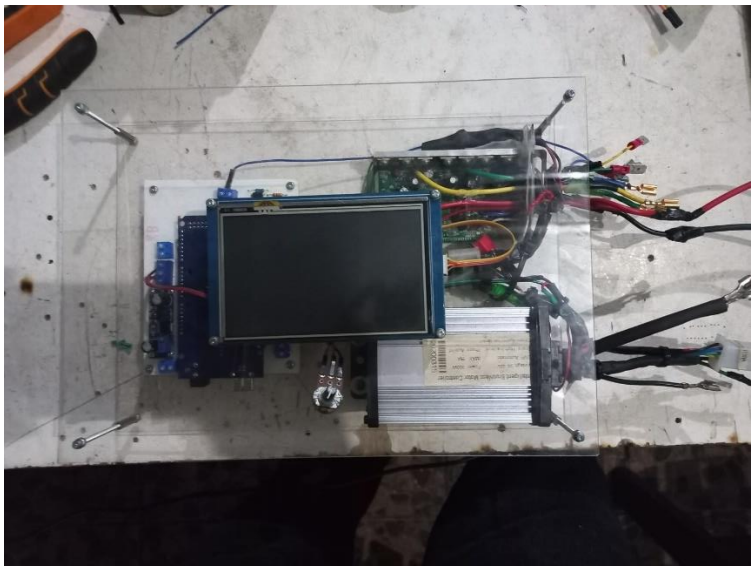
LAMPIRAN

LAMPIRAN 1




Variable Speed Drive
80 %
205
1425
RPM
back
graph

CODE ARDUINO

```
// ================== NEXTION
String data_from_display = "";
String dataIn;
char charArray[20];

// ================== DEKLARASI MULTITASKING
#define INTERVAL_MESSAGE1 100
#define INTERVAL_MESSAGE2 500
#define INTERVAL_MESSAGE3 5000

unsigned long time_1 = 0;
unsigned long time_2 = 0;
unsigned long time_3 = 0;

// ================== DEKLARASI PIN RPM
#define pin_rpm 2
float value = 0;
float rev = 0;
int rpm, setpoint;
int oldtime = 0;
int time;

// ================== DEKLARASI PIN
#define potensio A0
#define pin_button A1
bool prestate = 0;
```

```
int val_pot = 0;
int val_button = 1;
int tampilan = 0;

// ================= DEKLARASI PIN PWM BLDC
#define pin_bldc 9
#define pin_bldc2 6
int pwm_bldc = 0;

// ================ graph
int rpmap ;
int persen;



void isr() //interrupt service routine
{
  rev++;
}


void setup()
{
  Serial2.begin(9600);            //initialize LCD
  Serial.begin(9600);            //initialize LCD

  Serial2.print("page menu");
```

```
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);

  pinMode(potensio, INPUT);
  pinMode(pin_button, INPUT_PULLUP);
  pinMode(pin_bldc, OUTPUT);
  pinMode(pin_bldc2, OUTPUT);
  attachInterrupt(0, isr, FALLING); //attaching the interrupt

  //  String recivedData;
  //  recivedData = read_String(10);

  looping();
}

void loop()
{

}


void looping()
{
program_loop:

  analogWrite(pin_bldc, 0);
  analogWrite(pin_bldc2, 0);
```

```
 if (Serial2.available()) {
  delay (30);

  while (Serial2.available()) {
   data_from_display += char (Serial2.read());
  }

  Serial.println(data_from_display) ;
 }

 stringToCharArray(data_from_display, charArray, sizeof(charArray));
 parseReceivedData(charArray);


 if (data_from_display == "vsd") {
  Serial.println("MODE VSD");
  vsd();
 }

//  if (data_from_display == "pid") {
//    Serial.println("MODE PID");
//    pid_mode();
//  }
//
//  if (data_from_display == "fuzzy") {
//    Serial.println("MODE FUZZY");
//    fuzzy_mode();
```

```
// }

  data_from_display = "";
  Serial.println("menu");
  delay(100);
  goto program_loop;
}

void vsd()
{
m_vsd:
  if (Serial2.available()) {
    delay (30);

    while (Serial2.available()) {
      data_from_display += char (Serial2.read());
    }
    Serial.println(data_from_display) ;
  }

  if (millis() > time_1 + INTERVAL_MESSAGE1) {
    time_1 = millis();

    val_pot = analogRead(potensio);
    persen = map(val_pot, 0, 1023, 0, 100);
    pwm_bldc = map(val_pot, 0, 1023, 0, 255);
    analogWrite(pin_bldc, pwm_bldc);
    analogWrite(pin_bldc2, pwm_bldc);
```

```
  detachInterrupt(0);          //detaches the interrupt
  time = millis() - oldtime;    //finds the time
  rpm = (rev / time) * 6000;   //calculates rpm 60000
  oldtime = millis();          //saves the current time
  rev = 0;
  rpmap = map(rpm, 0, 1425, 0, 255);
  attachInterrupt(0, isr, RISING);

  Serial2.print("va0.val=");
  Serial2.print(rpmap);
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
}

if (millis() > time_2 + INTERVAL_MESSAGE2) {
  time_2 = millis();

  Serial.print("MODE VSD >> ");
  Serial.print("PWM: ");
  Serial.print(pwm_bldc);
  Serial.print("\t%: ");
  Serial.print(persen);
  Serial.print("\tRPM:  ");
  Serial.println(rpm);

  Serial2.print("rpm.txt=");
```

```
  Serial2.print("\"");
  Serial2.print(rpm);
  Serial2.print("\"");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);

  Serial2.print("persen.txt=");
  Serial2.print("\"");
  Serial2.print(persen);
  Serial2.print("\"");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);

  Serial2.print("pwm.txt=");
  Serial2.print("\"");
  Serial2.print(pwm_bldc);
  Serial2.print("\"");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
}

if (data_from_display == "back") {
  Serial.println("BACK");
  data_from_display = "";
  delay(1000);
```

```
    looping();
  }



  data_from_display = "";
  goto m_vsd;
}

void stringToCharArray(const String& str, char* charArray, size_t
bufferSize) {
  str.toCharArray(charArray, bufferSize);
}

void parseReceivedData(const char* data)
{
  if (strncmp(data, "set", 3) == 0) {
    int value = atoi(data + 3); // Mengambil angka setelah "set"

    // Cetak hasil parsing
    Serial.print("Nilai yang diambil: ");
    Serial.println(value);
    setpoint = value;
  }
  else {
    //    Serial.println("Format data tidak valid");
  }
}
```