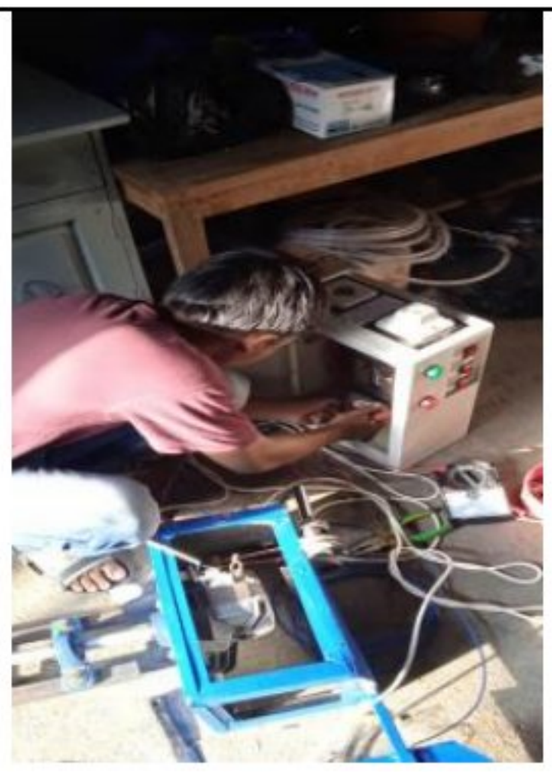
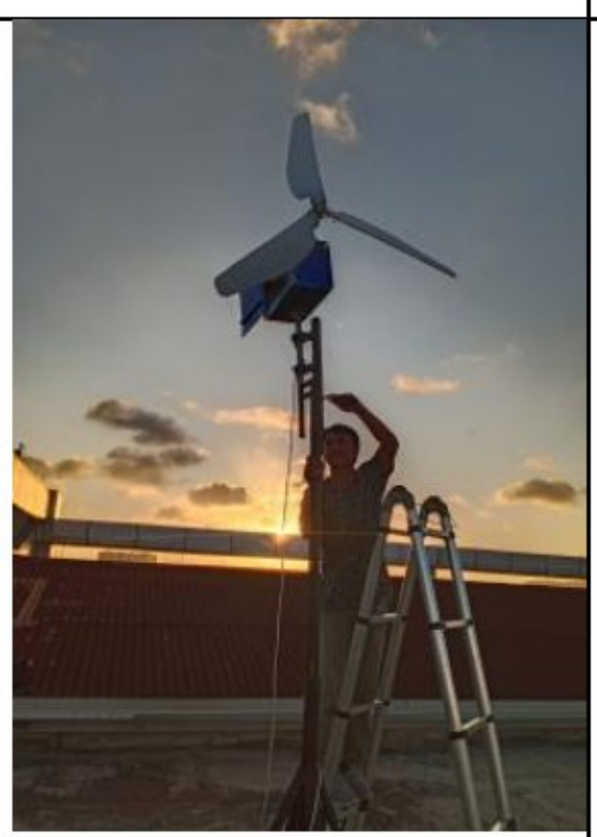
LAMPIRAN A
Dokumentasi Kegiatan

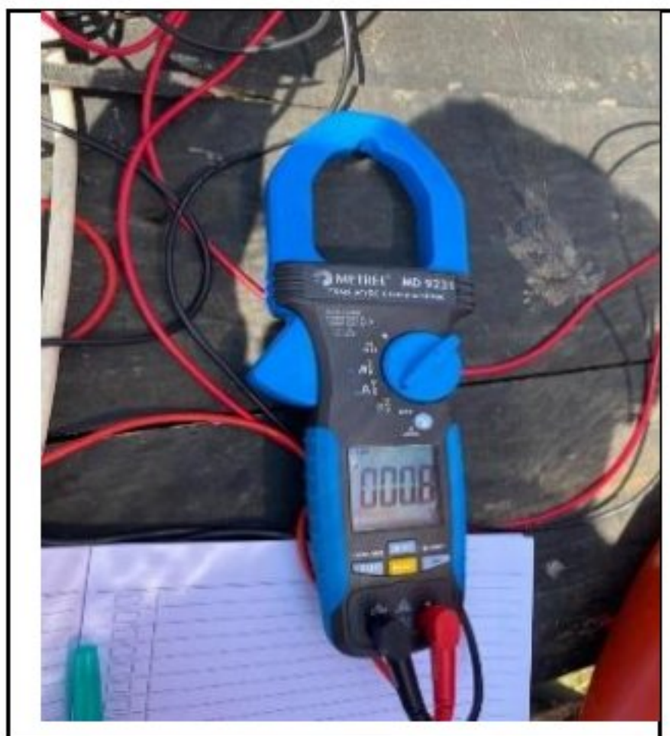
Perakitan Mekanikal



LAMPIRAN B
Pengambilan Data

Pengambilan Data Pengukuran Hari Ke-1		
		
		
		


METREL MD 9231
0008


METREL MD 9231
TRMS AC/DC Clamp-on DMM
OFF
ZERO
0000
PEAK/MIN
SELECT
COM


METREL MD 9231
TRMS AC/DC Clamp-on DMM
OFF
ZERO
000
PEAK/MIN
SELECT
COM


LM-8010


LM-8010


LM-8010


TYPE K
Humidity
LM-8010


LM-8010


TYPE K
Humidity
LM-8010

Lampiran C
Berikut Kode Pemograman

```
include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecureBearSSL.h>
#include <Adafruit_INA219.h>
#include <BlynkSimpleEsp8266.h>

#define BLYNK_PRINT Serial

#include "certs.h"

#define BLYNK_TEMPLATE_ID "TMPL6EYl8kuuL"
#define BLYNK_TEMPLATE_NAME "Monitoring Control Aji"
#define BLYNK_AUTH_TOKEN "Iz3QGT7Tg9uGnRpLVhFIynGzH3jXSHQS"


#define LED_PIN 2 //D2
#define LM393_PIN 14//pin sensor speed

String serverName = "https://script.google.com/macros/s/AKfycbwa8KFl6BDFyAu5KNyI07i4s4GOGSGBMoLCkaS--HGLgstOqoOcKPvs193zye9Bd1-e/exec";

const char auth[] = "Iz3QGT7Tg9uGnRpLVhFIynGzH3jXSHQS"; //TOKEN
const char ssid[] = "nandar pnc";
const char pass[] = "bismillah123";
float vVoltage, vCurrent, vPower, vSpeed, offsetRPM=1;;
unsigned long lastTime = 0, sendTime = 5000;
unsigned long speedTime= 0, vRpm=0;
unsigned long lastDisplayTime=0, displayTime=1000;
bool stateReady=false;

Adafruit_INA219 ina219;

void setup() {
```

```
Blynk.begin(auth, ssid, pass);

Serial.begin(9600); delay(100);
Serial.println("Inisialisasi....");
pinMode(LM393_PIN,INPUT_PULLUP);
//attachInterrupt(digitalPinToInterrupt(LM393_PIN),count,RISING);
if (! ina219.begin()) {
  Serial.println("Failed to find INA219 chip");
  while (1) {
    delay(10);
  }
}

pinMode(LED_PIN, OUTPUT); digitalWrite(LED_PIN, LOW);
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
int count = 0;
while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    digitalWrite(LED_PIN, HIGH); delay(250);
    digitalWrite(LED_PIN, LOW); delay(250);
count++;
    if (count >= 20)ESP.restart();
  }
  Serial.println("Connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  Blynk.run();


vVoltage = ina219.getBusVoltage_V()*15;
vCurrent = ina219.getCurrent_mA();
vPower   = ina219.getPower_mW()/ 1000;
vRpm     = vRpm*offsetRPM;
```

```
//   vVoltage = random(1,100);vCurrent = random(1,100);vPower    =
random(1,100);
  if ((WiFi.status() != WL_CONNECTED)) {
    reconnecting();
  }
  else {
    if (millis() > lastTime + sendTime) {
      updateDB();
      lastTime = millis();
    }
  }
  if(millis() > lastDisplayTime + displayTime){
Serial.println("Voltage\t: " + String(vVoltage) + " V");
    Serial.println("Current\t: " + String(vCurrent) + " mA");
    Serial.println("Power\t: " + String(vPower) + " W");
    Serial.println("Speed\t: " + String(vRpm) + " RpM");
    Serial.println("=================================");


    Blynk.virtualWrite(V0,String(vVoltage));
    Blynk.virtualWrite(V2, String(vPower));
    Blynk.virtualWrite(V1, String(vCurrent));
    Blynk.virtualWrite(V4, String(vRpm));
  }
  if(digitalRead(LM393_PIN)==LOW && stateReady == true){
    countRPM();
    stateReady=false;
  }
  if(digitalRead(LM393_PIN)==HIGH){
    stateReady=true;
  }
  delay(1);
}

void countRPM(){
  unsigned long endTime = millis();
  unsigned long duration = endTime-speedTime;
  vRpm = 60000/duration;
  speedTime = millis();
```

```
}
void reconnecting() {
  Serial.print("reConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  int count = 0;
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    digitalWrite(LED_PIN, HIGH); delay(250);
    digitalWrite(LED_PIN, LOW); delay(250);
    count++;
    if (count >= 20)ESP.restart();
   }
  Serial.println("Connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void updateDB() {
  if ((WiFi.status() == WL_CONNECTED)) {
    std::unique_ptr<BearSSL::WiFiClientSecure>client(new
BearSSL::WiFiClientSecure);
    client->setInsecure();

    HTTPClient https;

    Serial.print("[HTTPS] begin...\n");
    String serverPath = serverName + "?i1=" + String(vVoltage) + "&i2="
+ String(vCurrent) + "&i3=" + String(vPower)+ "&i4=" + String(vRpm);
    Serial.println(serverPath);
if (https.begin(*client, serverPath)) { // HTTPS

      Serial.print("[HTTPS] GET...\n");
      int httpCode = https.GET();
      if (httpCode > 0) {
       Serial.printf("[HTTPS] GET... code: %d\n", httpCode);
       if    (httpCode    ==    HTTP_CODE_OK    ||    httpCode    ==
HTTP_CODE_MOVED_PERMANENTLY) {
        //        String payload = https.getString();
        //        Serial.println(payload);
```

```
        Serial.println("OK");
       }
      } else {
       Serial.printf("[HTTPS]      GET...      failed,      error:      %s\n",
https.errorToString(httpCode).c_str());
      }
       https.end();
      } else {
       Serial.printf("[HTTPS] Unable to connect\n");
      }
    }
    else {
     Serial.println("CANNOT SEND TO GOOGLE SHEET ERROR WIFI
CONNECTION");
    }
}
```