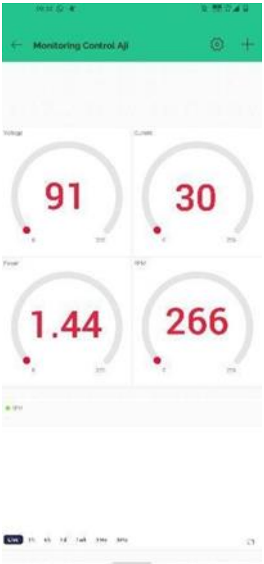
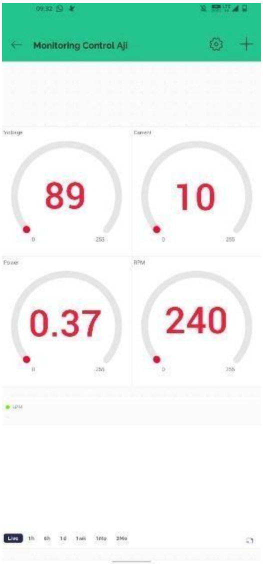
LAMPIRAN A

(Foto Kegiatan)

- Pembuatan Rangka Turbin Ulir

- Pengambilan Data Tegangan Dan Arus

- Pembacaan Pengukuran Pada Aplikasi *Bylnk*

 Monitoring Control Aji Voltage 91 Current 30 Power 1.44 RPM 266	 Monitoring Control Aji Voltage 89 Current 10 Power 0.37 RPM 240

LAMPIRAN B

(Kode Pemrograman)

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

#include "certs.h"
#define BLYNK_TEMPLATE_ID "TMPL6caTzbHJy" #define
BLYNK_TEMPLATE_NAME "Quickstart Template" #define
BLYNK_AUTH_TOKEN "QkjHBc76PJy3JaCHnzIKau8NwLbC7saq"
#define BLYNK_FIRMWARE_VERSION "0.1.0"
//#define APP_DEBUG #define BLYNK_PRINT Serial

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

String serverName =
"https://script.google.com/macros/s/AKfycbzbYlOOLwZ8Nvr
qoJNMHsgjI5hB0VR1M_gxuaBihorVrhu3Vy5Q1vYTpH6Cqh_VCith/e xec";

const char auth[] = "QkjHBc76PJy3JaCHnzIKau8NwLbC7saq";          //TOKEN
const char ssid[] = "zona nyaman";
const char pass[] = "eksekutif";
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),co unt,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() {
vVoltage = ina219.getBusVoltage_V()*8.6; vCurrent = ina219.getCurrent_mA()
; vPower= vVoltage*vCurrent;
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,vVoltage);
Blynk.virtualWrite(V2, vPower); Blynk.virtualWrite(V1, vCurrent);
Blynk.virtualWrite(V3, 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(ne
w 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");
}
}
```