LAMPIRAN A

SPESIFIKASI SENSOR GPS Neo-Ublox-M8

Parameter	Symbol	Module	Min	Typ	Max	Units	Condition
Power supply voltage	VCC	NEO-6G	1.75	1.8	1.95	V	
		NEO-6Q/M NEO-6P/V/T	2.7	3.0	3.6	V	
Supply voltage USB	VDDUSB	All	3.0	3.3	3.6	V	
Backup battery voltage	V_BCKP	All	1.4		3.6	V	
Backup battery current	I_BCKP	All		22		µA	V_BCKP = 1.8 V, VCC = 0V
Input pin voltage range	Vin	All	0		VCC	V	
Digital IO Pin Low level input voltage	Vil	All	0		0.2*VCC	V	
Digital IO Pin High level input voltage	Vih	All	0.7*VCC		VCC	V	
Digital IO Pin Low level output voltage	Vol	All			0.4	V	Iol=4mA
Digital IO Pin High level output voltage	Voh	All	VCC -0.4			V	Ioh=4mA
USB_DM, USB_DP	VinU	All	Compatible with USB with 22 Ohms series resistance				
VCC_RF voltage	VCC_RF	All		VCC-0.1		V	
VCC_RF output current	ICC_RF	All			50	mA	
Antenna gain	Gant	All			50	dB	
Receiver Chain Noise Figure	NFtot	All		3.0		dB	
Operating temperature	Topr	All	-40		85	°C	

LAMPIRAN B

SPESIFIKASI KENDARAAN MOBIL LISTRIK

DATA LOG

TATA NEXON EV

Price	Rs 16.25 lakh (ex-showroom, India)
Warranty	3 years or 1,25,000km + 8 years or 1,60,000km on battery and motor

BATTERY

Rating	30.2kWh
Type	Lithium-ion
Voltage	320V
Motor type	Permanent magnet synchronous motor
Power	129hp
Torque	245Nm
Power to weight	92hp per tonne
Torque to weight	175Nm per tonne

TRANSMISSION

Type	Single-speed automatic
Final drive ratio	9.1:1

CHASSIS & BODY

Construction	Five-door, monocoque SUV
Weight	1400kg
Tyres	215/60 R16
Spare	215/60 R16

SUSPENSION

Front	Independent, MacPherson strut with coil springs
Rear	Torsion beam with hydraulic shock absorbers, coil springs

STEERING

Type	Rack and pinion
Type of power assist	Electric
Turning circle	10.2m

BRAKES

Front	Disc
Rear	Drum
Anti-lock	Yes

RANGE

TEST

City	216km per charge
Highway	201km per charge

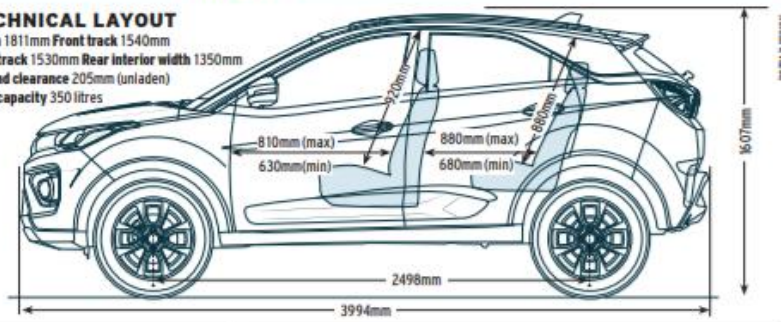
EQUIPMENT CHECK LIST

ESP	NA
Airbags	2
Sunroof	■
Cruise control	NA
Touchscreen	■
Android Auto	■
Apple CarPlay	■
LED headlamps	NA
Climate control	■
Rear AC vents	■
Wireless phone charging	NA

■ = Available, NA=Not Available,

TECHNICAL LAYOUT

Width 1811mm **Front track** 1540mm
Rear track 1530mm **Rear interior width** 1350mm
Ground clearance 205mm (unladen)
Boot capacity 350 litres



ACCELERATION

Kph	Sport (sec)	D (sec)
0-10	0.80	1.13
0-20	1.51	2.28
0-30	2.19	3.45
0-40	2.87	4.68
0-50	3.55	5.95
0-60	4.36	7.32
0-70	5.33	8.95
0-80	6.47	10.89
0-90	7.80	13.23
0-100	9.30	16.14
0-110	11.04	19.62
0-120	16.12	-

BRAKING

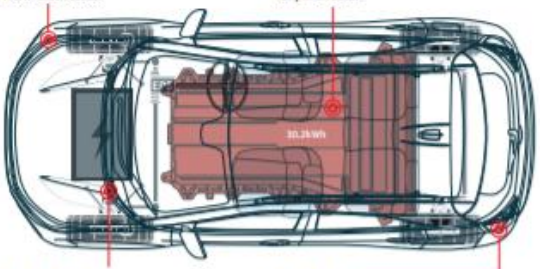
80-0kph	27.34m, 2.44s

NOISE LEVEL (dB)

Idle	33.8
Idle with AC at half	58.1
50kph in 4th gear	61.7
80kph in top gear	67.5

Five starrer
It's a modified version of a platform that has secured a solid five stars in Global NCAP crash tests.

IP67 rated
Its battery pack has been submerged one metre underwater for half an hour without any ill effects.



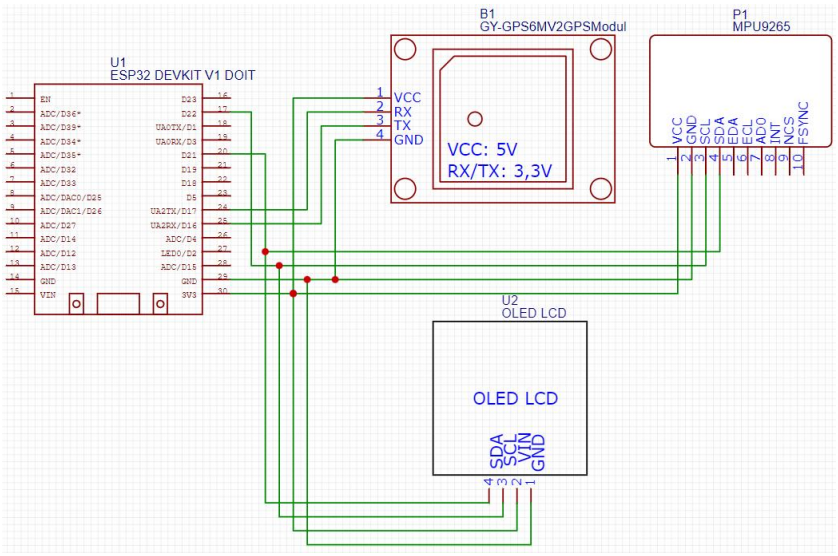
Equal distribution
With a 50:50 weight distribution across its axles, this one feels very sporty from behind the wheel.

4mm lower
The battery is tucked beneath the cabin, yet the ground clearance is only 4mm lower than the ICE version.

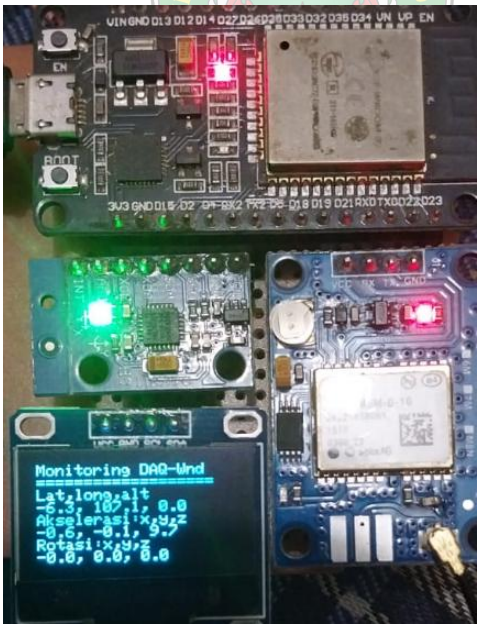
EV vs DIESEL AUTOMATIC, OWNERSHIP COST FOR 3 YEARS

	NEXON EV	NEXON D AMT	COMMENTS
VARIANT	XZ+ Lux	XZA+ (O)	
Ex-showroom Delhi	1625000	1250000	
0.75% Tax Collected at Source	12188	9375	
Registration + road tax	5000	163100	Full waiver of road tax for EV
Insurance	50863	39125	
On-road price	1693050	1461600	
Delhi state govt. subsidy	-150000	-	To be availed from the state govt. directly, not from the dealer.
Final buy price (A)	**1543050**	**1461600**	
Electricity / fuel cost per unit	8.00	73.87	Rs. 8 per unit considered as average cost of electricity Pan India.
Running cost per km	1.16	4.62	For EV - 240 (i.e. 8*30kWh) divide by 208 (range); For diesel - 73.87/16(kpl)
Running cost for 10000km	11615	46200	Cost per km* 10000
Running cost after 3 years (B)	**34846**	**138600**	
ANNUAL SERVICE COSTS			
Year 1	3279	3562	
Year 2	6046	8245	
Year 3	4250	7922	

LAMPIRAN C
DIAGRAM SKEMATIK DAN PROGRAM



Gambar B.1 Diagram skematik rangkaian



Program ESP32

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BusIO_Register.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include <WiFi.h>

const char* ssid = "Pena";
const char* password = "wandaadib";
const char* pub_topic_gps_enc = "publish_e_v_ta_gps_enc";
const char* pub_topic_gps_date = "publish_e_v_ta_gps_date";
const char* pub_topic_gps_year = "publish_e_v_ta_gps_year";
const char* pub_topic_gps_month = "publish_e_v_ta_gps_month";
const char* pub_topic_gps_day = "publish_e_v_ta_gps_day";
const char* pub_topic_gps_time = "publish_e_v_ta_gps_time";
const char* pub_topic_gps_time_hour = "publish_e_v_ta_gps_time_hour";
const char* pub_topic_gps_time_min = "publish_e_v_ta_gps_time_min";
const char* pub_topic_gps_time_sec = "publish_e_v_ta_gps_time_sec";
const char* pub_topic_gps_time_cen_sec = "publish_e_v_ta_gps_time_cen_sec";
const char* pub_topic_gps_speed = "publish_e_v_ta_gps_speed";
const char* pub_topic_gps_speed_knots = "publish_e_v_ta_gps_speed_knots";
const char* pub_topic_gps_speed_mph = "publish_e_v_ta_gps_speed_mph";
const char* pub_topic_gps_speed_mps = "publish_e_v_ta_gps_speed_mps";
const char* pub_topic_gps_speed_kmph = "publish_e_v_ta_gps_speed_kmph";
```

```
const char* pub_topic_gps_course_raw = "publish_e_v_ta_gps_cour_raw";
const char* pub_topic_gps_course_raw_deg = "publish_e_v_ta_gps_cour_raw_deg";
const char* pub_topic_gps_satelite = "publish_e_v_ta_gps_satelite";
const char* pub_topic_gps_hdop = "publish_e_v_ta_gps_hdop";
const char* pub_topic_gps_latitude = "publish_e_v_ta_latitude"; // publish/username/apiKeyIn
const char* pub_topic_gps_latitude_raw = "publish_e_v_ta_gps_lat_raw";
const char* pub_topic_gps_latitude_raw_deg = "publish_e_v_ta_gps_lat_raw_deg";
const char* pub_topic_gps_latitude_raw_bill = "publish_e_v_ta_gps_lat_raw_bill";
const char* pub_topic_gps_longitude = "publish_e_v_ta_longitude";
const char* pub_topic_gps_longitude_raw = "publish_e_v_ta_gps_long_raw";
const char* pub_topic_gps_longitude_raw_deg = "publish_e_v_ta_gps_long_deg";
const char* pub_topic_gps_longitude_raw_bill = "publish_e_v_ta_gps_long_bill";
const char* pub_topic_gps_altitude = "publish_e_v_ta_altitude";
const char* pub_topic_gps_altitude_meter = "publish_e_v_ta_gps_alt_meter";
const char* pub_topic_gps_altitude_mil = "publish_e_v_ta_gps_alt_mil";
const char* pub_topic_gps_altitude_km = "publish_e_v_ta_gps_alt_km";
const char* pub_topic_gps_altitude_feet = "publish_e_v_ta_gps_alt_feet";
const char* pub_topic_aks_x = "publish_e_v_ta_gyro_akselerasi_x";
const char* pub_topic_aks_y = "publish_e_v_ta_gyro_akselerasi_y";
const char* pub_topic_aks_z = "publish_e_v_ta_gyro_akselerasi_z";
const char* pub_topic_rts_x = "publish_e_v_ta_gyro_rotasi_x";
const char* pub_topic_rts_y = "publish_e_v_ta_gyro_rotasi_y";
const char* pub_topic_rts_z = "publish_e_v_ta_gyro_rotasi_z";
const char* pub_topic_temp = "publish_e_v_ta_gyro_temperature";
```

```
const unsigned int writeInterval = 25000;
static const int RXPin = 16, TXPin = 17;
static const uint32_t GPSBaud = 9600;
const char* mqtt_server = "broker.mqtt-dashboard.com";
unsigned int mqtt_port = 1883;
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE          (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

Adafruit_MPU6050 mpu;
WiFiClient espClient;
PubSubClient client(espClient);
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire);

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");
```

```
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
   Serial.print((char)payload[i]);
  }
  Serial.println();
}

void main_gyro(){
  while (!Serial)
    delay(10);
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
```

```
  Serial.println("+-4G");
  break;
case MPU6050_RANGE_8_G:
  Serial.println("+-8G");
  break;
case MPU6050_RANGE_16_G:
  Serial.println("+-16G");
  break;
}
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
Serial.print("Gyro range set to: ");
switch (mpu.getGyroRange()) {
case MPU6050_RANGE_250_DEG:
  Serial.println("+- 250 deg/s");
  break;
case MPU6050_RANGE_500_DEG:
  Serial.println("+- 500 deg/s");
  break;
case MPU6050_RANGE_1000_DEG:
  Serial.println("+- 1000 deg/s");
  break;
case MPU6050_RANGE_2000_DEG:
  Serial.println("+- 2000 deg/s");
  break;
}
mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
  Serial.println("260 Hz");
  break;
```

```
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }
  Serial.println("");
  delay(100);
}
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.publish("outTopic", "hello world");
      client.subscribe("inTopic");
```

```
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void control_program() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  if (gps.location.isValid()){
    double gps_enc = (gps.encode(ss.read()));
    double gps_date = (gps.date.value());
    double gps_year = (gps.date.year());
    double gps_month = (gps.date.month());
    double gps_day = (gps.date.day());
    double gps_time = (gps.time.value());
    double gps_time_hour = (gps.time.hour());
    double gps_time_min = (gps.time.minute());
    double gps_time_sec = (gps.time.second());
    double gps_time_cen_sec = (gps.time.centisecond());
    double gps_speed = (gps.speed.value());
    double gps_speed_knots = (gps.speed.knots());
    double gps_speed_mph = (gps.speed.mph());
    double gps_speed_mps = (gps.speed.mps());
    double gps_speed_kmph = (gps.speed.kmph());
    double gps_course_raw = (gps.course.value());
    double gps_course_raw_deg = (gps.course.deg());
    double gps_satelite = (gps.satellites.value());
```

```
    double gps_hdop = (gps.hdop.value());
    double latitude = (gps.location.lat());
    double latitude_raw = (gps.location.rawLat().negative);
    double latitude_raw_deg = (gps.location.rawLat().deg);
    double latitude_raw_bill = (gps.location.rawLat().billionths);
    double longitude = (gps.location.lng());
    double longitude_raw = (gps.location.rawLng().negative);
    double longitude_raw_deg = (gps.location.rawLng().deg);
    double longitude_raw_bill = (gps.location.rawLng().billionths);
    double altitude = (gps.altitude.value());
    double altitude_meter = (gps.altitude.meters());
    double altitude_mil = (gps.altitude.miles());
    double altitude_km = (gps.altitude.kilometers());
    double altitude_feet = (gps.altitude.feet());
    double akselerasi_x = a.acceleration.x;
    double akselerasi_y = a.acceleration.y;
    double akselerasi_z = a.acceleration.z;
    double rotasi_x = g.gyro.x;
    double rotasi_y = g.gyro.y;
    double rotasi_z = g.gyro.z;
    double temp_gyro = temp.temperature;
//_____________________________________________________________________
_____
    char mqtt_payload_gps_enc[50] = "";
    char mqtt_payload_gps_date[50] = "";
    char mqtt_payload_gps_year[50] = "";
    char mqtt_payload_gps_month[50] = "";
    char mqtt_payload_gps_day[50] = "";
    char mqtt_payload_gps_time[50] = "";
    char mqtt_payload_gps_time_hour[50] = "";
    char mqtt_payload_gps_time_min[50] = "";
```

```
char mqtt_payload_gps_time_sec[50] = "";
char mqtt_payload_gps_time_cen_sec[50] = "";
char mqtt_payload_gps_speed[50] = "";
char mqtt_payload_gps_speed_knots[50] = "";
char mqtt_payload_gps_speed_mph[50] = "";
char mqtt_payload_gps_speed_mps[50] = "";
char mqtt_payload_gps_speed_kmph[50] = "";
char mqtt_payload_gps_course_raw[50] = "";
char mqtt_payload_gps_course_raw_deg[50] = "";
char mqtt_payload_gps_satelite[50] = "";
char mqtt_payload_gps_hdop[50] = "";
char mqtt_payload_gps_latitude[50] = "";
char mqtt_payload_gps_latitude_raw[50] = "";
char mqtt_payload_gps_latitude_raw_deg[50] = "";
char mqtt_payload_gps_latitude_raw_bill[50] = "";
char mqtt_payload_gps_longitude[50] = "";
char mqtt_payload_gps_longitude_raw[50] = "";
char mqtt_payload_gps_longitude_raw_deg[50] = "";
char mqtt_payload_gps_longitude_raw_bill[50] = "";
char mqtt_payload_gps_altitude[50] = "";
char mqtt_payload_gps_altitude_meter[50] = "";
char mqtt_payload_gps_altitude_mil[50] = "";
char mqtt_payload_gps_altitude_km[50] = "";
char mqtt_payload_gps_altitude_feet[50] = "";
char mqtt_payload_akselerasi_x[50] = "";
char mqtt_payload_akselerasi_y[50] = "";
char mqtt_payload_akselerasi_z[50] = "";
char mqtt_payload_rotasi_x[50] = "";
char mqtt_payload_rotasi_y[50] = "";
char mqtt_payload_rotasi_z[50] = "";
char mqtt_payload_temp[50] = "";
```

```
//_____________________________________________________________________
_____

  snprintf (mqtt_payload_gps_enc, 50, "%lf", gps_enc);
  Serial.print("Publish message gps-enc: ");
  Serial.println(mqtt_payload_gps_enc);
  client.publish(pub_topic_gps_enc, mqtt_payload_gps_enc);

  snprintf (mqtt_payload_gps_date, 50, "%lf", gps_date);
  Serial.print("Publish message gps-date: ");
  Serial.println(mqtt_payload_gps_date);
  client.publish(pub_topic_gps_date, mqtt_payload_gps_date);

  snprintf (mqtt_payload_gps_year, 50, "%lf", gps_year);
  Serial.print("Publish message gps-year: ");
  Serial.println(mqtt_payload_gps_year);
  client.publish(pub_topic_gps_year, mqtt_payload_gps_year);

  snprintf (mqtt_payload_gps_month, 50, "%lf", gps_month);
  Serial.print("Publish message gps-month: ");
  Serial.println(mqtt_payload_gps_month);
  client.publish(pub_topic_gps_month, mqtt_payload_gps_month);

  snprintf (mqtt_payload_gps_day, 50, "%lf", gps_day);
  Serial.print("Publish message gps-day: ");
  Serial.println(mqtt_payload_gps_day);
  client.publish(pub_topic_gps_day, mqtt_payload_gps_day);

  snprintf (mqtt_payload_gps_time, 50, "%lf", gps_time);
  Serial.print("Publish message gps-time: ");
  Serial.println(mqtt_payload_gps_time);
```

```
client.publish(pub_topic_gps_time, mqtt_payload_gps_time);

snprintf (mqtt_payload_gps_time_hour, 50, "%lf", gps_time_hour);
Serial.print("Publish message gps-time-hour: ");
Serial.println(mqtt_payload_gps_time_hour);
client.publish(pub_topic_gps_time_hour, mqtt_payload_gps_time_hour);

snprintf (mqtt_payload_gps_time_min, 50, "%lf", gps_time_min);
Serial.print("Publish message gps-time-min: ");
Serial.println(mqtt_payload_gps_time_min);
client.publish(pub_topic_gps_time_min, mqtt_payload_gps_time_min);

snprintf (mqtt_payload_gps_time_sec, 50, "%lf", gps_time_sec);
Serial.print("Publish message gps-time-sec: ");
Serial.println(mqtt_payload_gps_time_sec);
client.publish(pub_topic_gps_time_sec, mqtt_payload_gps_time_sec);

snprintf (mqtt_payload_gps_time_cen_sec, 50, "%lf", gps_time_cen_sec);
Serial.print("Publish message gps-time-cen-min: ");
Serial.println(mqtt_payload_gps_time_cen_sec);
client.publish(pub_topic_gps_time_cen_sec, mqtt_payload_gps_time_cen_sec);

snprintf (mqtt_payload_gps_speed, 50, "%lf", gps_speed);
Serial.print("Publish message gps-speed: ");
Serial.println(mqtt_payload_gps_speed);
client.publish(pub_topic_gps_speed, mqtt_payload_gps_speed);

snprintf (mqtt_payload_gps_speed_knots, 50, "%lf", gps_speed_knots);
Serial.print("Publish message gps-speed-knots: ");
Serial.println(mqtt_payload_gps_speed_knots);
client.publish(pub_topic_gps_speed_knots, mqtt_payload_gps_speed_knots);
```

```
    snprintf (mqtt_payload_gps_speed_mph, 50, "%lf", gps_speed_mph);
    Serial.print("Publish message gps-speed-mph: ");
    Serial.println(mqtt_payload_gps_speed_mph);
    client.publish(pub_topic_gps_speed_mph, mqtt_payload_gps_speed_mph);

    snprintf (mqtt_payload_gps_speed_mps, 50, "%lf", gps_speed_mps);
    Serial.print("Publish message gps-speed-mps: ");
    Serial.println(mqtt_payload_gps_speed_mps);
    client.publish(pub_topic_gps_speed_mps, mqtt_payload_gps_speed_mps);

    snprintf (mqtt_payload_gps_speed_kmph, 50, "%lf", gps_speed_kmph);
    Serial.print("Publish message gps-speed-kmph: ");
    Serial.println(mqtt_payload_gps_speed_kmph);
    client.publish(pub_topic_gps_speed_kmph, mqtt_payload_gps_speed_kmph);

    snprintf (mqtt_payload_gps_course_raw, 50, "%lf", gps_course_raw);
    Serial.print("Publish message gps-course-raw: ");
    Serial.println(mqtt_payload_gps_course_raw);
    client.publish(pub_topic_gps_course_raw, mqtt_payload_gps_course_raw);

    snprintf (mqtt_payload_gps_course_raw_deg, 50, "%lf", gps_course_raw_deg);
    Serial.print("Publish message gps-course-raw-deg: ");
    Serial.println(mqtt_payload_gps_course_raw_deg);
    client.publish(pub_topic_gps_course_raw_deg,
mqtt_payload_gps_course_raw_deg);

    snprintf (mqtt_payload_gps_satelite, 50, "%lf", gps_satelite);
    Serial.print("Publish message gps-satelite: ");
    Serial.println(mqtt_payload_gps_satelite);
    client.publish(pub_topic_gps_satelite, mqtt_payload_gps_satelite);
```

```
    snprintf (mqtt_payload_gps_hdop, 50, "%lf", gps_hdop);
    Serial.print("Publish message gps-hdop: ");
    Serial.println(mqtt_payload_gps_hdop);
    client.publish(pub_topic_gps_hdop, mqtt_payload_gps_hdop);
//_____________________________________________________________________
_______

    snprintf (mqtt_payload_gps_latitude, 50, "%lf", latitude);
    Serial.print("Publish message gps-lat: ");
    Serial.println(mqtt_payload_gps_latitude);
    client.publish(pub_topic_gps_latitude, mqtt_payload_gps_latitude);

    snprintf (mqtt_payload_gps_latitude_raw, 50, "%lf", latitude_raw);
    Serial.print("Publish message gps-lat-raw: ");
    Serial.println(mqtt_payload_gps_latitude_raw);
    client.publish(pub_topic_gps_latitude_raw, mqtt_payload_gps_latitude_raw);

    snprintf (mqtt_payload_gps_latitude_raw_deg, 50, "%lf", latitude_raw_deg);
    Serial.print("Publish message gps-lat-raw-deg: ");
    Serial.println(mqtt_payload_gps_latitude_raw_deg);
    client.publish(pub_topic_gps_latitude_raw_deg,
mqtt_payload_gps_latitude_raw_deg);

    snprintf (mqtt_payload_gps_latitude_raw_bill, 50, "%lf", latitude_raw_bill);
    Serial.print("Publish message gps-lat-raw-bill: ");
    Serial.println(mqtt_payload_gps_latitude_raw_bill);
    client.publish(pub_topic_gps_latitude_raw_bill,
mqtt_payload_gps_latitude_raw_bill);

//_____________________________________________________________________
```

```
_____

    snprintf (mqtt_payload_gps_longitude, 50, "%lf", longitude);
    Serial.print("Publish message gps-long: ");
    Serial.println(mqtt_payload_gps_longitude);
    client.publish(pub_topic_gps_longitude, mqtt_payload_gps_longitude);

    snprintf (mqtt_payload_gps_longitude_raw, 50, "%lf", longitude_raw);
    Serial.print("Publish message gps-long-raw: ");
    Serial.println(mqtt_payload_gps_longitude_raw);
    client.publish(pub_topic_gps_longitude_raw,
mqtt_payload_gps_longitude_raw);

    snprintf (mqtt_payload_gps_longitude_raw_deg, 50, "%lf",
longitude_raw_deg);
    Serial.print("Publish message gps-lat-raw-deg: ");
    Serial.println(mqtt_payload_gps_longitude_raw_deg);
    client.publish(pub_topic_gps_longitude_raw_deg,
mqtt_payload_gps_longitude_raw_deg);

    snprintf (mqtt_payload_gps_longitude_raw_bill, 50, "%lf",
longitude_raw_bill);
    Serial.print("Publish message gps-lat-raw-bill: ");
    Serial.println(mqtt_payload_gps_longitude_raw_bill);
    client.publish(pub_topic_gps_longitude_raw_bill,
mqtt_payload_gps_longitude_raw_bill);

//_____

_____

    snprintf (mqtt_payload_gps_altitude, 50, "%lf", altitude);
    Serial.print("Publish message gps-alt: ");
```

```
    Serial.println(mqtt_payload_gps_altitude);
    client.publish(pub_topic_gps_altitude, mqtt_payload_gps_altitude);

    snprintf (mqtt_payload_gps_altitude_meter, 50, "%lf", altitude_meter);
    Serial.print("Publish message gps-alt-meter: ");
    Serial.println(mqtt_payload_gps_altitude_meter);
    client.publish(pub_topic_gps_altitude_meter,
mqtt_payload_gps_altitude_meter);

    snprintf (mqtt_payload_gps_altitude_mil, 50, "%lf", altitude_mil);
    Serial.print("Publish message gps-alt-mil: ");
    Serial.println(mqtt_payload_gps_altitude_mil);
    client.publish(pub_topic_gps_altitude_mil, mqtt_payload_gps_altitude_mil);

    snprintf (mqtt_payload_gps_altitude_km, 50, "%lf", altitude_km);
    Serial.print("Publish message gps-alt-km: ");
    Serial.println(mqtt_payload_gps_altitude_km);
    client.publish(pub_topic_gps_altitude_km, mqtt_payload_gps_altitude_km);

    snprintf (mqtt_payload_gps_altitude_feet, 50, "%lf", altitude_feet);
    Serial.print("Publish message gps-alt-feet: ");
    Serial.println(mqtt_payload_gps_altitude_feet);
    client.publish(pub_topic_gps_altitude_feet, mqtt_payload_gps_altitude_feet);

//_____________________________________________________________________
_____
    snprintf (mqtt_payload_akselerasi_x, 50, "%lf", akselerasi_x);
    Serial.print("Publish message akselerasi-x: ");
    Serial.println(mqtt_payload_akselerasi_x);
    client.publish(pub_topic_aks_x, mqtt_payload_akselerasi_x);
    snprintf (mqtt_payload_akselerasi_y, 50, "%lf", akselerasi_y);
```

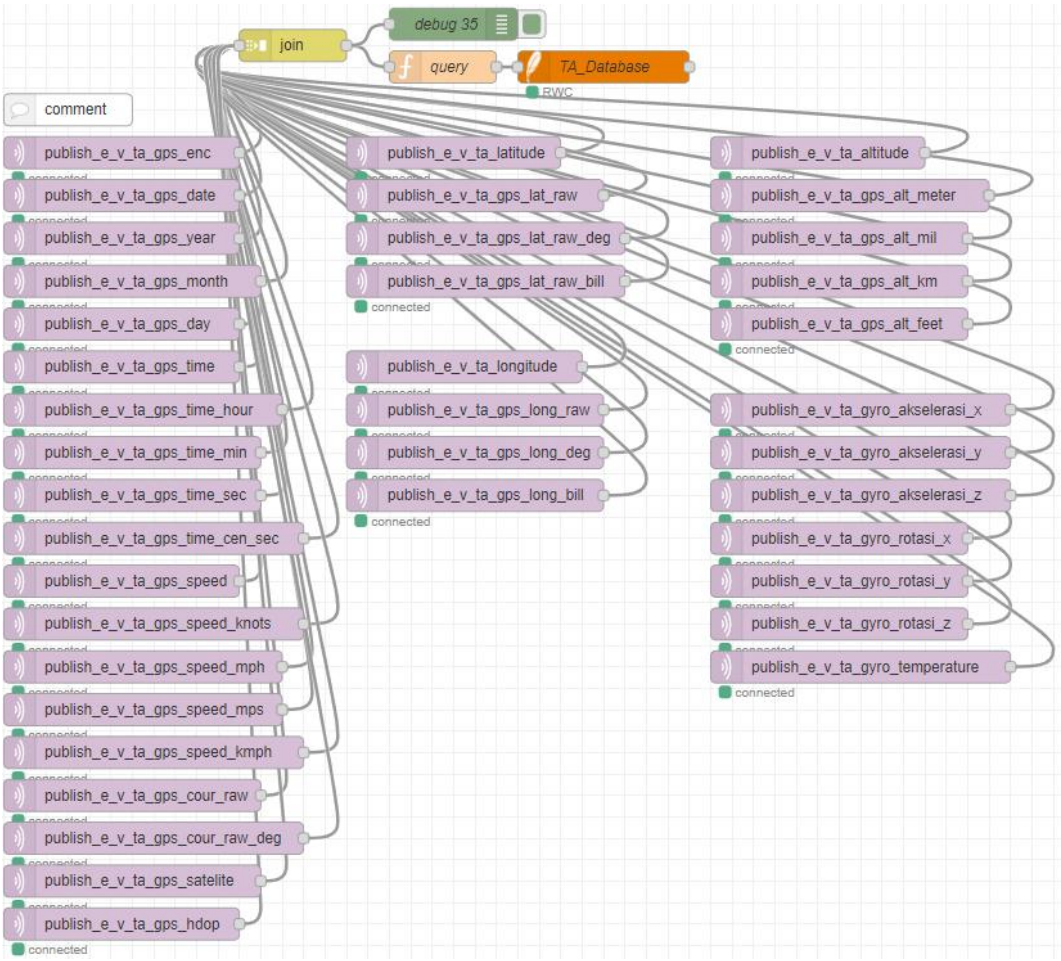
```
  Serial.print("Publish message akselerasi-y: ");
  Serial.println(mqtt_payload_akselerasi_y);
  client.publish(pub_topic_aks_y, mqtt_payload_akselerasi_y);
  snprintf (mqtt_payload_akselerasi_z, 50, "%lf", akselerasi_z);
  Serial.print("Publish message akselerasi-z: ");
  Serial.println(mqtt_payload_akselerasi_z);
  client.publish(pub_topic_aks_z, mqtt_payload_akselerasi_z);
  snprintf (mqtt_payload_rotasi_x, 50, "%lf", rotasi_x);
  Serial.print("Publish message rotasi-x: ");
  Serial.println(mqtt_payload_rotasi_x);
  client.publish(pub_topic_rts_x, mqtt_payload_rotasi_x);
  snprintf (mqtt_payload_rotasi_y, 50, "%lf", rotasi_y);
  Serial.print("Publish message rotasi-y: ");
  Serial.println(mqtt_payload_rotasi_y);
  client.publish(pub_topic_rts_y, mqtt_payload_rotasi_y);
  snprintf (mqtt_payload_rotasi_z, 50, "%lf", rotasi_z);
  Serial.print("Publish message rotasi-z: ");
  Serial.println(mqtt_payload_rotasi_z);
  client.publish(pub_topic_rts_z, mqtt_payload_rotasi_z);
  snprintf (mqtt_payload_temp, 50, "%lf", temp_gyro);
  Serial.print("Publish message temp-gyro: ");
  Serial.println(mqtt_payload_temp);
  client.publish(pub_topic_temp, mqtt_payload_temp);

  Serial.println("> MQTT data published");

// ____________________________________________________
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Monitoring DAQ-Wnd");
  display.println("==================");
```

```
    display.println("Lat,long,alt");
    display.print(latitude, 1);
    display.print(", ");
    display.print(longitude, 1);
    display.print(", ");
    display.print(altitude, 1);
    display.println("");
    display.display();
    display.println("Akselerasi:x,y,z");
    display.print(akselerasi_x, 1);
    display.print(", ");
    display.print(akselerasi_y, 1);
    display.print(", ");
    display.print(akselerasi_z, 1);
    display.println("");
    display.display();
    display.println("Rotasi:x,y,z");
    display.print(rotasi_x, 1);
    display.print(", ");
    display.print(rotasi_y, 1);
    display.print(", ");
    display.print(rotasi_z, 1);
    display.println("");
    display.display();
    delay(writeInterval);// delay
  }
  else{
    Serial.println(F("INVALID"));
  }
}
```

```
// setup
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  ss.begin(GPSBaud);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }
  delay(500);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setRotation(0);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  if (!mpu.begin()) {
    Serial.println("Sensor init failed");
    while (1)
      yield();
  }
  main_gyro();
}
void loop() {
  if (!client.connected())
  reconnect();
  client.loop();
  while (ss.available() > 0)
  if(gps.encode(ss.read()))
```

```
    control_program();
    if (millis() > 5000 && gps.charsProcessed() < 10){
        Serial.println(F("No GPS detected: check wiring."));
        while(true);
    }
}
```

Node pada Node-RED



Program Node Query

```
let gps_enc          = msg.payload.publish_e_v_ta_gps_enc;
let gps_date         = msg.payload.publish_e_v_ta_gps_date;
```

```
let gps_year          = msg.payload.publish_e_v_ta_gps_year;
let gps_month         = msg.payload.publish_e_v_ta_gps_month;
let gps_day           = msg.payload.publish_e_v_ta_gps_day;
let gps_time          = msg.payload.publish_e_v_ta_gps_time;
let gps_time_hour     = msg.payload.publish_e_v_ta_gps_time_hour;
let gps_time_min      = msg.payload.publish_e_v_ta_gps_time_min;
let gps_time_sec      = msg.payload.publish_e_v_ta_gps_time_sec;
let gps_time_cen_sec  = msg.payload.publish_e_v_ta_gps_time_cen_sec;
let gps_speed         = msg.payload.publish_e_v_ta_gps_speed;
let gps_speed_knots   = msg.payload.publish_e_v_ta_gps_speed_knots;
let gps_speed_mph     = msg.payload.publish_e_v_ta_gps_speed_mph;
let gps_speed_mps     = msg.payload.publish_e_v_ta_gps_speed_mps;
let gps_speed_kmph    = msg.payload.publish_e_v_ta_gps_speed_kmph;
let gps_course_raw    = msg.payload.publish_e_v_ta_gps_cour_raw;
let gps_course_raw_deg = msg.payload.publish_e_v_ta_gps_cour_raw_deg;
let gps_satelite      = msg.payload.publish_e_v_ta_gps_satelite;
let gps_hdop          = msg.payload.publish_e_v_ta_gps_hdop;
let latitude          = msg.payload.publish_e_v_ta_latitude;
let latitude_raw      = msg.payload.publish_e_v_ta_gps_lat_raw;
let latitude_raw_deg  = msg.payload.publish_e_v_ta_gps_lat_raw_deg;
let latitude_raw_bill = msg.payload.publish_e_v_ta_gps_lat_raw_bill;
let longitude         = msg.payload.publish_e_v_ta_longitude;
let longitude_raw     = msg.payload.publish_e_v_ta_gps_long_raw;
let longitude_raw_deg = msg.payload.publish_e_v_ta_gps_long_deg;
let longitude_raw_bill = msg.payload.publish_e_v_ta_gps_long_bill;
let altitude          = msg.payload.publish_e_v_ta_altitude;
let altitude_meter    = msg.payload.publish_e_v_ta_gps_alt_meter;
let altitude_mil      = msg.payload.publish_e_v_ta_gps_alt_mil;
let altitude_km       = msg.payload.publish_e_v_ta_gps_alt_km;
let altitude_feet     = msg.payload.publish_e_v_ta_gps_alt_feet;
let gyro_aks_x        = msg.payload.publish_e_v_ta_gyro_akselerasi_x;
```

```
let gyro_aks_y          = msg.payload.publish_e_v_ta_gyro_akselerasi_y;
let gyro_aks_z          = msg.payload.publish_e_v_ta_gyro_akselerasi_z;
let gyro_rts_x          = msg.payload.publish_e_v_ta_gyro_rotasi_x;
let gyro_rts_y          = msg.payload.publish_e_v_ta_gyro_rotasi_y;
let gyro_rts_z          = msg.payload.publish_e_v_ta_gyro_rotasi_z;
let gyro_temperature    = msg.payload.publish_e_v_ta_gyro_temperature;

msg.topic = 'INSERT INTO table_data(id, gps_encode, gps_date, gps_year,
gps_month, gps_day, gps_time, gps_time_hour, gps_time_minute,
gps_time_second, gps_time_centi_second, gps_speed, gps_speed_knots,
speed_mph, speed_mps, speed_kmph, gps_course_raw, gps_course_raw_deg,
gps_satelite, gps_hdop, latitude, latitude_raw, latitude_raw_deg,
latitude_raw_bill, longitude, longitude_raw, longitude_raw_deg,
longitude_raw_bill, altitude, altitude_meter, altitude_mil, altitude_km,
altitude_feet, akseleration_x, akseleration_y, akseleration_z, rotation_x,
rotation_y, rotation_z, gyro_temp, date, time, device) values(null, $val1, $val2,
$val3, $val4, $val5, $val6, $val7, $val8, $val9, $val10, $val11, $val12, $val13,
$val14, $val15, $val16, $val17, $val18, $val19, $val20, $val21, $val22, $val23,
$val24, $val25, $val26, $val27, $val28, $val29, $val30, $val31, $val32, $val33,
$val34, $val35, $val36, $val37, $val38, $val39, date("now"),time("now"), "Alat-
Params");'
msg.payload = [gps_enc, gps_date, gps_year, gps_month, gps_day, gps_time,
gps_time_hour, gps_time_min, gps_time_sec, gps_time_cen_sec, gps_speed,
gps_speed_knots, gps_speed_mph, gps_speed_mps, gps_speed_kmph,
gps_course_raw, gps_course_raw_deg, gps_satelite, gps_hdop, latitude,
latitude_raw, latitude_raw_deg, latitude_raw_bill, longitude, longitude_raw,
longitude_raw_deg, longitude_raw_bill, altitude, altitude_meter, altitude_mil,
altitude_km, altitude_feet, gyro_aks_x, gyro_aks_y, gyro_aks_z, gyro_rts_x,
gyro_rts_y, gyro_rts_z, gyro_temperature]

return msg;
```

LAMPIRAN D

Surat Edaran WR I Nomor :SE/12/UNJANI/I/2021

Tentang

Ketentuan Unggah Mandiri Tugas Akhir bagi Mahasiswa Unjani

- Memakai Watermark logo Unjani disetiap halaman **(ukuran 12 cm Berwarna)**
- Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
- Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditandatangani diatas Materai Rp 10.000 oleh Dosen Pembimbing dan Mahasiswa dalam format PDF ukuran maksimum file 1 MB
- Pindai Lembar Izin Publikasi yang sudah ditandai tangan oleh Dosen dan Mahasiswa
- Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb
- Isi BAB I - V Maksimum ukuran file 5 MB
- Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
- Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa

4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendaptkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.

5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.

LAMPIRAN E

Surat Edaran WR I Nomor :SE/33/UNJANI/I/2021

Tentang

Revisi Peraturan Pengumpulan Tugas Akhir di Perpustakaan Pusat bagi Mahasiswa Unjani Nomor :SE/10/UNJANI/2021

- Memakai Watermark logo Unjani disetiap halaman **ukuran 4x4 cm Berwarna)**
- Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
- Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditanda tangani diatas Materai Rp 10.000 oleh Mahasiswa dalam format PDF ukuran maksimum file 1 MB
- Pindai Lembar Izin Publikasi yang bersifat opsional (apabila dosen pembimbing tidak berkenan untuk dipublikasikan, maka yang mendatangani hanya mahasiswa bersangkutan.
- Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb
- Isi BAB I - V Maksimum ukuran file 5 MB
- Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
- Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa

4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendaptkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.

5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.

LAMPIRAN F

Surat Edaran WR I Nomor :SE/12/UNJANI/I/2021

Tentang

Ketentuan Unggah Mandiri Tugas Akhir bagi Mahasiswa Unjani

- Memakai Watermark logo Unjani disetiap halaman **(ukuran 12 cm Berwarna)**
- Cover berwarna dan di simpan dalam format JPG ukuran maksimum 500 Kb
- Pindai Lembar Pernyataan Bebas Plagiasi yang sudah ditandatangani diatas Materai Rp 10.000 oleh Dosen Pembimbing dan Mahasiswa dalam format PDF ukuran maksimum file 1 MB
- Pindai Lembar Izin Publikasi yang sudah ditandai tangan oleh Dosen dan Mahasiswa
- Abstrak dengan dua bahasa (Bahasa Indonesia dan Bahasa Inggris dalam bentuk PDF ukuran masimum file 500 Kb
- Isi BAB I - V Maksimum ukuran file 5 MB
- Daftar Pustaka dan Lampiran-lampiran dalam bentuk PDF ukuran maksimum 1 MB
- Soft file tugas Akhir di simpan dalam Compack Disk (CD) dan tempat CD berbentuk kotak yang sudah ditanda tangani oleh Dosem Pembimbing dan Mahasiswa

4 Apabila mahasiswa sudah unggah Laporan Tugas Akhir ke Website Perpustakaan Pusat Unjani dan Perpustakaan Fakultas Teknik, mahasiswa tersebut akan mendaptkan Report berupa Surat Keterangan Penyerahan Tugas Akhir ke email masing-masing.

5 Surat Keterangan Penyerahan Tugas Akhir harap dibawa saat penyerahan CD Tugas ke Perpustakaan Pusat Unjani saat verifikasi Data.

LAMPIRAN D Tim Penyusun Standarisasi Pedoman Draft Tugas Akhir

Penanggung jawab	: Ketua Prodi
	: Sekretaris Prodi
Ketua Tim	: Udin Komarudin, S.T., M.T.
Anggota	: Sunubroto, S.T., M.T.
	: Ahmad Daelami, S.T., M.M.
	: Ade Sena Permana, S.T., M.T.
	: M. Reza Hidayat, S.T., M.T.
	: Giri Angga Setia, S.T., M.T.
	: Fauzia Haz, S.T., M.T.