Sistem *Trajectory Mapping* Berbasis GPS dan FTDI

Buku Manual Penggunaan Sistem

Ditulis Oleh

1. Ryan Satria Wijaya
2. Vafin Julanba
3. Aditya Gading Purnama
4. Rizki Munandar
5. Neni Nur Hidayanti Ningsih
6. Maria Nirmala Dewi Situmorang
7. Gideon Butar Butar
8. Feri Ana
9. Kanaya Maghfira Ariska
10. Armansyah Dwi Prawira

Politeknik Negeri Batam

2024

KATA PENGANTAR

Puji Syukur kehadirat Tuhan Yang Maha Esa atas Rahmat dan karunia-Nya sehingga buku manual berjudul Sistem Trajectory Mapping Berbasis GPS dan FTDI ini dapat disusun dengan baik.

Buku manual ini berisi penjelasan lengkap mengenai system pemetaan lintasan (Trajectory mapping) menggunakan data GPS yang diolah melalui Bahasa pemrograman Cpp dan divisualisasikan ke dalam koordinat kartesian 2D menggunakan Python di system operasi Ubuntu.

Sistem ini diharapkan dapat digunakan sebagai referensi untuk pengembangan system monitoring pergerakan robot, khususnya pada bidang Autonomous Surface Vehicle (ASV), serta menjadi dasar dokumentasi dalam pengajuan Hak Kekayaan Intelektual (HKI).

Ucupan terima kasih disampaikan kepada semua pihak yang telah membantu dalam penyusunan system dan dokumentasi ini.

Kami menyadari bahwa buku manual ini masih memiliki kekurangan, oleh karena itu kritik dan saran yang membangun sangat diharapkan demi perbaikan ke depannya.

Akhir kata, semoga buku manual ini dapat bermanfaat bagi pembaca dan pihak-pihak yang membutuhkan.

DAFTAR ISI

DAFTAR GAMBAR

1. Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi navigasi dan pelacakan posisi berbasis GPS sangat pesat. Sistem Global Positioning System (GPS) kini telah digunakan secara luas dalam berbagai bidang seperti transportasi, pemetaan, pertanian, hingga otomasi robotik.

Pada bidang robotika divisi maritim seperti Autonomous Surface Vehicle (ASV), system navigasi dan pemetaan lintasan menjadi hal yang sangat penting untuk mengetahui posisi dan arah Gerak secara real-time. Oleh karena itu, dibutuhkan sistem yang mampu mengolah data GPS secara efisien dan menampilkan posisi robot dalam bentuk visual koordinat 2D yang mudah di pahami.

Dalam buku manual ini, dijelaskan system berbasis ubuntu yang dapat membaca data GPS dari modul Neo 6M v2 melalui koneksi FTDI, mengkonversi koordinat dari format NMEA ke system UTM, dan menampilkan posisi lintasan secara langsung menggunakan python.

1.2 Tujuan

Tujuan dari pengembangan sistem ini adalah untuk:

- Membuat sistem pembacaan data posisi menggunakan modul GPS Neo 6M v2 yang terhubung ke laptop Ubuntu via FTDI
- Melakukan konversi data GPS dari format NMEA ke koordinat derajat decimal, kemudia ke UTM (Universal Transverse Mercator)
- Menghasilkan output posisi relative terhadap titik awal dalam format kartesian 2d (X, Y) dalam satuan meter
- Menghasilkan data posisi ke file untuk keperluan visualisasi dan monitoring
- Menampilkan lintasan pergerakan robot dalam bentuk visual menggunakan python dan matplotlib secara real-time

1.3 Ruang Lingkup

Lingkup system yang dibahas dalam buku manual ini mencakup:

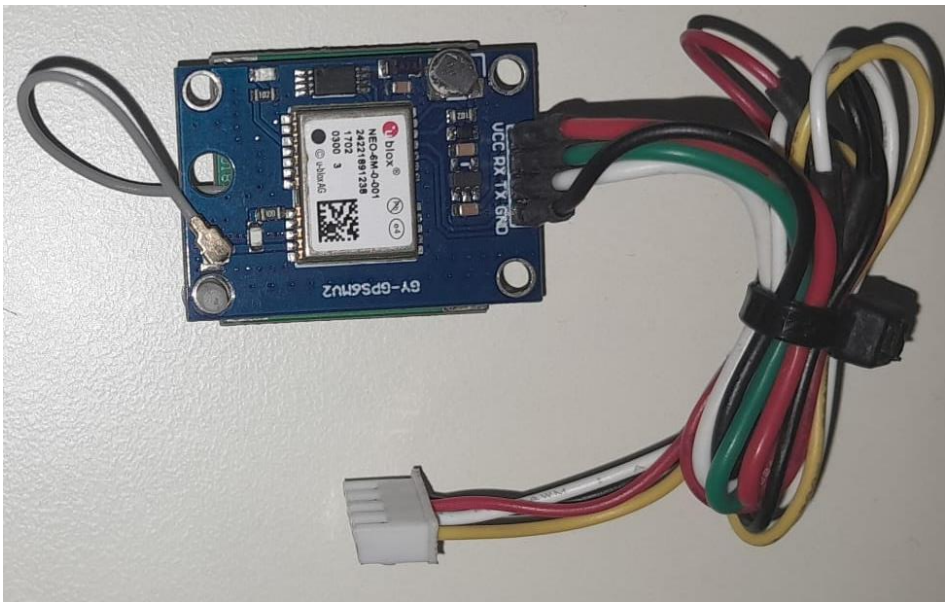
- Perolehan data GPS dengan format NMEA dari perangkat GPS melalui FTDI.
- Proses konversi koordinat dari format NMEA ke format derajat decimal.
- Transformasi data koordinat ke sistem UTM (Universal Transverse Mercator).
- Perhitungan posisi relative terhadap titik awal (origin) dalam sistem koordinat kartesian.
- Penyimpanan data posisi ke dalam file position.txt secara terus-menerus.
- Visualisasi lintasan Gerak robot ASV secara langsung menggunakan python dan matplotlib.

- Tampilan lingkungan visual arena berupa titik-titik bola, kotak start dan finish, dan arah lintasan.

2. Persiapan dan Instalasi Sistem

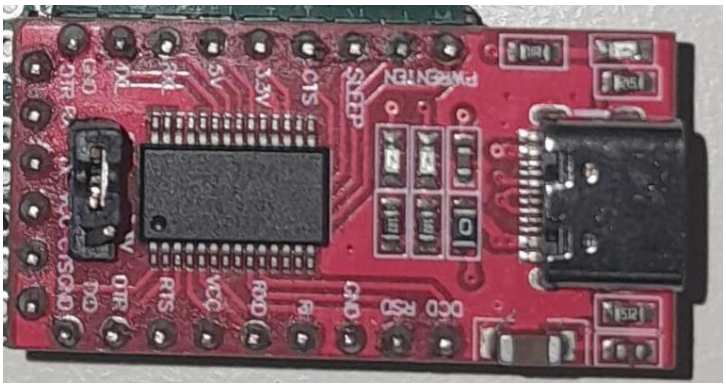
2.1 Perangkat Keras yang digunakan

- GPS Neo 6Mv2
  Modul GPS untuk mendapatkan koordinat posisi berupa latitude, longitude dan dikonversi ke UTM agar mendapatkan nilai X, Y.



Gambar 1. GPS Neo 6Mv2

- Modul FTDI
  Modul Konverter USB to Serial (RX, TX) agar mendapatkan data GPS melalui komunikasi serial dan sebagai jembatan komunikasi GPS ke Laptop.



Gambar 2. FTDI FT232RL

- Laptop (Ubuntu)
  Sistem Operasi ubuntu untuk bisa menjalankan program melalui terminal ubuntu.

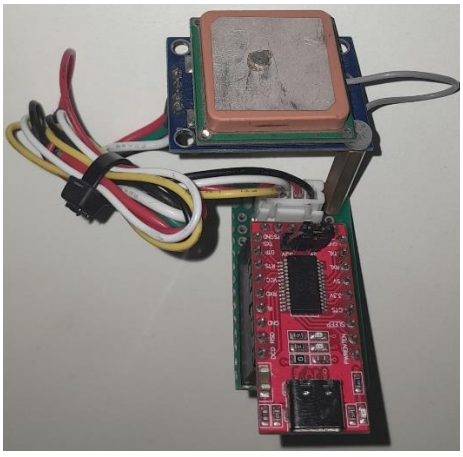
2.2 Perangkat ringan yang digunakan

- Visual Studio Code
  Software Visual Studio Code untuk membuat program dan menyimpan file ke home.
- g++ compiler
  Bahasa program yang mudah dalam mengolah data GPS dan Modul g++ untuk Kompilasi dalam menjalani program GPS.cpp.
- Python 3.13.2
  Bahasa program yang mudah dalam pembuatan GUI untuk divisualkan ke koordinat kartesian sesuai dengan data X dan Y yang diterima oleh data GPS.cpp

2.3 Instalasi dan Setup Awal

Berikut adalah instalasi beberapa modul yang dibutuhkan oleh program GPS.cpp dan monitoring.py, persiapan awal yang perlu disiapkan agar dapat menjalankan sistem sesuai dengan prosedur.

- Instalasi modul dan dependencies
  GPS.cpp yaitu, iostream, iomanip, fstream, string, cmath, Fcntl.h, termios, unistd, Sstream.
  Monitoring.py yaitu, Matplotlib.python, Matplotlib.animation, os, math.
- Koneksi port USB dan pengecekan



Gambar 3. Komponen Sistem

Sambungkan GPS ke Modul FTDI:

- GPS TX ke FTDI RX
- GPS RX ke FTDI TX
- VCC ke 5V
- GND ke GND

Colokkan FTDI ke port USB Laptop dan pastikan kabel port usb dapat menerima port serial dan bisa di chek Port Serial dengan cara buka terminal dan ketik: ls /dev/tty*, dalam percobaan ini dikenali sebagai /dev/ttyUSB0.

2.4 Struktur File Program

Berikut adalah struktur file dan program yang digunakan untuk memenuhi kebutuhan sistem:

- GPS.cpp
  Program utama untuk membaca data GPS dan menyimpan data X dan Y ke file "position.txt".
- Monitoring.py
  Program visualisasi real-time lintasan dan arah robot menggunakan matplotlib.
- Position.txt
  File output berisi koordinat X dan Y yang dihasilkan dari GPS.cpp.

3. Penjelasan Cara Kerja Program dan Konversi Data

3.1 Penjelasan GPS.cpp dan monitoring,py

Berikut adalah penjelasan dari program yang digunakan pada system:

3.1.1 Program GPS.cpp

- Membaca data GPS melalui kabel port serial untuk membaca data seperti NMEA GPRMC.
- Mengubah data NMEA ke data yang diperlukan seperti Latitude, Longitude, Time dan Status.
- Mekonversi data latitude longitude ke UTM X Y dalam satuan meter.
- Menetapkan titik awal sebagai origin X,Y menjadi 0,0.
- Menyimpan data X,Y ke file position.txt

3.1.2 Program monitoring.cpy

- Membuat grid koordinat dengan ukuran 25 x 25 dengan jarak 5 dalam satuan meter.
- Memberikan beberapa rintangan seperti titik merah dan titik hijau, kotak merah sebagai kondisi start dan finish, dan terdapat kotak hijau dan kotak biru sebagai jalur lintasan yang harus di lewati, rintangan ini dibuat sesuai dengan arena Latihan Autonomous Surface Vehicle (ASV).
- Membentuk robot sebagai titi biru dan arah pegerakannya dengan menggunakan panah.
- Menampilkan jejak pergerakan ASV dalam bentuk gsris putus-putus.

3.2 Alur Konversi NMEA ke UTM

Dalam data GPS yang diterima oleh satelit berupa data geografis NMEA yaitu latitude dan longitude dengan format derajat menit, setelah mendapatkan data NMEA selanjutnya akan di konversi ke latitude longitude dengan format derajat decimal, selanjutnya akan di konversi ke radian dan dilanjutkan dikonversi ke utm X dan Y yang dapat divisualkan kedalam koordinat kartesian 2D pada program monitoring.py

3.2.1 Konversi latitude longitude format NMEA ke format derajat decimal

konversi Latitude longitude format NMEA berupa satuan derajat menit ke derajat decimal sebagai berikut :

- Pisahkan derajat dan menit

  untuk latitude terdiri dari 2 digit derajat, longitude terdiri dari 3 digit derajat, menit
  terdiri dari 2 digit sebelum titik ditambah semua angka setelah titik, sebagai contoh berikut:

- Latitude: 0107.07451,N
- Longitude: 10402.94184,E

Penjelasan:
Dengan data latitude dan longitude diatas dapat disimpulkan

- 0107.07451 artinya 01 derajat, 07.07451 menit
- 10402.94184 artinya 104 derajat, 02.94184 menit

- Konversi ke derajat desimal:

$$\text{Desimalderajat} = \text{derajat} + (\frac{\text{minute}}{60.0})$$

Contoh : latitude = 0107.07451,N

- Derajat: 01
- Menit: 07.07451
- Konversi:

$$latitude = 01 + \frac{7.07451}{60} = \ 01 + 0.1179085 = 1.1179085^{\circ}$$

Contoh : longitude = 10402.94184,E

- Derajat: 104
- Menit: 02.94184
- Konversi:

$$longitude = 104 + \frac{2.94184}{60} = \ 104 + 0.0490306667 = 104.0490306667^{\circ}$$

Maka Latitude dan longitude yang akan digunakan sebagai berikut

- Latitude = 1.1179085°
- Longitude = 104.0490306667°

Ketika data sudah di convert dari derajat waktu ke derajat decimal maka sudah bisa di convert ke UTM.

3.2.2 Konversi Koordinat geodetik ke UTM

Rumus Matematika Konversi dari latitude dan longitude ke UTM X dan UTM Y

variablel yang akan digunakan
Ø˚= latitude derajat
λ˚= longitude derajat
λ0˚= meridian Tengah zona UTM derajat
Ø = latitude radian
λ = longitude radian
λ0 = meridian Tengah zona UTM radian

Parameter WGS84
$\alpha = 6378137.0$ (semi-major axis)

$f = \frac{1}{298.257223563}$ (flattening)
$k0 = 0.9996$ (skala zona sentral)
$e^2 = f \cdot (2 - f)$ (eksentisitas kuadrat)

3.2.2.1 Hitung parameter dasar

$$\boldsymbol{e^4 = e^2 \cdot e^2}$$
$$\boldsymbol{e^6 = e^4 \cdot e^2}$$

Hitung zona:

$$Zone = \left[\lambda^\circ + \frac{180}{6}\right] + 1$$

$$\lambda 0^\circ = (Zone - 1) \cdot 6 - 180 + 3$$

Hasil dalam derajat ini akan dikonversi ke radian

3.2.2.2 Konversi ke radian

$$\emptyset = \emptyset^\circ \cdot \frac{\pi}{180}$$
$$\lambda = \lambda^\circ \cdot \frac{\pi}{180}$$
$$\lambda 0 = \lambda 0^\circ \cdot \frac{\pi}{180}$$

3.2.2.3 Hitung parameter geodetic

$$N = \frac{a}{\sqrt{1 - e^2 \cdot \sin^2(\emptyset)}}$$
$$T = \tan^2(\emptyset)$$
$$C = \frac{e^2}{1 - e^2} \cdot \cos^2(\emptyset)$$
$$A = \cos(\emptyset) \cdot (\lambda - \lambda 0)$$

3.2.2.4 Hitung meridional arc M

$$M = a\left[\left(1 - \frac{e^2}{4} - \frac{3e^4}{64} - \frac{5e^6}{256}\right)\emptyset - \left(\frac{3e^2}{8} + \frac{3e^4}{32} + \frac{45e^6}{1024}\right)\sin(2\emptyset) + \left(\frac{15e^4}{256} + \frac{45e^6}{1024}\right)\sin(4\emptyset) - \left(\frac{35e^6}{3072}\right)\sin(6\emptyset)\right]$$

3.2.2.5 Hitung UTM X dan Y

UTM X (Eastings)

$$X = k0 \cdot N\left[A + \frac{(1 - T + C)A^3}{6} + \frac{(5 - 18T + T^2 + 72C - 58e'^2)A^5}{120}\right] + 500000$$

UTM Y (Northings)

$$Y = k0 \cdot \left[M + N \cdot \tan(Ø) \cdot \left(\frac{A^2}{2} + \frac{(5 - T + 9C + 4C^2)A^4}{24} + \frac{(61 - 58T + T^2 + 600C - 330e'^2)A^6}{720} \right) \right]$$

Lalu ditambahkan kondisi jika berada di belahan bumi Selatan, tambahkan 10.000.000 meter ke nilai Y: $if Ø < 0, maka\ Y = Y + 10^7$

Tujuan: agar semua nilai Y tetap bernilai positif, tidak peduli ada dibelahan bumi utara atau pun Selatan.

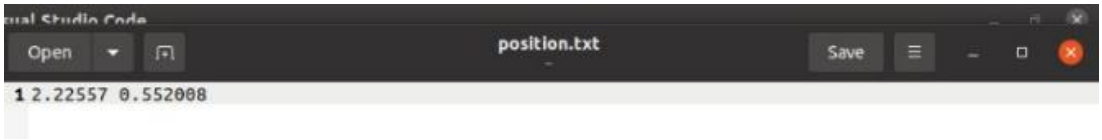
3.3 Visualisasi Data dan Format File Output

Berikut adalah Penempatan format dan hasil dari GPS divisualisasikan ke GUI Koordinat kartesian X dan Y.

- Format File Output (position.txt)

  Program GPS.cpp menghasilkan file Bernama position.txt yang menyimpan koordinat relative robot terhadap titik awal (origin). Data ini disimpan secara overwrite setiap kali pembacaan GPS baru tersedia, sehingga hanya terdapat satu baris data terbaru.

  Format isi file berupa nilai X Y:



Gambar 4. Format file tempat pengiriman data dan menerima data

  Artinya :

  - X = 2.22557 meter
  - Y = 0.552008 meter

  Koordinat ini merupakan posisi relative titik awal GPS (yang dianggap sebagai titik (0,0) dalam system UTM (Universal Transverse Mercator).

- Visualisasi Lintasan Robot (monitoring.py)

  Program monitoring.py membaca file dari position.txt secara berkala (setiap 0.5 detik) dan menampilkan:

  Elemen Rintangan Visual yang digunakan

Elemen	Warna	Deskripsi

Robot	Biru (bulat)	Posisi saat ini
Lintasan	Garis biru putus-putus	Jejak pergerakan
Panah Arah	Biru (Panah)	Menunjukkan arah Gerak
Bola Merah	Merah	Titik rintangan atau objek
Bola Hijau	Hijau	Titik objek lain
Kotak Merah	Merah	Posisi start awal
Kotak Hijau	Hijau	Rintangan yang dilalui
Kotak Biru	Biru	Rintangan yang dilalui

Sistem Koordinat Visual:

- Sistem koordinat dengan satuan meter dalam koordinat kartesian 2D
- Sumbu X negatif (kiri) dan sumbu Y positif (atas) sesuai dengan system plot matplotlib.

- Fungsi Animasi (Real-Time Update)

  Fungsi update() dalam monitoring.py melakukan hal beriku:

  - Membaca position.txt
  - Menyimpan lintasan sebelumnya
  - Memperbarui posisi titik robot
  - Menghitung arah gerak dari dua titik terakhir
  - Menggambar panah ke arah tersebut
  - Menampilkan hasil ke grid arena simulasi

4. Panduan Penggunaan dan *Troubleshooting*

Berikut adalah Langkah-langkah penggunaan dalam menggunakan Sistem Trajectory Mapping Berbasis GPS dan FTDI:

4.1 Kompilasi dan Menjalankan GPS.cpp

Kompilasi Program
Buka terminal di folder tempat file GPS.cpp berada, lalu jalankan:

G++ GPS.cpp -o GPS

Jalankan Program GPS
Setelah kompilasi berhasil, jalankan:

./GPS

Program ini akan membaca data GPS dan menyimpan koordinat relative (dalam meter) ke file position.txt

4.2 Menjalankan Program Visualisasi monitoring.py

Setelah program GPS berjalan dan menghasilkan position.txt, jalankan visualisasi lintasan:

Python3 monitoring.py

Visualisasi akan menampilkan

- Posisi robot sebagai titik biru.
- Jejak lintasan robot (garis biru putus-putus).
- Arah pergerakan sebagai panah.
- Elemen tetap seperti, titik bola merah, titik bola hijau, kotak hijau, kotak biru dan kotak merah sebagai start/finish

4.3 Langkah-langkah Penggunaan

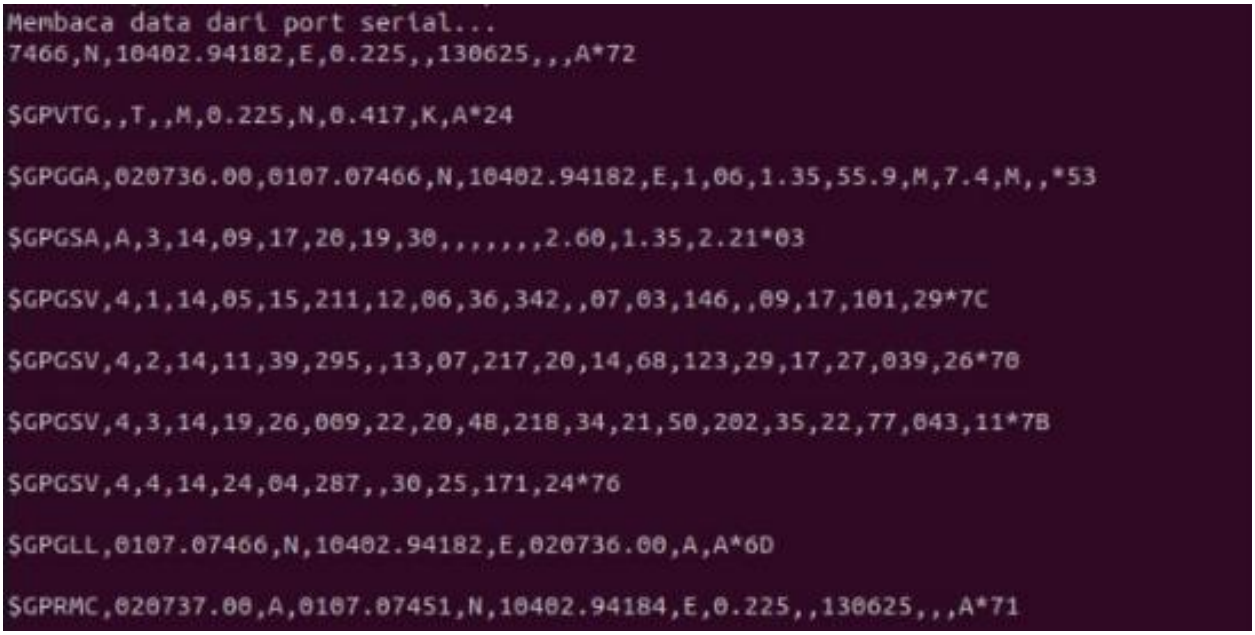
Berikut adalah Langkah-langkah penggunaan dalam menggunakan Sistem Trajectory Mapping Berbasis GPS dan FTDI:

- Chek port yang akan digunakan
  Cek Port Serial dengan cara menghubungkan GPS melalui kabel pin RX TX ke FTDI yang terhubung dengan port usb, klik ls /dev/ttyUSB* di terminal ubuntu dan bisa dilihat terdapat /dev/ttyUSB0 port usb yang terbaca di terminal ubuntu.

Gambar 5. Cara mengechek port yang terhubung

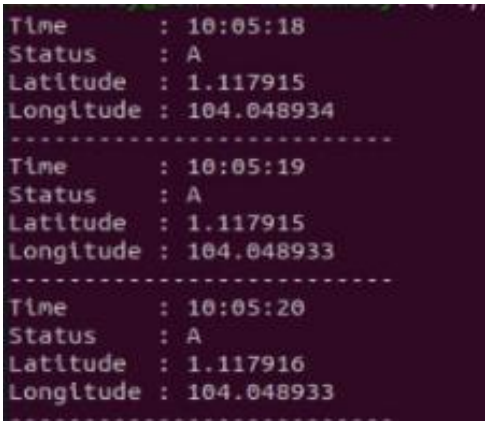
- Membuat program dan kompilasi program cpp
  Membuat program GPS.cpp di Visual Studio Code dan menyesuaikan dengan port yang terhubung, lalu dengan klik g++ GPS.cpp -o GPS.txt setelah di kompilasi dan tidak ada yang salah dalam programnya maka coba dijalankan dengan mengklik ./GPS.txt

- Pembacaan data GPS
  Berikut adalah pembacaan data GPS dengan menggunakan usb port "/dev/ttyUSB0" dan baudrate diatur 9600 dikarenakan pada gps terbaca baudrate 9600, hasilnya terdapat data NMEA bawaan dari setelit sebanyak 5 data dengan kode data yang berbeda-beda, berikut adalah hasil datanya:



Gambar 6. Pembacaan data GPS

- Pengambilan data yang penting dari data NMEA
  Berikut adalah pemilihan data NMEA sesuai dengan data yang dibutuhkan seperti data Waktu, Validasi status, Latitude dan Longitude, dengan memparsing data nmea dari kode yang diberikan oleh satelit.

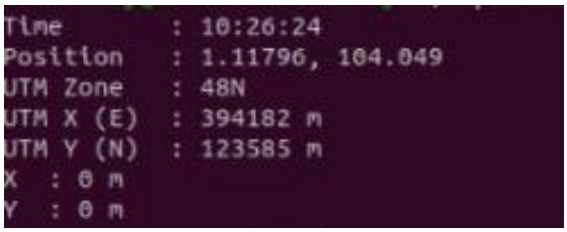


Gambar 7. Mengambil Data time, status, latitude, longitude

- Konversi data Latitude dan Longitude ke UTM X dan UTM Y
  Berikut data latitude dan longitude diconvert ke utm x dan utm y, agar dapat terbaca di monitoring dengan koordinat cartesian.

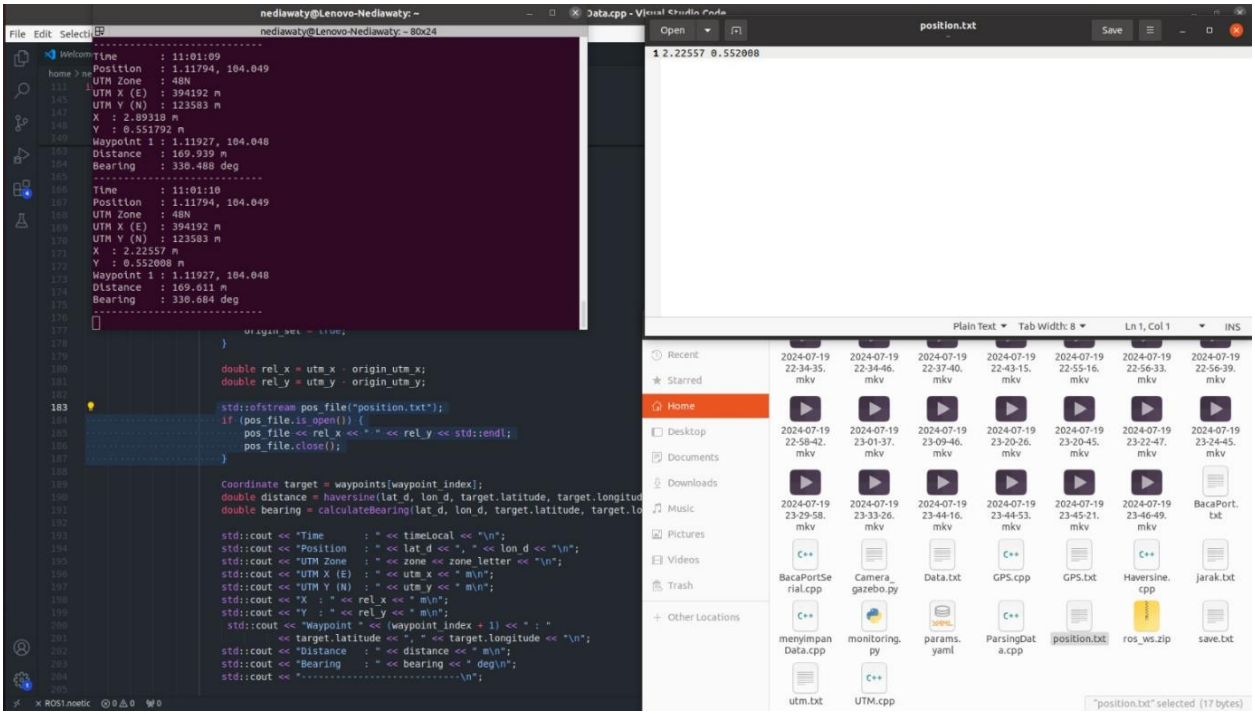
Gambar 8. Konversi latitude longitude ke UTM X dan Y

- Mereset data X dan Y menjadi 0,0
  Berikut adalah data UTM X dan UTM Y reset 0 ketika program dijalankan dengan algoritma x = utm x – utm x, y = utm y – utm y dan akan dimulai dari 0 hingga yang dikirim adalah data dengan variable x dan y.



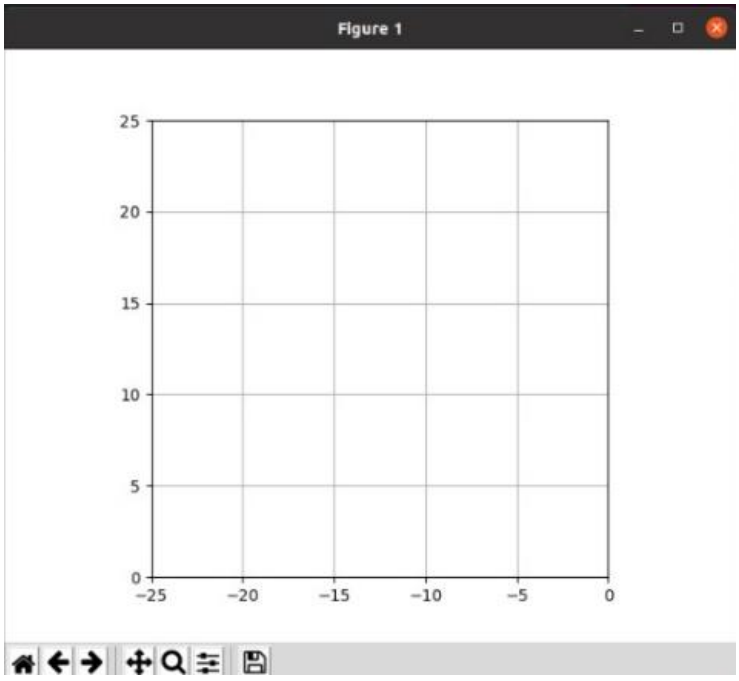
Gambar 9. Mereset data UTM X dan Y menjadi 0,0

- Mengirim data X dan Y ke file position.txt
  Berikut adalah pengiriman data x dan y ke file position.txt dengan menggunakan fungsi std::ofstream pos_file dan tertampil per satu baris tujuannya agar Ketika data yang diterima oleh monitoring.py tidak tertimpa dengan data sebelumnya, maka data akan diupdate terus menerus.



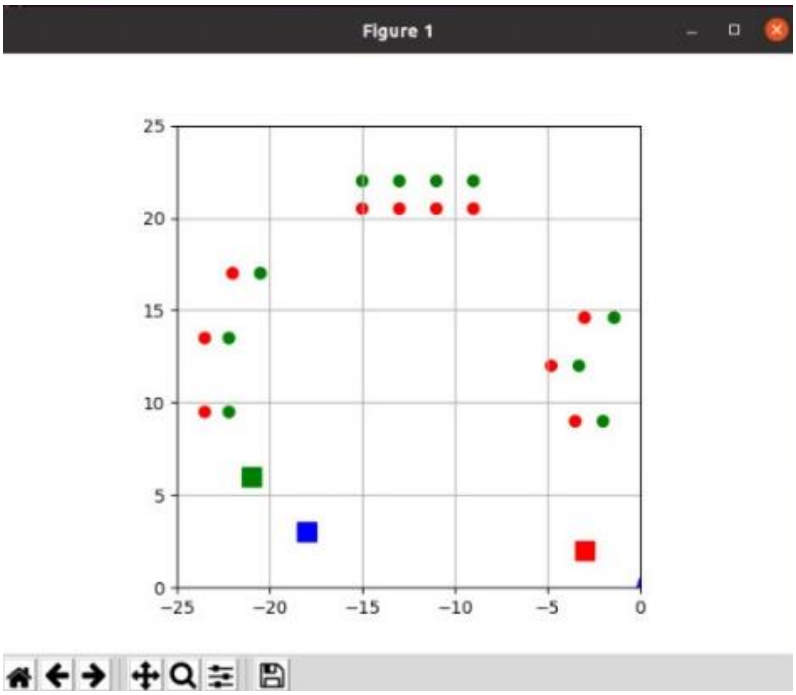
Gambar 10. Cara kerja pengiriman data X Y ke monitoring melalui format file.txt

- Membuat Grid kartesian menggunakan matplotlib python
  Berikut di bawah ini adalah program monitoring dengan menggunakan matplotlib untuk membuat grid 25 kali 25 dan ditampilkan dengan menggunakan fungsi plt.show()



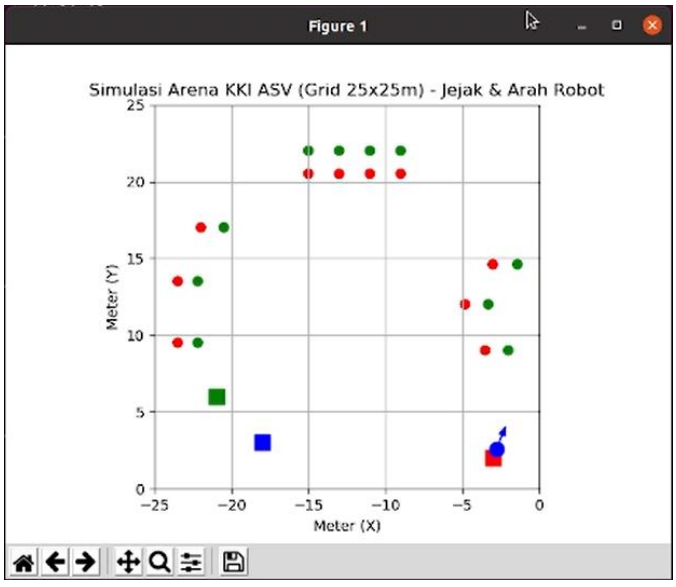
Gambar 11. Tampilan Grid kartesian x dan y

- Pembuatan rintangan pada grid menggunakan program python
  Berikut adalah lanjutan dari program sebelumnya dengan menambahkan obstacle yang dibutuhkan seperti dibawah ini.



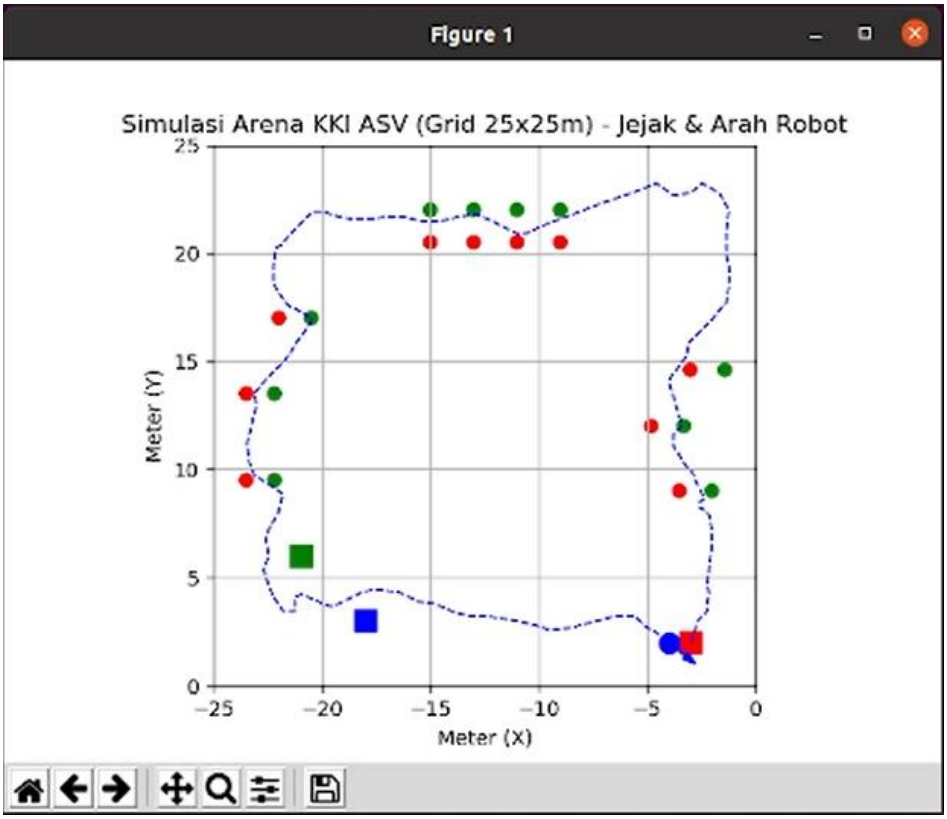
Gambar 12. Pemasangan Rintangan sesuai arena ASV

- Pembuatan fungsi pada posisi system gps
  Berikut adalah penambahan program untuk pemanggilan fungsi agar ada aksi pada system gps untuk ditampilkan ke monitoring dengan agen ditandai warna biru berbentuk bulat dan berarah panah untuk sesuai dengan arah panahnya



Gambar 13. pembuatan agent di dalam monitoring dengan berbentuk bulat berarah panah

- Hasil dari sistem gps yang ditampilkan ke grid kartesian
  Berikut adalah hasil sistem pemetaan lintasan berbasis gps dan ftdi dengan cara berjalan sesuai dengan rute yang sudah dibuat.



Gambar 14. Simulasi Full sesuai dengan sistem

4.4 Pemecahan Masalah

Berikut adalah permasalahan yang terjadi jika menggunakan sistem, permasalahan yang tercantum terlepas dari permasalahan diluar dari koneksi hardware seperti kabel port, kabel data, komponen dan laptop, dalam hal ini permasalahan pada software dan terbaca di terminal ubuntu sebagai berikut:

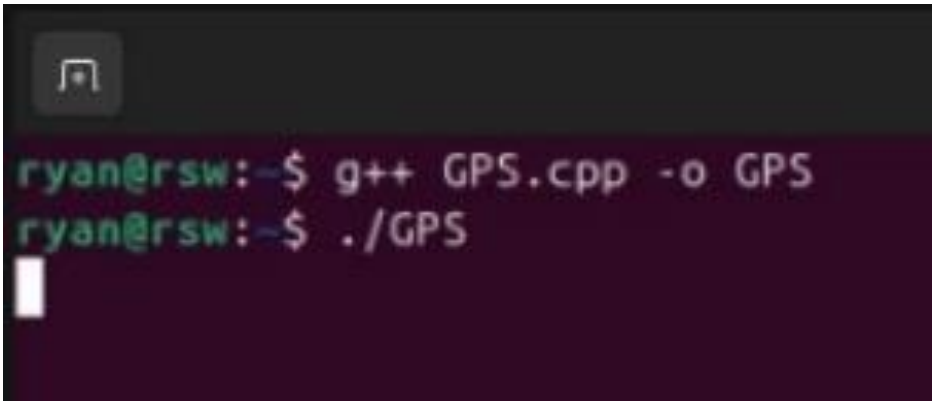
- Port serial gagal dibuka (gagal membuka port)
  Pastikan untuk chek port terlebih dahulu apakah terbaca atau tidaknya menggunakan ls /dev/tty*, jika tidak terbaca silahkan dichek kabel sesuai kebutuhan yang mendukung untuk pembacaan port, jika terbaca maka pastikan untuk tidak ada yang menggunakan port tersebut, dan berikut adalah gambar dari port tidak terbaca

Gambar 15. Gagal membuka port serial

- Kesalahan sintaks program
  Pastikan untuk selalu dicompile menggunakan g++ setiap programnya selesai, berikut adalah gambar dari syntax yang bermasalah

Gambar 16. Kesalahan sintaks program

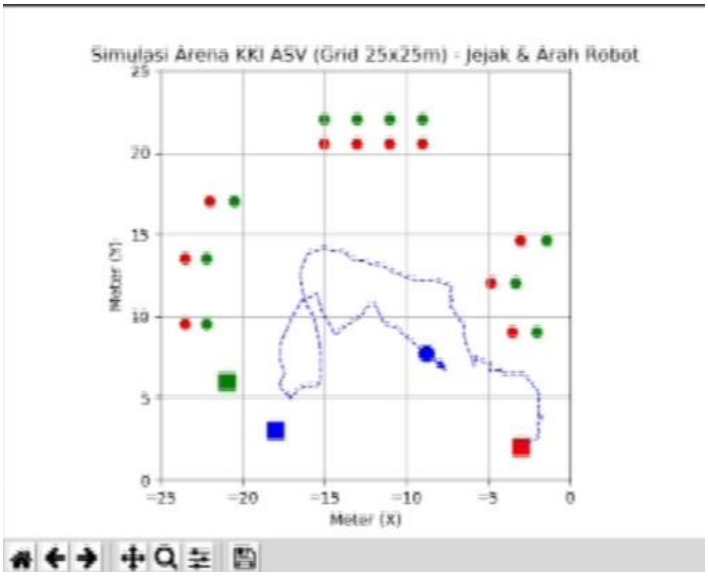
- GPS tidak memberikan data
  Hal ini terjadi dikarenakan gps tidak menerima sinyal data dari satelit, maka pastikan posisikan gps berada di tempat yang terbuka dan tidak tertutup seperti, dalam rumah yang bertingkat, dibawah pohon yang tertutup, dan dengan cuaca yang baik,.



Gambar 17. Tidak mendapatkan data

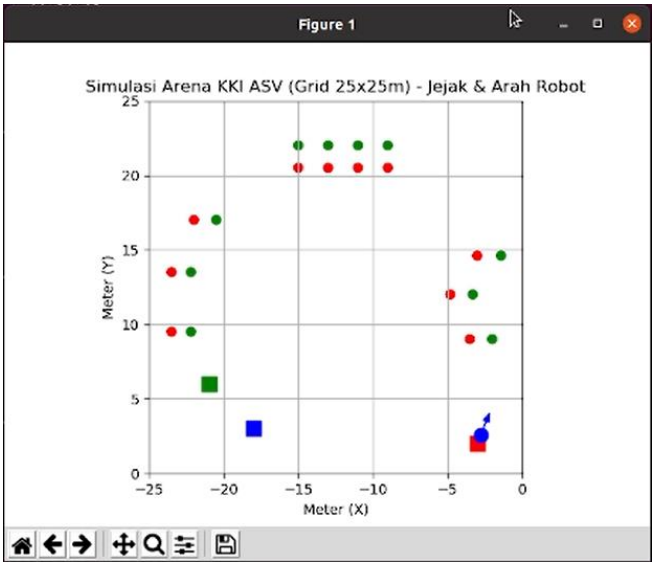
- Data GPS berantakan
  Jika mendapatkan data namun berantakan nilai x dan y nya seperti jika posisi diam

dan data bisa melompat jauh hingga 5 meter lebih, dikarenakan ada banyak hal yang menyebabkan data x dan y berantakan seperti berada didalam rumah yang masih mendapatkan sinyal, berada di tempat yang banyak jaring jaring logam, dan berikut adalah gambar Ketika tidak menerima data gps.



Gambar 18. Error data GPS

- Visualisasi tidak muncul atau error file
  hal tersebut terjadi jika file position.txt dihasilkan dari GPS.cpp, dan jangan tutup terminal saat dalam proses menjalankan program GPS.cpp
- Plot tidak bergerak
  dikarenakan plot tidak menerima data dari position.txt yang dikirim oleh GPS.cpp, maka pastikan port terbaca, data gps bener bener mendapatkan sinyal dari satelit dan dapat dipastikan dengan indicator lampu yang berkedip, selanjutnya pastikan GPS berada diruangan terbuka.



Gambar 19. Agent tidak menampilkan aksi

5. *Source Code* Program

5.1 Tautan *Source Code* Program

Program Sistem *Trajectory Mapping* Berbasis GPS dan FTDI dapat diunduh pada tautan berikut ini.

https://github.com/barelangmrt-robotics/Sistem-Trajectory-Mapping-Berbasis-GPS-dan-FTDI/tree/237ba19bf106e381890f286c588ff67c97368be9/Program%20Sistem

5.2 Skrip Program GPS.cpp

Skrip Program GPS.cpp adalah sebagai berikut.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <cmath>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <sstream>

#define EARTH_RADIUS 6371000.0

bool origin_set = false;
double origin_utm_x = 0.0;
double origin_utm_y = 0.0;

// ==========================
// Konversi dan perhitungan
// ==========================

std::string convertUTCtoLocal(const std::string& timeUTC, int offsetHours) {
    if (timeUTC.length() < 6) return "";
    int hour = std::stoi(timeUTC.substr(0, 2));
    int minute = std::stoi(timeUTC.substr(2, 2));
    int second = std::stoi(timeUTC.substr(4, 2));
    hour += offsetHours;
    if (hour >= 24) hour -= 24;
    std::ostringstream oss;
    oss << std::setw(2) << std::setfill('0') << hour << ":"
        << std::setw(2) << minute << ":"
        << std::setw(2) << second;
    return oss.str();
}

std::string convertToDecimalDegrees(const std::string& raw, const std::string& dir)
{
    if (raw.empty() || raw.find('.') == std::string::npos || raw.length() < 4)
return "";
    double deg = std::stod(raw.substr(0, raw.find('.') - 2));
    double min = std::stod(raw.substr(raw.find('.') - 2));
    double dec = deg + (min / 60.0);
    if (dir == "S" || dir == "W") dec *= -1;
    std::ostringstream oss;
    oss << std::fixed << std::setprecision(6) << dec;
    return oss.str();
}

// ==========================
// Konversi Ke UTM
// ==========================

void latLonToUTM(double lat, double lon, double& utm_x, double& utm_y, int& zone,
char& zone_letter) {
    const double a = 6378137.0;
    const double f = 1 / 298.257223563;
    const double k0 = 0.9996;
    const double e2 = f * (2 - f);
    const double n = f / (2 - f);
    const double e4 = e2 * e2;
    const double e6 = e4 * e2;

    zone = int((lon + 180) / 6) + 1;
    double lambda0 = (zone - 1) * 6 - 180 + 3;
```

```
    lambda0 *= M_PI / 180.0;

    double phi = lat * M_PI / 180.0;
    double lambda = lon * M_PI / 180.0;

    double N = a / sqrt(1 - e2 * sin(phi) * sin(phi));
    double T = tan(phi) * tan(phi);
    double C = e2 / (1 - e2) * cos(phi) * cos(phi);
    double A = cos(phi) * (lambda - lambda0);

    double M = a * ((1 - e2 / 4 - 3 * e4 / 64 - 5 * e6 / 256) * phi
        - (3 * e2 / 8 + 3 * e4 / 32 + 45 * e6 / 1024) * sin(2 * phi)
        + (15 * e4 / 256 + 45 * e6 / 1024) * sin(4 * phi)
        - (35 * e6 / 3072) * sin(6 * phi));

    utm_x = k0 * N * (A + (1 - T + C) * pow(A, 3) / 6
        + (5 - 18 * T + T * T + 72 * C - 58 * e2 / (1 - e2)) * pow(A, 5) / 120) +
500000.0;

    utm_y = k0 * (M + N * tan(phi) * (A * A / 2
        + (5 - T + 9 * C + 4 * C * C) * pow(A, 4) / 24
        + (61 - 58 * T + T * T + 600 * C - 330 * e2 / (1 - e2)) * pow(A, 6) /
720));

    if (lat < 0)
        utm_y += 10000000.0;

    const char letters[] = "CDEFGHJKLMNPQRSTUVWX";
    int idx = int((lat + 80) / 8);
    zone_letter = (idx >= 0 && idx < 20) ? letters[idx] : 'Z';
}

// ========================
// Program Utama
// ========================

int main() {
    const char* port = "/dev/ttyUSB0";
    int serial_port = open(port, O_RDWR | O_NOCTTY | O_SYNC);
    if (serial_port < 0) {
        std::cerr << "Gagal membuka port serial.\n";
            return 1;
        }

        termios tty{};
        tcgetattr(serial_port, &tty);
        cfsetospeed(&tty, B9600);
        cfsetispeed(&tty, B9600);
        tty.c_cflag |= (CLOCAL | CREAD);
        tty.c_cflag &= ~PARENB;
        tty.c_cflag &= ~CSTOPB;
        tty.c_cflag &= ~CSIZE;
        tty.c_cflag |= CS8;
        tty.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
        tty.c_oflag &= ~OPOST;
        tty.c_cc[VMIN] = 1;
        tty.c_cc[VTIME] = 1;
        tcsetattr(serial_port, TCSANOW, &tty);

        char buf = '\0';
        std::string line;

        std::cout << std::fixed << std::setprecision(6);

        while (true) {
            int n = read(serial_port, &buf, 1);
            if (n > 0) {
                if (buf == '\n') {
                    if (line.find("$GPRMC") == 0) {
                        std::stringstream ss(line);
                        std::string token, timeUTC, status, lat, latDir, lon,
lonDir;
                        std::getline(ss, token, ','); // $GPRMC
                        std::getline(ss, timeUTC, ',');
                        std::getline(ss, status, ',');
                        std::getline(ss, lat, ',');
                        std::getline(ss, latDir, ',');
                        std::getline(ss, lon, ',');
                        std::getline(ss, lonDir, ',');

                        std::string timeLocal = convertUTCtoLocal(timeUTC, 7);
                        std::string lat_str = convertToDecimalDegrees(lat,
latDir);
                        std::string lon_str = convertToDecimalDegrees(lon,
lonDir);

                        if (!lat_str.empty() && !lon_str.empty()) {
```

```
                                double lat_d = std::stod(lat_str);
                                double lon_d = std::stod(lon_str);

                                double utm_x, utm_y;
                                int zone;
                                char zone_letter;
                                latLonToUTM(lat_d, lon_d, utm_x, utm_y, zone,
    zone_letter);

                                if (!origin_set) {
                                    origin_utm_x = utm_x;
                                    origin_utm_y = utm_y;
                                    origin_set = true;
                                }

                                double rel_x = utm_x - origin_utm_x;
                                double rel_y = utm_y - origin_utm_y;

                                std::ofstream pos_file("position.txt");
                                if (pos_file.is_open()) {
                                    pos_file << rel_x << " " << rel_y << std::endl;
                                    pos_file.close();
                                }

                                std::cout << "Time       : " << timeLocal << "\n";
                                std::cout << "Position   : " << lat_d << ", " <<
    lon_d << "\n";
                                std::cout << "UTM Zone   : " << zone << zone_letter
    << "\n";
                                std::cout << "UTM X (E)  : " << utm_x << " m\n";
                                std::cout << "UTM Y (N)  : " << utm_y << " m\n";
                                std::cout << "X          : " << rel_x << " m\n";
                                std::cout << "Y          : " << rel_y << " m\n";
                                std::cout << "----------------------------\n";
                            }
                        }
                        line.clear();
                    } else if (buf != '\r') {
                        line += buf;
                    }
                }
            }

            close(serial_port);
            return 0;
        }

        </sstream></unistd.h></termios.h></fcntl.h></cmath></string></fstream></ioma
    nip></iostream>
```

5.3 Skrip Program monitoring.py

Skrip Program monitoring.py.

```
1.  import matplotlib.pyplot as plt
2.  import matplotlib.animation as animation
3.  import os
4.  import math
5.
6.  initial_position = None
7.
8.  # Ukuran grid arena (25m x 25m)
9.  grid_size = 25
10.
11. # Titik bola merah
12. red_balls = [(-3.5, 9), (-4.8, 12),(-3, 14.6), (-9, 20.5), (-11 ,20.5), (-13,20.5),
    (-15,20.5), (-22,17),(-23.5, 13.5),(-23.5, 9.5)]
13. # Titik bola hijau
14. green_balls = [(-2, 9), (-3.3, 12), (-1.4, 14.6), (-9,22),(-11,22),(-13,22),(-15,
    22),(-20.5,17),(-22.2,13.5),(-22.2,9.5)]
15. # Kotak start dan finish
16. red_square = (-3, 2)
17. blue_square = (-18, 3)
18. green_square = (-21, 6)
19.
20. # Setup plot
21. fig, ax = plt.subplots()
22. ax.set_xlim(-grid_size, 0)
23. ax.set_ylim(0, grid_size)
24. ax.set_aspect('equal')
```

```
ax.grid(True)

# Tambahkan elemen tetap
for x, y in red_balls:
    ax.add_patch(plt.Circle((x, y), 0.3, color='red'))
for x, y in green_balls:
    ax.add_patch(plt.Circle((x, y), 0.3, color='green'))
ax.add_patch(plt.Rectangle((red_square[0]-0.5, red_square[1]-0.5), 1, 1,
color='red'))
ax.add_patch(plt.Rectangle((blue_square[0]-0.5, blue_square[1]-0.5), 1, 1,
color='blue'))
ax.add_patch(plt.Rectangle((green_square[0]-0.5, green_square[1]-0.5), 1, 1,
color='green'))

# Titik robot (bergerak)
robot_dot, = ax.plot([], [], 'bo', markersize=10)
# Jejak lintasan
path_x, path_y = [], []
path_line, = ax.plot([], [], 'b--', linewidth=1)
# Arah panah
arrow = ax.arrow(0, 0, 0, 0, head_width=0.3, head_length=0.5, fc='blue', ec='blue')

# Fungsi update animasi
def update(frame):
    global arrow, initial_position
    if not os.path.exists("position.txt"):
        return robot_dot, path_line, arrow

    try:
        with open("position.txt", "r") as f:
            line = f.readline().strip()
            if line:
                x_str, y_str = line.split()
                x = float(x_str)
                y = float(y_str)

                # Inisialisasi posisi awal (saat pertama kali baca data)
                if initial_position is None:
                    initial_position = (x, y)

                # Hitung offset relatif posisi sekarang terhadap posisi awal
                dx = x - initial_position[0]
                dy = y - initial_position[1]

                # Jika posisi awal adalah (0,0), pindahkan ke koordinat kotak merah
                # Posisi robot di plot = kotak merah + offset
                robot_x = red_square[0] + dx
                robot_y = red_square[1] + dy

                # Update posisi robot
                robot_dot.set_data(robot_x, robot_y)

                # Simpan jejak
                path_x.append(robot_x)
                path_y.append(robot_y)
                path_line.set_data(path_x, path_y)

                # Update arah jika memungkinkan
                if len(path_x) >= 2:
                    vx = path_x[-1] - path_x[-2]
                    vy = path_y[-1] - path_y[-2]
                    mag = math.hypot(vx, vy)
                    if mag > 0.01:
                        vx /= mag
                        vy /= mag
                        arrow.remove()
                        arrow = ax.arrow(robot_x, robot_y, vx*1.0, vy*1.0,
head_width=0.4, head_length=0.6, fc='blue', ec='blue')

    except Exception as e:
        print("Error reading file:", e)

    return robot_dot, path_line, arrow


# Judul dan label
plt.title("Simulasi Arena KKI ASV (Grid 25x25m) - Jejak & Arah Robot")
plt.xlabel("Meter (X)")
plt.ylabel("Meter (Y)")

        # Jalankan animasi
        ani = animation.FuncAnimation(fig, update, interval=500)
        plt.show()
```