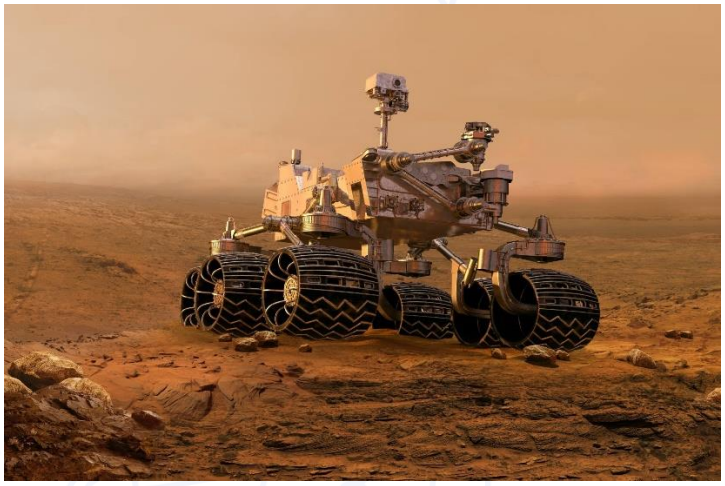
Tugas Post-Day Modul 3 Programming

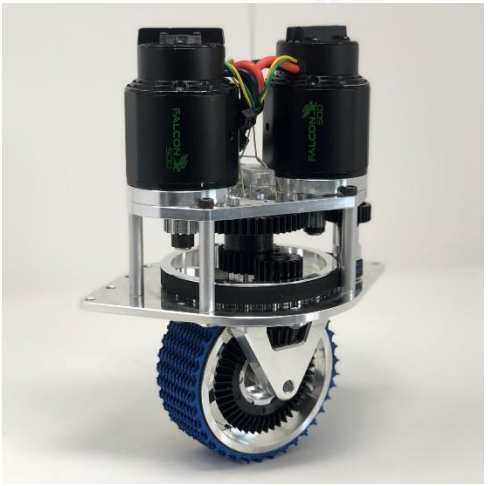
Object-Oriented Programming

Haloo Cakru 16! Setelah Anda mempelajari OOP, sekarang adalah waktunya untuk Anda mengaplikasikan ilmu yang telah Anda pelajari! Pembuatan kode program ditulis dalam bahasa C++ dengan tools/library yang dibebaskan. Tugas ini adalah tugas individu dan plagiarisme akan menyebabkan nilai tugas ini menjadi nol.



Gambar 1 Ilustrasi rover

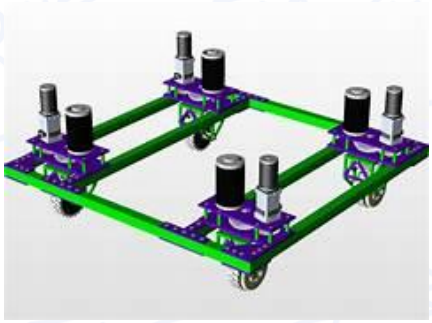
Jauh di luar angkasa sana, kita memerlukan suatu rover untuk melakukan eksplorasi. Guna memastikan bahwa rover ini dapat berjalan dengan baik, anda diminta untuk mensimulasikan kinematika dari rover tersebut. Salah satu hal yang perlu Anda cermati adalah drive train rover ini menggunakan system *swerve drive*.



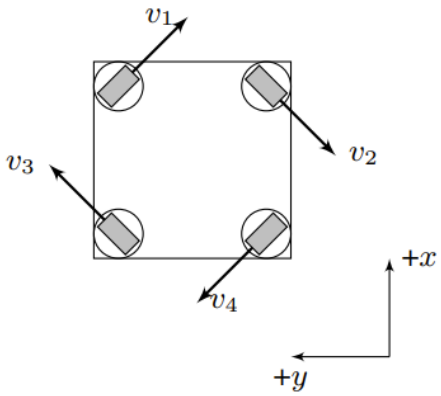
Gambar 2 Module swerve drive

System swerve drive adalah jenis drive train yang memiliki dua degree of freedom. Artinya, untuk menggunakan system swerve drive setidaknya memerlukan dua motor. Terdapat satu motor untuk membuat roda berotasi dan ada juga satu motor untuk memutar arah menghadapnya roda. Kelebihan dari system swerve, yaitu arah tiap roda dapat berubah secara independen dari chasis-nya. Agar Anda semakin memiliki gambaran mengenai system swerve drive, tontohlah video berikut
https://youtu.be/J1Z5rDxcSNk?si=dCEOHB0B09gZrAY3 dan
https://youtube.com/shorts/16NsjuQYUrI?si=cW5QdTRhy5fw7nzi

Permodelan system swerve yang akan kita gunakan seperti berikut



Gambar 3 Chasis robot



Gambar 4 Swerve drive free body diagram

Berikut adalah kinematika dari swerve drive

$$\begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r_{1y} \\ 0 & 1 & r_{1x} \\ 1 & 0 & -r_{2y} \\ 0 & 1 & r_{2x} \\ 1 & 0 & -r_{3y} \\ 0 & 1 & r_{3x} \\ 1 & 0 & -r_{4y} \\ 0 & 1 & r_{4x} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Keterangan: v_x, v_y, dan ω adalah kecepatan pada center dari chasis. Lalu r_{nx} dan r_{ny} secara berturut-turut adalah jarak dari center chasis ke roda *n* pada arah sumbu x dan sumbu y. Kemudian v_{nx} dan v_{ny} secara beruturut-turut adalah kecepatan roda *n* pada arah sumbu x dan sumbu y. (perhatikan sumbu x dan y mengikuti sumbu yang ditampilkan pada free body diagram).

1. Operasi Matrix

Objective: Buatlah class bernama Matrix yang isinya terdapat implementasi penjumlahan, pengurangan, dan perkalian dua matrix.

Implementasi pada class ini memerlukan beberapa constructor. Buatlah

- constructor yang dapat menerima input dalam bentuk array 2d.
- constructor yang dapat menerima input banyaknya baris dan kolom
- constructor yang dapat men-copy objek dengan class yang sama

Lalu operasi penjumlahan, pengurangan, dan perkalian matrix ***haruslah menggunakan operator overloading***. Berikut adalah contoh behaviour dari class Matrix yang harus Anda buat

```
int main()
{
    Matrix A = {{1, 2, 3}, {2, 3, 1}};
    Matrix B = {{0, 2, 4}, {1, 2, 5}, {8, 2, 1.2}};
    Matrix C (2, 3); // Parameter constructor adalah (banyak baris, banyak kolom) dan isi
dari matrix ini adalah angka nol
    std::cout << "Matrix A\n";
    A.display(); // print matrix A
    std::cout << "Matrix B\n";
    B.display();
    std::cout << "Matrix C\n";
    C.display();
    std::cout << '\n';

    Matrix C1 = A + C;
    std::cout << "Matrix C1\n";
    C1.display();
    std::cout << '\n';

    Matrix C2 = A - C;
    std::cout << "Matrix C2\n";
    C2.display();
    std::cout << '\n';

    Matrix C3 = A * B;
    std::cout << "Matrix C3\n";
    C3.display();
    std::cout << '\n';
```

```
    Matrix C4 = B * A;
    std::cout << "Matrix C4\n";
    C4.display();
    std::cout << '\n';

    return 0;
}
```

Akan mengeluarkan output

```
Matrix A
1 2 3
2 3 1
Matrix B
0 2 4
1 2 5
8 2 1.2
Matrix C
0 0 0
0 0 0

Matrix C1
1 2 3
2 3 1

Matrix C2
1 2 3
2 3 1

Matrix C3
26 12 17.6
11 12 24.2
```

```
Multiplication cannot be done. The number of columns in
the first matrix should be equal to the number of rows in
the second.
Return 1 by 1 of zeros matrix
Matrix C4
0
```

Note:

Jika penjumlahan atau pengurangan matrix tidak bisa dilakukan karena dimensi yang tidak cocok maka tolong berikan peringatan

“The matrix dimension is not valid!”

Dan return-kanlah matrix 1x1 dengan isi cell bernilai 0

Lakukanlah hal yang serupa pada perkalian matrix, hanya saja pesannya dirubah menjadi

“Multiplication cannot be done. The number of columns in the first matrix should be equal to the number of rows in the second”

2. Class Swerve

Objective: Buatlah class bernama Swerve yang isinya mencakup informasi mengenai posisi dan kecepatan dari chasis dan kecepatan roda

$$\begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r_{1y} \\ 0 & 1 & r_{1x} \\ 1 & 0 & -r_{2y} \\ 0 & 1 & r_{2x} \\ 1 & 0 & -r_{3y} \\ 0 & 1 & r_{3x} \\ 1 & 0 & -r_{4y} \\ 0 & 1 & r_{4x} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Data member dari class Swerve harus mencakup keseluruhan variable pada persamaan di atas. Kemudian, diketahui bahwa chasis swerve berbentuk persegi sehingga $\boldsymbol{r_{nx}}$ **dan** $\boldsymbol{r_{ny}}$ **bernilai sama, yaitu 0.26363 meter**. Pada kasus kita, kita belum mengetahui nilai v_{nx} dan v_{ny}. Hal yang kita ketahui (akan diberikan kepada Anda, lihat lah problem 3, simulasi) adalah nilai v_x, v_y, dan ω pada setiap waktunya. Oleh karena itu, buatlah member function seperti berikut

- velocityCommand(float vx, float vy, float omega)
  fungsi ini akan menghitung nilai v_{nx} dan v_{ny} dengan *n* menyatakan roda ke-n. Gunakanlah modul class Matrix yang telah Anda buat sebelumnya, untuk melakukan operasi perkalian matrix. Hasil perhitungan v_{nx} dan v_{ny} lalu disimpan dalam member data pada class Swerve ini. Anda dapat melakukan perhitungan dengan rumus berikut

$$\begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r_{1y} \\ 0 & 1 & r_{1x} \\ 1 & 0 & -r_{2y} \\ 0 & 1 & r_{2x} \\ 1 & 0 & -r_{3y} \\ 0 & 1 & r_{3x} \\ 1 & 0 & -r_{4y} \\ 0 & 1 & r_{4x} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

- updatePose(float deltaTime)
  Berdasarkan informasi v_x, v_y, dan ω yang diberikan pada tiap-tiap waktunya, hitunglah posisi chasis robot setiap waktunya. Perhatikan rumus berikut

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} (deltaTime)$$

Kumpulan posisi (x, y, θ) biasanya disebut *pose*. Perhatikan bahwa dengan menggunakan perumusan di atas, Anda dapat menghitung pose setiap waktunya dengan cara, nilai pose sebelumnya, dijumlah dengan kecepatan kali waktu.

3. Simulasi

Objective: Membuat grafik posisi chasis dan kecepatan resultan dari tiap-tiap roda

Anda diberikan fungsi v_x dan v_y sebagai berikut

$$v_x = 48\ \cos(t)\ \sin^2(\mathrm{t})$$

$$v_y = 4\ \sin(4t) + 6\sin(3\mathrm{t}) + 10\sin(2t) - 13\sin(t)$$

$$\omega = 0$$

Dengan konfigurasi pose awal adalah $x_o = 0$, $y_o = 5$, dan $\theta_o = 0$. (variabel t adalah waktu)

Buatkanlah simulasi dengan time bound dari 0 detik sampai 6,3 detik dengan increment waktu adalah 0.01 detik. Pada tiap-tiap detik tersebut hitunglah hal-hal berikut

- Hitunglah posisi dari chasis robot pada tiap-tiap waktunya. Gunakan fungsi updatePose() yang didalamnya terdapat implementasi berikut

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} (deltaTime)$$

- Hitunglah kecepatan resultan dari tiap-tiap roda. Ingat, Anda telah membuat fungsi velocityCommand() yang di dalamnya terdapat perumusan berikut

$$\begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r_{1y} \\ 0 & 1 & r_{1x} \\ 1 & 0 & -r_{2y} \\ 0 & 1 & r_{2x} \\ 1 & 0 & -r_{3y} \\ 0 & 1 & r_{3x} \\ 1 & 0 & -r_{4y} \\ 0 & 1 & r_{4x} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Perhatikan, yang perlu anda hitung adalah resultan kecepatannya. Resultan kecepatan didapatkan dengan menggunakan perumusan berikut

$$v_n = \sqrt{(v_{nx})^2 + \left(v_{ny}\right)^2}$$

dengan n merujuk pada roda ke-n

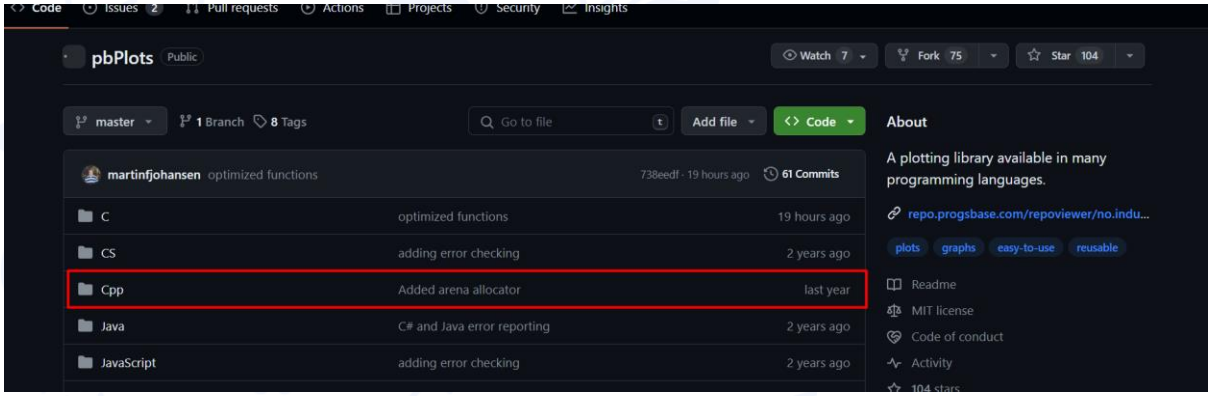
Setelah Anda berhasil menghitung keseluruhan hal yang diminta, tugas Anda adalah membuat grafik-nya. Buatlah grafik

- Grafik posisi y terhadap x
- Grafik v_1 terhadap waktu
- Grafik v_2 terhadap waktu
- Grafik v_3 terhadap waktu
- Grafik v_4 terhadap waktu

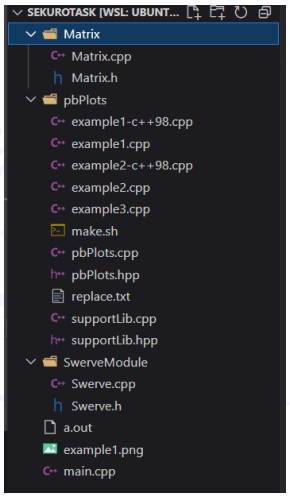
Cara untuk membuat grafik dapat dilihat dari tutorial berikut
https://youtu.be/RNKVHQzvaRM?si=d4wgEo_7FKCJAX94

Pada tutorial tersebut digunakan tools yang bernama pbPlot, Anda tidak harus terpatok pada library ini, Anda dibebaskan untuk menggunakan tools apapun dalam pembuatan grafik. Jika Anda menggunakan library pbPlot, berikut adalah tata cara pembuatan grafik

Clone-lah repository ini https://github.com/InductiveComputerScience/pbPlots.git. Perhatikan pada repository tersebut terdapat folder bernama CPP



Copy lah folder Cpp ke dalam workspace Anda dan silahkan ubah nama foldernya sesuka hati Anda. Contohnya misal seperti berikut saya merubah foldernya menjadi bernama pbPlots



Note: Pembuatan struktur folder pada tugas ini dibebaskan dan tidak diharuskan dibagi-bagi dalam beberapa folder yang berbeda seperti pada contoh struktur folder di atas.

Berikut adalah contoh penggunaan library pbPlots

```
#include "pbPlots.hpp"
#include "supportLib.hpp"

int main(){
    StringReference *errorMessage = CreateStringReferenceLengthValue(0, L' ');
    RGBABitmapImageReference *imageReference = CreateRGBABitmapImageReference();

    std::vector<double> xs{-2, -1, 0, 1, 2};
    std::vector<double> ys{2, -1, -2, -1, 2};

    // Pembuatan grafik
    DrawScatterPlot(imageReference, 600, 400, &xs, &ys, errorMessage);

    // Simpan grafik dalam format PNG
    std::vector<double> *pngdata = ConvertToPNG(imageReference->image);
    WriteToFile(pngdata, "example1.png"); // simpan nama file "example1.png"
    DeleteImage(imageReference->image);

    FreeAllocations();

    return 0;
```

Dalam pembuatan grafik, anda tinggal ubah array xs dan ys saja. Tidak perlu merubah hal-hal lainnya.

Pengumpulan tugas ini melalui repository yang telah kalian buat. Struktur folder pengumpulannya adalah seperti berikut

```
Tugas-Modul-OOP
      Tugas-OOP
          -- main.cpp
          -- (disesuaikan dengan struktur folder Anda)
      Hasil-Grafik
          -- grafik_y_terhadap_x.png
          -- grafik_v1_terhadap_time.png
          -- grafik_v2_terhadap_time.png
          -- grafik_v3_terhadap_time.png
          -- grafik_v4_terhadap_time.png
```

Note: Letakkan keseluruhan kerjaan Anda pada folder Tugas-OOP. Lalu, letakkan grafik hasil simulasi pada folder Hasil-Grafik