# URO TEST


INSTITUT TEKNOLOGI BANDUNG

2024

# PROGRAMMING

Selamat Datang CaKRU-17 !!
URO Test Divisi Programming, merupakan bagian dari proses seleksi URO pada divisi programming dimana CaKru-17 diwajibkan untuk memenuhi beberapa ketentuan berikut :

**PETUNJUK PENGERJAAN**

1. Peserta wajib mengerjakan semua soal, tidak menjawab sekurang-kurangnya 1 soal maka diberikan nilai 0
2. Peserta mengerjakan URO Test dengan serius dimana penilaian ini akan mempengaruhi kelulusan hingga tahap penentuan tim
3. Peserta mengerjakan URO Test pada waktu yang telah ditentukan, hingga 8 Oktober 2024 pukul 23.55 WIB. Peserta yang mengumpulkan melewati tenggat waktu maka dianggap tidak mengumpulkan
4. Peserta disarankan untuk memaksimalkan kemampuan riset dan mencari tahu sumber informasi terkait permasalahan yang ditemui pada URO Test di kanal youtube ataupun sumber lainnya
5. Peserta dilarang keras untuk menggunakan AI untuk menjawab soal-soal yang melibatkan kemampuan riset dan teknis
6. Semua Pengumpulan dilakukan dengan sebuah link Github dimana mencakup 4 folder utama pada sebuah repository :
   *Soal_1_(Nama)_(NIM)_Cakru17*
   *Soal_2_(Nama)_(NIM)_Cakru17*
   *Soal_3_(Nama)_(NIM)_Cakru17*
   *Soal_4_(Nama)_(NIM)_Cakru17*
7. Peserta mengumpulkan link github URO Test tersebut pada form : link berikut ini
8. Pastikan semua file/link yang dikumpul, aksesnya dapat dibuka dengan baik

## SOAL 1

a. Apa itu ROS (Robot Operating System), dan bagaimana peran utamanya dalam pengembangan robotik modern? (Jelaskan mengapa ROS penting untuk integrasi berbagai komponen robot seperti sensor, aktuator, dan kamera dalam satu sistem yang bekerja harmonis.)
b. Apa perbedaan utama antara ROS dan ROS2, dan mengapa pengembang cenderung memilih ROS2 untuk proyek baru? (Jelaskan keunggulan ROS2 dibandingkan ROS dalam hal performa, keamanan, dan pemeliharaan jangka panjang.)
c. Mengapa simulasi robotik penting dalam pengembangan robot, dan apa keuntungan menggunakan simulasi sebelum membangun robot fisik? (Berikan contoh kasus di mana simulasi dapat menghemat waktu dan biaya pengembangan robot.)
d. Apa itu Gazebo, dan bagaimana Gazebo digunakan untuk mensimulasikan lingkungan fisik bagi robot? (Jelaskan langkah-langkah dasar mengintegrasikan ROS dengan Gazebo untuk mengontrol robot dalam simulasi.)
e. Bagaimana cara kerja navigasi robot di dunia simulasi? (Apa saja konsep dasar seperti mapping dan lokalisasi yang perlu dipahami, dan bagaimana fitur ini dapat diimplementasikan pada robot Anda?)
f. Apa itu TF (Transform) dalam konteks ROS, dan bagaimana TF membantu robot memahami posisi dan orientasinya dalam ruang tiga dimensi? (Berikan contoh bagaimana TF digunakan untuk memastikan robot bergerak dengan benar dalam simulasi.)
g. (Opsional, tidak wajib dikerjakan, boleh dikerjakan untuk memaksimalkan kemampuan simulasi dan implementasi ROS) : Perhatikan Lampiran 1

**Format Pengumpulan :**
*Folder Soal_1_(Nama)_(NIM)_Cakru17* pada Github
yang berisikan :
*Jawaban_1* (file pdf yang berisi jawaban a-f)
bedakan jawaban untuk soal opsional (1.g), untuk bagian (1.g) format pengumpulan mengikuti lampiran 1

## SOAL 2

Buatlah sebuah game sederhana menggunakan Python/C++ dengan penerapan prinsip Object-Oriented Programming (OOP). Game yang dibuat adalah "Battle of Robots" di mana pemain akan mengendalikan robot untuk bertarung melawan robot musuh.

OOP diterapkan dengan penggunaan kelas dan objek untuk mendefinisikan robot, enkapsulasi untuk menyembunyikan detail internal dan mengakses atribut serta metode, abstraksi untuk menyederhanakan interaksi dengan objek, dan komposisi untuk mengelola interaksi antar kelas seperti robot dan pertarungan, dengan

**Ketentuan :**

- Implementasikan Class “Robot”: Implementasikan atribut dan metode sesuai dengan spesifikasi.
- Implementasikan Class “Battle”: Implementasikan metode “start_fight” untuk menjalankan pertarungan antara dua robot.
- Implementasikan Class “Game” : Implementasikan metode “add_robot” untuk menambahkan robot ke dalam daftar, Implementasikan metode “start_game” untuk memulai game dengan memilih dua robot untuk bertarung.

**Uji Program:**

- Buat beberapa objek Robot dengan berbagai atribut.
- Tambahkan robot-robot tersebut ke dalam objek Game.
- Jalankan game dan lakukan pertarungan antar robot.

**Output yang diharapkan kurang lebih seperti ini :**

*Choose robots for the battle:*
*1. RoboOne*
*2. RoboTwo*
*3. RoboThree*
*Select the first robot: 1*
*Select the second robot: 2*
*Battle Start!*
*RoboOne attacks RoboTwo for 23 damage!*
*RoboTwo attacks RoboOne for 14 damage!*
*RoboOne attacks RoboTwo for 7 damage!*

*RoboTwo attacks RoboOne for 18 damage!*
*RoboOne attacks RoboTwo for 28 damage!*
*RoboTwo attacks RoboOne for 8 damage!*
*RoboOne attacks RoboTwo for 30 damage!*
*RoboTwo is defeated!*
*RoboOne wins!*

**POIN BONUS KE NILAI KESELURUHAN URO TEST (OPSIONAL) :**

- Peserta mampu menambahkan fitur tambahan seperti berbagai jenis serangan, kemampuan khusus untuk setiap robot, dan lain-lain
- Peserta mampu membuat antarmuka grafis atau perbaikan pada cara pengguna memilih robot untuk bertarung
- Peserta mampu membangun algoritma untuk pertarungan yang lebih realistis atau menambahkan strategi berdasarkan atribut robot
- Peserta mampu mengimplementasikan kreativitas lainnya

**Format Pengumpulan :**
*Folder Soal_2_(Nama)_(NIM)_Cakru17* pada Github
yang berisikan :
*Jawaban_2* (program)
*readme program* (penjelasan program)

## SOAL 3

Programming memang menjadi komponen utama pada kecerdasan robot. Namun, robot tidak dapat melakukan tugasnya/mengaktifkan kecerdasan yang telah dibangun tanpa komponen yang melibatkan kecerdasan tersebut atau dengan kata lain, **yang mengatur kecerdasan** tersebut digunakan untuk memerintahkan komponen robot lainnya. Sistem pada robot memerlukan komponen elektrikal dan mekanikal.

Programming di URO setidaknya mampu menguasai sistem *Single Board Computer* dan Mikrokontroler.

a. Pada **TinkerCad**, Buatlah sebuah proyek sederhana yang melibatkan mikrokontroler dan beberapa sensor dengan pemrograman arduino sederhana, buatlah presentasi selama kurang lebih 5 menit tentang proyek dan upload di YT !
https://www.tinkercad.com/

b. Lakukan **Perbandingan** pada semua Single Board Computer yang kamu tau ! (Kelebihan, Kekurangan, Cocok dipakai untuk apa saja, dan lain-lain)

c. Jelaskan secara detail tentang **Raspberry Pi** dan **cara setupnya** ! (versi Raspi Dibebaskan)

d. Dynamixel AX-18A merupakan servo yang memiliki torsi yang cukup kuat dan sangat penting untuk pergerakan yang membawa bobot berat. Jelaskan secara bertahap **cara kontrol dan komunikasi servo** tersebut menggunakan sistem publish subscribe pada Raspberry Pi ! (perkirakan juga penggunaan komponen modul lainnya, U2D2 misalnya)

**Format Pengumpulan :**
*Folder Soal_3_(Nama)_(NIM)_Cakru17* pada Github
yang berisikan :
*Jawaban_3_a* (link yt)
*Jawaban_3_b* (PDF)
*Jawaban_3_c* (PDF)
*Jawaban_3_d* (PDF)

## SOAL 4

Algoritma dan sistem kontrol merupakan hal yang sangat penting dalam pengembangan robot.

a. Buatlah sebuah Blok Diagram pada draw.io dari 2 jenis sistem kontrol dan detail penjelasan skemanya !
   https://app.diagrams.net/
b. Jelaskan lebih jauh masing-masing algoritma di bawah dengan detail :
   - Kinematics (object detection, pose estimation, camera calibration)
   - ADRC (Active Disturbance Rejection Control)
   - PID (Proportional-Integral-Derivative) control algorithms
   - A* (A star) algorithm

**POIN BONUS KE NILAI KESELURUHAN URO TEST (OPSIONAL) :**
BUAT SEBUAH PROJECT YANG MENGGUNAKAN 1 ATAU LEBIH ALGORITMA DIATAS

**Format Pengumpulan :**
*Folder Soal_4_(Nama)_(NIM)_Cakru17* pada Github
yang berisikan :
*Jawaban_4_a* (pdf)
*Jawaban_4_b* (pdf)
*Jawaban_4_Bonus* (bebas apapun cara untuk menjelaskan proyekmu)

## LAMPIRAN 1

Buatlah sebuah robot (dibebaskan) pada **simulasi**, dengan Ketentuan :

- Misi Utama robot pada simulasi adalah **bebas** namun memiliki berbagai kelengkapan : Bentuk robot dibebaskan (boleh menggunakan mesh), Memiliki kemampuan bergerak dengan kontrol melalui keyboard/joystick, memiliki kamera sehingga mampu melakukan **Computer Vision**, memiliki kemampuan **mapping, lokalisasi dan navigasi** sehingga mampu memperkirakan koordinat dan bergerak otomatis, melibatkan action tertentu ketika menerima sebuah informasi (AUTOMATIC MODE).
- Semua pekerjaan Dilakukan pada **OS Ubuntu Linux**
- Simulasi Dijalankan pada **Gazebo**
- Menggunakan **Robot Operating System** sebagai Framework [Dibebaskan ROS/ROS2, disarankan untuk mengikuti versi terbaru (ROS2) dengan alasan pemeliharaan developer yang lebih baik

Kumpulkan Hasil Simulasi dalam bentuk github yang berisi (makalah, semua file project yang disusun rapi, link video YT presentasi)

Video YT berdurasi 5-15 menit, menjelaskan manajemen direktori, langkah-langkah menjalankan simulasi, penjelasan implementasi simulasi (hasil).

**File Makalah (PDF)** mencakup :

- Latar belakang dan tujuan robot tersebut dibuat
- Penjelasan Kenapa menggunakan versi Linux dan versi ROS/ROS2 tersebut
- Semua Tools Simulasi yang digunakan (Penjelasan Gazebo, URDF, Code, dan lainnya)
- Penjelasan manajemen direktori yang digunakan
- Keseluruhan Sistem Robot (programming, elektrik, mekanik) [JIKA KAMU JADI KETUA TIM, MAKSIMALKAN ROBOTMU DENGAN MEMBERIKAN ARAHAN PADA DIVISI ELEKTRIK DAN MEKANIK YANG ADA DI TIM MU, TULISKAN ARAHAN UNTUK 2 DIVISI LAIN TERSEBUT !]
- Flowchart Simulasi
- Komponen-komponen robot (link, joint, worldnya)
- TF pada RViz
- Movement Robot
- Sistem Navigasi robot
- Kamera dan Computer Vision pada robot

- Implementasi Sistem pada Simulasi (Hasil)
- Kendala
- Harapan dan Mimpi untuk robot tersebut (apakah mungkin direalisasikan dan dikembangkan di keadaan nyata?, pengembangan yang baik nya bagaimana ?, dan lain-lain)

**Format Pengumpulan :**

*Folder Soal_Opsional_(Nama)_(NIM)_Cakru17* pada Github
yang berisikan :

- SISTEM YANG RAPI YANG BERISIKAN SEMUA KOMPONEN KEPERLUAN SIMULASI (terapkan penggunaan github dan manajemen direktori yang baik dengan colcon/ lainnya dan ROS) [berisi semua folder dan file sistem]
- *Presentasi_1_(Nama)_(NIM)_Cakru17* (link yt presentasi)
- *Makalah_1_(Nama)_(NIM)_Cakru17* (file pdf makalah)
- *Instruksi_Menjalankan_Simulasi* (readme file) [bayangkan orang awam yang akan menjalankan, lengkap dengan apa saja yang harus di clone dari github, dll]

**Catatan :**

1. Bagi yang OS Windows, disarankan untuk melakukan dual boot untuk dapat menggunakan Linux Ubuntu

How to Dual Boot Windows 11 & Ubuntu Easily!

(disesuaikan dengan versi masing-masing)

2. Pada Linux 20.04 gunakan ROS/ROS2 yang sesuai dengan keperluan simulasi atau 22.04 untuk ROS2 Humble (dan versi terbarunya) untuk keperluan simulasi dengan fitur tertentu