



Arsitektur AI Hybrid & Conversational Loop

MODUL SOFTWARE ROLI

```
sode  
sodesee {  
  to teso-nas-hoe;  
  tosedate = 'acka-1'  
  code = 'agla'  
} 1-reeso;  
sowcelis: 'modul-1'
```



```
cede  
scenias {  
  //tetee-nas-no'  
  toeernee wname = cod1;  
} 1-woocost;  
senso; tat: 'toexl-?';  
reperst tech;
```

DOSEN UNY

Dr. Yeni Artanti, S.Pd., M.Hum.

MENTOR

Ivan Andremaulana

GURU SEKOLAH

Ichwan Aryono, S.Pd., M.Pd.Si.

**SISWA DIVISI
SOFTWARE**

Wilbert Reyhan Susabda
Aloysius Ivan Hanindito
Evan Alexander Dharma
Alraffi Belva Moreno

Dari nol baris kode, sampai RoLi bisa bicara

RoLi butuh "telinga", "otak", dan "mulut" sebelum ia bisa jadi teman ngobrol siswa dalam dua bahasa. Ini rekonstruksi materi 8 pertemuan Tim Software, disusun ulang dari kode

`RoboLingua_Prototype` yang benar-benar mereka bangun, untuk **4 siswa** yang mengerjakannya dari nol sampai robot bisa didemokan utuh.



`voice_listener` → `stt` → `brain` → `tts`

01 Pembukaan

02 Otak (LLM)

03 Suara

04 Wajah & UI

05 Video

06 Setup Wifi

07 Offline

08 Finishing

Kata Pengantar

Modul ini disusun sebagai bahan pengantar bagi siswa Tim Software, untuk dibaca sebelum memulai kegiatan praktik dalam program mentoring RoboLingua (RoLi), kerja sama Program Studi Linguistik Terapan Universitas Negeri Yogyakarta dengan SMA Negeri 3 Yogyakarta. Selain belajar robotika, program ini juga menghadirkan pertukaran bahasa dan budaya secara langsung lewat kunjungan penutur asli bahasa Prancis, Luis, di beberapa pertemuan.

Sebelum mulai menulis baris kode pertama, penting untuk memahami dulu gambaran besar tugas Tim Software: membangun "otak" dan "indera" RoLi, bagian yang membuat robot ini bisa mendengarkan, berpikir, dan menjawab. Modul ini membahas delapan pertemuan secara berurutan, mulai dari dasar kecerdasan buatan percakapan hingga menyatukan seluruh modul menjadi satu sistem yang utuh.

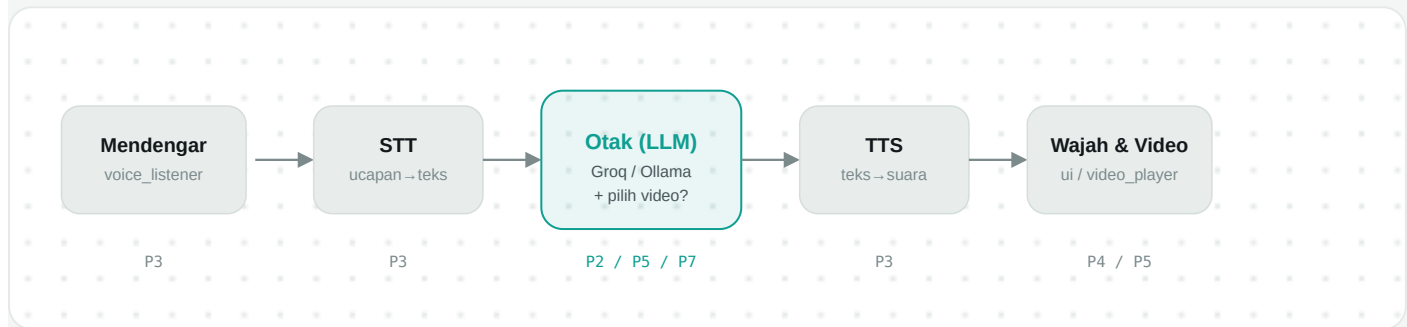
Setiap pertemuan disusun dengan tujuan pembelajaran, konsep kunci, alat & bahan, aktivitas hands-on, serta pertanyaan refleksi, supaya modul ini bisa langsung dipakai sebagai panduan belajar sebelum praktik dimulai. Selamat membaca, dan selamat memulai perjalanan membangun sistem percakapan RoLi.

Daftar Isi

- 01 Kata Pengantar
- 02 Satu Alur, Dibangun Bertahap
- 03 Pertemuan 1: Pembukaan & Brainstorming
- 04 Pertemuan 2: Otak Robot, LLM via API Groq
- 05 Pertemuan 3: Fitur Suara, STT & TTS
- 06 Pertemuan 4: Animasi & UI, Wajah RoLi
- 07 Pertemuan 5: Pemutar Video & Function Calling
- 08 Pertemuan 6: Setup Pembuka, Wifi, Mode, Tampilan Awal
- 09 Pertemuan 7: Versi Offline
- 10 Pertemuan 8: Sinkronisasi, Finishing, Bug Hunting

Satu alur, dibangun bertahap

Seluruh 8 pertemuan pada dasarnya membangun satu alur ini, sepotong demi sepotong.



01 Pembukaan & Brainstorming

Tujuan Pembelajaran

Siswa memahami gambaran besar RoLi: robot apa yang akan dibangun, untuk siapa, dan bagaimana 8 pertemuan ke depan akan berjalan.

Istilah Penting

Conversational AI

Program komputer yang dirancang untuk bisa "mengobrol" dengan manusia: memahami apa yang diucapkan, lalu merespons dengan cara yang terasa alami, bukan sekadar jawaban baku.

Modul (pemrograman)

Satu file kode kecil yang mengerjakan satu tugas spesifik dan jelas, misalnya `stt.py` yang hanya mengurus ucapan-jadi-teks, tanpa mencampuri tugas modul lain.

Brainstorming

Sesi diskusi bebas untuk mengumpulkan sebanyak mungkin ide tanpa buru-buru menilai benar-salah. Penyaringan mana ide yang akan benar-benar dikerjakan baru dilakukan setelahnya.

Arsitektur (sistem)

Rancangan besar tentang bagaimana bagian-bagian sebuah sistem saling terhubung dan bekerja sama, semacam "peta jalan" sebelum baris kode pertama ditulis.

Konsep Kunci

Conversational AI: program yang bisa memahami ucapan manusia dan merespons secara alami.

Alur Dengar–Pikir–Jawab: kerangka besar yang akan dibangun sepotong demi sepotong sepanjang 8 pertemuan.

Penjelasan Lengkap

RoLi bukan satu program tunggal. Ia adalah sekitar sepuluh program kecil yang saling berbicara lewat data: suara yang diubah jadi teks, teks yang diubah jadi keputusan, keputusan yang diubah jadi suara lagi, dan seterusnya. Delapan

pertemuan ke depan pada dasarnya adalah membangun program-program kecil itu satu per satu, lalu menyambungkannya jadi satu robot yang hidup.

Kenapa dipecah jadi banyak bagian kecil?

Kalau semua logika ditulis dalam satu file raksasa, satu kesalahan kecil saja bisa membuat seluruh robot berhenti bekerja. Karena itu, Tim Software memecah RoLi menjadi modul-modul terpisah yang masing-masing punya satu tanggung jawab jelas: `stt.py` cuma urusan mengubah suara jadi teks, `tts.py` cuma urusan mengubah teks jadi suara, `brain.py` cuma urusan "berpikir" dan memutuskan jawaban. Dalam rekayasa perangkat lunak, prinsip ini disebut *separation of concerns*, dan akan terus dipakai di setiap pertemuan berikutnya.

Peta yang akan kalian ikuti

Sepanjang 8 pertemuan, RoLi akan dibangun mulai dari "otak"-nya (bisa berpikir dan menjawab lewat teks), baru kemudian ditambah telinga dan mulut, lalu wajah, lalu kemampuan memutar video, lalu fitur-fitur pendukung seperti wifi dan mode offline, dan terakhir semuanya disatukan. Urutan ini bukan kebetulan: tiap pertemuan sengaja dibangun di atas fondasi pertemuan sebelumnya, supaya di setiap titik selalu ada sesuatu yang benar-benar bisa dicoba dan dilihat hasilnya.

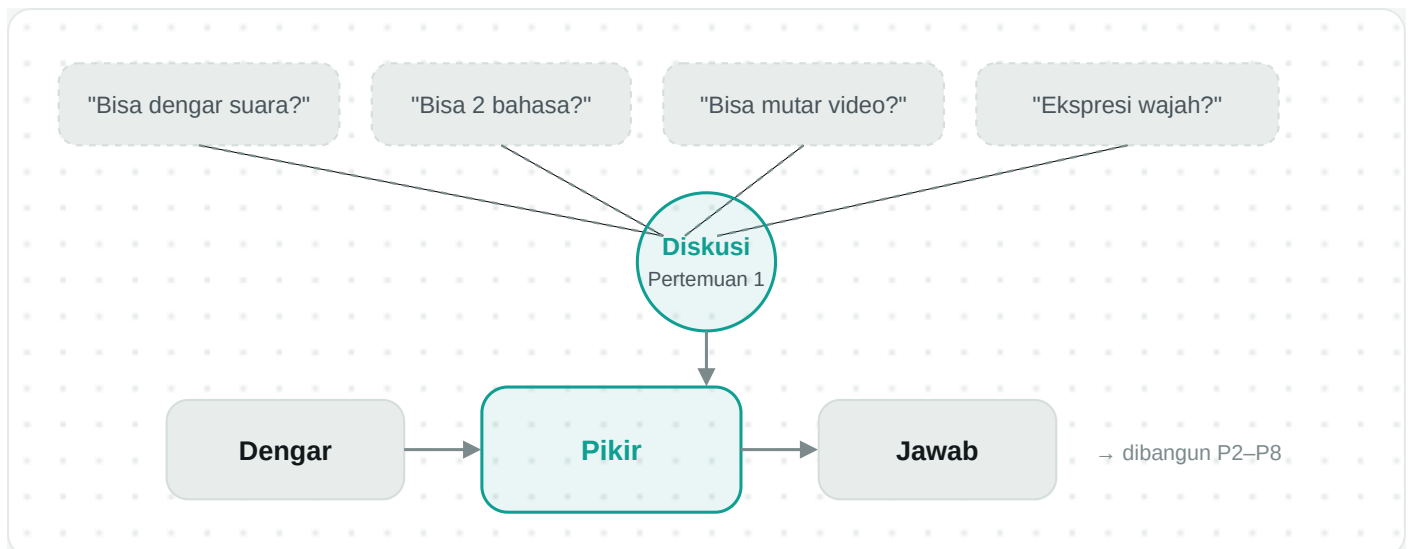
Jurang yang perlu diwaspadai

Proyek seperti ini kelihatan sederhana di atas kertas, tapi ada tiga hal yang sering menjatuhkan tim pemula di tengah jalan: internet yang tidak selalu stabil, sehingga robot yang bergantung penuh pada API online bisa "bisu" begitu saja saat wifi lab mati; empat orang menulis empat modul terpisah, dan bug paling berbahaya biasanya muncul justru di titik sambungan antar modul, bukan di dalam satu modul; dan kode yang berjalan mulus di laptop belum tentu berjalan sama di Raspberry Pi yang jadi otak fisik RoLi, karena perangkat kerasnya jauh lebih terbatas.

Materi di pertemuan-pertemuan berikutnya akan menunjukkan bagaimana Tim Software menghadapi ketiga jurang ini secara nyata, lengkap dengan bug asli yang mereka temukan, bukan cuma teori di atas kertas.

Kaitan dengan pelajaran sekolah

Membangun sistem lintas-modul seperti ini persis dengan elemen "Praktik Lintas Bidang" (PLB) pada mata pelajaran Informatika SMA Kurikulum Merdeka: proyek nyata yang menggabungkan beberapa elemen informatika sekaligus (jaringan, algoritma, kecerdasan buatan) dalam satu produk utuh, bukan dipelajari terpisah-pisah per bab.



Kaitan Kompetensi Lebih Luas

Brainstorming terstruktur dan menerjemahkan ide jadi peta kerja adalah keterampilan yang dipakai di hampir semua bidang, jauh sebelum baris kode pertama ditulis.

Alat & Bahan

papan tulis

diskusi kelompok

Aktivitas Hands-on

1. Diskusi kelompok: apa saja yang harus bisa dilakukan RoLi?
2. Mentor memaparkan gambaran arsitektur "dengar-pikir-jawab".

Hasil / Tolok Ukur

- ✓ Siswa bisa menjelaskan ulang alur "dengar-pikir-jawab" dengan kata-kata sendiri.

Kalau kamu jadi RoLi, hal pertama apa yang ingin kamu bisa lakukan?

02 Otak Robot: LLM via API Groq

Tujuan Pembelajaran

Siswa bisa memanggil LLM lewat API Groq dari kode Python dan mendapatkan balasan teks.

Istilah Penting

LLM (Large Language Model)

Model AI raksasa yang dilatih dari sangat banyak teks, sehingga bisa memprediksi & menyusun kalimat yang terdengar wajar, dasar dari "otak" RoLi.

API key

Semacam kata sandi khusus yang membuktikan ke server bahwa permintaan datang dari pengguna yang sah, dipakai untuk menghitung pemakaian & mengamankan akses.

Inference

Proses LLM benar-benar "berpikir" & menyusun jawaban dari sebuah pertanyaan. Semakin cepat proses ini, semakin cepat robot merespons.

API

Application Programming Interface, yaitu cara resmi bagi satu program untuk "meminta layanan" ke program atau server lain lewat internet, tanpa perlu tahu cara kerja di baliknya.

System prompt

Instruksi tersembunyi yang diberikan ke LLM di awal setiap percakapan, membentuk "kepribadian" & aturan main jawaban robot; pengguna tidak pernah melihatnya langsung.

Halusinasi (AI)

Ketika LLM menjawab dengan sangat yakin & meyakinkan, padahal isinya keliru atau mengada-ada, risiko yang selalu ada pada model bahasa, bukan cuma pada RoLi.

Konsep Kunci

LLM: model AI yang memprediksi & menyusun teks berdasarkan pola bahasa.

API key & request/response: cara program "berbicara" ke layanan AI lewat internet.

System prompt: instruksi tersembunyi yang membentuk "kepribadian" & aturan jawaban robot.

LLM (Large Language Model) paling gampang dibayangkan seperti fitur autocomplete di keyboard HP, tapi dilatih dari miliaran kalimat dalam buku, artikel, dan halaman web, bukan cuma dari kebiasaan mengetik satu orang. Karena polanya sangat luas, ia bisa "melanjutkan" hampir kalimat apa pun dengan cara yang terdengar masuk akal, termasuk membalas obrolan seolah-olah benar-benar memahami maksud kita.

Bagaimana LLM sebenarnya "berpikir"?

Penting untuk jujur soal ini sejak awal: LLM tidak benar-benar mengerti seperti manusia. Ia menyusun jawaban kata demi kata dengan menghitung kata apa yang paling mungkin muncul selanjutnya, berdasarkan pola statistik dari data latihannya. Karena itu ia bisa terdengar sangat yakin, padahal isinya keliru, kondisi yang biasa disebut "halusinasi". Membangun RoLi berarti belajar memakai alat yang sangat pintar sekaligus tidak sepenuhnya bisa dipercaya begitu saja.

Kenapa memilih Groq, bukan layanan LLM lain?

Untuk chatbot berbasis teks, jeda beberapa detik sebelum jawaban muncul biasanya tidak masalah. Tapi RoLi adalah robot yang diajak *bicara*: jeda yang sama akan terasa canggung, seperti robot yang "nge-lag" saat diajak ngobrol. Groq dipilih karena dikenal punya kecepatan pemrosesan (inference) yang sangat tinggi, sehingga jawaban bisa disiapkan lebih cepat, menjaga percakapan tetap terasa alami dan responsif.

System prompt asli RoLi

System prompt RoLi berisi aturan yang cukup spesifik: robot hanya boleh berbicara dalam Indonesia atau Prancis, harus menjawab dalam bahasa yang sama dengan lawan bicaranya kecuali diminta eksplisit memakai bahasa lain, dan setiap jawaban wajib diakhiri "tag" tersembunyi seperti `<TTS: ID>` atau `<TTS: FR>`; tag inilah yang nanti dipakai modul suara untuk memilih suara yang benar. Satu baris instruksi kecil seperti ini bisa menentukan apakah seluruh sistem berjalan mulus atau justru membingungkan modul-modul setelahnya.

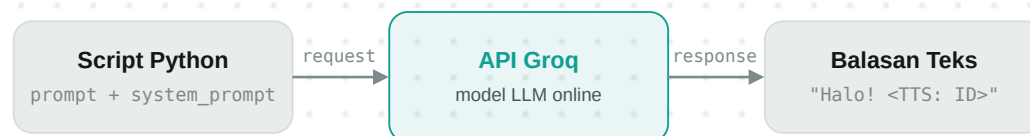
Jurang yang perlu diwaspadai

LLM bersifat tidak deterministik: kirim pertanyaan yang persis sama dua kali, jawabannya bisa sedikit berbeda setiap kali, berbeda jauh dari kode biasa yang untuk input sama selalu menghasilkan output yang sama. Ini bikin proses mencari bug terasa asing di awal, karena "kok kemarin jalan, sekarang jawabannya beda?" adalah hal yang wajar terjadi, bukan tanda ada yang rusak.

Di balik tiap panggilan API juga ada batas pemakaian (rate limit) dan biaya. Robot yang terus-menerus dipanggil tanpa jeda bisa kena limit atau menghabiskan kuota. Jadi kode yang efisien memanggil API bukan cuma soal kecepatan, tapi juga soal keberlangsungan.

Kaitan dengan pelajaran sekolah

Elemen "Algoritma dan Pemrograman" pada Informatika SMA Fase F secara eksplisit mencakup kemampuan mengimplementasikan *library* kecerdasan buatan dalam sebuah program. Pertemuan ini adalah versi nyatanya, bukan sekadar latihan di kelas, tapi library AI yang benar-benar dipakai robot yang akan berdiri di depan orang.



Kaitan Kompetensi Lebih Luas

Menulis system prompt yang jelas adalah bentuk komunikasi instruksi presisi, skill yang sama dipakai saat menyusun SOP atau instruksi kerja.

Alat & Bahan

Python

API key Groq

library groq

Aktivitas Hands-on

1. Setup API key Groq.
2. Kirim prompt sederhana lewat script, amati balasan.
3. Ubah system prompt, amati bagaimana gaya jawaban berubah.

Hasil / Tolok Ukur

- ✔ Script berhasil mengirim prompt & menerima balasan, dengan system prompt dasar RoLi (ramah, singkat, dwibahasa ID/FR), fondasi awal `brain.py`.

Instruksi apa yang paling menentukan supaya jawaban robot terasa seperti "RoLi", bukan robot generik?

03 Fitur Suara: STT & TTS

Istilah Penting

STT (Speech-to-Text)

Teknologi yang mengubah suara ucapan jadi teks tertulis. Inilah "telinga" RoLi, langkah pertama sebelum otaknya bisa memahami apa yang diucapkan.

Whisper

Model AI pengenalan suara buatan OpenAI, dilatih dari ratusan ribu jam rekaman berbagai bahasa, dipakai RoLi lewat dua jalur berbeda: online (Groq) dan offline (faster-whisper).

TTS (Text-to-Speech)

Teknologi kebalikan dari STT: mengubah teks jadi suara yang terdengar seperti orang berbicara. Inilah "mulut" RoLi.

Sampling (audio)

Proses mengubah gelombang suara analog jadi deretan angka digital, ribuan kali per detik, supaya bisa diproses komputer.

Blok A: Tujuan (STT)

Robot bisa mengubah ucapan jadi teks.

Blok A: Konsep (STT)

Speech-to-Text: mengubah gelombang suara jadi teks yang bisa diproses.

Groq Whisper vs faster-whisper: versi online (cepat, akurat) vs versi offline (jalan di perangkat sendiri).

Blok A: Aktivitas

1. Rekam suara dari mikrofon.
2. Jalankan lewat `stt.py`, amati hasil transkripsi.
3. Bandingkan hasil transkripsi Indonesia vs Prancis.

Blok B: Tujuan (TTS)

Robot bisa mengubah teks jadi suara.

Blok B: Konsep (TTS)

Text-to-Speech: mengubah teks jadi suara yang diucapkan.

Edge-TTS vs Piper: suara online (Microsoft Edge-TTS) vs model suara lokal offline (Piper).

Blok B: Aktivitas

1. Kirim teks ke `tts.py`, dengarkan hasilnya.
2. Coba ganti bahasa ID/FR dan amati perbedaan suara.

Penjelasan Lengkap

Setelah otak RoLi bisa berpikir lewat teks, saatnya memberinya telinga dan mulut. Speech-to-Text (STT) mengubah gelombang suara jadi teks yang bisa dipahami otak; Text-to-Speech (TTS) mengubah balasan otak jadi suara yang keluar dari speaker: dua arah yang berlawanan, tapi sama-sama harus akurat dan cepat supaya percakapan terasa alami.

Bagian STT: bagaimana komputer "mendengar"

Mikrofon merekam suara sebagai gelombang analog, lalu mengubahnya jadi deretan angka digital lewat proses yang disebut sampling. Angka-angka itulah yang diproses model STT untuk ditebak jadi teks. RoLi memakai dua model untuk tugas yang sama: versi online lewat Groq memakai Whisper (model pengenalan suara buatan OpenAI yang dilatih dari ratusan ribu jam rekaman suara berbagai bahasa), sementara versi offline memakai faster-whisper, yaitu model Whisper yang sama tapi dijalankan langsung di perangkat dan dioptimalkan supaya cukup ringan untuk komputer kecil seperti Raspberry Pi. Keduanya "sepupu dekat", tapi versi cloud jauh lebih cepat dan akurat karena berjalan di server yang jauh lebih kuat daripada Raspberry Pi.

Bagian TTS: bagaimana komputer "berbicara"

TTS modern bekerja kurang-lebih terbalik dari STT: modelnya dilatih dari rekaman suara manusia asli sampai bisa memprediksi bentuk gelombang suara yang sesuai untuk teks apa pun yang diberikan. RoLi memakai Edge-TTS (layanan online dari Microsoft, suaranya sangat natural) sebagai pilihan utama, dan Piper (model suara lokal yang jauh lebih ringan) sebagai cadangan saat offline. Karena RoLi harus bicara dua bahasa, sistem ini

juga harus memilih suara yang tepat: suara Indonesia untuk teks Indonesia, suara Prancis untuk teks Prancis, persis berdasarkan tag `<TTS: ID/FR>` yang disiapkan otak robot di Pertemuan 2.

Jurang yang perlu diwaspadai

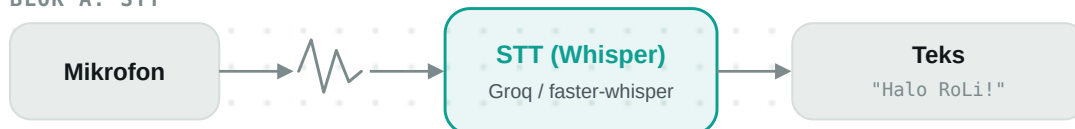
Akurasi STT gampang terganggu oleh suara latar (kipas angin, obrolan orang lain, dengung AC lab) dan aksen bicara yang berbeda-beda. Robot yang "salah dengar" adalah hal biasa yang harus diterima, bukan selalu tanda kode yang rusak. Menentukan kapan seseorang "selesai bicara" juga jauh lebih rumit dari kelihatannya: berhenti mendengarkan terlalu cepat memotong kalimat orang, terlalu lambat membuat robot terasa lelet merespons.

Tim Software juga pernah menemukan bug nyata di bagian audio: perangkat suara yang tersambung lewat HDMI ke layar kadang "hilang" tiba-tiba di tengah pemakaian, kemungkinan karena sambungan sempat putus sesaat. Solusinya, kode pemutar suara ditulis untuk mendeteksi kegagalan semacam itu dan otomatis menyambung ulang perangkat audio sebelum mencoba bicara sekali lagi. Ini contoh nyata kenapa kode yang mengendalikan perangkat fisik harus selalu siap menghadapi kejadian tak terduga, bukan cuma skenario ideal di atas kertas.

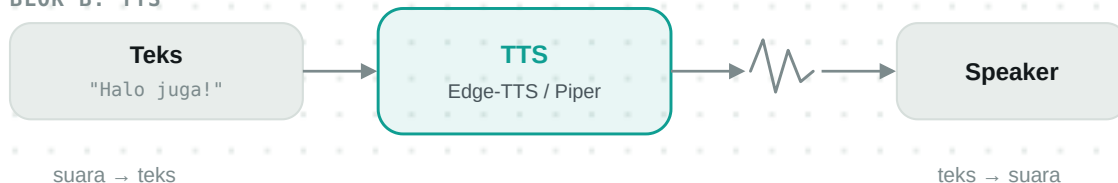
Fakta menarik

Whisper, model yang "mendengarkan" RoLi baik online maupun offline, adalah model open-source yang sama dipakai jutaan aplikasi lain di dunia untuk transkripsi suara. RoLi memanfaatkannya lewat dua jalur berbeda untuk dua kebutuhan berbeda: kecepatan maksimal saat online lewat server Groq, ketahanan saat offline lewat perangkat sendiri.

BLOK A: STT



BLOK B: TTS



Kaitan Kompetensi Lebih Luas

Merancang jalur cadangan (fallback) saat layanan online gagal adalah prinsip keandalan sistem (reliability engineering) yang dipakai di banyak sistem produksi, bukan cuma robot suara.

Alat & Bahan

mikrofon

stt.py

tts.py

piper_voices/

Milestone Penutup

STT (blok A) + Otak (P2) + TTS (blok B) disambungkan jadi loop dengar-pikir-jawab utuh untuk pertama kalinya. Robot benar-benar bisa "diajak ngobrol".

Hasil / Tolok Ukur

- ✓ Robot merespons suara secara lisan, end-to-end, dalam Indonesia maupun Prancis.

Kalau internet mati di tengah percakapan, apa yang seharusnya terjadi supaya robot tidak "diam saja"?

04 Animasi & UI: Wajah RoLi

Tujuan Pembelajaran

Robot punya wajah yang bereaksi sesuai status: mendengar, berpikir, atau bicara.

Istilah Penting

State machine

Cara berpikir dalam "kondisi-kondisi" tetap dan jelas (seperti lampu lalu lintas: merah/kuning/hijau); sistem selalu tahu persis kondisi mana yang sedang aktif.

Lip-sync

Menyelaraskan gerakan "mulut" karakter dengan suara yang sedang diucapkan, supaya terlihat seperti benar-benar sedang bicara.

Pygame

Library Python untuk menggambar & menganimasikan bentuk di layar secara langsung, dipilih karena ringan dan cepat menggambar ulang layar berkali-kali per detik.

Asynchronous (async)

Gaya menulis kode yang memungkinkan program mengerjakan hal lain (seperti tetap menganimasikan wajah) sambil menunggu proses lain selesai (seperti menunggu jawaban dari internet).

Konsep Kunci

State machine: mesin status sederhana: listening / thinking / speaking sebagai tiga kondisi berbeda.

Pemrograman grafis dengan pygame: menggambar & menganimasikan bentuk di layar secara langsung.

Lip-sync sederhana: gerakan "mulut" yang mengikuti level volume suara saat robot bicara.

Penjelasan Lengkap

Sejauh ini RoLi sudah bisa berpikir, mendengar, dan bicara, tapi dari luar, ia masih terlihat seperti kotak diam yang tiba-tiba mengeluarkan suara. Pertemuan

ini memberinya wajah, supaya orang yang berinteraksi bisa *melihat* robot sedang mendengarkan, berpikir, atau bicara, bukan cuma mendengarnya.

Apa itu state machine, sebenarnya?

Bayangkan lampu lalu lintas: ia hanya punya beberapa kondisi tetap (merah, kuning, hijau), dan selalu jelas kondisi mana yang sedang aktif serta kapan berpindah ke kondisi berikutnya. RoLi memakai ide yang sama untuk wajahnya: sistem selalu tahu persis apakah dirinya sedang berada di kondisi *listening*, *thinking*, atau *speaking*, dan kode menggambar tampilan yang berbeda tergantung kondisi mana yang aktif saat itu. Pendekatan berpikir dalam "kondisi yang jelas dan terbatas" ini disebut state machine, dan dipakai di hampir semua software interaktif.

Kenapa pygame, bukan framework yang lebih besar?

Wajah RoLi harus digambar ulang puluhan kali per detik supaya animasinya terlihat halus, bukan patah-patah, sekitar 30 kali per detik. Banyak framework antarmuka modern dirancang untuk aplikasi yang jarang berubah tampilannya (seperti formulir atau halaman pengaturan), bukan untuk animasi yang terus bergerak. Pygame dipilih karena ia memang dibuat untuk menggambar ulang layar berulang-ulang dengan cepat dan ringan, cocok untuk perangkat terbatas seperti Raspberry Pi yang menjadi otak fisik RoLi.

Lip-sync yang "jujur" disederhanakan

Lip-sync yang sesungguhnya (mulut bergerak sesuai bunyi huruf yang sedang diucapkan) butuh analisis suara yang rumit dan berat secara komputasi. RoLi memakai pendekatan yang jauh lebih sederhana: bukaan "mulut" mengikuti seberapa keras suara pada saat itu (level volume), bukan bunyi huruf tertentu. Hasilnya bukan lip-sync sempurna, tapi cukup meyakinkan mata manusia bahwa robot sedang bicara. Ini sebuah contoh nyata keputusan rekayasa: memilih solusi "cukup baik dan ringan" ketimbang "sempurna tapi berat", karena keterbatasan perangkat keras yang dipakai.

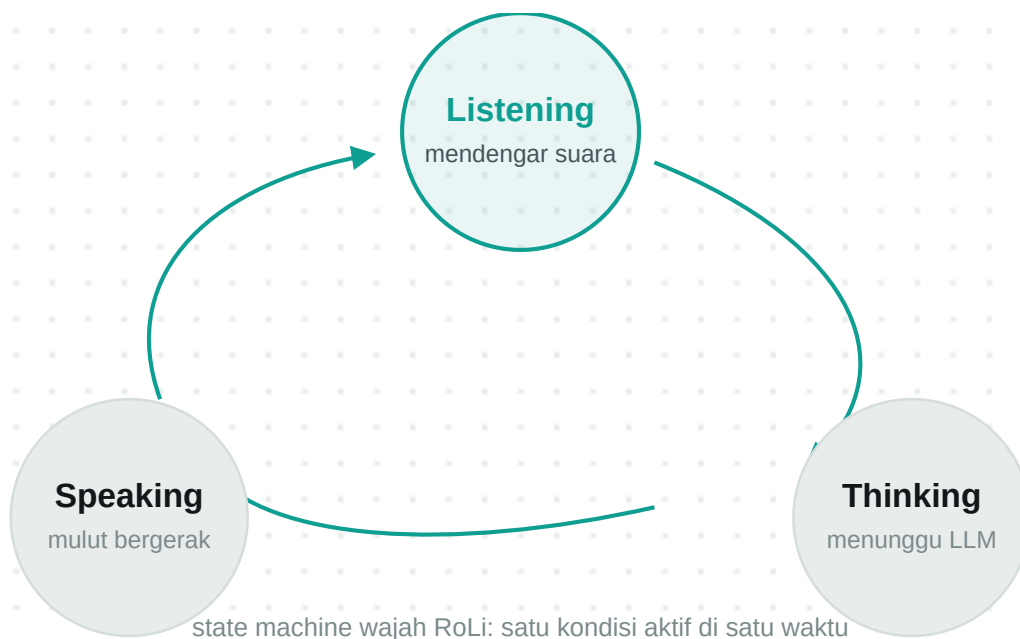
Jurang yang perlu diwaspadai

Menjaga animasi tetap mulus sambil robot juga sedang menunggu jawaban dari API di internet adalah tantangan klasik dalam sistem real-time: kalau proses utama berhenti sejenak menunggu jawaban jaringan, wajah robot akan ikut "membeku", dan ilusi bahwa ia hidup langsung runtuh di depan mata. Inilah alasan sebagian besar kode RoLi ditulis dengan gaya *asynchronous* (async), supaya animasi tetap berjalan sambil menunggu proses lain di latar belakang.

Fakta lain yang layak diperhatikan: `ui.py` adalah file terbesar di seluruh proyek RoLi (sekitar 47 kilobyte kode). Satu file yang terus membesar seiring bertambahnya fitur adalah tanda yang wajar sekaligus perlu diwaspadai dalam proyek nyata: semakin besar satu file, semakin sulit menemukan bagian tertentu dan semakin mudah satu perubahan kecil menimbulkan efek yang tidak terduga di bagian lain.

Fakta menarik

Riset interaksi manusia-komputer berulang kali menunjukkan bahwa manusia menilai seberapa "hidup" atau "dapat dipercaya" sebuah asisten robot lebih dari seberapa cepat & ekspresif ia merespons secara visual, bukan semata dari seberapa pintar jawabannya. Wajah yang bereaksi cepat bisa membuat RoLi terasa lebih "hidup" walau logika di baliknya sebenarnya sederhana.



Kaitan Kompetensi Lebih Luas

Berpikir dalam "state" (kondisi) adalah cara berpikir yang dipakai di hampir semua software interaktif, dari lampu lalu lintas sampai mesin ATM, bukan cuma wajah robot.

Alat & Bahan

ui.py

pygame

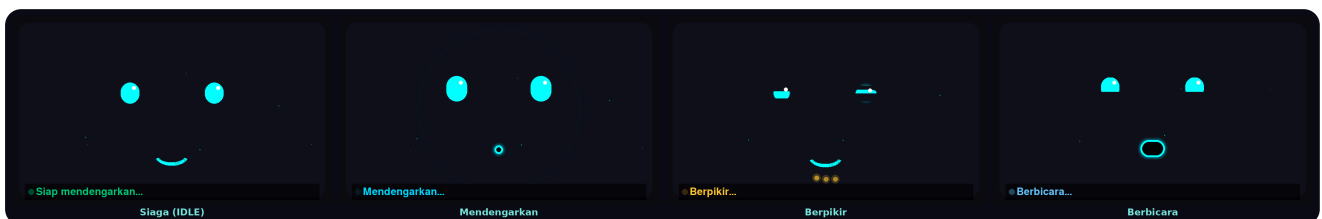
Aktivitas Hands-on

1. Ubah & uji tampilan tiap state wajah.
2. Sumbungkan animasi ke level volume output TTS agar "mulut" ikut bergerak saat bicara.

Hasil / Tolok Ukur

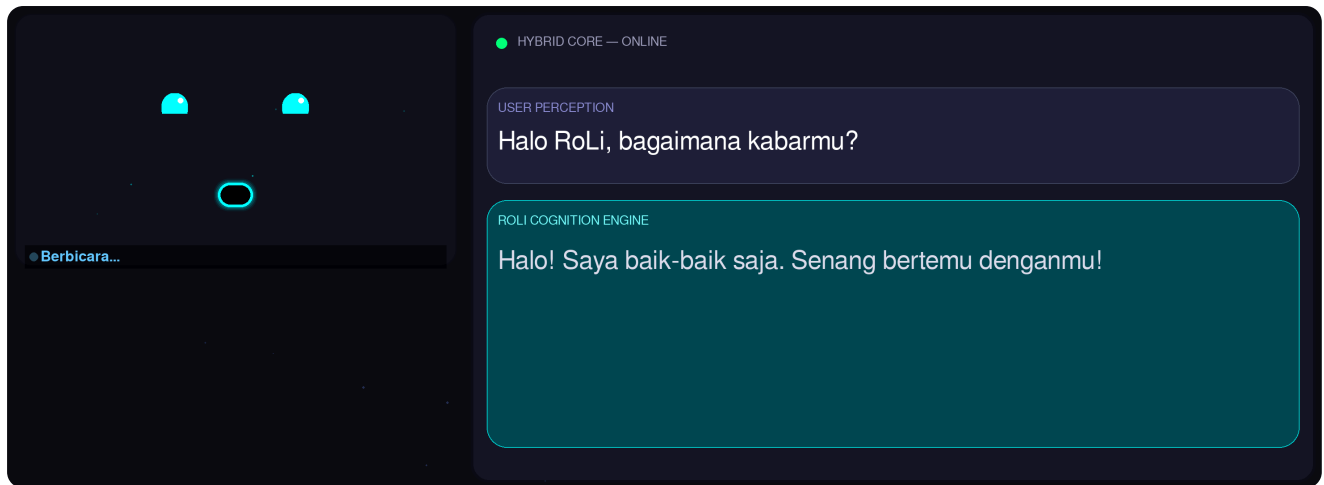
- ✓ Wajah RoLi berubah otomatis mengikuti apa yang sedang dilakukan robot, tanpa disentuh manual.

Pencapaian: Empat Ekspresi RoLi



Screenshot ui.py sungguhan saat dijalankan, empat status wajah yang dibangun Tim Software: siaga, mendengarkan, berpikir, dan berbicara.

Pencapaian: Layar Penuh Saat Bicara



Tampilan gabungan layar wajah dan layar transkrip percakapan, menunjukkan RoLi merespons pertanyaan secara real-time.

Ekspresi apa yang bikin robot terasa "hidup", bukan cuma layar yang menyala?

05 Pemutar Video & Function Calling

Tujuan Pembelajaran

Robot memutar video cerita/lagu atas permintaan suara, dan videonya dipilih otomatis oleh AI.

Istilah Penting

Function calling

Kemampuan LLM untuk tidak cuma menjawab teks, tapi memutuskan & "meminta" program menjalankan fungsi tertentu (misalnya memutar video) berdasarkan makna kalimat penggunaannya.

VLC

Pemutar media open-source yang sudah teruji di banyak jenis perangkat, termasuk Raspberry Pi, dipakai RoLi untuk benar-benar menampilkan video.

Manifest / katalog

Daftar berisi semua video yang tersedia lengkap dengan judul & deskripsi, disimpan di `manifest.json`, sumber informasi yang dipakai AI untuk memilih video.

Konsep Kunci

Function calling: AI tidak cuma menjawab teks, tapi bisa "memanggil fungsi" di dalam program.

Katalog `manifest.json`: daftar video beserta deskripsi, sumber AI memutuskan video mana yang cocok.

Penjelasan Lengkap

Sampai pertemuan sebelumnya, otak RoLi hanya bisa melakukan satu hal: menjawab dengan kata-kata. Pertemuan ini adalah lompatan penting: otak RoLi

belajar *bertindak*, bukan cuma menjawab, dengan cara memutuskan sendiri kapan harus memutar video dan video mana yang paling cocok.

Dari sekadar menjawab, menjadi bertindak

Cara lama yang sering dipakai program sederhana adalah mencocokkan kata kunci di kalimat pengguna, misalnya, kalau kalimat mengandung kata "video", jalankan pemutar video. Cara ini rapuh: orang punya banyak sekali cara berbeda untuk meminta hal yang sama ("puterin dong", "aku mau nonton", "ceritain yang kemarin"). Function calling mengubah pendekatannya: LLM diberi "menu" fungsi yang benar-benar bisa dijalankan program, lengkap dengan nama dan deskripsi tiap fungsi. LLM sendiri yang memutuskan apakah perlu memanggil salah satu fungsi itu berdasarkan makna kalimat penggunaanya, bukan sekadar mencocokkan kata per kata, dan mengembalikan permintaan pemanggilan fungsi yang terstruktur, bukan teks bebas yang harus ditebak-tebak programnya sendiri.

Kenapa deskripsi di manifest.json menentukan segalanya

Keputusan AI sepenuhnya bergantung pada seberapa jelas deskripsi tiap video ditulis di katalog. Panduan resmi Tim Software untuk menambah video bahkan menyebutkan eksplisit: video yang lupa didaftarkan dengan deskripsi lengkap tetap akan muncul otomatis di katalog, tapi hanya dengan judul file sebagai deskripsi seadanya, sehingga AI punya jauh lebih sedikit informasi untuk memutuskan kapan video itu relevan diputar. Ini adalah pelajaran "sampah masuk, sampah keluar" (garbage in, garbage out) versi nyata: sistem AI hanya sebaik data yang diberikan kepadanya.

VLC dan jendela layar penuh

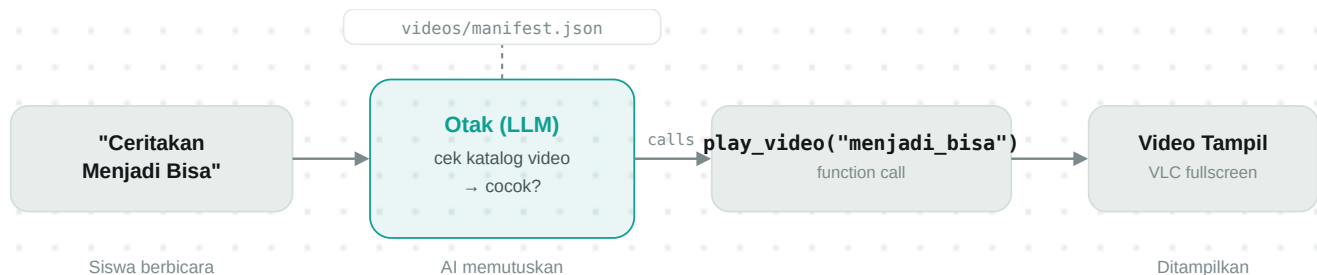
Saat video akhirnya diputar, RoLi memakai VLC, pemutar media open-source yang sudah teruji di banyak perangkat, termasuk Raspberry Pi, dan mengatur jendelanya secara langsung lewat perintah sistem X11 (Xlib) supaya video benar-benar tampil layar penuh tanpa bingkai jendela yang mengganggu, persis seperti kios/etalase digital sungguhan.

Jurang yang perlu diwaspadai

Permintaan suara yang ambigu tetap bisa membuat AI memilih video yang kurang tepat. Ini risiko yang tidak pernah hilang sepenuhnya, hanya bisa diperkecil dengan deskripsi yang lebih jelas dan pengujian kalimat yang beragam. Katalog video juga bisa "drift", perlahan tidak sinkron dari file yang sebenarnya ada di folder, kalau video baru ditambahkan tapi lupa didaftarkan, atau file dihapus tapi entrinya masih ada. Menjaga data (file video) dan metadata (manifest.json) tetap sinkron adalah disiplin berkelanjutan, bukan tugas sekali jadi yang selesai di satu pertemuan.

Fakta menarik

Contoh permintaan suara yang dipakai untuk menguji fitur ini di dokumentasi resmi Tim Software adalah "Ceritakan cerita Menjadi Bisa dong", merujuk langsung ke salah satu video hasil adaptasi Tim Konten. Function calling di pertemuan ini adalah titik pertama di mana hasil kerja dua divisi berbeda benar-benar bertemu dalam satu fitur yang berfungsi.



Kaitan Kompetensi Lebih Luas

Memberi AI "daftar pilihan yang jelas" (katalog dengan deskripsi baik) supaya keputusannya akurat adalah prinsip desain sistem AI yang berlaku luas, bukan cuma untuk video.

Alat & Bahan

video_player.py video_controls.py
manifest.json brain.py (tools)

Aktivitas Hands-on

1. Daftarkan video baru ke `manifest.json` (hasil dari Tim Konten).
2. Uji beberapa kalimat suara berbeda, amati kapan AI memilih video yang tepat.
3. Coba tombol kontrol: pause, maju/mundur, volume.

Hasil / Tolok Ukur

- ✓ AI berhasil memilih video yang sesuai dari kalimat suara natural, bukan dari kata kunci yang kaku.

Kenapa deskripsi video yang jelas lebih penting bagi AI daripada nama file yang rapi?

06 Setup Pembuka: Wifi, Mode, Tampilan Awal

Tujuan Pembelajaran

Robot punya alur onboarding yang jelas saat pertama kali dinyalakan.

Istilah Penting

SSID

Nama sebuah jaringan wifi yang terlihat saat memindai sinyal di sekitar, misalnya "Lab-Robotika" atau nama wifi rumah.

Alamat IP

Semacam "alamat rumah" digital yang diberikan ke sebuah perangkat begitu berhasil tersambung ke jaringan, supaya perangkat lain bisa mengirim data kepadanya.

Alur wizard

Rangkaian langkah yang dituntun satu per satu (bukan satu layar penuh pilihan sekaligus), cara umum menuntun pengguna baru yang belum familier dengan sistemnya.

nmcli

Network Manager Command Line Interface, yaitu alat bawaan Linux untuk mengatur koneksi jaringan lewat perintah teks, "dibungkus" RoLi jadi tombol-tombol layar sentuh.

Locale

Pengaturan bahasa & format regional pada sistem operasi, bisa diam-diam mengubah keluaran teks sebuah perintah, seperti "yes" menjadi "ya".

Konsep Kunci

SSID & koneksi wifi: menyambungkan perangkat ke jaringan lewat command-line `nmcli` di Linux.

On-screen keyboard: keyboard di layar sentuh untuk mengetik password tanpa keyboard fisik.

Alur wizard: langkah demi langkah yang menuntun pengguna baru, bukan satu layar yang membingungkan.

Penjelasan Lengkap

Semua fitur RoLi yang sudah dibangun sejauh ini (otak, telinga, mulut, wajah) kurang berguna kalau robotnya bahkan belum tahu cara tersambung ke internet saat pertama kali dinyalakan di tempat baru. Pertemuan ini membangun "gerbang masuk" RoLi: layar pertama yang dilihat siapa pun yang menyalakannya.

Apa yang sebenarnya terjadi saat "menyambung ke wifi"?

Di balik satu ketukan layar sentuh, ada beberapa langkah teknis: perangkat memindai (scan) sinyal-sinyal wifi di sekitarnya dan menampilkan nama jaringan (SSID) yang tertangkap; setelah nama jaringan dan kata sandi dimasukkan, perangkat mengirim keduanya untuk diverifikasi oleh router; kalau berhasil, perangkat baru diberi alamat identitas di jaringan itu (alamat IP) supaya bisa saling berkiriman data. RoLi tidak menciptakan ulang seluruh proses ini dari nol. Ia membungkus alat bawaan sistem operasi Linux bernama `nmcli` (Network Manager Command Line Interface) dengan tampilan layar sentuh yang ramah bagi siapa pun, bahkan yang tidak paham command-line sama sekali.

`nmcli`: jembatan ke sistem operasi

`wifi_manager.py` pada dasarnya menjalankan perintah `nmcli` dari dalam program Python, lalu membaca hasil teksnya untuk ditampilkan sebagai tombol-tombol yang bisa disentuh. Ini contoh nyata bagaimana software modern jarang membangun semuanya dari nol. Sering kali tugasnya adalah menyambungkan alat-alat yang sudah ada (dalam hal ini, alat bawaan Linux) dengan cara yang lebih mudah dipakai orang biasa.

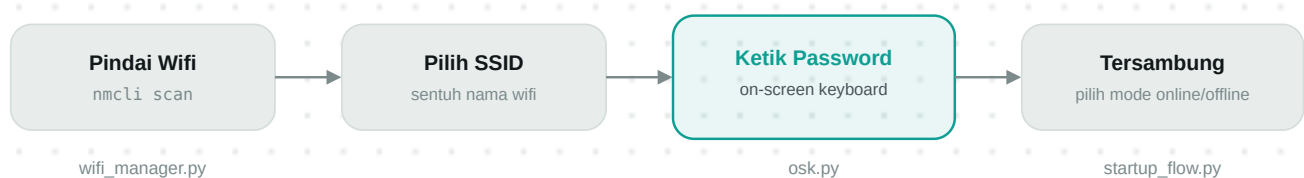
Jurang yang perlu diwaspadai

Tim Software menemukan bug yang halus tapi nyata: keluaran teks dari `nmcli` ternyata berubah bahasa mengikuti pengaturan bahasa sistem operasi (misalnya kata "ya"/"tidak" muncul menggantikan "yes"/"no" di sistem berbahasa Indonesia), dan ini diam-diam merusak kode yang membaca hasil tersebut, karena kode tadinya menganggap keluarannya akan selalu berbahasa Inggris. Solusinya adalah memaksa perintah selalu berjalan dalam pengaturan bahasa netral (disebut locale `C`), apa pun bahasa sistem operasinya. Ini contoh sempurna kenapa kode yang berjalan mulus di laptop seorang programmer bisa gagal secara diam-diam di perangkat lain dengan pengaturan berbeda.

Ada juga urusan izin akses: mengubah pengaturan jaringan di Linux butuh hak akses khusus, dan kode ini ditemukan hanya bekerja andal ketika aplikasi berjalan sebagai sesi pengguna yang sedang aktif login di layar itu sendiri, detail teknis yang tidak terlihat gejalanya sampai tiba-tiba gagal tanpa pesan error yang jelas. Fitur yang terasa sederhana bagi pengguna (ketuk nama wifi, ketik password, ketuk sambung) sering kali justru menyimpan kerumitan tersembunyi paling banyak di baliknya. Ini persis alasan kenapa fitur ini pantas mendapat satu pertemuan penuh.

Kaitan dengan pelajaran sekolah

Elemen "Jaringan Komputer dan Internet" pada Informatika SMA Fase F membahas topologi jaringan dan lapisan-lapisan sistem jaringan komputer (OSI Layer) secara konsep. Pertemuan ini adalah versi hands-on-nya: bukan menghafal diagram lapisan jaringan, tapi benar-benar membuat dua perangkat saling menemukan dan mempercayai satu sama lain di dunia nyata.



Kaitan Kompetensi Lebih Luas

Merancang alur onboarding yang jelas adalah bagian dari UX (pengalaman pengguna), dipakai di hampir semua perangkat konsumen, dari smart TV sampai ponsel baru.

Alat & Bahan

wifi_manager.py

osk.py

startup_flow.py

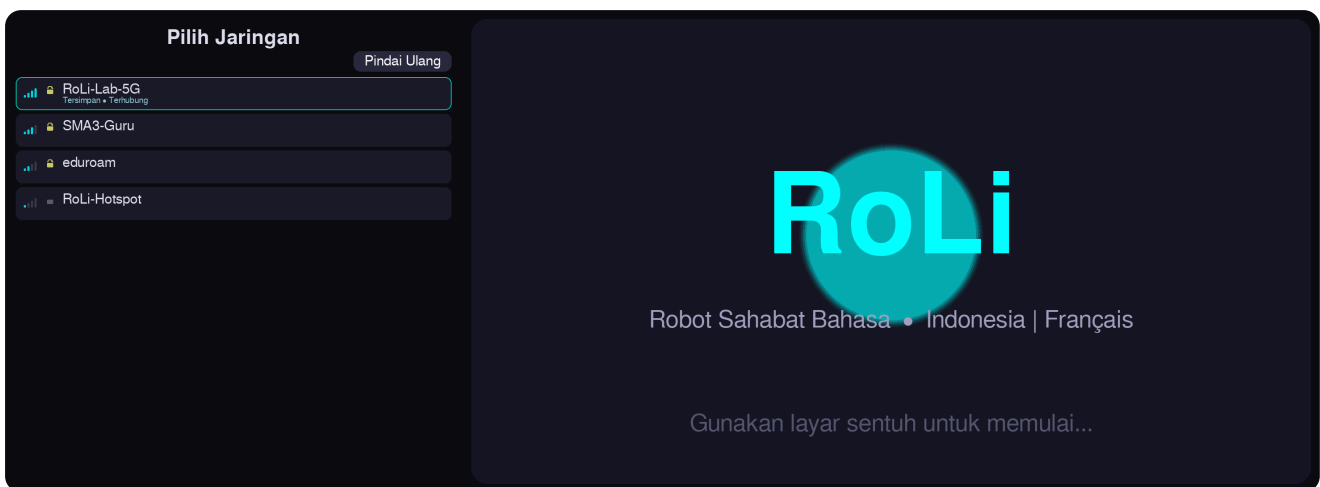
Aktivitas Hands-on

1. Uji alur scan → pilih → sambung wifi lewat touchscreen.
2. Coba layar pemilihan mode online/offline.

Hasil / Tolok Ukur

- ✓ Robot bisa disambungkan ke wifi baru langsung dari layar sentuh, tanpa bantuan command-line.

Pencapaian: Layar Pilih Jaringan



Screenshot WifiListPanel sungguhan dari startup_flow.py, daftar jaringan dicontohkan untuk demonstrasi tampilan.

Kalau kamu orang yang baru pertama kali pakai RoLi, langkah apa yang harus paling jelas terlihat di layar?

07 Versi Offline

Tujuan Pembelajaran

Robot tetap bisa dengar-mikir-jawab walau tanpa internet.

Istilah Penting

Ollama

Alat untuk menjalankan model AI secara lokal langsung di sebuah perangkat, tanpa perlu mengirim data ke server internet.

Fallback

Rencana cadangan yang otomatis diaktifkan sistem saat pilihan utama gagal, misalnya beralih ke mode offline saat internet mati.

Edge AI / on-device AI

Istilah umum untuk menjalankan kecerdasan buatan langsung di perangkat kecil (seperti Raspberry Pi), bukan di server raksasa, bidang yang sedang berkembang pesat.

Graceful degradation

Prinsip desain: sistem yang baik tidak berhenti total saat satu bagian gagal, tapi "mundur" ke versi lebih sederhana sambil tetap berfungsi.

Konsep Kunci

AI lokal (Ollama) vs cloud: trade-off kecepatan & kualitas vs ketersediaan tanpa internet.

Keterbatasan function calling offline: model lokal kecil ternyata tidak bisa diandalkan untuk tool-call terstruktur, jadi dipakai tag teks `<VIDEO: id>` sebagai gantinya.

Fallback otomatis: sistem beralih sendiri dari online ke offline saat koneksi gagal.

Penjelasan Lengkap

Semua fitur sejauh ini bergantung pada layanan online yang berjalan di server yang sangat kuat. Tapi RoLi dirancang untuk dipakai di sekolah, yang koneksi internetnya tidak selalu bisa diandalkan. Pertemuan ini menjawab pertanyaan

yang jarang dipikirkan pemula sejak awal: apa yang terjadi kalau internet tiba-tiba mati di tengah RoLi sedang dipakai?

Kenapa AI lokal itu sulit?

Model AI raksasa seperti yang dipakai Groq berjalan di server khusus yang jauh lebih kuat dari komputer rumahan mana pun. Menjalankan "otak" serupa langsung di perangkat kecil seperti Raspberry Pi berarti memakai model yang jauh lebih kecil dan sederhana lewat Ollama (alat untuk menjalankan model AI secara lokal). Model yang lebih kecil punya "kapasitas" yang jauh lebih terbatas untuk mengikuti instruksi rumit secara konsisten, mirip perbedaan antara meminta bantuan pada seseorang yang sangat berpengalaman dibanding seseorang yang baru belajar: keduanya bisa membantu, tapi kualitas dan keandalannya jauh berbeda.

Ketika function calling "resmi" ternyata tidak bisa diandalkan

Tim Software awalnya mencoba memakai function calling yang sama persis seperti di Pertemuan 5 untuk model offline. Hasil pengujian nyata menunjukkan ini bermasalah: beberapa model lokal langsung menolak permintaan begitu diberi daftar fungsi (parameter `tools`), sementara model lain malah menghabiskan seluruh jatah proses hanya untuk "berpikir keras" apakah perlu memanggil fungsi atau tidak, bukannya benar-benar menjawab. Solusinya justru menggunakan kembali pola yang sudah terbukti dari Pertemuan 2: bukan format function-calling yang kaku, tapi tag teks sederhana seperti `<VIDEO: id>`, meniru pola `<TTS: XX>` yang sudah lebih dulu terbukti andal. Model kecil ternyata jauh lebih bisa diandalkan untuk sekadar menulis pola teks di jawabannya, dibanding mengikuti format terstruktur yang ketat.

Fallback sebagai filosofi desain, bukan sekadar fitur tambahan

Prinsip di balik seluruh pertemuan ini disebut *graceful degradation*: sistem yang baik tidak berhenti total saat satu bagian gagal, tapi mundur ke versi yang lebih sederhana sambil tetap berfungsi. Lebih baik robot yang menjawab singkat dan terbatas daripada robot yang tiba-tiba bisu sama sekali.

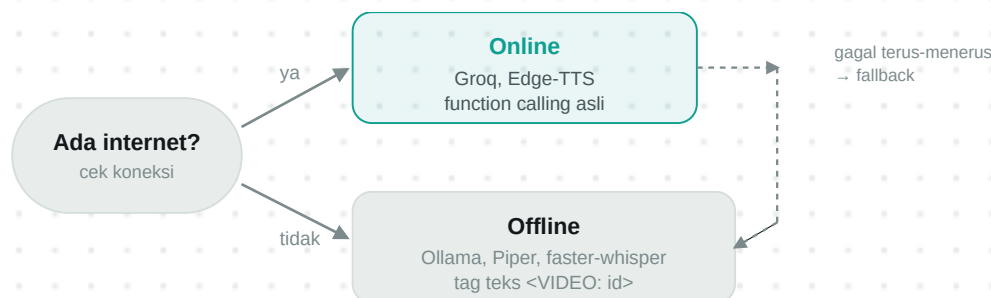
Jurang yang perlu diwaspadai

Model AI lokal berperilaku jauh lebih "tidak terduga" dibanding kode biasa. Solusi yang berhasil di satu model kecil bisa gagal total di model kecil lain, walau sama-sama dijalankan lewat Ollama. Ini pelajaran penting: sistem berbasis AI tidak selalu bisa digeneralisasi seperti kode konvensional, sehingga pengujian langsung jauh lebih penting daripada menebak dari dokumentasi saja.

Model lokal juga butuh waktu "pemanasan" (warm-up) untuk dimuat penuh ke memori sebelum bisa menjawab, jadi interaksi offline pertama setelah robot berpindah mode biasanya terasa lebih lambat dari interaksi-interaksi setelahnya. Solusi pragmatis yang dipakai tim: begitu satu model diketahui gagal memakai function calling, sistem "mengingat" hal itu dan berhenti mencoba format itu lagi untuk model yang sama sepanjang sesi. Ini bukan solusi sempurna, tapi cukup untuk menjaga percakapan tetap berjalan.

Fakta menarik

Menjalankan model kecerdasan buatan langsung di perangkat kecil seperti Raspberry Pi (bukan di server raksasa) adalah bidang yang sedang berkembang pesat secara global, sering disebut "edge AI" atau "on-device AI". Beberapa tahun lalu, ide menjalankan AI percakapan di komputer sekecil ini nyaris mustahil dilakukan.



Kaitan Kompetensi Lebih Luas

Memutuskan kapan harus menyederhanakan solusi karena keterbatasan sumber daya adalah keputusan rekayasa nyata (engineering trade-off) yang dipakai di banyak sistem embedded.

Alat & Bahan

brain.py (Ollama)

stt.py / tts.py (fallback)

startup_flow.py

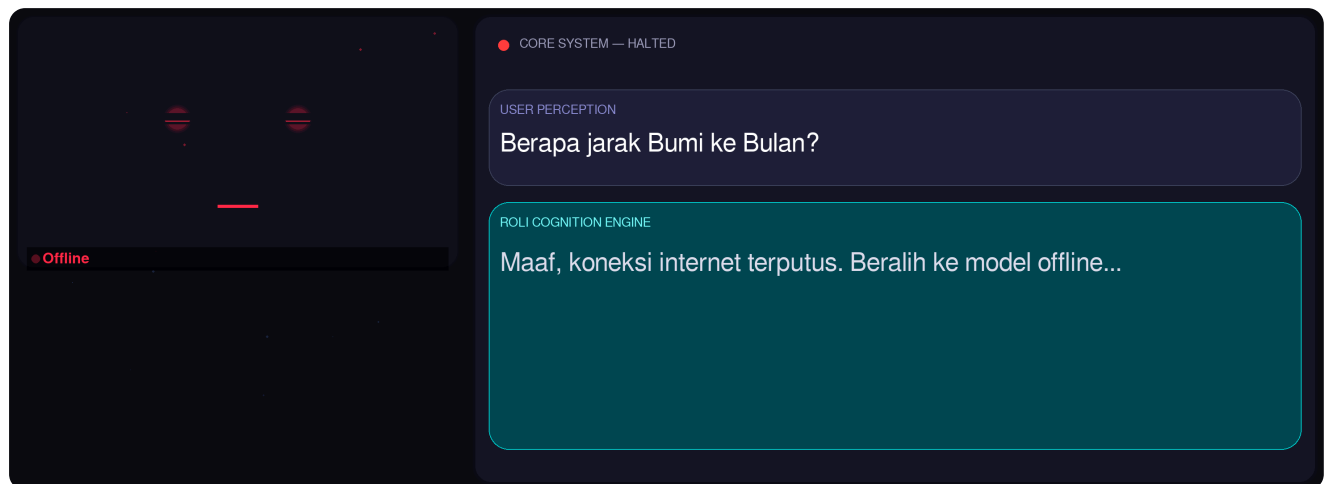
Aktivitas Hands-on

1. Matikan internet, uji robot tetap menjawab & memutar video.
2. Bandingkan kualitas & kecepatan jawaban online vs offline.

Hasil / Tolok Ukur

- ✓ Robot tetap berfungsi (dengan jawaban lebih singkat) walau benar-benar tanpa internet.

Pencapaian: Mode Offline Aktif



Screenshot ui.py saat `set_offline(True)` aktif, wajah dan status berubah merah menandakan robot beralih ke model lokal.

Kompromi apa yang rela kamu terima supaya robot tetap "hidup" walau internet mati?

Sinkronisasi,

08 Finishing, Bug Hunting

Tujuan Pembelajaran

Semua fitur dari Pertemuan 2-7 berjalan mulus sebagai satu sistem utuh, siap didemokan.

Istilah Penting

Integration testing

Menguji sistem sebagai satu kesatuan yang lengkap, bukan cuma menguji tiap modul sendiri-sendiri, mengungkap bug yang hanya muncul saat bagian-bagian digabung.

Deployment

Proses menyiapkan & menjalankan sebuah sistem software supaya benar-benar siap dipakai, bukan cuma jalan di komputer pengembangnya sendiri.

Debugging

Proses menelusuri & memperbaiki sumber sebuah kesalahan (bug) di dalam kode.

End-to-end

Menguji sebuah alur dari awal sampai akhir secara utuh, misalnya dari robot dinyalakan, tersambung wifi, sampai benar-benar menjawab pertanyaan.

Konsep Kunci

Integration testing: menguji sistem sebagai satu kesatuan, bukan cuma tiap bagian sendiri-sendiri.

Debugging lintas modul: menelusuri bug yang muncul justru karena dua fitur bertabrakan.

Deployment dasar: menjalankan seluruh sistem sekali jalan lewat `run.sh`.

Penjelasan Lengkap

Di Pertemuan 1, sempat disebutkan bahwa jurang terbesar tim pemula biasanya muncul justru di titik sambungan antar modul, bukan di dalam satu modul. Pertemuan terakhir ini adalah saat jurang itu benar-benar dihadapi langsung, setelah enam pertemuan sebelumnya membangun tiap potongan secara terpisah.

Kenapa bug paling berbahaya muncul di titik sambungan

Empat orang di Tim Software masing-masing membangun & menguji modulnya sendiri-sendiri, dan tiap modul sudah terbukti bekerja sendiri-sendiri. Tapi begitu semuanya digabung, muncul interaksi yang tidak pernah terlihat saat modul diuji terpisah: dua bagian kode yang sama-sama ingin memakai perangkat audio di saat bersamaan, asumsi soal urutan kejadian yang benar saat diuji sendirian tapi ternyata salah saat berjalan bersamaan dengan proses lain. Menguji "apakah bagian ini bekerja" adalah satu keterampilan; menguji "apakah semua bagian bekerja bersama-sama dengan benar" adalah keterampilan yang sama sekali berbeda, dan sering kali di situlah proyek nyata benar-benar diuji.

`run.sh` sebagai kontrak, bukan cuma jalan pintas

Skrip `run.sh` bukan sekadar tombol praktis untuk malas mengetik banyak perintah. Ia adalah cara menuliskan urutan langkah yang *pasti benar* untuk menyalakan seluruh sistem, setiap kali, tanpa ada langkah yang terlewat atau salah urutan. Tanpa skrip ini, menjalankan RoLi akan bergantung pada ingatan satu-dua orang tentang "urutan yang benar", pengetahuan yang mudah hilang begitu orang itu tidak ada di tempat. Skrip deployment mengubah pengetahuan yang tadinya tersimpan di kepala orang, menjadi sesuatu yang tertulis dan bisa diulang siapa saja.

Dari "kode yang jalan" menjadi "sistem yang siap didemokan"

Ada jarak yang sering diremehkan antara kode yang berhasil jalan sekali di laptop pengembang, dengan sistem yang cukup andal untuk didemokan langsung di depan orang banyak: jaringan wifi lab sungguhan (bukan hotspot laptop yang stabil), suara ramai penonton yang bisa mengganggu mikrofon, dan pertanyaan tak terduga yang diajukan orang yang belum pernah memakai RoLi sebelumnya. Pertemuan ini adalah saat tim sengaja menciptakan kondisi senyata mungkin itu sebelum hari-H, bukan berharap semuanya kebetulan lancar.

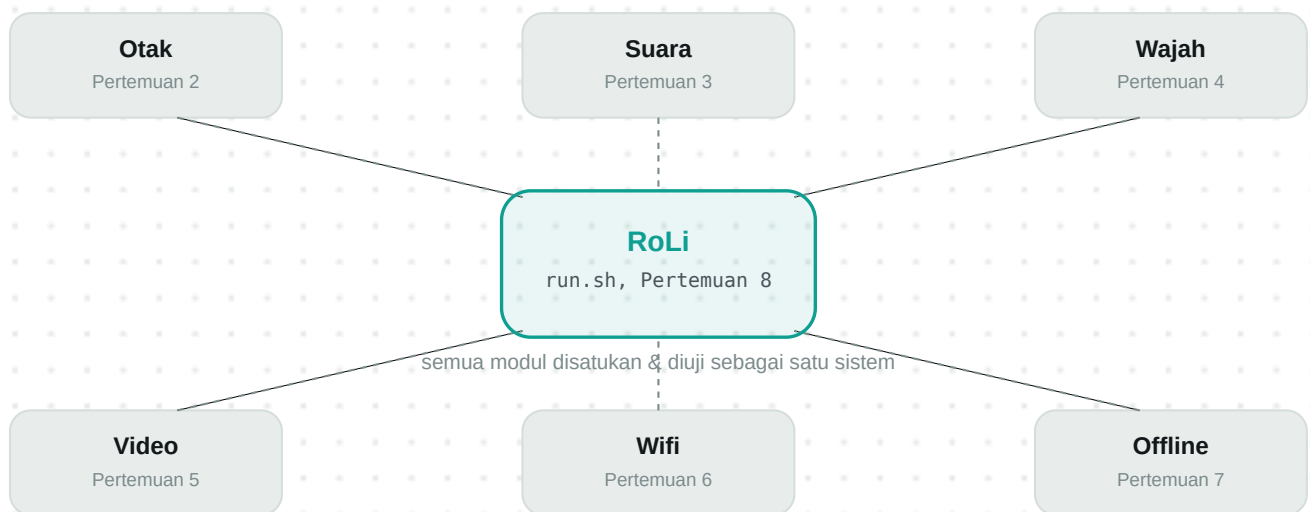
Jurang yang perlu diwaspadai

Bug klasik "kemarin jalan, sekarang kok tidak" biasanya bukan soal kode yang tiba-tiba berubah sendiri, tapi soal kondisi lingkungan yang berubah tanpa disadari: proses lama yang masih menyala di latar belakang, baterai yang mulai lemah, atau perangkat yang belum di-restart bersih. Kondisi seperti ini tidak selalu kelihatan saat menguji dalam kondisi terisolasi, tapi sangat nyata saat robot sudah menyala berjam-jam menjelang demo.

Tekanan waktu menjelang demo publik juga memaksa keputusan sulit yang sangat nyata: bug mana yang wajib diperbaiki sekarang juga, dan bug mana yang "cukup diketahui saja" lalu dihindari saat demo. Memutuskan ini dengan kepala dingin, bukan mencoba memperbaiki semuanya di menit-menit terakhir, adalah keterampilan rekayasa yang sama pentingnya dengan menulis kode itu sendiri.

Menutup lingkaran

Tiga jurang yang disebutkan di Pertemuan 1 (internet yang tidak stabil, bug di titik sambungan antar modul, dan perbedaan laptop vs Raspberry Pi) semuanya sudah punya jawabannya sekarang: mode offline (Pertemuan 7) untuk jurang pertama, pertemuan ini untuk jurang kedua, dan pengujian langsung di perangkat fisik sepanjang seluruh proses untuk jurang ketiga. Delapan pertemuan yang terasa terpisah-pisah di awal, ternyata membentuk satu jawaban utuh.



Kaitan Kompetensi Lebih Luas

Menguji sistem sebagai satu kesatuan adalah tahap yang sering dilupakan pemula, padahal krusial di proyek nyata: bug yang paling berbahaya justru muncul di titik sambungan antar bagian.

Alat & Bahan

`main.py``run.sh`

Aktivitas Hands-on

1. Uji skenario end-to-end nyata: nyalakan → sambung wifi → ngobrol → minta video → lepas internet, dst.
2. Catat & perbaiki bug bersama tim.
3. Latihan demo untuk presentasi akhir.

Hasil / Tolok Ukur

- ✓ RoLi versi prototipe siap didemokan secara utuh: dari nyala sampai ngobrol, semua fitur menyatu.

Bug paling "mengejutkan" apa yang kalian temukan saat menggabungkan semua fitur jadi satu?