Stage1 Proposal

۱. ساختار کلی سه مرحله (Stage 1, 2, 3)

Stage 1 – مبانی Raspberry Pi ،ROS 2، بینایی پایه و کنترل موتور

حوزه‌های ارزیابی:

- راه‌اندازی و پیکربندی **Raspberry Pi 5**
- مبانی **ROS 2** (نود، تاپیک، سرویس، workspace، لانچ)
- کار با **دوربین** و پیاده‌سازی پایپلاین‌های ساده بینایی با مدل‌های YOLO (Object Detection و Segmentation)
- کنترل سرعت موتور DC (RPM) از طریق **Arduino** با پروتکل **Serial**
- شبیه‌سازی ساده روی لپ‌تاپ + Teleop
- معماری **Client–Server**:
  - Raspberry Pi به‌عنوان **سرور** (پردازش)
  - لپ‌تاپ به‌عنوان **کلاینت** (نمایش، Teleop، شبیه‌سازی)

هدف Stage 1: کاهش ۲۰ داوطلب به حدود ۸ **تا** ۱۰ **نفر**.

Stage 2 – بینایی پیشرفته و ناوبری شبیه‌سازی‌شده

حوزه‌های ارزیابی:

- یکپارچه‌سازی مدل‌های **pretrained** بینایی (Object Detection / Segmentation پیشرفته)
- پیاده‌سازی ناوبری با **Nav2**
- پیاده‌سازی **SLAM / AMCL** در محیط شبیه‌سازی(isaacsim or gazebo)
- مدیریت سنسورها، نقشه، مسیر و رفتار ربات در شبیه‌ساز

هدف Stage 2: کاهش ۸–۱۰ نفر به **۴ نفر**.

Stage 3 – یکپارچه‌سازی کامل روی ربات واقعی

حوزه‌های ارزیابی:

- پیاده‌سازی ناوبری روی ربات واقعی
- یکپارچه‌سازی بینایی، ناوبری، بازوی رباتیک و **MoveIt2**
- طراحی و پیاده‌سازی سناریوی کامل مأموریت ربات
- ارزیابی **کار تیمی، استحکام سیستم، دیباگ و تحمل خطا**

هدف Stage 3: انتخاب نهایی **۲ عضو اصلی** و **۱ عضو ذخیره** برای تیم.

توجه: این سند، نسخه‌ی نهایی و حرفه‌ای مربوط به **Stage 1** را به‌طور کامل تشریح می‌کند و نقش آن را در فرآیند سه‌مرحله‌ای نشان می‌دهد.

۲. ساختار Stage 1

Stage 1 شامل دو بخش اصلی است:

1. **بخش تئوری**
   - مدت‌زمان: حدود **۶۰ تا ۹۰ دقیقه**
   - محورها:
- مفاهیم پایهRos2:

(node، topic، service، action، namespace، QoS)

- معرفی معماری DDS و تنظیمات شبکه (Domain ID، discovery)
- مفاهیم پایه لینوکس (simple command lines)و کار با Raspberry Pi
- معرفی ساختار کلی سیستم (Pi، Arduino، دوربین، لپ‌تاپ، شبیه‌ساز)

2. **بخش عملی**
   - مدت‌زمان: حدود **۲ تا ۳ ساعت**
   - شامل ۵ تسک اصلی:

- راه‌اندازی Raspberry Pi 5
- ساخت workspace و پیاده‌سازی Pub/Sub و Service ساده در ROS 2
- دوربین + بینایی (Detection و Segmentation) + کنترل RPM موتور از طریق Arduino (سریال)
- شبیه‌سازی ساده روی لپ‌تاپ + Teleop + انتشار RGB/Depth برای Raspberry Pi
- معماری Client–Server و یکپارچه‌سازی نهایی

۳. الزام مشترک: استفاده از rqt_graph و تحلیل گراف ROS

الزام عمومی برای تمام تسک‌های عملی Stage 1:

در هر مرحله‌ای که نودها و تاپیک‌های ROS 2 در حال اجرا هستند، داوطلب موظف است:

1. ابزار **rqt_graph** را اجرا کند.
2. گراف نودها، تاپیک‌ها و (در صورت پشتیبانی) سرویس‌ها را نمایش دهد.
3. به‌صورت شفاهی و دقیق توضیح دهد:
   - هر نود چه نقشی دارد (action، service client، service server، subscriber، publisher، …)
   - هر تاپیک چه داده‌ای را و در چه جهتی منتقل می‌کند.
   - تفاوت جریان داده در **Publish/Subscribe** و **Service (Request/Response)** چیست.
   - …

این الزام، بخش مهمی از ارزیابی **درک معماری سیستم** و توانایی **دیباگ و تحلیل** داوطلب است.

۴. Task 1 – راه‌اندازی Raspberry Pi 5

هدف: ارزیابی توانایی داوطلب در راه‌اندازی اولیه Pi، کار با لینوکس و آماده‌سازی محیط ROS 2.

الزامات Task 1

- بوت کردن Raspberry Pi 5:
  - با مانیتور (GUI) یا به‌صورت **Headless** با SSH.
- تنظیم شبکه:
  - اتصال به WiFi یا Ethernet
  - پیدا کردن IP دستگاه و برقراری ارتباط از لپ‌تاپ
- اطمینان از نصب صحیح ROS 2:
  - اجرای دستوراتی مانند:
- ros2 doctor
- ros2 topic list

الزام rqt_graph در Task 1 (در صورت اجرای نود تستی)

در صورتی‌که برای تست، نود ساده‌ای (مثلاً publisher نمونه‌ی demo_nodes_cpp) اجرا شود:

- داوطلب باید rqt_graph را باز کند، نودها و تاپیک‌های فعال را نمایش دهد و توضیح دهد:
  - هر نود چه وظیفه‌ای دارد
  - چه تاپیک‌هایی در حال انتشار / دریافت هستند
  - نحوه تشخیص جهت جریان داده در گراف.

۵. Task 2

(ROS 2 ساده در Service و Workspace، Pub/Sub)

هدف: سنجش درک پایه از مفاهیم **publisher**، **subscriber**، **service**، **client** در ROS 2 و توانایی ساخت و بیلد یک پکیج ساده.

۵.۱ ساخت workspace و Pub/Sub ساده

الزامات:

- ایجاد یک workspace (مثلاً ~/ros2_ws)
- ایجاد یک پکیج ساده (مثلاً my_first_pkg) به زبان C++ و Python
- پیاده‌سازی:
  - یک **Publisher** که به‌صورت دوره‌ای (مثلاً هر ۰.۵ ثانیه) یک پیام متنی (string) روی تاپیک / chatter منتشر کند.

(With rclpy)

- یک **Subscriber** که به تاپیک /chatter گوش دهد و پیام دریافتی را در ترمینال چاپ کند.

(With rclcpp)

- بیلد با colcon build و اجرای نودها.

(With only one launch file)

- استفاده از ابزارهای CLI:
  - ros2 topic list برای مشاهده لیست تاپیک‌ها
  - ros2 topic echo /chatter برای مشاهده پیام‌ها
  - ros2 node list برای مشاهده نودهای فعال

۵.۲ پیاده‌سازی Service ساده (Service + Client)

برای تکمیل درک معماری ROS 2، در این Task یک سرویس ساده نیز پیاده‌سازی می‌شود.

الزامات:

- استفاده از سرویس آماده‌ی example_interfaces/srv/AddTwoInts (یا معادل آن)
- پیاده‌سازی یک **Service Server**:
  - تعریف نودی که سرویس /add_two_ints را ارائه می‌کند
  - دریافت دو عدد صحیح به‌عنوان Request
  - محاسبه جمع دو عدد و ارسال آن به‌عنوان Response
  - چاپ Log از Request و Response در ترمینال
- پیاده‌سازی یک **Service Client**:
  - ارسال درخواست با دو عدد (مثلاً ۲ و ۳)
  - دریافت پاسخ (مثلاً ۵)
  - چاپ نتیجه‌ی دریافت‌شده
- آزمون سرویس با CLI (در صورت لزوم):
  - پس از اجرای server، ارسال درخواست:

"{a: 2, b: 3}"

- بررسی صحت پاسخ.

۵.۳ الزام rqt_graph در Task 2

- **برای Pub/Sub:**
  - اجرای rqt_graph توسط داوطلب
  - نمایش نود publisher، نود subscriber و تاپیک /chatter
  - توضیح:
- کدام نود publisher است
- کدام نود subscriber است

- تاپیک چه نقشی دارد و جهت فلش‌ها چه معنایی دارد
- **برای Service:**
  - نمایش (در حد امکانات rqt_graph و/یا توضیح شفاهی)
  - سرویس /add_two_ints و نودهای server و client
  - توضیح:
    - نام سرویس چیست
    - کدام نود سرویس را ارائه می‌کند (server)
    - کدام نود درخواست می‌فرستد (client)
    - تفاوت بین داده‌های **یک‌طرفه و پیوسته** در Pub/Sub و داده‌های **درخواست/پاسخ** در Service.

۶. Task 3 – دوربین + بینایی (Detection و Segmentation) + کنترل RPM موتور از طریق Arduino (Serial)

۶.۱ معماری سخت‌افزاری

- دوربین متصل به **Raspberry Pi**
- موتور DC مجهز به انکدر، متصل به **Arduino**
- اتصال Arduino به Raspberry Pi از طریق **Serial (USB)**
- تمام پردازش‌ها (بینایی، کنترل RPM، ارتباط سریال) روی **Raspberry Pi**
- لپ‌تاپ به‌عنوان **کلاینت** برای:
  - اجرای rviz2 برای نمایش
  - اجرای Teleop و شبیه‌سازی (در Task 4)

۶.۲ جریان دوربین: Raspberry Pi → Laptop

الزامات:

- راه‌اندازی نود درایور دوربین روی Raspberry Pi
- انتشار تصویر RGB روی تاپیکی مانند:
  - camera/rgb/image_raw/ یا /rgb
- تنظیمات شبکه / DDS به‌گونه‌ای که لپ‌تاپ بتواند تاپیک‌های Pi را مشاهده کند.
- اجرای rviz2 روی لپ‌تاپ و نمایش تصویر دوربین.

الزام rqt_graph – دوربین:

- نمایش نود دوربین، تاپیک تصویر و نودهای مصرف‌کننده (مثلاً نودهای بینایی) در rqt_graph
- توضیح مسیر حرکت داده‌ی تصویر از دوربین تا نودهای بعدی.

۶.۳ پایپلاین Object Detection

نود تشخیص (روی Raspberry Pi):

- subscribe - /camera/rgb/image_raw
- استفاده از مدل YOLO8-n

برای **تشخیص میوه‌ها** (یا مدل سبک مشابه)

- انتشار خروجی روی

/detections شامل:

  - Bounding Box
  - کلاس میوه (label)
  - Confidence

فایل لانچ object_detection.launch.py:

این فایل باید نودهای زیر را به‌صورت یکپارچه اجرا کند:

1. نود درایور دوربین
2. نود Object Detection
3. نود کنترل RPM (بخش ۶.۵)

نمایش روی لپ‌تاپ:

- در rviz2:
  - تصویر RGB
  - bounding box

الزام rqt_graph – پایپلاین Detection:

- اجرای rqt_graph و نمایش:
  - نود دوربین
  - نود Object Detection
  - نود کنترل RPM
  - تاپیک‌های /camera/rgb/image_raw و /detections
- توضیح تمامی مسیرهای داده:
  - تصویر از دوربین Detection -> کنترل Arduino -> RPM (از طریق سریال).

۶.۴ پایپلاین Segmentation

نود Segmentation (روی Raspberry Pi):

- subscribe -> /camera/rgb/image_raw
- استفاده از مدل Segmentation برای تشخیص و جداسازی میوه‌ها
- انتشار خروجی روی:
  - segmentation_result/ یا segmentation_mask/

فایل لانچ segmentation.launch.py:

- اجرای:
  1. نود درایور دوربین
  2. نود Segmentation
  3. نود کنترل RPM (مشترک با Detection)

نمایش روی لپ‌تاپ:

- در rviz2:
  - تصویر RGB
  - ماسک‌های Segmentation

الزام rqt_graph – پایپلاین Segmentation:

- نمایش گراف کامل:
  - نود دوربین
  - نود Segmentation
  - نود کنترل RPM
  - تاپیک‌های ورودی/خروجی
- توضیح تفاوت این گراف با گراف پایپلاین Detection (نود و تاپیک خروجی متفاوت).

۶.۵ نود کنترل RPM (مشترک برای Detection و Segmentation)

رفتار نود کنترل RPM:

1. **ورودی از Object Detection:**
   - subscribe به detections/
   - استخراج نوع میوه
   - نگاشت میوه -> RPM (مثلاً در روز آزمون گفته می‌شود:
     (apple → 10 RPM, orange → 15 RPM
   - محاسبه RPM هدف
2. **ورودی از Segmentation:**
   - subscribe به segmentation_result/ یا segmentation_mask/

- تعیین میوه / ناحیه هدف
- استفاده از همان نگاشت میوه
- محاسبه RPM هدف

3. **ارسال فرمان به Arduino:**

- ارسال فرمان RPM هدف از طریق Serial (USB) به Arduino

الزام rqt_graph – نود کنترل RPM:

- توضیح نود کنترل RPM در گراف:
  - کدام تاپیک‌ها را subscribe می‌کند
  - اگر تاپیک خروجی ROS دارد (مثلاً /motor_rpm_cmd) نمایش آن در گراف
- توضیح شفاهی:
  - اتصال سریال با Arduino چگونه در کنار گراف ROS قرار می‌گیرد (به‌عنوان پل بین ROS و سخت‌افزار).(bridge)

۶.۶ Teleop → Pi → Arduino (فرمان RPM از لپ‌تاپ)

سناریو:

- Teleop روی لپ‌تاپ فرمان حرکت را روی /cmd_vel (یا /motor_cmd) منتشر می‌کند.
- Raspberry Pi به /cmd_vel یا /motor_cmd subscribe می‌کند.
- نود کنترل RPM:
  - فرمان خطی/زاویه‌ای یا فرمان مجرد Teleop را به RPM مناسب تبدیل می‌کند
  - RPM را از طریق سریال به Arduino ارسال می‌کند
- Arduino موتور را با RPM مورد نظر کنترل می‌کند.

۷. Task 4 – شبیه‌سازی روی لپ‌تاپ + Teleop + انتشار RGB/Depth به Raspberry Pi

نکته: شبیه‌ساز فقط روی **لپ‌تاپ** اجرا می‌شود.

۷.۱ مدل ربات و محیط شبیه‌سازی

- استفاده از Gazebo یا Isaac Sim
- مدل URDF/Description ربات مشابه ربات واقعی Stage 0
- افزودن سنسور:
  - دوربین RGB-D (یا مجموعه RGB + Depth)

۷.۲ Teleop در شبیه‌سازی

- اجرای teleop_twist_keyboard روی لپ‌تاپ
- اطمینان از حرکت ربات در محیط شبیه‌ساز.

الزام rqt_graph – شبیه‌سازی و Teleop:

- نمایش نود Teleop، نود کنترل‌کننده ربات شبیه‌سازی‌شده و تاپیک در rqt_graph
- توضیح فرایند

۷.۳ انتشار RGB/Depth از شبیه‌ساز به Raspberry Pi

- روی لپ‌تاپ (در شبیه‌ساز یا Bridge ROS 2):
  - انتشار camera/rgb/image_raw/
  - انتشار camera/depth/image_raw/
- روی Raspberry Pi:
  - subscribe به این تاپیک‌ها
  - استفاده از داده‌ها در پایپلاین‌های بینایی (Detection و Segmentation)

الزام rqt_graph – اتصال شبیه‌ساز → Pi:

- نمایش نودهای publisher (شبیه‌ساز) و subscriber (روی Pi) در rqt_graph
- توضیح:
  - داده‌ی تصویر چگونه از شبیه‌ساز به Pi منتقل می‌شود
  - این داده چگونه وارد پایپلاین بینایی و سپس کنترل RPM می‌شود.

8. جمع‌بندی و معیارهای ارزیابی Stage 1

8.۱ محورهای اصلی ارزیابی

- درک مفاهیم پایه ROS 2 (نود، تاپیک، سرویس، workspace، لانچ)
- توانایی کار با Raspberry Pi و لینوکس
- پیاده‌سازی Pub/Sub و Service ساده (Task 2)
- راه‌اندازی دوربین و نمایش آن در rviz2 (Task 3)
- پیاده‌سازی پایپلاین‌های ساده بینایی (Detection و Segmentation) و اتصال آن‌ها به کنترل RPM (Task 3)
- راه‌اندازی شبیه‌سازی، Teleop و انتقال داده به Pi (Task 4)
- درک و پیاده‌سازی معماری Client–Server و توانایی تحلیل گراف سیستم در rqt_graph (Task 5)
- مهارت در دیباگ، مدیریت زمان، خوانایی و ساختار کد

8.۲ وزن‌دهی پیشنهادی (قابل تنظیم)

- مبانی ROS 2 (workspace، Pub/Sub، Service) – حدود ۱۰–۱۵٪
- دوربین و نمایش روی لپ‌تاپ – حدود ۱۰–۱۵٪
- پایپلاین Object Detection + کنترل RPM + تحلیل گراف – حدود ۱۵–۲۰٪
- پایپلاین Segmentation + کنترل RPM + تحلیل گراف – حدود ۱۵–۲۰٪
- شبیه‌سازی + Teleop + انتشار RGB/Depth به Pi + تحلیل گراف – حدود ۱۰–۱۵٪
- معماری کامل Client–Server و تسلط بر rqt_graph – حدود ۱۵٪
- کیفیت کلی کدنویسی، دیباگ و ارائه شفاهی – باقی‌مانده درصد.