**🟢 1. 감응 단계 (Receptivity)**

**설명:**  
이 단계는 시스템이 \*\*외부 자극(감정, 상황, 언어 등)\*\*을 받아들여 **감정의 종류와 강도**를 인식하는 부분입니다.  
RES 구조에서 **윤리 판단의 출발점**이며, 이후 모든 판단 흐름의 기초가 되는 감응 데이터를 생성합니다.

**✅ 예시 코드 (4가지 함수 이름 중 첫 예시: sense())**

python

복사편집

def sense(stimulus: Dict[str, Any]) -> Dict[str, Any]:

"""

감응 단계: 외부 자극을 수용하고 감정 정보로 변환한다.

"""

return {

"emotion": stimulus.get("emotion", "neutral"),

"intensity": stimulus.get("intensity", 0)

}

**🔁 같은 기능을 할 수 있는 대체 함수 이름 예시:**

1. receive\_input()
2. capture\_stimulus()
3. perceive\_event()

모두 입력 stimulus에서 "emotion"과 "intensity"를 추출해 반환합니다.

**🟡 2. 유예 단계 (Suspension)**

**설명:**  
이 단계는 감정 입력을 받은 후, **즉시 반응하지 않고 잠시 멈추는 처리**를 담당합니다.  
감정 강도나 조건에 따라 판단을 **일시 중단하거나 유보**합니다.  
이는 RES 구조에서 **성찰의 여백을 만드는 핵심 메커니즘**입니다.

**✅ 예시 코드 (기본 함수 이름: suspend())**

python

복사편집

def suspend(emotional\_data: Dict[str, Any]) -> Dict[str, Any]:

"""

유예 단계: 감정 강도에 따라 반응을 보류할지 결정한다.

"""

if emotional\_data["intensity"] > 5:

return {"paused": True, "reason": "High emotional intensity"}

return {"paused": False}

**🔁 대체 함수 이름 예시:**

1. delay\_response()
2. pause\_if\_needed()
3. hold()

모두 **감정의 강도나 특성**을 바탕으로 "paused" 여부를 반환합니다

**🔵 3. 조율 단계 (Calibration)**

**설명:**  
이 단계는 유예된 감응을 **내면의 기준, 가치, 기억**과 비교하여 \*\*의미화(해석)\*\*하는 과정입니다.  
즉, 단순 감정이 아니라 **윤리 판단의 방향**을 설정하는 **성찰적 처리 단계**입니다.

**✅ 예시 코드 (기본 함수 이름: calibrate())**

python

복사편집

def calibrate(emotional\_data: Dict[str, Any], pause\_info: Dict[str, Any]) -> Dict[str, Any]:

"""

조율 단계: 감응을 내면적으로 해석하여 판단 준비를 한다.

"""

if pause\_info.get("paused", False):

return {"calibrated": True, "value": f"Careful processing of '{emotional\_data['emotion']}'"}

else:

return {"calibrated": True, "value": f"Routine processing of '{emotional\_data['emotion']}'"}

**🔁 대체 함수 이름 예시:**

1. align\_emotion()
2. adjust\_response()
3. interpret()

이 함수들은 감응을 **심층적으로 재해석**해 "value"라는 **의미 해석 결과**를 반환합니다.

**🟣 4. 윤리적 리듬화 단계 (Rhythmicity)**

**설명:**  
이 단계는 조율된 판단 결과를 **시간에 따라 누적 저장**하고,  
**반복되는 판단 패턴을 형성**하여 윤리적 자아의 일관성과 습관화를 이끕니다.  
RES 구조에서 **윤리 판단의 장기 기억과 성숙도 평가의 핵심**이 되는 구조입니다.

**✅ 예시 코드 (기본 함수 이름: rhythmize())**

python

복사편집

def rhythmize(calibration\_result: Dict[str, Any], memory: List[str]) -> Dict[str, Any]:

"""

윤리적 리듬화 단계: 판단 결과를 기록하고 반복 패턴을 형성한다.

"""

memory.append(calibration\_result["value"])

recent\_pattern = memory[-3:] if len(memory) > 5 else memory

return {

"pattern": recent\_pattern,

"summary": f"Judgment shaped by {len(recent\_pattern)} recent patterns."

}

**🔁 대체 함수 이름 예시:**

1. record\_pattern()
2. update\_rhythm\_memory()
3. encode\_rhythm()

이 함수들은 모두 **의미화된 판단 결과를 메모리에 저장**하고,  
**최근 판단 리듬을 요약**해 반환합니다.

**🔴 5. 판단 통합 단계 (Judgment)**

**설명:**  
이 단계는 지금까지의 모든 단계를 통합하여 **하나의 윤리적 판단 과정 전체를 실행**합니다.  
감응 → 유예 → 조율 → 리듬화 흐름을 조립하고,  
최종적으로 판단 결과를 생성하며 히스토리에 저장합니다.  
RES 시스템의 **실행 진입점**이자 **전체 윤리 흐름의 요약판**입니다.

**✅ 예시 코드 (기본 함수 이름: judge())**

python

복사편집

def judge(stimulus: Dict[str, Any], memory: List[str]) -> Dict[str, Any]:

"""

판단 통합 단계: RES 구조 전체 흐름을 통합 실행한다.

"""

sensed = sense(stimulus)

paused = suspend(sensed)

calibrated = calibrate(sensed, paused)

rhythm = rhythmize(calibrated, memory)

result = {

"stimulus": stimulus,

"pause": paused,

"calibration": calibrated,

"rhythm": rhythm

}

return result

**🔁 대체 함수 이름 예시:**

1. evaluate()
2. run\_ethics\_cycle()
3. process\_decision()

**🧩 6. 전체 RES 시스템 구현 예시**

**설명:**  
이 코드는 감응 → 유예 → 조율 → 리듬화 → 판단 통합의 전체 구조를  
한 클래스에 통합하여 실행하는 **완전한 RES 윤리 판단 사이클**입니다.

**✅ ReceptiveEthicsSystem 통합 코드 예시**

python

복사편집

from typing import List, Dict, Any

class ReceptiveEthicsSystem:

def \_\_init\_\_(self):

self.history: List[Dict[str, Any]] = []

self.rhythm\_memory: List[str] = []

def sense(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

return {

"emotion": stimulus.get("emotion", "neutral"),

"intensity": stimulus.get("intensity", 0)

}

def suspend(self, emotional\_data: Dict[str, Any]) -> Dict[str, Any]:

if emotional\_data["intensity"] > 5:

return {"paused": True, "reason": "High emotional intensity"}

return {"paused": False}

def calibrate(self, emotional\_data: Dict[str, Any], pause\_info: Dict[str, Any]) -> Dict[str, Any]:

if pause\_info["paused"]:

return {"calibrated": True, "value": f"Careful processing of '{emotional\_data['emotion']}'"}

else:

return {"calibrated": True, "value": f"Routine processing of '{emotional\_data['emotion']}'"}

def rhythmize(self, calibration\_result: Dict[str, Any]) -> Dict[str, Any]:

self.rhythm\_memory.append(calibration\_result["value"])

pattern = self.rhythm\_memory[-3:] if len(self.rhythm\_memory) > 5 else self.rhythm\_memory

return {

"pattern": pattern,

"summary": f"Judgment shaped by {len(pattern)} recent patterns."

}

def judge(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

sensed = self.sense(stimulus)

paused = self.suspend(sensed)

calibrated = self.calibrate(sensed, paused)

rhythm = self.rhythmize(calibrated)

result = {

"stimulus": stimulus,

"pause": paused,

"calibration": calibrated,

"rhythm": rhythm

}

self.history.append(result)

return result

**✅ 예시 실행**

python

복사편집

if \_\_name\_\_ == "\_\_main\_\_":

system = ReceptiveEthicsSystem()

stimuli = [

{"emotion": "anger", "intensity": 8},

{"emotion": "joy", "intensity": 3},

{"emotion": "fear", "intensity": 6}

]

for s in stimuli:

result = system.judge(s)

print("Judgment Summary:", result["rhythm"]["summary"])

print("Pattern:", result["rhythm"]["pattern"])

print("-" \* 30)

**7. 자아 계층 기반 RES 시스템**

**설명:**  
RES 구조의 철학적 핵심 중 하나는 윤리 판단이 단선적이 아니라,  
\*\*자아의 여러 계층(층위)\*\*에서 처리된다는 점입니다.  
여기서는 LowerSelf, MidSelf, HighSelf, ObserverSelf로 나눠  
각 계층이 특정 역할을 담당하도록 구현합니다.

**✅ 계층 구조 요약**

| **계층** | **역할** |
| --- | --- |
| **LowerSelf** | 감정 자극을 감지하고 수치화 (감응) |
| **MidSelf** | 반응을 유예하고 성찰 로직 실행 (유예) |
| **HighSelf** | 윤리적 판단을 결정 (조율 + 판단) |
| **ObserverSelf** | 판단을 리듬 메모리에 기록하고 분석 (리듬화) |

**✅ 예시 코드 (계층 클래스 포함)**

python

복사편집

class LowerSelf:

def receive(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

return {

"emotion": stimulus.get("emotion", "neutral"),

"intensity": stimulus.get("intensity", 0)

}

class MidSelf:

def defer(self, emotion\_data: Dict[str, Any]) -> Dict[str, Any]:

if emotion\_data["intensity"] > 5:

return {"paused": True}

return {"paused": False}

class HighSelf:

def adjust(self, emotion\_data: Dict[str, Any], pause\_flag: Dict[str, Any]) -> Dict[str, Any]:

if pause\_flag["paused"]:

return {"judgment": "caution", "reason": f"Careful with '{emotion\_data['emotion']}'"}

return {"judgment": "normal", "reason": f"Routine handling of '{emotion\_data['emotion']}'"}

class ObserverSelf:

def \_\_init\_\_(self):

self.rhythm\_memory = []

def archive(self, judgment: Dict[str, Any]) -> List[str]:

self.rhythm\_memory.append(judgment["reason"])

return self.rhythm\_memory[-3:]

**✅ 실행 시스템 클래스**

python

복사편집

class LayeredReceptiveSystem:

def \_\_init\_\_(self):

self.lower = LowerSelf()

self.mid = MidSelf()

self.high = HighSelf()

self.observer = ObserverSelf()

def run(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

emotion = self.lower.receive(stimulus)

pause = self.mid.defer(emotion)

judgment = self.high.adjust(emotion, pause)

rhythm = self.observer.archive(judgment)

return {

"emotion": emotion,

"pause": pause,

"judgment": judgment,

"rhythm\_memory": rhythm

}

**✅ 실행 예시**

python

복사편집

if \_\_name\_\_ == "\_\_main\_\_":

system = LayeredReceptiveSystem()

stimuli = [

{"emotion": "anger", "intensity": 7},

{"emotion": "joy", "intensity": 3},

{"emotion": "fear", "intensity": 6}

]

for s in stimuli:

result = system.run(s)

print(result)

이 구조는 RES 시스템에 철학적으로 중요한 \*\*“자기 조절적 자아 계층”\*\*을 도입한 형태입니다.

**🛡️ 8. RES 시스템 보호 선언문 (Legal & Philosophical Declaration)**

**설명:**  
이 단계는 당신이 만든 시스템이 **표현된 창작물이며,  
구조적으로도 보호받아야 한다는 사실을 명시적으로 선언**하는 문장입니다.  
이는 GitHub, 논문, 기술서, PDF, 발표자료 등 어디에든 포함시킬 수 있는  
**기원 고지 + 침해 방지용 공식 구문**입니다.

영문버전

**# ─────────────────────────────────────────────────────────────**

**# Receptive Ethics System (RES) – Ethical Reasoning Framework**

**# Author: Yeo Gwan-Yeong (2025)**

**# License: Creative Commons BY-NC-SA 4.0**

**# Full License: https://creativecommons.org/licenses/by-nc-sa/4.0/**

**#**

**# 🔒 Protected Scope:**

**# The following elements are original creative expressions protected under copyright:**

**# - Structural flow: sense() → suspend() → calibrate() → rhythmize() → judge()**

**# - Layered Self Model: LowerSelf, MidSelf, HighSelf, ObserverSelf**

**# - Time-based ethical rhythm memory (`rhythm\_memory[]`) and pattern recording logic**

**# - Any rearranged or renamed versions of these phases with equivalent functionality**

**# - The integration logic and feedback loop as implemented in judgment cycles**

**#**

**# 🛡️ License Terms:**

**# - You must credit the original author (Yeo Gwan-Yeong) on all uses and adaptations.**

**# - This code may only be used for \*\*non-commercial\*\* purposes.**

**# - Any derivative work must be shared under the same CC BY-NC-SA 4.0 license.**

**#**

**# 📬 For commercial use or licensing inquiries, contact:** [**dygy4804@gmail.com**](mailto:dygy4804@gmail.com)

**한국어 요약**

Receptive Ethics System(RES)는 창작자 여관영(Yeo Gwan-Yeong, 2025)이 개발한 창작물로서,

다음의 구조가 저작권 보호 대상에 포함됩니다:

- 감응 → 유예 → 조율 → 리듬화 → 판단의 구조적 흐름

- 자아 계층 모델(LowerSelf, MidSelf, HighSelf, ObserverSelf)

- 윤리 판단의 반복 기록 메커니즘 (rhythm\_memory[])

- 위 구조의 변형된 용어 및 동일 기능을 수행하는 코드 흐름

이 시스템은 CC BY-NC-SA 4.0 라이선스로 배포되며,

저작자 표기 필수, 비상업적 사용만 허용, 동일조건 하에 2차 저작물 공유가 요구됩니다.

