**Avatar operating system based on sensors, data processing, and audiovisual response**

Git : [*Link*](https://github.com/ArielMobileLab/System/blob/main/Autonomous%20Response%20Unit/Avatar_Main_Project/Main%20Version/Brains/Tessy_Brain.py)

## **Overview**

The Avatar Operating System is designed to process environmental sensor data, detect challenges, and generate responses through an generated avatar.

The system consists of three core components:

1. **Perception** – filters sensor data from the environment
2. **Brain** – analyzes the filtered sensors and determines the avatar’s response
3. **Avatar** – generates an avatar using text-to-speech (TTS) and expressive rendering

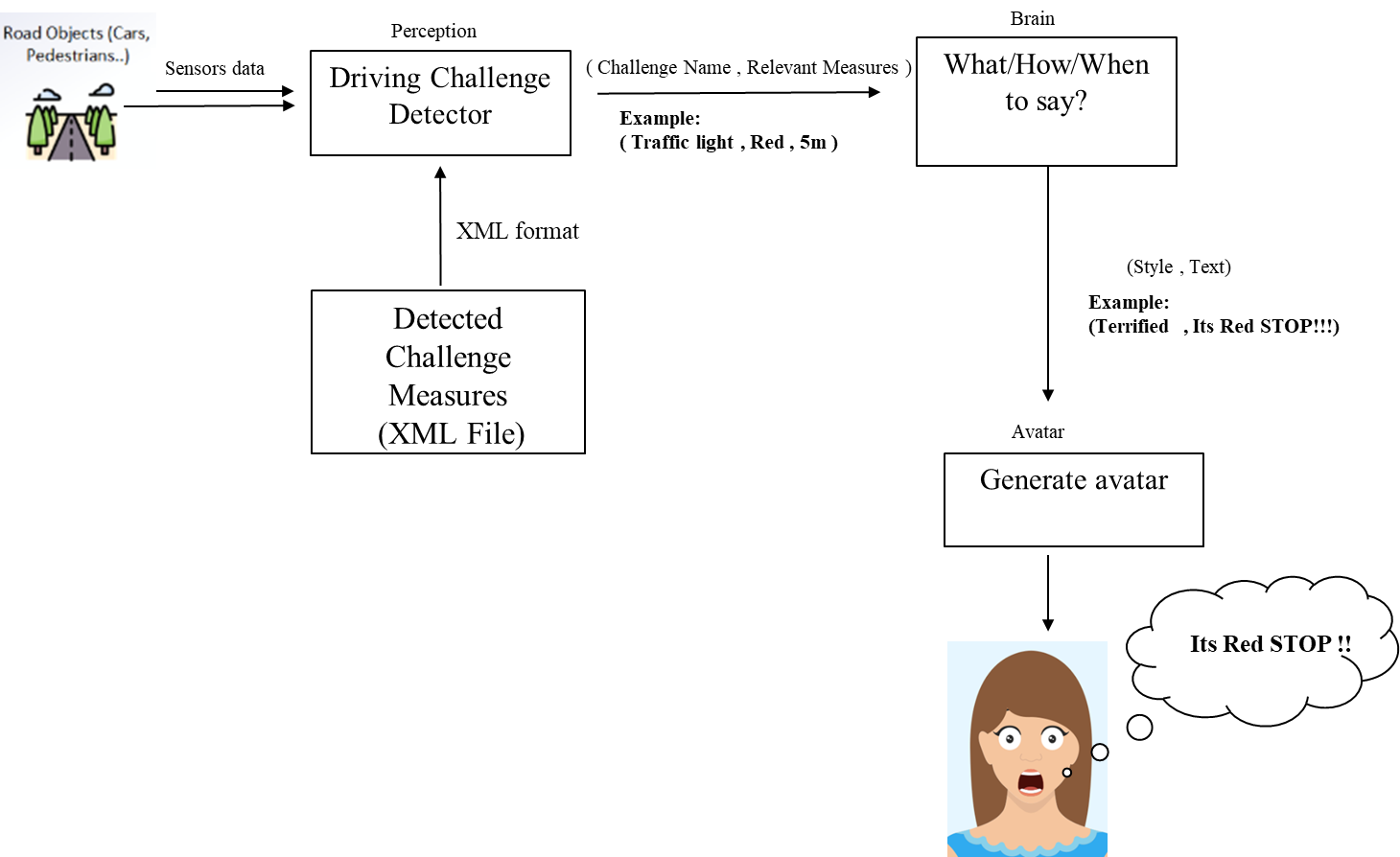


Figure 1: Architecture of Avatar Project

## **1. Perception**

The Perception component receives raw sensor input from the environment and extracts relevant information (traffic lights, pedestrians, emergency events, etc.).

### **Inputs**

* Sensor data from the environment (CARLA simulator API e.g., )
* Conditions from an XML configuration file

### **Processing**

* Filters data according to the XML file
* Converts results into a vector with challenge name and measurements

### **Outputs**

* Vector format: (Challenge name, Relevant measurements)

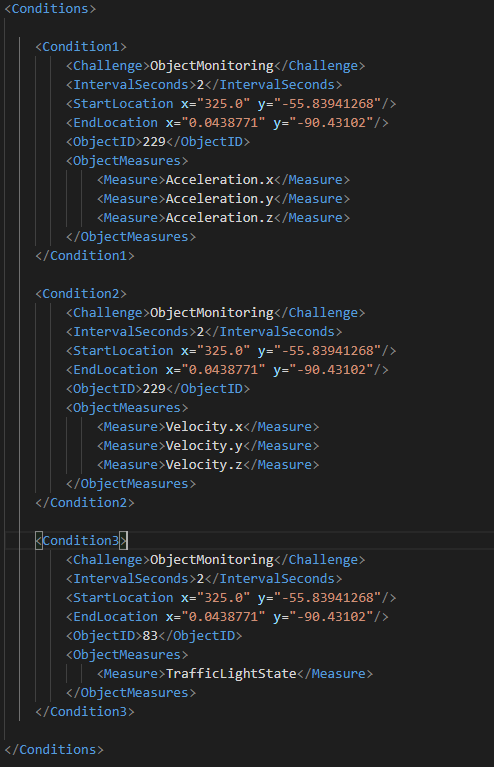


Figure 2: XML file of challenges measures

Reference**:** [CARLA Python API](https://carla.readthedocs.io/en/latest/python_api/)

**2. Brain**

The Brain interprets challenges from Perception and generates speech output specifications for the avatar.

### **Inputs**

* (Challenge name, Relevant measurements) from Perception.

### **Processing**

* Logic rules
* Speech and text rules

### **Outputs**

* Vector format: (Voice name, Category name, Style, Styledegree, Pitch, Rate, Volume, Text))

### Argument Explanation:

* **voice name** – Selected voice model from [Azure Voice Catalog](https://speech.microsoft.com/portal/voicegallery).
* **style** – Emotional tone (cheerful, angry, chat, etc.).
* **styledegree** – Intensity of emotion, range 0.01 – 2.0 (default = 1.0).
* **pitch** – Adjusts tone (low = -20%, medium = 0%, high = +20%).
* **rate** – Speaking speed (x-slow, slow, medium, fast, x-fast).
* **volume** – Loudness (x-soft = 0.2 → x-loud = 1.0).
* **text** – The actual utterance.

### Example – Brain Output:

("en-US-JennyNeural", "TrafficLight", "serious", 1.2, "low", "slow", "loud",

"Caution! Pedestrian ahead, slowing down.")

**Reference:**

* [Azure TTS Styles](https://learn.microsoft.com/nb-no/azure/ai-services/speech-service/language-support?tabs=tts#voice-styles-and-roles)
* [SSML Examples](https://learn.microsoft.com/en-us/azure/ai-services/speech-service/speech-synthesis-markup-voice#custom-voice-style-example)

### **Example brain:**

For example, a brain module that receives acceleration data from the experimental vehicle and processes it: *[*[*Mobile lab git*](https://github.com/ArielMobileLab/System/blob/main/Autonomous%20Response%20Unit/Avatar_Main_Project/Main%20Version/Brains/Tessy_Brain.py)]

## **3. Avatar**

The Avatar component converts the Brain’s output into audiovisual responses using Azure TTS models.

**Input**

* (Voice name, Category name, Style, Styledegree, Pitch, Rate, Volume, Text)) from Brain.

### **Processing**

* Sends requests to Azure’s TTS service
* Produces synthesized speech

### **Training**

For practice without accessing physical hardware, a training version of the AZURA TTS model is available:  *[*[*Mobile lab git*](https://github.com/ArielMobileLab/System/tree/main/Autonomous%20Response%20Unit/Avatar_Main_Project/Voice%20Version%20Azure)]

**Reference:**

* [Azure TTS Styles](https://learn.microsoft.com/nb-no/azure/ai-services/speech-service/language-support?tabs=tts#voice-styles-and-roles)
* [SSML Examples](https://learn.microsoft.com/en-us/azure/ai-services/speech-service/speech-synthesis-markup-voice#custom-voice-style-example)