<< File: sound_detection_implementation.py **>>**

SoundDetectionNode

- node_name: str
- config: dict
- unit_tests: bool
- frequency_sample: int
- speed_of_sound: float
- distance_between_ears: float
- intensity_threshold: float
- verbose_mode: bool
- noise_type: bool
- prop_decrease: float
- localization_buffer_size: int
- frontleft_buffer: np.ndarray
- frontright_buffer: np.ndarray
- accumulated_samples: int
- vad_aggressiveness: int
- vad: webrtcvad.Vad
- vad_frame_duration: float
- vad_frame_size: int
- target_rms: float
- context_duration: float
- context_size: int
- left_context_window: np.ndarray
- use_noise_reduction: bool
- save_audio_duration: float
- save_audio: bool
- sample_count: int
- max_samples_to_save: int
- saved_samples: int
- filtered_buffer: list
- unit_test_path: str
- lock: Lock
- last_status_time: float

+ __init__()
+ spin()
+ on_shutdown()
+ read_json_file(package_name: str) static
+ extract_topics(topic_key: str) static
+ normalize_rms(audio_data: np.ndarray, target_rms=None, min_rms=1e-10): np.ndarray
+ save_test_audio()
+ apply_noise_reduction(current_block: np.ndarray): np.ndarray
+ voice_detected(audio_frame: np.ndarray): bool
+ audio_callback(msg: sound_detection_microphone_msg_file)
+ process_audio_data(msg: sound_detection_microphone_msg_file): (np.ndarray, np.ndarray)
+ is_intense_enough(signal_data: np.ndarray): bool
+ update_buffers(sigIn_frontLeft: np.ndarray, sigIn_frontRight: np.ndarray)
+ localize(sigIn_frontLeft: np.ndarray, sigIn_frontRight: np.ndarray)
+ gcc_phat(sig: np.ndarray, ref_sig: np.ndarray, fs: int, max_tau=None, interp=16): float
+ calculate_angle(itd: float): float
+ publish_angle(angle: float)
+ publish_signal(signal_data: np.ndarray)