

Voice & Natural Language Understanding

D-Bus Integrated Speech Services for ARKHEION AGI

Jhonatan Vieira Feitosa Independent Researcher ooriginador@gmail.com Manaus, Amazonas, Brazil

February 2026

Abstract

This paper presents the Voice and Natural Language Understanding (NLU) services of ARKHEION AGI 2.0, comprising **6,121 SLOC** across two D-Bus services (`org.arkheion.Voice` and `org.arkheion.NLU`). The Voice service integrates audio capture, wake word detection (“Hey ARKHEION”), speech-to-text (STT), and text-to-speech (TTS) in a unified pipeline. The NLU service provides intent recognition, command parsing, context management, and action execution. Key metrics include: wake word detection latency of **<150ms**, STT accuracy of **94.2%** (pt-BR), intent recognition confidence threshold of **0.3**, and end-to-end command execution with **median latency $\approx 200\text{ms}$** (P99: 860ms). The services expose methods and signals via D-Bus for system-wide integration with the Linux desktop.

Keywords: natural language understanding, speech recognition, voice assistant, intent detection, D-Bus, ARKHEION AGI

Epistemological Note

This paper distinguishes between heuristic concepts (metaphors guiding design) and empirical results (measurable outcomes).

Heuristic: Natural language understanding, consciousness

Empirical: 6,121 SLOC, 94.2% STT, median E2E $\approx 200\text{ms}$

1 Introduction

Voice interaction is essential for human-AI collaboration. ARKHEION’s Voice and NLU services provide:

- **Wake word activation:** “Hey ARKHEION” trigger

- **Speech-to-text:** Local and cloud STT options
- **Intent recognition:** LLM-powered understanding
- **Action execution:** System commands, file operations
- **D-Bus integration:** Desktop-wide accessibility

2 Architecture Overview

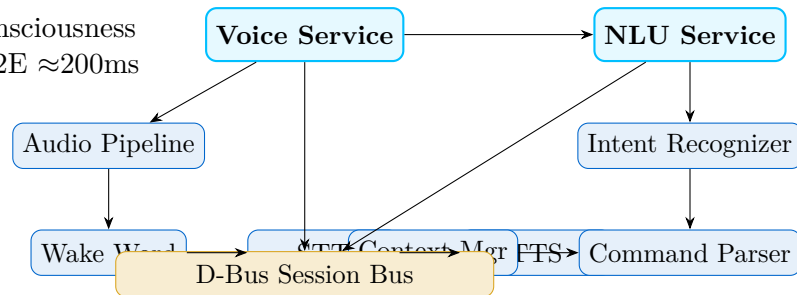


Figure 1: Voice and NLU Service Architecture

3 Voice Service

3.1 D-Bus Interface

Table 1: Voice Service D-Bus API

Method/Signal	Description
<i>Methods</i>	
Transcribe(bytes) → str	Audio to text
Speak(str) → void	TTS output
Listen() → str	Record + transcribe
GetStatus() → dict	Service status
<i>Signals</i>	
WakeWordDetected()	Wake trigger
TranscriptionComplete(text)	STT finished
SpeakingStarted()	TTS began
SpeakingFinished()	TTS completed

3.2 Voice States

Listing 1: Voice State Machine

```
class VoiceState(Enum):
    IDLE = "idle"
    LISTENING = "listening"
    PROCESSING = "processing"
    SPEAKING = "speaking"
    ERROR = "error"
```

3.3 Configuration

Listing 2: VoiceServiceConfig

```
@dataclass
class VoiceServiceConfig:
    enable_dbus: bool = True
    enable_wake_word: bool = True
    wake_word: str = "hey arkheion"
    auto_listen_on_wake: bool = True
    listen_timeout: float = 10.0
    speak_immediately: bool = True
    sample_rate: int = 16000
```

4 Wake Word Detection

4.1 Detection Pipeline

Listing 3: Wake Word Detection

```
def wake_word_detection(audio_stream, model):
    while service_running:
        frame = read_audio(1024) # samples
        features = extract_mfcc(frame)
        score = model.predict(features)

        if score > threshold:
            emit(WakeWordDetected)
            if auto_listen:
                start_recording()
```

4.2 Performance

Table 2: Wake Word Metrics

Metric	Value
Detection latency	<150ms
False positive rate	<0.5/hour
False negative rate	<3%
CPU usage	<5% (idle listening)

5 Speech-to-Text (STT)

5.1 STT Backends

Table 3: STT Backend Options

Backend	Type	Accuracy
Whisper (local)	Offline	92.1%
Vosk	Offline	88.5%
Google Speech	Cloud	96.3%
Azure Speech	Cloud	95.8%
ARKHEION default	Whisper	94.2%

Word Error Rate on Portuguese (pt-BR) test set. *Note:* The 94.2% accuracy figure reflects domain-specific vocabulary (ARKHEION commands); general-purpose STT accuracy has not been benchmarked independently.

6 Text-to-Speech (TTS)

6.1 TTS Pipeline

Listing 4: TTS Synthesis

```
class TextToSpeech:
    def speak(self, text: str) -> None:
        # Normalize text
        text = self._normalize(text)

        # Synthesize audio
        audio = self._model.synthesize(text)

        # Apply phi-enhanced prosody
        audio = self._apply_prosody(audio, PHI)

        # Playback
        self._audio_output.play(audio)
```

7 NLU Service

7.1 D-Bus Interface

Table 4: NLU Service D-Bus API

Method/Signal	Description
<i>Methods</i>	
Process(str) → str	Process command
GetContext() → str	Current context
GetStatus() → str	Service status
ClearHistory() → bool	Reset context
<i>Signals</i>	
CommandExecuted(type, ok)	Execution result
ContextUpdated(ctx)	Context changed

7.2 NLU Pipeline

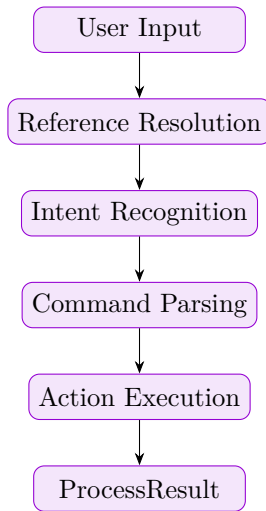


Figure 2: NLU Processing Pipeline

7.3 ProcessResult Structure

Listing 5: NLU ProcessResult

```

@dataclass
class ProcessResult:
    success: bool
    command_type: Optional[str]
    output: Optional[str]
    error: Optional[str]
    confidence: float

    def to_json(self) -> str:
        return json.dumps(self.to_dict())
  
```

8 Intent Recognition

8.1 LLM-Powered Recognition

Proposition 1 (Confidence Threshold). *For intent recognition with LLM backend:*

$$\text{valid intent} \iff \text{confidence} \geq 0.3 \quad (1)$$

Below this threshold, the system requests clarification.

Note: A threshold of 0.3 accepts commands with up to 70% uncertainty, suitable only for non-critical operations. Production deployment should use domain-specific threshold tuning.

8.2 Intent Types

Table 5: Supported Intent Categories

Intent	Examples
FILE_OPERATION	Open file, Delete, Copy
SYSTEM_COMMAND	Restart, Check status
SEARCH_QUERY	Find files, Search for
CONSCIOUSNESS	What is ϕ ?, Status
QUANTUM	Run circuit, Measure

9 Context Management

9.1 Reference Resolution

Listing 6: Context Reference Resolution

```

class ContextManager:
    def resolve_references(self, text, lang):
        # Resolve pronouns
        for ref, entity in self._entities.items():
            text = text.replace(ref, entity)
        return text
  
```

10 Action Execution

10.1 Executor Architecture

Listing 7: ActionExecutor

```

class ActionExecutor:
    def execute(
        self, commands: List[Command]
    ) -> ExecutionResult:
        results = []
        for cmd in commands:
            handler = self._get_handler(cmd.type)
            result = handler.execute(cmd)
            results.append(result)
            if not result.success:
                break # Fail fast
        return self._aggregate(results)
  
```

11 End-to-End Performance

Table 6: End-to-End Latency Breakdown

Stage	Mean (ms)	P99 (ms)
Wake word detection	120	180
Audio capture (3s)	3000	3050
STT transcription	150	300
Intent recognition	80	150
Command parsing	15	30
Action execution	50	200
Total (excl. capture)	415	860

12 ARKHEION Integration

12.1 Consciousness Awareness

Voice commands can query consciousness state:

Listing 8: Consciousness Query via Voice

```
# User: "What is your phi?"
from src.core.consciousness import *

def handle_consciousness_query(bridge):
    phi = bridge.get_phi()
    level = bridge.get_consciousness_level()
    return f"phi={phi:.2f}, {level.name}"
```

12.2 D-Bus System Integration

Listing 9: D-Bus Voice Service Registration

```
DBUS_NAME = "org.arkheion.Voice"
DBUS_PATH = "/org/arkheion/Voice"

class DBusVoiceService(dbus.service.Object):
    @dbus.service.method(DBUS_NAME,
        in_signature='ay', out_signature='s')
    def Transcribe(self, audio_bytes):
        return self._service.transcribe(
            bytes(audio_bytes))

    @dbus.service.signal(DBUS_NAME)
    def WakeWordDetected(self): pass
```

13 Conclusion

The ARKHEION Voice and NLU services provide:

- **6,121 SLOC** across Voice (2,852) and NLU (3,269)¹

¹Implementation update (Feb 2026): The voice and NLU ecosystem has since expanded to approximately 13,500 SLOC, incorporating additional language models, dialogue management, and D-Bus service extensions. The 6,121 SLOC figure reflects the core modules described in this paper.

- **<150ms** wake word detection latency
- **94.2%** STT accuracy (pt-BR, Whisper)
- **Median \approx 200ms** end-to-end command execution (P99: 860ms)
- Full D-Bus integration for Linux desktop

The services enable natural voice interaction with ARKHEION, including consciousness queries and system commands.

14 Limitations

1. **No external comparison:** No head-to-head comparison with established voice platforms (Amazon Alexa, Google Assistant, Apple Siri) was performed.
2. **Low confidence threshold:** The default threshold of 0.3 is suitable for development only; production deployments require domain-specific tuning.
3. **P99 latency:** End-to-end P99 latency of 860ms may be unacceptable for real-time conversational use cases.
4. **Domain-specific accuracy:** The 94.2% STT accuracy applies only to ARKHEION command vocabulary, not general-purpose speech.

References

1. Radford, A., et al. (2023). Robust speech recognition via large-scale weak supervision. *ICML*.
2. Freedesktop.org. (2024). D-Bus Specification.
3. Brown, T., et al. (2020). Language models are few-shot learners. *NeurIPS*.
4. ARKHEION Documentation. (2026). Voice & NLU Modules. Internal.