

Personal Infrastructure Projects

A family of related projects for speech-to-text, video transcription, voice interfaces, and personal knowledge management.

video-transcriber

****Status****: Published on PyPI

****Repo****: <https://github.com/romilly/video-transcriber>

A Python library that extracts visually distinct frames from videos and transcribes audio using Whisper. Designed for turning presentation recordings into portable markdown documentation.

Features

- Smart slide detection using perceptual hashing (captures only distinct frames)
- Audio transcription via Whisper (runs locally)
- Timeline merging - associates transcribed audio with corresponding slides
- Portable output - zip file containing markdown and images

Output Structure

```

\`
my-presentation_transcript.zip
├─ transcript.md
├─ img/
│   ├─ frame_000.png
│   ├─ frame_001.png
│   └─ ...
\`

```

Options

- Model size: tiny, base, small, medium, large-v3
- Sample interval for slide detection
- Timestamps in output
- Audio-only mode for podcasts/interviews

video-transcriber-gui

****Status****: Published on PyPI

****Repo****: <https://github.com/romilly/video-transcriber-gui>

A Textual-based TUI wrapper for video-transcriber. Provides a file browser for selecting videos and a simple interface for transcription.

the-great-dictator

****Status****: In development

****Host****: polwarth (Intel Linux box with GPU)

A browser-based dictation app for family use. Users speak into their browser; audio is transcribed server-side and saved per-user.

Architecture

- FastAPI backend with SQLite for user accounts and saved dictations
- Faster-Whisper with GPU acceleration for STT
- Cloudflare tunnel for HTTPS access from anywhere
- Cloudflare Access for authentication (free for up to 50 users)

Key Design Decisions

- Everything runs on polwarth - no need to split across multiple machines
- Cloudflare Access provides the login wall; app reads ``Cf-Access-Authenticated-User-Email`` header
- Dictations scoped by authenticated email address

watcher-on-the-Pi

****Status**:** Planned

A Raspberry Pi-based system to watch and transcribe free-to-air television broadcasts. Completely legitimate - just receiving public broadcasts and taking notes.

Concept

- DVB-T tuner receives Freeview broadcasts
- Audio stream piped through Whisper/Moonshine for continuous transcription
- Transcripts stored and indexed for search and summarisation
- No authentication, no APIs, no terms of service concerns

Potential Capabilities

- "What's in the news?" - summarise recent BBC News transcripts
- Search broadcasts for specific topics
- Alert on keywords or topics of interest
- Feed transcripts into knowledge base

Hardware Options

- USB DVB-T dongle (cheap, widely available)
- Pi 5 + AI HAT+ 2 for on-device STT
- Or route audio to polwarth for GPU-accelerated transcription

Open Questions

- Store video recordings or just transcripts?
- Which channels to monitor (BBC News, Parliament, etc.)?
- Storage and retention policy

gmail_reader

****Status**:** In use

****Repo**:** https://github.com/romilly/gmail_reader

A Python library for reading Gmail messages via the Gmail API, converting HTML content to Markdown, and storing messages in PostgreSQL.

Features

- Gmail API integration with OAuth2 authentication

```

106 - HTML to Markdown conversion with smart cleaning (removes tracking images, boilerplate)
107 - Email address extraction from headers
108 - PostgreSQL storage with markdown column
109 - Idempotent sync - fetches incrementally, safe to interrupt and resume
110 - Batch export to Markdown files
111 - Hexagonal architecture for testability
112
113 ### Key Capabilities
114 ```python
115 # Sync messages from a specific sender
116 result = sync_messages(
117     reader=reader,
118     repository=repository,
119     sender='newsletter@example.com',
120     after_date='2024/01/01',
121     verbose=True
122 )
123 ```
124
125 ### Architecture
126 - **Ports**: `MessageReaderPort`, `MessageRepositoryPort`
127 - **Adapters**: `GmailMessageReader` (Gmail API), `PostgresMessageRepository` (PostgreSQL)
128
129 ---
130
131 ## John Dee
132
133 **Status**: In use
134 **Repo**: https://github.com/romilly/john-dee
135
136 *Named after the Elizabethan polymath, mathematician, and keeper of the largest private library in England*
137
138 A FastAPI web application providing browser-based search access to personal knowledge stores, starting with Logseq.
139
140 ### Features
141 - Web UI at `/search` for browser-based searching
142 - Three search modes: full-text (FTS), semantic (embeddings), hybrid
143 - JSON API for programmatic access
144 - Hexagonal architecture
145
146 ### Stack
147 - PostgreSQL with pgvector for document storage
148 - Ollama with nomic-embed-text for semantic search
149 - logseq-searcher library for Logseq integration
150
151 ### API
152 ```
153 GET /search?q=query&type=hybrid # Web search
154 POST /api/search # JSON API
155 GET /api/sources # List sources
156 ```
157
158 ---

```

```

159
160 ## arXiv Librarian
161
162 **Status**: In production (runs on s2ag Pi)
163 **Repo**: https://github.com/romilly/arxiv-librarian
164
165 Monitors, analyzes, and summarizes research papers from arXiv and bioRxiv,
166 tracking research trends over time.
167
168 ### Features
169 - Automated paper collection from arXiv and bioRxiv APIs
170 - AI-powered summarization using Ollama (qwen2.5:7b on polwarth's GPU)
171 - Trend analysis - tracks frequency of acronyms and terms over time
172 - Flask web interface for visualization
173 - PostgreSQL storage with full-text search on AI summaries
174 - Hexagonal architecture with comprehensive test coverage
175
176 ### Production Setup
177 - Runs on **s2ag** (Raspberry Pi) with daily cron jobs
178 - Uses **polwarth** for GPU-accelerated summarization (NVIDIA 3060)
179 - Processes up to 12,000 papers per daily run
180
181 ### bioRxiv Integration
182 - Abstract Retriever interface for testability
183 - Automatic pagination and category filtering
184 - VCR.py for fast test execution
185
186 ---
187
188 ## s2ag-corpus
189
190 **Status**: In production (runs on Pi 5)
191 **Repo**: https://github.com/romilly/s2ag-corpus
192
193 Imports the Semantic Scholar Academic Graph Corpus into PostgreSQL. Handles the
194 full dataset (220M+ papers, 2.6B+ citations) with incremental diff updates.
195
196 ### Features
197 - Downloads and imports full S2AG dataset releases
198 - Applies incremental diffs from Semantic Scholar API
199 - Supports: abstracts, authors, citations, paper-ids, papers, publication-venues,
200 tldrs
201 - Citation graph visualization (generates SVG from paper relationships)
202
203 ### Hardware Requirements
204 - ~1TB storage for downloads
205 - Large database volume (4TB NVMe recommended)
206 - Initial import takes over a day on Pi 5
207
208 ### Citation Graphs
209 Generates interactive SVG visualizations of citation networks - hover for paper
210 details, click to open in Semantic Scholar.
211
212 ---
213
214 ## s2ag-web-researcher

```

```

212 **Status**: In development
213 **Repo**: https://github.com/romilly/s2ag-web-researcher
214
215 Web interface for accessing and caching Semantic Scholar Academic Graph data. Uses ↗
216 the S2AG API with local caching.
217
218 ### Features
219 - Access to S2AG via new API
220 - Local caching of results
221 - Can create Logseq-formatted pages for papers
222
223 ---
224 ## Much: Personal AI Assistant
225
226 **Status**: Planned (long-term)
227
228 A local voice assistant designed to be "done right" - legitimate APIs, no ToS ↗
229 breaches, proper integration with personal infrastructure.
230
231 The name comes from Much the Miller's Son, one of Robin Hood's Merry Men - a ↗
232 loyal, humble companion rather than a flashy presence.
233
234 ### Core Design Principles
235 - Legitimate integrations only: Official Google APIs, paid services where necessary
236 - Local-first where practical: Speech processing and simple commands on-device
237 - Incremental value: Build something simple that works, then extend
238
239 ### Planned Capabilities
240
241 **Voice Interface**
242 - Wake-word detection (note: "Much" may have false-positive issues - consider "Hey ↗
243 Much")
244 - Speech-to-text (Moonshine for low latency, Whisper for accuracy)
245 - Text-to-speech
246 - Intent routing for command classification
247
248 **Information Access**
249 - Personal knowledge base via John Dee (Logseq, PostgreSQL with pgvector)
250 - Email via gmail_reader (Gmail API, PostgreSQL storage)
251 - Research papers via arXiv Librarian and s2ag-corpus
252 - Google Calendar queries
253 - News from watcher-on-the-Pi transcripts
254 - Stock market data (legal APIs)
255
256 **Media Control**
257 - Personal music library
258 - DAB radio (BBC Radio 3, Radio 4) via uGreen or MonkeyBoard HAT
259
260 **Context Awareness**
261 - Lazydoro integration: presence data from lidar sensor
262 - VS Code plugin: time spent on each project
263 - Claude Code CLI: AI-assisted coding time
264 - Logseq: PKM integration
265
266 ### Hardware Architecture

```

```

265 **Pi 5 + AI HAT+ 2 ($130)**
266 - Hailo-10H accelerator: 40 TOPS (INT4)
267 - 8GB dedicated onboard RAM
268 - Runs Whisper/Moonshine for local STT
269 - Runs small LLMs for intent routing
270 - Keeps simple interactions local (low latency, no API cost)
271
272 **DAB Radio HAT**
273 - uGreen DAB Board or MonkeyBoard
274 - Receives BBC broadcasts over the air
275 - Avoids BBC Sounds authentication requirements
276
277 **Pico-based Touchscreen**
278 - Dedicated display interface
279 - Lightweight, always on
280
281 ### Tiered Processing
282 1. **Local (HAT+ 2)**: Wake-word, STT, intent classification, simple commands, TTS
283 2. **Claude API**: Complex queries, knowledge base reasoning
284
285 ### First Milestone
286 Voice control of music library and Radio 3/4 - contained scope, immediate daily value, exercises the core pipeline.
287
288 ---
289
290 ## Book Topics
291
292 Several components would make good content for "Build Great Python Projects Faster with AI":
293 - Wake-word detection and speech processing
294 - Browser-based audio capture and transcription
295 - Video frame extraction with perceptual hashing
296 - Gmail API integration with OAuth2
297 - HTML to Markdown conversion and cleaning
298 - Semantic search with pgvector and Ollama embeddings
299 - Hexagonal architecture for testability
300 - Working with research paper APIs (arXiv, bioRxiv, Semantic Scholar)
301 - AI-powered summarization with Ollama
302 - Large dataset ingestion and incremental updates
303 - Citation graph visualization
304 - Google Calendar/Email API integration
305 - Working with the AI HAT+ 2
306 - Building Pico-based interfaces
307 - Presence detection with lidar
308 - Local LLM inference on constrained hardware
309 - DAB radio reception on Raspberry Pi
310 - Building a personal news wire from broadcast transcription
311
312 ---
313
314 ## Related Tools
315
316 - **Moonshine**: Small-footprint STT (27-62M params), 5-15x faster than Whisper, good for edge devices
317 - **Faster-Whisper**: GPU-accelerated Whisper, best accuracy, runs on polwarth
318 - **Cloudflare Tunnels**: Free HTTPS access to home services, 1000 tunnels allowed

```

on free plan

- **ollama**: Local LLM hosting, used with nomic-embed-text for semantic search in John Dee
- **pgvector**: PostgreSQL extension for vector similarity search

Infrastructure

Host	Hardware	Role
polwarth	Intel Linux, NVIDIA 3060 GPU	the-great-dictator, Faster-Whisper STT, Ollama for summarization
s2ag	Raspberry Pi	arXiv Librarian cron jobs, production database
Pi 5	Raspberry Pi 5 + 4TB NVMe	s2ag-corpus (Semantic Scholar database)
Future	Pi 5 + AI HAT+ 2	Much voice assistant, local STT/intent routing

Updated 31 January 2026