1 Meta

- Add Lucas to authors
- add early collaborators like Chris, mackenzie, etc. who gave information and insight.
- add contribution statement

1.1 Formatting

- TOC spacing!
- Smart quotes for single quotes
- Margin note positioning throughout (do at end though)

2 Intro

2.1 Foreword to Manuscript v2

- Check the changelog!

2.2 Preamble

- Add: Also experimental software is a missed opportunity and a strong lbockage to our ambitions of "open science" – making an unnecessary rite of passage in struggling through behavioral setups,
- This is actually a challenging technical and theoretical problem. Tension between totalizing framework and integrating many tools: autopilot tries to be mindful about both – it is a logical framework for the different parts of an experiment, allows easily hooking into each of these components, but also lets them be used separately. Autopilot was designed to *play nice* with other tools, but also give them a place to play nice with each other.
- "whats missing is a glue framework"
- **Reproducibility para** - Mention of the wiki and plugin systems – using a linux system brings with it all a manner of technical challenges. We have designed autopilot as a system that can integrate fluid technical knowledge along with the code to use it. Also that we have designed it to let work at all stages of "completeness" work together with its plugin framework – example with a stepper motor. thsi one might not be **exactly** what you need but it's better than nothing.
- Note that autopilot is a continually developing system, so we don't necessarily do everything here perfectly, many things are in progress, but this document is intended to capture the ideas that animate its development.

2.3 Existing Systems

- "These are the minimum components for a complete behavior system, but there are of course many other things, means of conceptualizing and supporting. Each of these is far from monolithic and has within it a thousand design choices."
- add transforms to "data management"
- Update description for pyControl
- update description for BPod
- link to finite state machine section from "pyControl uses a finite state machine"
- Bpod also has pyBpod and whatever else
- remove "Bpod blocks operation" it's not necessary – refocus this section on just describing the architectural decisions without including shit talking the GUI

2.4 Limitations of existing systems

- Documentation is a big problem – don't have low-level docs so people can inspect what's going on. Researchers are expected to interact with it on a top-level only: the tool as is, rather than expand and hack on the library itself. Flexible tools need to have interfaces at multiple levels: yes, just use, but also be able to extend, and modify the existing system. This is an architectural, interface, and documentation question and it's a hard one to solve!
- **New Item: Community Tooling** - No facility for sharing tasks, hardware development, etc. No good means of sharing practical usage advice – basically the community tooling isn't there. Look for examples of Bpod tasks (eg. the IBL task) and how that doesn't make for a cumulative body of knowledge. Use the christof koch quote about organization being the major problem: in order to be a force multiplier, need to be able to have peer-to-peer knowledge sharing instead of treating a piece of software as a fixed entity.
- Data - no schemas, no formal structures, sharing is hard and both take the approach that you need to write a whole additional library to
- Devops and code tooling – Bpod stuff has no means of integrating other code because matlab doesn't have a package manager. No CI, no tests for either
- the bottom line is that these tools weren't designed with integration in mind, ithey might handle some subset of the experiment, but they don't really play well with the other parts of the experiment and ultimately serve as yet another complicated node rather than something that can be compared.

3 Design

- We take a broad lens on reproducibility – not just good data, but a tool that integrates into a broader ecosystem of tools and reduces labor duplication beyond itself. Reproducibility is not just a question of getting everyone to upload their data, but to build the tools that make reproducibility accessible to a broader base of researchers.
- Though only the GPIO is really unique to the raspi – autopilot can be used on any unixlike system.

3.1 Efficiency

- integrated circuit -> microcontroller
- efficiency of interpreted languages is not the only consideration in efficiency: how easy it is to use and the time spent troubleshooting and making do with other systems is itself part of the equation, so we try and balance the ease of use witht he performance of the program
- **Concurrency** - Arguably the most radical concurrency model in autopilot is the use of multiple comptuers. So if a single processor can't do what you want it's trivial to have other computers running other things. This also lets different system components be isolated from one another, something that blends into the notion of agents and behavioral topologies discussed later.
- **Low-Level** - We make use of Python to mature an API that we will then drop down into lower level languages like C and Rust when they are stable.

3.2 Flexibility

- **Modularity** - There is a tension between flexibiltiy and providing a complete behavioral package, to balance that we have to be very capable of keeping our code modular. We do that by implementing a sensible inheritance hierarchy with predictable and shared APIs for different objects. That way we can keep a predictable set of methods exposed while being able to modify the other
- **New Section: Plugins** - replace structured expansino/code transparency section with plugin sectino. By separating local configuration files and etc. we are able to have a very flexible plugin system: users are not expected to modify files in the repository to use it (which makes it difficult to contribute later) or implement everything in a single script. instead we provide a system for expansion that scaffolds their use of the software and reuse in tasks.
- CI is also really relevant here – to make sure that it's possible to change the system freely, we have CI and as we have been developing have been

3.3 Reproducibility

- New to data management section: schema stuff! We want to provide a well annotated data structure that can be exported to many different formats. To do that we have built a bunch of data modeling tools
- Also working on a related ingest project to be able to automatically bring in data from other sources made during use.

4 Structure

5 Tests