Tyelab Currently Existing Bruker-2P Technical Docs can be found here: Repository can be found here: User Guide can be found here: Program Execution + Some Ideas Start of Day's Imaging Requires: 2022 https://bruker-control.readthedocs.io/en/latest/about/in https://github.com/Tyelab/bruker_control https://bruker-control.readthedocs.io/ 1. Load Arduino Sketch dex.html en/latest/instructions/index.html Solid boxes mean Open Prairie View implemented. Local Directory Structure Server Directory Structure Use bruker_control CLI Local drive is a 1.8TB SSD dedicated to collecting raw data Note: This is what's Austin's project has currently Dashed boxes mean yet to be implemented or awaiting improvement L— teamname_projectname teamname_projectname config config.json ├── subjects
└── SUBJECT_ID bruker_control CLI **Prairie View** Arduino App --- microscopy ---- subject_id.json 2P-Microscope recording session Python Would be best if user didn't have to upload the - .ENV for microscope settings ---- 2p sketch themselves, instead did it via Arduino CLI – confia -.XML for image metadata subprocess. See Issue #71 Prairie View controls microscope through GUI. -.RAW binaries for images processed See here for "flight Have list of subjects https://qithub.com/Tyelab/bruker control/issues/ raw L.RAW binaries for behavior manifest" idea: L-SUBJECT_ID for imaging ready or --- video Move scope Z-axis through software or Micromanipulator to https://github.com/Tyelab L— DATE .mp4 via H264 encoding create one via GUI /bruker_control/issues/75 plane experimenter wants to image. config.json — yoked microscopy control - video.mp4 L.json of trial_arrays Ensure settings for scope are properly in place Upload team's Arduino Sketch zstacks ---- experimental "Flight Manifest" L .json of trial_arrays Number of images to collect set very high (>50k) Determination -zstacks Only done once at the start of This can probably be set in the Prairie recording_session_zstack# day's imaging .ENV for microscope settings View API, just not sure what PVShard to XML for image metadata reference... L.RAW binaries for images **Grab Project** Correct Voltage Recording Selected Behavior Config, Start with Input Trigger Subject Metadata Subject folder is where subject metadata is held. Config folder is where behavior configurations are held. https://bruker-control.readthedocs.io/en/latest/configurations/configs.html#example-Metadata & Configurations https://bruker-control.readthedocs.io/en/latest/configu rations/configs.html#example-project-config-behavior Additional metadata for imaging plane parameters could look something simple Loaded like this in .json, .yml, etc (includes multiple lasers as we have them at NLW and new Bruker will have it as well). Info could be found by parsing out numerous .ENV Move to Imaging Potential Re-Design in Class-Based Code and .XML files from Bruker, but accessing them this way would be simpler to use as Plane well as easier to read/maintain instead of having to search through Bruker's XML. https://github.com/Tvelab/bruker_control/blob/class_i n session/configs/project config.json "plane_1": See Pydantic datamodels validation for Class-Based location: -300.00, # Z-axis position, Connect to pockels_1: 500, # First laser power, used for one calcium indicator https://github.com/Tyelab/bruker_control/blob/class_i pockels_2: 600, # Second laser power, used for second calcium indicator pmts_1: 750, # Channel 1 PMTs, used to record from first laser Microscope API n session/main/datamodels.py pmts_2: 650, # Channel 2 PMTs, used to record from second laser Naming Conventions, File Formats, Etc "plane_2" Images typically averaged tiffs from a time series of microscopy data Much of this documentation resides on Confluence Can easily display with skimage/opencv at the moment, but it should be migrated to RTD. alongside video previews The essence is: See Issue #49 for reference plane **Present Selected** Configure https://github.com/Tvelab/bruker_control/issue YYYYMMDD_subjectid_plane#_planeposition Imaging Plane Camera(s) for "plane_n Reference (From Explorer Update would be structured way of Recording types, directories, etc have their docs Preview Exploration) finding imaging planes. See basic idea at Issue on Confluence and will be migrated to RTD or Currently only one https://aithub.com/Tvelab/bruker_control/issue camera is present, <u>s/79</u> Adopting this kind of metadata file would allow the program to know how many adding a second planes it needs to take, which laser/PMT settings to uses on an animal by animal can be done with basis or even plane by plane basis if necessary. Could also automatically use minimal refactoring The Bruker scope, unlike the NLW scope, has no functionality microscope's API to move down to different planes. Something like this could be for detecting if your current plane matches your reference Confirm Imaging image, so it must be done by eye. **Present Camera** next_plane_position = imaging_plane_metadata[n] We can access frames through their API and Plane via Neural Preview(s) perform recognition via inference through a Jetson pl.SendScriptCommands("-SetMotorPosition '-Z' '{}').format(next_plane_position) Network Nano or GPU if we upgrade the currently in place computer with one. We have a GPU (GTX1050) that may be suitable for this purpose lying around in lab. For making sure current plane is the same as reference plane, could use flattened images and do correlation value maybe... This would make it so users don't have to eyeball similarity and use Z-axis position alone Would also be great to be competent at for when the new SLM arrives and we want to do closed loop Yes stimulation from real time neural activity Could be made to automatically search for plane, as If yoked configuration in NLW, but would be much more time intensive to (meaning the same settings) exists... No Yoked Trials Yes No Write to yoked Load Yoked **Generate Trial** Write out directory if yoked, Configuration Arrays configuration .json standard config file written every time Arduino Sketch Communicate Starts the Trials to Arduino Experiment Theoretically should be expandable to many more cameras, but computational cost of encoding with H264 at runtime must be considered Placing USB hubs at separate controllers as Brian has taught us may Input trigger starts Camera is triggered by Configure be necessary. If too intensive, we could TTLs from the microscope, scope's recording of Camera(s) for use a separate machine, as the configuration is to set up microscope and Gogolla lab does, or perform little to no Recording behavior data to disk camera to receive them compression on video data at runtime. Could use campy with GPU accelerated compression to solve this problem Current camera has 300Mbs/sec data rate over ethernet Scope sends TTLs to camera Time Series Directory Video Data See note above right about yet to be vvvvmmdd subjectid plane# planeposition raw-### recorded to disk implemented pieces via H264 encoding VoltageRecording.xml VoltageRecordingRAW (<2GB) VoltageRecordingFilelist.txt MicroscopyMetadata.env MicroscopyRecording.xml This was previously in place, but later decided to be done at a later time. Would be easy to re-implement. MicroscopyRecordingFilelist.txt Write base NWB See closed Issue #58 for more here: MicroscopyRecordingRAW (30 min = 35*2GB/channel) File to Server https://github.com/Tyelab/bruker_control/issues/58 See code for utility here: https://github.com/Tyelab/bruker_pipeline/blob/main/nwb_utils. ŊΥ Requires use of specialk style subject metadata and project metadata Total # Planes Recorded No Yes Move to Configure subjects recorded Camera Preview Step Yes No Ideally have "Flight Start at Transfer data to server with Manifest" do this bruker_control CLI bruker_transfer_utility.sh part instead Approx 5hr over 1Gb ethernet (@ 29MB/sec)