Rapid Motor Characterization Using ODrive Motor Controller

Emily Goldberg

Master's Plan B Project Defense

April 30, 2021

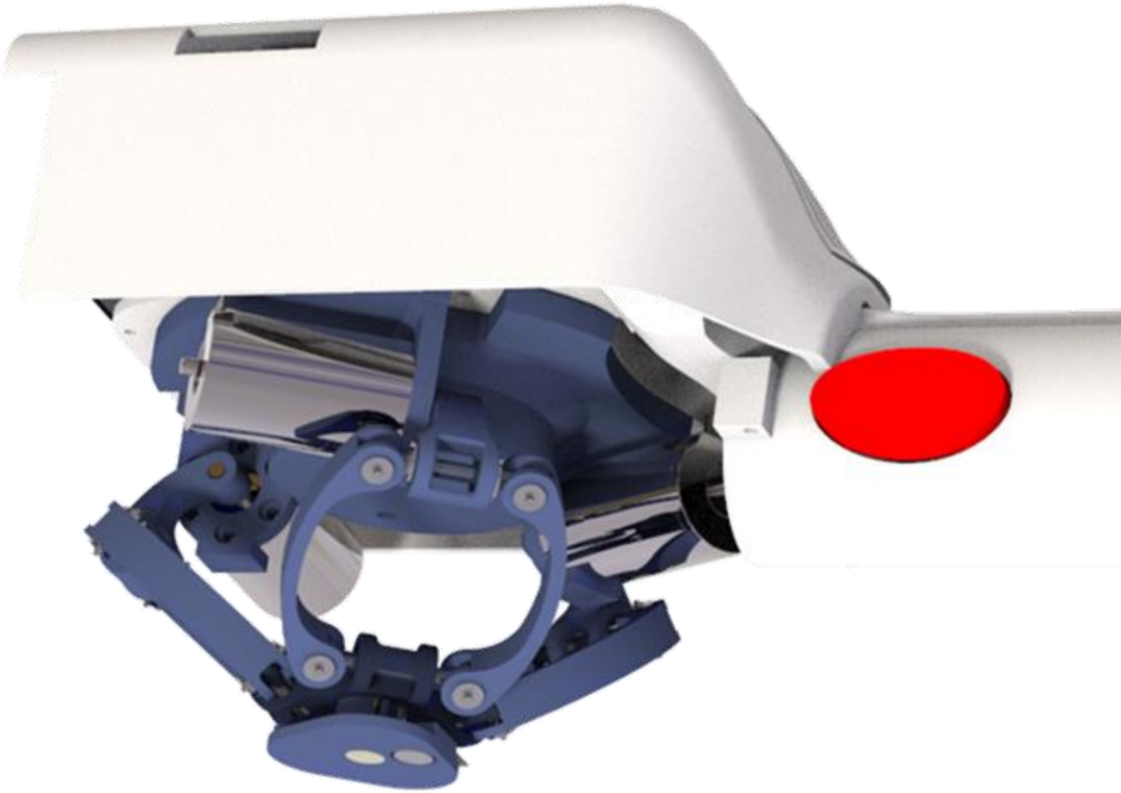
Department of Mechanical Engineering

Adviser/Committee Chair: Professor Tim Kowalewski

Committee: Professor Will Durfee, Professor Junaed Sattar

Introduction – BLDCs in robotics

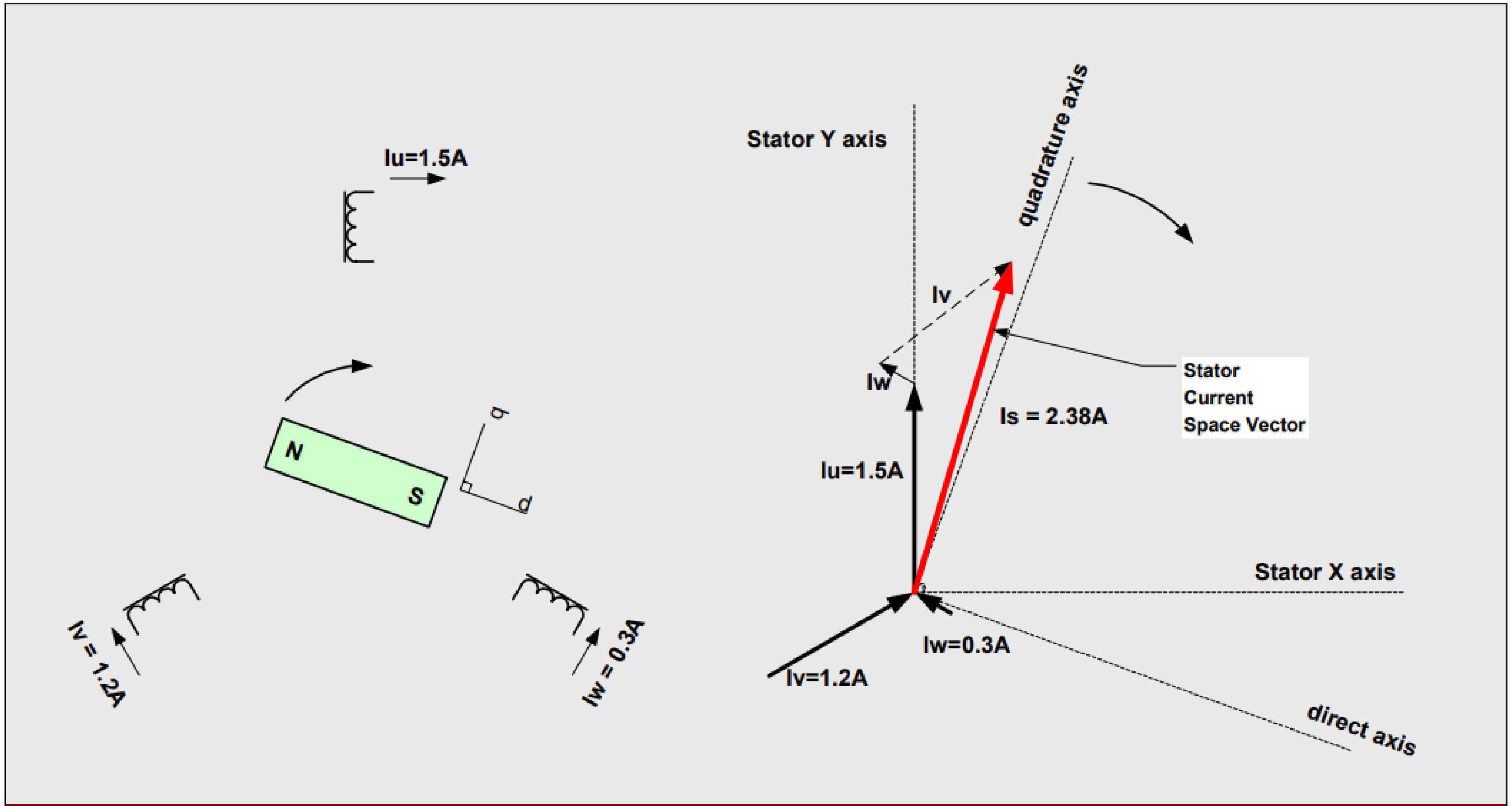
- Brushless DC motors
  - Better performance than brushed DCs
  - Less expensive than servos
  - Better high-speed performance than steppers
- But require good control



Introduction – BLDC control theory

[3]

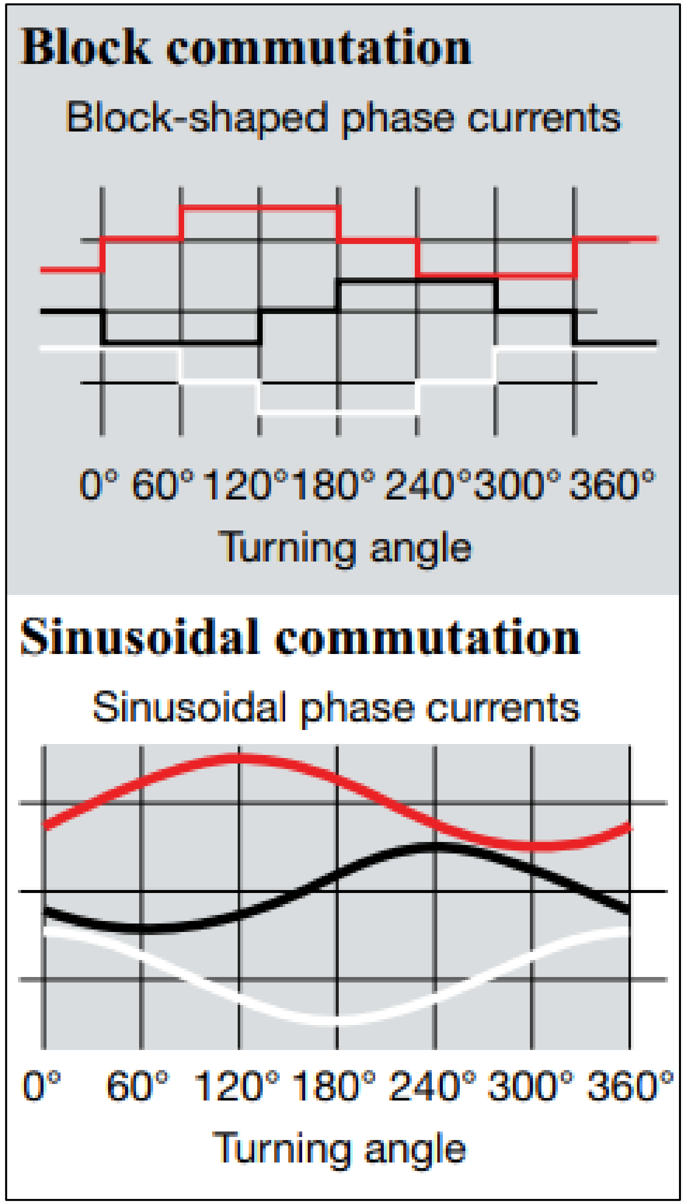
- Magnetic rotor, three windings
- Current space vector representation
    - Optimal vector is constant magnitude, perpendicular
    - -> sinusoidal winding currents



Stator current space vector is the sum of the space vectors for each winding. For maximum torque, the stator current space vector should rotate with the rotor so that it always points in the quadrature direction.

Introduction – commutation

- Trapezoidal/block commutation
  - Simple, efficient
  - High torque ripple, bad at low speeds
- Sinusoidal commutation
  - Very smooth at low speeds
  - Breaks down at high speeds
- Field-oriented control (FOC)
  - Like sinusoidal, but in d-q frame
  - Good for both low and high speeds

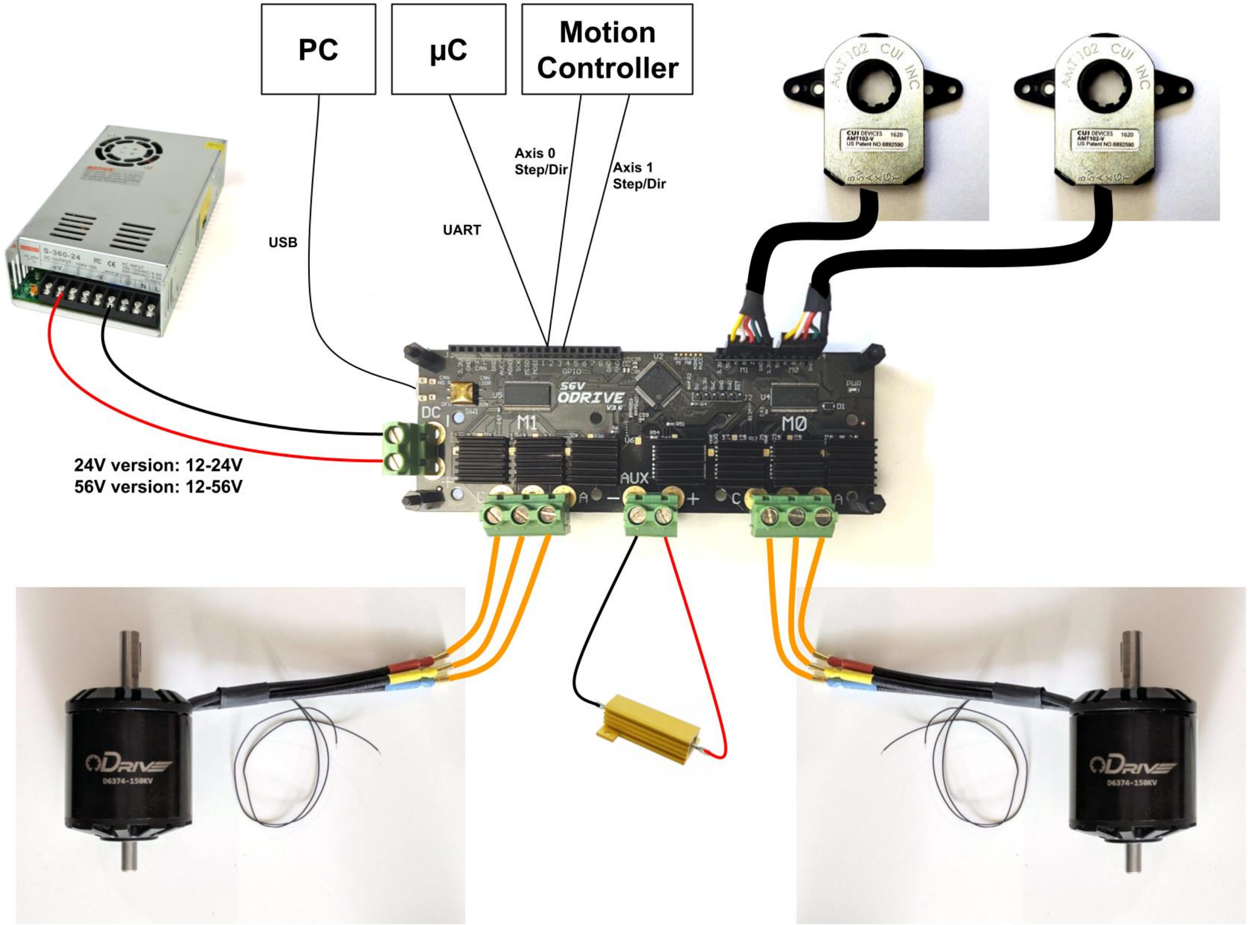


[2]

Introduction – challenges

[5]

- Challenges in BLDC use
  - Brand differences
  - Tuning – hard without characterization
- ODrive
  - Generalizable motor controller
  - Ignores brand differences
  - Still requires manual tuning
  - Open-source firmware



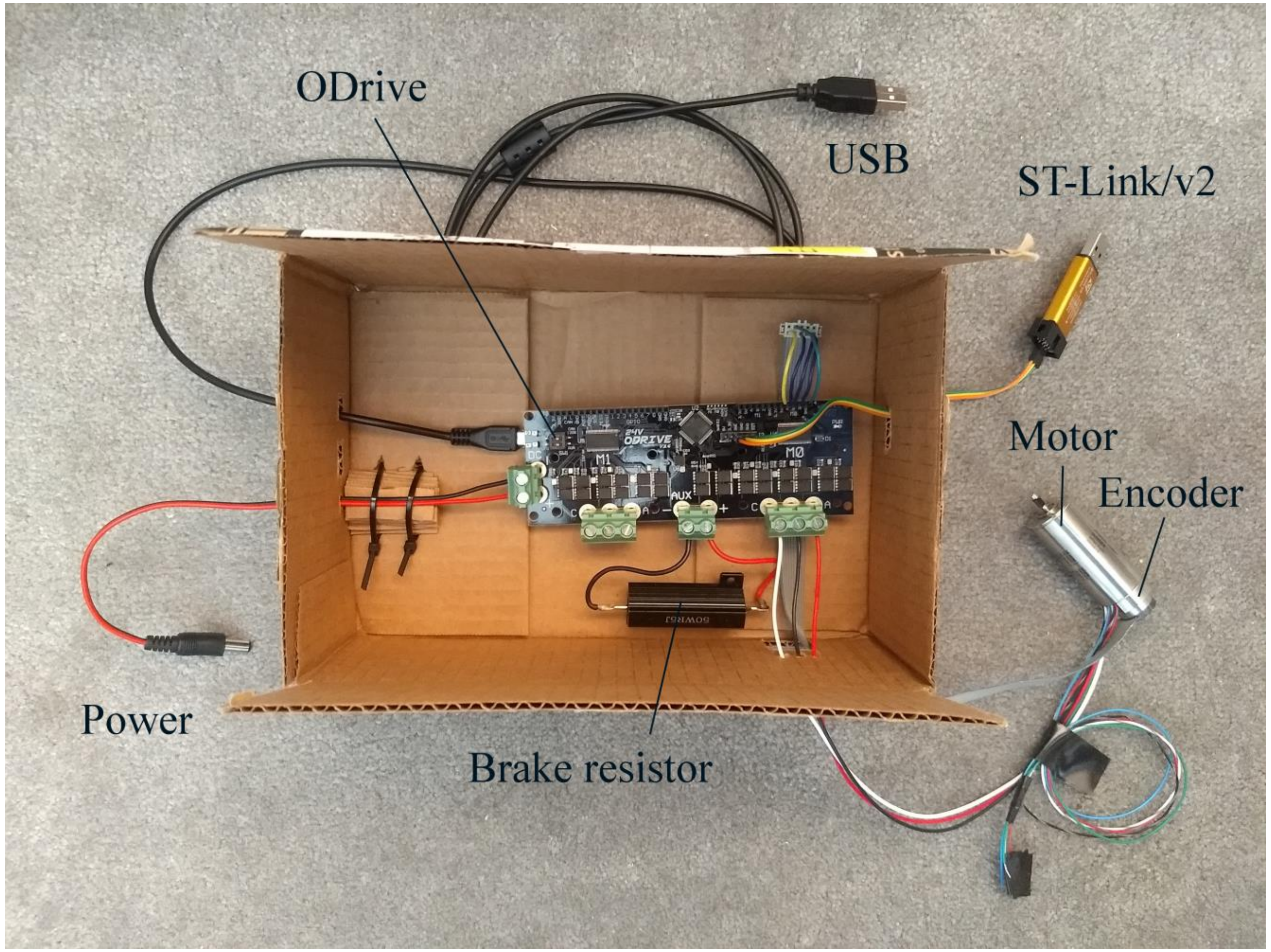
Introduction – project goals

Objective: add easy motor characterization to ODrive

1. High-resolution real-time recording of motor data (applied electrical input, motion outputs) with bounded or known sample time jitter
2. Ability to send known or desired test signals to characterize the motor (step, impulse, chirp, and noise) in the same time sample as the recorded parameters

Materials

- ODrive
  - Hardware v3.6-24V
  - Firmware v0.4.11 → v0.5.1 (C/C++)
  - odrivetool (Python)
- Motor – maxon 323128
  - 125 g
  - 45.1 mNm nominal torque
  - 90 W assigned power rating
- Encoder – maxon 575828



Methods – original ODrive firmware

- main.cpp creates two sets of:
  - Axis
  - Encoder
  - Motor
  - Controller
  - Configurations for each one
- Then each axis runs a state machine
  - run_control_loop() guaranteed at 8kHz
  - User-set requested_state

```cpp
template<typename T>
void run_control_loop(const T& update_handler) {
    while (requested_state_ == AXIS_STATE_UNDEFINED) {
        // look for errors at axis level and also all subcomponents
        bool checks_ok = do_checks();
        // Update all estimators
        // Note: updates run even if checks fail
        bool updates_ok = do_updates();

        // make sure the watchdog is being fed.
        bool watchdog_ok = watchdog_check();

        if (!checks_ok || !updates_ok || !watchdog_ok) {
            // It's not useful to quit idle since that is the safe action
            // Also leaving idle would rearm the motors
            if (current_state_ != AXIS_STATE_IDLE)
                break;
        }

        // Run main loop function, defer quitting for after wait
        // TODO: change arming logic to arm after waiting
        bool main_continue = update_handler();

        // Check we meet deadlines after queueing
        ++loop_counter_;
```

Methods – original ODrive firmware

- Time
  - loop_counter_ [#] (updates at 8kHz)
- Encoder readings
  - Position, velocity [encoder counts, counts/s in v4, turns and turns/s in v5]
  - Phase = position in electrical radians
  - Phase velocity calculated for motor updates (different in v4 and v5)
- Motor commands (in voltage control mode)
  - motor.update()
  - Translates command to FOC voltages, sends to low-level architecture

Methods – original ODrive firmware

- Motor settings “high-current” and “gimbal”
- Control options
  - v4: trajectory, position, velocity, current
  - v5: position, velocity, torque
- Voltage control mode technically exists, but is unused
  - Can simulate it with:
    - v4: gimbal motor mode + current control mode
    - v5: gimbal motor mode + torque control mode + torque constant set to 1
  - Caution: no built-in safety checks

Methods – original ODrive firmware

- USB accessibility requires a communication protocol
  - v4: make_protocol_xxx() methods
  - v5: YAML file

```
static inline auto make_obj_tree() {
    return make_protocol_member_list(
        make_protocol_ro_property("vbus_voltage", &vbus_voltage),
        make_protocol_ro_property("serial_number", &serial_number),
        make_protocol_ro_property("hw_version_major", &hw_version_major),
        make_protocol_ro_property("hw_version_minor", &hw_version_minor),
        make_protocol_ro_property("hw_version_variant", &hw_version_variant),
        make_protocol_ro_property("fw_version_major", &fw_version_major),
        make_protocol_ro_property("fw_version_minor", &fw_version_minor),
        make_protocol_ro_property("fw_version_revision", &fw_version_revision),
```

→

```
serial_number: readonly uint64
hw_version_major: readonly uint8
hw_version_minor: readonly uint8
hw_version_variant: readonly uint8
fw_version_major: readonly uint8
fw_version_minor: readonly uint8
fw_version_revision: readonly uint8
```

Methods – additions needed

- The ODrive does have:
  - Good control architecture
  - Voltage control
  - Communication protocols for USB interface
- The ODrive does not have:
  - Voltage test inputs
  - Data recording for anything other than bus voltage
  - Continuous data export (v5 has some, but not ideal for this)

Methods – voltage test inputs

- New axis state AXIS_STATE_MOTOR_CHARACTERIZE_INPUT
  - Checks motor mode, control mode, torque constant (v5 only)
  - Calls run_motor_characterize_input()
- run_motor_characterize_input()
  - Waits for delay time
  - Runs test input
  - Records data on each timestep
- New axis configuration struct input_config_ (type InputConfig_t)
  - Saves and reboots like other configurations

Methods – voltage test inputs

- Calls run_control_loop() with one of four handlers
- Each has format:
  1. Calculate voltage command at current time; cap if necessary
  2. Get latest encoder estimates; derive phase velocity
  3. Call motor update with desired voltage, observed phase/phase velocity
  4. Record data
  5. Repeat until test_duration time has passed

Methods – voltage test inputs

Input Type	Voltage command as a function of time (all parameters from `input_config_`)
Step	$V(t) = step\_voltage$
Impulse	$V(t) = \begin{cases} impulse\_voltage\ if\ [steps\ elapsed] < impulse\_peakDuration \\ 0\ otherwise \end{cases}$
Noise	$V(t) \in [-n, n]$ Where $n = \left(\frac{noise\_max}{100} * voltage\_lim\right)$ for $noise\_max\ \mathbb{Z} \in [1\ 100]$
Exponential chirp	$V(t) = chirp\_amplitude * \sin(phase) + chirp\_midline$ Where $phase = 2\pi * chirp\_freqLow * \left(\frac{k^{x*test\_duration} - 1}{\log(k)}\right)$ for $k = \left(\frac{chirp\_freqHigh}{chirp\_freqLow}\right)^{\frac{1}{test\_duration}}$

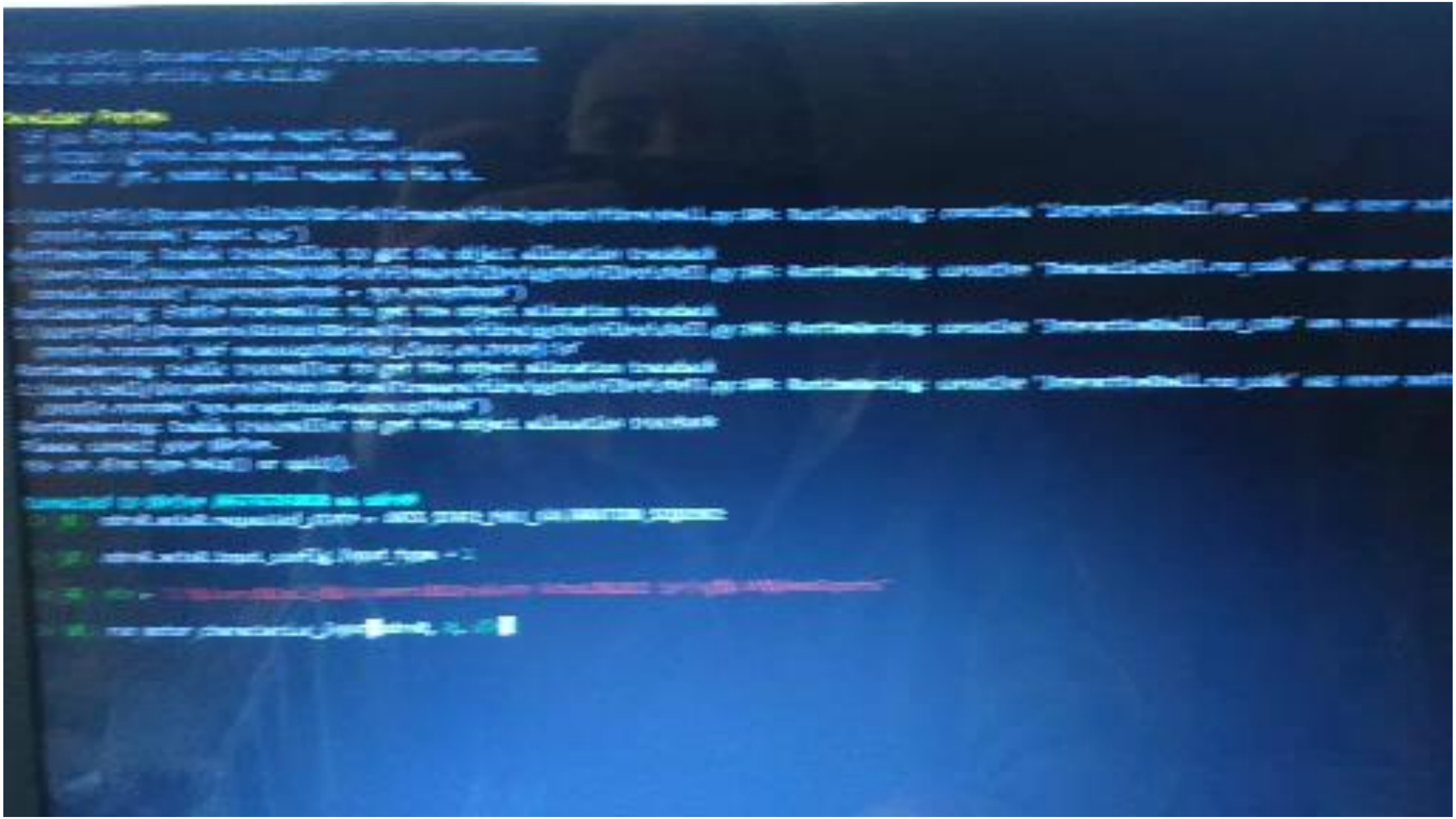
University of Minnesota
Driven to Discover®

Methods – data storage/access

- Ring buffer motor_characterize_data
  - 4x128
  - Timestep [s], command [V], position [counts → turns], velocity [counts/s → turns/s]
- record_motor_characterize_data() called in update handler(s)
- Buffer added to communication protocol
  - User can access by index in odrivetool
  - Latest observation is at motor_characterize_data_pos

Methods – data export

- Not enough storage space on ODrive; have to export continuously
- New odrivetool method run_motor_characterize_input()
  - Sets requested_state
  - Starts pulling data as fast as possible
  - Stops recording when the data returns to zero
  - Writes all data to CSV



Discussion – user interface

```
In [5]: odrv0.axis0.input_config
Out[5]:
input_type = 1 (int)
test_delay = 2.0 (float)
test_duration = 5.0 (float)
impulse_voltage = 2.0 (float)
impulse_peakDuration = 1 (int)
step_voltage = 0.25 (float)
chirp_amplitude = 0.25 (float)
chirp_midline = 0.0 (float)
chirp_freqLow = 1.0 (float)
chirp_freqHigh = 1000.0 (float)
noise_max = 2 (int)
```

- Launch odrivetool
- Confirm gimbal motor mode and:
  - v4: current control mode
  - v5: torque control mode, torque constant = 1
- Run motor calibration
- Edit <axis>.input_config (see report for details)
- Call run_motor_characterize_input(<odrive>, <axis>, <directory>)
  - <odrive> - ODrive object, odrv0 by default
  - <axis> - 0 or 1
  - <directory> - string, export directory location

Discussion – example commands

```
Connected to ODrive 2057357A3056 as odrv0
In [1]: odrv0.axis0.requested_state = AXIS_STATE_FULL_CALIBRATION_SEQUENCE

In [2]: odrv0.axis0.input_config.input_type = 2

In [3]: dir = "C:\\Users\\Emily\\Documents\\1Graduate School\\2021 Spring\\Lab\\demoExports"

In [4]: run_motor_characterize_input(odrv0, 0, dir)
```

ODrive object

export directory

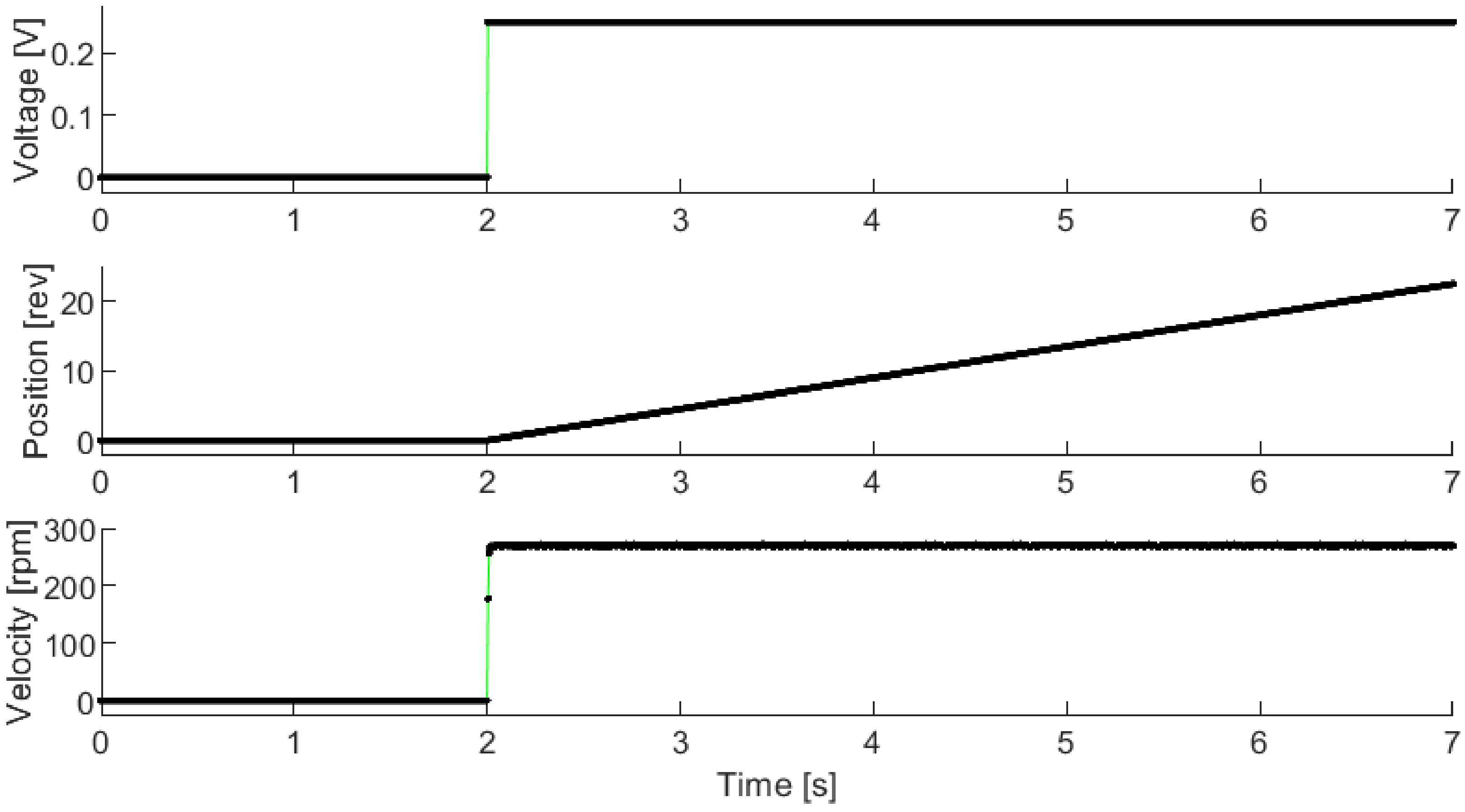
axis

Discussion – recorded data (step, impulse)

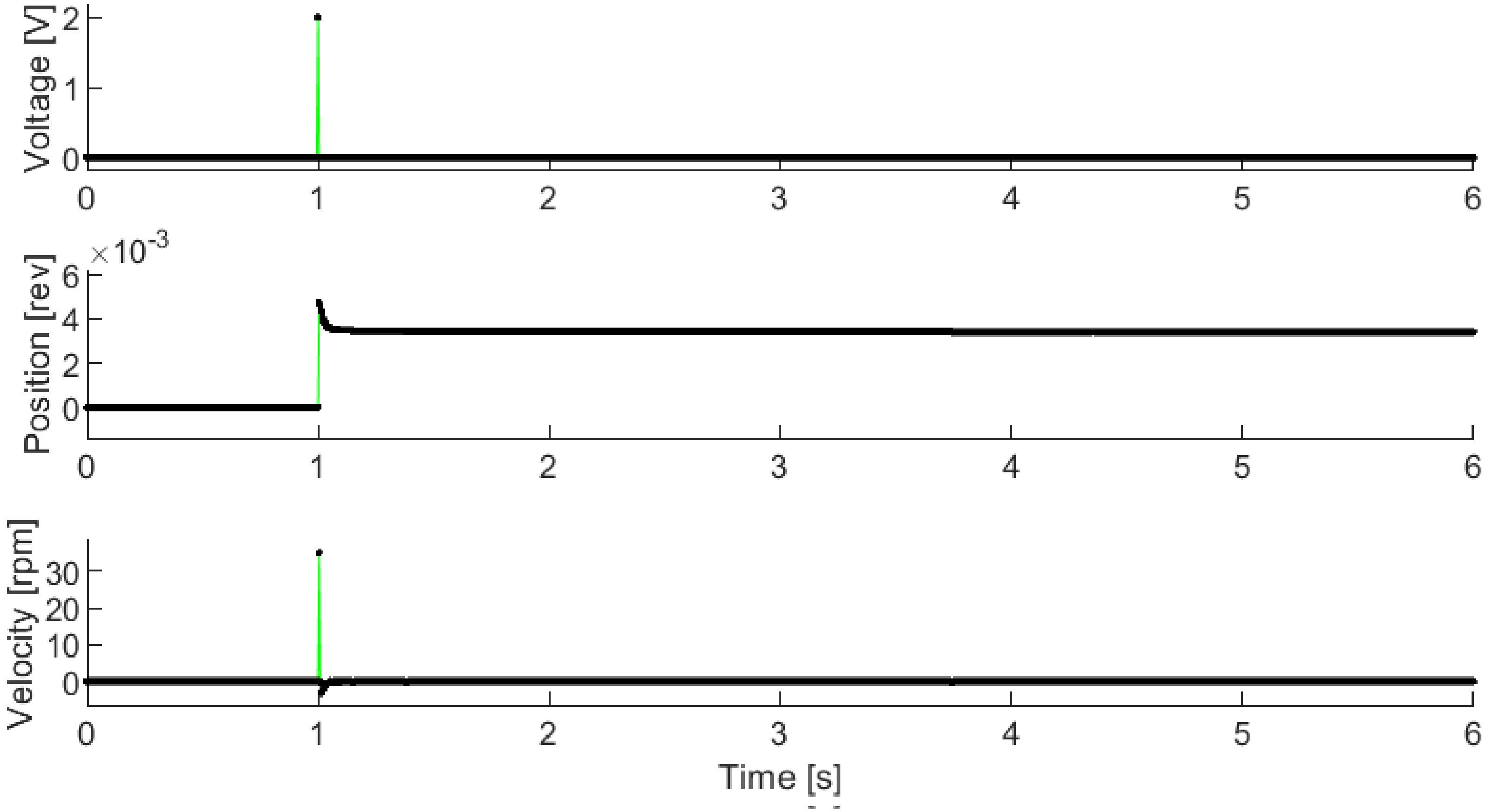
```
%Motor characterization data
%Each row's values were recorded on the same timestep
%Timestep increments at 8kHz

%Operator:
%Motor:
%ODrive axis: axis0
%Date:,29/04/2021
%Start time:,15:31:52

%timestep (8Hz),voltage,position,velocity
%[#],[V],[turns],[turns/s]
2.0,0.0,-0.16394783556461334,0.0;
57.0,0.0,-0.16394783556461334,0.0;
109.0,0.0,-0.16394783556461334,0.0;
//
15915.0,0.0,-0.16394783556461334,0.0;
15958.0,0.0,-0.16394783556461334,0.0;
16000.0,0.25,-0.16394783556461334,0.0;
16045.0,0.25,-0.14777781069278717,3.471374750137329;
16089.0,0.25,-0.12354542315006256,4.398346424102783;
16131.0,0.25,-0.0998385101556778,4.512787342071533;
```

```
%Motor characterization data
%Each row's values were recorded on the same timestep
%Timestep increments at 8kHz

%Operator:
%Motor:
%ODrive axis: axis0
%Date:,29/04/2021
%Start time:,15:34:20

%timestep (8Hz),voltage,position,velocity
%[#],[V],[turns],[turns/s]
2.0,0.0,114.09017181396484,0.0;
51.0,0.0,114.09017181396484,0.0;
98.0,0.0,114.09017181396484,0.0;
158.0,0.0,114.09017181396484,0.0;
//
15950.0,0.0,114.09017181396484,0.0;
15998.0,2.0,114.09017181396484,0.0;
16052.0,0.0,114.09513854980469,0.2899171113967895;
16104.0,0.0,114.09483337402344,-0.04196155443787575;
16158.0,0.0,114.09454345703125,-0.03814685717225075;
```

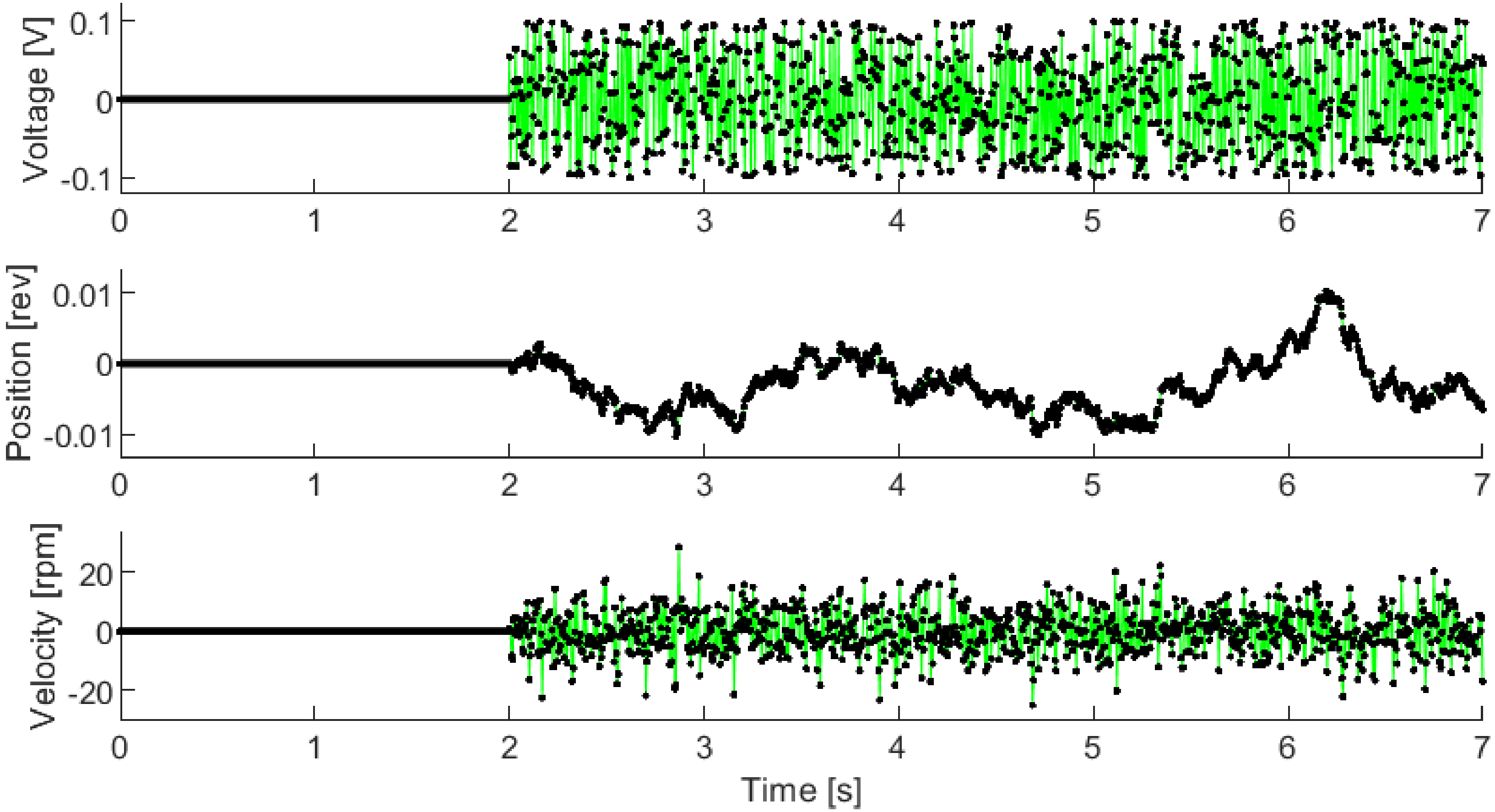
Discussion – recorded data (step)



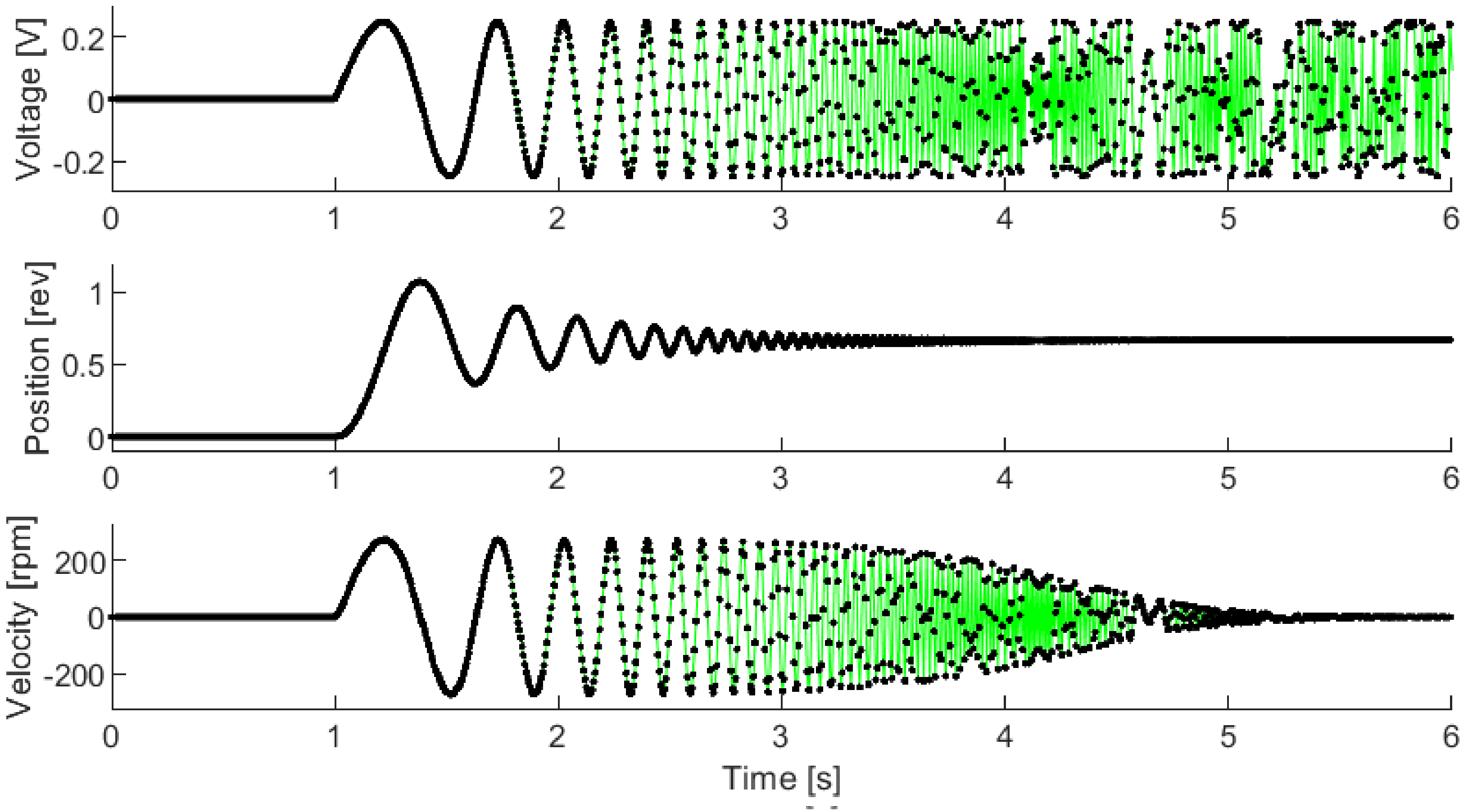
Discussion – recorded data (impulse)



Discussion – recorded data (noise)

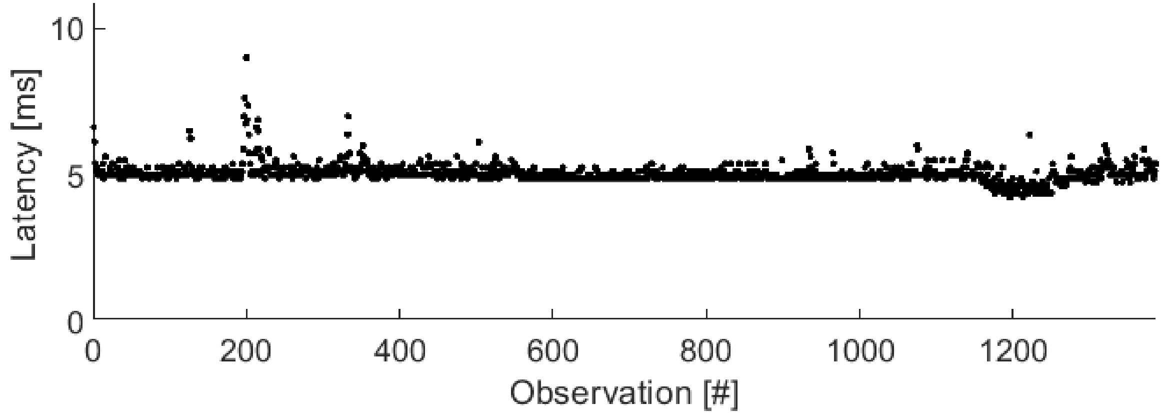


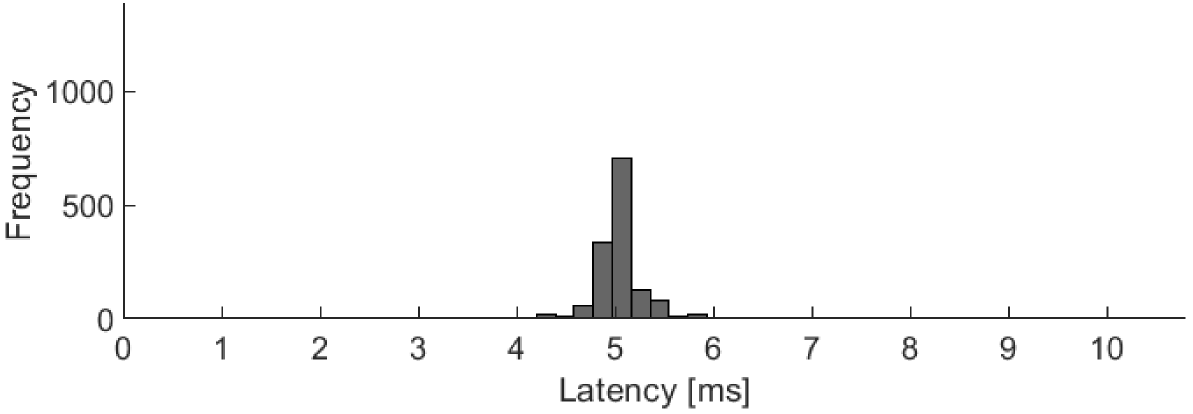
Discussion – recorded data (chirp)



Discussion – latency

- Limited to one value at a time
  - Only "latest" observation
  - Five calls for one timestep
  - Known ODrive limitation; in plans for future versions
- Optimized to get 150-200Hz





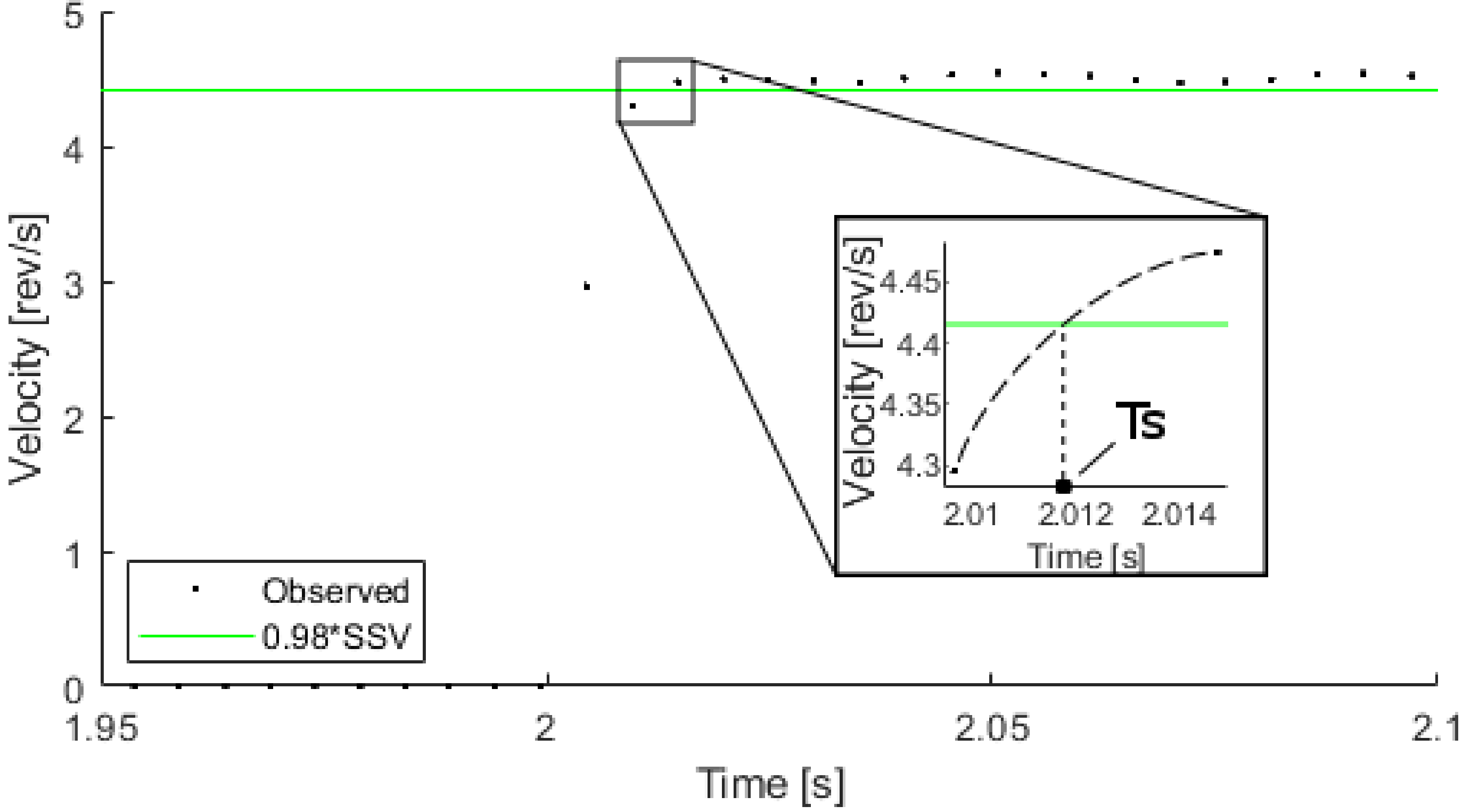
Discussion – characterization

- Settling time T_s
- Time constant

$$\frac{1}{a} = \frac{T_s}{4}$$

- Steady-state SSV
- Gain $K = SSV * a$

$$G(s) = \frac{K}{s(s+a)}$$



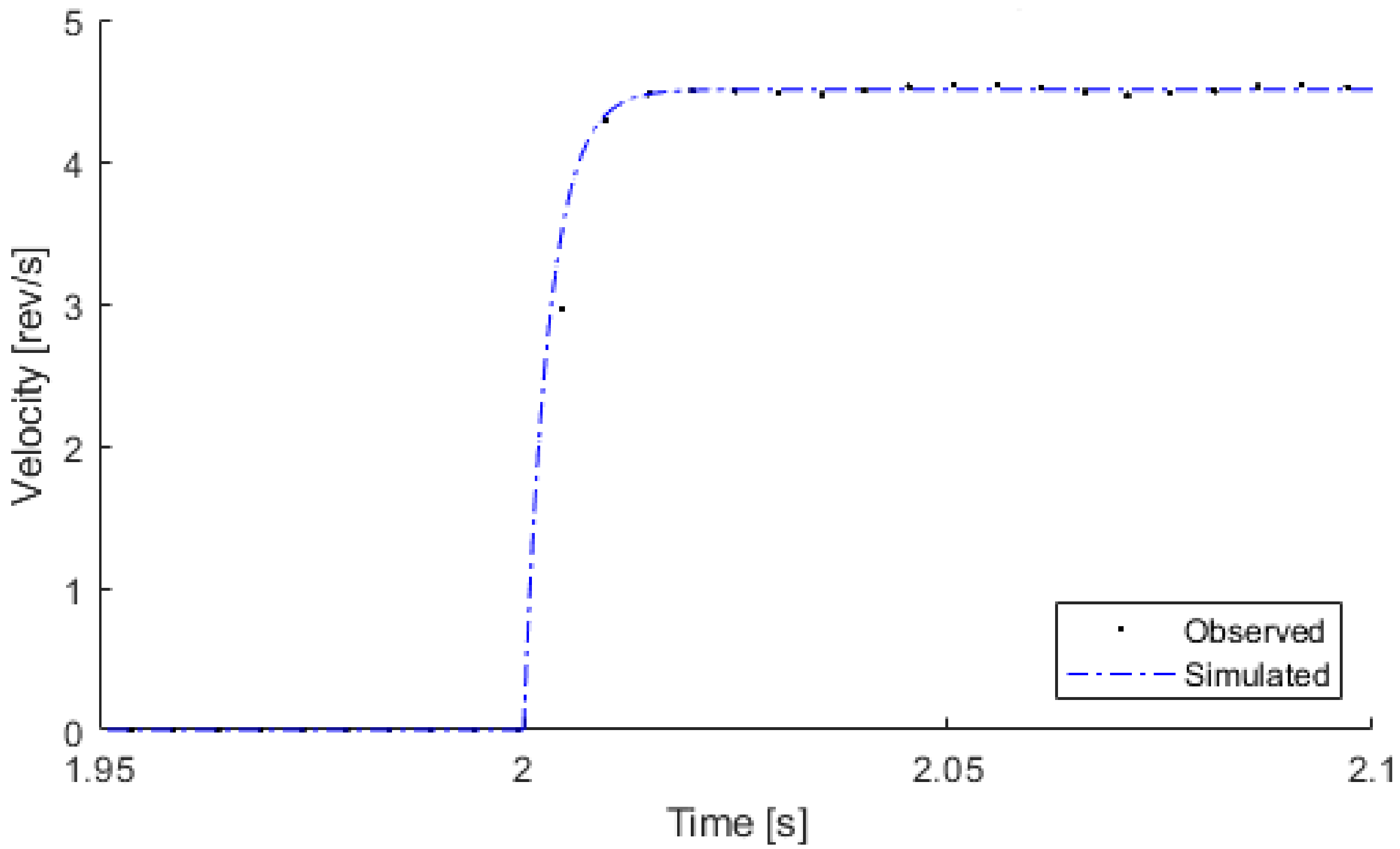
Discussion – characterization

- Settling time $T_s = 0.012\ s$
- Time constant

$$\frac{1}{a} = \frac{T_s}{4} = \frac{0.012\ s}{4} = 0.003\ s$$

- Steady-state $SSV = 4.5060 \frac{rev}{s}$
- Gain $K = SSV * a = 1502.0 \frac{rev}{s^2}$

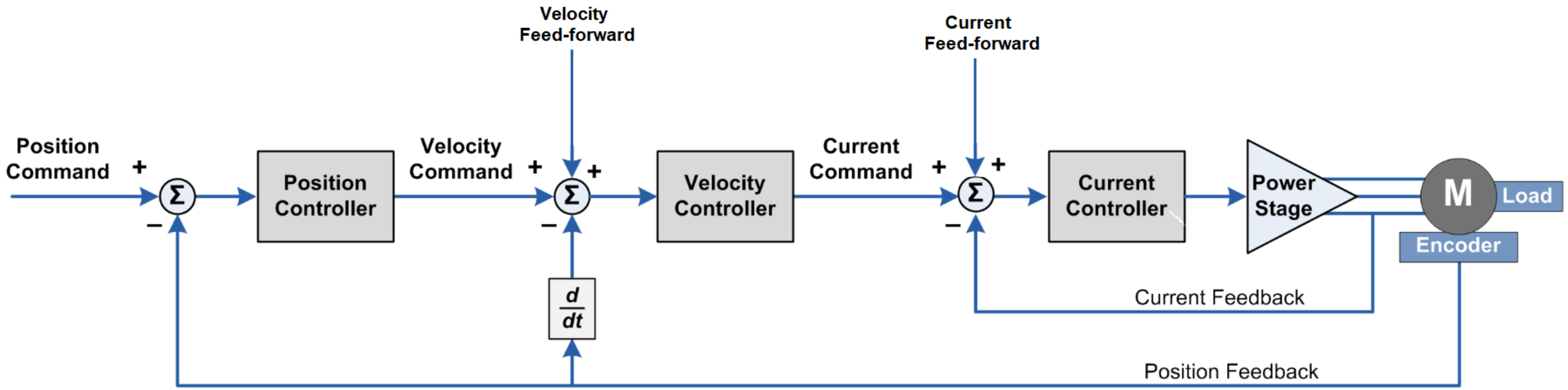
$$G(s) = \frac{1502}{s(s+333)}$$



Discussion – ODrive controller

$$C(s) = \frac{P(s)I(s)V(s)}{1 + I(s)V(s)s + P(s)I(s)V(s)}$$

Where $P(s) = K_p$, $V(s) = K_v + \frac{K_{vi}}{s}$, and $I(s) = K_i + \frac{K_{ii}}{s}$.



[5]

Discussion – ODrive controller

$$C(s) = \frac{P(s)I(s)V(s)}{1 + I(s)V(s)s + P(s)I(s)V(s)}$$

Where $P(s) = K_p$, $V(s) = K_v + \frac{K_{vi}}{s}$, and $I(s) = K_i + \frac{K_{ii}}{s}$.

ODrive config	ODrive field	Corresponds to	Set by
<odrive>.controller.config	pos_gain	K_p	User (default 20.0)
	vel_gain	K_v	User (default 0.0005)
	vel_integrator_gain	K_{vi}	User (default 0.001)
<odrive>.motor.current_control	p_gain	K_i	$current\_control\_bandwidth * phase\_inductance$
	i_gain	K_{ii}	$\left(\frac{phase\_resistance}{phase\_inductance}\right) * K_i$

- current_control_bandwidth, phase_resistance, and phase_inductance are accessible via <odrive>.motor.config
- phase_resistance and phase_inductance are set during motor calibration

Discussion – controller design/tuning

- Know all required information for pole placement, etc.
- Would need to:
  - Prove current control block will actually act as a simple PI loop
  - Confirm that feed-forward terms are not in use
  - Design controller adaptively *or* show that phase resistance and inductance are approximately constant
- Could also design new controller

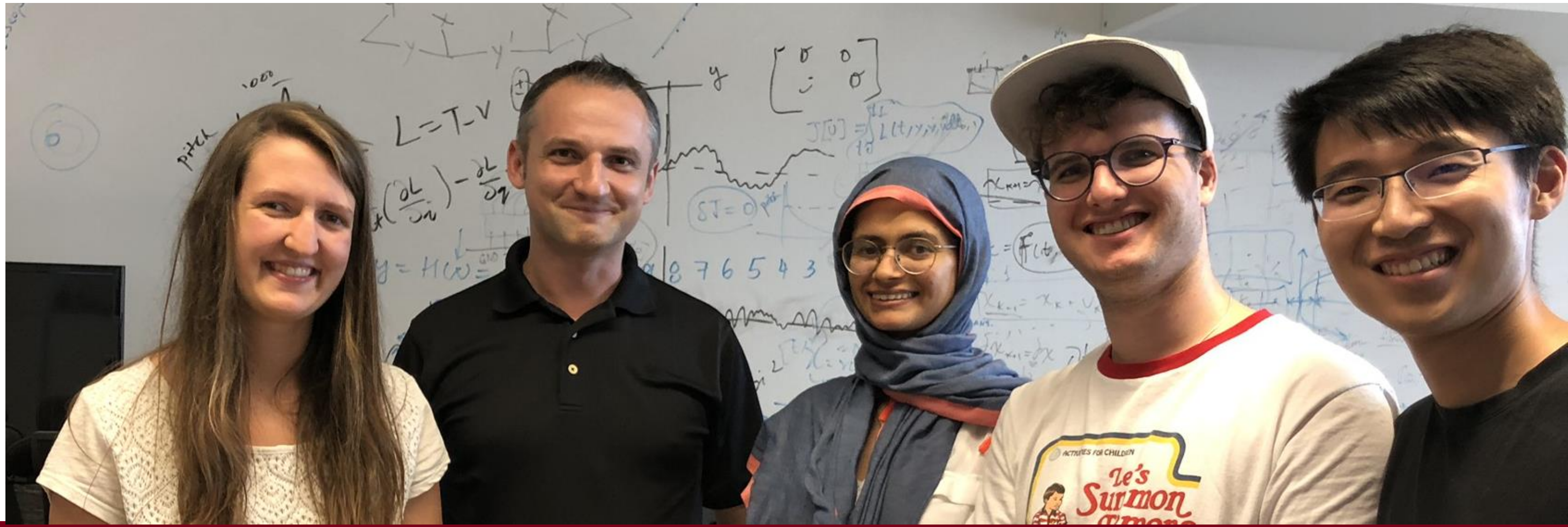
Conclusions

- Successful system for characterization
  - Potential for automation, more complex controls
- v0.5 updates will be formalized in the next week
- Lab reference at the U and in general
  - Public lab GitHub github.com/labmrd/odrive-with-motor-characterization
  - Main ODrive project github.com/odriverobotics/ODrive

Acknowledgements

- Professor Tim Kowalewski
- Yusra Farhat Ullah, Mark Gotthelf
- ODrive developers Oskar Weigl, Paul Guenette

This work was supported, in part, by the National Science Foundation CAREER Grant under 331 Award No. 1847610. Opinions, interpretations, conclusions and recommendations are those of the 332 author and are not necessarily endorsed by the National Science Foundation



References

[1] J. Zhao and Y. Yangwei, “Brushless DC Motor Fundamentals Application Note,” *MPS*. Monolithic Power, pp. 1–19, 2014.

[2] J. Braun, “Formulae Handbook,” *maxon academy*. maxon motor, Sechseln, pp. 1–60, 2012.

[3] C. Rollman, “What is ‘Field Oriented Control’ and what good is it?,” *Copley Controls Corp.* pp. 1–13, 2002.

[4] “Park, Inverse Park And Clarke, Inverse Clarke Transformations MSS Software Implementation User Guide,” *Microsemi Corporation*. Alieso Viejo, CA, pp. 5–9, 2013.

[5] “ODrive Documentation,” *ODrive Robotics*, 2021. [Online]. Available: docs.odriverobotics.com. [Accessed: 01-Apr-2021].

[6] “Motion System Tuning.” Rockwell Automation, Milwaukee, WI, pp. 22–23, 2020.

[7] “ODrive Robotics GitHub,” *ODrive Robotics*. [Online]. Available: github.com/odriverobotics. [Accessed: 01-Apr-2021].

[8] “ODrive Firmware Developer Guide,” *ODrive Robotics*. [Online]. Available: docs.odriverobotics.com/developer-guide. [Accessed: 01-Apr-2021].

[9] “ODrive encoder guide,” *ODrive Robotics*. [Online]. Available: https://docs.google.com/spreadsheets/d/1OBDwYrBb5zUPZLrhL98ezZbg94tUsZcdTuwiVNgVqpU/edit#gid=0. [Accessed: 01-Apr-2021].

[10] “ODrive motor guide,” *ODrive Robotics*. [Online]. Available: https://docs.google.com/spreadsheets/d/12vzz7XVEK6YNIOqH0jAz51F5VUpc-lJEs3mmkWP1H4Y/edit#gid=0. [Accessed: 01-Apr-2021].

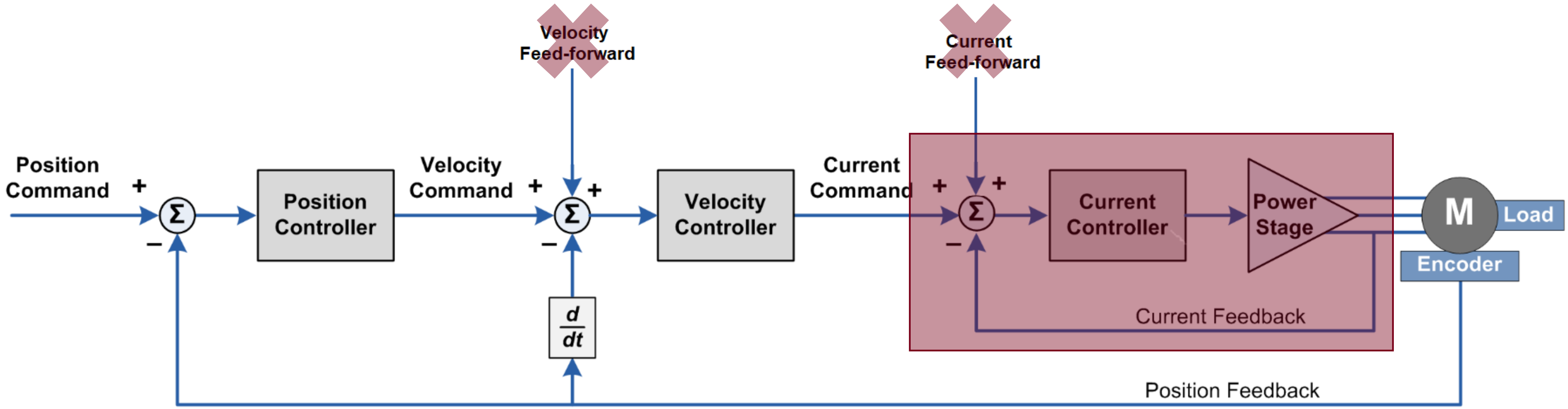
[11] R. Smoot, “What’s the Difference Between an Incremental Encoder’s PPR, CPR, and LPR?,” *CUI Insights: Motion*, 2021. [Online]. Available: cuidevices.com/blog/what-is-encoder-ppr-cpr-and-lpr. [Accessed: 01-Apr-2021].

[12] E. Goldberg, “ODrive Setup Walkthrough.” MRD Lab, University of Minnesota Department of Mechanical Engineering, Minneapolis, MN, 2021.

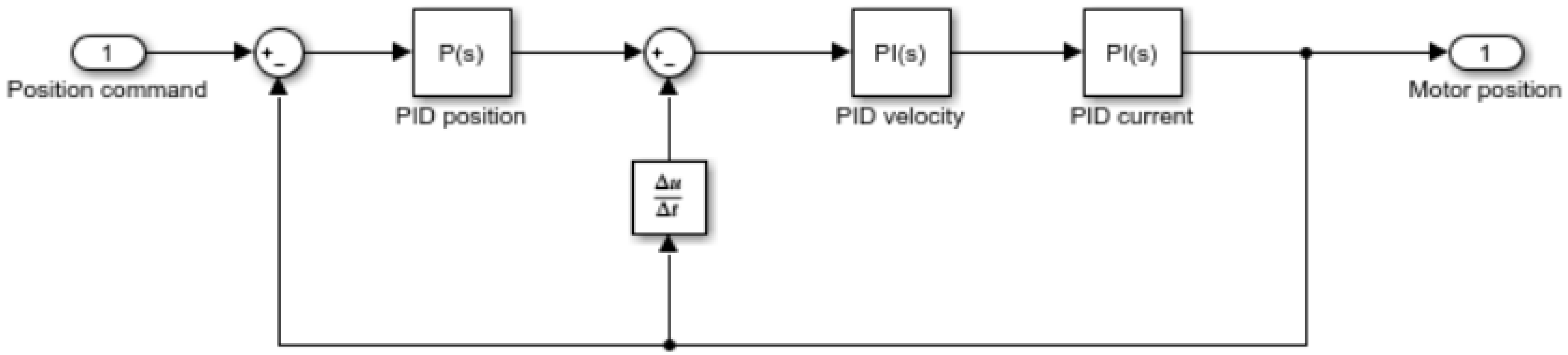
[13] “MRD Lab GitHub,” *Medical Robotics and Devices Lab*. [Online]. Available: github.com/labmrd. [Accessed: 01-Apr-2021].

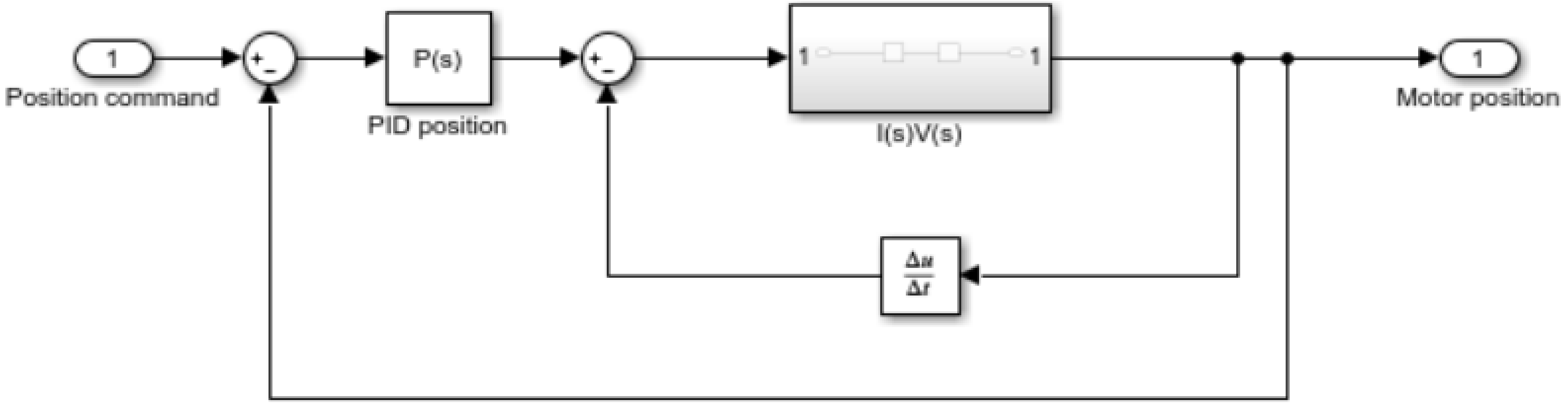
[14] E. Goldberg, “VS Code for ODrive.” MRD Lab, University of Minnesota Department of Mechanical Engineering, Minneapolis, MN, 2021.

Controller simplification



Controller simplification





Controller simplification