Derivative Control of Second-Order Plant

Digital Control System Design

Professor Fuller

Project Assignment

Assignment

1. Read and understand the remainder of this pdf.
2. Construct the circuit.[*1] Get hardware components per Canvas announcement
3. Use the provided *pyboard.py* and *<yourName>_lab1.py* to implement the difference equation of the discrete derivative controller.
4. Observe the response. With sample period of $T_s = 0.05$ seconds:
    1. Set $K_d = 0.0$, and $K_{ff} = 1.0$. Apply a unit step input and run for 10 seconds. Save the output as a .csv. Describe the response.
    2. Now add derivative control to see if we can reduce the overshoot in the plant output. Set $k_d = 0.2631$ and unit step the reference input by 1.0 volts and run for 4 seconds. Save the output as a .csv. Describe the response.
    3. Step the disturbance input by 1.0 volts and run for 10 seconds. Save the output as a separate .csv. Describe the response and compare to the ref step response.
5. Analyze your saved data and generate <u>your own</u> plots displaying the output, inputs, and control signal over time. Generate high quality figures.
6. Submit a report fulfilling the instructions on the next slide.

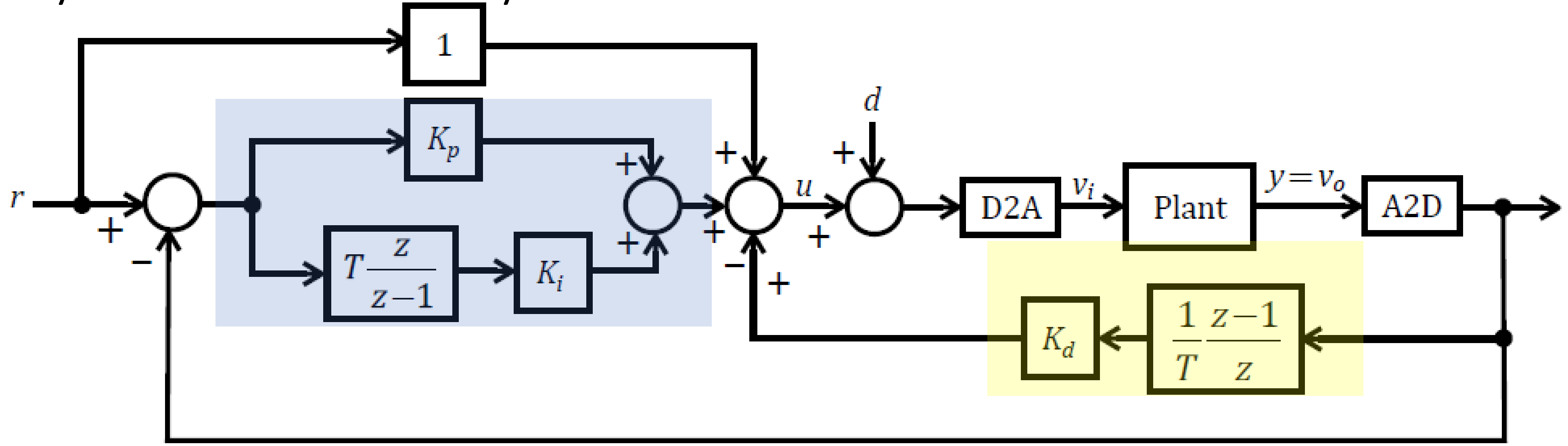
*1 Suggestion: sketch a circuit schematic showing your microcontroller and plant connections. This will help you to breadboard the system.

What to submit:

1. One PDF that includes:
    1. An “executive summary” that describes what you did, the results you obtained, and the problems/difficulties you ran into in your work to complete this project.
        - Think of this as an abstract but with a practical emphasis, as for someone intending to use your control system and plant and needs a quick reference to know everything important about it. Keep it succinct.
    2. Your plots of the reference and disturbance step experiments. Make sure to also plot the input signals.
    3. Brief (<=1 page) discussion of your interpretation of major features of the plant and controller, and your observations from tasks 4.1 – 4.3, referencing your plots. For example: explain the role of “Kff”; what is the difference other types of disturbances that could be modeled and where they would enter your block diagram. Discuss anything else you consider relevant.
2. Your code and saved data files.
3. Photo(s) of your hardware sufficient to understand your wiring

What we’re doing:

Our goal is to control a plant, in this lab we will first see if we can drive the output toward reference (control) with just feedforward and derivative control. In particular, we wish to damp the overshoot of the closed loop response. You will implement this block diagram on the microcontroller using CircuitPython and MicroPython!



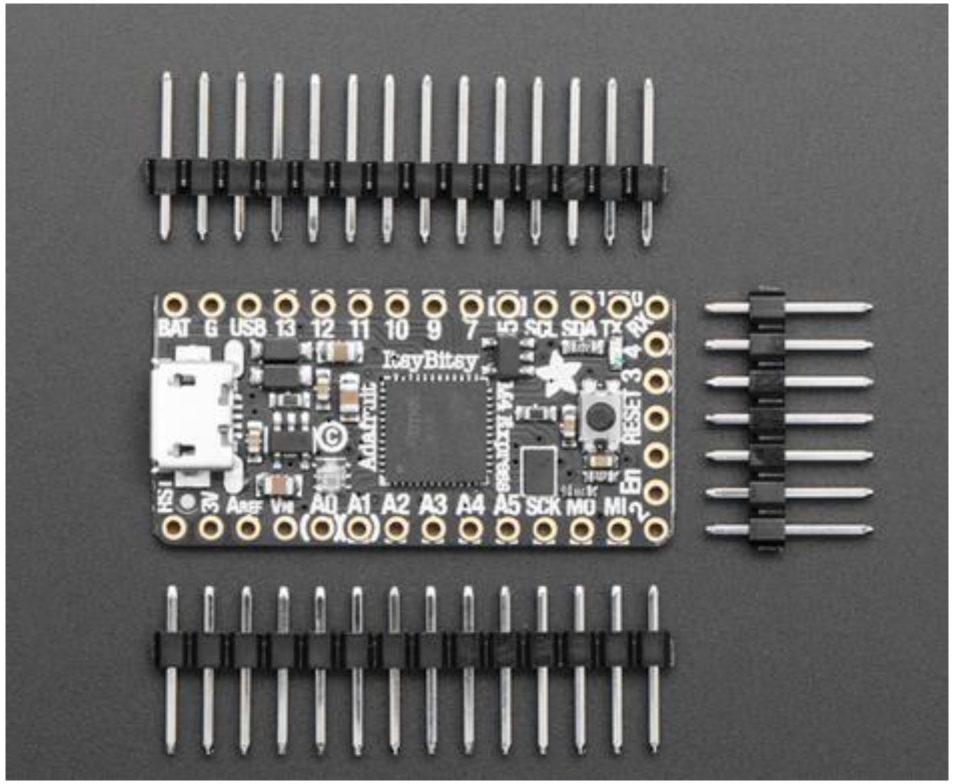
Block diagram showing disturbance input “d” affecting the control signal “u”. The disturbance affects “u” before it is actuated by the D2A.

- Note: This shows a PI controller (blue highlight) + derivative, whereas your controller will be D-only (yellow) with feed-forward (FF). This particular feed-forward control element is represented by the unity-gain block at top.

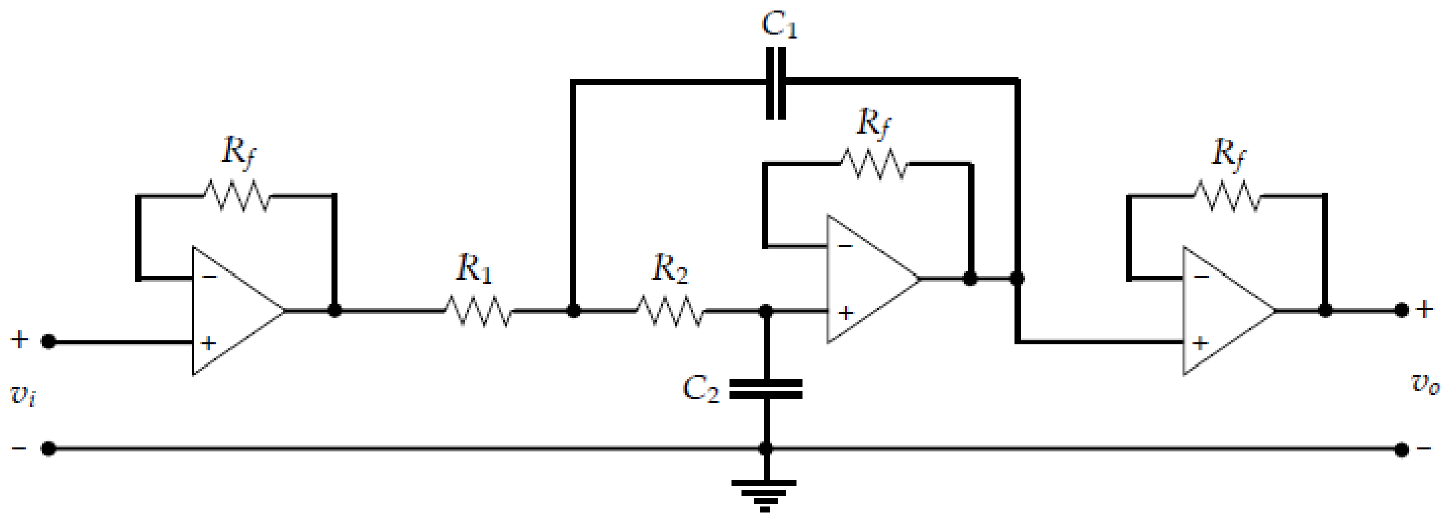
Microcontroller

controller, actuator (D2A converter “DAC”), and sensor (A2D converter “ADC”)

- Arm core M4 ATSAMD51 “Adafruit Itsy Bitsy”
  - 120 MHz, FPU, 12-bit DAC and ADC with plenty of pins
  - https://learn.adafruit.com/introducing-adafruit-itsybitsy-m4 (pdf also on canvas)
- Circuit Python
  - Runs simplified Python! Can import some libraries like Numpy.
  - Easy to use with Pyboard.py tool to execute your .py code
    - Command line or scripted call of pyboard.py (demo in class)


BAT G USB 13 12 11 10 9 7 SCL SDA TX RX
ItsyBitsy
Adafruit
RESET
En
3V Vhi A0 A1 A2 A3 A4 A5 SCK MO MI 2

Second Order Analog Plant



$$\underbrace{\quad\quad\quad}_{1}$$

Input Impedance $= \infty$

Output Impedance $= 0$

$R_f \neq \infty$

$$\underbrace{\quad\quad\quad}_{\frac{\omega_{n_1}^2}{s^2 + 2\zeta_1 \omega_{n_1} s + \omega_{n_1}^2}}$$

$$\omega_{n_1} = \frac{1}{\sqrt{R_1 C_1 R_2 C_2}} \quad \zeta_1 = \frac{(R_1 + R_2) C_2}{2\sqrt{R_1 C_1 R_2 C_2}}$$

$$\left.\begin{aligned} R_1 &= 160\text{K } \Omega \\ R_2 &= 200\text{K } \Omega \\ C_1 &= 10\ \mu\text{F} \\ C_2 &= 0.082\ \mu\text{F} \end{aligned}\right\} \Rightarrow \begin{cases} \omega_{n_1} = 6.173\ \text{rad/sec} \\ \zeta_1 = 0.09112 \end{cases}$$

$$\underbrace{\quad\quad\quad}_{1}$$

Input Impedance $= \infty$

Output Impedance $= 0$

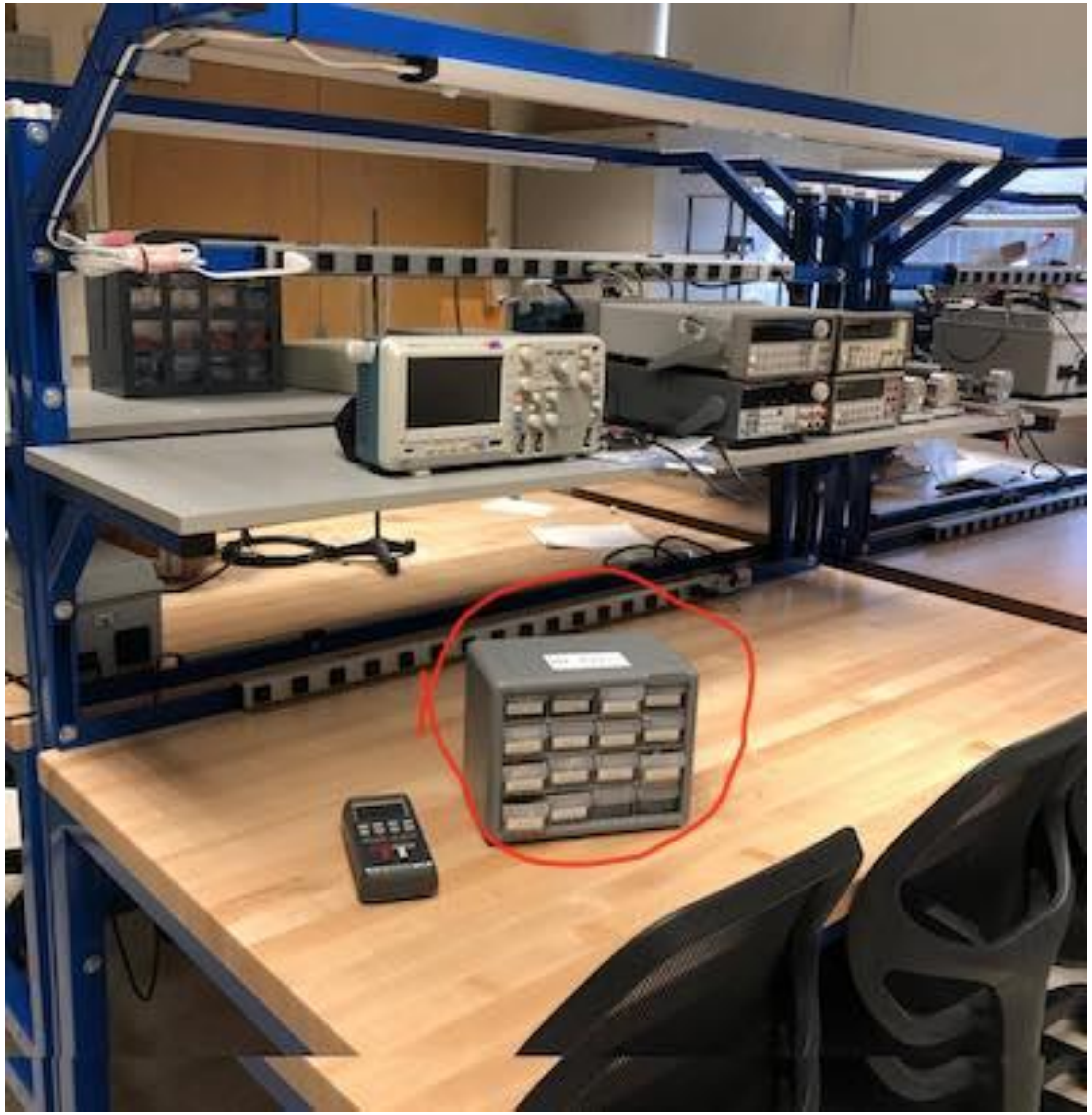
$R_f \neq \infty$

Electrical schematic representing a second order analog plant. Note that the voltage-followers “buffers” left and right (with transfer function of 1) are not part of the ~1Hz resonant mode we are modeling, but are instead there to avoid loading effects at the input and output of the plant.

Sampling theorem:

$\Rightarrow$ Absolute minimum advisable sampling rate =

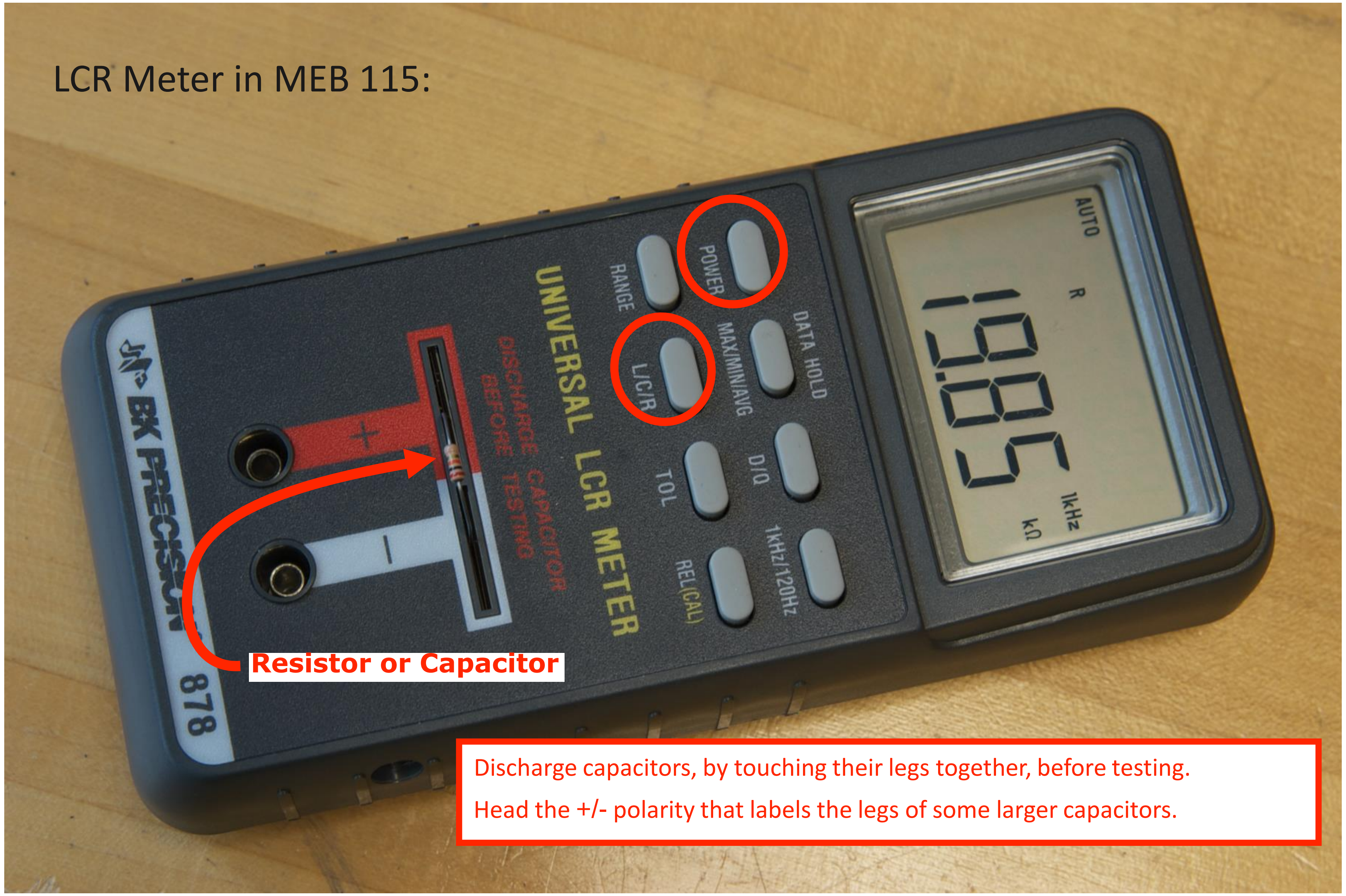
$$\frac{6.173\ \frac{rad}{sec}}{2\,\pi\ \frac{rad}{cycle}} \times 2\,\frac{samples}{cycle} = 1.965\,\frac{samples}{cycle} = 1.965\ Hz$$



Component location

MEB115 – Mechatronics lab
Door code: 2468
(also announced in Canvas)

How to measure component values

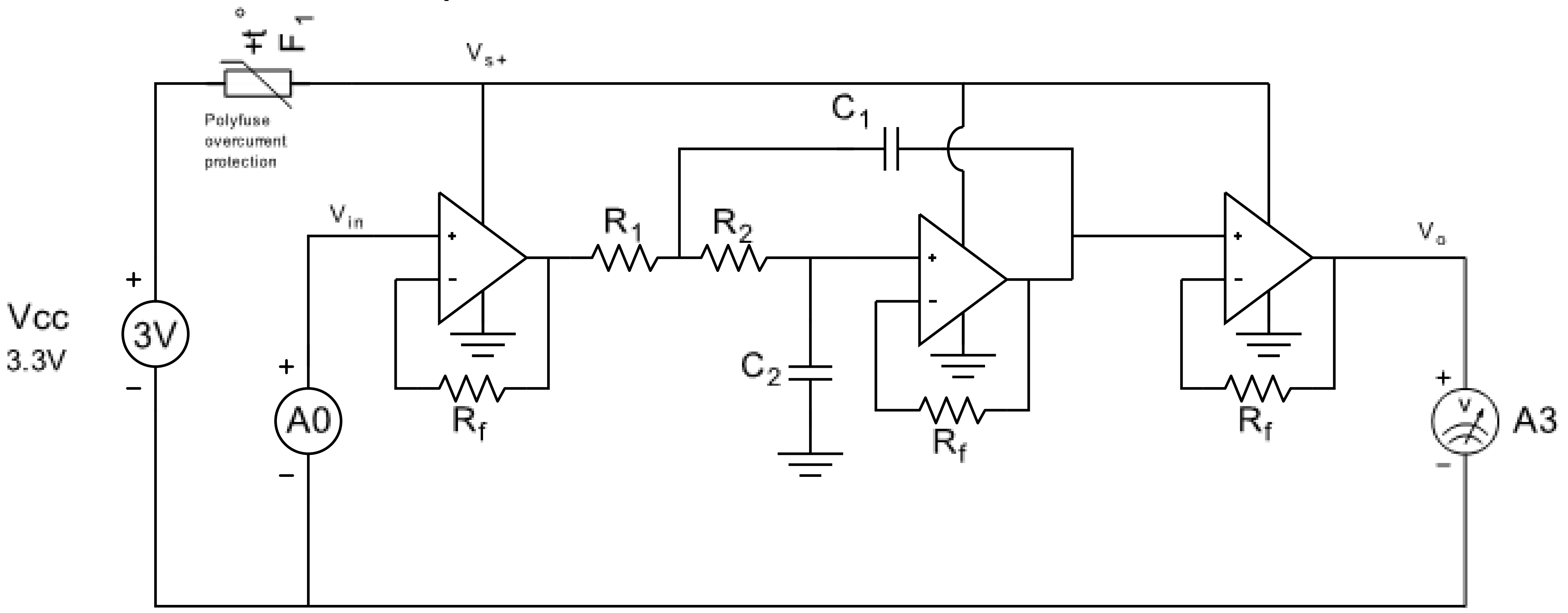


LCR Meter in MEB 115:

Discharge capacitors, by touching their legs together, before testing.

Head the +/- polarity that labels the legs of some larger capacitors.

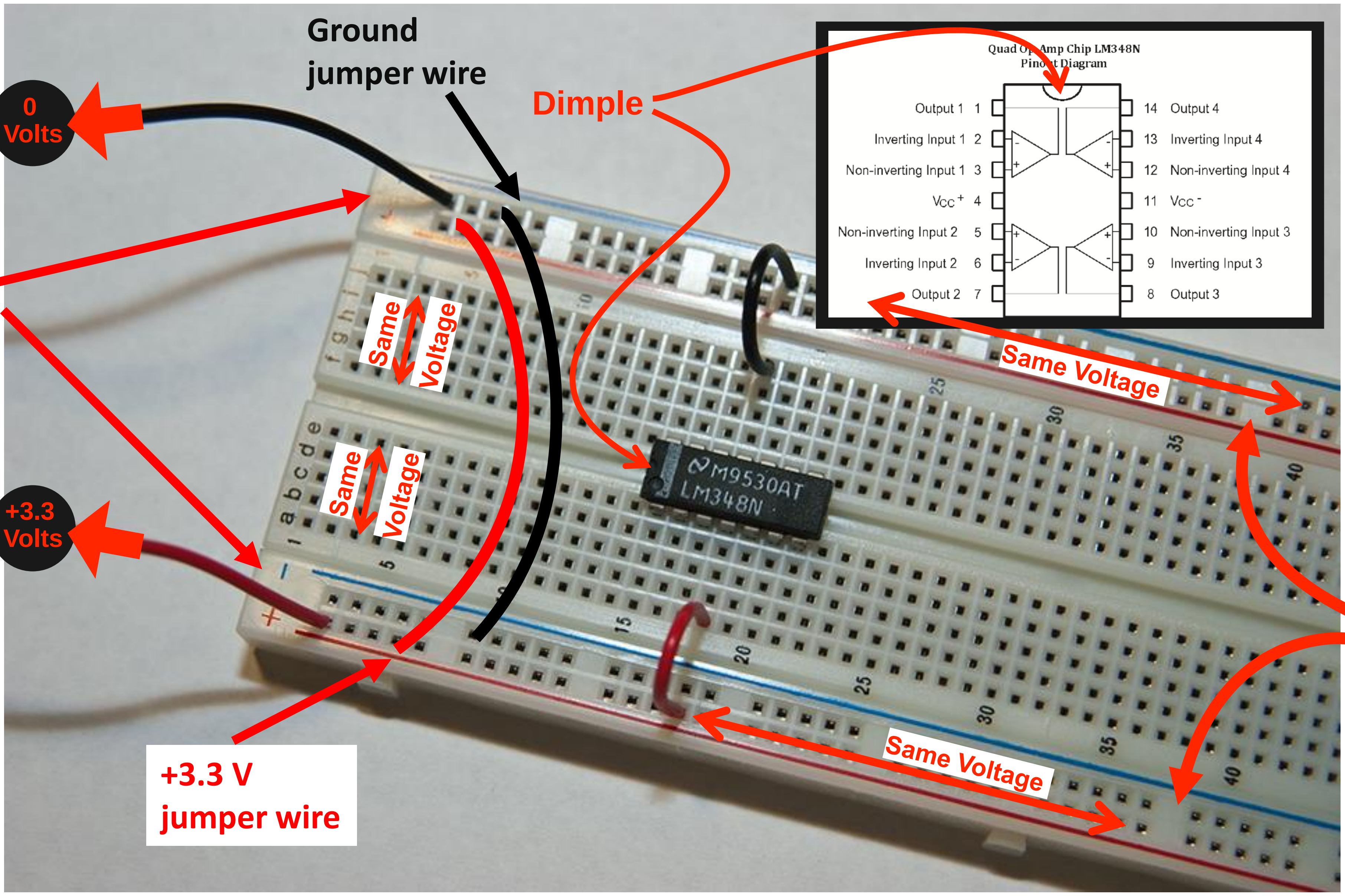
Complete circuit schematic



Voltage sources “3V”, “A0” and the analog input “A3” are labeled pins on the microcontroller. Consult the datasheet for the op amp pinout. Reference images are also below.

Caution: Be careful not to cause short circuits when placing the polyfuse “F1” or when moving the microcontroller, as the “USB” (5V) and “G” pins on the Itsy Bitsy are next to each other and shorts could easily occur by misplacing things by a single row.

Breadboard Wiring Basics

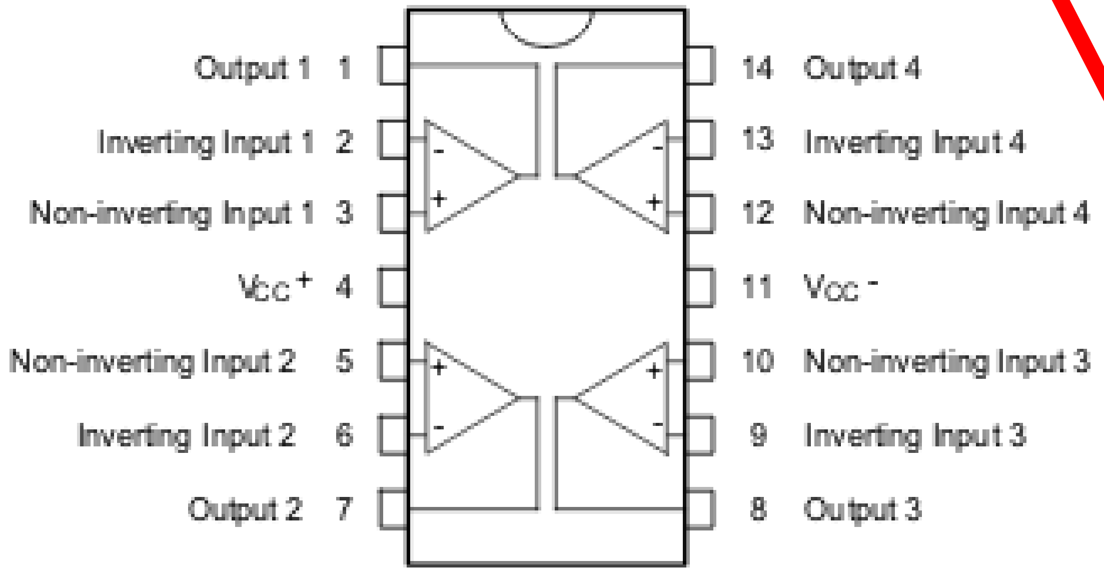


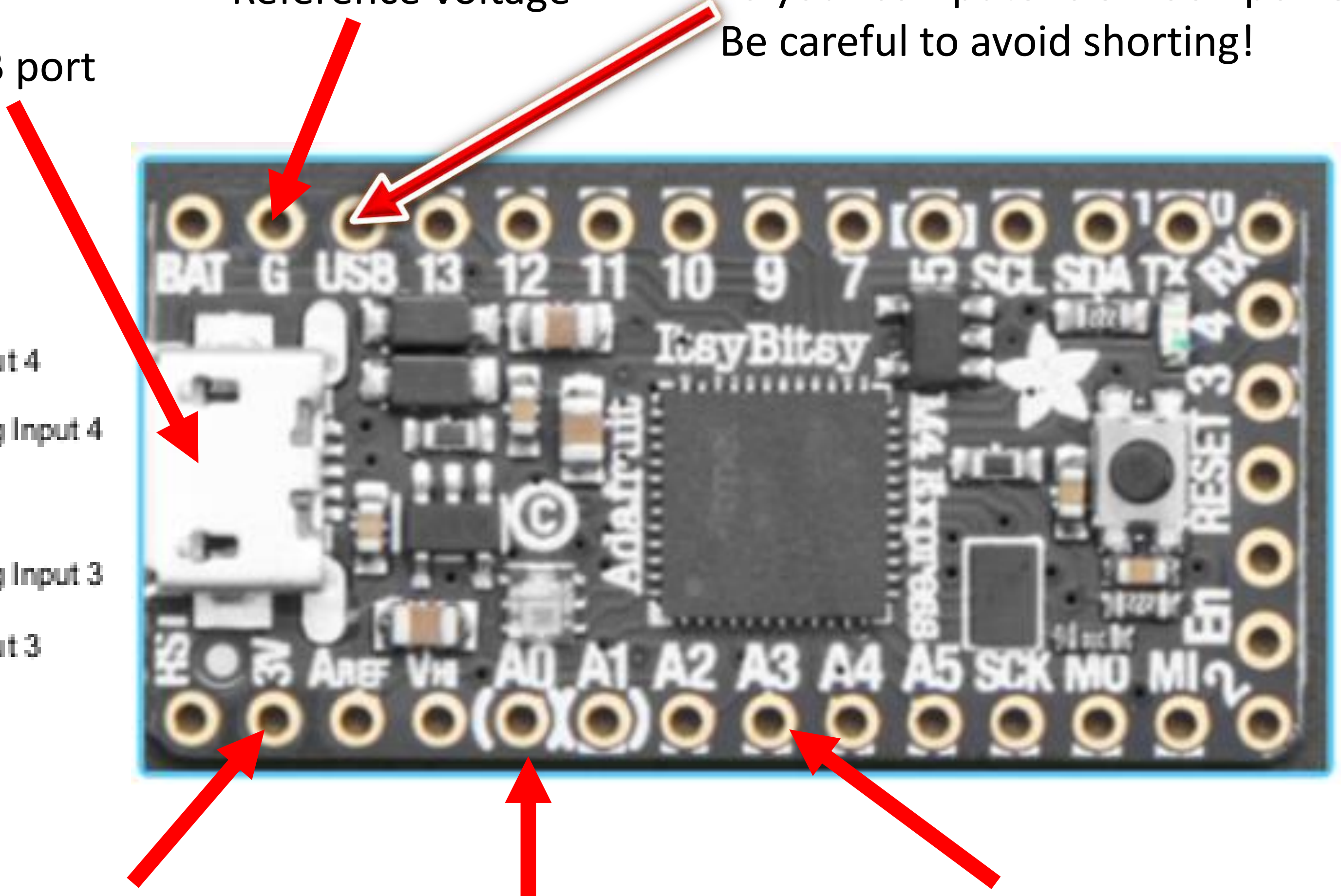
+ and –
Are useful as
power rails

Warning:
Some breadboards have a break in the power rails at these gaps in the middle of the breadboard, requiring a jumper wire to bridge to the rows of the power rails on the other side

Microcontroller and Op Amp

MCP6001





BAT G USB 13 12 11 10 9 7 5 SCL SDA TX RX
ItsyBitsy
Adafruit
RESET
3V AREF VHI A0 A1 A2 A3 A4 A5 SCK MO MI

USB port

G, ground
Reference voltage

USB, connects directly
To your computer's 5V USB power
Be careful to avoid shorting!

VCC, 3.3V supply
To polyfuse

A0, analog out

A3, analog input

Running your .py script and saving data

- Using your python package manager, install pyserial. ex: `pip install pyserial` or `conda install -c anaconda pyserial`
- Download the *pyboard.py* and *<yourName>_lab1.py* from the assignment page into the same directory.
- *<yourName>* is your name, you change the file name to reflect.
- Edit *<yourName>_lab1.py* to incorporate your *C(k)* difference equation for the discrete derivative control element.
- With the circuit and microcontroller safe to energize, plug in the microcontroller via usb and identify the USB device.
- Open a terminal or command prompt in that same directory.
- Terminal command:

  ```
  python pyboard.py --device <deviceName> <yourName>_lab1.py >datafile.csv
  ```

  Explanation:
  - Your system will call python executable (you may have to type "python3" instead of "python" depending on your Python environment) to run the *pyboard.py.*
  - *Pyboard.py* will look for the USB device with the specified name, e.g. `<deviceName>` is the USB device identified above, e.g.
    - *Windows: `COM`**`x`***, check the device manager: device manager->Ports (Com or LPT). If Itsy Bitsy is COM16 for ex, you would replace `<deviceName>` with `COM16`
    - Mac, easy to find device ex: or `/dev/tty.usbmodem*`.
    - Linux `/dev/ttyUSB2` if you're running linux you can figure it out
  - *Pyboard.py* will then execute the script **`<yourName>_lab1.py`** on that USB device. (pyboard.py will look in the current directory for that file unless a different directory is specified)
  - `>datafile.csv` is the part of the command which "redirects" (the > operator) any console output from that script into a comma-separated-variable (CSV) file of the specified file name. Since we print the measurements over time, that's what we'll get in the csv. If the script errors out, you will instead get an error log instead of saved data.
- Debugging: instead of redirecting console output to a .csv file, if you are debugging you may wish to see the output printed instead, so simply leave out the "`>datafile.csv`"

Reading the saved data and plotting

Example commands to help get started

csv columns are in order of: $k, t, y, r, d, u, u_{sat}$

Create another Python script that loads this data into an array using, for example,

```
import numpy as np
data = np.loadtxt('data1.csv', delimiter=',’)
```

then use Matplotlib (or your favorite plotting system) to plot various parameters.
For example (the indices may be different for you)

```
    t = data[:,1]
    y = data[:,2]
    plt.plot(t, y)
```

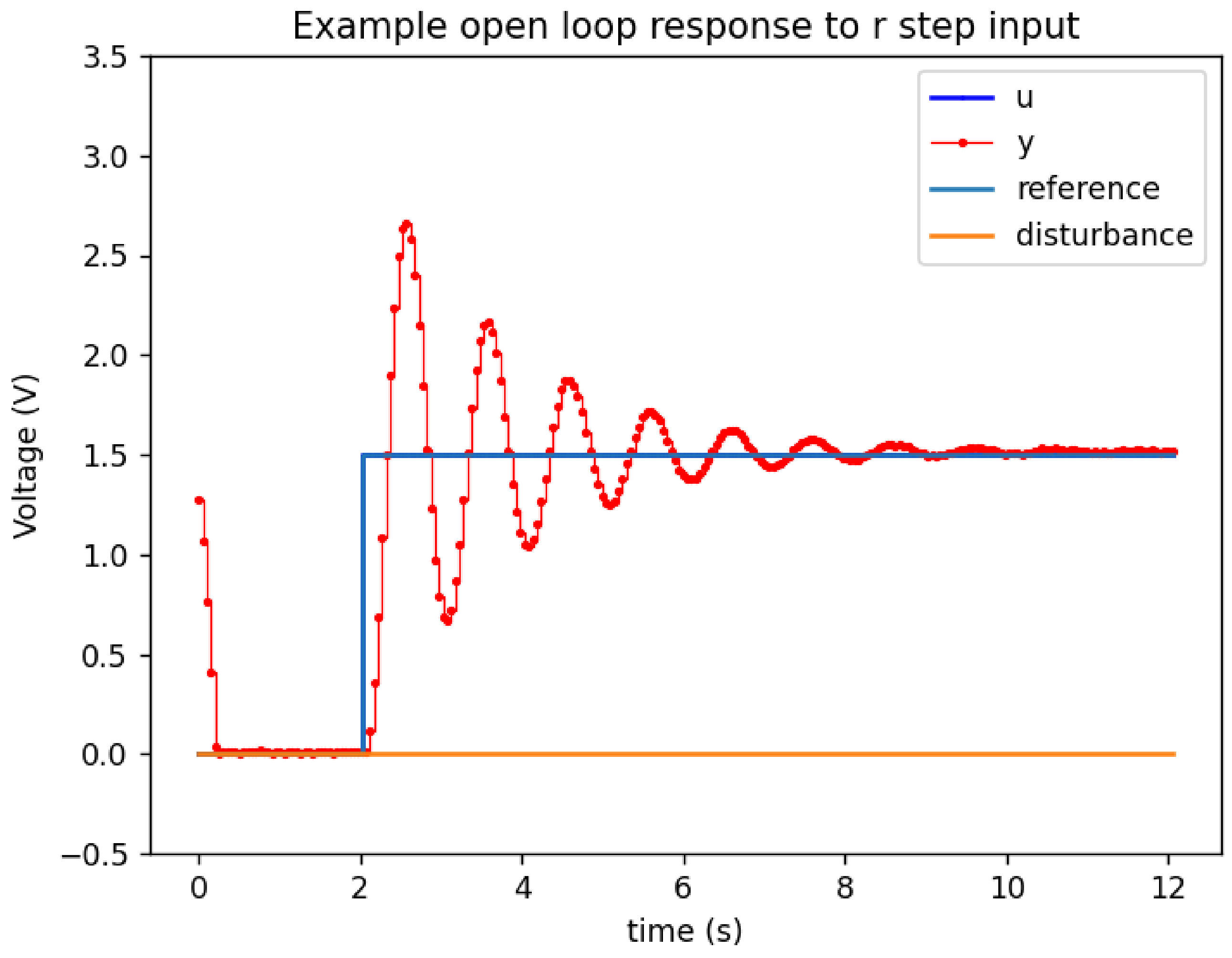
Make sure to label your plot axes.
Save your plot using, e.g. `plt.savefig('myfigname.pdf’)`

And an equivalent sketch for Matlab:

```
        data = readtable('myfile.csv’);
        t = data(:,2);
        y = data(:, 3);
        plot(t, y);
```

Example reference step response

Feel free to plot other ways



Example plotting commands

```
plt.figure()
plt.step(time, u, '.-', color='blue', markersize=1,
linewidth=1.5, label='u', where='post')
plt.step(time, y, '.-', color='red', markersize=4,
linewidth=0.7, label='y', where='post')
plt.step(time, r, label='reference', where='post')
plt.step(time, d, label='disturbance', where='post')
# plt.ylim([-0.5, 3.5])
plt.xlabel('time (s)')
plt.ylabel('Voltage (V)')
plt.title(r'Response to ref step input with PID
$k_p=0.0, k_i=0.0, k_d=0.0$ ')
plt.legend( loc='best')
```