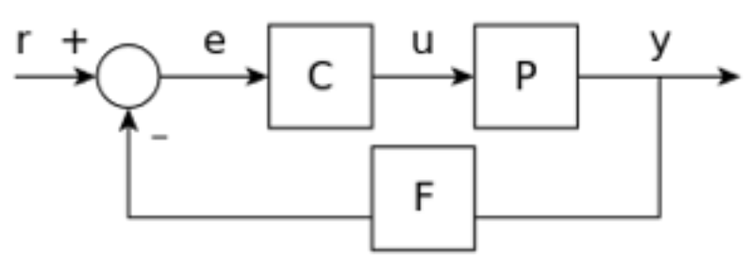
Pendulum
+ Beaglebone
+ Control Theory
= Profit



Justin Pearson
IEEE Club
University of California, Santa Barbara
Nov 21, 2016

Outline

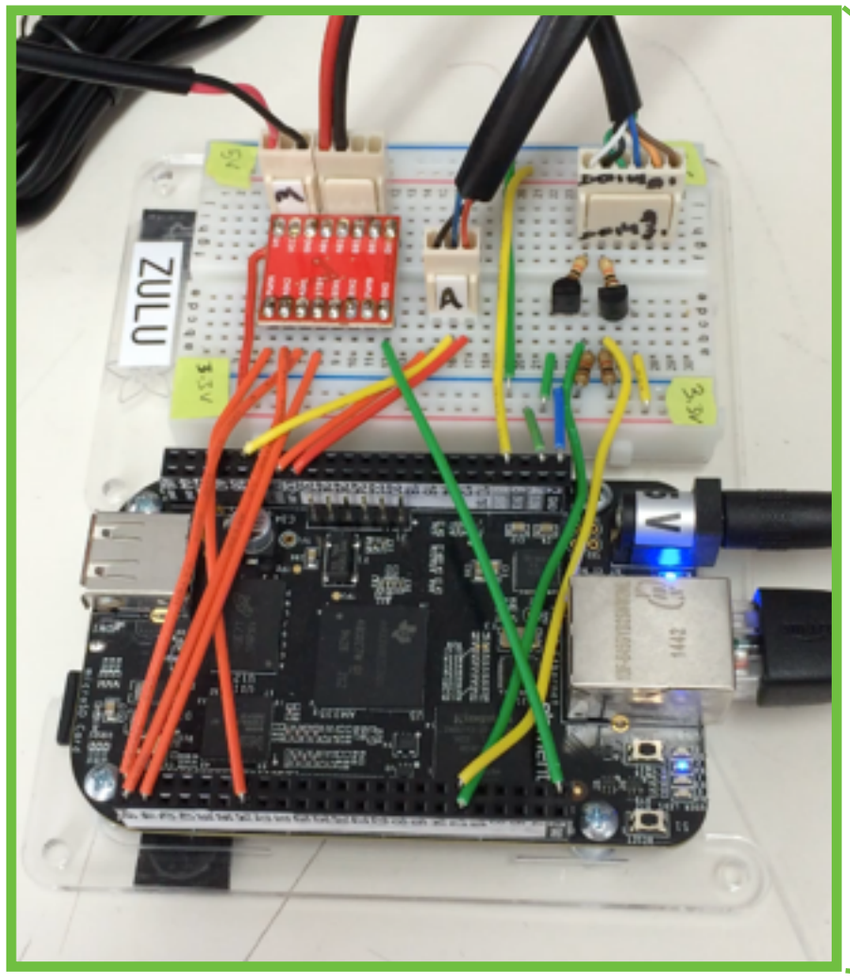
- **Omg Demo**
- Pendulum
- Beaglebone
- Control Theory

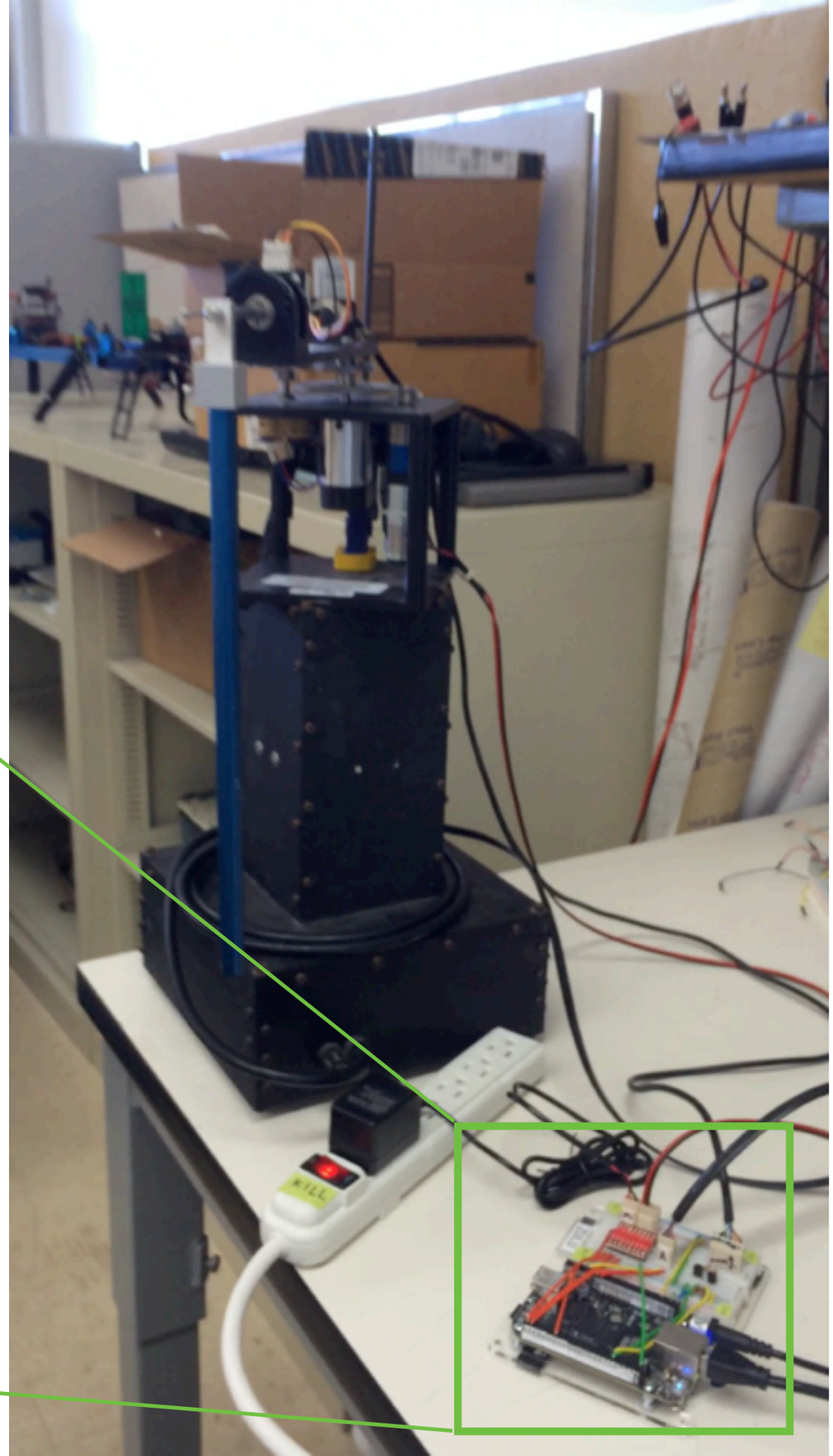
First:

omg demo

Demo

BBB
balances
pendulum





rotary-inverted-pendulum-balance-from-beaglebone-v2.MOV

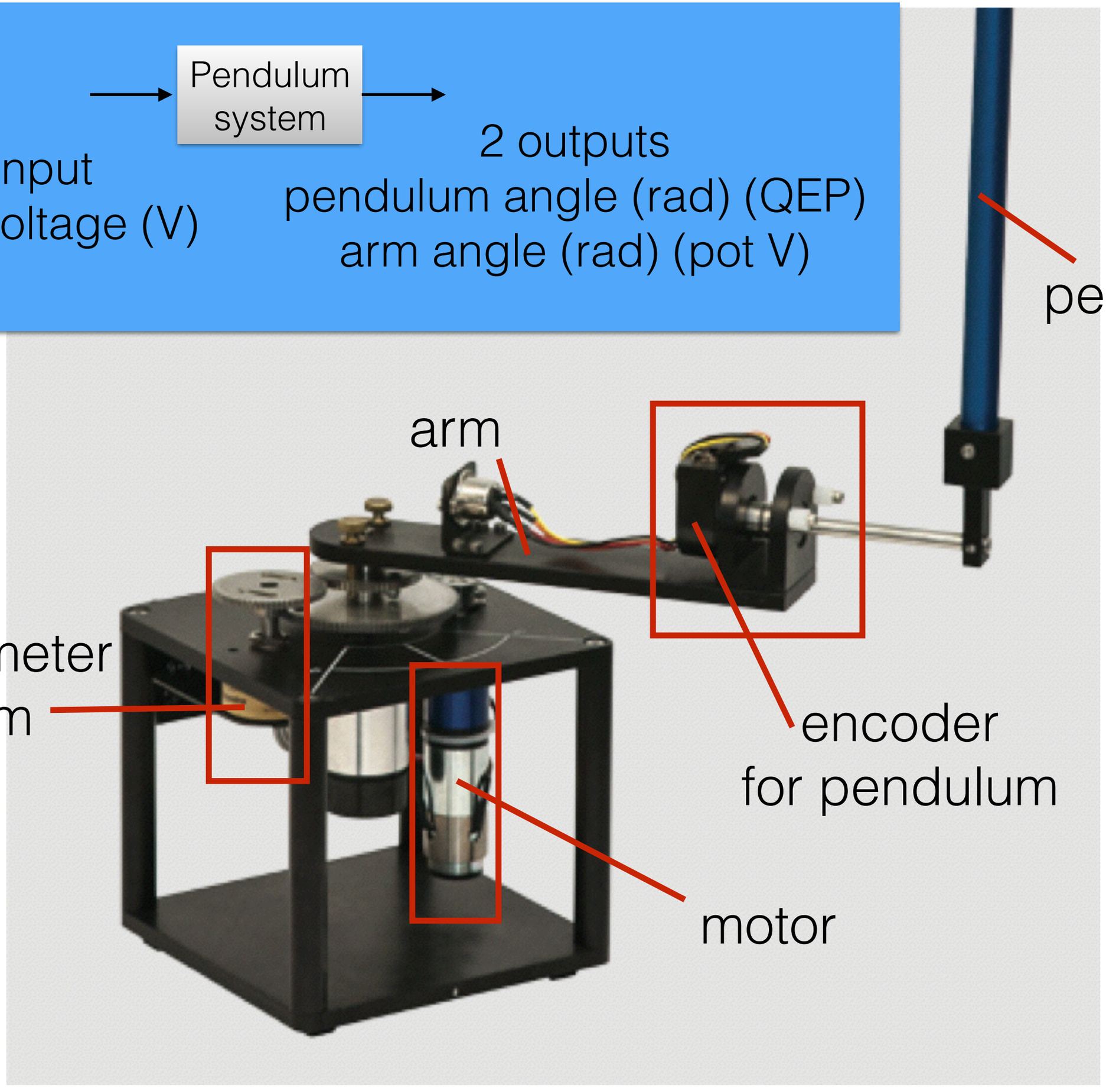
Outline

- Omg Demo
- **Pendulum**
- Beaglebone
- Control Theory

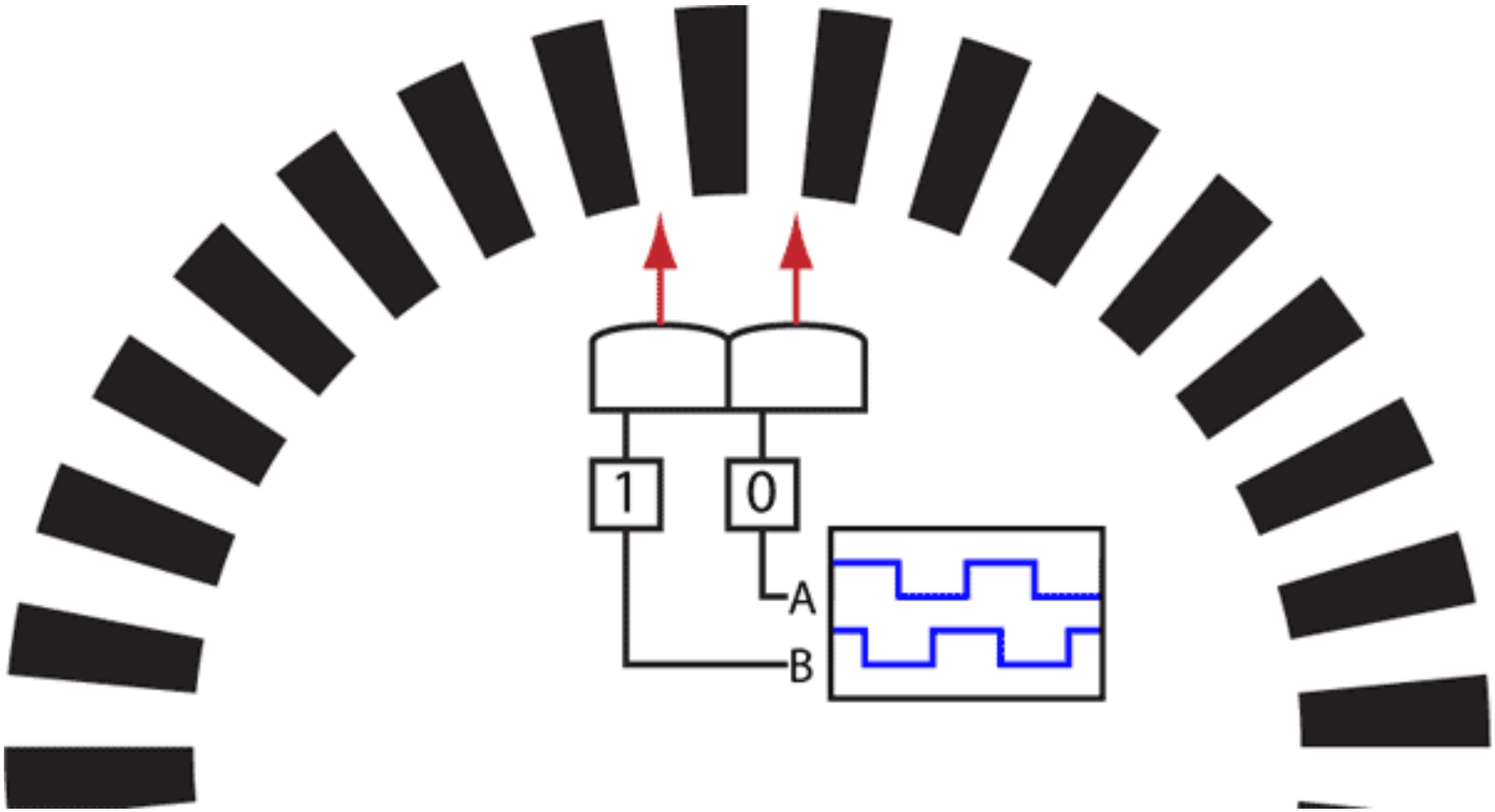
Pendulum system

1 input
motor voltage (V)

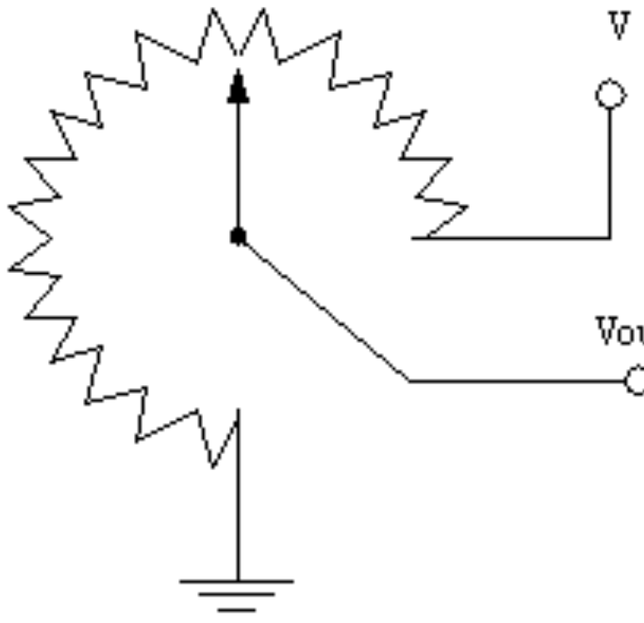
2 outputs
pendulum angle (rad) (QEP)
arm angle (rad) (pot V)



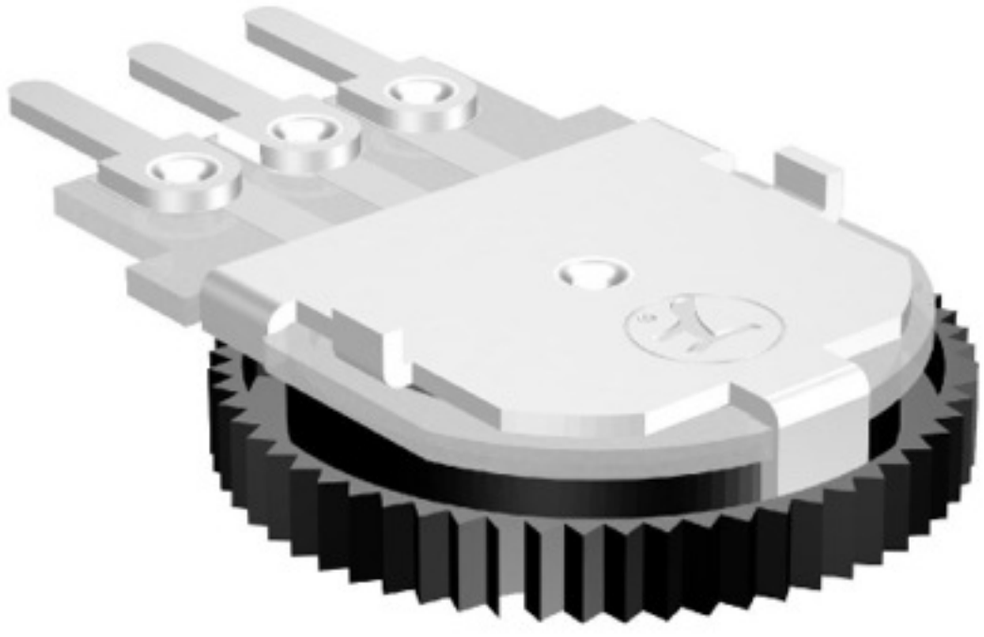
Quadrature-encoded pulses



Rotary Potentiometer



$$V_{out} = V\left(\frac{\theta}{\theta_{max}}\right)$$



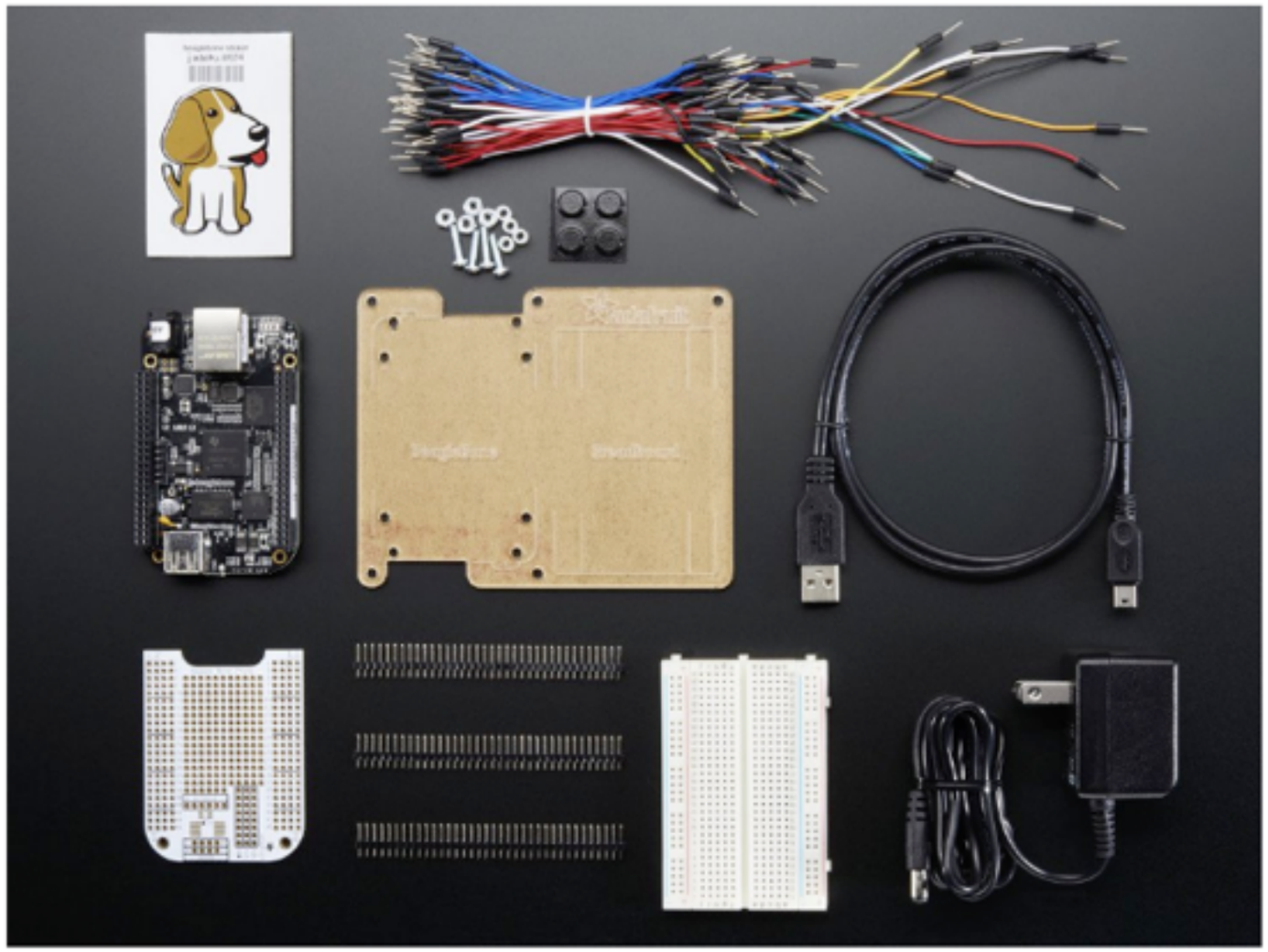
Motor

Faulhaber
Coreless
DC motor
-5V to 5V
we drive w/ PWM

Outline

- Omg Demo
- Pendulum
- **Beaglebone**
- Control Theory

Beaglebone Black



Adafruit Beagle Bone Black Starter Pack

PRODUCT ID: 703

$89.95

OUT OF STOCK

Please enter your details below and we will send you an email when this item is back in stock. You will only be emailed about this product!

YOUR NAME

YOUR EMAIL

NOTIFY ME

ADD TO WISHLIST

DESCRIPTION

If you've heard about the Beagle Bone Black and you want to hit the ground running, this starter pack is for you. We've picked out everything you need to start out, with essential parts and accessories to save on a bundle.

Includes:

- The latest Element 14 **Beagle Bone Black Rev C with 4GB Flash** - fully assembled, tested and ready to rock! **As of December 8th, we're including an Element 14 Bone with**

10/100 Ethernet

USB Host
Easily connects to almost any everyday device such as mouse or keyboard

microHDMI
Connect directly to monitors and TVs

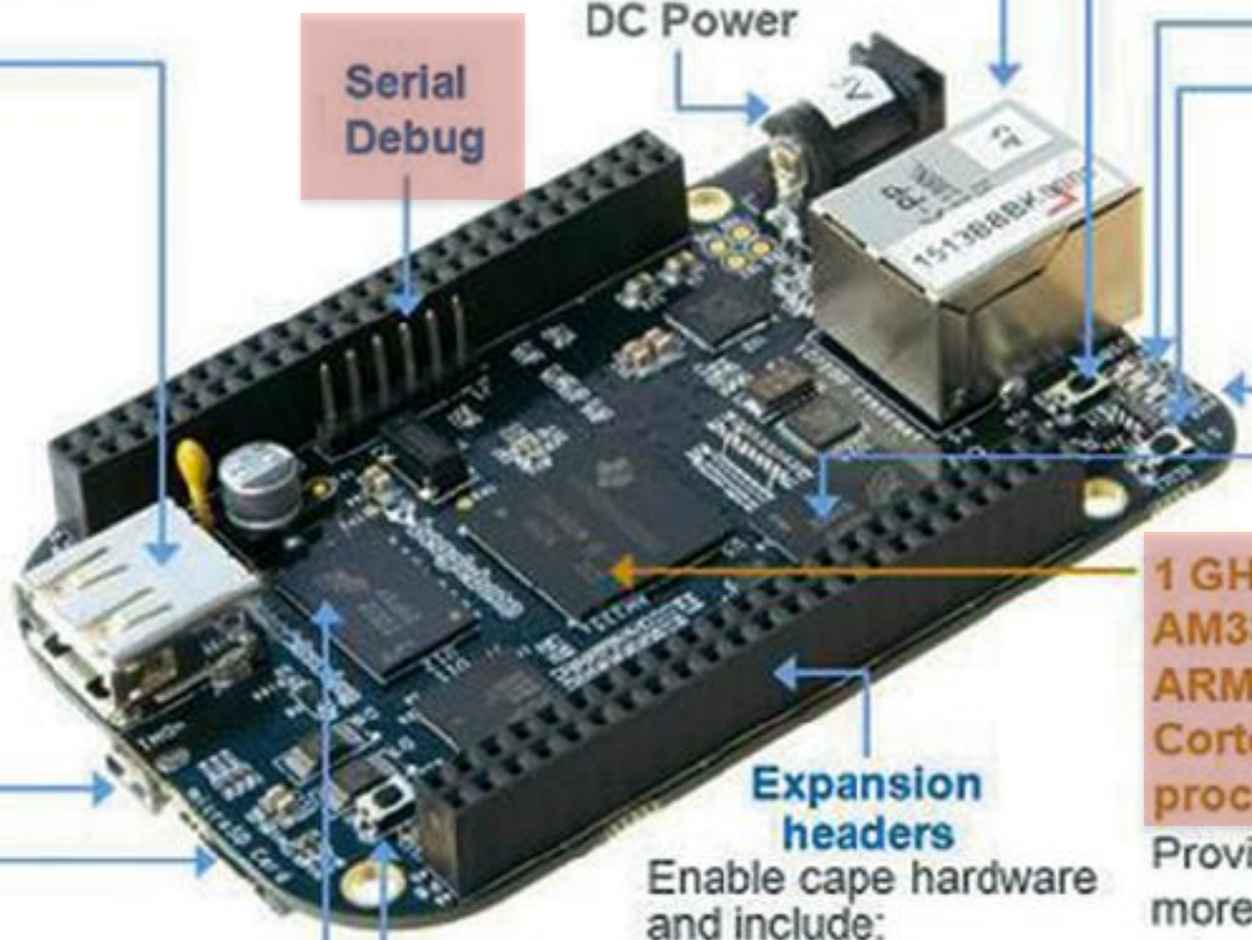
microSD
Expansion slot for additional storage

512MB DDR3
Faster, lower power RAM for enhanced user-friendly experience

Serial Debug

DC Power

Boot Button



Power Button

LEDS

Reset Button

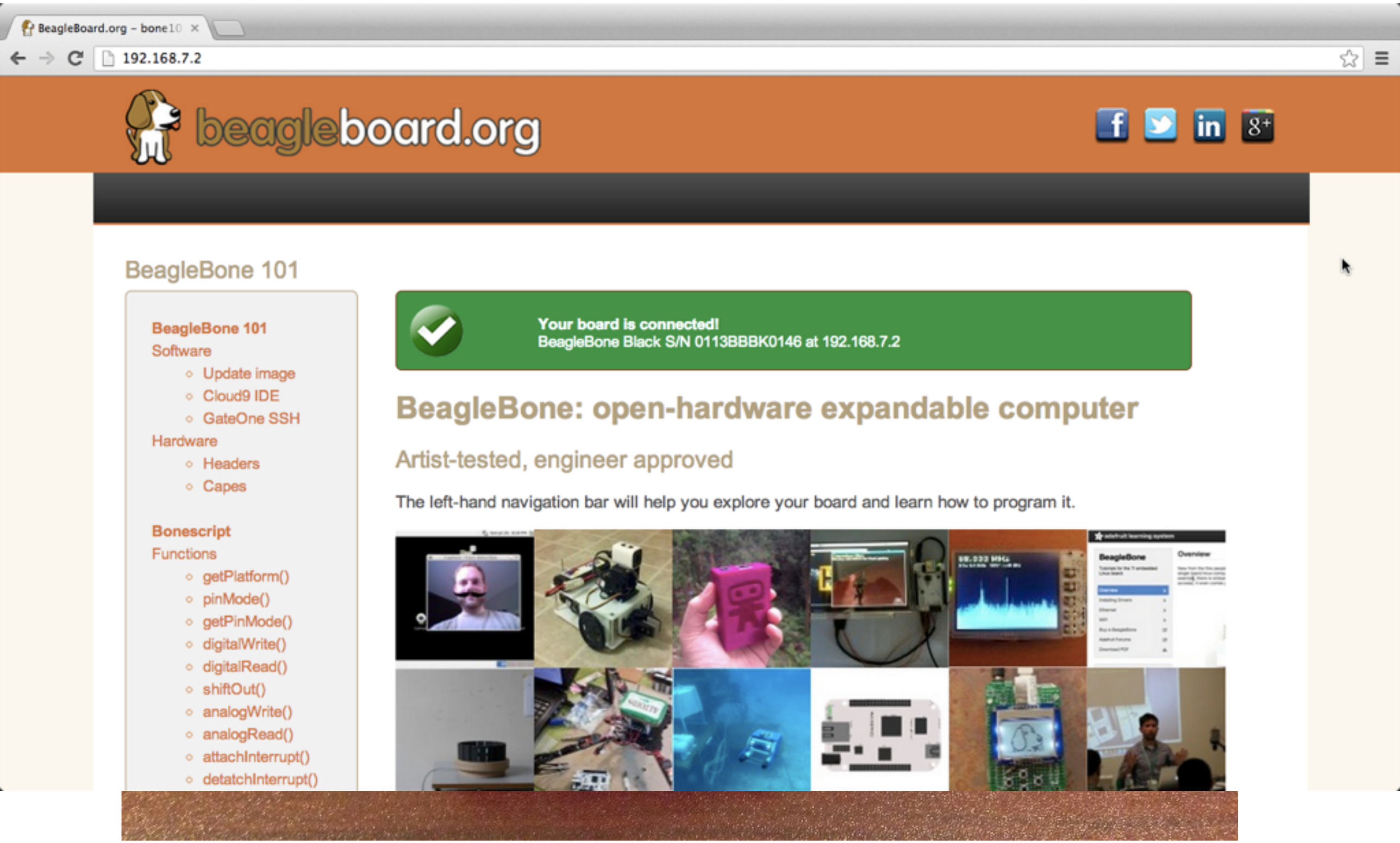
USB Client
Development interface and directly powers board from PC

2GB on-board storage using eMMC
- Pre-loaded with Ångström Linux Distribution
- 8-bit bus accelerates performance
- Frees the microSD slot to be used for additional storage for a less expensive solution than SD cards

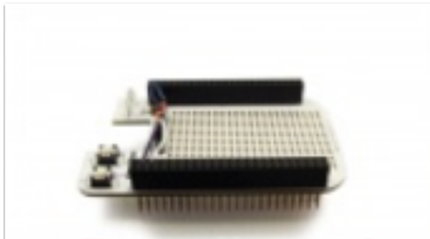
1 GHz Sitara AM335x ARM® Cortex™-A8 processor
Provides a more advanced user interface and up to 150% better performance than ARM11

Expansion headers
Enable cape hardware and include:
- 65 digital I/O
- 7 analog
- 4 serial
- 2 SPI
- 2 I2C
- 8 PWMs
- 4 timers
- And much much more!

Easy setup: USB & browser



Capes for I/O


Breadboard


Breakout

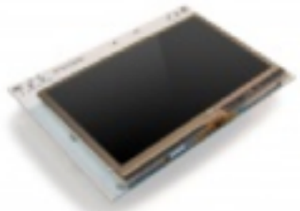

7” LCD


DVI-D


Can Bus


RS232


RS485


4.3”LCD

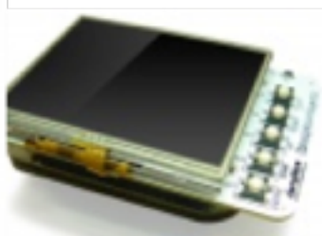

VGA


Battery


Profibus


Proto


RF-CC1101
CC2500
CC2530


3.5” LCD


Weather

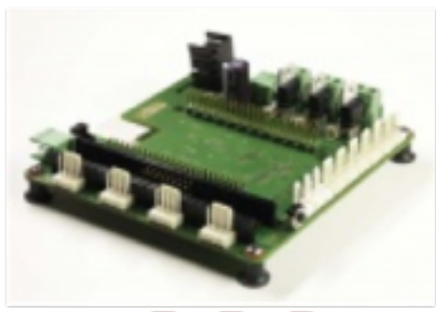

Camera


CAN


DVI-D w/
Audio


Audio

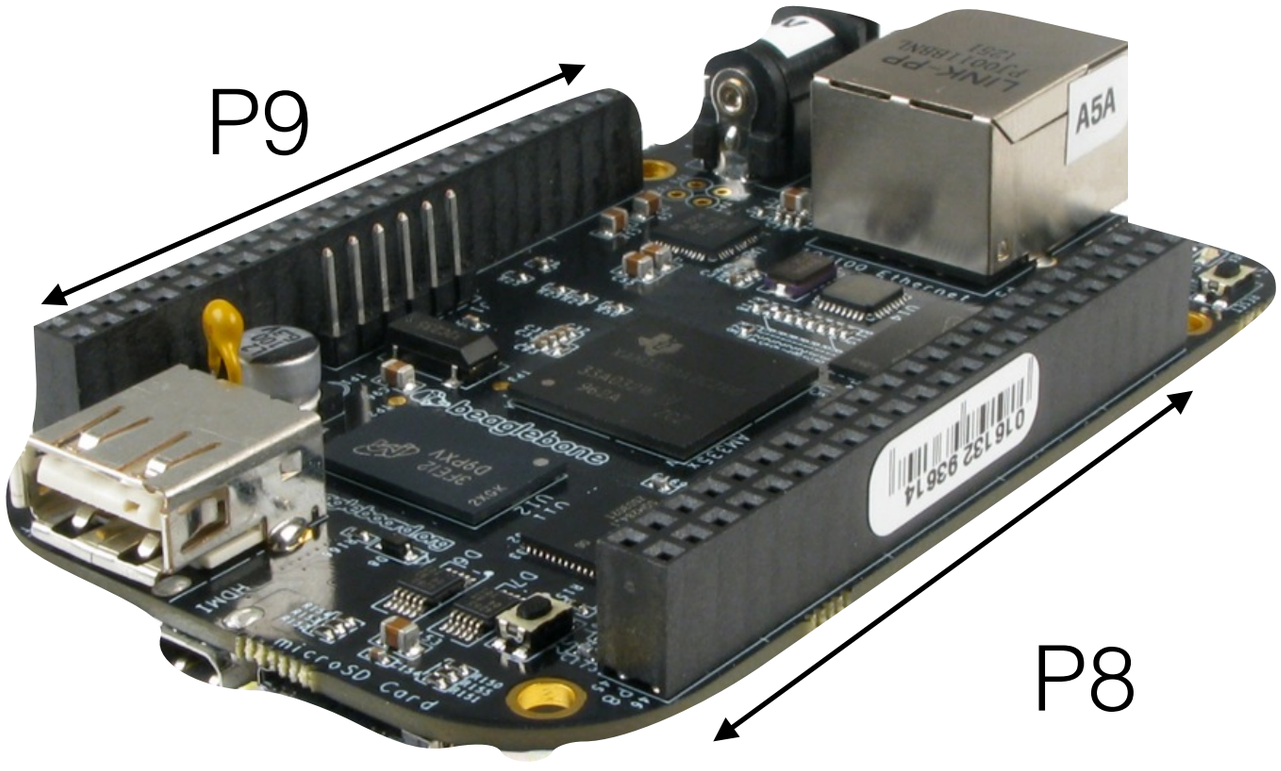

Radar


BeBoPr

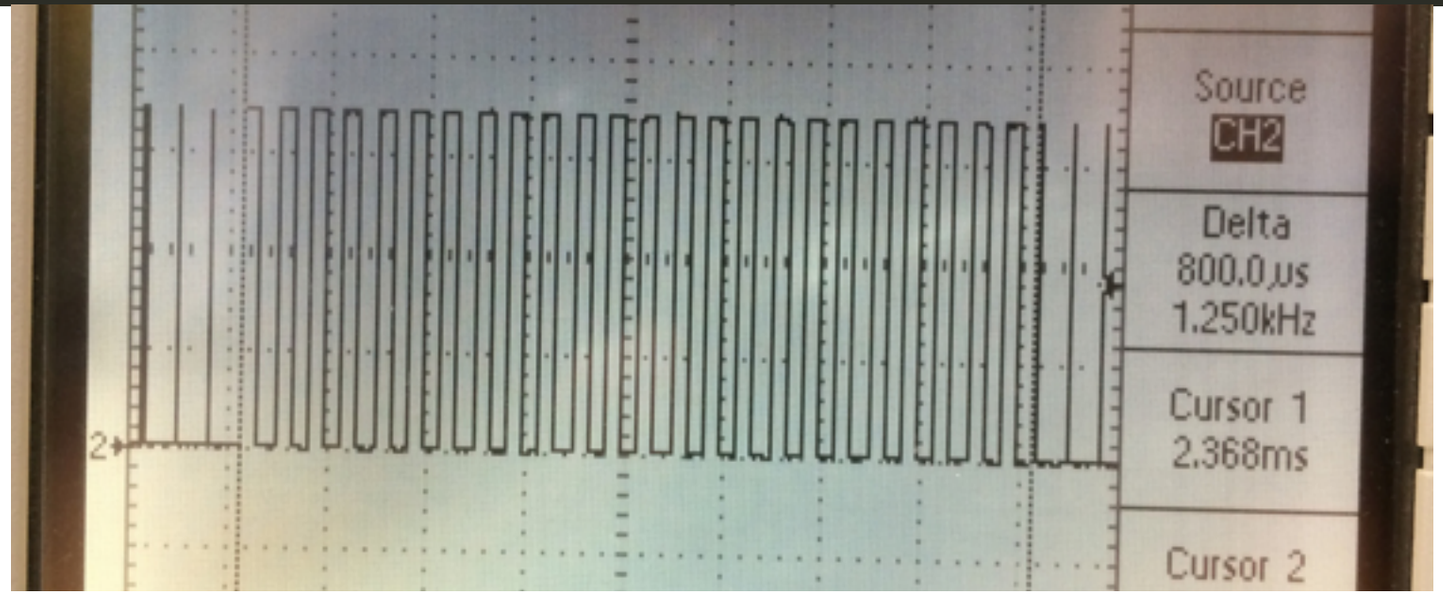

LVDS

BBB hw setup

- Exposed header pin rows: P8, P9
- Which pins do what?
- "Device tree overlays"



```
Loading the DTO created this sysfs entry:

    root@beaglebone:/lib/firmware# cd /sys/devices/ocp.3/pwm_test_P8_34.12/
    root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_34.12# ls -lstr
    total 0
    0 -rw-r--r-- 1 root root 4096 Apr 23 22:47 uevent
    0 lrwxrwxrwx 1 root root    0 Apr 23 22:47 subsystem -> ../../../bus/platform
    0 -rw------- 1 root root 4096 Apr 23 22:48 run
    0 drwxr-xr-x 2 root root    0 Apr 23 22:48 power
    0 -rw------- 1 root root 4096 Apr 23 22:48 polarity
    0 -r--r--r-- 1 root root 4096 Apr 23 22:48 modalias
    0 -rw------- 1 root root 4096 Apr 23 22:48 duty
    0 lrwxrwxrwx 1 root root    0 Apr 23 22:48 driver -> ../../../bus/platform/drivers/pwm_test
    0 -rw------- 1 root root 4096 Apr 23 22:48 period

Test it out:

    root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_34.12# echo 5000 > duty
    root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_34.12# echo 10000 > period
    root@beaglebone:/sys/devices/ocp.3/pwm_test_P8_34.12# echo 1 > run

The units are in nanoseconds, he says.

We observe the PWM on port 8 pin 34 on the scope, good.
```



Wiring for motor

\+

\-

6V lead-acid battery

3.3V (P9_3)

GND (P8_2)

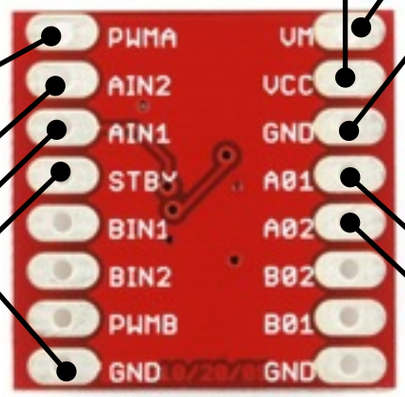
pwm

direction 2

direction 1

standby

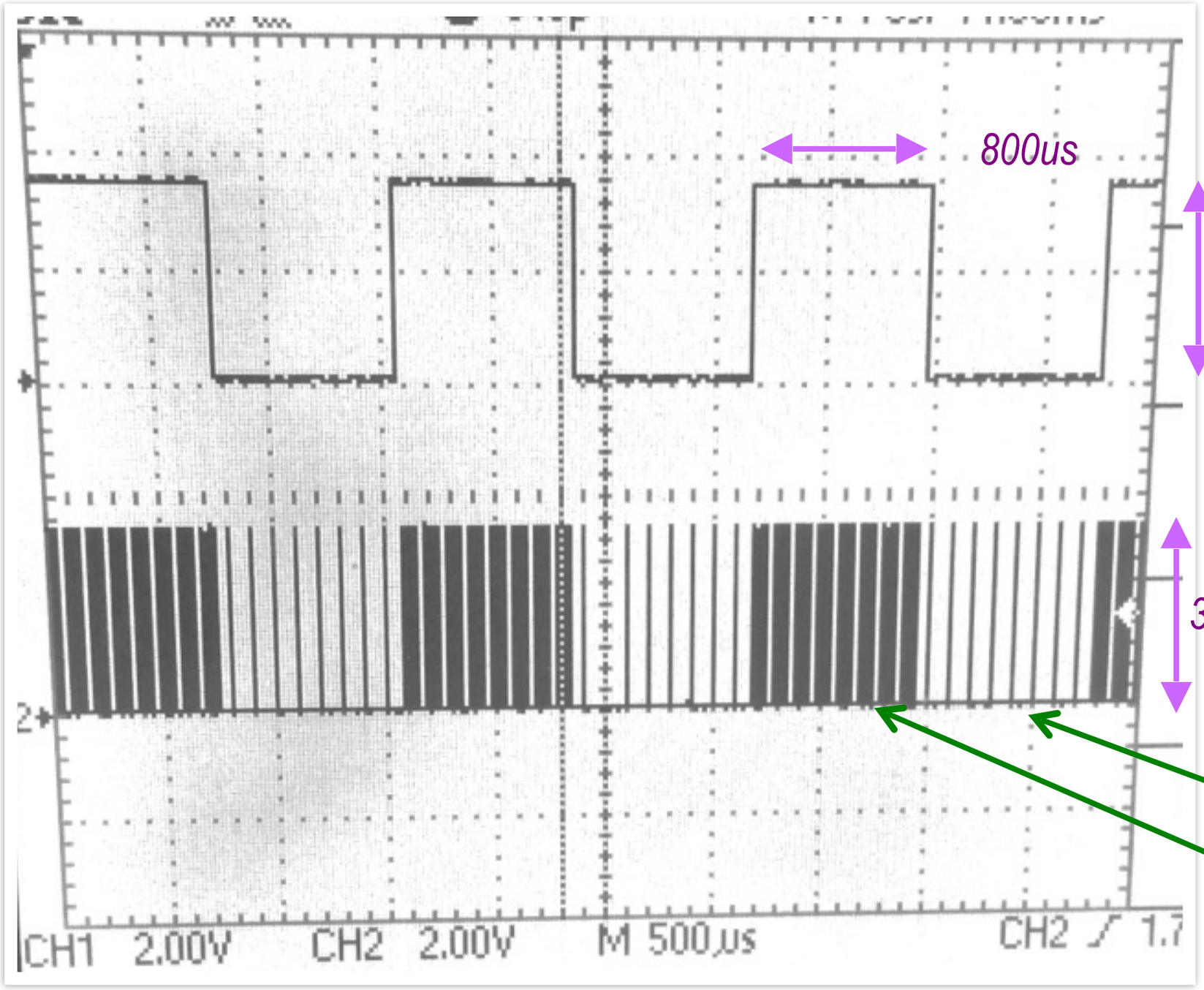
Beaglebone Black



SparkFun Motor Driver
(Dual TB6612FNG)
($35 on sparkfun)

Hobby Motor
(M260)

Pulse-width modulation to drive motor



motor direction
(cw, ccw)

motor speed
(PWM)

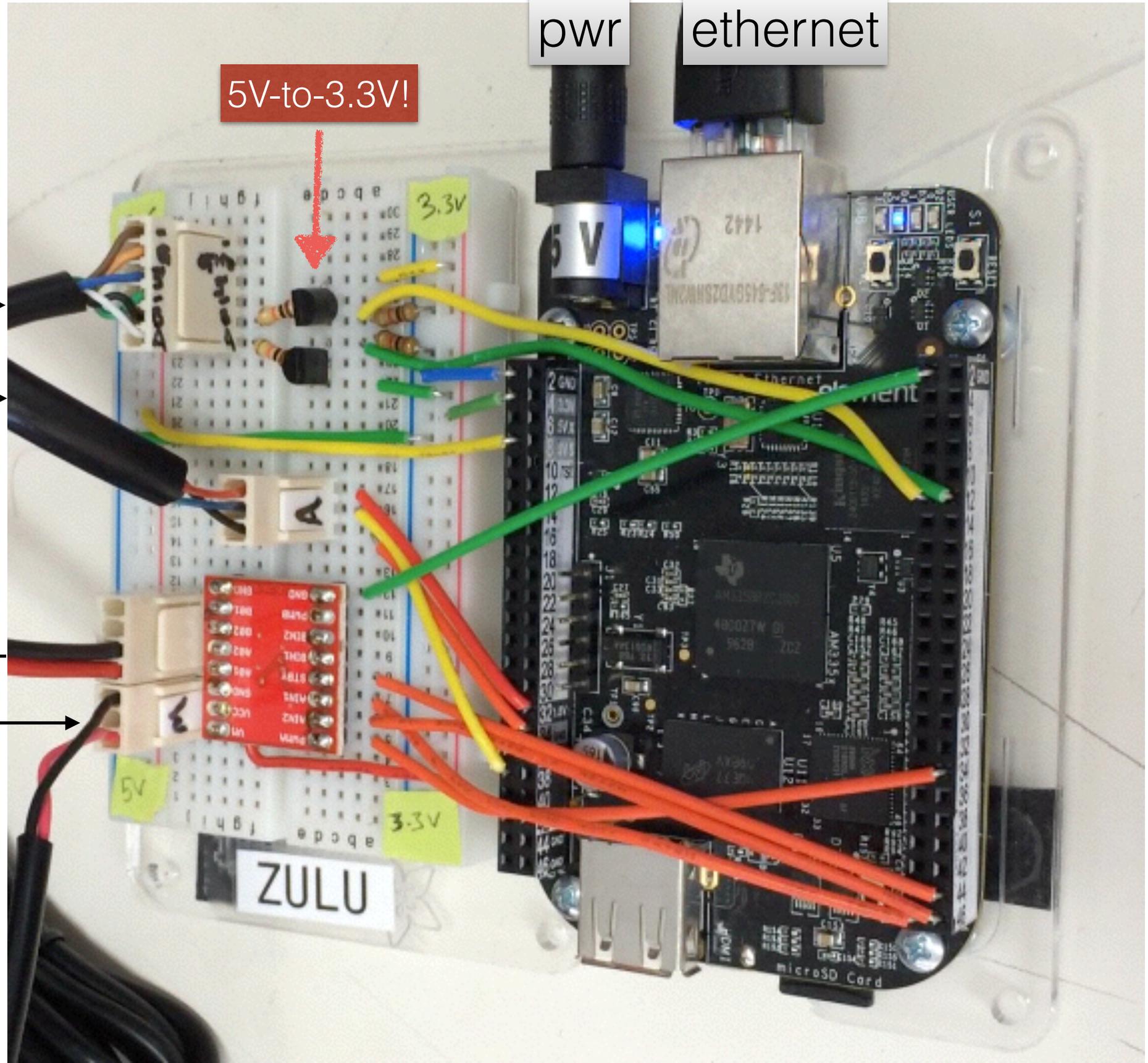
Beaglebone controller

pendulum encoder

arm pot.

to motor

motor pwr



```
# Configure I/O.

# - Pendulum measured by US Digital optical encoder that
#   outputs quadrature-encoded pulses, read by the eQEP module.
from eqep import eQEP  # Nathaniel Lewis's API
encoder = eQEP('/sys/devices/platform/ocp/48304000.epwmss/48304180.eqep',
                    eQEP.MODE_ABSOLUTE) # may need to change this path to suit

# - Actuator arm measured by 10kOhm potentiometer, read by the ADC.
# Beaglebone ADCs expect between 0 and 1.8V -- do not exceed.
import Adafruit_BBIO.ADC as ADC
ADC.setup()

# - Motor actuated by a Sparkfun motor driver that
# expects PWM signal and 2 digital outputs for direction.
import Adafruit_BBIO.PWM as PWM
MY_PWM_PIN = 'P8_34'

import Adafruit_BBIO.GPIO as GPIO
MY_PWM_DIR_PIN_1 = 'P8_45'
MY_PWM_DIR_PIN_2 = 'P8_46'
MY_PWM_STBY = 'P8_44'

GPIO.setup(MY_PWM_DIR_PIN_1, GPIO.OUT)
GPIO.setup(MY_PWM_DIR_PIN_2, GPIO.OUT)
GPIO.setup(MY_PWM_STBY, GPIO.OUT)

GPIO.output(MY_PWM_DIR_PIN_1, GPIO.LOW)
GPIO.output(MY_PWM_DIR_PIN_2, GPIO.LOW)


# start PWM
PWM.start(MY_PWM_PIN,0,50000) # duty cycle 0%, freq 1000HZ.
GPIO.output(MY_PWM_STBY, GPIO.HIGH)
```

read

```
adc = ADC.read("AIN2")
if debug_adc:
        print('Raw ADC: {}'.format(adc))

arm_rad = (adc-0.5)*2.0*pi
# ADC.read returns a number in [0,1]
# We've wired it so that 50% is "arm is straight ahead (0 deg)"


pend_rad = encoder.get_position()*2.0*pi/4096.0
# 2^12 = 4096 lines per rev
```

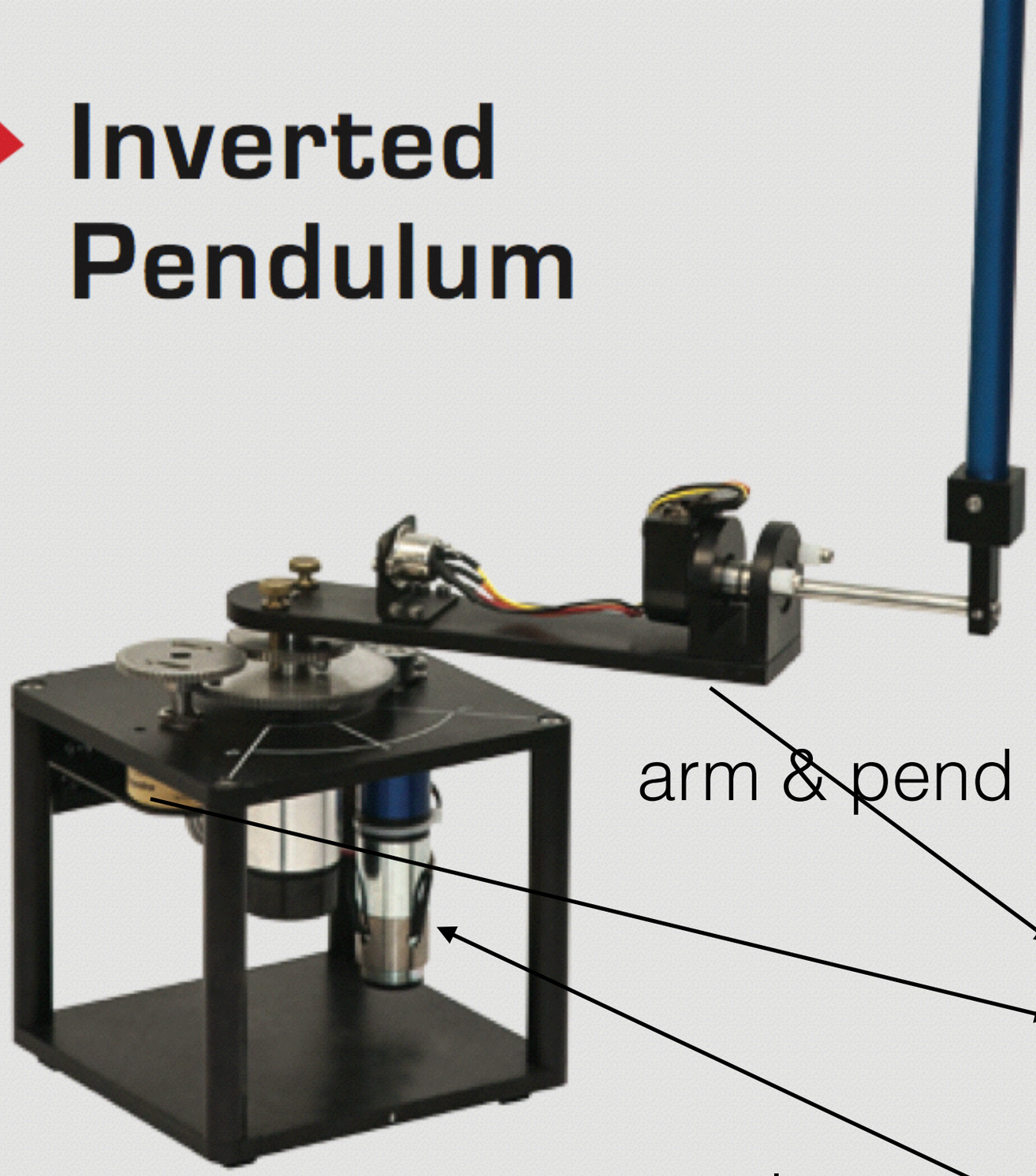
write

```
def write_motor_voltage(v):
    if v>0:
            # motor counter-clockwise -> arm cw
            GPIO.output(MY_PWM_DIR_PIN_1, GPIO.LOW)
            GPIO.output(MY_PWM_DIR_PIN_2, GPIO.HIGH)
    else:
            # motor clockwise -> arm cw
            GPIO.output(MY_PWM_DIR_PIN_1, GPIO.HIGH)
            GPIO.output(MY_PWM_DIR_PIN_2, GPIO.LOW)

    duty = abs(v)/5.0*100.0  # percentage, e.g., 15 not 0.15

    if duty>100:
        duty = 100
    elif duty<0:
        duty = 0

    PWM.set_duty_cycle(MY_PWM_PIN,duty)
```

Outline

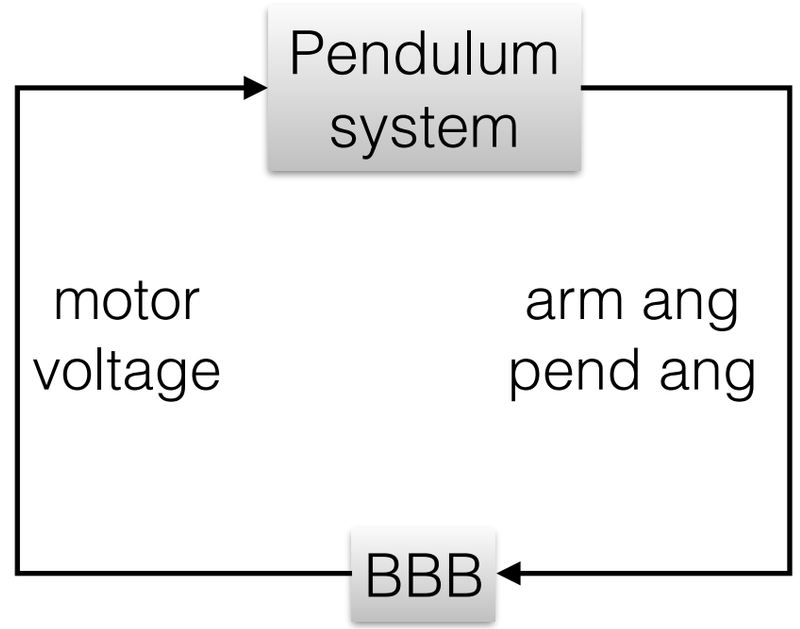
- Omg Demo
- Pendulum
- Beaglebone
- **Control Theory**

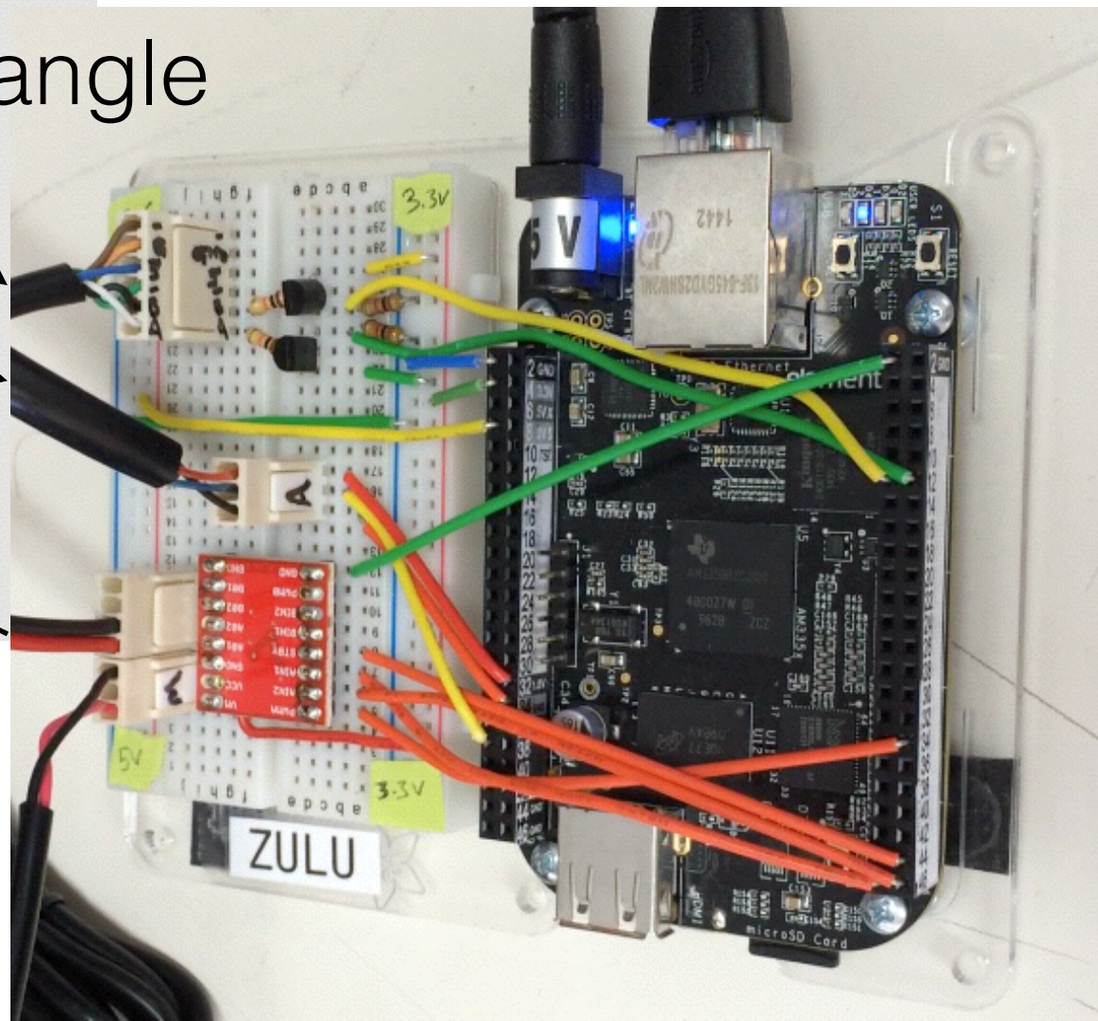
Inverted Pendulum



arm & pend angle

motor voltage





Linear Systems

A discrete-time linear time-invariant dynamical system is a set of matrix equations of the form

state input

$$x_{i+1} = Ax_i + Bu_i$$
$$y_i = Cx_i + Du_i$$

$$i \in \mathbb{N}_{\geq 0}$$

output

$$x_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}^m,\ u_i \in \mathbb{R}^r$$

$$x := \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}$$ arm ang, arm vel, pend ang, pend vel

u motor voltage

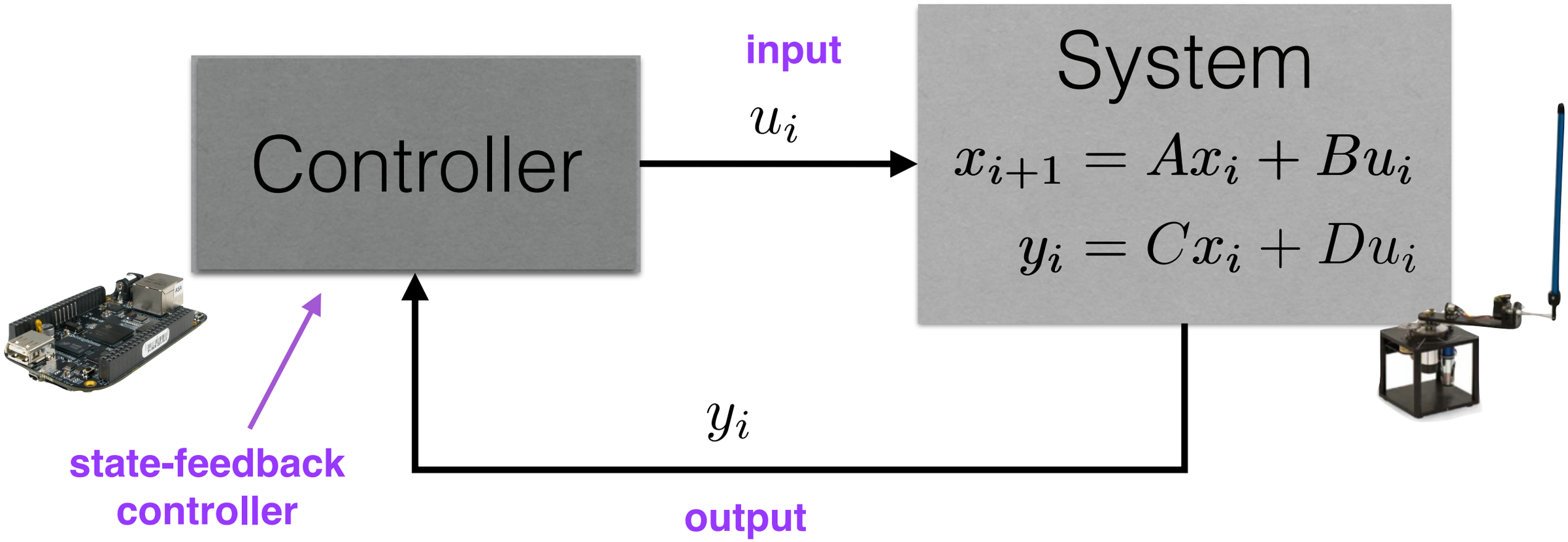
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$ only read arm ang and pend ang

Specifies a (vector) recursion

Given x_0 and {u}, we can calculate {x} and {y}

Control Theory: Observing {y}, pick {u} to make {x} “good”

State-feedback controllers are easy to design, analyze, and implement



$$u_i = \begin{bmatrix} 1 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}_i$$

$$x := \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}$$ arm ang, arm vel, pend ang

u: motor voltage

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$ only read arm ang and pend ang

Will this K stabilize the system?

State-feedback controllers are easy to design, analyze, and implement

$$x_{i+1} = Ax_i + Bu_i$$
$$y_i = Cx_i + Du_i$$

Suppose D=0 and C is identity, so we measure x directly.

By choosing input

$$u_i := Kx_i$$

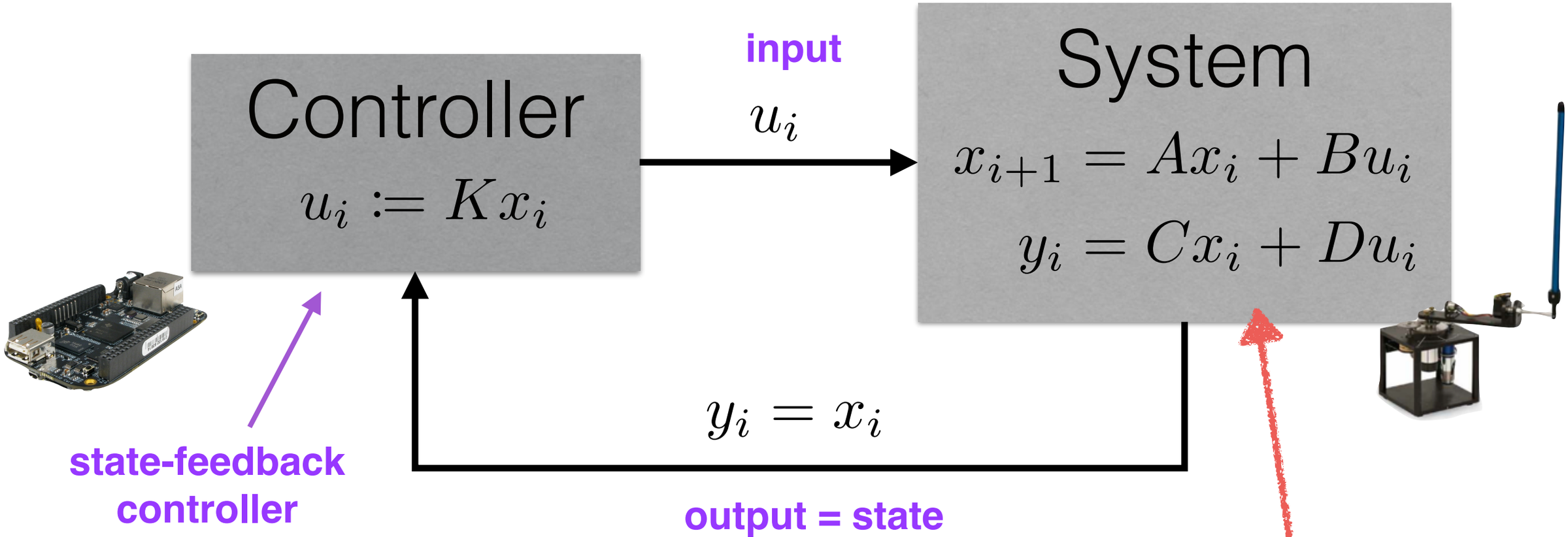
we have

$$u_i = \begin{bmatrix} 1 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}_i$$

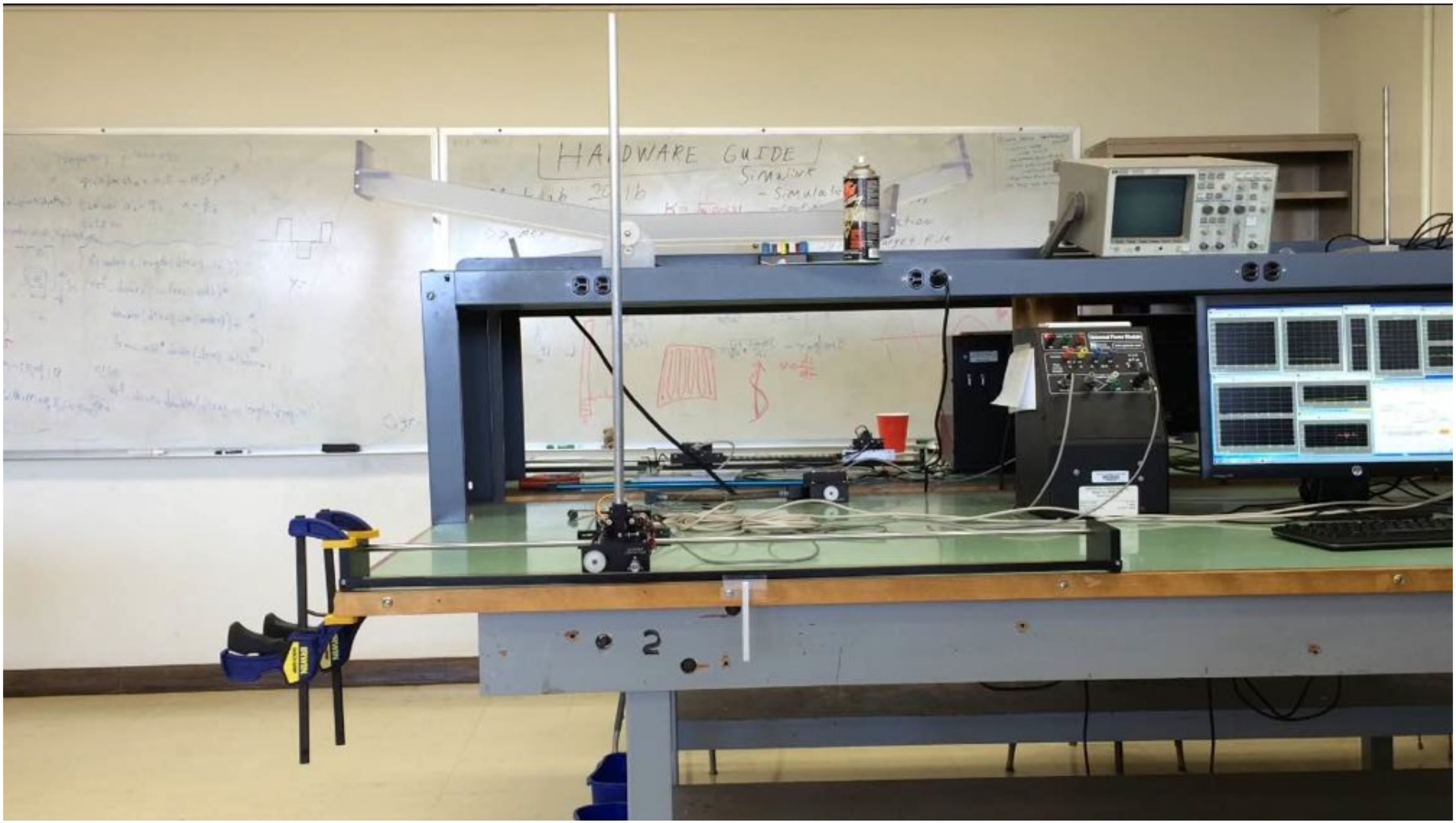
$$x_{i+1} = (A + BK)x_i \quad \Leftrightarrow \quad x_i = (A + BK)^i x_0$$

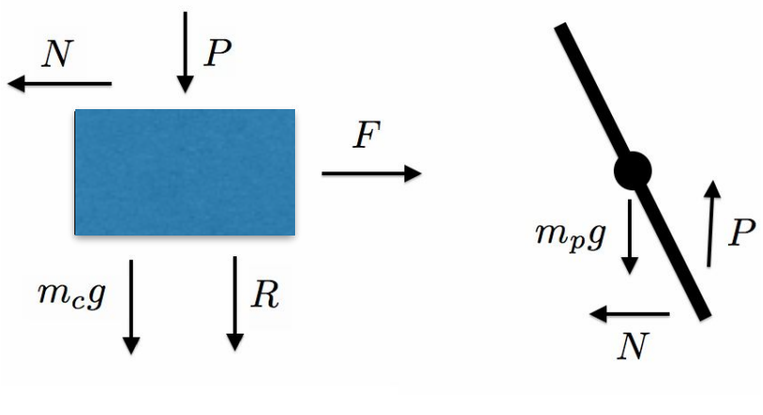
Pick K for which $(A+BK)^i \to 0$

State-feedback controllers are easy to design, analyze, and implement



How to express the pendulum system like this?


HARDWARE GUIDE
2



Free body diagram

Equations of motion

$$m_c\ddot{p} = F - N$$

$$I\ddot{\theta} = Plsin(\theta) - Nlcos(\theta)$$

$$F = \frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p}$$

$$N = m_p(\ddot{p} + lcos(\theta)\ddot{\theta} - lsin(\theta)\dot{\theta}^2)$$

$$P = m_p g + m_p l(-sin(\theta)\ddot{\theta} - cos(\theta)\dot{\theta}^2)$$

Core idea

Express

Equations of Motion

$$\ddot{p}(M - \frac{m_p l cos^2(\theta)}{L}) = \frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p} - \frac{m_p l g}{L} cos(\theta) sin(\theta) + m_p l sin(\theta) \dot{\theta}^2$$

$$\ddot{\theta}(L - \frac{m_p l cos^2(\theta)}{M}) = g sin(\theta) - \frac{m_p l \dot{\theta}^2}{M} cos(\theta) sin(\theta) - \frac{cos(\theta)}{M}(\frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p})$$

as a discrete-time linear time-invariant system

$$x_{i+1} = Ax_i + Bu_i$$
$$y_i = Cx_i + Du_i$$

$$\text{Let } x := \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}$$

$$u := V$$

then find matrix K satisfying

$$(A + BK)^i \to 0$$

so that the control law $u_i := Kx_i$ causes $x_i \to 0$

Simplify

Equations of Motion

$$\ddot{p}(M - \frac{m_p l cos^2(\theta)}{L}) = \frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p} - \frac{m_p l g}{L} cos(\theta) sin(\theta) + m_p l sin(\theta) \dot{\theta}^2$$

$$\ddot{\theta}(L - \frac{m_p l cos^2(\theta)}{M}) = g sin(\theta) - \frac{m_p l \dot{\theta}^2}{M} cos(\theta) sin(\theta) - \frac{cos(\theta)}{M}(\frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p})$$

Constants: $M, m_p, L, l, K_m, K_g, Rr, g$

Functions of time: $p(t), \theta(t), V(t)$

$$p''(1 - \cos(\theta)^2) = V - p' - \cos(\theta)\sin(\theta) + \sin(\theta)\theta'^2$$

$$\theta''(1 - \cdot) = \cdot$$

2nd-order to 1st-order vector eqn

$$p''(1-\cos(\theta)^2) = V - p' - \cos(\theta)\sin(\theta) + \sin(\theta)\theta'^2$$

$$\theta''(1-\cdot) = \cdot$$

$$\text{Let } x := \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix}, \text{ so } x' := \begin{bmatrix} p' \\ p'' \\ \theta' \\ \theta'' \end{bmatrix} = \begin{bmatrix} p' \\ \frac{V - p' - \cos(\theta)\sin(\theta) + \sin(\theta)\theta'^2}{(1-\cos(\theta)^2)} \\ \theta' \\ \cdot \end{bmatrix}$$

$$u := V$$

$$\boxed{x' = f(x) + \begin{bmatrix} 0 \\ u \\ 0 \\ \cdot \end{bmatrix}}$$

$$= \begin{bmatrix} x_2 \\ \frac{u - x_2 - \cos(x_3)\sin(x_3) + \sin(x_1)x_4^2}{(1-\cos(x_3)^2)} \\ x_3 \\ \cdot \end{bmatrix}$$

f(x)

Linearize

$$x' = f(x) + \begin{bmatrix} 0 \\ u \\ 0 \\ \cdot \end{bmatrix}$$

The general equation for the linearization of a multivariable function $f(\mathbf{x})$ at a point $\mathbf{p}$ is:

$$f(\mathbf{x}) \approx f(\mathbf{p}) + \nabla f|_{\mathbf{p}} \cdot (\mathbf{x} - \mathbf{p})$$

where $\mathbf{x}$ is the vector of variables, and $\mathbf{p}$ is the linearization point of interest .[2]

$$p := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Mathematica File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

Untitled-1

```
In[13]:= D[{x2, (-x2 - Cos[x3] Sin[x3] + Sin[x3] x4^2)/(1 - .2 Cos[x3]^2), x3}, {{x1, x2, x3, x4}}] // MatrixForm
```

Out[13]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{1-0.2\,\mathrm{Cos}[x3]^2} & -\frac{0.4\,\mathrm{Cos}[x3]\,\mathrm{Sin}[x3]\left(-x2+x4^2\,\mathrm{Sin}[x3]-\mathrm{Cos}[x3]\,\mathrm{Sin}[x3]\right)}{\left(1-0.2\,\mathrm{Cos}[x3]^2\right)^2}+\frac{x4^2\,\mathrm{Cos}[x3]-\mathrm{Cos}[x3]^2+\mathrm{Sin}[x3]^2}{1-0.2\,\mathrm{Cos}[x3]^2} & \frac{2\,x4\,\mathrm{Sin}[x3]}{1-0.2\,\mathrm{Cos}[x3]^2} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

```
In[14]:= % /. {x1 → 0, x2 → 0, x3 → 0, x4 → 0} // MatrixForm
```

Out[14]//MatrixForm=

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1.25 & -1.25 & 0. \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Linearize $x' = f(x) + \begin{bmatrix} 0 \\ u \\ 0 \\ \cdot \end{bmatrix}$ about $p := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$$f(\mathbf{x}) \approx f(\mathbf{p}) + \nabla f|_{\mathbf{p}} \cdot (\mathbf{x} - \mathbf{p})$$

0 0

$$A := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1.25 & -1.25 & 0. \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B := \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$x' = f(x) + \begin{bmatrix} 0 \\ u \\ 0 \\ 0 \end{bmatrix} \approx Ax + Bu$$

Core idea

Express

Equations of Motion

$$\ddot{p}(M - \frac{m_p l cos^2(\theta)}{L}) = \frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p} - \frac{m_p l g}{L} cos(\theta) sin(\theta) + m_p l sin(\theta) \dot{\theta}^2$$

$$\ddot{\theta}(L - \frac{m_p l cos^2(\theta)}{M}) = g sin(\theta) - \frac{m_p l \dot{\theta}^2}{M} cos(\theta) sin(\theta) - \frac{cos(\theta)}{M}(\frac{K_m K_g}{Rr} V - \frac{K_m^2 K_g^2}{Rr^2} \dot{p})$$

as a discrete-time linear time-invariant system

$$x_{i+1} = Ax_i + Bu_i$$
$$y_i = Cx_i + Du_i$$

$$\text{Let } x := \begin{bmatrix} p \\ p' \\ \theta \\ \theta' \end{bmatrix} \quad u := V$$

$$A := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -1.25 & -1.25 & 0. \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B := \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

then find matrix K satisfying

$$(A + BK)^i \to 0$$

so that the control law $u_i := Kx_i$ causes $x_i \to 0$

place

Pole placement design

```
p = [-1 -1.23 -5.0];
K = place(a,b,p)
```

Syntax

```
K = place(A,B,p)
[K,prec,message] = place(A,B,p)
```

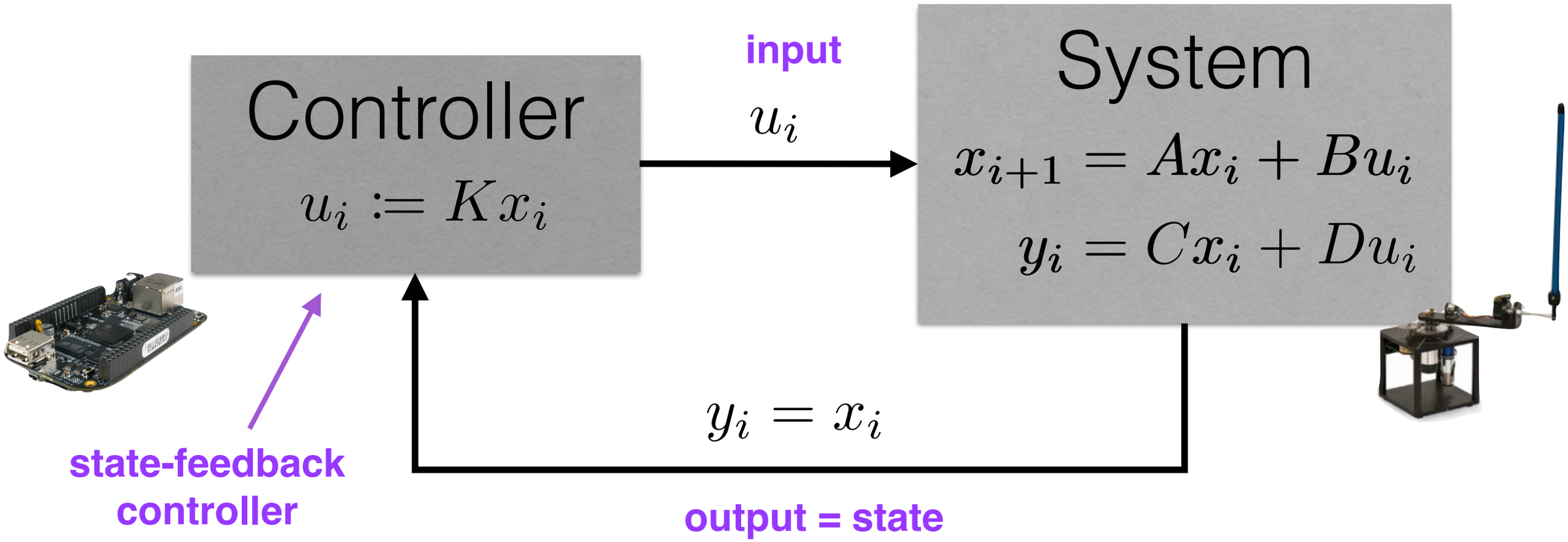
Description

Given the single- or multi-input system

$$\dot{x} = Ax + Bu$$

and a vector p of desired self-conjugate closed-loop pole locations, `place` computes a gain matrix K such that the state feedback $u = -Kx$ places the closed-loop poles at the locations p. In other words, the eigenvalues of $A - BK$ match the entries of p (up to the ordering).

State-feedback controllers are easy to design, analyze, and implement




```
        # state feedback control law
        Klqr = numpy.array([  -2.206024501362314  , -2.158429674965489 ,
        self.motor_voltage = numpy.dot(-Klqr,self.sys.x)[0] # get the el
```

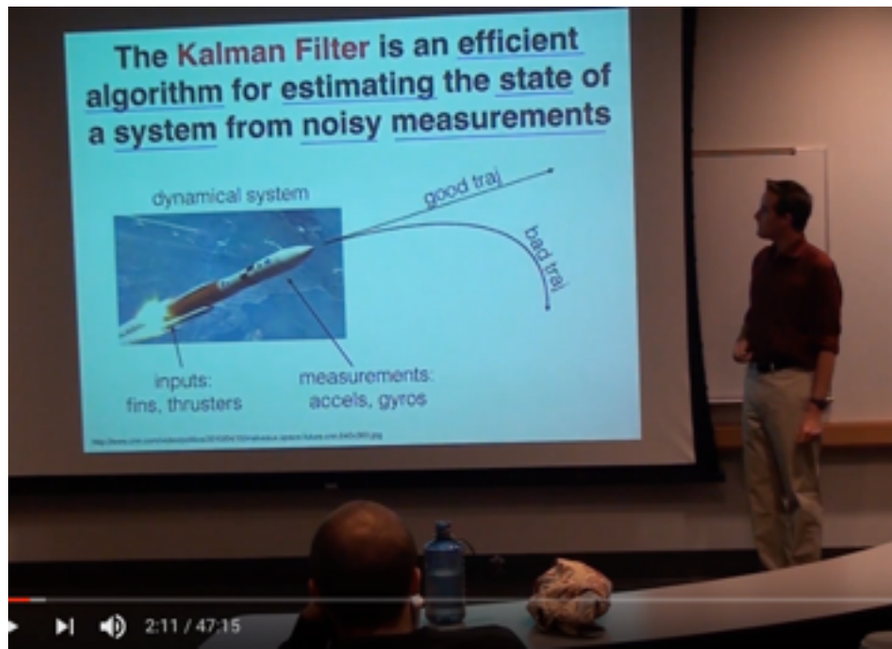
Recap: state-feedback conrol

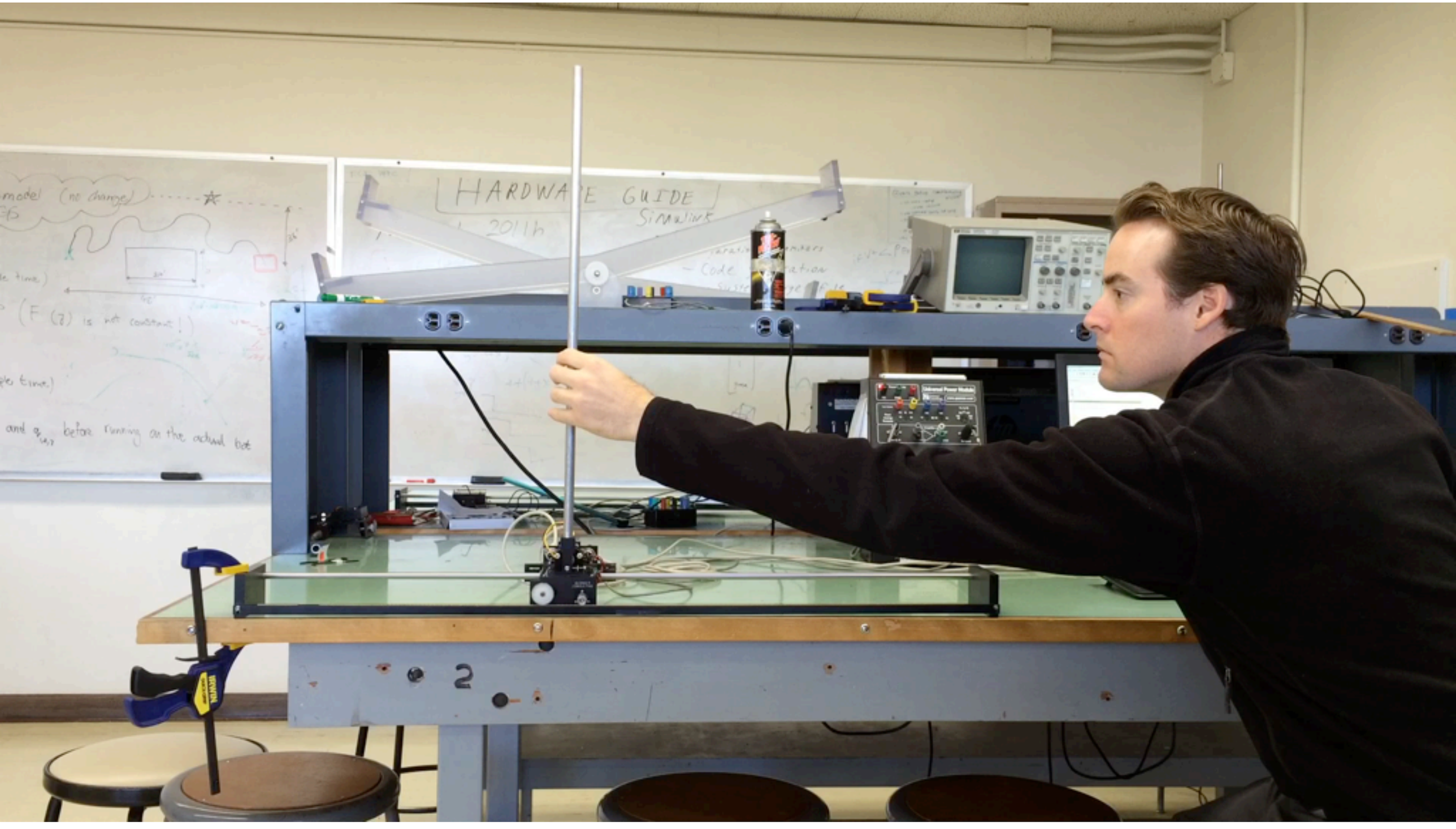
- Equations of motion
- Convert to vector 1st-order
- Linearize system
- Compute state-feedback gain K

Not pictured:

- *cts-to-discrete*
- *state estimation*

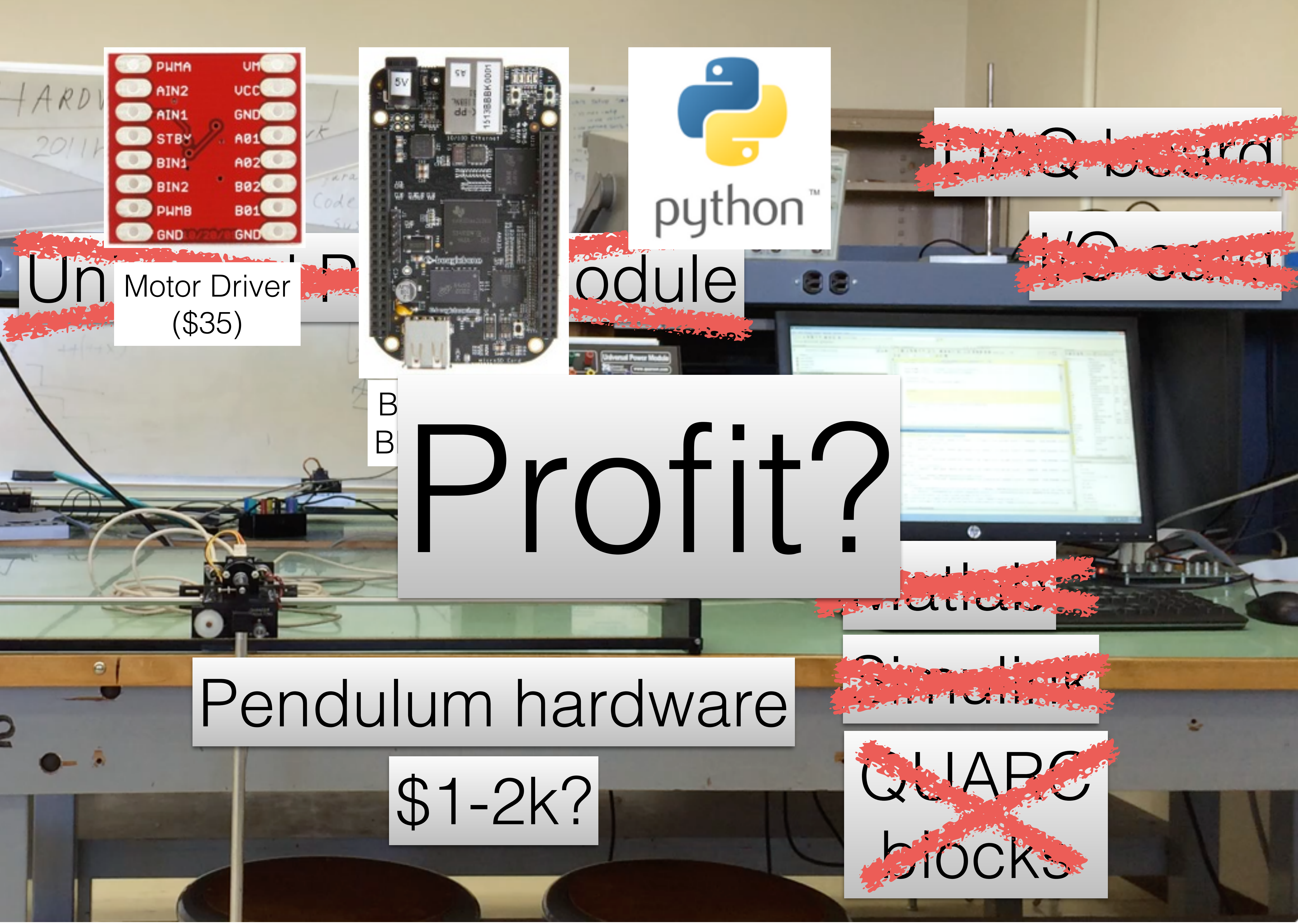
YouTube user "funskits"



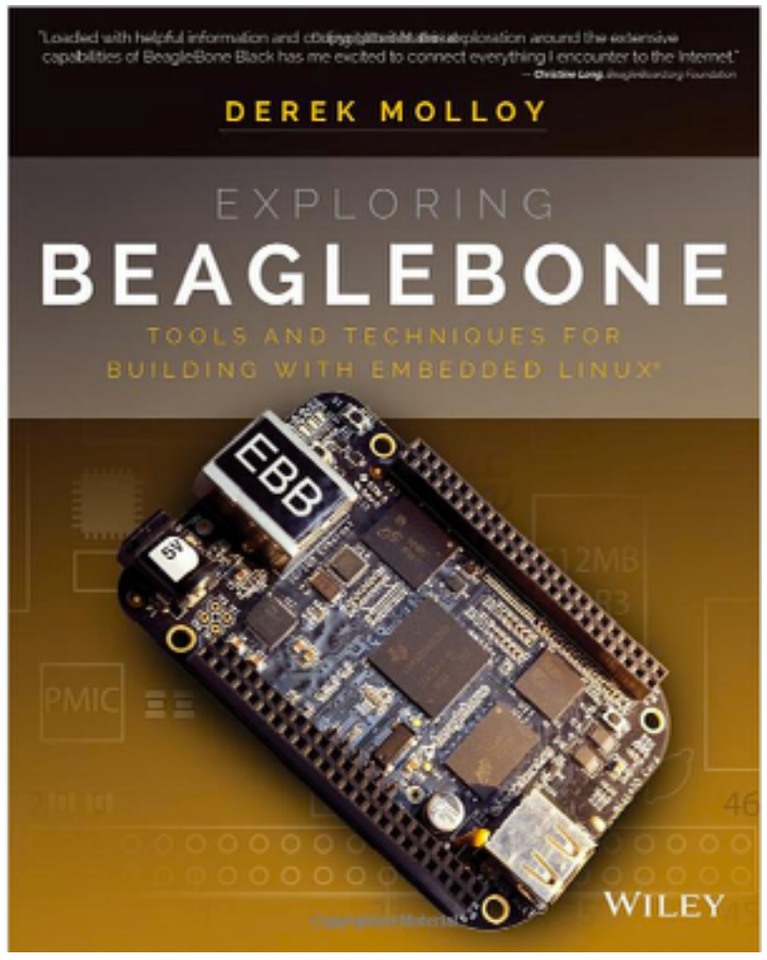

HARDWARE GUIDE
Simulink
2011b
2

Summary

- Omg Demo
- Pendulum
- Beaglebone
- Control Theory


Motor Driver
($35)
python
Profit?
Pendulum hardware
$1-2k?
QUARC blocks

Thanks

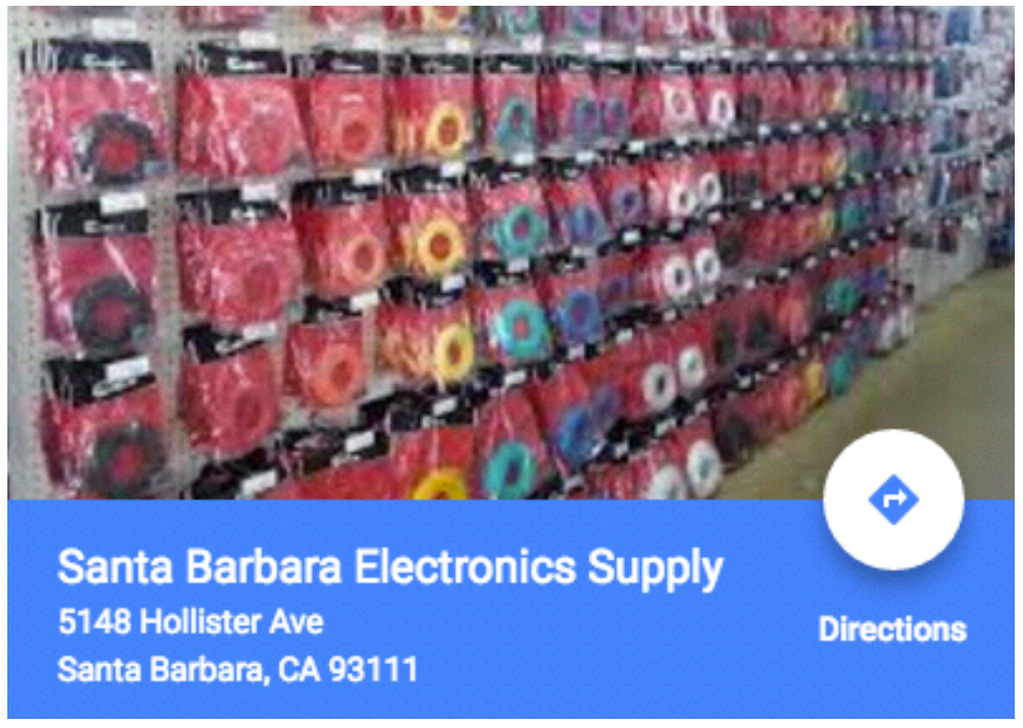


Derek

Andrew Symington

Caio Motta

Calvin Wan



Andrew, Brian

END