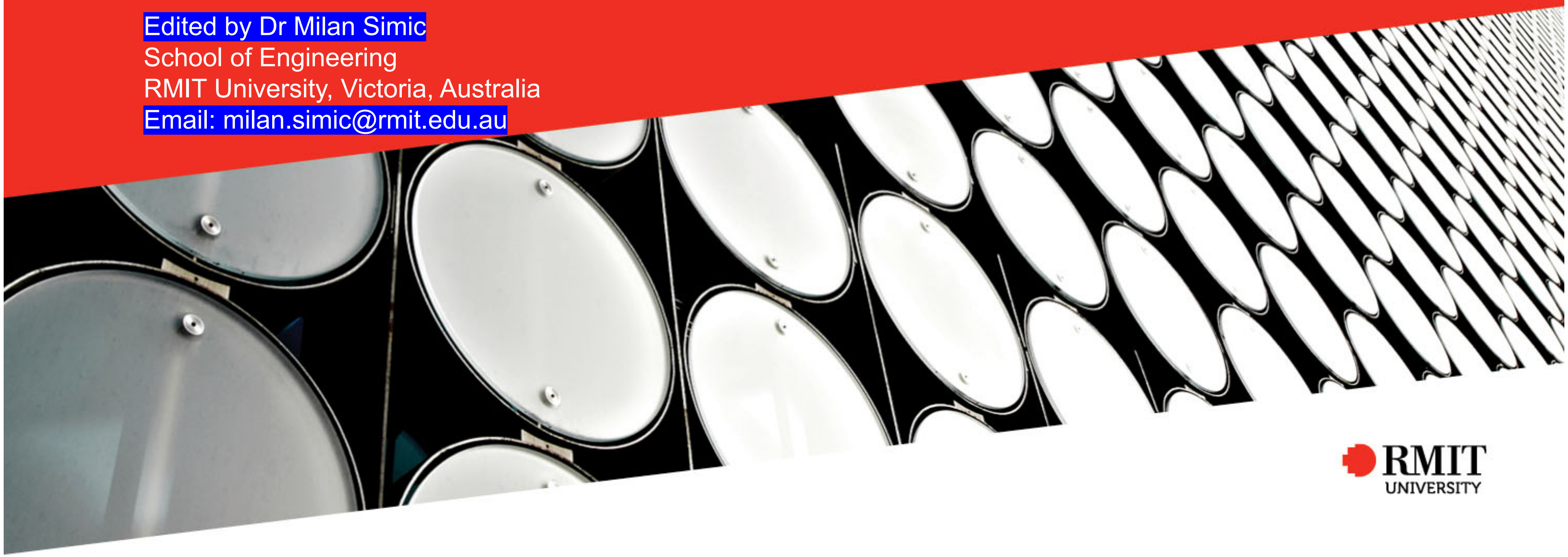
Week 8 – DC Motor Control Implementation

Advanced Mechatronics System Design – MANU2451

Dr Chow Yin LAI
Edited by Dr Milan Simic
School of Engineering
RMIT University, Victoria, Australia
Email: milan.simic@rmit.edu.au

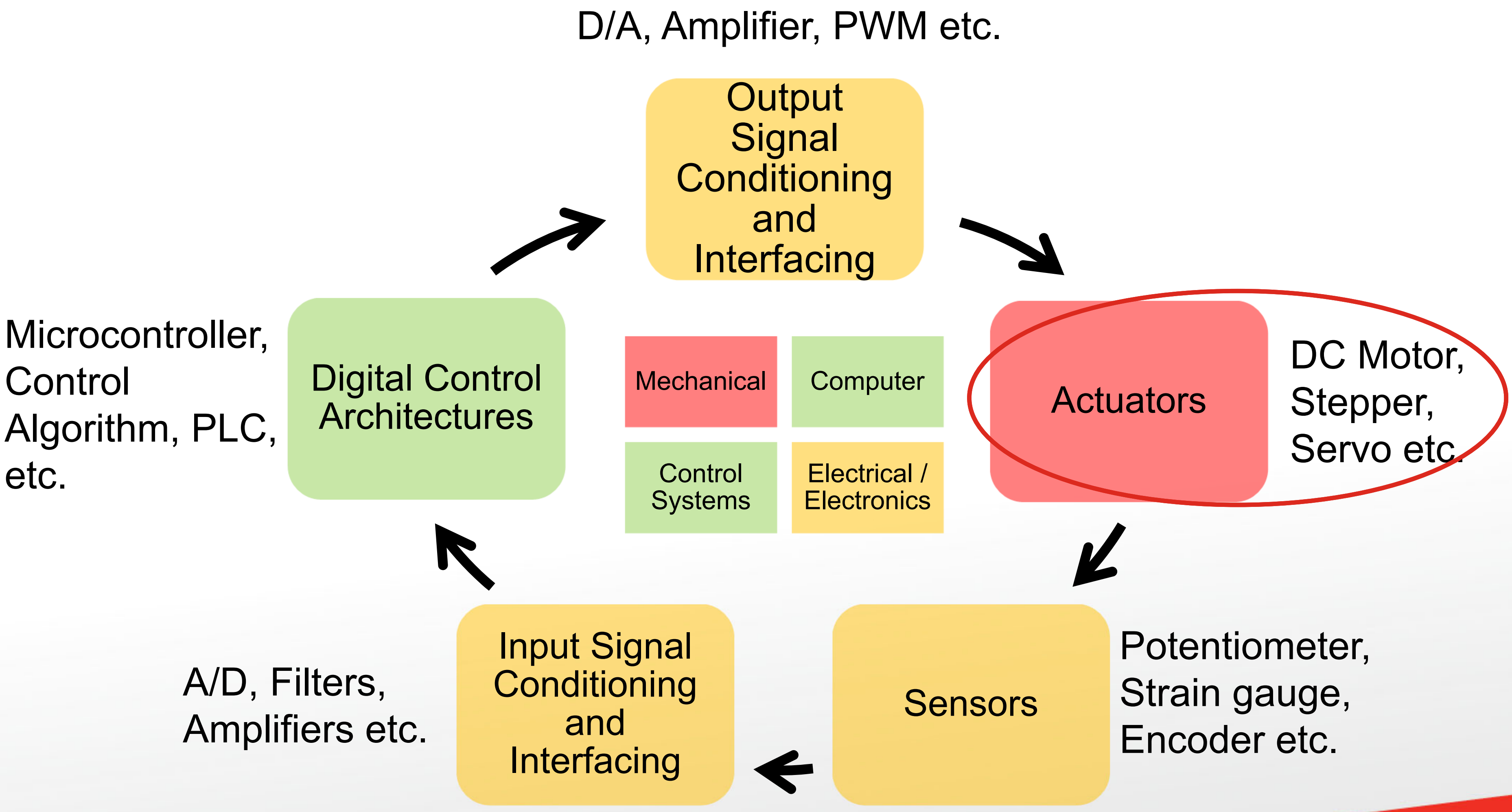


RMIT UNIVERSITY

New Teaching Schedule

Week		Class Activity Before	Lecture	Class Activity During or After
1			Introduction to the Course / Introduction to LabVIEW	LabVIEW Programming
2			Introduction to LabVIEW / Data Acquisition	LabVIEW Programming
3			Gripper / Introduction to Solidworks / Safety	Gripper Design
4			Sensors I	myRIO Programming for Sensor Signal Reading / Gripper Design
5			Sensors II	myRIO Programming for Sensor Signal Reading
6			Actuators I	LabVIEW Tutorial
7		LabVIEW Assessment.	DC Motors I	Matlab Simulink Simulation
8		Design report submission	DC Motors II	Matlab Simulink Simulation / myRIO Programming for Control
9			Actuators II	Matlab Simulink Simulation Gripper CAD
10			Modeling and System Identification	Matlab Simulink Simulation / Gripper simulation testing
11			Artificial Intelligence I	Matlab Simulation / Finalize Gripper
12		Gripper Simulation / Submission of Report	Artificial Intelligent II	Revision

Mechatronics System Components



Mechatronics System Components

Sensors:
Encoder at each joint

→ Input signal interfacing

↓

Robot controller:

- Generate desired motion trajectory
- Calculate current end-effector position based on angular position (kinematics)
- Calculate desired angular position for desired end-effector position and trajectory (inverse kinematics)
- Control algorithm
- Safety, collision detection etc.

↓

Output signal interfacing

←

Actuators:
Geared motor at each joint

Industrial Robots
https://commons.wikimedia.org/wiki/File:Float_Glass_Unloading.jpg

RMIT UNIVERSITY

Content

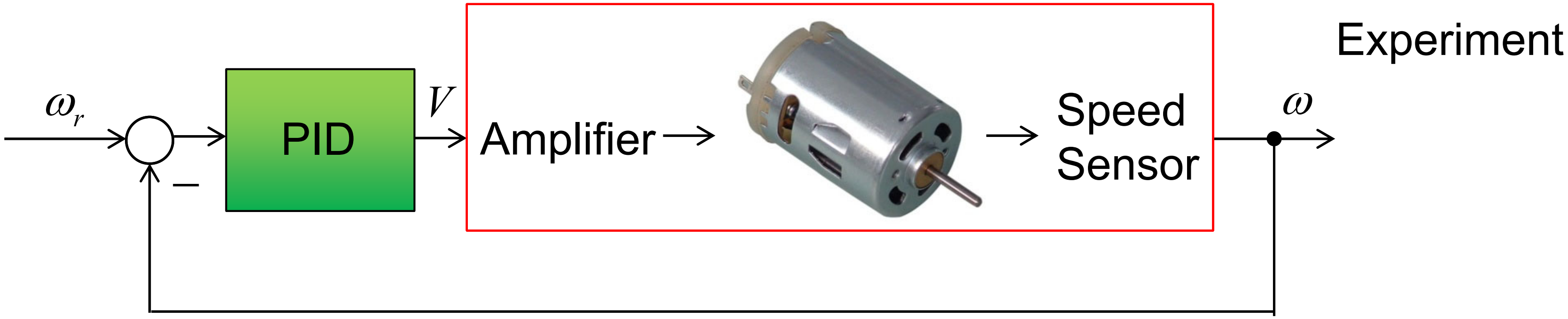
- Pulse Width Modulation & Motor Drivers
- Position Control of DC Motors
    - myRIO Implementation
- Force Control of DC Motors
    - myRIO Implementation

Content

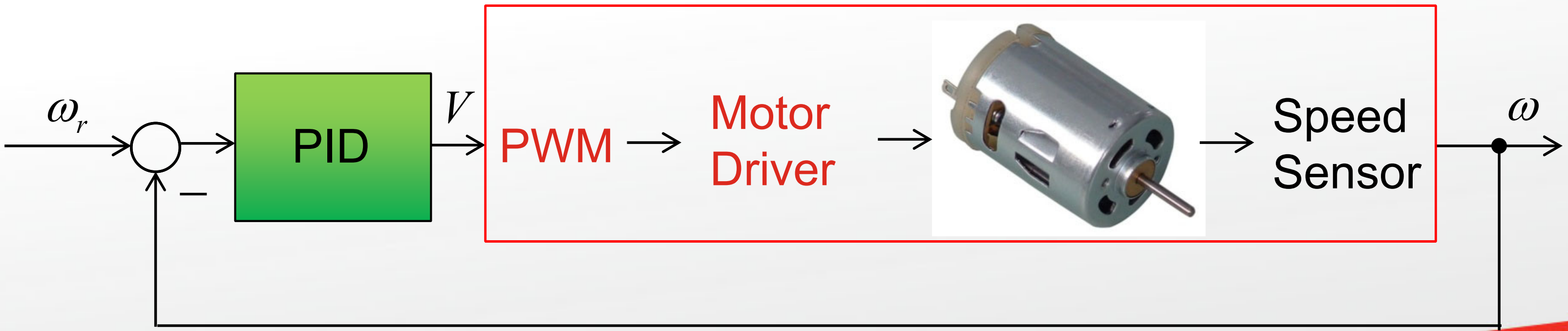
- Pulse Width Modulation & Motor Drivers
- Position Control of DC Motors
    - myRIO Implementation
- Force Control of DC Motors
    - myRIO Implementation

PWM and Motor Driver

- We saw this figure last week:



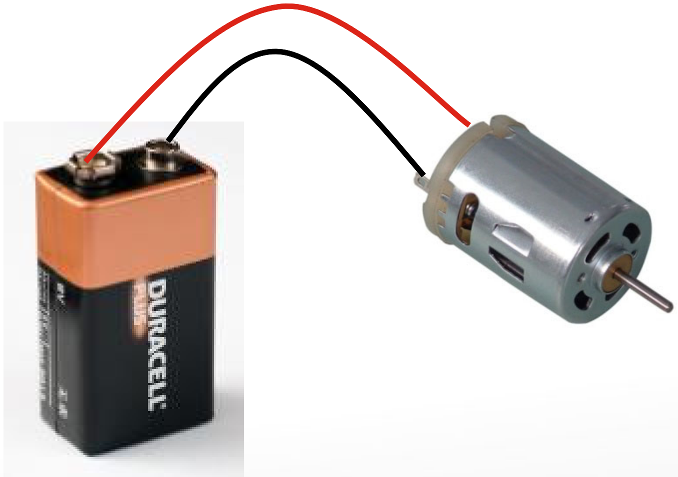
- The "Amplifier" part usually consists of two portions:



- What are these exactly?

PWM - Changing Motor Speed

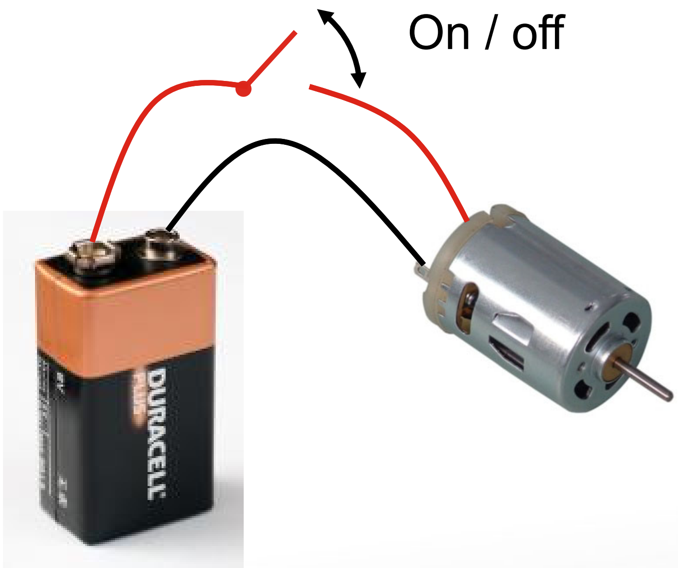
- Let's discuss about PWM first.
- Imagine you have a battery and a DC motor.

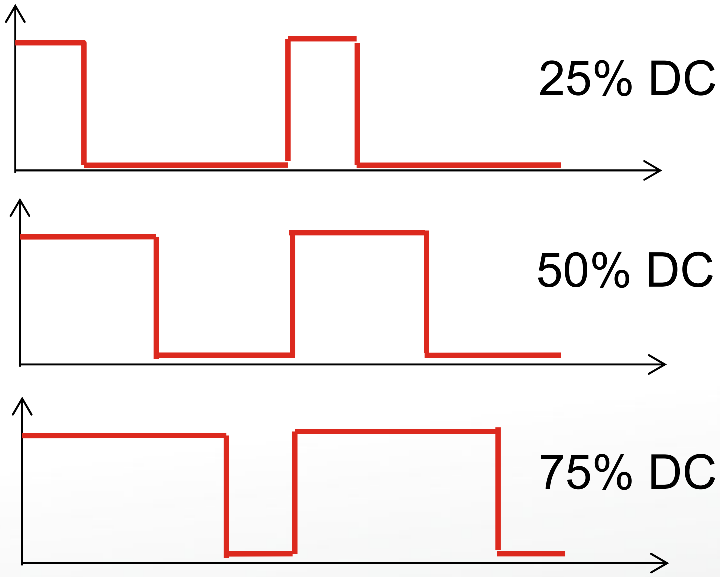


- If you connect the battery and the DC motor, the motor will spin at the full speed.
- How can you achieve a different / variable speed?
  - Remember that the battery gives constant voltage, not varying voltage.

PWM – Changing Motor Speed

- One method is to connect & disconnect & connect & disconnect the battery at different duty cycle (% of on-time within one period).





- Motor accelerates when switch is on, and decelerates when switch is off.
- If switching frequency is high (typically tens of kHz), the “mechanical filtering effect” will result in a relatively smooth rotation with speed determined by the duty cycle.

Torque

For a multi-turn coil the torque is expresses as following:

$$T = KBi_w \sin\alpha$$

where $\boldsymbol{K}$ is a constant, $\boldsymbol{i_w}$ is the current,
$\boldsymbol{B}$ is stator field magnetic flux density and
α is angle between $\boldsymbol{B}$ and normal to the coil plane.

DC Motor Control

$$T = KBi_w \sin\alpha$$

$$T = T(B, \alpha, i_w)$$

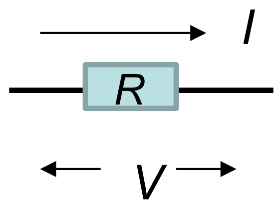
Torque and speed are proportional to the electrical current

- Current, or DC voltage control
- DC Pulse width control - PWM

Energy and Power

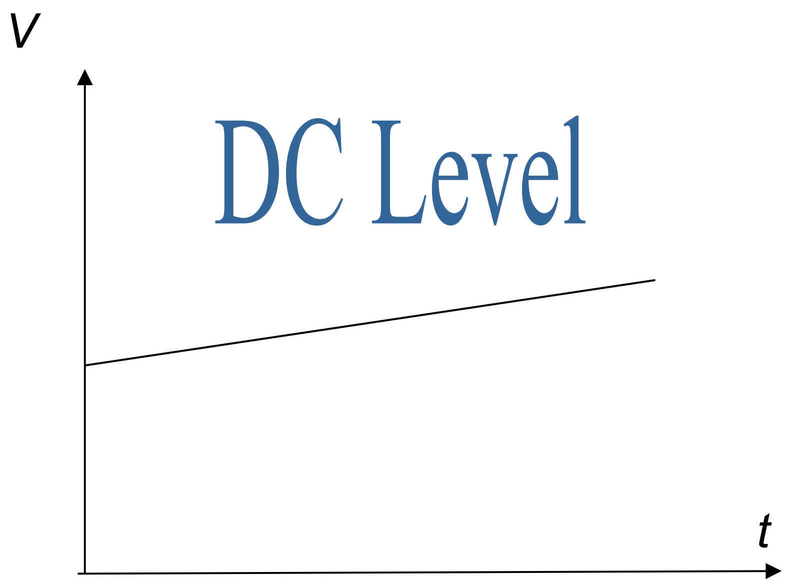
- Energy is the capability to do the Work [J = joule]
- Power [W = watt] is the energy use rate (in time)

$$P = Work/time$$



instant power	$p(t) = v(t)i(t)$
(DC power)	$P = VI$
$V = RI$	$P = VI = (RI)I = RI^2$
$I = V/R$	$P = VI = V(V/R) = V^2/R$

DC Motor Control Modes

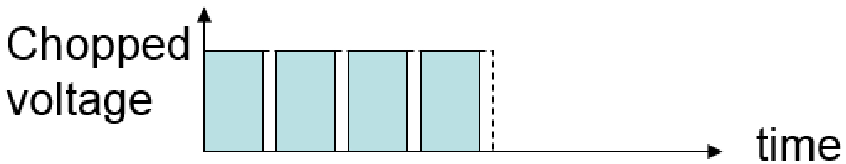


We need to know the amount of energy transferred to the motor,
Effective value, known as RMS (root-mean-square) value

DC Motor Control

- Voltage applied to armature controls speed

- Chopping DC supply voltage, known as PWM (Pulse Width Modulation)

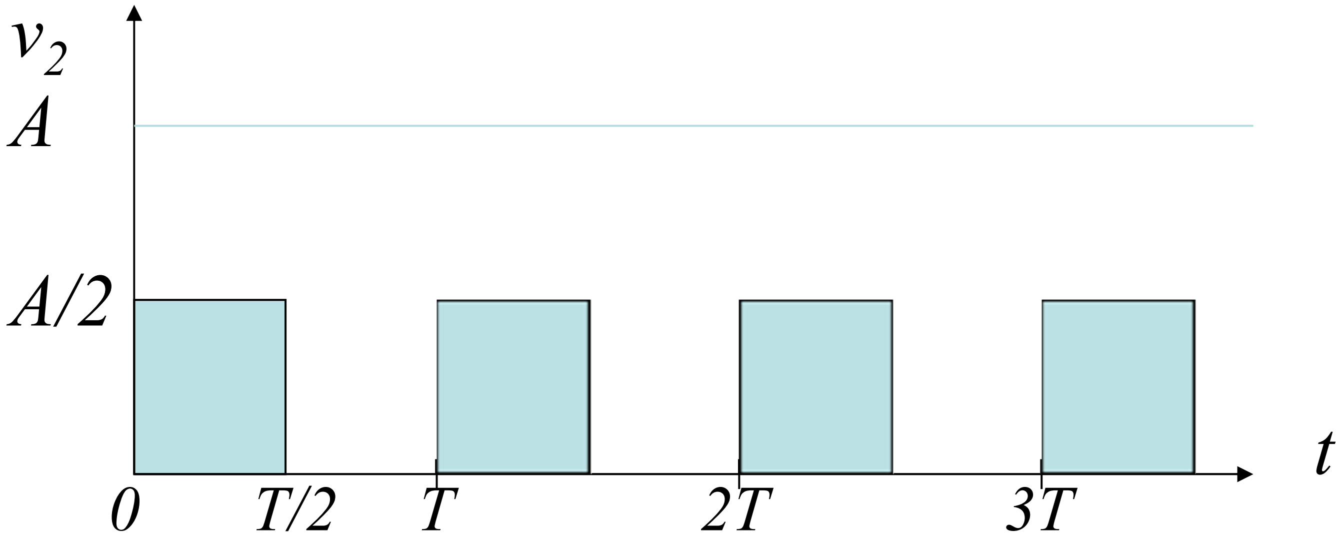


$$V_{RMS}^2 = \frac{1}{T}\int_0^T v^2 dt$$

The RMS value is equal to the DC voltage level that would produce the same amount of heat across a resistive load as variable voltage applied.

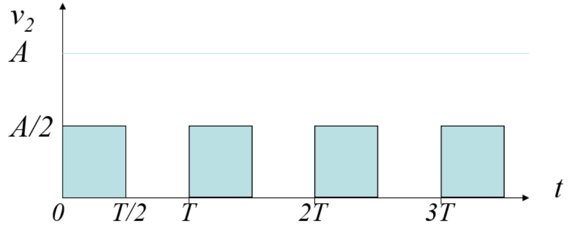
The amount of energy transferred to the motor, Effective value, RMS (root-mean-square)

Calculate Average and RMS Values as Functions of Pulse Width



$$V_{AWR} = \frac{1}{T}\int_0^T v \; dt \qquad\qquad V_{RMS}^2 = \frac{1}{T}\int_0^T v^2 dt$$

Average and RMS Values

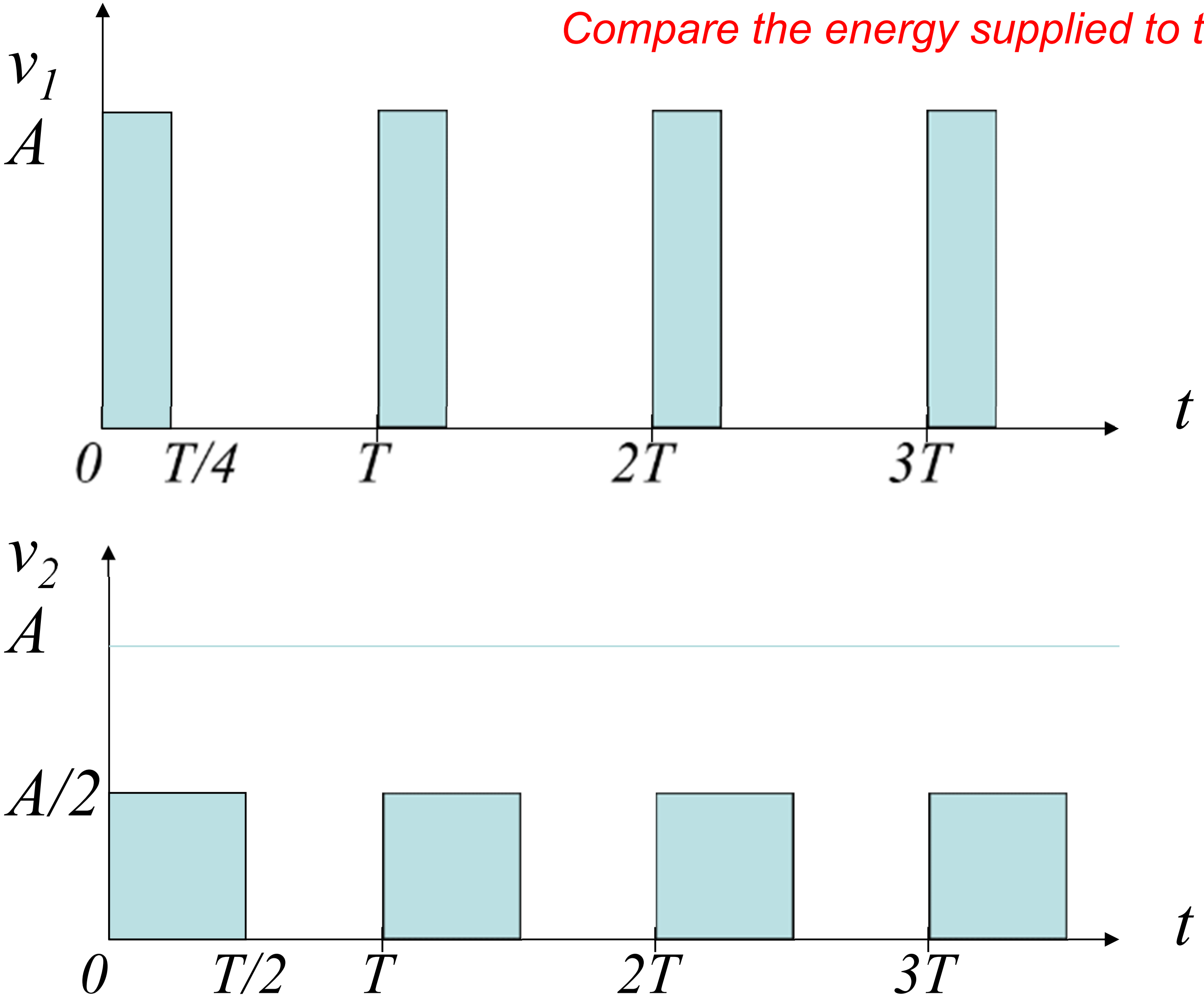


$$V_{AWR} = \frac{1}{T}\int_0^T v\ dt = \frac{1}{T}\int_0^{T/2} \frac{A}{2} dt = \frac{1}{T}\frac{A}{2} t \,\Big|_0^{A/2} = \frac{1}{T}\frac{A}{2} t \,\Big|_0^{T/2}\ \frac{1}{T}\frac{A}{2}\frac{T}{2} = \frac{A}{4}$$

$$V_{RMS}^2 = \frac{1}{T}\int_0^T v^2 dt = \frac{1}{T}\int_0^{T/2} (\frac{A}{2})^2 dt = \frac{1}{T}(\frac{A}{2})^2 t \,\Big|_0^{T/2} = \frac{1}{T}(\frac{A}{2})^2 \frac{T}{2} = (\frac{A}{2})^2 \frac{1}{2}$$

$$V_{RMS} = \frac{1}{\sqrt{2}}(\frac{A}{2}) = \frac{\sqrt{2}}{4} A$$

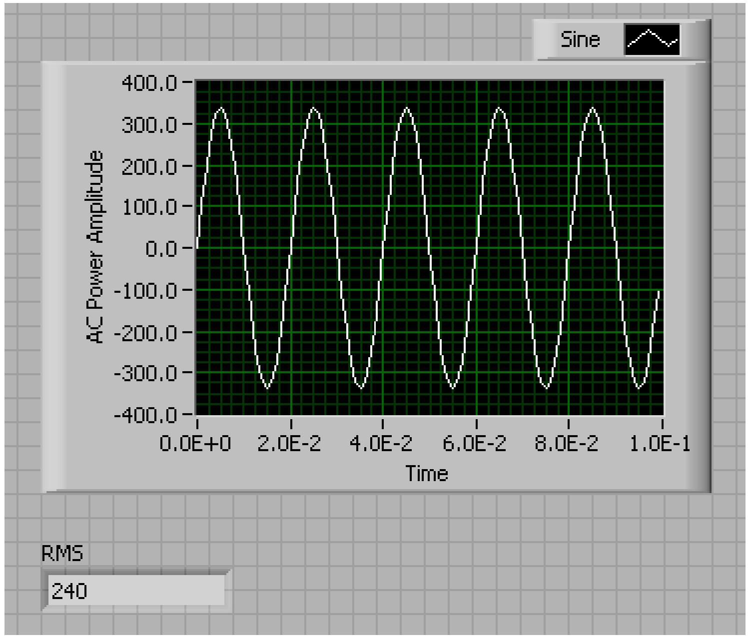
Compare the energy supplied to the motor



Effective Value, or RMS

The RMS value is equal to the DC voltage level that would produce the same amount of heat across a resistive load as variable voltage applied.

$$V_{RMS}^2 = \frac{1}{T}\int_0^T v^2 dt$$



AC Voltage Amplitude *340V*

Frequency *50Hz*

Find the expression for V_{RMS} if

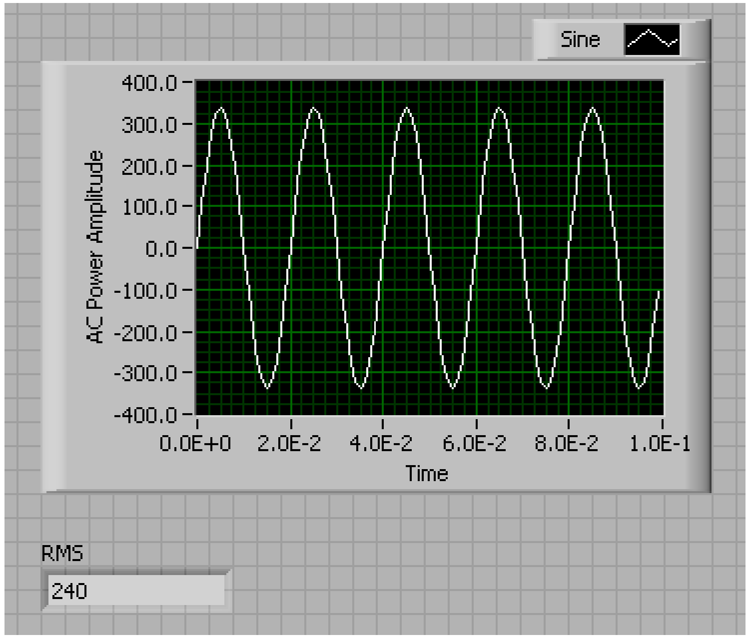
$v(t)=V_{max}\sin(wt)= V_{max}\sin(2\pi ft)$

Effective Value, or RMS

The RMS value is equal to the DC voltage level that would produce the same amount of heat across a resistive load as AC voltage applied.

$$V_{RMS}^2 = \frac{1}{T}\int_0^T v^2 dt$$

AC Voltage Amplitude 340V
Frequency 50Hz



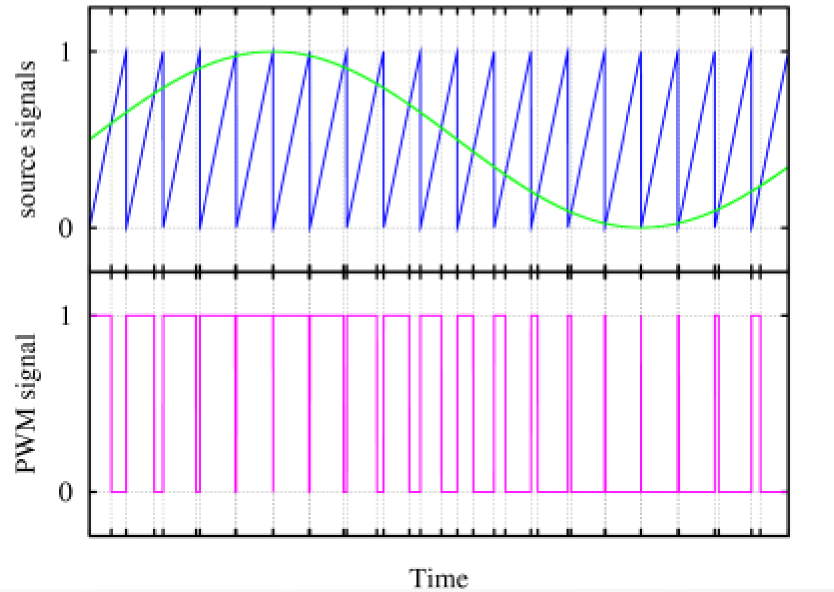
$V_{RMS} = 0.707\ V_{max}$

RMS (root-mean-square)
= Amplitude /√2=
340/ √2 = 340 * 0.707=240

Australia:	**240V, 50Hz**
Europe:	**220V, 50Hz**
USA:	**120V, 60Hz**

PWM

- How do we generate the digital pulse from a desired analog value?
  - By comparing with a base signal, most often a fixed-frequency saw wave:



- If analog signal is higher than the saw wave → Logical high.
- If analog signal is lower than saw wave → Logical low.
- With this we can obtain a varying pulse width in accordance to the analog value.

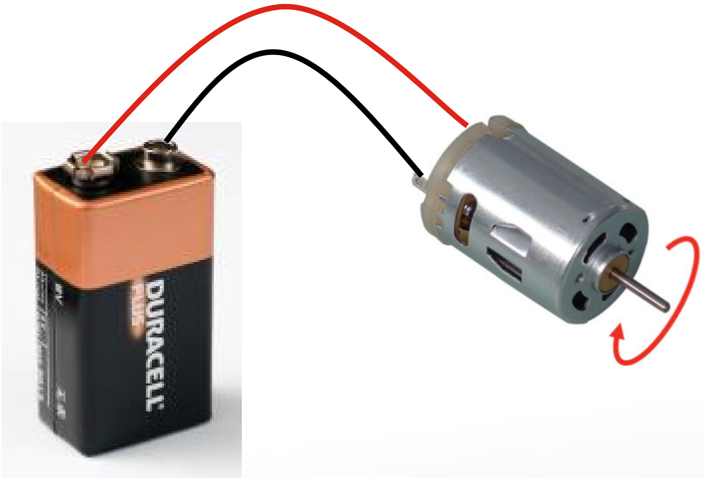
PWM Generation
https://commons.wikimedia.org/wiki/File:Pwm.png

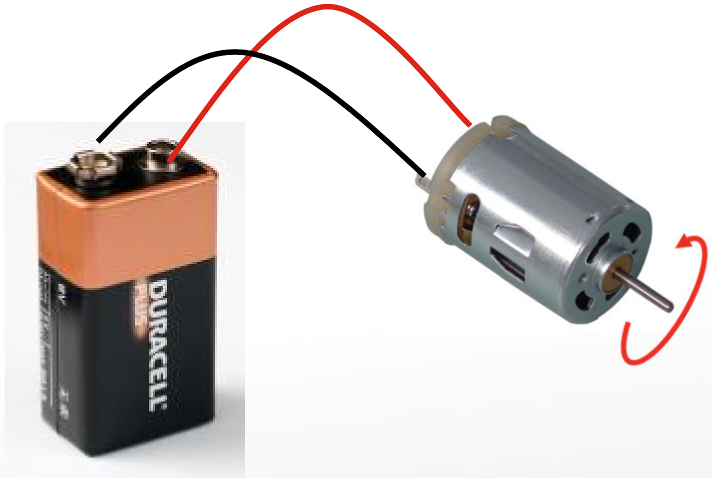
Automotive Electronics and Triacs

See Power Electronics Slides

Changing Motor Direction

- Next, let's think about how you can change the motor direction.
- One simple way is to swap the wires:

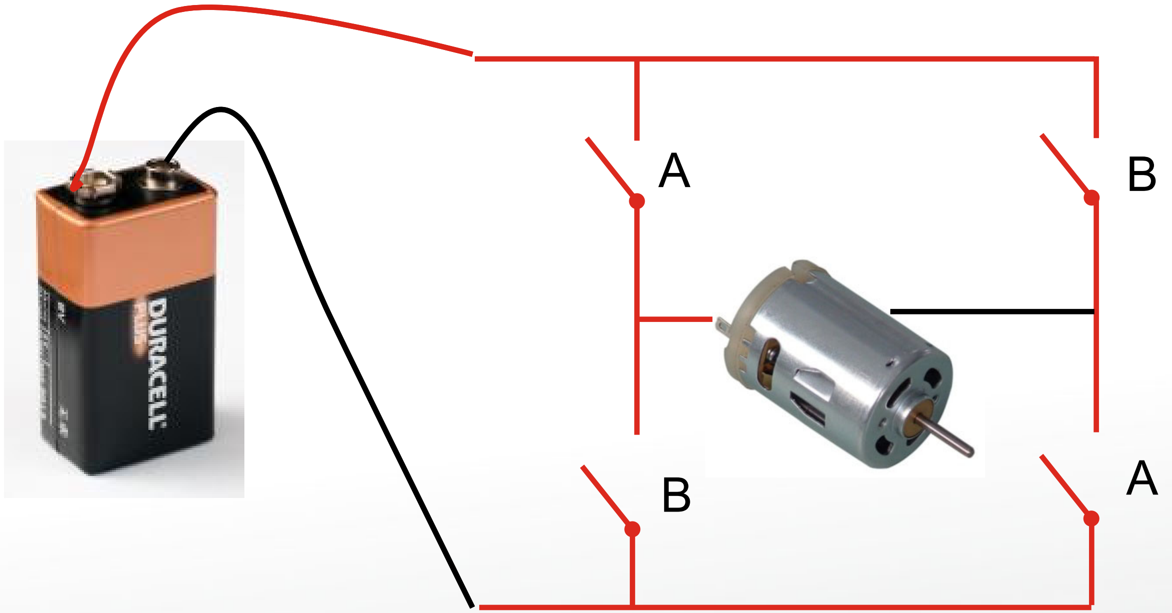




- But how can this be done automatically?

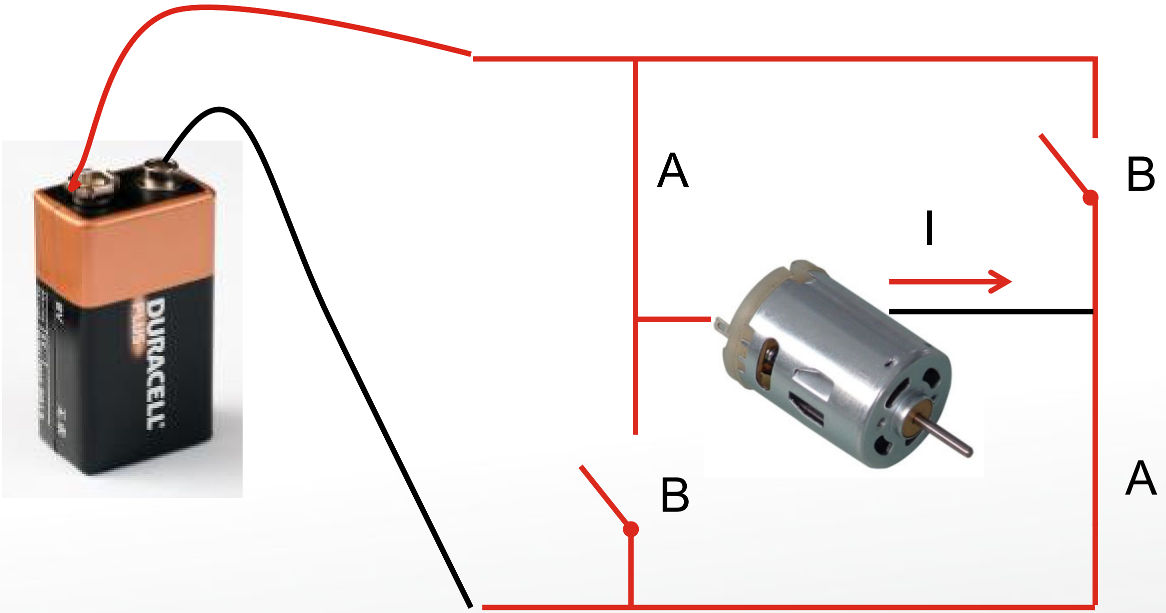
H-Bridge / Motor Driver

- This can be achieved by using an H-Bridge circuit:



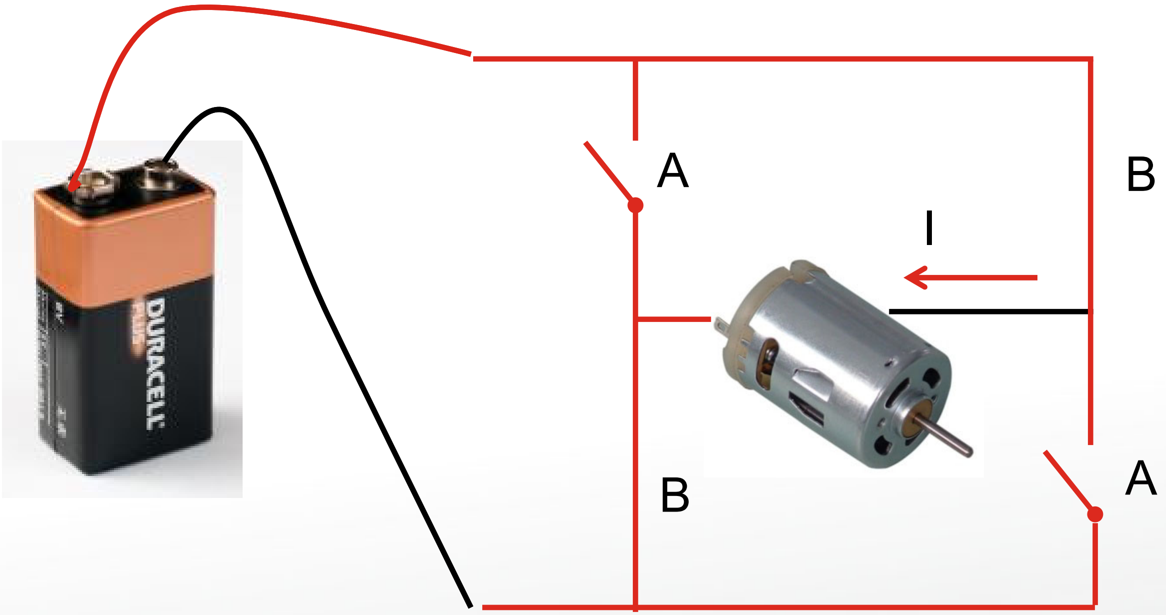
H-Bridge / Motor Driver

- If switches A are closed, current will move from left to right, and motor turns in one direction.



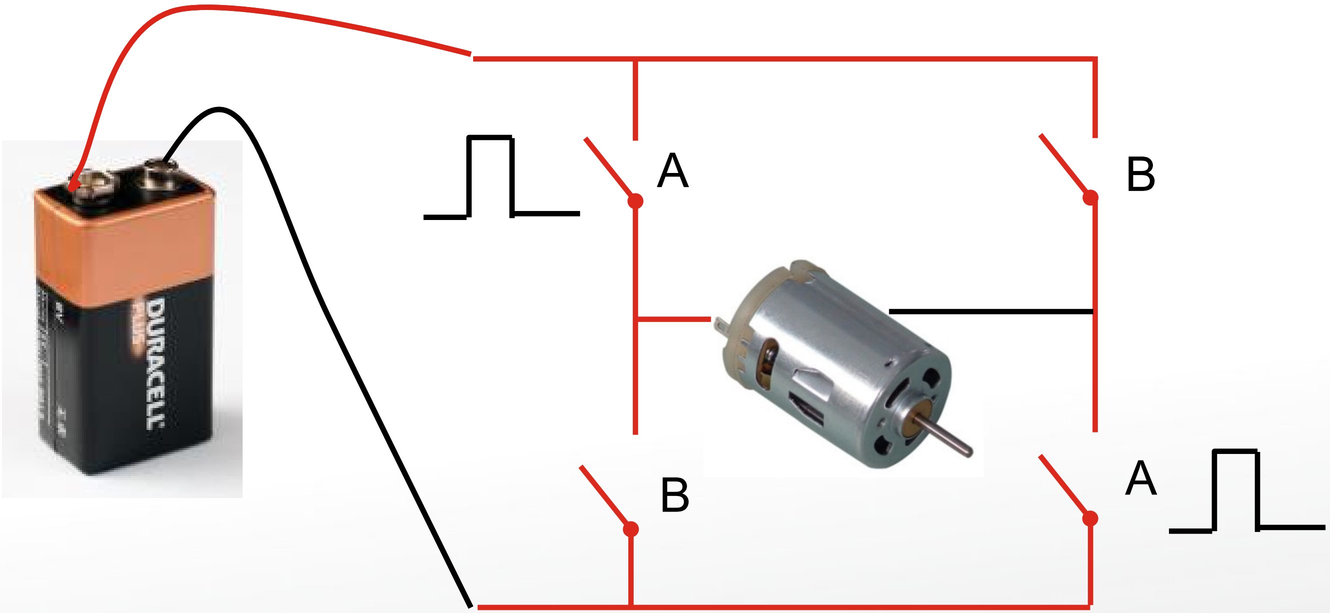
H-Bridge / Motor Driver

- If switches B are closed, current will move from right to left, and motor turns in the other direction.



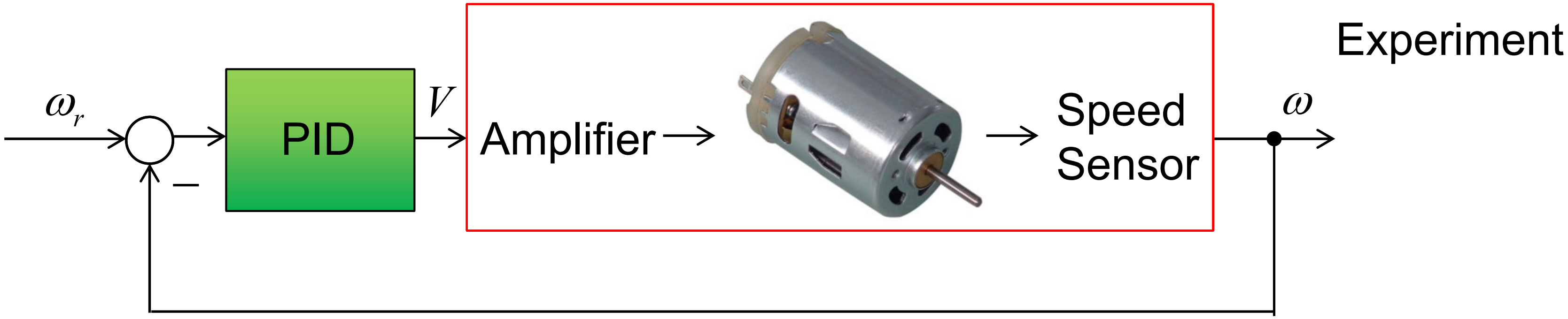
Changing Speed AND Direction

- Combining PWM and H-Bridge, we can now alter BOTH speed AND diection.



Amplifier?

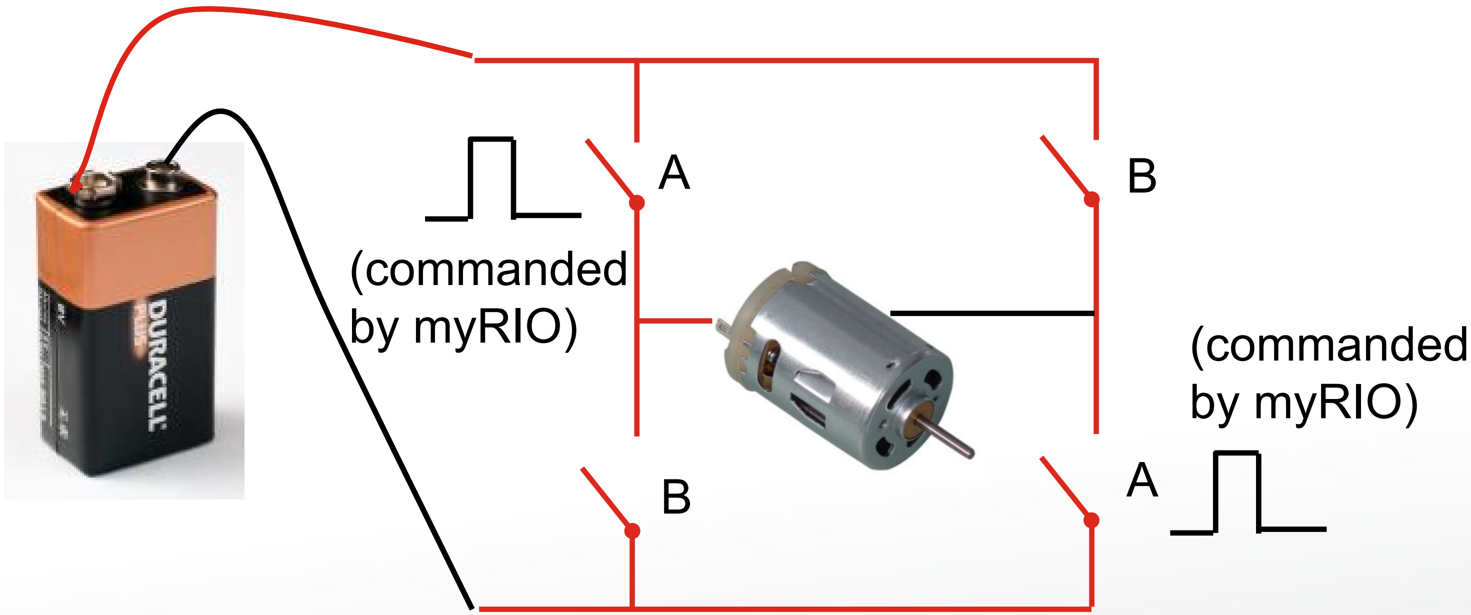
- Why is then the H-Bridge an amplifier?



- The current coming from the controller (e.g. myRIO) is very small (e.g. 2mA).
- This is not enough to power up the motor.
  - In fact, if you connect the myRIO analog output to the motor terminals directly, you may damage the myRIO device.

Amplifier?

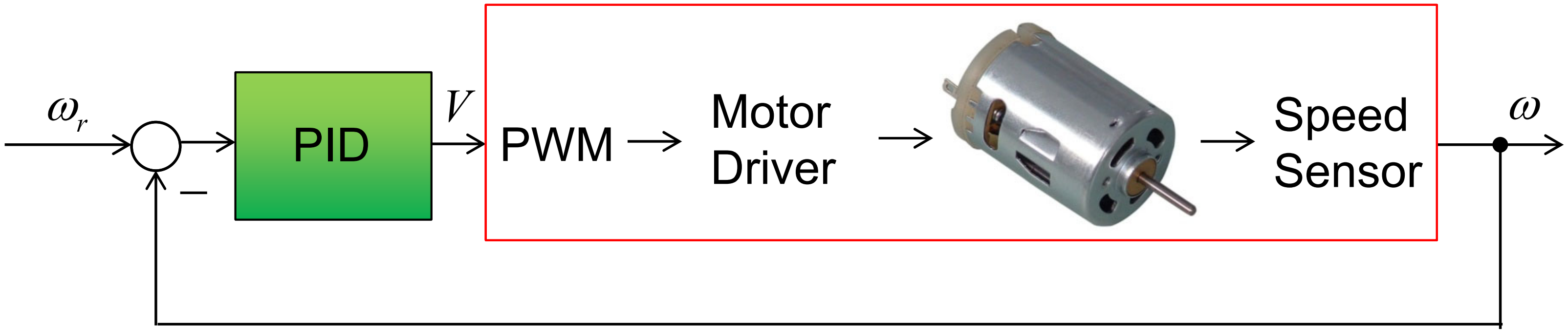
- Instead, we connect external battery for the motor, so that the current meets the requirement of the motor.



- The output from controller (e.g. myRIO) is now used to only turn on & off the switches. A low current is sufficient.
  - So, the H-Bridge receives a low current control input from the controller, and translate this into a high current output for the motor.
  - Thus it is an amplifier.

PWM and Motor Driver

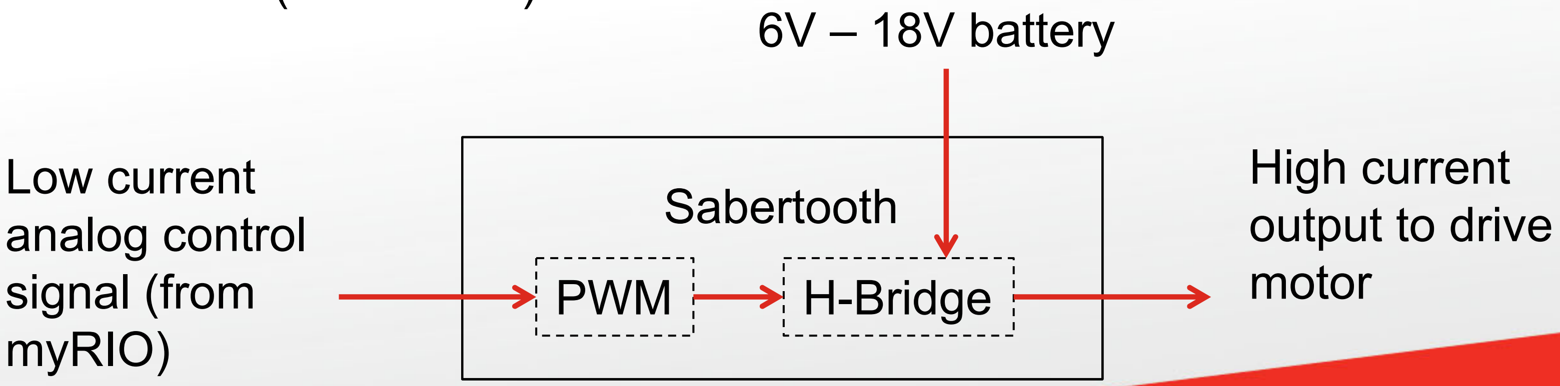
- Summary:



- In your project, you will use a Sabertooth 2x5 as the motor driver.
- See next slide…

Sabertooth 2x5

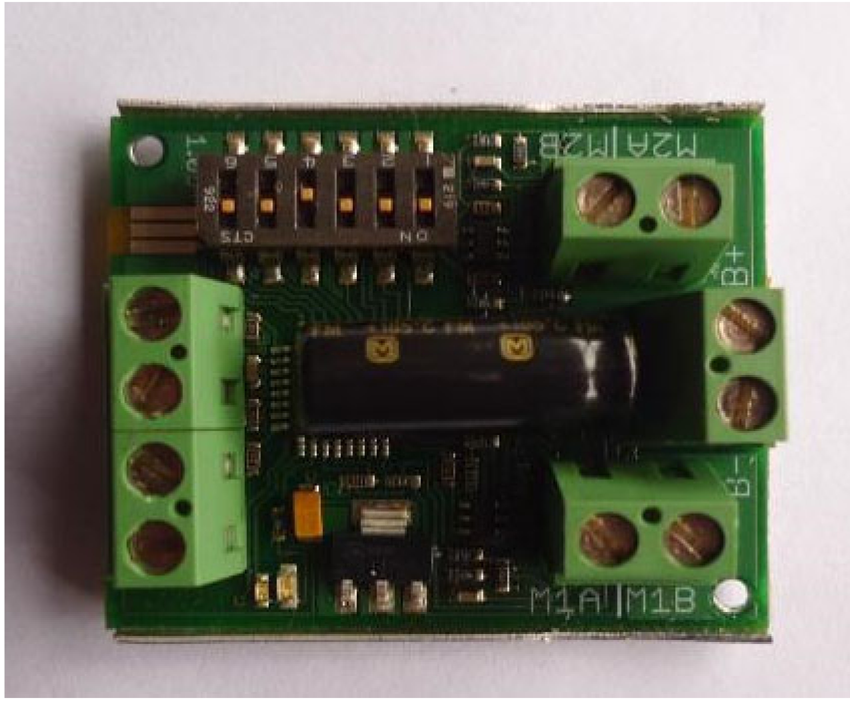
- About the Sabertooth 2x5 motor driver:
- It is able to drive two DC brushed motors simultaneously.
- Provides up to 5A continuous current per channel.
  - Peak current of 10A for a few seconds.
- Power supply (through B+ and B-): 6V to 18V.
- Low-current analog control signals are connected to S1 (for control of Motor 1) and S2 (for control of Motor 2).
- High-current output signals are provided by M1A & M1B (for Motor 1) and M2A & M2B (for Motor 2).



Outputs - Actuators - Motors

- Brushed DC Motor
- Gear Motor
- 7.2VDC
- 160rpm

Motor Driver



- Sabertooth 2x5
- Drive Two Motors
- 6-18V nominal
- 20V maximum
- 5A per channel
- Peak drive up to 10A per channel

Driver Modes of Operation

1. **Analog Input**
2. R/C Input
3. Simplified serial
4. Packetized serial

Set up the switches

Analog input, Independent mode, Linear response, Sensitivity mode disabled

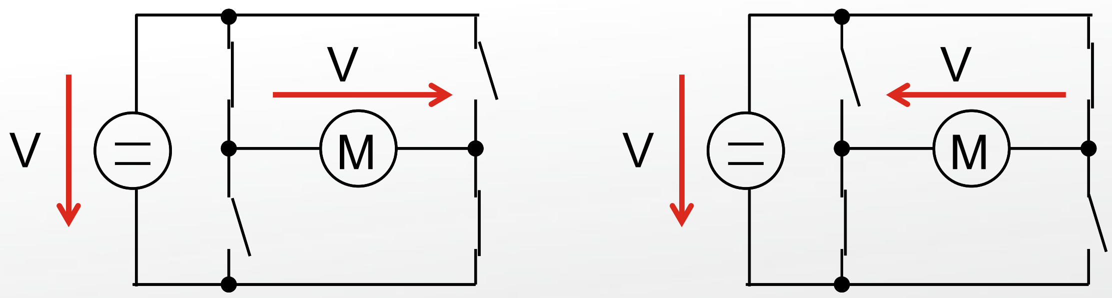
Analog Mode

- The input signals that control the Sabertooth are connected to terminals S1 and S2.
- If driver is running in analog mode, it is important to have both the signal inputs connected before applying power to the device.
- Otherwise, the motors may start unexpectedly.

Sabertooth 2x5

Analog Mode:

- Make sure that the switches on Sabertooth are "up-up-up-down-up-up".
- Input signal (at S1 / S2):
    - 2.5V = No movement
    - 5V = maximum speed in one direction
    - 0V = maximum speed in another direction
    - Values between 0V & 2.5V, and 2.5V & 5V → The switches turn on and off at certain rates to vary the speed of motor.

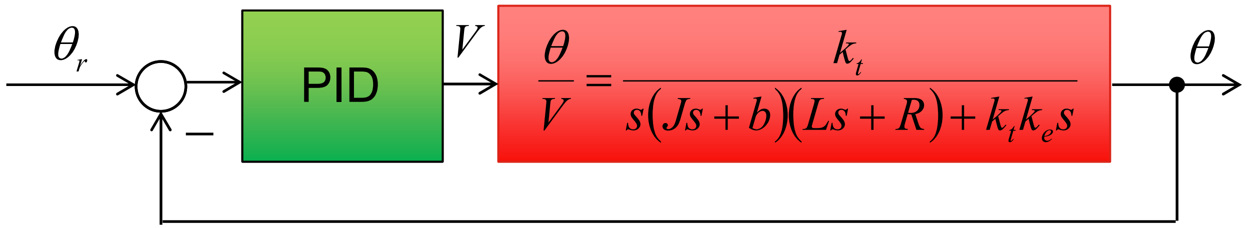


Content

- Pulse Width Modulation & Motor Drivers
- Position Control of DC Motors
    - myRIO Implementation
- Force Control of DC Motors
    - myRIO Implementation

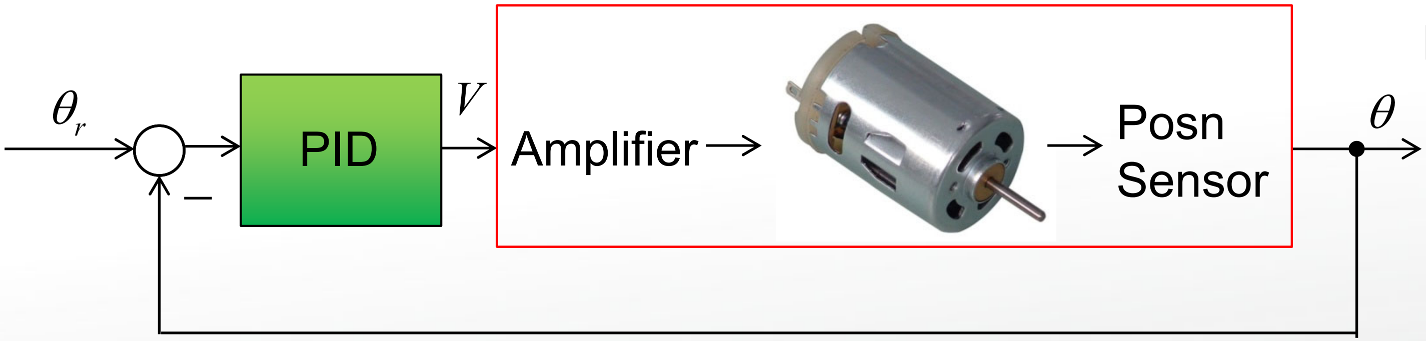
Position Control of Motor

- The closed loop block diagram of motor position control is given below:



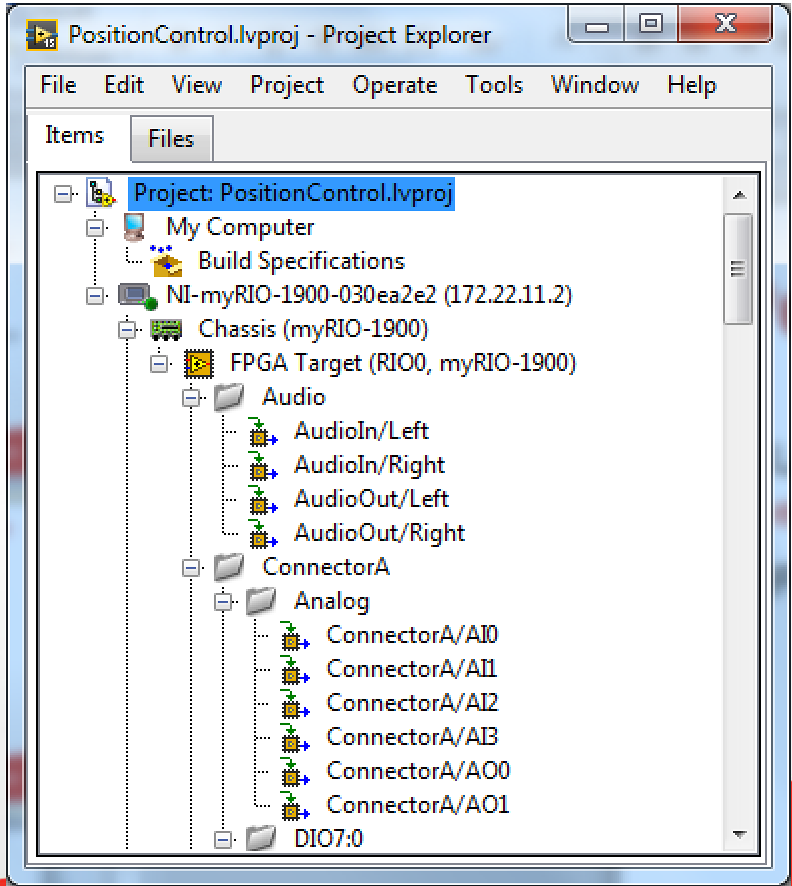
Simulation

Experiment



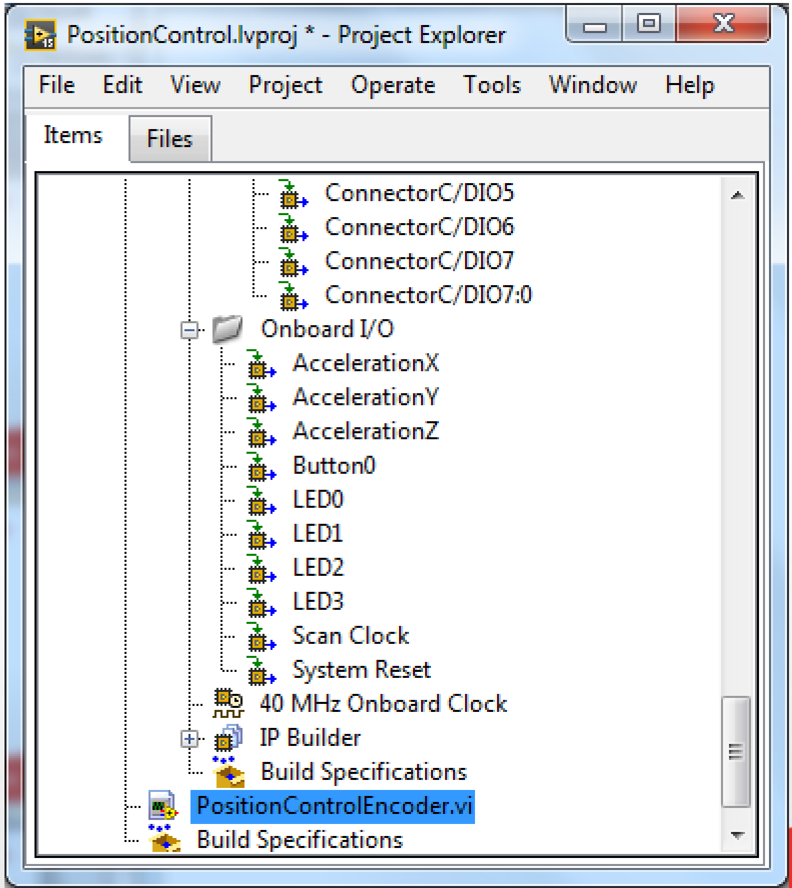
Position Control of Motor - Encoder

- Let's now build the control loop for position control of the motor, using encoder as the sensor.
- First of all, create a myRIO project called "Position Control".



Position Control of Motor - Encoder

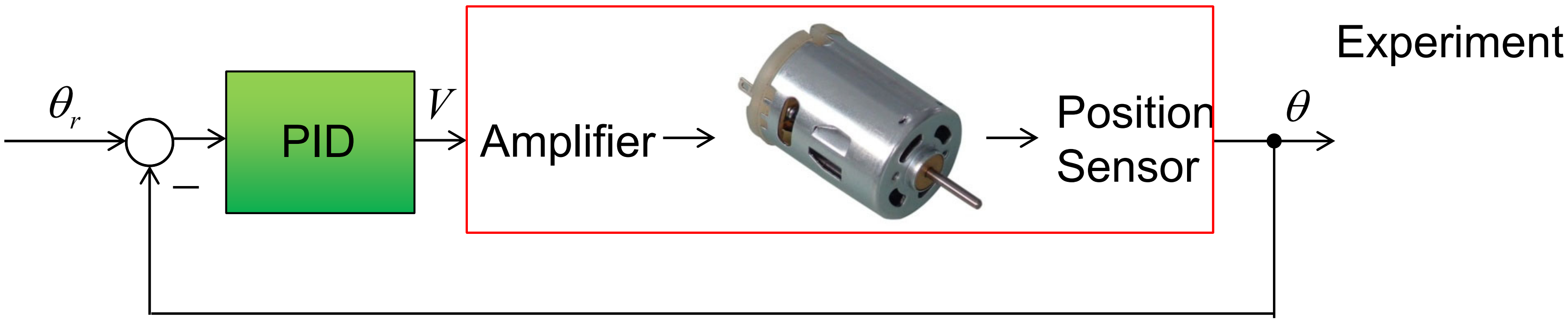
- Right click on “NI-myRIO-…” and choose “New VI”.
- Save the VI as “PositionControlEncoder.vi”.



RMIT UNIVERSITY

Position Control of Motor - Encoder

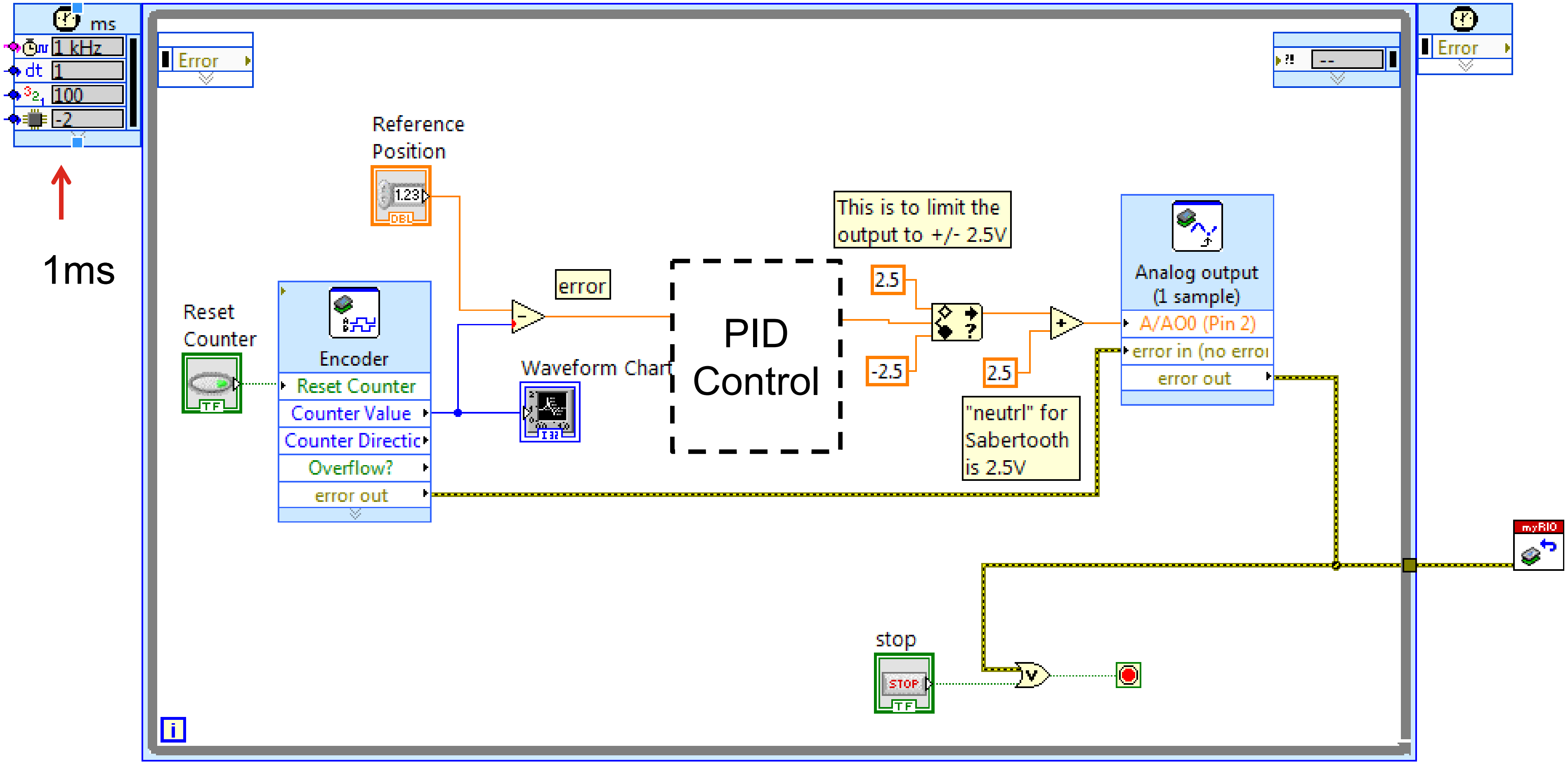
- We will write the code in LabVIEW to resemble the following block diagram:



- Looking at the block diagram, we see that we need the following in our code:
  - Key in reference position.
  - Read in encoder signal.
  - Compare the two and calculate the "error".
  - Pass the error through the PID controller.
  - PID controller calculates and provides the voltage output to Sabertooth.

Position Control of Motor - Encoder

- So let's code this up first:

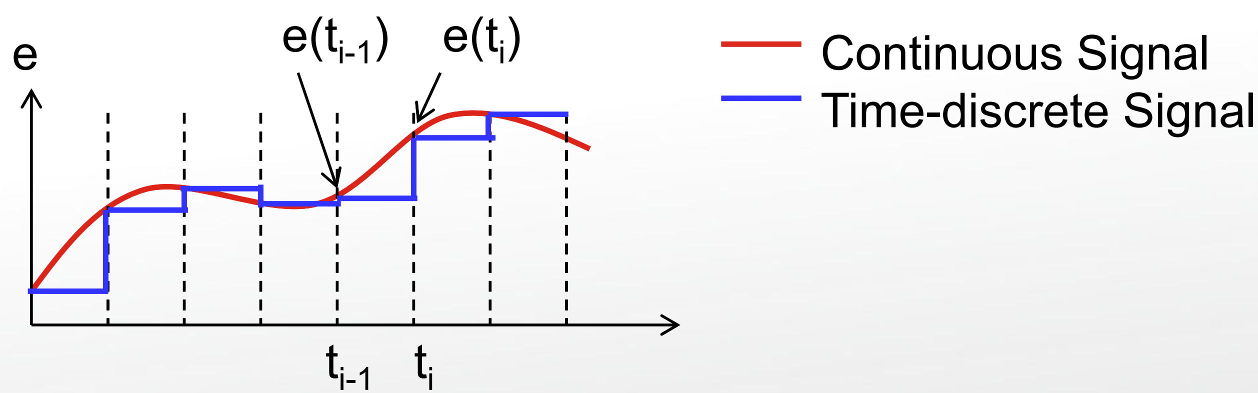


Aside: Controller Implementation

- Question: how do we code the PID controller in LabVIEW?
- The PID controller is:

$$u(t)_{PID} = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_D \dot{e}(t)$$

- Because we are implementing the controller in a digital system (myRIO), the signals are actually read in at a constant sampling rate.
- For simplicity, assume signal is constant in between sampling:

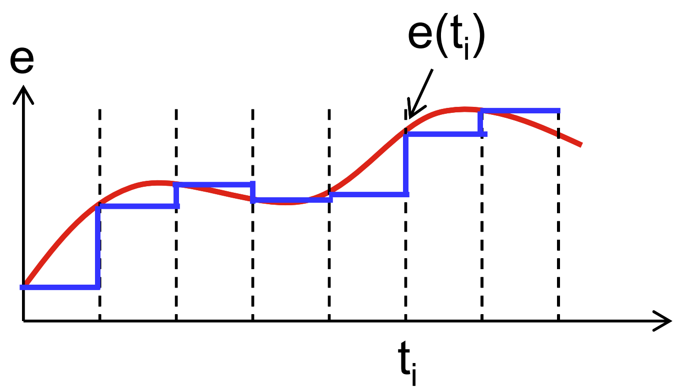


- Obviously, the faster the sampling rate, the closer the discretized signal is to the continuous signal.

Aside: Controller Implementation

- Assume we are now standing at time instant t_i.
- P part of control is straightforward:

$$u(t_i)_P = K_p e(t_i)$$



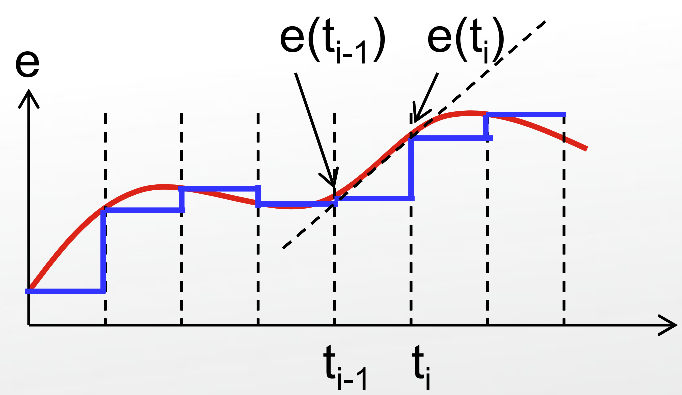
- D part of control was:

$$u(t)_D = K_D \dot{e}(t)$$

  - This means K_D multiply by gradient of e-curve.
  - This can be approximated as:

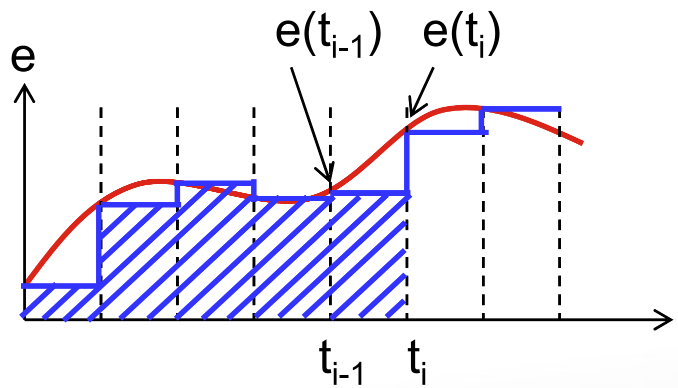
$$u(t_i)_D = K_D\left(\frac{e(t_i) - e(t_{i-1})}{T}\right)$$

  - Where T is the sampling interval.



Aside: Controller Implementation

- The I part was: $$u(t)_I = K_I \int_0^t e(\tau)d\tau$$
- Which means K_I multiplied by the area under the e-curve up to time t.
- The area under the curve can be approximated as summation of the rectangular blocks, each with height e(at the respective sampling time) and width T.



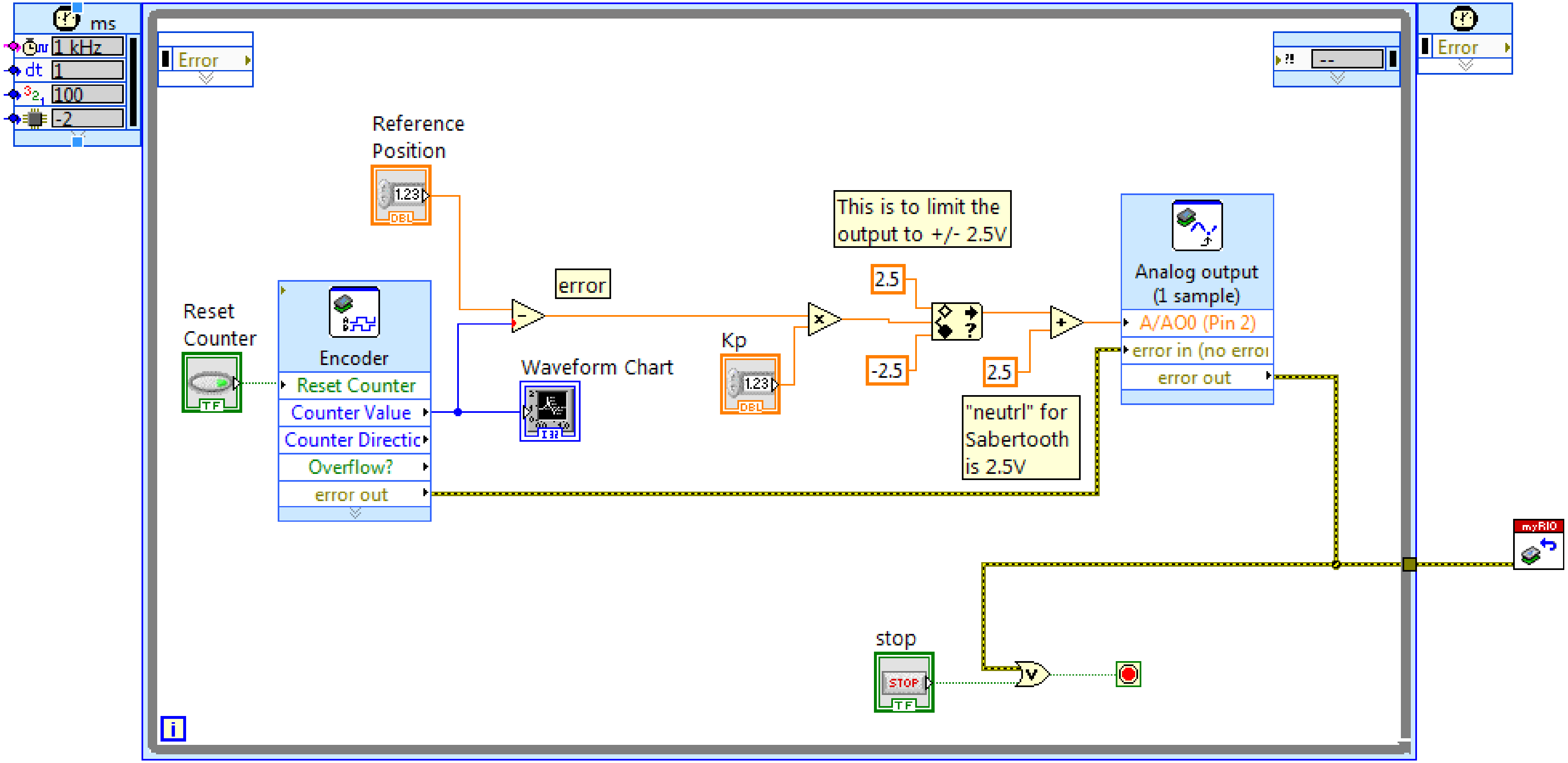
- Thus the I control is approximated as:

$$u(t)_I = K_I \sum_{i=0}^{i} e(t_i)T$$

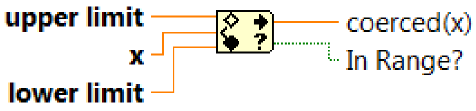
- Which can also be implemented as: $$u(t)_I = K_I e(t_i)T + \underbrace{\sum_{i=0}^{i-1} K_I e(t_i)T}_{u(t_{i-1})_I}$$

Position Control of Motor - Encoder

- We implement the P-control first:



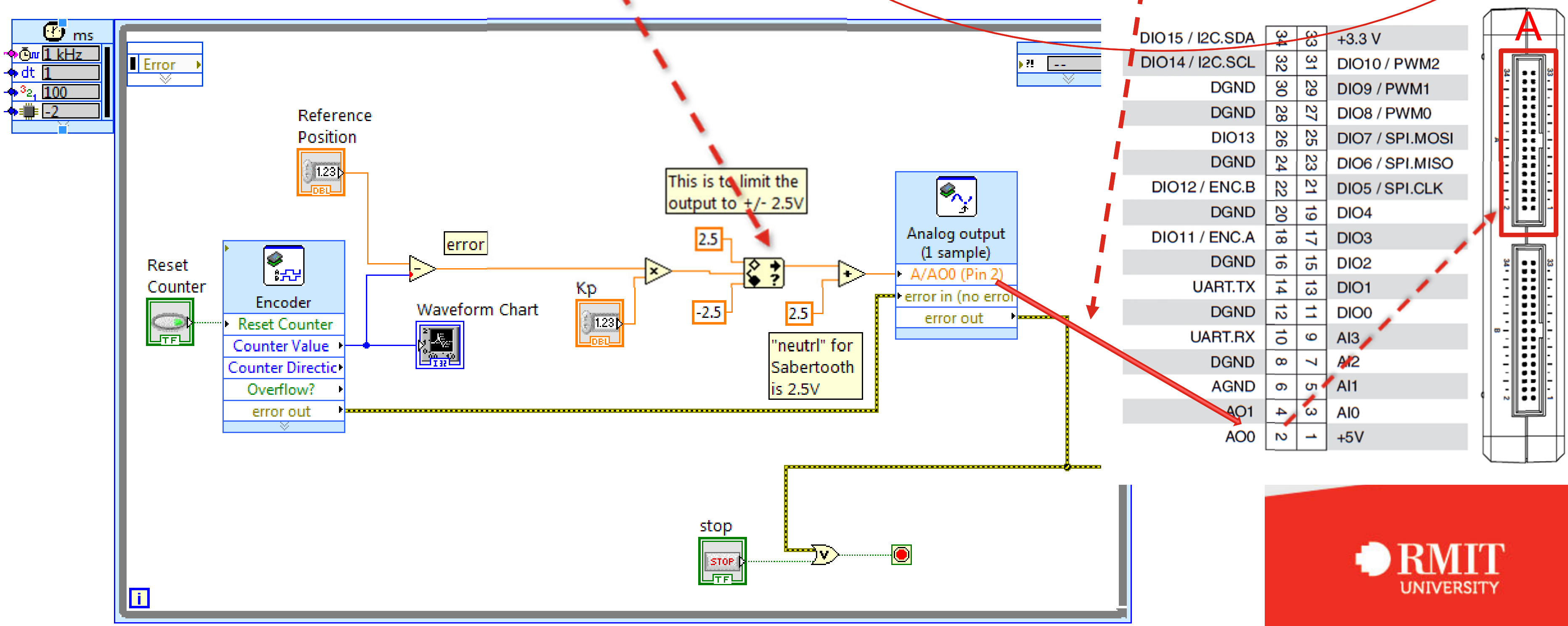
In Range and Coerce



Determines whether **x** falls within a range specified by the **upper limit** and **lower limit** inputs and optionally coerces the value to fall within the range. The function performs the coercion only in Compare Elements mode. This function accepts time stamp values if all inputs are time stamp values. You can change the comparison mode of this function.

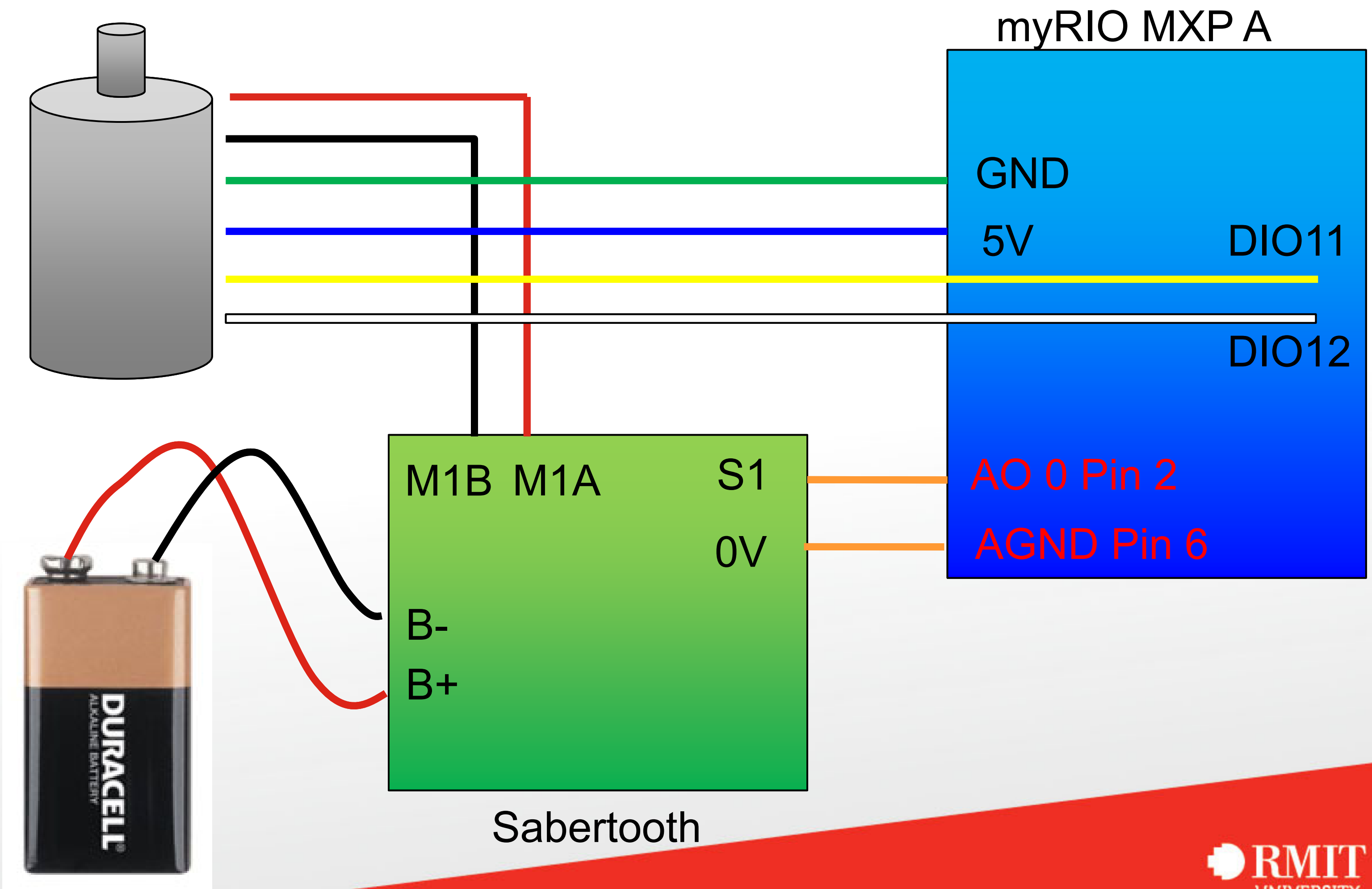
AO <0..1>

0-5 V referenced, single-ended analog output. Refer to the *Analog Output Channels* section for more information.



Position Control of Motor - Encoder

- Now, connect your motor to Sabertooth and/or myRIO as follows:

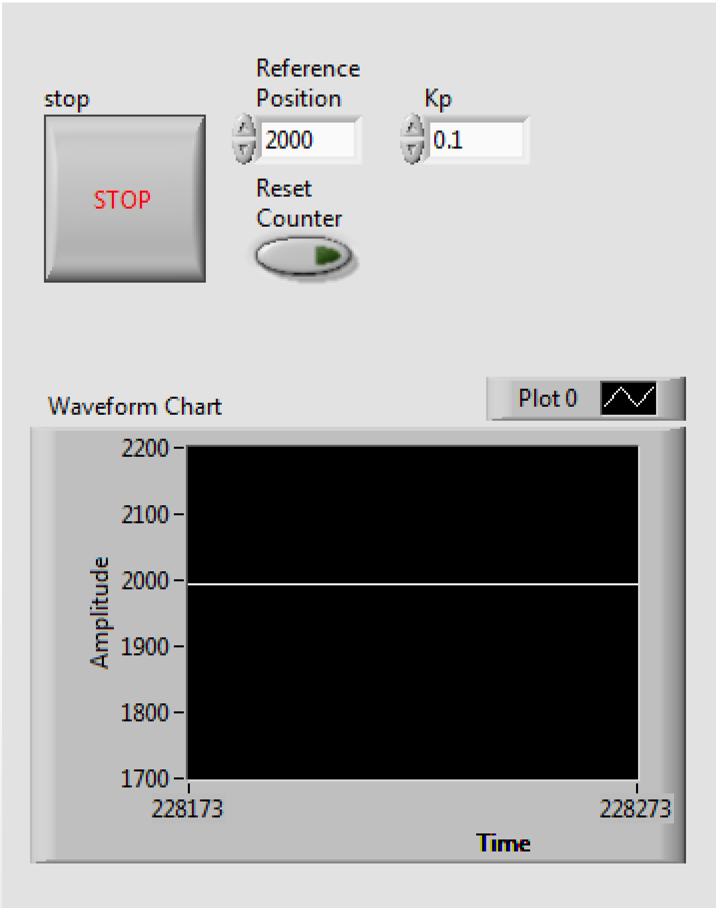


Position Control of Motor - Encoder

- Please follow the following sequence (very important!)
  - Set Reference Position as 0.
  - Set Kp as 0.
  - Click Play.
  - Reset Encoder Counter.
  - Plug in the battery.
  - Change Reference Position (e.g. 2000, i.e. 2000 Encoder Pulses).
  - Observe the response in Waveform Chart.
  - Tune Kp (e.g. 0.1, 0.2,….) until you are satisfied with the result.
  - When you are ready to turn this off:
    - Unplug the battery.
    - Click Stop in program.

Position Control of Motor - Encoder

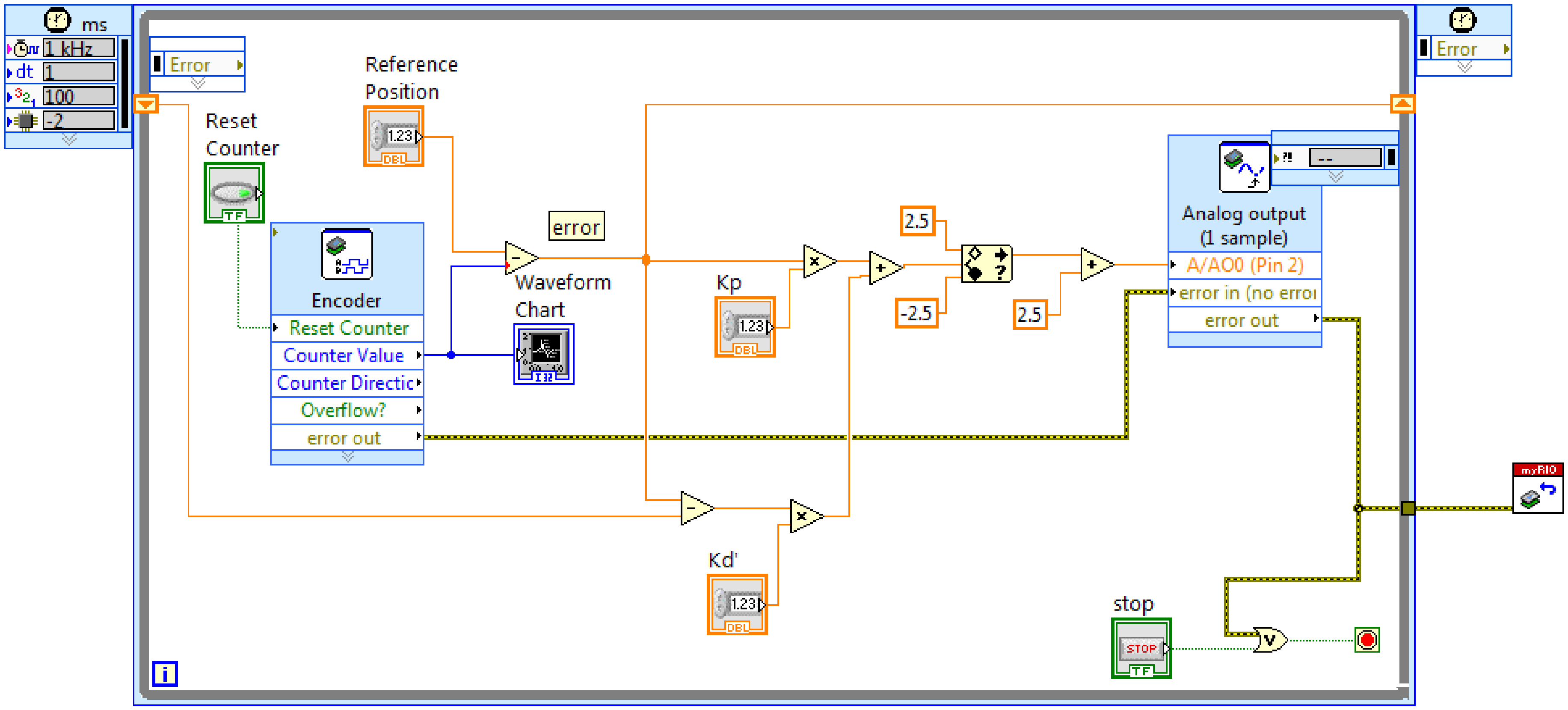
- This is an example of the result. The position is close to but not exactly 2000.



Position Control of Motor - Encoder

Shift register

- We can add in D-control.
- As mentioned, it is implemented as: $u(t_i)_D = K_D\left(\frac{e(t_i)-e(t_{i-1})}{T}\right)$
- We do not actually need to divide the change in error by T.
- Instead, we implement this: $u(t_i)_D = K_D^{'}\left(e(t_i)-e(t_{i-1})\right)$



Position Control of Motor - Encoder

- Finally, we add in I control.
- Its formula is:

$$u(t)_I = K_I e(t_i) T + \underbrace{\sum_{i=0}^{i-1} K_I e(t_i) T}_{u(t_{i-1})_I}$$

Shift register

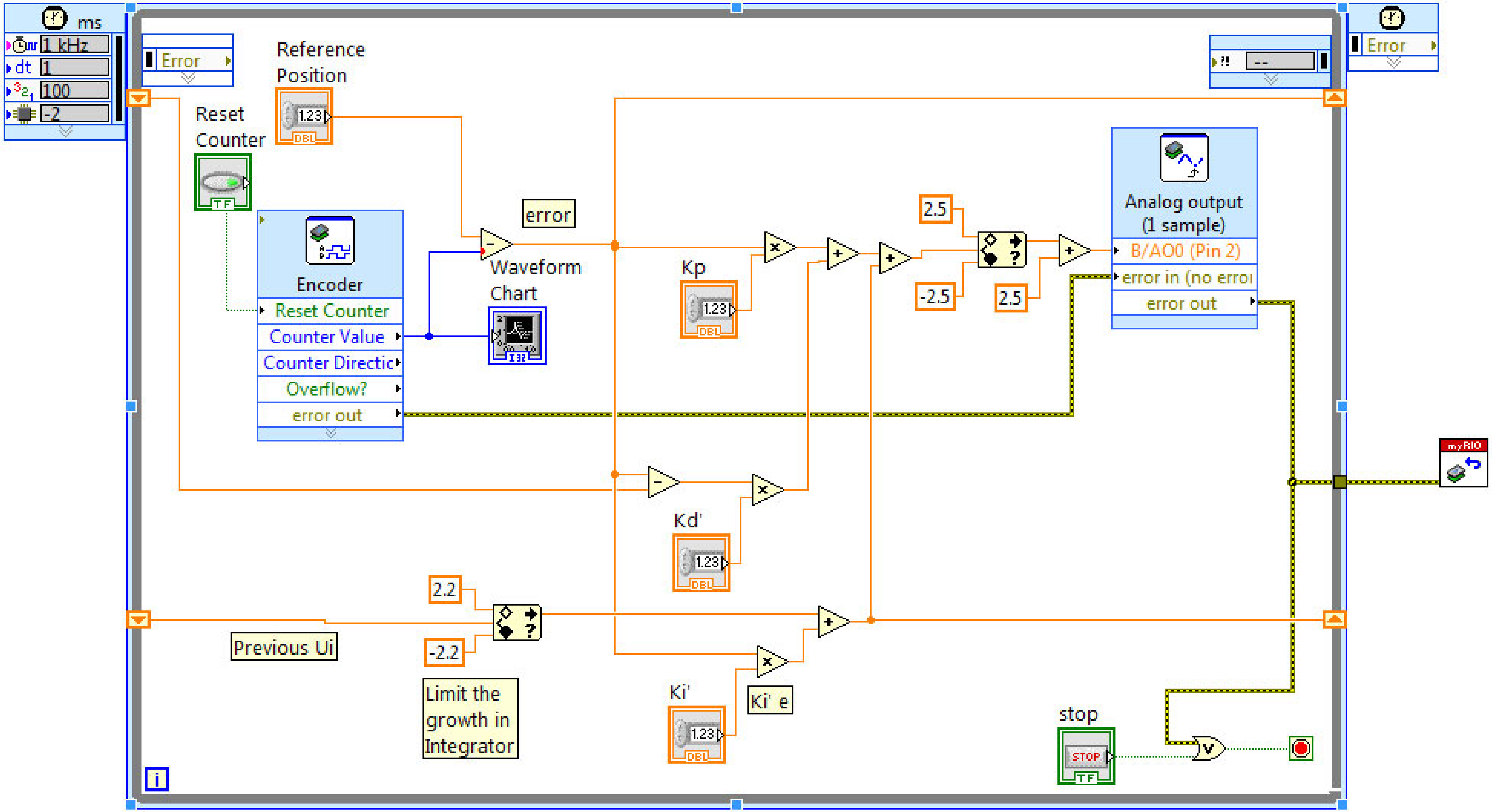
- Again, we do not need to multiply the terms with T.
- Instead, just code this up:

$$u(t)_I = K_I' e(t_i) + \underbrace{\sum_{i=0}^{i-1} K_I' e(t_i)}_{u(t_{i-1})_I}$$

- Note: We must limit the growth of this integration. Otherwise, it could become a large number in the background and creates unacceptable results, which is a phenomenon called “Integral Windup”.

Position Control of Motor - Encoder

- The code is as follows:

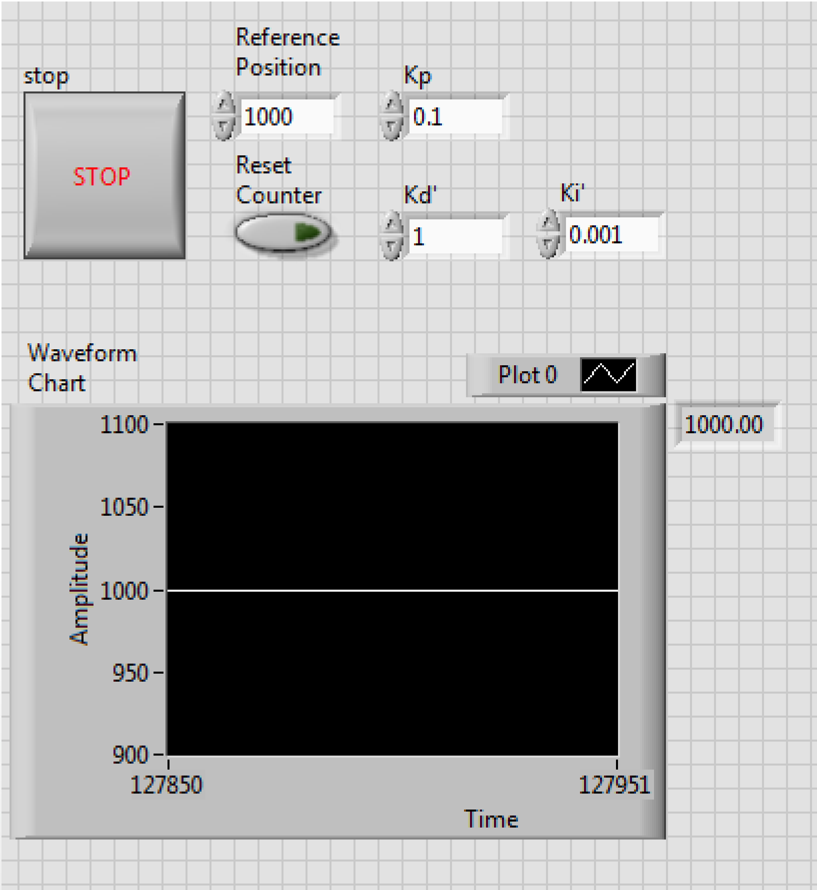


Position Control of Motor - Encoder

- Please follow the following sequence (very important!)
  - Set Reference Position as 0.
  - Set Kp, Ki and Kd as 0.
  - Click Play.
  - Reset Encoder Counter.
  - Plug in the battery.
  - Change Reference Position (e.g. 2000, i.e. 2000 Encoder Pulses).
  - Observe the response in Waveform Chart.
  - Tune Kp, Ki, Kd until you are satisfied with the result.
    - You may need to set Reference Position as 2000, then 1000, then 2000… for a few times during the tuning.
  - When you are ready to turn this off:
    - Unplug the battery.
    - Click Stop in program.

Position Control of Motor - Encoder

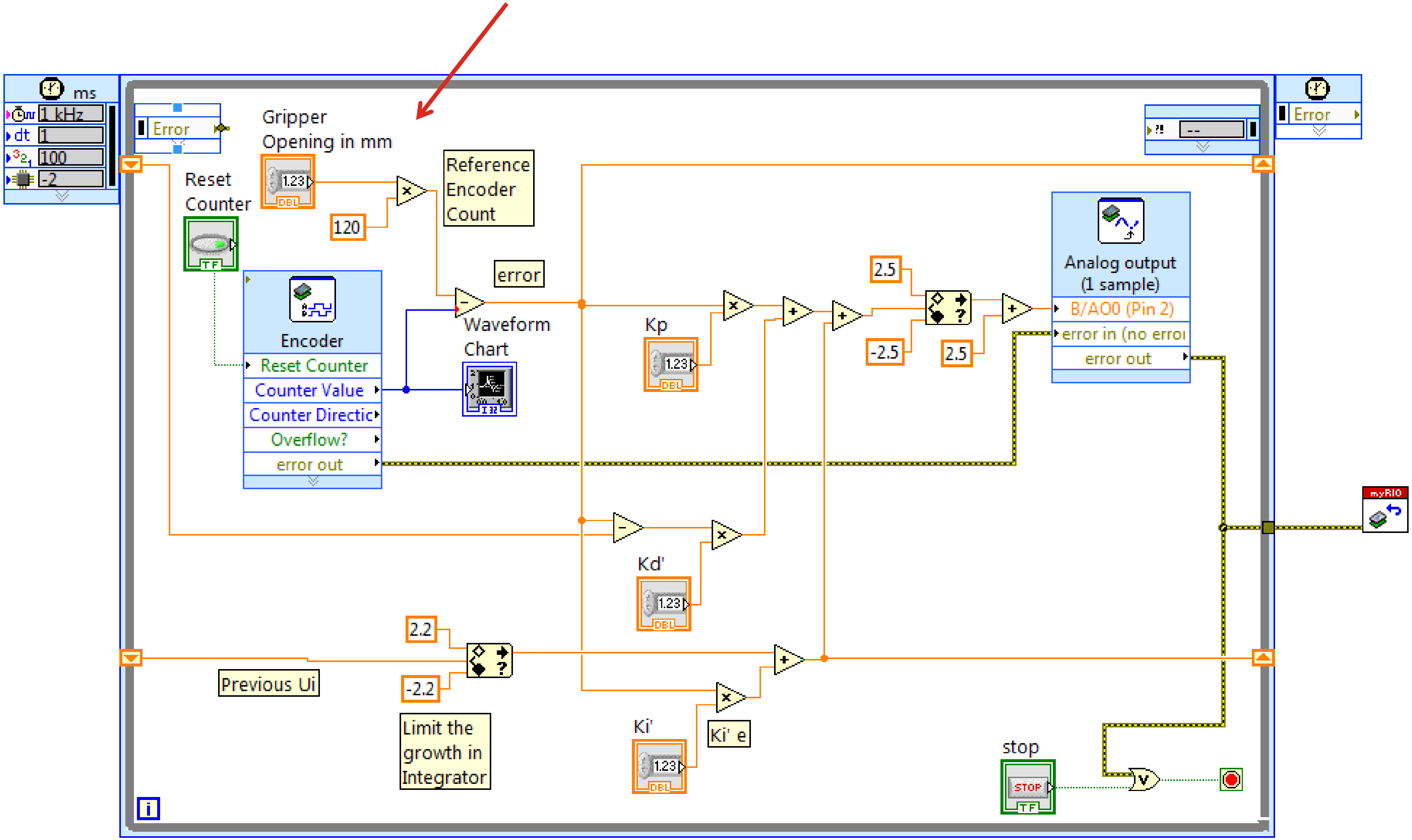
- For the given motor, the following controller parameters work well:
- Kp = 0.1 as previously used.
- When Ki = 0, there was steady state error.
- Tuned Ki = 0.001 and the error diminished.
- However, there is oscillation when step change between 1000 and 2000.
- Tuned Kd = 1 and the oscillation reduced.



Using Encoder for Your Gripper

- How do you use this for your gripper project?
- You need to find out the relationship between the encoder counts and your gripper opening.
- For e.g. 50mm means 6000 counts, i.e. 1mm = 6000/50 = 120 counts
- Then you can add this information in your code (see next page).

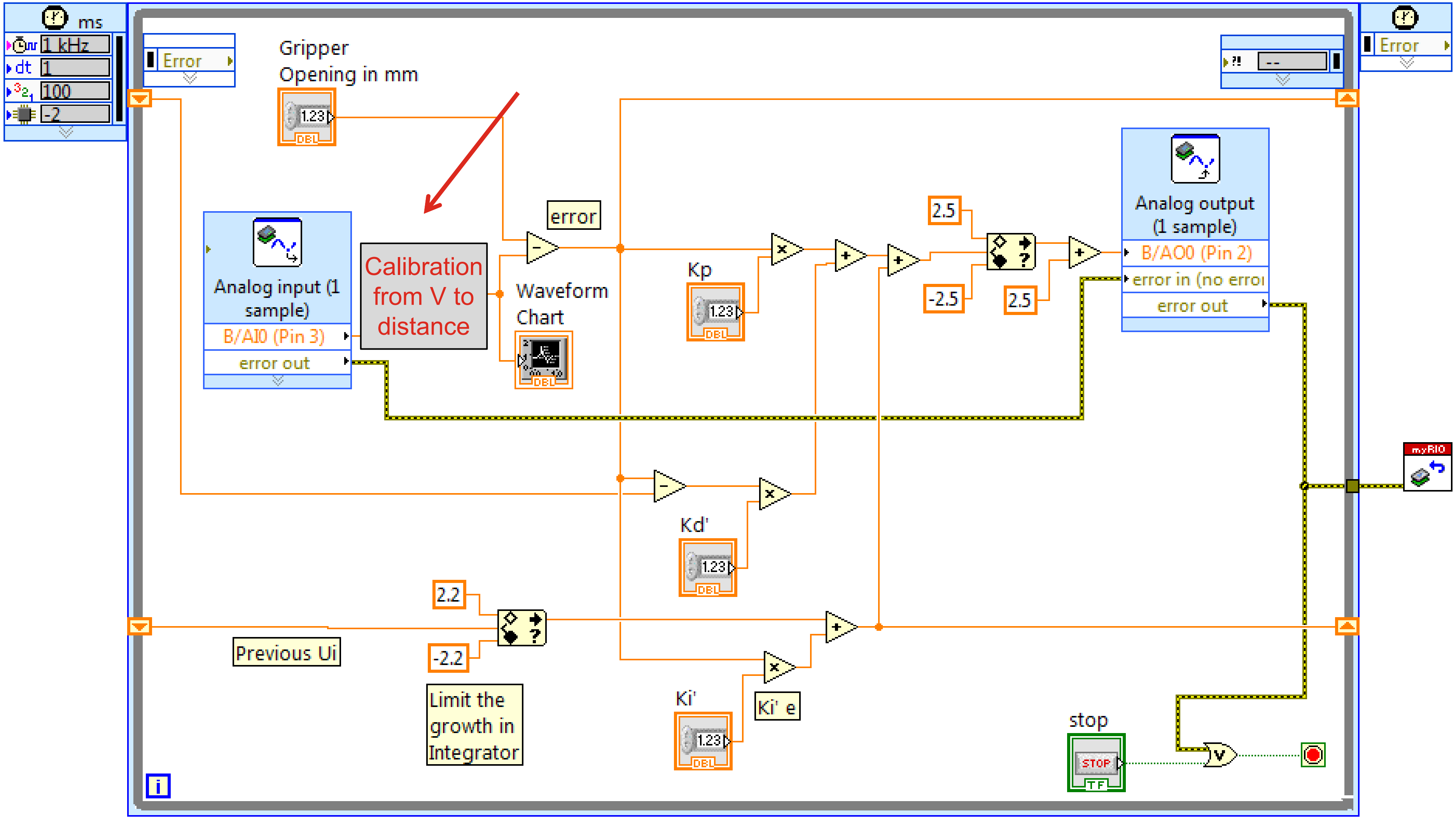
Using Encoder for Your Gripper



Position Control of Motor – IR

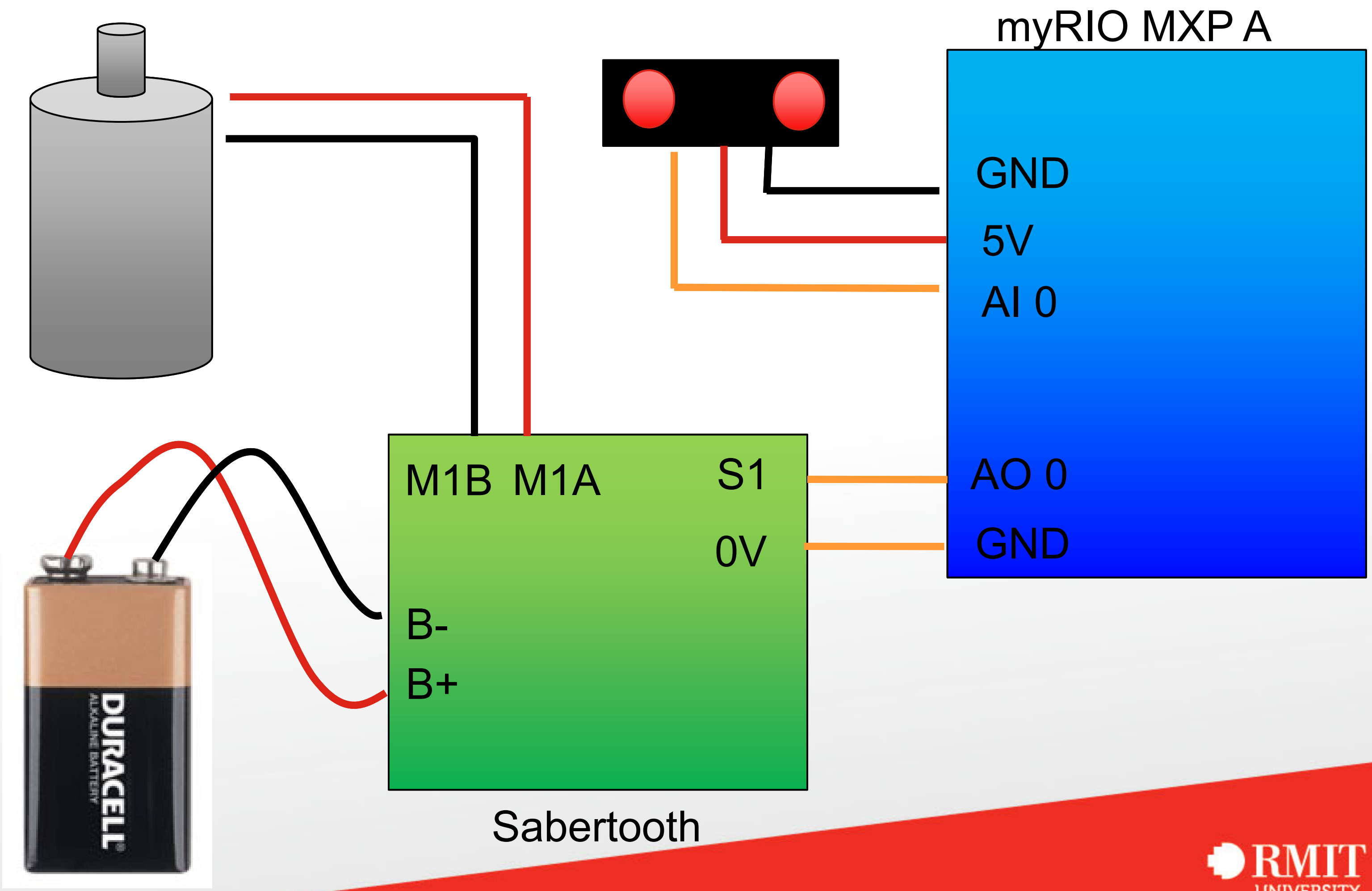
- Some teams might use IR range sensor to measure the gripper opening.
- So let's see how we can code this up.
- Save the previous file as "PositionControlIR.vi"
- We will make changes in the sensing portion of code. (see next page).

Position Control of Motor – IR



Position Control of Motor – IR

- Now, connect your motor to Sabertooth and/or myRIO as follows:



Position Control of Motor - IR

- Please follow the following sequence (very important!)
  - Set Gripper Opening as 10mm (or the minimum range of sensor).
  - Manually open/close gripper or turn the motor until it is indeed 10mm. Reason, we want error to start as zero.
  - Set Kp, Ki and Kd as 0.
  - Click Play.
  - Plug in the battery.
  - Change Reference Position (e.g. 20mm).
  - Observe the response in Waveform Chart.
  - Tune Kp, Ki, Kd until you are satisfied with the result.
    - You may need to set Reference Position as 20, then 10, then 20… for a few times during the tuning.
  - When you are ready to turn this off:
    - Unplug the battery.
    - Click Stop in program.

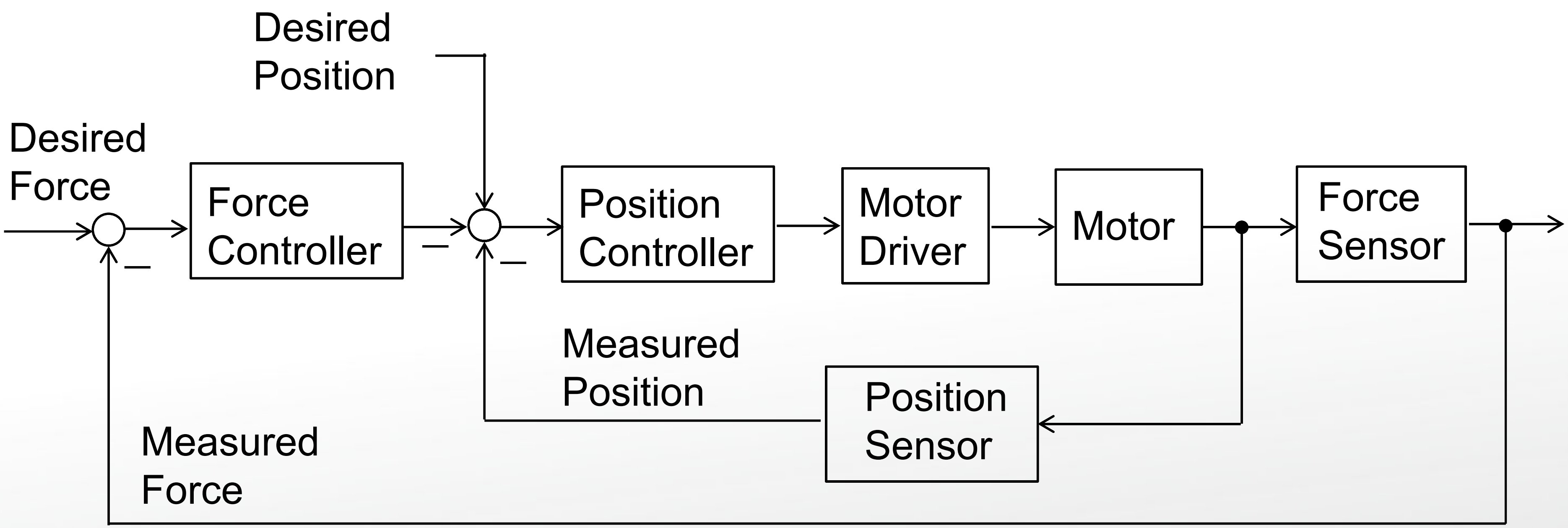
RMIT UNIVERSITY

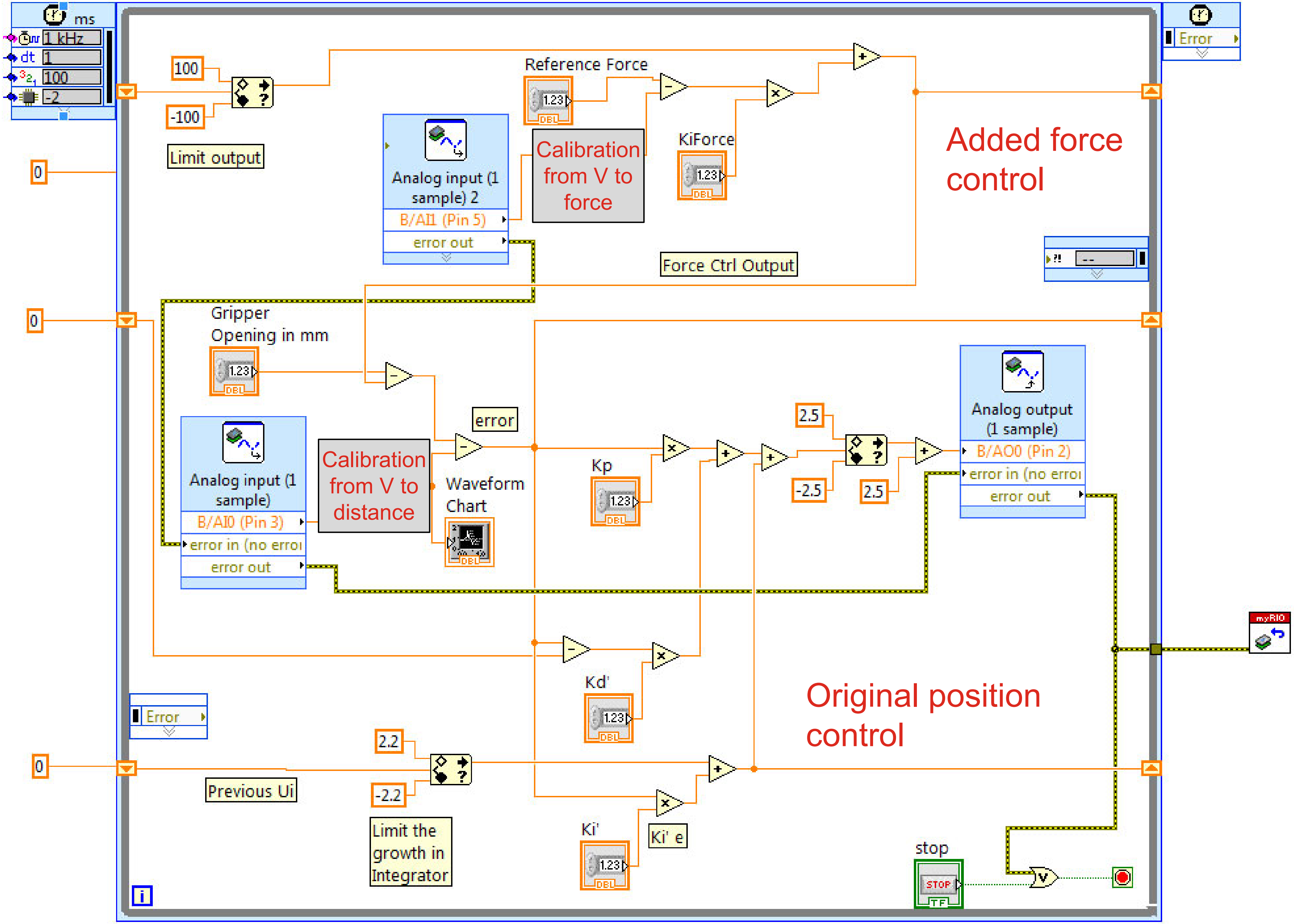
Content

- Pulse Width Modulation & Motor Drivers
- Position Control of DC Motors
    - myRIO Implementation
- Force Control of DC Motors
    - myRIO Implementation

Force Control Implementation

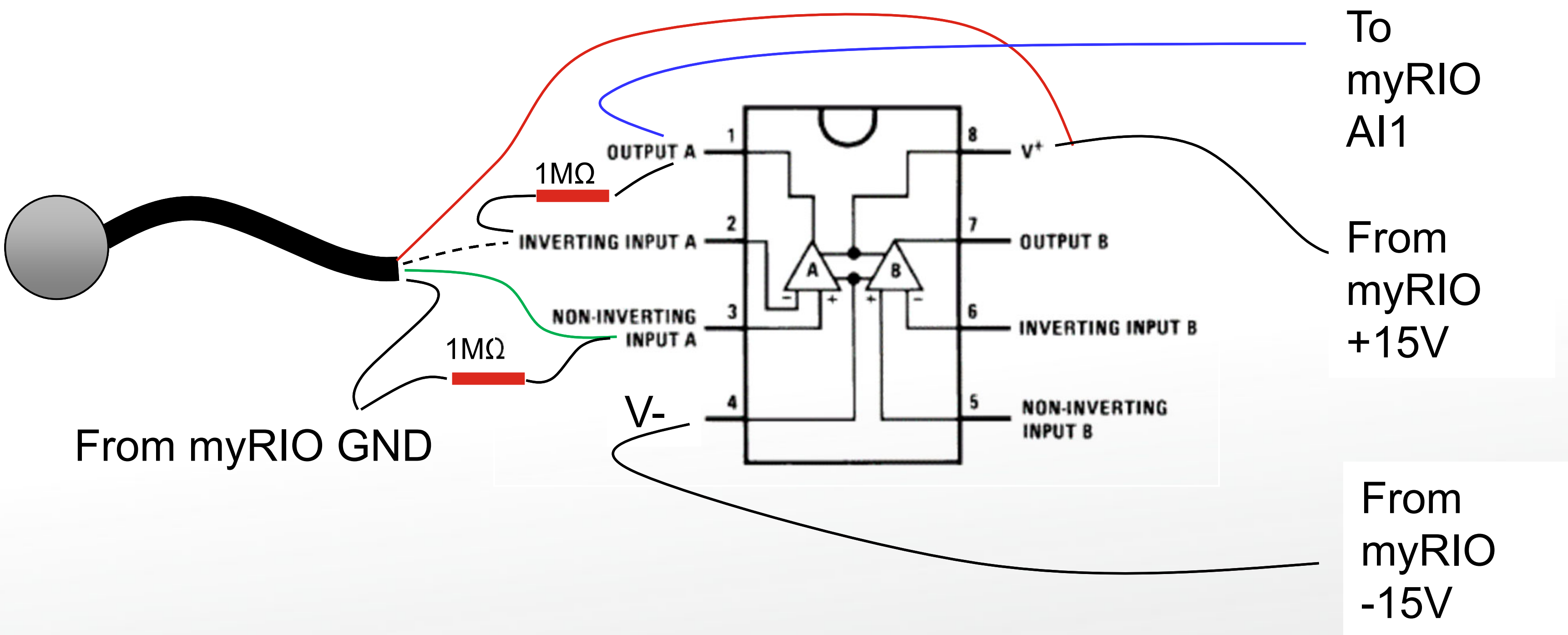
- To perform force control, we can use the cascaded control method:
  - Inner position control loop – This should be properly tuned first!
  - Outer force control loop




ms
1 kHz
dt
1
100
-2
Error
100
-100
Limit output
0
Reference Force
Analog input (1 sample) 2
B/AI1 (Pin 5)
error out
Calibration from V to force
KiForce
Force Ctrl Output
Added force control
0
Gripper Opening in mm
Analog input (1 sample)
B/AI0 (Pin 3)
error in (no error)
error out
Calibration from V to distance
error
Waveform Chart
Kp
2.5
-2.5
2.5
Analog output (1 sample)
B/AO0 (Pin 2)
error in (no error)
error out
Kd'
Error
Original position control
0
Previous Ui
2.2
-2.2
Limit the growth in Integrator
Ki'
Ki' e
stop
STOP
myRIO

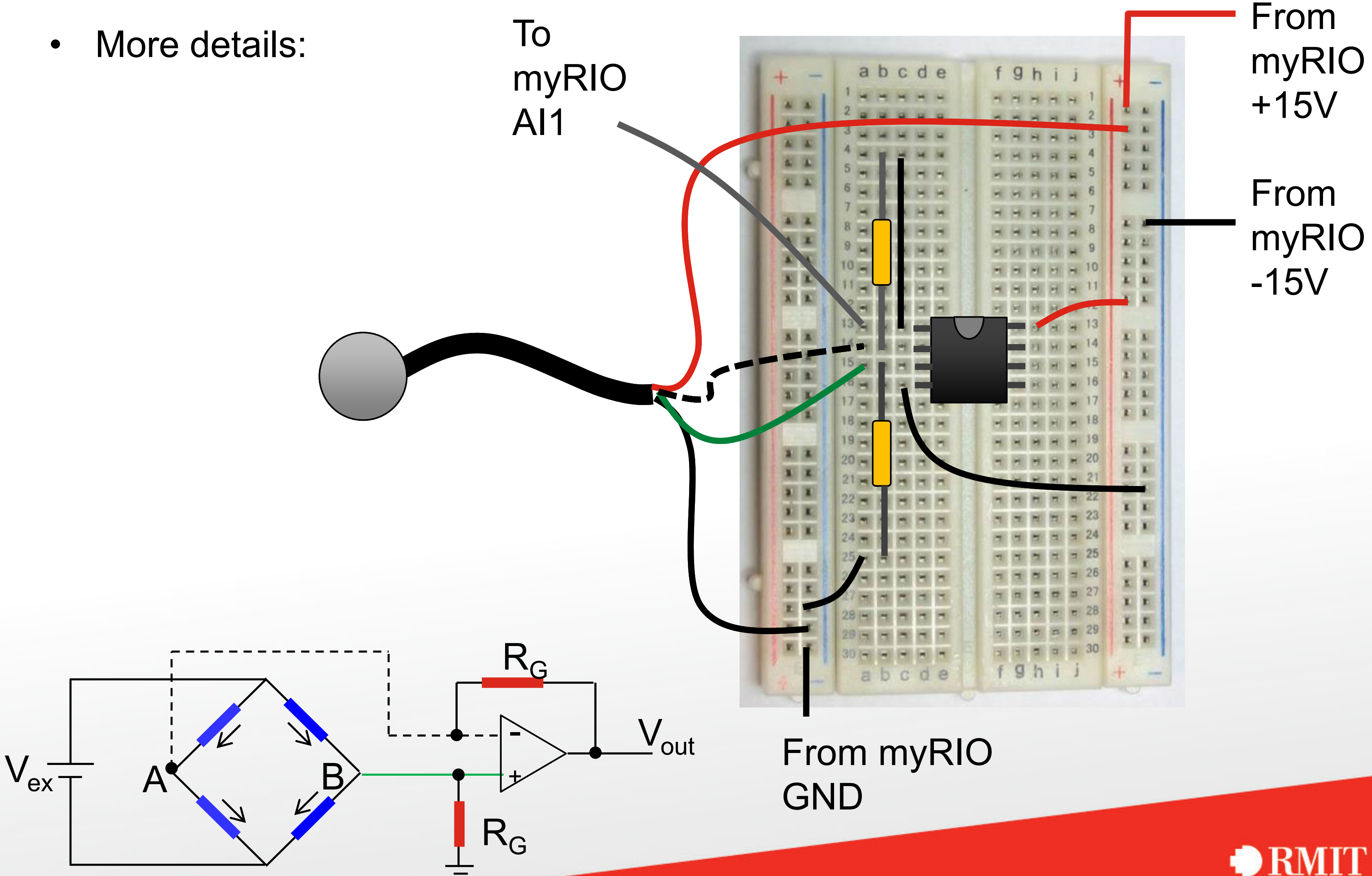
Force Control Implementation

- Connections for myRIO experiment:
  - Just the force sensing part; position sensing same as previous slides (either encoder or IR sensor).



Force Control Implementation

- More details:

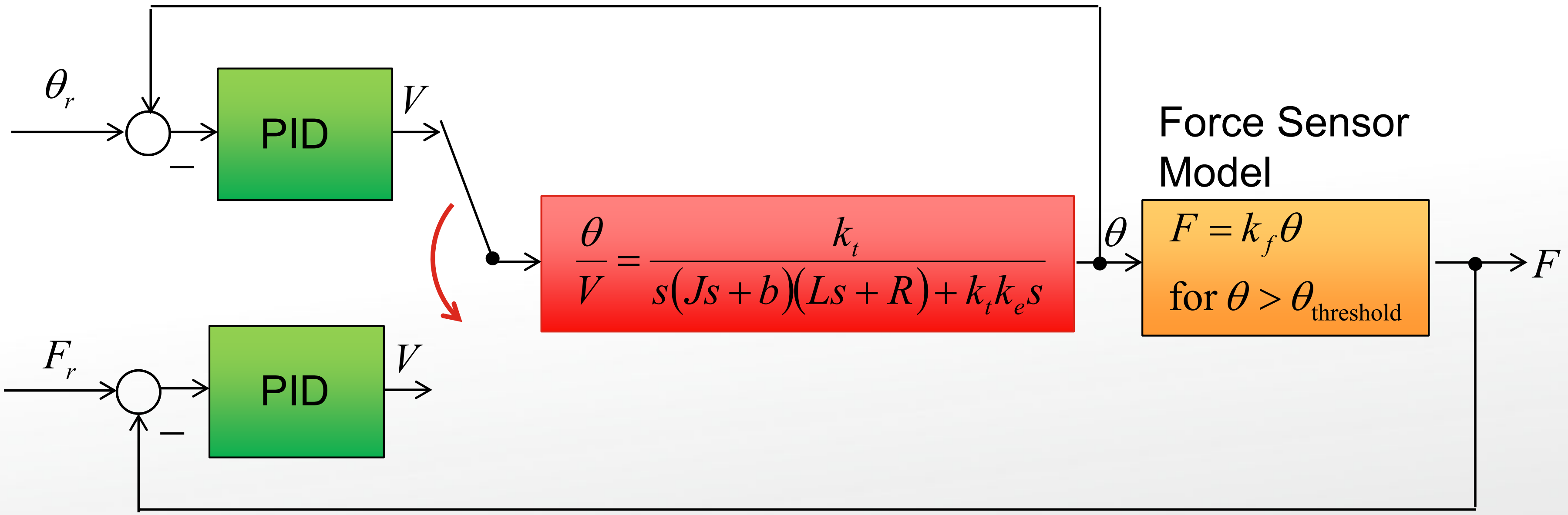


Force Control Implementation

- Please follow the following sequence (very important!)
  - Set Gripper Opening as 10mm (or the minimum range of IR sensor).
  - Manually open/close gripper or turn the motor until it is indeed 10mm. Reason, we want error to start as zero.
  - Use the same Kp, Ki, Kd as tuned before.
  - Set Reference Force = 0, and KiForce = 0
  - Click Play.
  - Plug in the battery.
  - Change Reference Force (e.g. 10N)
  - Observe the response in Waveform Chart.
  - Tune KiForce until you are satisfied with the result.
  - When you are ready to turn this off:
    - Unplug the battery.
    - Click Stop in program.

Exercise – At your own free time

- Please take the code for position control using encoder, and expand it for force control.
- Apart from the cascaded control method, you may also try out the Hybrid position/force control method, which uses a "switch" to determine either position or force control.



Thank you!

Any questions?