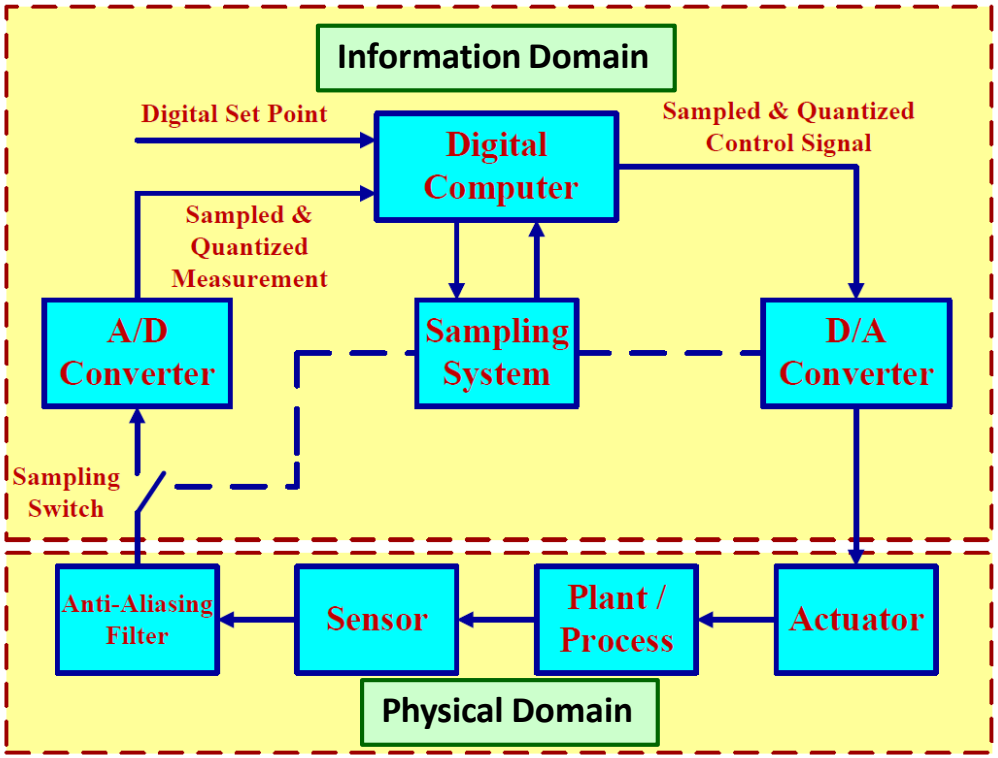
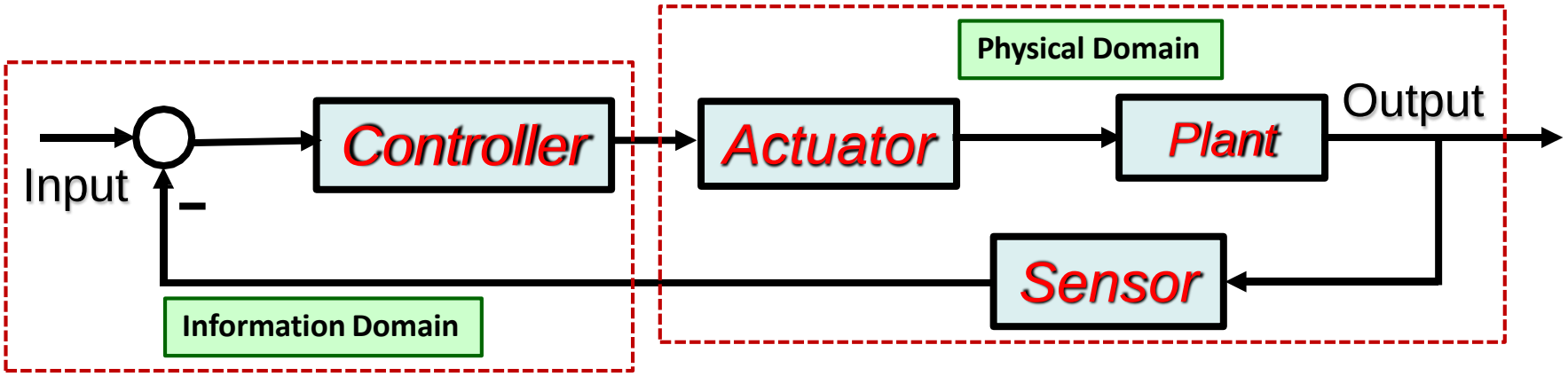
MAE/ECE 5320 Mechatronics

2025 Spring semester

Lecture 08

Information and Physical Domains





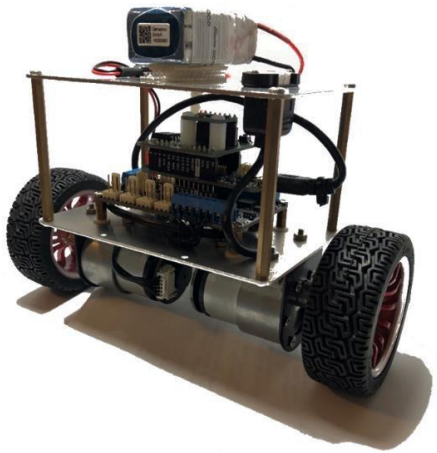
Content

- Mini Segway system model
  - DC Motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

Content

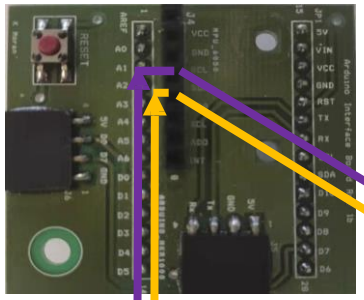
- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

Mini Segway system (1)

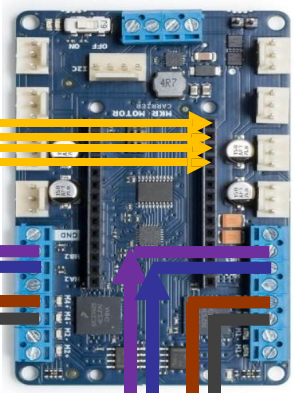


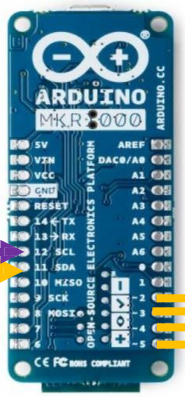
Mini Segway electronic hardware

interface board

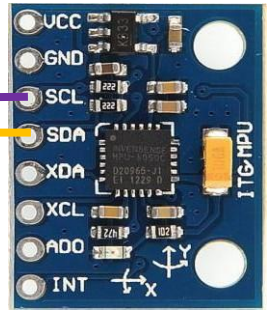


Motor drive

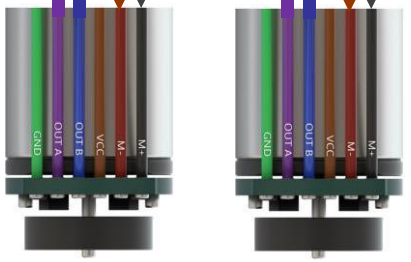




Arduino
MKR 1000

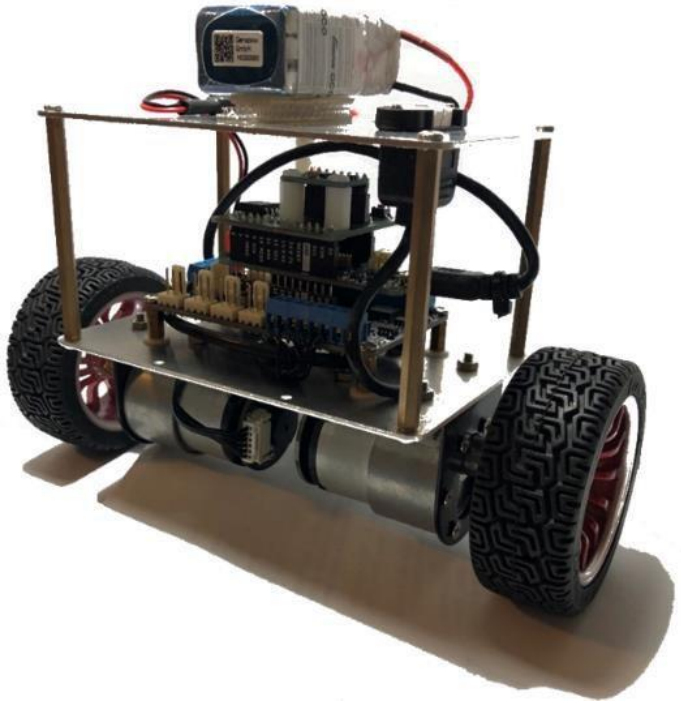


MPU 6050
Gyro & acce sensor



Motors with encoder

Mini Segway system (2)



Mechanical system model is divided into the following three groups and modeled one-by-one.

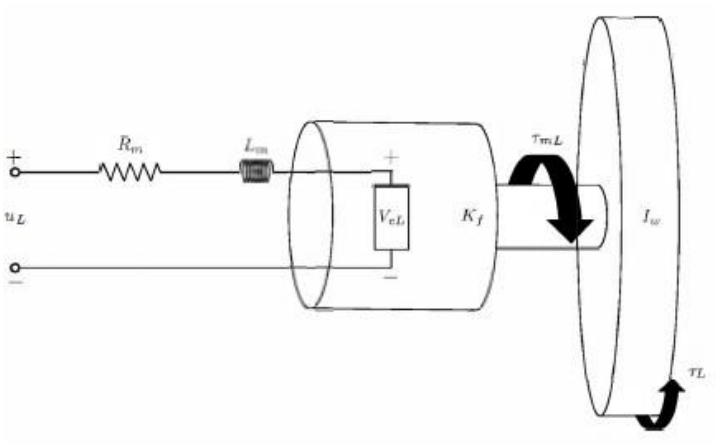
- ❑ DC motor
- ❑ Wheel and cart
- ❑ Inverted pendulum

Content

- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

DC Motor model (1)

Left and right motors are identical and located oppositely and only one motor is considered, where the subscript "L" denote for left motor and the right motor will be denoted by "R" .



When the current i_L is applied to motor, the motor torque is

$$\tau_{mL} = K_\tau i_L$$

Generated EMF voltage is

$$V_{eL} = K_{emf}\omega_L$$

And the Kirchhoff's voltage law leads to

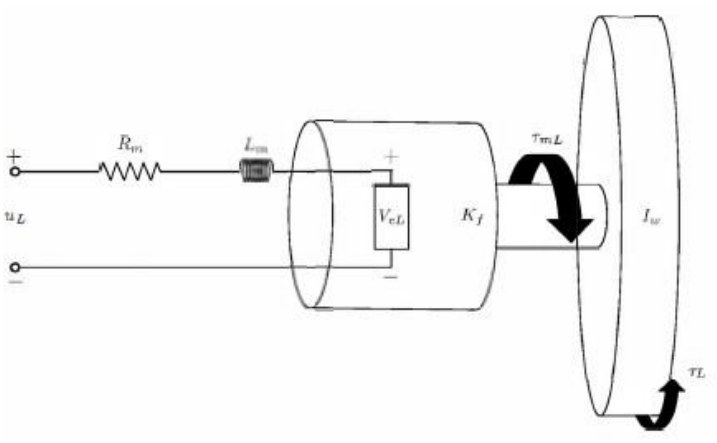
$$u_L - R_m i_L - L_m \frac{di_L}{dt} - V_{eL} = 0$$

Since $R \gg L_m$, letting $L_m = 0$ leads to

$$i_L = \frac{u_L}{R_m} - \frac{K_{emf}}{R_m}\omega_L \qquad (1)$$

DC Motor model (2)

Left and right motors are identical and located oppositely and only one motor is considered, where the subscript "L" denote for left motor and the right motor will be denoted by "R" .



DC motor torque equation (Newton's law)

$$\sum M = \tau_{mL} - K_f\omega_L - \tau_L = I_\omega \frac{d\omega_L}{dt} \qquad (2)$$

$$\Rightarrow \quad \frac{d\omega_L}{dt} = \frac{K_\tau}{I_\omega} i_L - \frac{K_f}{I_\omega}\omega_L - \frac{\tau_L}{I_\omega} \qquad (3)$$

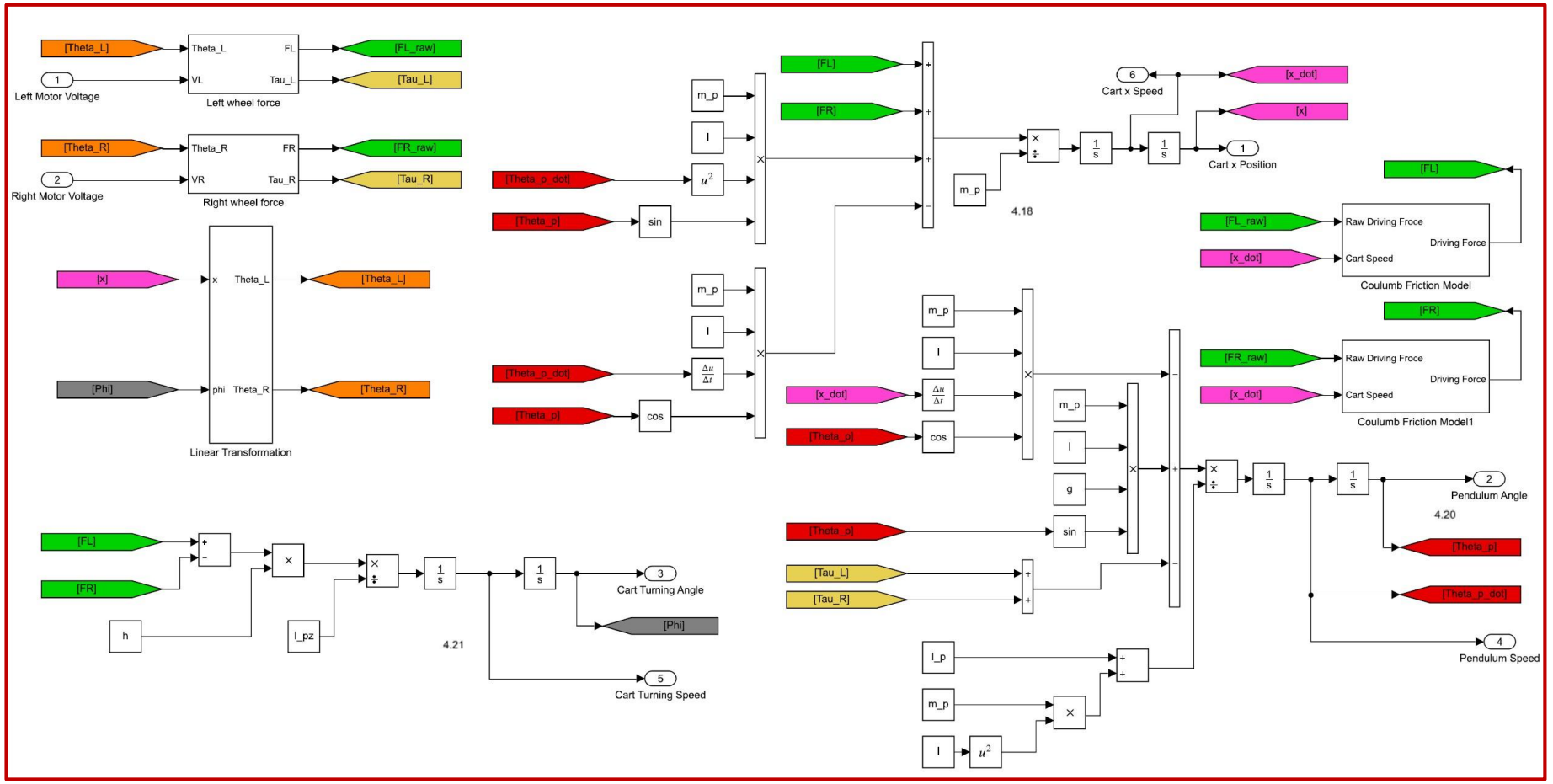
Substituting (1) to (3) yields

$$\Rightarrow \quad \frac{d\omega_L}{dt} = \frac{K_\tau}{I_\omega}\left(\frac{u_L}{R_m} - \frac{K_{emf}}{R_m}\omega_L\right) - \frac{K_f}{I_\omega}\omega_L - \frac{\tau_L}{I_\omega} \qquad (4)$$

Combining (2) and (4) leads to

$$\tau_{mL} = \frac{-K_\tau K_{emf}}{R_m}\omega_L + \frac{K_\tau}{R_m}u_L \qquad (5)$$

Segway model Simulink implementation



Content

- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

Segway control (1)

Model-based control:
Pole-placement;
LQR/LQG;
Optimal/Robust control

Model based control features:

- ❑ Accurate modeling is required that is time-consuming for development
- ❑ Relatively complex structure, but is able to address multiple-input-multiple-output (MIMO) system
- ❑ Control tuning is accomplished through design matrices
- ❑ Closed-loop system performance is assured if there is no modeling error, that is accurate model is required.

Normally covered in graduate-level class

Model-free control:
PID;
Data-driven;
Learning-based control

Model free control features:

- ❑ Requires no model;
- ❑ Need a tuning process for PID control or a learning process for learning-based control. This process can be time-consuming and low-efficient.
- ❑ Limited to single-input-single-output system for PID control
- ❑ Closed-loop system performance is achieved by control tuning that is heavily dependent on experience.

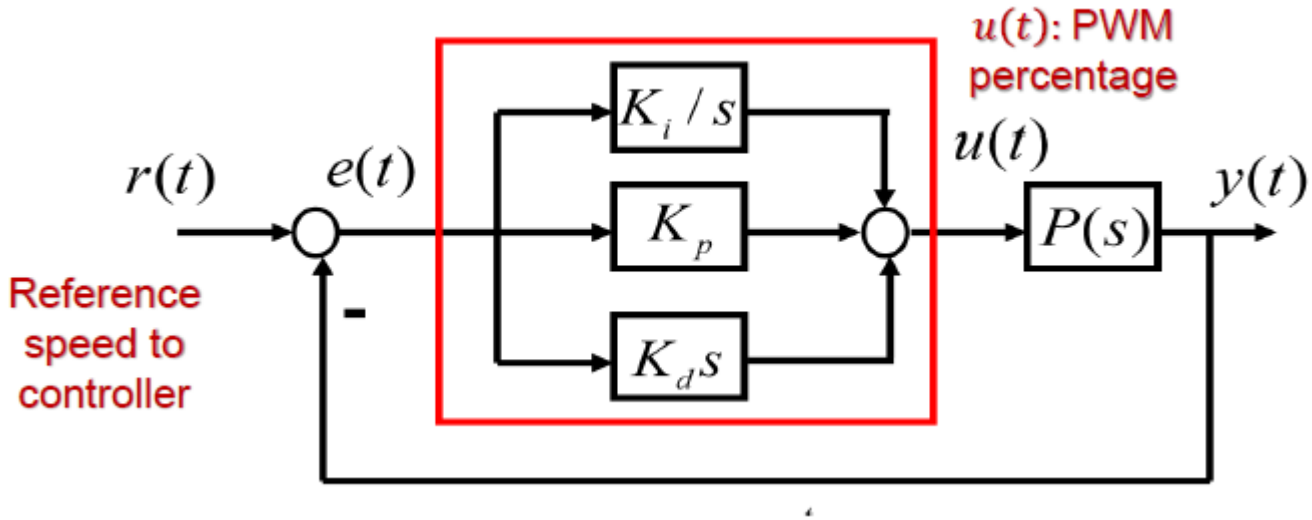
To be used in this class

Content

- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

PID control review

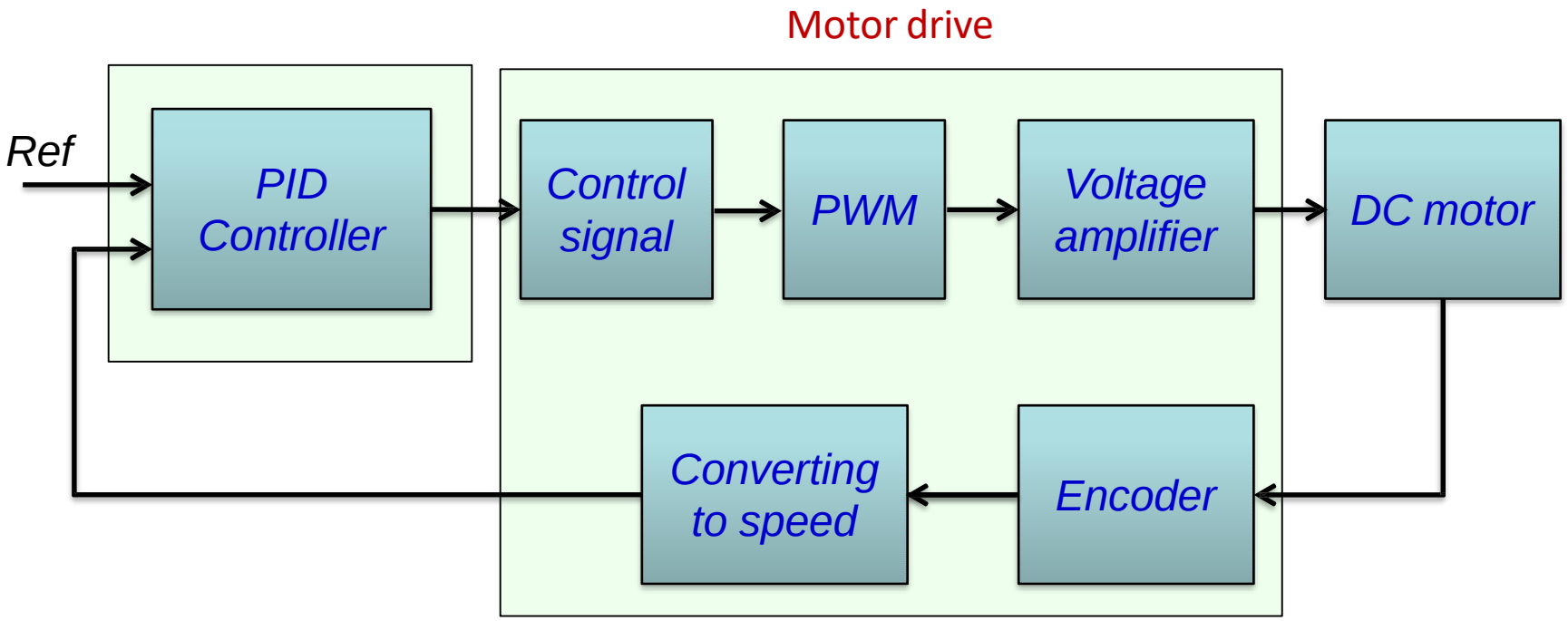
Closed-loop system with a PID controller.



PID gains need to be well tuned to achieve desired performances.

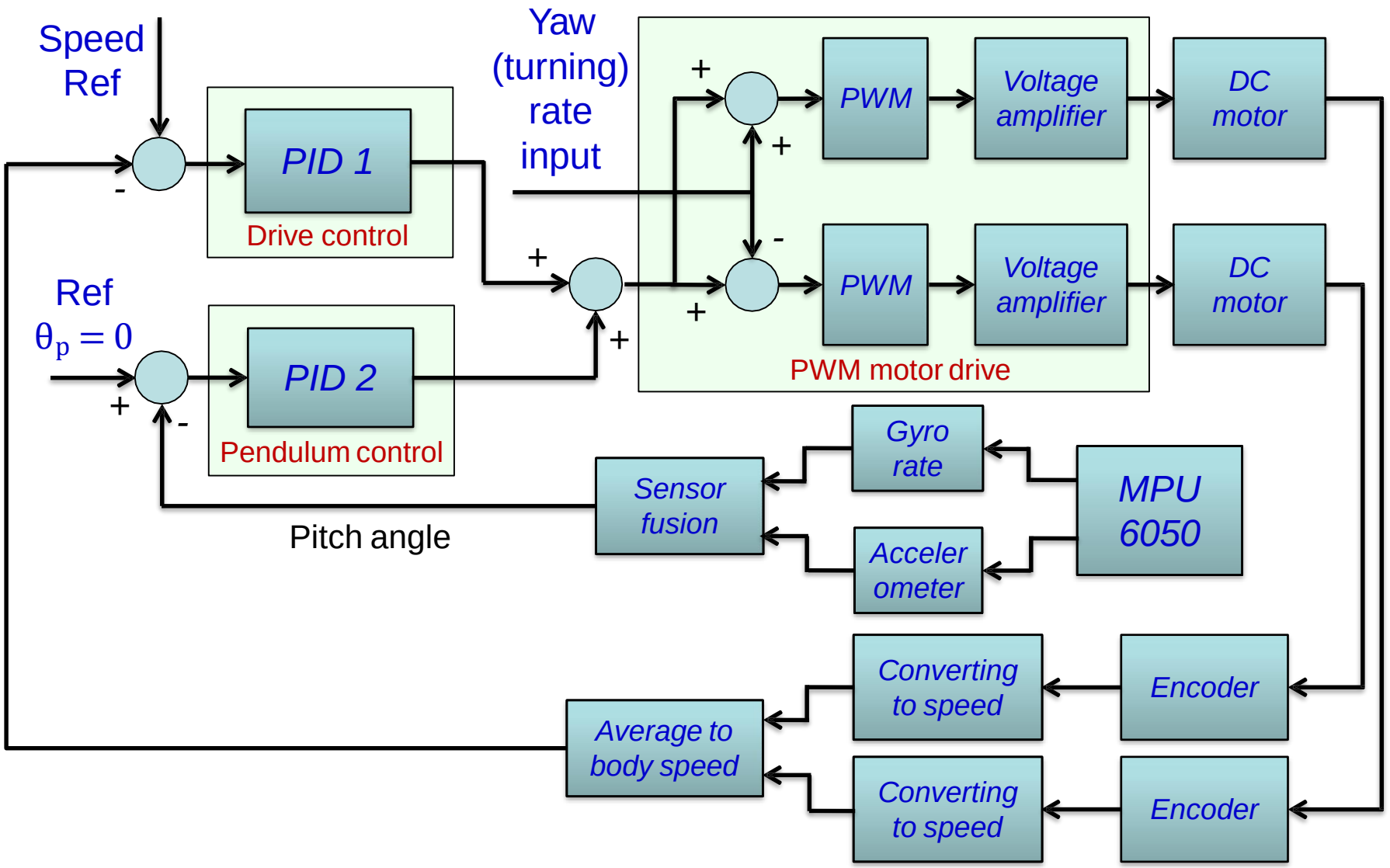
PID speed control of Segway (1)

Recall the closed-loop speed control system for one motor.



PID speed control of Segway (2)

Closed-loop control system for a two-motor Segway



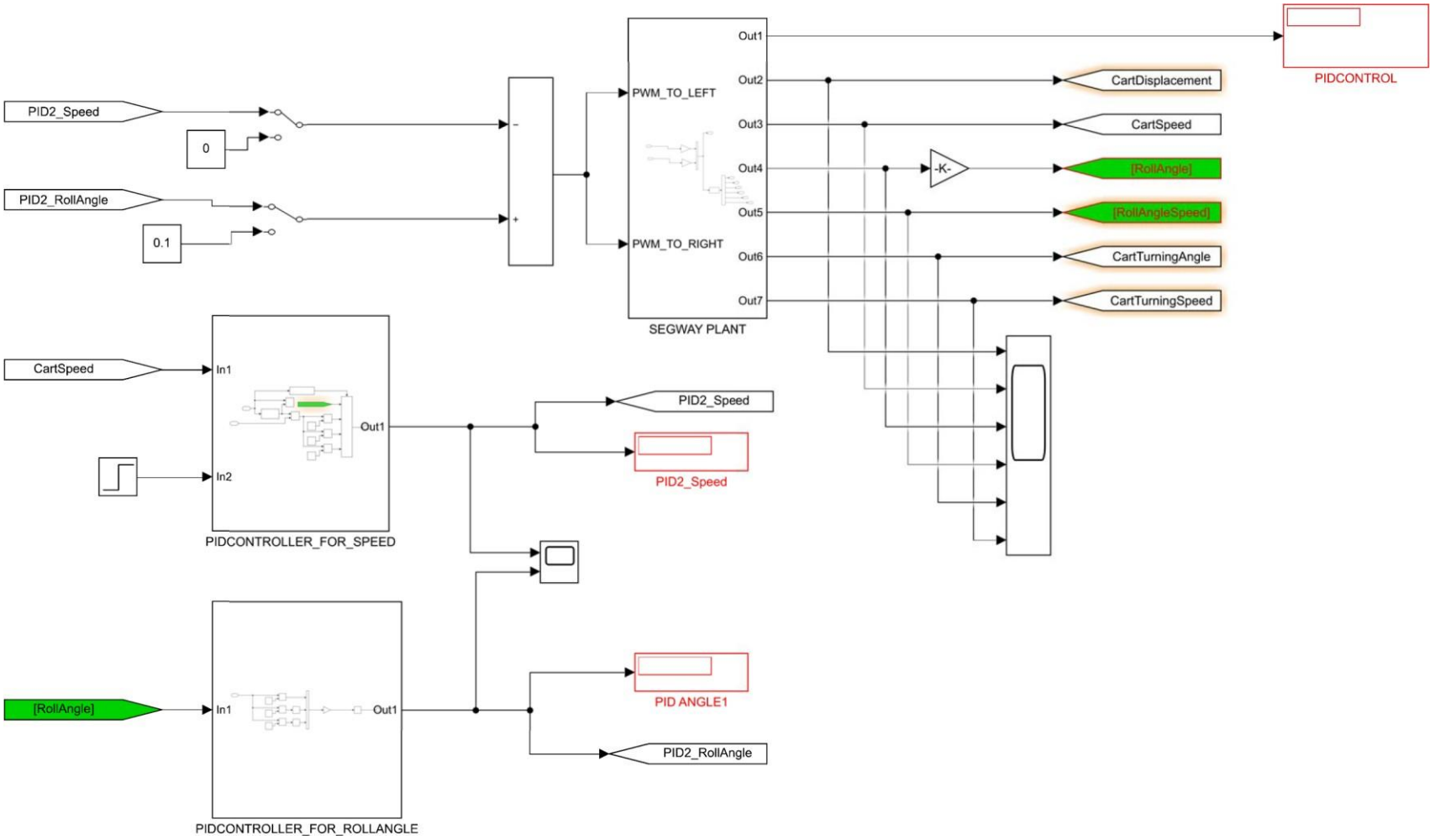
PID speed control of Segway (3)

Remarks:

- Two PID controllers are used to stabilize Segway and follow given speed reference (forward and backward). The turning is controlled in open-loop.
- As discussed, the pitch angle dynamics and Segway speed dynamics are NOT independent, thus PID controllers for these two dynamics will influence each other. In other words, two PID controllers are NOT independent. To avoid double integration, the pendulum PID integration gain is set to zero.
- Tuning of PID controllers can be conducted iteratively, meaning that, fix one PID gains and tune the other one, then fix the tuned gains and tune the first PID controller iteratively. Repeat until satisfactory performance is achieved.
- If two PID controllers were used for controlling two motors (actuator), respectively, tuning work would be even more difficult. This is one of the drawbacks mentioned early.

PID speed control of Segway (4)

PID control gain tuning in Simulink based on linearized Segway model

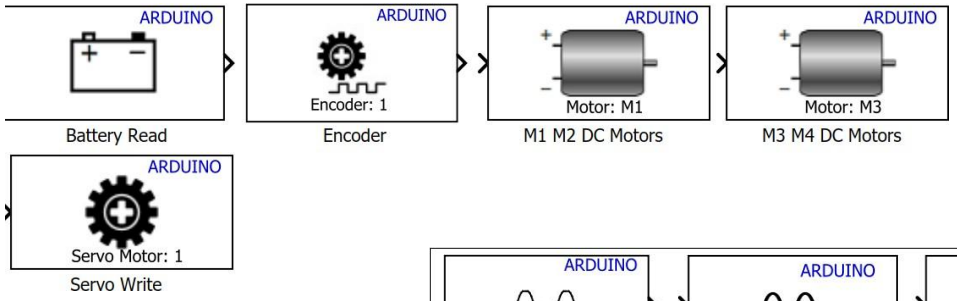


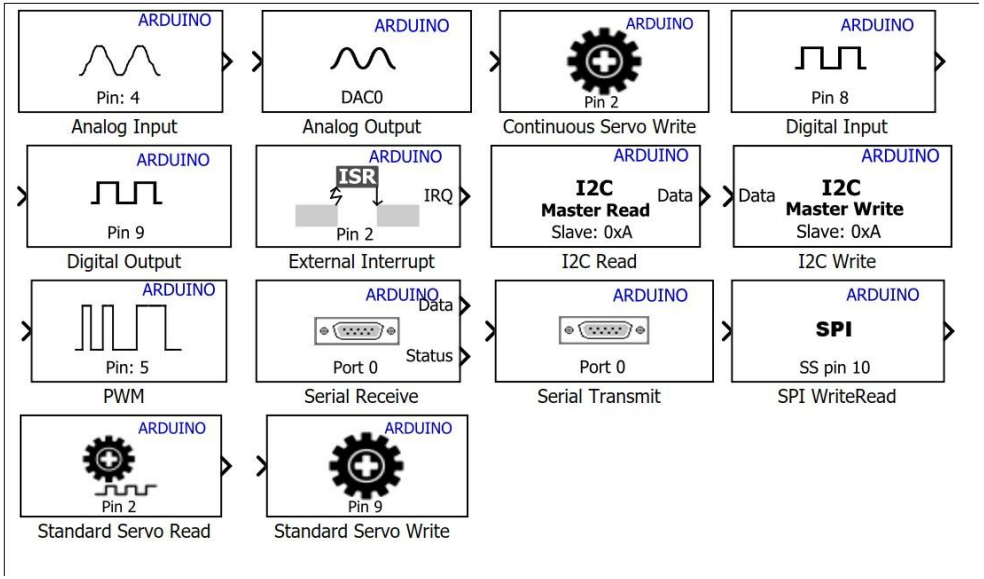
Content

- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

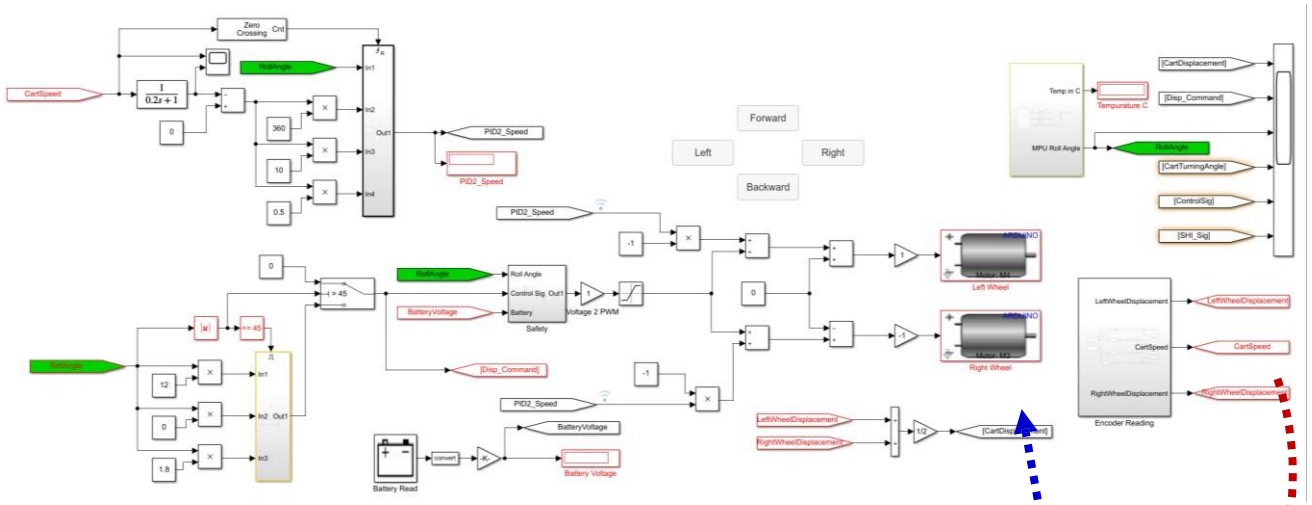
Hardware implementation

MATLAB/SIMULINK provides a platform for implementing mechatronics hardware, including pin specification, DC motor/servo motor module, MPU sensor module, encoder module, WiFi/serial communication, etc.





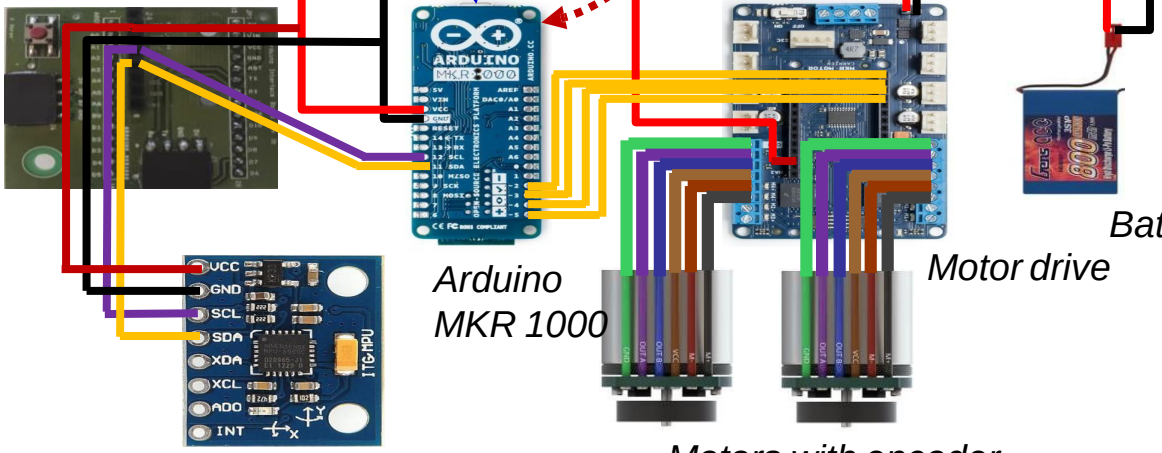
Segway control system hardware



wifi

serial

interface board



Arduino
MKR 1000

Motor drive

Battery

Motors with encoder

MPU 6050

Gyro&acce sensor

Hardware implementation (1)

- Relax the computational burden of Arduino microprocessor, the PID controllers are implemented in SIMULINK in the host computer and the computed control signals are passed to Arduino by WiFi communication.
- Programming and debugging are conducted on Simulink blocks. Hardware-based codes are automatically converted and burned to Arduino board. (Lab5, monitor and tune mode)
- At the same time, sensor readings are measured by Arduino and transmitted to corresponding hardware modules in SIMULINK by Arduino.

Hardware implementation (2)

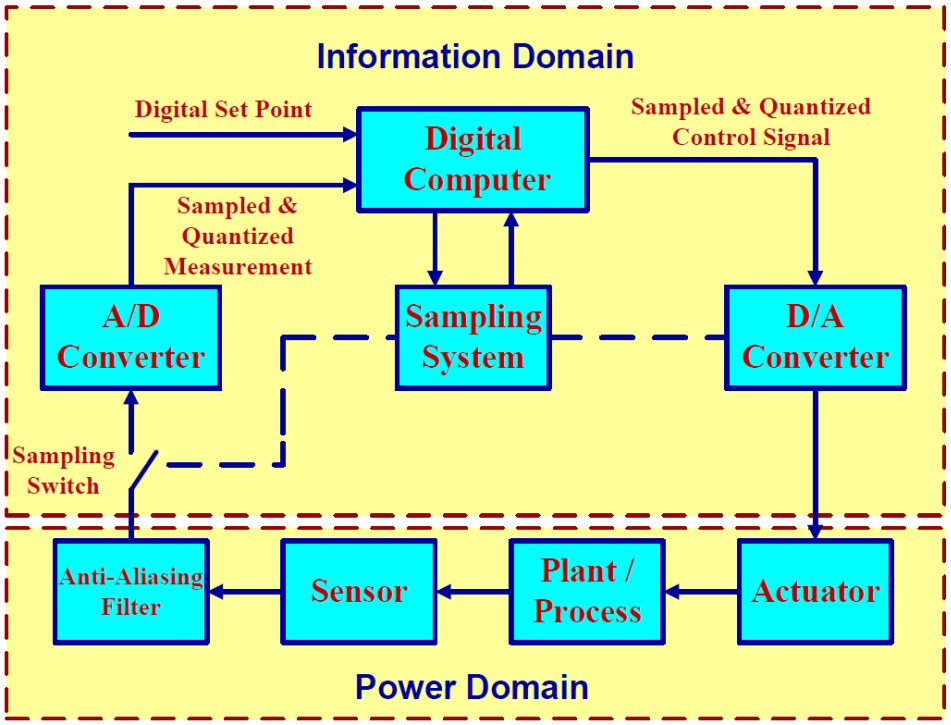
Protection mechanism for the class Segway:

- Motors and wheels are protected from high-voltage and long-time full duty cycle control inputs;
- System is enabled only when Segway pitch angle is less than $45°$ to make sure that when the Segway is about to lose balance, both motors will be stopped by set the duty-cycle to zero;
- System starts working after 8 seconds;
- System is enabled only when battery voltage is greater than 10.5 V.

Content

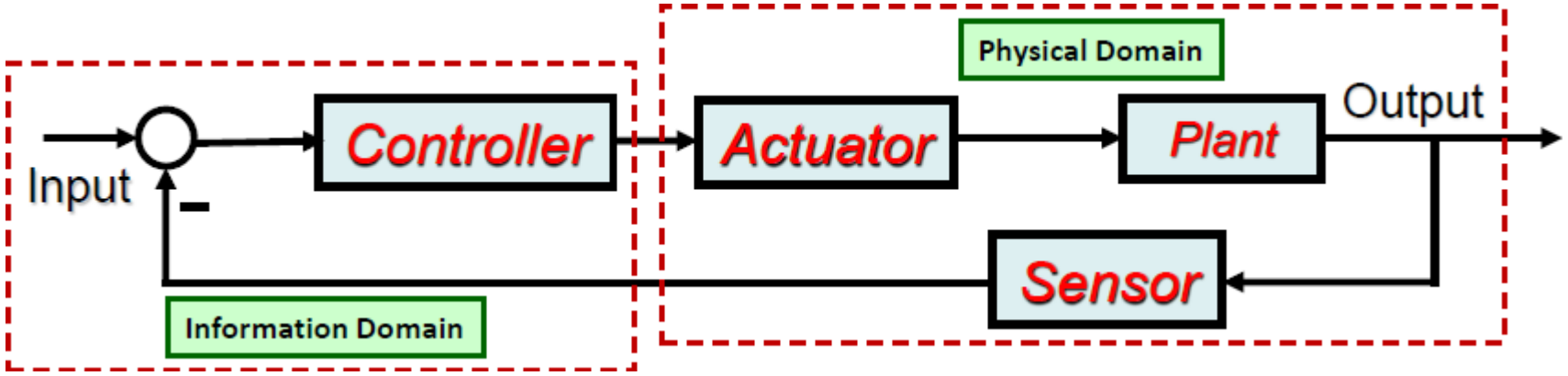
- Mini Segway system model
  - DC motor model
  - Wheel and cart model
  - Inverted pendulum model
- Model-based PID control tuning
  - Inverted pendulum PID loop
  - Driving PID loop
- Control system hardware and software
  - Arduino Simulink I/O library
  - Control system information flow

Information and power flow in Segway

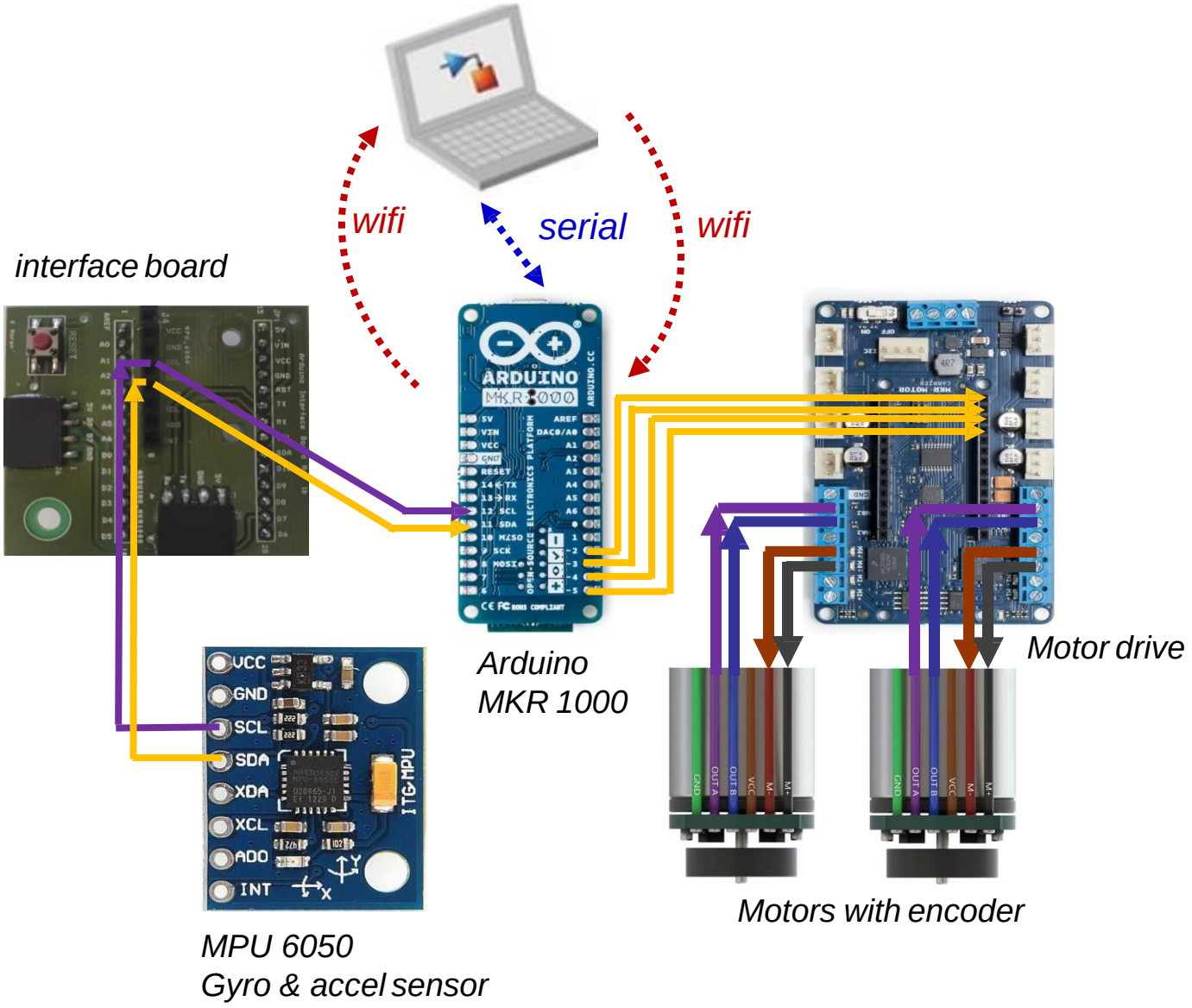


Recall the information and power flow in Segway.

Q: How are they implemented in hardware ?



Information flow in hardware



Power flow in hardware