CS65K Robotics

Modelling, Planning and Control

Chapter 8: Motion Control

Section 8.5-8.7

LECTURE 15: FINITE STATE MACHINE AND CONTROL IMPLEMENTATION

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Hardware Implementation for Robot Control – Control Unit and Data Path
- Robot Control through Finite State Machine
- PID Controller Using FPGA Technology

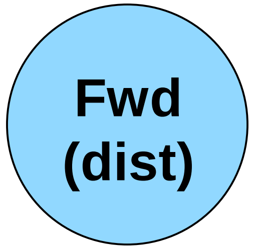
Finite State Machine

SECTION 1

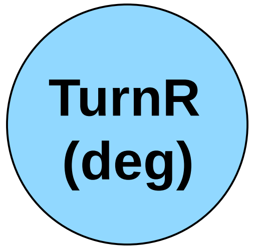
Goals

- Control Design – Finite State Machine
- Case Study of a Finite State Machine Control

Finite State Machines offer another alternative for combining behaviors

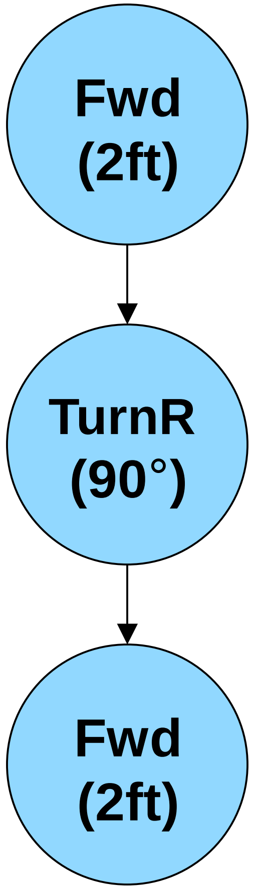


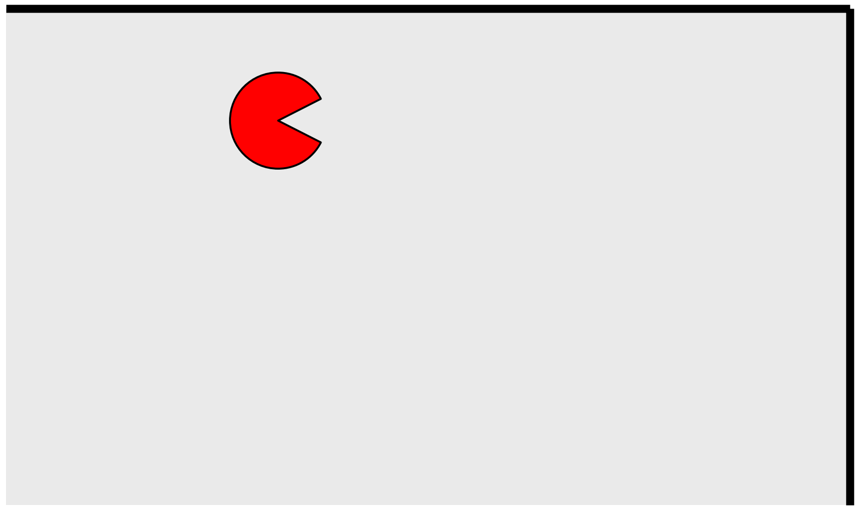
Fwd behavior moves robot straight forward a given distance



TurnR behavior turns robot to the right a given number of degrees

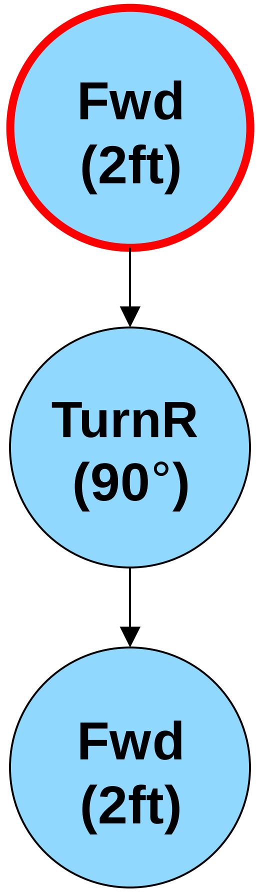
Finite State Machines offer another alternative for combining behaviors

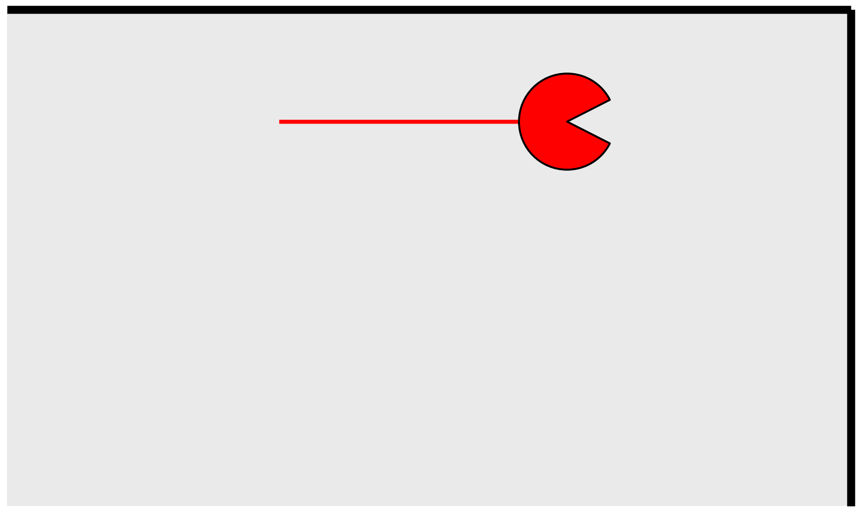




Each state is just a behavior and we can easily link them together to create an open loop control system

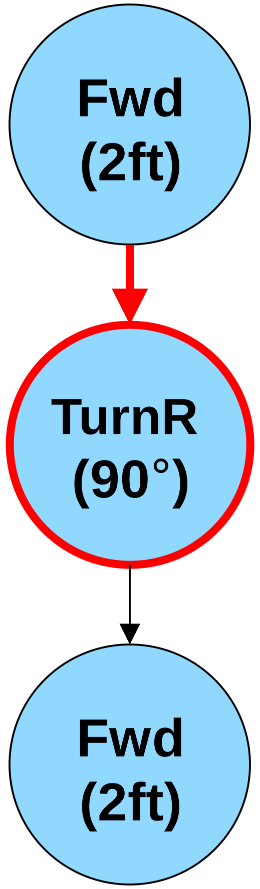
Finite State Machines offer another alternative for combining behaviors

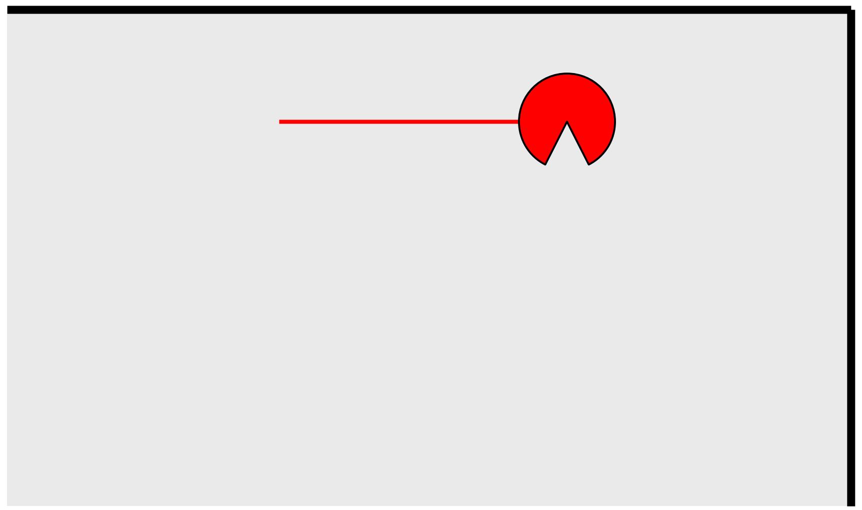




Each state is just a behavior and we can easily link them together to create an open loop control system

Finite State Machines offer another alternative for combining behaviors

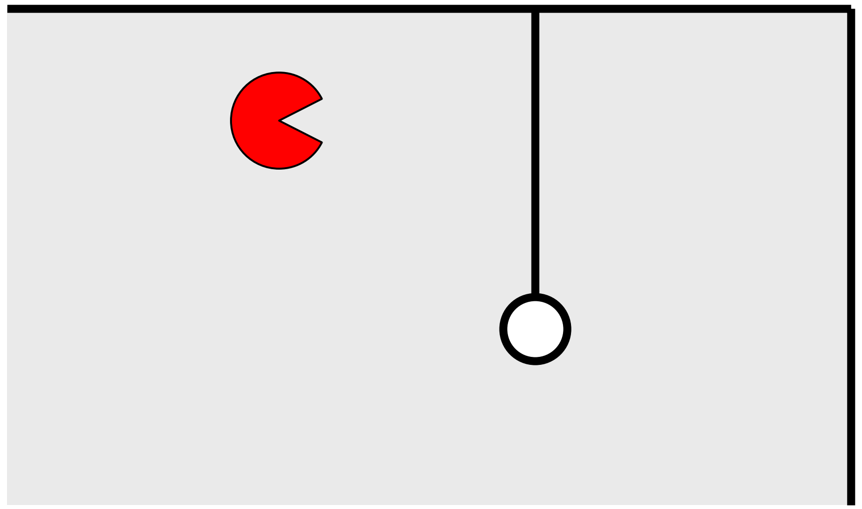




Each state is just a behavior and we can easily link them together to create an open loop control system

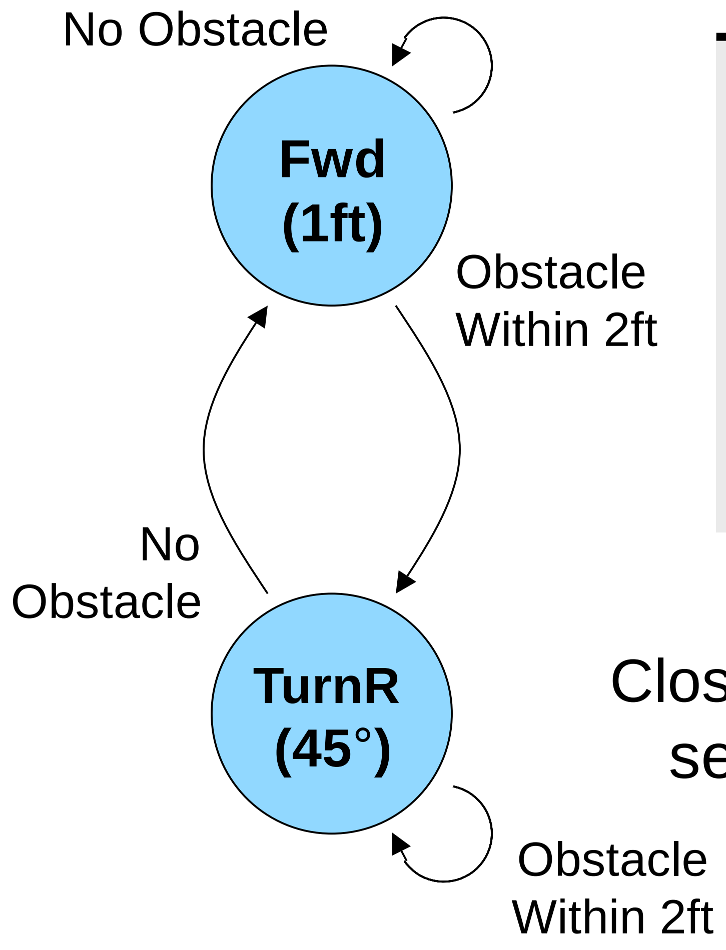
Finite State Machines offer another alternative for combining behaviors

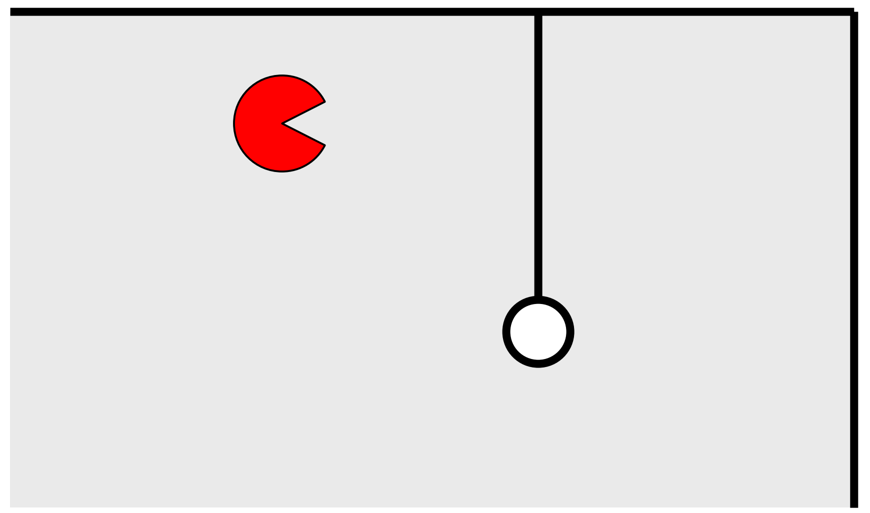
$$\begin{tikzcd}
\boxed{\text{Fwd (2ft)}} \arrow[d] \\
\boxed{\text{TurnR (90°)}} \arrow[d] \\
\boxed{\text{Fwd (2ft)}}
\end{tikzcd}$$



Since the Maslab playing field is unknown, open loop control systems have no hope of success!

Finite State Machines offer another alternative for combining behaviors

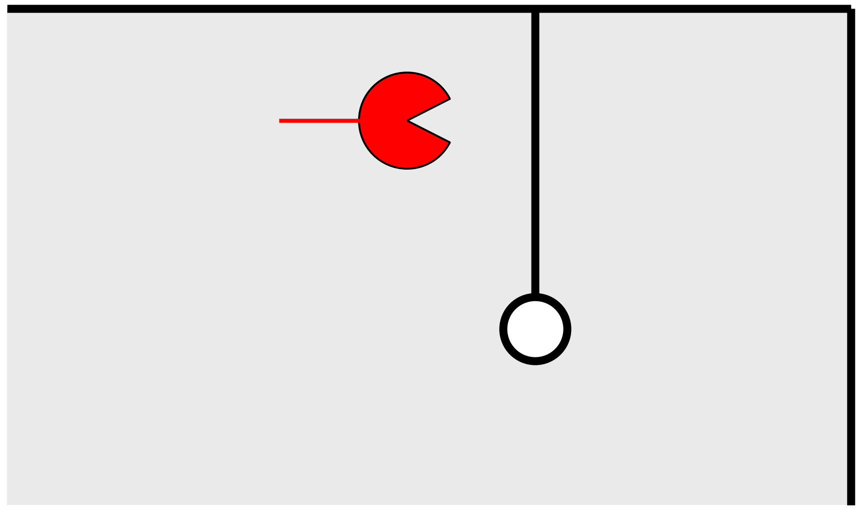




Closed loop finite state machines use sensor data as feedback to make state transitions

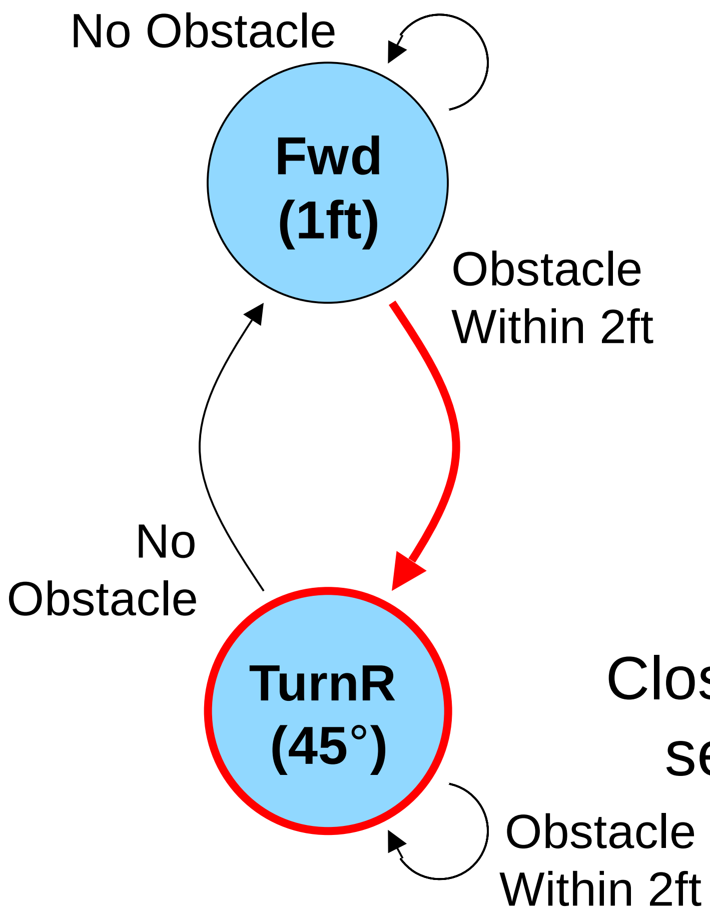
Finite State Machines offer another alternative for combining behaviors

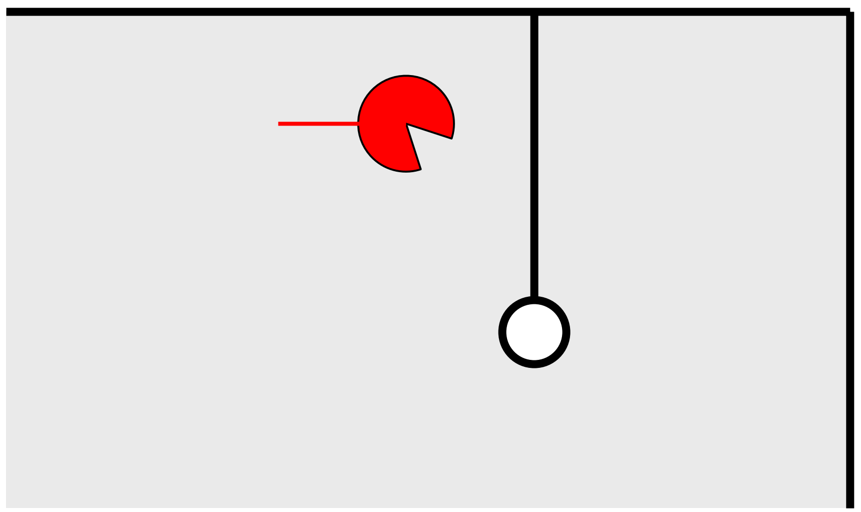
$$\begin{tikzcd}
\text{Fwd (1ft)} \arrow[loop above, "\text{No Obstacle}"] \arrow[d, bend left, "\text{Obstacle Within 2ft}"] \\
\text{TurnR (45°)} \arrow[u, bend left, "\text{No Obstacle}"] \arrow[loop below, "\text{Obstacle Within 2ft}"]
\end{tikzcd}$$



Closed loop finite state machines use sensor data as feedback to make state transitions

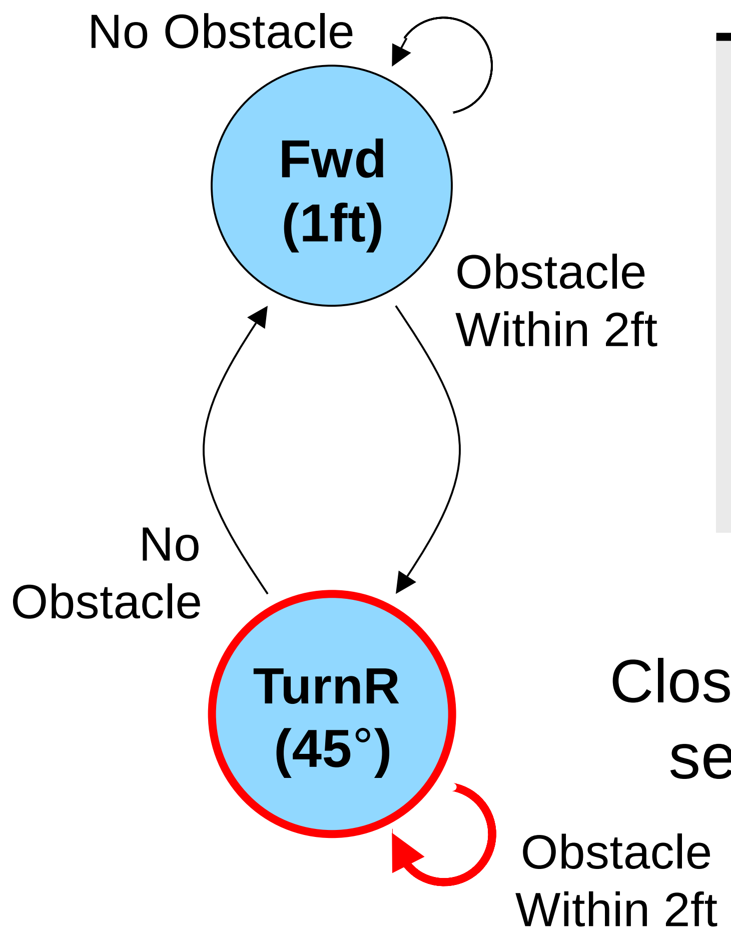
Finite State Machines offer another alternative for combining behaviors

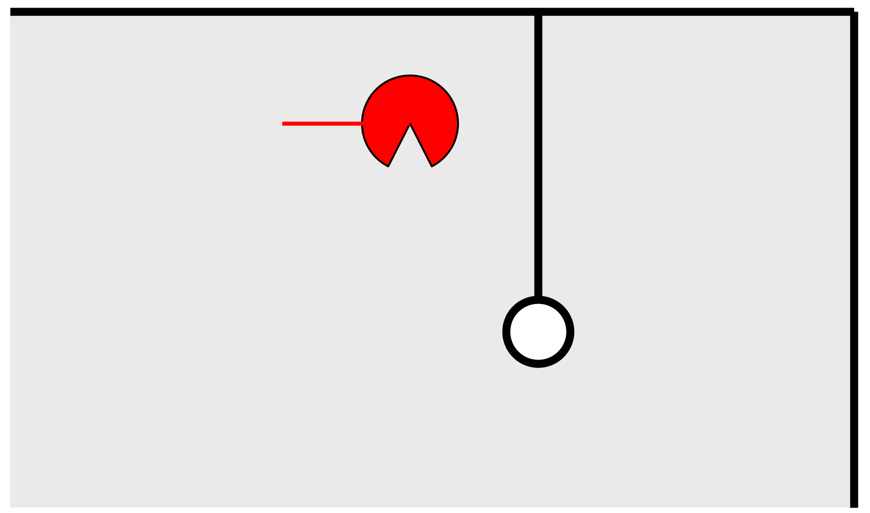




Closed loop finite state machines use sensor data as feedback to make state transitions

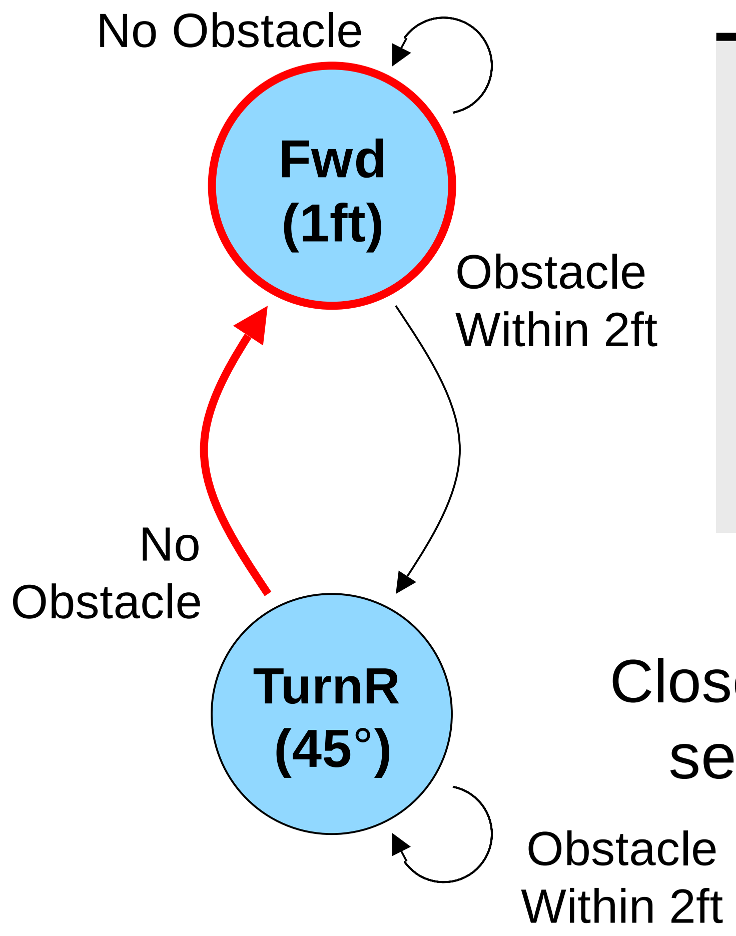
Finite State Machines offer another alternative for combining behaviors

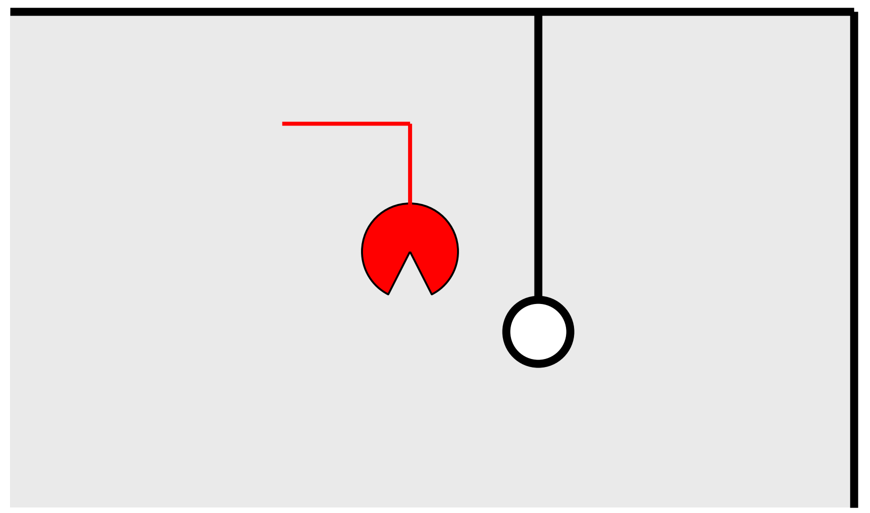




Closed loop finite state machines use sensor data as feedback to make state transitions

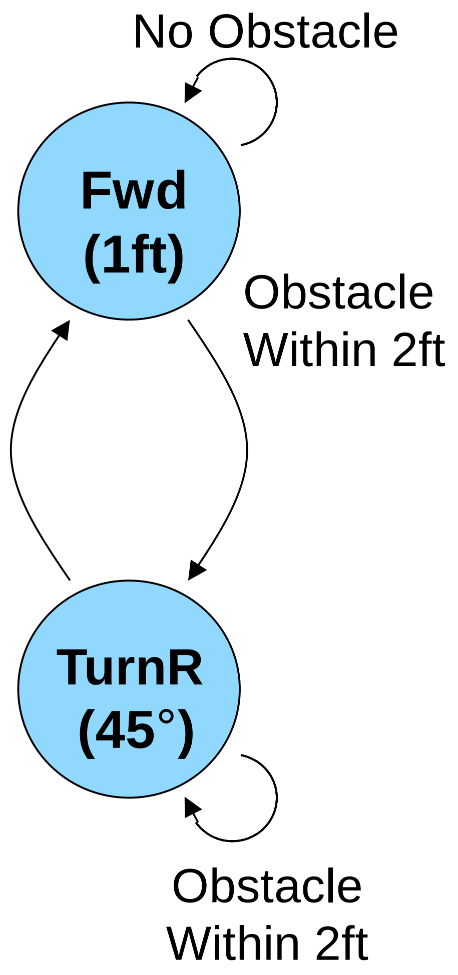
Finite State Machines offer another alternative for combining behaviors





Closed loop finite state machines use sensor data as feedback to make state transitions

Implementing a FSM in Java



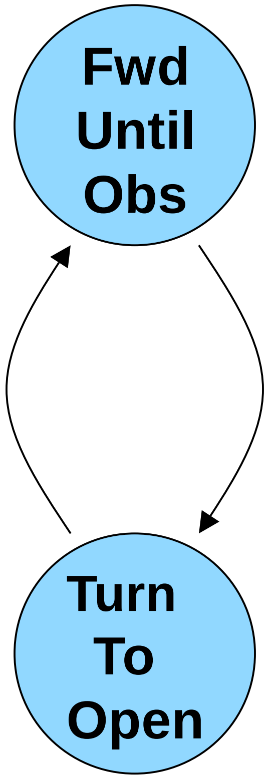
```
// State transitions
switch ( state ) {

  case States.Fwd_1 :
    if ( distanceToObstacle() < 2 )
      state = TurnR_45;
    break;

  case States.TurnR_45 :
    if ( distanceToObstacle() >= 2 )
      state = Fwd_1;
    break;

}

// State outputs
switch ( state ) {

  case States.Fwd_1 :
    moveFoward(1); break;

  case States.TurnR_45 :
    turnRight(45); break;

}
```

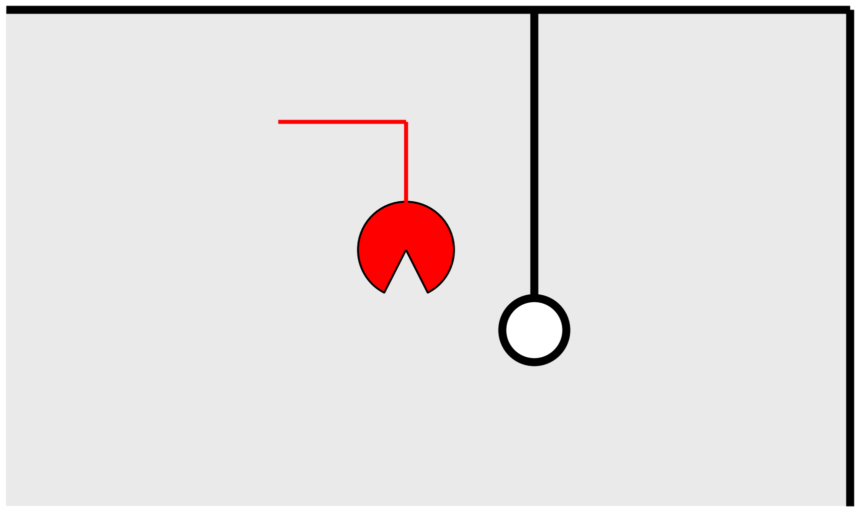
Implementing a FSM in Java

- Implement behaviors as parameterized functions
- First switch statement handles state transitions
- Second switch statement executes behaviors associated with each state
- Use enums for state variables

```
// State transitions
switch ( state ) {

  case States.Fwd_1 :
    if ( distanceToObstacle() < 2 )
      state = TurnR_45;
    break;

  case States.TurnR_45 :
    if ( distanceToObstacle() >= 2 )
      state = Fwd_1;
    break;

}

// State outputs
switch ( state ) {

  case States.Fwd_1 :
    moveFoward(1); break;

  case States.TurnR_45 :
    turnRight(45); break;

}
```

Finite State Machines offer another alternative for combining behaviors



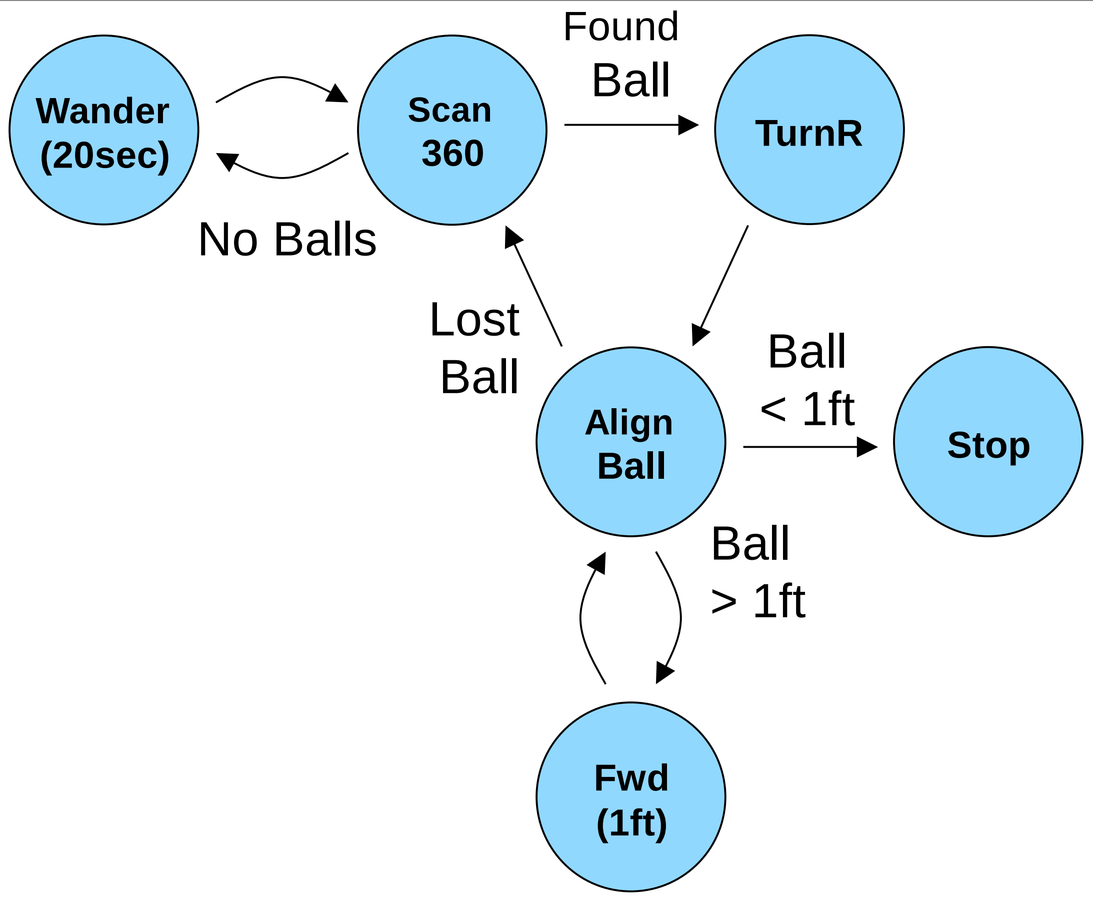


Can also fold closed loop feedback into the behaviors themselves

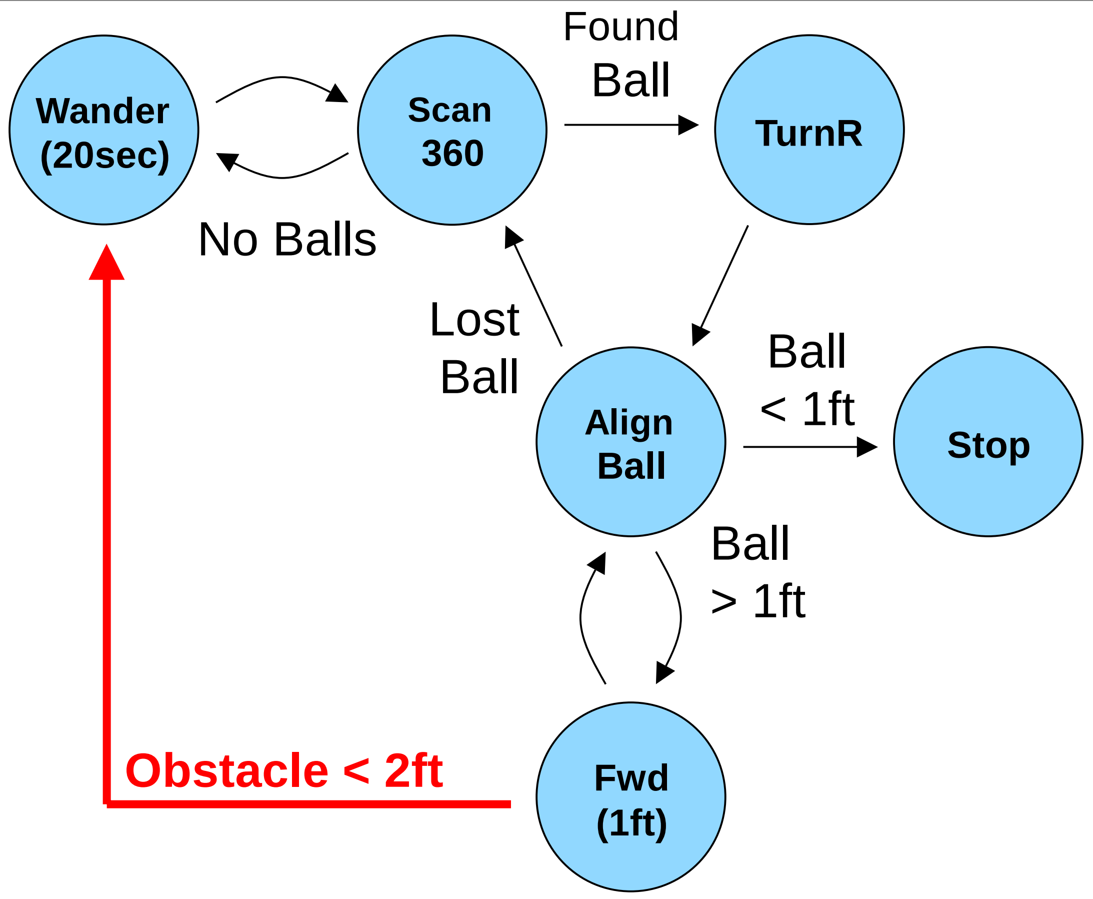
Case Study

SECTION 2

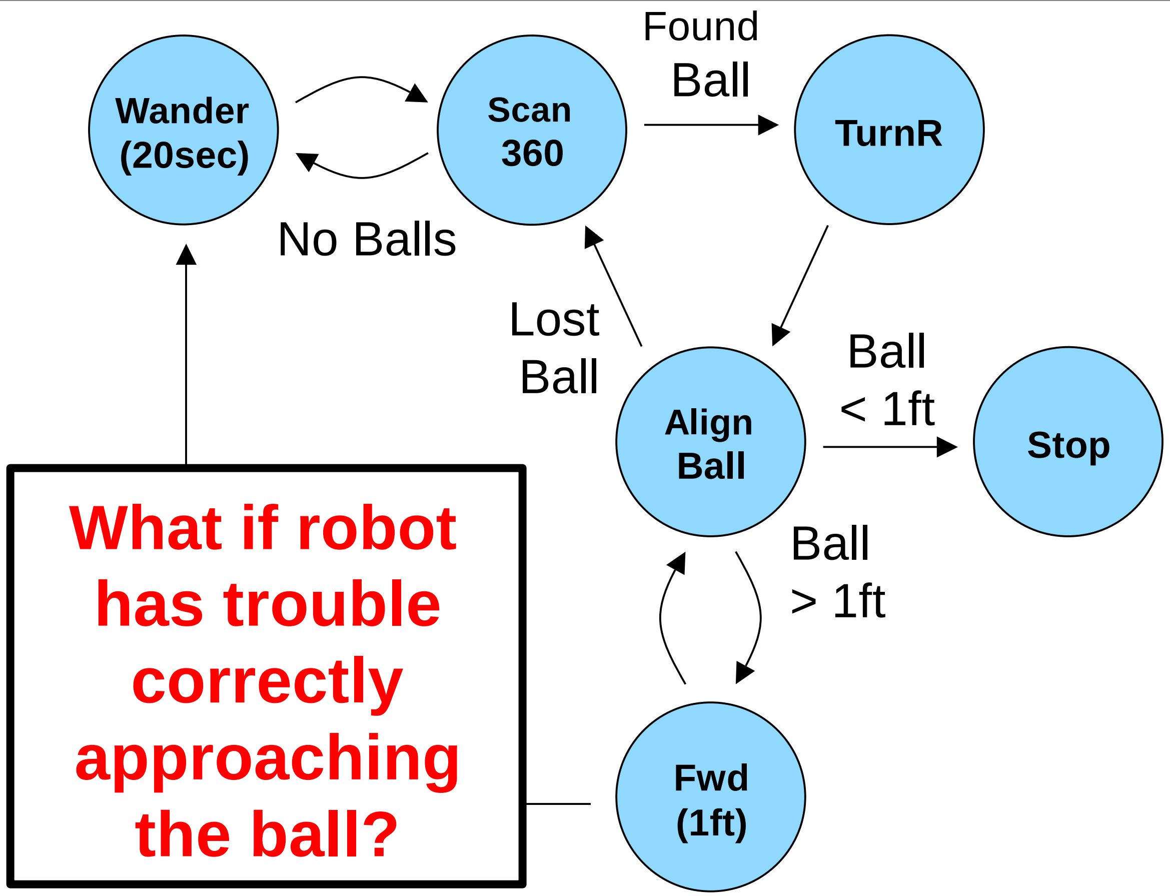
Simple finite state machine to locate red balls



Simple finite state machine to locate red balls

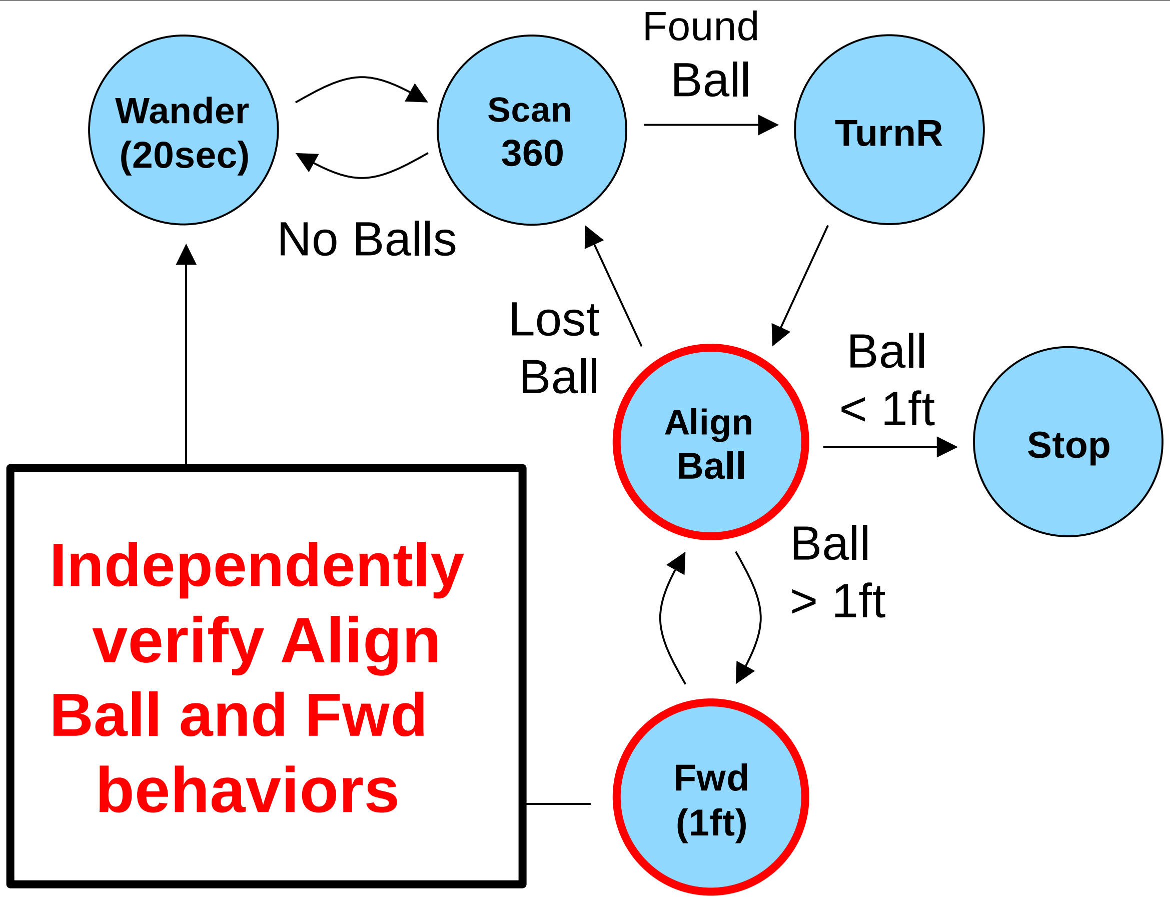


To debug a FSM control system verify behaviors and state transitions

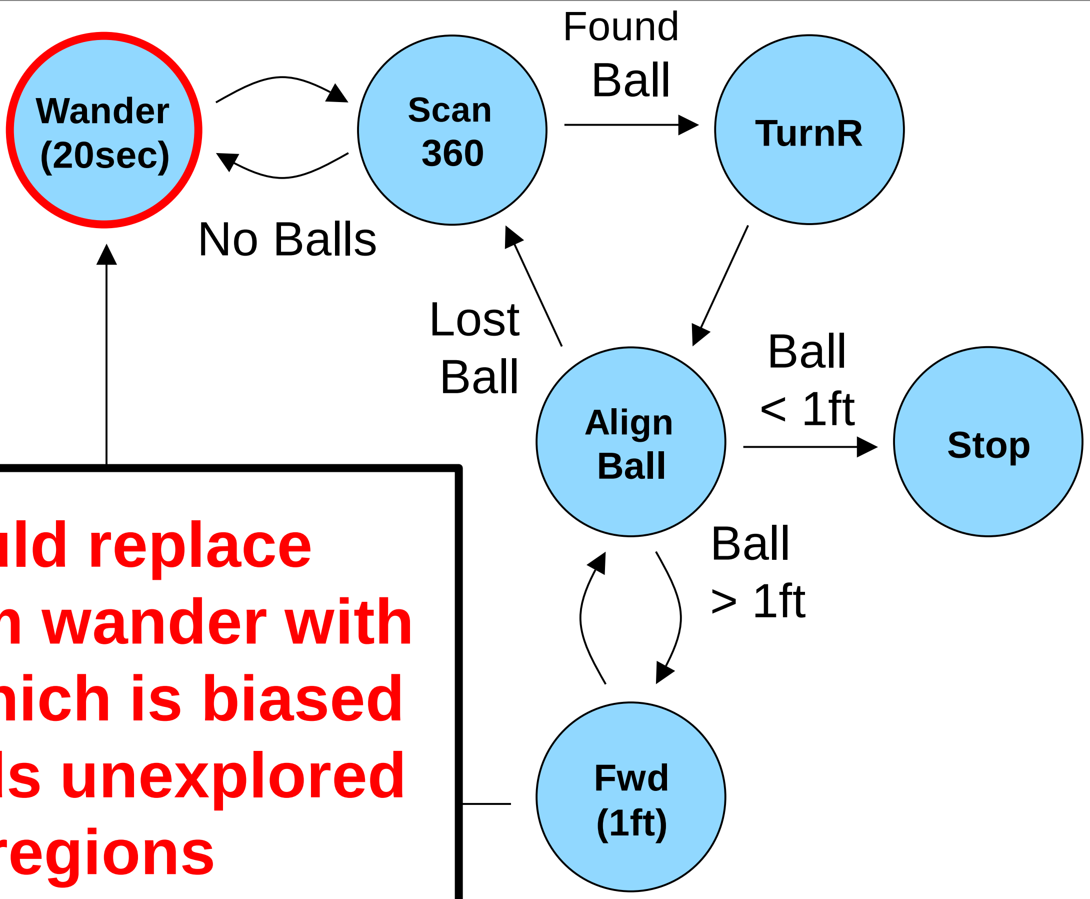


What if robot has trouble correctly approaching the ball?

To debug a FSM control system verify behaviors and state transitions



Improve FSM control system by replacing a state with a better implementation



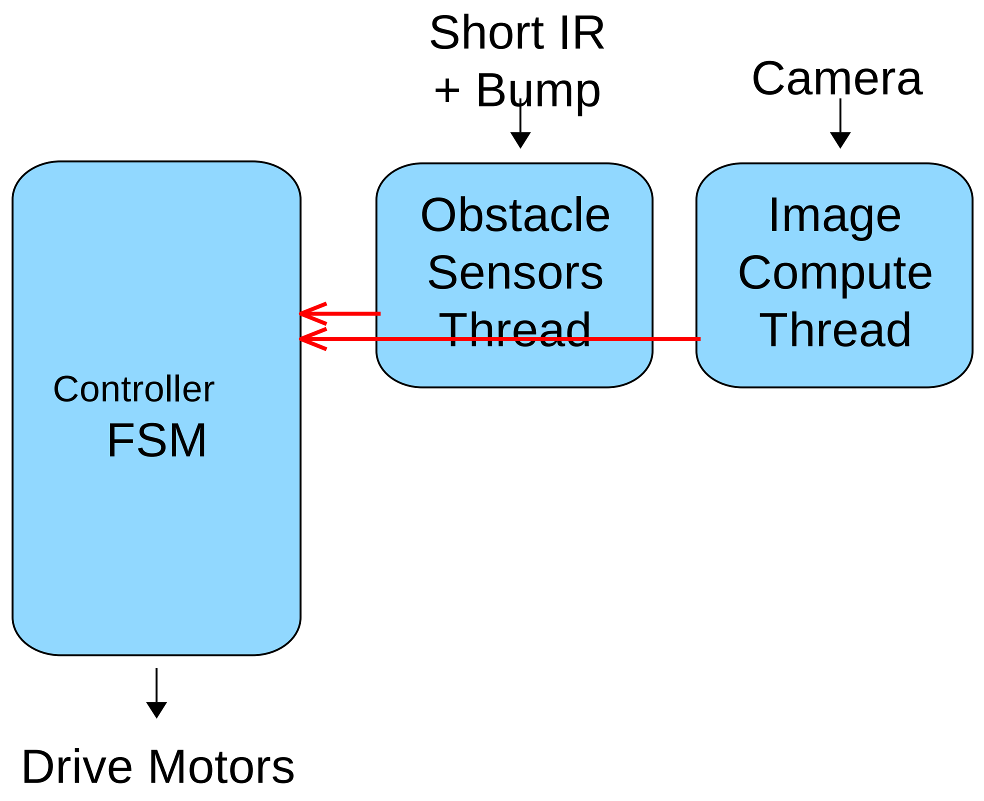
Could replace random wander with one which is biased towards unexplored regions

Improve FSM control system by replacing a state with a better implementation

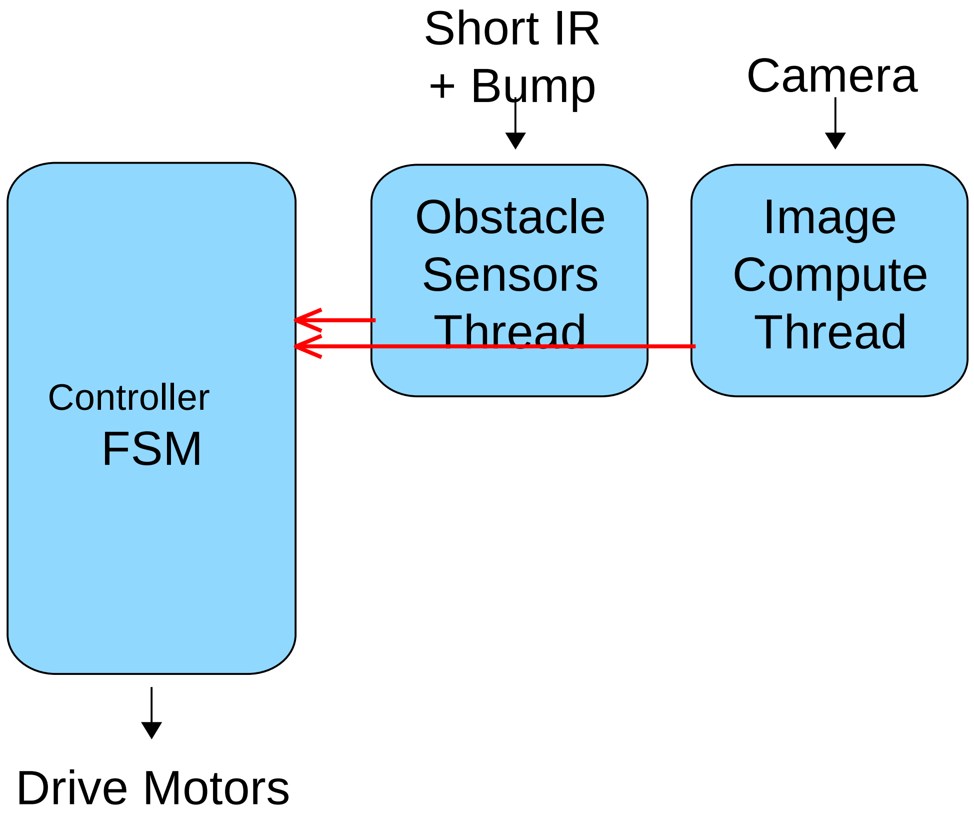
- What about integrating camera code into wander behavior so robot is always looking for red balls?
    - Image processing is time consuming so might not check for obstacles until too late
    - Not checking camera when rotating
    - Wander behavior begins to become monolithic

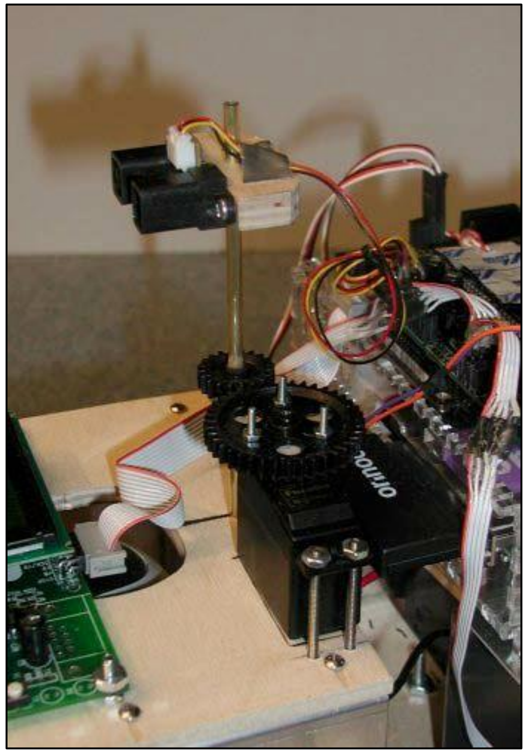
```
ball = false
turn both motors on
while ( !timeout and !ball )
  capture and process image
  if ( red ball ) ball = true

  read IR sensor
  if ( IR < thresh )
    stop motors
    rotate 90 degrees
    turn both motors on
  endif

endwhile
```

Multi-threaded finite state machine control systems

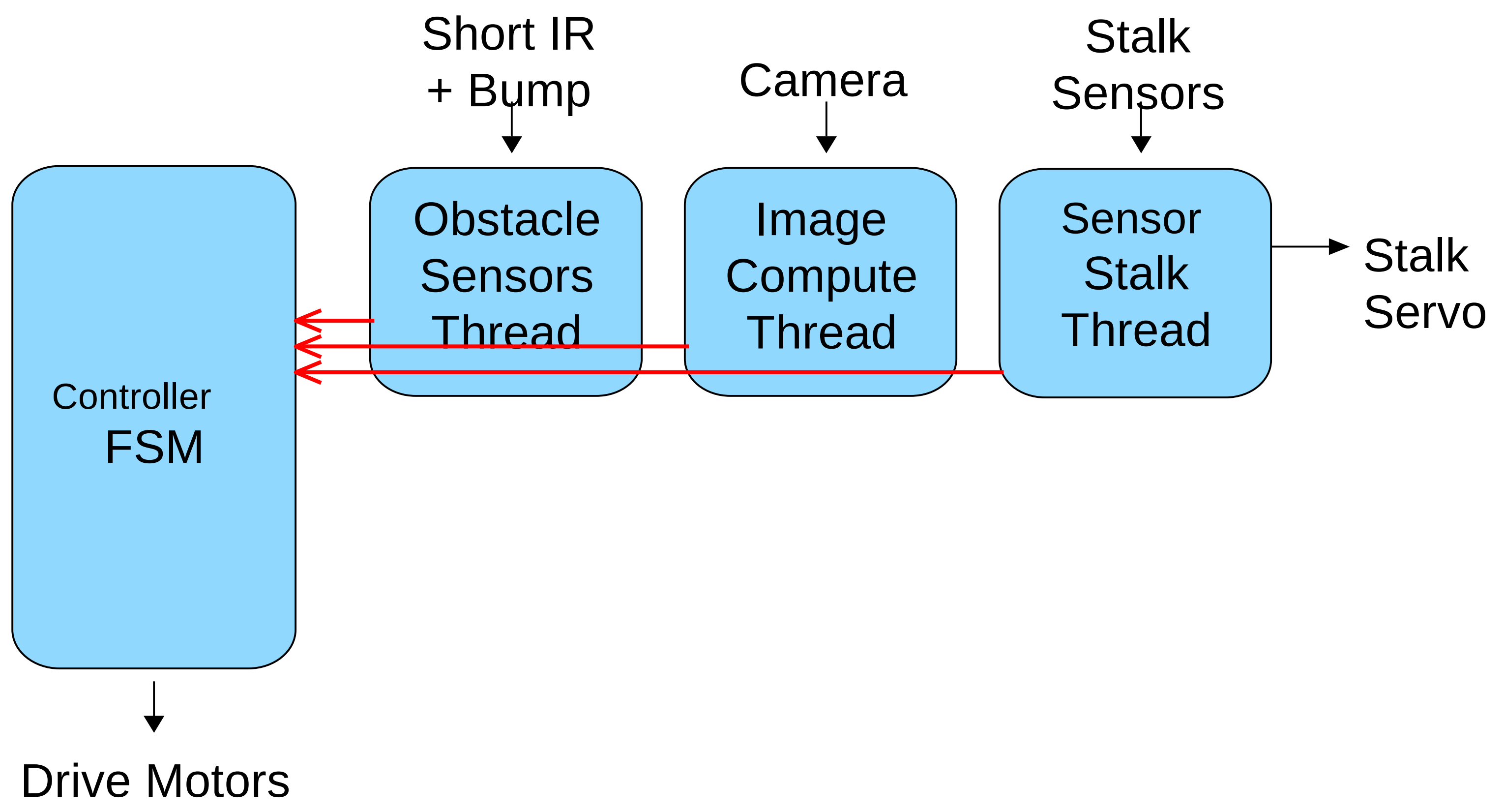


Multi-threaded finite state machine control systems

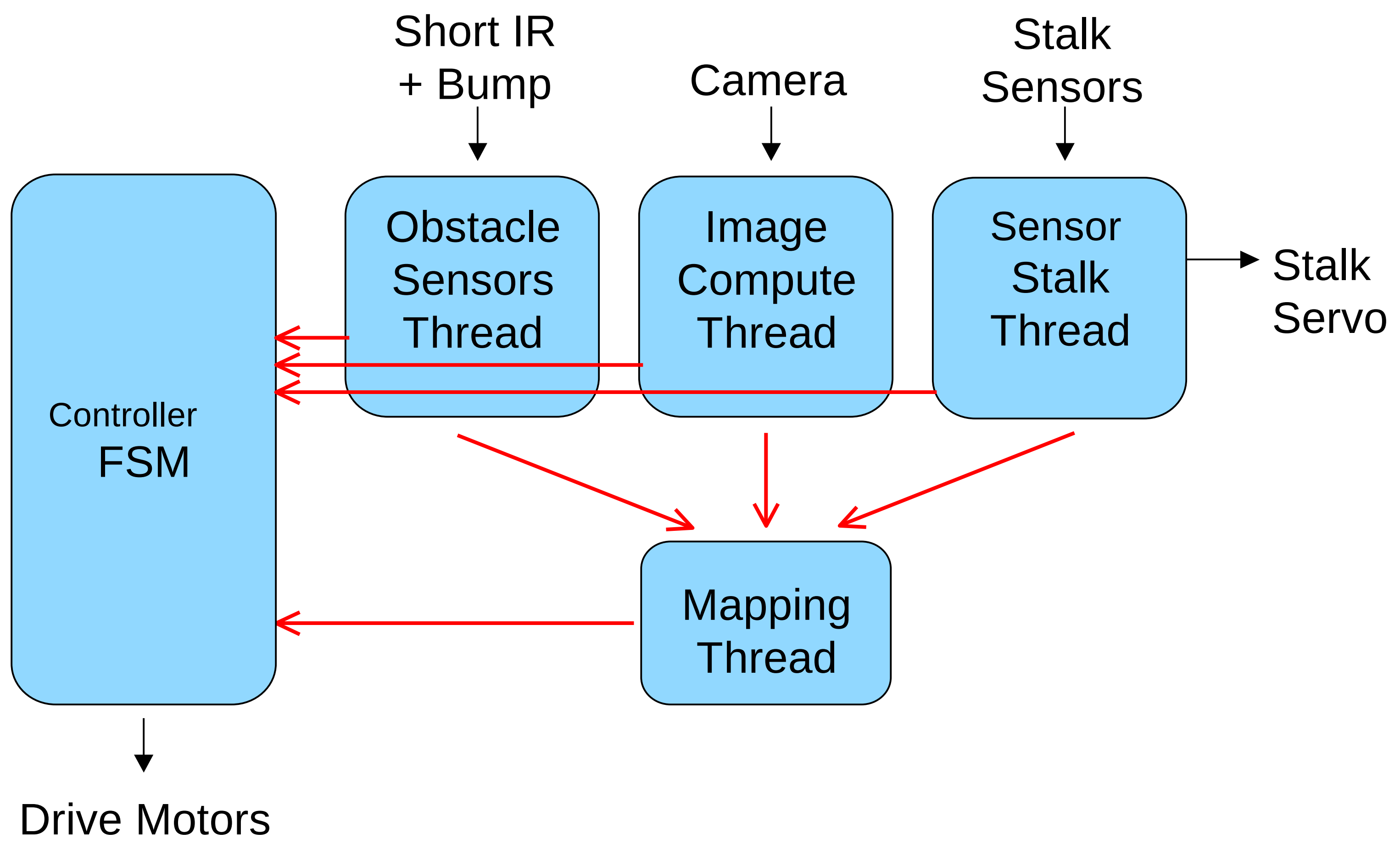




Multi-threaded finite state machine control systems



Multi-threaded finite state machine control systems



Python Coding

SECTION 3

FSM in Python

- Design of a Finite State Machine model in Python Language.
- Finite state machines can combine the model-plan-act and behavioral approaches and are a good starting point for your robotic control system modeling.

FPGA Implementation of PID Controller

SECTION 4

FPGA and PID Controller

- The **FPGAs** have been used then to make the fast samples of ASICs (Application Specific Integrated Circuits) and find since some years their place in many domains of applications. However, the order of the processes industrial requires more and more elements of powerful calculations. This type of order is in the same way in perpetual evolution with the development of the numeric circuits of calculation.
- Thus, the **PID controllers** represent the majority of the controllers used in the industrial systems control. Of this fact, it will be necessary to **digitalize** the PID algorithm. The modern digital control systems require more and more strong and fastest calculation components.

FPGA and PID Controller

- This type of elements becomes yet indispensable with the utilization of some new control algorithms like the fuzzy control, the adaptive control, the sliding mode control. Although the PID controllers are the oldest they represent the most used controllers in the industrial control systems

Discrete PID Equation

SECTION 5

Discrete PID Equation

- The PID algorithm consists of three basic modes, the Proportional mode, the Integral and the Derivative modes. When utilizing this algorithm, it is necessary to decide which modes are to be used (P, I or D) and then specify the parameters (or settings) for each mode used. Generally, three basic algorithms are used P, PI or PID.
- The implementation of PID controllers using microprocessors and DSP chips is old and well known, whereas very little works can be found in the literature on how to implement PID controllers using FPGAs.

Discrete PID Equation

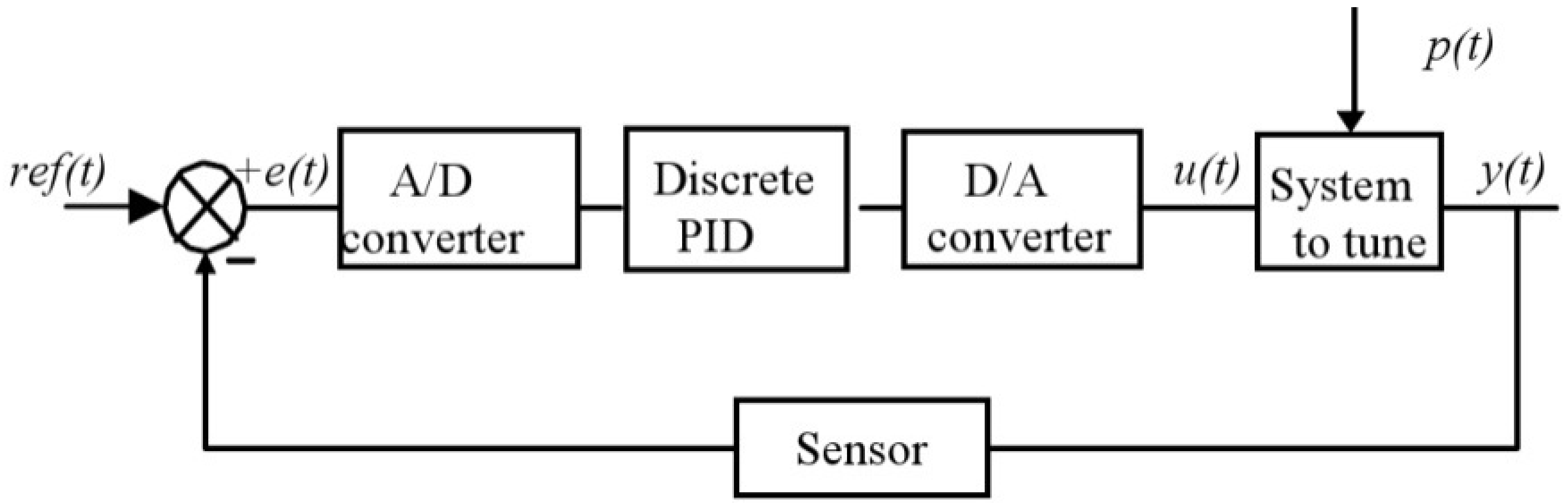
- Field Programmable Gate Arrays (FPGA) have become an alternative solution for the realization of digital control systems, previously dominated by the general-purpose microprocessor systems.

Discrete PID Equation

- A simple method for implementing PID controllers is introduced. Some other contributions focused on proposing algorithms for tuning the coefficients of PID controllers using FPGAs while the controller itself is still implemented in software.
- In this example project, data acquisition for the PID controller, which is implemented using Xilinx Spartan-3 Starter Kit Board, is based on 8-bitserial A/D converters extensible from Digilent board AO1. Similar converters are utilized to implement an adaptable strain gage conditioner using FPGAs.

Discrete PID Equation

- The application of a PID controller in a feedback control system is shown in fig1, where ref is the set point signal, y is the feedback signal, e is the error signal, and u is the control input.



A PID-based feedback control system.

Discrete PID Equation

- The simplest form of the PID control algorithm is given by:

$$u(t) = k_p(e(t) + \frac{1}{T_i}\int e(t)dt + T_d \frac{de(t)}{dt}) \qquad \text{E1}$$

- According to the study done in the digitized PID equation is brought back to:

$$u_k = u_{k-1} + b_0\ e_k + b_1\ e(k-1) + b_2\ e(k-2) \qquad \text{E2}$$

Discrete PID Equation

- Where the coefficients b_0, b_1, and b_2 are evaluated by the expressions:

$$b_0 = k_p \left(1 + \frac{T_d}{T}\right);$$

$$b_1 = k_p \left(-1 + \frac{T}{T_i} - 2\,\frac{T_d}{T}\right);$$

$$b_2 = k_p \,\frac{T_d}{T}; \qquad \text{E3}$$

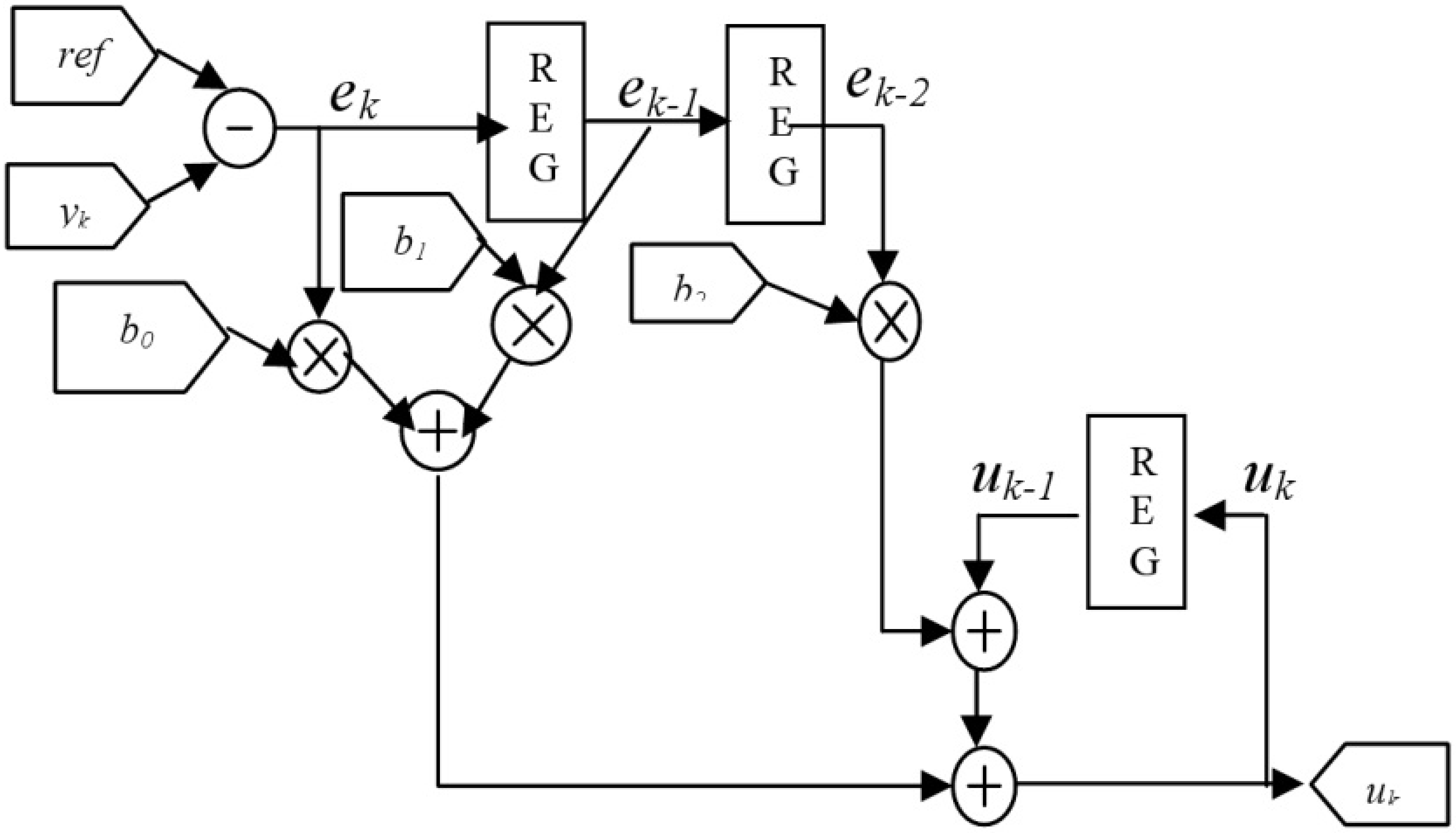
- The K_p, T_i and T_d, are PID parameters for tuning, and T is the sampling period in seconds.

Digital PID Architecture

SECTION 6

Digital PID Architecture

- To improve the speed and minimize the cost while offering clearly good performances, the adopted architecture used includes essentially three combinational logic multiplier, one subtractor three adders and three registers. The fig 2 gives the adopted architecture.
- Indeed, this architecture requires the availability of all calculation operators in each phase.



PID Architecture.

Conversion Blocks Presentation

- The AIO1 board is a peripheral board designed to work with Digilent’s family of system boards. The AIO1 contains analog to-digital and digital-to-analog converters from Analog Devices, two dual op amps, a variety of analog signal I /O connectors, and a solderless breadboard. All analog components use an on-board 5VDC voltage source. All unused I/O signals are passed through the AIO1 board so that it can be used between a system board and other peripheral boards.
- The AIO1 uses an 8-bit, 200Ks analog-to-digital converter (the AD7823), and an 8-bit, 1 MHz digital-to-analog converter (the AD7303), both from Analog Devices. The AD8534 op amps (also from Analog Devices) can drive 250mA outputs rail-to-rail with a 3 MHz bandwidth, so many useful devices can be driven directly.

Analog input interface

- FPGAs are well suited for serial Analog to Digital (A/D) converters. This is mainly because serial interface consumes less communication lines while the FPGA is fast enough to accommodate the high-speed serial data. The AD7823 is a high speed, low power, 8-bit A/D converter.
- The part contains a 4 µs typical successive approximation A/D converter and a high-speed serial interface that interfaces easily to FPGAs. The A/D interface adapter (ADIA) is implemented within the FPGA (Figure 5).

Analog input interface

- Inside the FPGA, this adapter facilitates parallel data acquisition. Sampling is initiated at the rising edge of a clock applied at the line sample. The timing diagram of the communication protocol is illustrated in figure 4. The whole conversion and acquisition period is 5.4 µs allowing sampling up to a rate of 185 Kilo Sample per second.
- This rate is more than sufficient for most PID control applications.

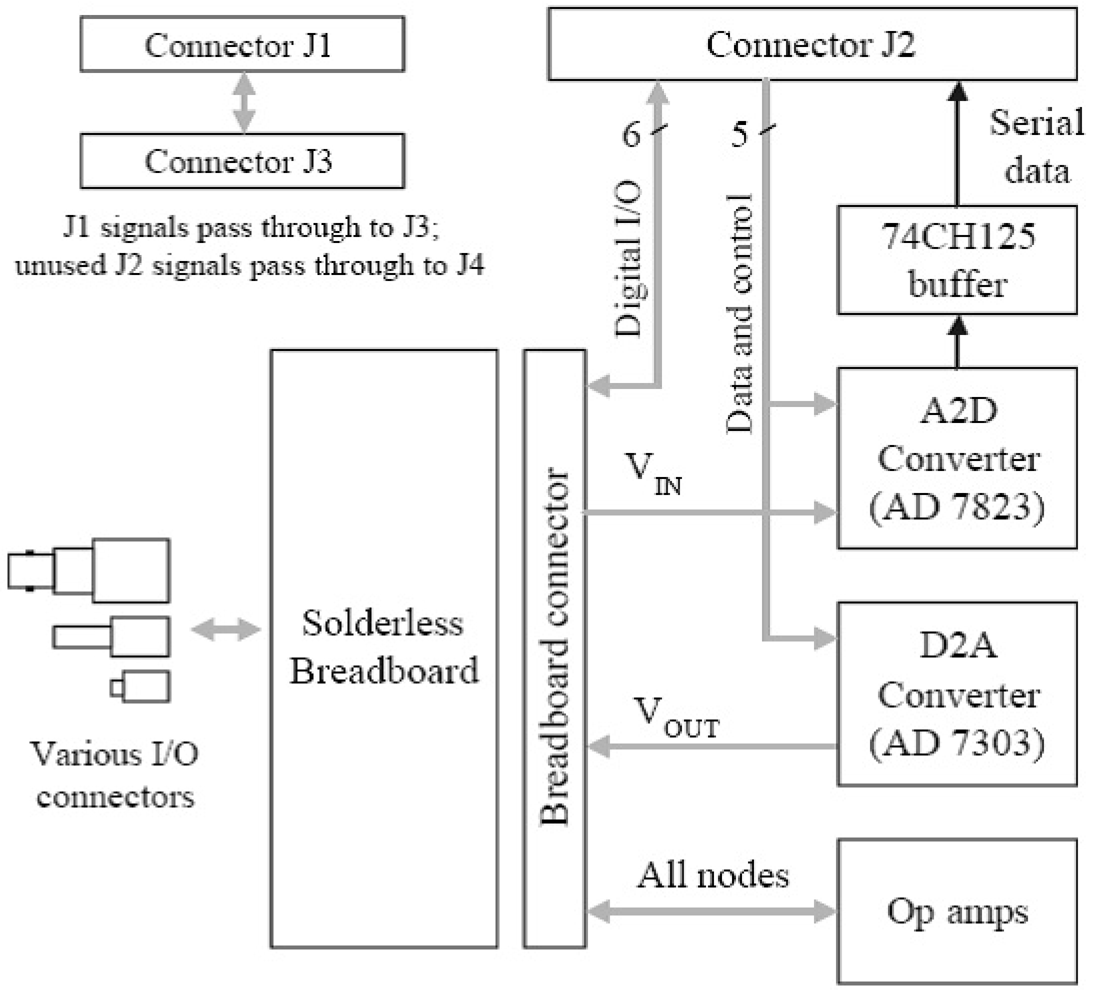
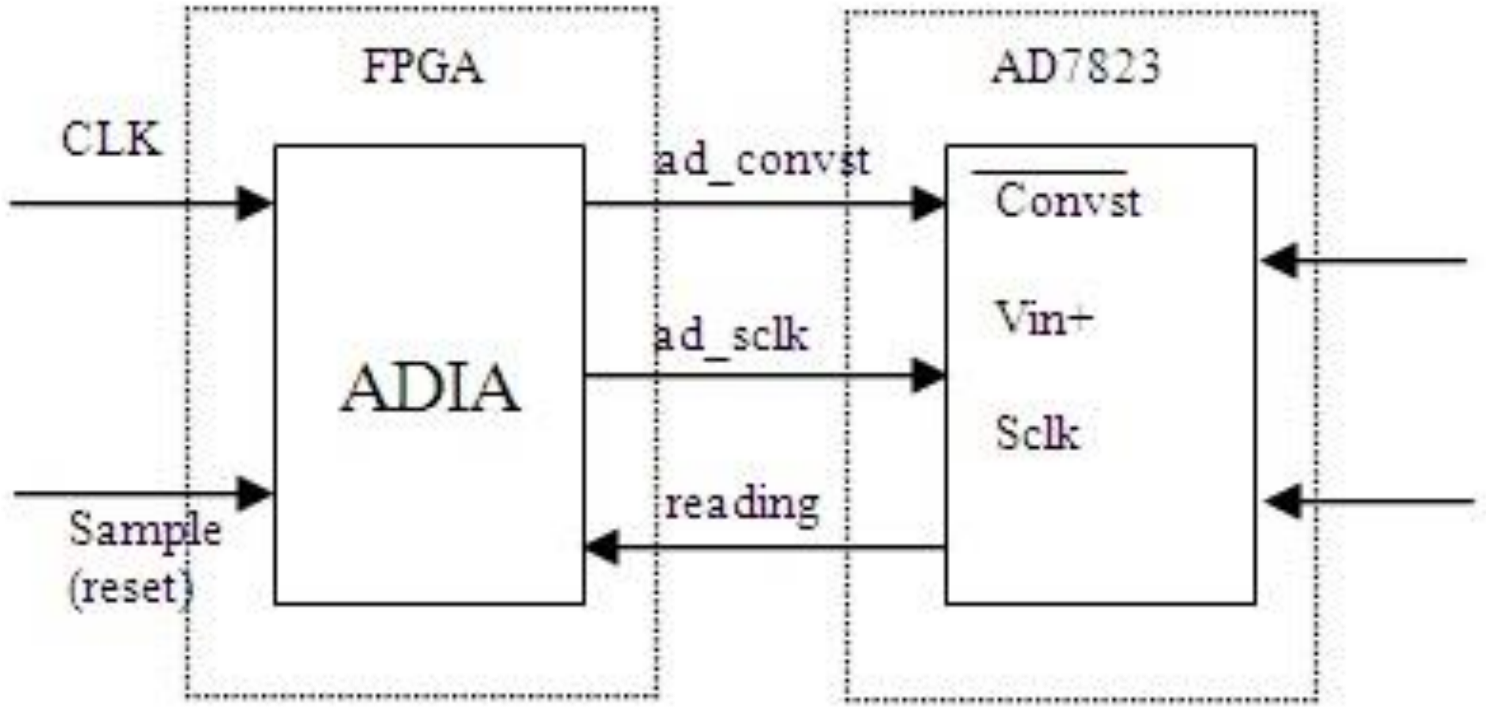
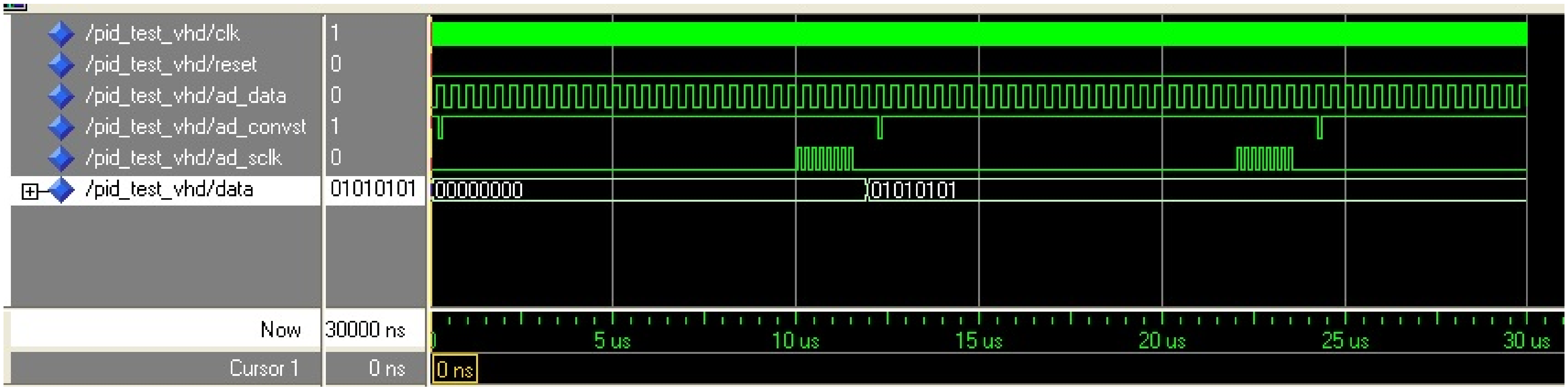


Diagram of DIGILENT AIO1



A/D interface converter.



/pid_test_vhd/clk	1	
/pid_test_vhd/reset	0	
/pid_test_vhd/ad_data	0	
/pid_test_vhd/ad_convst	1	
/pid_test_vhd/ad_sclk	0	
/pid_test_vhd/data	01010101	00000000, 01010101
Now	30000 ns	0, 5 us, 10 us, 15 us, 20 us, 25 us, 30 us
Cursor 1	0 ns	0 ns

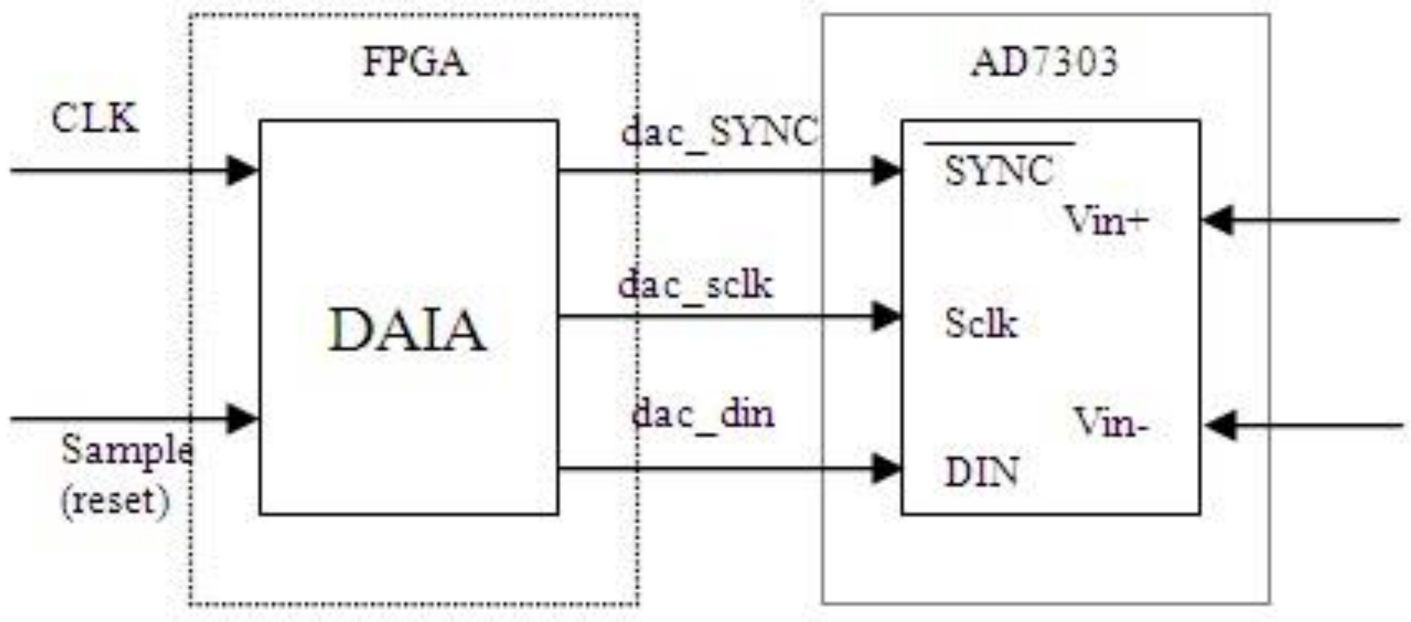
AD7823 Timing Diagram.

Analog input interface

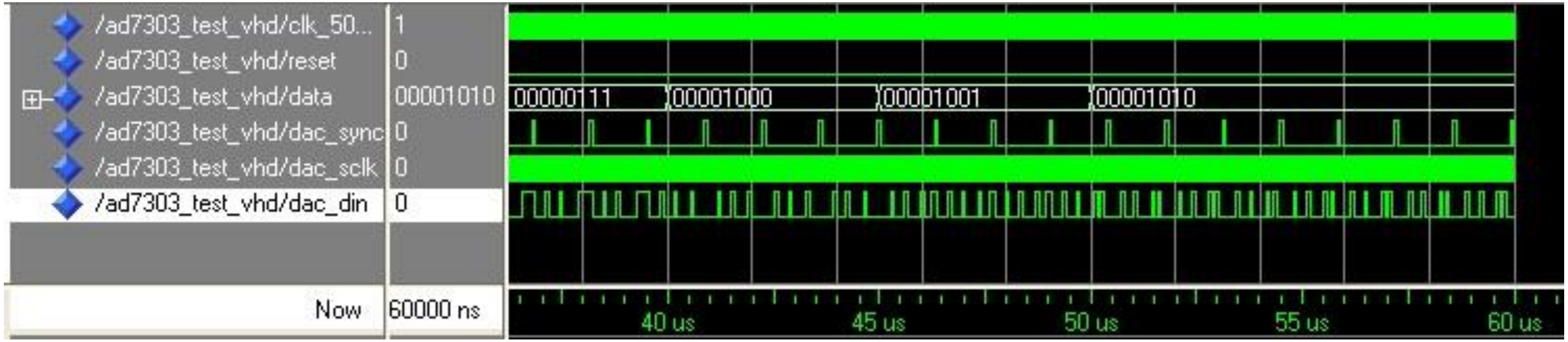
- The output coding of the AD7823 is straight binary. The designed code transitions occur at successive integer LSB values (i.e., 1 LSB, 2 LSBs, etc.). The LSB size is = VREF/256.

Analog output interface

- The AD7303 is a dual, 8-bit voltage out Digital to Analog (D/A) converter. This device uses a versatile 3-wire serial interface that operates at a clock up to 30 MHz. The serial input register is 16 bits wide; 8 bits act as data bits for the D/A converter, and the remaining 8 bits make up a control register. It is interfaced to an FPGA as illustrated in Figure 6. The D/A interface adapter (DAIA), which is implemented within the FPGA, facilitates parallel data input for the dual D/A converters.
- The timing diagram of the communication protocol is illustrated in figure 7. The transmission period of a sample is 680 ns allowing D/A conversion at an excellent rate of 1.47 MHZ.



A/D interface converter.



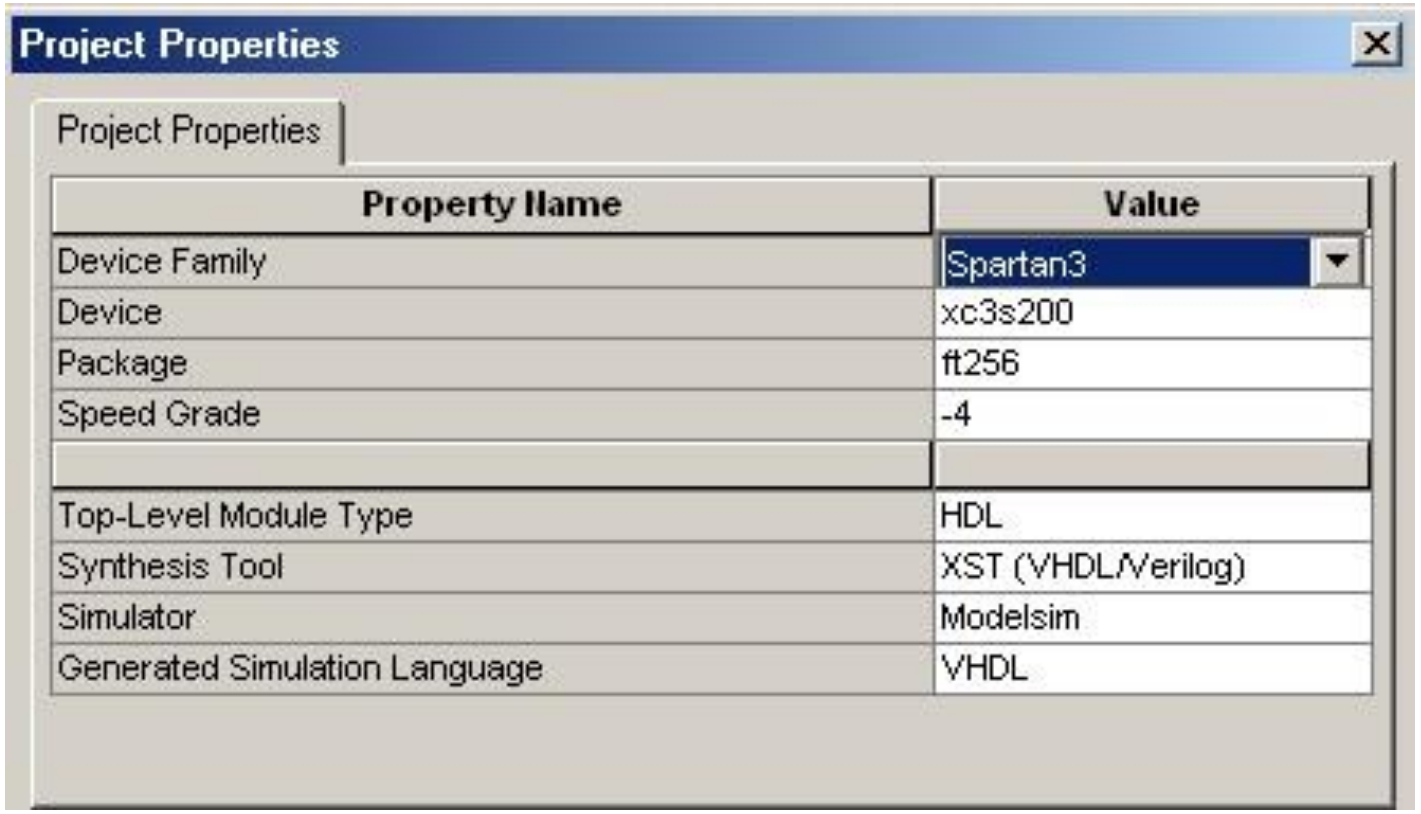
AD7303 Timing Diagram.

Analog output interface

- Any DAC output voltage can ideally be expressed as:
- VOUT = 2 × VREF × (N/256) where: N is the decimal equivalent of the binary input code. An N range from 0 to 255xVREF is the voltage applied to the external REF pin when the external reference is selected and is VDD/2 if the internal reference is used.

Implementation results

The proposed based PID controller is implemented using the Xilinx Inc FPGA technology and can be used as a general purpose controller for different applications. The simulation results obtained with the generated VHDL, in this work, the ModelSim® simulator was used. The circuits for the PID controllers have been obtained by logic synthesis and place and route using Xilinx ISE 7.1i, from the VHDL representation generated by the static analyzer. We use a Xilinx Spartan-3 xc3s200-ft256 -4 FPGA. The results presented herein are estimations directly obtained from Xilinx ISE 7.1i.



Project Properties

Project Properties

Property Name	Value
Device Family	Spartan3
Device	xc3s200
Package	ft256
Speed Grade	-4
Top-Level Module Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	Modelsim
Generated Simulation Language	VHDL

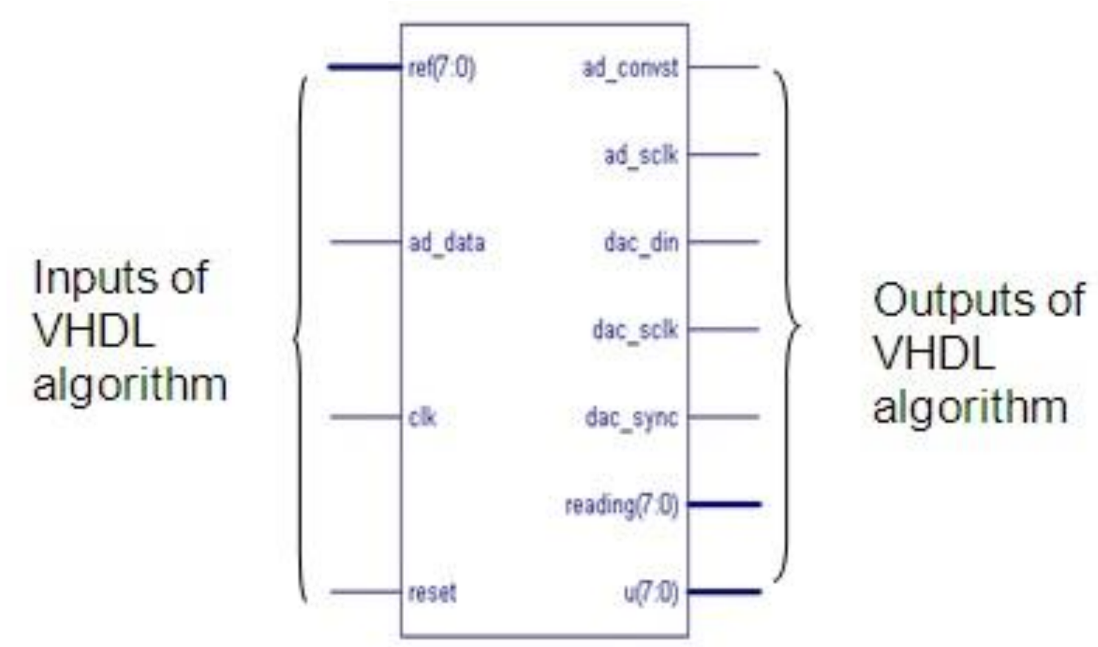
Design properties.

Table 1 shows the minimum number of multiplications, additions and registers required for the PID controller without conversions block.

Multiplication	3
Addition	3
subtraction	1
register	3
total	10

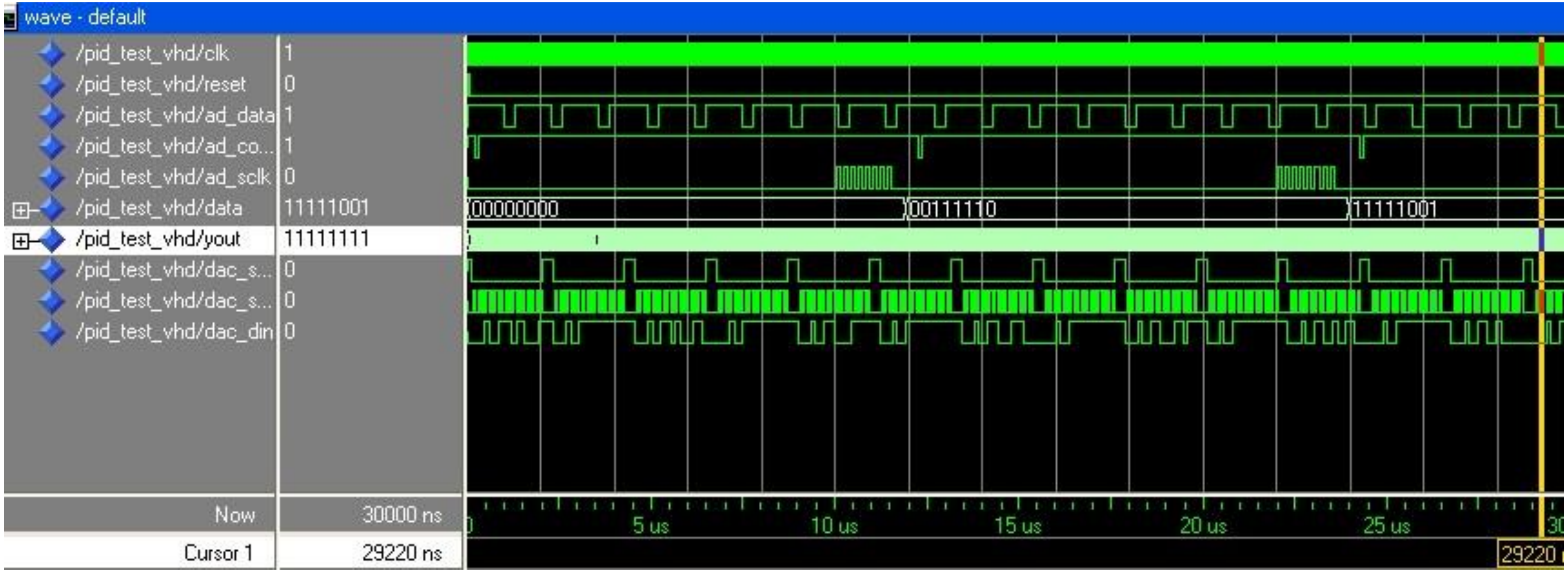
Arithmetic Number for PID controller.

The PID controller block, into a complete control system consisting of analog and digital I/O, is illustrated in figure 9.



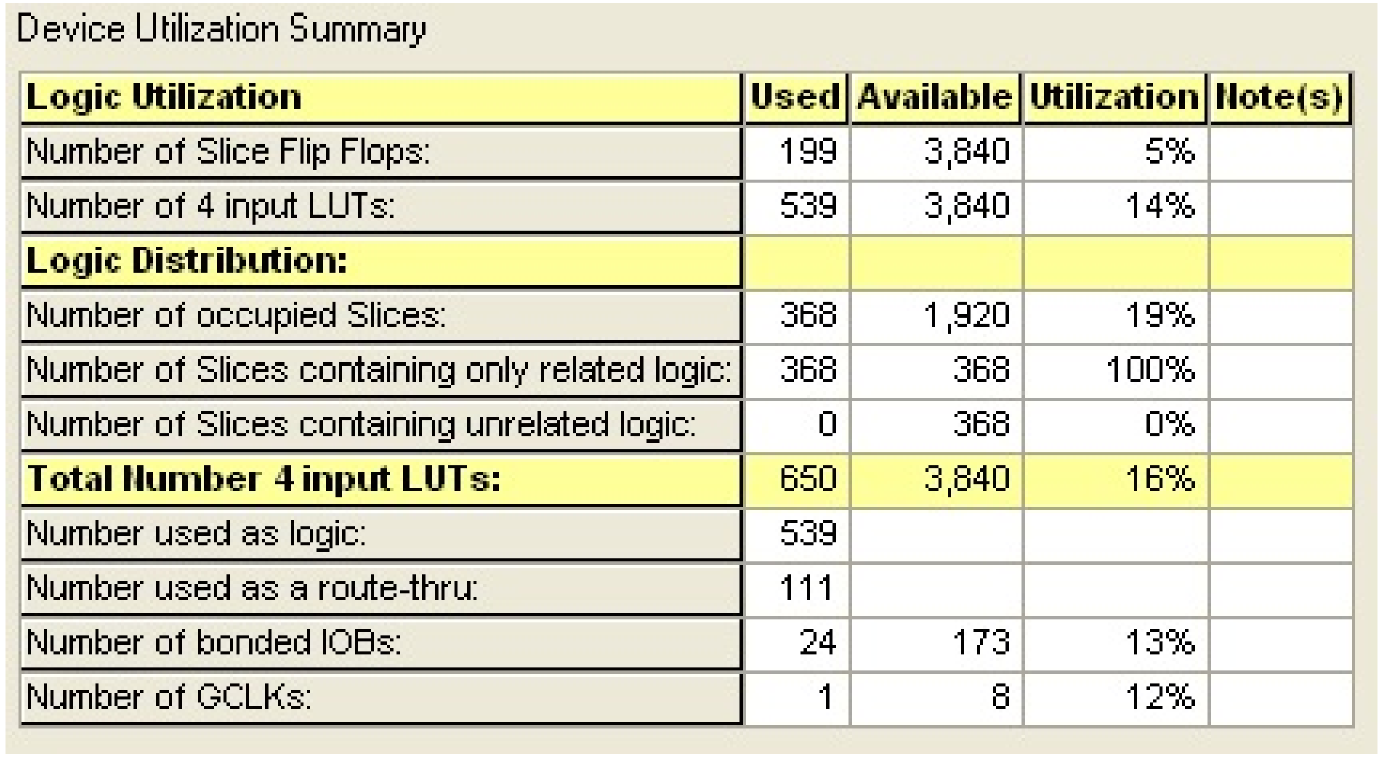
PID Controller Block.

The simulation results adapted to this block is shown in the next figure.



Simulation diagram of PID controller block

The synthesis of PID controller block using a Xilinx Spartan-3 xc3s200-ft256 -4 FPGA gives the following results.



Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops:	199	3,840	5%	
Number of 4 input LUTs:	539	3,840	14%	
Logic Distribution:				
Number of occupied Slices:	368	1,920	19%	
Number of Slices containing only related logic:	368	368	100%	
Number of Slices containing unrelated logic:	0	368	0%	
Total Number 4 input LUTs:	650	3,840	16%	
Number used as logic:	539			
Number used as a route-thru:	111			
Number of bonded IOBs:	24	173	13%	
Number of GCLKs:	1	8	12%	

Devices utilizations summary for the PID controller Implementation.

Implementation results

- A design which is efficient in terms of power consumption and chip area means that the FPGA chip can be used to accommodate more controllers with adequate speed and low power consumption, resulting in a cost reduction of the controller hardware.

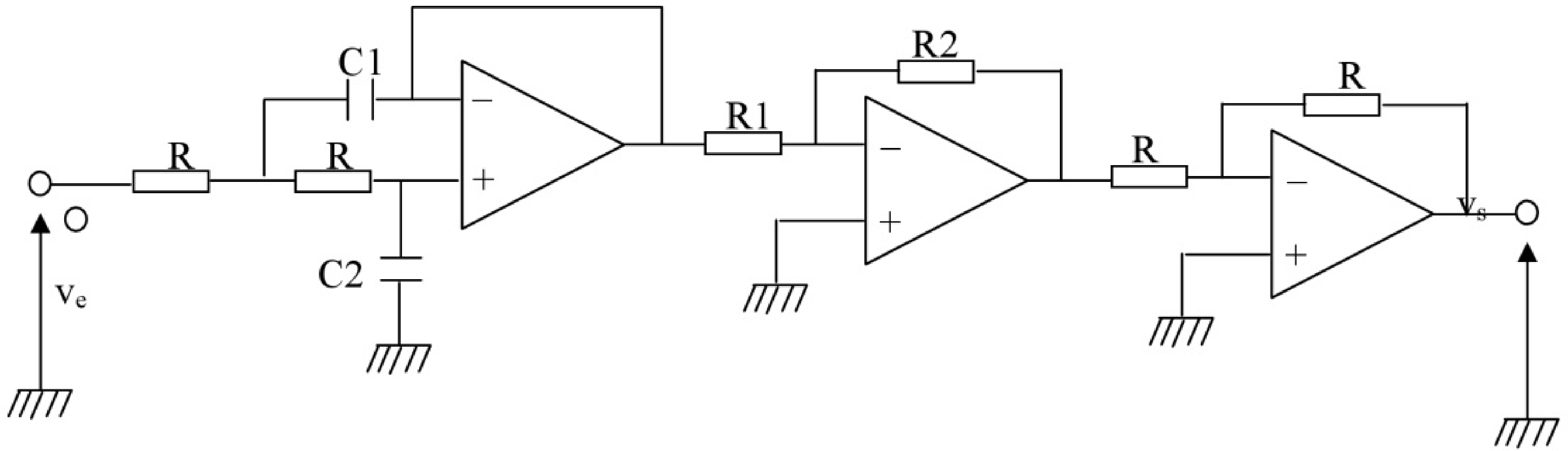
Case Study: A System of Second Order

SECTION 7

System Presentation

- The values of all the components are:

R = 12 k
R1 = 15 kΩ
R2 = 10 kΩ
C1 = 6.8 nF
C2 = 22 nF



Installation Diagram of the system of second order.

System Presentation

The Transfer function of the system is given by the following equation

$$G(p) = \frac{V_s(p)}{V_e(p)} = \frac{K_s}{1 + \frac{2m}{\omega_0} p + (\frac{1}{\omega_0})^2 p^2}$$

With:

$$K_s = \frac{R_2}{R_2} = 0.67\text{: static gain}$$

$$\omega_0 = \frac{1}{R\sqrt{C_1 C_2}} = 6.81 * 10^3 \text{ rad/s} \quad \text{E5}$$

$$m = \sqrt{\frac{C_2}{C_1}} = 1.80 > 1\text{: Amortization Factor.}$$

While permuting the positions of C_1 and C_2, the amortization factor becomes:

Experimental result

- At the time of the order of the system two cases appear; when the amortization is $m>1$ and $m<1$.

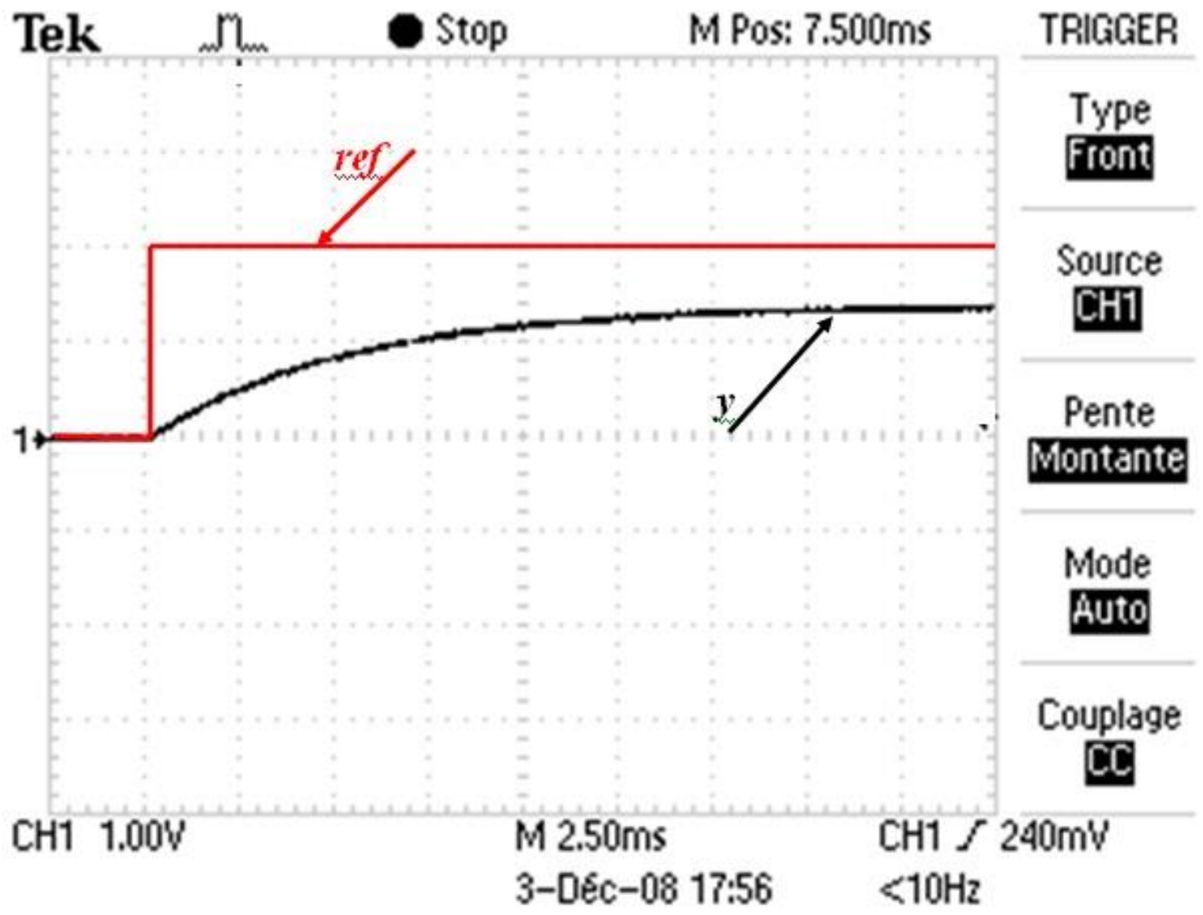
$$m = \sqrt{\frac{C_1}{C_2}} = 0.56 \qquad \text{E6}$$

Experimental results for an amortization $m>1$

CASE 1

System Answer Results in Open Buckle

- For an order (ref) of the order of 2V applied to the system in BO one gets the answer y (t) presented on the following figure.



System Answer in open buckle m>1.

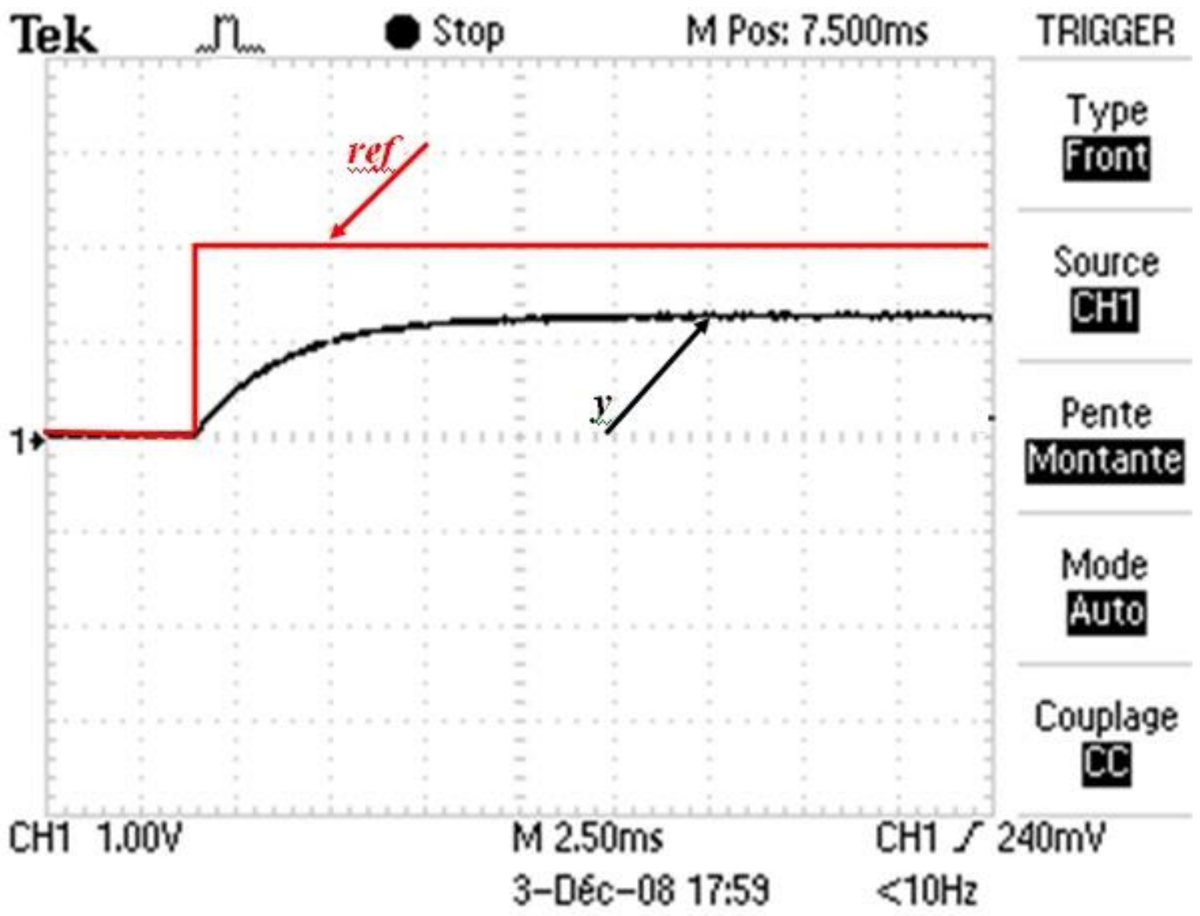
System answer results in open buckle

- According to this answer, one verifies that the tension of exit stabilizes without oscillations (m>1) nearly to the value 1,4V, that corresponds more or less to the theoretical value:

$$V_s = K_s V_e = 0.67 * 2 = 1.34\, V \qquad \text{E7}$$

System answer results with P regulator

- For an order (ref) of the order of 2V applied to the system ordered by a Proportional regulator (P) with KP = 2, one gets the answer y(t) presented on the following Figure



Answer of the system ordered by proportional regulator P.

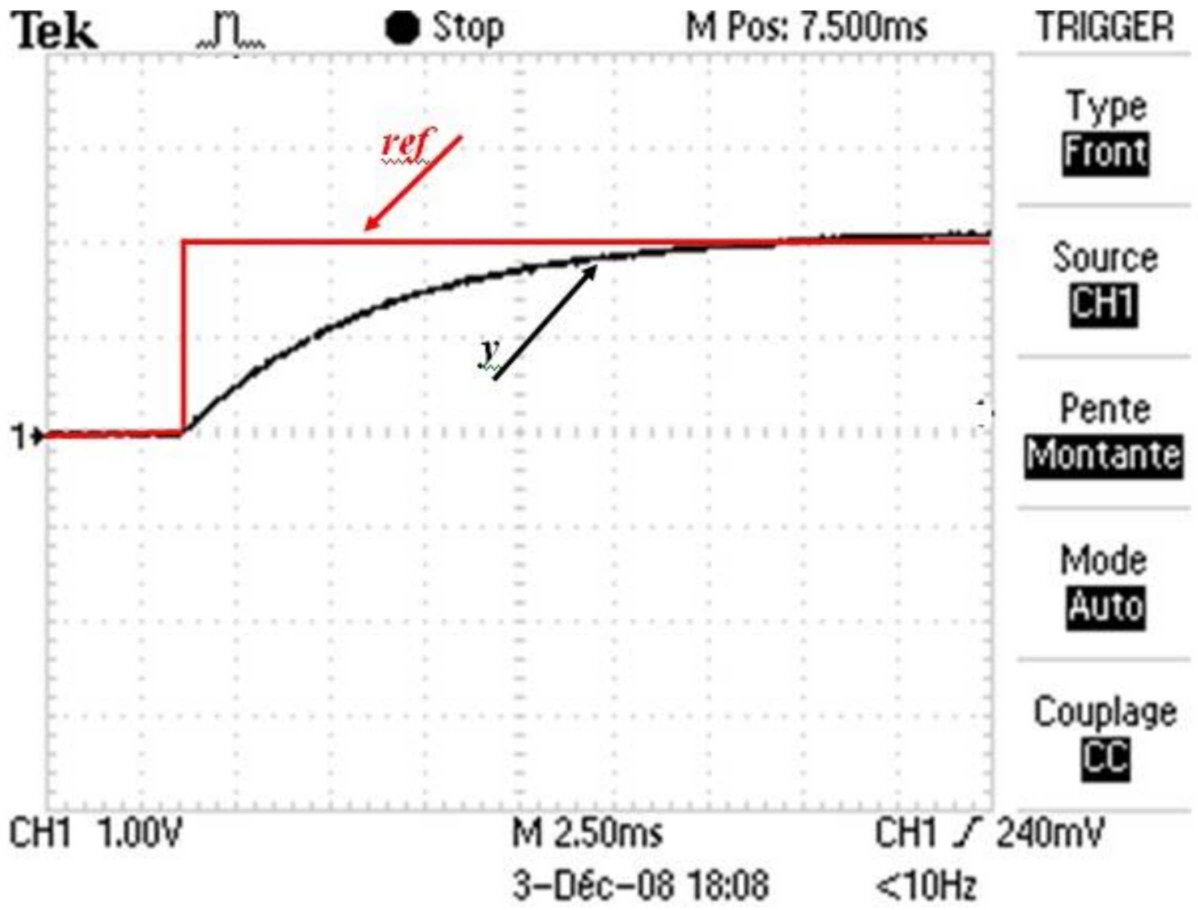
System answer results with P regulator

- One notices that this answer presents a static mistake of the order of 40%. Theoretically this mistake is given by:

$$\frac{1}{1+K_p\,K_s}\;100\% = 42\ \% \qquad \text{E8}$$

System answer results with PI regulator

- For an order (ref) applied of the order of 2V to the system ordered by a PI regulator with KP=2, KI=0.5, one gets the answer y(t) presented in the following figure:



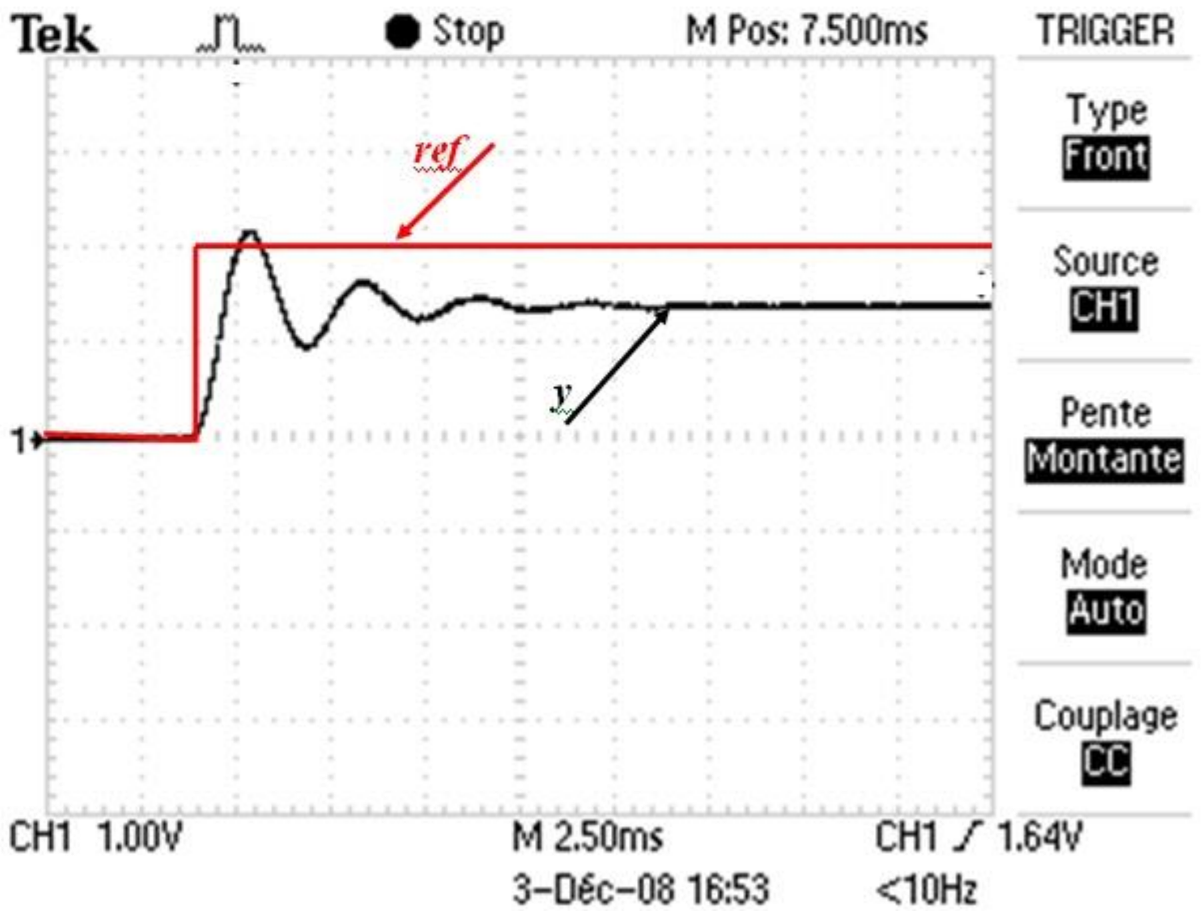
Answer of the system ordered by PI regulator (m>1).One notices the annulment of the static mistake well thanks to the introduction of the I action.

Experimental results for an amortization m<1

CASE 2

System answer results in open buckle

- For an order (ref) of the order of 2V applied to the system in BO one gets the answer *y(t)* presented on the following figure



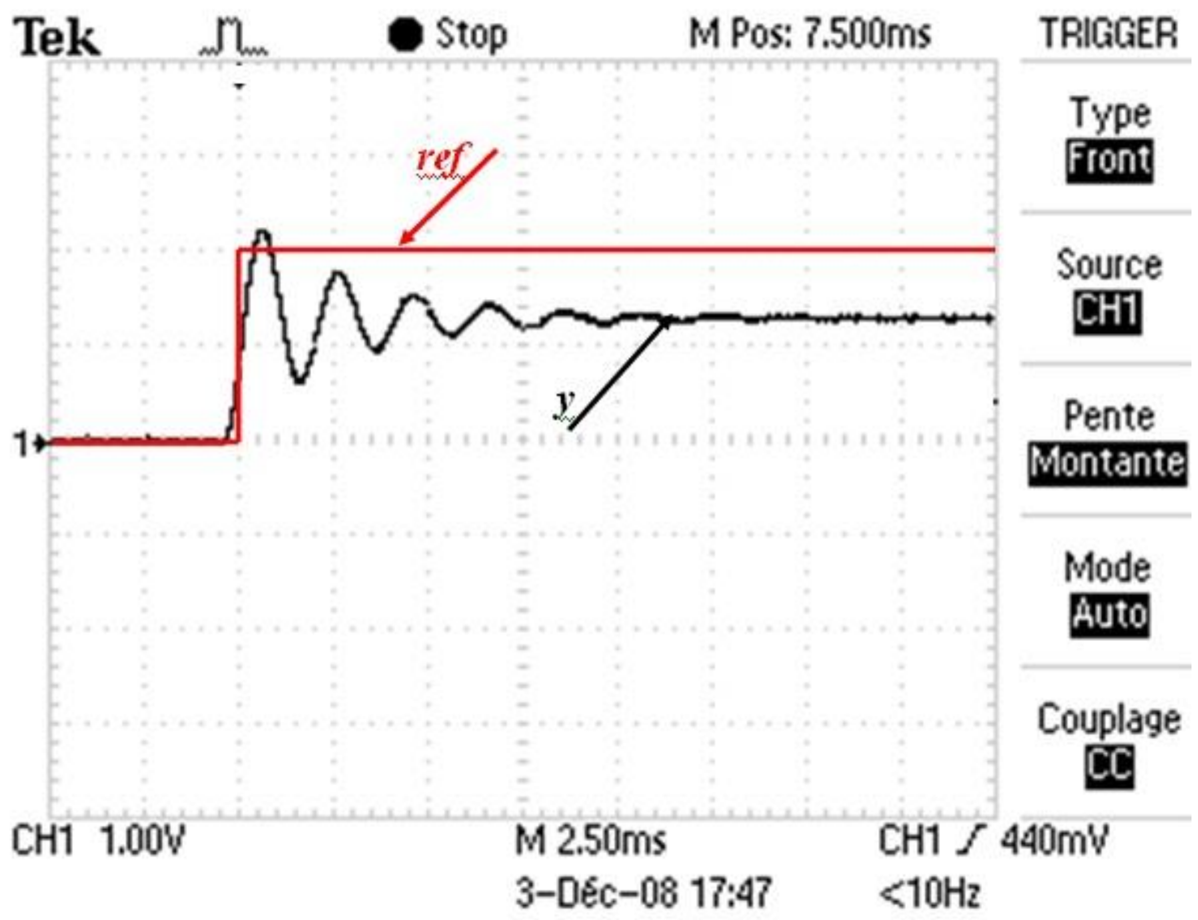
Answer of the system in open buckle(m<1).

System answer results in open buckle

- According to this answer, one verifies that the tension of exit stabilizes with oscillations (m <1) nearly to the value 1.4V.

System answer results with P regulator

- For an order (ref) of the order of 2V applied to the system ordered by a Proportional regulator (P) with KP = 2, one gets the answer y (t) presented on the following Figure



Answer of the system ordered by proportional regulator P.

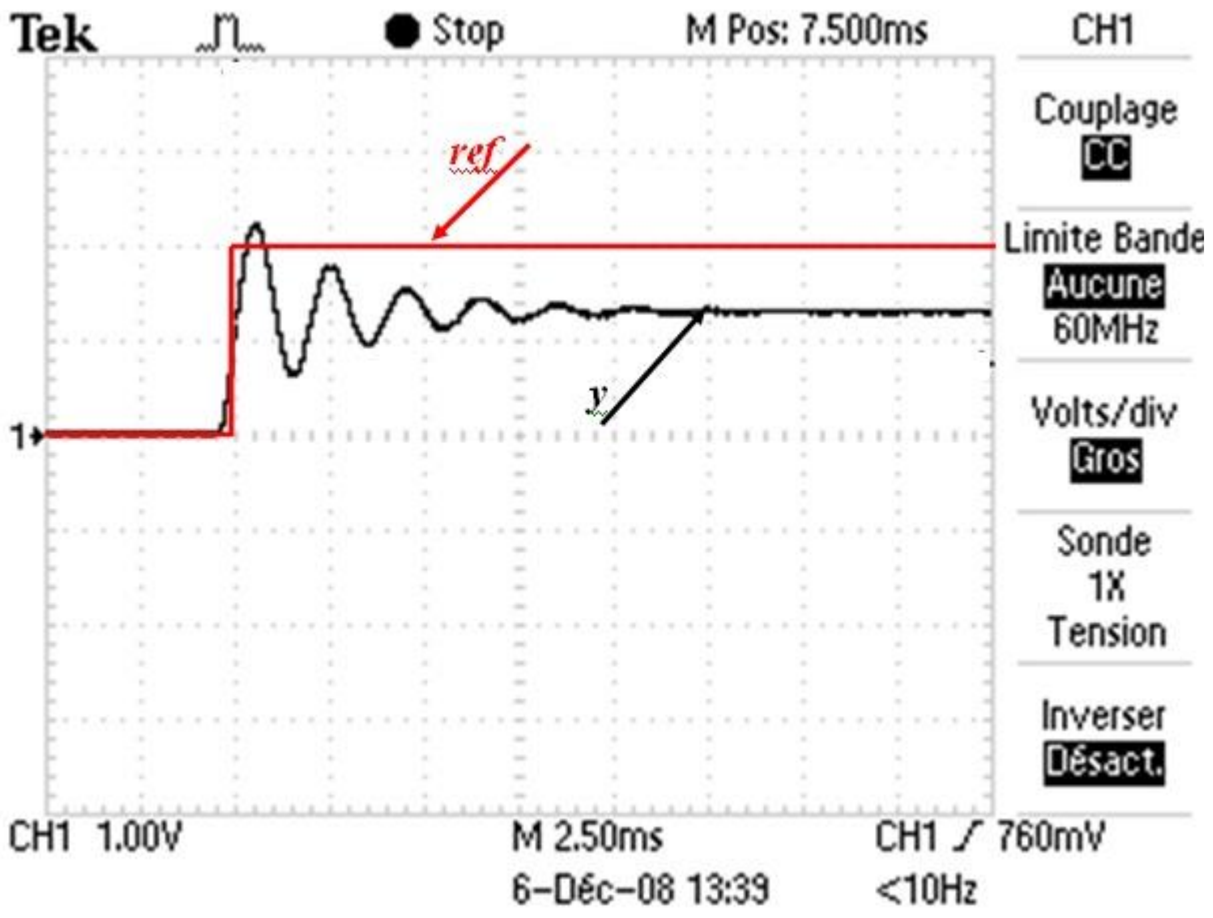
System answer results with P regulator

- One notices that this answer presents a static mistake of the order of 40%. Theoretically this mistake is given by:

$$\frac{1}{1+K_p\ K_s}\ 100\% = 42\ \% \qquad \text{E9}$$

System answer results with PD regulator

- For an order (ref) of the order of 2V applied to the system ordered by a PD regulator with KP = 2,KD=1 one gets the answer y (t) presented on the following Figure



Answer of the system ordered by proportional regulator PD.

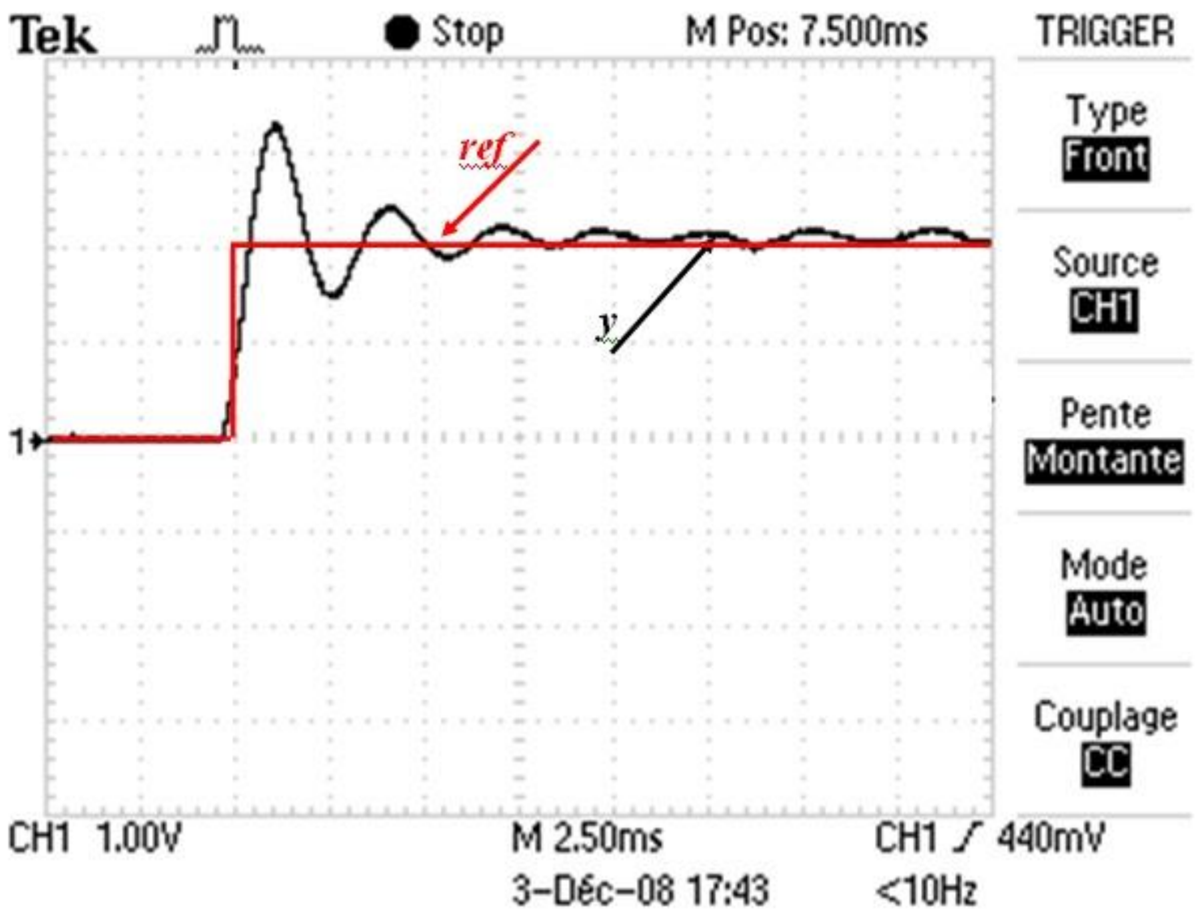
System answer results with PD regulator

(m<1)   E10

- One notices that the answer gotten present less oscillations that the one with the regulating P thanks to the Derivative action.

System answer results with PI regulator

- For an order (ref) of the order of 2V applied to the system ordered by a PI regulator with KP=2, KI=0,5 one gets the answer y (t) presented on the following Figure.



Answer of the system ordered by proportional regulator PI.

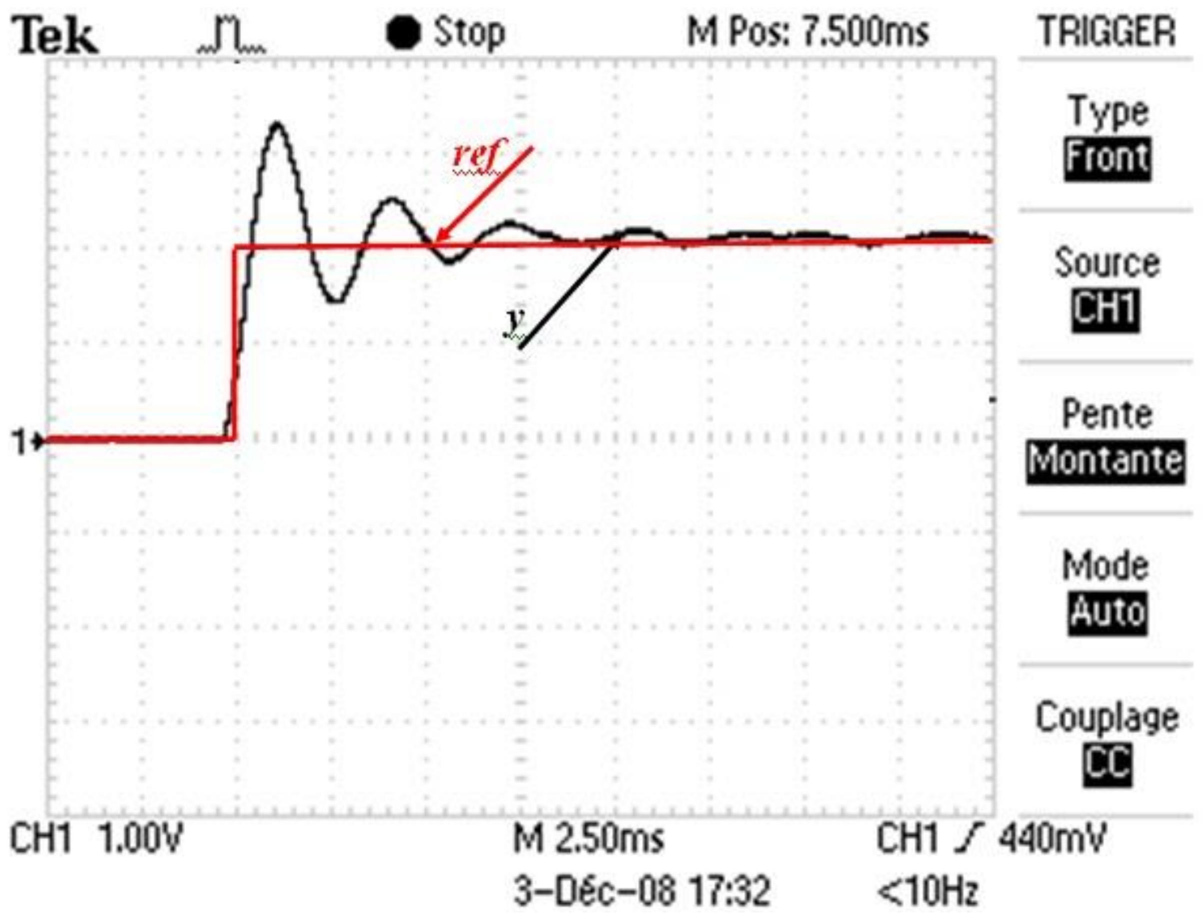
System answer results with PI regulator

$(m<1)$ E11

- one notices the annulment of the static mistake well thanks to the introduction of the **I** action

System answer results with PID regulator

- For an order (ref) of the order of 2V applied to the system ordered by a PID regulator with KP = 2, KD=1 and KI=0, 5 one gets the answer y (t) presented on the following Figure:



Answer of the system ordered by proportional regulator PID.

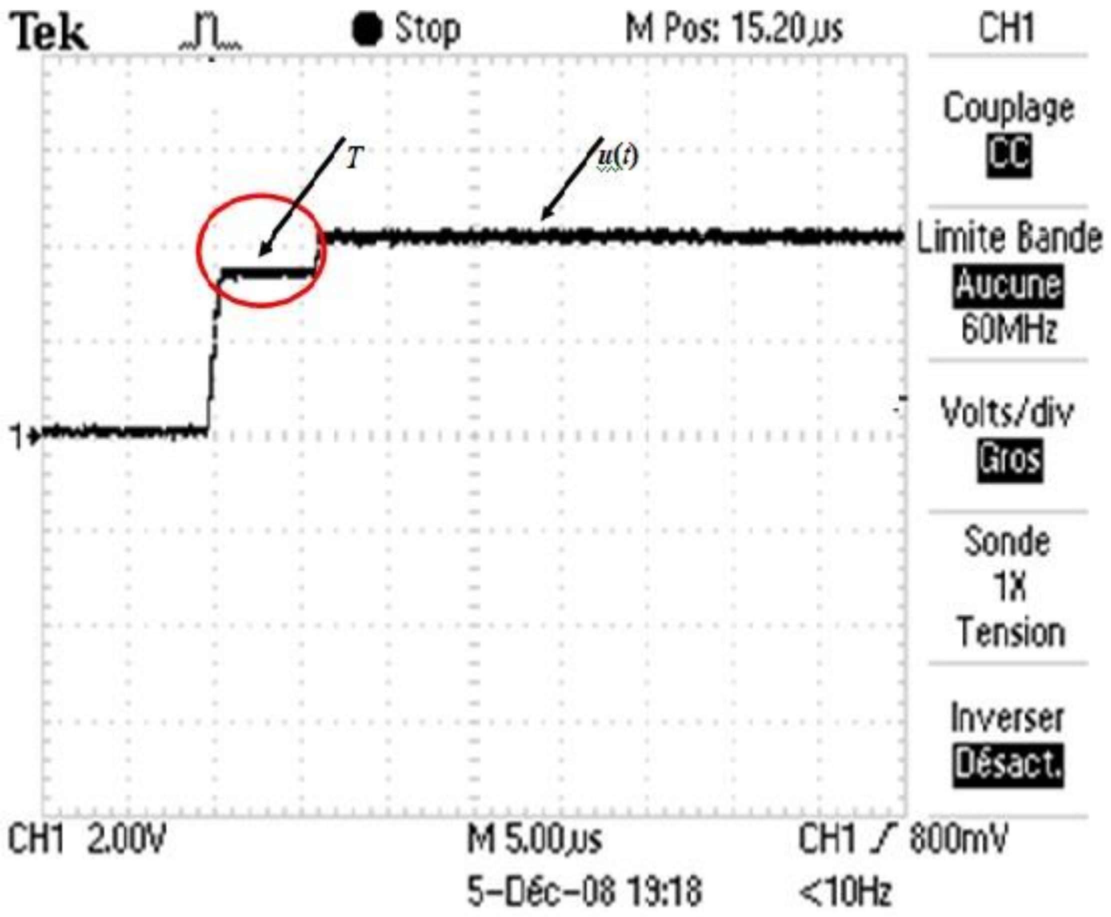
System answer results with PID regulator

(m<1) E12

- One notices that with the addition of the Derivative action, one has a light reduction of the oscillations in relation to the answer gotten by a regulating **PI**.

Tentative evaluation of sampling period

- While following the evolution of the order u(t) one could estimate the value of the sampling period experimentally (T) as it's indicated in the following figure:



Tentative evaluation of the sampling period (T).

Tentative evaluation of sampling period

- According to the figure 21, one estimates the value of the sampling period (T) that is the order of 6, 7µs.

Summary

SECTION 8

Summary

- A Finite State Machine control for a robot is presented. It is not the most advanced design, but it does show all the essential features of a robot control elements.
- An FPGA Implementation for the control feedback loop is also presented. This part can also be extended for many other robot designs.

Summary

- A digital PID controller implemented in FPGA technology is a configurable controller in terms of latency, resolution, and parallelism.
- The speed or execution or latency of the controller can be precisely controlled with the amount of reuse of arithmetic elements such as the speed of execution of FPGA based PID controller can be less then 100 ns if desired for high throughput requirements.
- Implementing PID controllers on FPGAs features speed, accuracy, power, compactness, and cost improvement over other digital implementation techniques.