EECE 499/693: Computers and Safety Critical Systems

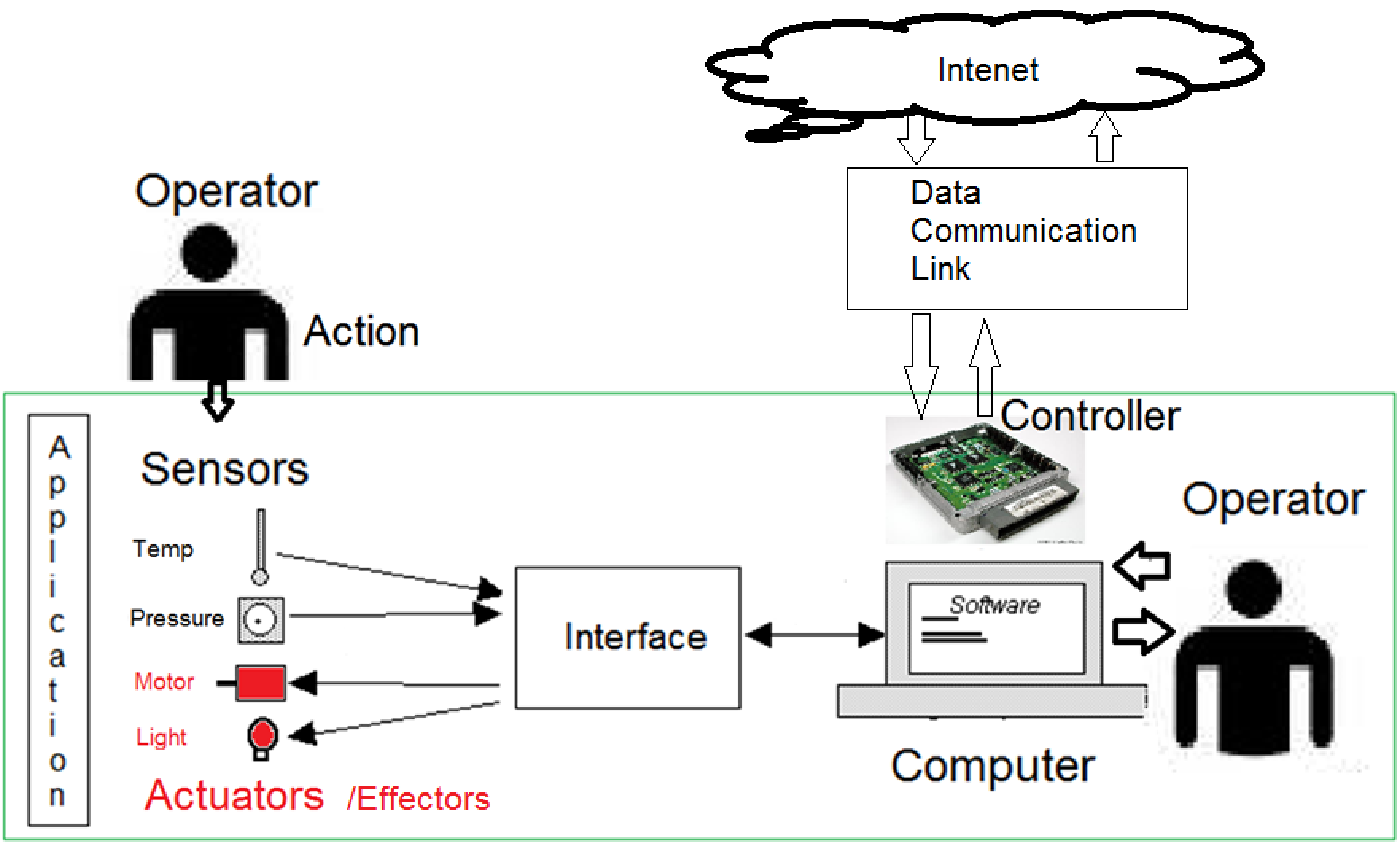
2 Computer Systems – Background Overview

Instructor: Dr. Charles Kim

Electrical and Computer Engineering
Howard University

www.mwftr.com/CS2.html

Computer System Overview

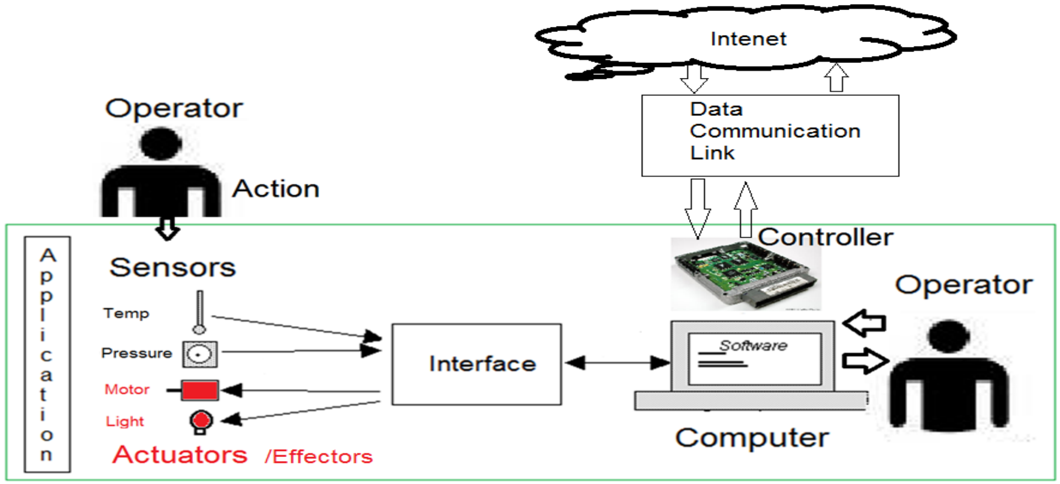


Modern Computer Control System

Hardware (Radio Network) behind the scene

Hardware (Radio Network) behind the scene

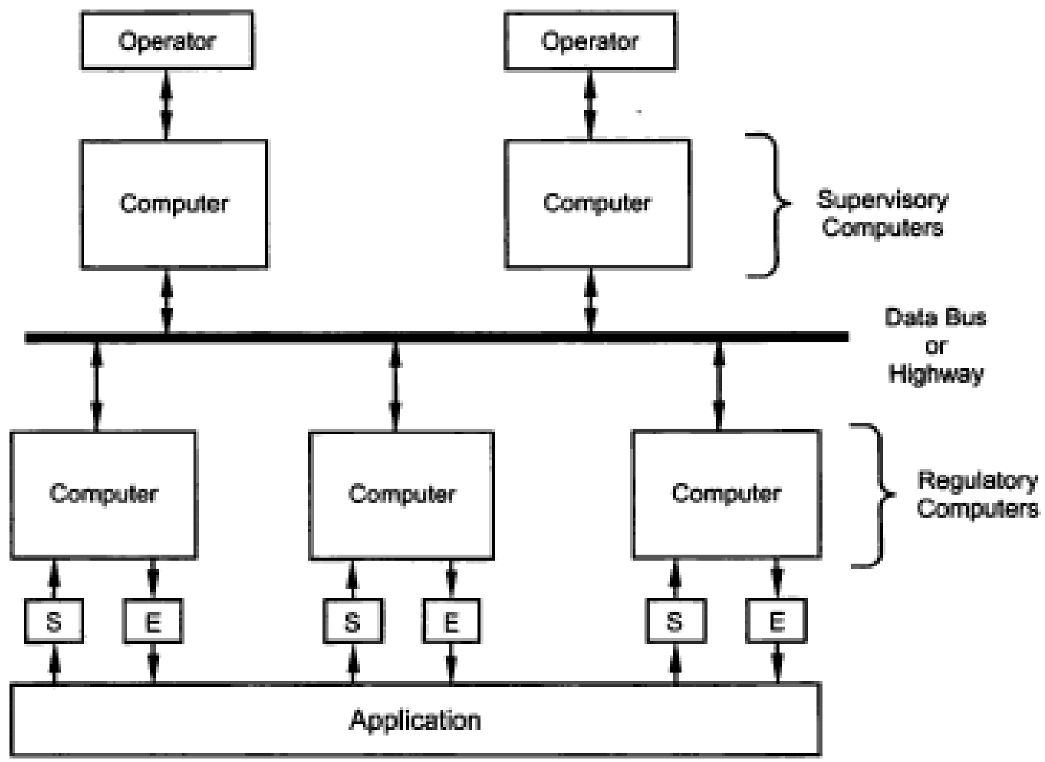
Computer System Overview



- Application
  - The application consists of the physical entity whose function and operation is being monitored and controlled
- Computer
  - The digital hardware and software that is monitoring and controlling the application in real time.
- Effector/Actuator
  - A device that converts an electrical signal from the output of the computer to a physical quantity which controls the function of the application.
- Sensor
  - A device that converts an application's physical quantity into an electrical signal for input into the computer
- Operator
  - The human or humans responsible for monitoring and operating the computer system in real time.
- Data Communication Link
  - A two-way pathway for transferring data between the computer and other external computers.

Distributed Computer Control System

- 2 Levels of Computer Control
  - Supervisory control
    - Supervise the regulatory computers
    - Monitoring
  - Regulatory control
    - Regulate or control the detailed tasks for given application
- Computer communication between 2 levels
  - Databus



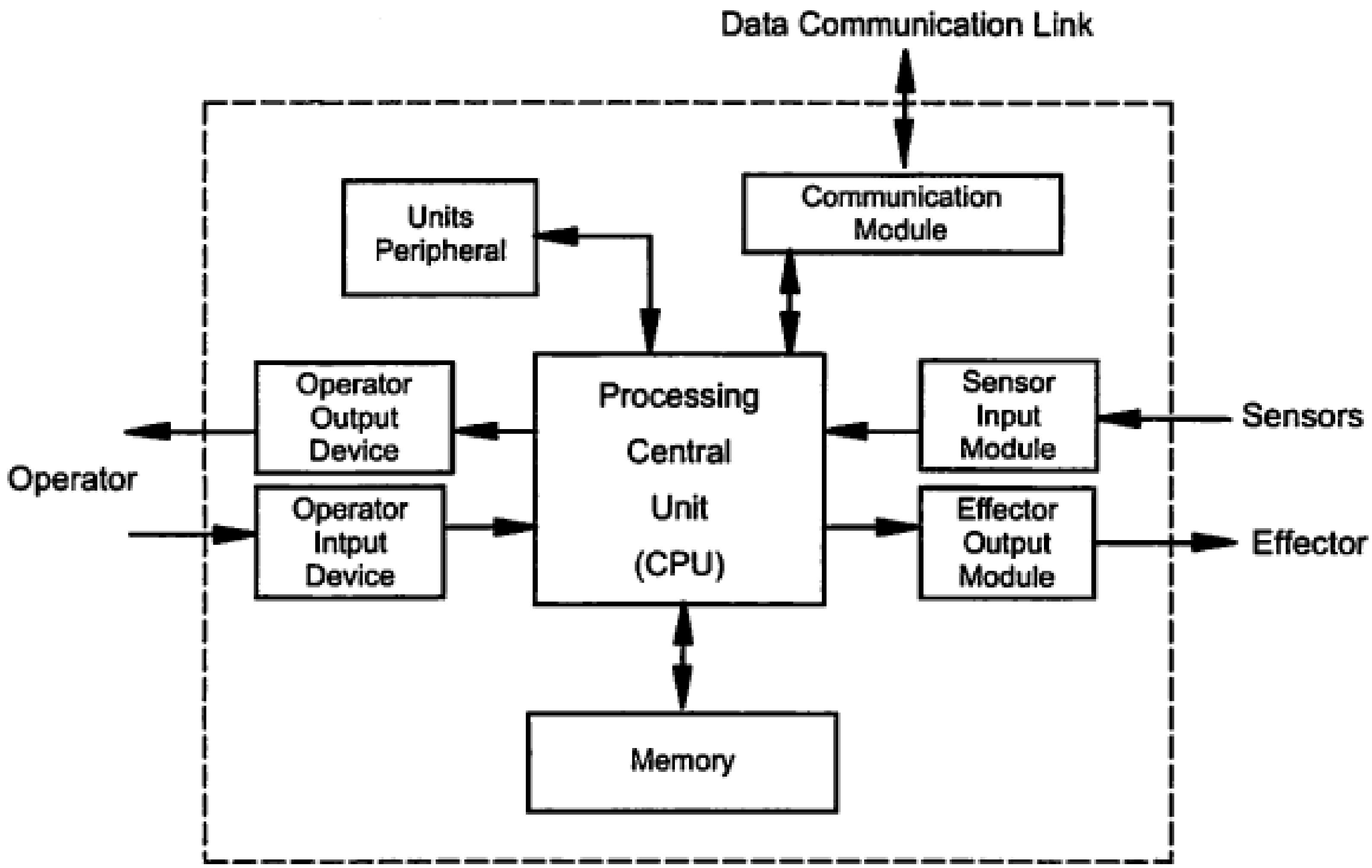
What do we want to accomplish from this subject?

- **<u>Eventual goal</u>:** Design a complete computer control system with practical hardware components and even with software (with pseudo coding)
- **<u>Learning Steps:</u>**
  - Computer Architecture for Control Application
  - Application examples and their computer control system architectures
  - How inputs (from sensors) and outputs (to the actuators/effectors) can be selected, ported, connected, and coded
  - How microprocessors/microcontrollers are programmed for the necessary I/O patterns

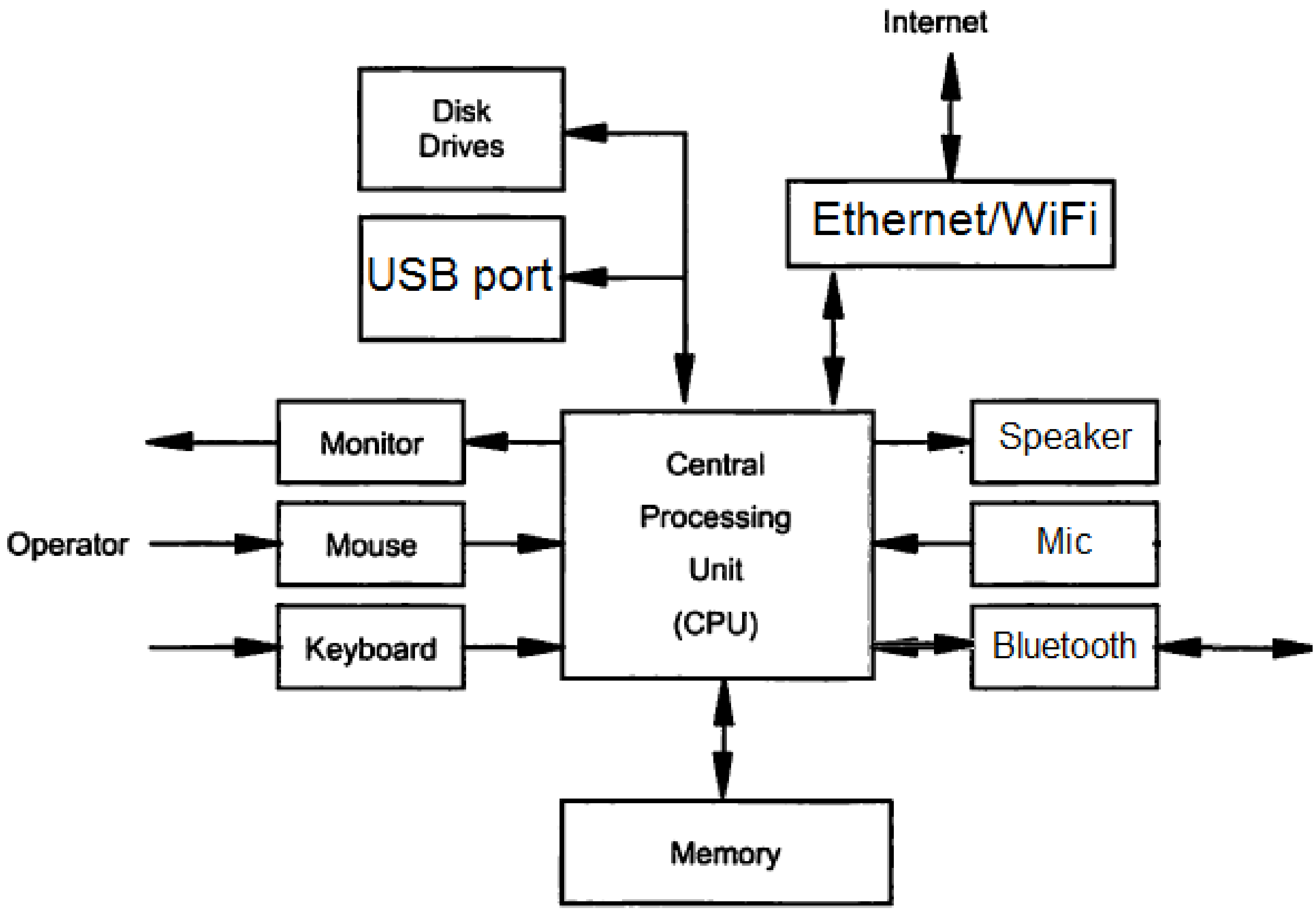
Application (control system) list

Computers and Safety Critical Systems	
Fall 2014	
Name	**Application (Control Systems)**
Ajayi, Joshua D.	Lighting Control System
Celestina Carvalho, Marcos	Breaker/Accelerator Control System
Davis, Caleb A.	Blind Spot Asisstant Control System
Green, Naja J.	Doors and Seats Control System
Little, Penny J.	
Lopera, Jonathan J.	Airbag Control System
Monette, Jordan A.	Lighting Control System
Mwandu, Sarah M.	Seats Control System
Robinson, Michael A.	Transmission Control System
Smith, Darrell E.	Fuel/Air Control System
West, Derek	
Woods II, De'Shawnn L.	Auto Collision Detection System
Wren, Jordan R.	Airbag Control System
Alaie Faradonbeh, Misagh	Collison Avoidance System
Ayele, Samuale	Electronic Accelerator and Breaker Control System
Charles, Pierre E.	Airbag Control System
Idubor, Ayobami O.	Lighting Control System
Nahar, Kamrun	Automatic Braking System
Sims, Nathaniel A.	Daylight Running Lighting Control System
Woods, D'Angelo R.	Automatic Parking Control System

Control Computer Architecture

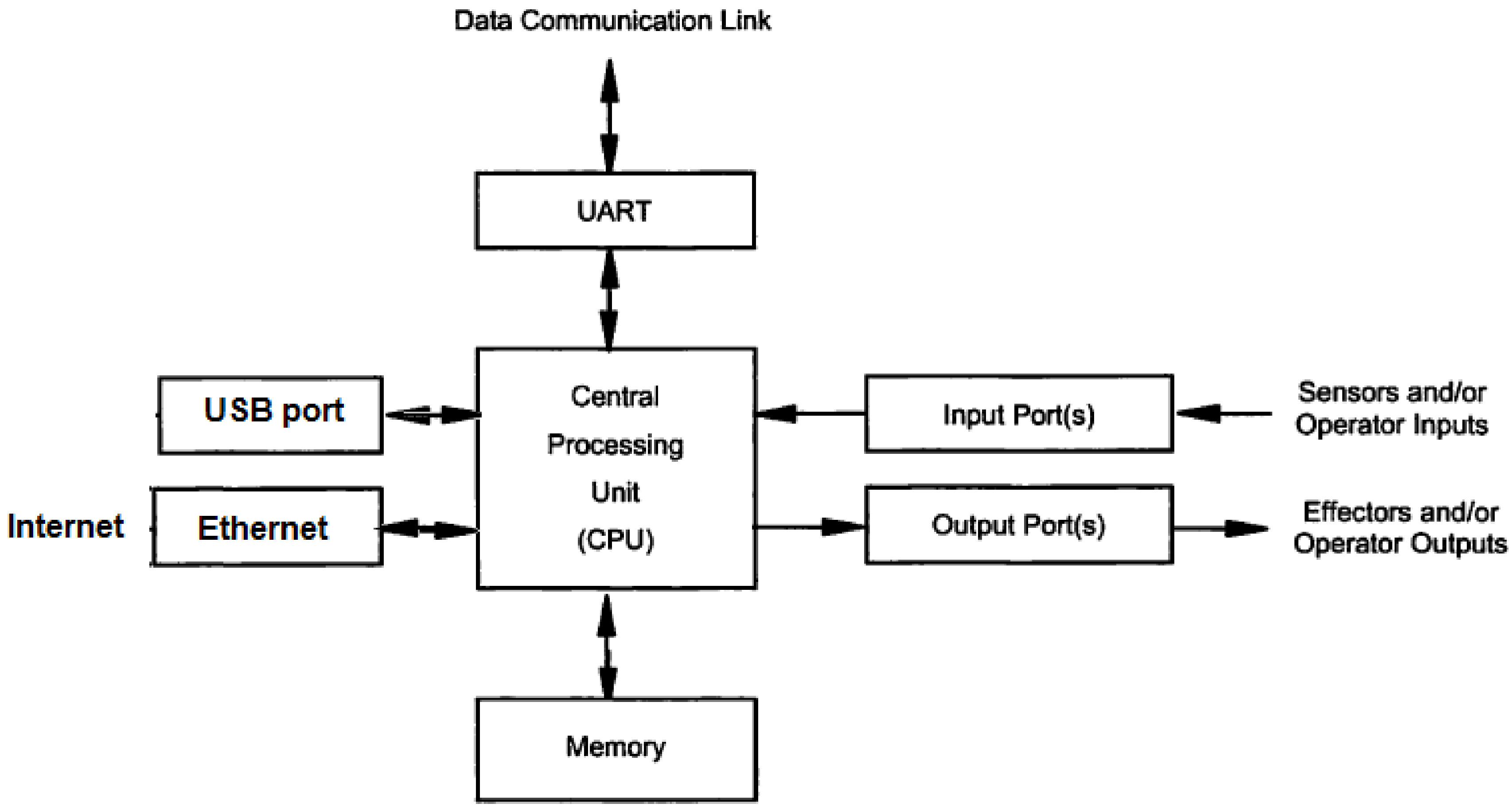


Control PC/Laptop Structure

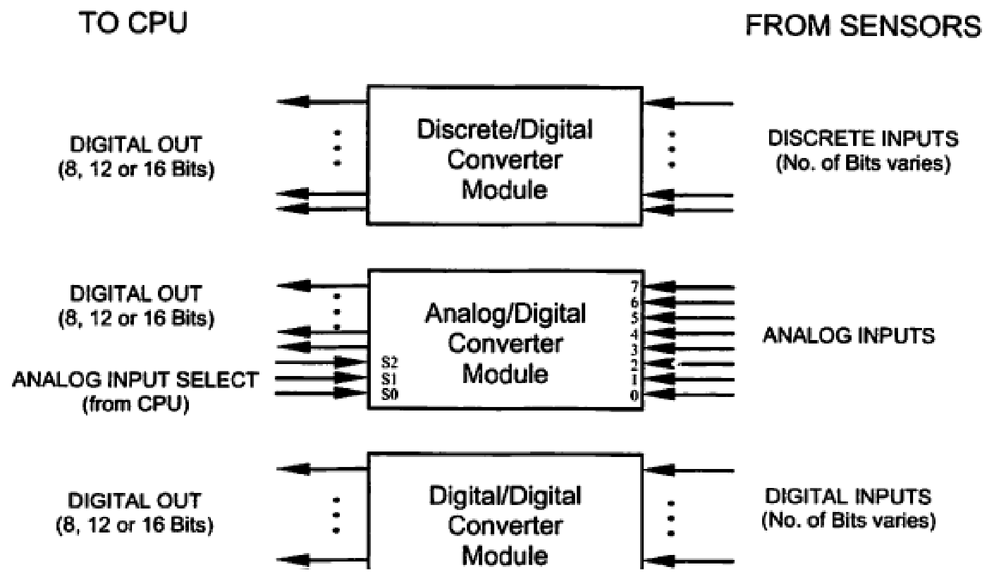


Control Single-Chip Microcontroller Structure

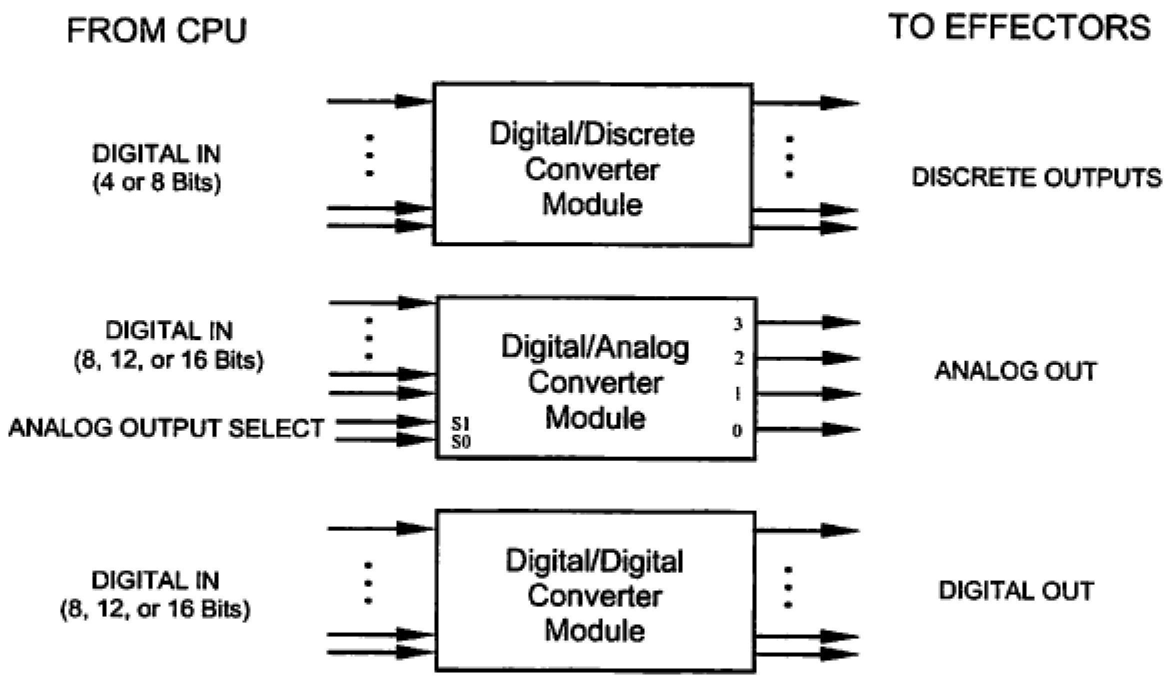
- SoC & Mobile Computing



Sensor Inputs with DI and AI



Actuator Outputs – DO and IO

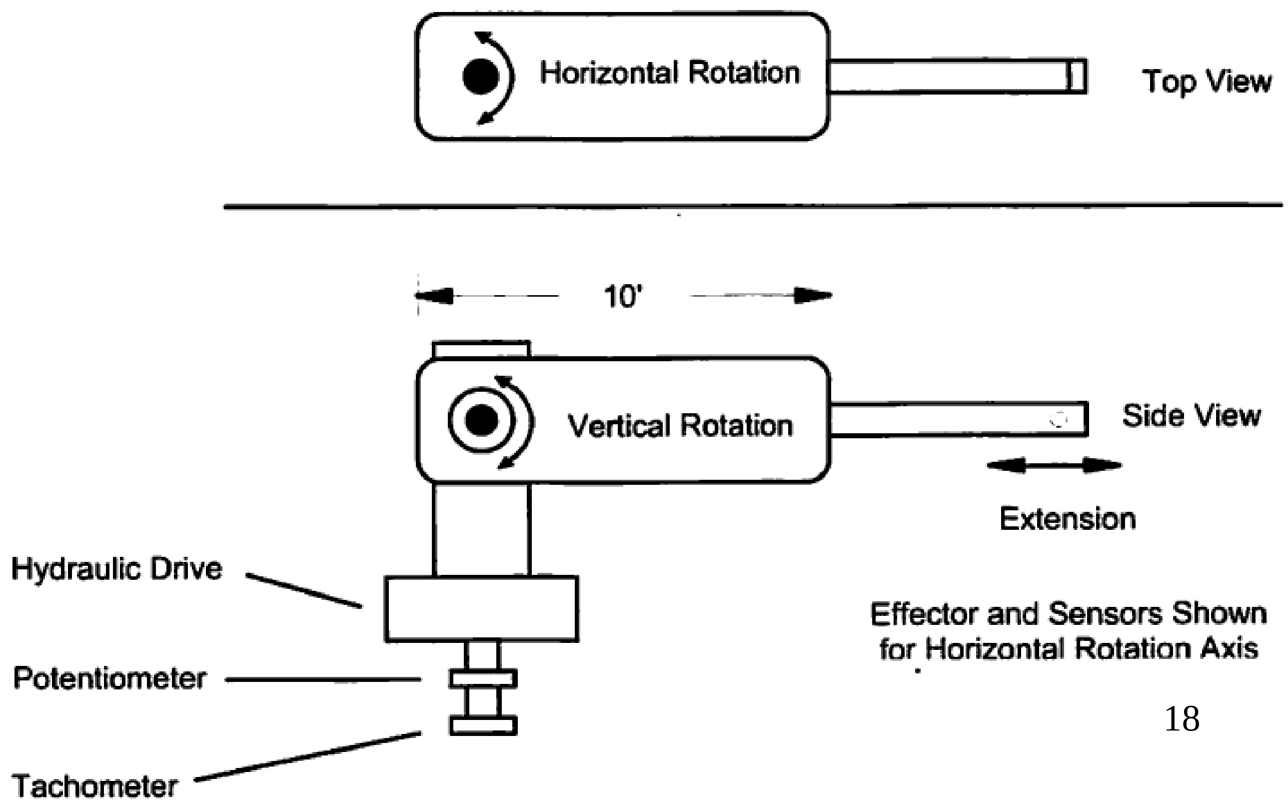


Operator I/O Devices

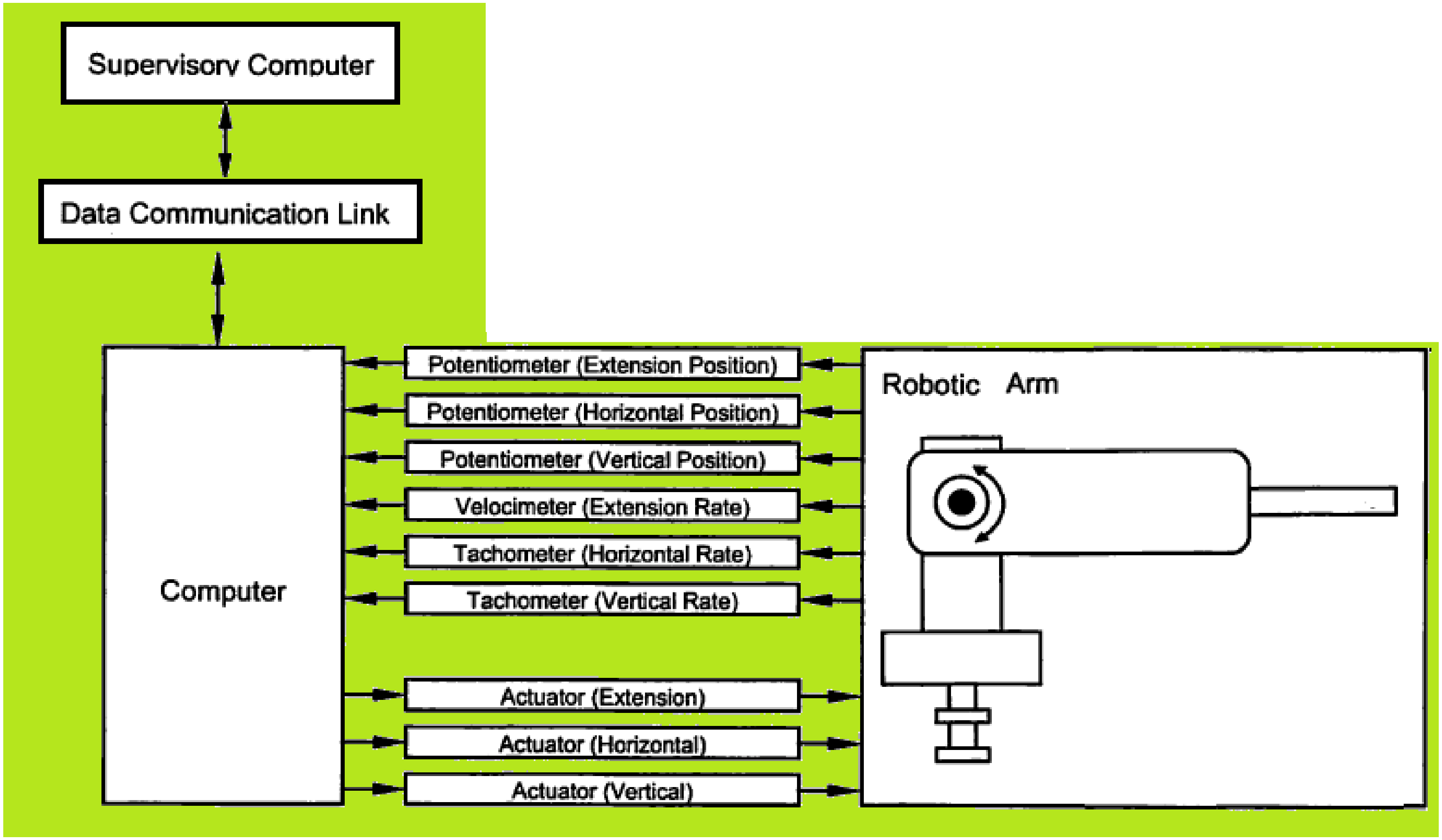
Input Device	Output Device
Joystick	Annunciator light
Keyboard (standard or enhanced)	CRT (cathode ray tube) video display
Keypad (numeric, touch-tone, or custom)	Incandescent lamp
Light pen	Klaxon
Microphone	Light emitting diode
Mouse	Liquid crystal display
Potentiometer	Loudspeaker
Switch (momentary)	Panel meters
Switch (on/off)	Seven segment display
Thumbwheel switch	
Touch screen	
Trackball	

Application Example 1

- **Robot Arm Control**
  - Movement: Horizontal, Vertical, Extension
  - Position Sensor: Potentiometer for measurement of vertical and horizontal angles and linear displacement of the end piece
  - Rate Sensor (for measurement of the speed of the changes of the position in each of the three axes): Tachometers (for horizontal and vertical axes) and Velocimeter (for the end piece)

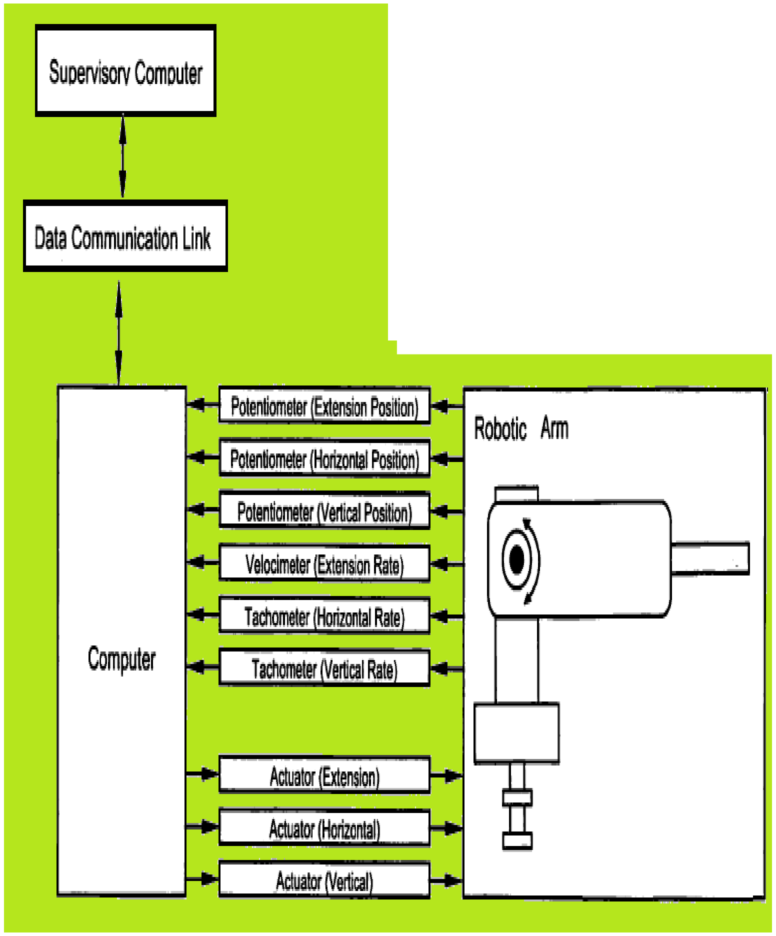


Robot Arm Computer Control System – Block Diagram



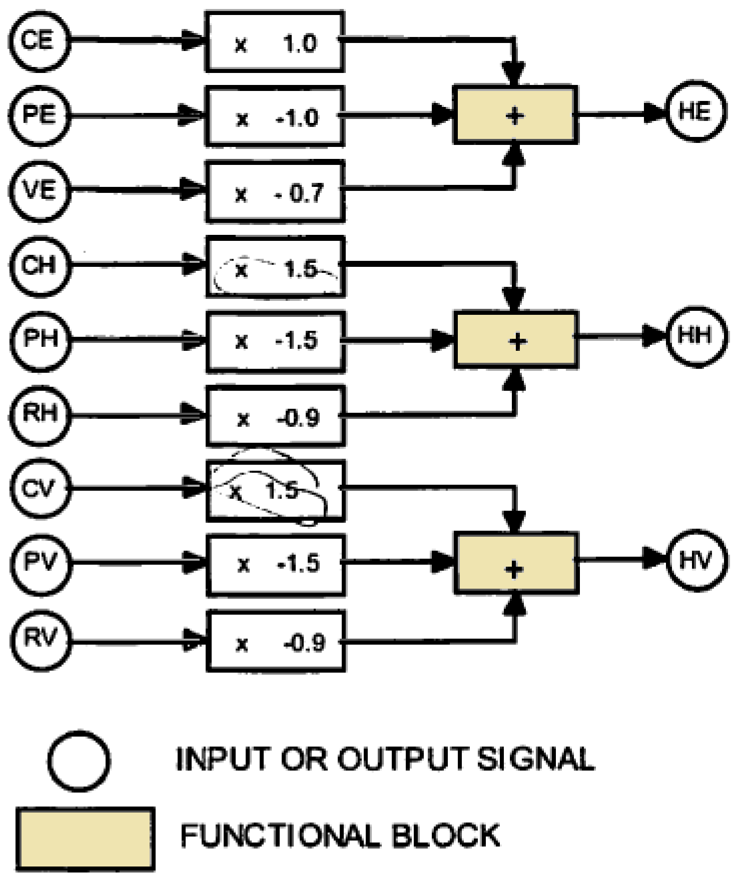
Robot Arm Computer Control System – Software Structure

- Input
  - Required arm positions (in 3 axes) transmitted from the supervisory computer
  - Required extension position from the supervisory computer
- Output
  - Move the arm so that its axes are aligned with the commanded position, while checking the speed so that extension is smoothly done
- Control Algorithm and Feedback for accurate position control



Functional Diagram

- Graphical expression of software requirement to represent Input/Output relationship



HE = 1.0 CE -1.0 PE - 0.7 VE
HH = 1.5 CH -1.5 PH - 0.9 RH
HV = 1.5 CV -1.5 PV - 0.9 RV

Symbol	Signal Name	Signal Type
CE	Commanded position – extension axis	Digital in
PE	Position – extension axis	Analog in
VE	Velocity – extension axis	Analog in
CH	Commanded position – horizontal axis	Digital in
PH	Position – horizontal axis	Analog in
RH	Rate – horizontal axis	Analog in
CV	Commanded position – vertical axis	Digital in
PV	Position – vertical axis	Analog in
RV	Rate – vertical axis	Analog in
HE	Hydraulic actuator – extension axis	Analog out
HH	Hydraulic actuator – horizontal axis	Analog out
HV	Hydraulic actuator – vertical axis	Analog out

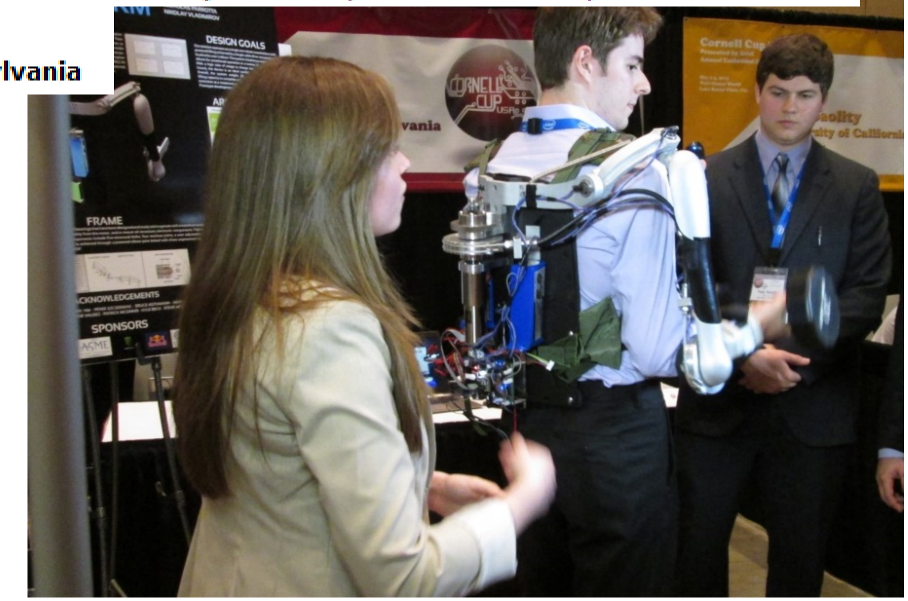
Intel-Cornell Cup

Robot Arm Computer Control System – Advanced Application

The challenge is to build an exoskeletal system that is inexpensive, streamlined, and wireless.

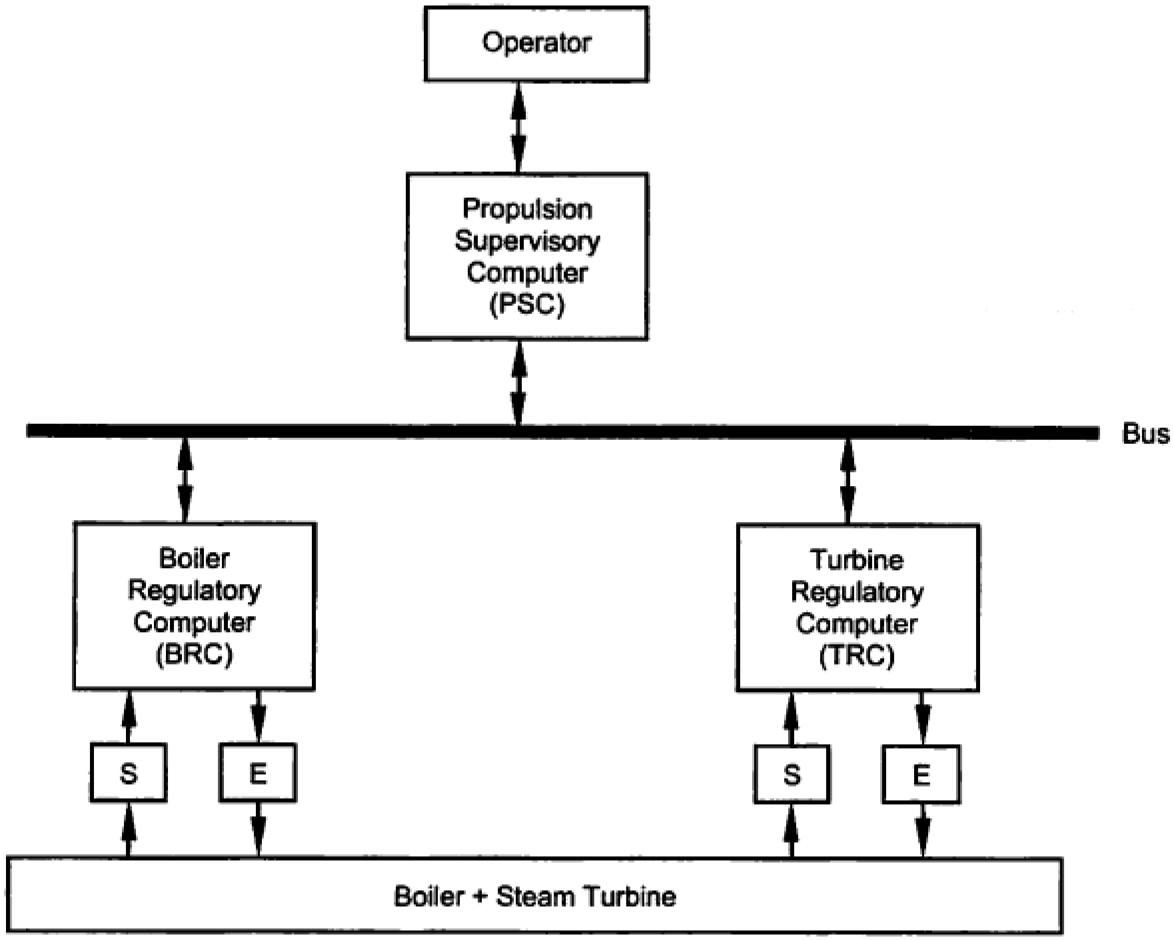
Team Summaries - 2013

Titan
University of Pennsylvania



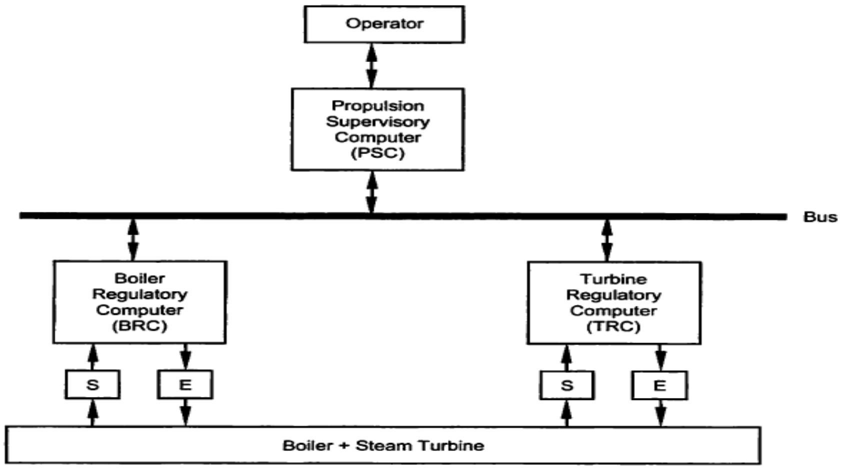
Award-Winning Robotic Exoskeleton Boosts Your Arm Strength

Application Example 2 – Steamboat Propulsion System



- 2 subsystems
- 1 Supervisory computer (located one the ship's bridge)
- Operator commands propulsion power
- 3 Computers are connected on a single bus

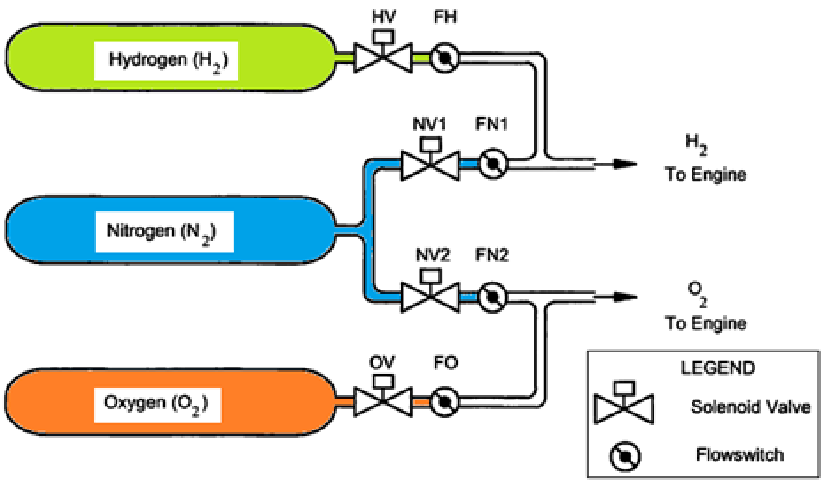
Application Example 2 – Steamboat Propulsion System



- Bus controller
  - To keep computers communicate orderly
  - Prevent multiple simultaneous transmission of command or data
- Communication Protocol
  - A set of pre-defined sequences to conduct orderly communication
  - Master or Slave
  - Sync + Source ID + Destination ID + Data

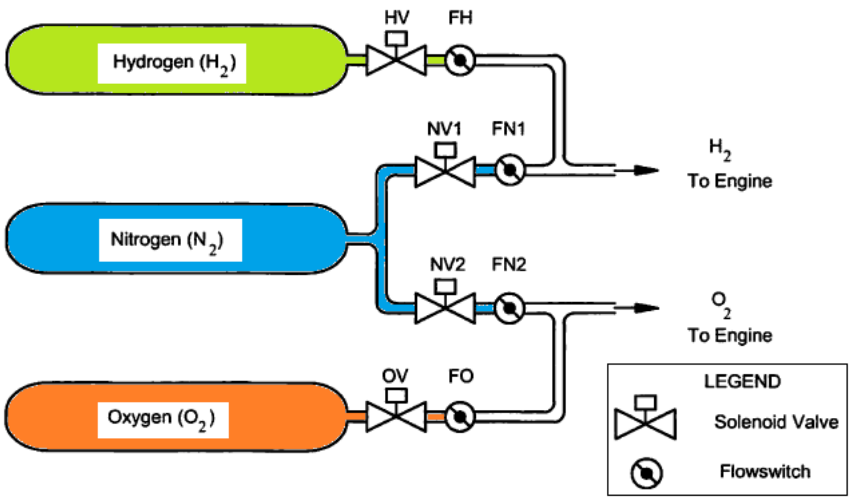
Application Example 3: Test Jet Propulsion System (for Software System discussion)

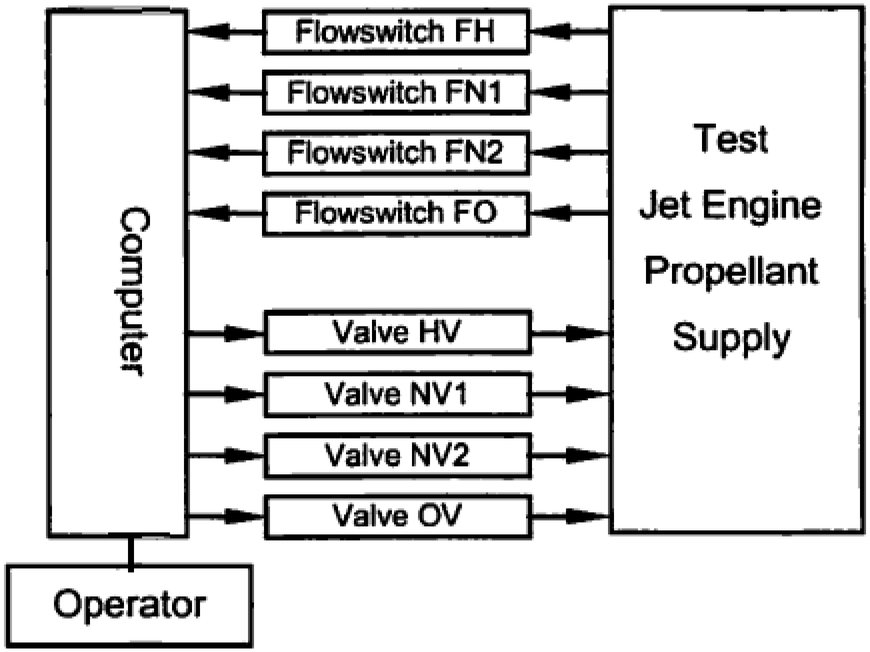
- Instructions located in the memory which are brought in by CPU to perform Arithmetic and Logic operations
- **Example** is given with Test Jet Engine Propellant Supply system: A jet engine is to be supplied with a mixture of hydrogen and oxygen to provide considerable propulsion energy with minimal air pollution.
- Nitrogen is available to purges the lines of any residual H2 and O2 to prevent fire or an explosion after experimental run.



Test Jet Propulsion – Computer System

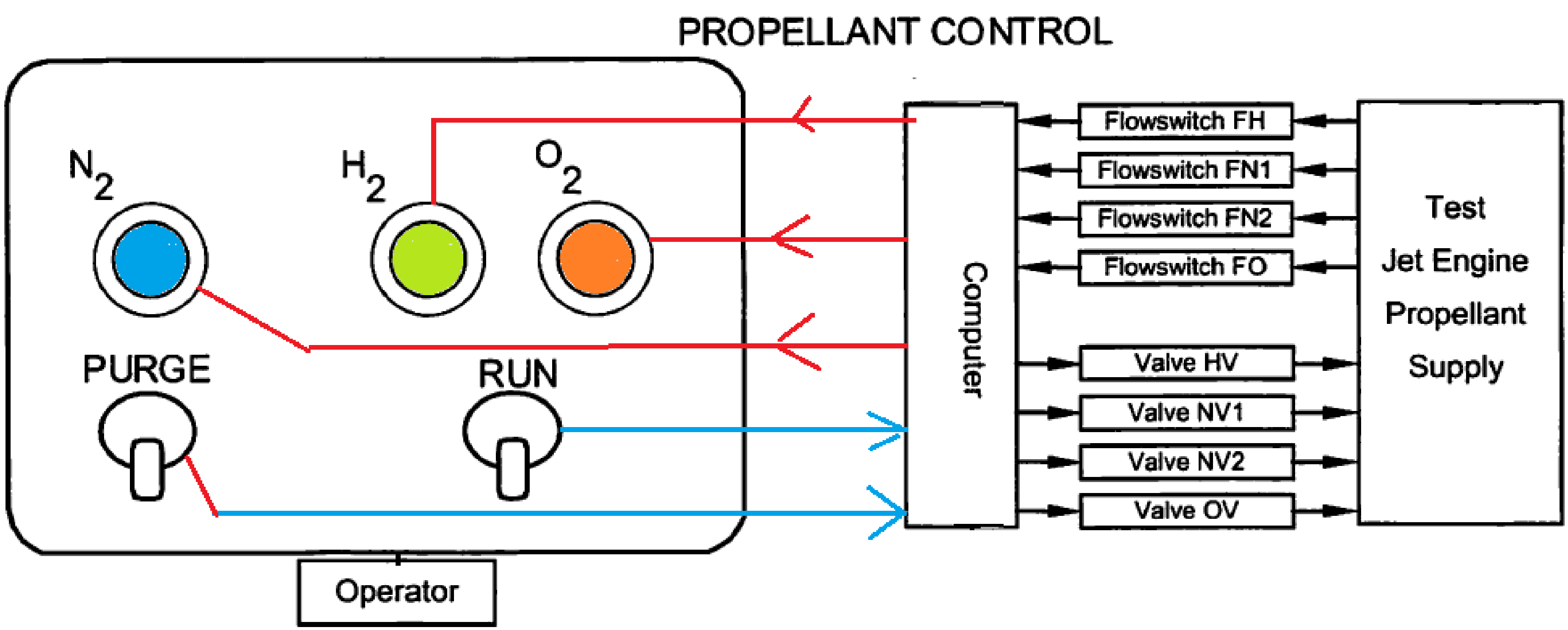
- Valves (HV, OV, NV1, and NV2) : actuators
  - NC (Normal-Close)
  - Opened by applying electric current
- Flow Switches (FH, FN1, FN2, FO) : Sensors
  - NO (normal Open) Position (Output): Or HIGH
  - Sensitive to the presence of gas flow
  - Closed Position (Output) when there is a gas flow: Or LOW





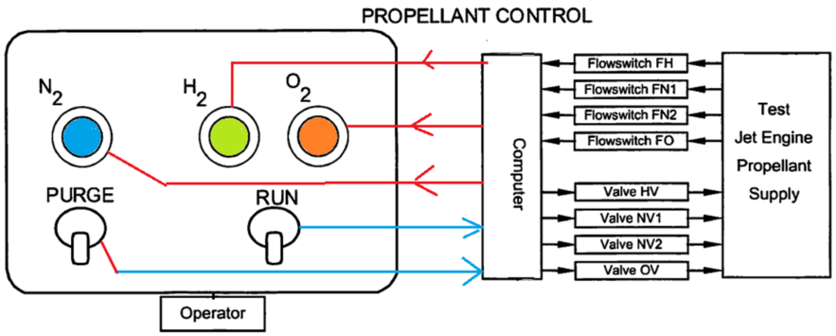
Complete Compute System with Operator Control Panel

- Two control switches:
  - RUN: turn on the flow of the propellants (H2 and O2) – initially OFF
  - PURGE: Supply purge gas (N2) to the system – initially OFF
- Three lamp indicators
  - Flows of H2, O2, and N2



Computer Software Requirements

- **<u>Software Requirement</u>** of the Test Jet Propulsion Supply system
  - Complete, clear, and correct definition of what the S/W is supposed to do
  - **Requirement (Truth Table) for Switch Position vs. Valve State**
  - **Requirement (Truth Table) for Flow Switch Output vs, Light State**

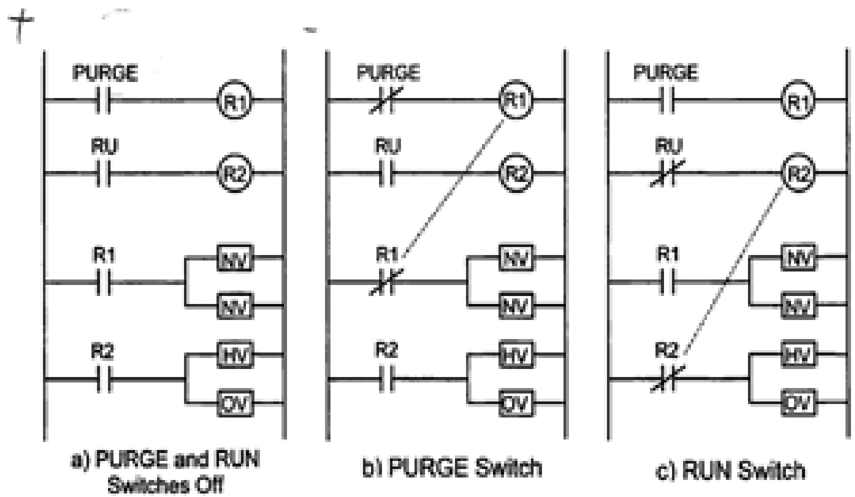


Switch Position		Valve State			
RUN	PURGE	HV	NV2	NV1	OV
0	0	0	0	0	0
0	1	0	1	1	0
1	0	1	0	0	1

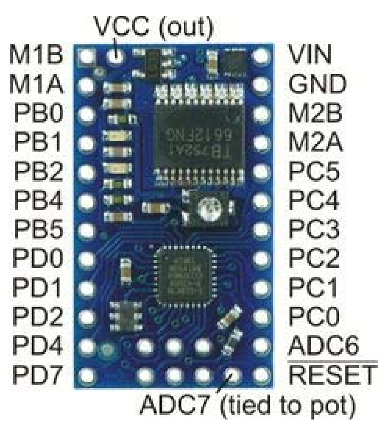
Flow Switch Position				Light State		
FH	FN2	FN1	FO	O	H	N
0	0	0	0	0	0	0
0	0	0	1	1	0	0
1	0	0	1	1	1	0
0	1	1	0	0	0	1
1	0	0	0	0	1	0

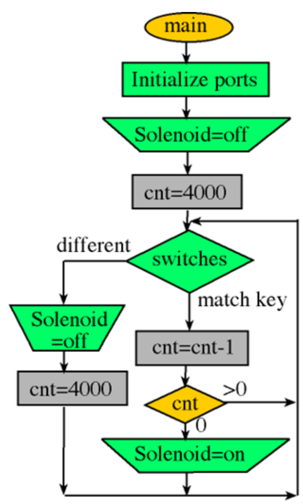
Conversion of Software Requirements to Coding/Programming

- **1 Ladder Diagram Approach (for PLC)**



- **2 Hardware Programming Approach (for Microcontroller)**





Pseudo Code

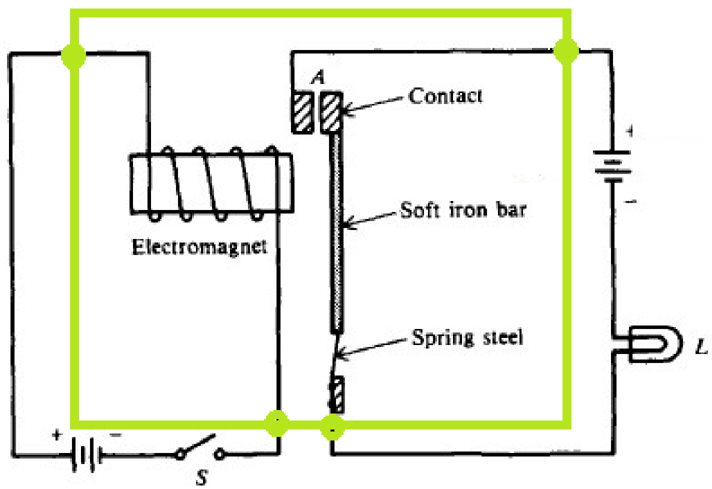
1) initialize ports
   PA6-PA0 inputs
   PA7 output
2) turn off solenoid
3) set counter to 4000
4) repeat indefinitely
   if switch matches key
     a) decrement counter
     b) if counter is zero
        turn on solenoid
   otherwise
     a) turn off solenoid
     b) set counter to 4000

C Code

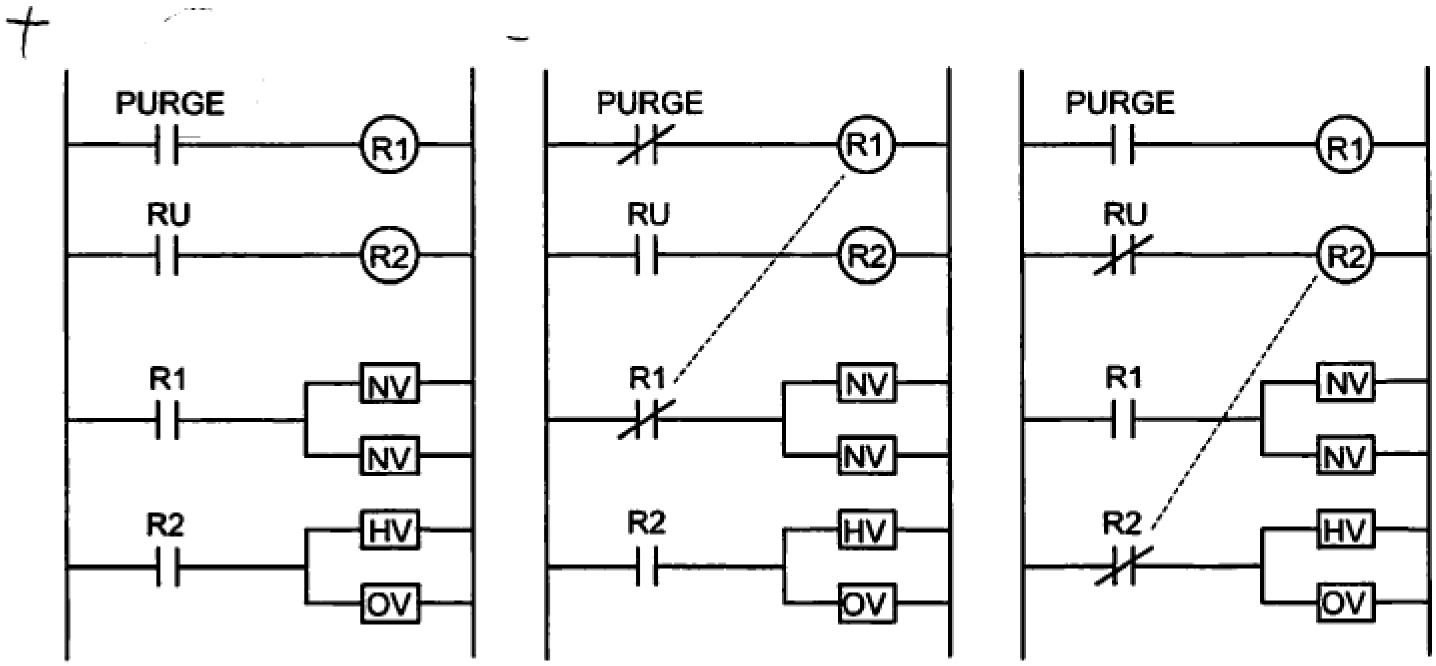
```
DDRA=0x80;
PORTA=0; cnt=4000;
while(1){
  if((PORTA&0x7F==key){
    if((--cnt)==0)
      PORTA|=0x80;}
  else{
    PORTA=0; cnt=4000;}}
```

1 Ladder Diagram Approach (for PLC)

- Relay
  - Digital logic was based on relay (before low-cost electronic systems)
  - Electric switch actuated by a magnetic field created by electric current
- Ladder Diagram
  - Vertical lines: Electrical wires (SRC and GND) on which current can flow horizontal components
  - Horizontal components
    - Switch Contacts
    - Relay Coils
    - Solenoids



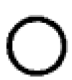
Ladder Diagram for RUN and PURGE



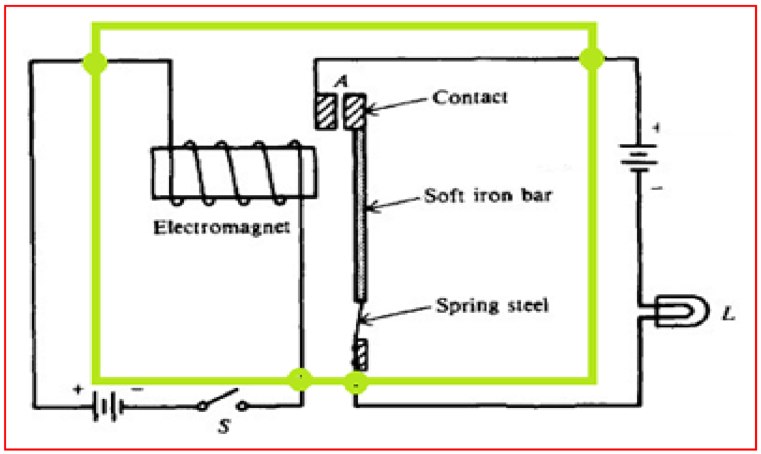
a) PURGE and RUN Switches Off

b) PURGE Switch

c) RUN Switch



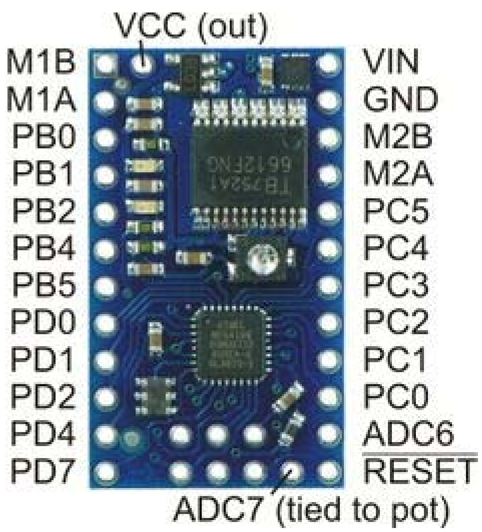
Relay Coil

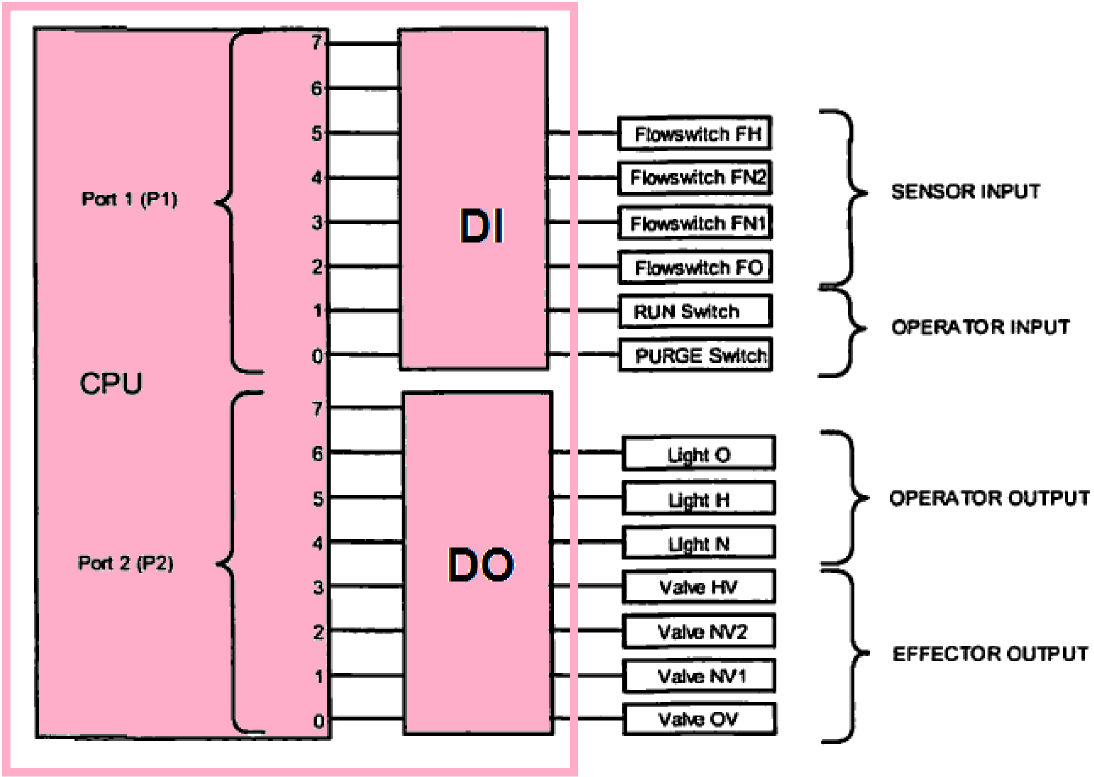


Contact (Open)

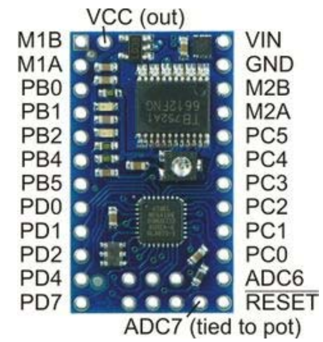
2 Hardware Programming Approach

- Programming Approach: “Picture” of hardware environment → Pin/Port assignment
- Pseudo-Coding
- Coding





Software Design for Programming Approach



- Step 1:
  - Read PORT 1 to get Flow Position Information & Operator SW positions (BYTE read or BIT read?)
- Step 2:
  - Send out correct information to PORT 2 by the Software Requirement Truth Table (BYTE out or BIT out?)
- **FLOWCHART**
- **Safety-Related Questions**
  - How to get the FLOW POSITION at the same time?
  - How to get the SW positions at the same time?
  - What would be a problem when FLOW POSITIONS are read sequentially one at a time?
  - What would be a problem when SW POSITIONS are read sequential one at a time?
  - What are you going to do if PURGE is ON while RUN is OFF? How do you avoid this?

Compute System with Operator Control Panel –Activity

- **Step 1**
  - **Complete block diagram of your SYSTEM (of automobile electronic control system)**
  - **Computer**
  - **Sensors**
  - **Actuators**
  - **Operator control panel**
  - **Others – if your control system requires**

Computer Software Requirements - Activity

- **Step 2:**
  - **<u>Software Requirement</u>** of YOUR SYSTEM of automobile electronic control system
  - Complete, clear, and correct definition of what the S/W is supposed to do
  - **Truth Table Approach**
  - **Ladder Diagram Approach**

Hardware Programming Approach - Activity

- **Step 3**
  - Programming Approach: “Picture” of hardware environment → Pin/Port assignment
  - Choose a microcontroller
    - Arduino
    - Galileo
    - Raspberry Pi
    - Altera FPGA
    - Digilent FPGA
    - Basic Stamp 2

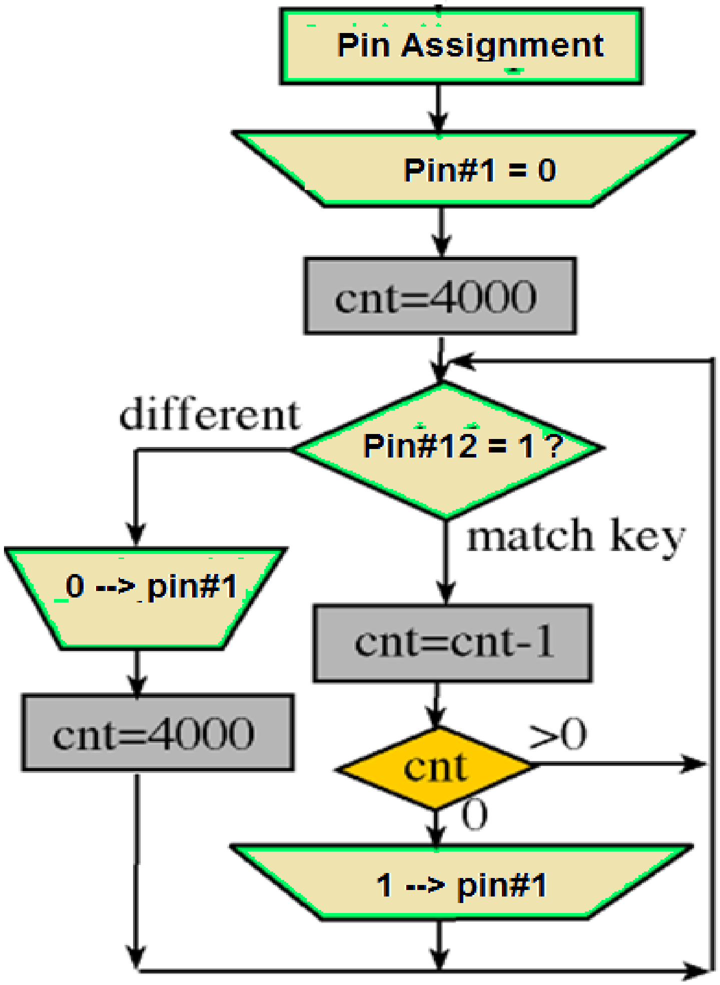
Side Bar:
Assignment #2: "Computer Caused Accidents"

Invited Presentation

- Accidents reported
  - Aircraft/Drone – 6
  - LAX – 1
  - Car – 4
  - Voting – 1
  - Rail – 1
  - Prison – 1
  - Metro – 1
  - Hospital – 1
  - Ship – 1
  - Elevator – 1
  - Y2K - 1
- Invited Presentations
  - Green – Maximum Security Prison
  - Mwandu – Costa Concordia
  - Sims – Drone
  - Woods – Nissan Airbag Recall

Flowchart & Pseudo-Coding - Activity

- **Step 4 – Flowchart**
  - Use **pin numbers** only



- **Step 5 – Pseudo Coding**
  - Use **pin numbers** as variables

Pseudo Code

1) initialize ports
   pin#1 set as output
   pin#12 set as input
2) **0 --> pin#1**
3) set counter to 4000
4) repeat indefinitely
   if pin#12=1
   a) decrement counter
   b) if counter is zero
      1 --> pin#1
   if pin#12=0
   a) **0 --> pin#1**
   b) set counter to 4000