Dec14-08 Powering the PUMA

Project Plan

Team Members:
Matt Bogenschultz
Alex Grieve
Nhat Pham
Zeyu Zhang

Client:
Dr. Greg Luecke

Advisor:
Dr. Greg Luecke

Revision History

Version	Description	Date
1.0	Initial project plan written	February 19, 2014
1.1	Updated to reflect new requirements – must use existing DAQ board provided by client, and must design a PID controller	March 14, 2014
1.2	Revised system block diagram, added testing plan and Gantt chart for project schedule	April 2, 2014

Table of Contents

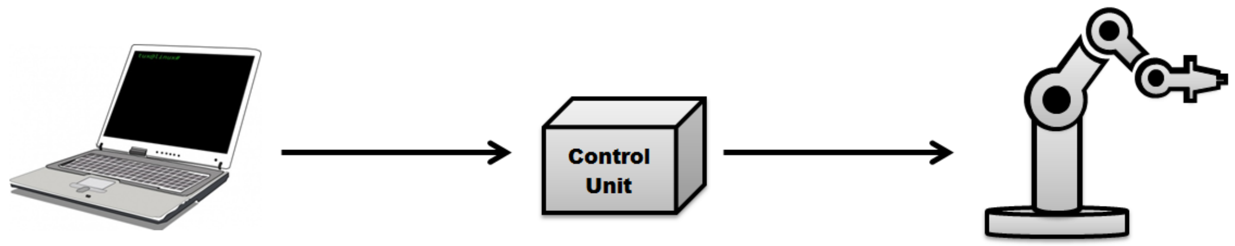
Background

The Unimation Programmable Universal Machine for Assembly (PUMA) is an industrial robot arm with six degrees of freedom. It has a main control unit that sends commands to the PUMA arm and samples feedback from the PUMA arm's sensors. The control unit also supports two mechanisms to program the PUMA arm. The first is a teach pendant that is used to manipulate the six joints and record their positions. The second is a terminal interface where the PUMA arm can be programmed in a language called VAL. The control unit has a floppy disk drive for storing and loading programs.

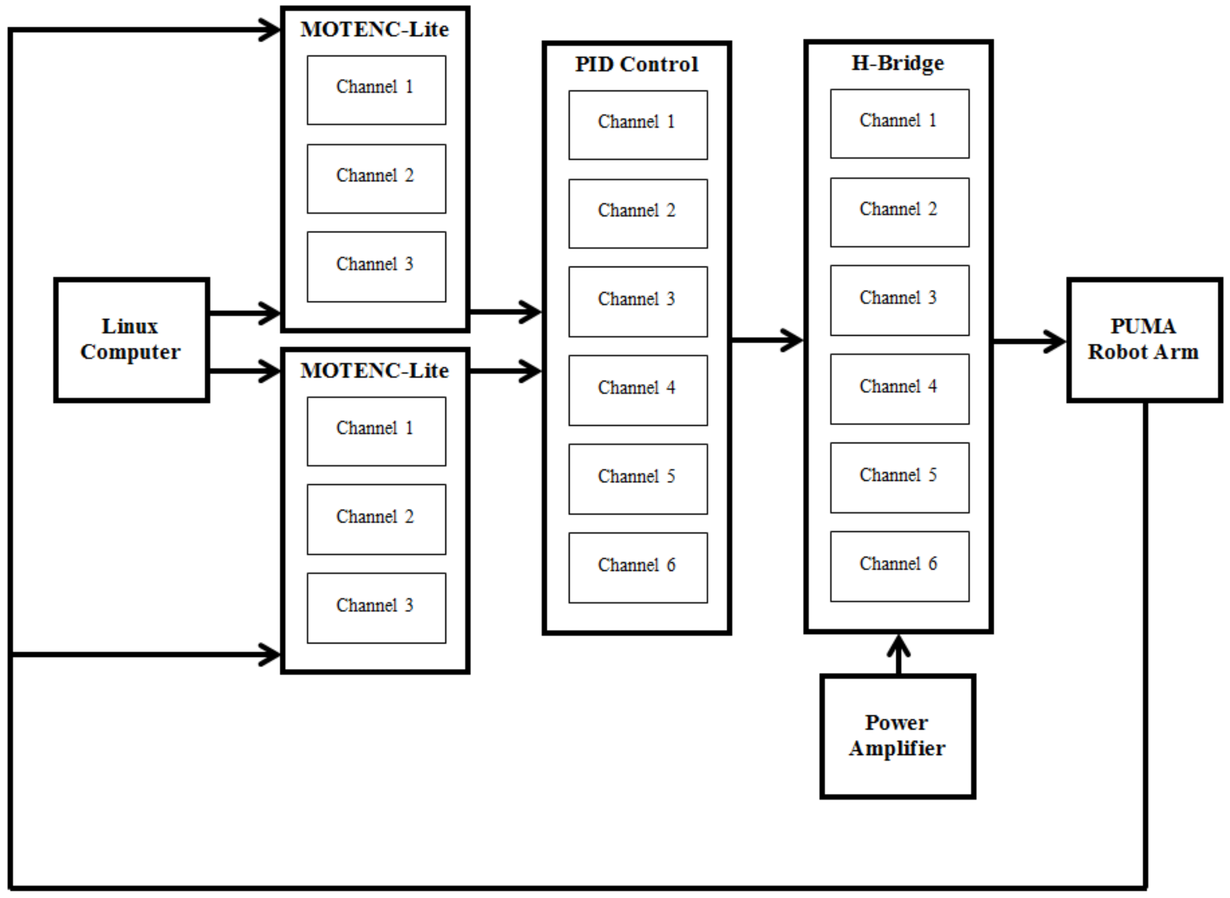
Problem Statement

Our client, Dr. Greg Luecke, has acquired two PUMA 500 robot arms, but the controllers have been damaged and/or lost. Our team objective is to develop a control system that will interact with the PUMA robot arm, which will replace the original controllers.

Concept Sketch



System Block Diagram



System Description

Linux Computer
The Linux Computer will have a C library that will allow for communication between the controller and custom application programs. The C library will serve as a basic API, allowing application programs to easily interact with the PUMA robot arm.

MOTENC-Lite Data Acquisition Board (DAQ)
The MOTENC-Lite boards will be connected to the Linux computer via PCI slots (one PCI slot per board.) The MOTENC-Lite boards will be accessed through C code and will output desired positions for each PUMA robot arm. Additionally, we will leverage the MOTENC-Lite boards quadrature encoder support to get the current position of each arm and output it for the PID controller. Note that a single MOTENC-Lite board only supports four degrees of freedom, while the PUMA robot has six degrees of freedom, hence the need for two boards.

PID Controller
There will be a PID Controller for each joint of the PUMA arm and they will be implemented digitally through the Linux computer and MOTENC-Lite boards. By taking in two input values from the DAQ associated with a desired and current angle value of each joint, it will output a given torque value. These circuits may differ based on the power demands or the degrees of freedom each joint is able to move. Also, within this block we will need to know the certain inertial values of each joint, so that we can allot the correct torque value to its given motor.

H-Bridges
The H-Bridge controls the direction of the DC motor. They switch current from the power amplifier in one of two directions at a time, depending on the input from the PID control circuit. We will also implement a logic circuit to avoid a short circuit if both sides of the H-bridge are being trigger by code bugs, machine errors, etc.

Power Amplifier
The power amplifier provides power to the six DC motors. Three joints only require 12 volts, and the other three require 40 volts. The three 40 volt motors also have brakes that must be unlocked with 24 volt input. Current requirements will be calculated based on the needs for each H-bridge and arm motor, and will be verified with measurements.

PUMA Robot Arm
The PUMA robot arm has six movable joints. Each joint has a DC motor and a quadrature encoder attached to its shaft. As the DC motor is energized, the encoder rotates, creating pulse trains. These pulse trains are used to indicate the current position, speed, and direction of rotation of a specific joint. Each joint on the arm has some inertial value associated with how hard it will be to move that given joint. These values will need to be taken into account when deciding how much torque a given joint will need to move it.

Operating Environment

Once operational, the PUMA robot arm will be used in the VRAC (Virtual Reality Applications Center) and the Laboratory for Advanced Robotics and Computer Control (LARRC) for research. We don’t anticipate any extreme conditions that will affect our design plan.

User Interface Description

The user interface will not have a GUI. Instead, we plan to provide a C library of functions to quickly build custom programs that control the PUMA robot for different research applications. The C library will be designed to be used in a Linux-based environment.

Functional Requirements

1. Six operational joints
    - All six joints of the PUMA robot arm will be operational, and their movement will be controlled by the user.

2. User interface through C code
    - The PUMA robot arm will be controlled by making specific C function calls. This will allow the user to write custom programs that control the PUMA robot arm.

3. Use H-Bridge design
    - The client, Dr. Luecke, already has an existing H-Bridge design that is very robust. He would like it to be refined and utilized in the controller. We will add a logic circuit to the existing H-bridge design that will prevent short circuits.

4. PID Controller
    - We will need to design a PID controller that is operational for six joints. We will implement a proportional response to the current angle and desired angle of each joint. If we wish to have a faster or better loop performance, we can include integral and/or derivative gains.

Non-Functional Requirements

1. Professional Quality
    - Our client would like the controller to look professional. Its circuits should be fabricated on PCBs, and the controller's inputs and outputs should be clearly labeled.

2. Ease of Use
    - The C library of functions will be easy to use, allowing for rapid development of custom applications for the PUMA robot arm.

3. Performance
    - There will be no noticeable lag from the time a command is given to when the PUMA robot arm moves. The output degree of the controller should be accurate with respect to the input command given for each joint.

Deliverables

Our client expects a controller for the PUMA robot arm, and a C library that will interface with the controller.

Work Breakdown

Alex will be handling the DAQ configuration and creating the C library to interface with the DAQ. Matt, Nhat, and Zeyu will work together to refine the H-Bridge design, PID controller, select a power amplifier, and map the PUMA robot arm wiring.

Resource Requirements

In order to accomplish this project, we will need the following resources:

1. A PCB fabrication company to fabricate our H-bridge circuit design onto a single PCB.
2. Electronic components to build an H-bridge prototype and other test circuits.
3. A Linux computer to test our C library and the control system.
4. MOTENC-Lite boards and corresponding software to develop and test the custom hardware design.
5. Once the PCBs are fabricated, we will need soldering equipment to assemble our circuit designs.
6. Electrical engineering test equipment such as a multi-meter, oscilloscope, function generator, and testing cables. This equipment is vital to ensure that our design meets the functional requirements.
7. Mechanical engineer(s) to build an enclosure for our control circuits. They also could perform maintenance on the PUMA robot arms.

Testing Plan

MOTENC-Lite DAQ
We will verify that the MOTENC-Lite boards are fully functional by utilizing test code provided by the manufacturer. The test code allows the encoder values to be read and reset, values to be written to the DAC, write to output registers, and read from input registers. This code will also serve as a starting point for building the C library.

C library
To test the C library, we will write small programs that call the C library functions we created. To verify that the library functions are operating correctly, we will use the gcc debugger to check proper variable values and function return values. If a library function call activates or

deactivates a pin or pins on the MOTENC-Lite boards, we will verify this behavior by connecting an oscilloscope to the affected MOTENC-Lite board pin(s).

PID Controller
The PID controller will be implemented in software. Thus, testing the PID controller will involve issuing movement commands to the PUMA robot arm and observing the output movement. Each of the PID controllers will have the same general form, but will differ given that each joint has a different inertia. We will tune each controller by changing the proportional, differential, or integral gain until the PUMA arm reaches the desired position with minimal overshoot.

H-Bridge
We will add a logic circuit that prevents both sides of the H-bridge from being trigger at the same time, which could potentially burn out the MOSFETS and short the PUMA robot arm motors. To test the logic circuit, we will apply different input and enable voltages, and verify that only one side of the H-bridge gets activated. To test the H-bridge circuit, we will trigger one side of the bridge and check for continuity, heat dissipation, and polarity. Once the entire H-bridge circuit successfully controls the motor's direction without any error, we will begin the next step of fabrication.

System Integration
Once all components are tested and are working individually, we can begin hooking components together. We will start with the MOTENC-Lite boards and C library and iteratively add a PID controller and verify that it can be controlled correctly by the C code and MOTENC-Lite boards. When all the PID controllers have been integrated with the C library and MOTENC-Lite boards, we will iteratively add an H-bridge and verify that it can be controlled correctly by the C code, MOTENC-Lite boards, and PID controllers. When all H-bridges have been successfully integrated, we can then connect our controller to the PUMA robot and verify that we can correctly control the robot through C code. We will need to make sure that the desired angle input to the C code results in the affected joint moving to the desired angle. If the output angle doesn't match the input angle, we will need to tune the PID controllers to match the two angles.

Project Schedule

	Spring 2014															
	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14	Week15	Week16
Map Pins	🟦 🟨	🟦 🟨	🟦 🟨	🟦 🟨	🟦 🟨 Alex, Nhat, and Matt	🟦 🟨	🟦 🟨	🟦 🟨	🟦 🟨							
H-Bridge Modification												🟦 🟨 Nhat	🟦	🟦		
H-Bridge Testing															🟦	🟦
MOTENC_Lite DAQ Design												🟦 🟨 Alex	🟦	🟦	🟦	🟦
MOTENC board testing															🟦	🟦
PID Controller										🟦 🟨 Matt and Zeyu	🟦 🟨	🟦 🟨	🟦	🟦	🟦	🟦
PID Testing Plan															🟦	🟦
	Fall 2014															
	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14	Week15	Week16
Spec Power Amplifier	🟦	🟦	🟦													
C-Library	🟦	🟦	🟦													
Implementation	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
Testing and Debug	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦			

Action Schedule	🟦
Action Completed	🟨