Jacob Harris Embedded Systems

3/3/2015

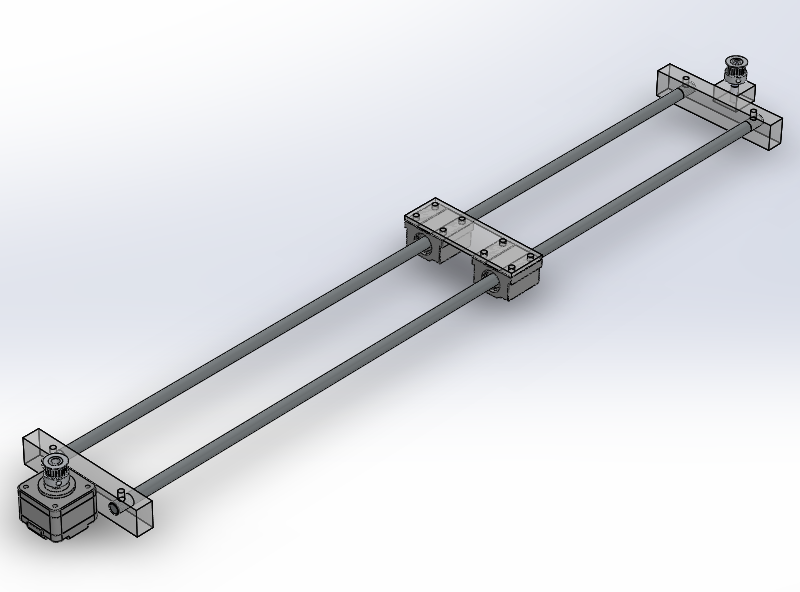
Project Proposal: Economical Linear Actuator with Wireless Networking

Linear actuators typically consist of some kind of cart or slide on a track or rail system, positioned by a belt or chain drive, or a linear motor. Both servo motors and stepper motors can be found in industrial units. Linear actuators find use in many commercial applications such as robotics equipment, in research applications such as fixturing, and consumer applications such as home-automation as well as hobby 3d printing and CNC.

The goal of this project is to design and implement a small consumer-grade linear actuator, which could find use in hobby or home-automation use, among others. Specifications of the prototype linear actuator are (subject to change due to availability of parts) as follows:

* Linear range of motion: at least one foot
* Load capacity: at least ½ pound at slow speed
* Linear speed (max): at least 4 inches/second
* Motor type: stepper motor, bipolar
* Rail system: 8 mm round rod with linear bearings
* Microcontroller: Ti MSP430 series
* Radio communication nRF24L01+ radio

Hardware implementation is relatively straightforward, with motor driving being handled by a Pololu stepper motor driver breakout, making use of the Allegro A4988 chip, which can current limit according to settings on the board. The driver chip accepts direction and step inputs and connects directly to the microcontroller and stepper motor. Early tests have proved effective, however, as is common with stepper motors, above a certain speed torque drops off rapidly and the motor stalls. Thus, it is necessary to accelerate and decelerate the motor at reasonable step rates (referred to as SPS for steps/second and SPS2 for steps/second^2). The stepper used will be sourced from Pololu, most likely a model 1208, or equivalent 200 step/rev motor. The linear actuator will operate on a network of radio modules, allowing it to receive position commands from any authorized node on the network. A nRF24L01+ module will be used to achieve wireless communication. A Solidworks model of the tentative mechanical design is shown below:



The software architecture is as follows:

|  |  |  |
| --- | --- | --- |
| **Module name** | **Description** | **Functions** |
| **Stepper Driving** | | |
| pulses | Interfaces with timer peripheral to generate pulse trains of varying frequency and length | void timerInit();  void pulse\_start();  void pulse\_stop();  void pulse\_setPPS(uint16\_t PPS);  void pulse\_update(void(\*fn)void); |
| stepper\_drive | Calculates shape of stepper velocity curve, sets appropriate speeds using the pulses module, keeps track of position | void stepperInit (uint16\_t max\_SPS, uint16\_t min\_SPS, uint16\_t SPS2);  void stepper\_start();  void stepper\_stop();  void stepper\_setSPS(uint16\_t SPS);  void stepper\_moveTo(uint16\_t position);  void stepper\_update();  uint16\_t stepper\_numSteps(uint16\_t SPS\_start, SPS\_finish) |
| Motion\_planner | Queues position commands. Can ignore points between two other points if desired. e.g. A🡪C not A🡪B🡪C | void motion\_planner\_Init(uint16\_t mode);  uint16\_t motion\_planner\_addToQueue(queue \*motion\_queue, uint16\_t position, uint16\_t time); |
| **Communication** | | |
| network | Deals with sending and receiving of messages, buffering said messages, and addressing | <to be implemented together in class, something like this>  void network\_setAddress(uint16\_t address);  uint16\_t network\_sendMessage(uint16\_t length, uint16\_t destination\_address, char\* data);  uint16\_t network\_receiveMessage(buffer\* rcvBuffer); |
| SPI | Initializes the hardware peripheral and gives send/receive functionality | void SPIInit(uint16\_t SPI\_module\_number);  uint16\_t SPI\_transmit(char\* data);  uint16\_t SPI\_receive(char\* data); |
| nRF24L01 | Implements send and receive as well as setup for channel selection and other options. Uses SPI | void radioInit();  void radio\_setChannels(uint16\_t send\_channel, uint16\_t receive\_channel);  void radio\_sendData(buffer\* TXBuf);  void radio\_receiveData(buffer\* RXBuf); |
| item\_buffer | Implements a circular buffer of arbitrary items | <as implemented in class> |
| **Organization** | | |
| task\_management | Implements a task scheduling system capable of running tasks at regular intervals | <as implemented in class> |
| timer | Keeps track of time since startup (in ms), handles rollover | void timer\_init(void);  void timer\_update(void);  time timer\_getCurrentTime(void);  time timer\_getTimeElapsed(time\* timeStamp);  void timer\_delayNon-Blocking(time\* delay); |

As the team consists only of one member, all responsibilities for each milestone fall on that team member. The Demonstration for April 14th milestone Beta is suggested to be a moving linear actuator, with acceleration and deceleration (potentially lacking radio communication), pending approval. The final presentation will include networking and receiving commands to move to given positions, and accurately prioritizing and executing those commands.