Elecanisms Miniproject 1
Haptic Control System

Ryan Eggert, Subhash Gubba, Amanda Sutherland
February 7, 2016

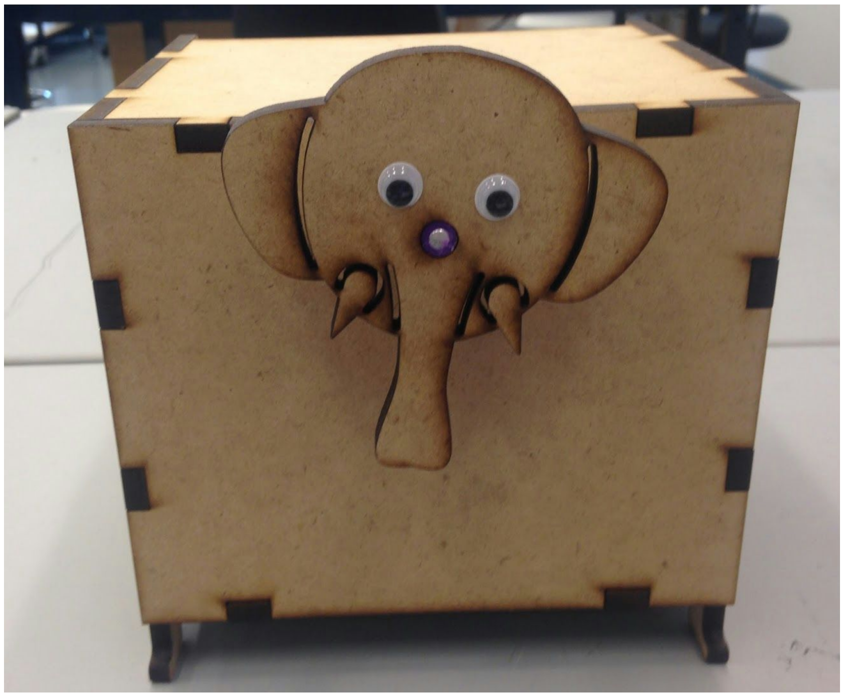


Table Of Contents

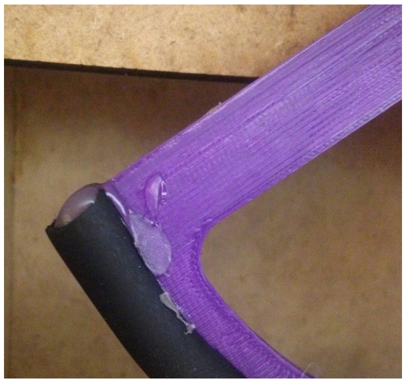
Hardware Summary

Electrical

Using the solder paste syringes and the reflow oven in the EE Prototyping room, we assembled a motor shield for the PIC, as well as a breakout board for the magnetic encoder. In addition to the surface mount soldering, we also soldered wire onto the leads of our DC motor and header pins for each component. Finally, we crimped wires for connecting the motor to the shield and the magnetic encoder to the same board as well.

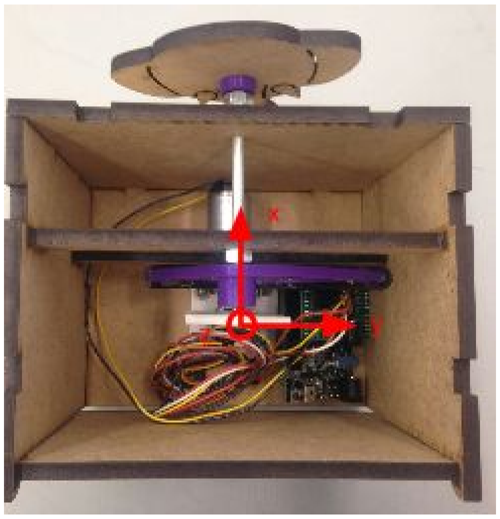
Our first angle-encoder board stopped working for yet-unknown reasons (we guess that something shorted) so we ended up making a second one. This one has continued to function properly.

Mechanical



The basis for the mechanical design was a box which could enclose all the components and also ensure they are still accessible. Also, we kept the bare essentials of the Haptic mechanical system. For testing purpose, we hot glued a nut onto the shaft of the motor, placed the magnet on it, and fit placed it in front of the magnetic encoder. This permitted testing of the electrical and software components while the mechanical aspects were assembled.

The mechanical system consists of a hand driven front wheel with a shaft into the box which is attached to a rocker, as per the hapkit system, which rotates drive wheel on the shaft of the motor. The rocker is a 3D printed piece which has a neoprene sleeve on the edge to give it greater friction, as in the figure at right. Increased friction keeps the rocker from slipping as it turns the drive wheel on the motor.



As the interaction between the rocker and the drive wheel, and between the magnet and the magnetic encoder have necessarily close tolerances. The rocker must continually be in contact with the drive wheel, thus the motor was mounted with bolts in slots so it's distance to the rocker could be slightly altered. The magnetic encoder was similarly mounted; it needs to be within a few millimeters from the magnet and therefore the mount itself could slide in the x and y direction, as in the figure below. The encoder was also mounted with slots and could move in the z direction. This allowed us to get elements in approximate position and then hand calibrate them. One issue with the rocker is that attaching the neoprene sleeve with hot glue created lumps along the curve, which gives non-continuous pressure contact on the drive wheel. This will be fixed by changing the attachment method of the neoprene sleeve - possibly with pins through the 3D printed part and sleeve.

Coding Summary

Firmware

Using the provided libraries, we wrote scripts to communicate with the PIC over SPI and spin the motor at variable rates. We initially chose a value of 24KHz for the SPI clock. We needed to increase this frequency to 9.5 MHz. in order to gather data faster for the spindown test.

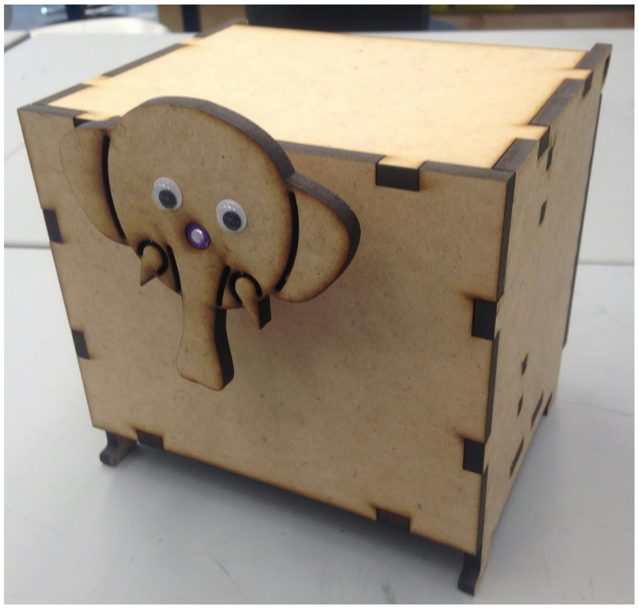
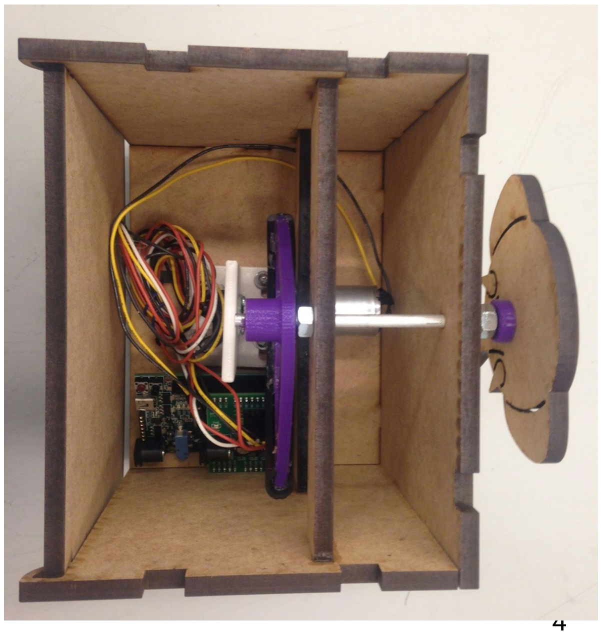
We chose our PWM variables like duty cycle and frequency based on trial and error. We found that a lower PWM frequency (the system's minimum appears to be approximately 244.7 Hz.) allowed us to drive our motor at slower rotational speeds (smaller duty cycles). Higher frequencies resulted in smoother, quieter motor operation. A PWM frequency of tens of kilohertz was used in the spindown test [see "Spindown Test" section, below].

The motor is very controllable from the C code running on the PIC. We maintain two PWM outputs on the motor's two controller input pins. These are used by a collection of functions which allow setting the motor's direction as well as it's speed. These functions currently assume operation in the motor driver's fast-decay operational mode. If it becomes necessary, these functions could be extended to the mixed-decay and brake modes.

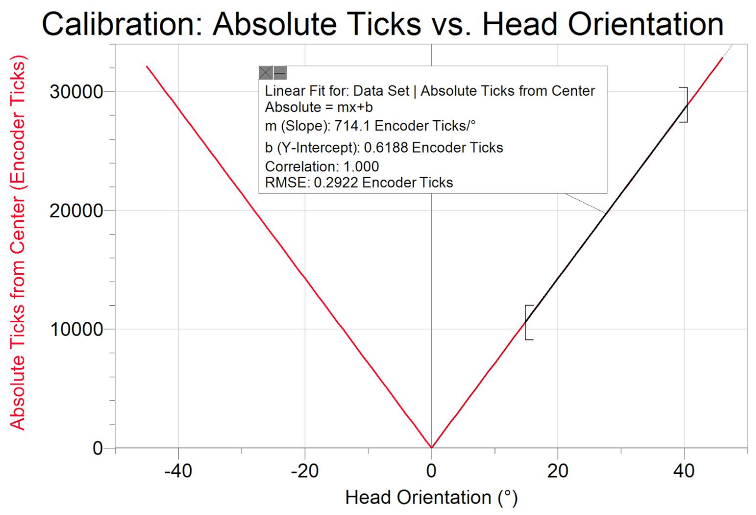
Software:

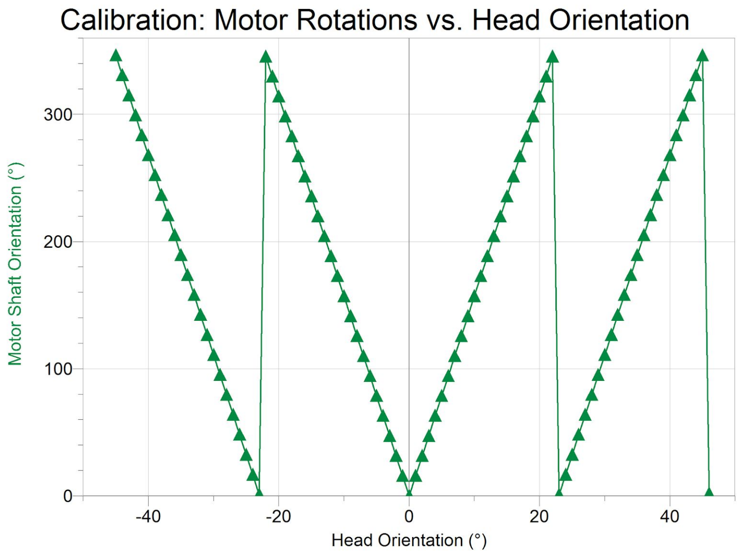
The PIC could communicate with a PC via UART and USB. A Python script running on a PC could request a read of the angle sensor and receive the raw, 16-bit response via USB. The Python script could parse this into an encoder reading, convert the encoder reading to motor shaft rotation (degrees), and store a timeseries of angle data to a csv. This CSV data could be plotted by another Python script, or most graphing tools (e.g., LoggerPro).

Integrated System

Calibration Curves

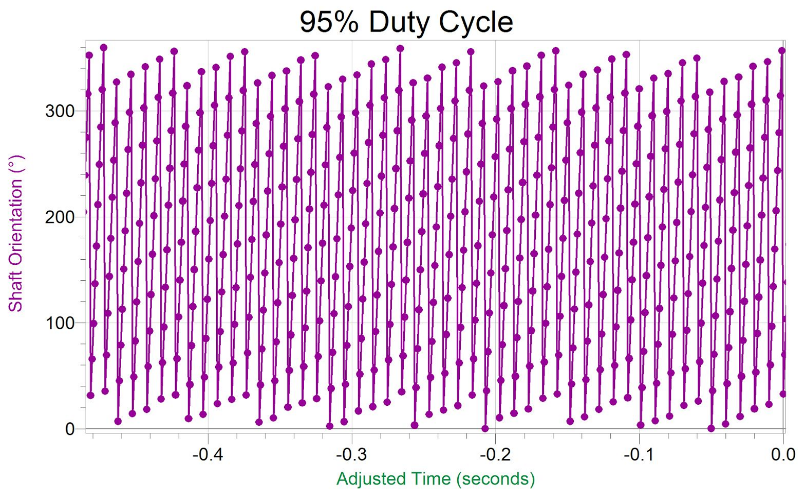




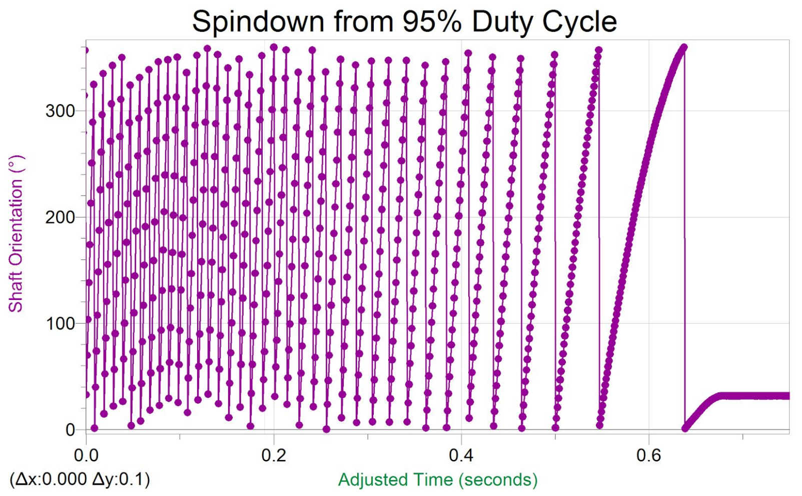
As the angle sensor was brought online, we performed two calibrations. We first related motor shaft angle to encoder ticks. A series of four manual measurements resulted in finding that approximately 45.5 encoder ticks correspond to a one degree rotation of the motor’s shaft.

We then extended these calculations to the rotating head of our elephant. A series of five encoder measurements showed that the elephant’s head could rotate just under 45 degrees from center in either direction. Rotating the head from center to either extreme took, on average, 32137 ticks (714.15 encoder ticks per degree of elephant head rotation). Some of these relations are graphically depicted in the two *Calibration:* graphs above.

Spindown Test



The first graph (above) shows our measurement of a steady 95% duty cycle (35kHz. PWM frequency). Data points were collected in 1ms intervals. Shortly after the 0 second mark (seen in the second *Spindown* figure), the motor's power was removed, allowing the shaft with mounted magnet to gradually spin down. Notice in the second graph how the distance between peaks increases, illustrating the motor's slowdown.



Github Repository

The code used in this project may be found at the following link--
https://github.com/gubbatuba/elecanisms/tree/master/haptic