BreakfastBot

David Box

April 24, 2012

University of Florida
Department of Electrical and Computer Engineering
EEL 5666C – IMDL – Final Written Report
Instructors: A. Antonio Arroyo, Eric M. Schwartz
TAs: Ryan Stevens, Josh Weaver, and Tim Martin

Table of Contents

Abstract

Given the pace of modern day lives many people leave for work or school without having a healthy breakfast. Research by health professionals commonly shows that having a breakfast is the key for an individual to unlock his or her peak performance and efficiency. Many people choose to maximize their sleep and therefore have very little time in the morning to cook. This paper describes the BreakfastBot which automates your breakfast cooking to save a person precious time in the morning. This robot will cook one of the best breakfast options there is: eggs.

Executive Summary

BreakfastBot is a robot designed to autonomously cook any user some delicious eggs. In order to accomplish this task on a college student's budget this robot has a novel architecture and platform. This robot is easily broken up into three major components: the mechanical, electrical, and computer components. This section will detail a summary of the most salient aspects of each of these components.

The mechanical component of the system is responsible for providing the structure for the 1'x1'x1' of workable area. In order to accomplish this task this system was made out of the study but light 1/8" aluminum. Straight pieces and L-beams were used to provide most of the structure with a piece of square tubing with a slot providing the moving along the main frame's length. Bearings were moved along a track powdered with graphite for one direction of movement. A bolt was moved along the slot underneath the square tubing for another direction of movement. Finally, the third axis of movement was directly mounted to the aforementioned bolt. In addition to the pieces of aluminum, a lot of wood was used for height adjustments for the system.

The main electrical components consist of pressure sensors, a touchscreen, linear actuators, servos, optical distance sensors, many amplifiers and drivers to make everything work. In order to provide enough power for all of these components a large 12V 7A power supply is utilized for the major components like the actuators. Additionally, 5V and 3.3V supplies are utilized. LM741 op amps are used to boost the output for analog signals like from the pressure sensors. Several mechanical and solid-state relays are used to provide precise and quick control of major components.

In order to provide the logic for the entire system two 32-bit ARM microcontrollers were used. The mbed NXP LPC1768 microcontrollers were preferred for this project for their high clock rate of 96MHz. This clock rate is useful to control this system because it requires a lot of temporal precision. In addition, I've wanted to become familiar with the ARM architecture and libraries for some time. Two mbeds were necessary because of the lack of peripheral ports, namely the serial and analogIn (which contain an ADC) ports. An arm-gcc compiler was used to program the controllers over an USB interface.

Introduction

Many people know from personal experience that their performance throughout the day suffers if you don't eat breakfast [1]. Yet, many people's modern lives cause them to habitually miss breakfast. I am proposing a robotic solution to this problem that will minimize the human involvement in the making of eggs. Eggs are considered one of if not the best breakfast foods [2].

Once the eggs are loaded into the robot and the options are selected on a touchscreen the robot will kick into action producing the amount of egg specified, scooped onto a plate for easy consumption. In a practical application, I could see a person just waking up, loading the eggs, then getting dressed and ready (possibly taking a shower), and then walking back to the robot with the eggs already prepared. The main overall objective is to be able to make high-quality eggs autonomously (by high-quality I mean without mishaps like having egg shells in the final dish served). A secondary objective is to speed up the egg making process as much as possible to reduce the time that would have to be spent waiting for the eggs.

In this paper I will be describing the details of operation of the BreakfastBot. Each subsequent section will focus on one of the major aspects of developing a robot. In particular I will describe the integrated system, platform, actuation, sensors, and behaviors of the robot followed by the results and conclusion.

Integrated System

Everything starts with the user loading the eggs into the pan attached to the robot and the user making the selections on a touchscreen. The options I will offer will be how many eggs to scramble and when to start. Once these steps have been fulfilled and the user presses the on-screen button to start the robot, the robot will begin the cooking process. Output is displayed real-time to the LCD touchscreen to notify the user of the robot's progress.

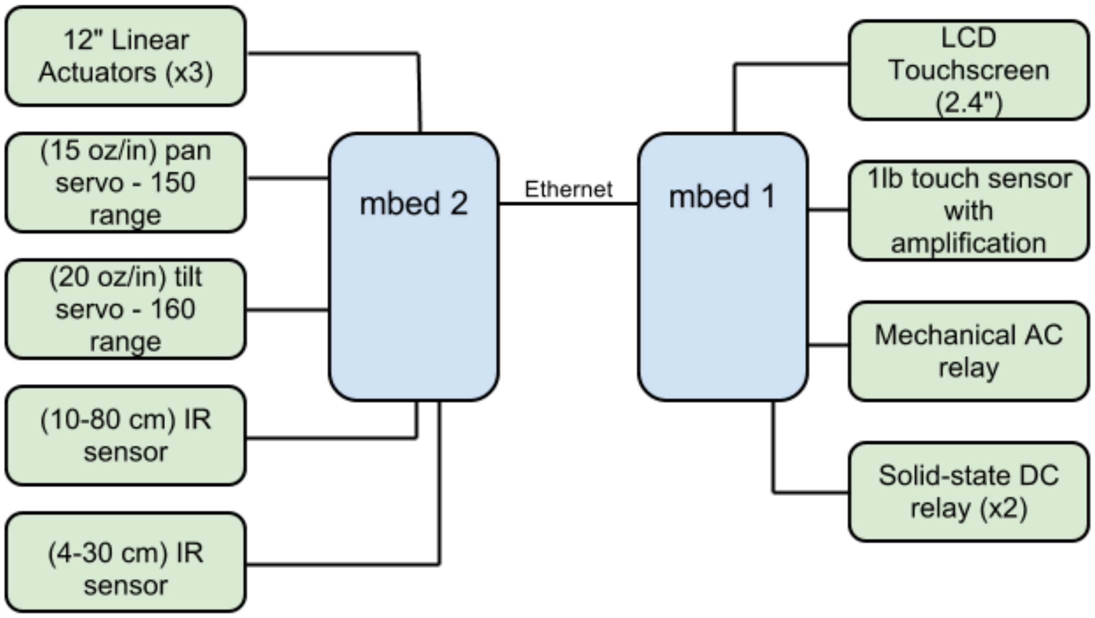
12" Linear Actuators (x3)
(15 oz/in) pan servo - 150 range
(20 oz/in) tilt servo - 160 range
(10-80 cm) IR sensor
(4-30 cm) IR sensor
mbed 2
Ethernet
mbed 1
LCD Touchscreen (2.4")
1lb touch sensor with amplification
Mechanical AC relay
Solid-state DC relay (x2)

Figure 1: High-level Diagram of System Components

The system is extremely sensitive to precision both mechanical and temporal. For this reason the system platform has to be put together very carefully and needs to be calibrated so that everything is level. The position of the skillet can be moved in the x-direction (length-wise) with the addition of IR sensors but needs to be placed in the "correct" place with regard to it's angle and y-direction. The IR sensors for the skillet and plate detect how far aware each component is from the stands the main frame rests on. With this information the system can cook the eggs. Figure 1 shows a high-level diagram of the components involved.

Once the system has been powered on and the power switches are in the "on" position, the system is waiting for the user to supply the egg batter and make the corresponding options on the LCD touchscreen. Once the selections have been made the system follows a routine like outlined in Figure 2. The initialization involves setting the right position and actuation values so that every subsequent procedure meets the accuracy requirements. In addition, the skillet is turned on via an AC relay and the eggs begin to cook.

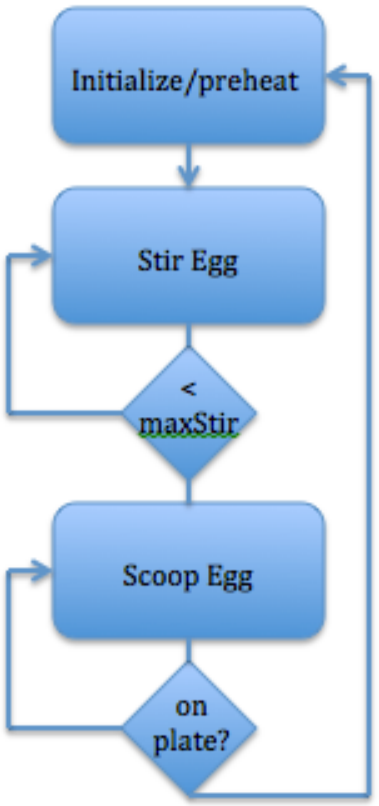

Figure 2: High-Level Flow Diagram of Cooking Eggs with BreakfastBot

Once the eggs have cooked enough that they need to be stirred (which depends on the amount of eggs but into the skillet), the stir routine is begun. The optical infrared sensor locates the skillet with relation to the stand and the linear actuators are put to work. The egg contents are stirred four times spaced out between a little over two minutes. The egg contents will eventually thicken and become ready to scoop onto a plate. A pressure sensor determines how many times the robot needs to attempt to scoop based on if it senses the egg contents' weight.

The main objective of preparing eggs and delivering them to the user on a plate is completed with the weight of the cooked eggs sensed. Originally the system was going to crack and stir eggs but this proved to be infeasible given the components I was able to buy. Many additional substructures and subsystems were developed that didn't make it into the final system. Although the final system went through many revisions throughout the semester as challenges arouse I am very happy with this objective completed.

Platform

The mechanical component is quite unusual for a robot. In order to save on costs linear actuators were used for linear motion that necessitated a large platform. This proved to be a daunting but exciting task for a computer engineering major (like myself) to undertake. Ninety percent of the platform was made out of aluminum to save on weight but to provide sturdiness that wood couldn't provide. The main frame for the platform houses the linear actuators and is the most mechanically complex part of the project. Four perforated angle iron beams form the basis of the design. Several solid pieces of 1/8" angle iron connect the frame together and are doubled up in several places to provide extra sturdiness. A square piece of tubing provides is used as a push rod for one of the linear actuators that is mounted in the center of the platform. The square tubing is moved by free spinning bearings along a track made with metal dowels and powdered with graphite. Underneath this push rod, a slot was drilled lengthwise so that another linear actuator also mounted underneath can linearly push a bolt.

Finally, the third linear actuator is mounted to this bolt so that it can provide vertical actuation. All together this frame is 4'Lx3'Wx30"H.

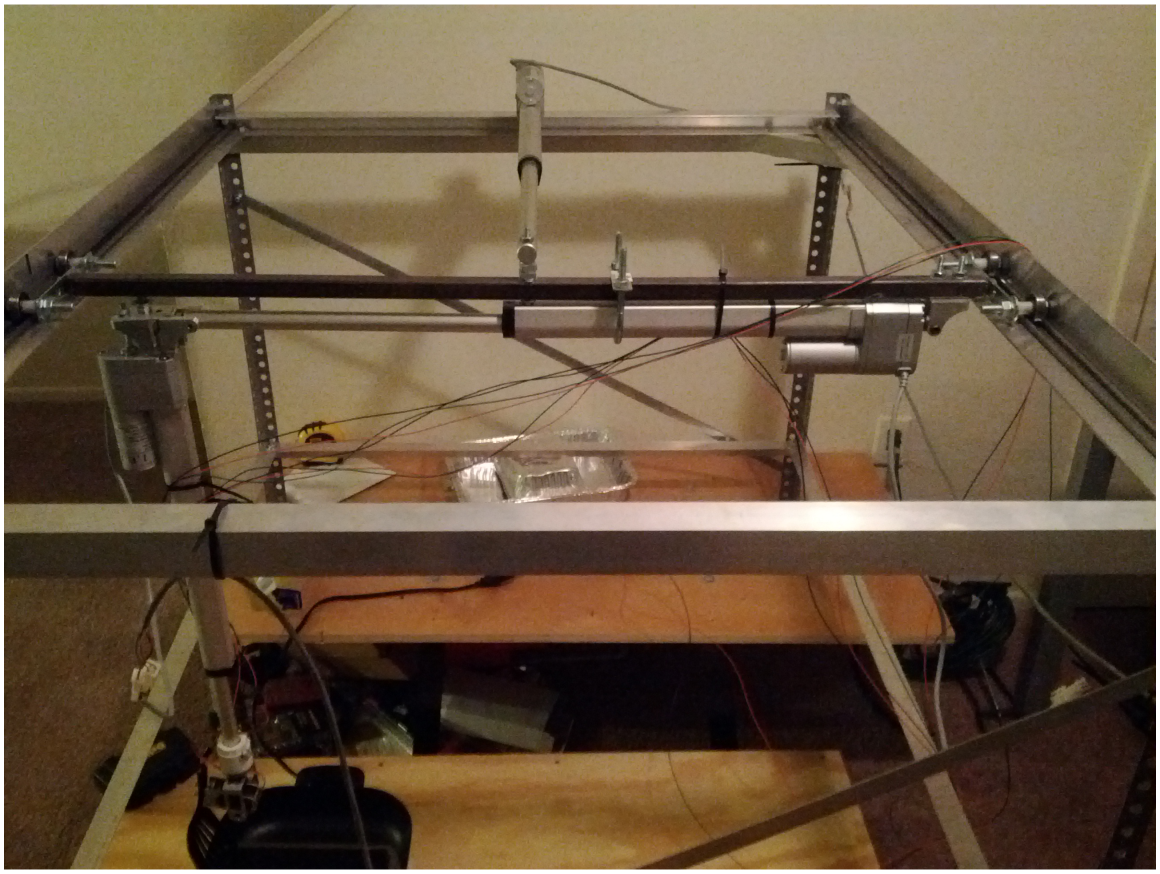
Figure 3: Platform from Front View

A pan/tilt system is mounted to the vertical linear actuator to provide ample motion for a spatula mounted underneath. To directly mount the spatula to the tilt bracket, two collars where attached horizontally to mount the spatula flush to the ground. Throughout the semester the overall height of the system was adjusted as the prototyping necessitated. In order to accommodate doorways and the height demands, the main frame is set on top of two metal stands with a 5/8" wooden platform on top. Additionally, another entirely wooden 4'L x 2'W x 5 3/4"H frame was constructed to act as a standoff for the skillet from the ground. Clamps, U-Bolts, and brackets where used extensively to mount and level-off actuators and sensors. The entire system's success relies heavily on precision so a level was used extensively to make the mechanical aspects of the system adequate.

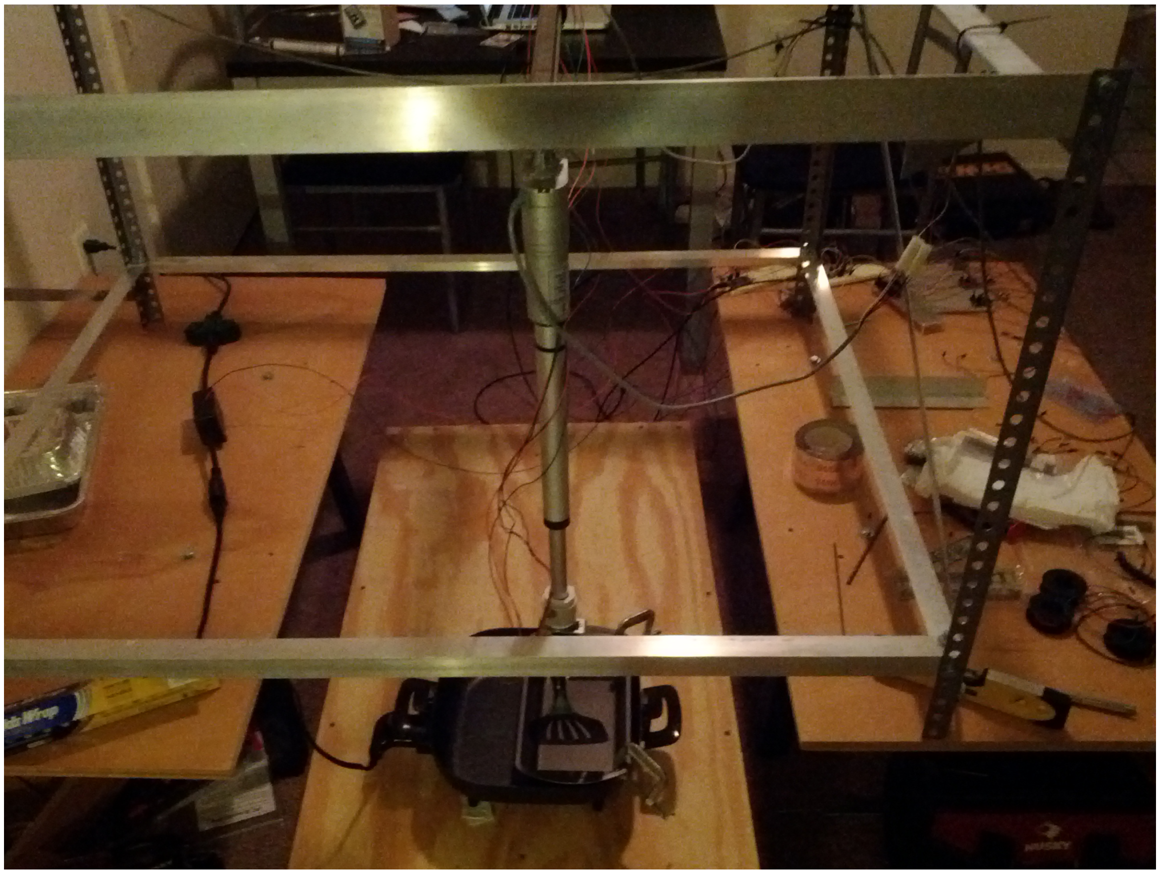
Figure 4: Platform Side View

This project ended up being a lot more difficult mechanically than I originally realized. Apart of the reason why was the need to go with the linear actuators instead of a CNC Router-like design because of cost constraints. A lot of the mechanical difficulties encountered were because the precision needed and my limited knowledge of mechanical components. Fortunately I learned a lot quick and know much more about mechanical parts, tools, and methods. I have also learned my way around a hardware store and many store employees have gotten to know me.

One of the most challenging and embarrassing lessons learned in developing the platform was getting the push rod to work well. I attached one linear actuator in the middle and attached two bearings to each end but this didn't work nearly as well as I initially thought it would. I spent a lot of time tightening/loosening the tracks, making things more level, and considered some drastic measures to get the actuator mounted. In the end I just needed to remember some basics about torque and friction. Simply separating the bearings and putting graphite along the track worked like a charm.

Actuation

In order to achieve the task of cooking an egg it was determined that this system would need a large "workable" area. The robot would then need to be able to manipulate objects within this area. Three linear actuators each with 12" strokes [3] [4] were attached to the metal platform to give this robot a cubic foot of workspace. Attached to the end of the vertical linear actuator is a pan and tilt system powered by small, lightweight servomotors. This pan and tilt system theoretically allows up to 180 degrees of rotation each although in real world tests the max rotation was around 150 degrees.

The linear actuators were by far the most crucial and most frustrating part of the design. The linear actuators were much cheaper than the alternatives considered for the main movement however this robot uses them in a way that they weren't originally designed for. Each linear actuator has a potentiometer for feedback and is powered and controlled by a driver that contains an 8-bit PIC running a PID controller. Originally when the linear actuators were bought I tuned and scaled the feedback so that the performance and precision was within the systems requirements. However, as the semester wore on the performance of the linear actuators gradually degraded and the PID controller and feedback scale needed to be continuously tweaked. In addition, the drivers for the actuators started to show anomalous behavior. A lot of debugging went into continuously tweaking the linear actuators and debugging the erratic behavior. Some of the issues were solved by continuously empting the buffers (although with simple serial communication this shouldn't have been the case). In the final implementation the linear actuators are very conservatively initialized and run.

Servomotors [5] [6] are used extensively in the stirring and scooping of the eggs as they provide the angular actuation. A pan and tilt system powered by the servomotors has been tested for about 150° total rotation. At 15 oz/in pan and 20 oz/in tilt this system provides more than enough torque to accomplish its objectives. Figure 5 shows the pan and tilt system being put to use scooping the egg contents onto a plate.

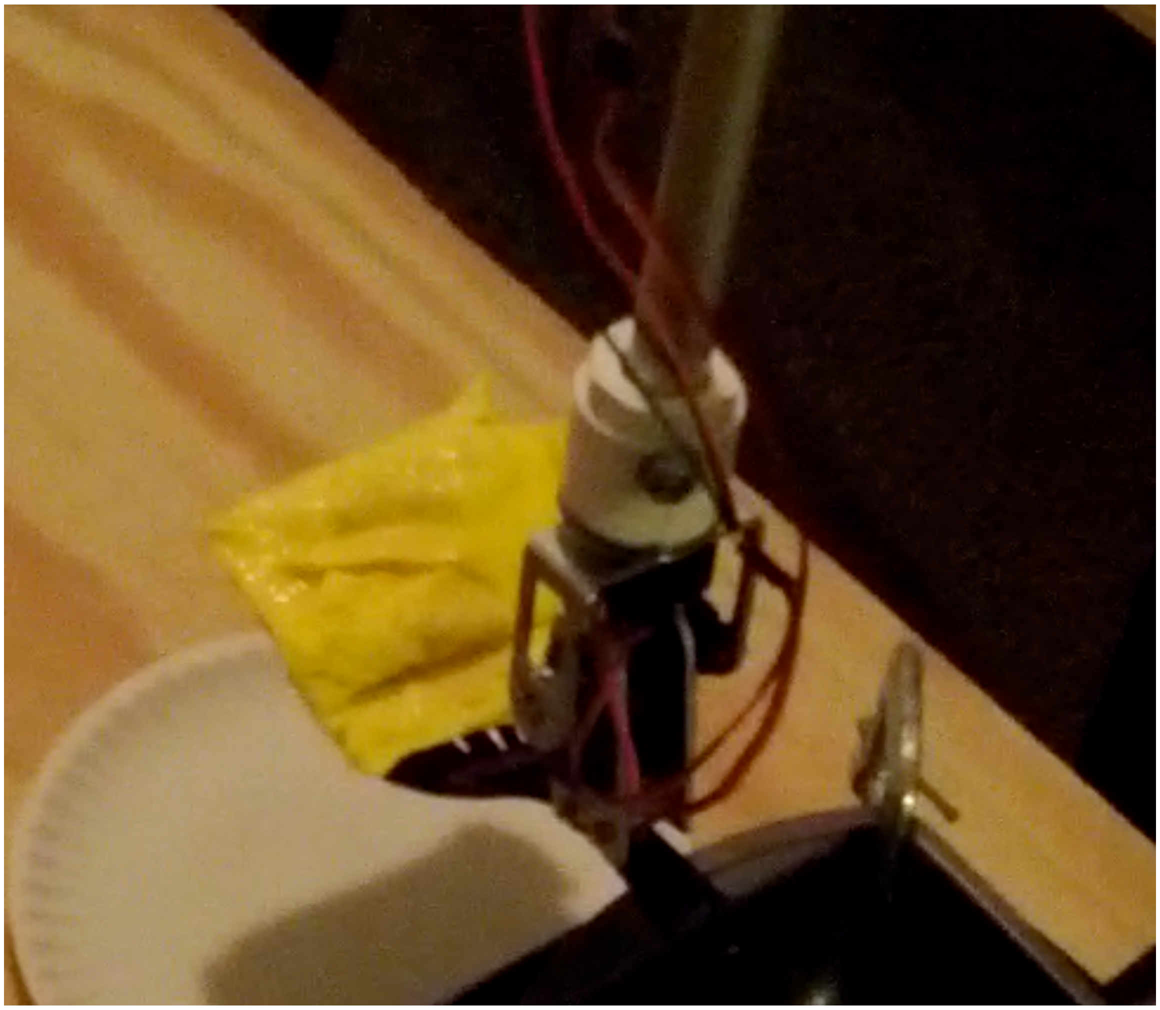
Figure 5: Scooping the Eggs onto a Plate

Initially this project also incorporated linear solenoids with 1" strokes to crack eggs. The solenoids had a spring attached to them to make them into the "pull" type (Figure 6). The ends of the strokes had very sharp blades to crack eggs. Also, a sevo-powered claw was used to manipulate the workspace but with a redesign of the egg cooking process a spatula directly mounted to the pan/tilt bracket was found to be optimal. Finally, a servomotor was attached to a cup holder to pour the egg batter for the robot (Figure 7). While this worked really well, the final design didn't call for this to be necessary as the egg batter can be directly poured into the pan. In the future, this system can be used when you want to cook more than just eggs.

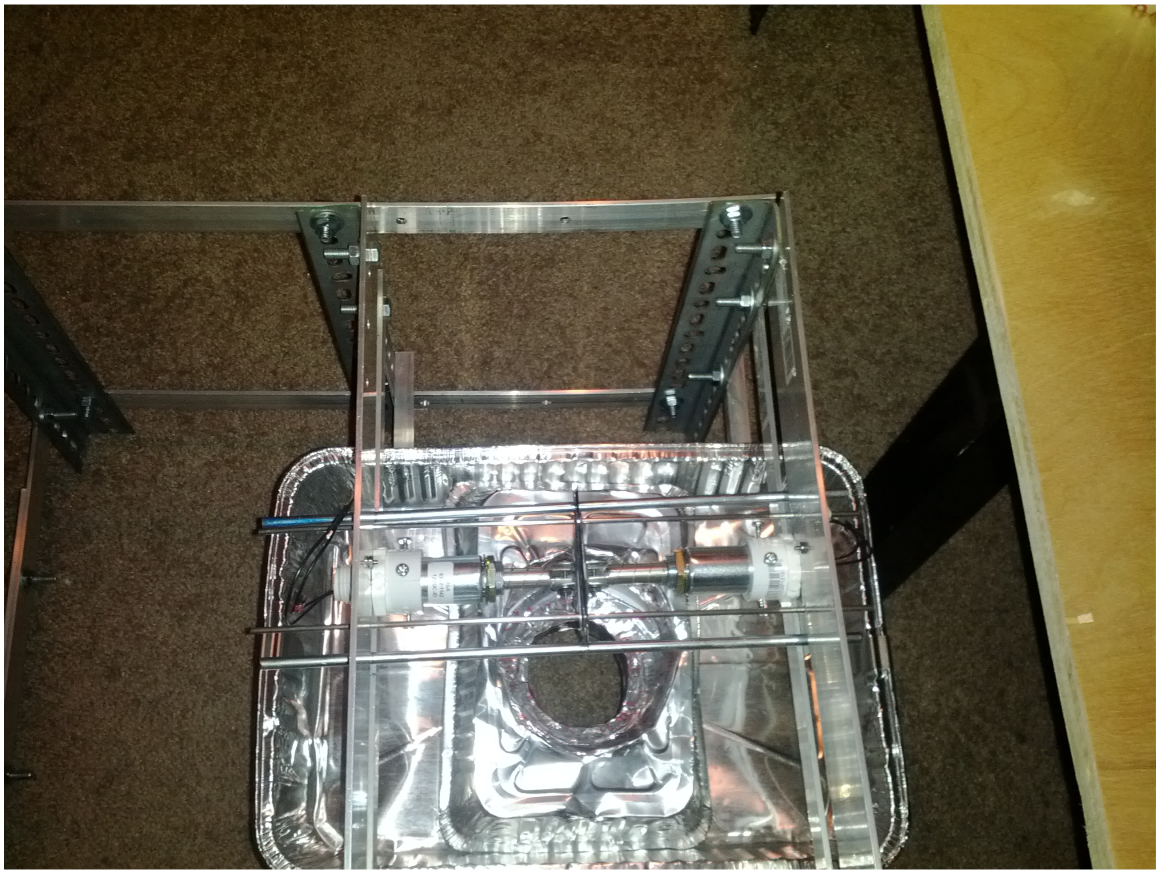
Figure 6: Egg Cracking Subsystem with Linear Solenoids

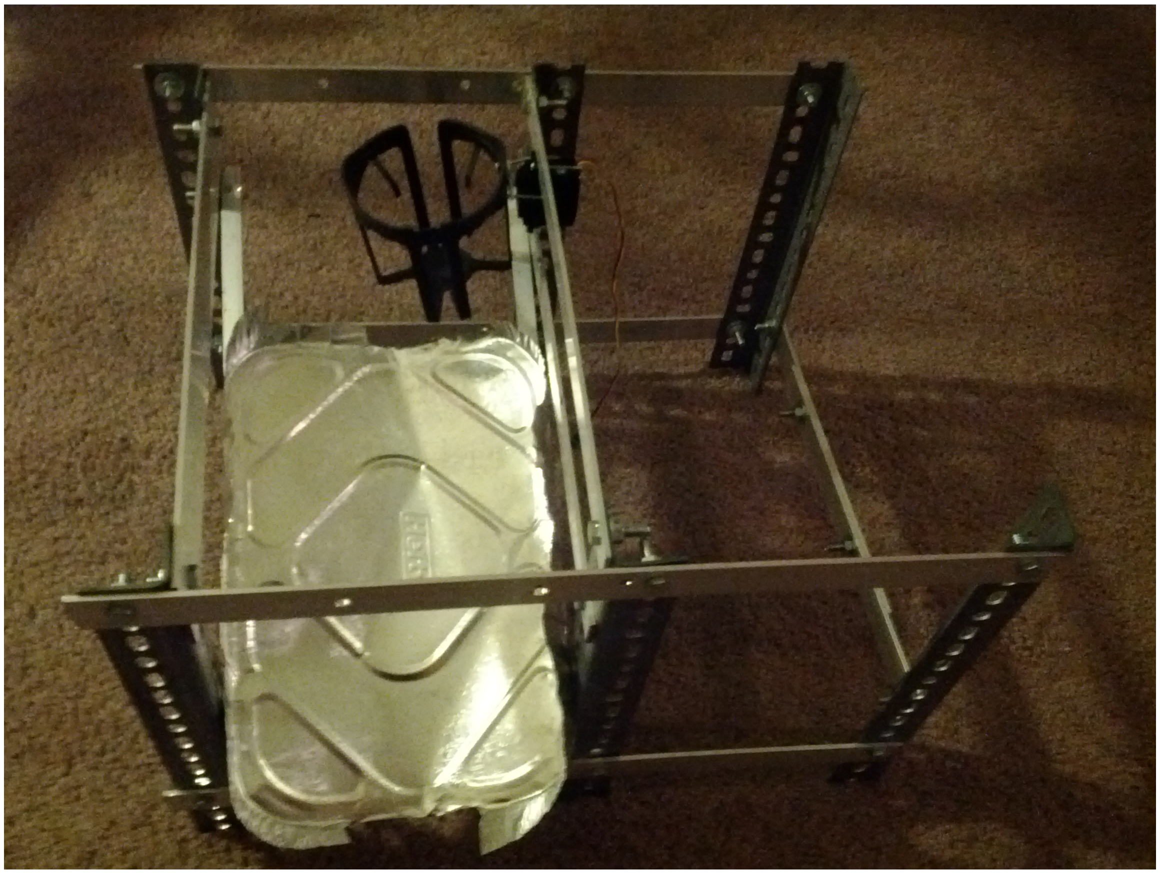
Figure 7: Egg Pouring Station with High Torque Servomotor

Sensors

A touchscreen [7] will be used to allow the user to make the different selections for the robot. This screen will also be used to give feedback to the user on which stage the robot is currently undergoing and to notify the user that the task has been completed. In order to control this 2.4" LCD touchscreen from Vizic Technologies, several images are preloaded on to a SDHC memory card for recall. The LCD is continually updated via a serial interface. The feedback from the capacitive screen is continually read and responded to.

IR distance sensors [8] are used to know the distance from the skillet and plate to the edge of wooden platform it rests on (Figure 8). This information is used to know where to stir, scoop, and deposit the eggs. In order to get the most accurate reading as possible I selected the IR sensors so that they had a big voltage drop-off between the 4" to 16" distance that I would need to be measuring in. Through testing I found that taking the median instead of the mean of 21 values gives the most reliable result. Since I can store up to 64-bit integer number I could keep taking more samples for the median but this becomes computationally too complex since I must then sort the number (selection sort). In order to get the reading I'm using the onboard 12-bit ADC that refreshes 200 KHz which is more than enough for this sensor. However, since this system requires ¼" precision in order to properly work this system's error combined with the error from the linear actuators feedback cause the system to become unstable. For more information on this problem see the results section.

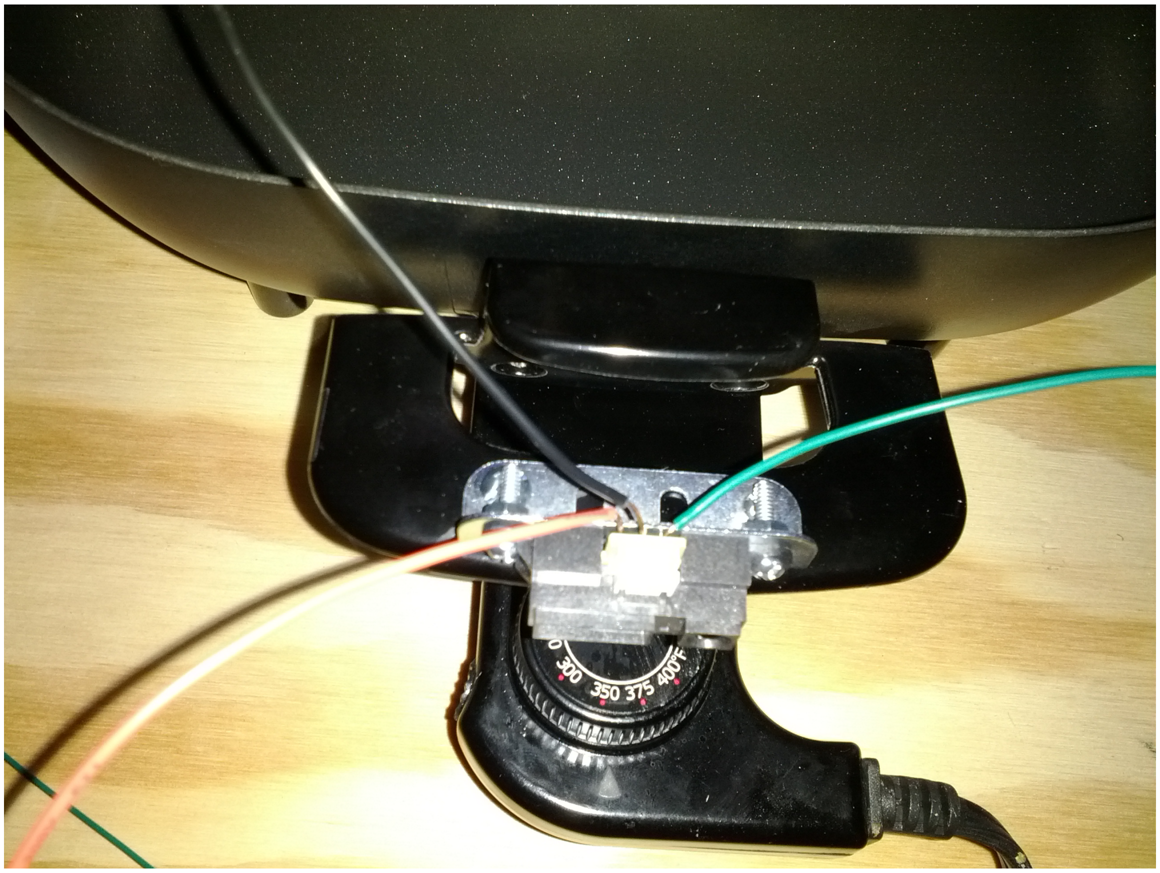

Figure 8: IR Sensor attached to handle of Electric Skillet

My special sensors are my pressure sensors that detect where the eggs have been placed and are used to know if the robot has deposited the eggs on the plate. If the scooping process fails the procedure is repeated until the eggs are properly dropped onto the plate. Since the eggs have little weight, I went with the Interlink Electronics FSR 406 [9] pressure sensors. These sensors are designed to work with forces as little as .1N and up to 10N (roughly 0 to 16 oz). At 43.69mm of area the pads are also small which is beneficial since the eggs are small. The sensor has two leads change resistance based on how much pressure is applied. Just directly reading back the voltage on my analog ports on my microprocessor I can read high-pressure changes but miss the small changes in pressure. In order to get a better reading I am using a Microchip Technology MCP6004 Op Amp to boost the Voltage reading into my microprocessors analog port's dynamic range.

Behaviors

Much of the behavior for this robot is predicated on the accurate physical and temporal precision. Therefore a lot of the effort for this robot was getting the accuracy I needed. Given the components that I had a budget for this robot is very integrated as opposed to modular. If a small modification is done to one component the consequences often affect every other component. One possible solution to this challenge is to add sensors that will detect position and pressure against the main components. One of the original attempts for this was to attach tiny pressure sensors [9] to the "pads" at the end of the claw. However, the error from the sensor was amplified from the drastic changes in heat due to the electric skillet. The error from the sensor coupled with the error from the linear actuators proved to cause too much imprecision to rely on this method. The final nail in the coffin was when the pressure sensor actually melted to the claw due from it accidently resting on the skillet during operation. Another attempt has been to incorporate IR sensors [8] but since the linear actuators are already on the upper limit of the amount of precision needed the additional error from the sensors seems to still cause the procedure to fail.

The solution to these problems has been to tweak the PID controller so that the settings are very conservative. In addition, I have relaxed my linear actuator positioning routines so that they make sure that the feedback is within the ¼" needed precision range. This causes the robot to move slowly but this ensures acceptable behavior. This robot also requires precise timing which is carried out using a Timer that runs in it's own thread.

Experimental Layout and Results

The main objective of this project has been to be able to reliably put some cooked eggs on a plate. The final routine has been run nine times now and only failed to place the eggs on the plate once (represented in Figure 9). Given the challenge-level of this project I am happy with this result.

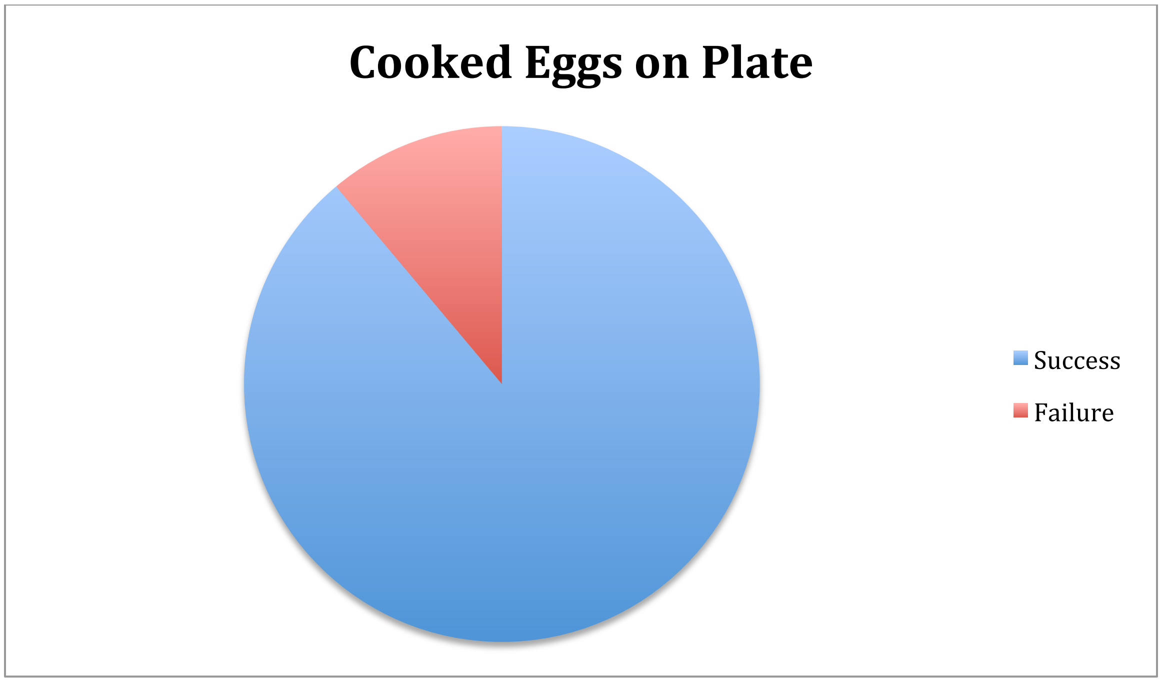

Figure 9: Success Rate of Cooked Eggs on a Plate from Testing

While the overall success rate is really the only results that ultimately matter there are several other results that have a huge impact on that success. The metric with the biggest impact on overall success is whether the linear actuators actually meet their target. The linear actuators started off being pretty good about getting close (determined by the feedback) to what their target was any given time. Despite the PID controller, as time went on the performance degraded. Unfortunately I didn't keep any metrics on the target error when I first got the linear actuators but I did collect this information for the y-direction linear actuator (the one used most often). Figure 10 shows what the error is from the target to the final stop's scaled feedback moving 6" both ways a total of 20 times. These results show that with the final tweaks to the PID controller I am getting acceptable or better results over 75% of the time.

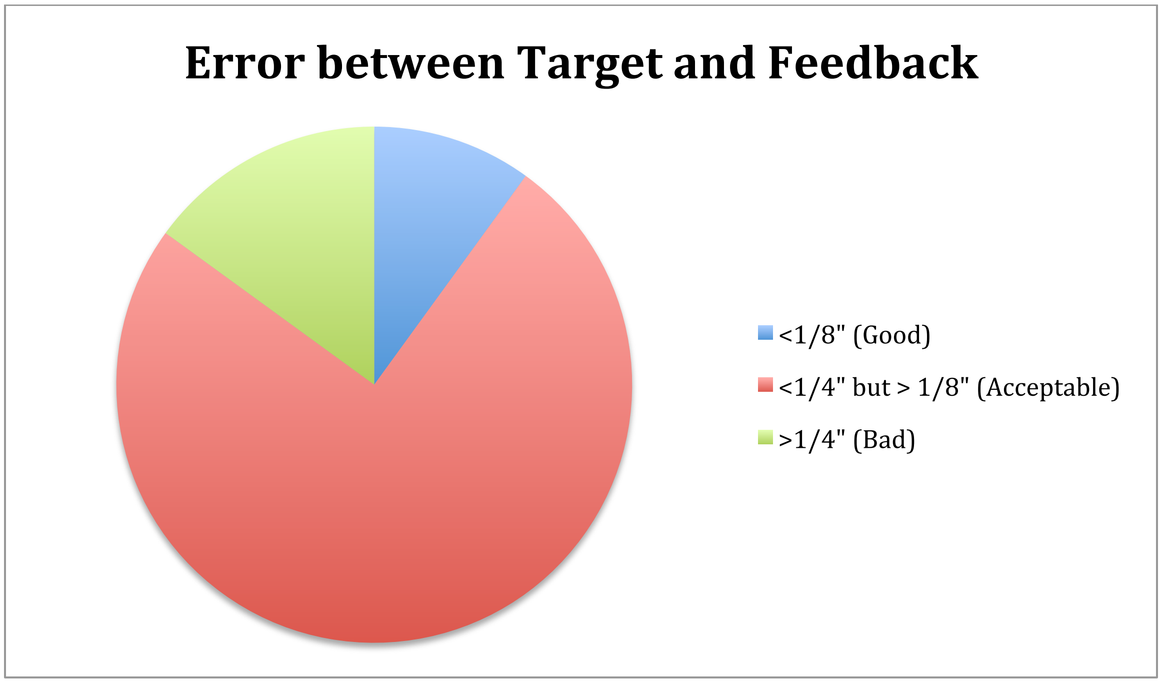

Figure10: Error between Target and Feedback (Good – 2, Acceptable – 15, Bad – 3)

Another source of error that has to be taken into account is in the distance measurement from the IR sensors. Since the linear actuators are very close to the maximum error threshold (and in a few cases already over), the IR sensors didn't have much room for error. For this test I moved took several samples with each sample consisting of the median of 21 measurements with 200ms spaced between each measurement. For my experiment I look at the error in the range from 10 to 20 cm (4-8inch). The ADC I'm using is sampling with 12-bits so with this voltage drop from Figure 10 (2.4-1.4V) I should see the hex range of 0xBA2-0x6C9. This gives me a range of about 10 bits, which is pretty good. Unfortunately in practice I was only getting 6-8 bits of resolution depending on where the sensor was in relation to the curve in Figure 11. Practicing with this resolution didn't seem to allow the stir and scoop parts of the routine ample enough accuracy so that they didn't hit the clamps for the skillet or miss the pan entirely. I really don't think that this is the IR sensor fault at all. I think the main problem is that the linear actuators are already on the edge of being too inaccurate to work and adding any source of error, no matter how small, is causing failure.

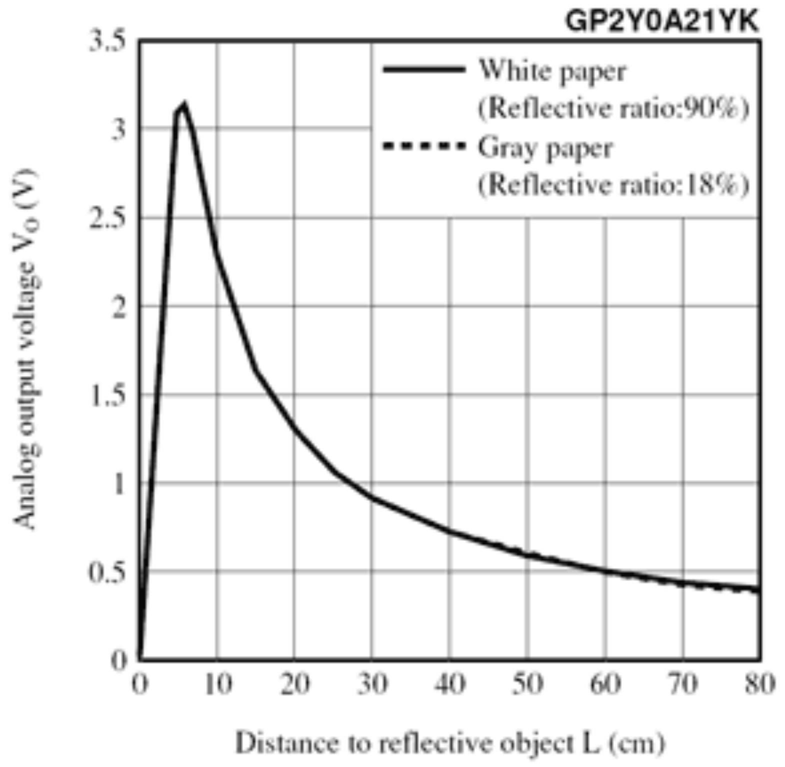

Figure 11: Expect output Voltage for IR Sensor from Sharp (Manufacturer)

Conclusion

This was a very challenging project for me to work on and I am very excited with the work that got accomplished. I developed a major platform, pushed the limits of linear actuator accuracy, integrated a pan/tilt system, built my own software and libraries for several components, included several sensors, and much more. A lot of unexpected challenges arose and sense this project was always very highly integrated this cause several redesigns of the entire system. However, I stuck with this project and proved very adaptable.

I believe cooking breakfast is a very hard task for a robot to do especially given the time limit of one semester and the budget of a college student. Working within my limits as one college student and with the components I could afford is an important accomplishment in and of itself (although I did stretch my budget quite a bit further than I intended to). Given the amount of money I spent at the end I would have made several changes to my parts list had I prepared to spend that amount from the beginning. The most crucial part would have been to trade the linear actuators for a full-blown CNC Router design. These designs go into 3D printers and other systems and are known to achieve accuracies of .1 mm. This would allow me to easily incorporate IR and other sensors into my design without worry of the compounding error.

Another important component would have been to get a much stronger/more accurate claw. I started to run out of money when I bought the claw and I purchased one that wasn't up to the task to firmly griping an egg. Some claws (really expensive ones) even have tactile feedback built-in so that I would have been able to know how hard to grip the egg without breaking it. Had I been able to buy these more expensive components I believe I would have been able to crack eggs as I originally hoped to.

I was aware from early in the semester that my robot was going to be big and that it's size would cause it to be difficult for me to move around. However, I drastically underestimated how much of a burden this would be and I would highly advise other students to not undertake tasks that require big robots. In addition, I also would have been more careful to not be stuck with such high accuracy requirements had I started this project over again.

I have many plans to improve my next breakfast robot. The first step is going to be to buy the more expensive components that are actually designed for this sort of precision work. With the more advanced system I will work towards adding more breakfast foods and much more flexibility to the software running the robot. I also plan to use a higher end microcontroller that has all the peripherals I need so that I'm not running with two microcontrollers. I think building the microcontroller myself might be a fun project in itself.

Most importantly I've had a lot of fun working on this project and I have learned tons about robotics through this course. I would highly recommend it to any senior or graduate student!

References

[1] John A. Seibel, MD, "Why Breakfast is the Most Important Meal", http://www.webmd.com/diet/guide/most-important-meal, WebMD, February 3, 2012

[2] Fiona Macrae, "Breakfast like a king: Why a high fat bacon and eggs meal is the healthiest start to the day", http://www.dailymail.co.uk/health/article-1262453/High-fat-bacon-eggs-breakfast-healthiest-start-day.html, Mail Online, March 31st 2010

[3] Concentric LACT12P-12V-20 Linear actuator with Feedback: 12" Stroke, 12V, 0.5"/s, Pololu, www.pololu.com/catalog/product/2313

[4] Concentric LACT12P-12V-5 Linear actuator with Feedback: 12" Stroke, 12V, 1.7"/s, Pololu, www.pololu.com/catalog/product/2327

[5] Servo – Medium, Sparkfun Electronics, ROB-10333, www.sparkfun.com/products/10333

[6] Servo – Small, Sparkfun Electronics, ROB-09065, www.sparkfun.com/products/9065

[7] SMARTGPU, Vizic Technologies, vizictechnologies.com/#/smart-gpu4554296549

[8] Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm, Pololu, www.pololu.com/catalog/product/136

[9] Interlink Electronics FSR 406, Interlink Electronics, http://www.interlinkelectronics.com/Product/Standard-406-FSR

[10] mbed NXP LPC1768, ARM microcontroller, MBED, http://mbed.org/handbook/mbed-NXP-LPC1768

Appendices

breakfastBotProto.cpp – used for main controller mbed1

```
#include "mbed.h"
#include "SMARTGPU.h"
#include "MyEthernet.h"
#include <string>

SMARTGPU lcd(p28,p27,p26);       //(TX,RX,Reset);
DigitalOut acRelay(p21);
DigitalOut led1(LED1);
DigitalOut led2(LED2);
//Each time we use the touchscreen we must define a int array that
stores the X and Y readed or touched coordinates.
int touch[2];
AnalogIn pressure_sensor(p20); //used for pressure sensor reading
(under plate)

void straightLine(int height, int thickness, int color) {
   int xstart, xend, i;
   xstart = 0;
   xend = 320;

   for (i=0; i< thickness; i++) {
      lcd.drawLine(xstart, height+i, xend, height+i, color);
   }
}

/* Screen that displays output when robot in operation */
void inOperationScreen(string output[]) {
   lcd.erase();
   int numOutputLines = 5;

   char *lcdout = (char*)malloc( sizeof( char ) * 25+1); //Can only
output 25 characters per line (+1 for terminating null char)
```

```
  //lcd.string(10,5,310,25,BLACK,FONT4,TRANS,"Output from
operation:");
  straightLine(30,4,BLACK);
  int i = 0;
  for (i; i<numOutputLines; i++) {
    strcpy(lcdout, output[i].c_str());
    lcd.string(5,40+i*28,315,58+i*28,BLACK,FONT3,TRANS,lcdout);
  }
  straightLine(180,4,BLACK);
//   lcd.imageSD(90,190,"resetbtn");
  free(lcdout);
}

int main() {
  string output[5]; //For output on inProgress Screen of LCD (for debug
output) [FIRST ITERATION]
  /* INITIALIZE LCD */
  lcd.reset();              //physically reset SMARTGPU
  lcd.start();              //initialize the SMARTGPU processor
  lcd.baudChange(2000000);       //set high baud for comm with LCD
  lcd.setScreenBackground(WHITE); //set background to white
  /* END INIT LCD */
  /* Declare and Init */
  unsigned char numEggs = 0;  //number of eggs to make (currently 1-
3)
  bool inOperation = false; //when false display selection screen on
LCD; when true display inProgress screen with debug output
  bool change = true; //when change == true we need a screen refresh
on LCD
  bool readyForBreakfast = true;
  bool preheat = true;
  bool stir = false;
  bool scoop = false;

  string txstr;
  string rxstr;
  MyEthernet eth;
  char buffer[6];
  string dis;
  acRelay = 0;
  led1 = 0;
  led2 = 0;
  /* End Declare and Init */

  eth.waitForLink();
  wait(1);

  while (1) {
    if (!inOperation) {
      if (change) {
        lcd.erase();
        change = false;
        readyForBreakfast = true;
        lcd.string(10,5,240,25,BLACK,FONT4,TRANS,"Number of
Eggs:");
        /* Draw Number of Egg Button Array (Horz) */
        if (numEggs == 1) {
          lcd.imageSD(10,35,"1btnalt");
          lcd.imageSD(85,35,"2btn");
          lcd.imageSD(160,35,"3btn");
        } else if (numEggs == 2) {
          lcd.imageSD(10,35,"1btn");
          lcd.imageSD(85,35,"2btnalt");
          lcd.imageSD(160,35,"3btn");
        } else if (numEggs == 3) {
          lcd.imageSD(10,35,"1btn");
          lcd.imageSD(85,35,"2btn");
          lcd.imageSD(160,35,"3btnalt");
        } else {
          lcd.imageSD(10,35,"1btn");
          lcd.imageSD(85,35,"2btn");
          lcd.imageSD(160,35,"3btn");
        }
        straightLine(80,4,BLACK);
        /* End of Button Array (EGGS) */
        //lcd.string(10,90,240,115,BLACK,FONT4,TRANS,"Options:");
        /* No options for now :( */
            straightLine(180,4,BLACK);
            lcd.imageSD(90,190,"startbtn");
         }

         /* Wait for touch to do something */
         while (lcd.touchScreen(touch)==0);

         //check for touch at number of eggs row
         if (touch[YCOORD]>33 && touch[YCOORD]<72) {
            if (touch[XCOORD]>8 && touch[XCOORD]<72) {
               numEggs = 1;
               change = true;
            } else if (touch[XCOORD]>83 && touch[XCOORD]<142) {
               numEggs = 2;
               change = true;
            } else if (touch[XCOORD]>158 && touch[XCOORD]<222) {
               numEggs = 3;
               change = true;
            }
         } /* touch at number of eggs row (if) */
         //check for touch at start main program
         else if (touch[YCOORD]>188 && touch[XCOORD]<240) {
            if (touch[XCOORD]>88 && touch[XCOORD]<172) {
               if (numEggs > 0) { /* else keep at same screen and wait for
user to make selection */
                  inOperation = true;
                  change = true;
               }
            }
         } /* touch at start button location (else if) */
      } /* if !inOperation */
      else { //we are inOperation
         if (change) {
            change = false;
            dis = "Preheat stage";
            output[0] = dis;
            output[1] = "         ";
            output[2] = "         ";
            output[3] = "         ";
```

```
output[4] = "          ";
inOperationScreen(output);
if (readyForBreakfast) {
   readyForBreakfast = false;
   /* Begin Breakfast Routine */
   if (preheat){
      acRelay = 1;
      wait(4*60);
      change = true;
      preheat = false;
      stir = true;
      dis = "Stir stage";
      output[1] = dis;
      inOperationScreen(output);
   }
   if (stir){
      txstr = 's';
      eth.txPKT(txstr);
      rxstr = eth.rxPKT();
      led1 = 1;
      change = true;
      stir = false;
      scoop = true;
      dis = "Scoop stage";
      output[2] = dis;
      inOperationScreen(output);
   }
   if (scoop){
      wait(60);
      acRelay = 0;
      txstr = 'c';
      eth.txPKT(txstr);
      rxstr = eth.rxPKT();
      //while(pressure_sensor <.8){
         //eth.txPKT(txstr);
         //rxstr = eth.rxPKT();
      //}
      led2 = 1;
      output[0] = "          ";
```

```
                    output[1] = "          ";
                    output[2] = "          ";
                    inOperation = false;
                    change = true;
                    preheat = true;
                    scoop = false;
                }
            } /* if readyForBreakfast */
        } /* if change */

    } /* else (inOperation) */
  } /* main while loop */
} /* int main function */
```

motionProto: used for mbed2

```
/*
 * Controlling linear actuators (x3) and servo motors for pan/tilt/bowl
 * using jrk21v3 to controll linear actuators (one driver for each lin act)
 * LinAct Range: 0-4000
*/
#include "mbed.h"
#include "Servo.h"
#include "MyEthernet.h"
#include <string>
#include <stdio.h>

Serial linactx(p9,p10);
Serial linacty(p13,p14);
Serial linactz(p28,p27);

Servo pan(p21);
Servo tilt(p22);
Servo bowl(p23);

Timer counter;


DigitalOut led1(LED1);
DigitalOut led2(LED2);
DigitalOut led3(LED3);

Serial pc(USBTX, USBRX);

int moveLinActx(int position);
int moveLinActy(int position);
int moveLinActz(int position);
void flushSerialBufferX(void) {
  char char1 = 0;
  while (linactx.readable()) {
    char1 = linactx.getc();
  }
  return;
}
void flushSerialBufferY(void) {
  char char1 = 0;
  while (linacty.readable()) {
    char1 = linacty.getc();
  }
  return;
}
void flushSerialBufferZ(void) {
  char char1 = 0;
  while (linactz.readable()) {
    char1 = linactz.getc();
  }
  return;
}

int main() {
  float panRange = 0.0008;
  float panPosition = 0.0;
  float tiltRange = 0.0009;
  float tiltPosition = 0.7;
  float bowlRange = 0.0009;
  float bowlPosition = 0.0;

  pan.calibrate(panRange, 45.0);
```

```
tilt.calibrate(tiltRange, 45.0);
bowl.calibrate(bowlRange, 45.0);

pan = panPosition;
tilt = tiltPosition;
bowl = bowlPosition;

//Static variables for the Stir Command!!!
int eggPanX = 4000;
int startEggPanY = 4000;
int eggPanZ = 2850;

int endEggPanY = 800;

int liftZ = 1600;

float stirTimer = 2.2*60;
int maxStirCount = 4;
int tipTimer = 15;

int panForward = 0;
int tiltStir = .7;

//ethernet
char rxchar;
float rxfloat;
string rxString, txString;
MyEthernet eth;

eth.waitForLink();
led1 = 1;

while (1) {
  //pc.printf("Input Command: ");
  rxString = eth.rxPKT();
  led2 = 1;
  sscanf(rxString.c_str(), "%c", &rxchar);
  pc.printf("rx char = %c\r\n", rxchar);
  switch (rxchar) {
        case 's':
          //get starting and ending position of the egg pan (static
variables)
          //start a timer for how long to do this
          //begin from one end and go to other
          pc.printf("\r\n Init\r\n");
          led3 = 1;
          tilt = .7;
          pan = 0;
          moveLinActz(liftZ);
          moveLinActx(eggPanX);
          moveLinActy(startEggPanY);
          moveLinActz(eggPanZ);
          pc.printf("\r\n Start\r\n");
          counter.start();
          int stirCount = 0;
          while (stirCount < maxStirCount) {
            if (counter.read()>stirTimer) {
              moveLinActy(endEggPanY+750);
              moveLinActz(liftZ+550);
              moveLinActy(endEggPanY);
              moveLinActz(eggPanZ);
              moveLinActy(startEggPanY-900);
              moveLinActz(liftZ+550);
              moveLinActy(startEggPanY);
              moveLinActz(eggPanZ);
              counter.stop();
              counter.reset();
              counter.start();
              stirCount = stirCount+1;
              //pc.printf("while\r\n");
            }
            wait(30);
            pc.printf("Counter=%f\r\n", counter.read());
          }
          counter.stop();
          counter.reset();
          moveLinActz(liftZ);
          pc.printf("\r\n Done\r\n");
```

```
    txString = 'd';
    eth.txPKT(txString);
    break;
  case 'c':
    pc.printf("\r\n Init\r\n");
    tilt = .7;
    pan = 0;
    /* move egg clump to oposite side and get into position*/
    moveLinActz(liftZ);
    moveLinActx(eggPanX);
    moveLinActy(startEggPanY);
    moveLinActz(eggPanZ);
    moveLinActy(endEggPanY+750);
    moveLinActz(liftZ);
    moveLinActy(startEggPanY);
    /* in position */
    pc.printf("\r\n Start\r\n");
     for (int i = 0; i< 1; i++) { //one go
      moveLinActz(eggPanZ);
      moveLinActy(startEggPanY-600); //y=3400
      tilt = .6;
      wait(.1);
      tilt = .5;
      wait(.1);
      moveLinActz(eggPanZ+200); // z=2850
      tilt = .4;
      moveLinActz(eggPanZ+300); // z=2950
      tilt = .3;
      moveLinActz(eggPanZ+450); // z=3100
      moveLinActy(startEggPanY-1250); //y=2750
      tilt = .2;
      moveLinActz(eggPanZ+650); // z=3300
      moveLinActy(startEggPanY-1550); //y=2450
      tilt = .1;
      tilt = 0;
      moveLinActz(liftZ+800); //z=2400
      wait(1);
      moveLinActy(endEggPanY); //y=800
      tilt = .7;
```

```
            wait(3);
            moveLinActz(liftZ);
            wait(2);
            moveLinActy(startEggPanY);
            pc.printf("\r\n For\r\n");
          }
          pc.printf("\r\n Done\r\n");
          txString = 'd';
          eth.txPKT(txString);
          break;
        case 't':
          bowl = .9;
          wait(tipTimer);
          bowl = .0;
          txString = 'd';
          eth.txPKT(txString);
          break;

      }
    }
  }

int moveLinActx(int position) {
  int target, feedback, prevfeedback, fcount, lowB, highB, c;
  int done=0;
  int timeout = 50;
  while (!done) {
    flushSerialBufferX();
    target = -9999;
    feedback = -1;
    while (1) {
      if (linactx.writeable()) {
        linactx.putc(0xC0 + (position & 0x1F));
        break;
      }
      pc.printf("\t\t\tpos1 putc while loop\r\n");
    }
    while (1) {
      if (linactx.writeable()) {
        linactx.putc((position >> 5) & 0x7F);
        break;
    }
    pc.printf("\t\t\tpos2 putc while loop\r\n");
}

while (1) {
    if (linactx.writeable()) {
        linactx.putc(0xA3);
        break;
    }
    pc.printf("\t\t\tA3 putc while loop\r\n");
}
lowB = 0;
highB=0;
c=0;
while (c<timeout) {
    if (linactx.readable() && lowB == 0) {
        lowB = linactx.getc();
    } else if (linactx.readable()) {
        highB = linactx.getc();
        target = lowB + (highB << 8);
        break;
    }
    target = -9999;
    c = c+1;
    pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
    flushSerialBufferX();
    while (1) {
        if (linactx.writeable()) {
            linactx.putc(0xA3);
            break;
        }
        pc.printf("\t\t\tA3over putc while loop\r\n");
    }
    lowB = 0;
    highB=0;
```

```
        c=0;
        while (c<timeout) {
            if (linactx.readable() && lowB == 0) {
                lowB = linactx.getc();
            } else if (linactx.readable()) {
                highB = linactx.getc();
                target = lowB + (highB << 8);
                break;
            }
            target = -9999;
            c = c+1;
        pc.printf("c=%i, timeout=%i\r\n", c, timeout);
        }
    }
    pc.printf("\ttarget=%i\r\n",target);
    if (target == -9999)
        continue;
    wait(1);
    fcount = 0;
    prevfeedback=-1;
    while (fcount < 20) {
        while (1) {
            if (linactx.writeable()) {
                linactx.putc(0xA7);
                break;
            }
            pc.printf("\t\t\tputc while loop\r\n");
        }
        lowB = 0;
        highB=0;
        c=0;
        while (c<timeout) {
            if (linactx.readable() && lowB == 0) {
                lowB = linactx.getc();
            } else if (linactx.readable()) {
                highB = linactx.getc();
                feedback = lowB + (highB << 8);
                break;
            }
            feedback = -1;
            c=c+1;
        }
        while (feedback>4095) {
            flushSerialBufferX();
            while (1) {
                if (linactx.writeable()) {
                    linactx.putc(0xA7);
                    break;
                }
                pc.printf("\t\t\tputc while loop\r\n");
            }
            lowB = 0;
            highB=0;
            c=0;
            while (c<timeout) {
                if (linactx.readable() && lowB == 0) {
                    lowB = linactx.getc();
                } else if (linactx.readable()) {
                    highB = linactx.getc();
                    feedback = lowB + (highB << 8);
                    break;
                }
                feedback = -1;
                c=c+1;
            }
        }
        if (!(feedback > target + 60 || feedback < target-60)) {
            pc.printf("\t\tdone feedback=%i\r\n", feedback);
            done = 1;
            break;
        } else if (prevfeedback == feedback) {
            break;
        }
        wait(1);
        fcount = fcount + 1;
        prevfeedback = feedback;
    } /* while fcount < 20 */
} /* while !done */
```

```
    pc.printf("\tX target=%i, feedback=%i\r\n", target, feedback);
    return 1;
}

int moveLinActy(int position) {
    int target, feedback, prevfeedback, fcount, lowB, highB, c;
    int done=0;
    int timeout = 50;
    while (!done) {
        flushSerialBufferY();
        target = -9999;
        feedback = -1;
        while (1) {
            if (linacty.writeable()) {
                linacty.putc(0xC0 + (position & 0x1F));
                break;
            }
            pc.printf("\t\t\tpos1 putc while loop\r\n");
        }
        while (1) {
            if (linacty.writeable()) {
                linacty.putc((position >> 5) & 0x7F);
                break;
            }
            pc.printf("\t\t\tpos2 putc while loop\r\n");
        }

        while (1) {
            if (linacty.writeable()) {
                linacty.putc(0xA3);
                break;
            }
            pc.printf("\t\t\tA3 putc while loop\r\n");
        }
        lowB = 0;
        highB=0;
        c=0;
        while (c<timeout) {
            if (linacty.readable() && lowB == 0) {
```

```
        lowB = linacty.getc();
    } else if (linacty.readable()) {
        highB = linacty.getc();
        target = lowB + (highB << 8);
        break;
    }
    target = -9999;
    c=c+1;
    pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
    flushSerialBufferY();
    while (1) {
        if (linacty.writeable()) {
            linacty.putc(0xA3);
            break;
        }
        pc.printf("\t\t\tA3over putc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linacty.readable() && lowB == 0) {
            lowB = linacty.getc();
        } else if (linacty.readable()) {
            highB = linacty.getc();
            target = lowB + (highB << 8);
            break;
        }
        target = -9999;
        c=c+1;
    }
}
pc.printf("\ttarget=%i\r\n",target);
if (target == -9999)
    continue;
wait(1);
fcount = 0;
```

```
prevfeedback=-1;
while (fcount < 20) {
   while (1) {
      if (linacty.writeable()) {
         linacty.putc(0xA7);
         break;
      }
      pc.printf("\t\t\tputc while loop\r\n");
   }
   lowB = 0;
   highB=0;
   c=0;
   while (c<timeout) {
      if (linacty.readable() && lowB == 0) {
         lowB = linacty.getc();
      } else if (linacty.readable()) {
         highB = linacty.getc();
         feedback = lowB + (highB << 8);
         break;
      }
      feedback = -1;
      c=c+1;
   }
   while (feedback>4095) {
      flushSerialBufferY();
      while (1) {
         if (linacty.writeable()) {
            linacty.putc(0xA7);
            break;
         }
         pc.printf("\t\t\tputc while loop\r\n");
      }
      lowB = 0;
      highB=0;
      c=0;
      while (c<timeout) {
         if (linacty.readable() && lowB == 0) {
            lowB = linacty.getc();
         } else if (linacty.readable()) {
```

```
                  highB = linacty.getc();
                  feedback = lowB + (highB << 8);
                  break;
               }
               feedback = -1;
               c=c+1;
            }
         }
         if (!(feedback > target + 60 || feedback < target-60)) {
            pc.printf("\t\tdone feedback=%i\r\n", feedback);
            done = 1;
            break;
         } else if (prevfeedback == feedback) {
            break;
         }
         wait(1);
         fcount = fcount + 1;
         prevfeedback = feedback;
      } /* while fcount < 20 */
   } /* while !done */
   pc.printf("\tY target=%i, feedback=%i\r\n", target, feedback);
   return 1;
}

int moveLinActz(int position) {
   int target, feedback, prevfeedback, fcount, lowB, highB, c;
   int done=0;
   int timeout = 200;
   while (!done) {
      flushSerialBufferZ();
      target = -9999;
      feedback = -1;
      while (1) {
         if (linactz.writeable()) {
            linactz.putc(0xC0 + (position & 0x1F));
            break;
         }
         pc.printf("\t\t\tpos1 putc while loop\r\n");
      }
```

```
while (1) {
   if (linactz.writeable()) {
      linactz.putc((position >> 5) & 0x7F);
      break;
   }
   pc.printf("\t\t\tpos2 putc while loop\r\n");
}

while (1) {
   if (linactz.writeable()) {
      linactz.putc(0xA3);
      break;
   }
   pc.printf("\t\t\tA3 putc while loop\r\n");
}
lowB = 0;
highB=0;
c=0;
while (c<timeout) {
   if (linactz.readable() && lowB == 0) {
      lowB = linactz.getc();
   } else if (linactz.readable()) {
      highB = linactz.getc();
      target = lowB + (highB << 8);
      break;
   }
   target = -9999;
   c=c+1;
   pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
   flushSerialBufferZ();
   while (1) {
      if (linactz.writeable()) {
         linactz.putc(0xA3);
         break;
      }
      pc.printf("\t\t\tA3over putc while loop\r\n");
   }
```

```
            lowB = 0;
            highB=0;
            c=0;
            while (c<timeout) {
               if (linactz.readable() && lowB == 0) {
                  lowB = linactz.getc();
               } else if (linactz.readable()) {
                  highB = linactz.getc();
                  target = lowB + (highB << 8);
                  break;
               }
               target = -9999;
               c=c+1;
               pc.printf("c=%i, timeout=%i\r\n", c, timeout);
            }
         }
         pc.printf("\ttarget=%i\r\n",target);
         if (target == -9999)
            continue;
         wait(1);
         fcount = 0;
         prevfeedback=-1;
         while (fcount < 20) {
            while (1) {
               if (linactz.writeable()) {
                  linactz.putc(0xA7);
                  break;
               }
               pc.printf("\t\t\tputc while loop\r\n");
            }
            lowB = 0;
            highB=0;
            c=0;
            while (c<timeout) {
               if (linactz.readable() && lowB == 0) {
                  lowB = linactz.getc();
               } else if (linactz.readable()) {
                  highB = linactz.getc();
                  feedback = lowB + (highB << 8);
```

```
        break;
    }
    feedback = -1;
    c=c+1;
}
while (feedback>4095) {
    flushSerialBufferZ();
    while (1) {
        if (linactz.writeable()) {
            linactz.putc(0xA7);
            break;
        }
        pc.printf("\t\t\tputc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linactz.readable() && lowB == 0) {
            lowB = linactz.getc();
        } else if (linactz.readable()) {
            highB = linactz.getc();
            feedback = lowB + (highB << 8);
            break;
        }
        feedback = -1;
        c=c+1;
    }
}
if (!(feedback > target + 60 || feedback < target-60)) {
    pc.printf("\t\tdone feedback=%i\r\n", feedback);
    done = 1;
    break;
} else if (prevfeedback == feedback) {
    break;
}
wait(1);
fcount = fcount + 1;
prevfeedback = feedback;
```

```
      } /* while fcount < 20 */
   } /* while !done */
   pc.printf("\tZ target=%i, feedback=%i\r\n", target, feedback);
   return 1;
}
```

MyEthernet.h

```
#ifndef MYETHERNET_H
#define MYETHERNET_H
#include <string>

class MyEthernet{

private:
   Ethernet eth;

public:
   MyEthernet();
   string rxPKT();
   void txPKT(string input);
   void waitForLink();
};
#endif
```

MyEthernet.cpp

```
#include "mbed.h"
#include "MyEthernet.h"

/* IMPORTANT:
   - Don't forget to wait(1) before first TX send
      * This is after both sides waitForLink()
      * I have no idea why you need to do this but it doesn't work
otherwise
*/

MyEthernet::MyEthernet(){ }
```

```
string MyEthernet::rxPKT(){  //pktsize is size
   string retString;

   while(1) {
      int size = eth.receive();
      if(size > 0) {
         char *rxbuf = (char*)malloc( sizeof( char ) * size); // the +1 is for
a null terminator
         eth.read(rxbuf, size);
         retString = rxbuf;
         free(rxbuf);
         break;
      }
   }
   return retString;
}

void MyEthernet::txPKT(string input){
   char *txbuf = (char*)malloc( sizeof( char ) * input.length()+1); //+1
for null terminator
   strcpy(txbuf, input.c_str());
   eth.write(txbuf, input.length()+1);
   eth.send();
   free(txbuf);
}

void MyEthernet::waitForLink(){
   int linkval = 0;
   while(!linkval){
      linkval = eth.link();
   }
}
```

analogINtoPC.cpp

```
#include "mbed.h"

AnalogIn ain(p20);
Serial pc(USBTX, USBRX);
```

```
int main() {
   long unsigned temp;
   while (1) {
      long unsigned  reg[7] = {0,0,0,0,0,0,0};
      wait(1);
      for (int i = 0; i<7; i++) {
         temp = (ain.read_u16()>>4);
         printf("\ttmp = 0x%04X\r\n", temp);
         reg[i] = temp;
         wait(.2);
      }
      //selection sort
      int position;
      int min;

      for (position = 0; position < 6; position++) {
         min = position;

         for (int i = position+1; i < 6; i++) {
            if (reg[i] < reg[min]) {
               min = i;
            }
         }
         if ( min != position ) {
            temp = reg[position];
            reg[position] = reg[min];
            reg[min] = temp;
         }
      }
      pc.printf("ain = 0x%04X\r\n", reg[3]);
   }
}
```

SevroProgram – used to calibrate servos

```
#include "mbed.h"
#include "Servo.h"
```

```
Servo myservo(p21);
Serial pc(USBTX, USBRX);

int main() {
   printf("Servo Calibration Controls:\n");
   printf("1,2,3 - Position Servo (full left, middle, full right)\n");
   printf("4,5 - Decrease or Increase range\n");

   float range = 0.0005;
   float position = 0.5;

   while(1) {
      switch(pc.getc()) {
         case '1': position = 0.0; break;
         case '2': position = 0.5; break;
         case '3': position = 1.0; break;
         case '4': range += 0.0001; break;
         case '5': range -= 0.0001; break;
         case '6': position += .1; break;
         case '7': position -= .1; break;
      }
      if (position > 1.0)
         position = 1.0;
      else if(position < 0.0)
         position = 0.0;
      printf("position = %.1f, range = +/-%0.4f\n", position, range);
      myservo.calibrate(range, 45.0);
      myservo = position;
   }
}
```

BreakfastBot – used to test code for touchscreen

```
#include "mbed.h"
#include "SMARTGPU.h"
#include "MyEthernet.h"
#include <string>
```

```
SMARTGPU lcd(p28,p27,p26);       //(TX,RX,Reset);
DigitalOut acRelay(p21);
DigitalOut led1(LED1);
DigitalOut led2(LED2);
//Each time we use the touchscreen we must define a int array that
stores the X and Y readed or touched coordinates.
int touch[2];

void straightLine(int height, int thickness, int color) {
   int xstart, xend, i;
   xstart = 0;
   xend = 320;

   for (i=0; i< thickness; i++) {
      lcd.drawLine(xstart, height+i, xend, height+i, color);
   }
}

void inOperationScreen(string output[]) {
   lcd.erase();
   int numOutputLines = 5;

   char *lcdout = (char*)malloc( sizeof( char ) * 25+1); //Can only
output 25 characters per line (+1 for terminating null char)
   lcd.string(10,5,310,25,BLACK,FONT4,TRANS,"Output from
operation:");
   straightLine(30,4,BLACK);
   int i = 0;
   for (i; i<numOutputLines; i++) {
      strcpy(lcdout, output[i].c_str());
      lcd.string(5,40+i*28,315,58+i*28,BLACK,FONT3,TRANS,lcdout);
   }
   straightLine(180,4,BLACK);
   lcd.imageSD(90,190,"resetbtn");
   free(lcdout);
}

int main() {
   string output[5]; //For output on inProgress Screen of LCD (for debug
output) [FIRST ITERATION]
   /* INITIALIZE LCD */
   lcd.reset();                  //physically reset SMARTGPU
   lcd.start();                  //initialize the SMARTGPU processor
   lcd.baudChange(2000000);        //set high baud for advanced
applications with LCD
   lcd.setScreenBackground(WHITE); //set background to white
   /* END INIT LCD */
   unsigned char numEggs = 0;  //number of eggs to make (currently 1-
3)
   bool inOperation = false; //when false display selection screen on
LCD; when true display inProgress screen with debug output
   bool change = true; //when change == true we need a screen refresh
on LCD
   bool readyForBreakfast = true;

   string txstr;
   string rxstr;
   MyEthernet eth;
   char buffer[6];

   acRelay = 0;
   led1 = 0;
   led2 = 0;

   eth.waitForLink();
   wait(1);

   while (1) {
      if (!inOperation) {
         if (change) {
            lcd.erase();
            change = false;
            readyForBreakfast = true;
            lcd.string(10,5,240,25,BLACK,FONT4,TRANS,"Number of
Eggs:");
            /* Draw Number of Egg Button Array (Horz) */
            if (numEggs == 1) {
```

```
      lcd.imageSD(10,35,"1btnalt");
      lcd.imageSD(85,35,"2btn");
      lcd.imageSD(160,35,"3btn");
   } else if (numEggs == 2) {
      lcd.imageSD(10,35,"1btn");
      lcd.imageSD(85,35,"2btnalt");
      lcd.imageSD(160,35,"3btn");
   } else if (numEggs == 3) {
      lcd.imageSD(10,35,"1btn");
      lcd.imageSD(85,35,"2btn");
      lcd.imageSD(160,35,"3btnalt");
   } else {
      lcd.imageSD(10,35,"1btn");
      lcd.imageSD(85,35,"2btn");
      lcd.imageSD(160,35,"3btn");
   }
   straightLine(80,4,BLACK);
   /* End of Button Array (EGGS) */
   lcd.string(10,90,240,115,BLACK,FONT4,TRANS,"Options:");
   /* No options for now :( */
   straightLine(180,4,BLACK);
   lcd.imageSD(90,190,"startbtn");
}

/* Wait for touch to do something */
while (lcd.touchScreen(touch)==0);

//check for touch at number of eggs row
if (touch[YCOORD]>33 && touch[YCOORD]<72) {
   if (touch[XCOORD]>8 && touch[XCOORD]<72) {
      numEggs = 1;
      change = true;
   } else if (touch[XCOORD]>83 && touch[XCOORD]<142) {
      numEggs = 2;
      change = true;
   } else if (touch[XCOORD]>158 && touch[XCOORD]<222) {
      numEggs = 3;
      change = true;
   }
```

```
         } /* touch at number of eggs row (if) */
         //check for touch at start main program
         else if (touch[YCOORD]>188 && touch[XCOORD]<240) {
            if (touch[XCOORD]>88 && touch[XCOORD]<172) {
               if (numEggs > 0) { /* else keep at same screen and wait for
user to make selection */
                  inOperation = true;
                  change = true;
               }
            }
         } /* touch at start button location (else if) */
      } /* if !inOperation */
      else { //we are inOperation
         if (change) {
            change = false;
            inOperationScreen(output);
            if (readyForBreakfast) {
               readyForBreakfast = false;
               /* Begin Breakfast Routine */
               wait(60);
            } /* if readyForBreakfast */
         } /* if change */

      } /* else (inOperation) */
   } /* main while loop */
} /* int main function */
```

EthernetCom1 – prototype Ethernet code

```
#include "mbed.h"
#include <string>
#include "MyEthernet.h"

Serial pc(USBTX, USBRX); // tx, rx

string convertInt(int number)
{
   if (number == 0)
```

```
        return "0";
    string temp="";
    string returnvalue="";
    while (number>0)
    {
        temp+=number%10+48;
        number/=10;
    }
    for (int i=0;i<temp.length();i++)
        returnvalue+=temp[temp.length()-i-1];
    return returnvalue;
}

int main() {
    string rxString, txString;
    MyEthernet x;
    char buffer[6];

    x.waitForLink();
    pc.printf("Got Linked!\n");
    rxString = x.rxPKT();
    pc.printf("RX: %s\n", rxString.c_str());
    txString = "I Recieved your value";
    x.txPKT(txString);
    pc.printf("TX: %s\n", txString.c_str());
    rxString = x.rxPKT();
    pc.printf("RX: %s\n", rxString.c_str());
    while(1)
    {
        pc.scanf("%s", &buffer);
        txString.assign(buffer, 6);
        pc.printf("\r\n%s\r\n", txString.c_str());
        x.txPKT(txString);
        rxString = x.rxPKT();
        pc.printf("\r\n%s\r\n", rxString.c_str());
    }

}
```

EthernetCom2 – went through sever revisions

```
#include "mbed.h"
#include <string>

Ethernet eth;
DigitalOut led(LED1);
DigitalOut led2(LED2);

string convertInt(int number)
{
   if (number == 0)
      return "0";
   string temp="";
   string returnvalue="";
   while (number>0)
   {
      temp+=number%10+48;
      number/=10;
   }
   for (int i=0;i<temp.length();i++)
      returnvalue+=temp[temp.length()-i-1];
   return returnvalue;
}

string rxPKT(){  //pktsize is size
   string retString;

   while(1) {
      int size = eth.receive();
      if(size > 0) {
         char *rxbuf = (char*)malloc( sizeof( char ) * size); // the +1 is for
a null terminator
         eth.read(rxbuf, size);
         retString = rxbuf;
         free(rxbuf);
         break;
      }
   }
```

```
  return retString;
}

void txPKT(string input){
  char *txbuf = (char*)malloc( sizeof( char ) * input.length()+1); //+1
for null terminator
  strcpy(txbuf, input.c_str());
  eth.write(txbuf, input.length()+1);
  eth.send();
  free(txbuf);
}

void waitForLink(){
  int linkval = 0;
  while(!linkval){
    linkval = eth.link();
  }
}

int main() {
  string rxString, txString;

  waitForLink();
  led = !led;
  wait(1);

  int counter = 0;

  txString = "This is a test" + convertInt(counter);
  txPKT(txString);
  led2 = !led2;

  rxString = rxPKT();

  counter++;
  txString = "This is a test" + convertInt(counter);
  txPKT(txString);
  led2 = !led2;
```

```
    txString = "Received";

    while(1)
    {
        rxString = rxPKT();
        txPKT(txString);
    }
}
```

linactservos.cpp – test/calibrate actuation

```
/*
 * Controlling linear actuators (x3) and servo motors for pan/tilt/bowl
 * using jrk21v3 to controll linear actuators (one driver for each lin act)
 * LinAct Range: 0-4000
*/
#include "mbed.h"
#include "Servo.h"

Serial linactx(p9,p10);
Serial linacty(p13,p14);
Serial linactz(p28,p27);

Servo pan(p21);
Servo tilt(p22);
Servo bowl(p23);

AnalogIn irskillet(p20);
AnalogIn irplate(p19);

Serial pc(USBTX, USBRX);

int moveLinActx(int position);
int moveLinActy(int position);
int moveLinActz(int position);

void flushSerialBufferX(void) {
    char char1 = 0;
    while (linactx.readable()) {
        char1 = linactx.getc();
    }
    return;
}
void flushSerialBufferY(void) {
    char char1 = 0;
    while (linacty.readable()) {
        char1 = linacty.getc();
    }
    return;
}
void flushSerialBufferZ(void) {
    char char1 = 0;
    while (linactz.readable()) {
        char1 = linactz.getc();
    }
    return;
}

int main() {
    int xPosition = 0.0;
    int yPosition = 0.0;
    int zPosition = 0.0;

    float panRange = 0.0008;
    float panPosition = 0;
    float tiltRange = 0.0009;
    float tiltPosition = 0.7;
    float bowlRange = 0.0009;
    float bowlPosition = 0.5;

    pan.calibrate(panRange, 45.0);
    tilt.calibrate(tiltRange, 45.0);
    bowl.calibrate(bowlRange, 45.0);

    //Static variables for the Stir Command!!!
    int eggPanX = 4000;
    int startEggPanY = 4000;
    int eggPanZ = 2850;

    int endEggPanY = 800;

    int liftZ = 1600;

    int panForward = 0;
    int tiltStir = .7;

    while (1) {
        pc.printf("Input Command: ");
        switch (pc.getc()) {
            case 'x':
                pc.printf("/n");
                pc.printf("Input position: ");
                pc.scanf("%d", &xPosition);
                pc.printf("\nxLinAct: position = %d\r\n", xPosition);
                moveLinActx(xPosition);
                break;
            case 'y':
                pc.printf("/n");
                pc.printf("Input position: ");
                pc.scanf("%d", &yPosition);
                pc.printf("\nyLinAct: position = %d\n\n", yPosition);
                moveLinActy(yPosition);
                break;
            case 'z':
                pc.printf("/n");
                pc.printf("Input position: ");
                pc.scanf("%d", &zPosition);
                pc.printf("\nzLinAct: position = %d\n\n", zPosition);
                moveLinActz(zPosition);
                break;
            case 'p':
                pc.printf("/n");
                pc.printf("Input position: ");
                pc.scanf("%f", &panPosition);
                pc.printf("\nPan: position = %.1f, range = +/-%0.4f\n\n",
panPosition, panRange);
```

```
            pan = panPosition;
            break;
         case 't':
            pc.printf("/n");
            pc.printf("Input position: ");
            pc.scanf("%f", &tiltPosition);
            pc.printf("\nTilt: position = %.1f, range = +/-%0.4f\n\n",
tiltPosition, tiltRange);
            tilt = tiltPosition;
            break;
         case 'b':
            pc.printf("/n");
            pc.printf("Input position: ");
            pc.scanf("%f", &bowlPosition);
            pc.printf("\nbowl: position = %.1f, range = +/-%0.4f\n\n",
bowlPosition, bowlRange);
            bowl = bowlPosition;
            break;
         case 'c':
            pc.printf("\r\n Init\r\n");
            tilt = .7;
            pan = 0;
            /* move egg clump to oposite side and get into position*/
            moveLinActz(liftZ);
            moveLinActx(eggPanX);
            moveLinActy(startEggPanY);
            moveLinActz(eggPanZ);
            moveLinActy(endEggPanY+750);
            moveLinActz(liftZ);
            moveLinActy(startEggPanY);
            /* in position */
            pc.printf("\r\n Start\r\n");
            for (int i = 0; i< 1; i++) { //one go
               moveLinActz(eggPanZ);
               moveLinActy(startEggPanY-600); //y=3400
               tilt = .6;
               wait(.1);
               tilt = .5;
               wait(.1);
               moveLinActz(eggPanZ+200); // z=3050
               tilt = .4;
               moveLinActz(eggPanZ+300); // z=3150
               tilt = .3;
               moveLinActz(eggPanZ+450); // z=3300
               moveLinActy(startEggPanY-1250); //y=2750
               tilt = .2;
               moveLinActz(eggPanZ+650); // z=3500
               moveLinActy(startEggPanY-1550); //y=2450
               tilt = .1;
               tilt = 0;
               moveLinActz(liftZ+800); //z=2400
               wait(1);
               moveLinActy(endEggPanY); //y=800
               tilt = .7;
               wait(3);
               moveLinActz(liftZ);
               wait(2);
               moveLinActy(startEggPanY);
               pc.printf("\r\n For\r\n");
            }
            pc.printf("\r\n Done\r\n");
            break;
         case 'i':
            long unsigned temp, ir_reading;
            long unsigned  reg[21];
            for (int j = 0; j<21; j++) {
               reg[j] = 0;
            }
            for (int i = 0; i<21; i++) {
               temp = (ain.read_u16()>>4);
               printf("\ttmp = 0x%04X\r\n", temp);
               reg[i] = temp;
               wait(.2);
            }
            //selection sort
            int position;
            int min;
```

```
for (position = 0; position < 20; position++) {
  min = position;

  for (int i = position+1; i < 20; i++) {
    if (reg[i] < reg[min]) {
      min = i;
    }
  }
  if ( min != position ) {
    temp = reg[position];
    reg[position] = reg[min];
    reg[min] = temp;
  }
}
ir_reading = reg[3];
if (ir_reading < 0x690) {
  moveLinActx(4000);
} else if (ir_reading < 0x6D0) {
  moveLinActx(3900);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3800);
} else if (ir_reading > 0x730) {
  moveLinActx(3700);
} else if (ir_reading > 0x750) {
  moveLinActx(3600);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3500);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3400);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3300);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3200);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3100);
} else if (ir_reading > 0x6E0) {
  moveLinActx(3000);
} else if (ir_reading > 0x6E0) {
  moveLinActx(2900);
```

```
            } else if (ir_reading > 0x6E0) {
               moveLinActx(2800);
            } else if (ir_reading > 0xB00) {
               moveLinActx(2700);
            } else if (ir_reading > 0xB10) {
               moveLinActx(2600);
            } else if (ir_reading > 0xB15) {
               moveLinActx(2500);
            } else if (ir_reading > 0xB20) {
               moveLinActx(2400);
            } else {
               moveLinActx(2300);
            }
            break;
      }
   }
}

int moveLinActx(int position) {
   int target, feedback, prevfeedback, fcount, lowB, highB, c;
   int done=0;
   int timeout = 50;
   while (!done) {
      flushSerialBufferX();
      target = -9999;
      feedback = -1;
      while (1) {
         if (linactx.writeable()) {
            linactx.putc(0xC0 + (position & 0x1F));
            break;
         }
         pc.printf("\t\t\tpos1 putc while loop\r\n");
      }
      while (1) {
         if (linactx.writeable()) {
            linactx.putc((position >> 5) & 0x7F);
            break;
         }
         pc.printf("\t\t\tpos2 putc while loop\r\n");
```

```
}

while (1) {
   if (linactx.writeable()) {
      linactx.putc(0xA3);
      break;
   }
   pc.printf("\t\t\tA3 putc while loop\r\n");
}
lowB = 0;
highB=0;
c=0;
while (c<timeout) {
   if (linactx.readable() && lowB == 0) {
      lowB = linactx.getc();
   } else if (linactx.readable()) {
      highB = linactx.getc();
      target = lowB + (highB << 8);
      break;
   }
   target = -9999;
   c = c+1;
   pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
   flushSerialBufferX();
   while (1) {
      if (linactx.writeable()) {
         linactx.putc(0xA3);
         break;
      }
      pc.printf("\t\t\tA3over putc while loop\r\n");
   }
   lowB = 0;
   highB=0;
   c=0;
   while (c<timeout) {
      if (linactx.readable() && lowB == 0) {
         lowB = linactx.getc();
```

```
        } else if (linactx.readable()) {
            highB = linactx.getc();
            target = lowB + (highB << 8);
            break;
        }
        target = -9999;
        c = c+1;
        pc.printf("c=%i, timeout=%i\r\n", c, timeout);
    }
}
pc.printf("\ttarget=%i\r\n",target);
if (target == -9999)
    continue;
wait(1);
fcount = 0;
prevfeedback=-1;
while (fcount < 20) {
    while (1) {
        if (linactx.writeable()) {
            linactx.putc(0xA7);
            break;
        }
        pc.printf("\t\t\tputc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linactx.readable() && lowB == 0) {
            lowB = linactx.getc();
        } else if (linactx.readable()) {
            highB = linactx.getc();
            feedback = lowB + (highB << 8);
            break;
        }
        feedback = -1;
        c=c+1;
    }
    while (feedback>4095) {
      flushSerialBufferX();
      while (1) {
        if (linactx.writeable()) {
          linactx.putc(0xA7);
          break;
        }
        pc.printf("\t\t\tputc while loop\r\n");
      }
      lowB = 0;
      highB=0;
      c=0;
      while (c<timeout) {
        if (linactx.readable() && lowB == 0) {
          lowB = linactx.getc();
        } else if (linactx.readable()) {
          highB = linactx.getc();
          feedback = lowB + (highB << 8);
          break;
        }
        feedback = -1;
        c=c+1;
      }
    }
    if (!(feedback > target + 60 || feedback < target-60)) {
      pc.printf("\t\tdone feedback=%i\r\n", feedback);
      done = 1;
      break;
    } else if (prevfeedback == feedback) {
      break;
    }
    wait(1);
    fcount = fcount + 1;
    prevfeedback = feedback;
  } /* while fcount < 20 */
 } /* while !done */
 pc.printf("\tX target=%i, feedback=%i\r\n", target, feedback);
 return 1;
}
```

```
int moveLinActy(int position) {
   int target, feedback, prevfeedback, fcount, lowB, highB, c;
   int done=0;
   int timeout = 50;
   while (!done) {
      flushSerialBufferY();
      target = -9999;
      feedback = -1;
      while (1) {
         if (linacty.writeable()) {
            linacty.putc(0xC0 + (position & 0x1F));
            break;
         }
         pc.printf("\t\t\tpos1 putc while loop\r\n");
      }
      while (1) {
         if (linacty.writeable()) {
            linacty.putc((position >> 5) & 0x7F);
            break;
         }
         pc.printf("\t\t\tpos2 putc while loop\r\n");
      }

      while (1) {
         if (linacty.writeable()) {
            linacty.putc(0xA3);
            break;
         }
         pc.printf("\t\t\tA3 putc while loop\r\n");
      }
      lowB = 0;
      highB=0;
      c=0;
      while (c<timeout) {
         if (linacty.readable() && lowB == 0) {
            lowB = linacty.getc();
         } else if (linacty.readable()) {
            highB = linacty.getc();
            target = lowB + (highB << 8);
```

```
        break;
    }
    target = -9999;
    c=c+1;
    pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
    flushSerialBufferY();
    while (1) {
        if (linacty.writeable()) {
            linacty.putc(0xA3);
            break;
        }
        pc.printf("\t\t\tA3over putc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linacty.readable() && lowB == 0) {
            lowB = linacty.getc();
        } else if (linacty.readable()) {
            highB = linacty.getc();
            target = lowB + (highB << 8);
            break;
        }
        target = -9999;
        c=c+1;
    }
}
pc.printf("\ttarget=%i\r\n",target);
if (target == -9999)
    continue;
wait(1);
fcount = 0;
prevfeedback=-1;
while (fcount < 20) {
    while (1) {
        if (linacty.writeable()) {
```

```
        linacty.putc(0xA7);
        break;
    }
    pc.printf("\t\t\tputc while loop\r\n");
}
lowB = 0;
highB=0;
c=0;
while (c<timeout) {
    if (linacty.readable() && lowB == 0) {
        lowB = linacty.getc();
    } else if (linacty.readable()) {
        highB = linacty.getc();
        feedback = lowB + (highB << 8);
        break;
    }
    feedback = -1;
    c=c+1;
}
while (feedback>4095) {
    flushSerialBufferY();
    while (1) {
        if (linacty.writeable()) {
            linacty.putc(0xA7);
            break;
        }
        pc.printf("\t\t\tputc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linacty.readable() && lowB == 0) {
            lowB = linacty.getc();
        } else if (linacty.readable()) {
            highB = linacty.getc();
            feedback = lowB + (highB << 8);
            break;
        }
```

```
               feedback = -1;
               c=c+1;
            }
         }
         if (!(feedback > target + 60 || feedback < target-60)) {
            pc.printf("\t\tdone feedback=%i\r\n", feedback);
            done = 1;
            break;
         } else if (prevfeedback == feedback) {
            break;
         }
         wait(1);
         fcount = fcount + 1;
         prevfeedback = feedback;
      } /* while fcount < 20 */
   } /* while !done */
   pc.printf("\tY target=%i, feedback=%i\r\n", target, feedback);
   return 1;
}

int moveLinActz(int position) {
   int target, feedback, prevfeedback, fcount, lowB, highB, c;
   int done=0;
   int timeout = 200;
   while (!done) {
      flushSerialBufferZ();
      target = -9999;
      feedback = -1;
      while (1) {
         if (linactz.writeable()) {
            linactz.putc(0xC0 + (position & 0x1F));
            break;
         }
         pc.printf("\t\t\tpos1 putc while loop\r\n");
      }
      while (1) {
         if (linactz.writeable()) {
            linactz.putc((position >> 5) & 0x7F);
            break;
```

```
    }
    pc.printf("\t\t\tpos2 putc while loop\r\n");
}

while (1) {
    if (linactz.writeable()) {
        linactz.putc(0xA3);
        break;
    }
    pc.printf("\t\t\tA3 putc while loop\r\n");
}
lowB = 0;
highB=0;
c=0;
while (c<timeout) {
    if (linactz.readable() && lowB == 0) {
        lowB = linactz.getc();
    } else if (linactz.readable()) {
        highB = linactz.getc();
        target = lowB + (highB << 8);
        break;
    }
    target = -9999;
    c=c+1;
    pc.printf("c=%i, timeout=%i\r\n", c, timeout);
}
while (target>4095) {
    flushSerialBufferZ();
    while (1) {
        if (linactz.writeable()) {
            linactz.putc(0xA3);
            break;
        }
        pc.printf("\t\t\tA3over putc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linactz.readable() && lowB == 0) {
            lowB = linactz.getc();
        } else if (linactz.readable()) {
            highB = linactz.getc();
            target = lowB + (highB << 8);
            break;
        }
        target = -9999;
        c=c+1;
        pc.printf("c=%i, timeout=%i\r\n", c, timeout);
    }
}
pc.printf("\ttarget=%i\r\n",target);
if (target == -9999)
    continue;
wait(1);
fcount = 0;
prevfeedback=-1;
while (fcount < 20) {
    while (1) {
        if (linactz.writeable()) {
            linactz.putc(0xA7);
            break;
        }
        pc.printf("\t\t\tputc while loop\r\n");
    }
    lowB = 0;
    highB=0;
    c=0;
    while (c<timeout) {
        if (linactz.readable() && lowB == 0) {
            lowB = linactz.getc();
        } else if (linactz.readable()) {
            highB = linactz.getc();
            feedback = lowB + (highB << 8);
            break;
        }
        feedback = -1;
        c=c+1;
        }
        while (feedback>4095) {
          flushSerialBufferZ();
          while (1) {
            if (linactz.writeable()) {
              linactz.putc(0xA7);
              break;
            }
            pc.printf("\t\t\tputc while loop\r\n");
          }
          lowB = 0;
          highB=0;
          c=0;
          while (c<timeout) {
            if (linactz.readable() && lowB == 0) {
              lowB = linactz.getc();
            } else if (linactz.readable()) {
              highB = linactz.getc();
              feedback = lowB + (highB << 8);
              break;
            }
            feedback = -1;
            c=c+1;
          }
        }
        if (!(feedback > target + 60 || feedback < target-60)) {
          pc.printf("\t\tdone feedback=%i\r\n", feedback);
          done = 1;
          break;
        } else if (prevfeedback == feedback) {
          break;
        }
        wait(1);
        fcount = fcount + 1;
        prevfeedback = feedback;
      } /* while fcount < 20 */
    } /* while !done */
    pc.printf("\tZ target=%i, feedback=%i\r\n", target, feedback);
    return 1;
```

```
}
```

(I had several other files with code that I don’t have anymore because I switched to another project design, in addition some of the code attached with through several revisions … if this code is needed I can look and see if I can find it in one of my backups)