Non-Contact Respiration Monitoring

Oklahoma State University

ECEN 4024: Capstone Design
Spring 2021

Andrew Fry, Hasan Hamoud,
Brenden Martin, Kyle Roth

Table of Contents

Responsibility Breakdown

Student	Responsibility
Andrew Fry *Andrew.fry@okstate.edu*	Motor Control UI Programming
Hasan Hamoud *hasan.hamoud@okstate.edu*	Recorder Hardware Assembly/Design Actuator Design
Brenden Martin *brenden.martin@okstate.edu*	Hardware Assembly/Design Alternative Sensor Interfacing Editor
Kyle Roth *Kyle.roth@okstate.edu*	Point of Contact Physical Design Sensor Interfacing

Background

Respiration rate is a valuable indicator of many respiratory ailments. However, due to the difficulty and time consumption associated with taking respiration rate manually—especially with children—it is often forgone. Non-contact systems for monitoring respiration increase the likelihood of these measurements actually being taken. Determination of respiration rate in a timely and automatic manner frees medical professionals to complete tasks not yet realizable by machine. Another advantage of using an automated system is the richness of the data. When respiration rate is taken manually (counting breaths over a period of time), only the rate is obtained. With a monitoring system in play, the shape and extent of the breathing can also be readily observed, providing greater insight into the condition of the patient. Such systems have already proven their usefulness in the field, but there are some concerns regarding the existing solutions that the parent project of our capstone design seeks to address.

Parent Project

Health monitoring devices are a nearly $60 billion industry. The goal of the parent project (NSF CNS 2008556) is to evaluate and lay a foundation for a new method for non-contact respiration and heart rate monitoring with non-coherent light-wave sensing (LWS) technology. Existing methods use approaches based on radio frequency (radar) and visual methods (imaging with cameras). The argument against camera-based sensing lies in the privacy concerns associated with this method. On the other hand, the RF-based method raises concerns of electromagnetic interference with other nearby medical instruments. To test the LWS devices, a suitable measurement setup was required, which thereby motivated our design project.

Breathing Phantom

The goal of our work was to create a breathing-motion-replicating dummy (sometimes called a breathing phantom) to be used in the testing and calibration of the aforementioned non-contact respiration monitoring apparatuses. Usage of such a device has two distinct advantages over human subjects: no special permissions must be obtained to perform measurements on the phantom and the breathing patterns can be precisely controlled. This allows for better understanding of the behavior of the instrumentation.

Design

The design of the breathing phantom was split into a handful of sub-elements, which are detailed below.

DESIGN ELEMENTS:

- Endoskeleton
  - Actuators
    - Ribs
    - Upper Body

- Skin
- Microcontroller
- Wiring Diagram
- Software
  - Control System
  - Data Acquisition
  - Graphical User Interface

Following, we discuss each of these design components in greater depth.

ENDOSKELETON

The endoskeleton of the breathing phantom is a primarily 3D-printed structure designed in SolidWorks mounted to a tripod for height adjustment. The structure is smooth and round to approximately the shape of a chest cavity. The endoskeleton is comprised of a back panel which is fixed to a base connected to the tripod. The base serves as a mounting point for the Raspberry Pi and the other wiring. Attached to the top of the back plate is an overhanging pivot about which the chest plate rotates. The back panel and the overhanging pivot each possess a platform for attaching the actuators. The rib plate is stabilized and restricted to linear motion by the presence of four linear bearings, which enable the rib plate to freely move in and out.

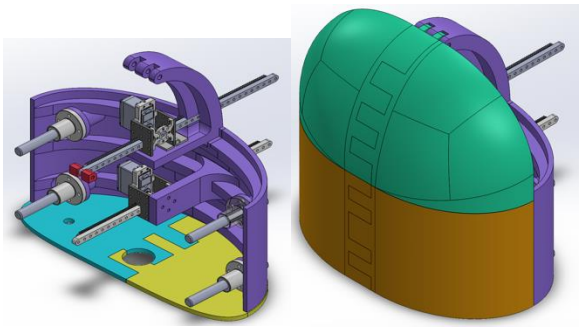


Figure 1: Endoskeleton Design

The plastic segments of the endoskeleton (base, back plate, chest plate, and rib plate) were printed on the CraftBot 2 using PLA filament. Because of the size limitations of the print beds, the individual segments had to be further divided and required the addition of finger joints to help align and connect the finished parts.

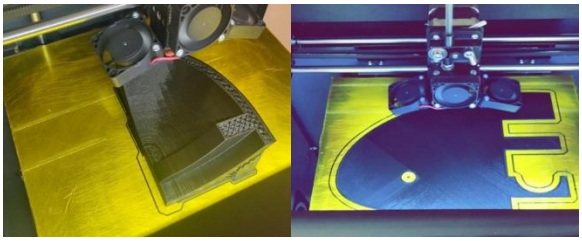


Figure 2: 3D-Printing

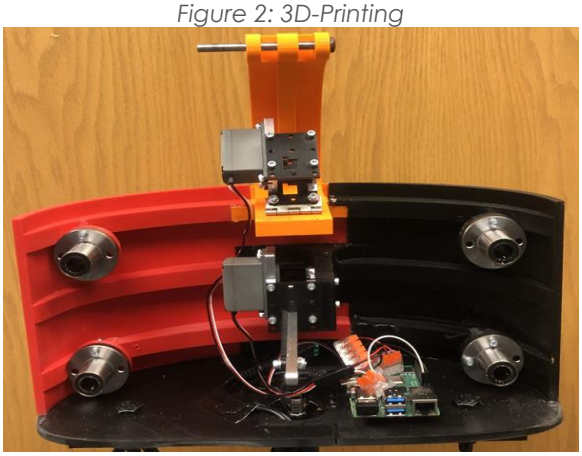


Figure 3: Endoskeleton Interior

Figure 4: PLA Base

Figure 5: Assembled Mechanism

ACTUATORS

Several different actuation methods were considered for producing the desired motions: piston style using DC motors, rack & pinion using servos, and pneumatic cylinders. We decided not to pursue pneumatics because of the difficulty to accurately control them. There were several key motor-actuator requirements to be considered, predominantly the force, linear velocity, and full travel of the actuator. The upper bounds of human respiration are around 30 millimeters of travel (during deep breathing) and the maximum rate is upwards of 55 breaths per minute (after heavy exercise). From this information, we knew we needed to ensure the actuator could both extend at least 30 millimeters and have a linear velocity of at least 60 millimeters a second (one full in-out cycle per second equates to 60 breaths per minute. None of the piston style kits we found possessed both a long enough stroke and high enough speed at the same time, so in order to use this actuation style, we would have needed to build our own. The piston kit would also have been far less direct to control, since we would have to add an encoder, and the rotation-extension is a nonlinear function. The 2000 Series motor was chosen in conjunction with rack & pinion style, meeting our travel requirement, and well exceeding the necessary linear velocity. Motor calculations & details are shown in the table below.

Motor	1501MG	2000 Series	AX-18A	AX-A12	Robotis
Deg Throw	200	300	360	360	300
Torque (oz-in)	240	65	254.901	212	55.29
RPM	71	230	97	59	114
Price	$19.95	$31.99	$94.90	$44.90	$21.90
Linear Velocity (mm/s)	47.2129016	152.943202	64.5021332	39.2332563	75.8066307
Full Travel (mm)	22.1656815	33.2485222	39.8982267	39.8982267	33.2485222
Gear/Rail Style					
Radius (in)	0.25	0.25	0.25	0.25	0.25
Force (lbs)	60	16.25	63.72525	53	13.8225
Piston Style					
Force (lbs) full rotation	10	2.70833333	10.620875	8.83333333	2.30375
Force (lbs) half rotation	5	1.35416667	5.3104375	4.41666667	1.151875

Figure 6: Motor Calculations Table

Equations used for motor calculation:

Linear Velocity: $(\frac{RPM}{60}) \times (Radius(in) \times 2\pi) \times 25.4(mm)$

Full Travel: $(\frac{Deg\ Throw}{360}) \times (Radius(in) \times 2\pi) \times 25.4(mm)$

Force: $(\frac{Torque(oz-in)}{Radius(in)}) \times 0.0625(lb)$

Figure 7: 2000 Series Super Speed Servo

The 2000 Series Super Speed Servo is a standard size servo that is driven with a pulse width modulation (PWM) signal from a microcontroller and has a supply voltage range of 4.8-7.4 V. This servo's no-load speed can reach up to 230 RPM. See Appendix C for more servo specifications.

The rack and pinion kit we selected is called a single perpendicular gear rack and was found at Servocity, a company specializing in servos, actuator kits, and associated hardware. When purchased altogether, the selected kit includes a Hitec HS-785HB servo, so it was necessary to purchase each part alone along with our chosen 2000 Series Super Speed Servo. The parts comprising the kit are:

- (1) Standard Servo Plate B
- (2) Beam Bracket
- (1) Side Tapped Pattern Mount A
- (25 pack) 1/4 Zinc-Plated Socket Head Machine Screws
- (25 pack) 5/16 Zinc-Plated Socket Head Machine Screws
- (25 pack) 5/8 Zinc-Plated Socket Head Machine Screws
- (25 pack) 6-32 Nylock Nuts
- (25 pack) #6 Undersized Washers
- (1) 32P, 16 Tooth, 25T 3F Spline Servo Mount Gear (Metal)
- (1) Actobotics Beam Gear Rack (32 Pitch, Acetal)
- (1) Aluminum Flat Beam (33 Hole, 12.32" Length)

A two-actuator design was selected (each rack & pinion), one for each of the primary manifestations of motion caused by respiration: the outward movement of the ribs and the upward motion of the chest. The core of each actuator is identical, varying only in means of affixture. Both rails are attached to their respective actuation plates via a pin and slot for easy removal when access to the interior is required.

For the lower plate, the actuator is mounted to the platform of the back panel aligned with and centered between the linear bearings. It was necessary because of the rotating motion and the position of the actuator in the chest piece, to add an additional degree of freedom. This

was accomplished using a hinge coupling the servo bracket to the mounting platform.

SKIN

To accurately replicate the movements of the human body, we used a flexible membrane as a covering of the internal armature. This skin aids in distribution of the movement of the actuation plates smoothly across the surface of the breathing phantom. We were able to locate a fairly realistic silicone torso being sold as a costuming element online. The thickness of the skin proved to be far greater than our suspicions, causing more tension than expected on the internal workings. For this reason, and to better adjust the fit of the skin, we made an incision from neck to waist down the middle of the back and added grommets down each side. The grommets were laced like a corset using a length of paracord.

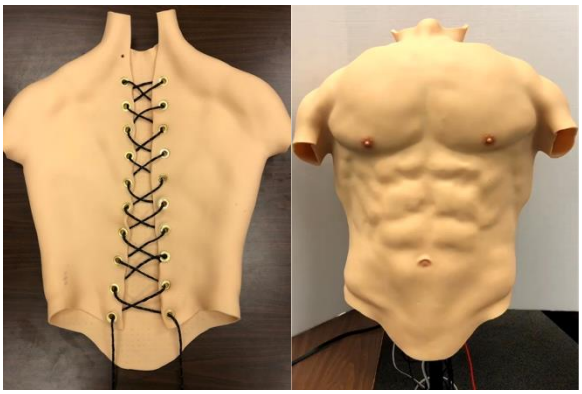


Figure 8: Front & Back View of The Skin

MICROCONTROLLER

To control the actuators in the dummy, a microcontroller was used. This was to allow for greater variability and control of breathing patterns than a simple speed controller could provide. Deliberations were made to decide between the Raspberry Pi and the Arduino as the best microcontroller for this application. The Arduino has far less overhead, making it generally more ideal for simple embedded tasks. This initially suggested that it could be the correct choice. However, we elected to use the Raspberry Pi because of its onboard memory and remote accessibility. Using the Arduino, to add or update breathing patterns, it would be necessary to either reprogram the device each time or to have designed an interface for entering and storing new patterns in an offboard memory. The Raspberry Pi, on the other hand, can be interfaced with over network using a program such as VNC or TeamViewer or directly observed by attaching a standard computer monitor and peripherals. Our hope was that this would provide the necessary means to create a simple interface for altering and switching breathing files that would not necessarily require the user to physically access the microcontroller in the breathing phantom nor recompile the program. Use of the Raspberry Pi also makes it possible to keep a copy of the entire

documentation for the device onboard, making it readily available wherever the phantom is taken.

During the programming phase, we discovered some unexpected challenges with the Raspberry Pi. A more involved description of the software iterations and accompanying issues can be found in the software section, but the essence of the problem was that the Graphical User Interface (GUI) was bogging down the Raspberry Pi to the extent that it could no longer drive the actuators smoothly while also plotting data. This eventually led to a reevaluation of the Raspberry Pi as the optimal microcontroller. We came to realize that the features we desired from the Pi could be realized from an external computer, and further, that a microcontroller within the phantom could be made responsible for dispatching instructions to the actuators exclusively, while the control interface runs on the external device. For this alternative architecture, an Arduino became the better option due to its ease of serial communication. In this final design variant, the external computer sends the position information to the Arduino over serial via USB cable. The Arduino converts the information from the computer into a PWM signal to control the actuators. The computer is also responsible for reading sensor data and plotting the target motion and measured data.

WIRING

The wiring within the dummy has been implemented utilizing lever nut terminals. These terminals maintain an exceptionally strong grasp upon the wires but facilitate easy troubleshooting and part replacement through easy wire removal. The proper pins to use can be found in the code documentation (and edited if a user so desires), but also seen in the schematic below.

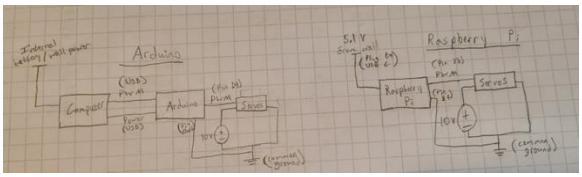


Figure 9: Schematic

SOFTWARE

- *Motor Control Text Interface*

The first control software programmed was a simple text-based interface written in C++, which was navigated by entering corresponding instruction and option numbers. This enabled us to evaluate the performance of the actuators but would be a nonideal interface for the end-user.

- *Motor Control GUI*

In the interest of user-friendliness, we began to work on a GUI. Sliders were now used to set the rate, offset, and amplitude of the actuation. A

dropdown menu was used for waveform selection. Shown below is the final version of this option mentioned in the Setup section.

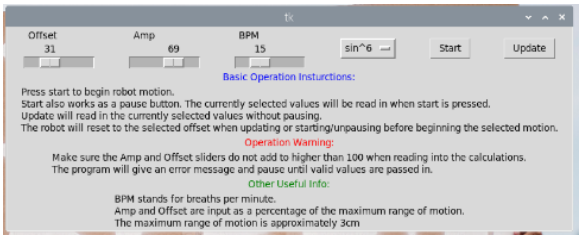


Figure 10: Motor Control Menu

- *GUI with Realtime Data Plots on Raspberry Pi*

Once the motor control was implemented in the GUI, we began work to add a display for the comparison of the commanded motions from the Raspberry Pi and the measured motions from a Time of Flight (ToF) sensor.

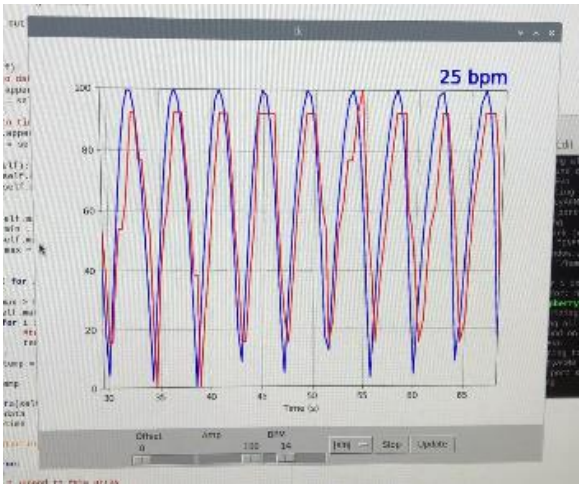


Figure 11: Real Time Data Plot (blue = sent, red = read)

At this point, it became evident that the Raspberry Pi was struggling to manage the real-time plotting while controlling the motors. Attempts at multithreading were made, but though irreplicable errors (the biggest being a memory access error) and lack of background knowledge made this a highly difficult challenge. Rather than scramble to accrue years of programming expertise over the course of a few weeks, we devised an alternative solution using a microcontroller as a slave to an external computer.

- *GUI with Realtime Data Plots on External Computer*

The final version of the control software is essentially identical to the prior except that it is run on an external computer and therefore runs with minimal jitter. Jitter is a term used to describe the time between calculations of the location information from the patterns to the servos. A smaller time between calculations means there are more points in a period of the waveform, so what you see is less jitter. Migrating the software to a desktop or laptop computer was a fairly easy task as that the graphical versions of the interface were programmed in Python. For its tradeoffs, being written in Python does add considerable versatility to the program in that it can run on Linux, MacOS, or Windows so long as the host machine has Python3 installed.

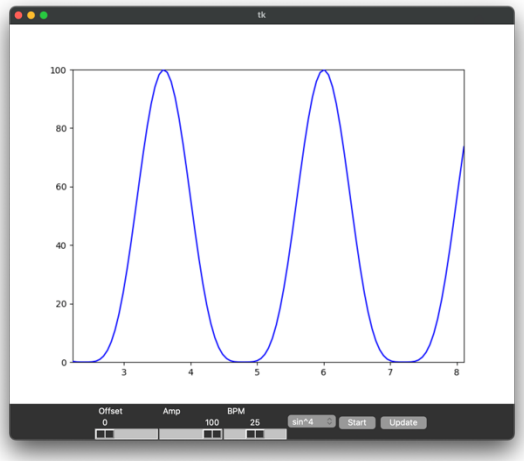


Figure 12: GUI for the Master-Slave Setup

Expandability is key for the usefulness of the breathing phantom in a research environment. The code has been written generally, such that it can be adapted to work with new sensors by simply changing the function which retrieves the sensor value. Assuming that the new data retrieval function has already been written, the change is merely to include its library at the top of the program and swap the new one out for the previous data retrieval function. Though the program currently includes sinusoidal functions only, piecewise functions can also be added using the same framework as the already implemented patterns. Further details for operating and altering the code can be found in its dedicated documentation and in the operating instructions provided in the appendix.

Bill of Materials and Budget Summary

For the budget and bill of materials, we had four total orders from several websites and companies. We were given $1400 budget to spend on this project from the Electrical and Computer Engineering department. In the first week of the semester, we had our first meeting discussing the initial parts list and budget for the materials that we need to accomplish this mission. For the initial parts list, the cost was $1090, and we knew that this is the minimum cost to do project, but at the same time we knew that the range of the price to make this robot would be between $1090-$1400. Planning ahead and working on the list we needed to order in time paid off especially during the extreme weather condition and all of the shipment delays. The first order took place in week 4. We ordered actuators and the linear motion kit, lighting stand, power supply, Raspberry pi 4 & its power supply, fake muscles, and male/female jumper wires. We sent the second list of the rest of the actuators in week 5 since they never got ordered in the first order. The third order was in week 7, and this time ordering some standard servo plates and gears for the servos' kits. Our fourth order took place in week 10 to purchase linear ball bearings & shafts, lever wire connectors, and a PCB kit. We also had to go to Walmart in week 14 to different sizes/colors T-shirts for the dummy as well as a pillow for the abdominal part of it. Overall, we spent $1303.21 of our budget, coming in well under our $1400 limit. Final cost and parts list are shown in the figure below.

Item	Brand	Quantity	Price/one	Total Price	Number	Link
2000 Series Dual Mode Servo (25-4, Super Speed)	ServoCity	6	$ 31.99	$ 191.94	2000-0025-0004	2000 Series Dual Mode Servo
Standard Servo Plate A	ServoCity	6	$ 4.99	$ 29.94	575112	Servo Plate
Standard Servo Plate B	ServoCity	6	$ 6.99	$ 41.94	575124	Servo Plate B
Beam Bracket U (2 pack)	ServoCity	6	$ 2.49	$ 14.94	585658	Beam Bracket U
90° Dual Side Mount B (2 pack)	ServoCity	6	$ 5.99	$ 35.94	585508	90° Dual Side Mount B
Side Tapped Pattern Mount A	ServoCity	6	$ 4.99	$ 29.94	545420	Side Tapped Pattern Mount A
Side Tapped Pattern Mount B	ServoCity	6	$ 4.99	$ 29.94	545424	Side Tapped Pattern Mount B
6-32 x 0.250" (1/4) Zinc-Plated Socket Head Machine Screw (25 pack)	ServoCity	3	$ 1.89	$ 5.67	632106	6-32 x 0.250" (1/4) Screw
6-32 x 0.3125" (5/16) Zinc-Plated Socket Head Machine Screw (25 Pack)	ServoCity	1	$ 1.99	$ 1.99	632108	6-32 x 0.3125" (5/16) Screw
6-32 x 0.625" (5/8) Zinc-Plated Socket Head Machine Screw (25 pack)	ServoCity	1	$ 2.79	$ 2.79	632118	6-32 x 0.625" (5/8) Screw
6-32 Nylock Nuts Pack (25 pack)	ServoCity	1	$ 1.89	$ 1.89	632142	6-32 Nylock Nuts Pack
#6 Undersized Washers (25 pack)	ServoCity	2	$ 0.99	$ 1.98	632144	#6 Undersized Washers
32P, 16 Tooth, 24T C1 Spline Servo Mount Gear (Metal)	ServoCity	6	$ 14.99	$ 89.94	615278	Spline Servo Mount Gear (Metal)
32P, 16 Tooth, 25T 3F Spline Servo Mount Gears (Metal)	ServoCity	6	$ 14.99	$ 89.94	615298	32p-16-tooth-25t-3f-spline
32P, 16 Tooth, 25T 3F Spline Servo Mount Gear (Acetyl)	ServoCity	6	$ 3.96	$ 23.76	RSA32-2F5-16	32P, 16 Tooth, 25T 3F Spline
Actobotics Beam Gear Rack (32 Pitch, Acetal)	ServoCity	6	$ 5.99	$ 35.94	615410	Actobotics Beam Gear Rack
Aluminum Flat Beam (33 Hole, 12.32" Length) - 2 Pack	ServoCity	3	$ 13.99	$ 41.97	585423	Aluminum Flat Beam
False Muscles	Ebay	1	$ 159.00	$ 159.00		Muscles
Lighting Stand	Sweetwater	1	$ 38.40	$ 38.40		Lighting Stand
Raspberry Pi 4 8 Gb	adafruit	2	$ 75.00	$ 150.00	4564	Raspberry Pi 4
Raspberry Pi Power Supply	adafruit	1	$ 7.95	$ 7.95	4298	Raspberry Pi Supply
Power Supply	Amazon	2	$ 20.00	$ 40.00		Power Supply
1/2" Flange-Mounted Linear Ball Bearing	McMASTER-CARR	4	$ 26.87	$ 107.48	6483K53	Ball Bearing
1/2" Linear Motion Shaft	McMASTER-CARR	4	$ 5.19	$ 20.76	6061K428	Linear motion shaft
Plastic Hinges without Holes	McMASTER-CARR	3	$ 1.02	$ 3.06	1118SA141	Plastic Hinges
Lever Wire Connectors	Amazon	1	$ 19.99	$ 19.99		Connectors
Smraza 100pcs Double Sided PCB Board Kit	Amazon	1	$ 14.29	$ 14.29		PCBs kit
T-shirts different colors/sizes (Pack)	Walmart	2	$ 19.98	$ 39.96		
T-shirts white (Pack)	Walmart	1	$ 14.16	$ 14.16		
Pillow	Walmart	1	$ 3.72	$ 3.72		
Assorted Jumper Cables (50cm)	Amazon	1	$ 13.99	$ 13.99		Jumper Cables
		Total		$ 1,303.21		

Figure 13: Parts List & Total Cost

Reflections

ANDREW

Ideally C++ would have a GUI library that is efficient and easy to use. Tkinter is great for GUI programming, but Python cannot keep up. I suppose then that I would have preferred to delve into the more difficult C++ GUI programming early on then attempted plotting along with that. I do not know if that would solve the problem in terms of plotting time, but then I guess the same issues would result regardless if not. I believe having the Raspberry Pi upload its information to a database would be the best for ease of use in terms of maintaining the simple user interface while allowing an experimenter to have access to the data from anywhere they wanted. As for the way the calculations are done, it could have been set up in a way that calculations are done for one period at startup and stored in an array for every function you wanted then looped through when called. This would allow for completely arbitrary functions. With the current framework, a future user could easily set something like this up if they wished to use arbitrary functions. Also, the Raspberry Pi does the mathematical calculations so quickly that this would not be a reasonable method to save calculation time. The main issue was how long it takes to plot data. Luckily, switching to the Arduino made that much better (still not as smooth as with no plotting at all), but it would be nice to have a system that was completely self-contained while meeting all the desires of the researchers.

HASAN

The current design of the dummy is very useful for research purposes and can be a reference for future medical breathing phantom devices. For the physical design, 3d printing option is not a bad idea for this kind of project, however it took a lot of time and effort from designing the module in SolidWorks to printing it on CraftBot 2 using PLA filament. The first design that we discussed as a team was to create our robot using a flexible thin-long metal sheets as ribs and aluminum plates that act as lungs with a total of six actuators to provide a stable and smooth linear motion. The current actuation design with our servos can handle up to 16.25 lb of force. That is more than enough to us, but I would recommend searching for a more expensive servo that has a reasonably high speed and can handle more force when using heavy materials to also avoid the overload heat in the servo. For the software part, we have a variety of breathing patterns that is helpful for the researcher who wants to collect data of a human-like breathing rates such as $|\sin(wt)|$, $\sin^2(wt)$, $\sin^4(wt)$, and $\sin^6(wt)$ with a user-friendly motor

control GUI. Adding more functions such as exponentials would also let the user to have a wider range of patterns to study. Overall, I believe

BRENDEN

My preferred design would use flexible metal ribs being deflected by actuator pairs (pairs for increased force). Each rib would be vertically adjustable along a rail. The ribs would be tunable, but identical. They would be designed from simple extruded forms to make their manufacture at the physics machine shop a straightforward task. I would use aluminum for the frame and spring steel for the deflecting plates. I would gladly employ the same approximate actuators, although I would attempt to find a servo with slightly greater force output to make it more rugged. At each the top and bottom of the vertical mounting rail would be a metal pipe ring upon which the silicone skin could be fastened. Individual adjustment of the ribs would be used to set the body shape and help fit it to the skin. I would keep the master-slave configuration between the microcontroller and an external computer, but rather than implement breathing patterns using functions, I would use a file-based method in which a completely arbitrary breathing pattern can be defined. A breathing description file would have a column for each actuator, and time would be the vertical axis. Waveforms would be of a fixed length and be normalized to the same frequency. I would select a frequency of around 0.1 Hertz (slow breathing) so that there are plenty of data points for smooth motion even at the low end of respiration rates. For the data monitoring software, I would use the serial and plotting capabilities of MATLAB, which we have reason to suspect may be better able to update plots quickly. This would also allow easy integration with other laboratory code already written for the MATLAB environment.

KYLE

We spent a lot of time discussing the physical design and construction of the

Breathing Phantom and lot of preliminary design work was done in SolidWorks to help cull bad or overly complex designs. Then, we 3D printed a prototype of the chosen design. This prototype helped to teach us many things about the dummy, and it showed a few places it could be improved. First, the frame of the dummy was slightly too large for the skin to fit nicely over it. Shrinking the overall dimensions of the frame would allow ‘dressing’ the robot to be easier and reduce load on the servos. Second, the chest piece doesn’t appear to fill out the dummy well. I’m quite certain this is because there are no shoulders on the dummy. While this shouldn’t affect the sensor data horribly, adding shoulders would make the dummy more lifelike. Third, the lower plate region could stand to be slightly taller. While there is motion in the lower stomach region, extending this plate just 2-3 inches would help exaggerate this motion. Finally, when testing the prototype, there was a decent amount of strain put on the servos. We had only one servo actuating each plate. The

design could be altered to allow multiple servos on each plate, distributing the load amongst them all. Given all of these, the Breathing Phantom produces very lifelike breathing motion. With these design considerations in mind, I would slightly modify the current design and get the dummy built from a more rigid material, maybe aluminum.

Appendix

Operating Instructions

SETUP

The control software is written in Python 3 and therefore, a valid installation of Python 3 is required to run the program. Complete and detailed instructions on the installation process can be found on the Python website:
https://wiki.python.org/moin/BeginnersGuide/Download

To install the additional required libraries, run the following in a command line in the software's directory:

> *pip3 install --user -r "requirements.txt"*

For completeness, we have provided several variations of the control software from throughout the design process. Following is a list of these versions and accompanying instructions:

RASPBERRY PI

Inside the 'NoGraph' folder are programs to run the breathing phantom that produce no respiration plots. *For maximally smooth actuation, it is recommended that the phantom be run with this version of the program.* To run this version, navigate in the command-line to the 'NoGraph' folder and run:

> *python3 DNFM.py*

Inside the 'RPi' folder are programs which both poll the sensor and control the phantom from the Raspberry Pi. These utilities produce a graph of the data. Because of the resultant degradation of actuation, these programs are not recommended, but are included should they become helpful.

> python3 DNFM.py does not plot either measured or target motion.
> python3 DNFM2.py plots both data sets.

Stored within the 'threading' folder are our attempts to utilize threading to smooth the motion of the phantom despite also plotting data. Though it consumed much energy and many hours, this approach unfortunately did not come to fruition. Though the Raspberry Pi has multiple cores, each core is single threaded, so this code implemented scheduling rather than true threading.

ARDUINO

Inside the 'Arduino' folder is code to control the phantom using an Arduino. The Arduino listens for serial data from an external computer over USB and converts it into a PWM signal to drive the actuators. On the external computer:

> python3 DNFMServo.py
> python3 DNFMEvo.py

This will open two separate windows. The DNFMServo window controls the phantom and the DNFMEvo window plots the sensor data. Alternatively, the DNFM.py program in the Arduino folder can be used to attempt running both plots in the same window, but is not recommended for its choppiness.

SENSOR ONLY

In the 'TeraRanger_Evo' folder, there are programs that poll the sensor only. There is a C++ program to write the data from the sensor to a Binary file. This was tested on a MacBook, however, it should run in Linux as well.

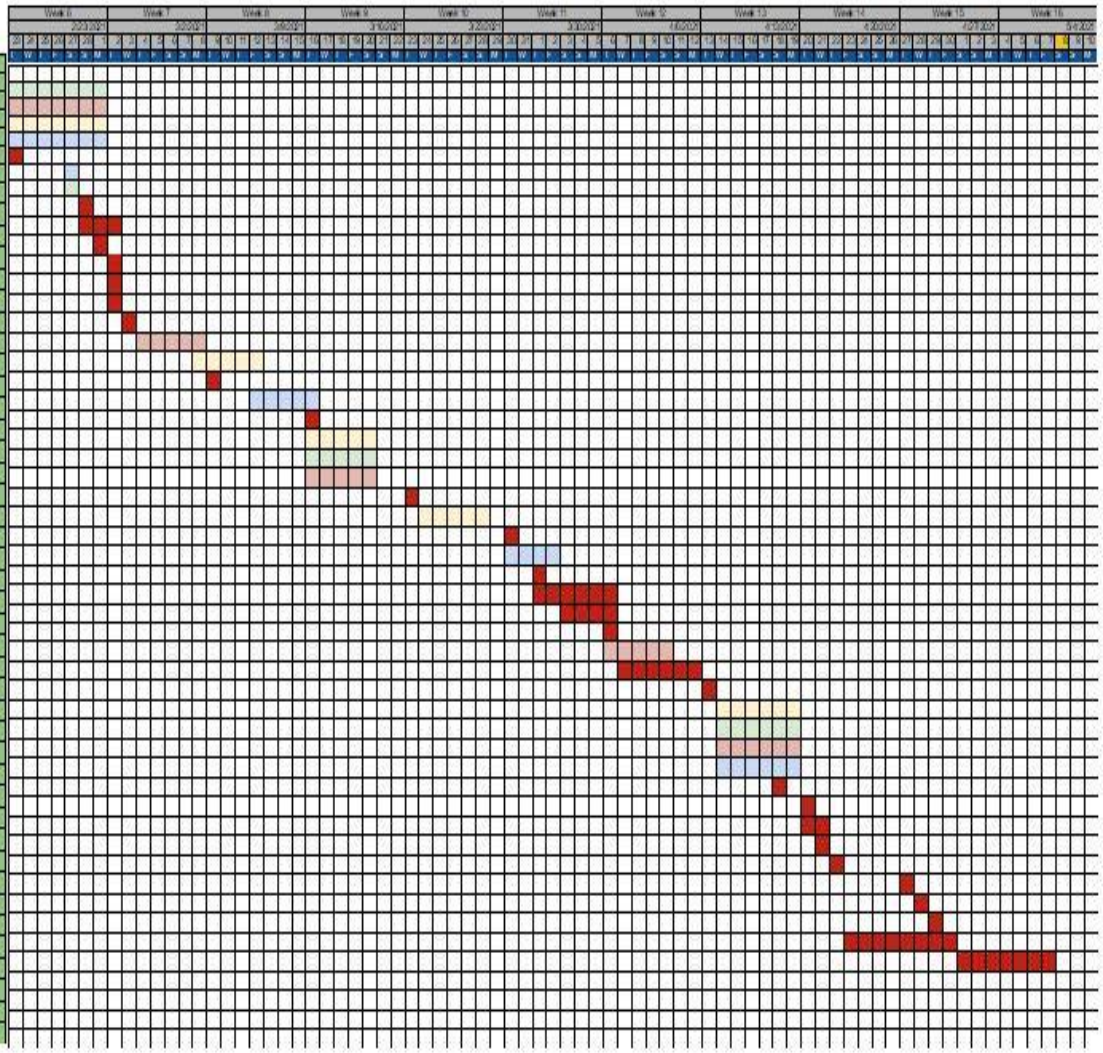
Motor Calculations

Motor	1501MG	2000 Series	AX-18A	AX-A12	Robotis	Paralax	P25	MG995R	Savox	HSR-2648CR
Deg Throw	200	300	360	360	300	360	360	120	130	360
Torque (oz-in)	240	65	254.901	212	55.29	34.7	96	140	277.7	166.64
RPM	71	230	97	59	114	120	1732	62.5	71	50
Price	$19.95	$31.99	$94.90	$44.90	$21.90	$30.01	$77.00	$19.95	$75.70	$32.99
Linear Velocity (mm/s)	47.2129016	152.943202	64.5021332	39.2332563	75.8066307	79.7964534	1151.72881	41.5606528	47.2129016	33.2485222
Full Travel (mm)	22.1656815	33.2485222	39.8982267	39.8982267	33.2485222	39.8982267	39.8982267	13.2994089	14.407693	39.8982267
Gear/Rail Style										
Radius (in)	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Force (lbs)	60	16.25	63.72525	53	13.8225	8.675	24	35	69.425	41.66
Piston Style										
Force (lbs) full rotation	10	2.70833333	10.620875	8.83333333	2.30375	1.44583333	4	5.83333333	11.5708333	6.94333333
Force (lbs) half rotation	5	1.35416667	5.3104375	4.41666667	1.151875	0.72291667	2	2.91666667	5.78541667	3.47166667

Gantt Chart

Project Title Non Contact Respiration and Heart Rates Monitoring
Project Advisors
Dr. Sabit Ekin
Dr. John O'Hara
Important Dates
Design Proposal 3rd week Complete
Prototype Demonstration 7th week Complete
Trial Project Inspection 13th week If possible
Oral Report and Demonstratior 14th week Next Goal
Final Report Due 15th week
Peer Evaluation 16th week
Group Member Color
Andrew Fry
Brenden Martin
Hasan Hamoud
Kyle Roth
Team
TASK STATUS
Meeting 1 (with advisors) Complete
Assign POC Complete
Read/Research about the project Complete
Estimate the budget & make initial parts list Complete
Meeting 2 (Team) Complete
Meeting 3 (with advisors) Complete
Meet with Hisham (discuss sensors) Complete
Read in brainstorm & find out new Complete
Meeting 4 Complete
Work on budget & parts Complete
Meeting 5 Complete
Design proposal Complete
Meeting 6 (with advisors) Complete
Assign team member (for job assignments) Complete
30 Meeting 7 Complete
Search for the perfect motor/servo Complete
Search for mechanical parts for the servo Complete
Search for mechanical parts for the servo Complete
Look for useful codes/libraries Complete
150 Sensors & real time display Complete
Meeting 8 Complete
Meeting 9 (with advisors) Complete
Search for faster servo/rack kit that fits it Complete
Search for faster servo/rack kit that fits it Complete
42 Meeting 10 Complete
Double check rack & pinion kit Complete
Double check rack & pinion kit Complete
Send parts list to Travis Complete
Pick up Lab key Complete
Mechanical equipment/tools/attachement Complete
Mechanical equipment/tools/attachement Complete
120 Ribs 3D modeling using solidworks Complete
Test and run codes on raspberry pi Complete
Meeting 11 (with advisors) Complete
Go to Lowe's to narrow down the choices Complete
Go to Lowe's to narrow down the choices Complete
Meeting 12 Complete
Search for motor control Complete
PWM raspberry pi/ motor controlling Complete
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
1/19/2021
1/26/2021
2/2/2021
2/9/2021
2/16/2021
2/23/2021

Search for motor control	Complete
PWM raspberry pi/ motor controlling	Complete
3D printing options/places	Complete
Make Hardware parts list/ order gears	Complete
Meeting 13 (with advisors)	Complete
Servo kit assembly & test mechanical design	Complete
Servo kit assembly & test mechanical design	Complete
Meeting 14	Complete
Work on PowerPoint presentation slides	Complete
Meeting 15	Complete
Meeting 16	Complete
Meeting 17 (with advisors)	Complete
Meet with Hisham	Complete
Prototype demonstration presentation (2:30)	Complete
Motor control version 1	Complete
3D printing options/ Sensor program	Complete
Meeting 18 (with advisors)	Complete
Servo kit assembly	Complete
Meeting 19 (with advisors)	Complete
Print chest/ abdomen	Complete
Search for linear bearing & data display	Complete
Add more breathing patterns	Complete
Meeting 20 (with advisors)	Complete
Print 3d design (back part)	Complete
Meeting 21 (with advisors)	Complete
Design a 3d platform (Solidworks)	Complete
Meeting 22	Complete
Start building the robot	Complete
Print robot base (3D module)	Complete
Meeting 23 (with advisors)	Complete
Work on GUI program	Complete
Purchase different sizes/colors T-shirts	Complete
Meeting 24 (with advisors)	Complete
Show real time sensor data	Complete
Work on different sensor	Complete
Inhance GUI and info showing	Complete
Work on ultrasonic sensor for measurments	Complete
Meeting 25	Complete
Meeting 26 (with advisors)	Complete
Work on PowerPoint presentation slides	Complete
Meeting 27	Complete
Oral Report and Demonstration (12:00 PM)	Complete
Meeting 28	Complete
Meeting 29	Complete
Meeting 30 (with advisors)	Complete
Final report	Complete
Peer evaluation	Complete



Servo Specifications

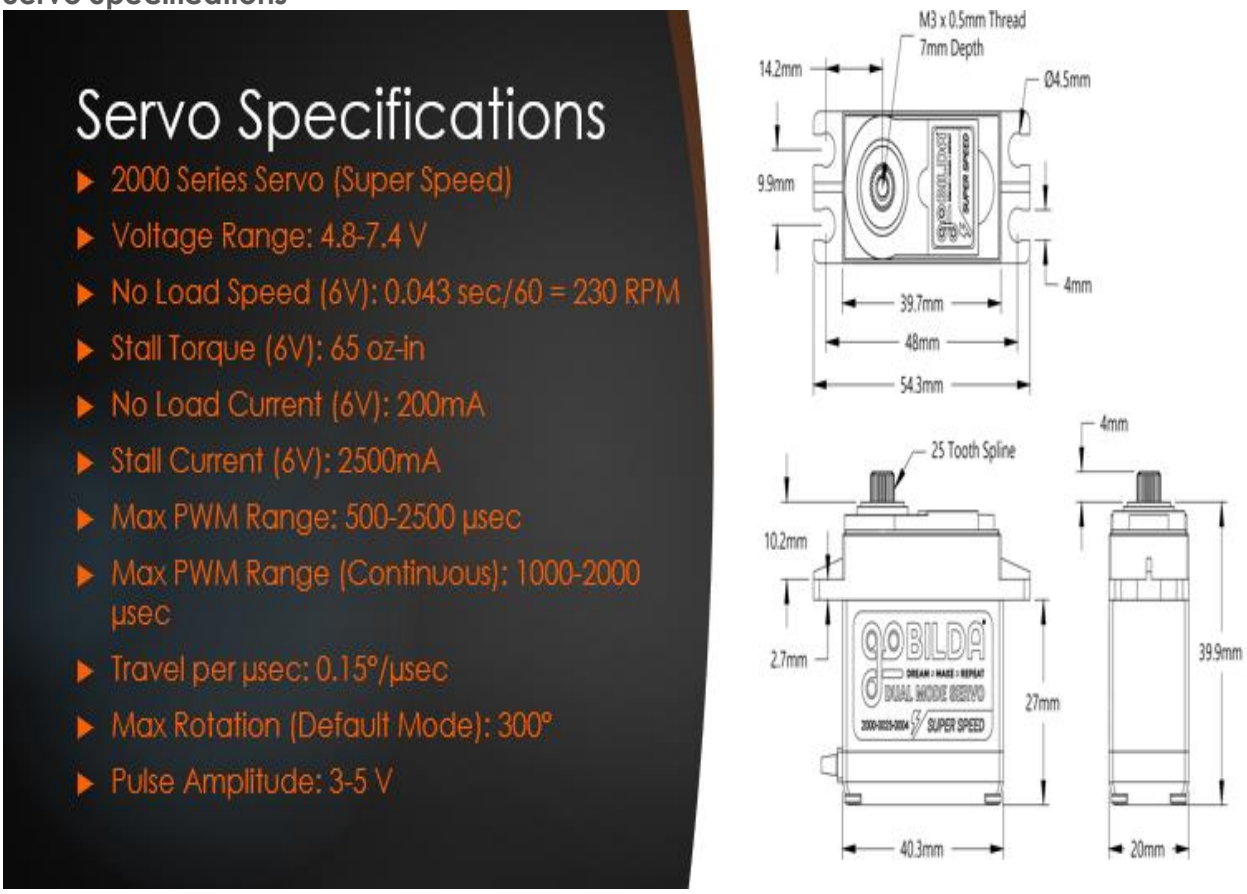


Figure 14: Servo Specifications