LEAH HARTWELL

MECHANICAL ENGINEERING STUDENT

Richmond, BC, Canada
778-833-3921
leahh222@student.ubc.ca
www.linkedin.com/in/leahhlhartwell
http://leahhartwell.tech/

CO-OP WORK EXPERIENCE

Mechanical Engineering Co-op → entrepreneurship@UBC JUL 2020 → DEC 2020

Simultaneously working at two startups within the e@UBC HATCH Accelerator Program where I am designing and building mechanical systems while learning the nuances of entrepreneurship

Mechanical Design Engineer → BIOFORM Solutions Inc. JUL 2020 → ONGOING

Startup developing a low-energy and small-footprint production platform that rapidly produces biodegradable alternatives to plastics

→ Responsible for designing and prototyping custom multi-layer nozzles, posttreatment and winding systems for production platform which will create stretch wrap and medical tubing for our pilot trial

→ Working with CTO to program the production platform control system and GUI in LabView for multi-pump operation through VFDs based on data from cameras and an inline rheometer

Hardware Engineer → Verdi Expeditions Inc. SEP 2020 → ONGOING

Startup developing a platform based around intelligent irrigation systems for efficient, climate-resilient agriculture

→ Designed, built and programmed an automated test jig in order to rigorously test solenoid valves to choose the best and most economic option to be used within each of the company's smart valves

→ Working on proprietary sensing unit to be used in new seed-round smart valves to sense flow/no-flow conditions through drip tube using a sensor and background neural network model

RECENT PROJECTS

UBC Open Robotics Design Team

Mechanical Co-Lead → Pianobot SEP 2020 → ONGOING

→ Responsible for guiding and leading junior members while overseeing all mechanical aspects of a robot which can read and play the piano at Level 4 RCM

→ Wrote a Python script which calculates force transferred from a linear actuator to the fingertip in order to find optimal dimensions for the finger design to press down piano keys

→ Currently implementing mechanisms to hit black keys and play arpeggios

Mechanical Engineer → RoboCup@Home SEP 2019 → AUG 2020

→ Designed belt-driven differential gear systems for elbow/wrist allowing for compact joints and decreasing material cost for larger carbon-fibre chassis of arms for our autonomous service robot

→ Analyzed components with FEA in Solidworks Simulations to verify that parts could withstand known forces and torques on arm in static and dynamic states

3D Print Checker → Personal Project MAY 2020 → ONGOING

→ Building a 3D printer monitoring system that notifies the user when the 3D print is failing and provides possible causes based on information and sources from the internet

→ Created multiple computer vision models with AutoML Vision/Tensorflow while training the models on web-scraped images and labelling them in LabelImg

→ Programmed 3D print failures model into a Python script with OpenCV library and ran it on a Raspberry Pi equipped with a PiCamera

K-Cup Cleaner → Personal Project APR 2020

→ Built a cleaning system for reusable Keurig cups which spins an angled cup holder using a stepper motor while water rinses coffee grounds out into a filter

→ Soldered Arduino, water pump, stepper motor driver, button, transistor, resistor, wires to PCB and programmed Arduino to control pump and motor with a button

→ Modelled and 3D printed custom nozzle, stand, water reservoir and cup holder

PeakStudy (Wolfram Research Award Winner) → cmd-f Hackathon MAR 2020

→ Placed in Top 10 out of 61 teams with my partner to create a web app using HTML/CSS/Javascript that optimizes student productivity by ranking homework importance

→ Rankings are based on type, difficulty, due date, duration and priority entered by a user and multiplied by backend weights determined by the Analytical Hierarchy Process

EDUCATION

University of British Columbia

BASc in Mechanical Engineering - Biomedical Option,
Third Year | CGPA 3.30
SEP 2019 → EXPECTED MAY 2023

Co-op: Completed 2/4 work terms; Available for 8 months beginning May 2021

Kwantlen Polytechnic University

Engineering First Year Certificate Program,
First Year | CGPA 3.89
SEP 2018 → MAY 2019

Certificate in Engineering (w/ Distinction)

TECHNICAL SKILLS

Software/Programming:

CAD/FEA/CFD
→ SolidWorks · OnShape · AutoCAD
Data Analysis
→ MATLAB · Maple · Excel
Languages
→ Python · C/C++ · HTML/CSS/Javascript
Machine Learning:
→ Tensorflow · Google Cloud Vision

Hardware:

Machining
→ Mill · Lathe · Drill · Band Saw · Water Jet · Spot Weld · 3D Print
Microcontrollers
→ Microchip PIC · Raspberry Pi · Arduino
Actuators/Sensors/Transducers
→ Pumps · Motors · Infrared Sensor · Flow Rate Sensor · Pressure Transducer
Testing/Validation
→ Circuit Analysis · Soldering · Oscilloscope · Function Generator

Design:

→ DFM/DFA · FMEA/FEA/CFD · CAD · GD&T · Prototyping · BOM Specification · LCA

INTERESTS AND ACTIVITIES

Art/Design · Entrepreneurship · Biotech · Sustainability · Machine Learning · IoT · Programming · Rugby · Taekwondo

TECHNICAL PROJECTS

Project:
Autonomous Service Robot – *7DOF Robotic Arm*

Team:
UBC Open Robotics Design Team – RoboCup@Home Arms Subteam

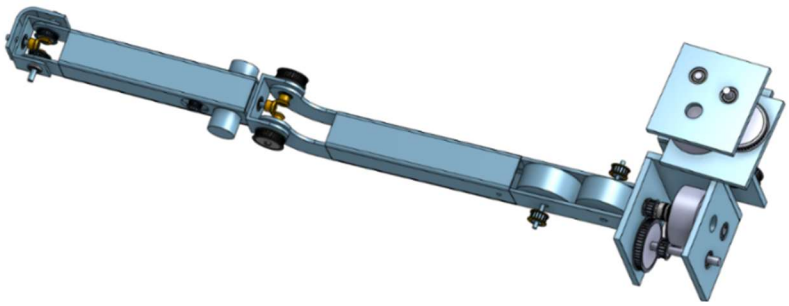
Timeline:
September 2019 – Present

Competition:
RoboCup@Home Competition 2020

Goal:

The UBC Open Robotics RoboCup@Home division's goal is to create an autonomous service robot that will be able to complete basic household tasks via voice command. For the Arms subteam, our focus is to create robotic arms able to carry up to a 3.5 kg load with minimal deflection and seven degrees of freedom. As a Mechanical Member, my goal is to design arms for the robot given the constraints by using both custom and off-the-shelf components. These constraints were based on competition requirements, cost, technical readiness, and feasibility.

Current Design:

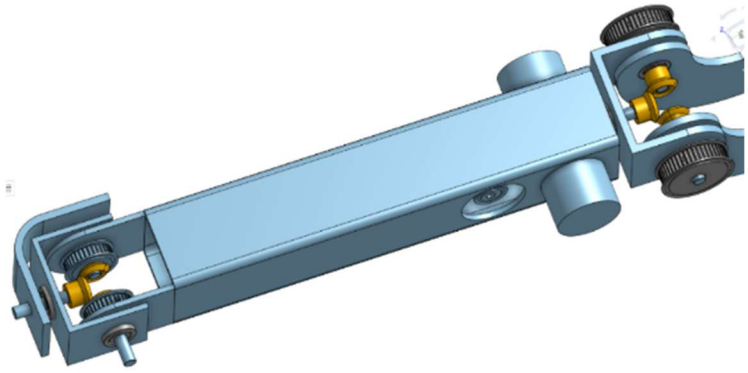


← Fully Mated CAD Assembly of Design:
2 X Differential gears at elbow and wrist for compact / spherical joints
3 X 100KV motors at shoulder with 1:10 reduction
2 X 100KV motors near shoulder with 1:5 reduction
2 X 400KV motors near elbow with 1:3 reduction
2 X carbon-fiber square tube
5 X custom aluminum brackets
reduction/transmission = GT5 timing pulley/belts

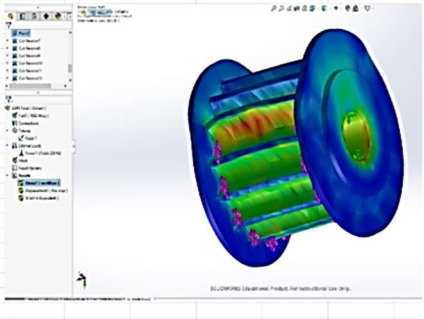
Personal Contribution:

Forearm-Wrist Design/CAD →
Worked on the forearm-wrist section of assembly which consisted of:

- Wrist transmission/reduction calculations
- Choosing best components off Misumi to match calculations
- Creating custom brackets for joint and gripper connections
- Fitting components into main carbon fibre chassis
- Mating all custom and off-the-shelf components together to display degrees of freedom



MOTOR SPECS			PULLEY SPECS	T14	T16	T32	T36	T40	T44	
Angular Velocity, [RPM]	3567		Diameter, d [cm]	2.114	2.432	4.979	5.616	6.252	6.889	
Angular Velocity, w [rad/s]	373.5353665		Radius, r [m]	0.1057	0.1216	0.24895	0.2808	0.3126	0.34445	
Power, P [W]	2671									
Torque, T [Nm]	7.150594668									
MOTOR SHAFT										
Angular Velocity, w [rad/s]	373.5353665		Force T14 [N]	67.64990225						
Torque, T [Nm]	7.150594668									
2X32:16										
SHAFT 2										
Angular Velocity, w [rad/s]	182.4539087		Force T14 [N]	120.3890927						
Torque, T [Nm]	14.63931397		Force T36 [N]	58.80423247						
SHAFT 3										
Angular Velocity, w [rad/s]	89.1198847		Force T36 [N]	120.3890927						
Torque, T [Nm]	29.97086463									
2X36:14										
SHAFT 2										
Angular Velocity, w [rad/s]	140.6078641		Force T14 [N]	179.7170535						
Torque, T [Nm]	18.99609255		Force T36 [N]	67.64990225						
SHAFT 3										
Angular Velocity, w [rad/s]	52.92824514		Force T36 [N]	179.7170535						
Torque, T [Nm]	50.46454861									



← Transmission Calculations / Finite Element Analysis
Analyzed transmission for the three different motor reduction configurations which consisted of:

- Finding specifications for motors (KV, power, rpm), smallest GT5 pulley (fits to motor shaft), multiple larger GT5 pulleys (for testing reduction ratios)
- Compiling all specifications in a spreadsheet to do reduction calculations with different reductions ratios
- Found max force and torques on weakest pulleys given best reduction ratio
- Using SolidWorks Simulations FEA on pulley to verify that max force/torque does not cause failure

Challenges:

Fitting Components → Difficult to fit multiple parts to one shaft because all of the inner diameters of each part had to match the shaft size of either the motor or the wrist. Since we wanted a compact arm, finding the smallest size of each part would be optimal. However, the smallest sizes would differ in inner diameter. This led to a need for a lot of comparisons between parts to makes sure they would work together with our preferred reduction. Fitting the systems within our carbon fibre tube was another difficulty since the first set of motors were too large to fit within the tubing. A larger reduction ratio which would be driven by a smaller motor with less torque would resolve this issue. I found that I was unable to fit a larger ratio within the tube, which pushed me to do a two-step reduction with smaller pulleys in the end, which solved my problem.

Specifying Components → Difficult because components of the failure calculations were based on initially unspecified pieces of the arm. This meant we had to estimate the unknowns during failure calculations and continuously redo them as other values continued to change.

Learned:

Engineering Is an Iterative Process → Many components rely on other, initially undetermined parts, which means estimations are important and calculations must be continuously updated as new information emerges.

Meeting Minutes / Logbook → Keeping up a record of your current components and how they factor into calculations is useful as you may leave the idea and later come back to it.

Next Steps:

Our next steps will be finishing our FEA analysis and timing belt failure calculations to ensure our arms will not break or endanger others around them. Once our PAF funding I helped apply for comes in we will be able to buy all of our off-the-shelf components and start to build the custom brackets.

Project:
Task Optimizer Web App - *PeakStudy*

Team:
Leah Hartwell, Joya Lee

Timeline:
March 2020 - Present

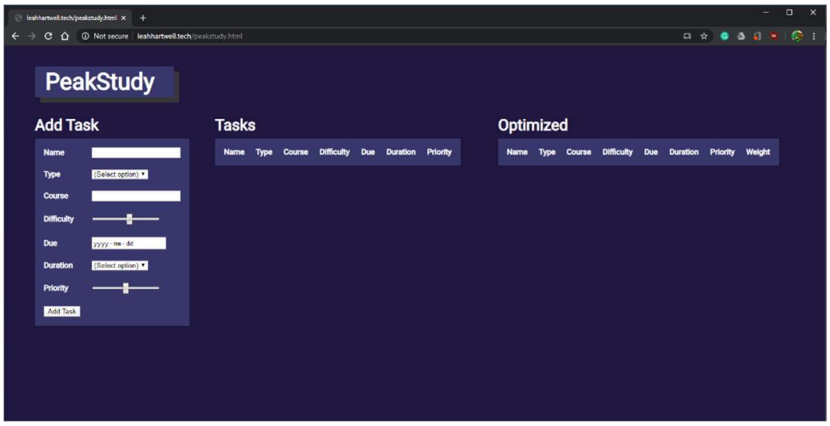
Competition:
cmd-f Hackathon 2020

Goal:

To create a web app that optimizes the user's productivity by quantifying their most important tasks through calculating/sorting them from greatest to least importance according to specific input parameters. As an engineering student with a constantly busy schedule, I thought it would be useful to develop an app that takes the burden off manual task planning.

Current Design:

PeakStudy was created for the cmd-f Hackathon 2020 by Joya Lee and me, using HTML, CSS and JavaScript. Both of us had never attended a hackathon before and only had experience in C and C++ from courses before this competition, so we learned almost all the web development code we used in our app during this 24hr hackathon. The main focus of our app was the background calculations for the task weights which were determined through algorithms based on the analytical hierarchy process and weighted decision matrix. Though our web app looked less aesthetically pleasing compared to many of the other teams, we placed in the top 10 out of 61 projects! The judges seemed quite interested in our algorithm and even suggested that we continue working on the app and broaden our scope to help companies and non-profits with project management.



Message from **nwPlus UBC** about your submission PeakStudy

Hi, as one of the top ten teams you will be receiving the Wolfram Award! This award is for a year of Wolfram|One Personal Edition plus a one-year subscription to Wolfram|Alpha Pro for each team member. I just need to grab emails from all the team members. If one of you could message all the emails to us that would be great! Thanks!

Personal Contribution:

Background Algorithm → I based the weighting algorithm off of two values that are similar to the AHP and WDM. I used the AHP created in Excel to determine the weights each parameter without user input. I used a WDM approach when analyzing the user input from the "Add Task" bar. This was done by assigning values to options within each parameter in the backend. These two values would be multiplied together for each parameter and then added up to give an overall weight to the task.

Domain/Hosting → I bought a domain and hosting service to upload our PeakStudy HTML/CSS/JavaScript code to an accessible URL. To connect the domain to the server, I changed the nameservers of the domain to the hosting services nameservers. After propagation, I added the files to my public_html folder and the site was live.

HTML/JavaScript Content → I displayed all visual components within tables for organization. Each time the "Add Task" button was clicked, a new row would appear with filled-in cells according to the most recent input parameters. The last column of the "Optimized" list displayed the calculated weight indicating the importance of each task. Each task input is also saved in an array of objects in the background. The calculated weights are used to sort tasks from greatest to least. The sorted task objects will be displayed in the next version as we were unable to figure out the display method for the sorted array within the 24hr period.

CSS Visuals → I did some basic web design with CSS connected to my HTML file. This was done through giving specific placement, sizing, and colour attributes to each of the different headings, tables and shapes. This was simply done through hard coding in CSS which was ideal due to our limited knowledge of front-end web development.

Challenges:

Team Size → As a two-person team, it was difficult to do everything within 24hrs compared to the teams of four, which was the majority of the teams at the hackathon. Even so, we managed to create a functional web app due to our ability to work together efficiently and without conflict.

New Language → As we only had experience with C and C++ prior to building PeakStudy, it was difficult to learn how to use objects to store information in our arrays and display them. Due to extensive online JavaScript documentation, we still ended up learning what we need to build our algorithm.

Domain/Hosting → Most website builders deal with domains, hosting, and SSL certificates in the backend without the user needing to understand it, but creating your own script makes it more difficult in the backend as you need to connect all these components yourself. Learning how to set it all up was confusing, but through online research, I was able to set PeakStudy up on my personal domain.

Learned:

Learning Online → Researching and reading documentation has been very helpful when learning how to do create anything I need/want in my design projects. Through this process, I feel that I have learned how to find information faster and with more accuracy than before which will be helpful for any future projects that I want to pursue.

Object-Oriented Languages → Learning how to code with objects has opened up new possibilities in the next programming projects I'll do in the future. It also has given me the confidence to look deeper into other app development languages and different services that offer machine learning and databases such as Google Cloud.

Next Steps:

Features we hope to implement are the display of our sorting function that is coded in the backend, an account service that enables saving tasks lists and connecting our app to services such as Google Calendar, iCal, etc. to suggest scheduling blocks directly based on your current schedule. We also hope to explore the business side of project management apps to see if our app would be marketable to companies and non-profit organizations as the judges suggested.

Project:
Line Following Robot – *SOP (Slap On Plastic)*

Team:
Leah Hartwell, Joya Lee

Timeline:
March 2019

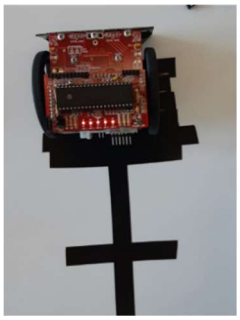
Competition:
KPU APSC 1299 Richmond-Surrey Campus Line Following Robot Competition

Goal:

To program the Microchip microcontroller of a Sumovore Robot equipped with infrared sensors and give it the ability to autonomously move obstacle course made of black electrical tape within three weeks. During the competition, our robot's ability to follow a curve, turn, jump gaps, turn around at dead ends, and stop on a landing pad was tested.

Design:

Our Sumovore named SOP was programmed using C in Microchips' MPLAB XC8 C Compiler. To program SOP we needed to use a few different libraries, some standard, some created by other users, and some entirely created by us. The most extensive library we created for our robot was motor_control.c. This library housed all of our subfunctions that told SOP what to do when he ran into specific situations. SOP utilized continuous infrared-sensor readings as his eyes to gather data from the obstacle course. These sensors would have one side transmit infrared light outwards and depending on the amount of light that was reflected back to the receiver, the microcontroller could tell whether the area beneath was black or white. We programmed SOP for each case in the competition by seeing what sensors would see black in each of the testing situations. Then given that reading we would assign a specific maneuver to pass through that obstacle. This allowed the robot to run autonomously given current sensor readings and allow it to decide and select the correct case to use in real-time.

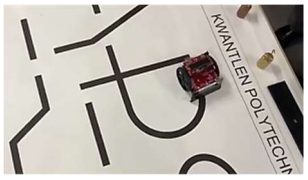


```
#include <xc.h>
#include "sumovore.h"
#include "motor_control.h"

void follow_simple_curves(void);
void follow_right_angle_turn(void);
void follow_acute_angle_turn(void);
void follow_landing_pad(void);
void follow_gaps(void);

unsigned char left_counter3(unsigned char left_value3);
unsigned char right_counter3(unsigned char right_value3);

void spin_left(void);
void turn_left(void);
void straight_fwd(void);
void turn_right(void);
void spin_right(void);

void straight_forward_fast(void);
void straight_forward_medium(void);
void straight_forward_slow(void);
void straight_stop(void);
void straight_reverse_slow(void);
void straight_reverse_medium(void);
void straight_reverse_fast(void);
void default_settings(void);

unsigned char left_value3 = 0, right_value3 = 0;
unsigned char detect_left_value3 = 0, detect_right_value3 = 0;


void motor_control(void)
{
    left_counter3(left_value3);
    right_counter3(right_value3);
    follow_simple_curves();
    follow_right_angle_turn();
    follow_acute_angle_turn();
    follow_landing_pad();
    follow_gaps();
}
```

Personal Contribution:

follow_simple_curves() → The most basic function which gave SOP the ability to follow straight and curved lines. Essentially, the SeeLine.b variable held the value of five-bits which corresponded to the five sensors in the order in which they occur on the front-facing robot. The if-else statements check to see if the current value of the sensor matched any of the SeeLine.b values and if so, would give the motors a command so that that SOP stays along the black line.

```
void follow_simple_curves(void)
{
    if ( SeeLine.b.Center ) straight_fwd();        //sensors bit 2, 0b00100u
    else if (SeeLine.b.Left) spin_left();          //sensors bit 4, 0b10000u
    else if (SeeLine.b.CntLeft) turn_left();       //sensors bit 3, 0b01000u
    else if (SeeLine.b.CntRight) turn_right();     //sensors bit 1, 0b00010u
    else if (SeeLine.b.Right) spin_right();        //sensors bit 0, 0b00001u
}
```




```
unsigned char left_counter3(unsigned char left_value3)
{
    if (SeeLine.B == 0b11100u)
        left_value3++;

    return left_value3;
}
```

```
void follow_right_angle_turn(void)
{
    detect_right_value3 = right_counter3(right_value3);
    detect_left_value3 = left_counter3(left_value3);

    if (detect_right_value3 > detect_left_value3)
    {
        spin_right;
        right_counter3(0);
        left_counter3(0);
    }
    else if (detect_right_value3 < detect_left_value3)
    {
        spin_left;
        right_counter3(0);
        left_counter3(0);
    }
    else if (detect_right_value3 == detect_left_value3)
    {
        straight_fwd;
        right_counter3(0);
        left_counter3(0);
    }
        _delay(100000ul);
}
```

← Counter Method

I implemented counters into SOP's logic:

Turns – Used two counters, one that counted the right sensors and the other the left. The code would then use these values and compare the sides, turning towards the side with a greater counter value. Counters reset after realigning.

Gaps – Used counter that logged how many times SOP saw all white. The code then had a threshold value that would tell SOP to either continue to follow the next line he sees if the count was under the threshold or turn around if the count passed the threshold and was a dead end.

Challenges:

Testing Without Digital Read Out → Due to the fact that we were unable to visually see the counters increment as the robot was running, we had a hard time debugging the logic when we ran into problems. To move pasts this, we had to physically hold SOP's wheels off the ground as we ran him over different test surfaces to see how the logic would react and try to collect data based on the wheel spin directions and speeds.

Learned:

Low-Level Language → Learned the theory behind bits and binary, octal and hexadecimal numbers and making functions and creating my own libraries of reusable functions. Learning languages like C and C++ made it easier to learn higher-level languages.

Microcontrollers/Actuators/Sensors → Used pins of microcontrollers like Digital I/O and Analog I/O with PWM. I feel that learning lower-level languages gave me a deeper knowledge about how sensors and actuators work with the logic of code.