NSF Workshop
The Heavy Lifting

Last modification: December 28, 2021

Workshop Note:
The document here includes several tasks that a student in a course would do – more than we have time for in the actual Workshop. For those who want to work through the preliminary activities, we provide some skeleton code, the instructions below, and nominal solutions.
In the Workshop, we will start in Section 6 and do the experiments needed to control the robot's motion. If you choose not to do the preliminary activities, you can start with the solution code that we provide.

1 Introduction

In this activity, you'll add the ability to detect and lift one of the "bags" for the final demo. You will use theory from mechanics to predict and improve the performance of your Romi and then do experiments to verify the results.

1.1 Objectives

To successfully complete this lab, the student will be able to:

- Use analytical tools to assess and improve the performance of a robot,
- Implement sensors and actuators to accomplish a specified task, in this case, lifting a bag as required for the final demo, and
- Develop and execute test plans to verify performance.

2 Overview

2.1 Problem statement

Your goal is to program your Romi such that when you press a specific button on the IR remote, it will initiate the pickup sequence, where your Romi will detect a delivery bag, approach it, and lift it. You will also determine the geometric configuration that will allow to lift heavy weights to sufficient heights.

2.2 Materials

- Your Romi, fully assembled with the lifting assembly and ultrasonic rangefinder.
- An arena for doing the deliveries (see project description).
- A bag. A small soup can with a paperclip for a handle works well, but exact size is not important.

2.3 Resources

- A MATLAB tool for analyzing the lifter performance can be obtained from the installer here (Windows only – if you have a mac, contact the workshop staff).

Workshop Note:
In our courses, we have the students do some mechanics calculations as a Pre-lab exercise. We specify a basic configuration and have them prove that they can analyze it. Once they have done that, we give them a calculator (developed in MATLAB) that allows them to do optimizations quickly. In the interest of time, we'll just start you with the calculator.
The student will have options for which direction to face the lifting arm. This activity assumes the hook is towards the front of the Romi (the same direction as the ultrasonic rangefinder), though some students will turn the assembly around the other direction.

3 Improving performance of the lifting arm

Figure 1 shows a basic configuration of the Romi with a lifter arm. The dimensions are defined as follows:

- L_{fore} is the length from the pivot point of the arm to the hook;
- L_{aft} is the length from the pivot point of the arm to the mounting point of the counterweight;
- L_{rocker} is the length from the pivot point of the arm to the mounting point of the coupler;
- H_{arm} is the height of the pivot above the mounting plate;
- H_{servo} is the height of the servo hub from the mounting plate;
- L_{servo} is the *horizontal* distance from the arm mount (pivot) to the servo hub;
- L_{horn} is the length of the servo horn;
- $L_{coupler}$ is the length of the coupler that connects the servo horn to the arm.

For any configuration, it will be important to calculate the following performance metrics:

- The *stroke* of the lifting arm, defined as the difference between the lowest and highest point of the hook.

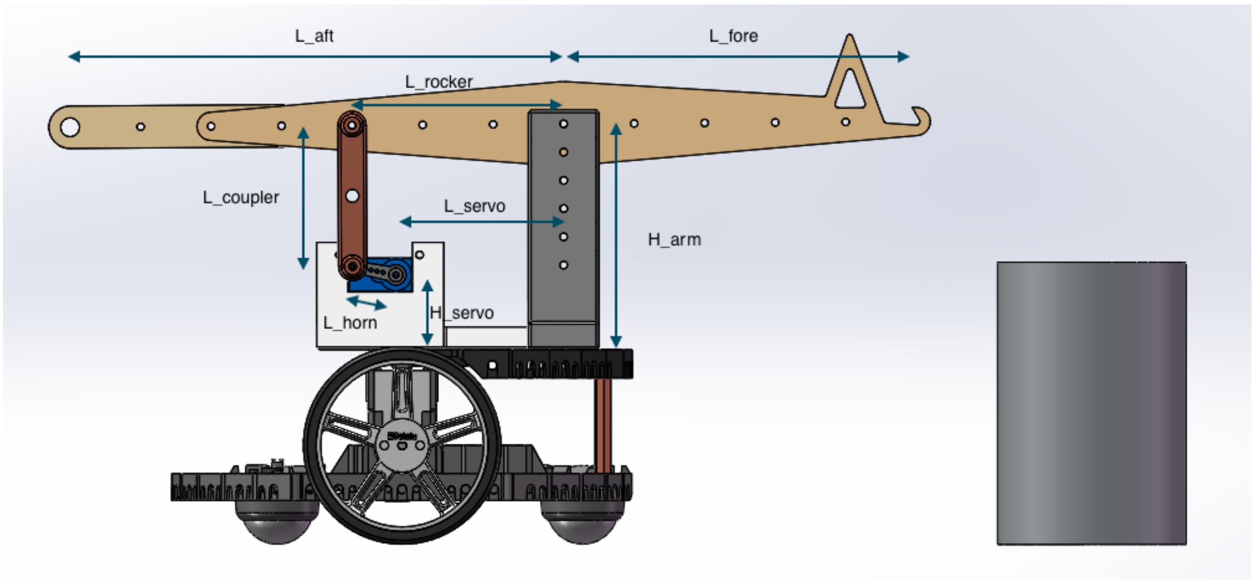

Figure 1: The Romi with a basic lifter configuration.

- For a given bag mass, the maximum torque on the motor in both the laden and unladen scenarios.
- The location of the center-of-gravity of the system (robot + bag + counterweight) for both the laden and unladen scenarios.

Procedure

1. Use the analysis tool from Section 2.3 to improve the performance of your system. Record the results of your investigations in Table 2. You do not need to record every simulation, but put down a few intermediate results and certainly your final results.
2. *As a group*, determine a configuration that will increase the *stroke* of the arm. Record the configuration parameters in Table 1. Calculate the performance metrics and record the results in Table 2. For each configuration, you'll need to find the center-of-gravity of the unladen Romi, which you can do by balancing it in the horizontal direction. To calculate the center-of-gravity when laden, you will need to use analysis, since you likely won't have bags of the exact weight.
3. As a group, discuss how a counterweight could be used to reduce the maximum torque on the servo motor. Repeat the procedure above with this new configuration.

Solution: There will be a wide variety of configurations. An ideal solution typically involves a counterweight that reduces the torque by half. Obviously, a long arm is useful, though stability can be a concern.

4 Detecting the bag

Lifting the bag is only part of the battle. Before your robot can pick up the bag (autonomously), it must detect the bag and position the hook under the bag's handle. To do that, you'll pair the dead reckoning routine from an earlier activity with readings from the ultrasonic rangefinder.

4.1 the ultrasonic rangefinder

An ultrasonic rangefinder is common sensor for measuring the distance to an object. It works by emitting a series of "chirps" and then listening for the echo. The amount of time it takes to detect the echo corresponds to the distance to the object (but remember: the chirp has to travel there *and back*!). This particular ultrasonic sensor creates a pulse on a pin that indicates the length of time it took to receive the echo. That is, when the sensor sends out the chirp, it sets the `ECHO` pin to a `HIGH` voltage and then it sets it to `LOW` when it receives the echo. A microcontroller can easily measure the time that the voltage is `HIGH`.[1]

We've provided a library to manage the ultrasonic sensor for you. The library keeps track of the echo in the background using interrupts. All you have to do is call a function to get the latest reading!

Procedure

1. Open the `rangefinder` example program and load it onto your Romi. This program will check the ultrasonic sensor every 50 ms and print the ultrasonic readings (in cm) to the Serial Monitor.

2. Point the ultrasonic sensor at an object 30 or 40 cm away and run the program. Does it print reasonable values? If you don't get decent readings, review your system.

3. What happens when you point it at objects very far away?

4. Now add code to have your robot read the ultrasonic 50 times *each time you push a button* and print each value to the Serial Monitor. Consult previous examples for ways to detect the button press. What kind of loop will you use? For now, you will want keep the `delay()` line in the loop to be sure that the sensor has had time to take a new reading and that all the ultrasonic echos from the previous ping have died down before sending out another chirp.

5. Run your code and point your sensor at an object (20 - 30 cm away again) to verify that your code still gives reasonable values and that it takes 50 readings. Are they all consistent?

6. Using a tape measure for precision, place your sensor 20 cm from a wall or other reflective object. Take 50 readings and calculate the average distance read by the sensor and the standard deviation. You're can copy-paste Serial Monitor output to a spreadsheet. This page discusses how to save directly to a file, if you'd like to try that. Be sure to exclude any wildly wrong results from your average. Record the results in the Activity Worksheet. Do you notice any potential problems with the ultrasonic readings?

[1] You'll explore how the microcontroller does this in future classes – RBE 200n and ECE 2049.

7. Repeat the process for distances of 40 cm, 60 cm, 80 cm, and 100 cm. Does the output get less reliable as the distance increases? Record the data for later analysis in the Activity Worksheet.

Solution:
The distance measurements should be within a few cm of the actual distance and the variation under 1 cm. Note that there is the occasional time-out where the sensor doesn't receive an echo. It is important to somehow filter these readings out, typically with an `if` statement that checks if the results are reasonable.

8. Hold a cell phone or other flat surface about 20 cm from the sensor, and aim it so you get a reasonable reading. Now tilt the cell phone about 45 degrees. Can you get the sound to bounce off the ceiling? How can you tell?

Solution: It's usually pretty easy to get an echo that corresponds to bounding off of a flat surface, up to the ceiling, and back down.

5 Servo motor

To actually lift a bag, you'll need an *actuator*. We have provided you with a servo motor. We have also provided a set of library routines to control the motor.

The servo motor is controlled by sending it pulses, where the duration of the pulse acts as a command to tell the servo where to move to. The library uses nominal limits of 1000 - 2000 μs, but the servo can actually move more than that, so the first thing you will need to do is adjust the limits of motion.

Procedure

1. Open and upload the `servo` example code from. This program will use library routines to command your servo motor to move back and forth. Note that, while the servo will move when powered by the USB cable, it is better to turn the battery power on to avoid drawing too much current from your USB port.

2. Adjust the parameters, `SERVO_MIN` and `SERVO_MAX` so that the servo limits correspond to vertical positions. Note that the library has built in limits to prevent driving the servo to an unreachable position. The code calls `setMinMaxMicroseconds()` to adjust those limits. Note that you may need to adjust your servo horn if you can't get it to reach the vertical positions. If you turn the power off, you can *gently* move the lifter arm to find the limits of the physical motion.

Solution: Though your results may vary, we find that limits of 500 - 2500 typically work well.

3. Answer the questions in the Worksheet about the library commands.

6 Lifting

Now that you have explored all the "pieces of the puzzle," it's time to integrate the components and methods into a working system. Two options are given below for how to program your robot. **You only need to do one or the other!**

Workshop Note: There are lots of ways to solve the problem. In this workshop, we'll use the "Drive and look" solution, where the robot drives slowly until it sees that the bag is close enough to be picked up. Note that this method assumes the arm reaches far enough out in front of the rangefinder that the pickup distance is at least 4 cm – the sensor doesn't give accurate results closer than that.

6.1 Look, then Drive

Using this method, the robot takes a range for the bag and then drives that amount. While the method can be made to work, it often requires repeated ranging as the robot nears the target to get more and more accurate results. We include it here, since it is a popular approach (and it is also useful when it comes time to find the "free range bag" in later activities. We do not give detailed instructions here.

Procedure

1. Starting with your code from last week, add a `ROBOT_BAGGING` state. Choose a key on the IR remote that will put the robot into the `ROBOT_BAGGING` state. When the key is pressed, the robot will
   (a) Read the distance to the bag,
   (b) Position the arm for pickup,
   (c) Drive the proper distance,
   (d) Stop and pick up the bag, and
   (e) Return to the `ROBOT_IDLE` state.
2. Set a bag in front of the Romi and press the key on the remote that starts it driving. Does it pick up the bag? Adjust your parameters, as needed, but be sure to make notes of your decisions – you're asked to explain your findings in the Worksheet.

6.2 Drive and Look

Using this method, the robot drives (using line following) and stops when the bag is the correct range.

Procedure

1. Starting with your code from the previous activity, add a `ROBOT_BAGGING` state.

2. Add code to `handleKeyPress` that responds to the `REWIND` key on the IR remote. When pressed, the robot will raise the servo and enter the `ROBOT_BAGGING` state.
3. Add code to the state machine in `loop()` such that when the robot is in the `ROBOT_BAGGING` state, it will:
   (a) Line follow, and
   (b) Call `checkForBag()` to see if the robot has approached a bag.

   You will probably want to set the drive speed to a low value.
4. Add the `checkForBag()` function, which will use event checking to determine if a bag has been reached.
5. When a bag is reached (`checkForBag()` returns `true`), call `pickupBag()`, which will lift the bag by commanding the servo motor and return the robot to the idle state. Don't forget to adjust the servo limits in the `setup()` function using the values you found in the preparatory activities.
6. Now set a bag on a line in front of an intersection and command your robot to drive to the intersection (where it should stop). Press the `REWIND` key. Does it pick up the bag? Adjust your parameters, as needed, but be sure to make notes of your decisions – you're asked to explain your findings in the Worksheet.

6.3 Interactions

A major theme of robotics is how concepts from different disciplines (mechanical, electrical, computer science, etc.) are integrated to make a functioning system, and this exercise certainly follows that theme. Sensor integration requires skills that are nominally in the domain of ECE and CS. Driving the motor includes those two, as well as ME. The lifting mechanism not only requires an understanding of mechanics, but also the concepts that go into controlling the arm.

So far, we have focused on the intentional interactions – the servo is used to drive the arm; the wheels are turned the correct amount to move the robot a certain distance.

There are, however, many unintended interactions that can occur when designing a system. When these unintended interactions are discovered after the fact, they can cause expensive and untimely delays. While you'll never be able to anticipate all such interactions, it is a useful exercise to identify as many interactions as possible as early as you can. Doing so will reduce the overall design time and cost.

Before you begin formal testing, spend some time identifying as many unintended interactions that you can and record them in the Worksheet.

6.4 Testing

Getting the Romi to pick up a single bag likely required some adjustment to get everything "just right." For better or for worse, though, your job doesn't end when you first get the system working. It is important that your robot is able to perform reliably, and a good way to assess reliably is to do thorough tests.

The Worksheet asks you to define a test strategy and explain your results.

Worksheet

In addition to the sign-offs in the worksheet (done electronically), submit the following to canvas. **One per group.**

1. **Be sure to submit your code in the proper assignment on canvas.** One per team. There will also be a place to submit your test plan.
2. Record the parameters for each of your lifting arm configurations in Table 1. Record the results of your analysis in Table 2. Use as many rows as needed. The first line is the same calculation in the pre-lab, without the simplification of right angles.

Configuration	L_{fore}	L_{aft}	L_{rocker}	H_{arm}	H_{servo}	L_{servo}	L_{horn}	$L_{coupler}$	$m_{counter}$
1	15cm	15cm	5 cm	8.3cm	3.5cm	4cm	1.5cm	5cm	0g
2									
3									
4									
5									
6									
7									
8									
9									
10									

Table 1: Record the parameters for each of your lifting arm configurations here.

Configuration	Bag mass	Stroke	T_{max} (laden)	T_{max} (unladen)	x_{CoG} (laden)	x_{CoG} (unladen)
1	100g					

Table 2: Record the results of your lifting calculations here. The first column should correspond to a row in Table 1. Be sure to include the bag mass for each calculation, one per row.

3. Across all your calculations/simulations, which one gives you the best performance? Why did you select that configuration?

Solution: They'll have to balance weight and height. They should have a good rationale for how they're going to score points (weight vs. number of deliveries).

4. For the ultrasonic sensor experiment, record the average reading for each distance for each student's robot in Table 3.

Distance	Student 1	Student 2	Student 3	Student 4
20 cm				
40 cm				
60 cm				
80 cm				
100 cm				

Table 3: Record the results of your ultrasonic experiments here. Use as many columns as needed for your team. Record your results as the average $\pm$ standard deviation (all in cm).

Solution: Distances should be within a few percent.

5. Using the datasets in the previous problem, create a single graph that combines the results of each student's data. For each student, plot the average distance reported by each student's sensor as a function of the actual distance. Add error bars that correspond to the standard deviation. Find the *linear* best fit for each dataset.[2] What is the offset (zero shift) error? What would the ideal slope be? How close is the actual slope to the ideal? Attach your graph to your submission.

Solution: Best fit line should have slope close to 1.0 and offset near 0.0, but neither value exactly.

6. What is the largest deviation of your best fit calculation from the actual distance?

Solution: Should be fairly small.

7. What are some of the sources of error that arise from using the ultrasonic sensor?

Solution: Temperature affects the speed of sound. Internal electronics lead to noise or bias.

8. Briefly describe one or two things you learned while optimizing the lifter mechanism.

Solution: Varies widely.

9. Identify some unintended interactions, as described in Section 6.3. List one or two actions you could take to mitigate those interactions.

[2] That is, you should have a set of points and a trendline for each student in your group.

Solution: Among others:

- When the Romi is holding a bag, or even when the arm is empty, but down, it blocks the ultrasonic. Mitigated by not reading the sensor when a bag has been picked up and keeping the arm up while sensing. Can also be used to your advantage by verifying that a bag has been lifted.
- Carrying weight may affect steering performance. Mitigated by driving slower. Possibly adjusting the counterweight to keep the CoG near the center of the vehicle.
- Lifting weight will compress the sprung caster, which will affect the line sensor readings. Mitigated by stiffening the caster or possibly by having two sets of line sensor coefficients: one for laden and one for unladen.

7 Pre-lab

This is an individual assignment.

1. Draw a free-body-diagram of your Romi as a rigid body when the arm is in a horizontal position. Consider the bag weight and the counterweight as external forces. Be sure to include important dimensions, applied forces (including reaction forces), and a coordinate system.

Solution: It's not pretty, but this shows all the important points. You could also use something like the schematic we showed in class, but don't forget the important dimensions.

2. Identify the key functions that your Romi needs to do to complete the prescribed activity in Table 4. Indicate if the function fits best with Sense/Think/Act and the component/process that will be used to meet that function. Try to break the functionality down into basic components, but you do not have to limit yourself to one of S/T/A.

   For example, for the LED in the first lab, you might write:

```
Indicate state | A | Light LED
```

Function	S/T/A	How it's met

Table 4: Identify functionality here.

Solution: Some good answers are:

- Detect bag / S / ultrasonic rangefinder
- Move (self) / A / drive motors
- Detect motion / S / encoders
- Determine position / T / Processor monitors encoders
- Lift bag / A / servo motor

One could certainly make the argument that some functions are more than one theme (e.g., "Position self" requires all three.

3. Identify two or three important requirements for this activity. While we normally want you to discuss ideas with others, in this case, make the list yourself – you'll combine your ideas with your team later. Just give a good try and you'll be fine.

> **Solution:** Important reqs are accuracy of driving (e.g., <1cm error); must remain stable; servo must have sufficient torque; must be able to lift bag to sufficient height

4. For the arm configuration in Figure 1 with the dimensions shown in Table 5, calculate the torque on the motor if the bag weighs 100g. Ignore the weight of the arm itself. Even though the servo horn is at a slight angle in the figure, assume it is horizontal. That is, assume all important angles in the figure are 90 degrees.

Configuration	L_{fore}	L_{aft}	L_{rocker}	H_{arm}	H_{servo}	L_{servo}	L_{horn}	$L_{coupler}$	$m_{counter}$
1	15cm	15cm	5 cm	8.3cm	3.5cm	4cm	1.5cm	5cm	0g

Table 5: Parameters for the torque calculation.

> **Solution:** Following the FBDs shown in lecture, we first calculate the tension force on the coupler,
>
> $$F_{coupler} = \frac{W_{bag} \cdot L_{fore}}{L_{rocker}} = \frac{0.1kg \cdot 9.81m/s^2 \cdot 15cm}{5cm} = 2.94N$$
>
> the force of the coupler is transmitted through the coupler to the servo. A moment balance on the servo horn then leads to,
>
> $$T_{motor} = -L_{horn} \cdot F + coupler = 1.5cm \cdot 2.94N = \boxed{4.41N \cdot cm}$$
>
> Note that we explicitly said the horn was horizontal, so the moment arm is the length of the horn (i.e., no cosine term).

5. *Estimate* the stroke of the lifter arm, that is, the difference between the highest and lowest point of the hook's motion. You do not have to go into trigonometric detail – just use (and state!) some simplifying approximations.

> **Solution:** A good estimate is to note that the vertical change in the pin connecting the horn to the coupler is 3cm. Though the coupler is not vertical at this point, we can make a reasonable approximation that the joint between the coupler and the arm will move by that much, as well. From the geometry, the hook will move by three times that much, so roughly $\boxed{9cm}$.
>
> (Unfortunately, I made this default configuration with a stroke of 9cm, which is nominally more than the needed 8cm. But you may find that it's not sufficient in practice.)

6. If the Romi weighs 500 g and center of gravity (without the bag) is directly over the drive

wheels, how much weight could the arm hold and still remain stable if the hook is 20 cm in front of the axle? In this case, assume that the robot tips by pivoting on the front caster and ignore the rear caster.

Solution: From the FBD in Prob. 1, we need to take the moment about the contact point of the front caster. Then, we find when the normal force at the wheel goes to zero. With the CoG over the wheel, $x_{COG} = 0$, so

$$\sum M_C = 0 = L_{caster} \cdot mg - L_{caster} \cdot N_w - (L_{hook} - L_{caster}) \cdot W_{bag}$$

With $N_w = 0$,

$$W_{bag} = \frac{L_{caster}}{(L_{hook} - L_{caster})} \cdot mg$$

I measure $L_{caster} = 4.5cm$, so

$$W_{bag} = \frac{4.5cm}{(20cm - 4.5cm} \cdot 500g \cdot 9.81m/s^2 = \boxed{145g}$$

7. Consult the datasheet for the servo motor to estimate the maximum torque when it is powered by 5V.

Solution: The stall torque is listed as 1.5 kg-cm (14.7 N-cm) at 6V and 1.3 kg-cm (12.8 N-cm) at 4.8V. Interpolating between the two gives $\boxed{13.1N \cdot cm}$.

8. For the configuration given, what is the maximum bag weight that can be lifted without exceeding 25% of the maximum torque at 5V?

Solution: 25% of the maximum torque is $3.28N \cdot cm$, which is less than the torque needed in Prob. 4. Because the equations are all linear, the torque scales directly with the weight, so we can lift

$$max = 100g \cdot \frac{3.28N \cdot cm}{4.41N \cdot cm} = \boxed{75g}$$