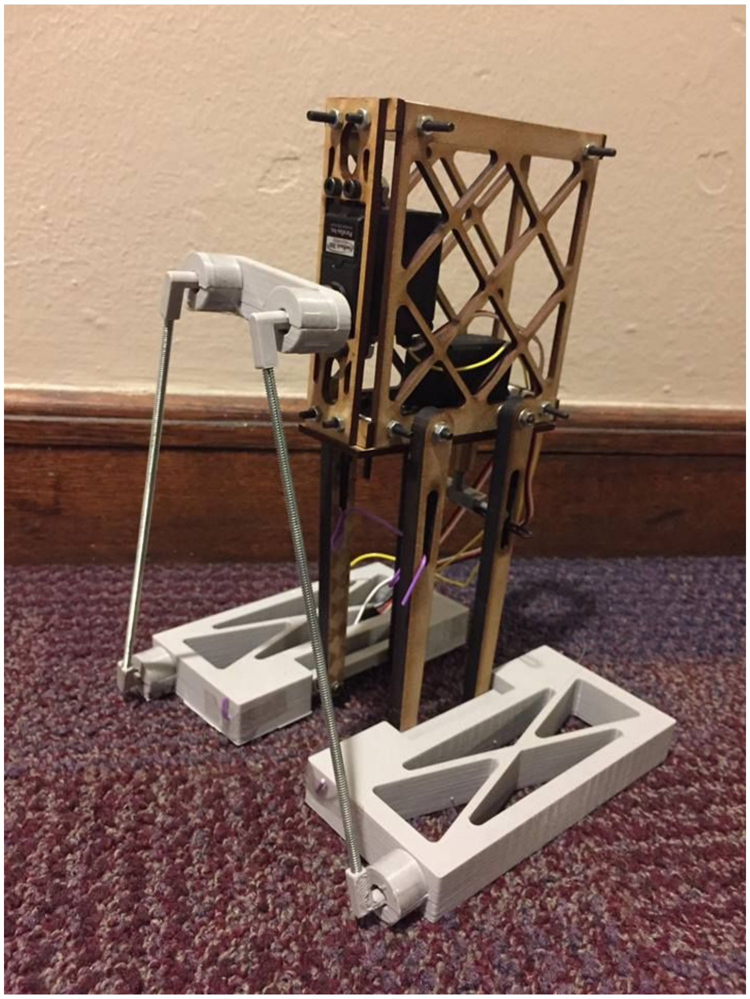
ME/CS 133

Dec 2018

Zoey Flynn, Cindy Huang, Emily Pan, Bianca Yang

Fabrication and Analysis of a Powered Walked Mechanism



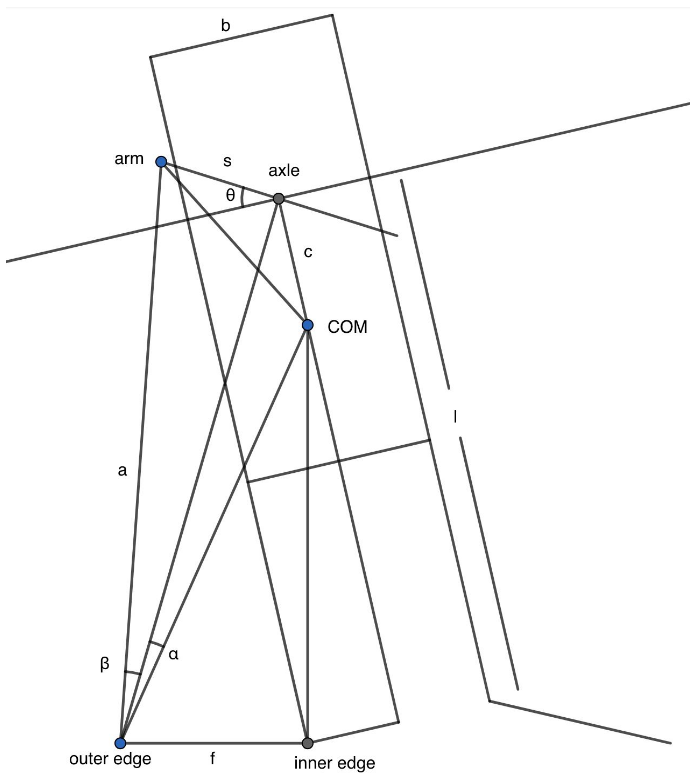
1. Mechanical Design

The design was inspired by the Penguin Robot Kit manufactured by Parallax Inc. When adapting the design for our robot we needed to be mindful of three major design considerations:

1. The robot should be able to balance when standing
2. The mass needed to be light enough that the servos could move the legs
3. The robot's center of mass needed to fully shift over one of the feet in order to pick up the opposite foot

In order to meet the first consideration we made sure that the COM was in between the robot's legs at its default position. For the second consideration we ensured that the robot was built with the lightest materials so that the weight would not exceed each servo's limit of 30.5 oz.

To determine the angle of the servo motor needed for the robot to take a step, we assume that the center of mass of the robot needs to be brought wholly above one foot to be able to move the other foot forward. The center of mass must then be limited to being above either end of the standing foot at all times. However, because it is significantly more dangerous for the robot to fall over on its side rather than fall onto the other foot, the center of mass is further limited to being at most 75% above the inner foot, or 0.75 away from the inner edge. The dimensions of the robot are given as follows:



The robot would be positioned neutrally at first (so that the base servomotor is at 0°). Assuming that the center of mass (referred to as the COM) of the robot is positioned 2.2 inches below the servomotor (calculated through Solidworks/CAD), the distance between the COM and the inner edge of the foot can be calculated via a right triangle:

$$(l-c)^2 + (0.5b)^2 = d^2_{COM,\ inner\ edge} \quad d_{COM,\ inner\ edge} = \sqrt{(l-c)^2 + 0.25b^2}$$

The distance from the COM to the outer edge can be calculated in the same way:

$$d^2_{COM,\ outer\ edge} = f^2 + d^2_{COM,\ inner\ edge} \quad d_{COM,\ outer\ edge} = \sqrt{f^2 + (l-c)^2 + 0.25b^2}$$

$$d_{COM,\ outer\ edge} = \sqrt{(1.90'')^2 + (7.05''-1.15'')^2 + 0.25(1.8'')^2} = 6.263''$$

The angle between the outer edge, the COM, and the servo axle would be equivalent to $\theta_{COM} = \pi - arctan\frac{0.5b}{l-c} - arctan\frac{f}{d_{COM,\ outer\ edge}} = \pi\ rad - arctan\frac{0.5(1.8'')}{7.05''-1.15''} - arctan\frac{1.90''}{6.263''} \approx 2.696\ rad$. We can then find the distance from the axle of the servo motor to the outer edge of the foot ($d_{axle,\ outer\ edge}$) through the law of cosines.

$$d^2_{axle,\ outer\ edge} = d^2_{COM,\ outer\ edge} + c^2 - 2cd_{COM,\ outer\ edge} cos\theta_{COM} \quad d_{axle,\ outer\ edge}$$

$$d_{axle,\ outer\ edge} = \sqrt{(6.263'')^2 + (1.15'')^2 - 2(1.15'')(6.263'')cos2.696} \approx 7.318''$$

From this value, we can calculate the total angle between the end of the servo motor arm, the outer edge of the foot, and the COM, by summing the two angles created by the line $d_{axle,\ outer\ edge}$. These angles can be calculated through the law of sines and the law of cosines:

$$\alpha = arcsin\frac{sin\theta_{COM}}{d_{axle,\ outer\ edge}}c = arcsin\frac{sin2.696}{7.318''}(1.15'') \approx 0.0678\ rad$$

$$\beta = arccos\frac{a^2 + d^2_{axle,\ outer\ edge} - s^2}{2ad_{axle,\ outer\ edge}} = arccos\frac{a^2 + d^2_{axle,\ outer\ edge} - s^2}{2ad_{axle,\ outer\ edge}}$$

The value of a is calculated by the right triangle it forms with l and f in the robot's neutral position: $a = \sqrt{l^2 + f^2} = \sqrt{(7.05'')^2 + (1.90'')^2} = 7.302''$.

$$\beta = arccos\frac{(7.302'')^2 + (7.318'')^2 - (1.00'')^2}{2(7.302'')(7.318'')} \approx 0.137\ rad$$

We can then calculate the angle θ by first calculating the distance between the end of the servo motor arm from $\alpha + \beta$ and then $\theta + \frac{\pi}{2}$, both through the law of cosines.

$$d^2_{arm,\ COM} = a^2 + d^2_{COM,\ outer\ edge} - 2ad_{COM,\ outer\ edge} cos\alpha + \beta \quad d_{arm,\ COM}$$

$$= \sqrt{(7.302'')^2 + (6.263'')^2 - 2(7.302'')(6.263'')cos(0.137 + 0.0678)} \approx 1.729''$$

$$\theta + \frac{\pi}{2} = arccos\frac{s^2 + c^2 - d^2_{arm,\ COM}}{2sc} \quad \theta = arccos\frac{(1.00'')^2 + (1.15'')^2 - (1.729'')^2}{2(1.00'')(1.15'')} - \frac{\pi}{2} \approx 0.294\ rad$$

So the servo motor must move a minimum of 0.294 radians, or 16.9°, in order for the body of the robot to rest completely above the foot. To calculate the maximum angle at $0.75f$, the horizontal distance from the COM to the inner edge of the foot is changed to $0.75f$. The vertical distance $d_{y,\,COM}$ can then be calculated as

$$d^2_{y,\,COM} = d^2_{COM,\,inner\ edge} - (0.75f)^2 \quad d^2_{COM,\,outer\ edge} = d^2_{y,\,COM} + (0.25f)^2 \quad d_{COM,\,outer\ edge}$$

$$= \sqrt{d^2_{COM,\,inner\ edge} + (0.25f)^2 - (0.75f)^2} = \sqrt{(l-c)^2 + 0.25b^2 - 0.5f^2}$$

$$= \sqrt{((7.05''-1.15'')^2 + 0.25(1.8'')^2 - 0.5(1.90'')^2} \approx 5.815''$$

Getting the servo motor angle is essentially the same process, but with the new $d_{COM,\,outer\ edge}$ substituted in when necessary. The angle θ_{COM} must be recalculated as well, using the law of cosines:

$$\theta_{COM} = \pi - arccos\frac{d^2_{COM,\,outer\ edge}+d^2_{COM,\,inner\ edge}-f^2}{2d_{COM,\,outer\ edge}d_{COM,\,inner\ edge}} = \pi - arccos\frac{d^2_{COM,\,outer\ edge}+(l-c)^2+0.25b^2-f^2}{2d_{COM,\,outer\ edge}\sqrt{(l-c)^2+0.25b^2}}$$

$$= \pi - arccos\frac{(5.815'')^2+(7.05''-1.15'')^2+0.25(1.8'')^2-(1.90'')^2}{2(5.815'')\sqrt{(7.05''-1.15'')^2+0.25(1.8'')^2}} \approx 2.819\ rad$$

We can then proceed with the servo angle calculations:

$$d_{axle,\,outer\ edge} = \sqrt{(5.815'')^2 + (1.15'')^2 - 2(1.15'')(6.263'')cos2.829} \approx 6.989''$$

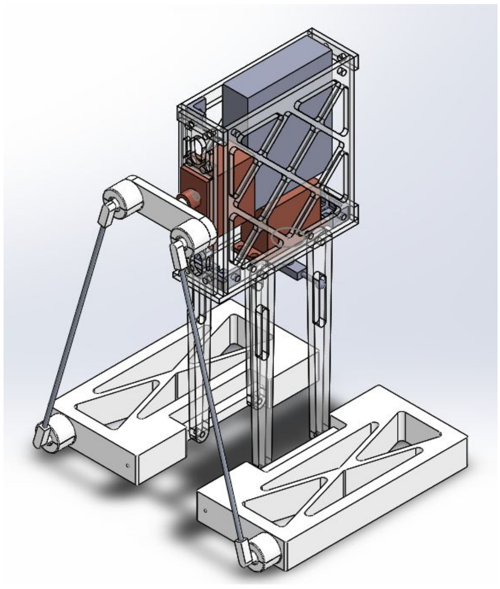
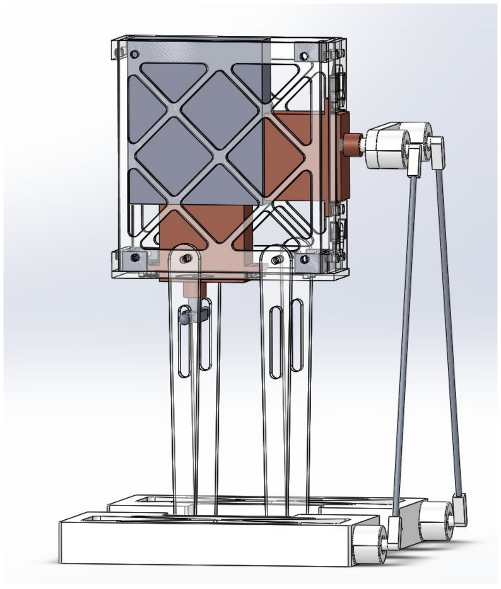
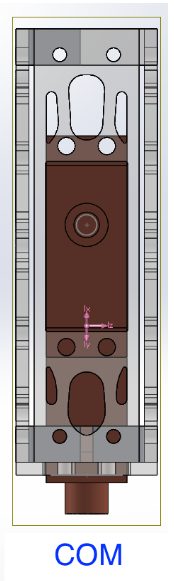
$$\alpha = arcsin\frac{sin2.829}{6.989''}(1.15'') \approx 0.0506$$

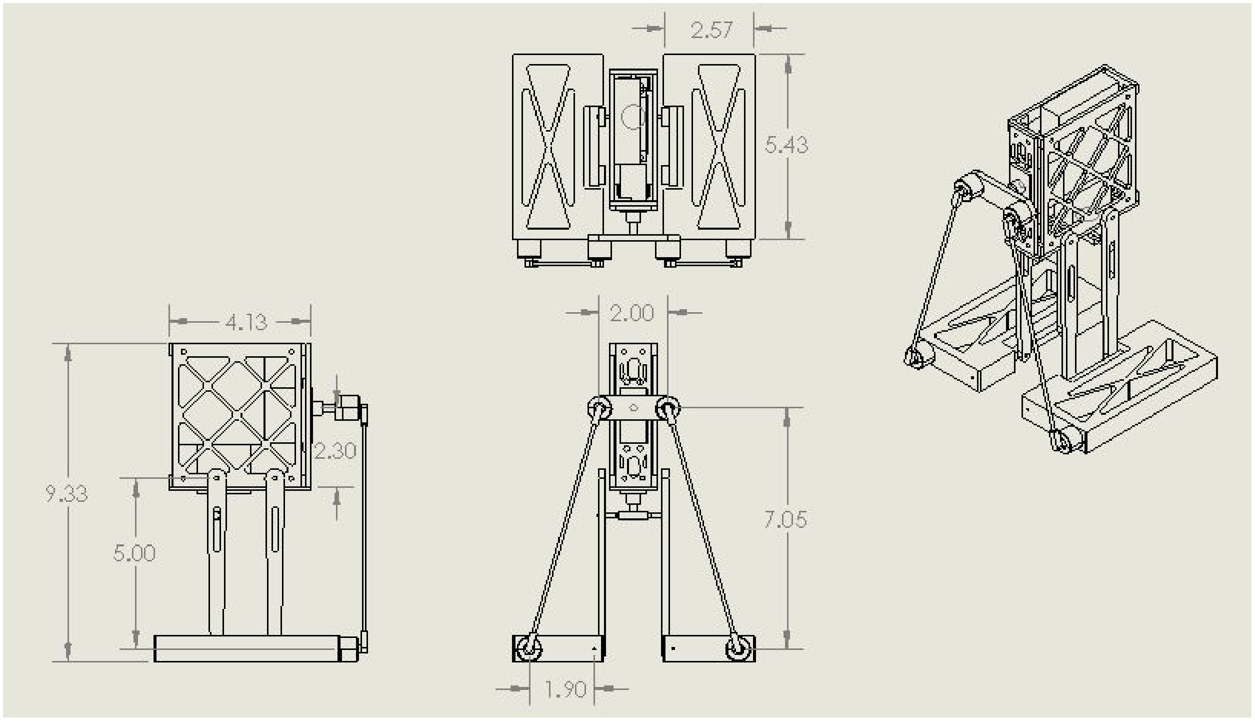
$$rad\ \beta = arccos\frac{(7.302'')^2+(6.989'')^2-(1.00'')^2}{2(7.302'')(6.989'')} \approx 0.133\ rad$$

$$d_{arm,\,COM} = \sqrt{(7.302'')^2 + (5.815'')^2 - 2(7.302'')(5.815'')cos(0.133 + 0.0506)} \approx 1.907''$$

$$theta = arccos\frac{(1.00'')^2+(1.15'')^2-(1.907'')^2}{2(1.00'')(1.15'')} - \frac{\pi}{2} \approx 0.608\ rad$$

So the servo motor arm can rotate a maximum of 0.608 radians, or 34.8°.



2. Electronics

The robot is controlled by two Parallax Feedback 360 servos. These continuous rotation servos are directly controlled by a Raspberry Pi Model 3 B+ (slave), which is connected to a Raspberry Pi Model 3 B (master) through Bluetooth.

The slave sends PWM commands to the servo, which controls how fast it rotates clockwise or counterclockwise. The amount of rotation is controlled by sending signals to continue rotation for various amounts of time. The slave Pi can be controlled over Bluetooth be pressing buttons that are attached to the master Pi. Pressing the “up” button on the master Pi will trigger the “move forward” sequence on the robot.

3. Performance

The robot was programmed to walk forward for 6 steps. Due to imprecision in the Python code which controls the servos, stability during the walk and the amount of distance covered varies between runs. Performance also varied based on initial setup of the robot. Improvements on precision can be made by using C code and setting up a feedback control loop so that exact angular positions can be encoded.

4. References

Servo references:
https://www.parallax.com/sites/default/files/downloads/900-00360-Feedback-360-HS-Servo-v1.2.pdf
Penguin Robot Kit:
https://www.adrirobot.it/parallax/penguin/pdf/Manuale_Penguin-v1.3.pdf

5. Code

Slave code:

```
import pigpio
import sys
import time
import serial

pi = pigpio.pi()
# uses BCM numbering

servoPIN1 = 4
servoFEEDBACK1 = 27
servoPIN2 = 22
# servoFEEDBACK2 =

pi.set_mode(servoPIN1, pigpio.OUTPUT)
pi.set_mode(servoPIN2, pigpio.OUTPUT)

channel = serial.Serial('/dev/rfcomm0')

def forward():
    pi.set_servo_pulsewidth(servoPIN1, 1540)
    time.sleep(.65)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1380)
    time.sleep(.2)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)

    pi.set_servo_pulsewidth(servoPIN1, 1440)
    time.sleep(.65)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1620)
    time.sleep(.4)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)

    pi.set_servo_pulsewidth(servoPIN1, 1560)
    time.sleep(.5)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1380)
    time.sleep(.6)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN1, 1440)
    time.sleep(.5)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1620)
    time.sleep(.4)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)

    pi.set_servo_pulsewidth(servoPIN1, 1560)
    time.sleep(.5)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1380)
    time.sleep(.6)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)

    pi.set_servo_pulsewidth(servoPIN1, 1440)
    time.sleep(.5)
    pi.set_servo_pulsewidth(servoPIN1, 0)
    time.sleep(.3)
    pi.set_servo_pulsewidth(servoPIN2, 1620)
    time.sleep(.4)
    pi.set_servo_pulsewidth(servoPIN2, 0)
    time.sleep(.3)

while True:
        data = channel.read()
        if data == 'u':
                forward()
                break
```

Master Code:

```
import RPi.GPIO as GPIO
import time
import serial

GPIO.setmode(GPIO.BOARD)

channel = serial.Serial('/dev/rfcomm0')

RIGHT = 7
LEFT = 11
UP = 15
DOWN = 13
GPIO.setup(RIGHT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(LEFT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(UP, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(DOWN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    right_state = GPIO.input(RIGHT)
    left_state = GPIO.input(LEFT)
    up_state = GPIO.input(UP)
    down_state = GPIO.input(DOWN)

    if not right_state:
        print('right pressed')
            channel.write('r')
        time.sleep(.2)
    elif not left_state:
        print('left pressed')
            channel.write('l')
        time.sleep(.2)
    elif not up_state:
        print('up pressed')
            channel.write('u')
        time.sleep(.2)
```

```
elif not down_state:
    print('down pressed')
          channel.write('d')
    time.sleep(.2)
```