Lab report 3 : See-Saw

Rana Muaddi 260163563

Ionut Georgian Ciobanu 260242300

Curly - the robot

1) Analysis of rolling object

The analysis of the motion of a rolling object involves applying Newton’s second law.

There are three forces acting on an object on an inclined plane. The Normal force (N), the friction (f) with the incline, and the gravitational force (Mg). All forces act on the center of mass of the rolling body except for friction.

This setup causes two simultaneous motions to occur: translational motion and rotational motion. Translational motion is the movement of the object down the incline and rotational motion is the rotation of the object as it rolls down.

As the body rolls down, both its angular velocity and its linear velocity increase. The x-component of the gravitational force (Mgsin ɵ) which is parallel to the incline causes the object to accelerate linearly as it goes down the incline and decelerate as it goes up the incline. The acceleration is constant in both directions.

The friction prevents the sliding of the object. The normal force and the y component of the gravitational force cancel each other out. What we are left with is the x component of the gravitational force and the friction which are acting on slightly different points of the rolling body. This creates a torque which causes the rolling motion.

The angular velocity ω is found to be equal to v/r. v being the translational velocity of the wheel and r being the radius of the wheel.

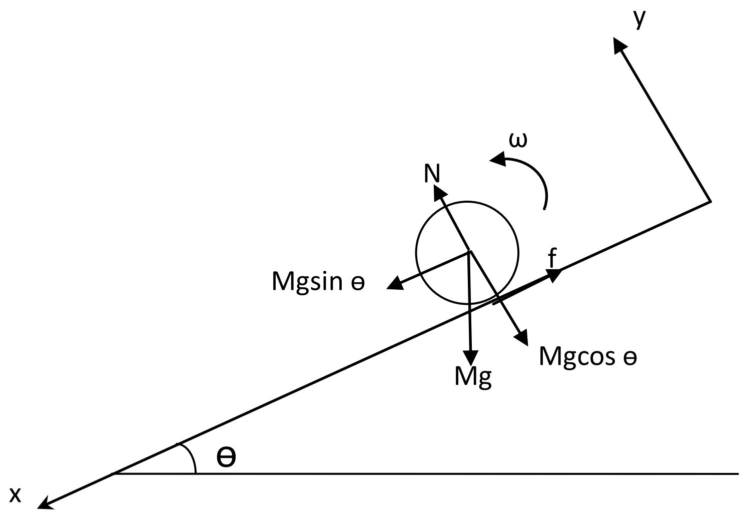

Figure 1: forces acting on a rolling object

2) Proposed algorithm

Our algorithm uses a motor to control the inline and a light sensor placed at the pivot underneath the inline to detect the wheel when it passes on top.

We set a threshold for the light sensor which was slightly less than the reading the sensor gets from the natural light in the room.

When the wheel passes in front of the sensor it casts a shadow which decreases the reading from the sensor. When the reading from the sensor is below the threshold point we cause the motor to rotate 180 degrees which Reverses the direction of the incline and makes the wheel decelerate to a stop and start rolling in the other direction and so on.

We have a 200 millisecond wait before the motor rotates so that we ensure the wheel reaches the end on the incline.

The wheel should start rolling from one end of the incline and the inline should be set at the highest or lowest possible.

3) NXC Code

```
//Global variables used



#define Down 1;

#define Up 0;


int detected = 0;  //Has the wheel passed the sensor?

int UpDown = 0;

int threshold = 49; //Less than this amount means we are looking at the wheel

int WaitTimeDown, WaitTimeUp, counter;
```

```
//Rotate the motor a specified number of degrees

void Rotate(int degrees, int power){

    if (degrees < 0)

        RotateMotor(OUT_B, -power, degrees);

    else

        RotateMotor(OUT_B, power, degrees);

}


//Print sensor information to the screen

task PrintInfo()

{

 while (true){

    //ClearScreen();


    TextOut(0, LCD_LINE1, "Light Sensor: ");

    NumOut(80, LCD_LINE1, Sensor(S3));


    TextOut(0, LCD_LINE2, "Counter: ");

    NumOut(80, LCD_LINE2, counter);


    Wait(250); //just so that we do not overload the processor

 }

}
```

```
//Move the arm such that the ball does not fall off

//We started off with very smart and complex algorithms and ended up with a very simple one

task Drive(){


    while(true){

        if(Sensor(S3)<threshold) {

            Wait(200);

            RotateMotor(OUT_B, -60, 180);

            Wait(250);

            counter++;

        }

    }

}




//Check to see if the wheel is passing in front of the sensor.

//If yes, than take a couple of samples, to make sure it's not noise.

//This also helps against detecting the passing wheel multiple times on one pass

task UpdateValues(){

  while(true){

  long sum = 0;

  int i;

  for ( i = 0; i < 5; i ++){

    sum += SENSOR_3;
```

```
 }
   if (sum < threshold){
    detected = 1;
   }
 }
}



//Initializes the sensors on the correct ports
//Starts the tasks
task main()
{
 SetSensorLight(IN_3);
 counter = 0;
 detected = 0;

    start PrintInfo;
    start Drive;
    start UpdateValues;
}
```

4) Discussion

Discuss the limitations of your theoretical design process, and the process you had to follow to arrive at a working system.

While we tried our best to fully describe the problem and come up with a very intelligent solution, there are a couple of factors that we could not really measure or control: the variation of the Lego assembly over the distance of the ramp, the variations in the layer of tape we applied to the wheel and the ramp, the wheel imperfections. There is also the usual problem of the motors not being exactly precise and, of course, not being able to measure the value of the friction force. We also had difficulties as the wheel would not always roll perfectly, and the movement on the ramp was not always symmetrical. Another significant issue was that the initial conditions (the start speed and distance of the wheel) would vary from experiment to experiment.

We tried measuring the time it takes between successive passes, the time the wheel spends in front of the sensor (essentially measuring the speed of the wheel), and a couple other approaches. In the end, tweaking the times, based on the movement analysis of the actual wheel yielded a much better result.

We faced many limitations while implementing the seesaw. First, the light sensor did not detect the wheel the way we were told to set it up. So we placed the light sensor underneath and very close to the incline. Second, the ramp and the wheel had bumps which caused the wheel to get stuck and stop moving. We solved that by putting tape of the ramp and on the wheel. It took a good number of tries, experimenting with several types of tape and combinations, to arrive at a good solution.

An interesting problem was detecting whether the wheel passed or not. We could detect its presence in front of the sensor, but if it moved fairly slowly or there was some fluctuation in the ambient light, we would easily get false positives. For that, we decided to aggregate a couple of measurements and take a majority vote. If we had 5 “clean” samples, then we would decide the ball was really in front of the sensor. This approach proved robust enough even in the presence of ambient light fluctuations.