LeJOS RTS Train Project 2018

Original Write-Up Done By:

Cody Martin

Cdm1322@jagmail.southalabma.edu

If you're reading this I wish you good luck. I had a good time working on this project and I hope you can do great things with it. If anything is unclear here feel free to contact me for any questions or help understanding. I'll likely be at USA for another 3-5 years.

Contents

LeJOS Overview

Setup (Windows)

Initial setup consists of downloading the newest version of Eclipse and installing the LeJOS plugin. You can find the install information at https://lejos.sourceforge.io/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm .

Once you have Eclipse setup and the plugin installed, copy the folders mentioned below into your LeJOS workspace directory.

General Program Functioning

Variables and their meaning

I. TrainNXT Grab1 -> Grasper Variables
   a. downCart – the number of degrees Motor B turns to be in the loading/unloading position for the train cart.
   b. downHopper – the number of degrees Motor B turns to be in the loading/unloading position for the hopper (brick was blocks on it)
   c. num_blocks – should start at zero. Keeps track of how many blocks have been loaded or unloaded. Loading and unloading processes limit this to four blocks. Through each loop this will be returned to zero.
   d. Time – keeps track of how long it takes for a grabber to load or unload.
   e. Light – Sensor on sensor port 1. Checks to see if there is a block that the grabber can pick up.
   f. Sound – Sensor on sensor port 2. This is one of the track sensors that tells the grabber arm and the control brick that the train has reached the appropriate position for loading/unloading.
   g. Timer – used to measure how much time is elapsing while loading/unloading
   h. toMas – output stream that sends flags to the master control brick
   i. fromMas – input stream that receives flags from the master control brick

II. TrainNXT Grab1 -> MasterBrick Variables
   a. Seconds – holds the total amount of time elapsed. This works in conjunction with the built-in scheduling process. Before grabbers were added this functioned within a fraction of a second to the deadline given. Grabbers have broken this precision due to erroneous behavior while loading/unloading.
   b. Record – time taken to complete tasks. Plays a role in automating speed control.
   c. Timer – used to measure how much time is elapsing between trips.
   d. prevSensor – initialized to true. This is used to help know whether the train has taken the outer track or inner track.

e. LENGTH_BETWEEN_SENSORS – This is a hard coded value that is the distance between each original track sensor. Grabber sensors have a difference in distance that must be accounted for later.
f. Start – holds the starting sensor position of the train.
g. scheduledTime – This is entered by the user of the system. It is the intended deadline to be met. Originally if there was time to spare a calculation was done to see how fast the train had to move to still meet time requirement. If it was running short on time, train would accelerate to its maximum speed. If you enter too small of a number for this value, the entire system may not function properly. Too large of a value would also cause the system to not function properly.
h. totalDistance – is used to first determine the total distance to be travelled initially, then is decremented as the train makes progress around the track.
i. desiredSpeed – this is what speed is desired for the amount of distance, time, and compensation left. The equation is
   desiredSpeed = totalDistance / (scheduledTime - compensation)
   IF compensation is too high (higher than scheduledTime), this equation leads to a negative desiredSpeed. This needs to be investigated more.
j. startingSpeed – From what I can tell this is not used.
k. Schedule – Two dimensional array that holds values for subtrips and what to do at the scheduled location. [0][0] is the starting position. The first dimension is simply a counter. The second dimension holds the following values. [x][0] holds the destination, [x][1] holds the action to perform at the loader, [x][2] holds the number of blocks. Currently the system is limited to 5 subtrips (x = [0,5]).
l. Turnout – Initialized to true. This can be changed depending on how the track is initialized before operation. If initialized to true you are indicating that both turnouts are set for the outside track.
m. Loader – tells the grabber arm when activated what action to perform, 1 for unload and 2 for load. If set to 1 for unload, the second pass by the grabber will perform a load, the third pass will do nothing.
n. numBlocks – number of blocks that will be loaded/unloaded at next stop.
o. toPC , fromPC, toLoader, fromLoader – input/output streams so the main control brick can communicate with the grabber and PC.

III. TrainNXT Grab1 -> TrainSlave Variables – This does not have any class variables, only instance variables.
   a. Four, five, six – sensor ports 1, 2, and 3 respectively.
   b. Con, dos – con holds the connection to the master control brick. Dos output stream for con.

IV. TrainPC(1, 2, 3) -> Master
   a. Out, in – input/output streams for talking with the master control brick
   b. Kb – scanner object for reading in information provided by the user
   c. Stops – array that holds destinations entered by the user
   d. Link – connection object for connecting to the NXT control brick
   e. outData, inData – data input/output streams for talking with the master control brick
   f. start – location of where the train will start from
   g. dest – holds the next destination and is sent to the master control brick

h. time – holds the desired schedule deadline time entered by the user
i. speed, sec, and location – all are used to display current train information received from the master brick controller.

V. For other versions of the main program set, more variables are added but most are simply copies of the one stated here already.

Functions/Methods and their meaning

I. TrainNXT Grab1 -> Grasper Methods
   a. Main()
      i. Input blocks until a trigger from the master control brick is received. If the trigger is one an unloading process begins. If the trigger is 2 then a loading process begins. Once the trigger is received the grabber brick sends a signal to the master control brick which is interpreted as slow down the train until next signal. Once the train is at the correct sensor position for loading/unloading the master brick is notified which will stop the train.
   b. Unload()
      i. The grabber arm begins the unloading process. First the arm is moved into the downCart position, then the method search2Unload() is called in an if statement. If searchToUnload() returns true, then the block is delivered to the hopper location and returns back to search for more until either the num_blocks exceeds three or the counter exceeds five. The master control brick is then notified that the unload() has finished so it can continue controlling the train's speed.
   c. Load()
      i. Essentially the same process as unload() except in reverse and calling search2Load().
   d. Search2Unload()
      i. Degree is how many degrees the arm will move each time it iterates through searching for a block to move. l_value will hold the current light sensor value on the arm. The first Motor C rotation was added to overcome inaccuracies when the train stops. Arm then goes through ten iterations of searching for a block. If the light sensor ever is greater than 49 the arm attempts to grab the block. Grabber then performs a check to ensure that the block was grabbed correctly by checking if l_value is less than 35. If all is well the arm returns to where it started with the block and returns true causing the arm to bring the block to the hopper area. This process is repeated as many times as unload() requests.
   e. Search2Load()
      i. Behaves essentially the same as Search2Unload(), except in reverse.
   f. timedOut()
      i. Handles keeping up with the current elapsed time

II. TrainNXT Grab1 -> MasterBrick
   a. Main
      i. Connections and streams are opened between TrainSlave, Grabber(s) and the PC. Once connections are made, input is received from the PC for starting position, destinations (up to five) and the desired schedule time. The desired speed is calculated based on the total distance, compensation and requested schedule time. The total distance and compensation is calculated within the distance() method. The

main loop is entered. Each iteration through the loop checks to see if the destination is 4, 5, or 6. Depending on which it is the turnouts will be flipped with the turnout() method. ID holds the current sensor that was hit. Depending on the sensor different actions will be taken. Locations 1-3 are read in from the master brick while 4-6 are read in from the trainslave. Each time a location is hit, speedometer() is called with the current position being passed to it. If sensor two is hit, loader() is called given that the variable loader does not equal zero. This starts the loading/unloading behavior.

b. timedOut()
    i. keeps track of the current elapsed time to be reported to the PC
c. speedometer(currentSensor)
    i. Reports what location was hit to the PC. Checks to see if prevSensor is 6 or 5 which determines if the outside or inside track was taken. Depending on the current sensor, length is removed from totalDistance and a new speed is calculated. The relevant information is then reported to the PC.
d. speedAdjuster(currentSpeed)
    i. This determines how much to turn the speed control knob. Previously the top speed was determined to be 28, however I had accuracy issues at that speed and lowered it to twelve.
e. Distance(id, tripEnd, subtrip)
    i. This determines distance between current sensor and destination. If passing by a loader location, compensation is increased based on numBlocks.
f. Turnout(direction)
    i. Direction is either positive or negative. Method simply tells the turn out motors (Motors B and C) to turn in order to open or close the path to the outer track.
g. Loader()
    i. When the train has hit a loader location, loader() slows the train down enough so that it can stop in a semi-accurate location for loading/unloading. It blocks on reading from the loader brick until the loader brick sends a signal to stop. Then it blocks while reading from the loader to acknowledge the loading process is completed. Compensation is then adjusted according to numBlocks and totalDistance is also decremented to account for the difference between the loader location and stop location. Finally the train is returned to it's previous speed.

III. TrainNXT Grab1 -> TrainSlave
a. Main()
    i. This method simply waits for a sensor to be hit and reports it to the main control brick.

IV. TrainPC (1,2,3) -> Master
a. Main()
    i. Method connects to the master control brick and relays user input to the control brick including schedule time, starting position, and requested destination. Once data is sent out, it waits for data to be returned by the control brick and prints out location, speed, and sec to the PC terminal window.

Specific Projects

I. **TrainNXT-Grab1** – Very close to the original design. Grabber location 1 and sensor location 2 functions, Grabber 2 does not.

II. **TrainNXT2-Grab2** – Also very close to the original design, except Grabber 2 works instead of Grabber 1. Grabber 2 is located at sensor location four.

III. **TrainNXT3-Grab1&Grab2-NotWorking** – This was the first attempt at getting both grabbers working together. The second grabber was added on almost identically as Grabber 1 (through a Bluetooth connection). Having multiple connections through Bluetooth is possible but dramatically increases latency. This module demonstrates how this method causes catastrophic failure.

IV. **TrainNXT4-Grab1&Grab2-Working** – After reworking how everything was set up it was determined the easiest way to get the second grabber to work with the first is to divert control of the second grabber through the PC. This requires the second grabber to be connected to the PC via USB. There is a USB hub that should be connected to the PC that has USB wires connecting the hub to the main control brick and grabber 2.

V. **TrainPC(1,2,3)** is what is ran on the PC for TrainNXT projects 1,2 and 3. Before running this program, ensure that all NXT bricks are on and running the correct programs.

VI. **TrainPC(1,2,3)2** is what is ran on the PC for TrainNXT4-Grab1&Grab2-Working. Again, ensure that all NXTs are powered on and running the correct software before executing this program.