Lola’s Programming Chain as of 4/9/2014

Main : output.txt (Contains output

robotWindow:

Up:

Down:

Left:

Right:

Forward:

Backward:

EForward:

EBackward:

ELeft:(Overloaded)

:ELeft (Robot)- Turns 90 degrees left.

:ELeft(Robot, Time) – Turns left for number of ticks specified.

ERight:

:ERight(Robot)- Turns 90 degrees Right.

:ERight(Robot, Time)- Turns Right for number of ticks specified.

waterTests: -Tests both Turbidity[0] and Salinity[1], returns double array of values.

salinityTest: -Tests Salinity, returns double Salinity.

turbidityTest: -Tests Turbidity, returns double Turbidity.

FullRun: OBSELETE, deleted from Github 4/3/14

setPing: (Overloaded)

:setPing(Robot, whichPing, Lola)-Detects Ping X[0] or Y[1] based on int whichPing input, gives the Ping detected to Lola, and returns Ping.

:setPing(Robot,whichPing)- Detects X[0] or Y[1] based on int whichPing input, and returns Ping.  
:setPing(Robot,Lola)- Detects both X and Y and passes them to Lola.

Retrieve:(Robot, Times)- Rams forward then backs up, based on int Times.

coverOpen: - Opens cover with servo 1.

coverClose: - Closes cover with servo 1.

bridgeRun: (Overloaded)

:bridgeRun(Robot, Time)- Runs forward at high speed for specified time. :bridgeRun(Robot)-Runs up ramp and zigzags across bridge following color change. OBSELETE

getPosition: (Robot, int[] expectedPosition) - Calls setPing(Robot,whichPing) to detect PingX and PingY, compares it to expectedPosition, and returns Coordinates[changes needed X,Y, PingX,Y].

correctPosition: (Robot, int[] Changes) – Runs motors to correct position based on changes as specified in int[] Changes.

testPosition: (Robot, int[] expectedPosition) – Driver function for getPosition and correctPosition.

UTurn: (Robot, int Direction)- Executes a 180 degree turn, with direction based on int Direction. 0 for Left, 1 for Right.

testBridge: (Robot) – Runs along in front of bridge searching with line sensor, then once found executed bridgeRun. OBSELETE

Bump: -Detects if bump sensor is pressed.

IR: -Detects from IR Proximity Sensor and does math to convert to useful value.

ColorTest: -Returns output from line sensor.

ballArm :(Robot, Angle) -Moves ‘Arm’ servo to specified angle.

getEPosition: - Returns the two motors’ positions in an int array[Right,Left].

Bump: -Detects if bump sensor is pressed.

Final:( Robot, int First,Second,Third,Fourth)- Driver program for full robot run. Accepts four ints for the four dispenser values. Calls most functions.

LolaObjectMichael: Constructor class for LolaObjectMichael, usually referred to as Lola. Contains attributes X and Y Coordinates as well as Coordinates array, as well as current number of balls for Salinity and Turbidity. Also contains getters and setters for all attributes. NOTE: Setters interact with each other. i.e. setX sets the X attribute, but also sets Coordinates[0], as they are the same attribute, just called different ways.

NormalObjectMichael: Constructor class for NormalObjectMichael, which is a basic obstacle like a dispenser. Contains attributes XLength and YLength as well as Lengths array, along with getters and setters for all attributes. Used to check for if the Robot will hit an obstacle, as seen by ping sensors.

BoundaryObjectMichael: Constructor class for BoundaryObjectMichael, which is the field. Contains attributes XLimit and YLimit as well as Limits array, along with getters and setters for all attributes. Used to check that the Robot will stay within the object, aka the limits of the course, and not get stuck on a wall.

checkPosition: (POSSIBLE METHOD, NOT WRITTEN) – Either uses getPing to determine Lola’s Position or calls getCoordinates method in Lola object to get position. Then compares those values to Normal objects and the Boundary object, to make sure that planned movement doesn’t hit anything AND is in the expected position compared to those objects. CAN do this at specific points in code or dynamically. CAN change position if needed or pass to another function to change the position.