Exercise 7 (IR P & D):

I lied…a MEMs gyro does not give you angular position, it provides angular rate (degrees/sec). So we will **integrate** the angular rate from the gyro to get an angular position.

The MEMs gyro will also have a calibration step performed to help cancel any offset in its reading.

The verilog for the block that performs both the integration and the calibration will be provided. So why do I mention it?

What happens if you integrate a constant? No matter how small the constant?

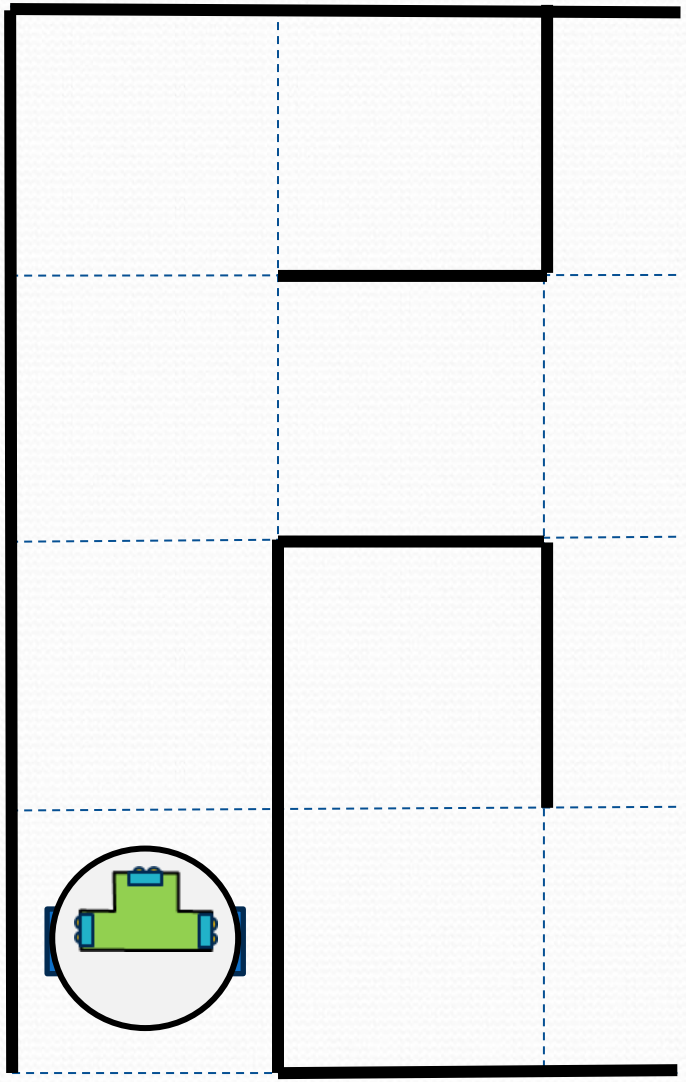
Yes…the integral ramps up over time. Since our heading comes from integrating angular rate over time it will eventually drift and become inaccurate. A MEMs gyro alone cannot provide the navigation accuracy we need.

Exercise 7 (IR Sensors)

Imagine our robot navigating a maze

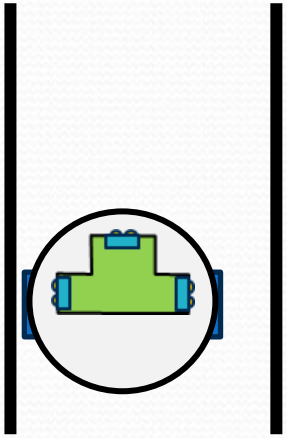
Not only do we need to correct the long term drift of the gyro, we need to know when there is a wall in front, and when there are openings to the side.

The MazeRunner has 3 IR sensors. One pointing forward to detect a wall in front, and left and right sensors to seek openings, but to also correct for gyro drift.



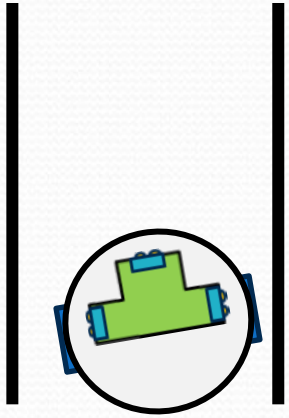
This exercise is about the math needed to correct gyro drift

Exercise 7 (IR PD correction)



There are two scenarios the lft/rght IR sensors help us correct.

If the MazeRunner is too close to a wall as shown the left IR sensor will read significantly lower than the right. A **P**-term *(proportional)* of IR sensors will help us correct this. *(of course there is the sub-case of too close to the right wall as well, but we have signed math and that handles both error directions)*



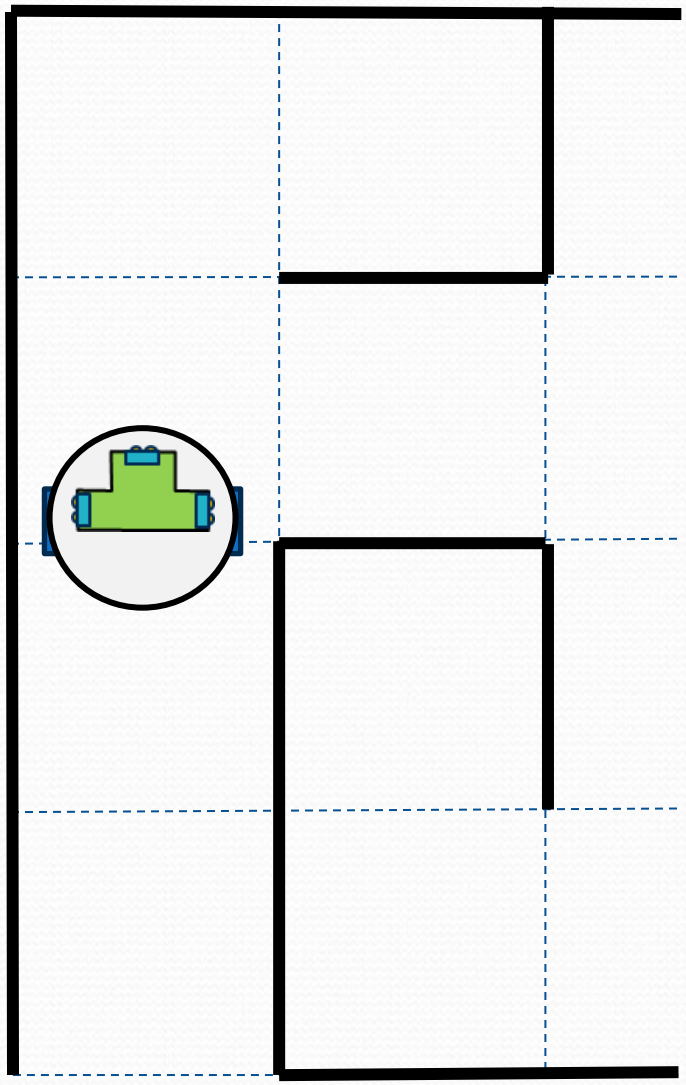
The 2nd scenario is the direction of the robot being off and the robot is veering toward a wall. The D-term *(time rate of change of the IR sensors)* can help us correct this.

Exercise 7 (IR sensors…both not always available)

Navigation is best done using the difference of the left vs right IR sensors. However, depending on the MazeRunner's position in the maze both sensors may not be available.

Shown here is a scenario where the left sensor is available and the right IR sensor is not.

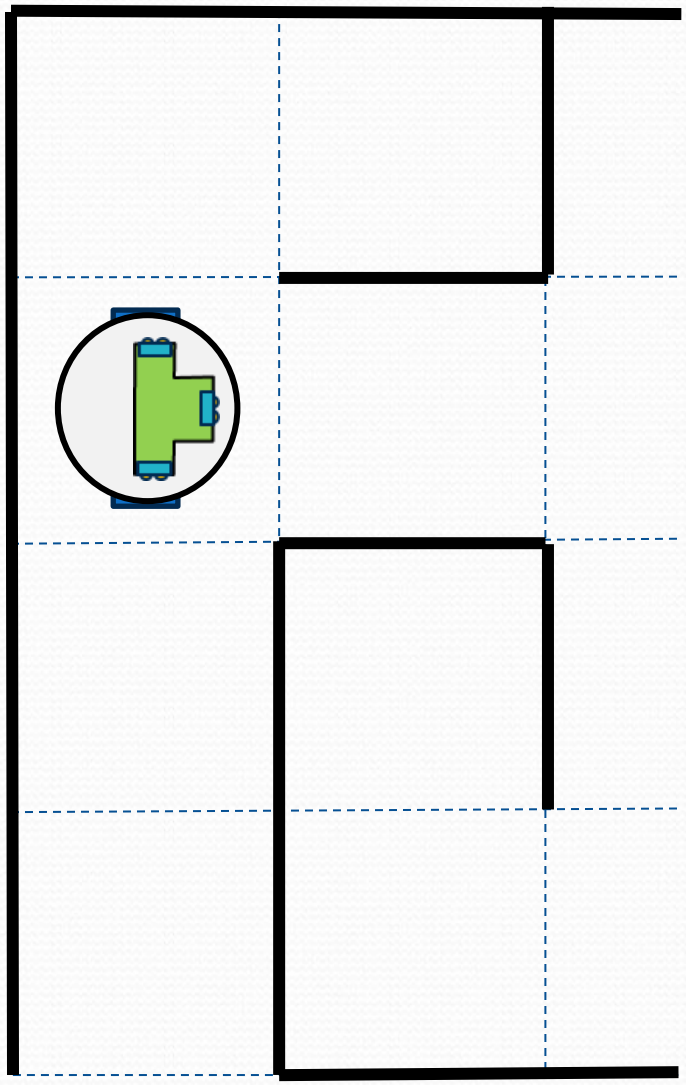
We will run with just info from the left IR sensor in this case.



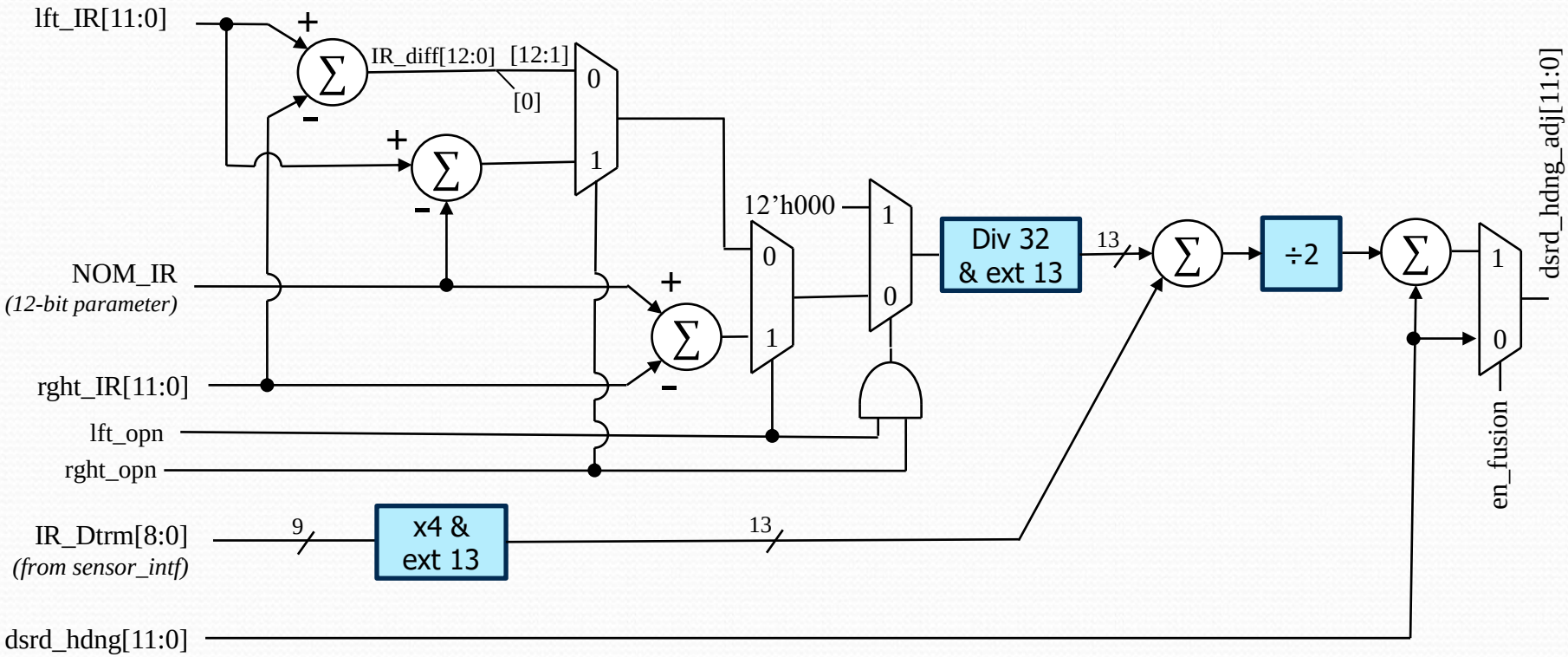
Exercise 7 (IR sensors…both not always available)

There are even brief scenarios *(1/2 square of travel)* where both IR sensors are unavailable.

In these cases, the IR correction will be zeroed out, and the MazeRunner will briefly run with only gyro based heading.



Exercise 7 (IR sensor Math)



IR_Math Interface

Signal:	Dir:	Description:
lft_opn, right_opn	in	Indicate left/right IR sensors have no reading *(the path is open)*
lft_IR[11:0], right_IR[11:0]	in	IR readings *(decrease closer object (maze wall) is)*
NOM_IR	param	12-bit parameter defaulted to 12'h900
IR_Dtrm[8:0]	in	Derivative of IR readings, comes from ***sensor_intf*** block
en_fusion	in	IR fused with gyro only when moving at decent speed
dsrd_hdng[11:0]	in	Desired heading of MazeRunner (comes from ***cmd_proc***)
dsrd_hdng_adj[11:0]	out	What we are making….the adjusted desired heading.

Produce a block (***IR_Math.sv***) with the above specified interface and the functionality specified on the preceeding slide. A self-checking testbench (***IR_Math_tb.sv***) is provided. Debug and submit as much as you have completed by end of class. This is problem on HW2.