Lab 6 Group 6

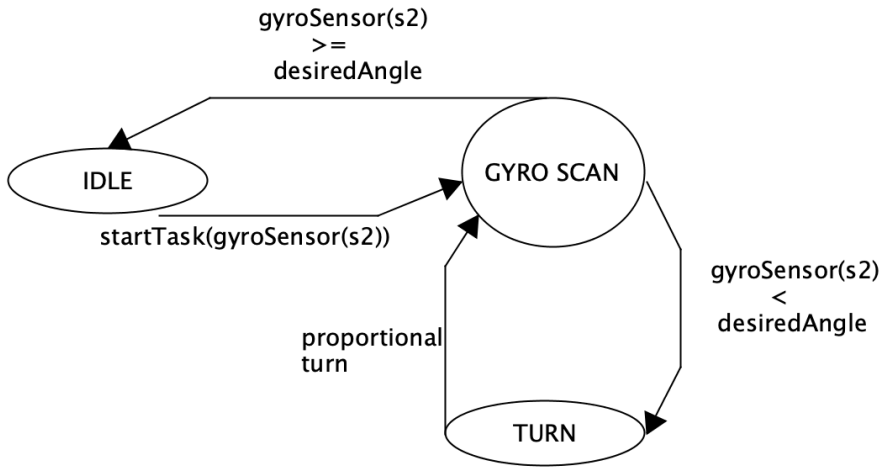
Participation:

Seth Weber (50%)

William Oakes (50%)

QUESTION 1 :

A) *State-Machine:*



B) *Pseudocode:*

1) Declare task function for controller

2) Set infinite while loop

3) Within while loop call proportional_TurnController() function

4) Declare proportional_TurnController() function outside and above task for controller

5) Within turn controller declare an error variable set equal to k * (desiredAngle – currentAngle)

6) Within turn controller declare

C)

D) This implementation compares better than the previous implementations in context to turn degree accuracy, with a slight loss in speed. The final gyroSensor value is steadily holding at within 2 degrees of the user designated turn degree. The robot typically lands slightly below the desired angle which contrasts with previous implementations that would land an average of 3-4 degrees above the desired turn angle.

QUESTION 2 :

A) *Pseudocode:*

1) reuse the proportional turn controller code (or append logic into that function)

2) check whether current motor power + slewRate greater than or equal to maximum motor power

I) if true -> enter if statement – set current motor power to the maximum motor Power

II) if false -> enter else statement – increment current motor power by the SlewRate

3) return the current motor power

B) **Source code included in .c files*

C) The accuracy is about the same where we are getting within a degree of our desired turn angle, however the overall performance seems more seamless and does appear quicker. We believe this is due to the robot's motors having better ability and capacity to hit narrow targets efficiently with proportional controlling of the motors' speeds.

What this means is that comparing the solution in Question 1 and Question 2 yields the observation that in Question 1 the robot spends more time slowing down and perfecting it's proportional turning. In Question 2 the proportional acceleration, caused by implementing the slewRate into the algorithm, causes the robot to spend less time in a deceleration phase in the latter motions of execution past the fulcrum point variable designated in our program.