Author: Gabe Harms

Date: 12/1/14

Class: CMPEN 417

Project Objective

**Intro**

The purpose of this memo is to get a better sense of direction for our project. Last meeting, before break, not much was accomplished aside from gaining a better understanding of the message protocol. Since we have the RS-232 read/write interface set up, the next thing we must do is use it to communicate with the robot.

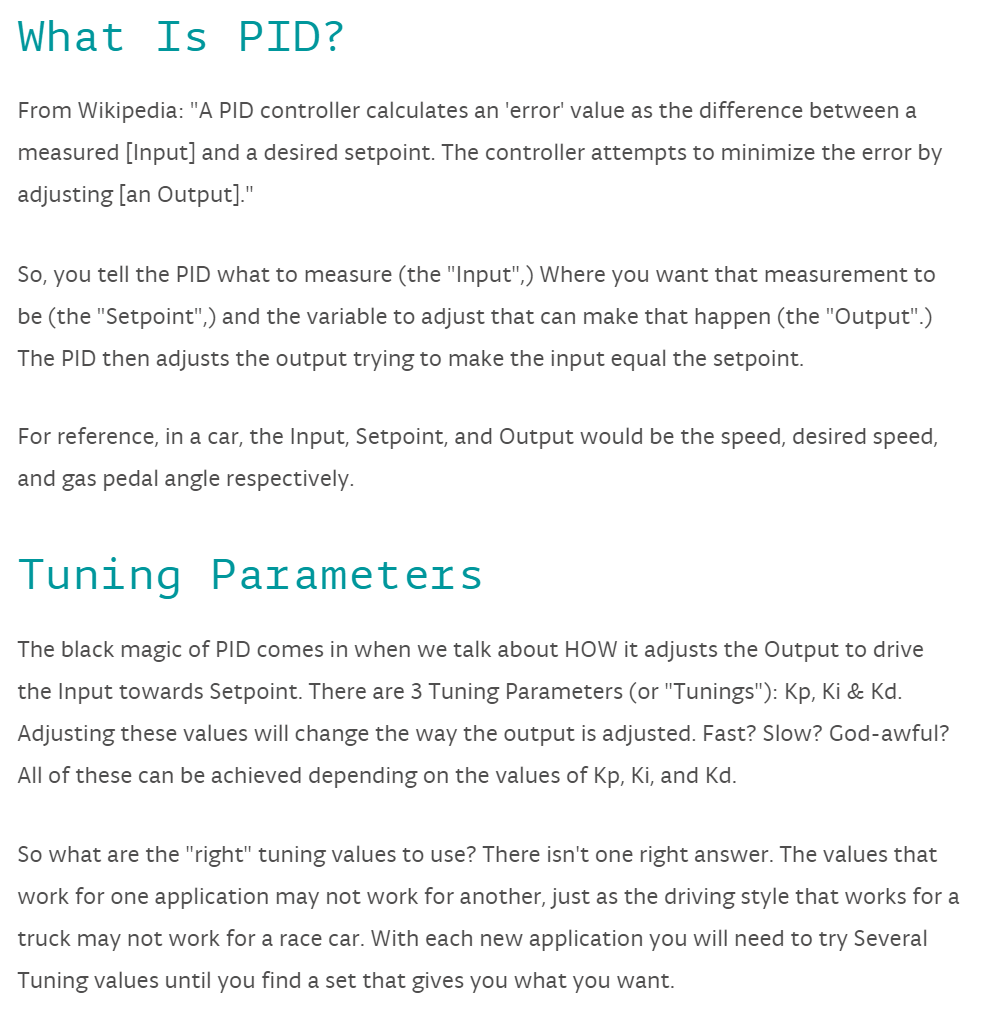
The majority of this document will explain what our project is, where we are, and what we need to do tonight. Please read this and use it as a prep before our meeting tonight.

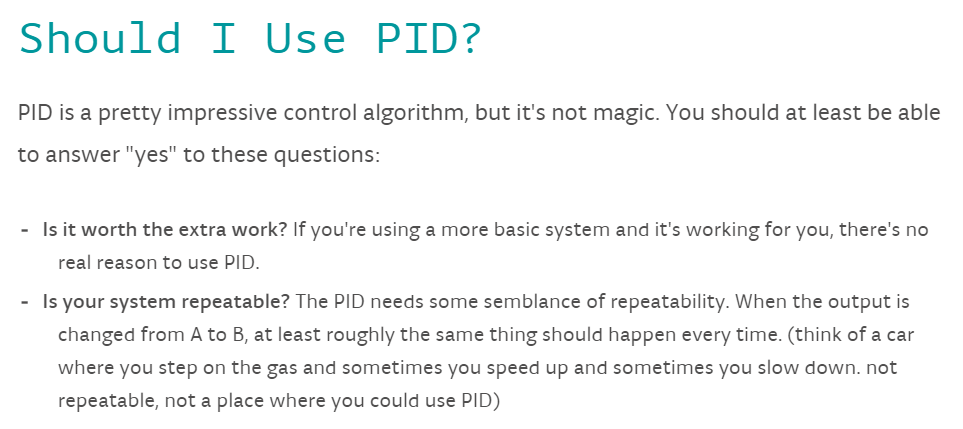
**Where we are**

First we need to clarify what our goal is. Right now, we have been instructed that we need to make a PID controller:

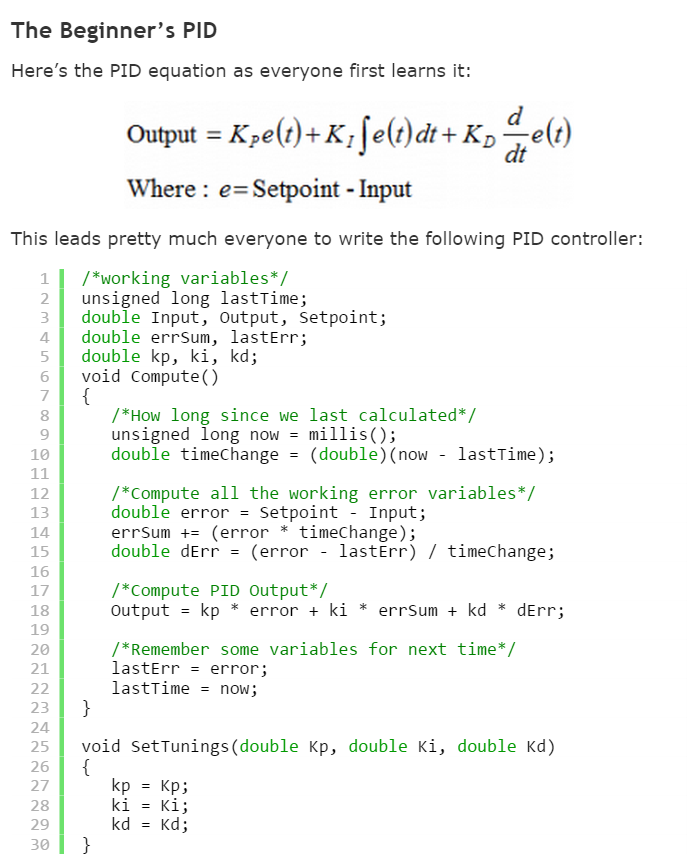
“As part of your project group, create a linear PID control system and testbench. Inputs to your controller should consist of a fixed-point control set point, fixed-point **measured output**, fixed point P, I and D control gains…”

In order for us to understand why a PID controller doesn’t make sense, we have to understand what a PID controller is:





So now that we have a small understanding of what a PID is, let’s look at some C code that defines a PID:



The PID equation is just a fancy way of saying the output will be adjusted if it is not equal to the “set point”. The set point is just a desired speed, voltage or whatever you want it to be

As you can see, the output is a continuous function of itself. Understand that last sentence because it is crucial to why we do not need a PID. A PID would be necessary for us if we were directly controlling the motors of the HUSKY. However, we are not sending PWM signals to these motors, the processor on board the HUSKY is.

What I am trying to say, is that the HUSKY has abstracted the implementation of its movement by providing us with an API, and command protocol. All we have to do is send a command to the HUSKY with the right parameters, and it will handle all motor and speed control.

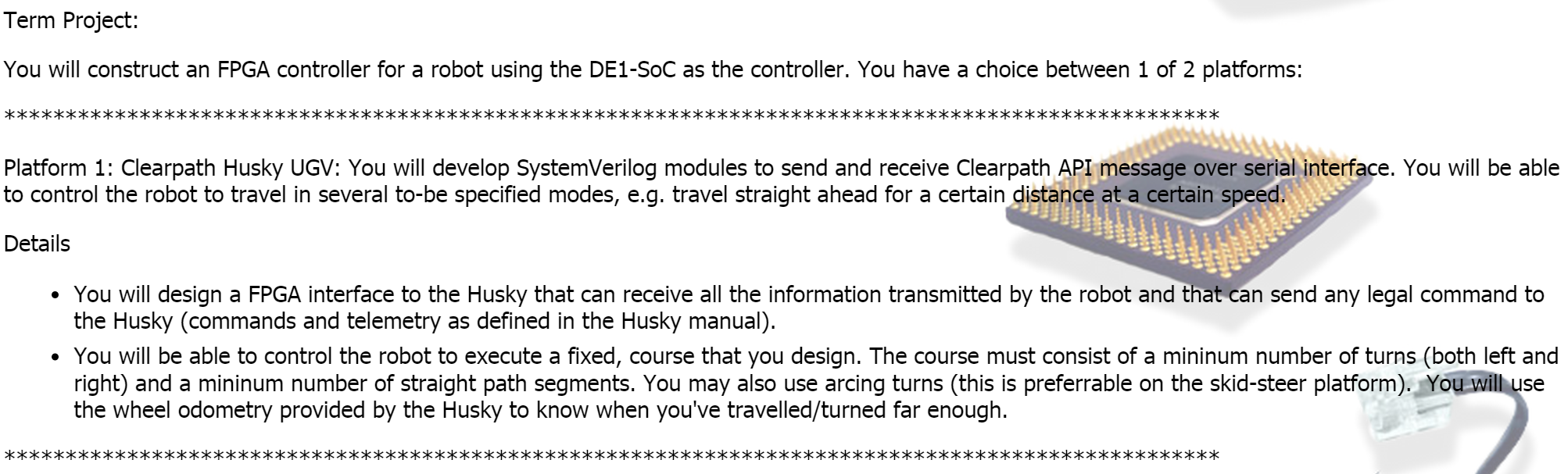
Students doing the flying king project need a PID because they don’t have commands that they can send to the air plane. Rather they are directly controlling all motors on board. Since we are not directly controlling the motors of the HUSKY we do not need a PID controller for them.

So now that we understand why we don’t need a PID, we need to figure out what we are supposed to do.

**What we need to do**

Now that we know that our homework 6 description does not describe our goal, lets figure out what our goal is. Since the PID controller for the flying king will be most of the project, we can assume that the professor wants us to complete a majority of our project for this homework.

That being said, what is the goal of our project? According to the project page on angel:

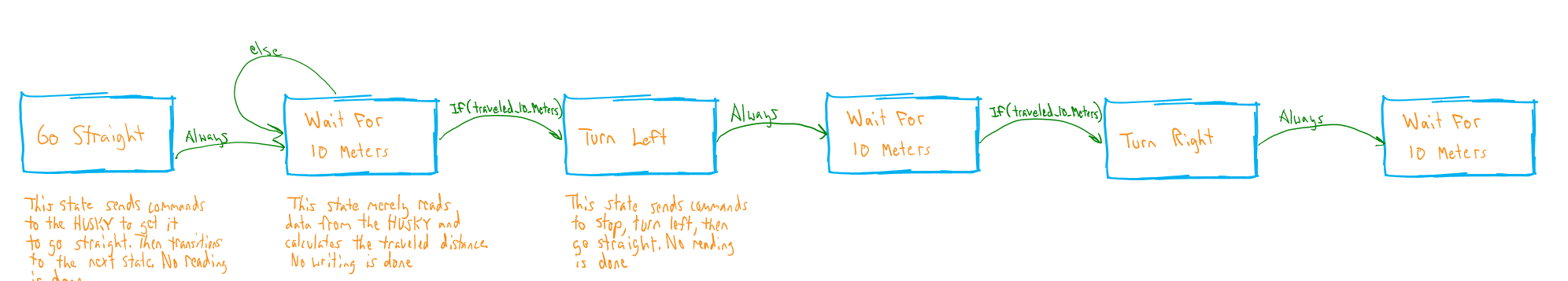


Since this is vague, I will more strictly define our goal. For our project we will want to accomplish the following things in order:

1. Set the HUSKY moving straight at a fixed speed (TBD) for 10 meters.
2. After ten meters the HUSKY will take a 90 degree left turn.
3. Then it will move straight for another 10 meters
4. After the next 10 meters, the HUSKY will make a 90 degree right turn
5. Finally, the HUSKY will move straight for another 10 meters

Now that we have defined our goal, we can have a more focused effort for our work.

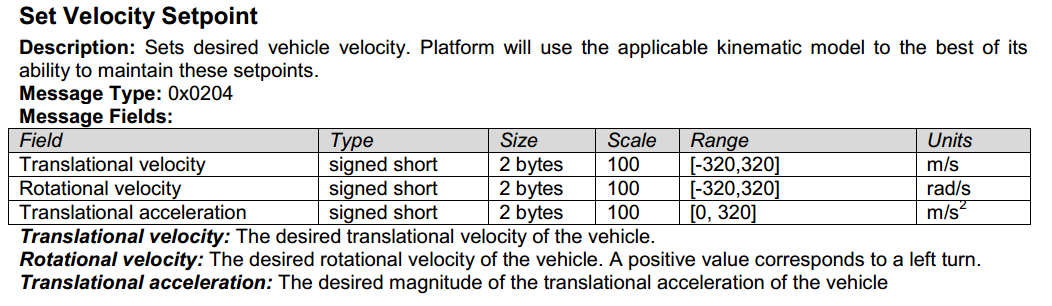
The way I see it, we will have a top level state machine, outlined below:



Now we just need to figure out what goes inside each state.

Go Straight:

Will use our transmitter to send a Set\_Velocity\_Setpoint command outlined in the protocol as:



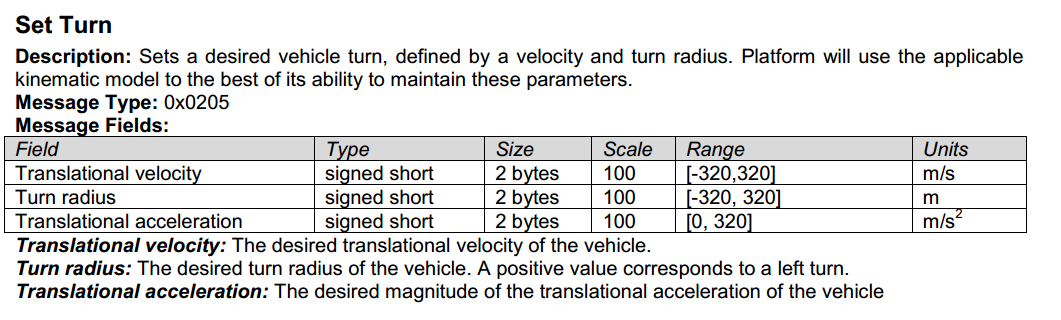
The translational velocity would have a value of 2, while rotational velocity and translational acceleration would be set to zero. This is all that will be required in this state. So we will have constants input to the transmitter that send this command

Wait for 10 meters:

This state will only look at incoming data. We will use the serial reading function we created to create messages that are received from the HUSKY. We will then figure out what part of this data we need, and then calculate the distance traveled. When the distance is 10 meters, we will set a flag that transitions to the next state.

Turn Left/Right:

To begin the turn we will send a Set\_Turn command outlined below:



Where velocity will be 1 m/s, radius will be 1 meter, and acceleration will be 0m/s^2. Following this command, we will then send another Set\_Turn command, ceasing the turning action. The parameters for this command will be a velocity of 2 m/s, a turn radius of 0 meters, and an acceleration of 0m/s^2. This will set the HUSKY to 2 m/s heading straight. We then automatically transition to the next state

**Conclusion**

We now have a good direction for tonight’s meeting. However, we have not discussed a key point. If you look at the command protocol document, you will see the structure that each message to the HUSKY must take. We will need to generate this structure for each message we send. We can discuss how to tackle this further.

At our meeting tonight I would like to accomplish:

1. Writing the code for our top level state machine
   1. Should be simple
2. Writing the code for the go straight state machine
   1. Will require research on command formatting
   2. Once we find the command format, we will need to figure out how to send the appropriate data to the transmitter module
3. Write the code for the wait for 10 meters state
   1. Will need to research what data we are receiving. This will require figuring out the message formatting
   2. We will then need to find out how to construct an incoming message
   3. After we construct a message we need to research which part of it we are interested in.
   4. Finally, we will need to use the data we want to compute distance
4. Write the code for the turning states
   1. Once we have the code for go straight, these states should be simple