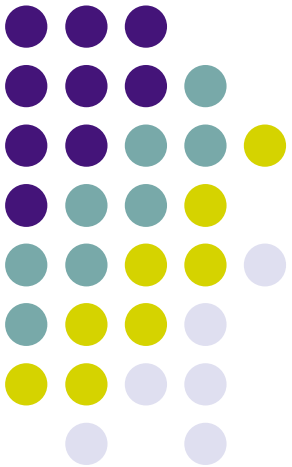
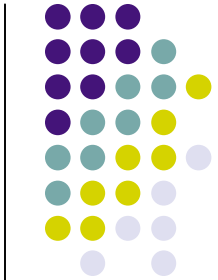
Programming FRC Robots

PID Control

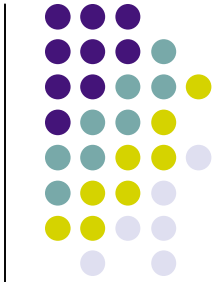


Presenters



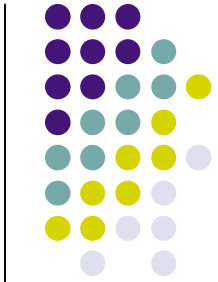
- Michael DiRamio:
  - Programming and Controls mentor on 1114 for the past 2 years.
  - High School teacher for District School Board of Niagara
  - E-mail: Michael.DiRamio@dsbn.edu.on.ca
- Tyler Holtzman:
  - Mechanical and Programming mentor on 1114 and 2056.
  - Started in 2003 as a student on 1114.
  - Software Engineering Student at McMaster University

Autonomous



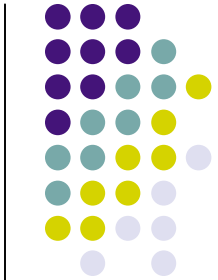
- For the last few years each game has started out with an autonomous period, in which the robots move based only on pre-programmed commands and sensor input.

Timer Based



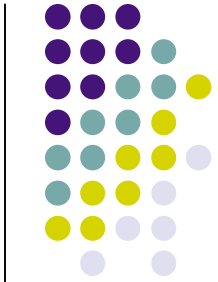
- Robot follows pre-programmed commands that each last for a given length of time.
- Time can be judged by either using the processors timers or by counting the number of cycles that have passed.

Common Sensors



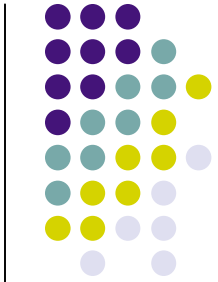
- Range Finders
- Light Sensors
- Limit Switches
- Hall Effect
- Encoders
- Gyros
- CMU Camera
- etc…..

Basic Sensor Use



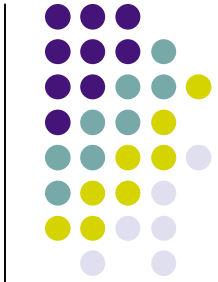
- There is always some value you are trying to "get" to.
- Go until you get to that value, then stop
- Problems: tends to overshoot the goal

PID Control



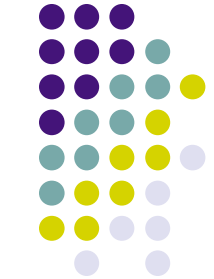
- PID stands for Proportional-Integral-Derivative
- A way of using what you know from your sensors to compute an “intelligent” motor output.
- Incorporates the idea that you need to slow down as you get close so you don’t overshoot the target.

Our Example



- We want to drive our robot forward 10 metres, and then stop.
- The concepts we explore do not always have to involve distance. (ie your goal may be to maintain a certain speed or certain rate of acceleration)

Proportional (P)

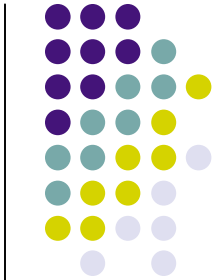


- A measure of how far you are away from your goal.
- The larger the P value, the larger the motor output should be, since you have farther to go.
- In our example this would be the distance remaining between our current position and our goal of 10 metres.

Calculating Distance

- An encoder or hall effect sensor can be used to keep track of how far a wheel has turned.
- Hall effect sensors (or “gear tooth counters”) signal the RC when a gear tooth passes in front of it. It is possible to calculate the distance each gear tooth covers.
- Encoders signal the RC for each part of a revolution turned. The number of signals per revolution varies.
- An advantage of encoders is that some (“quadrature encoders” are able to distinguish between forward and reverse movement)

Derivative (D)



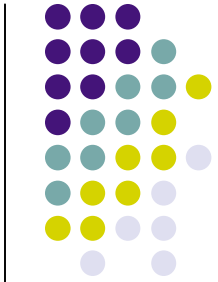
- How quickly you are moving towards your goal.
- Unlike P, when D is higher you want to “pull back” your motor output. This keeps you from heading towards you target too quickly. If P is still large (ie you are far from your target) then it will “overpower” the D.
- In our example this would be the speed we are traveling towards the goal.

Calculating Speed

- We have already seen how to keep track of the distance we have covered.
- Since cycles occur at set time intervals, it is possible to keep track of speed by calculating the distance travelled since the last cycle.

speed = last cycle distance - current cycle distance

Integral (I)

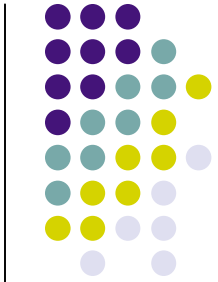


- The integral is how long you have been away from your target.
- The idea is that the longer the robot has not made it to the target, the more power should be applied to get it there.
- The most difficult part to incorporate. PD control alone can be very effective.

Measuring I

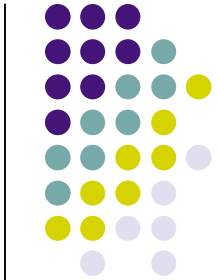
- I is the sum of the distances you are away each time. It can be calculated by adding up the P value from each loop.
- In actual use, this value can grow VERY quickly, especially with a cycle time of ~ 27ms.
- A useful alternative is to “cap” the contribution to I each cycle to 1.
  - If the robot is not at the target, add 1 to I.
  - If the robot is close to the target, reset I.

Calculating the Motor Output



- The basic formula is:
  output = $P * C_P + I * C_I - D * C_D$
- Where C_P, C_I, and C_D are values tweaked to get the proper result.
- There are some methods to calculate the values of C_P, C_I, and C_D, but it is generally more effective to find them by trial in error in robotics.

Method for Setting Values



- Start with C_P small and C_I, C_D both zero.
- Raise C_P until the robot is oscillating consistently around the target.
- Once this is accomplished, start increasing C_D until the robot stops oscillating.
- Then add C_I until the robot stops within a desired range of the target.