Rampaging Chariots

Requirements

# Technical Obligations

# I2c Motor commands

* Repetition for command message
  + 16ms (or board assumes contact lost or other failure -> safe stop)
* Payload bytes :
* 0 : Speed (equivalent to vertical stick)
* 1 : Rotation  (equivalent to horizontal stick)
* Value of 127 == stick centred.
* Motor controller will shutdown if demanded changes are too sudden (self protection)
  + Use 2 or 3 steps to go from min(127) to max-/+ (0 or 255)
  + (0 or 255 is FAST!)

# I2c Range sensor commands

* Message payload bytes :
* 0 : Control Code (?)
* 1 : Scan speed
* 2 : Left scan angle
* 3 : Right scan angle
* Byte 2= byte 3 : aim at angle, no scanning
* Angles are robot centric

# I2C Info

* Pi as i2c master. Initiates sends (e.g. motor and sensor control) and data requests
* Data-providing slaves (e.g. sensor outputs, odometers) do not initiate
* Data rate            115KBaud
* I2c message bytes – (This detail implicit in Python?)
  + 0 : Address
    - 7 bit number [in 7 top bits of transmitted byte]
    - [LSB :0 == request/write , 1==response]
  + 1 : Message payload size (0-255) demanded longitudinal speed
  + 2 : Payload byte (0 – 255) demanded lateral turn rate
  + N+2 : Payload byte N
* Following address byte,
* Range sensor on i2c address 8    ?

# I2C Odometer data readings format

* Left wheel odometer count, starts at 0                  16bit ? can roll over
* Right wheel odometer count, start at 0                  16bit ? can roll over
* Direction inferred from difference between L and R count

# Facts about the operating environment

# Chariot Dimensions

* Body Length: 400 + 50mm (page 21)
* Body Width: 310
* Wheelbase: front/rear axle centres: 200
* Track width: R/L mid wheel: 246?
* Wheel width: ???
* Wheel radius: 0.08m
* Wheel diameter: 150

# Waypoints

* [X,Y,Heading]

# Arena Coordinates

* Left=0 (X +ive rightwards )
* Bottom/near = 0(Y +ive upwards/far)
* Arena units
* Arena max X
* Arena max Y
* Angle reference : 0 along X axis; counter-clockwise in degrees

# Software Design Choices

# Visualiser

1. To show three lines
   1. One showing predicted line of movement
   2. One showing line from odometer reading
   3. One showing line based on reading from sensors (corrected)

* Different types of objects will have different collision result.

# Supporting detail