public Robot()

Note: The Robot class will take care of getting the UI/Field values from the Radio

Methods:

public int[] Scores() // returns all teams' scores

public int WinningScore() // returns score to win Note: may note be needed

public bool CanMove() // returns true if robot can move
Note: if false, Robot class will not allow any actuator change.

public bool IsAutonomous() // returns true if robot should be autonomous
Note: if true, Robot class will not allow user change.
Note: definition of auton could change to "limited usage"

public int[] UIAnalogVals() // returns interface's analog values
Note: the interface values are user-input (like joystick)

public int[] UIDigitalVals() // returns interface's digital values
Note: the interface values are user-input (like button)

public int[] FieldAnalogVals() // returns field's analog values
Note: the field values are user-input (like score, etc.)

public int[] FieldDigitalVals() // returns field's digital values
Note: the fields values are user-input (like other info about game)

public Radio(Robot robot, String port, String sixtyFourID, String sixteenID)

SimpleMotorController Class(Robot robot, String port, int device number)

Note: If daisychained with an already connected SimpleMotorController, pass the port name of the already existing one.

Properties:

double speed; // accepts between -100, full reverse, to 100, full forward.

double brake; // accepts between 0, coasting, to 10, full braking.

double maxSpeedForward; // accepts 0 to 100

double maxSpeedBackward; // accepts -100 to 0

double maxAccelerationForward; // accepts 0 to 100

double maxAccelerationBackward; // accepts -100 to 0

MicroMaestro Class(Robot robot, String port, int device number)

Note: If daisychained with an already connected MicroMaestro, pass the port name of the already existing one.
Note: Each property is an array of 6 elements, corresponding to the 6 different channels that servos could be connected on.
Note: defaults occur during initialization of instances

Properties:

double[6] minRotation; // assumed in degrees, defaults to 0 degrees

double[6] maxRotation; // assumed in degrees, defaults to 180 degrees

double[6] targets; // accepts from minRotation to maxRotation

double[6] speeds; // changes how fast servo rotates, accepts 0 to 100
Note: defaults to 50

double[6] accelerations; // changes how fast servo accelerates, accepts 0 to 100
Note: defaults to 50

Methods:

public void rotate(int channel, double degrees) // The argument degrees can be ±. If the rotation moves servo beyond rotation range, the servo will just turn to as extreme of a value it can.
Note: This method is primarily for the students' code's use.

Potentiometer Class(Robot robot, int pin, int rotationRange)

Note: rotationRange is measured in degrees

Methods:

public int getAngle() // returns the angle at which the pot currently is

public double getVoltage() // returns the voltage corresponding to the angle
Note: If the potentiometer is not calibrated, will return approximation

public void setMinVoltage(double min) // used to calibrate pot for min angle
Note: See sample code for examples to calibrate

public void setMaxVoltage(double max) // used to calibrate pot for max angle

ForceSensingResistor Class(Robot robot, int pin)

Note: this may change to be more specific to different types of FSRs

Methods:

public int getPressure() // returns the pressure on FSR (in g)

public double getVoltage() // returns the voltage corresponding to a pressure
Note: If the FSR is not calibrated, will return approximation

public void setMinVoltage(double min) // used to calibrate FSR for min pressure
Note: See sample code for examples to calibrate

public void setMaxVoltage(double max) // used to calibrate pot for max angle

IRLongDistanceTracker Class(Robot robot, int pin)

Note: minCm and maxCm are measured in cm
Note: This measures from 20-150 cm

Methods:

public double getDistance() // returns the distance (in cm) it is tracking

IRShortDistanceTracker Class(Robot robot, int pin)

Note: minCm and maxCm are measured in cm
Note: This measures from 4-30 cm

Methods:

public double getDistance() // returns the distance (in cm) it is tracking

SonarThickDistanceTracker Class(Robot robot, int pin)

Note: This has a thicker beam angle

Methods:

public double GetDistance() // returns the distance (in cm) it is tracking

SonarThinDistanceTracker Class(Robot robot, int pin)

Note: This has a thinner beam angle

Methods:

public double GetDistance() // returns the distance (in cm) it is tracking

Switch Class(Robot robot, int pin)

Methods:

public bool IsPressed() // true if pushed