## LocoXtreme AP CSA API Documentation

### API Classes:

* MessagesCodes
* GroundRobot
* RobotControl
* Plot2D

#### 

#### MessageCodes Class

##### Static Access Strings:

Sensor Type:

MessageCodes.D\_Ultrasonic

MessageCodes.D\_UltrasonicRaw

MessageCodes.D\_Accelerometer

MessageCodes.D\_AccelerometerRaw

MessageCodes.D\_GyroscopeRaw

MessageCodes.D\_MagnetometerRaw

MessageCodes.D\_Magnetometer

MessageCodes.D\_Temperature

MessageCodes.D\_Heading

MessageCodes.D\_RunningEncoders

Motor Direction:

MessageCodes.MD\_Forward

MessageCodes.MD\_Backward

Setup Wait Type:

MessageCodes.W\_Time

MessageCodes.W\_UltrasonicGreaterThan

MessageCodes.W\_UltrasonicLessThan

MessageCodes.W\_Song

MessageCodes.W\_Distance

MessageCodes.W\_Rotation

Note:

MessageCodes.N\_C

MessageCodes.N\_CSharp

MessageCodes.N\_D

MessageCodes.N\_EFlat

MessageCodes.N\_E

MessageCodes.N\_F

MessageCodes.N\_FSharp

MessageCodes.N\_G

MessageCodes.N\_GSharp

MessageCodes.N\_A

MessageCodes.N\_BFlat

MessageCodes.N\_B

Song:

MessageCodes.SO\_StarWars

MessageCodes.SO\_Rocky

MessageCodes.SO\_EyeOfTheTiger

MessageCodes.SO\_Mario

MessageCodes.SO\_Pokemon

#### RobotControl Class:

##### Setup Dongle Method

public void setup(String usb\_address)

* Parameters: String of the USB Port Address on which the Dongle is connected
* Returns: None

##### Scan for Robots

public ArrayList<GroundRobot> scan(long timeoutTime)

* Parameters: Duration for which to scan, in milliseconds
* Returns: ArrayList of detected robots

##### Connect to a Named Robot

public GroundRobot connect(String botName)

* Parameters: String of the name of the robot of which to connect
* Returns: GroundRobot object instance

##### Disconnect from a Robot

public void disconnect(GroundRobot robot)

* Parameters: Robot object from which to disconnect
* Returns: None

##### Listen for Robot Data

public void listen()

* Parameters: None
* Returns: None

##### Pause Main Thread Execution

public void waitTime(long waitTimeSpan)

* Parameters: Long value of time to wait, in milliseconds
* Returns: None

#### GroundRobot Class:

##### Activate Robot's Motors

public void activateMotors()

* Parameters: None
* Returns: None

##### Deactivate Robot's Motors

public void deactivateMotors()

* Parameters: None
* Returns: None

##### Enable/Disable A Sensor

public void enableSensor(String sensorName, int sensorState)

* Parameters:
  + String from MessageCodes of the Sensor to Enable
  + An integer value of 1 for enable or 0 for disable
* Returns: None

##### Set an LED's Color

public void setLight(int index, int red, int green, int blue)

* Parameters:
  + Integer of LED index (0 to 7)
  + Integer of red color value (0 to 255)
  + Integer of green color value (0 to 255)
  + Integer of blue color value (0 to 255)
* Returns: None

##### Set All LEDs to the Same Color

public void setLights(int red, int green, int blue)

* Parameters:
  + Integer of red color value (0 to 255)
  + Integer of green color value (0 to 255)
  + Integer of blue color value (0 to 255)
* Returns: None

##### Update LEDs to Currently Set Colors

public void syncLights()

* Parameters: None
* Returns: None

##### Change the Robot's Name

public void setName(String nameStr)

* Parameters: String of new robot name (No special characters, such as “:”, and no white-space)
* Returns: None

##### Get Unfiltered Ultrasonic Sensor Data

public int getUSDistanceCMFiltered()

* Parameters: None
* Returns: Integer of distance data in centimeters

##### Get Filtered Ultrasonic Sensor Data

public int getUSDistanceCMRaw()

* Parameters: None
* Returns: Integer of distance data in centimeters

##### Get Temperature Data

public float getTempCelsius()

* Parameters: None
* Returns: Float of temperature data in Celsius

##### Get Filtered X-Axis Accelerometer Data

public float getAccelXFiltered()

* Parameters: None
* Returns: Float of x-axis acceleration value in units of *g*

##### Get Filtered Y-Axis Accelerometer Data

public float getAccelYFiltered()

* Parameters: None
* Returns: Float of y-axis acceleration value in units of *g*

##### Get Filtered Z-Axis Accelerometer Data

public float getAccelZFiltered()

* Parameters: None
* Returns: Float of z-axis acceleration value in units of *g*

##### Get Unfiltered X-Axis Accelerometer Data

public float getAccelXRaw()

* Parameters: None
* Returns: Float of x-axis acceleration value in units of *g*

##### Get Unfiltered Y-Axis Accelerometer Data

public float getAccelYRaw()

* Parameters: None
* Returns: Float of y-axis acceleration value in units of *g*

##### Get Unfiltered Z-Axis Accelerometer Data

public float getAccelZRaw()

* Parameters: None
* Returns: Float of z-axis acceleration value in units of *g*

##### Get Unfiltered X-Axis Gyroscope Data

public float getGyroXRaw()

* Parameters: None
* Returns: Float of angular velocity about the x-axis in units of *degrees per second*

##### Get Unfiltered Y-Axis Gyroscope Data

public float getGyroYRaw()

* Parameters: None
* Returns: Float of angular velocity about the y-axis in units of *degrees per second*

##### Get Unfiltered Z-Axis Gyroscope Data

public float getGyroZRaw()

* Parameters: None
* Returns: Float of angular velocity about the z-axis in units of *degrees per second*

##### Get Filtered X-Axis Magnetometer Data

public float getMagnetometerXFiltered()

* Parameters: None
* Returns: Float of x-axis magnetic field value in units of *micro-tesla*

##### Get Filtered Y-Axis Magnetometer Data

public float getMagnetometerYFiltered()

* Parameters: None
* Returns: Float of y-axis magnetic field value in units of *micro-tesla*

##### Get Filtered Z-Axis Magnetometer Data

public float getMagnetometerZFiltered()

* Parameters: None
* Returns: Float of z-axis magnetic field value in units of *micro-tesla*

##### Get Unfiltered X-Axis Magnetometer Data

public float getMagnetometerXRaw()

* Parameters: None
* Returns: Float of x-axis magnetic field value in units of *micro-tesla*

##### Get Unfiltered Y-Axis Magnetometer Data

public float getMagnetometerYRaw()

* Parameters: None
* Returns: Float of y-axis magnetic field value in units of *micro-tesla*

##### Get Unfiltered Z-Axis Magnetometer Data

public float getMagnetometerZRaw()

* Parameters: None
* Returns: Float of z-axis magnetic field value in units of *micro-tesla*

##### Get Magnetometer-Based Heading Angle

public float getHeadingDegrees()

* Parameters: None
* Returns: Float of the heading angle in degrees (0 to 360)

##### Get Left Motor Encoder Tick Count

public int getLeftEncCount()

* Parameters: None
* Returns: Integer of the current left motor tick count

##### Get Right Motor Encoder Tick Count

public int getRightEncCount()

* Parameters: None
* Returns: Integer of the current right motor tick count

##### Configure the Robot for a Wait Type Command

public void setupWait(String sensor)

* Parameters:
  + String of the wait type, defined in the MessageCodes class
  + (optional, first parameter dependent) Integer value
* Returns: None

##### Instruct the Robot to Play a Note

public void playNote(String note, int duration, boolean wait)

* Parameters:
  + String of the note type, defined in the MessageCodes class
  + Integer of the note duration
  + Boolean of whether there has been a setup wait command to pause execution of the code while the note plays
* Returns: None

##### Instruct the Robot to Play a Song

public void playSong(String song, boolean wait)

* Parameters:
  + String of the song type, defined in the MessageCodes class
  + Boolean of whether there has been a setup wait command to pause execution of the code while the note plays
* Returns: None

##### Instruct the Robot to Move

public void move(String lmd, String rmd, float lms, float rms, boolean wait, boolean encoders)

* Parameters:
  + String of the left motor direction, defined in the MessageCodes class
  + String of the right motor direction, defined in the MessageCodes class
  + Float of the left motor speed (0 to 1)
  + Float of the right motor speed (0 to 1)
  + Boolean of whether there has been a setup wait command to pause execution of the code while the note plays
  + Boolean of whether encoders are enabled
* Returns: None

#### Plot2D Class

##### Set Curve Color For Plot

public void setColor(int r, int g, int b)

* Parameters:
  + Integer value of red color (0 to 255)
  + Integer value of green color (0 to 255)
  + Integer value of blue color (0 to 255)
* Returns: None

##### Set X-Axis Limits For Plot

public void setXLimits(double xMin, double xMax)

* Parameters:
  + Double minimum x-axis value
  + Double maximum x-axis value
* Returns: None

##### Set Y-Axis Limits For Plot

public void setYLimits(double yMin, double yMax)

* Parameters:
  + Double minimum y-axis value
  + Double maximum y-axis value
* Returns: None

##### Set Plot Data

public void addData(double[] curveX, double[] curveY)

* Parameters:
  + Double array of x-axis data
  + Double array of y-axis data
* Returns: None

##### Set Plot Title

public void setTitle(String titleString)

* Parameters: Title name String
* Returns: None

##### Set Plot X-Axis Label

public void setXLabel(String xLblString)

* Parameters: X-axis label name String
* Returns: None

##### Set Plot Y-Axis Label

public void setYLabel(String yLblString)

* Parameters: Y-axis label name String
* Returns: None

##### Create Plot

public void plotData()

* Parameters: None
* Returns: None