ExoNaut Camera Libraries

ExoNaut_AICam

General ExoNaut AI Cam Functions:

This library interfaces with a computer vision camera module for the ExoNaut robot. It provides functions for various vision capabilities including face detection, object detection, classification, QR/barcode scanning, color detection, line following, landmark recognition, and number recognition. The library allows changing between different vision modes and retrieving detected information through I2C communication.

ExoNaut_AICam.begin() - Initializes the camera module and sets up I2C communication.

ExoNaut_AICam.readFromAddr() - Reads data from a specific memory address on the camera.

ExoNaut_AICam.writeToAddr() - Writes data to a specific memory address on the camera.

ExoNaut_AICam.firmwareVersion() - Retrieves the firmware version of the camera.

ExoNaut_AICam.currentFunc() - Gets the current application mode that the camera is operating in.

ExoNaut_AICam.changeFunc() - Changes the current application mode to a new function.

ExoNaut_AICam.setLed() - Turns the camera's LED on or off.

updateResult() - Updates all recognition results from the camera based on current mode.

Face Detection Functions:

anyFaceDetected() - Checks if any face has been detected.

numOfTotalFaceDetected() - Returns the total number of faces detected.

anyLearnedFaceDetected() - Checks if any previously learned face has been detected.

numOfTotalLearnedFaceDetected() - Returns the number of learned faces detected.

anyUnlearnedFaceDetected() - Checks if any unlearned face has been detected.

numOfTotalUnlearnedFaceDetected() - Returns the number of unlearned faces detected.

faceOfIdDetected() - Checks if a specific face ID has been detected.

getFaceOfId() - Gets information about a face with a specific ID.

getFaceOfIndex() - Gets information about an unidentified face at a specific index.

Object Detection Functions:

anyObjDetected() - Checks if any object has been detected.

numOfObjDetected() - Returns the number of objects detected.

objIdDetected() - Checks if a specific object ID has been detected.

numOfObjIdDetected() - Returns the number of instances of a specific object ID detected.

objDetected() - Gets information about a specific object.

Image Classification Functions:

classIdOfMaxProb() - Returns the ID of the class with highest probability.

classMaxProb() - Returns the highest probability value from classification.

classProbOfId() - Returns the probability for a specific class ID.

Feature Learning Functions:

featureIdOfMaxProb() - Returns the ID of the feature with highest probability.

featureMaxProb() - Returns the highest probability value for features.

featureProbOfId() - Returns the probability for a specific feature ID.

QR Code Functions:

qrCodeDetected() - Checks if a QR code has been detected.

qrCodeDataLength() - Returns the length of data in the detected QR code.

qrCodeData() - Retrieves the data from the detected QR code.

Line Following Functions:

anyLineDetected() - Checks if any line has been detected.

numOfLineDetected() - Returns the number of lines detected.

lineIdDetected() - Checks if a specific line ID has been detected.

lineId() - Gets information about a specific detected line.

Landmark Recognition Functions:

anyLandmarkDetected() - Checks if any landmark has been detected.

numOfLandmarksDetected() - Returns the number of landmarks detected.

landmarkIdDetected() - Checks if a specific landmark ID has been detected.

numOfLandmarkIdDetected() - Returns the number of instances of a specific landmark ID.

getLandmarkById() - Gets information about a specific landmark.

landmarkIdWithMaxProb() - Returns the ID of the landmark with highest probability.

landmarkMaxProb() - Returns the highest probability value for landmarks.

landmarkProbOfId() - Returns the probability for a specific landmark ID.

Number Recognition Functions:

numberWithMaxProb() - Returns the number with highest recognition probability.

numberMaxProb() - Returns the highest probability value for number recognition.

numberProbOfId() - Returns the probability for a specific number ID.

ExoNaut_AccelGyro:

This library manages the MPU6050 accelerometer and gyroscope sensor on the ExoNaut robot. It handles initialization, calibration, and provides processed motion data including orientation (pitch, roll, yaw), acceleration, rotation rates, and temperature. It includes features like shake detection and surface orientation detection through sensor fusion algorithms.

ExoNautAccelGyro() - Constructor that initializes member variables for the accelerometer/gyroscope module.

writeMPU6050Register() - Writes a value to a specified register on the MPU6050 sensor.

readMPU6050Register() - Reads a value from a specified register on the MPU6050 sensor.

readMPU6050Data() - Reads raw acceleration, temperature, and gyroscope data from the MPU6050.

beginIMU() - Initializes the MPU6050 sensor by configuring its registers and verifying its identity.

updateIMU() - Reads the latest sensor data and updates the internal data structure with calibrated values.

calibrateIMU() - Performs gyroscope calibration by taking multiple samples while the device is stationary.

calculateOrientation() - Computes pitch, roll, and yaw angles using sensor fusion algorithms.

ExoNaut_DotMatrix:

This library controls a TM1640-based LED dot matrix display for the ExoNaut robot. It provides functions for displaying numbers, characters, and scrolling text, with support for different brightness levels and animations. The library includes a built-in font for characters and digits and manages the complexity of interfacing with the LED matrix hardware.

ExoNaut_DotMatrix(uint8_t port) - Constructor that initializes the dot matrix display with specific pins based on port number.

setPins(uint8_t clkPin, uint8_t dinPin) - Manually configures the clock and data pins for custom configurations.

begin() - Initializes the display hardware and prepares it for use.

clear() - Turns off all LEDs and clears the display buffer.

setBrightness(uint8_t brightness) - Adjusts the brightness level of the display.

setAllOn() - Turns on all LEDs in the display.

setAllOff() - Turns off all LEDs (same as clear).

setLED(uint8_t row, uint8_t col, bool state) - Controls the state of an individual LED.

setRow(uint8_t row, uint8_t pattern) - Sets an entire row of LEDs using a byte pattern.

setMatrix(const uint8_t data[]) - Updates the entire display at once with a pattern array.

clearBuffer() - Clears the internal display buffer without affecting the physical display.

setPixel(int x, int y, bool state) - Sets a pixel in the internal buffer.

drawDigit(int digit, int xOffset, int yOffset) - Draws a numeric digit (0-9) at specified position.

drawChar(char character, int xOffset, int yOffset) - Draws an ASCII character at specified position.

displayNumber(uint8_t number) - Shows a number (0-99) on the display.

displayNumberWithEffect(uint8_t number) - Shows a number with animation effects.

getCharWidth(char character) - Returns the pixel width of a character
calculateTextPixelWidth(const char *text) - Calculates the total width of a text string in pixels.

scrollText(const char *text, uint8_t numScrolls, uint8_t scrollSpeed) - Starts scrolling text across the display.

stopScroll() - Halts the current scrolling text animation.

isScrolling() - Checks if text is currently scrolling.

updateScroll() - Updates the scrolling animation (called regularly from the main loop).

sendCommand(uint8_t cmd) - Sends a command to the TM1640 controller.

startTransmission() - Begins a data transmission sequence to the display controller.

endTransmission() - Completes a data transmission sequence.

sendData(uint8_t address, uint8_t data) - Sends data to a specific address on the display.

sendData(uint8_t data) - Sends a single byte of data to the display controller.

ExoNaut_Knob:

This library handles analog potentiometer (knob) input for the ExoNaut robot. It provides functions to read raw values, map them to specific ranges, detect changes, and implement value smoothing to reduce noise. It automatically configures the correct input pin based on the specified port number.

ExoNaut_Knob() - Constructor that initializes member variables for the analog knob interface.

begin(uint8_t port) - Sets up the knob based on the port number, configuring the correct pin and taking initial readings.

readRaw(void) - Reads the raw analog value from the knob (0-4095 on ESP32).

readMapped(int minVal, int maxVal) - Reads the knob and maps its value to a specific range.

setRange(int minVal, int maxVal) - Sets the default output range for the knob readings.

readPercent(void) - Reads the knob and returns its position as a percentage (0-100%).

hasChanged(int threshold) - Detects if the knob value has changed by more than the specified threshold.

setSmoothing(bool enable, int samples) - Enables or disables value smoothing and sets the number of samples to average.
update(void) - Updates the current knob reading and the smoothing buffer.

calculateSmoothedValue(void) - Computes an averaged value from the samples in the smoothing buffer.

ExoNaut_LineFollower:

This library interfaces with line following sensors on the ExoNaut robot. It provides I2C communication functions to read the state of four sensor channels that detect lines beneath the robot, allowing it to follow paths or detect boundaries.

wireWriteByte(uint8_t addr, uint8_t val) - Writes a single byte to an I2C device at the specified address.

wireWriteDataArray(uint8_t addr, uint8_t reg, uint8_t *val, unsigned int len) - Writes multiple bytes to a specific register of an I2C device.

wireReadDataArray(uint8_t addr, uint8_t reg, uint8_t *val, unsigned int len) - Reads a specified number of bytes from a register on an I2C device.

readLineFollower(uint8_t &val) - Reads the current state of the line follower sensors and updates the individual channel values (CH1-CH4).

ExoNaut_MP3:

This library controls a Hiwonder MP3 module on the ExoNaut robot. It manages audio playback with functions for playing, pausing, skipping tracks, adjusting volume, and enabling loop mode. It communicates with the module via I2C and maintains track of the current state.

begin(void) - Initializes the MP3 module, sets initial volume, and detects the total number of tracks available.

sendCommand(uint8_t cmd) - Helper function that sends a simple command to the MP3 module with no parameters.

readResponseData(uint8_t cmd, uint8_t *buffer, uint8_t length) - Helper function that reads response data from the MP3 module.

sendCommandWithParam(uint8_t cmd, uint8_t param) - Helper function that sends a command with a single byte parameter.

sendCommandWithParam16(uint8_t cmd, uint16_t param) - Helper function that sends a command with a 16-bit parameter.

play(void) - Starts or resumes playback of the current track.

pause(void) - Pauses the current playback.
next(void) - Skips to the next track.

previous(void) - Returns to the previous track.

setVolume(uint8_t volume) - Sets the playback volume (0-30).

getVolume(void) - Returns the current volume setting.

volumeUp(void) - Increases the volume by one step.

volumeDown(void) - Decreases the volume by one step.

playTrack(uint16_t trackNumber) - Plays a specific track by number.

getCurrentTrack(void) - Returns the current track number.

enableLoop(bool enable) - Enables or disables track looping.

isLoopEnabled(void) - Checks if loop mode is currently enabled.

isPlaying(void) - Checks if the MP3 module is currently playing.

getTotalTracks(void) - Returns the total number of tracks available.

detectTotalTracks(void) - Detects and updates the count of available tracks on the module.

ExoNaut_RGB_LED:

This library controls NeoPixel RGB LEDs on the ExoNaut robot. It provides functions to set individual LED colors, set all LEDs to the same color, update the display, and clear all LEDs. It automatically configures based on the port number specified.

begin(uint8_t port) - Initializes the RGB LED module based on the specified port number, setting up the correct pin and preparing the NeoPixel library.

setColor(uint16_t n, uint8_t r, uint8_t g, uint8_t b) - Sets a specific LED (by index) to a particular RGB color.

setColorAll(uint8_t r, uint8_t g, uint8_t b) - Sets all LEDs to the same RGB color and updates the display.

show(void) - Updates the physical LED display with the current color settings.

clear(void) - Turns off all LEDs (sets them to black)

ExoNaut_TempHumid:

This library interfaces with a temperature and humidity sensor (AHT10/AHT20) on the ExoNaut robot. It provides functions to initialize the sensor and read temperature (in both Celsius and Fahrenheit) and relative humidity values.

begin() - Initializes the temperature and humidity sensor (AHT10/AHT20) and returns whether initialization was successful
readSensor() - Reads the current temperature and humidity data from the sensor and updates the internal class variables:

- temperatureC (temperature in Celsius)
- temperature (temperature in Fahrenheit)
- humidity (relative humidity percentage)

ExoNaut_UltraSonic:

This library controls an ultrasonic distance sensor with integrated RGB LEDs. It provides functions to read distance measurements in centimeters and control the RGB LEDs with solid colors or breathing effects. It communicates with the sensor module via I2C.

wireWriteByte(uint8_t addr, uint8_t val) - Helper function that writes a single byte to an I2C device at the specified address.

wireWriteDataArray(uint8_t addr, uint8_t reg, uint8_t *val, unsigned int len) - Helper function that writes multiple bytes to a specific register of an I2C device.

wireReadDataArray(uint8_t addr, uint8_t reg, uint8_t *val, unsigned int len) - Helper function that reads a specified number of bytes from a register on an I2C device.

breathing(uint8_t r1, uint8_t g1, uint8_t b1, uint8_t r2, uint8_t g2, uint8_t b2) - Sets the ultrasonic sensor's RGB LEDs to breathing light mode with specified color cycles for left and right LEDs.

color(uint8_t r1, uint8_t g1, uint8_t b1, uint8_t r2, uint8_t g2, uint8_t b2) - Sets the color of the ultrasonic sensor's RGB LEDs with specified RGB values for left and right LEDs.

getDistance() - Reads the distance measured by the ultrasonic sensor in centimeters.

getColor() - Reads the current color settings of the RGB LEDs and stores them in the class's rgb1 and rgb2 arrays.

getBreathing() - Appears to be an unimplemented function (empty body) intended to read breathing mode settings.

getBrightness() - Appears to be an unimplemented function (returns 0) intended to read the brightness setting of the LEDs.

getLEDMode() - Appears to be an unimplemented function (returns 0) intended to read the current operating mode of the LEDs