e-puck2_cpp_remote_library

Generated by Doxygen 1.7.6.1

Fri Nov 15 2019 11:00:47

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Epuck2 Class Reference

`#include <epuck2.h>`

Public Member Functions

- int8_t startCommunication (char *portName)

  To be called once at the beginning, it handles the communication with the robot (receive sensors data and send actuators values).
- void stopCommunication (void)

  To be called once at the end, it closes the Bluetooth communication and stop the communication thread.
- char * getPortName (void)

  This function can be used to know what serial port the robot is connected.
- uint8_t **waitForUpdate** (uint32_t us)
- int16_t getAccelerometerRaw (uint8_t id)

  Accelerometer raw axes values.
- float getAcceleration (void)

  Accelerometer magnitude.
- float getOrientation (void)

  Orientation of the robot.
- float getInclination (void)

  Inclination of the robot.
- int16_t getGyroRaw (uint8_t id)

  Gyroscope raw axes values.
- float getMagneticField (uint8_t id)

  Magnetometer raw axes values.
- int8_t getTemperature (void)

  Temperature given from the IMU.

- uint16_t getProximity (uint8_t id)

  IR proximity sensors values.
- uint16_t getAmbient (uint8_t id)

  IR proximity sensors values (ambient light measure).
- uint16_t getDistanceMillimeters (void)

  Time of flight sensor value.
- uint16_t getMicVolume (uint8_t id)

  Microphones volume.
- int16_t getMotorSteps (uint8_t id)

  Motors steps: 1000 steps per wheel revolution.
- uint16_t getBatteryRaw (void)

  Battery sampled value from the ADC.
- uint8_t getSdState (void)

  Micro SD state.
- uint8_t getTvRemoteData (void)

  TV remote received data (RC5 protocol).
- uint8_t getSelector (void)

  Selector position (16 positions available).
- uint16_t getGroundProximity (uint8_t id)

  IR proximity sensors values from the ground sensors extension.
- uint16_t getGroundAmbient (uint8_t id)

  IR proximity sensors values (ambient light measure) from the ground sensors extension.
- void setSpeed (int16_t left, int16_t right)

  Set motors speed.
- void setLed (uint8_t id, uint8_t state)

  Turn on/off the LEDs around the robots, the body LED and the front LED. Refer to https://www.gctronic.com/doc/index.php?title=e-puck2#Overview *for the position of the LEDs.*
- void setRgbLeds (uint8_t red2, uint8_t green2, uint8_t blue2, uint8_t red4, uint8_t green4, uint8_t blue4, uint8_t red6, uint8_t green6, uint8_t blue6, uint8_t red8, uint8_t green8, uint8_t blue8)

  Set the color and power of the RGB LEDs around the robots. Refer to https://www.gctronic.com/doc/index.php?title=e-puck2#Overview *for the position of the RGB LEDs.*
- void setSound (uint8_t id)

  Start playing predefined sound tracks or specific frequencies. The play can also be stopped.

2.1.1 Detailed Description

The Epuck2 class represents an e-puck2 robot.

Author

Stefano Morgani

Version

1.0

Date

07/11/19

2.1.2 Member Function Documentation

2.1.2.1 float Epuck2::getAcceleration (void)

Accelerometer magnitude.

Returns

values between 0.0 and about 2600.0 (about 3.45 g).

2.1.2.2 int16_t Epuck2::getAccelerometerRaw (uint8_t *id*)

Accelerometer raw axes values.

Parameters

id	X_AXIS, Y_AXIS or Z_AXIS.

Returns

values between -1500 and 1500, resolution is +/- 2g.

2.1.2.3 uint16_t Epuck2::getAmbient (uint8_t *id*)

IR proximity sensors values (ambient light measure).

Parameters

id	id of the proximity from which to get the value, refer to https://www.gctronic.com/doc/index.php?title=e-puck2#Overview for the position of the proximity sensors.

Returns

values between 0 (strong light) and 4095 (dark).

2.1.2.4 uint16_t Epuck2::getBatteryRaw (void)

Battery sampled value from the ADC.

Returns

sampled value.

2.1.2.5 uint16_t Epuck2::getDistanceMillimeters (void)

Time of flight sensor value.

Returns

distance measured in millimeters.

2.1.2.6 uint16_t Epuck2::getGroundAmbient (uint8_t *id*)

IR proximity sensors values (ambient light measure) from the ground sensors extension.

Parameters

id	id of the proximity from which to get the value: GROUND_LEFT, GROUND_CENTER, GROUND_RIGHT.

Returns

values between 0 (strong light) and 1023 (dark).

2.1.2.7 uint16_t Epuck2::getGroundProximity (uint8_t *id*)

IR proximity sensors values from the ground sensors extension.

Parameters

id	id of the proximity from which to get the value: GROUND_LEFT, GROUND_CENTER, GROUND_RIGHT.

Returns

values between 0 (no surface at all or not reflective surface e.g. black) and 1023 (very reflective surface e.g. white).

2.1.2.8 int16_t Epuck2::getGyroRaw (uint8_t *id*)

Gyroscope raw axes values.

Parameters

id	X_AXIS, Y_AXIS or Z_AXIS.

Returns

values between -32768 and 32767, range is +/- 250 dps.

2.1.2.9 float Epuck2::getInclination (void)

Inclination of the robot.

Returns

values between 0.0 and 90.0 degrees (when tilted in any direction)

2.1.2.10 float Epuck2::getMagneticField (uint8_t *id*)

Magnetometer raw axes values.

Parameters

id	X_AXIS, Y_AXIS or Z_AXIS.

Returns

values in float, range is +/- 4912.0 uT (magnetic flux density expressed in micro Tesla).

2.1.2.11 uint16_t Epuck2::getMicVolume (uint8_t *id*)

Microphones volume.

Parameters

id	MIC_RIGHT, MIC_LEFT, MIC_BACK, MIC_FRONT.

Returns

values between 0 and 4095.

2.1.2.12 int16_t Epuck2::getMotorSteps (uint8_t *id*)

Motors steps: 1000 steps per wheel revolution.

Parameters

id	LEFT, RIGHT.

Returns

motor steps.

2.1.2.13 float Epuck2::getOrientation (void)

Orientation of the robot.

Returns

values between 0.0 and 360.0 (degrees).

2.1.2.14 char * Epuck2::getPortName (void)

This function can be used to know what serial port the robot is connected.

Returns

serial port name.

2.1.2.15 uint16_t Epuck2::getProximity (uint8_t *id*)

IR proximity sensors values.

Parameters

id	id of the proximity from which to get the value, refer to https://www.gctronic.com/doc/index.php?title=e-puck2#Overview for the position of the proximity sensors.

Returns

values between 0 (no objects detected) and 4095 (object near the sensor).

2.1.2.16 uint8_t Epuck2::getSdState (void)

Micro SD state.

Returns

1 if the micro SD is present and can be read/write, 0 otherwise.

2.1.2.17 uint8_t Epuck2::getSelector (void)

Selector position (16 positions available).

Returns

values between 0 and 15.

2.1.2.18 int8_t Epuck2::getTemperature (void)

Temperature given from the IMU.

Returns

Celsius degrees.

2.1.2.19 uint8_t Epuck2::getTvRemoteData (void)

TV remote received data (RC5 protocol).

Returns

received command.

2.1.2.20 void Epuck2::setLed (uint8_t *id*, uint8_t *state*)

Turn on/off the LEDs around the robots, the body LED and the front LED. Refer to https://www.gctronic.com/doc/index.php?title=e-puck2#-Overview for the position of the LEDs.

Parameters

id	LED1, LED3, LED5, LED7, BODY_LED, FRONT_LED.
state	0=off, 1=on

Returns

none.

2.1.2.21 void Epuck2::setRgbLeds (uint8_t *red2,* uint8_t *green2,* uint8_t *blue2,* uint8_t *red4,* uint8_t *green4,* uint8_t *blue4,* uint8_t *red6,* uint8_t *green6,* uint8_t *blue6,* uint8_t *red8,* uint8_t *green8,* uint8_t *blue8*)

Set the color and power of the RGB LEDs around the robots. Refer to https://www.gctronic.com/doc/index.php?title=e-puck2#Overview for the position of the RGB LEDs.

Parameters

red2	RGB LED2 red component (0..100).
green2	RGB LED2 green component (0..100).
blue2	RGB LED2 blue component (0..100).
red4	RGB LED4 red component (0..100).
green4	RGB LED4 green component (0..100).
blue4	RGB LED4 blue component (0..100).
red6	RGB LED6 red component (0..100).
green6	RGB LED6 green component (0..100).
blue6	RGB LED6 blue component (0..100).
red8	RGB LED8 red component (0..100).
green8	RGB LED8 green component (0..100).
blue8	RGB LED8 blue component (0..100).

Returns

none.

2.1.2.22 void Epuck2::setSound (uint8_t *id*)

Start playing predefined sound tracks or specific frequencies. The play can also be stopped.

Parameters

id	SOUND_MUTE (no sound), SOUND_MARIO, SOUND_UNDERWORLD, SOUND_STARWARS, SOUND_4KHZ, SOUND_10KHZ, SOUND_STOP (stop the currently played sound if any).

Returns

none.

2.1.2.23 void Epuck2::setSpeed (int16_t *left*, int16_t *right*)

Set motors speed.

Parameters

left	desired left speed (-1000..1000)
right	desired right speed (-1000..1000)

Returns

none.

2.1.2.24 int8_t Epuck2::startCommunication (char * *portName*)

To be called once at the beginning, it handles the communication with the robot (receive sensors data and send actuators values).

Parameters

portName	name of the serial port, must be null terminated.

Returns

0 if communication started correctly, $<$ 0 in case of errors.

2.1.2.25 void Epuck2::stopCommunication (void)

To be called once at the end, it closes the Bluetooth communication and stop the communication thread.

Returns

none

The documentation for this class was generated from the following files:

- epuck2.h
- epuck2.cpp

2.2 SerialComm Class Reference

Class handling the serial communication under Linux/MacOS systems.

`#include <SerialComm.h>`

Public Member Functions

- int connect (char *path)
- void disconnect ()
- int32_t writeData (char *buf, int num_bytes, int usleep_time)
- int32_t readData (char *buf, int num_bytes, uint32_t usleep_time)
- void flush ()
- void discard (int num_bytes)

2.2.1 Detailed Description

Class handling the serial communication under Linux/MacOS systems.

Author

Stefano Morgani

Version

1.0

Date

11/18/08

This class contains all the functions needed to communicate with the E-Puck robot.

2.2.2 Member Function Documentation

2.2.2.1 int SerialComm::connect (char * *path*)

This function is used to initialize and open the connection to the bluetooth device; the serial settings are: 115200-8-N-1 and no hw/sw control.

Parameters

path	string indicating the path of the device to which the connection will be established (e.g. /dev/tty.e-puck_1594-COM1-1).

Returns

returns the file descriptor returned by the "open" function.

2.2.2.2 void SerialComm::discard (int *num_bytes*)

This function is used to discard data from the input buffer; note that this function is blocking, that is it will continue to read data from the buffer until all the requested bytes are discarded.

Parameters

num_bytes	the number of bytes to be discarded from the input buffer.

2.2.2.3 void SerialComm::disconnect ( )

This function closes the connection opened.

2.2.2.4 void SerialComm::flush ( )

This function flushes both the input and output buffers.

2.2.2.5 int32_t SerialComm::readData (char * *buf,* int *num_bytes,* uint32_t *usleep_time*)

This function is used to receive data from the device.

Parameters

buf	buffer that will contain the data received from the device.
num_bytes	the number of bytes to be read from the device (maximum length of the buffer).
usleep_time	specifies the total waiting time before exiting the reading operation in case of errors; the value is specified in microseconds unit.

Returns

returns the number of bytes correctly red from the device or -1 in case of error..

2.2.2.6 int32_t SerialComm::writeData (char * *buf,* int *num_bytes,* int *usleep_time*)

This function is used to send data to the device.

Parameters

buf	buffer that contains the characters (command) to be written to the serial port.
num_bytes	the number of bytes that will be written to the serial port (length of the buffer).
usleep_time	specifies the total waiting time before exiting the writing operation in case of errors; the value is specified in microseconds unit.

Returns

returns the number of bytes correctly written to the serial port or -1 in case of error.

The documentation for this class was generated from the following files:

- SerialComm.h
- SerialComm.cpp

Index