189 lines (139 sloc) 10.7 KB

Arduino Joystick Library

Version 2.0.5

This library can be used with Arduino IDE 1.6.6 (or above) to add one or more joysticks (or gamepads) to the list of HID devices an Arduino Leonardo or Arduino Micro (or any Arduino clone that is based on the ATmega32u4) can support. This library will also work with the Arduino Due, thanks to @Palakis. A complete list of supported boards can be found in the Wiki. This will not work with Arduino IDE 1.6.5 (or below) or with non-32u4 based Arduino devices (e.g. Arduino UNO, Arduino MEGA, etc.).

Features

The joystick or gamepad can have the following features:

- Buttons (default: 32)
- Up to 2 Hat Switches
- X, Y, and/or Z Axis (up to 16-bit precision)
- X, Y, and/or Z Axis Rotation (up to 16-bit precision)
- Rudder (up to 16-bit precision)
- Throttle (up to 16-bit precision)
- Accelerator (up to 16-bit precision)
- Brake (up to 16-bit precision)
- Steering (up to 16-bit precision)

Installation Instructions

Copy the `Joystick` folder to the Arduino libraries folder. Once the folder has been copied, the Joystick library should appear in the Arduino IDE list of libraries. The examples should also appear in the examples menu in the Arduino IDE.

Microsoft Windows

On Microsoft Windows machines, this is typically `%userprofile%\Documents\Arduino\libraries`. The `deploy.bat` file can be executed to install the Joystick folder on Microsoft Windows machines (assuming a default Arduino installation).

Linux

On Linux machines, this is typically `$HOME/Arduino/libraries`. The `deploy.sh` file can be executed to install the Joystick folder on Linux machines (assuming a default Arduino installation). [Thanks to @Nihlus (Jarl Gullberg) for his help with this.]

Examples

The following example Arduino sketch files are included in this library:

- `JoystickTest` - Simple test of the Joystick library. It exercises many of the Joystick library's functions when pin A0 is grounded.
- `MultipleJoystickTest` - Creates 4 Joysticks using the library and exercises the first 16 buttons, the X axis, and the Y axis of each joystick when pin A0 is grounded.
- `JoystickButton` - Creates a Joystick and maps pin 9 to button 0 of the joystick, pin 10 to button 1, pin 11 to button 2, and pin 12 to button 3.
- `JoystickKeyboard` - Creates a Joystick and a Keyboard. Maps pin 9 to Joystick Button 0, pin 10 to Joystick Button 1, pin 11 to Keyboard key 1, and pin 12 to Keyboard key 2.
- `GamepadExample` - Creates a simple Gamepad with an Up, Down, Left, Right, and Fire button.
- `DrivingControllerTest` - Creates a Driving Controller and tests 4 buttons, the Steering, Brake, and Accelerator when pin A0 is grounded.
- `FlightControllerTest` - Creates a Flight Controller and tests 32 buttons, the X and Y axis, the Throttle, and the Rudder when pin A0 is grounded.
- `HatSwitchTest` - Creates a joystick with two hat switches. Grounding pins 4 - 11 cause the hat switches to change position.

Simple example

```
#include <Joystick.h>

// Create the Joystick
Joystick_ Joystick;

// Constant that maps the phyical pin to the joystick button.
const int pinToButtonMap = 9;

void setup() {
  // Initialize Button Pins
  pinMode(pinToButtonMap, INPUT_PULLUP);

  // Initialize Joystick Library
  Joystick.begin();
}

// Last state of the button
int lastButtonState = 0;

void loop() {

  // Read pin values
  int currentButtonState = !digitalRead(pinToButtonMap);
  if (currentButtonState != lastButtonState)
  {
    Joystick.setButton(0, currentButtonState);
    lastButtonState = currentButtonState;
  }

  delay(50);
}
```

Joystick Library API

The following API is available if the Joystick library in included in a sketch file.

Joystick_(...)

Constructor used to initialize and setup the Joystick. The following optional parameters are available:

- `uint8_t hidReportId` - Default: `0x03` - Indicates the joystick's HID report ID. This value must be unique if you are creating multiple instances of Joystick. Do not use `0x01` or `0x02` as they are used by the built-in Arduino Keyboard and Mouse libraries.

- `uint8_t joystickType` - Default: `JOYSTICK_TYPE_JOYSTICK` or `0x04` - Indicates the HID input device type. Supported values:
  - `JOYSTICK_TYPE_JOYSTICK` or `0x04` - Joystick
  - `JOYSTICK_TYPE_GAMEPAD` or `0x05` - Gamepad
  - `JOYSTICK_TYPE_MULTI_AXIS` or `0x08` - Multi-axis Controller
- `uint8_t buttonCount` - Default: `32` - Indicates how many buttons will be available on the joystick.
- `uint8_t hatSwitchCount` - Default: `2` - Indicates how many hat switches will be available on the joystick. Range: `0` - `2`
- `bool includeXAxis` - Default: `true` - Indicates if the X Axis is available on the joystick.
- `bool includeYAxis` - Default: `true` - Indicates if the Y Axis is available on the joystick.
- `bool includeZAxis` - Default: `true` - Indicates if the Z Axis (in some situations this is the right X Axis) is available on the joystick.
- `bool includeRxAxis` - Default: `true` - Indicates if the X Axis Rotation (in some situations this is the right Y Axis) is available on the joystick.
- `bool includeRyAxis` - Default: `true` - Indicates if the Y Axis Rotation is available on the joystick.
- `bool includeRzAxis` - Default: `true` - Indicates if the Z Axis Rotation is available on the joystick.
- `bool includeRudder` - Default: `true` - Indicates if the Rudder is available on the joystick.
- `bool includeThrottle` - Default: `true` - Indicates if the Throttle is available on the joystick.
- `bool includeAccelerator` - Default: `true` - Indicates if the Accelerator is available on the joystick.
- `bool includeBrake` - Default: `true` - Indicates if the Brake is available on the joystick.
- `bool includeSteering` - Default: `true` - Indicates if the Steering is available on the joystick.

The following constants define the default values for the constructor parameters listed above:

- `JOYSTICK_DEFAULT_REPORT_ID` is set to `0x03`
- `JOYSTICK_DEFAULT_BUTTON_COUNT` is set to `32`
- `JOYSTICK_DEFAULT_HATSWITCH_COUNT` is set to `2`

Joystick.begin(bool initAutoSendState)

Starts emulating a game controller connected to a computer. By default, all methods update the game controller state immediately. If `initAutoSendState` is set to `false`, the `Joystick.sendState` method must be called to update the game controller state.

Joystick.end()

Stops the game controller emulation to a connected computer.

Joystick.setXAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the X axis. Default: `0` to `1023`

Joystick.setXAxis(int16_t value)

Sets the X axis value. See `setXAxisRange` for the range.

Joystick.setYAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Y axis. Default: `0` to `1023`

Joystick.setYAxis(int16_t value)

Sets the Y axis value. See `setYAxisRange` for the range.

Joystick.setZAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Z axis. Default: `0` to `1023`

Joystick.setZAxis(int16_t value)

Sets the Z axis value. See `setZAxisRange` for the range.

Joystick.setRxAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the X axis rotation. Default: `0` to `1023`

Joystick.setRxAxis(int16_t value)

Sets the X axis rotation value. See `setRxAxisRange` for the range.

Joystick.setRyAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Y axis rotation. Default: `0` to `1023`

Joystick.setRyAxis(int16_t value)

Sets the Y axis rotation value. See `setRyAxisRange` for the range.

Joystick.setRzAxisRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Z axis rotation. Default: `0` to `1023`

Joystick.setRzAxis(int16_t value)

Sets the Z axis rotation value. See `setRzAxisRange` for the range.

Joystick.setRudderRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Rudder. Default: `0` to `1023`

Joystick.setRudder(int16_t value)

Sets the Rudder value. See `setRudderRange` for the range.

Joystick.setThrottleRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Throttle. Default: `0` to `1023`

Joystick.setThrottle(int16_t value)

Sets the Throttle value. See `setThrottleRange` for the range.

Joystick.setAcceleratorRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Accelerator. Default: `0` to `1023`

Joystick.setAccelerator(int16_t value)

Sets the Accelerator value. See `setAcceleratorRange` for the range.

Joystick.setBrakeRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Brake. Default: `0` to `1023`

Joystick.setBrake(int16_t value)

Sets the Brake value. See `setBrakeRange` for the range.

Joystick.setSteeringRange(int16_t minimum, int16_t maximum)

Sets the range of values that will be used for the Steering. Default: `0` to `1023`

Joystick.setSteering(int16_t value)

Sets the Steering value. See `setSteeringRange` for the range.

Joystick.setButton(uint8_t button, uint8_t value)

Sets the state (`0` or `1`) of the specified button (range: `0` - (`buttonCount - 1`)). The button is the 0-based button number (i.e. button #1 is `0`, button #2 is `1`, etc.). The value is `1` if the button is pressed and `0` if the button is released.

Joystick.pressButton(uint8_t button)

Press the indicated button (range: `0` - (`buttonCount - 1`)). The button is the 0-based button number (i.e. button #1 is `0`, button #2 is `1`, etc.).

Joystick.releaseButton(uint8_t button)

Release the indicated button (range: `0` - (`buttonCount - 1`)). The button is the 0-based button number (i.e. button #1 is `0`, button #2 is `1`, etc.).

Joystick.setHatSwitch(int8_t hatSwitch, int16_t value)

Sets the value of the specified hat switch. The hatSwitch is 0-based (i.e. hat switch #1 is `0` and hat switch #2 is `1`). The value is from 0° to 360°, but in 45° increments. Any value less than 45° will be rounded down (i.e. 44° is rounded down to 0°, 89° is rounded down to 45°, etc.). Set the value to `JOYSTICK_HATSWITCH_RELEASE` or `-1` to release the hat switch.

Joystick.sendState()

Sends the updated joystick state to the host computer. Only needs to be called if `AutoSendState` is `false` (see `Joystick.begin` for more details).

See the Wiki for more details on things like FAQ, supported boards, testing, etc.