
arduino-serial-fixture Documentation

Release stable

Jul 12, 2019

Contents:

1	Introduction	3
2	Installation	5
2.1	Prerequisites	5
2.2	Download	5
3	Usage	7
3.1	Compilation	7
3.2	Library	7
4	Contributors	9

This library provides a fixture for the [Arduino Serial](#) library to allow for offline unit tests.

Please see [ReadTheDocs](#) for the latest documentation.

CHAPTER 1

Introduction

Offline unit testing of Arduino code is challenging because of missing libraries. In this project, we aim to make code using the standard Serial interface testable using the [Catch2](#) unit testing framework.

In this section we cover retrieval of the latest release or development version of the code.

2.1 Prerequisites

This project uses the [Catch2](#) unit testing framework. On Debian based systems, this can be installed via the package manager.

```
apt install catch
```

For other distributions and operating systems, see the [Catch2 documentation](#).

2.2 Download

2.2.1 Latest release

Navigate to the [latest release](#) and either download the `.zip` or the `.tar.gz` file.

Unpack the downloaded archive.

2.2.2 From source

The source is hosted on [GitHub](#), to install the latest development version, use the following command.

```
git clone https://github.com/jfjlaros/arduino-serial-fixture.git
```


In this section we describe how to use the Arduino Serial fixture and how to use a number of additional convenience functions. We assume that the library is installed in the directory where the unit tests reside.

3.1 Compilation

First compile the fixture.

```
g++ -c arduino-serial-fixture/src/Arduino.cc
```

For all tests that require the serial interface, i.e., tests that require `Arduino.h`, make sure that the path to the fixture is set.

```
g++ -I arduino-serial-fixture/src -c test_something.cc
```

Finally, compile the test main code and link.

```
g++ -o run_tests test_lib.cc test_something.o Arduino.o
```

3.2 Library

The fixture includes most commonly used functions for serial communication. Additionally, some convenience functions are included to make testing easier. These functions can be accessed by including the header.

```
#include <Arduino.h>
```

3.2.1 Easy reading and writing

The functions `autoRead()` and `autoWrite()` can be used to read or write to the serial device. These functions take care of type encoding automatically. If, for example, we want to read an integer and a float, we use `autoRead()` as follows.

```
int i = Serial.autoRead<int>();
float f = Serial.autoRead<float>();
```

Conversely, easy writing of an integer and a float can be done with `autoWrite()` as follows.

```
autoWrite(1234);
autoWrite(3.14F);
```

3.2.2 Inspecting and preparing data

The functions `inspect()` and `prepare()` can be used to inspect the output buffer and to prepare the input buffer. The `inspect()` function works like the `autoRead()` function, except that it operates on the output buffer and does not change any of the internal buffer offsets. If for example, the output buffer contains the string `xyz`, we can use `inspect()` as follows.

```
String s = Serial.inspect<String>(); // Yields "xyz".
char c = Serial.inspect<char>(); // Yields 'x'.
```

The function `prepare()` is used to prepare the input buffer. It accepts an arbitrary amount of variables that are of either basic types (e.g., `int`, `char`, `float`, etc.) or of type `String`. If for example, we want to put a `char`, a `string` and an `integer` in the input buffer, we can use `prepare()` as follows.

```
Serial.prepare('c', "xyz", 10);
```

CHAPTER 4

Contributors

- Jeroen F.J. Laros <jlaros@fixedpoint.nl> (Original author, maintainer)

Find out who contributed:

```
git shortlog -s -e
```