Robust Arduino Serial Protocol Documentation

Release 0.2

Antonin Raffin

Apr 10, 2023

User Guide

Robust Arduino Serial is a simple and robust serial communication protocol. It was designed to make two arduinos communicate, but can also be useful when you want a computer (e.g. a Raspberry Pi) to communicate with an Arduino.

It supports both Python 3.8+.

This repository is part of the Robust Arduino Serial project, main repository: https://github.com/araffin/arduino-robust-serial

> **Warning:** Please read the Medium Article to have an overview of this protocol.

Implementations are available in various programming languages:

- Arduino
- Python
- C++
- Rust

CHAPTER 1

Installation

Using pip:

```
pip install robust_serial
```

From Source:

```
git clone https://github.com/araffin/python-arduino-serial.git
pip install -e .
```

CHAPTER 2

Available Functions

class robust_serial.robust_serial.**Order**
Pre-defined orders

robust_serial.robust_serial.**decode_order**(*f: BinaryIO*, *byte: int*, *debug: bool = False*) → None

Parameters

- **f** – file handler or serial file
- **byte** – (int8_t)
- **debug** – (bool) whether to print or not received messages

robust_serial.robust_serial.**read_i16**(*f: BinaryIO*) → robust_serial.robust_serial.Order

Parameters **f** – file handler or serial file

Returns (int16_t)

robust_serial.robust_serial.**read_i32**(*f*)

Parameters **f** – file handler or serial file

Returns (int32_t)

robust_serial.robust_serial.**read_i8**(*f: BinaryIO*) → robust_serial.robust_serial.Order

Parameters **f** – file handler or serial file

Returns (int8_t)

robust_serial.robust_serial.**read_order**(*f: BinaryIO*) → robust_serial.robust_serial.Order

Parameters **f** – file handler or serial file

Returns (Order Enum Object)

robust_serial.robust_serial.**write_i16**(*f: BinaryIO*, *value: int*) → None

Parameters

- **f** – file handler or serial file
- **value** – (int16_t)

robust_serial.robust_serial.**write_i32**(*f: BinaryIO*, *value: int*) → None

Parameters

- **f** – file handler or serial file
- **value** – (int32_t)

robust_serial.robust_serial.**write_i8**(*f: BinaryIO*, *value: int*) → None

Parameters

- **f** – file handler or serial file
- **value** – (int8_t)

robust_serial.robust_serial.**write_order**(*f: BinaryIO*, *order: robust_serial.robust_serial.Order*) → None

Parameters

- **f** – file handler or serial file
- **order** – (Order Enum Object)

CHAPTER 3

Utils

CHAPTER 4

Threads

CHAPTER 5

Examples

Examples provided here are also in the `examples/` folder of the repo.

5.1 Arduino Serial Communication

Serial communication with an Arduino: Arduino Source Code

```
from __future__ import print_function, division, absolute_import

import time

from robust_serial import write_order, Order, write_i8, write_i16, read_i8, read_order
from robust_serial.utils import open_serial_port


try:
    serial_file = open_serial_port(baudrate=115200, timeout=None)
except Exception as e:
    raise e

is_connected = False
# Initialize communication with Arduino
while not is_connected:
    print("Waiting for arduino...")
    write_order(serial_file, Order.HELLO)
    bytes_array = bytearray(serial_file.read(1))
    if not bytes_array:
        time.sleep(2)
        continue
    byte = bytes_array[0]
    if byte in [Order.HELLO.value, Order.ALREADY_CONNECTED.value]:
        is_connected = True
```

(continues on next page)

(continued from previous page)

```
print("Connected to Arduino")

motor_speed = -56

# Equivalent to write_i8(serial_file, Order.MOTOR.value)
write_order(serial_file, Order.MOTOR)
write_i8(serial_file, motor_speed)

write_order(serial_file, Order.SERVO)
write_i16(serial_file, 120)

for _ in range(10):
    order = read_order(serial_file)
    print("Ordered received: {:?}", order)
```

5.2 Reading / Writing in a file

Read write in a file (WARNING: the file will be deleted when the script exits)

```
from __future__ import print_function, division, absolute_import
import os

from robust_serial import Order, write_order, write_i8, write_i16, write_i32, read_i8,
↪ read_i16, read_i32, read_order

test_file = "test.txt"

with open(test_file, 'wb') as f:
    write_order(f, Order.HELLO)

    write_i8(f, Order.MOTOR.value)
    write_i16(f, -56)
    write_i32(f, 131072)

with open(test_file, 'rb') as f:
    # Equivalent to Order(read_i8(f))
    order = read_order(f)
    print(order)

    motor_order = read_order(f)
    print(motor_order)
    print(read_i16(f))
    print(read_i32(f))

# Delete file
os.remove(test_file)
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index