SerialTalk

antonvh

Sep 02, 2023

CONTENTS

This is a library for robust, near real-time communication between two UART devices. We developed it on python 3.9 with LEGO EV3, SPIKE Prime and other MicroPython (ESP/STM32) modules. The library is available on github: UartRemote on GitHub. The library has the following properties:

- It is fast enough to read sensor data at 30-50Hz.
- It is fully symmetrical, so master and slave can have the same import.
- It includes a RAW REPL mode to upload code to a slave module. This means you can develop code for both modules in one file.
- It is implemented in MicroPython and Arduino/C code. With Arduino code, much higher sensor reading speeds are possible, but flashing is a bit less user friendly.
- The library has a command loop to wait and listen for calls. That loop is customizable and non-blocking so you can add your own code to it.
- The python-struct-like encoding is included in the payload, so the other side always knows how to decode it.
- Compatable with most RS232-TTL 3.3v/5v converter board to further expand i/o possibilities.
- Remote module importing

Usage: you can use all of the parts of this library for your own projects. Please give us credits at least. We put a lot spare time in this. You are also welcome to contribute. Please fork and PR.

CHAPTER

ONE

GOAL

The package aims to facilitate communication between devices like Robots and peripheral embedded systems or monitors over a serial communication line. Sounds abstract? Think connecting an OpenMV camera to a LEGO SPIKE Prime Robot. Or linking up two pyboards.

CHAPTER

TWO

INSTALLATION

The easiest way to install it is with the mpy-robot-tools installer.

CHAPTER

THREE

USAGE

When you want default UART for the platform you're running on, just go: `from serialtalk.auto import SerialTalk`

When you want special channels like sockets or Bluetooth, do it like this:

```
from serialtalk import SerialTalk
from serialtalk.sockets import ClientSocketSerial

ser = SerialTalk(ClientSocketSerial("127.0.0.1",8080))
ser.call('echo','read?')
```

CHAPTER

FOUR

EXAMPLE WITH OPENMV H7

1. Copy the complete serialtalk directory to the OpenMV flash (not the whole repo, just the library)
2. Create a main.py with this code. It is an adaptation of the OpenMV Hello world

```
import sensor, image, time
from serialtalk.auto import SerialTalk

sensor.reset()                      # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)   # Set frame size to QVGA (320x240)
sensor.skip_frames(time = 2000)     # Wait for settings take effect.
clock = time.clock()                # Create a clock object to track the FPS.

st = SerialTalk()                   # Create UART comm object
def fps():                          # Create function to call from uart
    return clock.fps()
st.add_command(fps,"repr")          # Add function to callable uart commands

while(True):
    clock.tick()                    # Update the FPS clock.
    img = sensor.snapshot()         # Take a picture and return the image.
    st.process_uart()               # Process aurt calls
    print(clock.fps())              # Note: OpenMV Cam runs about half as fast when␣
↪connected
                                    # to the IDE. The FPS should increase once␣
↪disconnected.
```

3. On the SPIKE Prime Install mpy-robot-tools with the installer script. Note that the installer may seem unresponsive. Just have some patience.
4. Run this script on SPIKE Prime:

```
from projects.mpy_robot_tools.serialtalk import SerialTalk
from projects.mpy_robot_tools.mshub import MSHubSerial

st = SerialTalk(MSHubSerial('F'))

print(st.call('echo','Hello there OpenMV!'))
print(st.call('fps'))
```

This should be the result: Spike result

CHAPTER

FIVE

SERIALTALK MODULES

CHAPTER

SIX

ROADMAP, TODO

- test on esp8266 platform
- test on bt comm channels
- create pyserial/desktop channels