RTBox v4: USTC Response Time Box

http://lobes.osu.edu/rt-box.php

What is it for?

Computer keyboard and mouse can be used to record response time (RT) to an event, such as visual or auditory stimulus. However, the accuracy depends on many factors, such as computer hardware specification, operating system, programming software, user code, and so on. Even if the user code is well written, these devices are inaccurate, and even worse, often introduce bias. The major reason is that, the user program can get the time when the program reads the key or mouse response, not the time when the key or mouse is pressed.

The USTC Response Time Box (RTBox) is designed to measure response time with high accuracy. The microcontroller in the device will record the event time and button identity. The user code can read them anytime when convenient. Unlike keyboard or mouse response, the timing is independent of time your program reads the response.

Features

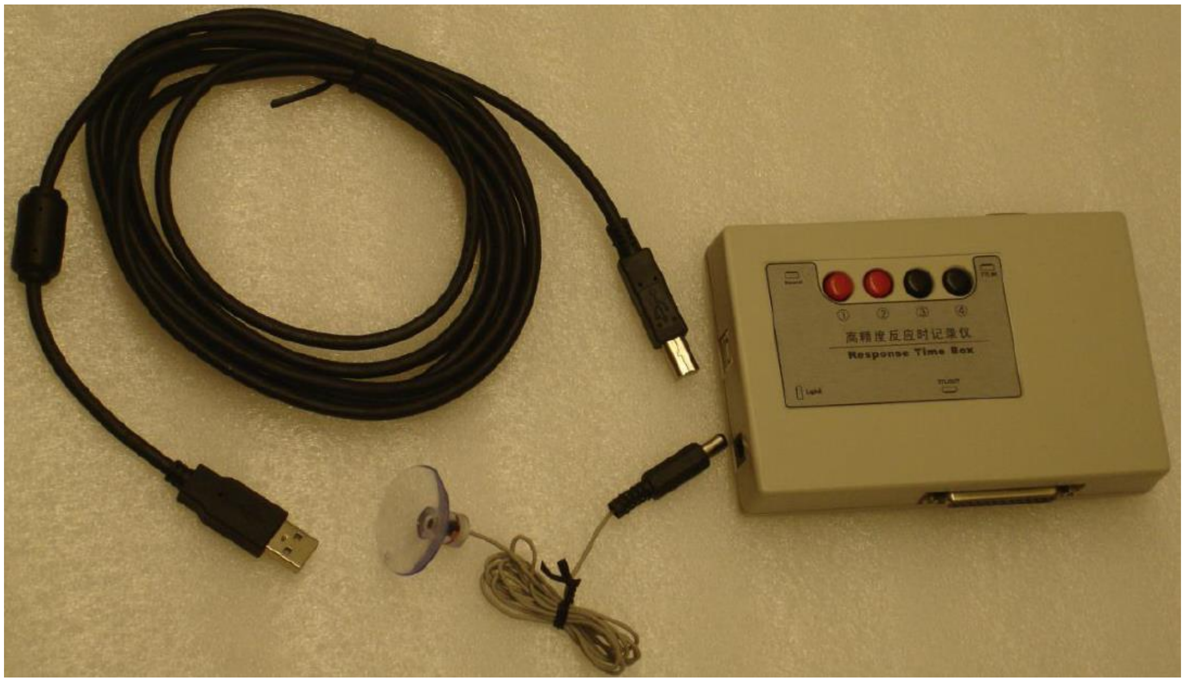
- Compatible to major computer systems, including Windows, MAC and Linux
- USB 1.1 and 2.0 compatible
- Measure both the button press and release time
- Built-in light port and pulse/sound port for trigger and calibration
- Built-in firmware upgrade so the device will never expire
- External buttons or button-driven TTL input for specialized keypad, such as MRI compatible keypad
- TTL output registering all input events and user event code, useful for EEG
- TTL input for TR (repetition time) trigger from MRI scanner
- Analog-to-Digital Converter function

Specification

- Four buttons allowing user to label with descriptive names
- Detection time resolution: about 6 μs (90 μs prior to v5.1)
- Dimensions: 5.5 x 4.5 x 1 (h) inches, 14 x 11 x 2.5 (h) cm
- Weight: ~5 oz

Parts of the device

- Photodiode with rubber suction and cable
- Four buttons
- USB port: connect to computer USB port with the provided cable
- Light port: receive light signal from the provided light sensor
- Sound/Pulse port: receive pulse signal from audio/stimulator device
- DB-9 input port: external switches or button-driven TTL, and TR
- DB-25 output port: output TTL for input events and 4-bit event code



How it works?

When the device is connected to a USB port of a computer, it will be recognized as a serial port. The device is powered by the USB port.

For principle about how the device works, you can check the paper on *Behavior Research Methods*.

Basically, the device detects button and port events with interval about 6 µs. When it detects an event, it records the event code and time based on its own clock, and sends them to the computer serial port. Each event contains 7 bytes of data. The first byte is the event code, and the rest 6 bytes contain the absolute time since the device is powered.

At the computer side, the device driver reads the data from serial buffer, identifies the event type, and calculates the response time based on device clock.

Install driver

When the device is connected to a USB port of a computer for the first time, the computer system may need the driver to recognize the device. You can download the latest version of the driver for your operating system from http://www.ftdichip.com/Drivers/VCP.htm.

For MAC OSX, the downloaded file is a dmg file. Run it and the driver will be installed. You may be asked to restart the computer.

For Linux, the driver should be included in your system. In case it is not, you can download the driver and follow the instruction in the ReadMe file. In case the Matlab or Python code fails to open the serial port, it is likely due to access issue, and you will need to do the following once:

```
  sudoedit /etc/udev/rules.d/50-myusb.rules
```

Paste the following line into the file:

```
  KERNEL=="ttyUSB[0-9]*",MODE="0666"
```

Then save the file, and re-plug the USB cable.

How to use it?

There are two ways to use the device to measure response time. If the user code has accurate stimulus onset timestamp, we need only to get the button time based on the same clock as the onset timestamp, and then do a subtraction to get the response time. This is the recommended method. This method relies on the method we developed to synchronize the device clock with computer clock.

The second way is to provide a trigger to the device to indicate the onset of stimulus. The device will detect both trigger and button events. We get the response time by calculating the time difference between the two events. This method is only needed when the user code doesn't have accurate stimulus onset time.

If there is a TTL pulse synchronized with stimulus, you can connect it to the sound port with a cable (not provided). Some stimulus equipment, such as an electrical stimulator and audio stimulator, has built-in trigger output for this purpose.

For computer-controlled visual stimulus, it may not be easy to generate an accurate trigger. The timing error could be caused by many factors, such as the time difference between code and real display, time difference due to the stimulus location on computer screen etc. We provide a photodiode as light trigger for visual experiments. You can mount the suction onto your screen, and program a light square which is within the same frame as the onset of your stimulus. If you use only grayscale visual stimulus, you can use our video switcher to provide a pulse trigger (http://lobes.usc.edu/videoswitcher).

The second method requires additional hardware connection. Although it is the most accurate solution, it is less convenient. Also the photodiode and the trigger light may be a distractor for the subjects.

Driver code in MatLab/Octave (with PsychToolbox) and Python

We provided code for both MatLab/Octave code (RTBox.m) based on PsychToolbox and Python (RTBox.py). The code can detect the device automatically, and use all its functionality. If you don't use these packages, you may "translate" the code into your own language, suppose your software environment has similar functions for serial communication and timestamp.

There are also some demo codes, showing how to use RTBox in an experiment.

How to do calibration?

The timing of device is very accurate and you normally don't need to calibrate it. However, there may be a time difference between the nominal stimulus onset and the real onset of the stimulus, and you can need to measure this difference and apply the difference to the measured response time. Ideally, this difference should be fixed for a certain setup.

The light and pulse/sound ports can be used for calibration purpose, especially for the computer-based visual stimulus. The included photo sensor is designed to detect the light onset, so we can compute the difference from nominal onset from video buffer flip time.

Both Matlab and Python code package have the demo code for the calibration.

How to test synchronization of computer and device clocks?

We synchronize the computer and device clocks by a serial trigger. When we send a trigger to the device, we record the computer time when the trigger is sent, and get the device time when the device receives the trigger. Then we get the difference between the two clocks. Later when we have device time for an event, we can calculate its computer time based on the difference. This is not guaranteed to work with all system setup. When possible inaccurate clock synchronization is detected, you will see a warning message. This typically indicates your system is not accurate on timing, not only for the response box, but also for other timing related measurement.

The speed of the computer and device clocks may be slightly different. The driver code will recommend to correct clock ratio for each host computer. The code will measure and apply the correction automatically.

How to use the ADC function of RTBox?

The RTBox can be used as a simple analog-to-digital converter (ADC). For details and instruction, check the code RTBoxADCDemo.m in Matlab.

Note that the device will stop its event report during ADC function.

Frequently asked questions and answers

1. I am warned to do RTBox('ClockRatio'). What is it for, and do I have to do it?

Answer: the short answer is no. However, clock ratio correction will further improve timing accuracy. And you'd better to run it once a while, or if you have any change to your host computer, like hardware change or major system update. Another simply way is to run the 30-sec RTBoxSyncTest, and it will inform to correct ratio if needed.

2. I am asked to change latency timer to 2ms, or warned "Failed to change latency timer due to insufficient privilege" or asked sudo password to change it. What does that mean, and how do I proceed?

Answer: It is recommended to set the USB serial port latency timer to 2ms. This won't affect the accuracy of RTBox timing, but will make serial port reading much faster. For example, on Windows and MACI systems, if the latency time is default 16 ms, 'clear' method in each trial will take about 380 ms, while with 2 ms, it takes about only 60 ms.

You can follow the instruction in the warning to change the timer, or do it by yourself. The manual method varies with operating system. Under Linux, the driver will likely adjust the timer by itself.

Under Windows, you need to go to Device Manager (right click Computer -> Manage) -> Ports (COM & LPT) -> Right click the RTBox USB serial port -> Properties -> Port Settings -> Advanced -> Change Latency Timer (msec) to 2.

Under MAC, it is a little complicated. You will need to identify the driver the serial port is using. It is typically the following file, but has some variations:
'/Library/Extensions/FTDIUSBSerialDriver.kext/Contents/Info.plist'
You'd better make a copy of the file, and then open it with sudo user, and search for '<key>FTDI2XXB'. Inside the group, you will see the latency timer key '<key>LatencyTimer</key>', and change the integer value to 2. Save the file and restart computer.

3. Does it make difference if I plug the RTBox to different USB port?

Answer: from our test, we didn't see time accuracy difference for different USB ports. However, we do see, on some computers, that reading speed is a little slower if the RTBox is connected to a port from a USB hub. As a rule of thumb, the USB ports at the back of a desktop computer may be better.

4. Why some of the TTL outputs are inverted, while others are not?

Answer: the diagram of RTBox with TTL outputs is shown in Appendix 2. By factory default, the output pins 1~8 are normal TTL (high active), while the output pins 17~24 of DB-25 port are inverted TTL (low active). This is compatible to the default polarity of Neuroscan EEG system. For other system, such as BrainProduct EEG, you can either set the polarity in the EEG software so it can receive the TTL with correct polarity, or you can change the RTBox TTL polarity to meet your system's requirement. Note that you need to change the polarity, as well as the TTL width, only once for one RTBox.

Contact us

If you have question or suggestion about the device, please contact us at:

Xiangrui Li
B71 Psychology Building
The Ohio State University
1835 Neil Ave
Columbus, OH 43210

Phone: (614) 292-1847
Email: xiangrui.li@gmail.com

Appendix 1: Serial Commands

This lists all the serial commands, and the returned data, if any, by the device. All commands are a single byte data. This is based on firmware v4.7. If you are using earlier version, please update to latest.

Uint8	Char	Returned	Function	Comments
0~15			4-bit TTL	3.0+
63	?	[63 state]	Ask button states	
66	B		Enter boot mode from simple	R to return from boot
69	E	[69 state]	Ask Enable state	1.4+
101	e	e	Wait for enable byte	4.1+
16			Send data to EEPROM	4.3+
17			Read data from EEPROM	4.3+
88	X	USTCRTBOX,921600,v5.x	Ask device identity	Also switch to advance
120	x		Switch to simple mode	1.4+
89	Y	[89 6-byte time]	Ask device time	
71	G		Go to ADC firmware	4.2+

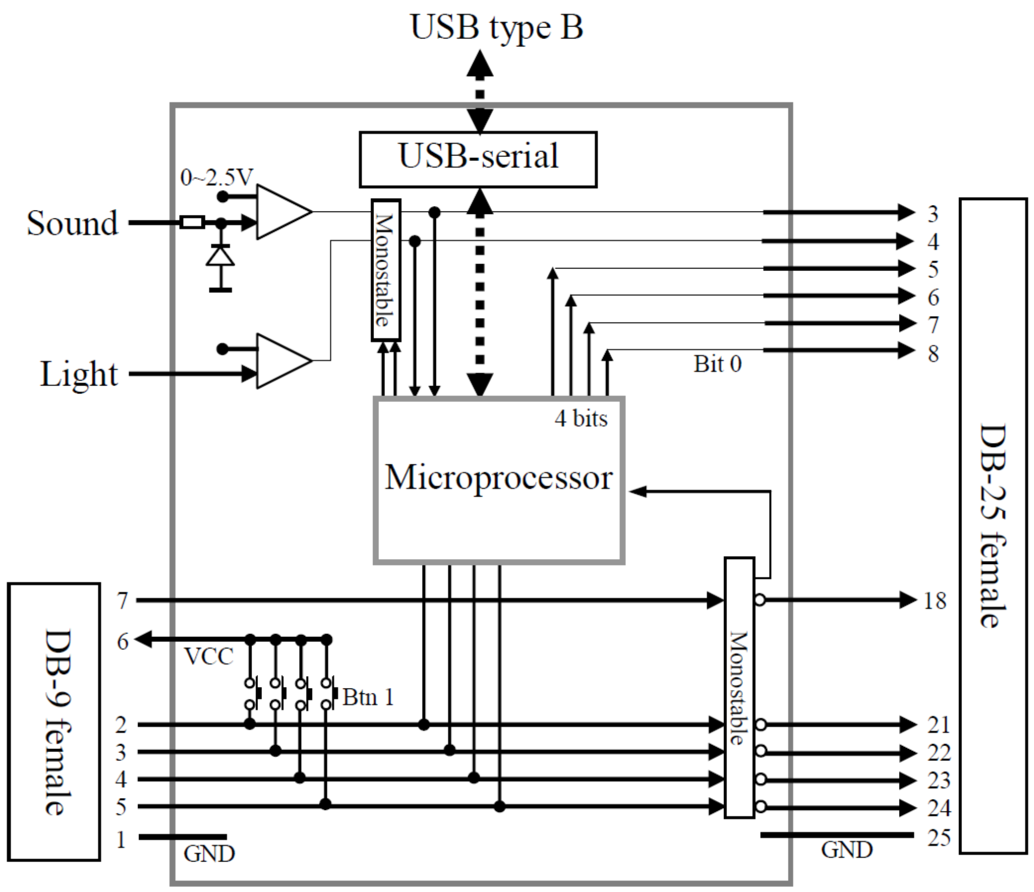
For command “?”, bits 0~3 of the returned state byte are for buttons 1 to 4.

Command “B” is used for firmware update, so you should not use it.

For command “E”, the 1~5th bits of the returned state byte are for button press, button release, pulse/sound, light and TR respectively.

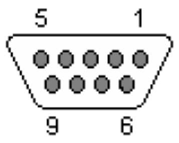
Command “X” switches device into advanced mode, and returns the device ID “USTCRTBOX”, device clock frequency 921600, and version of firmware.

Appendix 2: Schematic of RTBox v4.x



Appendix 3: Pinout of DB-9 port

Looking at the female DB-9 port on the RTBox, the pins appear as the following picture. This port is designed to receive external button-driven TTL, or to connect external buttons (switches). There is also a pin to receive MRI scanner TR trigger.



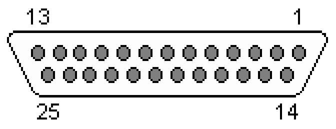
Pin	Signal	Comments
1	Ground	
2~5	Buttons 4~1	Pulled down in the box
6	5V	
7	TR trigger	Can also be used as a TTL input
8-9	No connection	

If you want to connect your button-driven TTL input to RTBox, use pin 1 as ground, and connect your signal to pins 2 to 5. Don’t use the power from pin 6.

If you want to use your external buttons, you don’t need pin 1 (ground), but only connect pin 6 (5V) to one of your button pins, and connect another button pins to one of the input pins (2~5).

Appendix 4: Pinout of DB-25 port

Looking at the female DB-25 port on the RTBox, the pins appear as the following picture. This port is designed to output TTL signal to EEG system. For now, we know it is compatible with Neuroscan, BrainProduct and Biosemi EEG system.



Pin	Signal	Comments
1~2	Ground	Required by Neuroscan System Unit
3	Sound/Pulse	There will be only one signal after it is enabled each time
4	Light	Same as Sound/Pulse
5~8	TTL Bits 3~0	TTL output (resting is low, but is configurable)
9~17	No connection	
18	TR trigger	Signal from DB-9 pin 7. Resting is high
19~20	No connection	
21~24	Buttons 4~1	Resting is high, and goes low for ~ 1ms when pressed
25	Ground	

The light, sound (pulse) and TR trigger TTL output are all direct through from input, so there is no delay in time. But the light and sound TTL will be disabled after each signal, while the TR TTL won't be disabled.