RTBox: USTC Response Time Box, v1.x

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 things, such as computer hardware specification, operating system, programming software, and user code. Even with well written user code, these devices are often inaccurate and introduce bias which is even worse. 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 button 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 time and button identity. The host computer can read them anytime when user code is convenient. Unlike keyboard or mouse response, the timing is independent of time your program reads the response.

Features

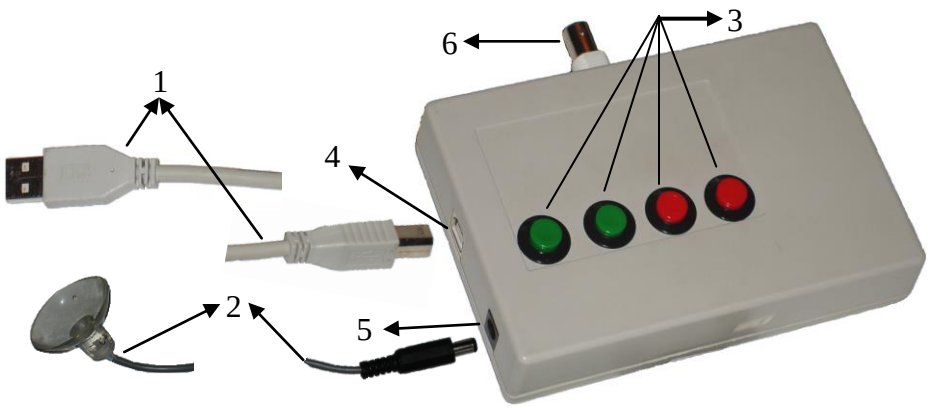
- Compatible to major computer systems, including Windows, MAC and Linux
- USB 1.1 and 2.0 compatible
- Measure both the button press and release time
- Built-in light port and pulse port for trigger and calibration
- Built-in firmware upgrade so the device will never expire (since v1.4)
- Simple ADC function to measure the light signal and more (since v1.8)
- The 10-pin port can be used as external switches (since v1.8)
- Upon request, the 10-pin port can be programmed for 4-bit TTL output

Specification

- Four buttons allowing user to label with descriptive names
- Detection time resolution: about 6 μs (90 μs prior to v1.9)
- 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

1. USB cable
2. Photodiode with rubber suction and cable
3. Four buttons: the space above is for button label
4. USB port: connect to computer USB port with the provided cable
5. Light port: receive light signal from the provided photodiode
6. Pulse port: receive pulse signal from stimulator/audio device
7. There is a 10-pin port which is close to BNC port. It was used to program the microcontroller. Since v1.8, it can be used to connect 4 external switches or switch-driven TTL. Upon request, it can also be programmed as 4-bit TTL output or other purpose.



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 our paper on *Behavior Research Methods*.

Basically, the device detects button and port events with an 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 serial port. Each event contains 7 bytes of data. The first byte is the event code, and the rest 6 bytes contain the device time of the event.

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

Install driver

When the device is connected to a USB port of the 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 the system. In case it is not, you can download the driver and follow the instruction in the ReadMe file.

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 response time with the same timestamp, and then 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 start 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 pulse 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.osu.edu/videoSwitcher/).

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

Driver code based on PsychToolbox in MatLab

We provided a MatLab/Octave code (RTBox.m) based on PsychToolbox. The code can detect the device automatically, and use all its functionality. If you don't use MatLab/Octave, you may "translate" this 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 stimulus and the trigger marking the onset of the stimulus, and you need to measure this difference so you can correct the measured response time. Ideally, this difference should be fixed for a certain setup.

The light and pulse ports can be used for calibration purpose, especially for the computer-based visual stimulus.

The demo code RTBoxdemo.m can also be used for calibration. It will show a flashing light. You can connect the photodiode to the light port so it can detect the flash. The measured response time will be the difference between Screen('flip') time and the onset of flash, taking all time delays into account. Then you can simply correct response time by the measured difference. Note that you need to enable the detection of 'light', since it is disabled by default.

How to test synchronization of computer and device clocks?

We synchronize the clocks by a serial trigger. When we send a trigger to the device, we record the computer time when the trigger is sent, and we also get the 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 convert it into 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 RTBox('ClockRatio') command will measure it and apply correction automatically. You'd better to run it when suggested by the driver code, although this is not mandatory.

How to use the ADC function of RTBox?

Since v1.8, the ADC function is implemented into the firmware. For v1.x hardware, the light signal is connected to one of ADC channels. So you can check the light signal as

shown in the RTBoxADCDemo.m. However, for historical reason, there is an RC circuit for the light signal, so the shape is smoothed and is not real light signal. If you are comfortable to do soldering, you can remove the C1 capacitor to get rid of the smoothing.

If you like to measure other signal, you will need to do some minor hardware modification. This is because the ADC input pins are not directly accessible, but are in the unused pin header on the PCB. You can open the box, and connect one or more ADC channels. If you need more information about ADC channels in the header, contact us.

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 time accuracy.

   If you are using version 1.1, you may need to run as super user so the code can save the ratio into a file. Otherwise, you will be asked to correct the ratio each time the device is closed.

2. I am 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: the warning indicates you are running as a restricted user. The code tries to change the latency timer of the USB serial port to the shortest 1 ms, so the reading of device will be faster. For example, on Windows and MACI systems, if the latency time is default 16 ms, RTBox('clear') will take about 380 ms, while with 1 ms, it takes about only 60 ms. The failure to change the value affects only the speed of reading. As the warning message suggests, you can run the code as administrator once (or simply input sudo password for MAC), and it will change the latency time. After the change, you can run as a restricted user. On Windows system, if you plug the box into a different USB port, you might need to make the change again for that port. You may need to right-click the Matlab icon or exe file, and *Run as administrator* so Matlab can change the LatencyTimer. You need to do this only once. On Linux, it seems that you have to run as super user anyway to access the USB-serial port. On MAC, the change will take effect after you reboot the computer, and the change is permanent for all USB ports.

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.

Contact us

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

Dr. 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 is a list of all the serial commands, and the returned data, if any, by the device. All commands are a single byte data. In Matlab, you need uint8 to convert a number before sending it via IOPort.

uint8	Char	Returned	function	Comments
63	?	[63 state]	Ask button states	Return [state 63] for 1.4-
65/97	A/a	A/a	Enable/Disable All	
66	B		Enter boot mode from simple	R to return from boot
68/100	D/d	D/d	Enable/Disable button-Down	
69	E	[69 state]	Ask Enable state	1.4+
79/111	O/o	O/o	Enable/Disable light-On	
80/112	P/p	P/p	Enable/Disable Pulse	
83	S	S	Waiting for 8-byte data	
115	s	8-byte data	Get 8-byte data	
64	T	14-byte time	Test firmware speed	For developers only
85/117	U/u	U/u	Enable/Disable button-Up	
88	X	USTCRTBOX,921600,v1.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	

For command "?", the 5th to 8th bits 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 1st to 4th bits of the returned state byte are for button press, button release, pulse, and light, respectively.

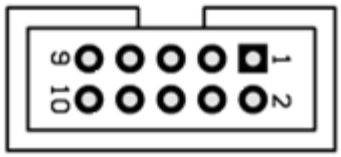
After command "S", the device will wait for 8-byte data for 1 second. The 8-byte data are some important parameters. The first byte is the TTL width in unit of 0.14 ms. The second byte is the debounce interval in unit of 1.1 ms. The third byte is the resting level of TTL. The fourth byte is 0 or 115 ("s"), meaning the clock ratio is not saved or saved. The rest 4 bytes are the clock ratio related data.

Command "T" is used by firmware developers to test the round trip speed, and is not available for normal device.

Command "X" switches device into advanced mode, and returns the device ID "USTCRTBOX", device clock frequency 921600 or 115200, and version of firmware.

Appendix 2: Pinout of external 10-pin port

The 10-pin external port looks like the right figure. This port was originally designed for developers to upload firmware to the microcontroller, and was not suggested to use by users.



Since v1.8, we developed a way to use the 4 data pins in this port for flexible purpose. In v1.8 firmware, we use them as external switches or external button driven TTL. The pinout of the port is as following:

Pin	Signal	Comments
1	Button 1	JTAG TCK, PC2
2	Ground	
3	Button 3	JTAG TDO, PC4
4	3.3V	Use with care
5	Button 2	JTAG TMS, PC3
6	RST	Don't connect anything to it
7	3.3V	Use with care
8	No connection	
9	Button 4	JTAG TDI, PC5
10	Ground	

If you want to connect your external switches to RTBox, use pin 2 or 10 as common ground, and connect 4 switches to pins 1, 5, 3 and 9. When a pin is connected to ground, the RTBox will treat is as a button press.

If you want to connect button-driven TTL input to RTBox, use pin 2 or 10 as ground, and connect your signal to pins 1, 5, 3 and 9 for the 4 buttons. The TTL must be low active.

The above 4 pins can also be configured to export 4-bit TTL. Please contact us if you like to use them for TTL output or other purpose.

Warning: Please don't connect pin 6 (RST) to anything. Also be careful when you use the 3.3V power (pin 4 and 7). If it is shorted, you can destroy the RTBox or your computer's USB port.