Daily Struggle Button

Poll Flow Chart

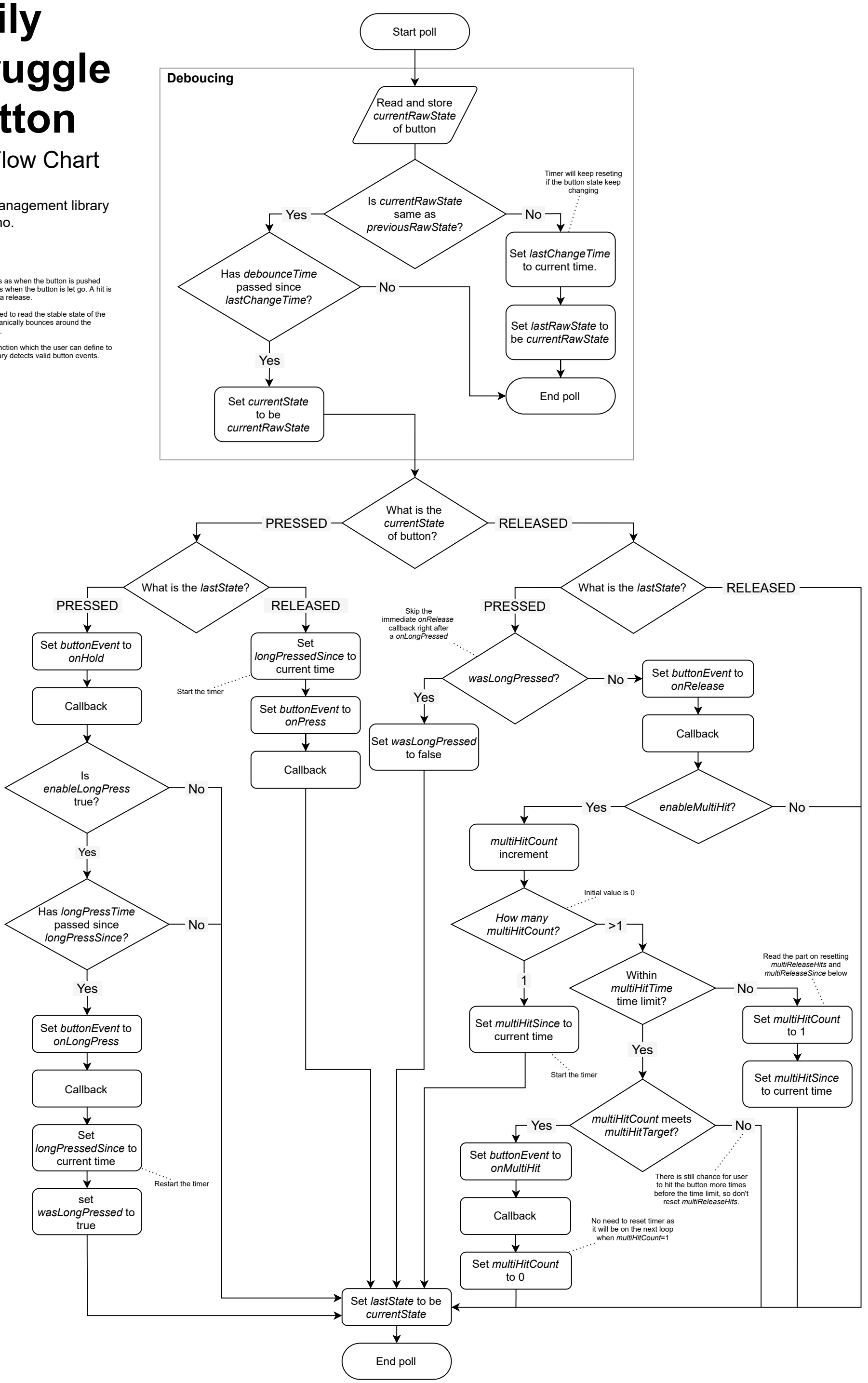
Button management library for Arduino.

Note:

We define a press as when the button is pushed down. A release is when the button is let go. A hit is a press and then a release.

Debouncing is used to read the stable state of the button as it mechanically bounces around the contacts when hit.

A callback is a function which the user can define to run when the library detects valid button events.



DailyStruggleButton
by cygig 2021
github.com/cygig/DailyStruggleButton

Variables

Button States

PRESSED

Button is pressed down.

RELEASED

Button is released; no longer held down.

debounceTime

The current stable state of the button after debouncing.

LastChangeTime

Period of time to wait when a button state change is registered before reading again to confirm the change.

currentRawState

The most recent unstable reading of the button state.

lastRawState

The previous unstable reading of the button state.

currentState

The most recent stable reading of the button state.

lastState

The previous stable reading of the button state.

Events

buttonEvent

Indicate to callback function what was the event that happened using numbers.

onPress

The button is pushed.

onRelease

The button is let go.

onHold

The button is held down. Be careful as this will keep calling back multiple times until the button is let go.

onLongPress

The button was held down for *longPressTime* length of time.

onMultiHit

The button was pushed and let go for *onMultiHitTarget* number of times within *onMultiHit* milliseconds.

Long Press

enableLongPress

Enable or disable detection of long pressing of button.

longPressTime

The time it takes to hold down a button to be considered a long press.

longPressedSince

The time it takes to hold down a button to be considered a long press.

wasLongPressed

Flag to indicate not to trigger onRelease after a onLongPress.

Multi-Hit

enableMultiHit

Enable or disable detection of multi-hit sequence.

multiHitCount

The number of hits the user had achieved in a multi-hit sequence.

multiHitTarget

The number of button hit a user needs to achieve in *multiHitTime* to be considered a multi-hit sequence.

multiHitTime

The timespan where *multiHitTarget* number of presses and releases needs to be achieved to be considered a multi-hit sequence.

multiHitSince

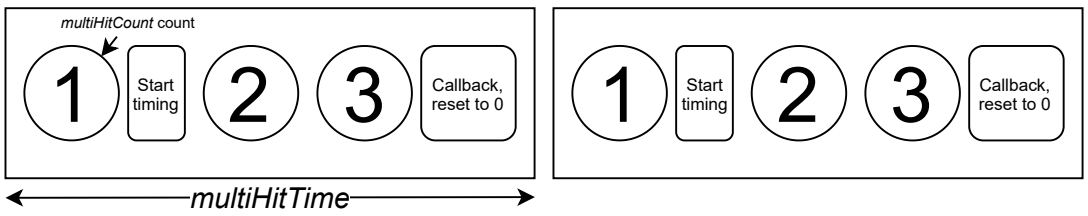
The time when the multi-hit sequence started.

When to reset *multiReleaseHits* and *multiReleaseSince?*

Assuming *multiHitTime* is 1000 and *multiHitTarget* is 3, which means the user needs to hit the button 3 times in a time limit of 1000ms for *onMultiHit* to register. When do we reset *multiHitCount* and *multiHitSince*?

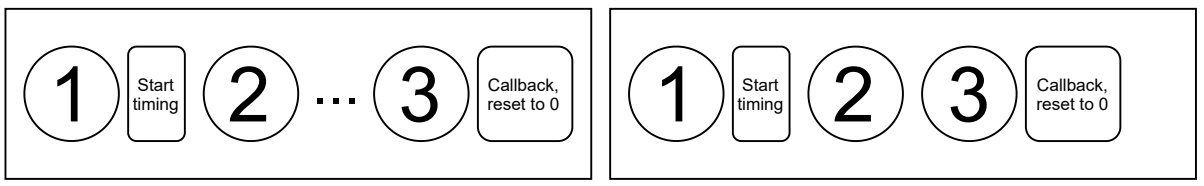
Scenario 1: Within time limit, all 3 hits connect

This is the perfect scenario where button is hit 3 times within the time limit. *multiHitCount* resets to 0 after the third hit.



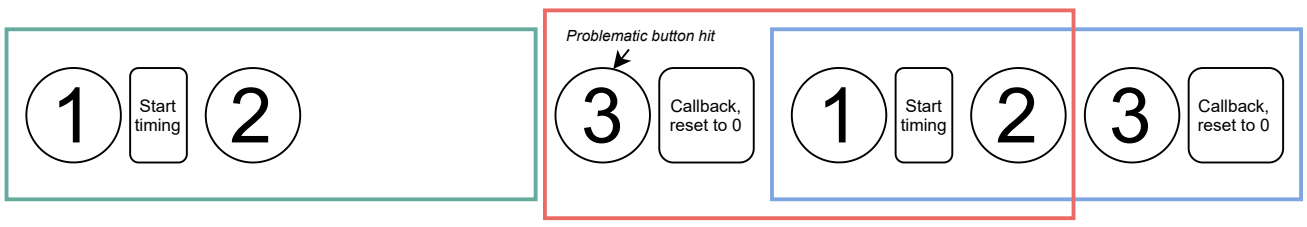
Scenario 2: Within time limit, not all hits connect

This happens when the last one/few is/are pressed slower, but still within the time limit, essentially similar to Scenario 1.
If those buttons were pressed out of the time limit, this becomes Scenario 3.

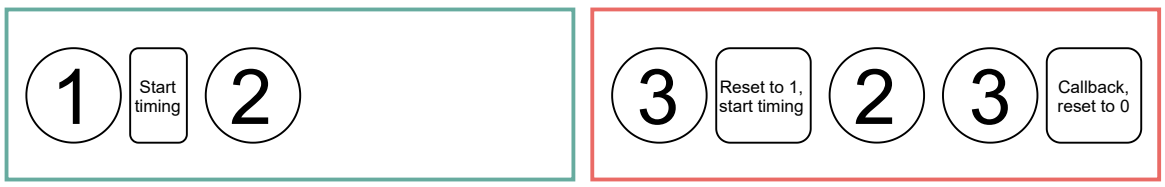


Scenario 3: Out of time limit, all three hits connect

The most interesting of all. The problematic button hit is the third late one. It is the closing hit for the green sequence where it is determined not to invoke *onMultiHit* since it is late. It resets *multiHitCount* to 0 and the next button hit forms a new blue sequence. However. the user might want to quickly try again after a failed multi-hit sequence attempt like in the red sequence. In this case, the problematic hit becomes an extra button to push.

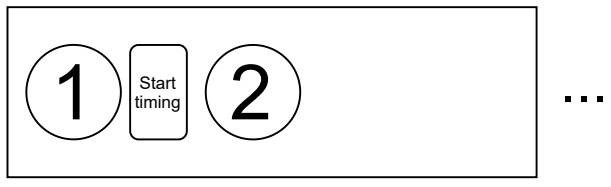


To fix this, after detecting that the time limit is over, we set *multiHitCount* to 1 instead. Then we set *multiReleaseSince* to the current time. This effectively ends the green sequence and immediately start the new red sequence.



Scenario 4: Out of time limit, not all hits connect

As long as the user keep hitting buttons, the multi-hit sequence will always connect, just whether within the time limit or not. In this scenario, it simply means that the user stopped hitting buttons altogether. As there is no event for the button being constantly released, we can ignore this scenario. If the user hits the button again, this will be back to Scenario 3.



DailyStruggleButton
by cygig 2021
github.com/cygig/DailyStruggleButton