I. Switch Debouncing

A Switch is Pressed, So What???

Problem: Switch Bounce

Initial Connection

Finally Contact Closed

Typically 10-20ms
Debouncing

- When a switch (any type) changes state (on -> off or off -> on), it presents a mechanical bouncing which generates a signal similar to the one shown at the right.
- The resistor R is needed because the signal S can not be left “floating” in an undefined state when the switch changes from state 1 to 2.
- Without debouncing the signal can generate several interrupts (or status changes) corresponding to just one action.
- Debouncing consists in “Filtering” the signal S so that a proper operation of the switch action is sensed.
- Debouncing can be done in hardware or software.

Techniques that can be used:
- If status loop: after first status change, program timer and after elapsed time read key status.
- If Interrupt: on first interrupt program timer which will interrupt after elapsed time. Then read key status.

```
Debouncing

1. Setup a counter variable, initialize to zero.
2. Setup a regular sampling event, perhaps using a timer. Use a period of about 1ms.
3. On a sample event:
4.   if switch signal is high then
5.     Reset the counter variable to zero
6.     Set internal switch state to released
7.   else
8.     Increment the counter variable to a maximum of 10
9.   end if
10.  if counter=10 then
11.    Set internal switch state to pressed
12.  end if

tkeydown event occurs on the state transition of released to pressed

void setup() {

    // configure pin2 as an input and enable the internal pull-up resistor
    pinMode(2, INPUT_PULLUP);
}
```

CSEP567
Debouncing

- A Better Way:
  - Use the Bounce library, check examples
- Remember that debouncing is not the same as one event per button press:

  keyDown event occurs on the state transition of released to pressed

Debouncing State Machine

keyDown event occurs on the state transition of MaybePush to Pushed