ELEC 327 – Ball V2– *Smart* Height Measuring Device

Yiqiu Wang, Tingkai Liu, Yuqiang Heng

Central Concepts

- **MOTION !**
- Distance = 0.5 * g * time^2 (g = 981 cm s^-2)
- The device generally have inelastic collision with the ceiling, hence we neglect the initial acceleration (beyond g) when the device starts to free fall

Functionality

In a room, user can repetitively throw this device to hit the ceiling and let the device free fall to the floor, and the device will calculate the distance of free fall and take the average.

Push Button Inputs

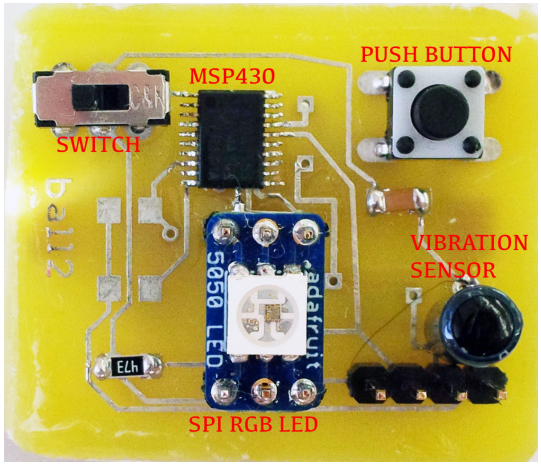
1. Press button for 0.4 seconds or longer but shorter than 1.5 seconds – the device will calculate the current distance measured and average into the previous measurements. Then it immediately gives the output.
2. Press button for 1.5 seconds or longer – the device will discard the current measurement and get ready for the next measurement.
3. Press button twice quickly – the device will reset itself and get ready for a new set of measurements.

Led Outputs

1. The device gives output in decimal number with unit cm:
    a. Red LED: 100 * number of blinks
    b. Green LED: 10 * number of blinks
    c. Purple LED: 1 * number of blinks

    Eg. 3 Red blinks + 2 Green blinks * 1 Purple blink = 3*100 + 2*10 +1 = 321cm
2. Discard – Red Red short blink
3. Reset – Red Green Blue short blink

PCB Design



- **Compact design**

We would not want a huge PCB, which is clumsy and also unsafe when tossed with higher distance.

- **Fall proof**

A larger PCB is also more prone to damage given because it has higher inertia. Use of simple components also gives a higher reliability.

Software Design

The code focuses on the following things:

- **Extending the functionality and control based on a hardware design with few components.** We could not use an LCD display because tossing would easily damage it, hence we used SPI LED and used three different colors to represent the 100, 10 and 1 digits of a number. This is accurate and interesting. We also extended the functionality of the button by setting different patterns of pressing, instead of multiple buttons.
- **Increasing accuracy given the unpredictability of tossing.** Multiple tosses can have slight variations, taking average increases accuracy to great extends. Sometimes a toss can be bad, and it would be time-consuming to start over the entire measurement, hence user are allowed to discard a toss and continue the measurement.
- **Low power.** The entire design utilized multiple timers to achieve low power. The CPU would be off during most of the time, and most of the jobs are done in interrupts.

Code Structure

- *Settings*
    - *WDT settings*
    - *Timer settings*
    - *SPI settings*
    - *Port 2 Settings*
    - *Enter Low Power Mode and Enable Interrupts*
- *Port 2 Interrupts*
    - *Vibration Sensor*
        - *Timer1*
        - *Timer2*
    - *Push Button*
        - *Average*
        - *Discard*
        - *Reset*
- *WDT Interrupts*
    - *Increment timers*

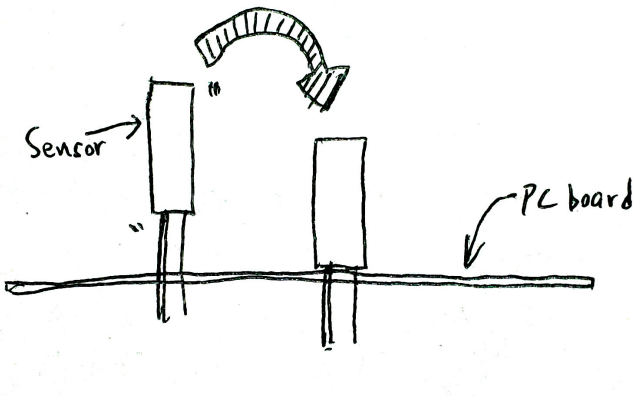
Challenges

1. A normal toss of the device might trigger the vibration switch **2-5 times, it's important for the software to recognize the correct time interval**. There are roughly **four cases** of different triggering situations:
    a. Case 1: Triggered when hit ceiling, hit floor.
    b. Case 2: Triggered when tossed, hit ceiling, hit floor.

c. Case 3: Triggered when tossed, hit ceiling, hit floor, bounced on floor for 1 or 2 times.
d. Case 4: Triggered when hit ceiling, hit floor, bounced on floor for 1 or 2 times.

Based on the complicated situations, one thing we can first make sure is that in one round of measurement, any vibration switch trigger after the third one is redundant. In the first two time intervals in between three triggers, the correct time interval between ceiling and floor triggering must be the longer one. This is because it takes shorter to arrive at the ceiling due to initial acceleration and shorter distance; and the bounce most likely has a much shorter distance. Based on these facts, it helps to distinguish the correct interval.

2. Vibration Switch debouncing. The general idea of the debouncing of this sensor is similar to the push button, but from later observations: the vibration switch "vibrates" for a long time after it is impacted, because the connection of the sensor and the PCB acts like a pivot for the sensor to vibrate around. We solved this problem in both hardware and software.
   a. Hardware: We inserted the sensor further into the PCB to reduce unwanted vibrations.



   b. Software: an additional debouncing time is added to the vibration sensor.

3. Time measurement calibration. We realize that the WDT is not very accurate when measuring time interval, which is essential in our measurement. Therefore, we took a long time calibrating the WDT by tracking our counter in the debug of CCS using stopwatch. In the end, we created a time calibration variable *timeCalibration* to adjust the accuracy in different circumstances.

Further Improvements

1. Though we have a focus on low power, we still used some __delay_cycles() function mainly in LED display interval and vibration switch debouncing. This can be improved by using WDT timer for LED display intervals and moving vibration switch to ports other than port2, and use another timer to do the debouncing, which we could not do now because vibration switch requires a different debouncing scheme to that of the button, which we did not think of when making our first PCB.

2. Despite the fact that the current device gives pretty accurate measurements, it would be better to use accelerometers to measure initial accelerations and integrate over time to measure the most accurate distance. But given the limited calculation capability of the MSP430 and few choices of sensors, we eventually decided on a simpler but much more reliable, low-power and compact design like this.