INTRODUCTION

A transducer translates intensity of sunlight into current, which is sampled and accumulated. An indicator clearly changes state when ( ) of sunlight falls on the object. We suggest the use of a tiny Solar Panel that keeps charging a battery for most time. At constant intervals (about 4 times a second), the current output of the panel is dropped against a resistor and the voltage is measured. Solar Panels have perfectly linear current vs light intensity graphs [4]. The voltage measured is accumulated by a cheap microcontroller. The controller also changes the state of an LCD once enough sun light is accumulated, indicating that the water safe to drink. The solution proposed costs $x at the maximum, and is almost perpetual, except for device wear and tear. Its water proof, needs no maintenance and is 45mm x 25mm x 7mm (or smaller)

In this document, we have argued the reason behind using a solar panel, the microcontroller and LCD to display, and included their costs, a description of the complete setup, and experimental results. In addition a section also describes how the idea was reached exploring other options and further enhancements.

IMPLEMENTATION

A solar panel charges a button cell battery. It also doubles up as a transducer, translating the intensity of sunlight into voltage across a resistor. This is read by a microcontroller and accumulated. Once the required amount of sunlight is gathered, the microcontroller drives and LCD display that masks out the default “Don't Drink” picture (or bright red color) indicating its safe to drink the water.

The microcontroller (operating at 4MHz) contains an analog to digital convertor that converts the voltage level into a convenient 8bit number. A value zero corresponds to 0V and 255 to 3.6v (battery voltage). To save power we do not keep using the ADC always. Instead, once ever nominal time (limited by how quickly sunlight conditions can change) we acquire a sample. In our code we have sampled the value once every 250ms, that's 4 samples a second. The ADC takes 13 cycles (325 nano seconds) to convert. When the ADC is not being used (for the next 250 milliseconds), a General Purpose Input Output (GPIO) to which the other end of the resistor is connected to is tristated leaving the pin floating. There is no current flowing through resistor R1, all current produced by the panel is used to charge the battery. During this state, the GPIO controlling the LCD goes on and off every 125milli seconds, blinking the LCD. Ie., the display blinks between don't drink and black 4 times a second indicating that the device is active and the required sunlight is not accumulated as yet.

Over time the sampled values are accumulated. This resultant value gives us amount of sunlight collected since the device was last reset. When a pre-calculated value is reached, the device goes into a state where the controller only drives the output LCD continuously

until reset. Note that the LCD is a single pixel display device that is transparent when off. When switched on, it turns black, hence covering the background image.

The proposed microcontroller (10f222), is a simple, inexpensive ($0.39/device) low power version manufactured by Microchip. With numerous industrial applications, it is commonly available, easy to program and well characterized. Using a microcontroller is significantly cost effective as sourcing even a few passive components can bring the cost up to 50 us cents. The increase in flexibility and reliability is an added bonus.

For resetting the device using a micro-switch was our first choice. Unfortunately, using one is not practical, as this would add an additional ~$0.50 and would have to have a exposed mechanical part that would degrade with rain. Using a small circuit with the microcontroller, a simple touch of 2 wires can be easily detected. 2 strands of open wire (copper/any conductor that does not rust) is left open on either side of the device. When the device is held as shown in fig, the device resets.

Schematics

At the heart of proposed solution lies a PIC10f222 device. This is 6 pin 8 bit microcontroller that is connected as follows.

Pin Name	Connected to
Vcc/GND	Power
AN0	ADC input pin, measures output of Solar Panel
GPIO1	Switches between Tristate (default) and Ground (when ADC conversion is done)
GPIO2	LCD driver
GPIO3	Touch sensor input

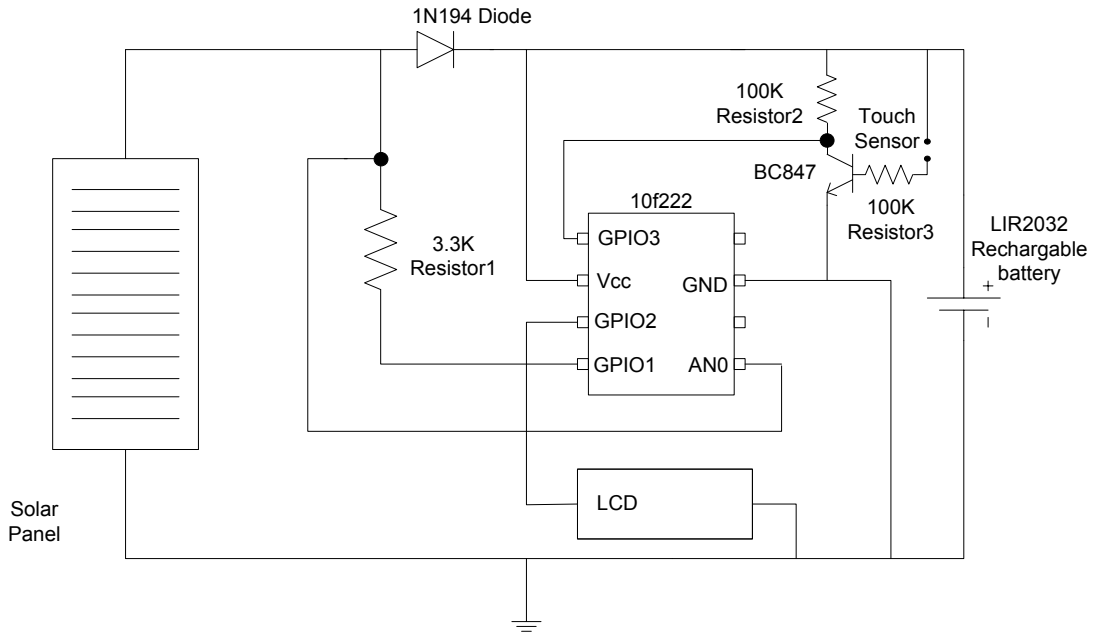

Fig 1: Circuit Diagram

The diode d1 allows flow of current in one direction only. This prevents the battery from draining through the solar panel when the panel voltage is less than that of the battery.

The touch sensor circuit uses a BC 847 transistor and 2 100k resistors, shorting the points shown by touching them causes GPIO3 to go to '0'. This value is picked up by the PIC. A BC847 is used instead of the usual BC547 for its higher β, that improves sensitivity and reduces power consumption (15nA drain when inactive). [5]

Characteristics of Components used

(1) Solar Panel

The current output of the solar panel depends on the intensity of light falling on it and the load connected to it. At a constant load, the current output varies is directly proportional to the light intensity (energy).

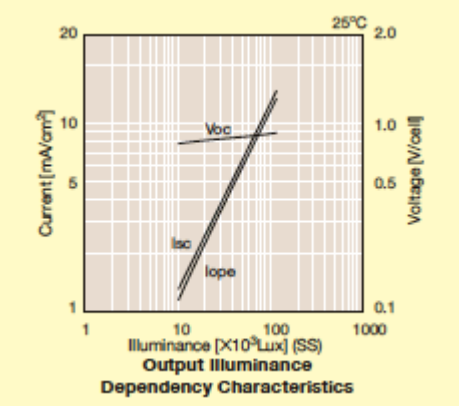

Fig 2. Sun Intensity vs Current output [4]

100 lux	Very dark overcast day[2]
320–500 lux	Office lighting[8][9][10]
400 lux	Sunrise or sunset on a clear day.
1,000 lux	Overcast day[2]; typical TV studio lighting
10,000–25,000 lux	Full daylight (not direct sun)[2]
32,000–130,000 lux	Direct sunlight

Fig 3. Typical Lux values (http://en.wikipedia.org/wiki/Lux)

This was corroborated by experimenting on the solar panel, measuring intensity vs current output over a 3.3KΩ resistor.

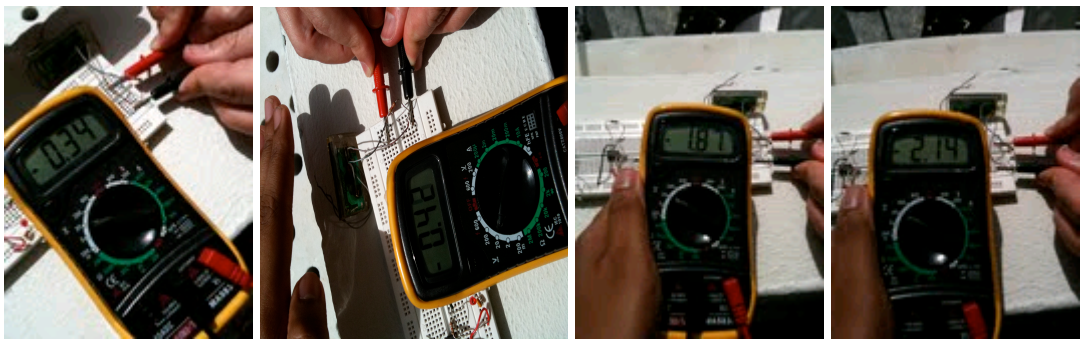

Fig 4.Voltage generated by a solar panel across a 3K resistor when (a) dark (solar panel in shade) (b) slightly exposed (c) cloudy (d) bright sunlight at 4PM

(2) Microcontroller
Detailed characteristics of the 10f2xx series is found in their datasheets [6]

(3) LCD
Power consumed in off state (display state "Don't Drink"): 0
Power consumed in on state (display black) : ~1µA

(4) Rechargeable battery
3.7V, 35mAh. The device's life is most likely limited by the battery charge discharge cycle. The datasheet [7] claims 500 cycles at 35mAh. That's 70hrs*500 = 35000 hours (4 years) of continuous usage.

(5) Passive components:
Diode: Voltage drop 0.6V. Resistor: 3.3K, Transistor BC857 [5].

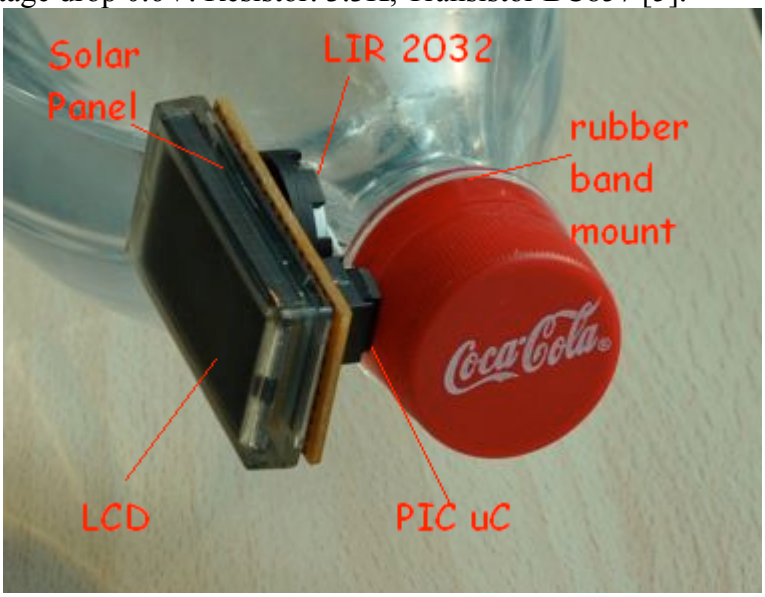

Fig 5. Components Used
Prototype only, entire actual device should fit in the epoxy material engulfing the LCD and the Panel in this picture

Power budgets
At V = 3.6Volts,

Current from the Solar Panel (at 2500kJ/m2) = 3mA. [a]

Current consumed by the microcontroller (active) = 400uA [a] (200uA [b])
Current consumed by the microcontroller (sleep) = 40uA [a] (30uA [b])
Current consumed by LCD (always on) = 0.5uA - 1uA [a]
Current consumed by LCD (blinking, during accumulation) = 50% of 1uA = 0.5uA.

Assuming LCD is on all the time and its very cloudy for over 2 days, we see that the device can sustain for about 50 hrs or 2 days for every hour of good sunlight.

Once the battery is fully charged (35mAh), the device can run without sunlight (keeps blinking) for 70hrs.

[a] Values obtained in our experiments.
[b] Claimed according to PIC10F2xx series electrical specs. Results may vary due to temperature, voltage etc.,

Packaging
Embedding the complete device in epoxy is suggested, an idea inherited from the blinking LCD-solar powered keychains [8]. this makes it water proof, resistant to normal wear and tear and severe heat. in my initial prototype I used the keychain's solar panel and LCD, I had to drill through the epoxy to get to the contacts.

Operation
Step 1: Fill bottle as described by SODIS. The device is in default state and will display "Don't drink". [Fig 8]
Step 2: Touch either ends of the device as shown (device resets)

Fig 6 : Resetting the Device (by touching the contacts on either side)

Step 3: When in operation the device keeps blinking, wait for it to go completely black.
Step 4: When LCD stops blinking and the Don't Drink sign is not visible any more, the water is safe to drink. [fig 9]

Software

The following flowchart illustrates the software implemented on the PIC.

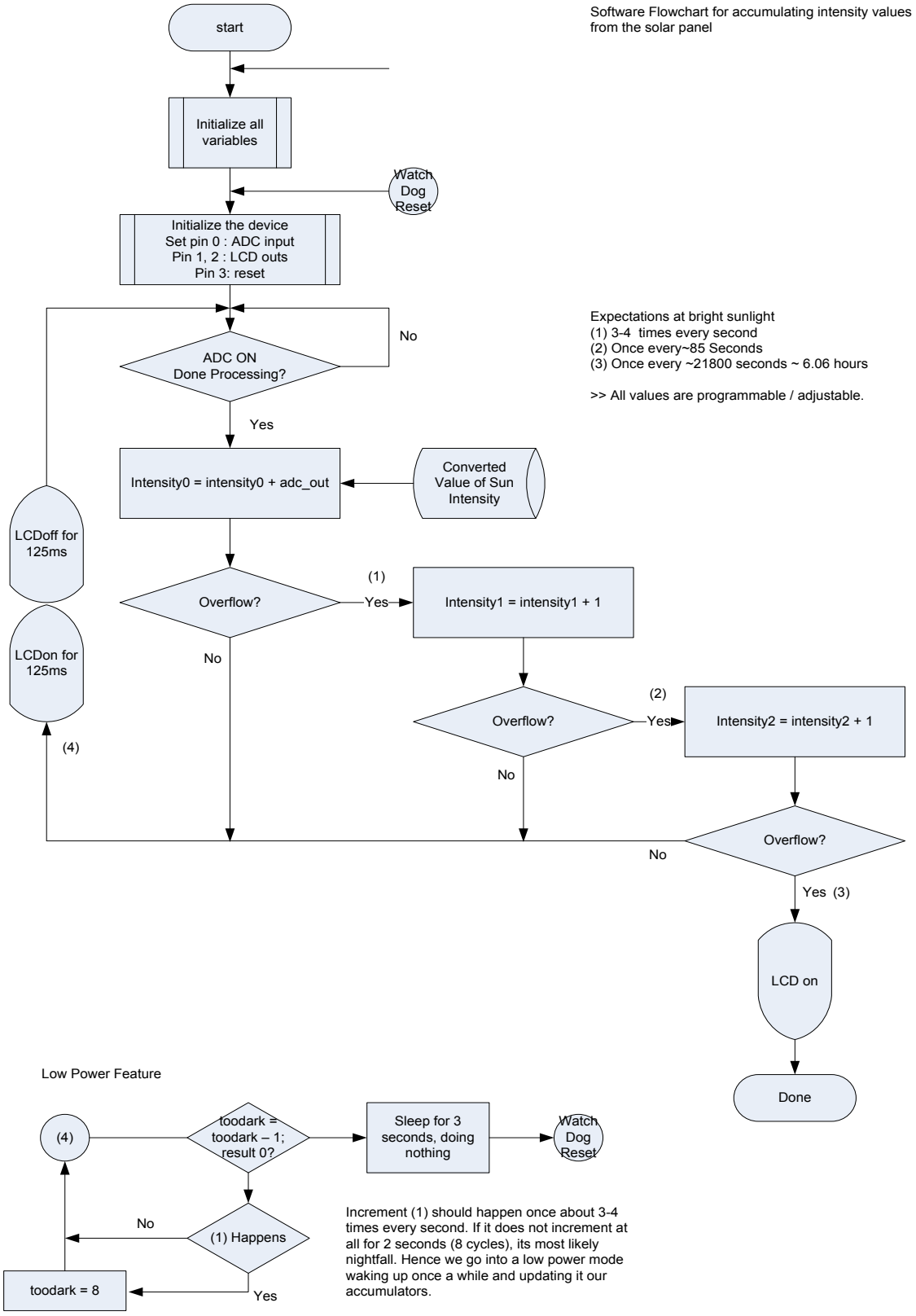

Fig 4: Software flowchart (please see attached jpg for better resolution)

Mounting on the bottle

Depending on the location of the panel and the LCD on the final manufactured product, the device can be mounted such that the LCD is visible clearly and the solar panel faces the sun. In our proposed configuration, its best mounted on the cap (so that it does not obstruct sunlight to the bottle) as shown in the figure. Here a rubber band is used to mount the device on to the cap.

Fig 7 Mounting the device on a PET bottle.

A single device can be used with several bottles if they are all on the same location and are filled in at the same time.

COSTS

Our estimate for the final product (all inclusive) is at most $2. We base this on products that include similar components sold widely [8].

Component	Cost	Cost source
PIC10F222	$0.39	Microchip store [2]
Transistor BC847 + Resistors + Diode	$0.10 (total maximum)	www.digikey.com
Solar Panel	$1 (estimate)	Devices with these components sold for $0.35-$0.5 by Chinese manufacturers. [8] [9]
LCD		
Manufactuing/Packaging (Epoxy)		

One time costs:
Programmer: as PICs are widely used, both official and custom-made programmers are widely available. They cost between $50-500 in production, microchip has a facility to deliver the devices pre programmed.
Experiments for calibration: the 6 hour value used is based on hand calculations, an on the field calibration will definitely yield more accurate results.
Engineering development costs.

Prototyping cost
The Programmer: I used a k182 USB programmer [3], that cost me 82 Singapore dollars. A 10F222 (cost of 1 piece purchase) costs about S$0.76 from the local farnell store (sg.farnell.com), the other components cost S$0.60, and the solar panel and LCD were ripped from a keychain purchased for S$4.

EXPERIMENTS

Two prototype boards were built. The first one used normal components as proof of concept and is used in the following pictures

Fig 8: Device in Default state (when the device is not active) The NISSAN logo is replaced with full red to indicate “water not safe”.

When the device is operating, the display blinks at the rate of 4 times per second (adjustable) switching between fig 8 and fig 9.

The keychain came with the NISSAN logo stuck to the LCD, I could not remove it without damaging the LCD. Manufactures give a choice of backgrounds (one even has replaceable backgrounds.

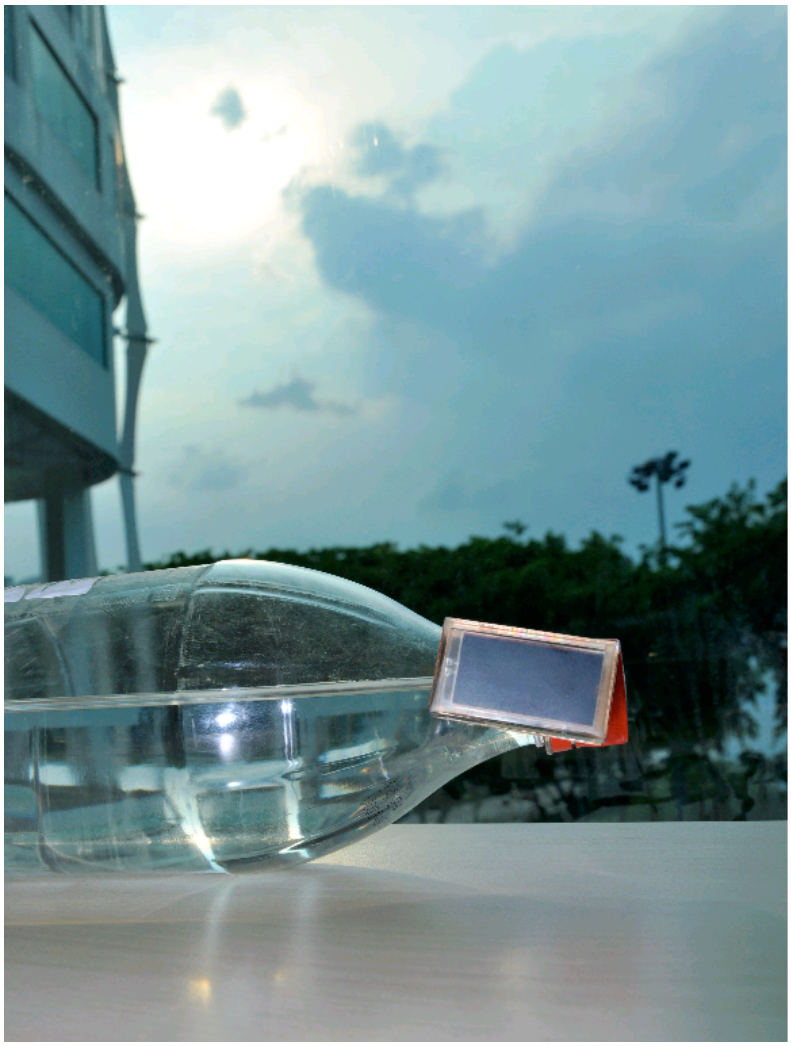

Fig 9: Display when enough light has been accumulated.

The images where shot at 4PM on a not so sunny day in Singapore. The device took 8 hours to reach the done state. The calibration was done the earlier day under bright sunlight.

In addition to this, SMD (Surface Mount Devices) which are much smaller were assembled to check if the volume fits in the epoxy into a thin device.

w

Fig 10 Assembly using SMD components embedded in epoxy. Notice the touch leads protruding outside.

ADVANTAGES AND ENHANCEMENTS

Software Usage

The solution is very flexible, thanks to the use of software to actually measure, decide and indicate. Amongst many other things the following can be controlled

(1) Intensity measurements

(i) Frequency of sampling: Depending on how fast the sun intensity changes, the number of samples / minute can be varied to further optimize

(ii) Sampled value scaling: Following experiments, the actual effectiveness of various intensities of sunlight can be factored in. ie., if

the assumption that 4 hrs of sunlight at 10W/m2 = 1 hr of sunlight at 40W/m2 is found incorrect, it can be accounted for in software. Furthermore negative effects of keeping the bottles overnight can be accounted for by subtracting a part the accumulated value of intensity. For example nonlinearities due to weather and climate as tabulated in Fig 10 on Page 14, of the SODIS manual [] can be factored in.

(iii) The type of PET bottle used affects SODIS effectiveness (Page 17 of []). A small piece of code can be written that initially measures direct sunlight and sunlight falling behind a filled bottle. This way the amount of sunlight blocked by the bottle (due to turbidity, use of bottles with low UV-transmittance, colored PETs) can be measured. This can then be used to scale up the exposure requirement automatically.

(2) Indication options With the LCDs/any other outputs used fully controllable, a more effective way of expression if found can be easily implemented.

(3) Device self check : With the other ADC hooked the battery a small piece of bootup code can check the condition of the battery

LCD vs LED

Our initial idea for state out was using low power LEDs which we had to discard for following reasons.

(i) a commercially available cheap LED consumes atleast 1mA, about 1000 times that consumed by an LCD

(ii) In bright sunlight, even a powerful red LED is barely visible

(iii) In the default state (LCD off), the indication given is that the water is not safe. The device stays in this state even before starting up. Powering up the LCD (blinking during operation and always on when done) is better a logic (LCD is on only when needed) than indicating using an LED that needs to be powered up to express any state.

Additional hardware ideas

A temperature sensor if integrated can enhance the system as SODIS method is also temperature dependent. However we could not think of a noninstrusive way to mount a sensor to the bottle. [1]

As explained earlier, an additional panel placed under the bottle can generate bottle/water quality information that can then be used by the processor.

Using Advanced PIC Controllers

Initially the circuit was built with a more advanced 12F615 microcontroller. To save power significantly, a crystal oscillator was used at 32KHz. The device now operates 125 times slower than the 10F222 at 4MHz, hence consumes much lower power. Further 12F615 supports sleep states and wake ups based on timers, when the power consumption is a 100 times smaller. (Current consumption was ~10μA in operation and ~1μA in sleep, the device can be on sleep 99% of the time). Effectively, such a device would last about 50 times longer than the proposed solution. However the cost was almost double (a 12F615 costs $0.59, with addition components required such as capacitors and a crystal

oscillator). If this is acceptable, a solution very similar to the one proposed but using the features of a 12F615 is highly recommended. This device can even double up as a powerful white LED light at night (powered by the battery as the device itself consumes negligible current).

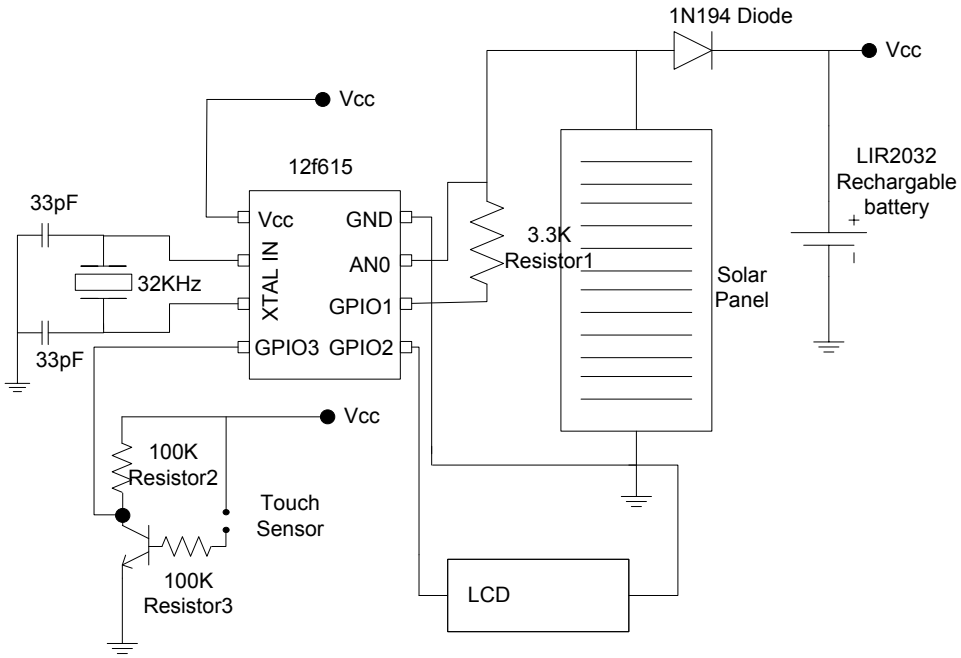

Fig 11: Solution with 12F615, costlier but more effective.

CONCLUSIONS

A device that uses a solar panel as a transducer to measure solar intensity is proposed. A controller adds intelligence to the device, making it flexible. A single pixel LCD display serves as the output, consuming very little current, yet clearly visible in broad daylight. Further, enhancements are proposed. A prototype of the device was assembled and tested and the results matched as required by the challenge. The author would be glad to send the prototype for further analysis on request.

REFERENCES

[1] SODIS manual
[2] PIC 10F series pricing
[3] PIC Programmer (USB, 3rd Party)
[4] Sanyo Characteristic plots of Solar Cells
[5] Transistor BC847 datasheet
[6] PIC10f2xx series datasheet
[7] LIR 2032 battery datasheet
[8] Solar LCD keychain (Alibaba link of a manufacturer)
[9] List of Solar LCD manufacturers (Alibaba link)

The code used in the prototype is attached. This code handles the touch sensor, sampling and accumulation, done signaling and LCD driving. Further the code puts the controller in sleep for most time during night times.

```
        list    p=10F222        ; tell MPLAB what processor is used
        #include <p10F222.inc>     ; has defines specific to 10f222

        __CONFIG   _MCLRE_OFF & _CP_OFF & _WDT_ON & _MCPU_OFF & _IOFSCS_4MHZ
        ; Disable everything except Watchdog timer

        cblock 0x10
                count1                          ; used in delay routine
                counta                          ; used in delay routine
                countb                          ; used in delay routine
                intensity0                      ; used in accumulation
                intensity1                      ; used in accumulation
                intensity2                      ; used in accumulation
                toodark                         ; used for power down
        endc

        org 0xff                                ; processor resets here
        ; Internal RC calibration value is placed at location 0xFF by Microchip
        ; as a movlw k, where the k is a literal value.

        org 0x00                                ; and immediately steps here
start
        movwf OSCCAL                            ; Factory calibrated value goes in to OSCCAL.
        btfsc STATUS, 4
        goto WDTTimeOut                         ; If only a timeout occured, no need to clear
                                                ; Timer, our base values etc.,
TouchReset
        movwf 0x0
        movlw TMR0
        movlw intensity0
        movlw intensity1
        movlw intensity2
        movlw toodark
WDTTimeOut
        clrwdt                                  ; A few setups on reset : clear registers etc.,
        movlw 0xcf                              ; no pull ups, no wake ups, timers set inside
        option                                  ; Also, set WatchDog timer to reset every 2-3
seconds.

        movlw 0x41                              ; ADC control :
        movwf ADCON0                            ; Channel 0 On. Dont switch on yet

loop
        clrwdt                                  ; Clear Watchdog timer
        movlw 0x9                               ; GP1 to output : LCD Drive
        movwf GPIO
        TRIS GPIO                               ; Set GP2 to output, and ground it
                                                ; Now the solar panel's current drops as voltage
```

```
across
                                        ; the 3K resistor
        btfss GPIO, 3                   ; Check GPIO3
        goto TouchReset                 ; reset if if value = 0 (touch active)

        bsf ADCON0, 1                   ; GO! for the ADC
ADCOperating
        btfsc ADCON0, 1                 ; Wait till Conversion done
        goto ADCOperating
                                        ; Conversion Done


        movlw 0xd                       ; 1101 : GP1 to output : LCD Drive
        movwf GPIO                      ; Also sets GP2 to input => Tristate
        TRIS GPIO                       ; This means the resistor is floating, solar panel
charges the battery.


                                        ; Update Timer start
                                        ; +1 to value
        movf ADRES                      ; Put ADC result in W
        addwf intensity0, 1             ; Add that with previous value
        btfss STATUS, C                 ; Check if there is carry over
        goto DoneAccumulation
        movlw 0x8
        movwf toodark
        movlw 0x1
        addwf intensity1, 1             ; if intensity0 overflows, then intensity1 ++;
        btfss STATUS, C                 ; Check carry
        goto DoneAccumulation
        addwf intensity2, 1             ; if intensity1 overflows, then intensity2 ++;
        btfsc STATUS, C                 ; Wait for it to overflow, skip if NOT
        goto WaterClear                 ; if this also overflows => We are done!
DoneAccumulation
        call LCDOnDelay                 ; Do nothing for another 1/8th second. Switch On
LCD
        call LCDOffDelay                ; Do nothing for another 1/8th second.

        movlw 0x1
        decfsz toodark, 1
        goto loop                       ; normally, keep looping.

        movwf toodark                   ; reset it to 1, so that next time it checks, it goes back
to sleep
                                        ; immediately if there is no Sun.
        sleep                   ; if toodark is zero => the intensity0 did not overflow for
                                ; 8 cycles or 2 seconds => light intensity < 1/8th of 256
                                ; for 2 seconds => mostly night fall... go to sleep and get up
                                ; at watch dog time out! Low power mode

WaterClear
loop1
        call LCDOnDelay                 ; switch on the LCD, and forget about it
        clrwdt                          ; no more watch dog needed
        goto loop1
        retlw 0x00

LCDOn
```

```
        bsf GPIO, 1
        goto back

LCDOnDelay
        movlw   d'250'                  ;delay 250 ms (4 MHz clock)
        movwf   count1
d0      movlw   0xC7
        movwf   counta
        movlw   0x01
        movwf   countb
        btfss GPIO, 1                   ; Turning on the LCD is tricky :
        goto LCDOn                      ; It requires on and off toggling.
        bcf GPIO, 1
back
Delay_0
        decfsz  counta, f
        goto    $+2
        decfsz  countb, f
        goto    Delay_0

        decfsz  count1  ,f
        goto    d0
        retlw   0x00

LCDOffDelay
        bcf GPIO, 1                     ; Turn off LCD
        movlw   d'250'                  ;delay 250 ms (4 MHz clock)
        movwf   count1
d1      movlw   0xC7
        movwf   counta
        movlw   0x01
        movwf   countb
Delay_1
        decfsz  counta, f
        goto    $+2
        decfsz  countb, f
        goto    Delay_1

        decfsz  count1  ,f
        goto    d1
        retlw   0x00

        end
```