Hardware Details Tilt Switch Indicator

Date: 8.12.13
Author: Lorenz Gruber
Code for MCU in apendix

Functionality

Movement detection

To detect the “indication” gesture of the cyclist, a tilt switch tracks the tilt of the user’s arm. The specifications of the tilt switch guarantee an “on” position if the hand is higher than 15 degrees from ground (Figure 1). The same holds true for an “off” position and less than -15 degrees (Figure 1). In the range between +15 and -15 degrees, the position could be either.

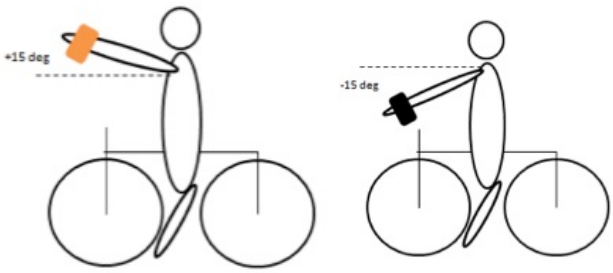

Figure 1 On and off position

The state of the tilt switch is sampled every 50ms by the microchip. To avoid false detection, 7 consecutive samples of the tilt switch need to be “on” to start the LEDs. Once an indication movement is detected, the LEDs will blink for four seconds. During that period the position of the tilt switch is neglected.

Lighting

Four high quality LEDs are used to achieve visibility in each direction. The LEDs are connected via two power wires and then pulled into the slap band. They achieve maximum efficiency of 70 lumen per watt at a forward voltage of 2V. At 2V, the combined consumption is around 120mA. The blinking frequency is 1Hz with a duty cycle of 50%.

User interaction

By turning the on/off switch on, the microchip and the movement detection start. Detecting the indication movement is the only mode that is available to the user.

Power supply

The battery is a 3V Li-Ion cell, which has 160mAh capacity and is not rechargeable. If the LEDs were lit constantly, the battery lasts for 1.3h, neglecting the consumption of the microcontroller and the transistor.

Circuit and PCB design

Circuit diagram

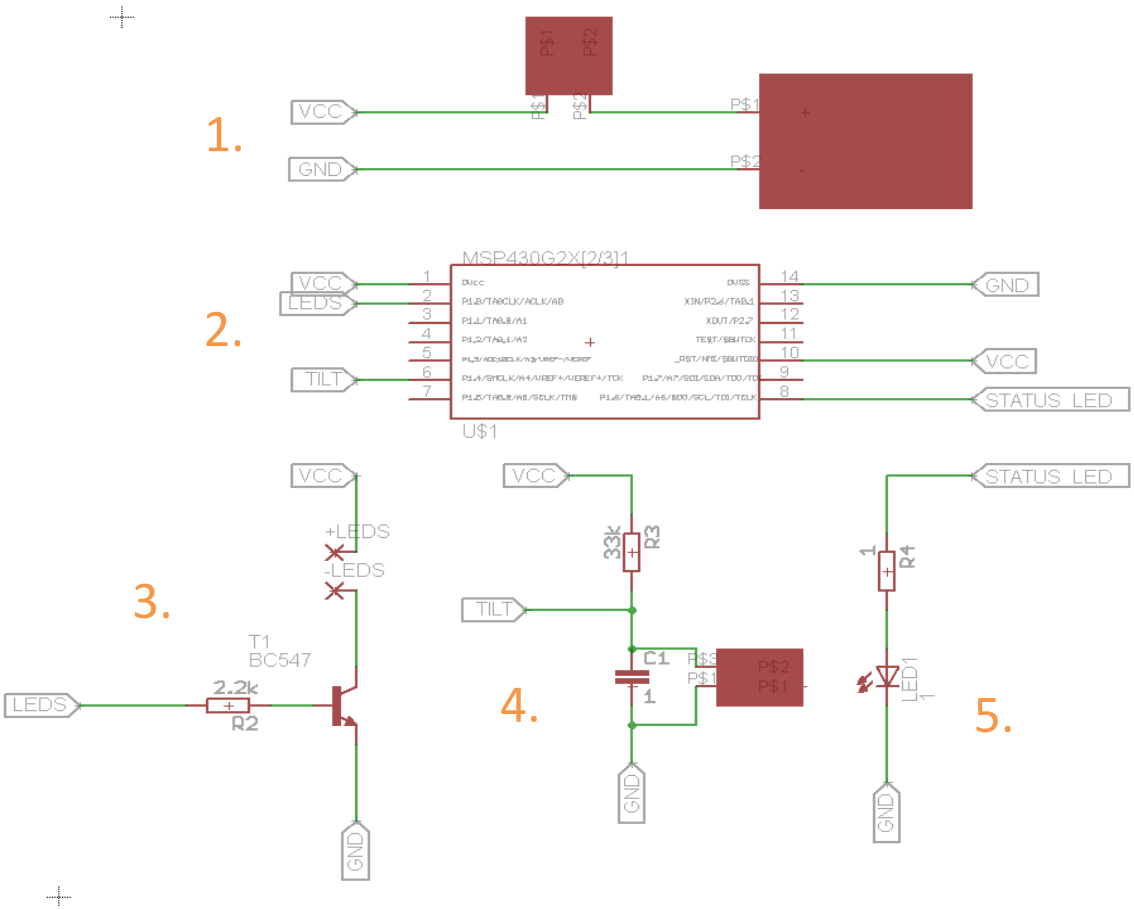

Figure 2 Circuit diagram v.1

The three components with red background had to be made as no existing components were found. The circuit is divided into 5 parts (indicated by orange numbers) that are connected with labels.

1. Battery connecter (large red rectangle) and SDPT switch
2. MSP430 chip
3. LED control
4. Tilt switch detection
5. Status LED control

The driving circuit for the 4 main LEDs (pictured bottom left of figure 2) is not using a current limiting resistor. Reason for that is the battery voltage which drops sharply from 3V to around 2.2V when is supplying around 120mA to the 4 LEDs. The LEDs have their highest optical efficiency when they are driven with 2V. As the transistor has a drop of around 0.2V, there is no need for a current limiting resistor.

PCB – Design

The slap band is 30mm wide. This width was used as a reference for the PCB. The dimensions of the PCB below are 65x30mm.

The chip is mounted from the top. The top area which the chip is covering should have been restricted. With a DIP-14 socket for the chip, soldering the connections at the top was hard. Also, the top restrictions for the SPDT switch and tilt switch are are unnecessary.

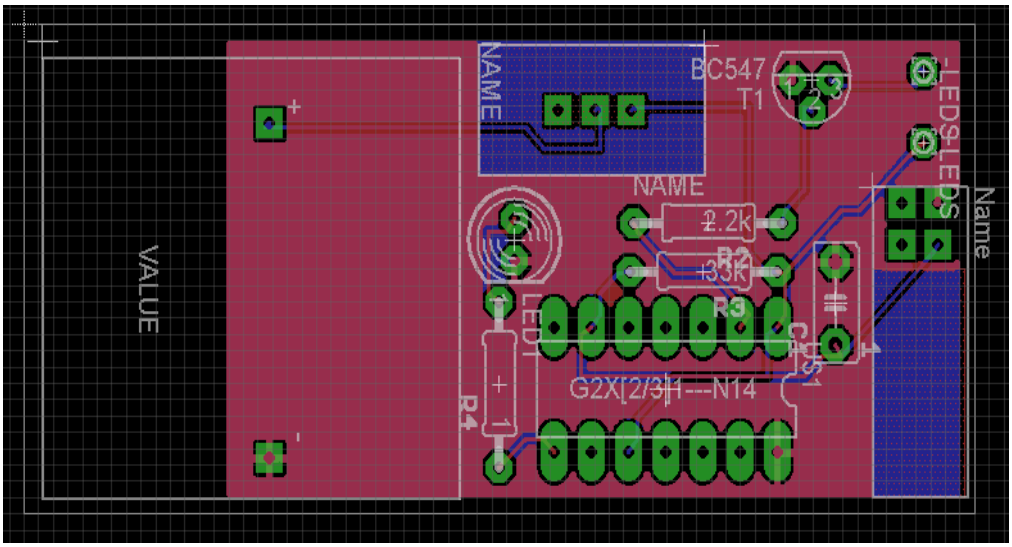

Figure 3 PCB layout

Finished Prototype

For the first prototype a batch size of 10 was produced.

The manual assembly of the prototype pictured in Figure 6 took around 3 hours. To protect the PCB, a customized box was 3d printed. Yet, the PCB is too big. When a user wears the slap band, pictured in figure 7, the box that is glued to the slap band becomes loose.

The following pictures show as sketch with dimension of the slap band, the assembled prototype, and a zoom of the PCB. The band has to be worn, such that the arrow on the PCB points away from the user.

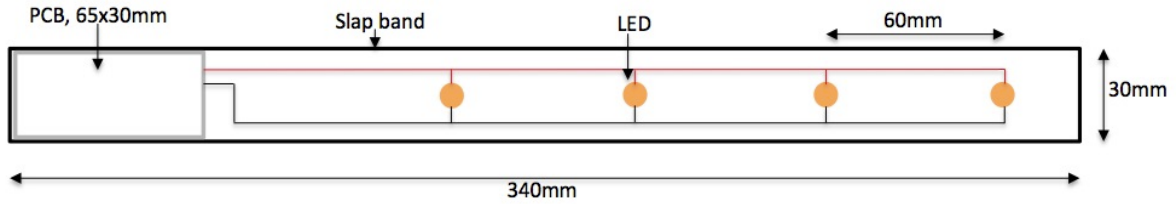

Figure 4 Sketch of v.1 with dimension

Figure 5 Finished prototype

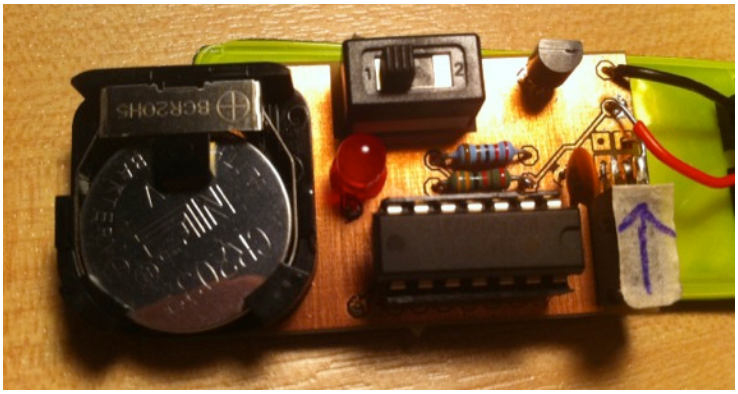

Figure 6 Assembled PCB

Figure 7 v.1 blinking

Microcontroller

Choice

The microcontroller used was the MSP430G2231 from Texas Instruments. It was chosen as a TI Launch Pad development board was available. The MSP430G2231 has 2kB of flash memory and 128B RAM.

For testing, the Launch Pad was used. The reset pin of the chip needs to be connected to VCC, if the chip is used without the launch pad.

Software

The code was written in C using the Code Composer Studio v.4 from Texas Instruments. The full code can be found in the appendix.

Three #define at beginning of the code are used to change the parameters for the low pass filter and the blinking duration / frequency quickly.

The compiled code uses 238 bytes of flash memory (~ 10%) and 2 bytes of RAM.

Component / Price List

The following table shows a cost breakdown of prototype v.1. The order date for the components from RS was 13.11.13. The total component cost is £6.48 including VAT. The cost for the PCB and several other components are neglected as the EEE department provided them.

Item	RS Stock No.	Quantity	Price (excl. VAT)	Notes
PCB	-	1	£0.00	Printed at Imperial College
Battery Holder BR/CR Verticle Holder	745-0938	1	£0.83	
90 deg pin tilt switch,24Vac 25mA	361-5093	1	£0.46	
SIDE LED Orange,LO A67F-V2BB-24	361-5093	4	£0.70	
CR2032 Lithium Coin Cell 3V	597-201	1	£1.12	
MSP430G2231IN14	724-4654	1	£1.17	
5mm status LED	-	1	£0.00	EEE Level 5 lab
BC547C NPN Transistor	-	1	£0.00	EEE Level 5 lab
Current Limiting Resitors	-	2	£0.00	EEE Level 5 lab
SPDT switch	-	1	£0.00	EEE Level 5 lab
Slap band	-	1	£0.90	From E-Bay
Total excl. VAT			**£5.18**	
Total incl. VAT			**£6.48**	

Appendix

Code for MSP430G2231 for v.1

```
// #include "msp430" --> change to the line below:
#include "msp430g2231.h"

#define LED1 BIT0
#define STATUS_LED BIT6
#define TILT BIT4

#define DELAY 10 // blinking frequency
// delay of 20 is too slow
// delay of 10 is a good speed (1Hz)
// blinking frequency = 1 / (DELAY * 50ms * 2)

#define BLINK_TIME 4 // how often will it blink
// total blink time (s) = DELAY * 50ms * 2 * BLINK_TIME
// for v.1: total blink time = 10 * 50ms * 2 * 4 = 4 seconds

#define SENSITIVITY 7   // how often one must detect the tilt switch in
correct position before turning it on.
                                    // 1 -> 50ms
                                    // 20 -> 1s is too long
                                    // 3 --> 150ms too short
                                    // 7 --> 350ms seems quite ok. I tested
a bit walking round in the lab.


// function definitions
void blink_leds();
void turn_on_led();
void turn_off_led();
void turn_status_led_off();
void turn_status_led_on();

int main(void)
{
      WDTCTL = WDTPW + WDTHOLD; // Stop WDT --> otherwise it will reset
    BCSCTL1 = CALBC1_1MHZ; // --> set the timers to 1mhz
    DCOCTL = CALDCO_1MHZ; // --> set the timers to 1mhz
    P1DIR |= (LED1 + STATUS_LED); // Set LED1 and status led to output
direction
    P1OUT &= ~(LED1 + STATUS_LED); // Set LED1 and status led to 0 (LED
OFF)


      turn_status_led_on();
    while(1)
    {
            int i = 0;
            int success = 0;

            for (i = 0; i <= SENSITIVITY; i++){
                  if ((TILT & P1IN)){
                  success++;
                  _delay_cycles(50000); // 50000 clock cycles is 50ms with
1mhz speed
                  }
```

```
                  else {
                        break;
                        //i = SENSITIVITY + 1; // exit the loop
                  }
            }
      if (success == SENSITIVITY + 1){ // it needs to be sens + 1 as the
for loop above is <= sens
                  blink_leds();
            }
    }
}

void blink_leds(){
      int i = 0;
      int j = 0;
      for (j=0; j <= (BLINK_TIME*2-1); j++) // need the -1 that the leds
stay off afterwards
      {
            P1OUT ^= (LED1); // toggle led
            for (i=0; i <= DELAY; i++)
                  {
                  _delay_cycles(50000);
                  }
      }
      // make sure both LEDs are off in the end
      turn_off_led();
}

void turn_on_led(){
      P1OUT |= LED1; // the pin is now high // need the logical or to set
it to 1 for sure
}

void turn_off_led(){
      P1OUT &= ~LED1; // need the logical and to set it to zero for sure
}

void turn_status_led_off(){
      P1OUT &= ~STATUS_LED;
}

void turn_status_led_on(){
      P1OUT |= STATUS_LED;
}
```