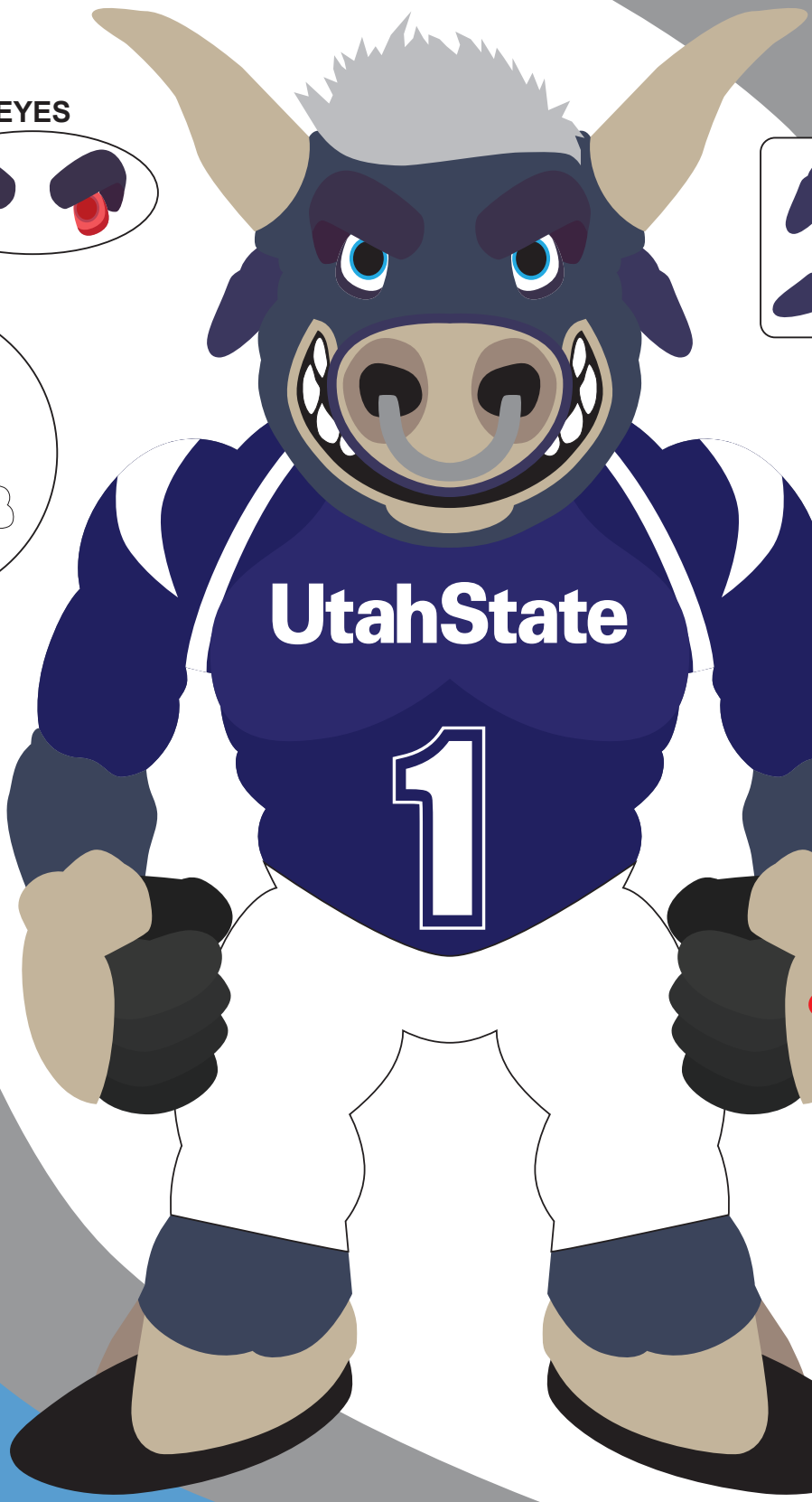
PROJECT

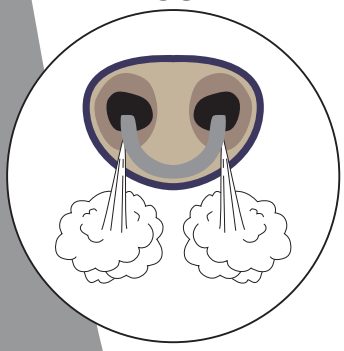
BIG BLUE 2.0

Nathan Yost & Joel Meine

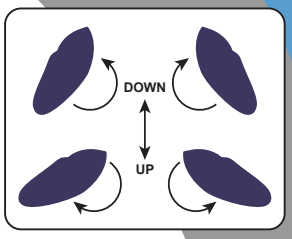


EYES

NOSE



EARS



MOUTH

BUTTON

TABLE OF CONTENTS

1. Introduction

This document describes the design of a system of embedded electronic special effects developed in a working prototype. Collectively these special effects are intended to be placed into a new costume for the USU bull mascot known as Big Blue. The effects will be a unique enhancement to the mascot to attract increased audience attention and excitement to athletic events. Combined, the electronic effects embedded into the head piece will further animate the bull mascot to express an angry mode that is invoked at user command when a bull is about to charge a target. Practical application of this technology would yield the attraction of other cultural industries including theme parks, theaters, film production, and sporting events. Although generally unique to the character of a bull, the special effects utilized in this project could also readily translate over to a dragon costume.

2. Overview

This document discusses, in general, the electrical and software design for the special effects system prototype. It includes the requirements, dependencies, and operation details. A hardware design diagram, software design diagram, and testing commentary with logic analyzer results is included to reinforce the results and conclusions of the project. The complete code is located in Appendix A.

3. Requirements

The following are the given requirements for the special effects system prototype:

1. An external button will be wired to the mascot's glove or wrist. When the button is pushed, the following electronic effects will be invoked simultaneously:

    - Eyes will glow red. (Optional) Set a specified duration of glow time.
    - Ears will raise up. (Optional) Switch control of ears between TI board & NeuroSky board.
    - Mouth will emit bull noise. Sound is output on a mini speaker.
    - Nose will blow steam. Metal evaporator element is heated to produce vapor.

The requirements deemed "optional" are stipulated so since the original project proposal did not mention the optional requirements.

3.1 Dependencies

The following are dependencies for the special effects system prototype:

- Transistors
    i. Quantity: 4 (3 x 2N3904 + 1 x TIP33)
    ii. Component is used for the control of power received by each external device.
- AA & AAA Batteries
    i. Quantity (AA): 2+6, Quantity (AAA): 4, Voltage (AA & AAA): 1.5 V
    ii. Power source used by each external device used as follows:

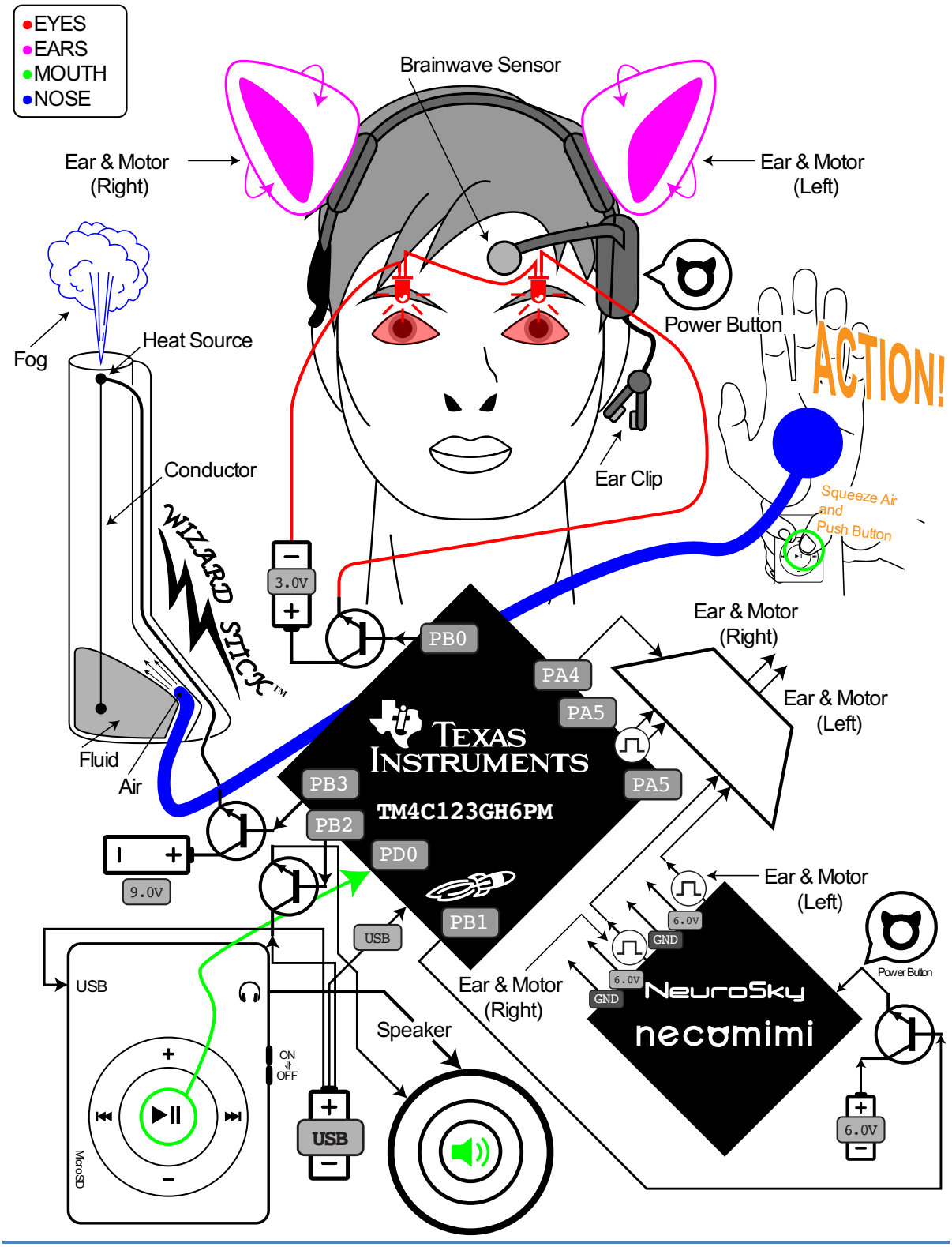
- Eyes: 2 x AA, Ears: 4 x AAA, Mouth: 0, Nose: 6 x AA
- USB Backup Battery
    i. Quantity: 1
    ii. Power source used by the TI microcontroller and external mini speaker.

3.2 Operation

When the system is powered on via button, the TI board sends a logic high to each of the four transistors to which each external device is interfaced with. Thus when pushed, the button will simultaneously active all four special effects denoted in the project requirements. When the special effects system is in the ON state, the next button push will turn the system OFF.

4. Design Details

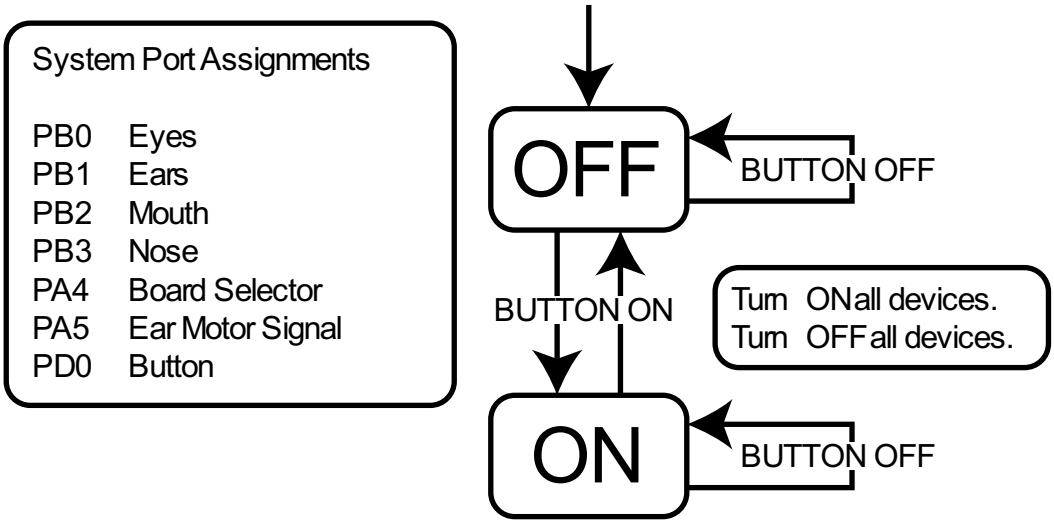
4.1 Hardware Design



4.2 Software Design

The software was written in Keil uVision and programmed onto a TM4C123GH6PM microcontroller. The software consists of two interrupts. The first interrupt is triggered on Port D0 when the button is pressed. When this occurs, the code will output high through Port B0-B3 to turn on the hardware. It will also turn on a 1 ms timer which interrupts every time it finishes. When it finishes it will check and if 16 ms has passed, it will output high on Port A5 which sends the correct data to the ears to make them perk up. When the button is pressed again it will turn off.

One problem encountered was with regards button debouncing. The button would send a high for a very short time and then go low and back high. Due to debouncing, the button interrupt would get entered twice and turn on the equipment and turn it off right after faster than the eye could see. The debouncing problem was fixed by using a counter that doesn't recognize the interrupt the first time it enters but it does the second time.



5. Testing

5.1 Eyes (RED LEDs)

The eyes are made from two red LEDs connected in series with a battery source and a potentiometer to adjust the brightness. As with all the other external devices, a transistor is used to readily connect or disconnect the power source of each respective external device depending on the low or high state of the source pin on the TI board.

5.2 Ears (Necomimi™)

The ears natively work on the basis of the user's brainwave state to invoke specific patterns of movement in the ear motors. Each ear receives its own independent power, ground, and data wires from the native microcontroller of the ear device. Using the TI board, the desired behavior was to emulate the signal necessary to quickly move the ears into the up position. Careful measurements from the logic analyzer of the data signal emitted from the native microcontroller

yielded successful results. The code written for the TI board emulates the signal needed to move the ears into the up position. See **Fig. 5-2(a,b,c,d,e)** in the Appendix.

5.3 Mouth (Speaker)

The mini speaker and associated mini MP3 player devices unfortunately yielded the most significant delays of the overall system. Destructive interference was evident as the sound from the speaker would come out distorted once interfaced with a transistor component. A solution was reached once a suitable mini speaker unit was located that could interface with the selected transistor component without distortion.

5.4 Nose (Wizard Stick™)

The handheld fog generator bears the unique consideration of a device dependent on high voltage to power it. Unlike the other external devices bearing lower voltage requirements that used transistors of lower tolerances, the fog generator required the use of a physically larger transistor of higher voltage tolerance. To emit larger concentrations of fog, the user may squeeze an air pump at the same time they push the button.

5.5 Button

The interrupt invoked by the user's push of the button is shown to be successful as demonstrated by **Fig. 5-5** in the Appendix. However, the button push shows that debouncing behavior is evident. The effects of debouncing are bypassed by incorporating the appropriate delay in the ISR handler function assigned to the button.

6. Conclusions

The design described in this document provides a functional prototype for a system of electronic special effects that can safely be integrated with a bull mascot costume. Sufficient testing of all requirements covering each of the four external device systems has yielded successful results.

However, this design is not optimal. Mainly due to the compromise in native idle behavior of the ear motors. Ideally it was desired that a multiplexor component be able to control when the TI board and the NeuroSky board would have independent control of the ears. Thus when under TI board control, the ears would instantly raise up. While under NeuroSky board control, the ears would resume their native intended behavior. For the final implementation of the actual mascot costume, it is likely that most of the hardware used would actually have to be physically modified or substituted for smaller sized equivalent parts. While the prototype does work, it would not be able to stand the environment and physical endurance requirements of a sports team mascot.

7. Credits

Programming, Testing, and Demonstration ..... Nathan Yost
Illustrations, Documentation, Inventory ..... Joel Meine
Commentary ..... Nathan Yost & Joel Meine

Appendix

A.1 Program Code

```
// Project BIG BLUE
// Code by Nathan Yost
// Edited by Joel Meine

// TI TM4C123GH6PM Micro-Controller
// MDS = Microcontroller Data Sheet

// System Base Addresses
unsigned char *SYSCTL = (unsigned char *) 0x400FE000;
unsigned char *M4CP = (unsigned char *) 0xE000E000;
unsigned char *GPIOA = (unsigned char *) 0x40004000;
unsigned char *GPIOB = (unsigned char *) 0x40005000;
unsigned char *GPIOD = (unsigned char *) 0x40007000;

// System Port Assignments
// GPIOA_   Sources the output to move the ear motors in conjunction with SysTick.
// GPIOB_   Sources the output used to turn on each transistor used by each external device.
// GPIOD_   Sources the input used to monitor for user button push.

// Global Variables
volatile int Debounce = 0; // To account for debouncing, the button sends two signals.
unsigned int Count = 0;    // Counts the number of timer expirations.
unsigned int Level = 0;    // Records the state of the timer signal; LOW=0, HIGH=1.

void GPIOInit()
{
  // 1. Enable the Clock.
  SYSCTL[0x608] = 0xFF;    // Enable Clock

  // 2. Unlock Pins and/or Disable Alt. Function.
  GPIOA[0x420] = 0x0;      // Disable Alt. Function
  GPIOB[0x420] = 0x0;      // Disable Alt. Function
  GPIOD[0x520] = 0x4B;     // Unlock Code (1 of 4)
  GPIOD[0x521] = 0x43;     // Unlock Code (2 of 4)
  GPIOD[0x522] = 0x4F;     // Unlock Code (3 of 4)
  GPIOD[0x523] = 0x4C;     // Unlock Code (4 of 4)
  GPIOD[0x524] = 0x03;     // Unlock Pins 0 and 1.

  // 3. Set Pins as Input or Output.
  GPIOA[0x400] = 0xFF;     // Output: Pins 0-7
  GPIOB[0x400] = 0xFF;     // Output: Pins 0-7
  GPIOD[0x400] = 0xFE;     // Input: Pin 0 (BUTTON)

  // 4. Set for Pull-Up or Pull-Down (Optional).
  GPIOD[0x510] = 0x01;     // Pull-Up (Enable): Pin 0

  // 5. Enable Interrupts (Optional).
```

```
  M4CP[0x100] = 0x08;       // Interrupts (Enable): GPIOD
  GPIOD[0x404] = 0x0;       // Set to Edge-Sensitive.
  GPIOD[0x40C] = 0x1;       // Set to Rising Edge.
  GPIOD[0x408] = 0x0;       // Set to Not Interrupt on Both Edges.
  GPIOD[0x410] = 0x1;       // Interrupts (Enable): Pin 0

  // 6. Enable Pins.
  GPIOA[0x51C] = 0xFF;      // Enable Pins: 0-7
  GPIOB[0x51C] = 0xFF;      // Enable Pins: 0-7
  GPIOD[0x51C] = 0x01;      // Enable Pins: 0
}

void SysTickInit()
{
  // 1. Stop the Timer.
  M4CP[0x010] = 0;

  // 2. Set the Initial Value.
  M4CP[0x014] = 0xFF;       // (INITIAL + 1)*(1/SYSCLK) = T
  M4CP[0x015] = 0x3F;       // SYSCLK = 12 MHz, T = 1.365 ms
                            // ..INITIAL = 16383 = 0x3FFF

  // 3. Clear Current Timer Value.
  M4CP[0x018] = 0;

  // 4. Set Timer Options & Start Timer.
  // M4CP[0x010] = 0x7;     // Use System Clock; Enable Interrupts; Start Timer
}

void GPIOD_Handler(void)    // User has pressed the button to power on the external devices.
{
  if(Debounce==0)           // Delay start of button interrupt due to debouncing.
  {
    Debounce++;
  }
  else
  {
    if (GPIOB[0x3FC]==0x0)        // Are the external devices, connected to GPIOB, turned OFF?
    {
      GPIOB[0x3FC] = 0xF;         // If YES, then turn ON all external devices through their
                                  respective transistor via GPIO pins 0 through 3.
      M4CP[0x010] = 0x7;    // Use System Clock; Enable Interrupts; Start Timer >> Move Ears Up.
    }
    else
    {
      GPIOB[0x3FC] = 0x0;         // If NO, then turn OFF all external devices through their
                                  respective transistor via GPIO pins 0 through 3.
      M4CP[0x010] = 0x0;    // ...; ...; Stop Timer >> Move Ears Down.
    }
    Debounce = 0;           // Enable debounce delay to trigger next time the button is pushed.
  }
  GPIOD[0x41C] = 0x1;       // Clear Interrupt
}

void SysTick_Handler(void)
{
  Count++;                  // Increment count of timer expirations.

  if(Level==0)              // If signal to ear motors is low...
  {
    if(Count==16)           // If timer has expired 16 times (i.e. apx. 16 ms)...
    {
```

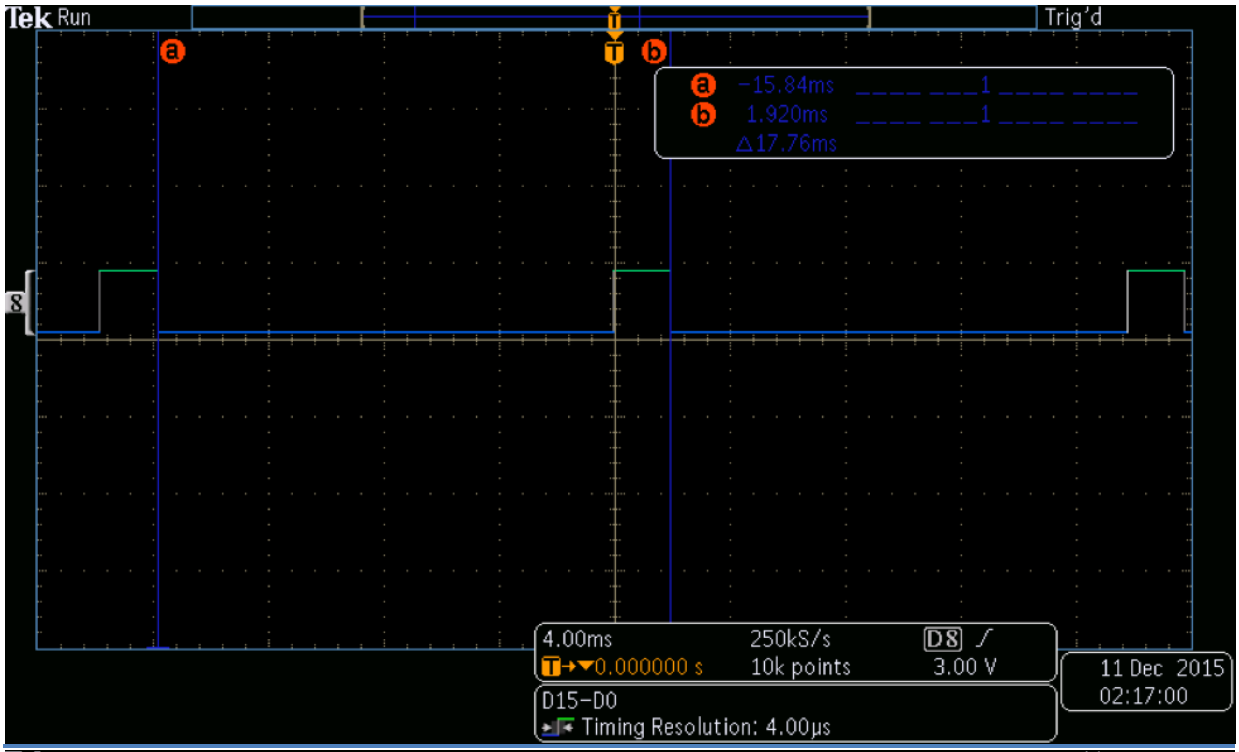
```
      Count = 0;           // Reset count of timer expirations...
      GPIOA[0x3FC] = 0x20; // Raise signal high on GPIOA:5 for apx. 1 ms...
      Level = 1;           // Record that signal is now high.
    }
    else
    {
      GPIOA[0x3FC] = 0x00; // If timer has not yet expired 16 times, then keep signal low.
    }
  }
  else if(Level==1)        // If signal to ear motors is high...
  {
    if(Count==16 || Count==17)    // If timer has expired 16 or 17 times (i.e. apx. 16-17 ms)...
    {
      GPIOA[0x3FC] = 0x00; // Drop signal low on GPIOA:5 for apx. 1 ms.
    }
    if(Count==17)
    {
      Count = 0;           // Reset count of timer expirations...
      GPIOA[0x3FC] = 0x00; // Drop signal low on GPIOA:5 for apx. 1 ms...
      Level = 0;           // Record that signal is now low.
    }
    else
    {
      GPIOA[0x3FC] = 0x00; // If timer has not yet expired 16-17 times, then force signal low.
    }
  }
}

int main(void)
{
GPIOInit();
SysTickInit();
while(1)
{
}
}
```

A.2 Logic Analyzer Data

Fig. 5-2

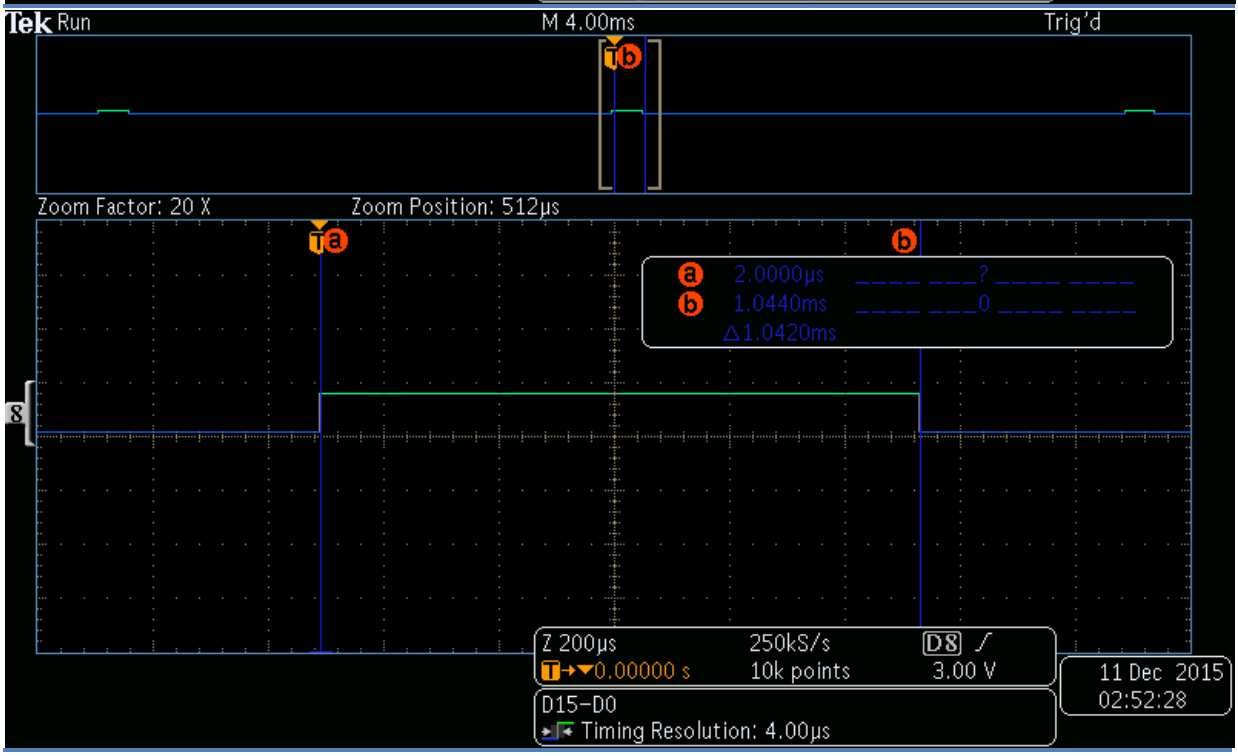
a.

Ears
• No Motion



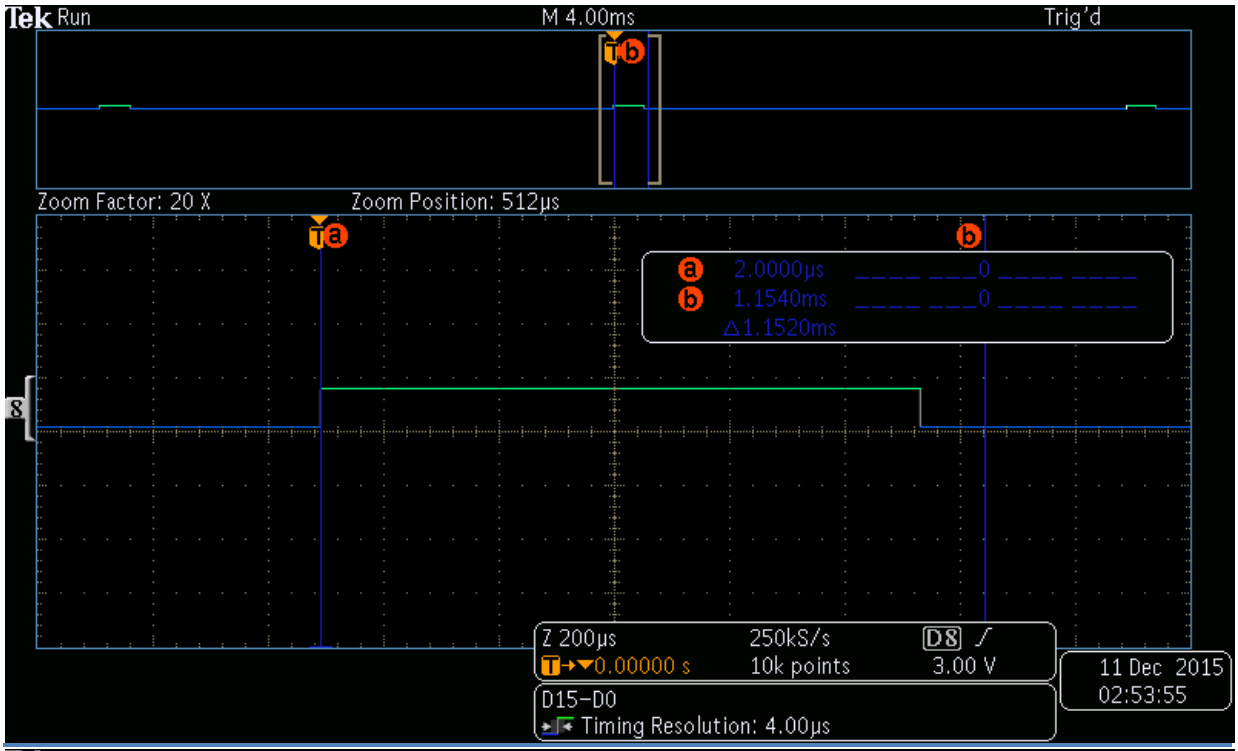
b.

Ears
• In Motion



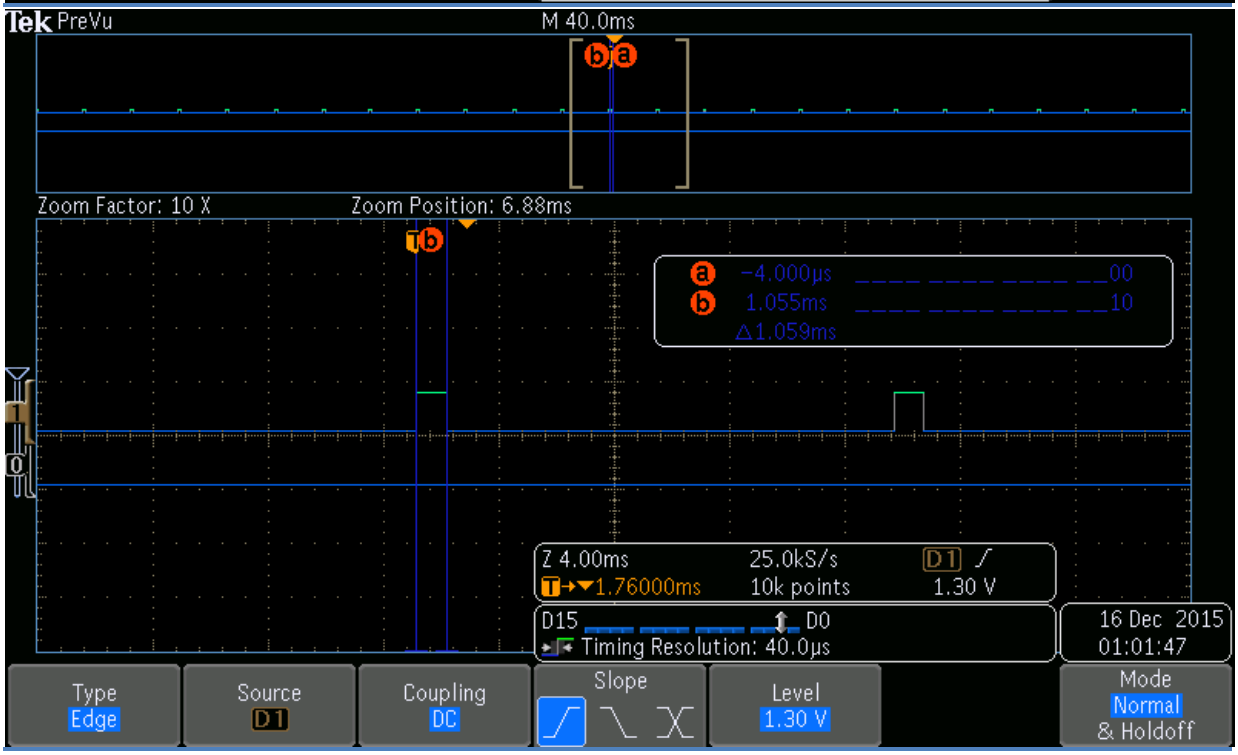
c.

Ears
• In Motion



d.

Ears
• uC SysTick



e.

Ears
• Freq. Period

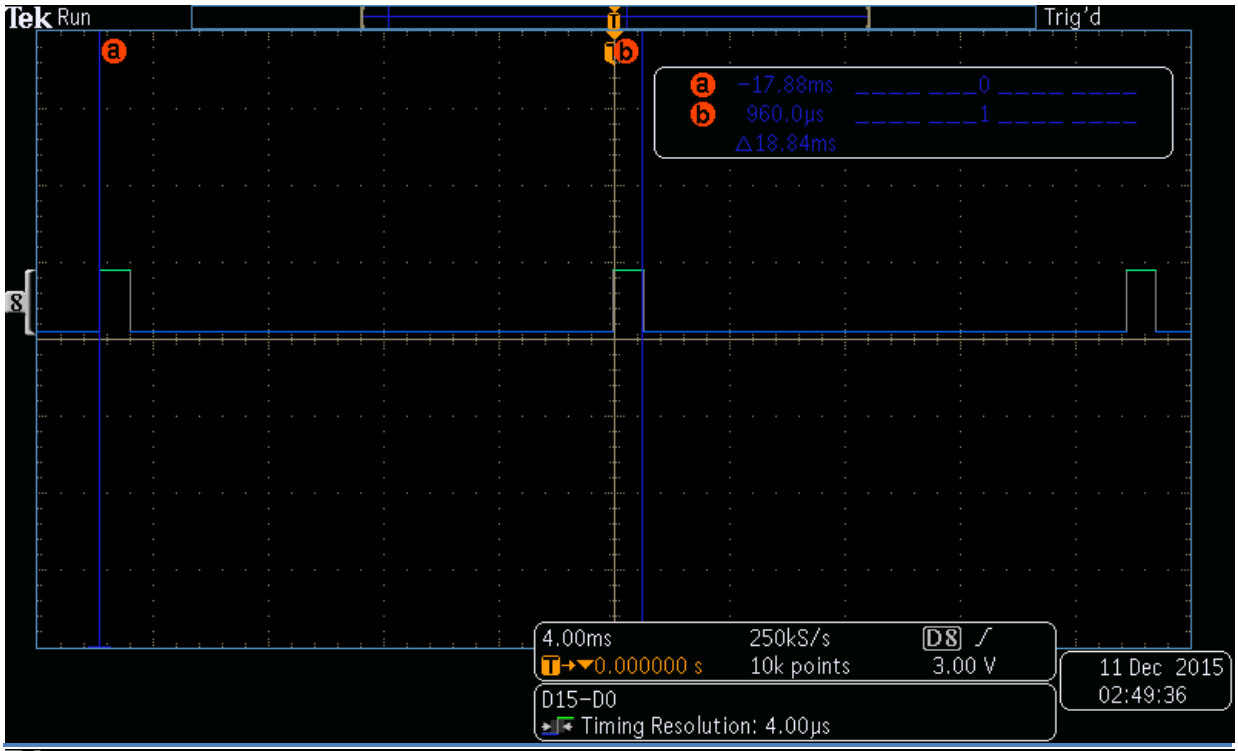


Fig. 5-5

Button
• Debouncing