ACCELEROMETER MOUSE

MANAV KATARIA
VINU BHASKAR
ABHISHEK SHARMA
MIZAN ABRAHA

SUPERVISOR: P.C.PANDEY

airmouse.iitb** [at] **gmail** [dot] **com

Abstract

[Many tasks that are performed on the computer require the use of both a keyboard on the mouse, and many people find it frustrating and awkward to have to switch back and forth between them. The air mouse can be a great solution for this.] As computers have evolved from 66MHz to the 3+GHz today, the mouse has remained relatively unchanged. There have been only two major changes to the mouse since its inception: more buttons, and interface changes (PS/2, USB, Wireless) but the core interaction method has not changed. The touch pad mouse and the stick mouse have been designed for a laptop, but have not gained momentum outside of laptops.[And most laptop users usually carry a USB mouse with the laptop as well.]

The accelerometer based mouse can be treated as the new age input device. It is more natural in its feel and provides the user with better ease of use. The user interface of applications can be changed to utilize the free hand movement possible with the device.

1. Introduction

The aim behind the project is to be able to sense the movement of a user's hand and use it to control the movement of mouse on screen. For sensing hand movement, we use an accelerometer. This would give us the acceleration sensed by the device in each axis. We use this data to find the change in movement of mouse pointer, which is then transmitted serially to the host computer using RS 232 communication.

2. High Level Design

We are using accelerometer to measure the acceleration of a user's hand and integrate that acceleration into a change in position. The math behind this is very easy - using Verlet Integration, we would approximate the integral of the acceleration to the second degree. Verlet integration interpolates between to measured accelerations, using the average slope between them to derive velocity. This is sometimes called "trapezoidal integraton." Here are the equations:

$$x(n) = x(n-1) + v(n-1)\Delta t + 1/2\alpha\Delta t^2$$

But gravity would affect the accelerometers for sure , this acceleration would be added into the integral, and with no negative acceleration to remove it, the change in position would grow boundlessly.

Because of this, we decided to construct a tilt mouse instead. While not as impressive as a position tracking device, the tilt mouse is easy to use and almost as neat as a position tracking mouse. The math behind this scheme is very easy - measure the acceleration due to gravity on the mouse, and multiply this by some constant to scale your output to a desirable level. The Microsoft serial mouse protocol uses an 8-bit twos complement number scheme to send data to the computer, and the numbers outputted by the Atmel Mega32's analog-to-digital converter can be conveniently represented in the same number of bits. However, the numbers outputted by the accelerometer had to be altered to normalize voltage extremes to -128 and 127 and the accelerometer's neutral output to 0. In addition to scaling the output, we also used a step filter on the data to make the mouse easier to use. Our accelerometer was so sensitive that the slightest motion of one's hand would cause the device to output a nonzero acceleration. To give the user a more stable region around the zero point, we quantized all outputs below a certain level to 0, then normalized any outputs out of this cutoff range by the breadth of the cutoff range. For example, if the cutoff were abs(10), the intersection of all numbers > -10 and all numbers < 10 defined the cutoff region, and any outputs outside of this range would be normalized by +10 or -10, depending on whether the output were negative or positive, respectively.

Our design is related to two standards. We use the Microsoft serial mouse protocol and an RS232 serial connection. RS232 uses -+12V for high and low signals, respectively. RS232 idles at +12V. Our microcontroller outputs 5V high, 0V low serial (TTL level), so we had to use a serial level shifter chip to allow the microcontroller to properly speak with the computer. The Microsoft serial mouse protocol works as such: when the RTS line of the serial connection is pulled logic high then falls logic low again, the computer expects you to send an identifier string to it so that it can determine what type of device you are. We are using "M" microsoft protocol, 2 buttons.

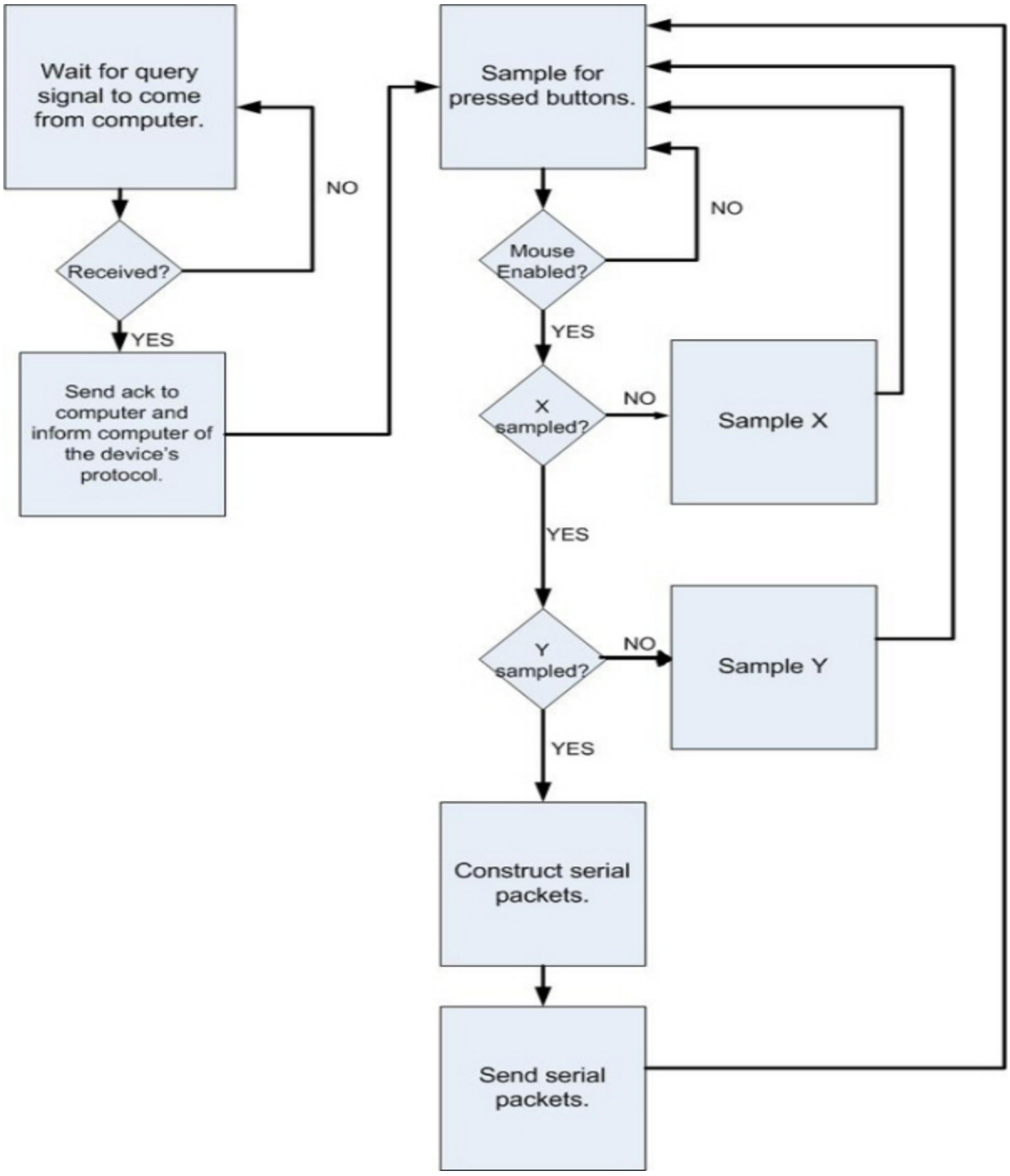
	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 1	'1'	L	R	Y7	Y6	X7	X6
byte 2	'0'	X5	X4	X3	X2	X1	X0
byte 3	'0'	Y5	Y4	Y3	Y2	Y1	Y0

Only the first byte has MSB set, for sequence identification.

- L and R are left and right button status (1 = pressed).
- X7..X0, packed to form an 8-bit integer (X0 is LSB), is the two's complement increment (in mouse unit) of the X axis.
- Y7..Y0, packed to form an 8-bit integer (Y0 is LSB), is the two's complement increment (in mouse unit) of the Y axis.

3. Program Design

Here is a block diagram of our program



First we wait for the computer to toggle the RTS line. When it does, we send it "M," the indicator that there is a Microsoft serial mouse. We respond to 5 queries, then disable query response and start mouse functioning.
After the mouse has started functioning, the buttons are sampled on every cycle. If the mouse is not enabled, the program cycles through the button sampling until the mouse is enabled. After the buttons are sampled and the mouse is enabled, a check is done to see which acceleration axes have been sampled. If X has not been sampled, set the sample multiplexer to Y, sleep until X is ready, and sample X. The multiplexer is set to Y before we sample X because the current sample being processed is always taken off of the current multiplexer value, which is X if X has not been sampled. If X has been sampled, we check to see if Y has been sampled. If it has not been, set the multiplexer to X, sleep until Y is ready, and sample Y. If Y has been sampled, modify the samples to the correct two's complement numbers, scale them, and create the serial packets. Send the serial packets, indicate that X needs to be sampled, and return to the beginning of the loop. The mouse is enabled and disabled by one of the sampled buttons. If the button value is read as "pressed" and its previous value was "not pressed," the mouse enable/disable is toggled.

4. Hardware Design

We used the MMA7260QT accelerometer from Freescale Semiconductors. It is a low cost capacitive micromachined accelerometer features signal conditioning, a 1-pole low pass filter, temperature compensation and g-Select which allows for the selection among 4 sensitivities. Zero-g offset full scale span and filter cut-off are factory set and require no external devices. Includes a Sleep Mode that makes it ideal for handheld battery powered electronics.

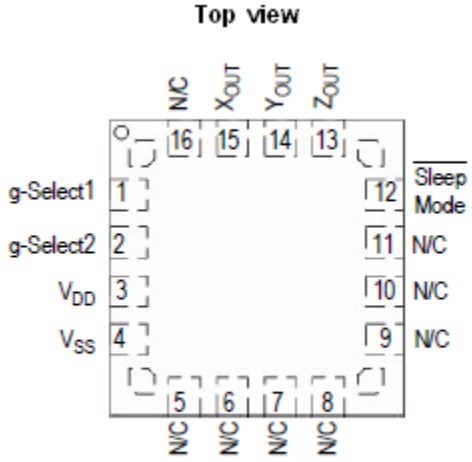
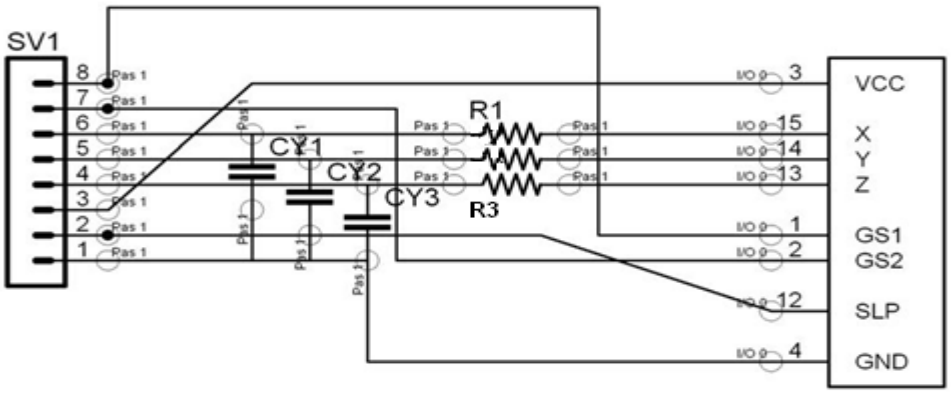
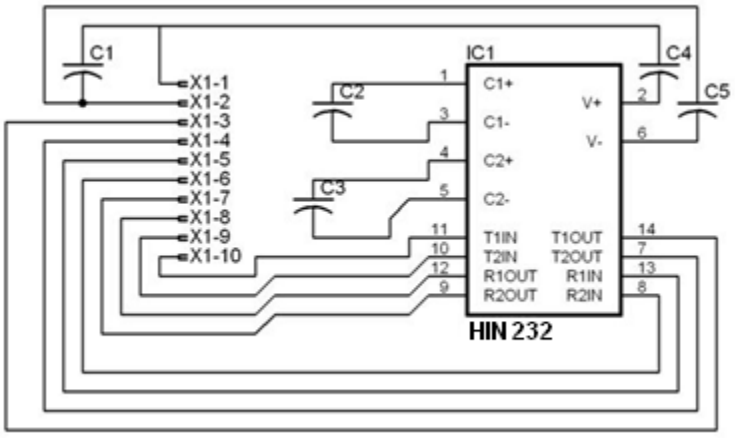


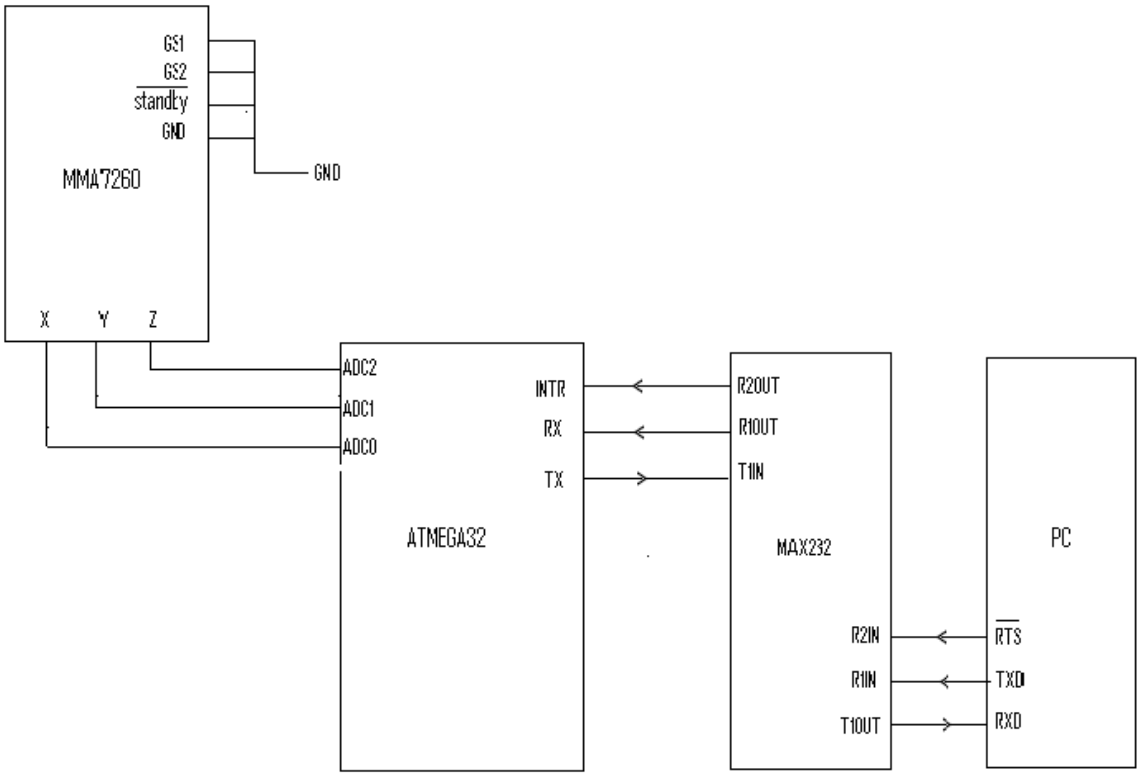
fig. pin connections



Schematic for MMA7260QT



Schematic for HIN232CP


GS1
GS2
standby
GND
MMA7260
GND
X
Y
Z
ADC2
ADC1
ADC0
INTR
RX
TX
ATMEGA32
R2OUT
R1OUT
T1IN
MAX232
R2IN
R1IN
T1OUT
PC
RTS
TXD
RXD