A REPORT TITLED

Human Interface Device for the differently abled

Submitted
in partial fulfillment of
the requirements of
Electronics in Service of Society Lab

By

Nilay Sheth (121060065)

Swapnil Zambare(121060057)

Akshay Budale(121060060)

Ashish Sharma(121060035)

Sameer Deogaokar(121060061)

Akash Chopade(121060063)

T.Y B.TECH.(ELECTRONICS)

Guide

DR. R.D.DARUWALA
PROFESSOR & DEAN, ACADEMICS
VTJI , Mumbai.

Veermata Jijabai Technological Institute, Mumbai 400 019

CERTIFICATE

The report, **“**Human Interface Device for the blind**”** is found to be satisfactory and is approved for **Electronics in Service of Society Lab (T.Y.BTech Electronics).**

Guide	**Examiner**

Date:

Place :

Acknowledgments:

It gives me great pleasure to present this report, a written testimonial of a fruitful experience. I take this opportunity to thank my project guide **Dr. R.D.Daruwala**, Professor & Dean Academics, VJTI, working with whom is a delightful and learning experience.

I would also like to express my sincere thanks to Lab Co-ordinator of Electrical Engineering department **Ms. Riya Saini**, for her contribution towards developing my interest in this field.

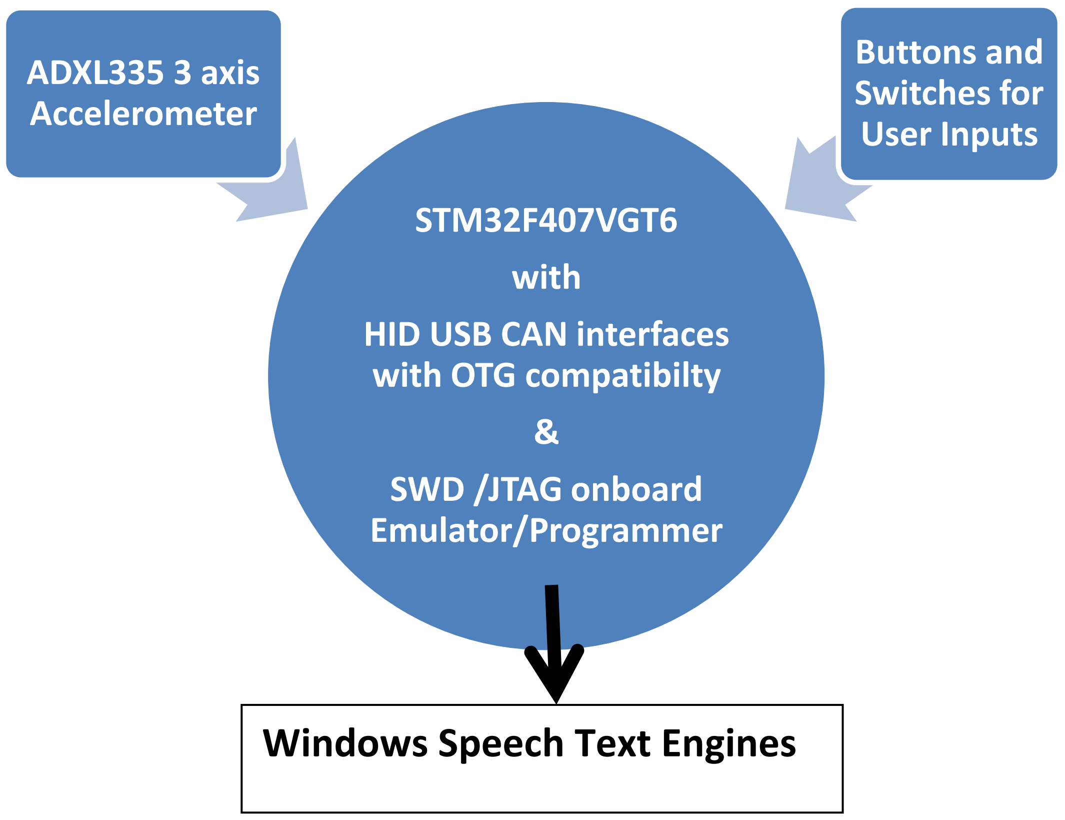
Abstract

The problem statement being to ease the difficulties of using a computer by replacing standard input devices by a cheap smart mouse. The interface enables users to quickly navigate between windows and opening frequent tabs in order to access the mail, accessibility options & PDF files. It is compatible with Microsoft Narrator, the text to speech convertor to give out feedback through audio. The gesture based device has buttons and accelerometers with haptic feedback to set up this HID environment.

CONTENTS

INTRODUCTION

Aim being to simplify use of computers for the disabled, we first interfaced the speech text engines of windows to the inputs that a user gave. Later a HID Device was prototyped with USB capabilities to detect itself as a Keyboard and a Mouse at the same time. So at the end result we were able to incorporate a Gesture based Mouse & Keyboard and Windows Engines at the software end which give user a easier way to use computers. This device does not require any drivers or special setups to recognize itself , it is a standalone device itself. The report further will break down the device's working into block diagrams and how we implemented the hardware and software required to accomplish this aim.


ADXL335 3 axis Accelerometer
Buttons and Switches for User Inputs
STM32F407VGT6
with
HID USB CAN interfaces with OTG compatibilty
&
SWD /JTAG onboard Emulator/Programmer
Windows Speech Text Engines

1) An STM32F407-ARM Cortex M4 Based uC

The features Include :-

• Core: ARM® Cortex®-M4 32-bit CPU with FPU(72 MHz max), single-cycle multiplication and HW division, 90 DMIPS (from CCM)/1.25 DMIPS/MHz (Dhrystone 2.1)performance at 0 wait state memory access, DSP instruction and MPU (memory protection unit)

• Operating conditions:– VDD, VDDA voltage range: 2.0 V to 3.6 V

• Memories

– 128 to 256 Kbytes of Flash memory

– Up to 40 Kbytes of SRAM, with HW parity check implemented on the first 16 Kbytes.

– Routine booster: 8 Kbytes of SRAM on instruction and data bus, with HW parity check (CCM)

• CRC calculation unit

• Reset and supply management

– Power-on/Power-down reset (POR/PDR)

– Programmable voltage detector (PVD)

– Low power modes: Sleep, Stop and Standby

VBAT supply for RTC and backup registers

• Clock management

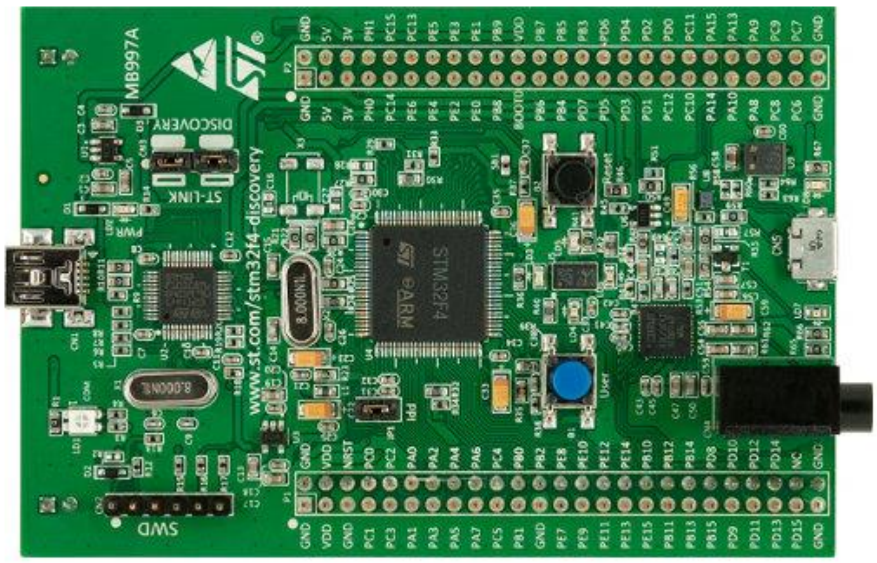
– 4 to 32 MHz crystal oscillator

– 32 kHz oscillator for RTC with calibration

– Internal 8 MHz RC with x 16 PLL option

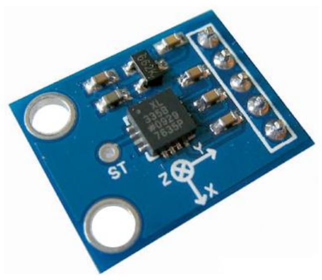
– Internal 40 kHz oscillator

• Up to 87 fast I/Os



– All mappable on external interrupt vectors
– Several 5 V-tolerant
• 12-channel DMA controller
• Four ADCs 0.20 µS (up to 39 channels) with selectable resolution of 12/10/8/6 bits, 0 to3.6 V conversion range, single ended/differential input, separate analog supply from 2 to 3.6 V
• Two 12-bit DAC channels with analog supply from 2.4 to 3.6 V
• Seven fast rail-to-rail analog comparators with analog supply from 2 to 3.6 V
• Four operational amplifiers that can be used in PGA mode, all terminals accessible with analog supply from 2.4 to 3.6 V
• Up to 24 capacitive sensing channels supporting touchkey, linear and rotary touch sensors
• Up to 13 timers
– One 32-bit timer and two 16-bit timers withup to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
– Two 16-bit 6-channel advanced-control timers, with up to 6 PWM channels, deadtime generation and emergency stop
– One 16-bit timer with 2 IC/OCs, 1OCN/PWM, deadtime generation and emergency stop
– Two 16-bit timers with IC/OC/OCN/PWM, deadtime generation and emergency stop
– Two watchdog timers (independent, window)
– SysTick timer: 24-bit down counter
– Two 16-bit basic timers to drive the DAC
• Calendar RTC with Alarm, periodic wakeup from Stop/Standby
• Communication interfaces
– CAN interface (2.0B Active)
– Two I2C Fast mode plus (1 Mbit/s) with20 mA current sink, SMBus/PMBus, wakeupfrom STOP
– Up to five USART/UARTs (ISO 7816 interface, LIN, IrDA, modem control)
– Up to three SPIs, two with multiplexed half/full duplex I2S interface, 4 to 16 programmable bit frames
– USB 2.0 full speed interface
– Infrared transmitter
• Serial wire debug, Cortex®-M4 with FPU ETM,
JTAG
• 96-bit unique ID.

2) ADXL 335- Accelerometer

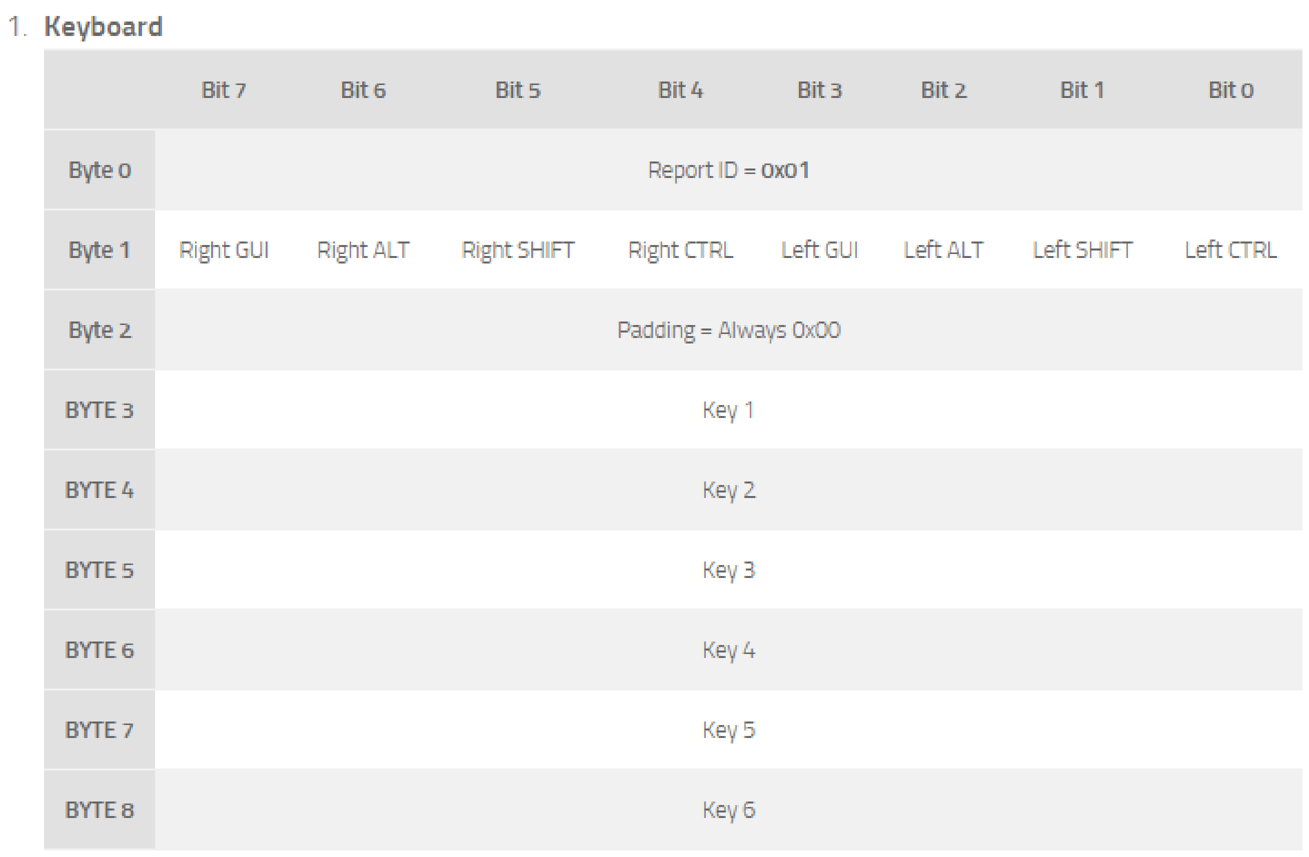


The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 g. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration. The user selects the bandwidth of the accelerometer using the CX, CY, and CZ capacitors at the XOUT, YOUT, and ZOUT pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

3) Buttons and Haptic Feedback
4) USB HID OTG interfacing components

A. Technical Background

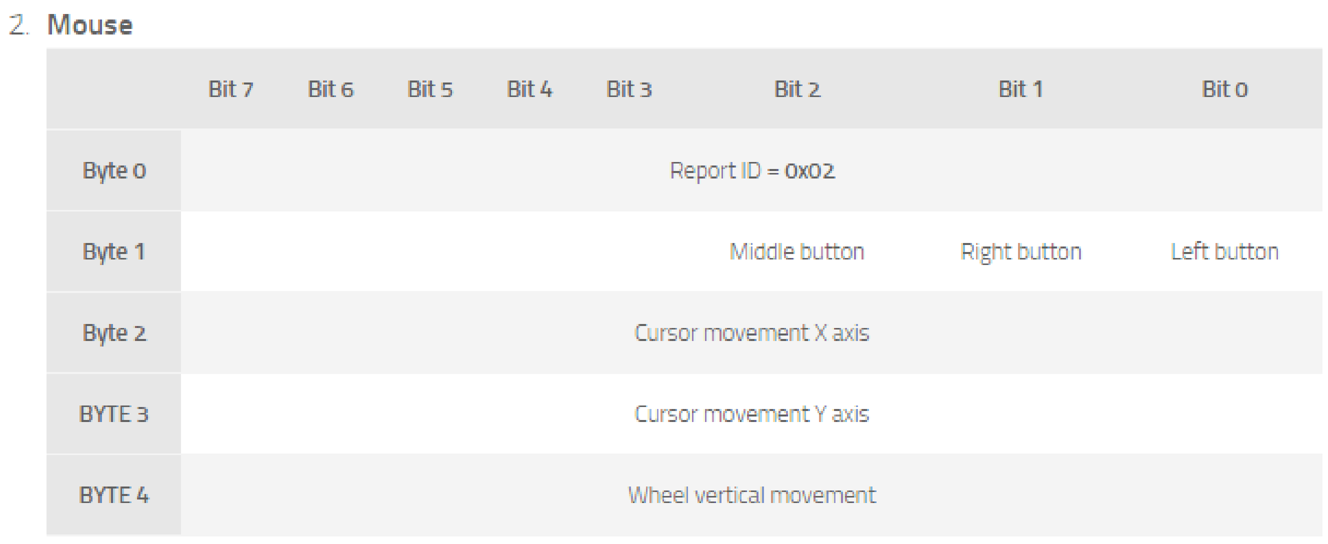
Why STM32F407 was the uC used?
-It is an SoC , bringing numerous peripherals together .
-It is the cheapest and latest ARM controller available.
-It has inbuilt HID options, requires strong structure based C codes.



1. **Keyboard**

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Report ID = 0x01							
Byte 1	Right GUI	Right ALT	Right SHIFT	Right CTRL	Left GUI	Left ALT	Left SHIFT	Left CTRL
Byte 2	Padding = Always 0x00							
BYTE 3	Key 1							
BYTE 4	Key 2							
BYTE 5	Key 3							
BYTE 6	Key 4							
BYTE 7	Key 5							
BYTE 8	Key 6							

<u>The Report Sent to the PC via our STM for emulating it as a Keyboard</u>



2. **Mouse**

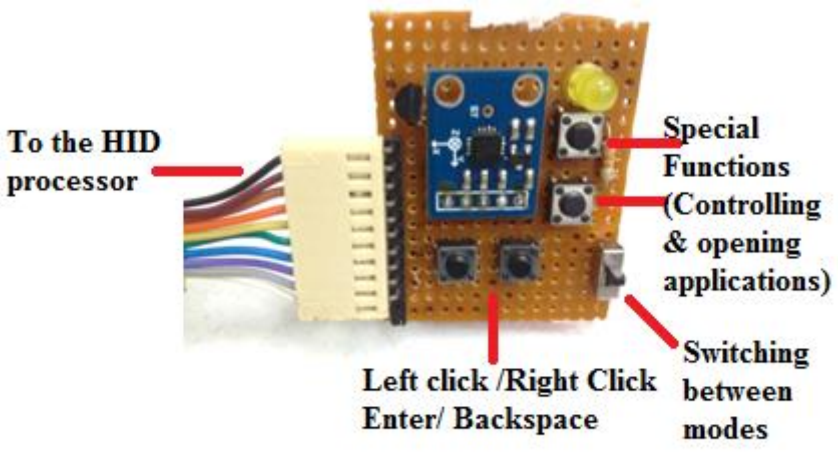
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Report ID = 0x02							
Byte 1						Middle button	Right button	Left button
Byte 2	Cursor movement X axis							
BYTE 3	Cursor movement Y axis							
BYTE 4	Wheel vertical movement							

<u>The Report Sent to the PC via our STM for emulating it as a MOUSE</u>

PROPOSED SOLUTION

In order to ease and alleviate the usage of HID devices, we incorporated a hands-free device which helped users navigate via gestures and clicks directly to any computer without installing different Drivers/softwares. We also interfaced this HID directly to the Windows Inbuilt Text to Speech engine which reads out whatever is selected on the screen. It also reads out lengthy PDF documents to users.

WORKING


To the HID processor
Special Functions (Controlling & opening applications)
Left click /Right Click Enter/ Backspace
Switching between modes

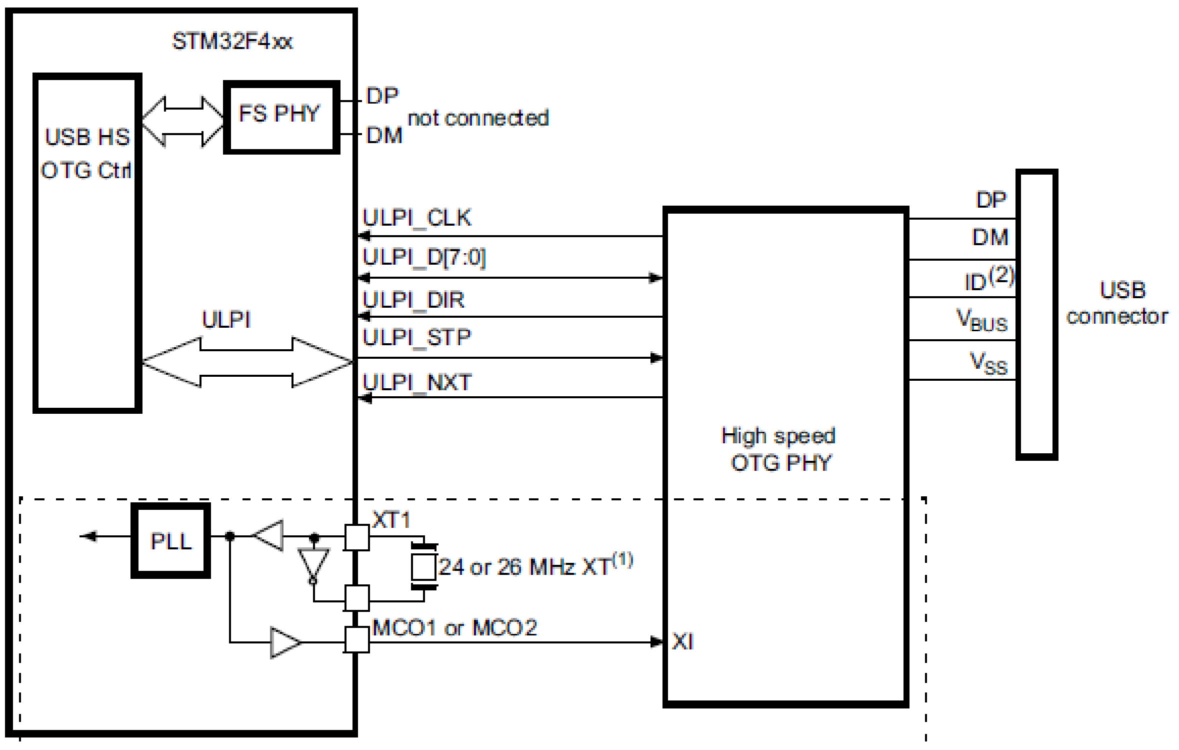
<u>**The Input Device**</u>

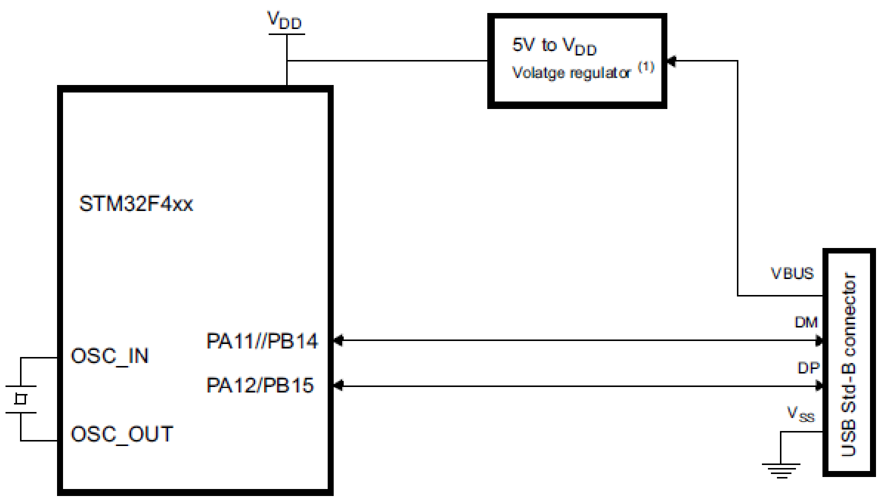
USB	FS MODE	HS IN FS MODE	DESCRIPTION
Data +	PA12	PB15	USB Data+ line
Data -	PA11	PB14	USB Data- line
ID	PA10	PB12	USB ID pin
VBUS	PA9	PB13	USB activate

IMPLEMENTATION

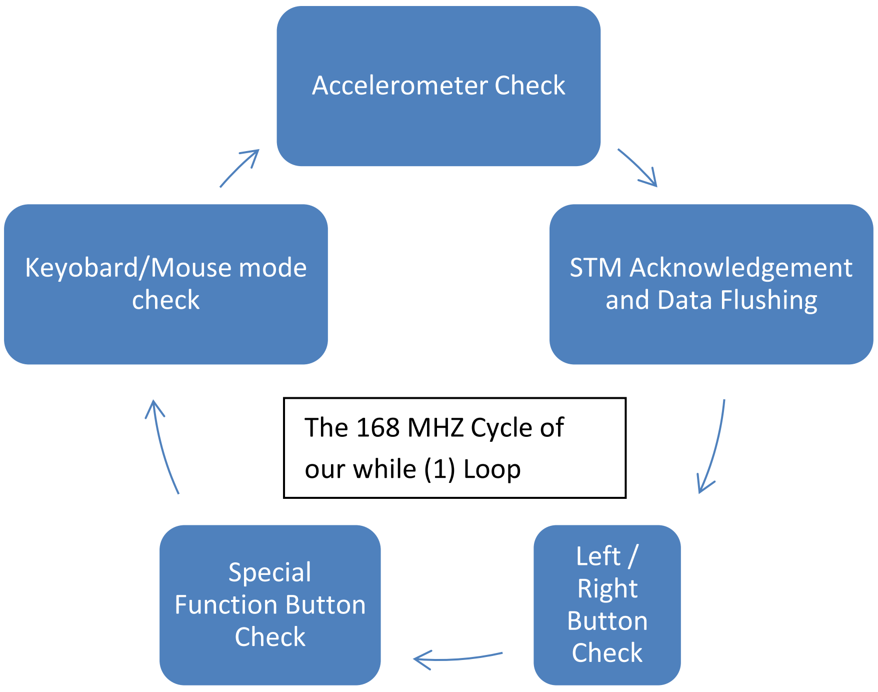
A. Hardware Implementation

<u>BLOCK DIAGRAM OF uC</u>





B. *Software Implementation*


Accelerometer Check
STM Acknowledgement and Data Flushing
Left / Right Button Check
Special Function Button Check
Keyobard/Mouse mode check
The 168 MHZ Cycle of our while (1) Loop

Features of our library

- Interface STM32F4 with computer as keyboard & mouse
- Supports up to 2 gamepads, 1 keyboard and 1 mouse
- Keyboard supports all special keys
- Mouse supports 3 main buttons, cursor movement and wheel vertical rotation
- Each of 2 gamepads supports up to 16 buttons and 2 joysticks
- Library works in **USB FS** or **USB HS in FS** mode
- Just plug USB in computer and works. No need for drivers

Dependencies

CMSIS
STM32F4xx
STM32F4xx RCC
STM32F4xx GPIO
STM32F4xx EXTI
HID Device stack provided by STMicroelectronics (included in library)

Our code :

```
/* Include core modules */
#include "stm32f4xx.h"
/* Include my libraries here */
#include "defines.h"
#include "tm_stm32f4_usb_hid_device.h"
#include "tm_stm32f4_delay.h"
#include "tm_stm32f4_disco.h"
#include "tm_stm32f4_adc.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "tm_stm32f4_adc.h"

/*Connections
PD6=Mouse/Keyboard mode switch
PD7= left/enter
PD8= right/backspace
PD10= keyboard special
PA2= x axis
PA1= y axis
*/

int main()
{
   uint8_t state1= 0;
   uint8_t state2= 0;
   uint8_t state3= 0;
   uint8_t state4= 0;
   uint8_t state5= 0;
   uint8_t state6= 0;
   uint8_t counter= 0;
   uint8_t counter1=0;
   TM_USB_HIDDEVICE_Keyboard_t Keyboard;
   TM_USB_HIDDEVICE_Mouse_t Mouse;

   GPIO_InitTypeDef GPIO_InitDef;

   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
   GPIO_InitDef.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7| GPIO_Pin_8 | GPIO_Pin_9 |
GPIO_Pin_10;
   GPIO_InitDef.GPIO_Mode = GPIO_Mode_IN;
   GPIO_InitDef.GPIO_OType = GPIO_OType_PP;
   GPIO_InitDef.GPIO_PuPd = GPIO_PuPd_UP;
   GPIO_InitDef.GPIO_Speed = GPIO_Speed_100MHz;
   GPIO_Init(GPIOD, &GPIO_InitDef);

   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

   GPIO_InitDef.GPIO_Pin = GPIO_Pin_11;
```

```
    GPIO_InitDef.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitDef.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDef.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitDef.GPIO_Speed = GPIO_Speed_100MHz;
    //Initialize pins
    GPIO_Init(GPIOD, &GPIO_InitDef);


    SystemInit();

    TM_ADC_Init(ADC1, ADC_Channel_1);   //for accx
    TM_ADC_Init(ADC1, ADC_Channel_2);   //for accy

    TM_DISCO_LedInit();

    TM_DELAY_Init();

    TM_USB_HIDDEVICE_Init();

    TM_USB_HIDDEVICE_MouseStructInit(&Mouse);
  TM_USB_HIDDEVICE_KeyboardStructInit(&Keyboard);
    GPIO_ResetBits(GPIOD, GPIO_Pin_11);

while (1)
{
if (TM_USB_HIDDEVICE_GetStatus() == TM_USB_HIDDEVICE_Status_Connected)
{

TM_DISCO_LedOn(LED_GREEN);


//////////////////////////Mouse Accelero/////////////////////////////////////////////
if(TM_ADC_Read(ADC1, ADC_Channel_2)<=1700&&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
Mouse.XAxis=-15;
    TM_DISCO_LedOn(LED_BLUE);
TM_USB_HIDDEVICE_MouseSend(&Mouse);
}
else if(TM_ADC_Read(ADC1, ADC_Channel_2)>2300&&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
Mouse.XAxis=15;
    TM_DISCO_LedOn(LED_BLUE);
TM_USB_HIDDEVICE_MouseSend(&Mouse);
}
if(TM_ADC_Read(ADC1, ADC_Channel_1)<=1700&&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
Mouse.YAxis=-15;
```

```
   TM_DISCO_LedOn(LED_BLUE);
TM_USB_HIDDEVICE_MouseSend(&Mouse);
}
else if(TM_ADC_Read(ADC1, ADC_Channel_1)>2300&&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
Mouse.YAxis=15;
   TM_DISCO_LedOn(LED_BLUE);
TM_USB_HIDDEVICE_MouseSend(&Mouse);
}
Mouse.XAxis=0;

Mouse.YAxis=0;
TM_DISCO_LedOff(LED_BLUE);
TM_USB_HIDDEVICE_MouseSend(&Mouse);
////////////////////////////////////////////////////////////////////////

//////////////////////Keyboard Accelero////////////////////////////////////////////////
if(TM_ADC_Read(ADC1, ADC_Channel_2)<=1700&&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
Keyboard.Key1=0x50;
TM_DISCO_LedOn(LED_ORANGE);
TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}

else if(TM_ADC_Read(ADC1, ADC_Channel_2)>2300&&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
Keyboard.Key1=0x4F;
TM_DISCO_LedOn(LED_ORANGE);
TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}

if(TM_ADC_Read(ADC1, ADC_Channel_1)<=1700&&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
Keyboard.Key1=0x51;
TM_DISCO_LedOn(LED_ORANGE);
TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}

else if(TM_ADC_Read(ADC1, ADC_Channel_1)>2300&&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
Keyboard.Key1=0x52;
TM_DISCO_LedOn(LED_ORANGE);
TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}
```

```
Keyboard.Key1=0x00;
TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
TM_DISCO_LedOff(LED_ORANGE);

//////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////////////

if ((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_7)) && state1 == 0 &&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
    state1 = 1;
    GPIO_SetBits(GPIOD, GPIO_Pin_11);
    TM_DISCO_LedOn(LED_BLUE);
    Mouse.LeftButton = 1;
    TM_USB_HIDDEVICE_MouseSend(&Mouse);
}

else if (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_7) && state1 == 1 &&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
    state1 = 0;
    Mouse.LeftButton = 0;
    GPIO_ResetBits(GPIOD, GPIO_Pin_11);
    TM_USB_HIDDEVICE_MouseSend(&Mouse);
    TM_DISCO_LedOff(LED_BLUE);
}
if ((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_7)) && state2== 0 &&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
    state2 = 1;
    GPIO_SetBits(GPIOD, GPIO_Pin_11);
    TM_DISCO_LedOn(LED_ORANGE);
  Keyboard.Key1 = 0x28;   //enter
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}
else if(GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_7) && state2 == 1 &&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
  state2 = 0;
    Keyboard.Key1=0x00;
    GPIO_ResetBits(GPIOD, GPIO_Pin_11);
    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
    TM_DISCO_LedOff(LED_ORANGE);
}
if ((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_8)) && state3 == 0 &&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
    state3 = 1;
    GPIO_SetBits(GPIOD, GPIO_Pin_11);
    TM_DISCO_LedOn(LED_BLUE);
    Mouse.RightButton = 1;
    TM_USB_HIDDEVICE_MouseSend(&Mouse);
}
else if (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_8) && state3 == 1 &&
GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6))
{
    state3 = 0;
    Mouse.RightButton = 0;
    TM_USB_HIDDEVICE_MouseSend(&Mouse);
    GPIO_ResetBits(GPIOD, GPIO_Pin_11);
    TM_DISCO_LedOff(LED_BLUE);
}
if ((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_8)) && state4== 0 &&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
    state4 = 1;
    GPIO_SetBits(GPIOD, GPIO_Pin_11);
    TM_DISCO_LedOn(LED_ORANGE);
  Keyboard.Key1 = 0x2A;   //backspace
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}
else if(GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_8) && state4 == 1 &&
(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_6)))
{
  state4 = 0;
    Keyboard.Key1=0x00;
    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
    TM_DISCO_LedOff(LED_ORANGE);
    GPIO_ResetBits(GPIOD, GPIO_Pin_11);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////

if((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_9)) && state6 == 0)
{
    while(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_9))
    {
     //start the motor from PinD11
     GPIO_SetBits(GPIOD, GPIO_Pin_11);
     counter1++;
     Delayms(500);
     //stop the motor
     GPIO_ResetBits(GPIOD, GPIO_Pin_11);
     Delayms(500);
    }
    state6=1;
}
else if(GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_9) && state6 == 1 )
{
```

```
if(counter1==1)
{
                    Keyboard.L_GUI = 1;/* Win button */
                    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
                    Delayms(100);
                    Keyboard.L_GUI=0;
                    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x11;    //n
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x04;    //a
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x28;   //enter
         TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}

if(counter1==2)
{
      Keyboard.L_CTRL=1;
      Keyboard.L_SHIFT=1;
      Keyboard.Key1=0x19;     //pdf reader mode
}
if(counter1==3)
{
                    Keyboard.L_GUI = 1;/* Win button */
                    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
                    Delayms(100);
                    Keyboard.L_GUI=0;
                    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x16;    //s
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x13;    //p
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x08;    //e
  TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);

  Keyboard.Key1 = 0x28;   //enter
         TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
}

TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
state6=0;
counter1=0;
Keyboard.L_GUI=0;
Keyboard.L_CTRL=0;
```

```
    Keyboard.L_SHIFT=0;
    Keyboard.L_ALT=0;
    Keyboard.Key1=0x00;
    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
    TM_DISCO_LedOff(LED_ORANGE);
}

///////////////////////////////////////////////////////////////////////////

if((!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_10)) && state5 == 0)
{
    while(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_10))
    {
     //start the motor from PinD11
     GPIO_SetBits(GPIOD, GPIO_Pin_11);
     counter++;
     Delayms(500);
     //stop the motor
     GPIO_ResetBits(GPIOD, GPIO_Pin_11);
     Delayms(500);
    }
    state5=1;
}
else if(GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_10) && state5 == 1 )
{
    if(counter==1)
    {
            //function1-alt tab
            Keyboard.L_ALT=1;
            Keyboard.Key1=0x2B;
    }
    if(counter==2)
    {
            //function2-alt f4
            Keyboard.L_ALT=1;
            Keyboard.Key1=0x3D;     //f4
    }

    if(counter==3)
    {
            //function3 - desktop
            Keyboard.L_GUI=1;
            Keyboard.Key1=0x07;     //d
    }

    if(counter==4)
    {
            //function4- my computer
            Keyboard.L_GUI=1;
            Keyboard.Key1=0x08;     //d
    }

    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
    state5=0;
    counter=0;
    Keyboard.L_GUI=0;
    Keyboard.L_CTRL=0;
    Keyboard.L_ALT=0;
    Keyboard.Key1=0x00;
    TM_USB_HIDDEVICE_KeyboardSend(&Keyboard);
    TM_DISCO_LedOff(LED_ORANGE);
}
///////////////////////////////////////////////////////////////////////

}
else
{
    TM_DISCO_LedOff(LED_GREEN);
}

}
}
```

RESULTS

-Communicated using protocols with the PC

-Interfaced STM as HID to the PC

-Interfaced sensors as inputs and haptic feedback as output to the STM

BILL OF MATERIALS

	Component	Manufacturer	Cost per component	Quantity	Total cost of components
1	STM32f407	ST Mirco	$ 2.95	1	$2.95
2	ADXL335	Analog Dev	$3	1	$3
3	Haptic F/b	Motor	$0.95	1	$0.95
4	Components	Ω,Buttons	$2	1	$2
				Total Cost of the Project	$8

CONCLUSIONS

We could reach to our main idea to make computers easy to use by changing the HID devices optimizing the ease of use and bringing majority of the controllers on hands.

After a blind/disable person is able to train himself to use this device it becomes relatively easy for him to use a computer. The computer NEED Not Require any extra drivers to support this device as it connects itself as a peripheral which sends reports similar to a mouse or a keyboard.

FUTURE SCOPE

Making it wireless was an expensive issue; we managed to prototype a cheap wired device. The uC could not be soldered separately because of clock skew after soldering on a matrix board. USB required clean noise free environment with proper biasing and precise paths on a PCB.

After adequate designing it could be shifted to a stand-alone device, and could have a huge market in Input devices as it is configured to enact itself as a hybrid Keyboard/Mouse optimized for the disabled.

REFERENCES

[1] http://stm32f4-discovery.com/

[2] http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419