University of Washington Bothell

School of Science, Technology Engineering, and Mathematics (STEM)
Division of Engineering & Mathematics

IR Theremin

Capstone - Summer Quarter 2020

Prepared by:
"Ryan" Young Beum Cho
Scott Bryar
Geoffrey Powell-Isom
Patricia Fang

Advised by:
Elaine Reeves
Joe Decuir

Table of Contents: **1**

Introduction

Abstract

Our team was asked by the company Hexabitz to create a project using their newly-released 1-dimensional LIDAR sensor module, which is part of their larger ecosystem of modular electronics. We were given a great deal of freedom in what to make, with the only restrictions being that it must use the LIDAR module and must be related to user interface. We made a touchless musical instrument that acts as a PC peripheral; an IR theremin, to use a loose definition of the word.

Motivation and Background

Hexabitz produces a variety of modular electronic components, each containing a microcontroller and one of a variety of sensors and actuators, such as speakers, lights, and thermometers. These components are designed to be combined into networks and programmed to interact with each other such that the network behaves in a desired way, similarly to an Arduino or Raspberry Pi with external sensors and actuators attached.
Hexabitz recently released a new module, the H08R6x, which is a 1-dimensional LIDAR sensor capable of accurately measuring the distance to an object less than 2 meters away. We were given an open-ended task; to invent and demonstrate some application of the device to the field of user interface, to serve as an inspiration to others.

Deliverable

The goal of this project was to create a musical instrument similar to a theremin or touchless drum kit using the Hexabitz LIDAR modules as a PC peripheral. A network of sensors, each acting as a single instrument, would detect the presence of a user's hand and transmit that detection to a PC via a USB cable. The PC would then play a user-configurable sound corresponding to the sensor the user moved their hand over (e.g sensor one is a drum and sensor two is a cymbal). The PC driver program would also have a command line interface for configuration. Finally, the project would have a page on Hackster.io explaining it, demonstrating it, and sharing the source code.

Hardware Specification

Overview

The hardware portion of our product consisted of two H08R60 sensor modules and the wiring connecting them to each other and to the PC. Module 2 had its power and communication ports connected using jumper wires to power and communication ports (respectively) on module 1, which in turn had other power and communication ports connected to the PC's USB port using the USB serial cable.

H08R60 Module

The sensor modules are small hexagonal circuit boards with an IR laser and receiver on the top and a microcontroller running the module firmware on the bottom. They arrive loaded with default firmware, but can be reprogrammed with new firmware (likely modifications of the existing firmware) compiled by users.
Every hexagonal Hexabitz module (pentagonal ones also available) features 12 pairs of pads onto which wires can be clipped or soldered. (Figure 1.2) The six pairs on the corners are for 3.3V power and ground on the front and back, respectively, while the six pairs on the sides are for individual I2C connections. Additional pads away from the edges exist for debugging purposes, but are generally not used in completed systems.

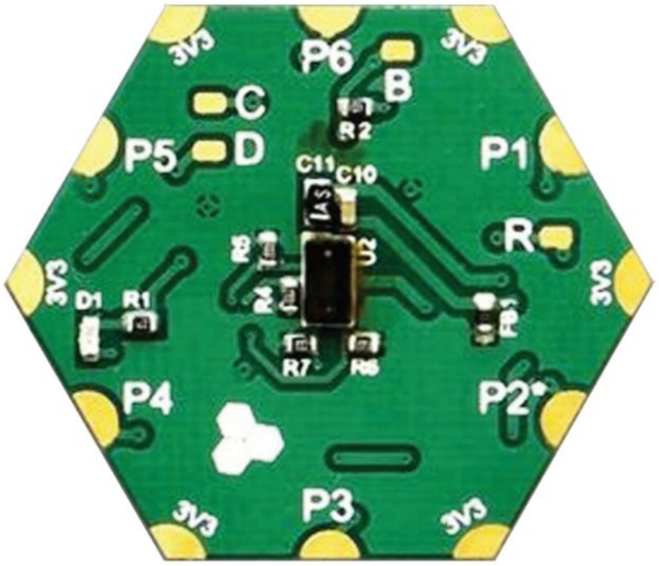


Figure 1.1: H08R60 Top View

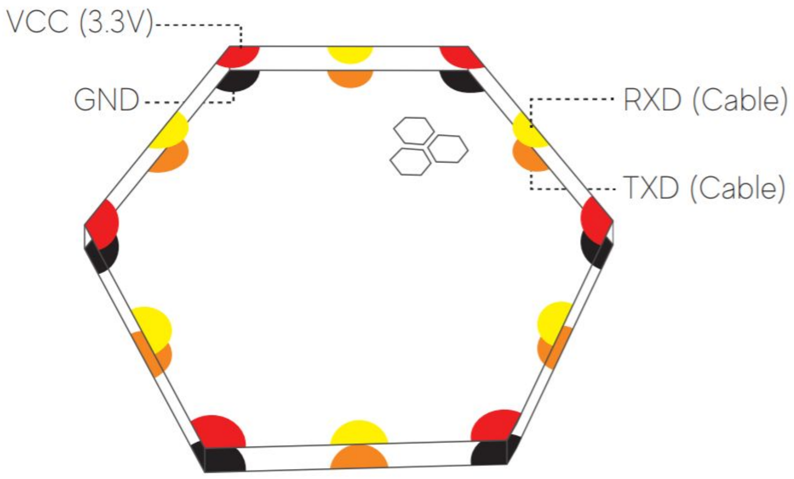


Figure 1.2: Hexabitz Module Connection Points

USB Serial Cable

Hexabitz makes a USB serial cable that has a serial-to-USB chip inside one end and splits into four jumper wires on the other. (Figure 1.3) Two of the wires are used for serial communication with the module while the other two are used for power and ground. This component was used to provide power to the modules and allow communication between them and the PC.

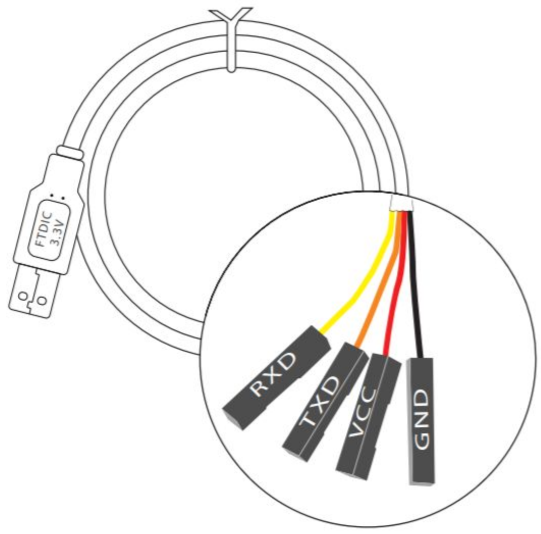


Figure 1.3: USB Serial Cable Diagram

Module Holder

The 3D Model of a module holder is designed to fix the module's position while in operation, while enabling the holder's height and angle to be adjusted according to the user's needs. Cinema 4D software was used to create the model (Figure 1.4). The left side of the figure shows how to adjust the height of the stand by plugging out a T-shaped lever from the grey cylinder. The right side of the figure shows a rotational axis of module holder's angle adjusted by rotating the two levers that fixes the module at the top of the module holder.

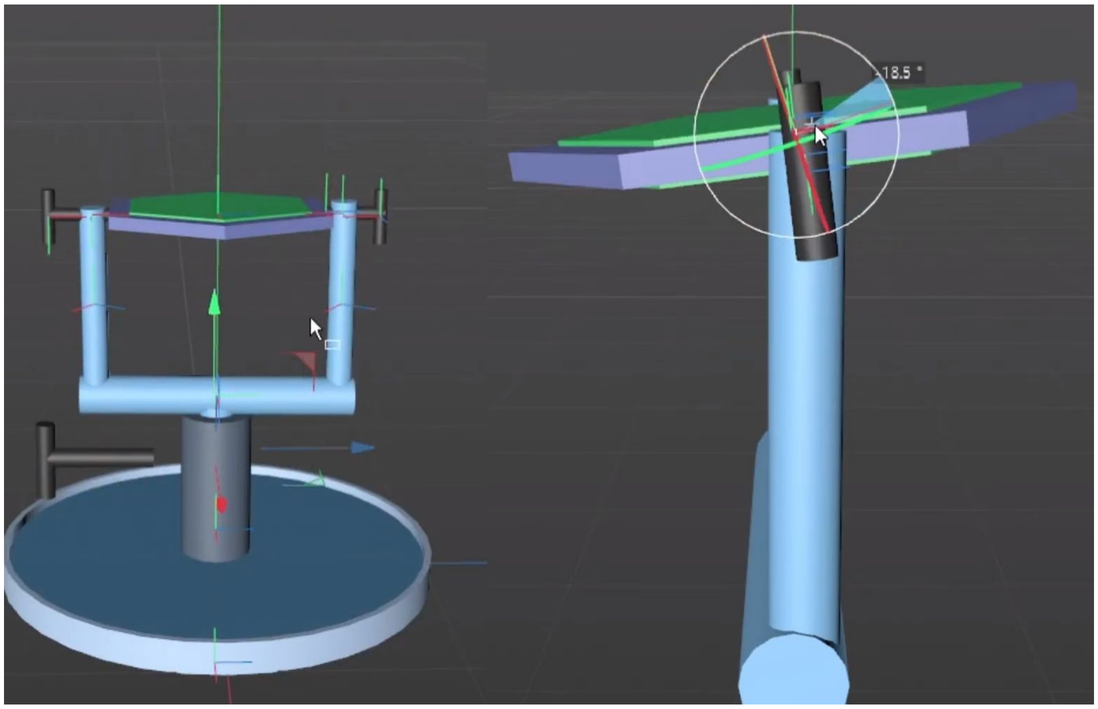


Figure 1.4: Module Holder

Software Specification

Overview

The software portion of our product mainly consists of module firmware which is used to send a serial message to the PC and Processing IDE Code which receives the message and processes the user instruction in the menu and sounds for IR Theremin accordingly.
Keil uVision was used to code module firmware and compile it to a HEX file for module 1 and module 2. Then, using STM32 Flash Loader, the compiled hex files were installed into each of the modules. Processing IDE code opens a CLI interface for users to run a sound test feature using keyboard input, open or close streaming the readings from a serial communication with the modules, and processes sounds using replay and overlay algorithm given the measured distances from each of the modules streamed in 50 milliseconds period.

Module Firmware

Overview

Coded in C language, module firmware includes modified module topology, algorithms in infinite loop in main source code, and functions in the BitzOS source code.
Topology file enables inter-array communication between multiple modules. It describes the number of modules and how they are connected to each other.
In the infinite loop of main source code of the firmware, modules are designed to check whether there is an object within the distance threshold (20cm) every 50 milliseconds. For module1’s case, if an object is detected, then it outputs a serial message “1” to the PC. For module2’s case, if an object is detected, then it sends a message to module 1 to execute a function called CODE_USER_MESSAGE_01, which is defined in the BitzOS source code. This enables module 1 to output a serial message “2” to the PC as well.
While BitzOS is running (i.e. module is connected to PC and powered), whenever it receives a string message from the user, a corresponding function to the string message is executed. However, when an unknown string message is received, in default, BitzOS acknowledges it as an unknown message and outputs an error. Thus, to enable module 1 to receive messages from module 2 and execute the functions successfully, the module firmware was modified to see if an unknown message received is called “CODE_USER_MESSAGE_01”, output a serial message “2” to the PC.

Software Flowchart

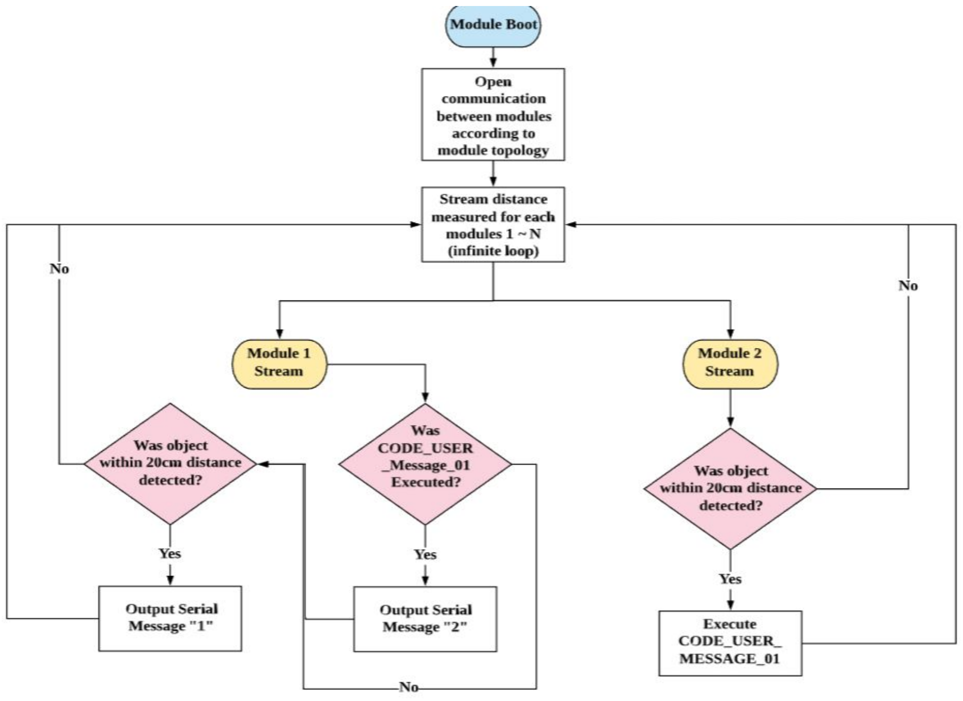

Module Boot
Open communication between modules according to module topology
Stream distance measured for each modules 1 ~ N (infinite loop)
Module 1 Stream
Module 2 Stream
Was object within 20cm distance detected?
Was CODE_USER _Message_01 Executed?
Was object within 20cm distance detected?
No
Yes
Output Serial Message "1"
Output Serial Message "2"
Execute CODE_USER_ MESSAGE_01

Figure 2.1: Firmware Flowchart

BitzOS

BitzOS (BOS) is Hexabitz's own operating system, built on top of FreeRTOS, it is designed to support Hexabitz arrays. Our product uses its functions which are able to be interacted via its Command Line Interface. BOS consists of five main parts: Commands, Messages, Data Streams, APIs, Internal Functions, and Peripheral Drivers.

The BitzOS firmware after modification experienced a software bug which its cause wasn't able to be tracked down as its occurrence was random. Precisely, module 1 delays the execution of CODE_USER_MESSAGE_01, and stacks multiple when module 2 is played while being delayed during IR Theremin operation. Then, after a few seconds (no more than 10 seconds), stacked CODE_USER_MESSAGE_01 is released at instant. At this time, fixing the software bug is included as a future work in this report.

Processing IDE Software

Overview

We used the Processing IDE for creating the computer program that interfaces with the HexaBitz Theremin. Processing has many libraries and was easier to use than creating equivalent C++ code for Windows. One of the biggest advantages was that Processing had sound that simply worked with little effort. The serial communication was also very straightforward. The main catch was in the interface. We had planned to make a text-based menu-driven application, but Processing didn't have any good options to do that easily. For that, we made our own virtual terminal.

Software Flowchart

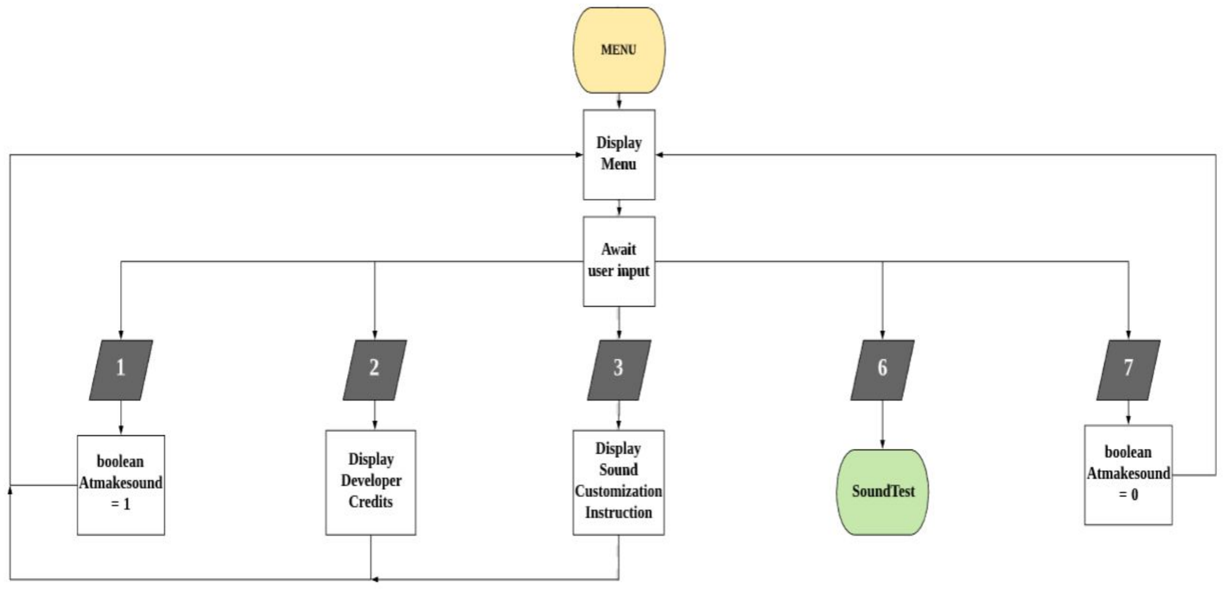


Figure 2.2: Main PC Software Flowchart

Sound Processing

The sound play of the system is fairly straightforward. Sound can be invoked with either the HexaBitz Lidar modules (when in active mode) or by using the Sound Test menu. The big difference is that the Sound Test menu allows keyboard control of sounds. When a particular sound is played again before a previous instance of the same sound is done playing, it will stop the playing instance and start a new instance of the sound from the beginning. However, multiple different sounds will play side-by-side. These are referred to as “replay” and “overlay”, respectively.

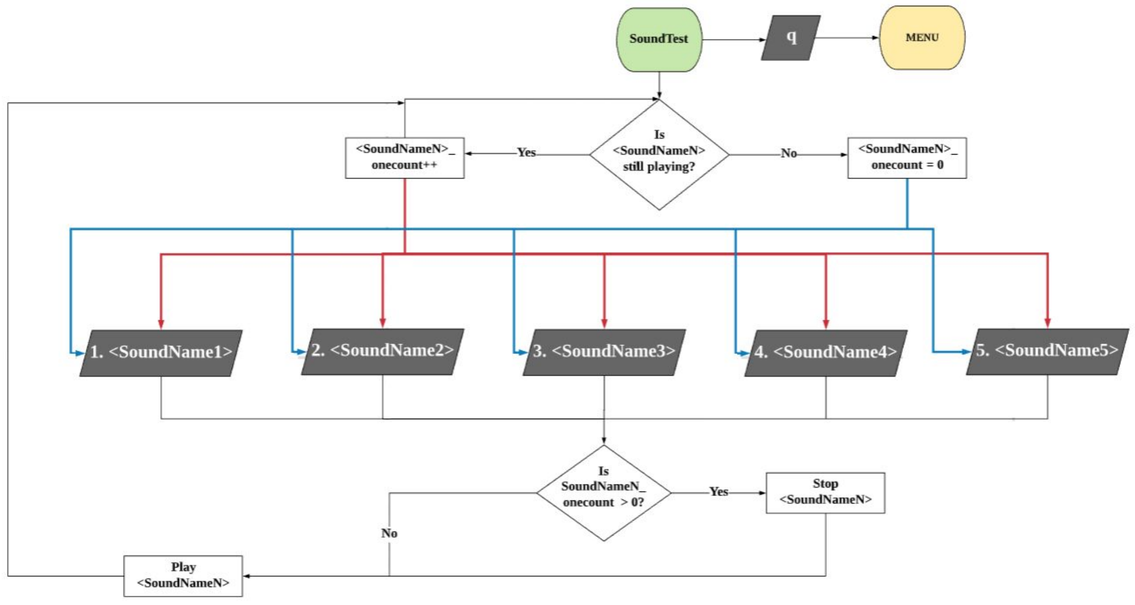


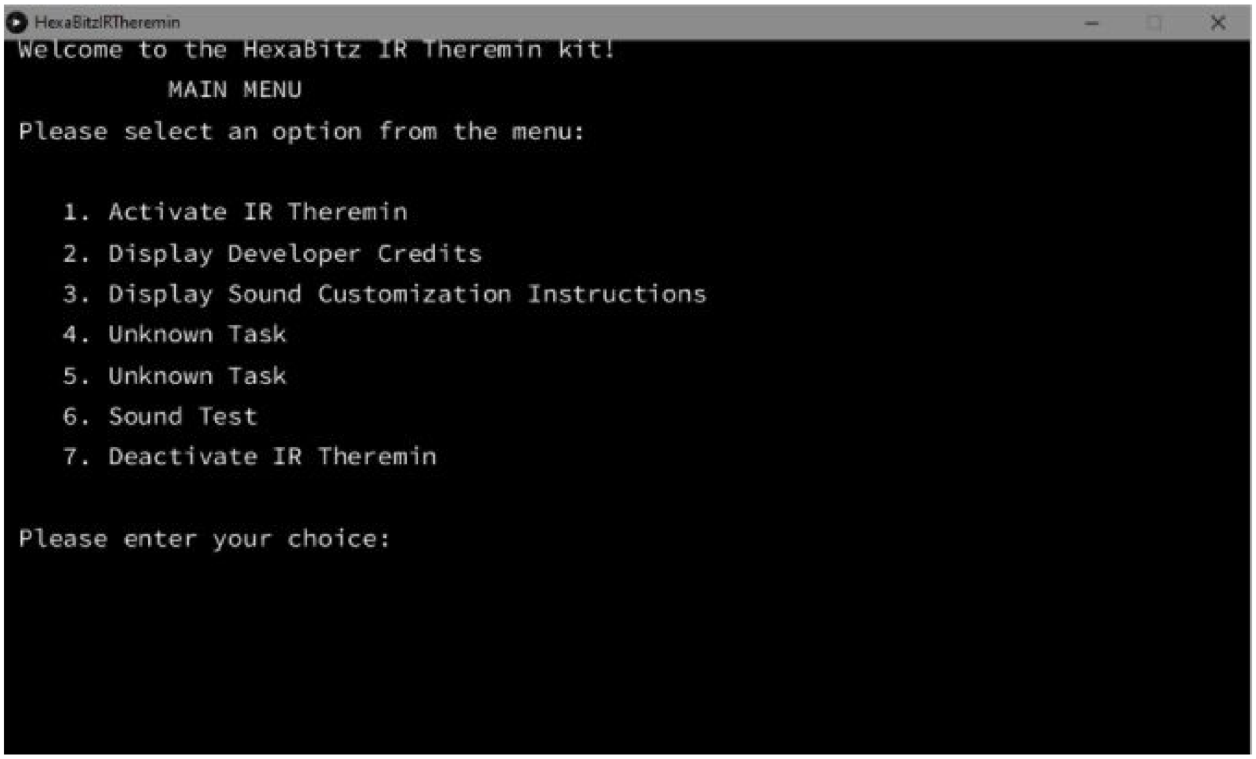
FIgure 2.3: General Sound Flow

Serial Communication

Our program uses the standard serial library from Processing, to communicate with the main Hexabitz module. Processing connects to the selected serial port, to talk to the module. We also made some functions to make it easier to handle some peculiarities of communicating with HexaBitz modules.

CLI Interface

Processing's built-in CLI proved to be insufficient for what our program needed. We needed to be able to write a text-based, menu-driven application, using keyboard input. We also realized that Processing's execution flow was not well-suited for this design style, since it likes to have a main loop method (called draw()), and it can misbehave when blocking functions are used in this method. We applied two solutions, side-by-side, to deal with these problems.
First, we made a new CLI interface, essentially a virtual terminal of our own. This included three main parts: a screen text buffer, a keyboard input buffer, and a line of code to redraw the screen text buffer to our window every time draw() is run. To make use of these, the following methods were created: clearKeyboardBuffer(), conprint(), conprintln(), getCharHidden(), getLine(), keyboardAvailable(), prompt(), and putChar(). They work very much like other methods with similar names. The exception is that putChar() also handles management of the screen buffer, and all text output eventually calls it.
Second, we put the menu system's code in a separate execution thread. This allowed it to use blocking methods, and leave the draw() method running as many times a second as Processing required.



```
Welcome to the HexaBitz IR Theremin kit!
          MAIN MENU
Please select an option from the menu:

   1. Activate IR Theremin
   2. Display Developer Credits
   3. Display Sound Customization Instructions
   4. Unknown Task
   5. Unknown Task
   6. Sound Test
   7. Deactivate IR Theremin

Please enter your choice:
```

Figure 2.4: CLI Interface Main menu

Testing

Testing Before Release

Our product is requested to be released once to the website called hackster.io. Then, as an open source project, project files are downloaded via the internet for further modification, experiment, etc. Thus, we checked if the sound processing algorithm works correctly.

Checklist	Functional Capability (Capable/Incapable)
Does the sound sample match with the assigned modules?	Capable
When users play the same module while the sound assigned to it is playing, does the software produce the sound for the first strike, then restart the equivalent sound when the user plays the module for the second strike? (replay)	Capable
When users play two different modules with a time period in between, and if the second module is struck before the first sound sample fades away, does the software play the first sound until fading, and layer the second sound over it? (Overlay)	Capable

Table 3.1: Testing Checklist

Data Sample Period Testing

Sample periods of the modules are adjusted using a firmware function called Stream_ToF_Memory which takes three inputs as its parameter: period, timeout, buffer.
To find the suitable data sample period, we collected a whole note sound sample and assigned it to a module. Then, using the “replay technique” of Processing IDE software functionality,we repeated playing the sound samples for a duration of sixteenth note ~ whole note (0.25 sec ~ 4 sec) with four sequential times for each type of the note. Out of 10 ms, 50ms, 100ms, 150sm, 200ms of different sample periods, 50ms showed the highest success rate in playing the sound, which can be seen in the figure below.

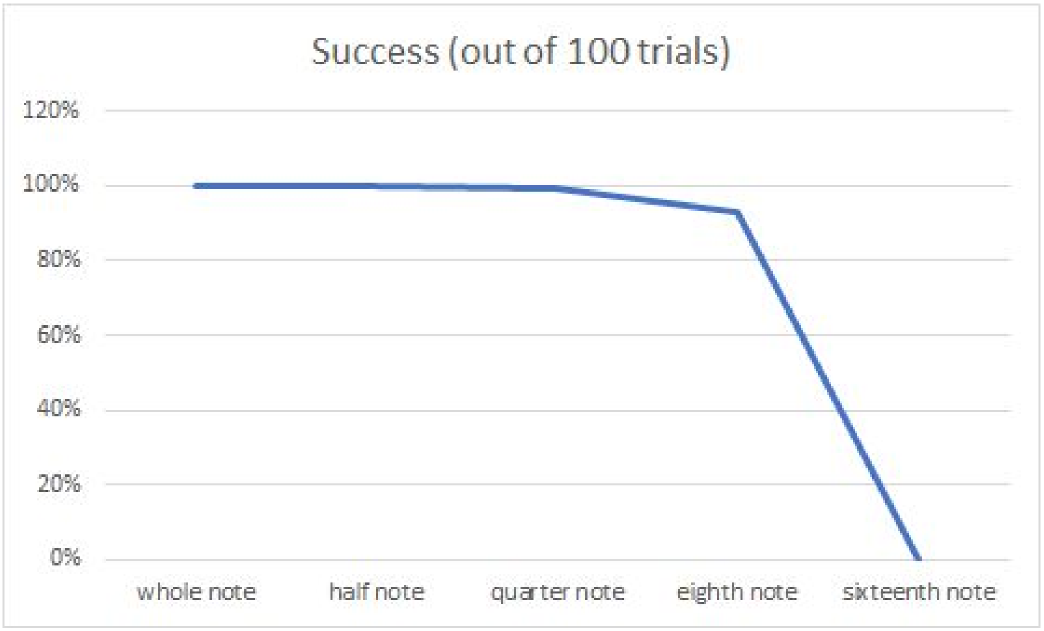


Figure 3.2: Result (Streaming in 50ms period, 20Hz)

Software Distance Threshold Testing

We also experimented to find the best distance threshold; how far away the module would acknowledge the presence of a hand. The reason was that the sensors sampled too slowly to consistently detect anything briefly within their field, so some distance was necessary to make sure that the user's hand would stay in the field long enough to be detected. For various distances, we tried playing one hundred quarter-notes and counted the number of false negatives where the system did not detect the hand. As a minimum distance to test, we chose 5 centimeters, as any closer and it would be hard to avoid hitting the sensors. For the maximum, we chose 20 centimeters, at which point it gets difficult to move your hands past the sensor without accidentally triggering it.
Based on the false negative rate by distance (Figure 3.1) we found that it was hard to reliably play the module at 5 cm, but the false negative rate was low and didn't vary much for 10 to 20 cm.
Unfortunately, these tests could not reflect false positives resulting from the sensor detecting a hand moving past it on the way to something else; this depends largely on the physical arrangement of the sensors relative to the user, which we intended to be user-determined. Therefore, these tests cannot be fully conclusive, though they do indicate that 10 cm is a reasonable default setting.

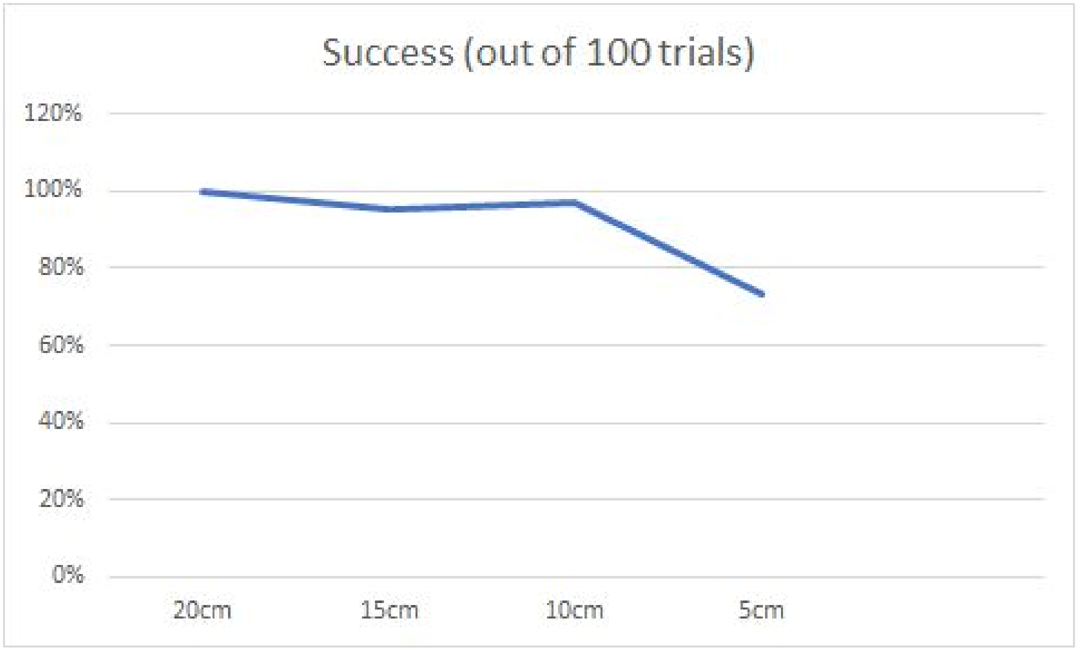


Figure 3.3: False Negative Rate By Distance

Conclusion and Future Work

IR Theremin design enabled modules to be played as fast upto eighth notes in terms of musical scale. Basic sound processing algorithms, replay and overlay has been successfully implemented to play sounds accordingly. Graphical 3D model of module holder was created to further support modules to be physically set up in an organized way. Through this project, we learned methods to interpret and modify operating system & firmware, create CLI interface, enable serial communication with STM32 MCU and PC, define module topology to enable communication between multiple MCUs, and recreate musical techniques such as replay and overlay in terms of IR Theremin's functionality.

From this point, next steps and future's tasks would be to fix the firmware bug and reorganize the code. An idea to create a synthesizer to allow special effects like vibrato using a non-vibrato, or steady/flat sound sample was brought as the idea for the future as well. Finally, implementing visualization of musical performance with real-time generated graphs (e.g. waterfall and waveform display) would be a good way to enhance the quality of the software design as well.

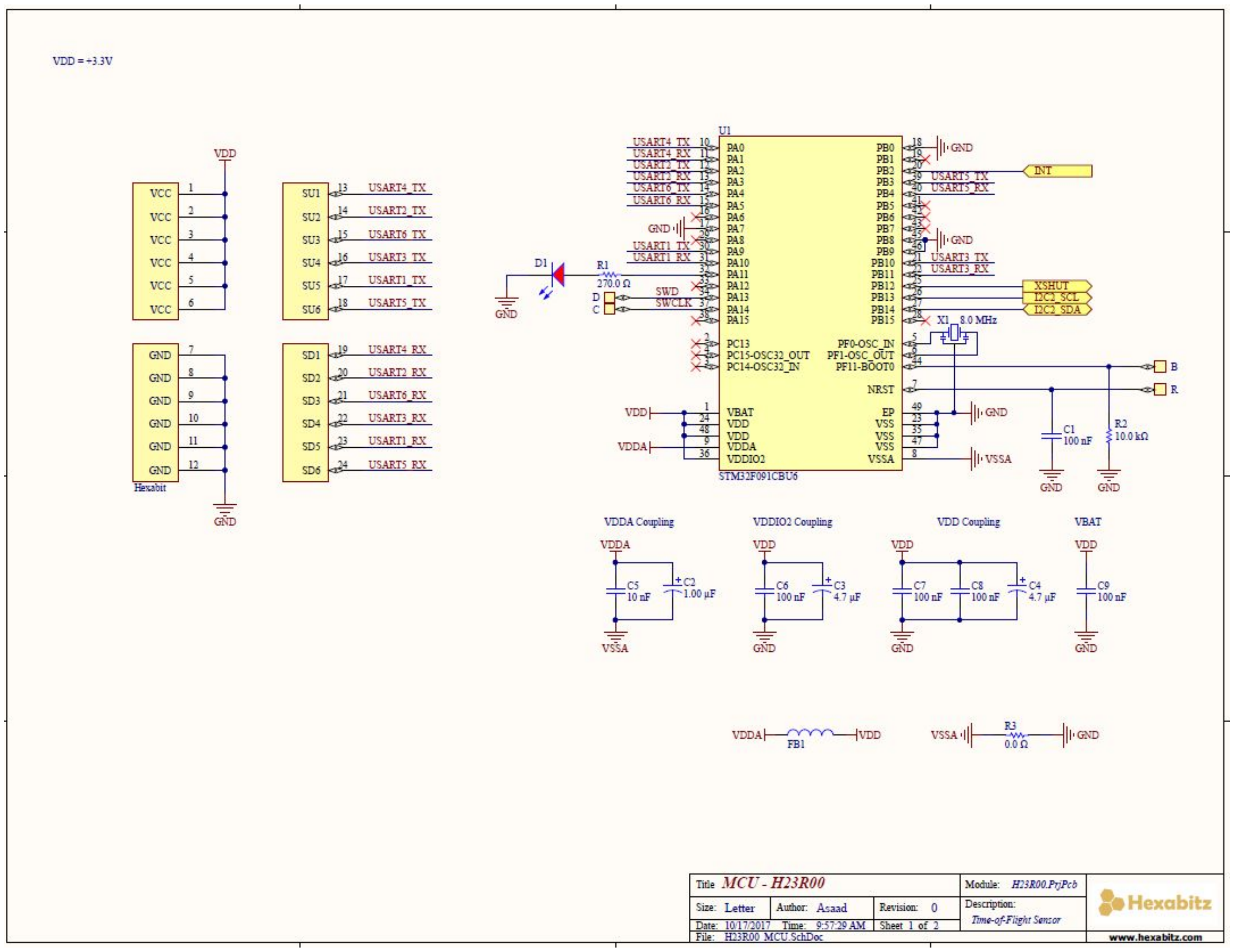
Acknowledgements

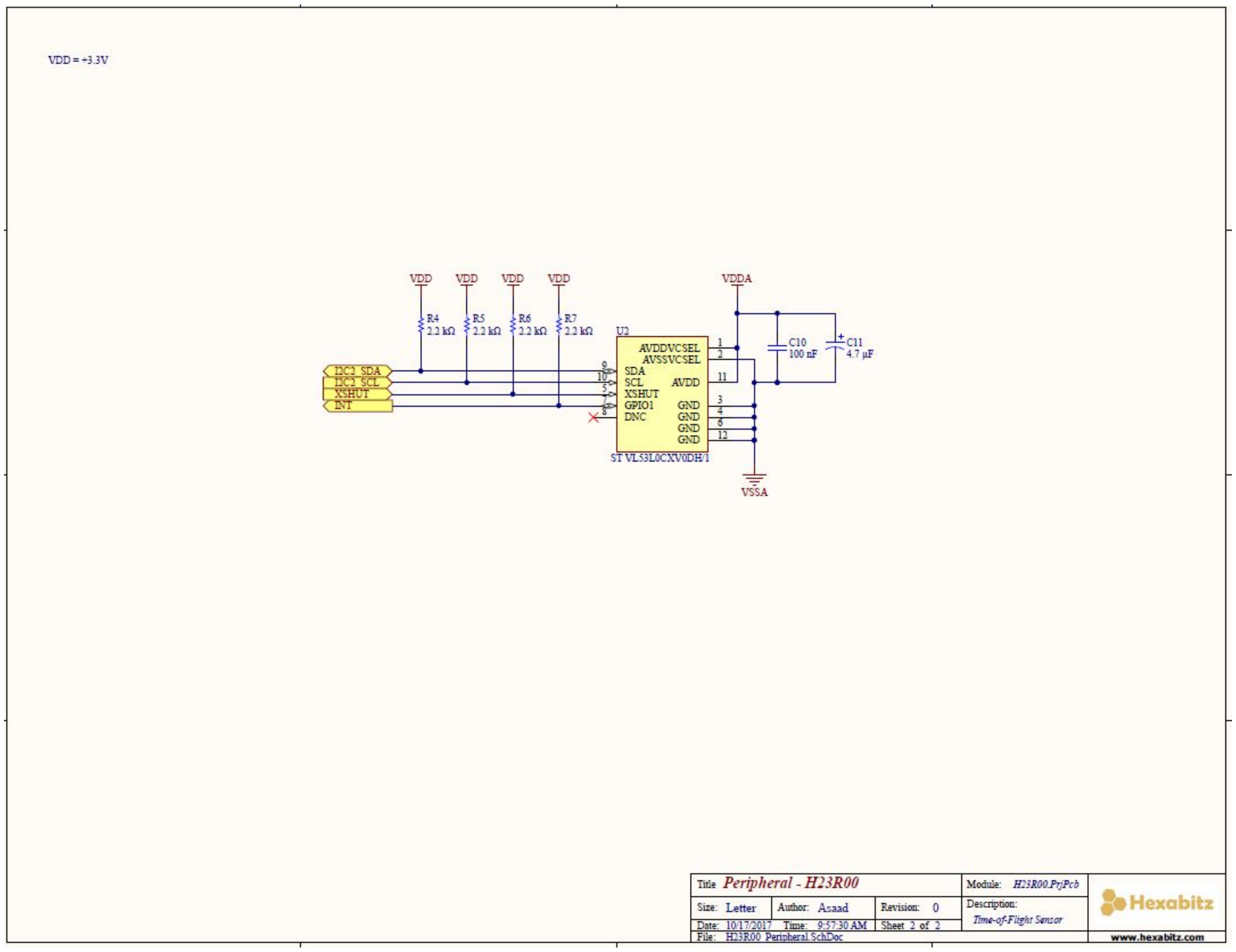
We would like to thank our academic advisors Elaine Reeves and Joseph Decuir, for giving us many valuable insights, feedback, and hours to help make this project a success.
We would also like to thank our industry advisor Asaad Kaadan, for creating the Hexabitz platform, for the inspiration for this project, and for some technical help.

Appendix

Datasheets

Module Schematic




VDD = +3.3V
VDD
VDD
VDD
VDD
R4
2.2 kΩ
R5
2.2 kΩ
R6
2.2 kΩ
R7
2.2 kΩ
VDDA
U2
AVDDVCSEL
AVSSVCSEL
SDA
SCL
XSHUT
GPIO1
DNC
AVDD
GND
GND
GND
GND
1
2
9
10
5
7
8
11
3
4
6
12
C10
100 nF
C11
4.7 µF
I2C2_SDA
I2C2_SCL
XSHUT
INT
ST VL53L0CXV0DH/1
VSSA
Title Peripheral - H23R00
Module: H23R00.PrjPcb
Size: Letter
Author: Asaad
Revision: 0
Description:
Time-of-Flight Sensor
Date: 10/17/2017
Time: 9:57:30 AM
Sheet 2 of 2
File: H23R00_Peripheral.SchDoc
Hexabitz
www.hexabitz.com

Technical Specifications

Top (1:1)

Bottom (1:1)

- Six array ports and six power ports (+3.3V and GND).
- Access to 6xUART, 2xI^{2}C, SWD, BOOT0, RESET.
- ST VL53L0X Time-of-Flight (ToF) ranging and gesture detection sensor:
  - 940nm Laser VCSEL.
  - Eye safe Class I laser.
  - Measures absolute range up to 2m.
  - Reported range is independent of the target reflectance.
- STM32F091CBU6 MCU.
- 8MHz external oscillator.

Available colors: 🟩

Bill of Materials

Product Name: Hexabitz IR Theremin Product Description: Compilable sets of hardware and software to perform as IR Theremin Created Date: 08/28/2020 Created By: Ryan Cho Comments: Hardware components for assembly follows the list below					
Product	Company	Link	Quantity	Price	Description
1D Lidar IR Sensor (H08R6x)	Hexabitz	H08R6x	2	$12.60x2 = $25.20	IR Sensor
4-Pin USB-Serial Prototype Cable	Hexabitz	Cable	1	$7.99	PC to module 1 connection
Jumper Wires	Elegoo	Wire	1	$6.98	Module to module connection, inserted at the header pins
Header Pins	MCIGICM	Pins	1	$4.59	Soldered each on to 3v3, GND, and ports of modules
Total: $44.76 (Excludes Tax and Shipping Price)					

Firmware Installation/Repair Guide

Installing a new .hex file to a module is a fairly simple process. It requires:

- Module firmware compiled into .hex format
- A terminal software (we used PuTTY)
- The Hexabitz USB Serial Cable
- The STM Flash Loader Demonstrator program

1. Physically connect the module to the computer as shown in the Figure 4.1.
2. Using the terminal software, establish the serial connection with the parameters:
    a. Baudrate = 921600
    b. Display Format = ANSI
    c. Parity = None
    d. Data Bits = 1
    e. Stop Bits = 1
    f. Flow Control = None
3. Press enter. A message introducing BitzOS should appear.
4. Type "update" and then press enter. A message should appear saying that the module has entered bootloader mode. If this does not happen, repeat this step. Press no further keys in the terminal window. (*If this step cannot be executed, refer to **4-a:** firmware code damaged*)
    a. If the module OS cannot be excessed through terminal (e.g. Putty), Apply 3.3V on module B (Boot0) pad while power cycling. Easiest way to achieve this is to connect a female-to-male jumper wire to one of power ports (edge ports) top connector, and then touch the B pad while power cycling the MCU.
    b. Skip to step **6** after power cycling.
5. Close the terminal window.
6. Open the STM Flashloader Demonstrator.
7. On the first page, set the COM port to the one the module is plugged into. Leave all other fields default.
8. The second page should show a green traffic light. Click next.
9. Leave the third page's setting default.
10. On the fourth page, select "download to device," browse for the .hex file you want to use, and leave everything else default.
11. Wait for the program to complete.
12. Disconnect and reconnect the module to power. The process is now complete.

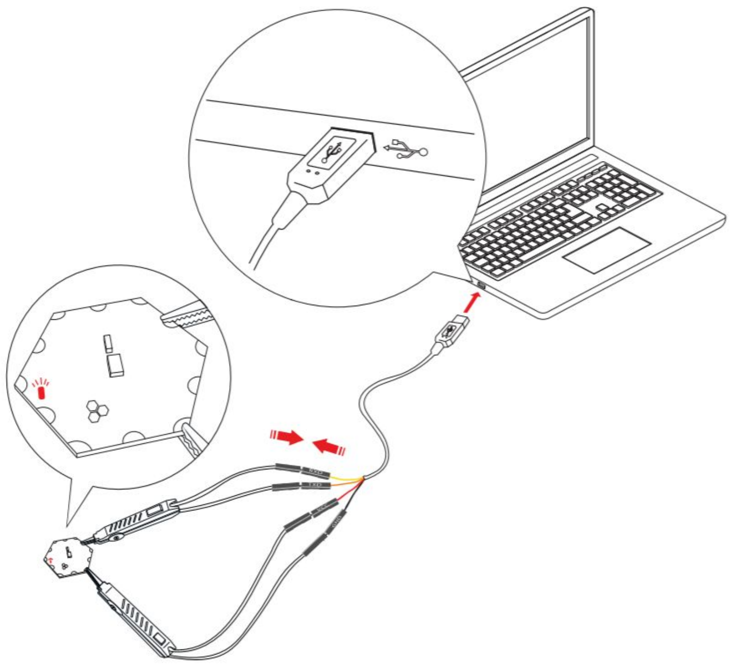


Figure 4.1: Module Connection to PC

Firmware Code

Topology_1.h

```
/*
    BitzOS (BOS) V0.0.0 - Copyright (C) 2016 Hexabitz
    All rights reserved

    File Name     : topology_1.h
    Description   : Array topology definition.
*/

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __topology_1_H
#define __topology_1_H
#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f0xx_hal.h"


#define __N    2  // Number of array modules

// Array modules
#define _mod1          1<<3
#define _mod2          2<<3


// Topology
static uint16_t array[__N][7] = {
          { _H08R6, 0, 0, 0, _mod2|P1, 0, 0},   // Module 1: uses P4 to connect to module 2
          { _H08R6, _mod1|P4, 0, 0, 0, 0, 0},   // Module 2: Uses P1 to connect to module 1
};

// Configurations for duplex serial ports
#if ( _module == 1 )
          #define          H08R6 1
          #define          _P1pol_normal          1
          #define          _P2pol_normal          1
          #define          _P3pol_normal          1
          #define          _P4pol_normal          1
          #define          _P5pol_normal          1
          #define          _P6pol_normal          1
#endif
#if ( _module == 2 )
```

```
	#define		H08R6 1
	#define		_P1pol_reversed	1 //RXD of module 1 connects to TXD of module 
2.
	#define		_P2pol_normal	1
	#define		_P3pol_normal	1
	#define		_P4pol_normal	1
	#define		_P5pol_normal	1
	#define		_P6pol_normal	1
#endif

#ifdef __cplusplus
}
#endif
#endif /*__ topology_1_H */


/************************ (C) COPYRIGHT HEXABITZ *****END OF FILE****/
```

main.c

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  *
  * COPYRIGHT(c) 2015 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND 
CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, 
THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR 
PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
```

```
BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/*
MODIFIED by Hexabitz for BitzOS (BOS) V0.2.2 - Copyright (C) 2017-2020 Hexabitz
   All rights reserved
*/

/* Includes ------------------------------------------------------------------*/
#include "BOS.h"
#include "H08R6.h"


/* Private variables ---------------------------------------------------------*/
float sensor = 0.0f;
uint32_t period = 100;
/* Private function prototypes -----------------------------------------------*/
bool DetectedPing(float distance);


/* Main functions ------------------------------------------------------------*/

int main(void)
{


  /* MCU Configuration----------------------------------------------------------*/

  /* Reset all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all user peripherals */

          /* Initialize BitzOS */
          BOS_Init();

  /* Call init function for freertos objects (in freertos.c) */
  MX_FREERTOS_Init();

  /* Start scheduler */
  osKernelStart();

  /* We should never get here as control is now taken by the scheduler */

  /* Infinite loop */
  while (1)
  {


  }


}

/*-----------------------------------------------------------*/

/* User Task */
void UserTask(void * argument)
{
          //Start stream data to each other among modules
          #if _module == 1
                    // Set units to mm
                    SetRangeUnit(UNIT_MEASUREMENT_MM);
                    // Stream to memory
                    Stream_ToF_Memory(50, portMAX_DELAY, &sensor);
          #endif
          #if _module == 2
                    // Set units to cm
                    SetRangeUnit(UNIT_MEASUREMENT_MM);
                    // Stream to memory
                    Stream_ToF_Memory(50, portMAX_DELAY, &sensor);
          #endif

  /* Infinite loop */
  for(;;)
  {
 //For the case of module 1, if DetectedPing(sensor) is true, send a serial message to PC.
"1\r\n\"
```

```
        #if _module == 1 //1st instrument
                if (DetectedPing(sensor))
                {
                sprintf( ( char * ) pcUserMessage, "1\r\n");
                writePxMutex(PcPort, pcUserMessage, strlen(pcUserMessage), cmd50ms,
HAL_MAX_DELAY);
                responseStatus = BOS_OK;
                IND_blink(1);

                }
                #endif

//For the case of module 1, if DetectedPing(sensor) is true, send a serial message to PC.
"2\r\n\"
//Since module 2 cannot send a message directly to PC without passing module 1,
//Program uses CODE_USER_MESSAGE_01 to tell module 1 to print out given Message
        #if _module == 2 //2nd instrument
                if (DetectedPing(sensor))
                {
                /*sprintf( ( char * ) pcUserMessage, "2\r\n");
                writePxMutex(PcPort, pcUserMessage, strlen(pcUserMessage), cmd50ms,
HAL_MAX_DELAY);
                responseStatus = BOS_OK;
                IND_blink(1);*/
                SendMessageToModule(1, CODE_USER_MESSAGE_01, 5);
                IND_blink(1);
                //Delay_ms(1);
                }
                #endif
        }
}

//Method to find out whether an object is within distance for each of the modules to trigger
// if conditions at for(;;)
bool DetectedPing(float distance)
{
                static float state;
        if (distance < 200.0f && state != 1)
        {
                state = 1;      // Detected an object
                return true;
        }
        else if (distance >= 200.0f && state == 1) {
                state = 2;      // The object cleared
        }
        return false;
}
```

```
/*-----------------------------------------------------------*/

/************************ (C) COPYRIGHT HEXABITZ *****END OF FILE****/
```

Bos.c (Line 2861 ~ 2885)

```
/*-----------------------------------------------------------*/

/* --- User message parser.
This function is declared as __weak to be overwritten by other implementations in the user
file.
CODE_USER_MESSAGE_01 is used to send a message from module 2 to module 1
If adding more modules, suggestion is to make a case for CODE_USER_MESSAGE_02
*/
__weak BOS_Status User_MessagingParser(uint16_t code, uint8_t port, uint8_t src, uint8_t
dst, uint8_t shift)

{
	BOS_Status result = BOS_ERR_UnknownMessage;
	switch (code){
		case CODE_USER_MESSAGE_01 :
		    sprintf( ( char * ) pcUserMessage, "2 \r\n");
		    writePxMutex(PcPort, pcUserMessage, strlen(pcUserMessage), cmd50ms,
HAL_MAX_DELAY);
		     responseStatus = BOS_OK;
		     //IND_blink(1);
		    break;
	}

	return result;
}

/*-----------------------------------------------------------*/
```

Processing IDE Code

HexaBitzIRTheramin.pde

```
/* HEXABITZ IR THEREMIN TEAM: Ryan Cho, Geoffrey Powell-Isom, Scott Bryar, Patricia
Fang
*  Date: 2020-08-16
*  Read comments at setup() to run the software without COM connections to modules
*
*  Instruction:
*  If using COM connections to module, use STL Flash software to install the following files to
*  each of the corresponding modules.
*  FilePath: H08R6 Firmware (ver2)\MDK-ARM\Objects\H08R6-Module 1
*  FilePath: H08R6 Firmware (ver2)\MDK-ARM\Objects\H08R6-Module 2
*
*  Use the instructions on hackster.io and Hexabitz tutorials
*  to enable a module connection for more than 2 modules.
*
*/

import processing.serial.*; //import the serial library
import processing.sound.*;  //import the sound  library
import ddf.minim.*; //import the Minim sound library, because it's reported less buggy

Serial myPort;  // Create object from Serial class
String portName = "COM12"; //change this if you use different COM port
String myString; //variable used to read message from serial port
char module; //a character received from module when a module is struck (different char for
every modules)
int nl = 10; //delimiter to read indents between read serial messages


//Sound setup (Another option to play sound)
Minim minim;

//Sound Files Variables (Activate IR Theremin)
SoundFile[] file;
int first_onecount = 0;
int second_onecount = 0;
boolean atmakesound;

//Sound Files Variables (soundTest)
SoundFile[] file1;
```

```
int tomTom_onecount = 0;
int hihat_onecount = 0;
int timpaniPlay_onecount = 0;
int cymballPlay_onecount = 0;
int voicePlay_onecount = 0;
boolean tomTomPlay;
boolean hiHatPlay;
boolean timpaniPlay;
boolean cymbalPlay;
boolean voicePlay;
boolean atSoundTest;


int instrumentLimit = 5; //not limiting this can allow a slowdown bug in the audio that loops for
a very long time, maybe infinitely

//CLI setup
PFont f;
StringBuilder screenBuffer = new StringBuilder();
byte lineLength;
byte lineCount;


StringBuilder keyboardInputBuffer = new StringBuilder();
String            inputBackup;
boolean inputReady  = false;
boolean newKeyPress = false;

void setup() {
  size(900, 600);
  background(0);

  // Create the font
  printArray(PFont.list());
  f = createFont("SourceCodePro-Regular.ttf", 18);
  textFont(f);
  textAlign(TOP, LEFT);


  //Engage communication with module
  myPort = new Serial(this, portName,921600);        //comment out this line and a line under to
run without serial communication (i.e. COM not connected to module)
  serialWrite("\r");                                 //comment out this line and a line above to run without
serial communication (i.e. COM not connected to module)

  //file(.wav) array for Activate IR Theremin
  //The size of the file is equal to the number of modules set up. (for this code, 2 modules is
used)
  file = new SoundFile[2];
  file[0] = new SoundFile(this, "CrashCymbal.wav");//Change the name of the
"CrashCymbal.wav" in order to swap sound files
  file[1] = new SoundFile(this, "TomTom.wav");


  //file(.wav) array for soundTest
  file1 = new SoundFile[5];
  file1[0] = new SoundFile(this, "TomTom.wav");
  file1[1] = new SoundFile(this, "HiHat.wav");
  file1[2] = new SoundFile(this, "timpani.wav");
  file1[3] = new SoundFile(this, "CrashCymbal.wav");
  file1[4] = new SoundFile(this, "voice.wav");



  NewMain main01 = new NewMain();
  Thread t1 = new Thread(main01);
  t1.start();
}

void draw() {
  background(0);

  // Set the left and top margin
  int margin = 3;
  translate(margin*4, margin*4);

        fill(255);

        // Draw the letter to the screen
        text(screenBuffer.toString(), 0, 0);

        /* At 'start' if Activate IR Theremin(is selected) make infinite loop turn on (turned off by
pressing 7 at the main menu, which is Deactive IR Theremin)
        *  Functionalities: Read string messages separated by '\n' from the first module
connection via COM port
        *  Then, retrieve the first character of the string message
        *  Play the corresponding sound from 'file', which is a wav array
        *  Sound Processing: (1) Capable of overlapping multiple sound
        *                    (2) Capable of replaying the sound
        */
        if(atmakesound == true){
                while(myPort.available() >0){
                //serial
                myString = myPort.readStringUntil('\n');
```

```
                    if(myString == null){
                    }
                    else{
                    module = myString.charAt(0);
                    println(module);
                    }
                    //sound

                    if(file[0].isPlaying() == false){
                    first_onecount = 0;
                    }
                    else{
                    first_onecount++;
                    }

                    if(file[1].isPlaying() == false){
                    second_onecount = 0;
                    }
                    else{
                    second_onecount++;
                    }


                    playMusic(module);
                    delay(10);

          }

          }
}

///////NEW MAIN METHOD TO USE NORMAL EXECUTION FLOW///////
//Do not change this subclass. Instead, edit or implement the NewMainFunction();.
//From there, you can write a program in normal Java style, like the normal Java
//main() method;
class NewMain implements Runnable{ public void run(){
  NewMainFunction();
}}
///////END OF MAIN CODE SECTION///////

// conprintln prints out string messages on the CLI window
void conprintln(){ conprint("\n");}
void conprintln(String text){ conprint(text + '\n');}
void conprint(String text){
  if(text == null) return;
  for(int i = 0; i < text.length(); ++i) putChar(text.charAt(i));
}
```

```
// interacts with screenBuffer to manage length of strings and lines
void putChar(char c){
  if(c != '\n' && lineLength == 80){
          screenBuffer.append('\n');
          lineLength = 0;
          ++lineCount;
  }
  if(c == '\n'){
          lineLength = 0;
          ++lineCount;
  } else {
          ++lineLength;
  }
  screenBuffer.append(c);

  if(lineCount > 19){
          while(screenBuffer.toString().charAt(0) != '\n') screenBuffer.delete(0, 1);
          screenBuffer.delete(0, 1);
          --lineCount;
  }
}

// Used to receive message input from the keyboard
String getLine(){
  //while(keyboardInputBuffer.toString().charAt(keyboardInputBuffer.length() - 1) != '\n');
  //while(inputBackup == null);
  StringBuilder input = new StringBuilder();
  while(keyboardInputBuffer.length() == 0 ||
keyboardInputBuffer.toString().charAt(keyboardInputBuffer.length() - 1) != '\n'){
          if(keyboardInputBuffer.length() > 0){
          input.append(keyboardInputBuffer.charAt(0));
          putChar(keyboardInputBuffer.charAt(0));
          keyboardInputBuffer.delete(0,1);
          }
          delay(1);
  }
  keyboardInputBuffer.delete(0,1);
  putChar('\n');
  //String input = inputBackup;
  inputReady = false;
  return input.toString();
  //return "input";
}

//receives the message from userPrompt and returns it as a string
String prompt(String userPrompt){
  conprint(userPrompt);
  String input = getLine();
  return input;
}

//Hide the characters being typed
char getCharHidden(){
  while(keyboardInputBuffer.length() == 0) delay(1);
  char input = keyboardInputBuffer.charAt(0);
  keyboardInputBuffer.delete(0,1);
  return input;
}

//Learn if the keyboardInputBuffer has content
boolean keyboardAvailable(){
  return keyboardInputBuffer.length() > 0;
}

//clear
void clearKeyboardBuffer(){
  keyboardInputBuffer.delete(0, keyboardInputBuffer.length());
}


/* At 'start' if soundTest(is selected) make infinite loop turn on (turned off by pressing q)
        *  Functionalities: Read string messages separated by '\n' from the first module
connection via COM port
        *  Then, retrieve the first character of the string message
        *  Play the corresponding sound from 'file', which is a wav array
        *  interacts with keyReleased to manage variable [Syntax: Soundname_onecount]
        *  Sound Processing: (1) Capable of overlapping multiple sound
        *                    (2) Capable of replaying of each sound (i.e. Sound called 'A' is played
again while 'A' is playing. In this case, stop playing the previous sound 'A')
        *
        */
void keyPressed() {

  keyboardInputBuffer.append(key);
  if(atSoundTest == true){
        if (key == '1') {
        tomTom_onecount++;
        if(file1[0].isPlaying() && tomTom_onecount == 1){
        file1[0].stop();
        }
        if(tomTom_onecount == 1){
        file1[0].play();
        }
        }

        else if (key == '2') {
        hihat_onecount++;
        if(file1[1].isPlaying() && hihat_onecount == 1){
        file1[1].stop();

        }
        if(hihat_onecount == 1){
        file1[1].play();
        }
        }

        else if (key == '3') {
        timpaniPlay_onecount++;
        if(file1[2].isPlaying() && timpaniPlay_onecount == 1){
        file1[2].stop();

        }
        if(timpaniPlay_onecount == 1){
        file1[2].play();
        }
        }
        else if (key == '4') {
        cymballPlay_onecount++;
        if(file1[3].isPlaying() && cymballPlay_onecount == 1){
        file1[3].stop();

        }
        if(cymballPlay_onecount == 1){
        file1[3].play();
        }
        }

        else if (key == '5') {
        voicePlay_onecount++;
        if(file1[4].isPlaying() && voicePlay_onecount == 1){
        file1[4].stop();

        }
        if(voicePlay_onecount == 1){
        file1[4].play();
        }
        }

        }
```

```
}

// Used for soundTest,
// If key is released, turn onecount variable to 0
void keyReleased() {
  if(atSoundTest == true){
          if(key == '1'){
          tomTom_onecount=0;
          }
          else if(key == '2'){
          hihat_onecount=0;
          }
          else if(key == '3'){
          timpaniPlay_onecount=0;
          }
          else if(key == '4'){
          cymballPlay_onecount=0;
          }
          else if(key == '5'){
          voicePlay_onecount=0;
          }
  }
}

void serialWrite(String content){
  int strLength = content.length();
  for (int i = 0; i<strLength; i++)
  {
          myPort.write(content.charAt(i));
  }
}

void playMusic(char module){
          if (module == '1') {
          first_onecount++;
          if(file[0].isPlaying() && first_onecount >= 1){
          file[0].stop();
          first_onecount = 1;
          }
          if(first_onecount == 1){
          file[0].play();
          }
          }

          else if (module == '2') {
            second_onecount++;
            if(file[1].isPlaying() && second_onecount >= 1){
            file[1].stop();
            second_onecount = 1;
            }
            if(second_onecount == 1){
            file[1].play();
            }
            }

}

//This method simply tells the respective module the new min and max distances, or prints an
error, if it doesn't know the module.
void setNewDistances(int minimun, int maximum, int mod){
  switch(module){
            default: conprintln("ERROR: Unknown module # " + module + " specified.");
            case 1:  //send new min command to theremin module 1
            //send new max command to theremin module 1
            case 2:  //send new min command to theramin module 2
            //send new max command to theremin module 2
  }
}

//This asks the module for its min distance, and returns that value to the calling code
short getMinDistance(short newModule){
  short minDistance = 0;
  //get minDistance from module
  return minDistance;
}

//This asks the module for its max distance, and returns that value to the calling code
short getMaxDistance(short newModule){
  short maxDistance = 0;
  //get maxDistance from module
  return maxDistance;
}
```

start.pde

```
void NewMainFunction(){
            conprintln("Welcome to the HexaBitz IR Theremin kit!");
            boolean running = true;
            while(running){
            conprintln("                 MAIN MENU");
            conprintln("Please select an option from the menu:");
            conprintln();
            conprintln("   1. Activate IR Theremin");
            conprintln("   2. Display Developer Credits");
            conprintln("   3. Display Sound Customization Instructions");
            conprintln("   4. Distance Calibrator");
            conprintln("   5. Sound Test");
            conprintln("   6. Deactivate IR Theremin");
            conprintln();
            String input = prompt("Please enter your choice: " );

            switch(input){
            default: break;
            case "1": atmakesound = true; break;
            case "2": credits(); break;
            case "3": soundInstructions(); break;
            case "4": calibrator(); break;
            case "5": soundTest(); break;
            case "6": atmakesound = false; break;
            }
  }
}

void credits(){
  conprintln("HexaBitz IR Theremin Kit was brought to you by:\n"
  + "The HexaBitz IR Theremin Kit Team!\n"
  + "   ** \"Ryan\" Young Beum Cho\n"
  + "   ** Scott Bryar\n"
  + "   ** Patricia Fang\n"
  + "   ** Geoffrey Powell-Isom");
  conprintln();
  prompt("Press enter, to continue...");
}

void soundInstructions(){
  conprintln("HexaBitz IR Theremin Kit sound customization instructions:\n"
  + "The HexaBitz IR Theremin Kit's sounds can be customized by changing the sound files in
the HexaBitz IR Theremin Kit source folder.");
  conprintln();
  prompt("Press enter, to continue...");
}
```

soundTest.pde

```
void soundTest(){
            boolean running = true;
            atSoundTest = true;

            conprintln("                SOUND TEST");
```

```
        conprintln("Press buttons for sounds you want to play from the menu, or \"q\" to quit.");
        conprintln();
        conprintln("   1. TomTom");
        conprintln("   2. HiHat");
        conprintln("   3. Timpani");
        conprintln("   4. Cymbal");
        conprintln("   5. voice????");
        conprintln("   q. quit");
        conprintln();
        conprintln("Press the button of your choice!");

        while(running){
        switch(getCharHidden()){
        default: break;
        case '1':  break;
        case '2':  break;
        case '3': break;
        case '4': break;
        case '5':  break;
        case 'q': case 'Q': running = false; atSoundTest=false; break;
        }
        clearKeyboardBuffer();
        }

}
```

References

Module Info:
https://hexabitz.com/product/1d-lidar-ir-sensor-h08r6x/

Module Firmware Update Guide:
https://hexabitz.com/docs/how-to/update-module-firmware/

STM Flashloader Demonstrator:
https://www.st.com/en/development-tools/flasher-stm32.html

Module Array Communication:
https://hexabitz.com/docs/how-to/make-a-pre-built-array-topology-file/

Module User Defined Message:
https://hexabitz.com/docs/code-overview-2/user-defined-messages/

Overview of Hexabitz Software Architecture:
https://hexabitz.com/docs/what-is-hexabitz/overview-of-hexabitz-software-architecture/

Inspired by (Air Piano & Theremin):
https://www.hackster.io/mahmoud-mardnly/air-piano-59d375
https://www.carolinaeyck.com/theremin