

THEREMIN GLOVE

Digitally Controlled Analog
Synthesizer with a Wireless Glove

Project Documentation

Johns Hopkins Whiting School of
Engineering

EN.525.743 Fall 2019

Tyler Huddleston

TABLE OF CONTENTS

1	PROJECT DESCRIPTION	1
1.1	Overview	1
1.2	Capabilities	2
1.2.1	Controls in one hand	2
1.2.2	Additional Features (vs traditional theremin)	3
1.3	Limitations	4
2	FUNCTIONAL DESCRIPTION	5
2.1	System Block Diagram	5
2.2	Major Components	5
2.2.1	The Glove	5
2.2.2	The Synthesizer	6
2.3	Microcontroller Uses	7
2.4	VCO Circuit Topology	7
2.5	VCF Circuit Topology	8
2.6	VCA Circuit Topology	9
2.7	A Note on Hardware Capability	9
3	INTERFACE DESCRIPTIONS	10

3.1	Microcontroller Interfaces	10
3.2	IO Requirements.....	10
3.3	Power Requirements	11
3.3.1	Wireless Processor Power Estimate	11
3.3.2	Synthesizer Power Estimate.....	12

4 MATERIAL AND RESOURCE REQUIREMENTS 12

4.1	Development Tools.....	12
4.2	Test Equipment	12
4.3	Build Equipment	13
4.4	Knowledge Base	13
4.5	Part Specifications	14
4.6	Components.....	14
4.7	WiProC PCB	16
4.8	Synthesizer PCB.....	16
4.9	PCB Production	16

5 DEVELOPMENT PLAN AND SCHEDULE 17

5.1	Approach	17
5.2	Order of Development	17
5.3	Schedule.....	18
5.4	Hour Tracking	19
5.5	Risk	20
5.6	Risk Mitigation	20

6	RESULTS.....	20
6.1	Wireless Processor Boards	20
6.1.1	Verification procedure and results	21
6.1.2	Communication.....	21
6.2	Analog Synthesizer Boards	22
6.2.1	Verification procedure and results	22
6.2.2	Waveforms.....	22
6.2.3	Breadboarded filters.....	24
7	ASSEMBLY INSTRUCTIONS.....	24
7.1	Components.....	24
7.2	Connections.....	25
7.3	Using the device	25
7.4	Modifying the design	25
8	DOCUMENTATION.....	25
8.1	github Configuration Management	25
9	REFERENCES.....	26
9.1	Schematic Re-Use.....	26
9.2	Design Topologies and Theory	26
9.3	General	26
	APPENDIX I: PROGRESS.....	26

APPENDIX II: LESSONS LEARNED 30

XBee3 SPI30

UART Interface for Debugging.....30

In System Programming Header.....30

1 PROJECT DESCRIPTION



(speaker not included)

1.1 OVERVIEW

The Theremin Glove is a theremin inspired electronic music instrument. Like the theremin, it is played with hand movements and gestures in the air to generate audio waveforms. Unlike the theremin, this instrument is played with one hand instead of two.

A similar device called the Mi.Mu glove (<https://mimugloves.com/>) was recently introduced. This glove uses similar sensors and serves as a controller to software running on a PC. While the Mi.Mu glove enables a wider range of controls and instruments, the Theremin Glove system is standalone and generates analog audio waveforms.

This project involves several aspects of electrical engineering:

- Analog circuit design
- Digital circuit design

- *Firmware design (C programming for AVR based microcontroller)*
- *PCB design and construction*

And of course, making the design work through verification test and debugging.

Changes from the proposed design at CDR to the project demonstration are noted throughout this document.

1.2 CAPABILITIES

The Theremin Glove will emulate a traditional theremin's sound and characteristics, but with the advantage of an underlying digital system for control and processing. This allows for additional features and modifications via firmware updates rather than hardware updates.

1.2.1 Controls in one hand

The proposed glove controls and the actual controls developed for the project demonstration are compared in the following table. The controls that were proposed but not developed, did not get implemented because of time, though the hardware capability is there for these features to be implemented in the future.

CONTROL	PROPOSED AT CDR	DEVELOPED
Volume Down	Bend finger	Bend finger
Volume Up	Extend finger	Extend finger
Pitch Up	Raise hand	Raise or tilt hand in (<i>changed</i>)
Pitch Down	Lower hand	Lower or tilt hand out (<i>changed</i>)
Tone Low	Tilt hand in	(<i>not developed</i>)
Tone High	Tilt hand out	(<i>not developed</i>)

CONTROL	PROPOSED AT CDR	DEVELOPED
Pitch mode select	Button	(not developed)
Waveform select	Button	Button
Filter select	Button	(not developed)

The most difficult feature to implement is using an accelerometer to measure the vertical distance traveled by the glove to control the pitch. Instead, the z-axis value from the accelerometer is forwarded to the pitch control, so that rotating the glove through a constant gravity field alters the pitch. The glove can be rotated in any direction to change the pitch, since any rotation alters the direction of the z-axis relative to the Earth.

1.2.2 Additional Features (vs traditional theremin)

- **Selectable pitch mode:**
 - Continuous – any frequency possible. Accelerometer data is scaled and forwarded to the pitch control.
 - Quantized (not developed) – frequency is rounded to the nearest musical note. Accelerometer data is compared to a lookup table to determine the pitch control.
- **Physical pitch range mapping is definable (not developed)**, i.e. the height at which the hand is located for the lowest and highest notes are configurable. This is required for the system to work, since it relies on an accelerometer to measure vertical movement and the accelerometer cannot measure absolute distance. Before playing, the user must set the range by:
 - 1. Placing hand at the location of the lowest note
 - 2. Pressing a button on the glove to start recording the range
 - 3. Raise hand to the location of the highest note
 - 4. Pressing a button on the glove to end recording
 - 5. Play
- **Filter selection:** Low-Pass (doesn't work), Band-Pass (doesn't work), and High-Pass filters can be selected from the glove controller to alter the tone of the waveform

- **Waveform selection:** Square, triangle (*doesn't work*), and sine waveforms can be selected from the glove controller

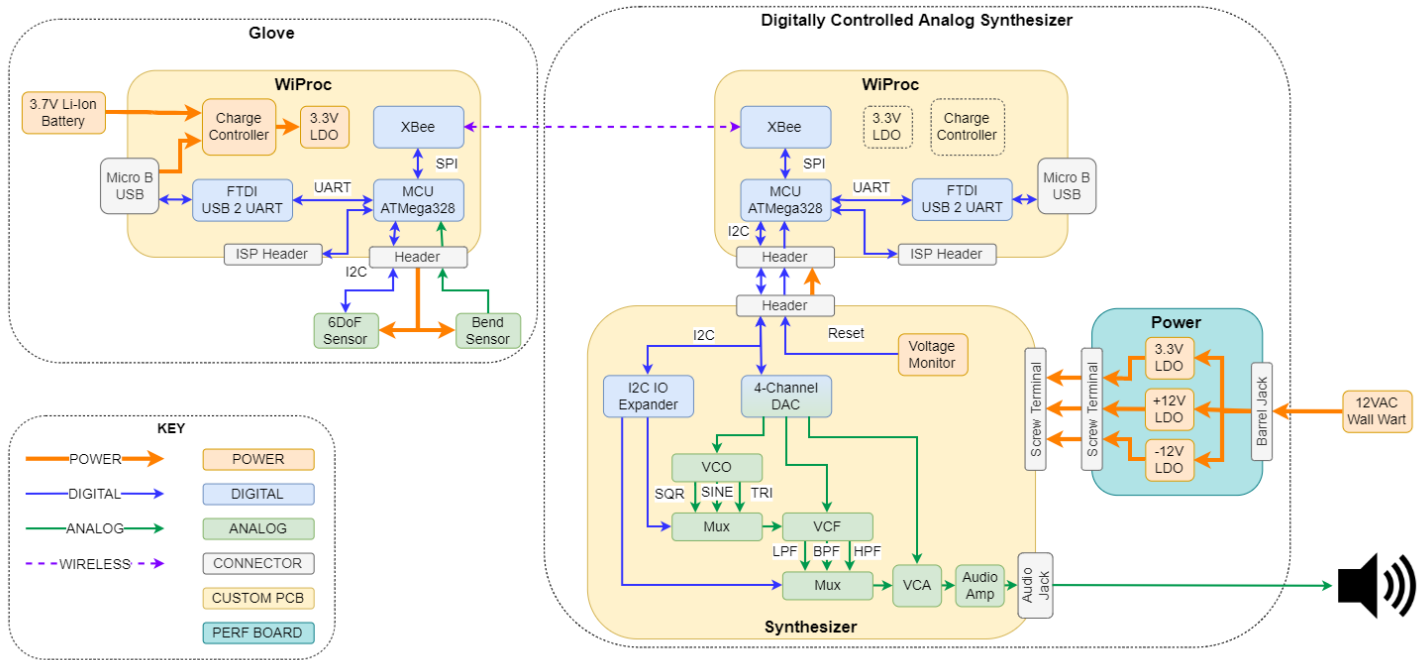
The low-pass filter, band-pass filter, and triangle waveform circuitry match the breadboarded circuitry that was tested and measured. These circuits may not work due to defects in the PCB. After receiving the PCBs, OSH Park sent notification that the fabrication used a drill setup with the wrong-sized drills, which may cause some opens in the board, and that there may have been over etching in some areas, which could have caused traces to be lifted and shorted to adjacent traces. These circuits have not been diagnosed yet.

1.3 LIMITATIONS

- *Right-hand glove only – this prototype will not support the glove being worn on the left hand (accelerometer would be upside-down). (*Not true for project demonstration*)*
- *The glove will not support absolute position measuring – the range mapping must be performed for every use.*
- *The hardware is capable of more features, but only those listed in section 1.2 are promised.*

2 FUNCTIONAL DESCRIPTION

2.1 SYSTEM BLOCK DIAGRAM



2.2 MAJOR COMPONENTS

As an electronics system, the Theremin Glove is a digitally controlled analog synthesizer with a wireless glove controller. Thus, the two major components are the glove and the synthesizer.

2.2.1 The Glove

The glove is a wearable device whose purpose is to capture and transmit hand movements, gestures, and button presses to the synthesizer module. It consists of:

- **Wireless processor CCA (WiProc)**
 - **ATMega328P microcontroller** – collects data from sensors and controls wireless transceiver
 - **Digi XBee3 wireless transceiver** – transmits sensor data to synthesizer module
 - **USB interface** for programming, debug, and battery charging
 - **Battery charging manager**

- **In System Programming header**
- **6DoF Sensor**
 - **3D Accelerometer** captures positional motion
 - **3D Gyroscope** captures rotational motion
- **Bend Sensor**
 - Captures finger bending
- **3.7V Li-Ion Battery** for completely untethered glove

2.2.2 The Synthesizer

The synthesizer module's purpose is to receive sensor data from the glove and translate it into parameters by which to synthesize audio waveforms. It consists of:

- **Wireless processor CCA (WiProc)**
 - **Common CCA** with glove controller with different firmware and battery components not populated
 - **ATMega328P microcontroller** – receives data from XBee and converts it to controls the analog synthesizer
 - **Digi XBee3 wireless transceiver** – receives sensor data from the glove
 - **USB interface** for programming and debugging
- **Analog synthesizer CCA**
 - **4-Channel DAC** generates voltages to the voltage-controlled circuits
 - **Voltage-Controlled Oscillator(s)** synthesizes the square, triangle, and sine waveforms of the frequency set by the control voltage
 - **Voltage-Controlled Filter** filters the audio waveform with the resonant frequency set by the control voltage
 - **Voltage-Controlled Amplifier** sets the gain of the waveform based on the control voltage
 - **I2C IO Expander** allows for fewer pins from the microcontroller and all synthesizer control over an I2C interface – making each control neatly addressable
 - **4:1 Analog Multiplexers** are used for the waveform and filter selection
 - ~~**Audio Amplifier** allows for direct interface with a speaker. The audio amplifier was removed after breadboard testing determined it was not necessary.~~
- **Power Perf Board Assembly**
 - **12VAC input from wall wart**
 - **+3.3V DC 1.5A out** for digital circuits
 - **+/-12V DC 1.5A out** for analog circuits
 - Traditional analog synthesizer instruments typically use an analog voltage of +/-9V to +/-15V, so +/-12V makes for a nice place in the middle.

2.3 MICROCONTROLLER USES

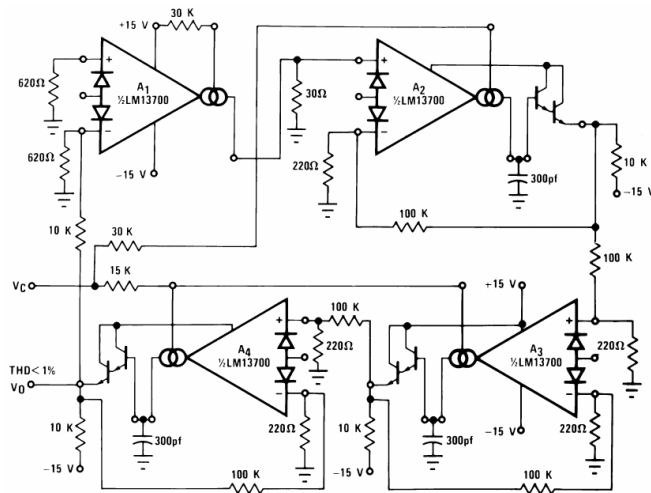
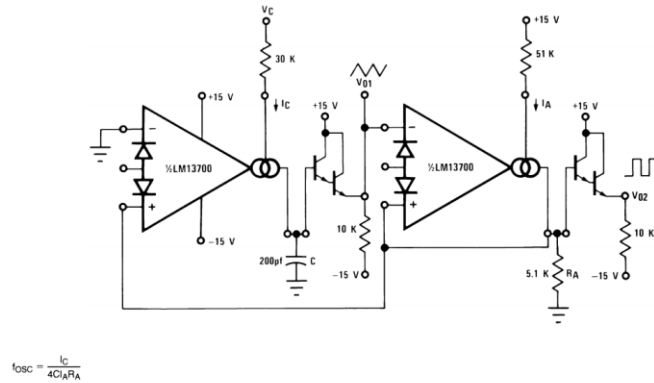
The following table highlights the major functionality of the microcontroller on each WiProc CCA in the system.

COMPONENT	GLOVE	SYNTHESIZER
MAIN PROCESS	MONITOR AND TRANSFER SENSOR DATA	RECEIVE DATA AND PROCESS TO CONTROL SYNTHESIZER FREQUENCY, WAVEFORM, TONE, AND VOLUME. EXPONENTIAL CONVERSION IN MCU ELIMINATES ANALOG CIRCUITS WITH MATCHED TRANSISTORS AND TUNING.
ADC	MEASURE BEND SENSOR AND BOARD VOLTAGES	MEASURE BOARD VOLTAGES
I2C	POLL 6DOF SENSOR	SYNTHESIZER CONTROLS: DAC AND MUX SELECTS
SPI	MASTER XBEE TRANSMIT	MASTER XBEE RECEIVE
UART	PROGRAMMING AND DEBUG CONSOLE	PROGRAMMING AND DEBUG CONSOLE
LOGIC DISCRETES	LEDS AND BUTTONS	LEDS AND BUTTONS

2.4 VCO CIRCUIT TOPOLOGY

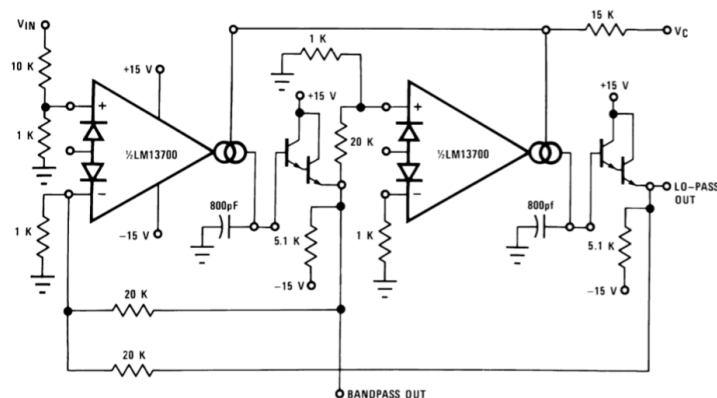
The following circuits were taken from the LM13700 datasheet and will be the basis for the VCO design. The LM13700 is an Operational Transconductance Amplifier with built in output buffers and will be the main device for each of the voltage-controlled circuits.

These circuits were simulated in LTSpice and verified by on a breadboard.



2.5 VCF CIRCUIT TOPOLOGY

The following circuit schematics are the basis for the VCF design. They were sourced from the LM13700 datasheet. These circuits were verified by simulation in LTSpice and breadboard testing.



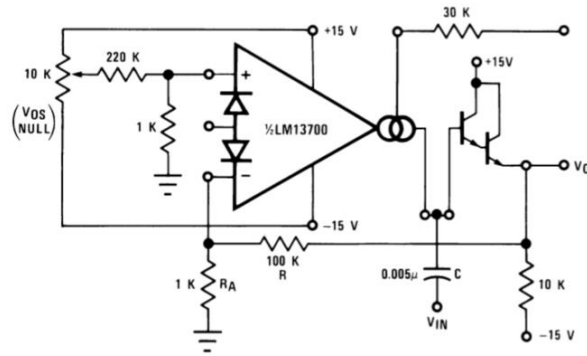


Figure 30. Voltage-Controlled Hi-Pass Filter

2.6 VCA CIRCUIT TOPOLOGY

The following circuit is from the LM13700 datasheet and will serve as the basis for the VCA design.

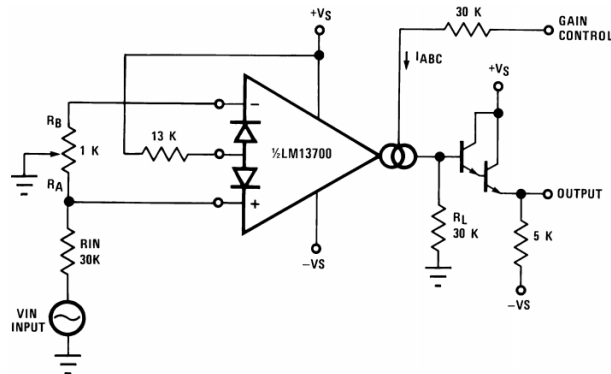


Figure 17. Voltage Controlled Amplifier

2.7 A NOTE ON HARDWARE CAPABILITY

The hardware design is generic enough that it can be reused in other applications. For example, the glove controller could control any device with a wireless receiver and the synthesizer could be controlled by any device with a wireless transmitter (such as a smartphone via Bluetooth) or over USB.

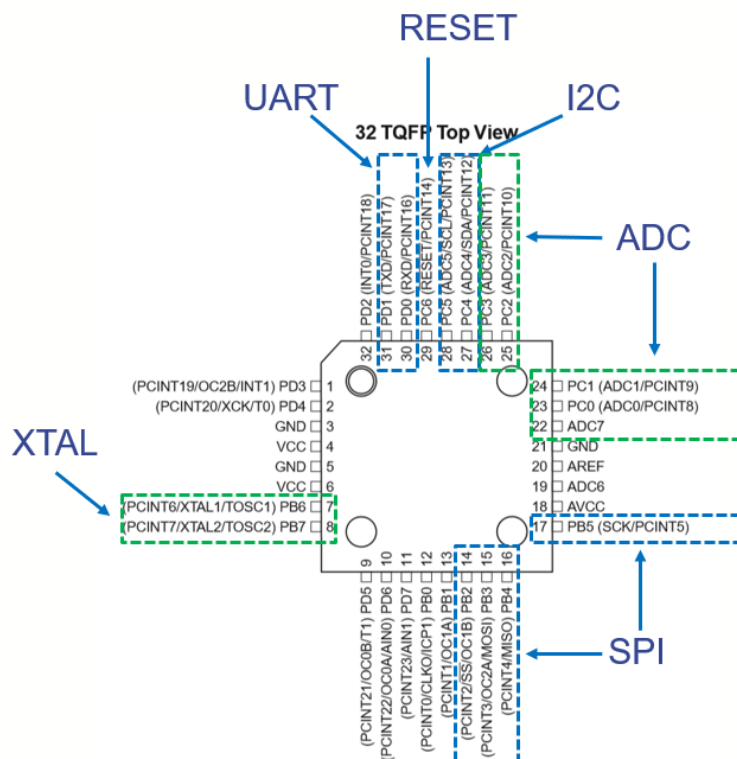
3 INTERFACE DESCRIPTIONS

3.1 MICROCONTROLLER INTERFACES

The major interfaces of the system are between the microcontroller and other digital devices. The ATmega328P's built-in interfaces will be used so that no custom ones need to be written. The interfaces are:

- **SPI** – to control XBee3
- **I2C** – to control 6DoF sensor or synthesizer CCA
- **UART** – for serial console to PC via FTDI USB-UART converter
- **ADC** – to measure bend sensor and supply voltages
- **Discretes** – for button input and LED control

The following diagram shows the pin allocation of the ATmega328P and its built-in interfaces.



3.2 IO REQUIREMENTS

- All digital IO is +3.3V logic.
- Analog inputs to the ATmega328P cannot exceed +3.3V

- Small-signal voltage supplies are +/-12V

3.3 POWER REQUIREMENTS

- The glove is powered by a 3.7V 2000mAh Li-Ion battery or by 5V at the USB connector.
- The synthesizer is powered by +3.3V and +/-12V with a ground common to all supplies.
- The synthesizer LDOs are rated for 1.5A

3.3.1 Wireless Processor Power Estimate

Part Number	+3.3V		+5V		Typ Power [mW]	Max Power [mW]
	Typ [mA]	Max [mA]	Typ [mA]	Max [mA]		
XBee3	40.0	52.0			132.00	171.60
ATMega328P	5.2	9.0			17.16	29.70
LSM6DS3	1.3	1.6			4.13	5.36
FSL0095103ST	0.3	0.3			1.09	1.09
LMV118	0.6	0.9			1.98	2.97
LMV118	0.6	0.9			1.98	2.97
FT231X			8.0	8.4	40.00	42.00
FODM8071	3.3	4.8			10.89	15.84
SN74AUP1G97	0.5	0.9			1.65	2.97
MCP73831			0.5	1.5	2.55	7.50
220 Ω RES LED, MCU1	12.3	12.3			40.50	40.50
220 Ω RES LED, MCU2	12.3	12.3			40.50	40.50
220 Ω RES LED, UART TX	12.3	12.3			40.50	40.50
220 Ω RES LED, UART RX	12.3	12.3			40.50	40.50
470 Ω RES LED, BATT CHRG			9.4	9.4	46.81	46.81
Sub Total	100.9	119.5	17.9	19.3	422.23	490.81
Regulator	Vin	Vout	GND Current [mA]		Typ Power [mW]	Max Power [mW]
TLV70033	3.7	3.3	0.3		40.3	47.8
Total Power					462.58	538.63

3.3.2 Synthesizer Power Estimate

Part Number	+3.3V		+12V		-12V		Qty	Typ Power [mW]	Max Power [mW]
	Typ [mA]	Max [mA]	Typ [mA]	Max [mA]	Typ [mA]	Max [mA]			
LM13700			1.300	1.690	1.300	1.690	5	156	202.80
OPA172			0.508	0.725	0.508	0.725	4	48.72	69.60
OPA4172			0.508	0.725	0.508	0.725	12	146.16	208.80
MUX508			0.045	0.059	0.025	0.034	2	1.68	2.23
PCF8574A	0.040	0.100					1	0.132	0.33
AD5694R	1.100	1.300					1	3.63	4.29
Sub Total (x Qty)	1.1	1.4	14.7	20.2	14.7	20.1		356.32	488.05
Total Power								356.32	488.05

4 MATERIAL AND RESOURCE REQUIREMENTS

4.1 DEVELOPMENT TOOLS

The following list of software was used for the design, simulation, and testing of this project. It is all available online for **FREE**.

- **LTSpice** for analog and power simulations
- **KiCAD** for schematic capture and PCB layout
- **Atmel Studio** for microcontroller C code development and programming
- **FT_PROG** for FTDI USB to UART configuration
- **XCTU** for XBee3 configuration
- **PuTTY** for the debug serial console
- **Waveforms** for the Digilent Analog Discovery 2 measurements

4.2 TEST EQUIPMENT

The following list of test equipment is pretty standard and available at home and in the lab:

- **Digilent Analog Discovery 2** – to measure analog waveforms and filter frequency responses
- **Oscilloscope** – to measure analog waveforms and digital timing
- **Signal Generator** – to source analog waveform for analog circuits as needed
- **Digital Multimeter** – for voltage measurements, continuity checks, etc.
- **Breadboard** – for testing and developing circuits with through-hole component versions

The power board was constructed first so that no power supply and wiring adapters were needed.

4.3 BUILD EQUIPMENT

The following list of build equipment is required to construct the CCA's by hand and is available mostly at home, but all is available in the lab:

- **Soldering station with fine tip**
- **Microscope**
- **ESD safe tweezers**
- **Solder wick**
- **Work holder**
- **Ventilation**

4.4 KNOWLEDGE BASE

I have done all of the work that this project requires in one form or another, however not all at once in a big project like this. I have experience with:

- *PCB design*
- *ATMega328P development in C which involved the built-in serial interfaces*
- *Digital, analog, and power circuit design, testing, and troubleshooting*
- *Hand-soldered surface mount components – ATMega328P 32-TQFP being the smallest device I've done with 9mm pitch leads*

I do not feel out of my comfort zone in the undertaking of this project and I plan to discuss and review my designs before PCB fabrication with engineering peers in and out of class.

Where applicable, the design schematics for Adafruit and Sparkfun are referenced where they use the same device in similar applications.

4.5 PART SPECIFICATIONS

The following part specifications are based on my construction capabilities, budget, schedule, and power supply requirements. They are also dependent on the availability of the microscope and soldering station in the lab.

- *All parts must be hand-solderable*
 - *Leaded packages with no exposed pads*
 - *No smaller than 9mm pitch leads*
 - *Passives not smaller than 0603*
- *Nominally rated for +3.3V and +/-12V*
- *Immediately available to ship*
- *In production and not expensive*
- *Order samples when available*

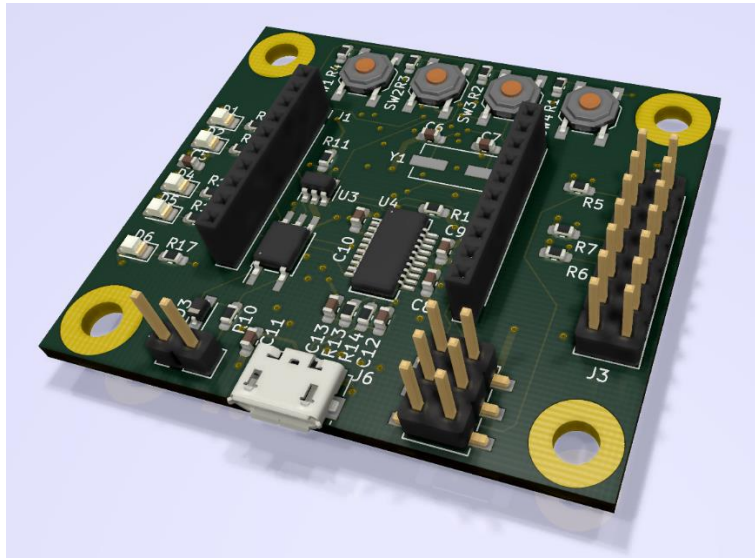
4.6 COMPONENTS

The major components are listed in the following table. They have large quantities in stock at Digi-Key and Sparkfun and some parts are available as samples from Analog Devices.

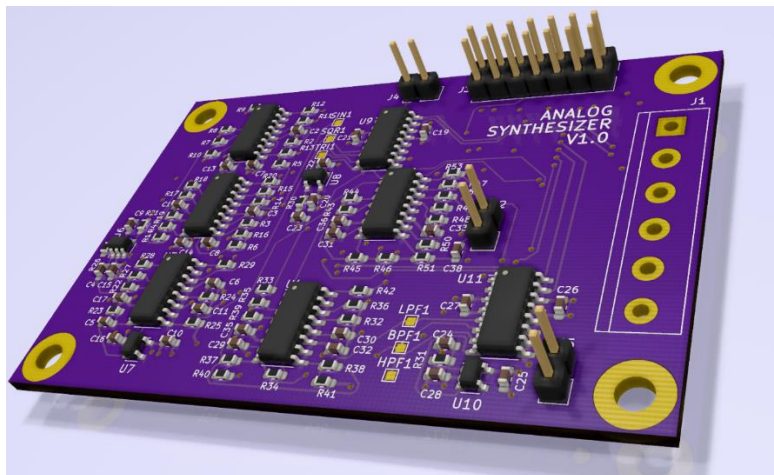
COMPONENT	DESCRIPTION	WHERE USED
AD5694	12-BIT DAC, 4-CHANNEL, I2C	SYNTHESIZER
LM13700	OPERATIONAL TRANSCONDUCTANCE AMPLIFIER	SYNTHESIZER
MUX508	ANALOG MULTIPLEXER WITH ENABLE	SYNTHESIZER
PCF8574A	8 CHANNEL I2C IO EXPANDER	SYNTHESIZER
OPA172/ OPA4172	GENERAL PURPOSE OP-AMP	SYNTHESIZER

COMPONENT	DESCRIPTION	WHERE USED
FT231X	FTDI USB TO UART	WIPROC
MCP73831	LI-ION BATTERY CHARGING MANAGER	WIPROC
TLV118	LOW VOLTAGE OP-AMP	WIPROC
TLV70033	3.3V LDO	WIPROC
SN74LVC1G97	2:1 MUX / CONFIGURABLE LOGIC GATE	WIPROC
FSL0095103ST	FLEX SENSOR	GLOVE
LSM6DS3	6DOF BREAKOUT BOARD	GLOVE
LM1086-3.3	+3.3V LDO, TO-220-3	POWER
UA7812	+12V LDO, TO-220-3	POWER
L7912	-12V LDO, TO-220-3	POWER
WAU12-2000	12VAC WALL ADAPTER	POWER

4.7 WIPROC PCB



4.8 SYNTHESIZER PCB



4.9 PCB PRODUCTION

The PCBs were manufactured by OSH Park using their 4-layer board process. The WiProc board measured 1.93" x 1.65" and the Synth board measured 2" x 3". OSH Park delivered on their 12 day turn-around with no hidden fees, with free shipping as an option.

OSH Park offers:

- 2-layer boards at \$5/sq in (includes 3 PCBs)
- 4-layer boards at \$10/sq in (includes 3 PCBs)
- Ships in 12 days

- *A lot of documentation on the website on PCB production and how to provide them the correct files they need*
- *Purple soldermask for added flair*

5 DEVELOPMENT PLAN AND SCHEDULE

5.1 APPROACH

The approach to completing the project in the time available involved reducing the risk of custom PCB design by verifying all circuits on breadboards first. This decreased the need for any work-arounds and less than optimal fixes to the CCAs in the end. The goal was to not need hardware changes after PCBs are in production.

Firmware was developed in between hardware down-time (waiting on parts, PCBs, schematic/layout reviews). Breadboarding both the analog and digital circuits greatly aided in the development of the MCU code while waiting for the PCBs to arrive.

5.2 ORDER OF DEVELOPMENT

1. *Preliminary design phase*
 - a. *Project definition*
 - b. *Requirements definition*
 - c. *PDR*
 - d. *CDR*
2. *Design phase*
 - a. *Schematic development*
 - b. *Microcontroller development*
 - c. *Circuit simulations*
 - d. *Breadboard verification*
 - e. *PCB layout*
 - f. *PCB fabrication*
3. *Post design phase*
 - a. *CCA construction*
 - b. *CCA verification*
 - c. *Update microcontroller with real glove parameters*

4. Final demonstration

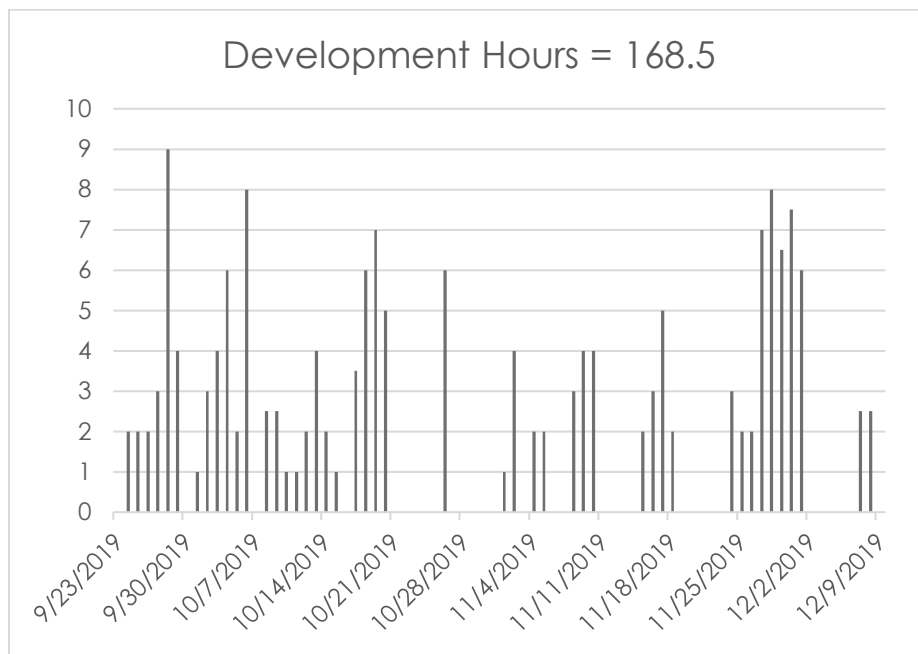
5.3 SCHEDULE

WEEK	TASK	ON TIME?
8/26 – 9/23	<ul style="list-style-type: none">- Defined project scope and developed PDR- Designed and simulated power board- Tested XBee to XBee communication- Selected most components	Y
9/30	<ul style="list-style-type: none">- Refined project concept- Microcontroller to XBee breadboarded and tested- UART and SPI interfaces developed and tested	Y
10/7	<ul style="list-style-type: none">- Build power board- Analog circuit design	Y
10/14	<ul style="list-style-type: none">- Analog circuit breadboard and test	Y
10/21	<ul style="list-style-type: none">- PCB design and part procurement.	Y
10/28	<ul style="list-style-type: none">- Finalize PCB design and place order.	Y
11/4	<ul style="list-style-type: none">- Microcontroller development	Y
11/11	<ul style="list-style-type: none">- Microcontroller development and glove construction	Y

WEEK	TASK	ON TIME?
11/18	- CCA construction	Y
11/25	- CCA Testing - Finalize microcontroller code	Y
12/2	- Finalize documentation	Y
12/9	- Demonstration	Y

5.4 HOUR TRACKING

For future reference, I wanted to know how much time a project like this would take me, so I kept track of how many hours I spent on it.



5.5 RISK

- **Schedule** – this project had a lot going on for one person and the schedule was pretty tight with the PCB lead time. The schedule required consistent progress to be made almost daily.
- **Custom PCBs** – there was no room in the schedule for PCB re-spin. The PCBs had to be designed correctly for the project to be completed and to work as intended.
- **SPI interface between microcontroller and XBee** – the documentation suggests that the XBee3 is I2C compatible, however other sources suggest that this feature is not enabled. There is little documentation on the SPI interface, so getting it to work may be a challenge, if it is indeed available in the current firmware.

5.6 RISK MITIGATION

- **Schedule** – I am passionate about this project and spent an average of 14 hours per week on it to meet the milestones I set for myself. The pressure of a time constraint is something I prefer as it forces engineering decisions to make something work one way or another.
- **Custom PCBs** – All circuits were verified with simulation and breadboarding prior to PCB fabrication.
- **SPI interface between microcontroller and XBee** – this risk has been eliminated. A working SPI interface between the XBee and microcontroller has been developed and verified. It was tricky as expected, since the XBee3 SPI documentation is minimal and its API command documentation is incomplete (see the XBee3 SPI section in Appendix II: Lessons Learned).

6 RESULTS

6.1 WIRELESS PROCESSOR BOARDS

The glove and synthesizer controller wireless processor boards worked without issues, except for one minor pin that needed to be corrected. The low-power op-amp used for the ADC input conditioning has an “enable (EN)” pin that is active high, but in the KiCAD library this pin was called “shutdown (SD)”, and I assumed the opposite control logic of what is given in the datasheet. This pin was directly tied to ground so that it was always in the disabled state. The correction was to lift the pin from the board, bend it, and short it to an adjacent VCC pin.

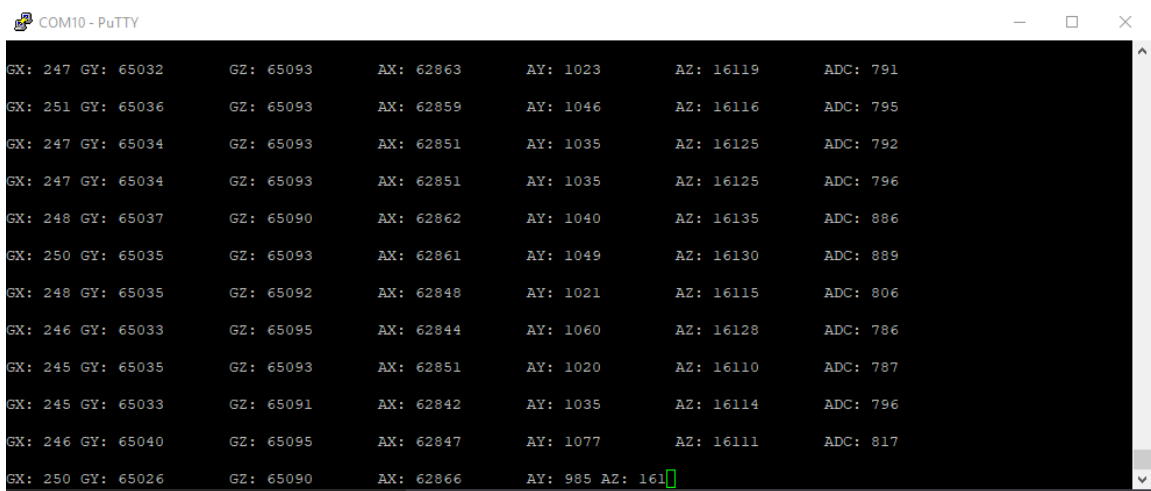
6.1.1 Verification procedure and results

The following steps were performed to verify the board design once the boards were constructed. These steps were done in parallel with code development. The table shows the final results after hardware debugging.

Step	Procedure	P/F
1	Ohm out power/gnd rails	P
2	Check continuity of board IO and select signals	P
3	Power on via USB and measure voltages	P
4	Power on via power supply and measure voltages	P
5	Charge batter via USB	P
6	Power on via battery and measure voltages	P
7	Detect MCU through programming tool	P
8	Program that toggles LEDs when buttons pressed	P
9	Board to board communication – transmitting WiProc sends character on button press, receiver prints to serial terminal	P
10	Transmit and print ADC data to serial terminal	P
11	Transmit and print 6DoF data to serial terminal	P

6.1.2 Communication

Transferring sensor data from the glove to the synthesizer controller is a major component of the design and it worked as intended. The capture below shows the end result of streaming data to a serial terminal at 20 Hz. The data path is: 6DoF I2C & Flex Sensor Analog → Glove MCU SPI → XBee3 Wireless TX → XBee3 Wireless RX → XBee3 SPI → Synth MCU UART → FTDI UART to USB → PC Serial terminal



GX: 247	GY: 65032	GZ: 65093	AX: 62863	AY: 1023	AZ: 16119	ADC: 791
GX: 251	GY: 65036	GZ: 65093	AX: 62859	AY: 1046	AZ: 16116	ADC: 795
GX: 247	GY: 65034	GZ: 65093	AX: 62851	AY: 1035	AZ: 16125	ADC: 792
GX: 247	GY: 65034	GZ: 65093	AX: 62851	AY: 1035	AZ: 16125	ADC: 796
GX: 248	GY: 65037	GZ: 65090	AX: 62862	AY: 1040	AZ: 16135	ADC: 886
GX: 250	GY: 65035	GZ: 65093	AX: 62861	AY: 1049	AZ: 16130	ADC: 889
GX: 248	GY: 65035	GZ: 65092	AX: 62848	AY: 1021	AZ: 16115	ADC: 806
GX: 246	GY: 65033	GZ: 65095	AX: 62844	AY: 1060	AZ: 16128	ADC: 786
GX: 245	GY: 65035	GZ: 65093	AX: 62851	AY: 1020	AZ: 16110	ADC: 787
GX: 245	GY: 65033	GZ: 65091	AX: 62842	AY: 1035	AZ: 16114	ADC: 796
GX: 246	GY: 65040	GZ: 65095	AX: 62847	AY: 1077	AZ: 16111	ADC: 817
GX: 250	GY: 65026	GZ: 65090	AX: 62866	AY: 985	AZ: 161	

6.2 ANALOG SYNTHESIZER BOARDS

The analog synthesizer circuits worked as intended except for the VCF as mentioned before. While debugging the board, I noticed the VCF amplifier got pretty warm, and I think at one point I heard some sizzling. I haven't diagnosed the issue to this circuit, except that continuity seems to be okay from passives to IC pins and the circuit on the board matches the breadboarded circuit. However, it is possible there is a short in the board that has not been detected yet due to the OSH Park issue.

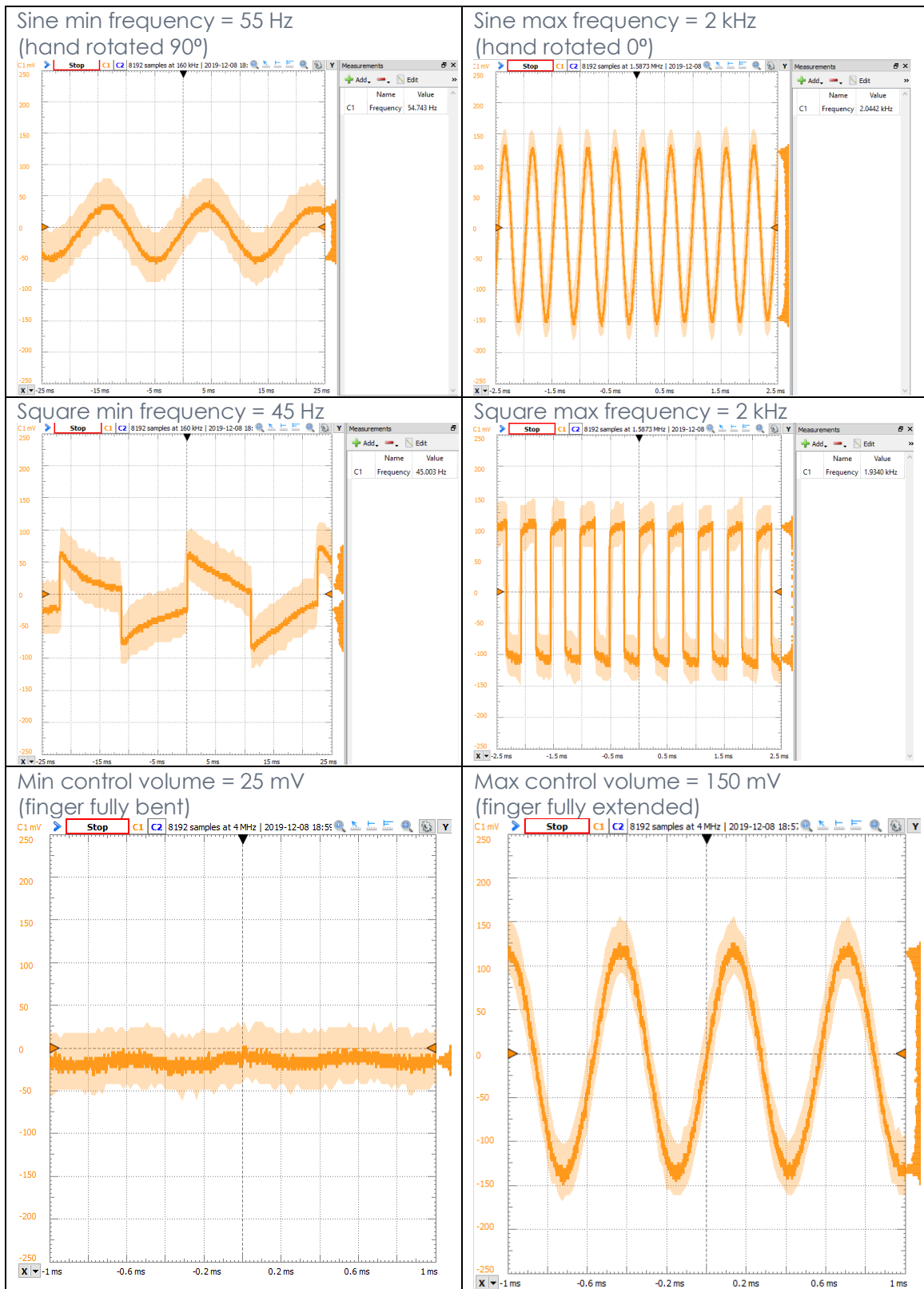
6.2.1 Verification procedure and results

The following steps were performed to verify the board design once the boards were constructed. These steps were done in parallel with code development. The table shows the final results after hardware debugging.

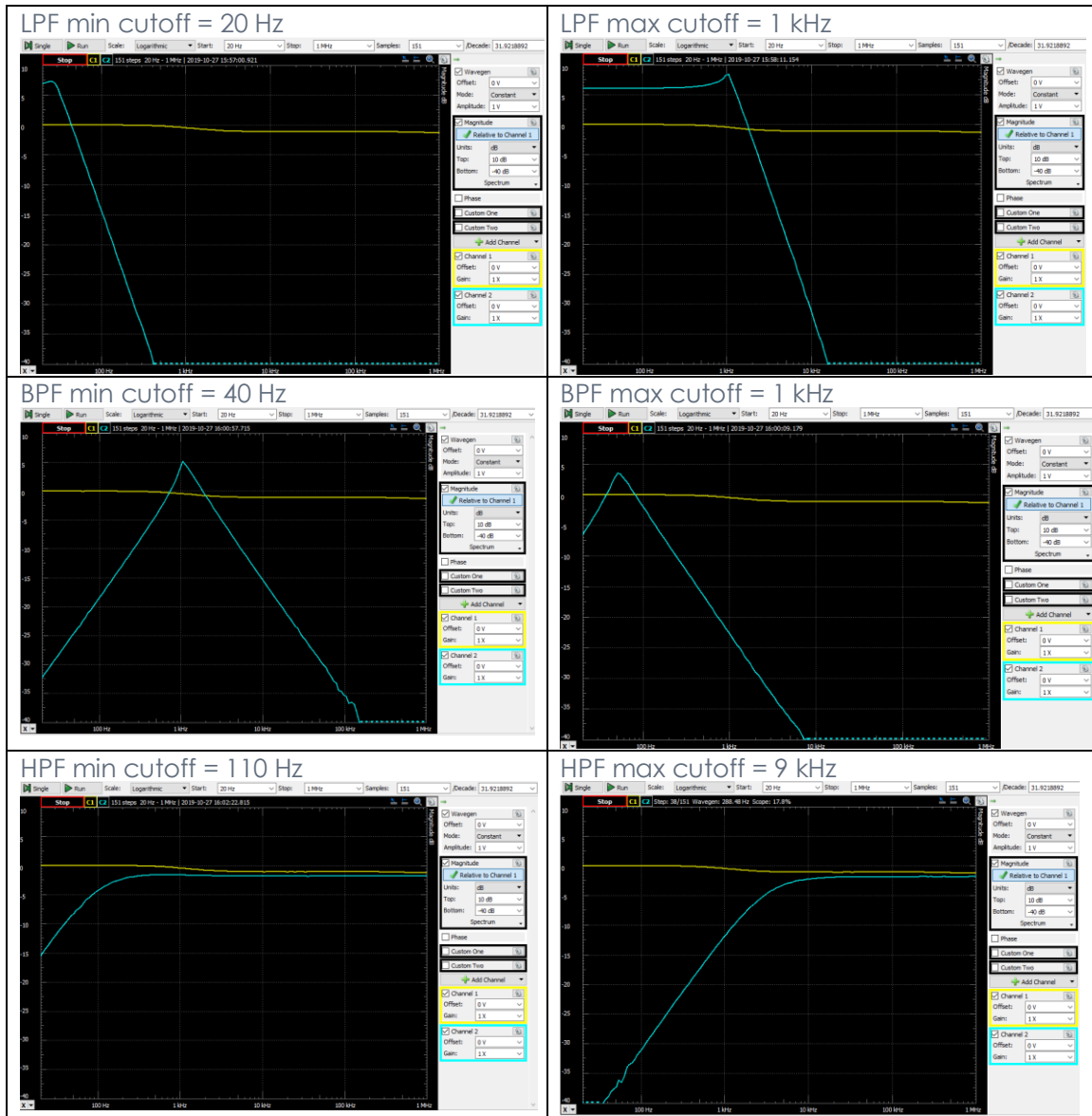
Step	Procedure	P/F
1	Ohm out power/gnd rails	P
2	Check continuity of board IO and select signals	P
3	Power on via power supply and measure voltages	P
4	Set DAC and MUX signals and measure voltages	P
5	Measure sine wave from VCO	P
6	Measure square wave from VCO	P
7	Measure triangle wave from VCO	F
8	Vary VCO control voltage and measure frequency range	P
9	Measure LPF output from VCF	F
10	Measure BPF output from VCF	F
11	Measure HPF output from VCF	P
12	Vary VCF control voltage and measure cutoff frequency range	F
13	Vary VCA control voltage and measure volume range	P

6.2.2 Waveforms

The waveforms in the table below were measured at the VCA output as controlled by the glove.



6.2.3 Breadboarded filters



7 ASSEMBLY INSTRUCTIONS

7.1 COMPONENTS

- Glove
- Analog Synthesizer
- Speaker

- 12VAC wall-wart
- 3.7V Li-Ion Battery

7.2 CONNECTIONS

1. Ensure 6DoF and flex sensor are connected to J3 on the glove according to the schematic
2. Connect Li-Ion battery to J7 on glove WiProc and verify LEDs D1 and D2 are illuminated
3. Ensure the +3.3V, +12V, and -12V are wired correctly from the power supply board to J1 of the analog synthesizer.
4. Ensure J3 of the analog synthesizer is connected to J3 of the WiProc synthesizer controller.
5. Ensure speaker is connected to J2 of the analog synthesizer.
6. Connect the 12VAC wall-wart to the barrel plug connected to the power supply.

7.3 USING THE DEVICE

1. Verify all connections are good
2. Turn on the power supply
3. Press SW1 on the glove to turn on/off audio signal
4. Rotate hand to vary the pitch
5. Extend or bend middle finger to vary the volume
6. Press SW4 on the glove to toggle between sine and square wave signals

7.4 MODIFYING THE DESIGN

- Refer to section 8.1 for microcontroller code, schematics, and layouts
- Refer to section 4.1 for required tools

8 DOCUMENTATION

8.1 GITHUB CONFIGURATION MANAGEMENT

The microcontroller development and PCB designs are under configuration control with github. The online repository can be found here:

https://github.com/tylerhudd/theremin_glove

9 REFERENCES

9.1 SCHEMATIC RE-USE

- **SparkFun XBee Explorer USB** – XBee3 schematic and FTDI USB to UART conversion
<https://www.sparkfun.com/products/11812>
- **Adafruit MCP73831** – for USB battery charging
<https://www.adafruit.com/product/259>

9.2 DESIGN TOPOLOGIES AND THEORY

- **Music From Outerspace** – for general synthesizer design and DC power supply from AC wall-wart
<http://musicfromouterspace.com/>
- **Birth of a Synth** – for VCF circuit
https://www.birhofasynth.com/Thomas_Henry/Pages/VCF-1.html
- **LM13700 datasheet** – for VCO and VCA circuit
<http://www.ti.com/lit/ds/symlink/lm13700.pdf>
- **ATmega328 datasheet** – for code examples and register definitions
http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

9.3 GENERAL

- **The Art of Electronics 3rd Edition** by Horowitz and Hill
- **The C Programming Language 2nd Edition** by Kernighan and Ritchie

APPENDIX I: PROGRESS

A progress log was kept for the sake of gathering data and holding myself accountable to continually work on the project.

Date	Worked On	Issues	Next Time	Hours
9/24/2019	<ul style="list-style-type: none">- Breadboarded 2x ATmega328 to XBee SPI interface- Verified no Vcc to GND shorts- Verified detection of both XBee's in XCTU- Verified detection of both MCU's in Atmel Studio	<ul style="list-style-type: none">- None	<ul style="list-style-type: none">- Write code to configure SPI interface- Configure XBee for SPI- Verify some event (button press) that generates SPI message to XBee then read from XBee in XCTU console- Verify even to SPI message to XBee to other XBee read from XCTU console	2
9/25/2019	<ul style="list-style-type: none">- Created spi.c/h files for testing SPI to XBee3- These files configure Atmega328 as SPI master and	<ul style="list-style-type: none">- None	<ul style="list-style-type: none">- Continue code and configure XBee3 for SPI mode- Program devices and test	2

	include functions to transmit 1 character and strings			
9/26/2019	- Continued code to configure xbee in SPI mode - didn't get any results	- trying to get spi interface to work	- continue working on spi interface - try getting uart to work for debug messages	2
9/27/2019	- found useful uart files here: https://gist.github.com/gshrikant/8549474 - wrote simple program to receive data from mcu over uart with success after trying baud of 2400 - discovered internal clock is 1 MHz (internal RC oscillator is 8 MHz, but clock/8 fuse is enabled)	- need to get spi interface working	- work on SPI interface and add debug messages - investigate faster uart baud	3
9/28/2019	- got spi interface working: mcu transmits to xbee over spi, then xbee to xbee, then received data read over serial console from 2nd xbee shows successful transfer - issues overcome: 1. few bugs in code - wrong ports and pins assigned 2. mixed up 2 wires on breadboard 3. spi has to be in API frame format - needs to send start byte, length, command, data, checksum, etc.	- None	- build power board - implement serial1 -> mcu1 -> xbee1 -> xbee2 -> mcu2 -> serial2	9
9/29/2019	- started git repository for configuration control - started github projects board to track things to do - replaced soldering iron heating element - gathered components for power board - need to determine output connector	- None	- build power board - implement serial1 -> mcu1 -> xbee1 -> xbee2 -> mcu2 -> serial2	4
9/30/2019	-class, demonstrated working RF transmit and receive	- None		0
10/1/2019	- finalizing power board design - ordered screw terminals for input/output power - laid out / fit check component placement	- None	- serial comms	1
10/2/2019	- started SPI MISO reading and decoding	- None	- finish SPI MISO reading	3
10/3/2019	- successfully decoded SPI received messages after transmit: the xbee returns a status message for transmit success and a modem status message after reset	- None	- xbee receive packet decoding	4
10/4/2019	- successfully transmitted from PC terminal to uC to Xbee to Xbee to uC to PC terminal: uart works in both directions and SPI works in both directions - the trick for receiving SPI messages was figuring out that the XBee uses different frame types than those listed in the datasheet, but are shown in XCTU	- None	- clean up and finalize interface functions	6
10/5/2019	- optimized to pc terminal to pc terminal chain and cleaned up code with macros and debug statements for readability and reduced size	- None	- build power board, all parts in today	2
10/6/2019	- CDR document - started power board, input filtering and rectifying installed	- None	- finish power board (regulators and output filtering)	8
10/7/2019	- Went to class	- None	- finish power board	0
10/8/2019	- worked on power board (+3.3V and +12V regulator circuits and LED header)	- None	- finish power board (-12V regulator and ground connections)	2.5

			- safe to turn on check - wire input barrel jack and switch	
10/9/2019	- finished building power board - measured outputs with DMM and they are +3.3V, +12V, -12V as designed	- None	- start breadboarding analog circuits	2.5
10/10/2019	- created LTSpice symbol for LTM13700 from TI's Pspice model - simulated VCA from LTM13700 datasheet	- None	- simulate topologies of other analog circuits	1
10/11/2019	- started VCO simulation using example in datasheet, couldn't get it to run	- Getting VCO sim working	- simulate VCO	1
10/12/2019	- got VCO sim to run, needed initial condition of cap used in integrator set to 0 - simulation doesn't quite work for all conditions, it stops after first oscillation, seems to be an LTSpice issue - read up on OTA's	- None	- breadboard VCO	2
10/13/2019	- breadboarded VCO - modified design for desired frequency range - designed circuit to shift range of DAC output from 0 to +2.5V to -12V to +12V	- None	- breadboard VCA, start VCF	4
10/14/2019	- went to class, demonstrated VCO and power board - started WiProc schematic	- None	- continue schematic - breadboard VCA, start VCF	2
10/15/2019	- worked on WiProc schematic	- None	- continue schematic	1
10/17/2019	- finished first cut of WiProc schematic	- None	- assign footprints and layout WiProc	3.5
10/18/2019	- initial part placement and board outline	- None	- switch gears to analog design: sim, breadboard, schematic, layout	6
10/19/2019	- added battery connector to WiProc - simulated sine VCO - started single VCO engine design with multiple waveform outputs	- None		7
10/20/2019	- simulated and breadboarded VCF - breadboarded VCA and speaker driving stage - tested audio, need to make some adjustments	- None	- finalize analog circuitry - draft analog schematic and initial layout	5
10/26/2019	- routed wiproc board	- None	- submit to osh park for quote	6
11/1/2019	- ordered wiproc PCB	- None	- finalize analog circuit - make analog circuit schematic - analog layout	1
11/2/2019	- tweaked analog breadboard - fixed resistors, changed a few values - measured HPF and exported data - worked on schematic	- None	- continue analog schematic - analog layout	4
11/4/2019	- finished analog schematic	- None	- work on layout	2
11/5/2019	- initial analog layout	- None	- work on layout	2
11/8/2019	- worked on layout and started routing	- None	- work on routing	3
11/9/2019	- finished routing	- None	- Double check design - submit to osh park - order parts	4
11/10/2019	- finalized analog PCB - submitted to OSH park - submitted parts order to DigiKey	- None	- continue mcu development	4
11/15/2019	- wiproc pcbs arrived: continuity and shorts check, parts fit check	- ISP connector footprint is for unshrouded connector and planned for shrouded. Will order unshrouded	- start soldering	2
11/16/2019	- started soldering	- USB connector will be difficult	- more soldering	3
11/17/2019	- soldered board except for USB connector - powered up	- USB connector will be difficult	- code development	5

	<ul style="list-style-type: none"> - detected MCU - programmed MCU to light LEDs on button press 			
11/18/2019	<ul style="list-style-type: none"> - soldered USB connectors on - test USB power - 5V LED turned on, MCU works - test battery circuit, looks good - programmed FTDI chip 	- none	- code development	2
11/24/2019	- started soldering IC's to analog board	-none	- solder analog board	3
11/25/2019	- finished soldering IC's and started capacitors	- none	- solder analog board	2
11/26/2019	- finished soldering capacitors	- none	- finish analog board	2
11/27/2019	<ul style="list-style-type: none"> - finished soldering analog board - checked for shorts: safe to turn on - verified PC to MCU UART via FTDI interface - verified PC->PCB_XBEE->Breadboard_XBEE->PC - coded and verified i2c interface by reading ID register from accelerometer 	- received email from OSH park that the vias in my boards may have been drilled too big and there may be opens	- test analog board	7
11/28/2019	<ul style="list-style-type: none"> - created 6DoF, dac, and i2c expander libraries - read accelerometer and temp values - powered up analog synthesizer and verified voltages - verified i2c io expander is written and outputs io, but 1 pin to mux isn't written high - maybe bad solder joint or open circuit from OSH park large drill issue, because other pins are written correctly - measured sine and square waves, triangle doesn't look good 	<ul style="list-style-type: none"> - triangle wave no good - need to add missed resistor in design - don't seem to be writing DAC - mux enable from i2c io expander doesn't get written as a full logic high 	<ul style="list-style-type: none"> - add new resistor - work on setting DAC output - figure out io expander issue 	8
11/29/2019	<ul style="list-style-type: none"> - added jumpers to fix control voltage scaling: now correctly goes from -11.8V to 11.8V - added jumper to hold mux enable high - added new resistor and measured better square wave - fixed DAC output - 2 pins shorted together - measured signal out of filter mux with filter bypass selected - measured signal at output! - read back and displayed all glove sensor data to serial terminal 	<ul style="list-style-type: none"> - LPF and BPF do not output any signal, maybe chip is damaged? - triangle wave is not output 	- build up wiproc board for glove to complete system	6.5
11/30/2019	<ul style="list-style-type: none"> - built 2nd wiproc to test complete system - implemented code to stream sensor from glove to controller wiproc and print out to PC terminal 	<ul style="list-style-type: none"> - sensor data updates mostly with large movements - LPF and BPF do not output any signal, maybe chip is damaged? - triangle wave is not output 	- process sensor data to control analog synthesizer	7.5
12/1/2019	<ul style="list-style-type: none"> - added controls from glove - first tests with sound output - put glove prototype together 	<ul style="list-style-type: none"> - LPF and BPF do not output any signal, maybe chip is damaged? - triangle wave is not output 	- perfect and debug system	6
12/7/2019	- documentation	- none	- finish documentation	2.5
12/8/2019	<ul style="list-style-type: none"> - documentation - code tweak 	- none	- demonstration	2.5

APPENDIX II: LESSONS LEARNED

XBEE3 SPI

It was assumed that the XBee3 could work in transparency mode over the SPI interface just as it does over UART – i.e. it transmits RF data as received over the serial interface. However, this is not true and further inspection of the XBee3 User's Guide SPI communications section specifies that it only supports the API mode over SPI.

So, data needs to be formatted to the API frames for the XBee3 to recognize it. The frame format generally consists of the start command, length bytes, command type, destination addresses, data bytes, and checksum.

Another snag was that all of the frame types are not listed in the user guide, especially the default one used by the XBee3 for receive frames. The user guide specifies a receive packet frame with the type (0x90), however there is another 16-bit address version that the XBee3 was receiving with the value 0x81, which is not listed in the documentation. This was discovered after some time in the XCTU software.

The XBee3 SPI interface with the microcontroller is fully operational now, however it could have been avoided using the ATmega328PB. This version has two built-in UART interfaces (and two SPI), but it does not come in a DIP package and could not have been as easily breadboarded. Using the API frame format is a bit more robust though, as it reduces any communication error by including a checksum. The SPI interface can also be run faster than UART, however it requires about 14 extra bytes per transmit message.

UART INTERFACE FOR DEBUGGING

The serial interface with the MCU was an invaluable feature for code development and debugging. Adding the FTDI USB-UART converter and USB connector made it very easy to connect to the MCU. Getting it up and running first was a good strategy that I will continue to use on future projects where applicable.

I used this interface to print sensor data to the PC terminal to ensure that it was being measured and received correctly. I also used print statements to determine where the processor would hang up.

IN SYSTEM PROGRAMMING HEADER

Adding the mating header to the microcontroller programmer made reprogramming it very easy. It took up board space and might be excluded from the glove's final design, but at the prototyping stage, it was worth it.