

Sponsored by:



# Hybrid Synthesizer

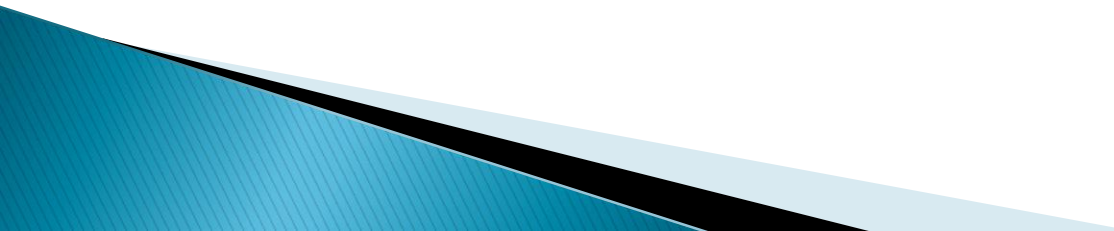
Group 32

Marlena Brammer (CpE)

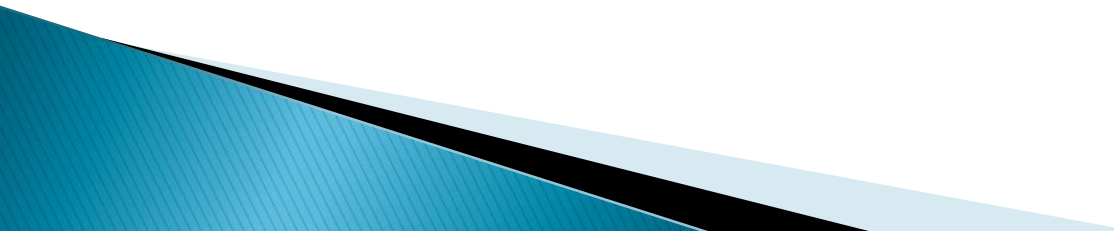
Josef Hirmann (EE)

Michael Smith (EE)

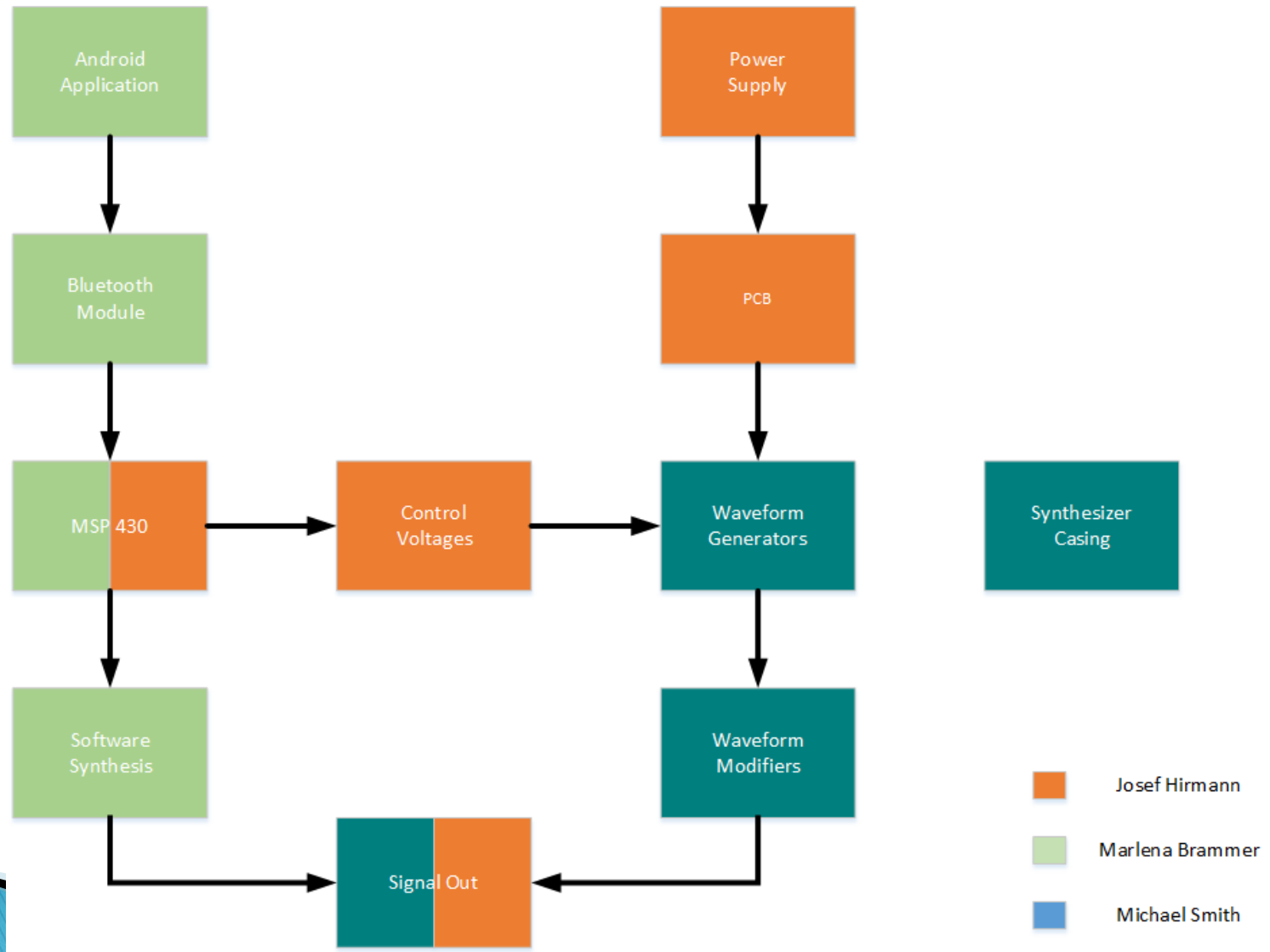
# Goals & Objectives

- ▶ To provide a portable, robust synthesizer for travelling musicians that can be used both in the studio and for live performance
  - ▶ Should have both traditional analog waveforms as well as simulated orchestral instruments for versatility
  - ▶ Should be affordable to produce to prevent the price barrier seen for most modern synths and allow greater accessibility for young artists
- 

# Specifications

- ▶ Needs to be light, ideally five pounds or less
  - ▶ Must be at least monophonic for at least two octaves in the analog design
  - ▶ Must support at least 3 waveforms in the analog design that generate a consistent, reliable signal
  - ▶ Must support at least 3 common orchestral instruments in software synthesis (piano, flute, clarinet, etc)
  - ▶ Must support connecting to an android phone with custom keyboard application
  - ▶ Must consume relatively low power
- 

# Project Block Diagram



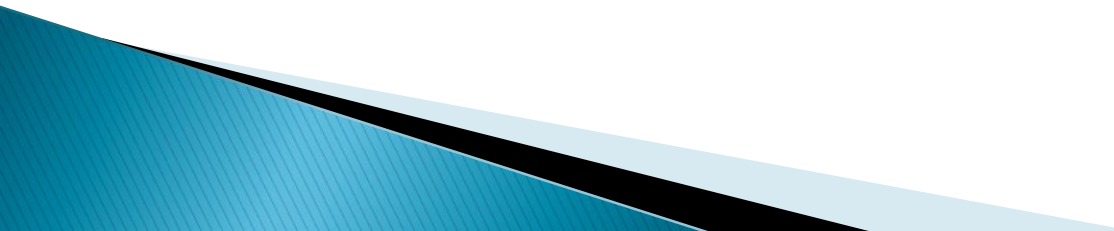
# Android application



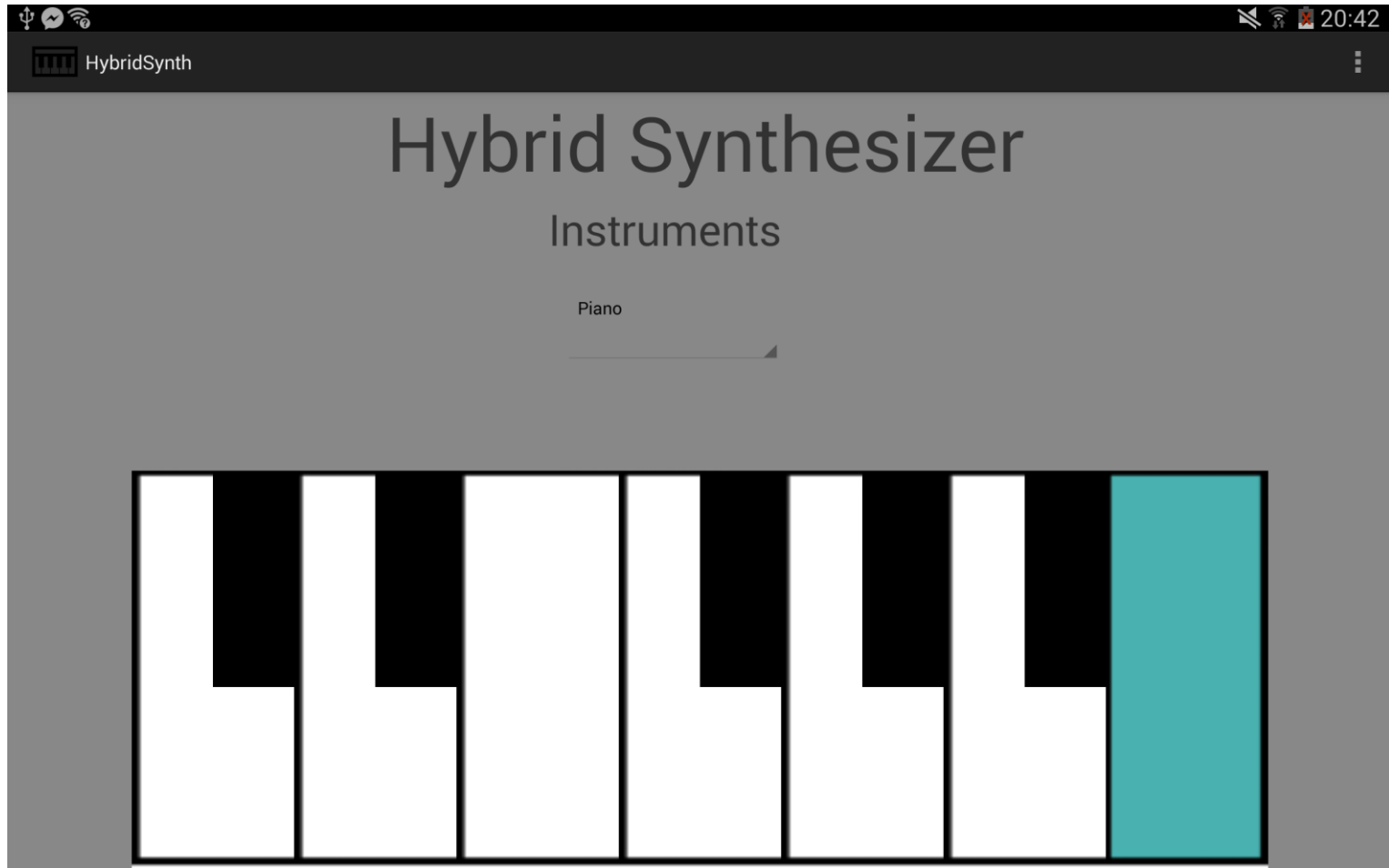
- ▶ Development Environment:
  - Eclipse for Android
  - Using API level 19 (Kit Kat)
  - We chose Android because it is Open Source and essentially free

	Language	Devices Readily Available ?	Familiarity	Cost to Develop
Android	Java/XML	Yes	High	Free
iOs	Obj-C	No	Low	\$99 + hardware
Windows Phone	XAML, C# or Visual Basic	No	Low	Free

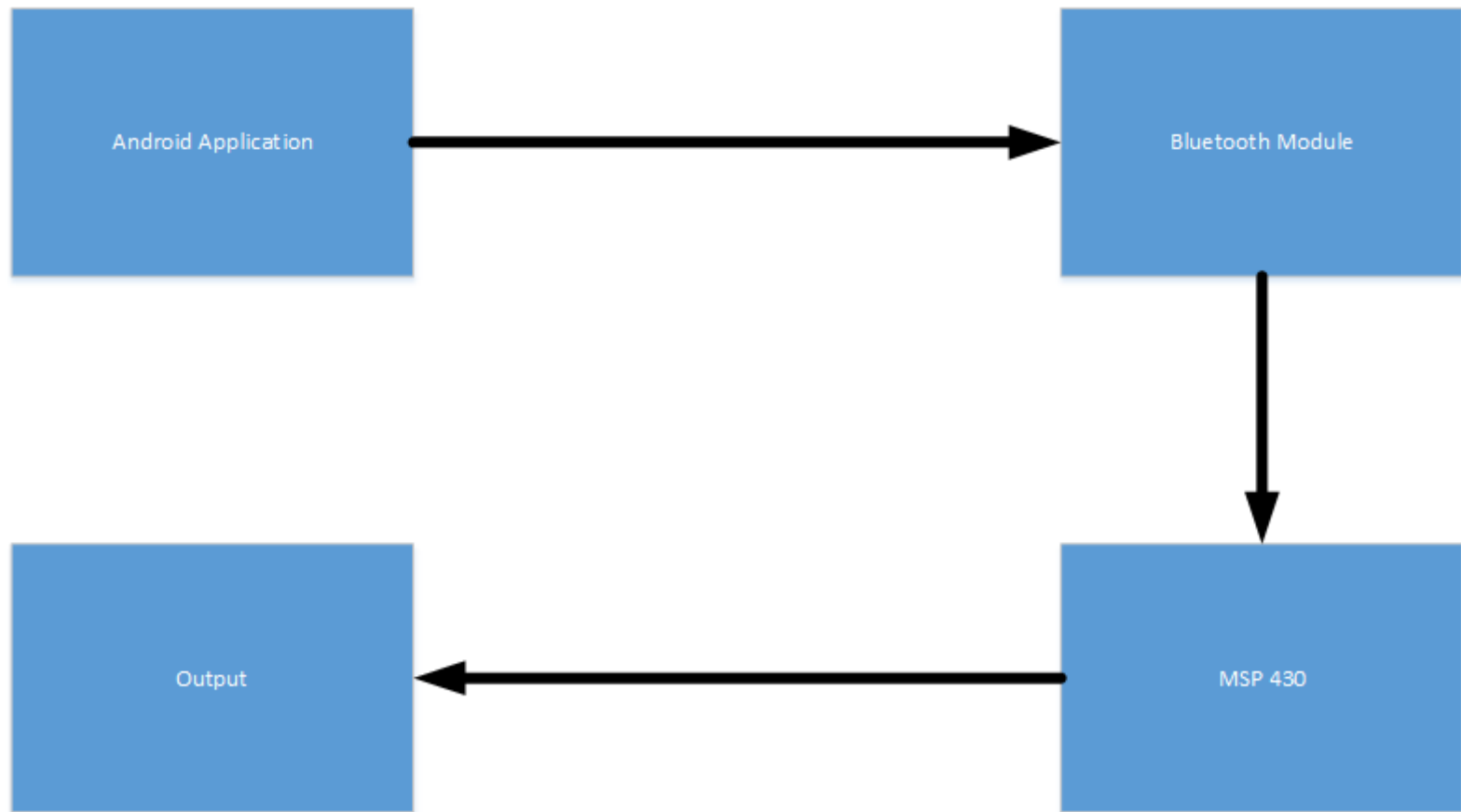
# Application Interface

- ▶ The Main Activity will look similarly to a Piano, each key press will generate the corresponding note
  - ▶ The drop down menu will allow the user to choose an instrument
  - ▶ The Instrument name and note will be sent to the Bluetooth Module as an array of characters
  - ▶ The key being pressed will change the key color
- 

# Application Interface



# Flow of information





# Android Application

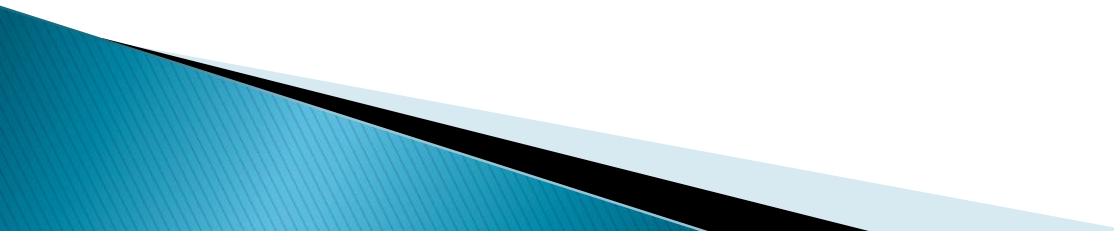
- ▶ At the start of the application, it will check to see if Bluetooth on the device is enabled
  - If so, it will connect to the MAC address of the Bluetooth Module by looping through all paired devices
- ▶ It will create a socket and use the `getOutputStream();` and `write byte[]` methods to send information to the module
- ▶ Since we are not listening for input from the Bluetooth Module, we don't need to enable read functions
- ▶ The MSP430 will receive this info from the Bluetooth module via UART

# Bluetooth Communication

- ▶ Using the TI CC2564 as the Bluetooth Module
- ▶ Will use the TI Bluetooth Stack
- ▶ It will be used in conjunction with the MSP430
- ▶ It is Bluetooth Specification v4.1
- ▶ Has dual mode capabilities, meaning that it can use Bluetooth Classic and Bluetooth Low Energy



# Bluetooth Connection

- ▶ This module will receive characters from the Android App and send it to the MSP430
  - ▶ Connected using the RX and TX pins
  - ▶ The instrument names will be set up as case statements
  - ▶ If we are sending the note frequency to the analog circuit, we will send it through a lookup table
  - ▶ If we are generating via software synthesis, we will choose the sine harmonics that correspond to the specific instrument
  - ▶ When the user has finished pressing the note, it will stop the flow of information and cause an interrupt
- 

# MSP430 Software Synthesis

- Generally, a pure tone is not all that we hear, we actually hear many frequencies that have higher pitches
- These are called the Harmonics of the note, and mathematically speaking, they are multiples that occur on top of the fundamental frequency

Ex:

$$\text{tone harmonic} = A1 * \cos(2 * \pi * f * t) + A2 * \cos(2 * \pi * f * t) + \dots An * \cos(2 * \pi * f * t)$$

# MSP430 Software Synthesis

- ▶ **Step 1: Note Generation**

- ▶  $f_n = f_0 * (2)^{n/12}$

where

- ▶  $f_0$  = the frequency of one reference note which must be defined.
- ▶  $n$  = the number of half steps away from the fixed note you are. This number can be negative or positive, depending on reference note.
- ▶  $f_n$  = the frequency of the note  $n$  half steps away.

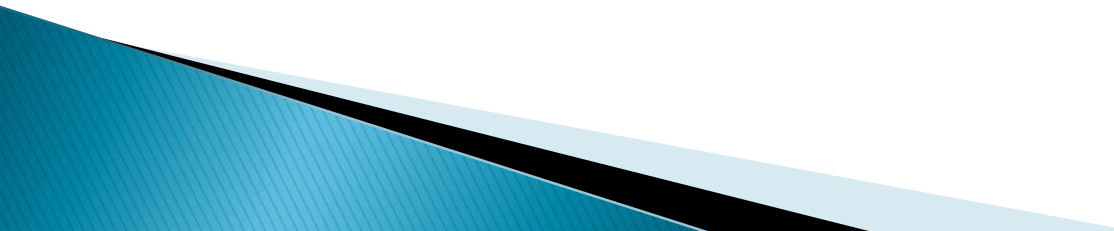
# Example

- ▶ To Calculate the frequency of Middle C
- ▶  $A_4 = 440\text{Hz}$  as  $f_0$
- ▶ Using,  $f_n = f_0 * (2)^{n/12}$
- ▶ Middle C is 9 half steps below  $A_4$
- ▶  $F_{-9} = 440 * (2)^{-9/12} = 261.63 \text{ Hz}$

# Table of Frequencies

Note	Frequency
C (Middle)	261.63
C#/D	277.18
D	293.66
Eflat/D#	311.13
E	329.63
F	349.23
F#/Gflat	369.99
G	392.00
Aflat/G#	415.30
A	440.00

# MSP430 Software Synthesis

- ▶ **Step 2: Envelope Generation**
  - ▶ This is where we simulate the instrument sound
  - ▶ The note and various harmonic values will be passed through an additional sine function that will create an array of values that we will loop through on output
- 



# Microcontroller – MSP430

**MSP430F5529**

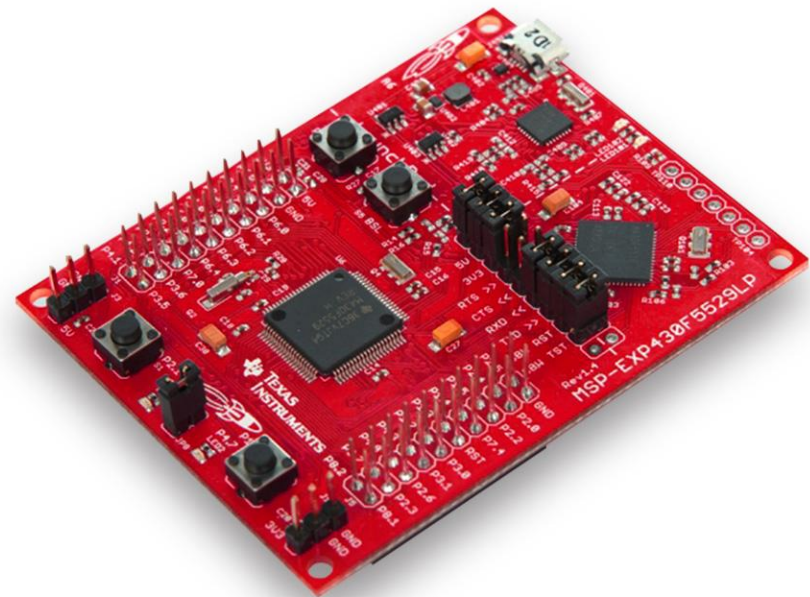
128KB Flash and 8KB RAM

Up to 25 MHz

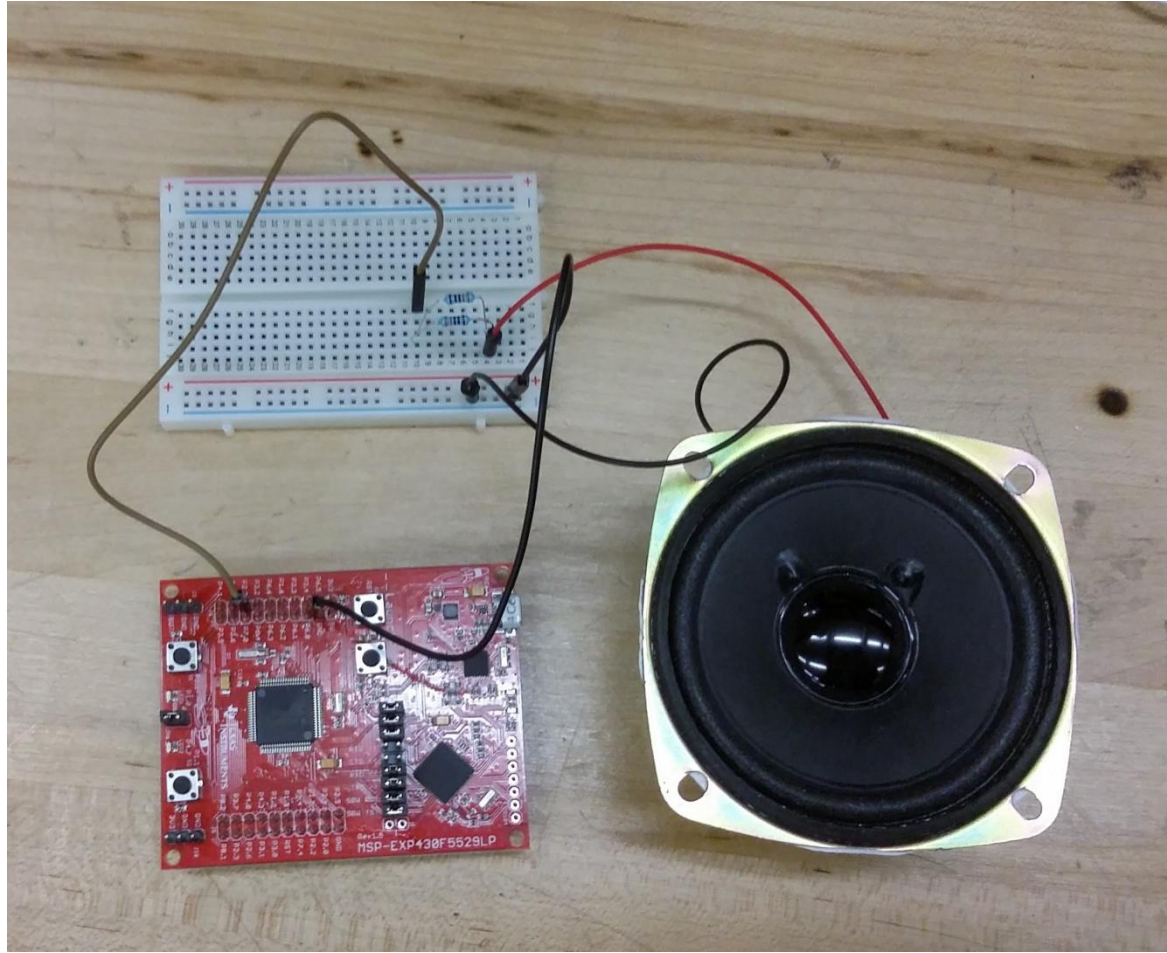
UART, PWM

5–3.3V

C, Assembly



# MSP 430 – Software Synthesis Prototype



# Hardware Synthesis – CV

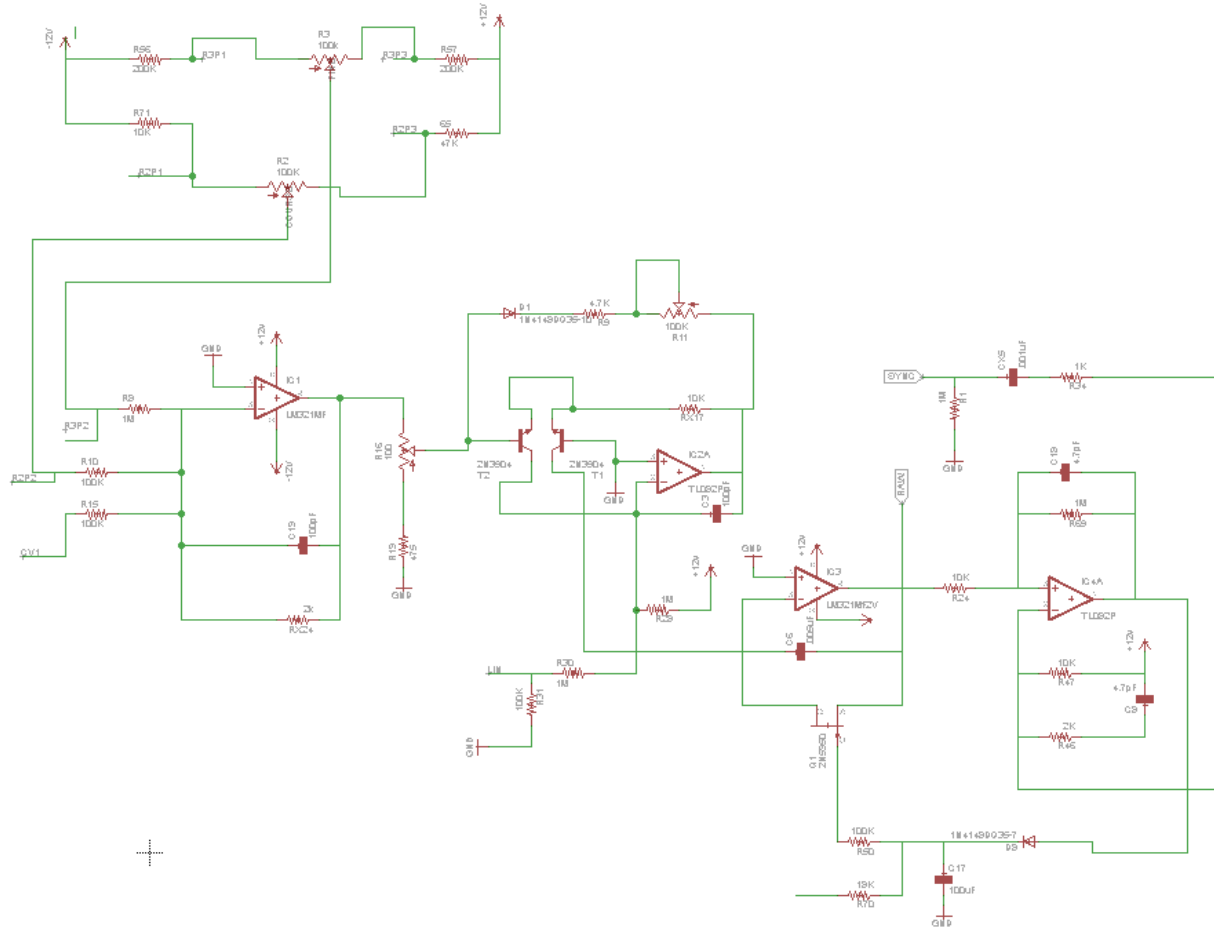
- ▶ If the user selects “analog waveforms”, the MSP430 will switch into CV/Gate Mode
- ▶ The MSP430 will use a hard-coded table to convert the frequency value received from the Android Application into a DC control voltage signal that is sent as an input to the hardware oscillators

# CV/Gate Voltage Examples

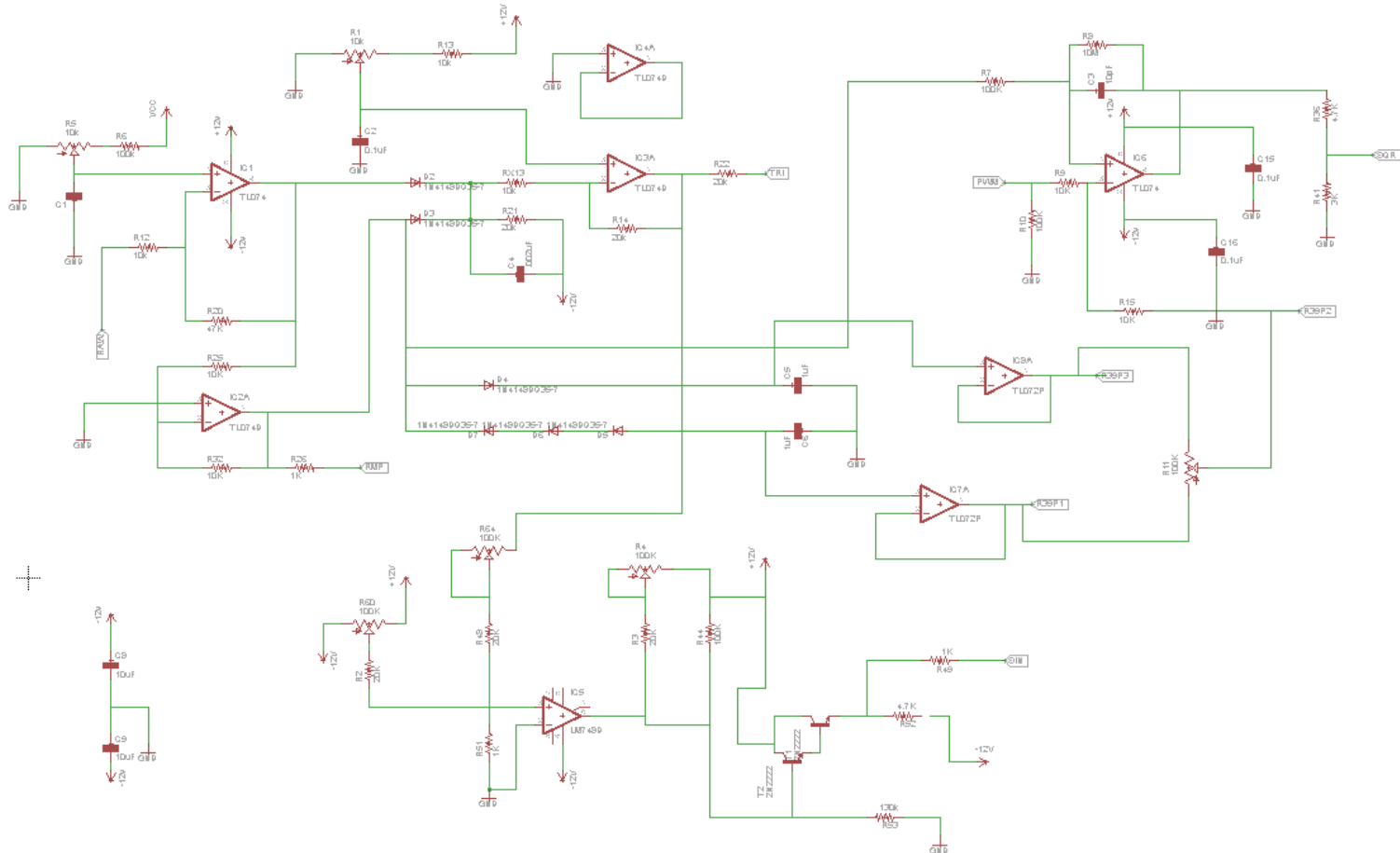
Note	A1	A2	A3	B3	C4	D4	E4	A4	A5
Volts per octave (V)	1	2	3	3.167	3.25	3.417	3.583	4	5
Frequency (Hz)	55	110	220	247	261	294	330	440	880

- ▶ The oscillators follow a 1V/Octave linear scaling system, meaning each half step will require a rise in voltage of 0.083V

# Oscillator Schematic Page 1

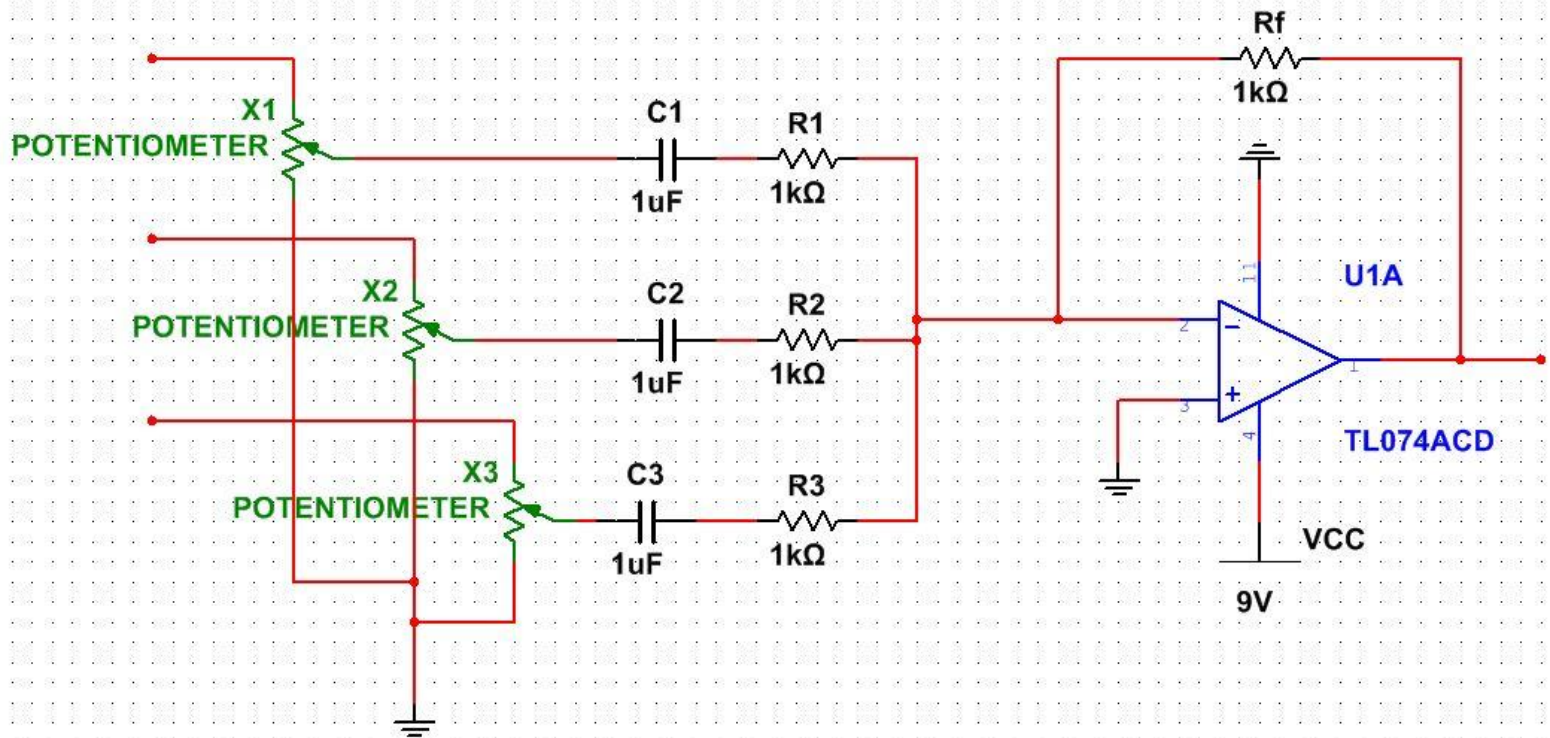


# Oscillator Schematic Page 2



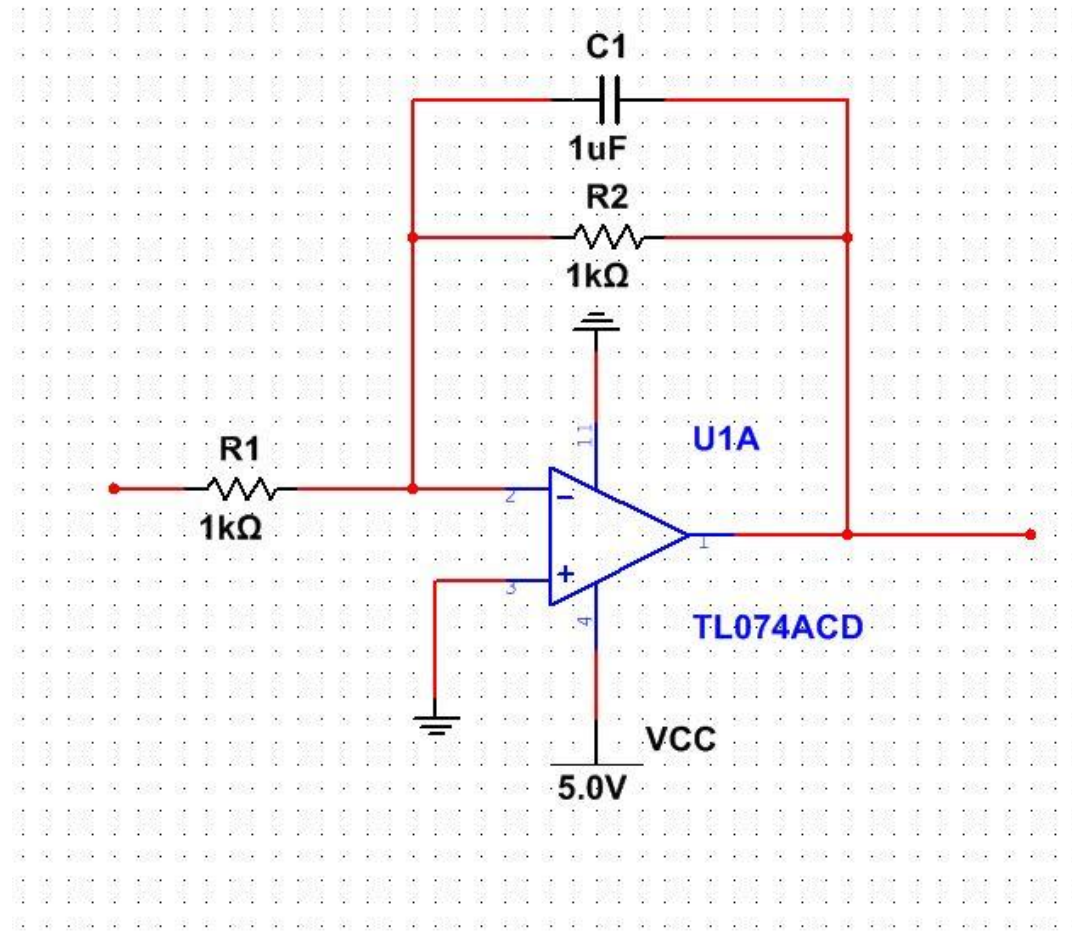


# Mixer Schematic



$$-V_{out} = \frac{R_f}{R_{IN}} (V_1 + V_2 + V_3 + \dots + V_N)$$

# Filter Schematic – Low Pass

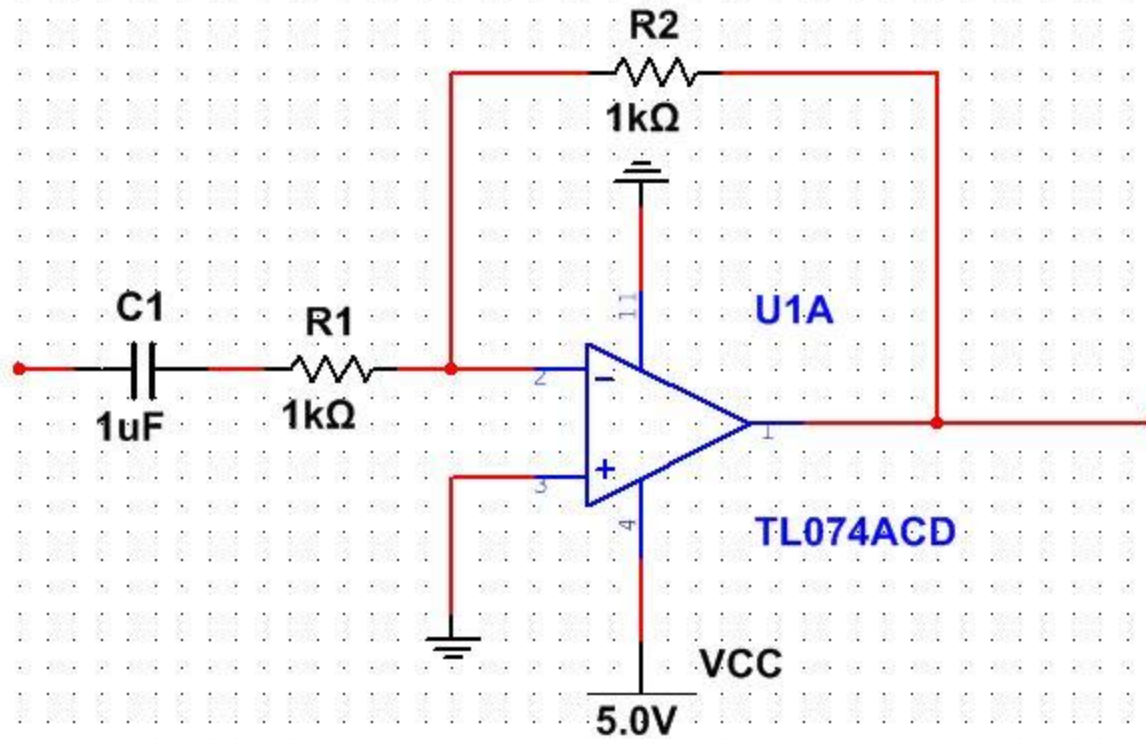


$$f_c = \frac{1}{2\pi R_2 C}$$

$$A_v = -\frac{R_2}{R_1}$$



# Filter Schematic – High Pass



$$f_c = \frac{1}{2\pi R_1 C}$$

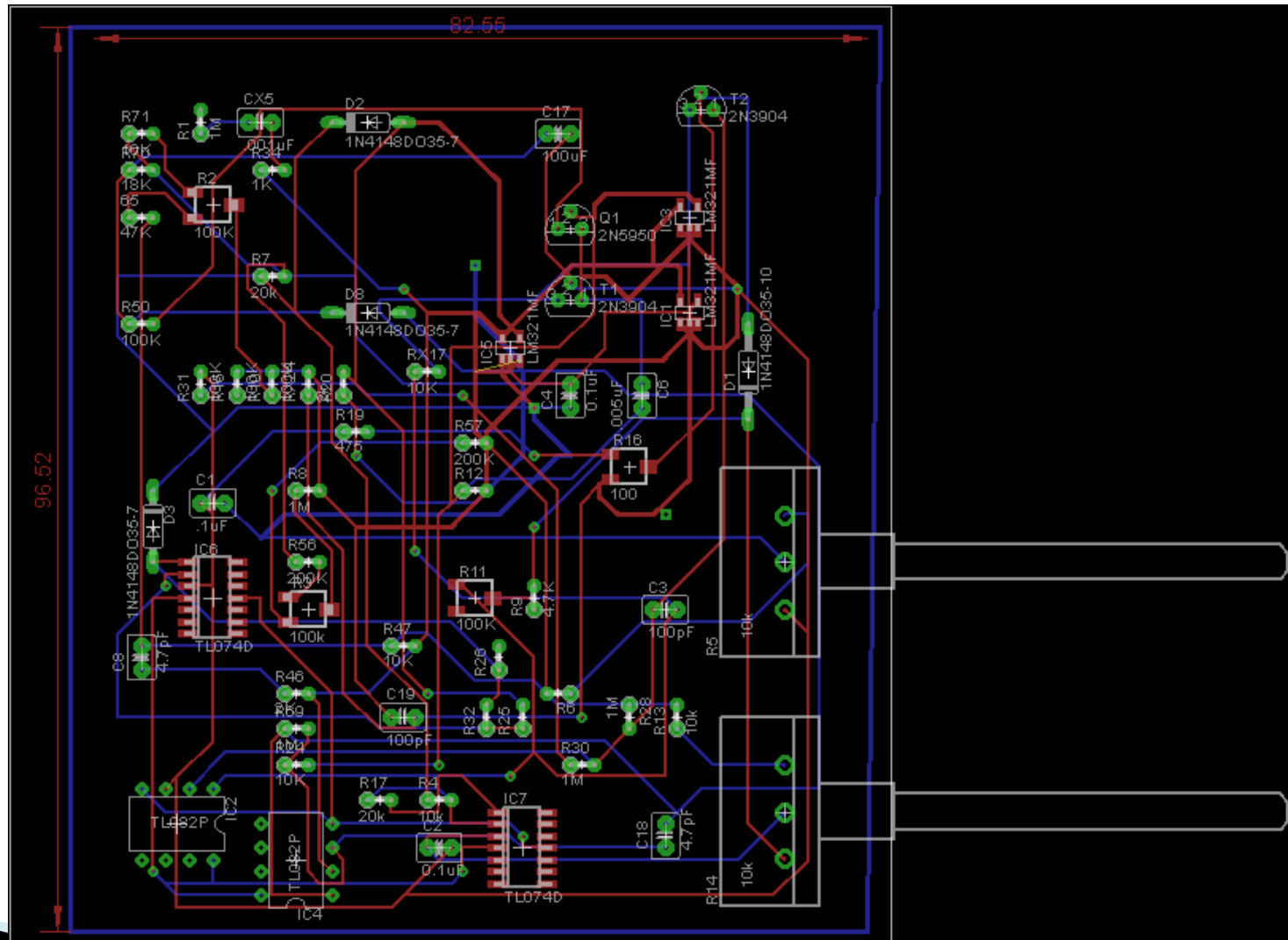
$$A_v = -\frac{R_2}{R_1}$$

# TL07X JFET OP-AMPS

- ▶ Low noise family of general-purpose JFET op-amps
- ▶ Can operate comfortably at our desired voltage rails of  $\pm 12\text{V}$
- ▶ Low heat output, low distortion
- ▶ Easy to replace if burnt out or if new revision of part is developed



# PCB – Oscillator



# Power Supply

## ▶ Dual Power Supply

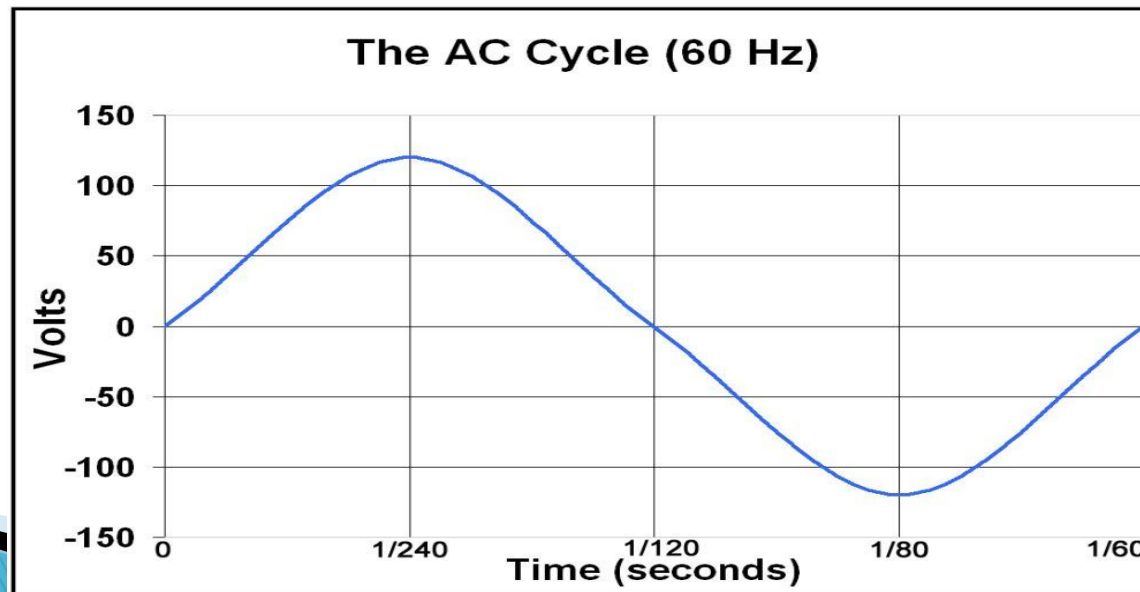
- Wall Wart Power (9, 12, or 15 VAC options)
- Goal is a 12 Volt regulation
- Supports plenty of capacitance for a clean supply
- Low part count
- Convenient source of bipolar voltage from the wall.

## MSP430F5529

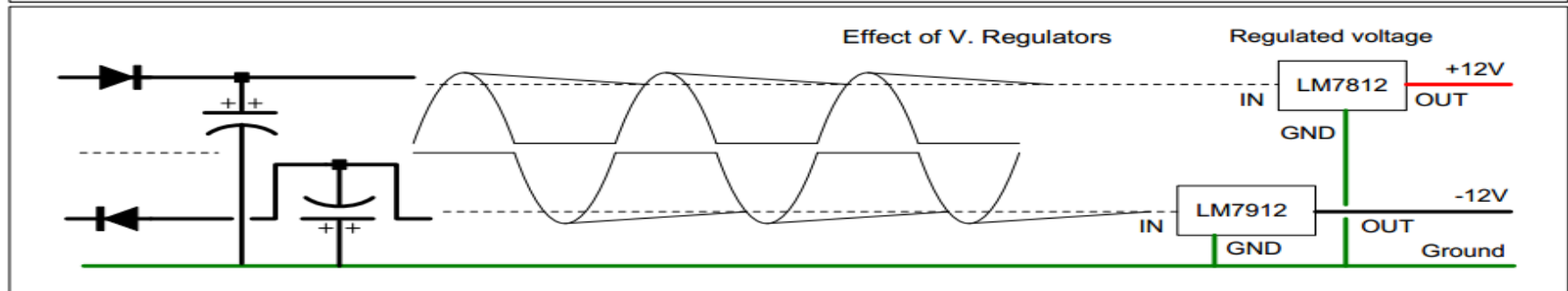
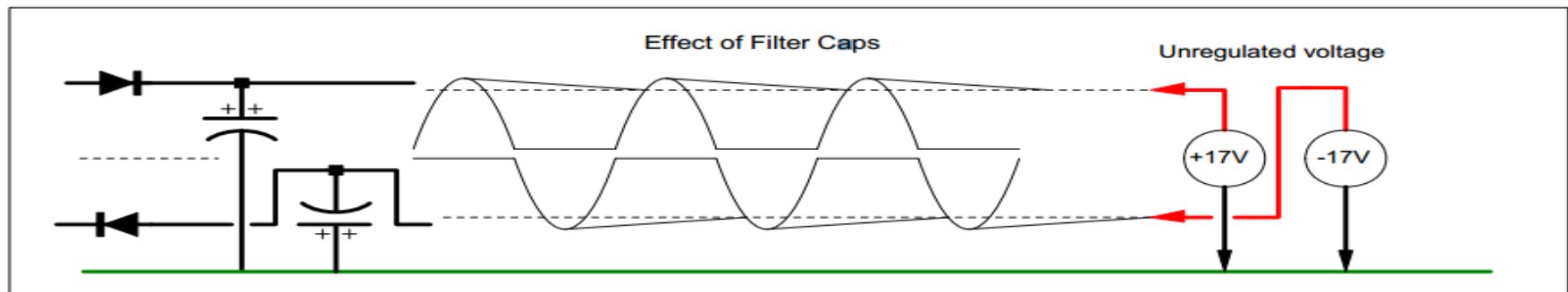
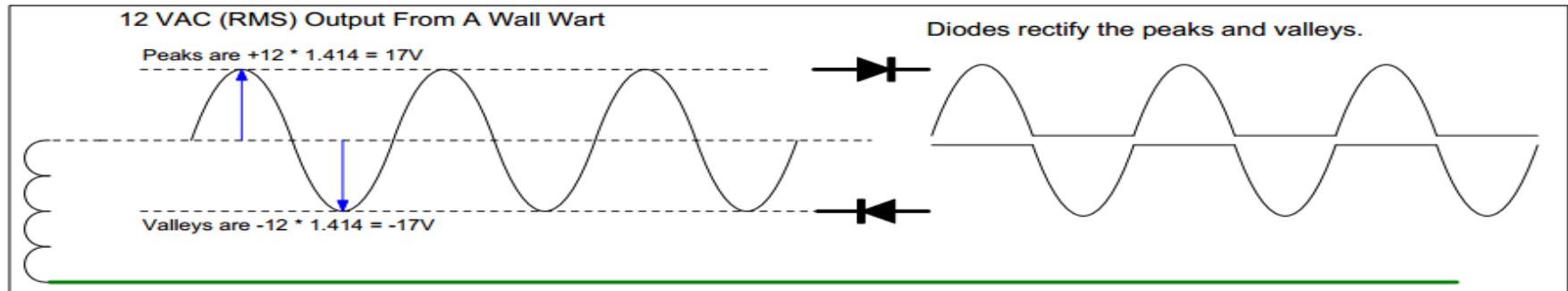
- Chain of regulators to power the MSP430 while mobile
- Possible to use the Lithium Ion Battery
- Currently using laptops and computers to power MSP

# Wall Wart Voltage Example

- ▶ If the Wall Wart outputs 12VAC, the VAC value of a transformer is an RMS value.
- ▶ So, 15VAC gives about 21.21V output. 12VAC gives about 17V.
- ▶  $12\text{VAC} * 1.414\text{V} = 16.968\text{V}$

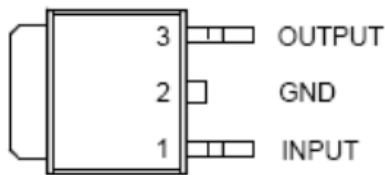


# PSU Signal Flow Theory

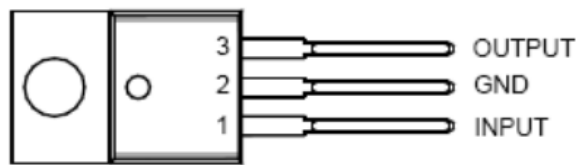


# Voltage Regulators: LM78XX or LM79XX

## Pin Configuration



Outline: D  
D-PACK  
(TO-252)



Outline: T  
TO-220

## Regulators (left)

- High Efficiency Linear Regulators
- Post Regulation for Switching Supply
- Microprocessor Power Supply



- **Wall Wart (right)**
- 12VAC @ 1000mA or 500mA
- AC to AC adaptor
- Cheap in cost

# Major Power Supply Components

## LMXX Regulators

- ▶ 1000mA current

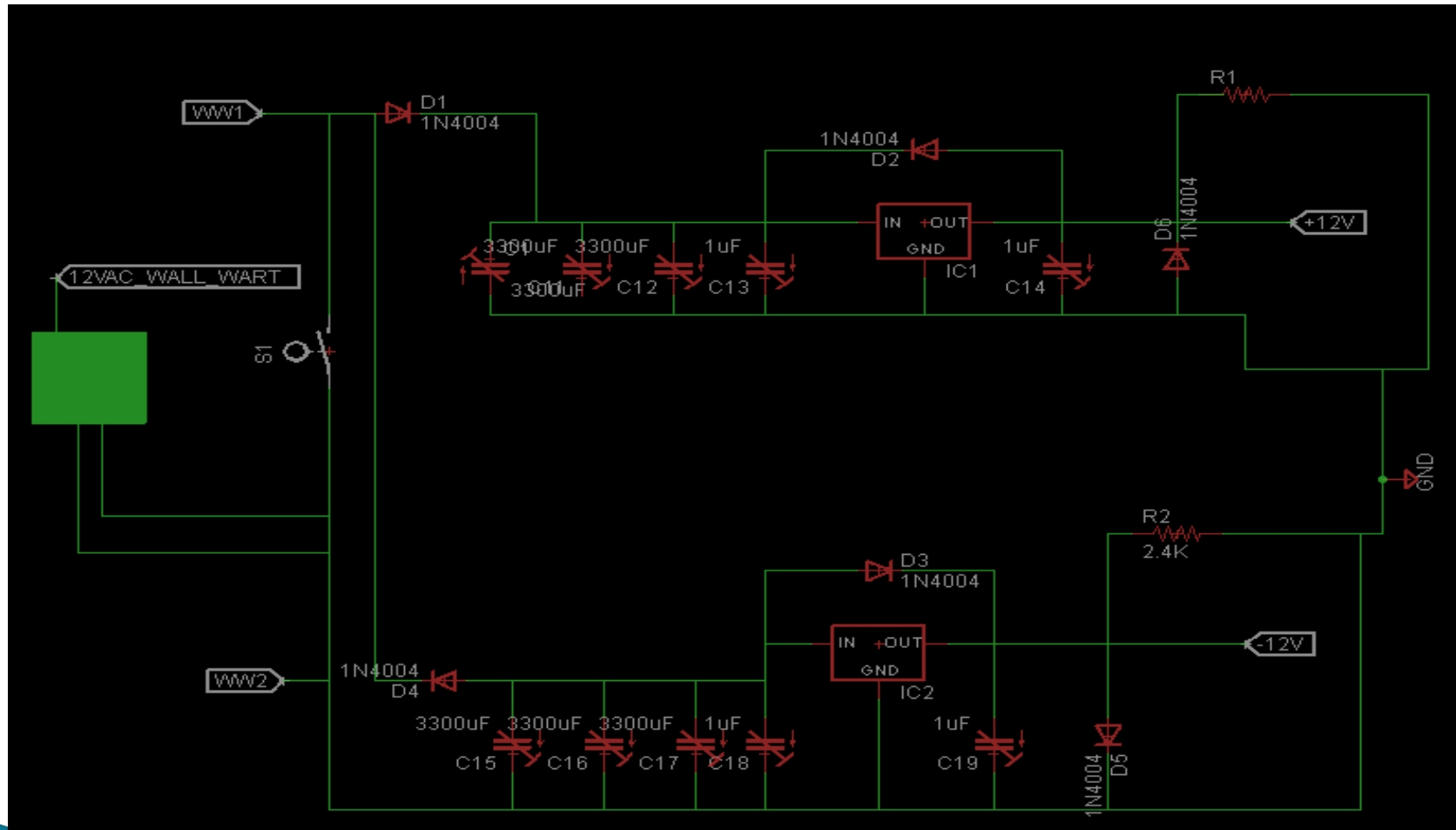
## LMLXX Regulators

- ▶ 100mA currents

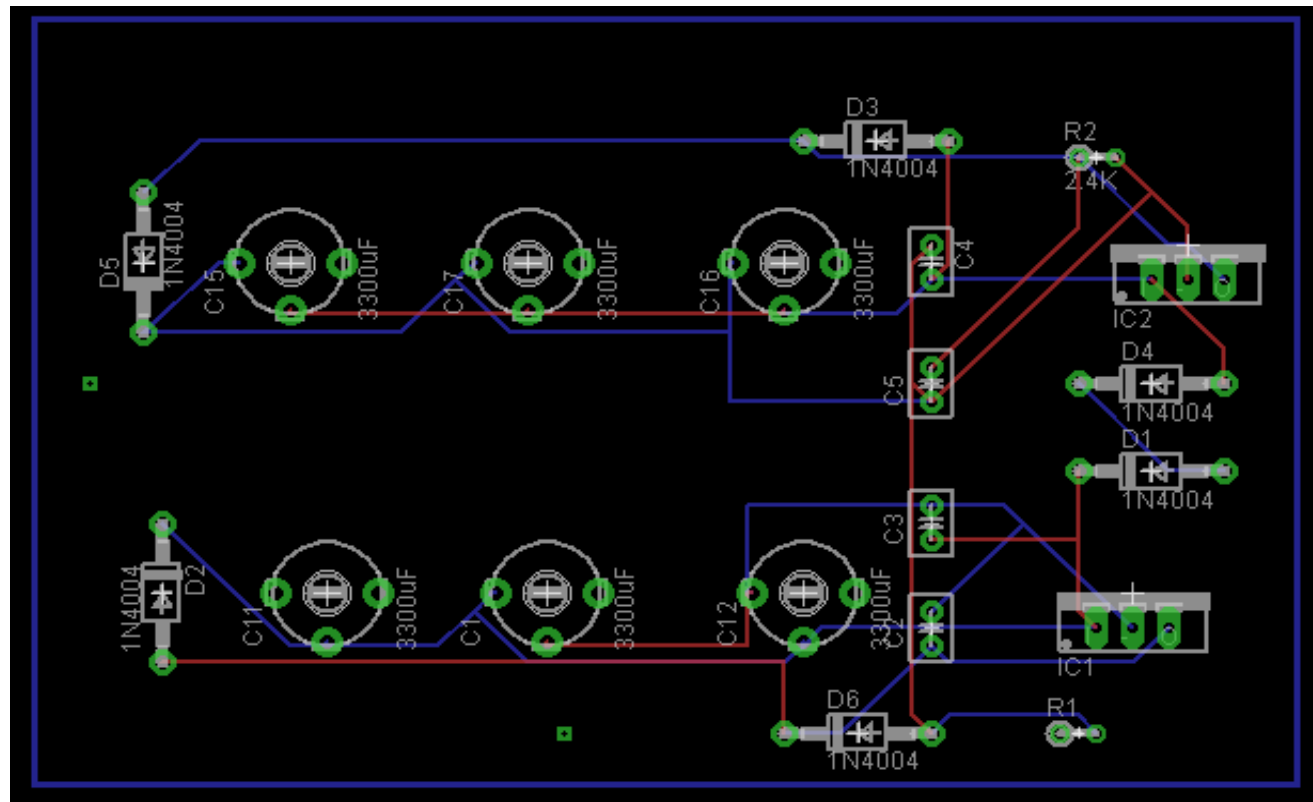
Desired Voltages	Higher Currents U1/U2	Lower Current U1/U2	Wall Wart Output
+/-9V	LM7809/LM7909	LM78L09/LM79L09	9 VAC Output
+/-12V	LM7812/LM7912	LM78L12/LM79L12	12 VAC Output
+/-15V	LM7815/LM7915	LM78L15/LM79L15	15 VAC Output



# PSU Eagle Schematic



# Power Supply PCB



# Budget

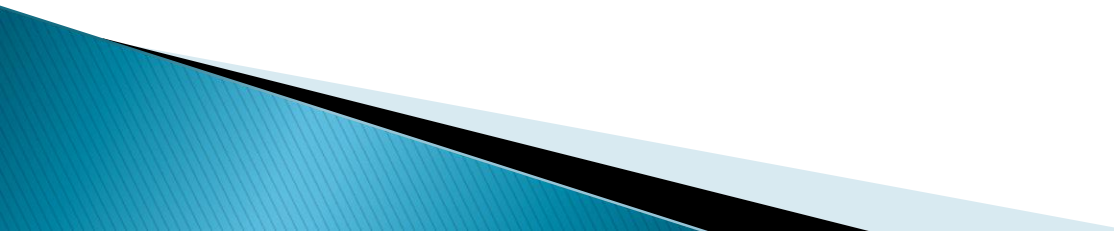
TI Coupon: \$200

Boeing Funding: \$502.80

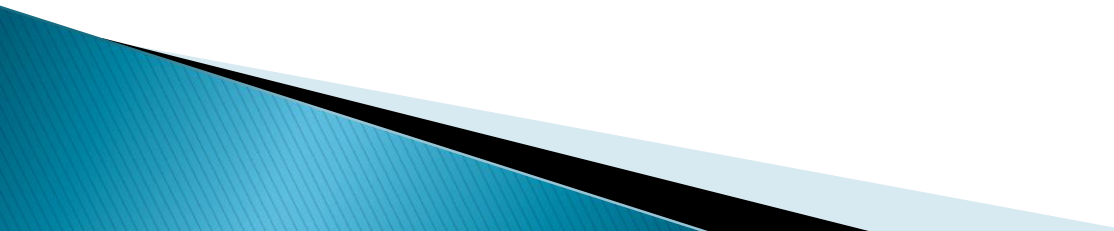
Total Remaining: \$97.38

PCB	140.85
Resistors	19.65
Casing	25.00
Jumper Wire	10.00
MSP430	0.00
AudioSink	0.00
TICC2564	0.00
5V Step-Up Breakout	15.91
Regulators	1.99
PCB Mounting	80.00
HC-05	8.00
Oscillator PCB Parts	48.00
Samsung Tablet	0.00
Software	0.00
12VAC Output	8.95
Op-Amps	7.74
Diode	1.64
Capacitors	10.69
Potentiometers	12.00
JFET	2.00
Transistors	13.00
Total	405.42

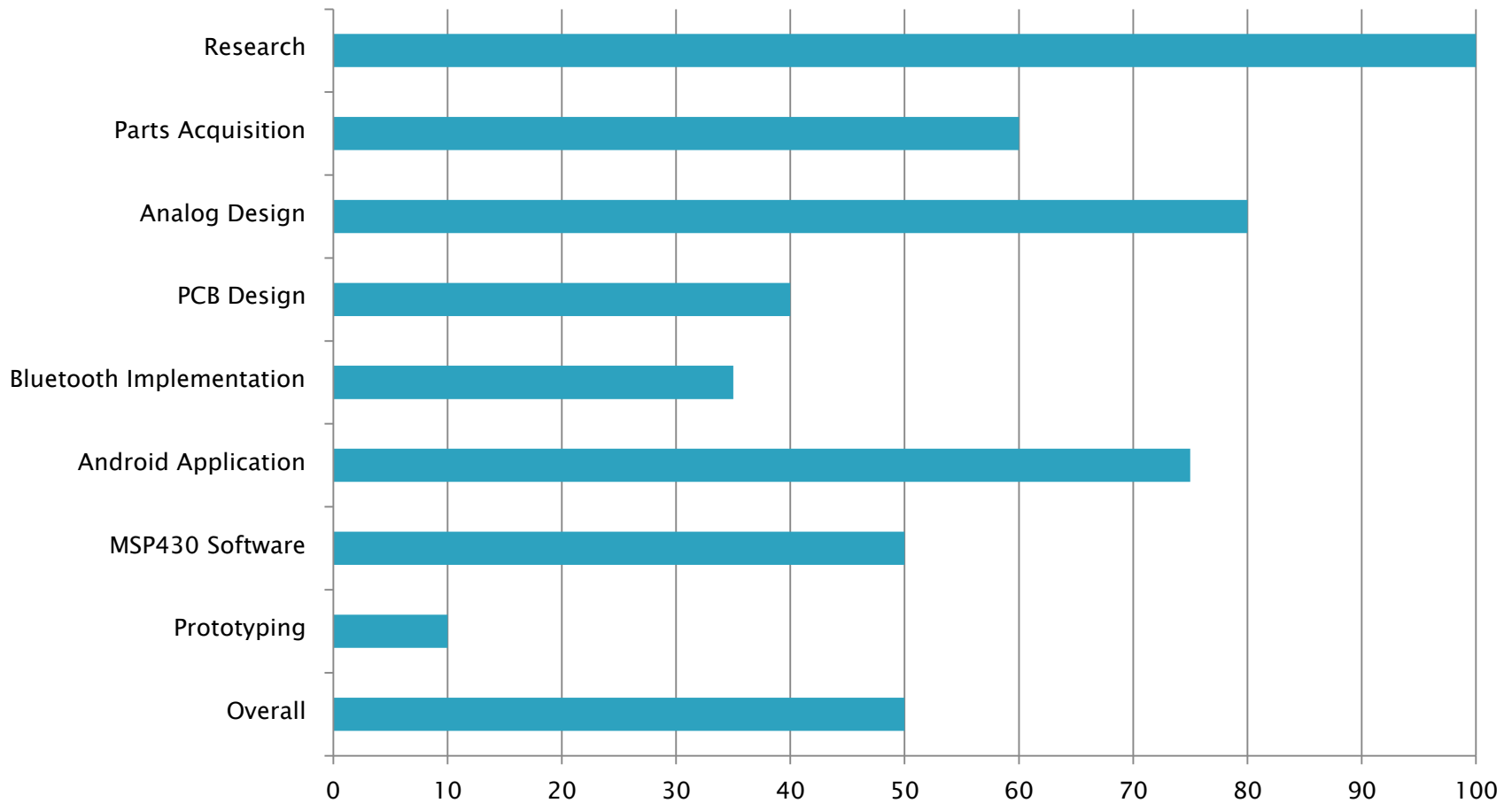
# Successes

- ▶ Oscillator design is robust and should track well between 6 to 8 octaves
  - ▶ Even with modules, current draw will not be significant, resulting in a need of 250 mA or less to power the entire circuit
  - ▶ Hobbyist network online ensures rare/unusual parts can still be purchased from individuals if they have been discontinued
- 

# Challenges

- ▶ Numerous hardware design revisions
  - ▶ Not sure if the MSP430 will be able to output precise control voltages down to the tens of millivolts, could result in out of tune notes
  - ▶ Changes in software synthesis implementation required updated MSP430 Boards to be purchased
- 

# Percent Completion



# Questions?

