Sponsored by:

Hybrid Synthesizer

Group 32
Marlena Brammer (CpE)
Josef Hirmann (EE)
Michael Smith (EE)

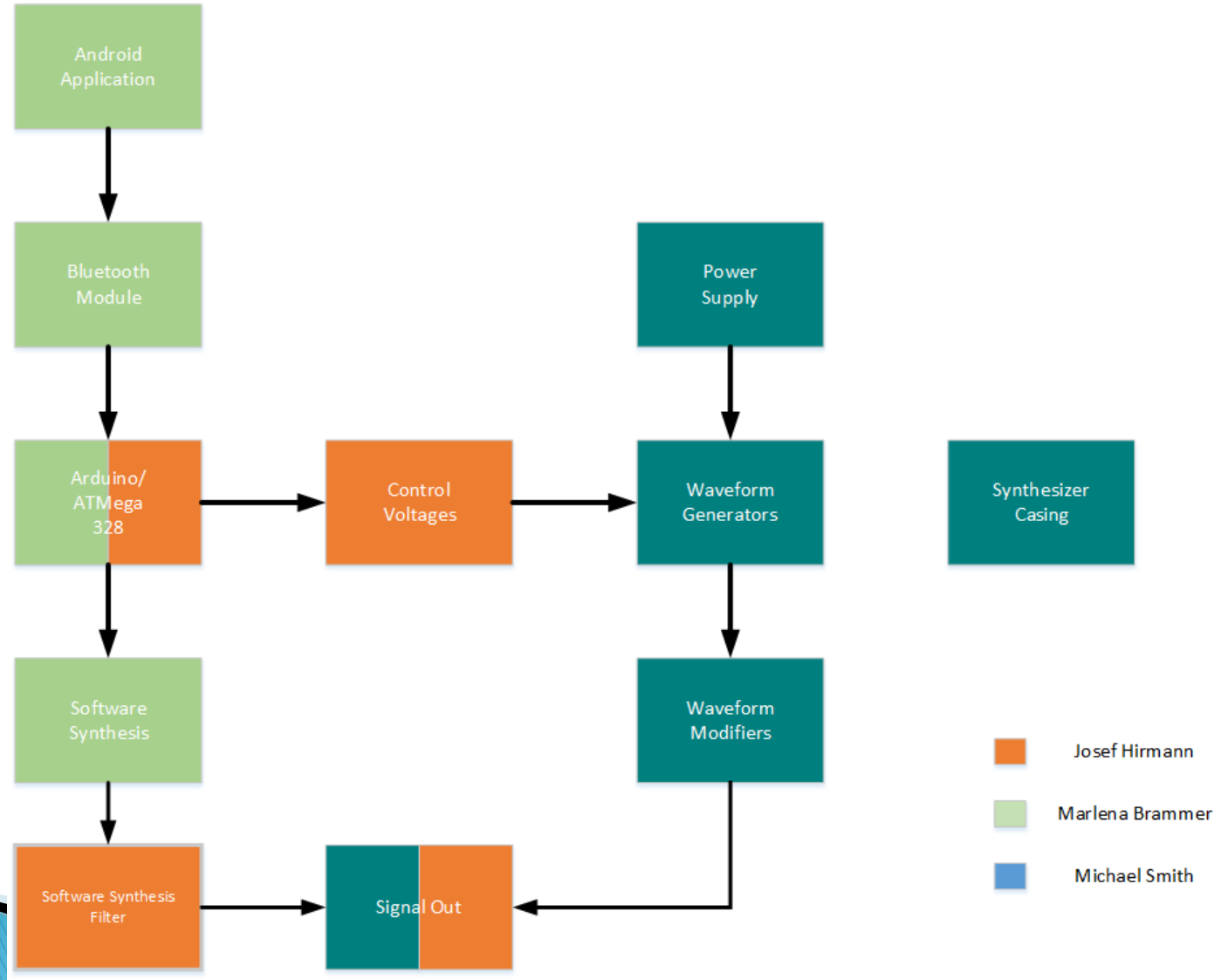
Goals & Objectives

- 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

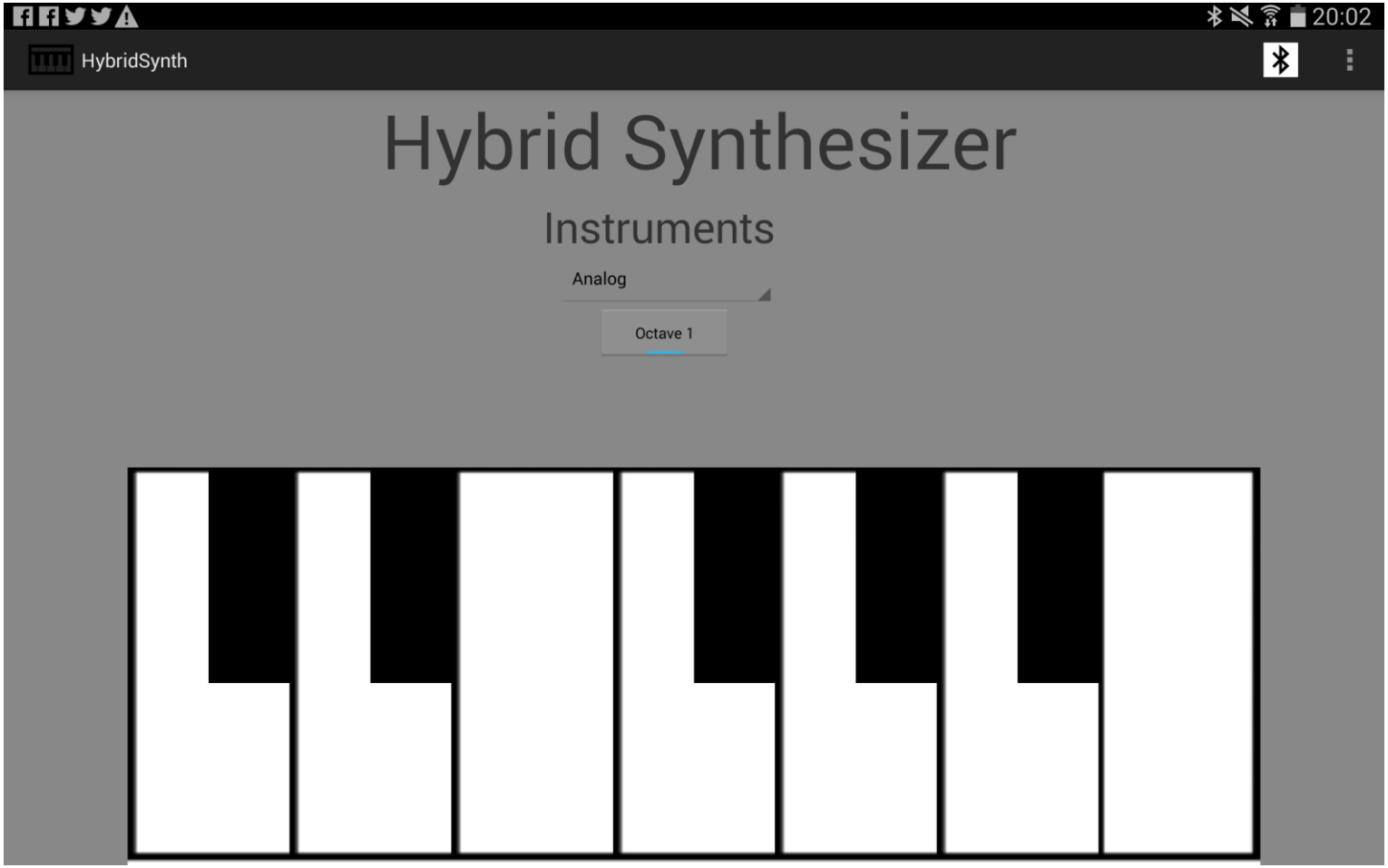
- Development Environment:
  - Eclipse for Android
  - Using API level 19 (Kit Kat)
  - We chose Android because it is Open Source and essentially free

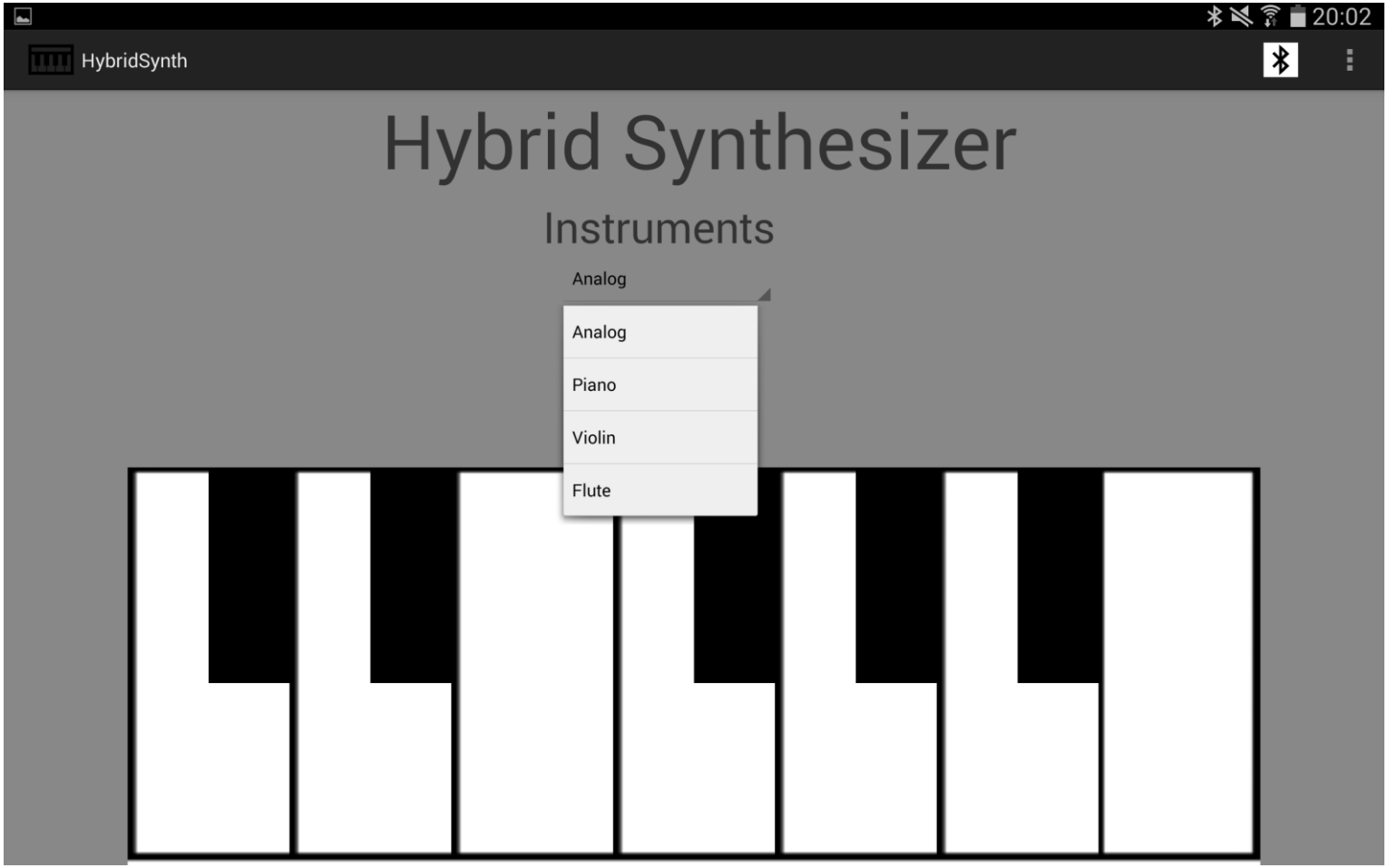
	Language	Devices Readily Available ?	Familiarity	Cost to Develop
Android	Java/XML	Yes	High	Free
iOs	Obj-C	No	Low	$99 + hardware
Windows Phone	XAML, C# or Visual Basic	No	Low	Free

Application Interface

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

Application Interface




20:02
HybridSynth
Hybrid Synthesizer
Instruments
Analog
Analog
Piano
Violin
Flute

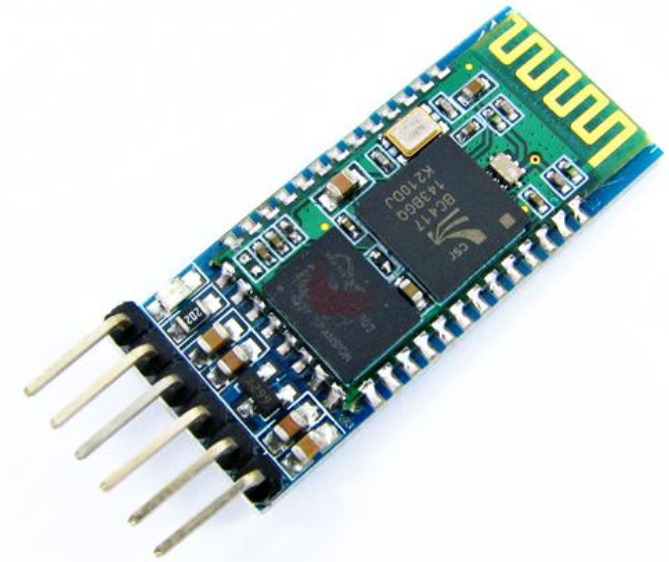
Android Application

- 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 to the Android Application we don't need to enable read functions
- The ATMega328 will receive these bytes using a UART connection to the Bluetooth Module.

Bluetooth Communication

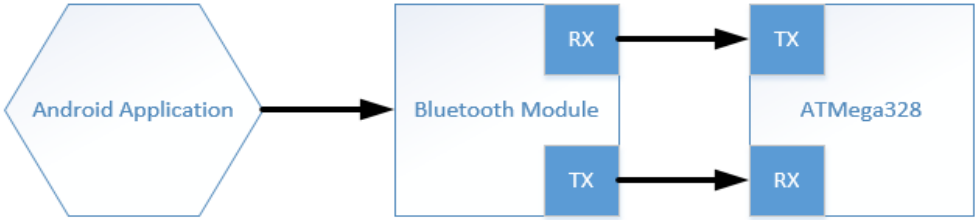
- Using the HC-05 as the Bluetooth Module
- With a Baud Rate of 9600
- In slave mode

Specifications:
•Bluetooth Specification v 2.0
•Range of up to 30m
•3.3vDC Power Input
•Baud Rate Adjustment



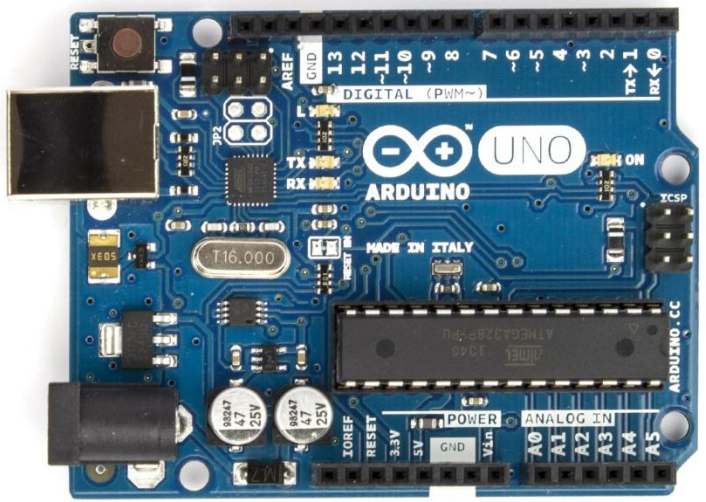
Bluetooth Connection

- This module will receive characters from the Android App and send it to the HC-05
- 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

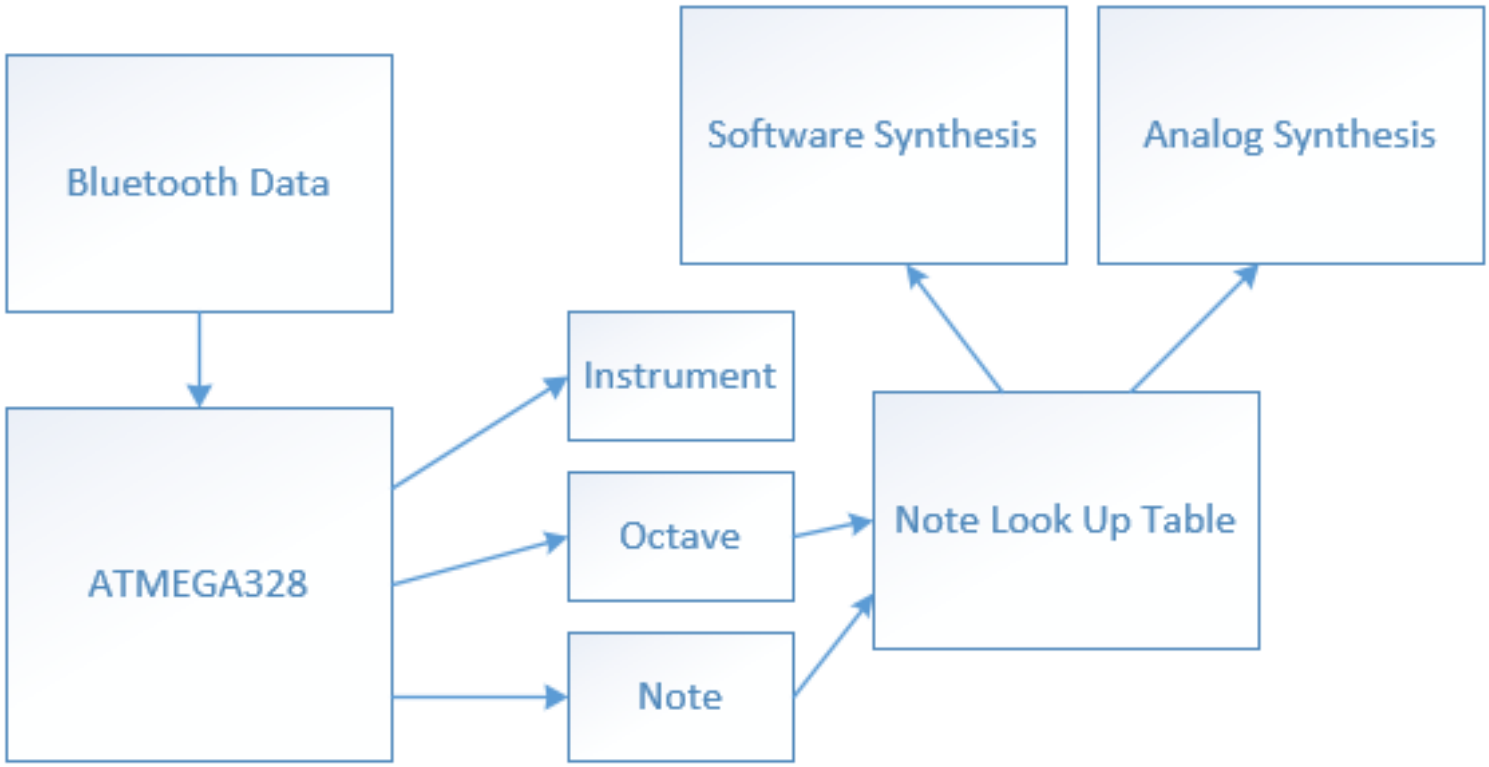


Microcontroller – Arduino/ ATMega328p

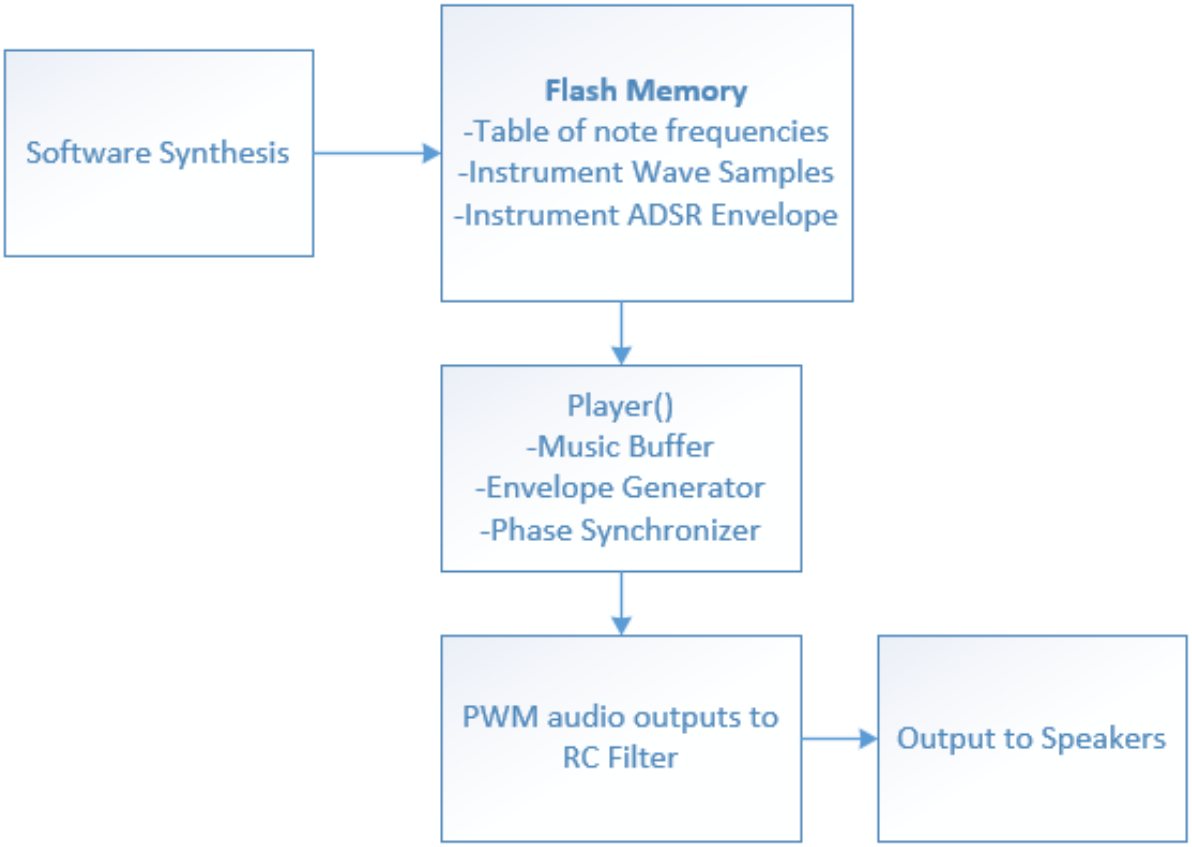
32KB of Flash Memory
16MHz Clock Speed
14 Digital I/O Pins (6 PWM)
5v operating voltage
Programmed using C



Arduino Synthesis Signal Flow



Software Synthesis



Software Synthesis

Step 1: Note Generation using harmonics

Step 2: Instrument Wave Table

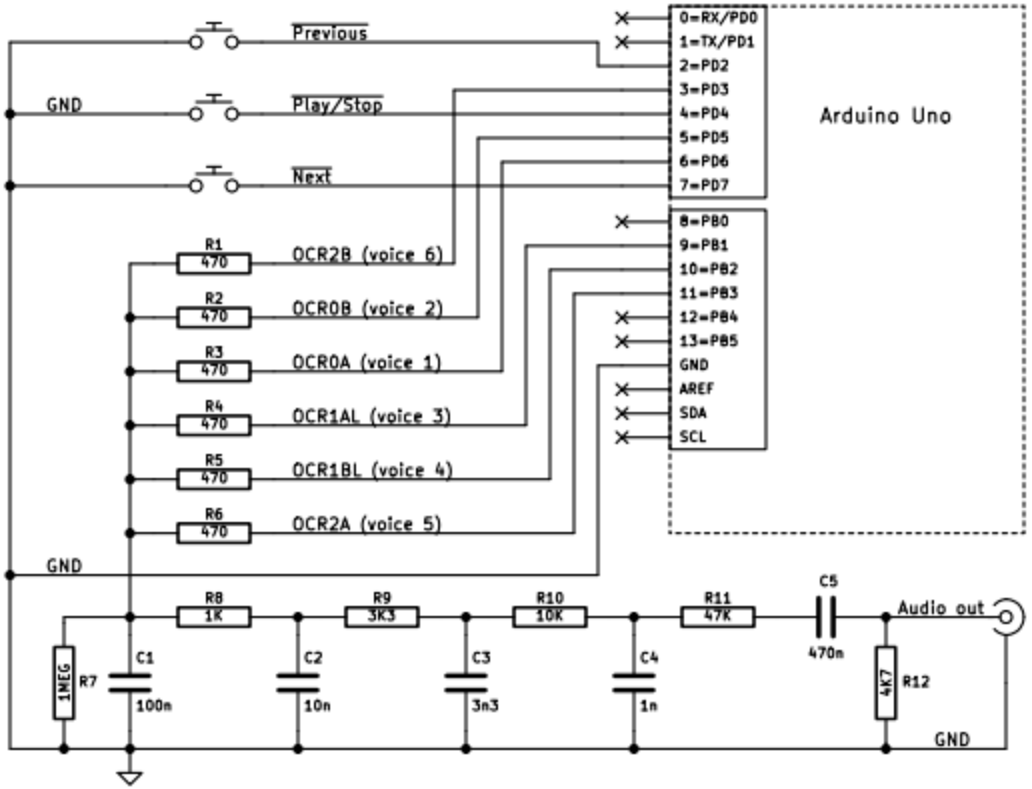
Step 3: ADSR Envelope

```
const Instrument instrum_violin PROGMEM __attribute__ ((aligned (256))) = {
    .wave = {
        0x00, 0x14, 0x28, 0x3b, 0x4b, 0x59, 0x65, 0x6f, 0x76, 0x7b,
        0x7e, 0x7f, 0x7e, 0x7d, 0x7a, 0x76, 0x71, 0x6c, 0x65, 0x5f,
        0x58, 0x50, 0x49, 0x41, 0x3a, 0x33, 0x2c, 0x26, 0x20, 0x1b,
        0x16, 0x12, 0x0e, 0x0b, 0x08, 0x06, 0x04, 0x03, 0x02, 0x01,
        0x01, 0x01, 0x01, 0x01, 0x01, 0x02, 0x02, 0x02, 0x02, 0x02,
        0x02, 0x02, 0x03, 0x03, 0x04, 0x05, 0x07, 0x09, 0x0b, 0x0d,
        0x0f, 0x12, 0x15, 0x17, 0x1a, 0x1d, 0x20, 0x22, 0x25, 0x28,
        0x2a, 0x2d, 0x2f, 0x31, 0x33, 0x34, 0x35, 0x35, 0x35, 0x34,
        0x33, 0x31, 0x2e, 0x2b, 0x28, 0x24, 0x1f, 0x1b, 0x16, 0x11,
        0x0c, 0x08, 0x03, 0xff, 0xfc, 0xf9, 0xf7, 0xf5, 0xf4, 0xf3,
        0xf3, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xfa, 0xfb, 0xfc,
        0xfd, 0xfe, 0xff, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x08,
        0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0d, 0x0d, 0x0c, 0x0b, 0x09,
        0x07, 0x04, 0x01, 0xfd, 0xf8, 0xf4, 0xef, 0xea, 0xe5, 0xe1,
        0xdc, 0xd8, 0xd5, 0xd2, 0xcf, 0xcd, 0xcc, 0xcb, 0xcb, 0xcb,
        0xcc, 0xcd, 0xcf, 0xd1, 0xd3, 0xd6, 0xd8, 0xdb, 0xde, 0xe0,
        0xe3, 0xe6, 0xe9, 0xeb, 0xee, 0xf1, 0xf3, 0xf5, 0xf7, 0xf9,
        0xfb, 0xfc, 0xfd, 0xfd, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe,
        0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xfc,
        0xfa, 0xf8, 0xf5, 0xf2, 0xee, 0xea, 0xe5, 0xe0, 0xda, 0xd4,
        0xcd, 0xc6, 0xbf, 0xb7, 0xb0, 0xa8, 0xa1, 0x9b, 0x94, 0x8f,
        0x8a, 0x86, 0x83, 0x82, 0x81, 0x82, 0x85, 0x8a, 0x91, 0x9b,
        0xa7, 0xb5, 0xc5, 0xd8, 0xec
    },
```

```
,,
.attackStep = 0x03ee,
.decayStep = 0x01f7,
.sustainLevel = 0xd9,
.releaseStep = 0x0108
```

Software Synthesis

- Step 3: Sound Output – Filter



Hardware Synthesis – CV

- If the user selects “analog waveforms”, the Arduino will switch into CV/Gate Mode
- Arduino 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

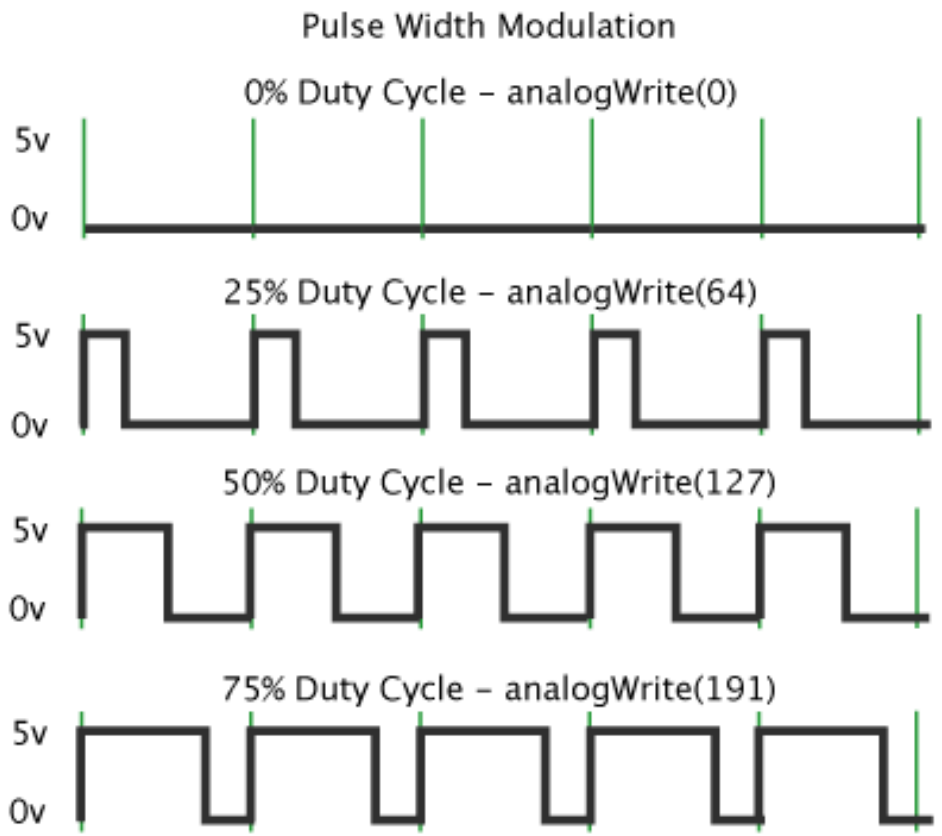
Pulse Width Modulation

The pulses are sent at a frequency of 62.5 kHz

They create the 5 voice outputs and 1 control voltage output

The PWM frequency gets sent into our Player Command Loop

Program memory sends musical outputs

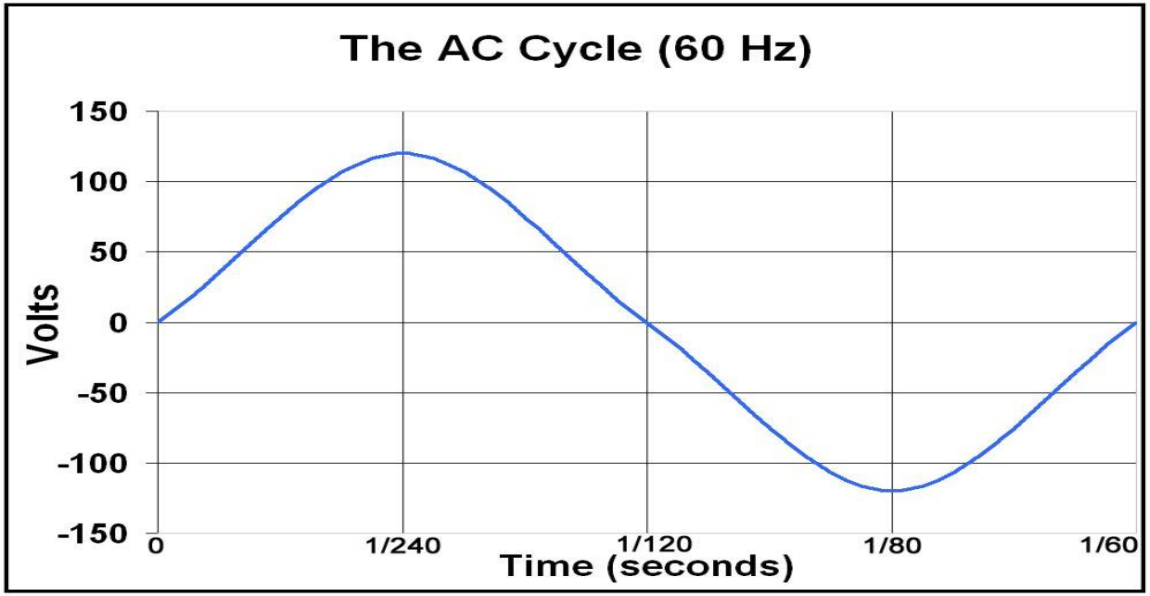


Power Supply

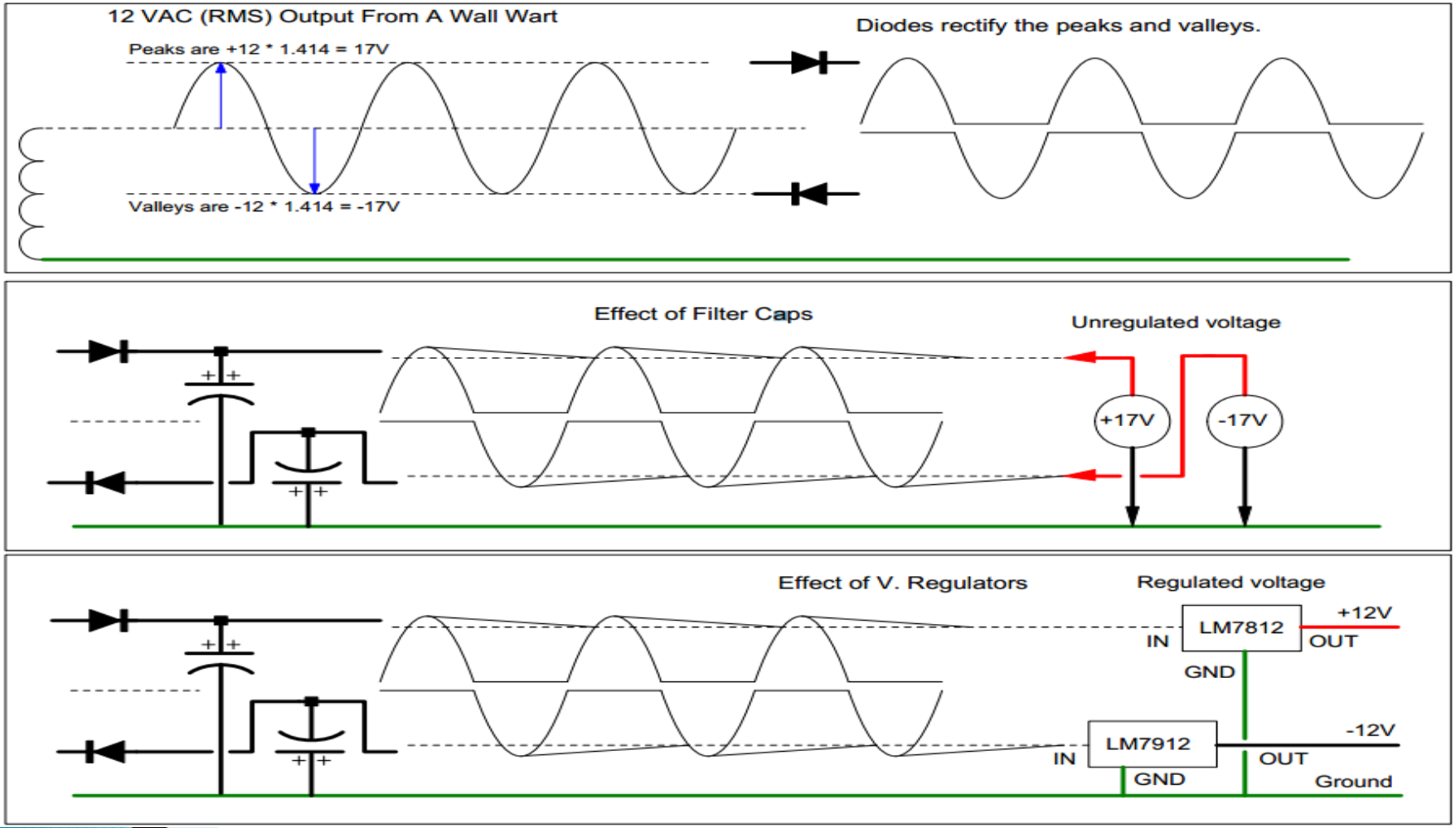
- Dual Power Supply
  - Wall Wart Power 12 VAC
  - Supports plenty of capacitance for a clean supply
  - Low part count
  - Creates bipolar (+ and -) 12 volts DC
  - Convenient source of bipolar voltage from the wall

Wall Wart Voltage Example

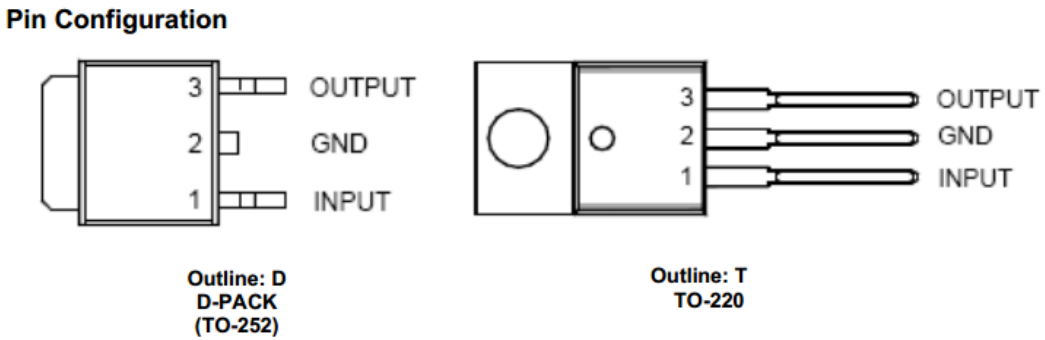
- If the Wall Wart outputs 12VAC, the VAC value of a transformer is an RMS value.
- So, 15VAC gives about 21.21V output. 12VAC gives about 17V.
- 12VAC * 1.414V = 16.968V



PSU Signal Flow Theory



Voltage Regulators: LM78XX or LM79XX



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

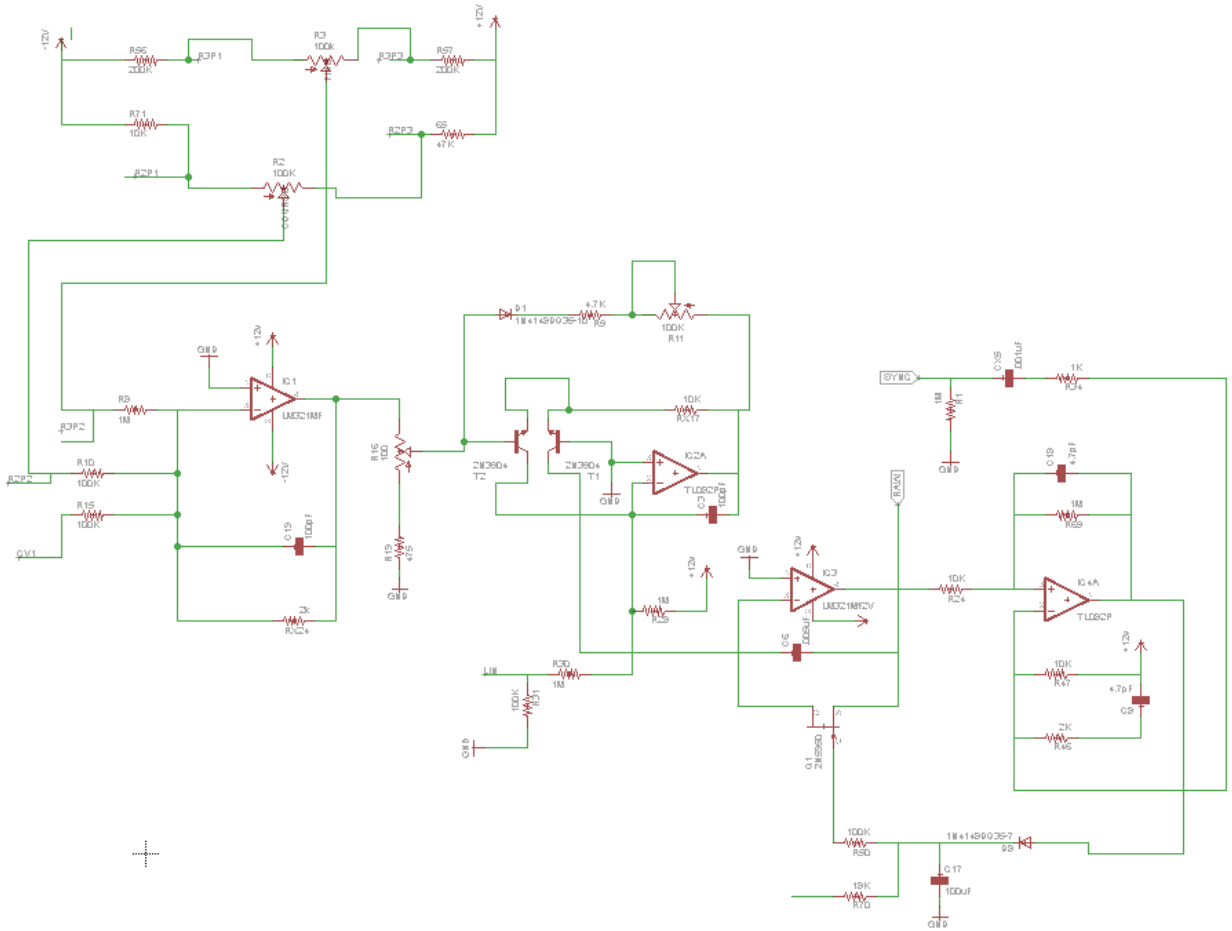
Major Power Supply Components

LMXX Regulators

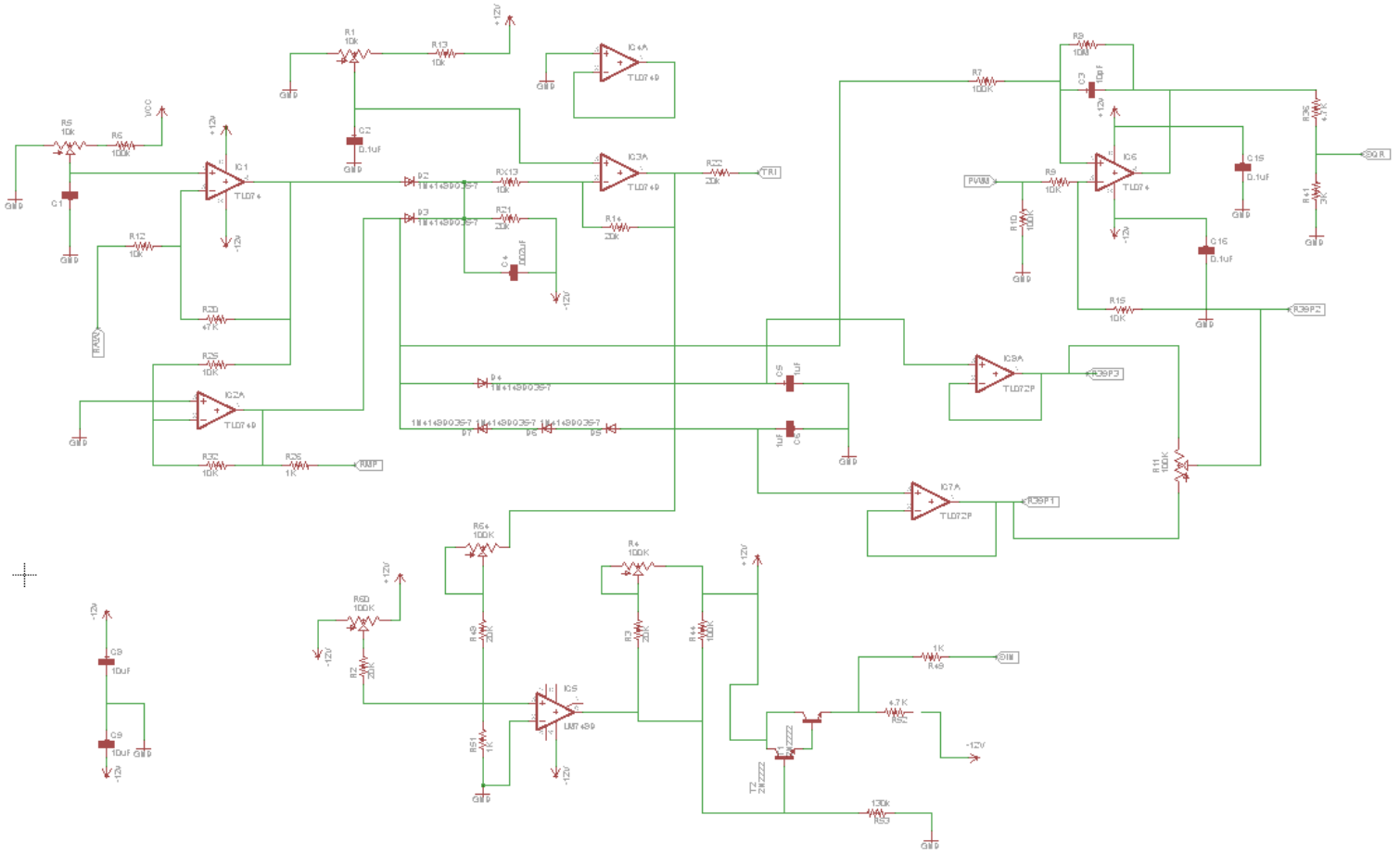
- 1000mA current

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

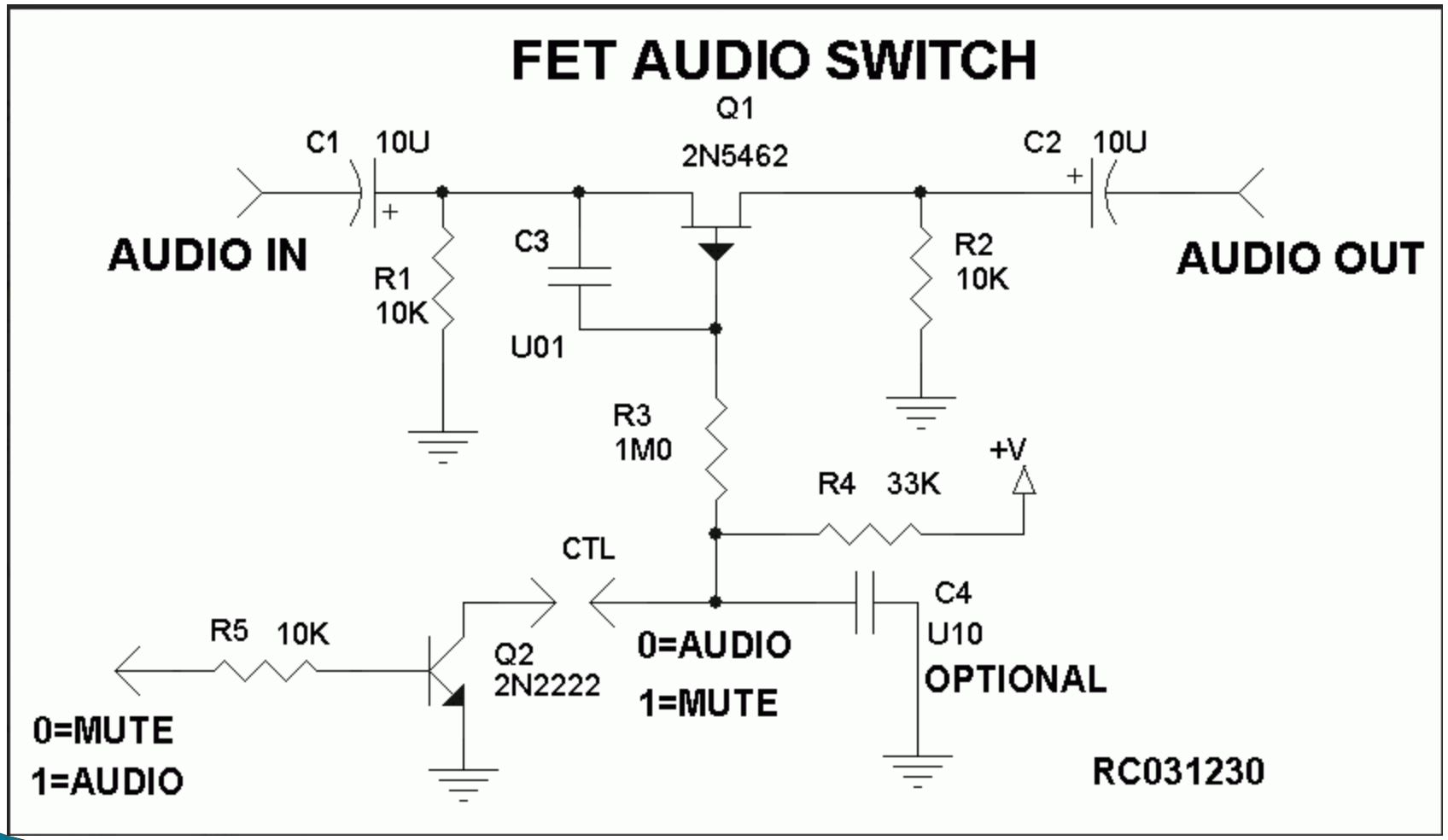
Oscillator Schematic Page 1



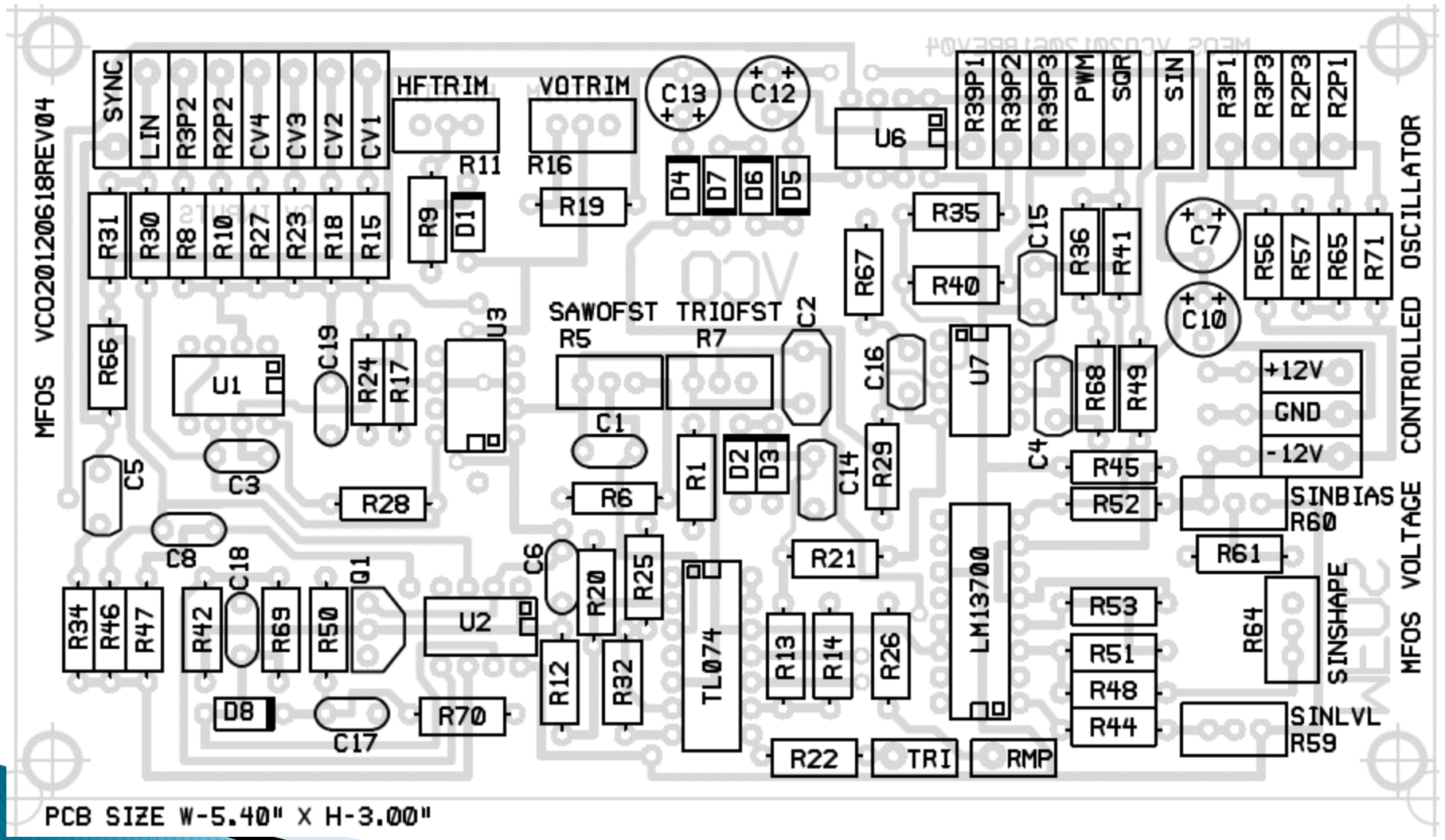
Oscillator Schematic Page 2



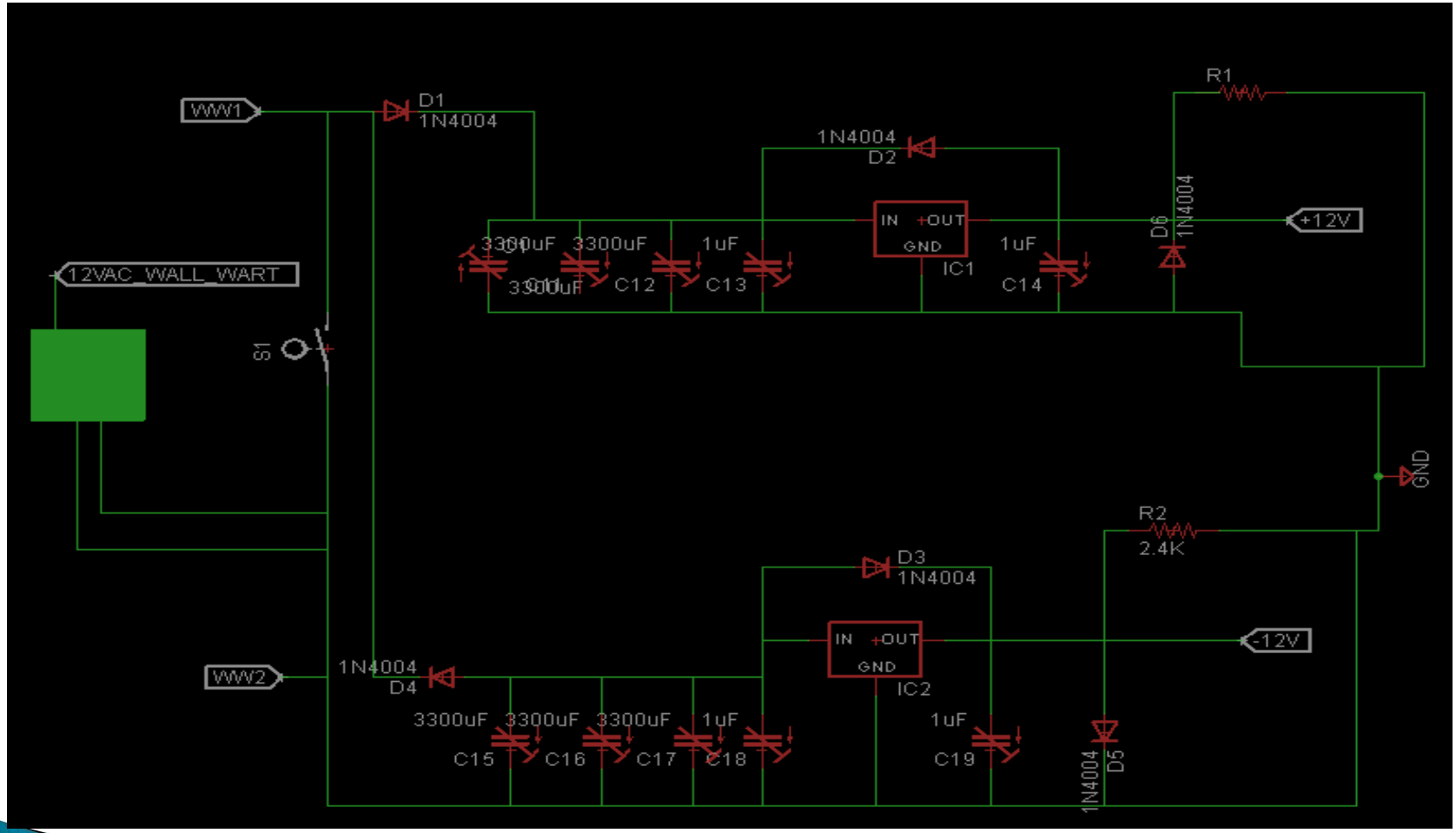
Muting Circuit



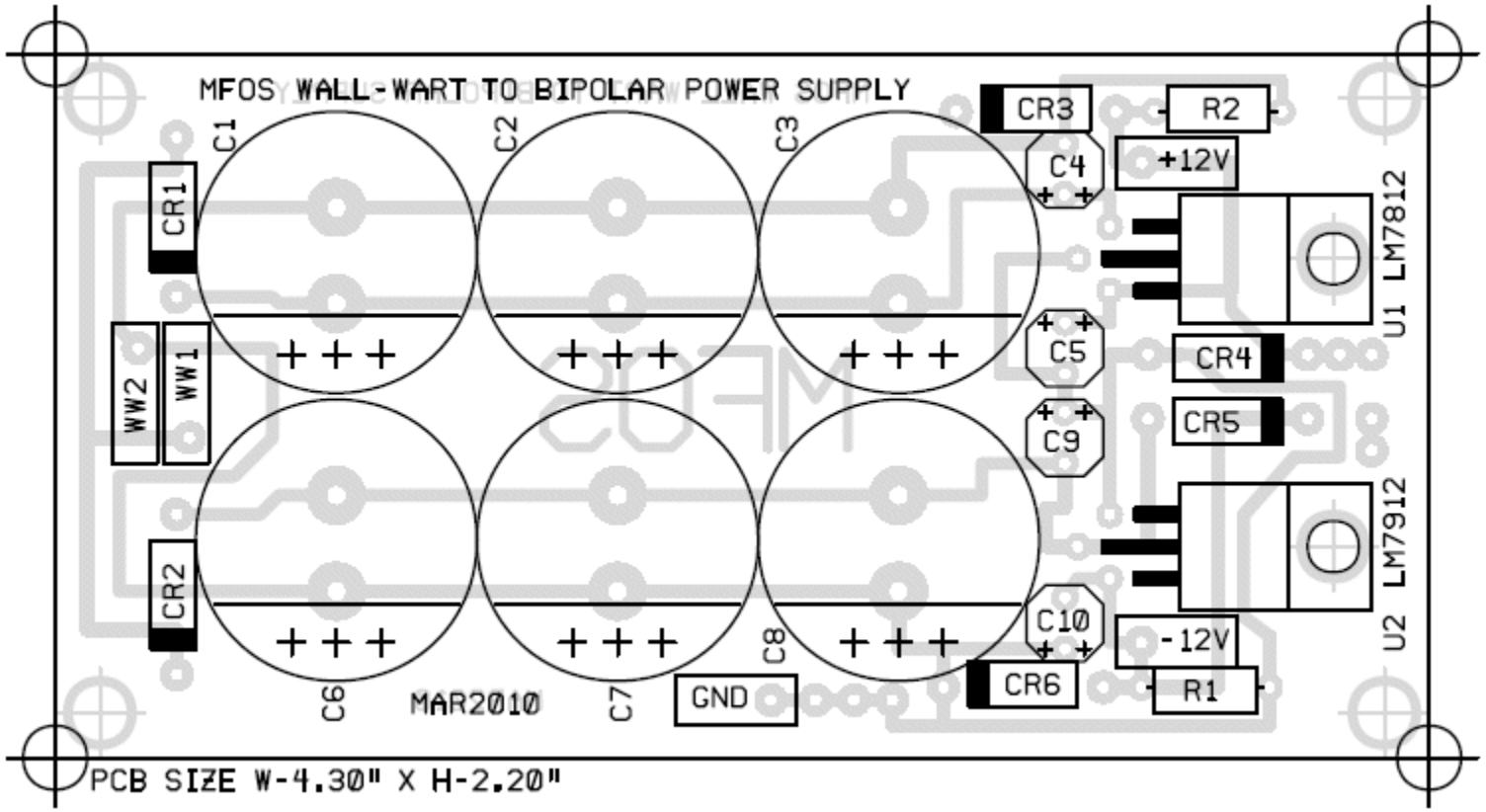
PCB – Oscillator



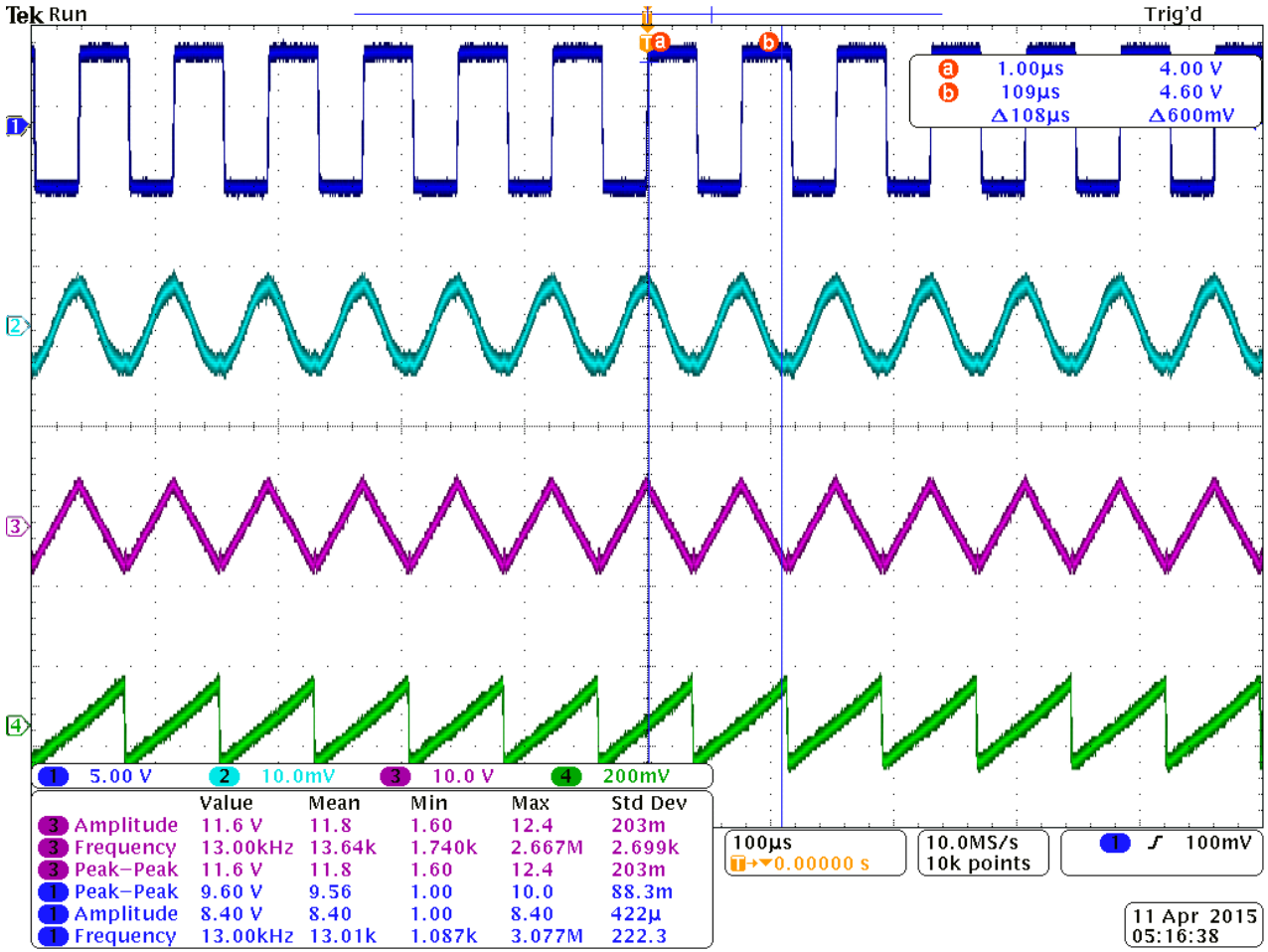
PSU Eagle Schematic



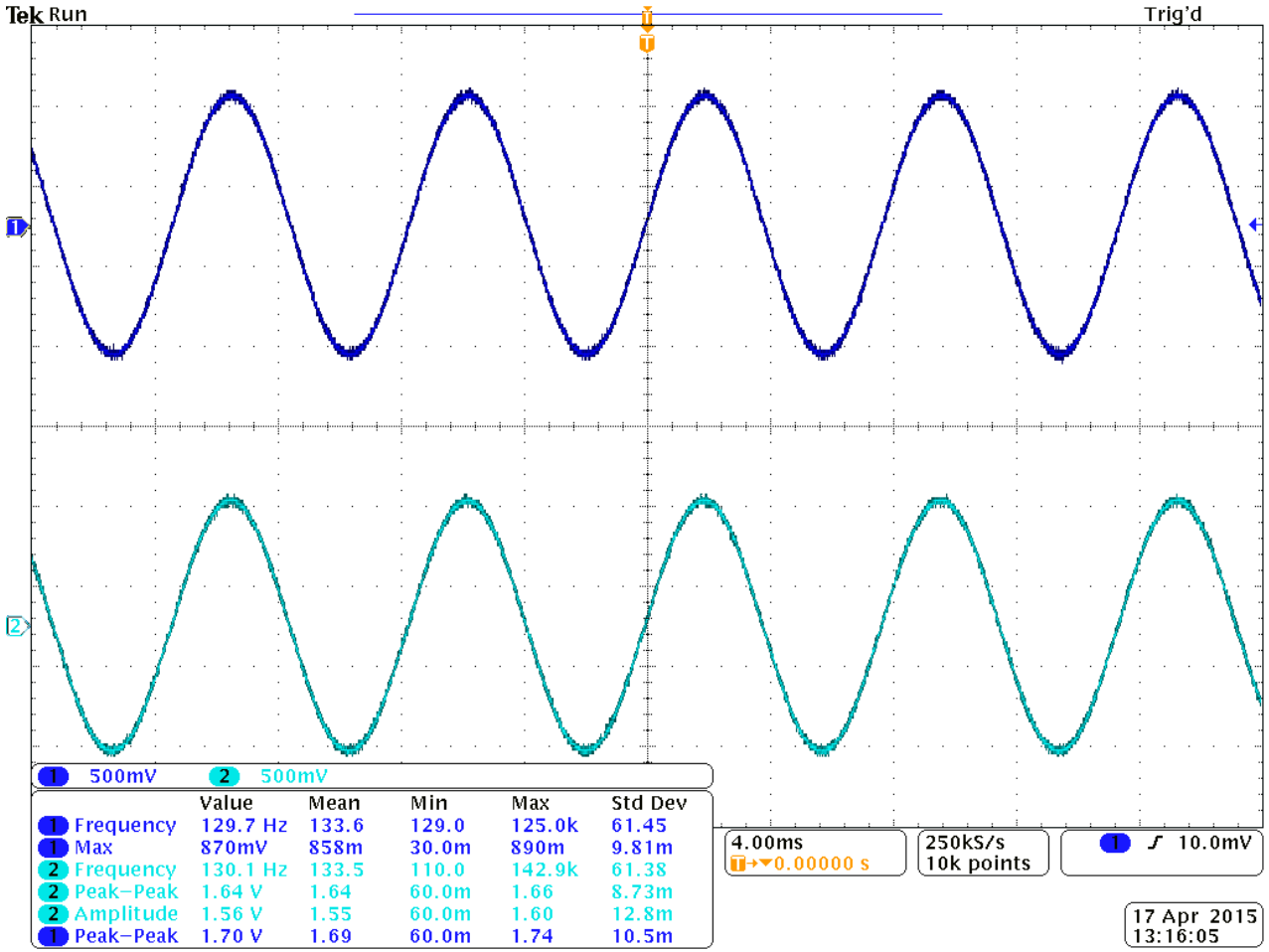
Power Supply PCB

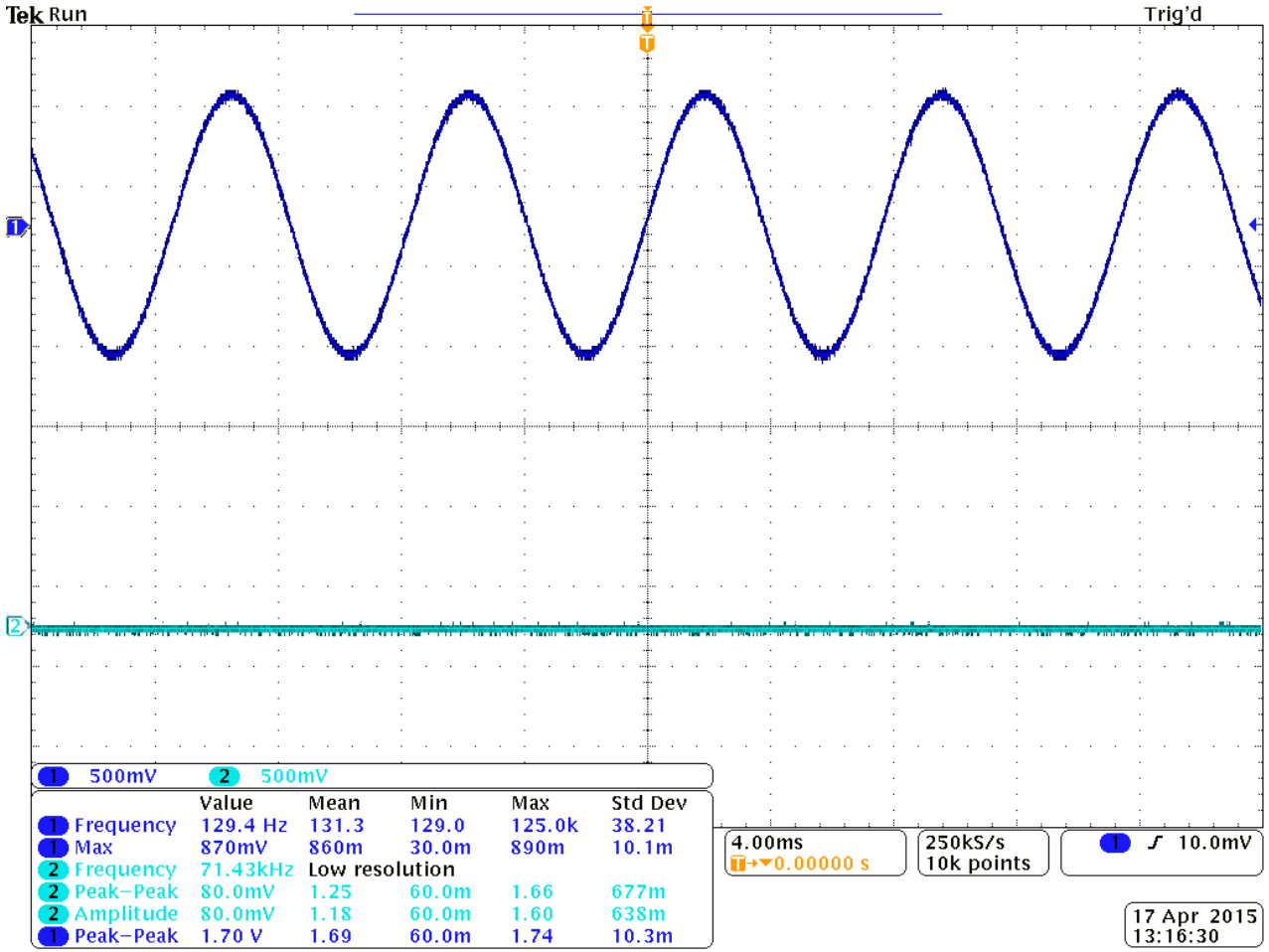


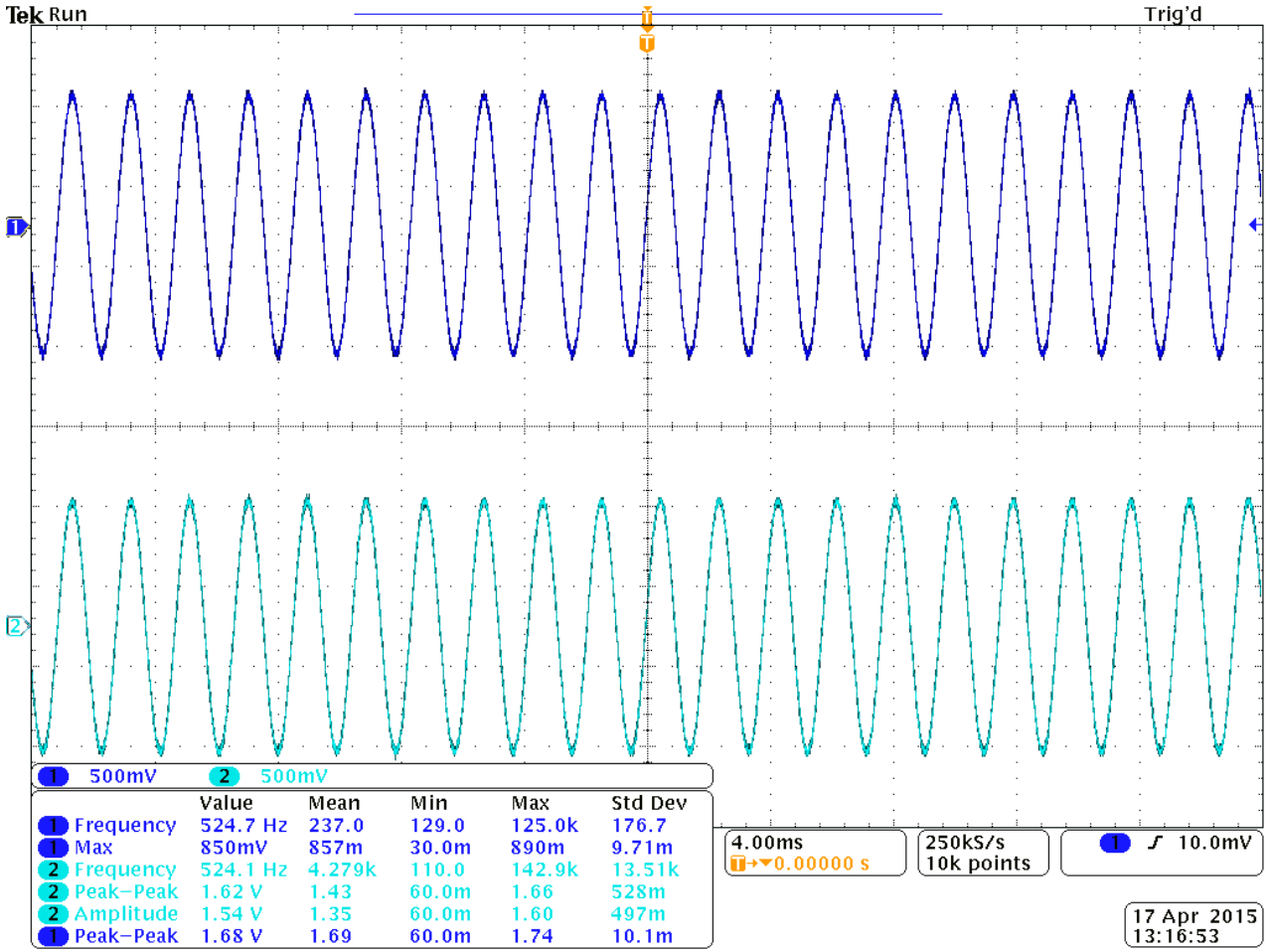
Hardware Testing

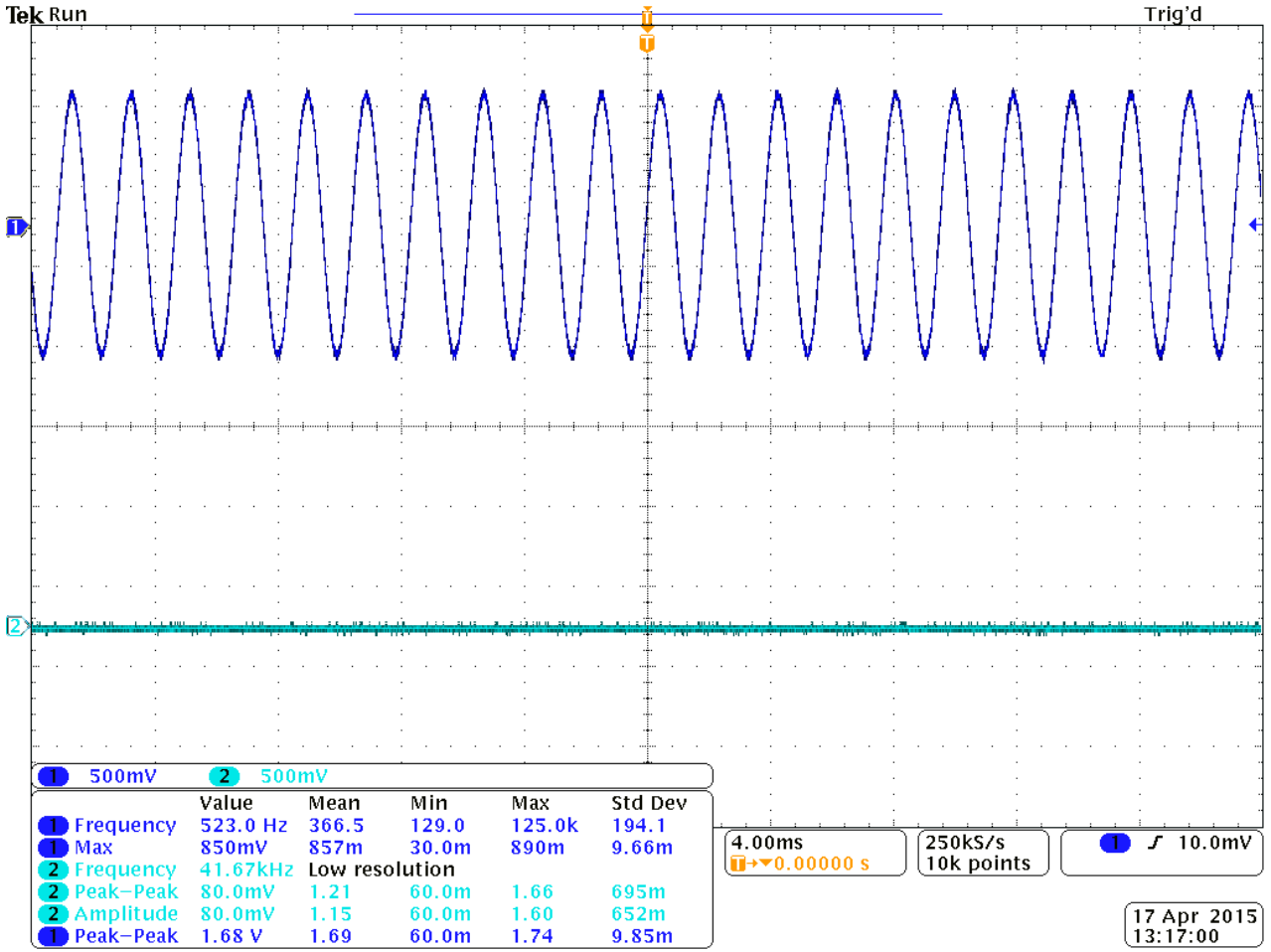


Mute Testing

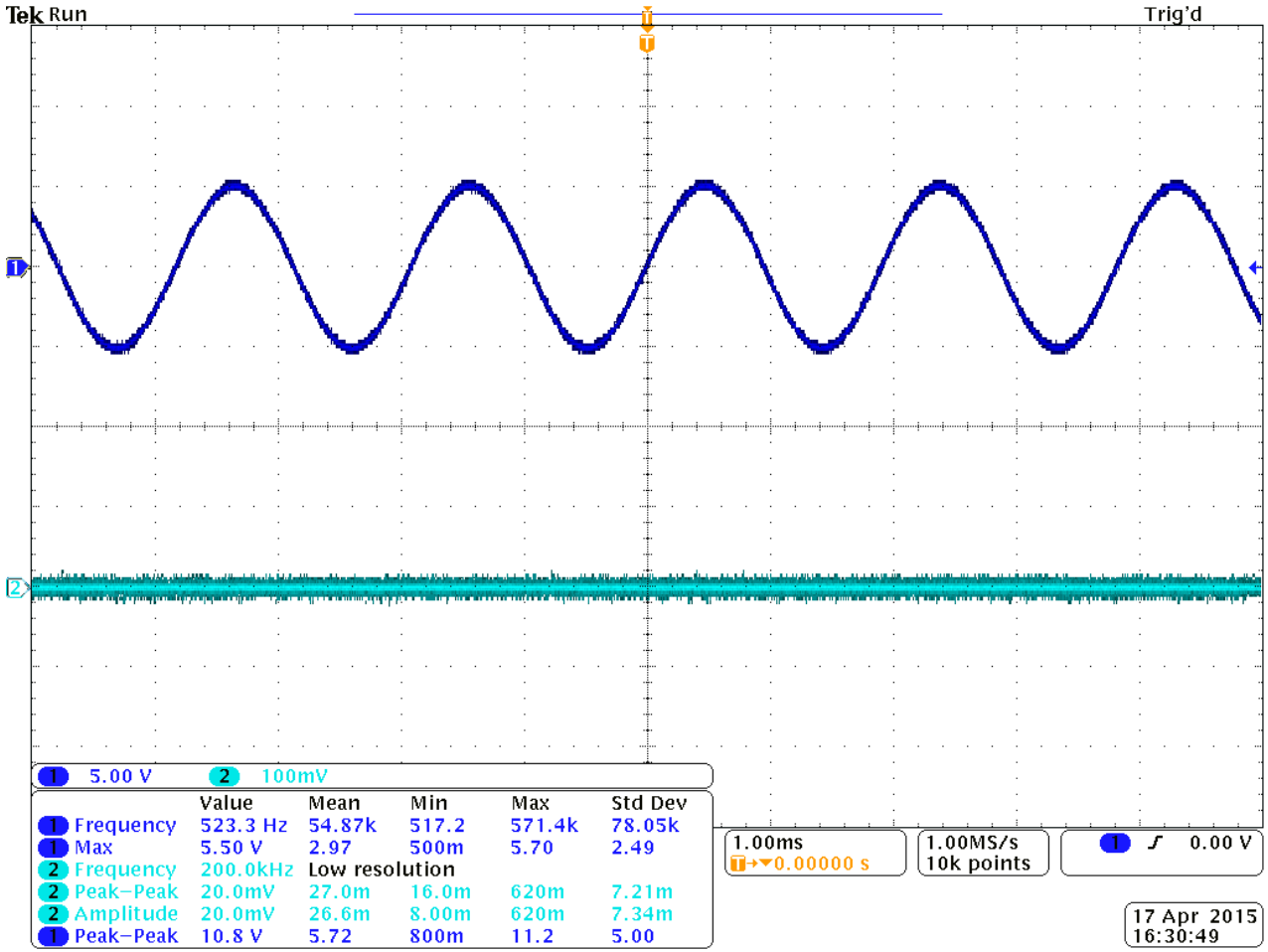


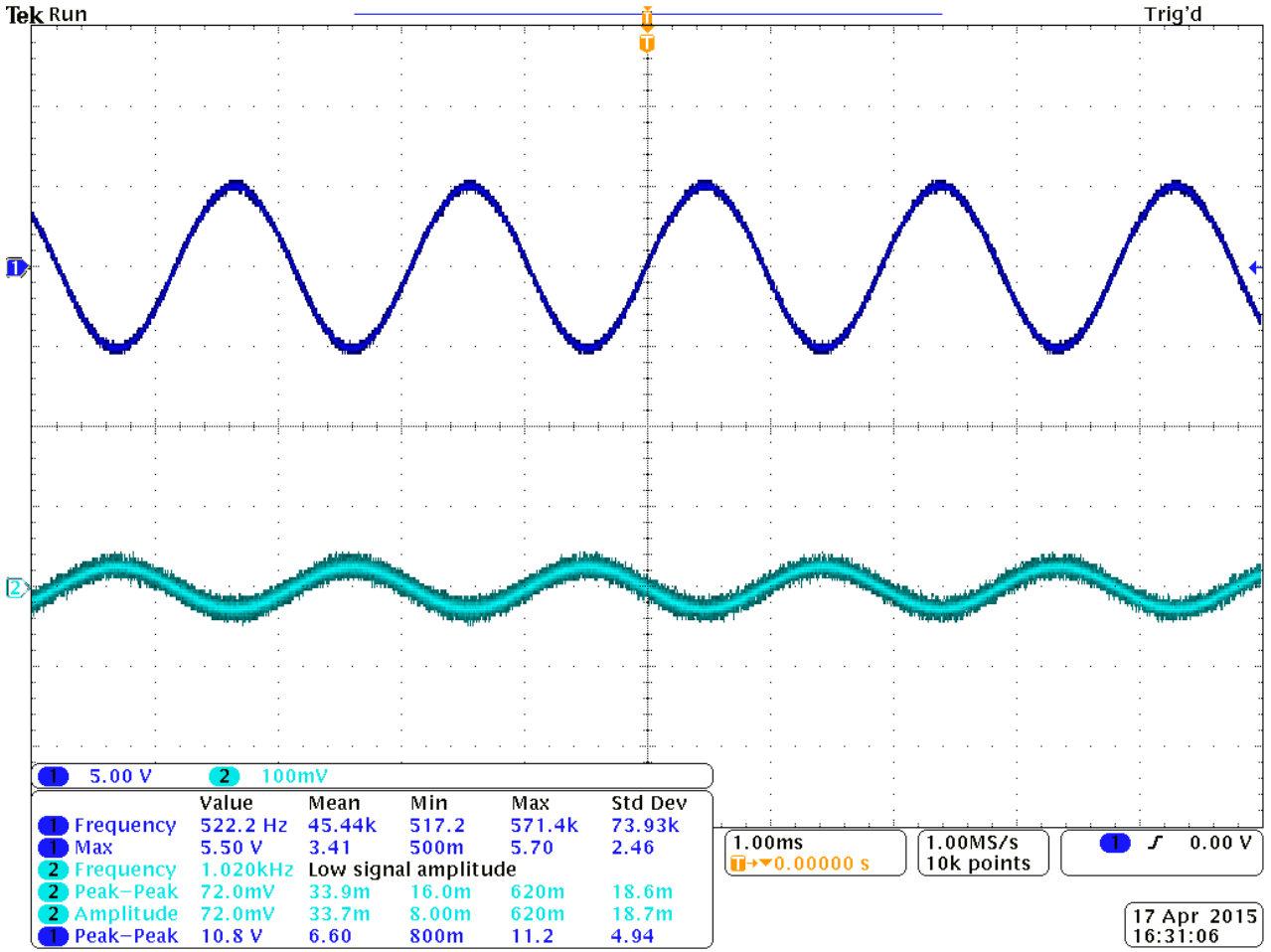

Tek Run
Trig'd
1 500mV
2 500mV
Value Mean Min Max Std Dev
1 Frequency 129.4 Hz 131.3 129.0 125.0k 38.21
1 Max 870mV 860m 30.0m 890m 10.1m
2 Frequency 71.43kHz Low resolution
2 Peak-Peak 80.0mV 1.25 60.0m 1.66 677m
2 Amplitude 80.0mV 1.18 60.0m 1.60 638m
1 Peak-Peak 1.70 V 1.69 60.0m 1.74 10.3m
4.00ms
0.00000 s
250kS/s
10k points
1 10.0mV
17 Apr 2015
13:16:30

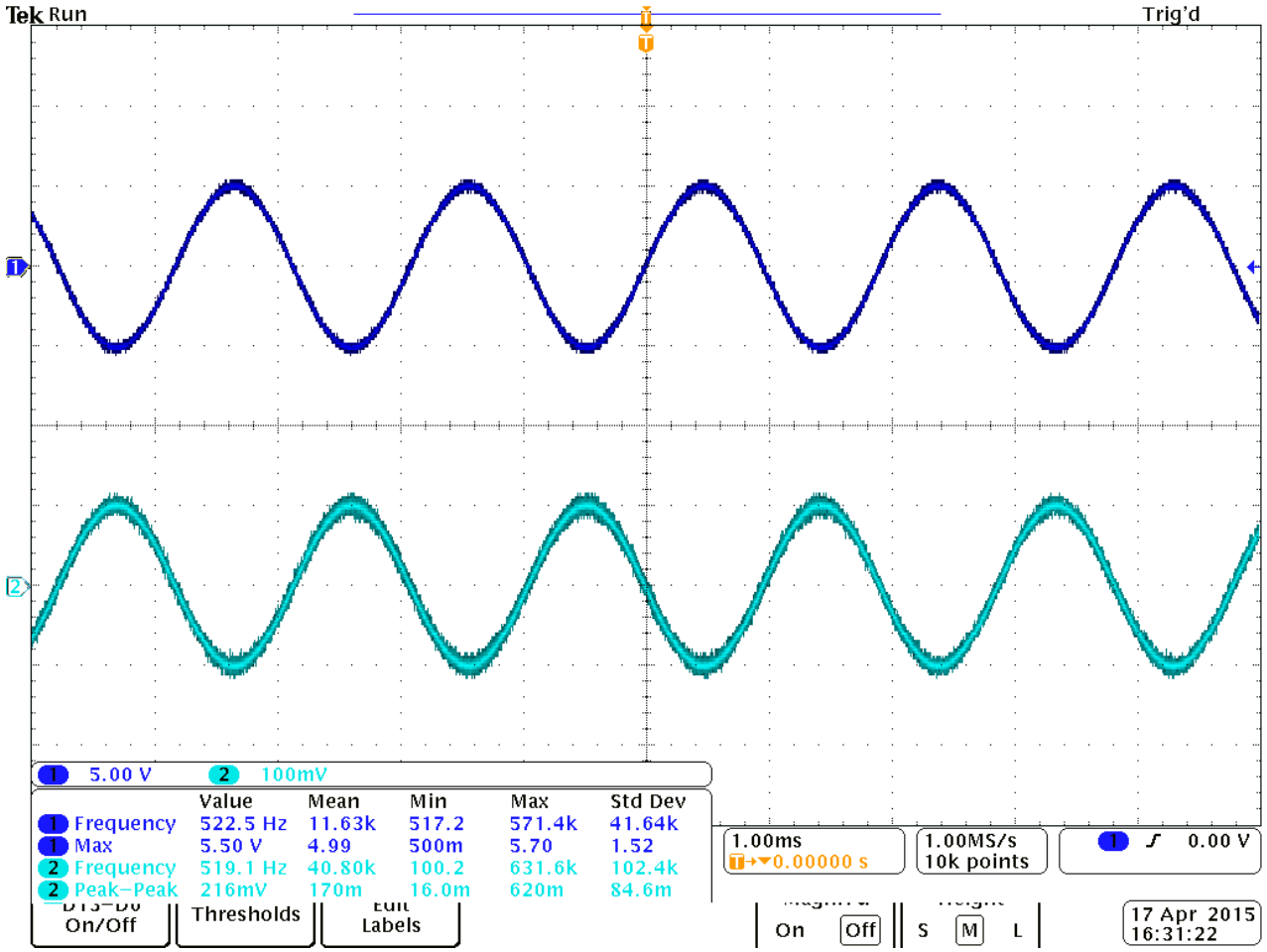

Tek Run
Trig'd
1 500mV
2 500mV
Value Mean Min Max Std Dev
1 Frequency 524.7 Hz 237.0 129.0 125.0k 176.7
1 Max 850mV 857m 30.0m 890m 9.71m
2 Frequency 524.1 Hz 4.279k 110.0 142.9k 13.51k
2 Peak-Peak 1.62 V 1.43 60.0m 1.66 528m
2 Amplitude 1.54 V 1.35 60.0m 1.60 497m
1 Peak-Peak 1.68 V 1.69 60.0m 1.74 10.1m
4.00ms
0.00000 s
250kS/s
10k points
1 10.0mV
17 Apr 2015
13:16:53


Tek Run
Trig'd
1 500mV
2 500mV
Value Mean Min Max Std Dev
1 Frequency 523.0 Hz 366.5 129.0 125.0k 194.1
1 Max 850mV 857m 30.0m 890m 9.66m
2 Frequency 41.67kHz Low resolution
2 Peak-Peak 80.0mV 1.21 60.0m 1.66 695m
2 Amplitude 80.0mV 1.15 60.0m 1.60 652m
1 Peak-Peak 1.68 V 1.69 60.0m 1.74 9.85m
4.00ms
0.00000 s
250kS/s
10k points
1 10.0mV
17 Apr 2015
13:17:00

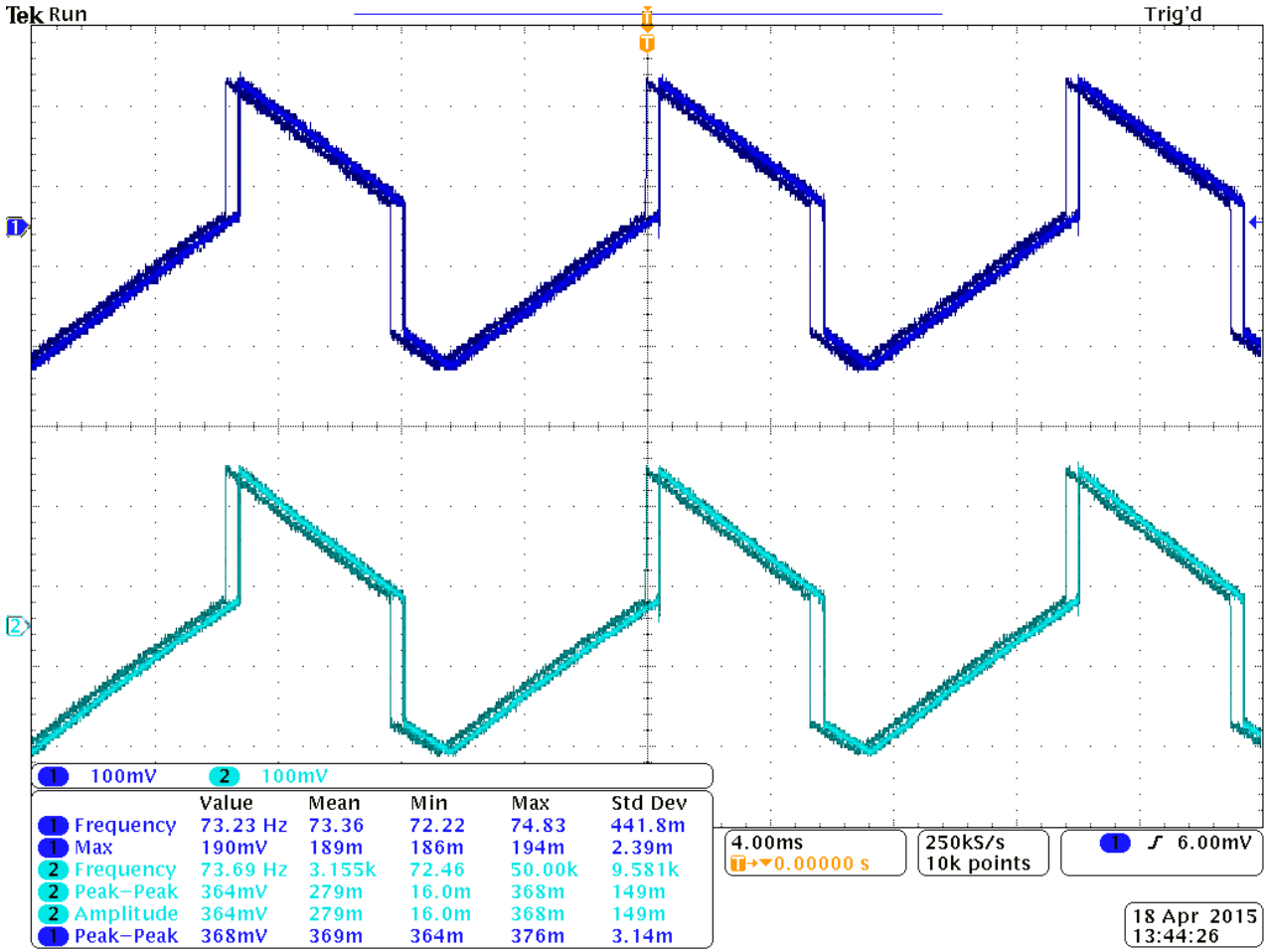
Mixer Testing

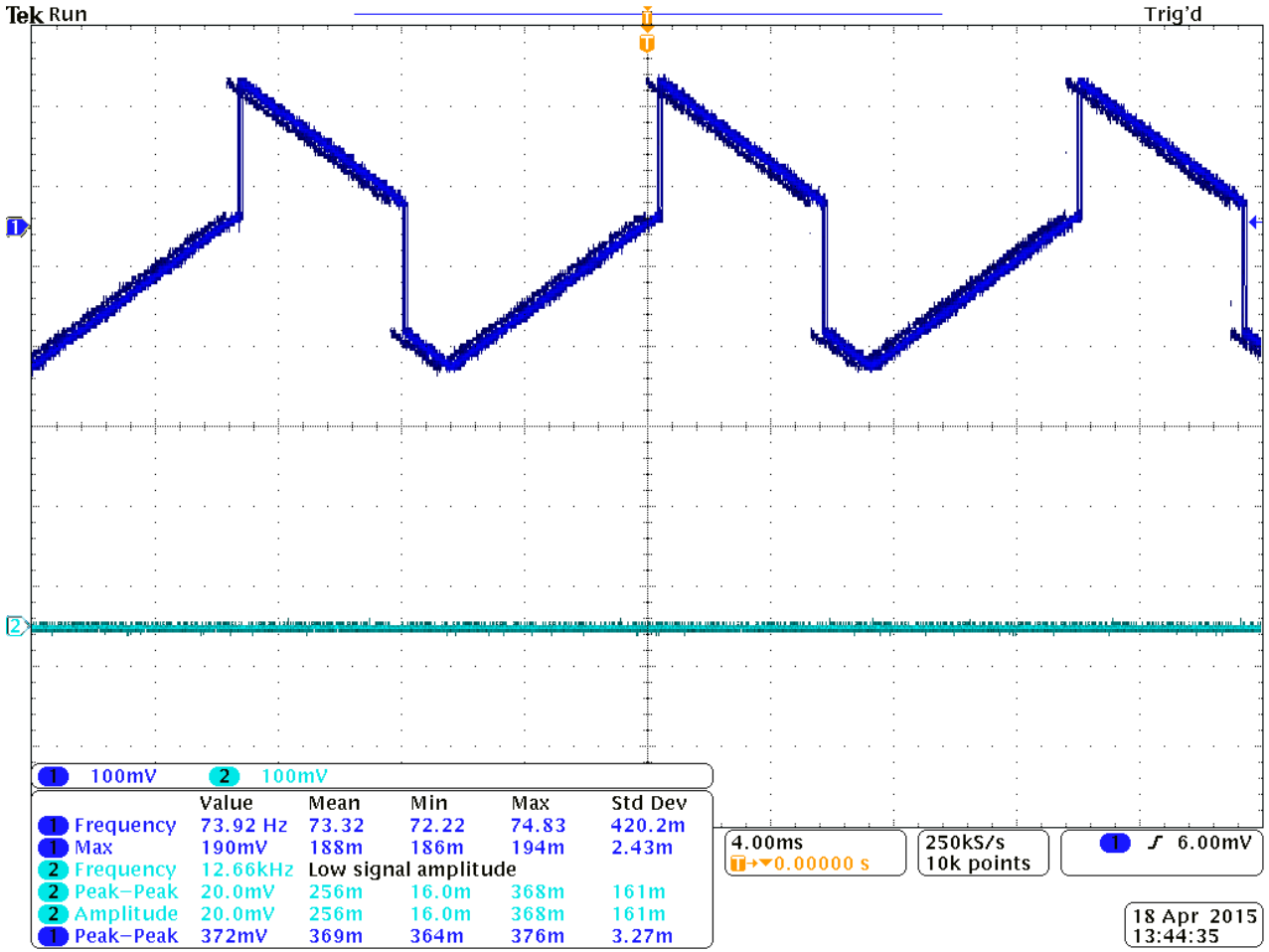



Tek Run
Trig'd
1 5.00 V
2 100mV
Value Mean Min Max Std Dev
1 Frequency 522.2 Hz 45.44k 517.2 571.4k 73.93k
1 Max 5.50 V 3.41 500m 5.70 2.46
2 Frequency 1.020kHz Low signal amplitude
2 Peak-Peak 72.0mV 33.9m 16.0m 620m 18.6m
2 Amplitude 72.0mV 33.7m 8.00m 620m 18.7m
1 Peak-Peak 10.8 V 6.60 800m 11.2 4.94
1.00ms
0.00000 s
1.00MS/s
10k points
1 0.00 V
17 Apr 2015
16:31:06

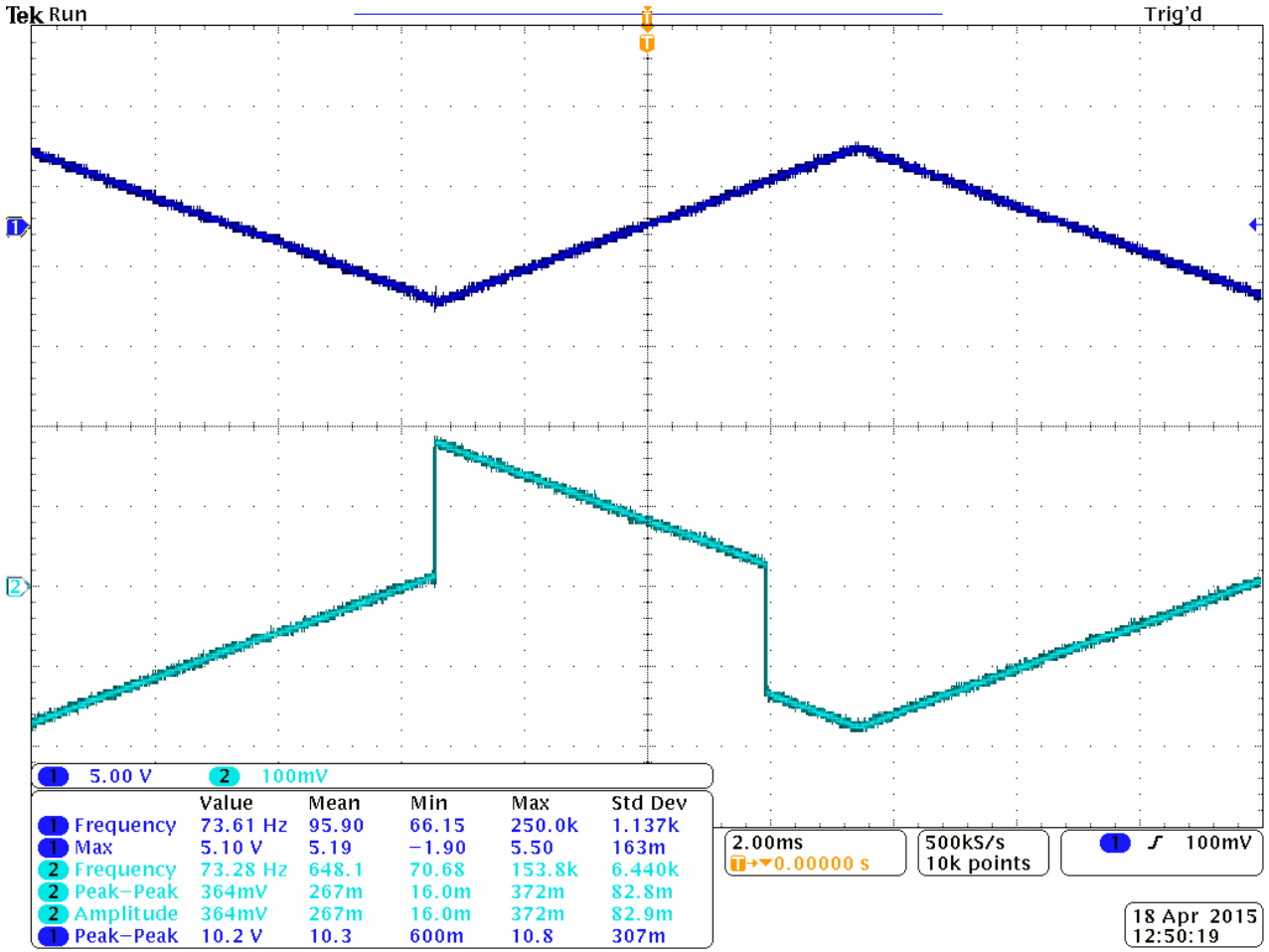

Tek Run
Trig'd
1 5.00 V
2 100mV
Value Mean Min Max Std Dev
1 Frequency 522.5 Hz 11.63k 517.2 571.4k 41.64k
1 Max 5.50 V 4.99 500m 5.70 1.52
2 Frequency 519.1 Hz 40.80k 100.2 631.6k 102.4k
2 Peak-Peak 216mV 170m 16.0m 620m 84.6m
1.00ms
0.00000 s
1.00MS/s
10k points
1 0.00 V
On/Off
Thresholds
Edit Labels
On
Off
S
M
L
17 Apr 2015
16:31:22

Mixed Waveform Mute Testing

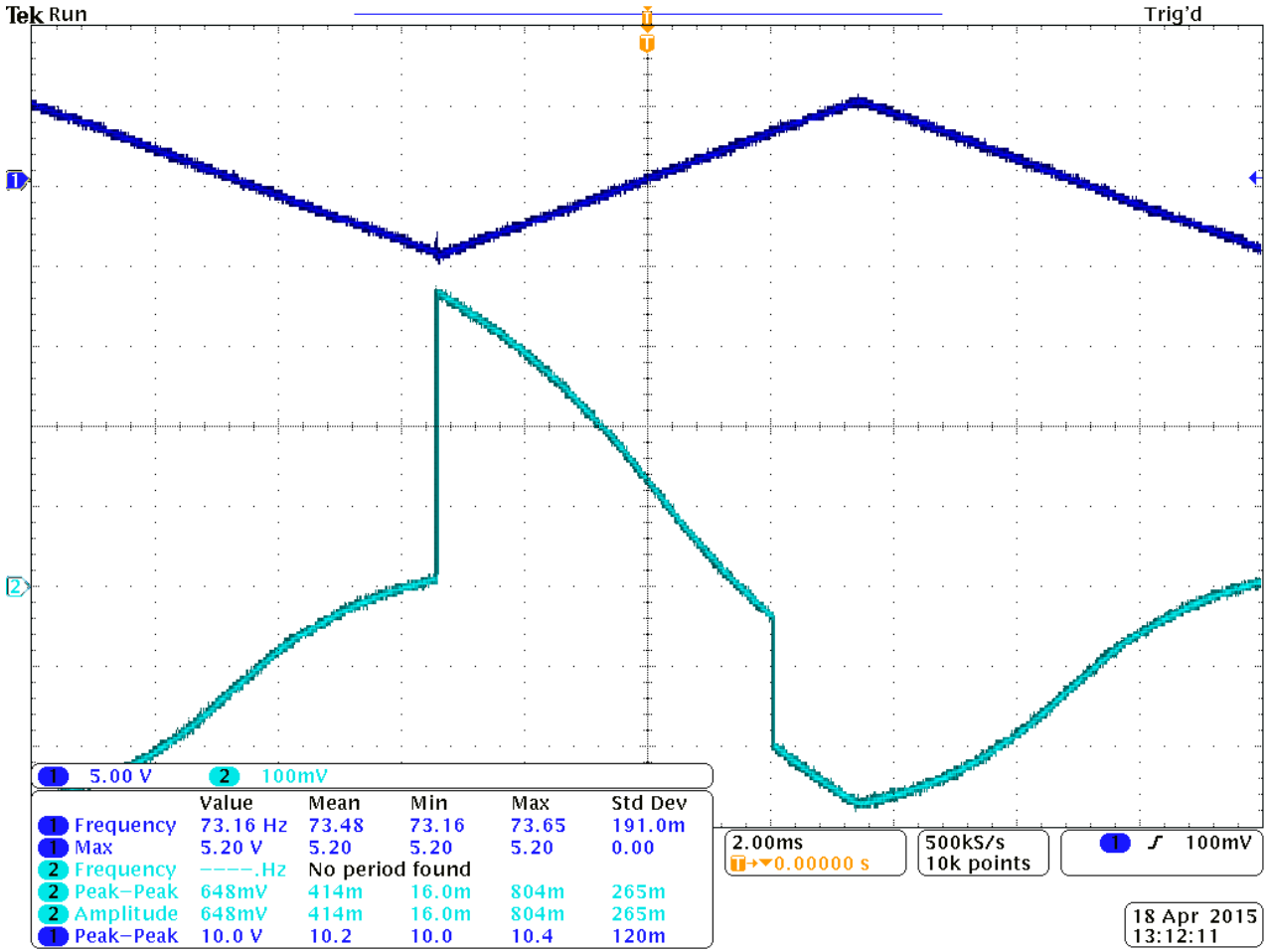



Tek Run
Trig'd
1 100mV
2 100mV
Value Mean Min Max Std Dev
1 Frequency 73.92 Hz 73.32 72.22 74.83 420.2m
1 Max 190mV 188m 186m 194m 2.43m
2 Frequency 12.66kHz Low signal amplitude
2 Peak-Peak 20.0mV 256m 16.0m 368m 161m
2 Amplitude 20.0mV 256m 16.0m 368m 161m
1 Peak-Peak 372mV 369m 364m 376m 3.27m
4.00ms
0.00000 s
250kS/s
10k points
1 6.00mV
18 Apr 2015
13:44:35

Two Mixed Waveforms



Four Mixed Waveforms



Software Testing

- Tested Android to Bluetooth Communication
  - Used Log files in Android Code to see if the BT Socket was connecting
  - Wrote Arduino Code to Take in Data and display it onto the screen
  - Connected the Arduino to the USB port of the computer and connected to the HC-05
  - Configured the Hyperterminal to receive the values that were sent
  - This test passed when the characters matched up on both ends

Software Testing cont.

- Tested Note Sounds
  - This was tested by creating the filter that and hooking this up to the Arduino and a headphone jack
  - We iterated through the array of notes until we heard all 72 notes, to make sure they were going up the scale using two octaves
  - Then, we configured the Android Application to send strings that correspond to each note and repeated the same process.

Budget

Parts	Cost
VCO Parts	117.07
Power Supply Parts	20.48
Power Supply PCB	20
VCO PCB	20
Arduino	25.61
HC-05	8.37
9v Batteries & Holder	10.59
Wood Enclosure	37.5
Tablet	0
Soldering	228.23

Total Cost: 487.85

Successes

- Oscillator design is robust and tracks accurately above and below the range of human hearing
- Even with modules, current draw will not be significant, resulting in a need of 250 mA or less to power the entire circuit
- Control Voltages are accurate and allow for held analog notes
- Muting circuit is accurate and does not introduce noise or clicks or pops

Challenges

- Design is 1 month old due to project rehaul
- Bluetooth module was difficult to work with and took a long time to diagnose and fix
- The oscillator being always on required the team to come up with a muting solution (filters were also attempted)

Future Improvements

- More analog modules (ADSR, Adjustable Filters, drum machine, etc)
- Extending the range of our digital software synthesis and adding more notes to create a 6 octave range
- Using a DAC to drive the Analog Oscillator control voltages to extend the VCO's range to 6 octaves, with additional CV precision resulting in finer tuning
- Implementing a looping system to allow digital software synthesis to play held notes for longer durations

Questions?