Midterm Presentation
EML 4552C/EEL 4914C- Senior Design

Team # 19
Jordan Berke
Dustin McRae
Khristofer Thomas
Luis Bonilla
Trevor Hubbard

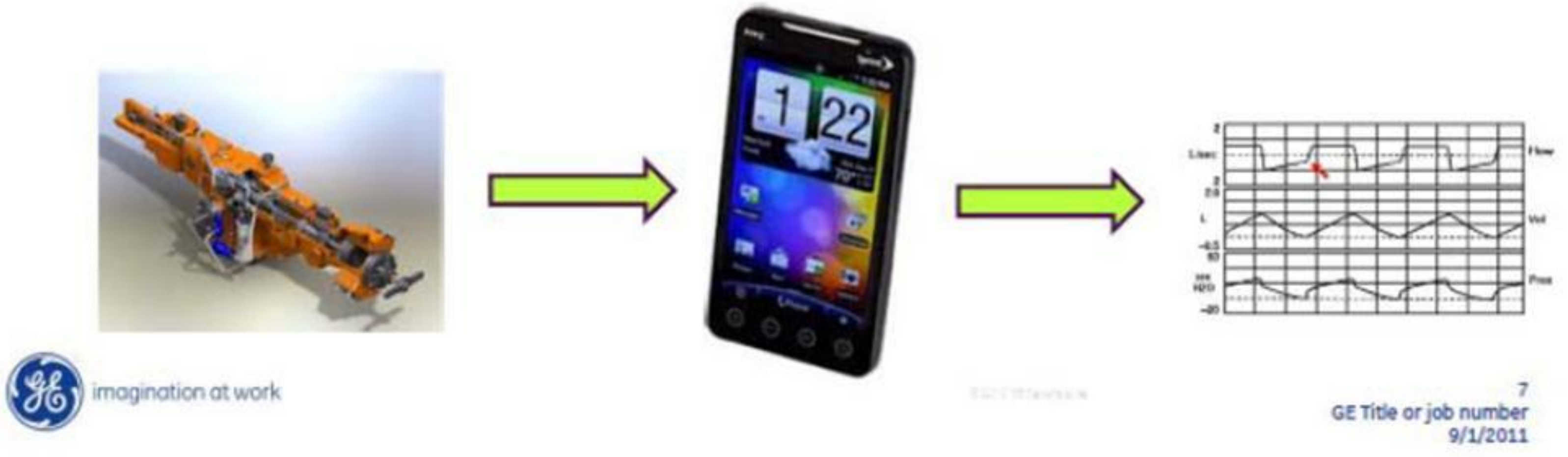
Google Mobile App for Compressor Performance (GE)

Project Sponsor
General Electric

Project Advisors:

Russell Wilburn
Industry Advisor, GE Oil & Gas
Dr. Sam Taira
Department of Mechanical Engineering
Dr. Linda DeBrunner
Department of Electrical and Computer Engineering

Scope of the Project

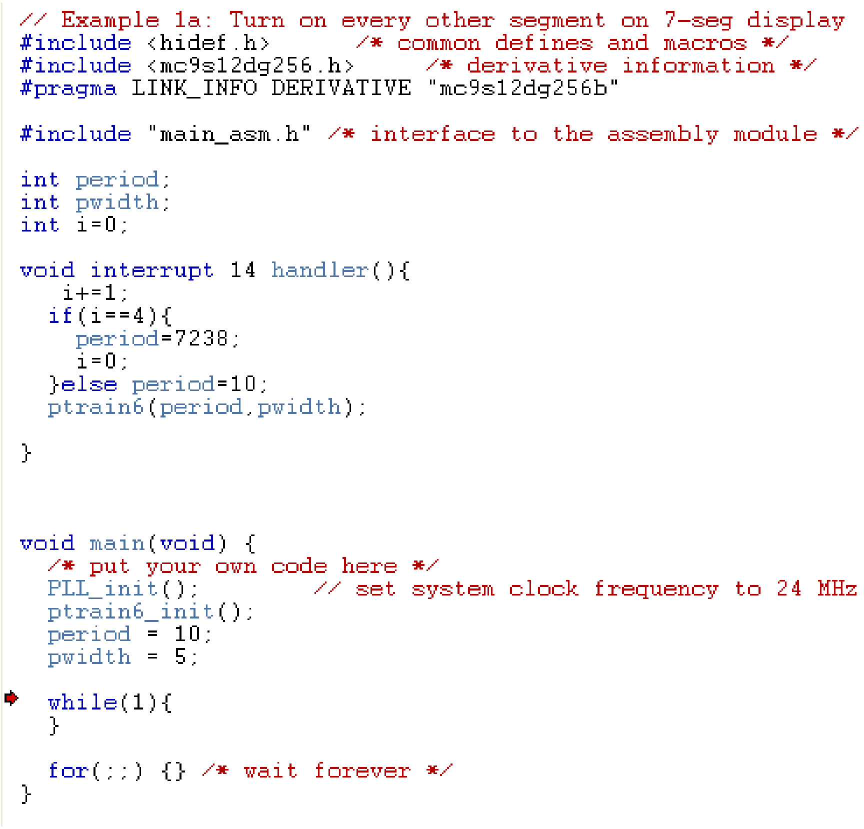


7
GE Title or job number
9/1/2011

- Customer Needs
  - – Transfer data wirelessly to an Android phone.
  - – Assembly time less than 5 minutes.
  - – No modifications to pipes; non-intrusive method.
  - – Software must collect, store and display data.
  - – Working Demo.

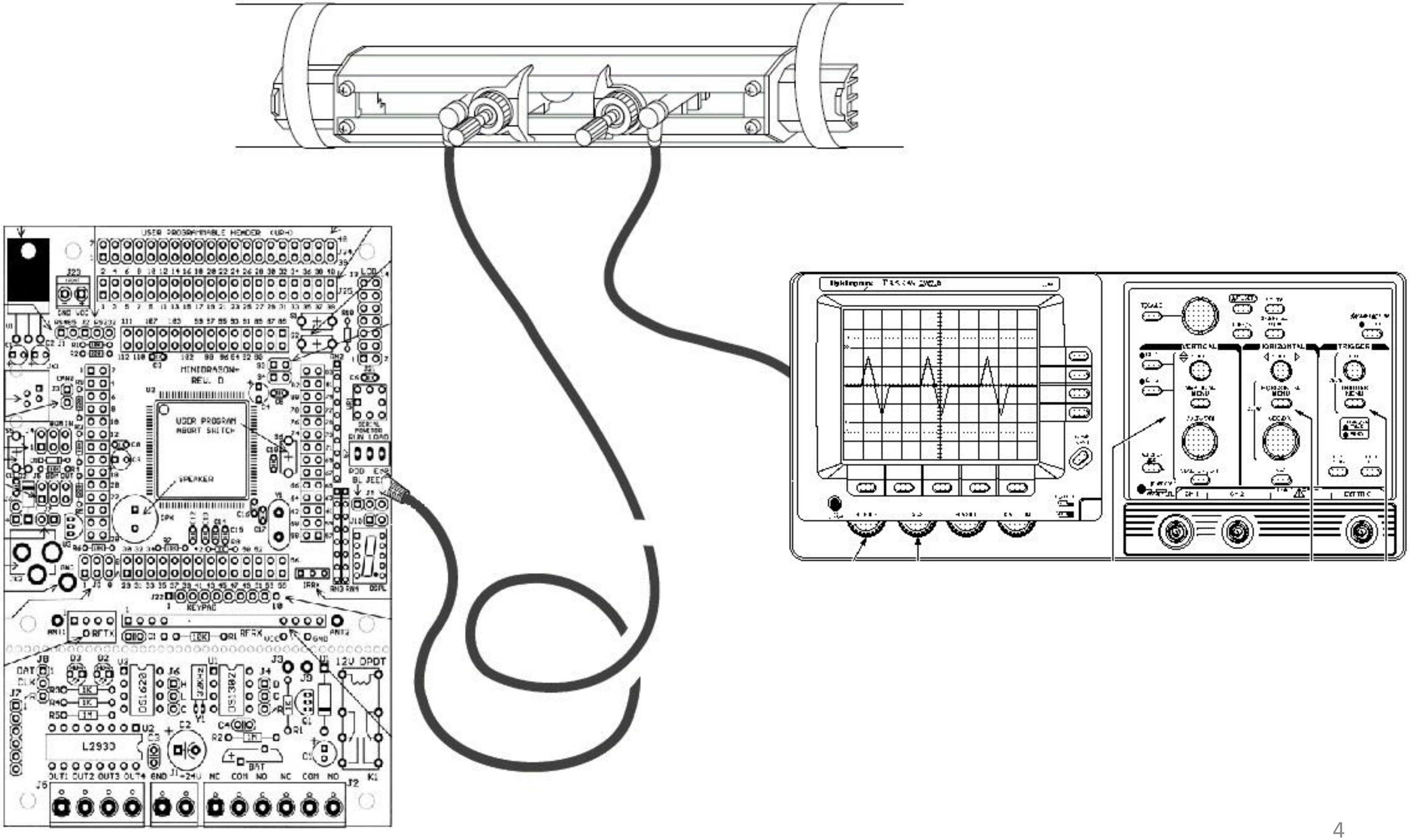
Sensor Update

- Utilized the DragonBoard to output signal to ultrasonic transducers
- Calls 16-bit up counter with resolution of 1.5 MHz
- Close enough to start testing
- Using basic low level functions for fast accurate signal reproduction



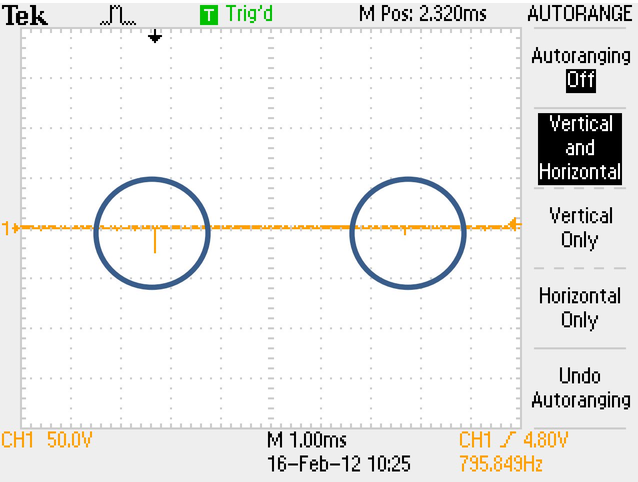
```
// Example 1a: Turn on every other segment on 7-seg display
#include <hidef.h>      /* common defines and macros */
#include <mc9s12dg256.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dg256b"

#include "main_asm.h" /* interface to the assembly module */

int period;
int pwidth;
int i=0;

void interrupt 14 handler(){
   i+=1;
  if(i==4){
    period=7238;
    i=0;
  }else period=10;
  ptrain6(period,pwidth);

}



void main(void) {
  /* put your own code here */
  PLL_init();        // set system clock frequency to 24 MHz
  ptrain6_init();
  period = 10;
  pwidth = 5;

  while(1){
  }

  for(;;) {} /* wait forever */
}
```

Experimental Setup



Flow Meter Output

- Burst of four square waves with 250 ns period
- Multiple bursts located 500 µs apart
- Approximately 50V peak to peak



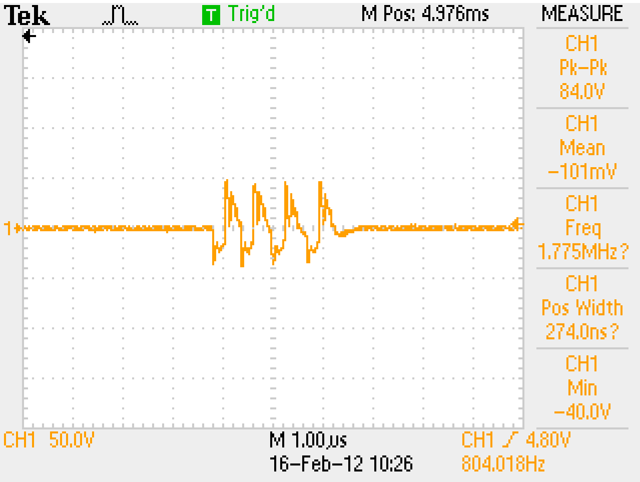
Tek Trig'd M Pos: 2.320ms

AUTORANGE

Autoranging Off

CH1 50.0V M 1.00ms CH1 ⁄ 4.80V

16–Feb–12 10:25 795.849Hz



Tek Trig'd M Pos: 4.976ms

MEASURE

CH1 Pk–Pk 84.0V

CH1 Mean –101mV

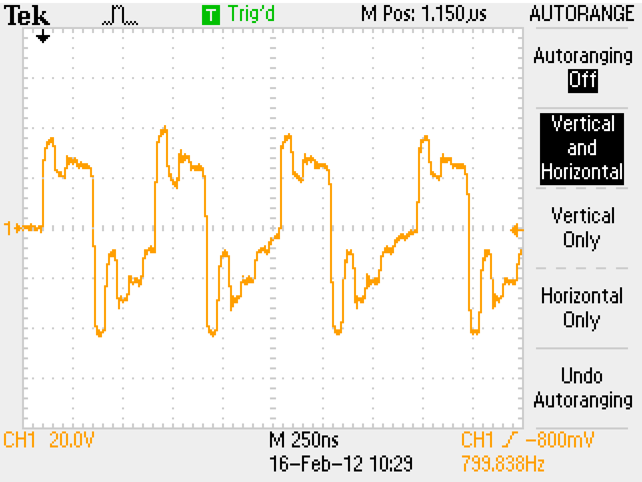
CH1 Freq 1.775MHz?

CH1 Pos Width 274.0ns?

CH1 Min –40.0V

CH1 50.0V M 1.00µs CH1 ⁄ 4.80V

16–Feb–12 10:26 804.018Hz



Tek Trig'd M Pos: 1.150µs

AUTORANGE

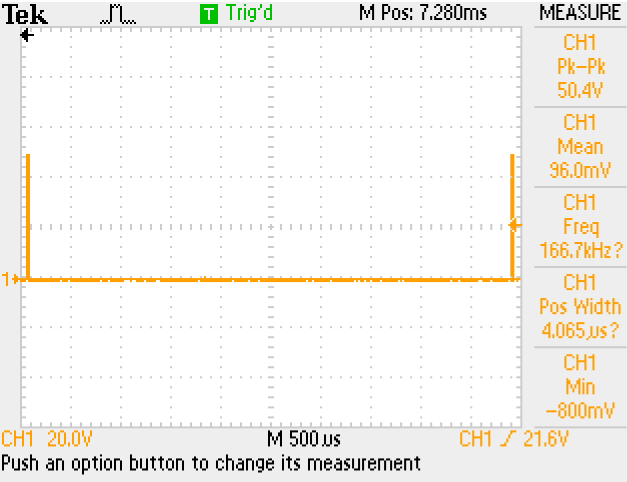
Autoranging Off

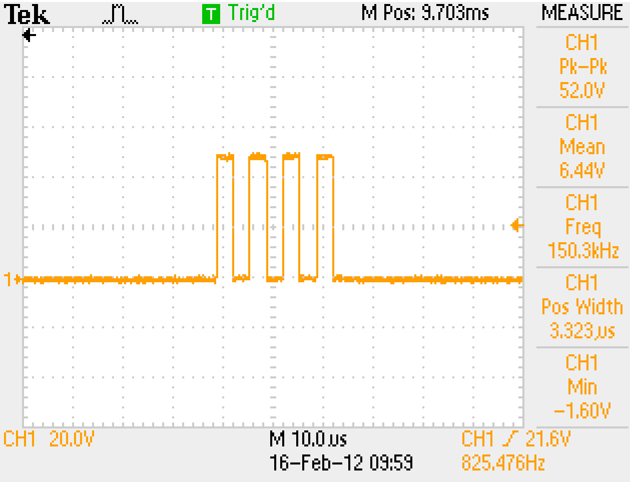
CH1 20.0V M 250ns CH1 ⁄ –800mV

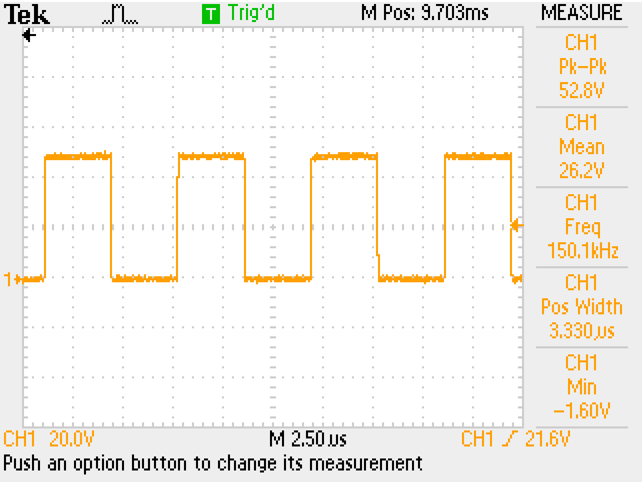
16–Feb–12 10:29 799.838Hz

DragonBoard Output

- Burst of four square waves with 2500 ns period (constraint of the TCNT)
- Multiple bursts located 500 µs apart
- Approximately 50 V peak to peak

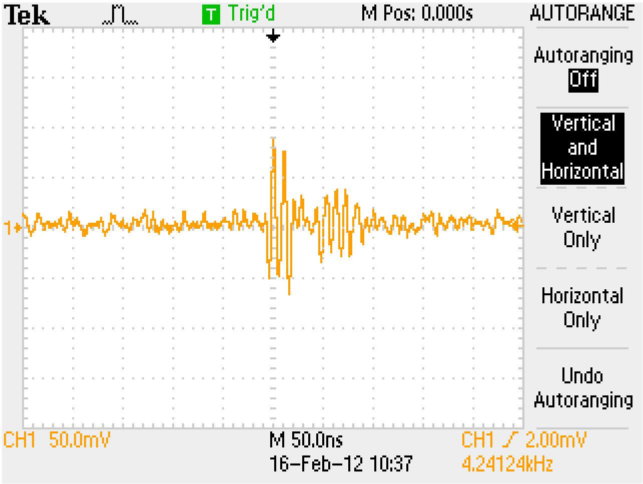






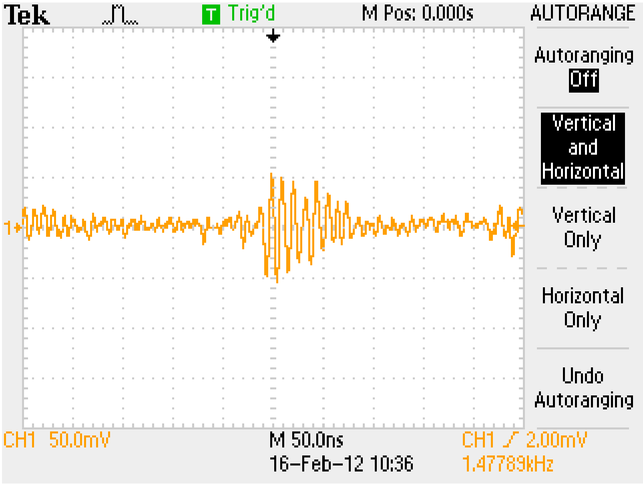
Sensor Output

- Sensor output in air
- Proof of concept, real testing and refinement will be done on the test apparatus



M Pos: 0.000s
CH1 50.0mV
M 50.0ns
16–Feb–12 10:37
CH1 ∕ 2.00mV
4.24124kHz

Flow Meter Output



M Pos: 0.000s
CH1 50.0mV
M 50.0ns
16–Feb–12 10:36
CH1 ∕ 2.00mV
1.47789kHz

DragonBoard Output

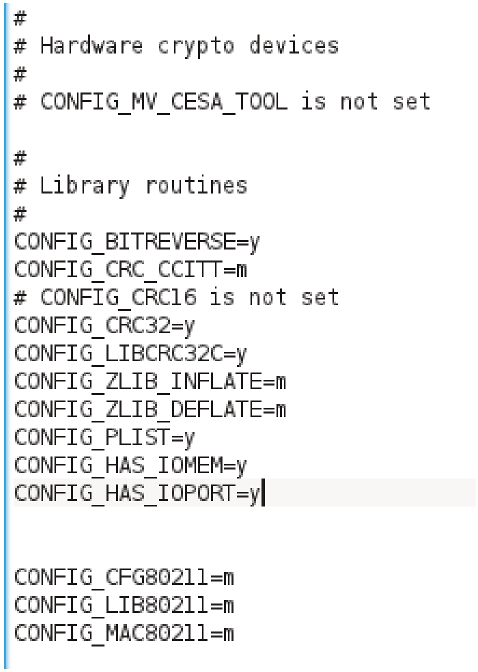
Wi-Fi on Single Board PC

- Original Wi-Fi USB adapter shipped from Technologic Systems did not support master mode with Linux driver, just Ad-Hoc.
- **Problem**: Most Android phones do not connect to Ad-Hoc networks with out rooting your phone.
- **Solution**: New Wi-Fi module ordered:

Penguin Wireless N USB Adapter for GNU/Linux

Wi-Fi on Single Board PC

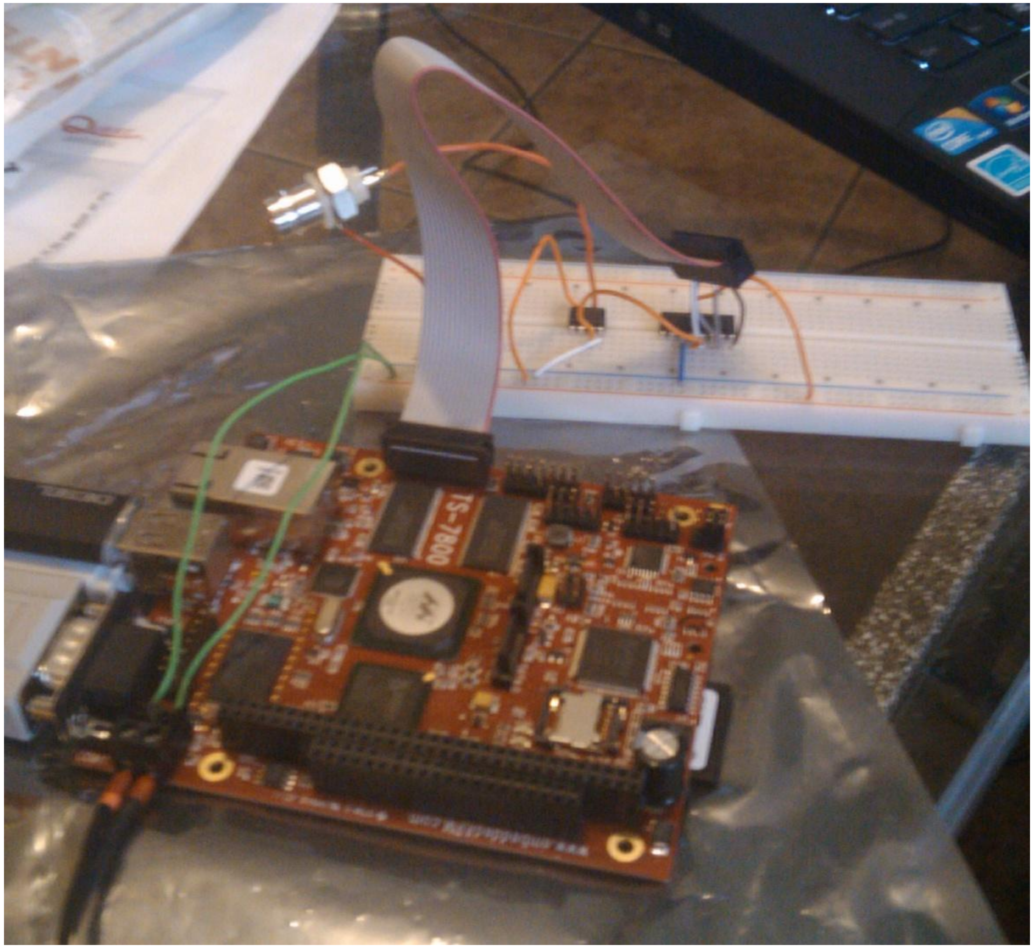
- **Problem**: Penguin adapter drivers require Linux kernel 2.6.37+. Technologic Systems only supports up to Linux kernel 2.6.34. (Most board components such as the SD Card are buggy on newer kernels)
- **Solution**: Cross compile own drivers with ARM architecture for 2.6.34.
- Drivers were compiled, but currently having trouble getting hostapd to work with firmware.
- Hostapd is what implements access point management in order to turn the Wi-Fi module into an access point.
- Matter of time. Cross compiling and transferring from Linux machine to SBC via SD Card is very time consuming.



```
#
# Hardware crypto devices
#
# CONFIG_MV_CESA_TOOL is not set

#
# Library routines
#
CONFIG_BITREVERSE=y
CONFIG_CRC_CCITT=m
# CONFIG_CRC16 is not set
CONFIG_CRC32=y
CONFIG_LIBCRC32C=y
CONFIG_ZLIB_INFLATE=m
CONFIG_ZLIB_DEFLATE=m
CONFIG_PLIST=y
CONFIG_HAS_IOMEM=y
CONFIG_HAS_IOPORT=y


CONFIG_CFG80211=m
CONFIG_LIB80211=m
CONFIG_MAC80211=m
```

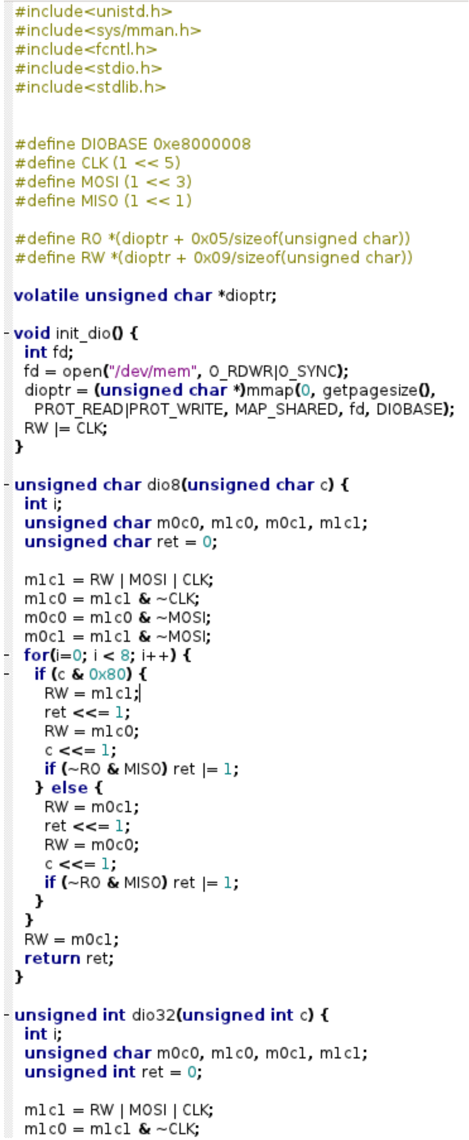
Signal Generation From SBPC

- Preliminary circuit built.
- Not yet tested in lab.
- Still figuring out how to control Digital I/O from user space programs.



Signal Generation From SBPC

- Operating System built on top of hardware => cannot access hardware directly.
- Memory mapped IO is used in order to access hardware from user space.
- Still figuring out how to control Digital I/O from user space programs.
- mmap() system call is used in order to achieve this.

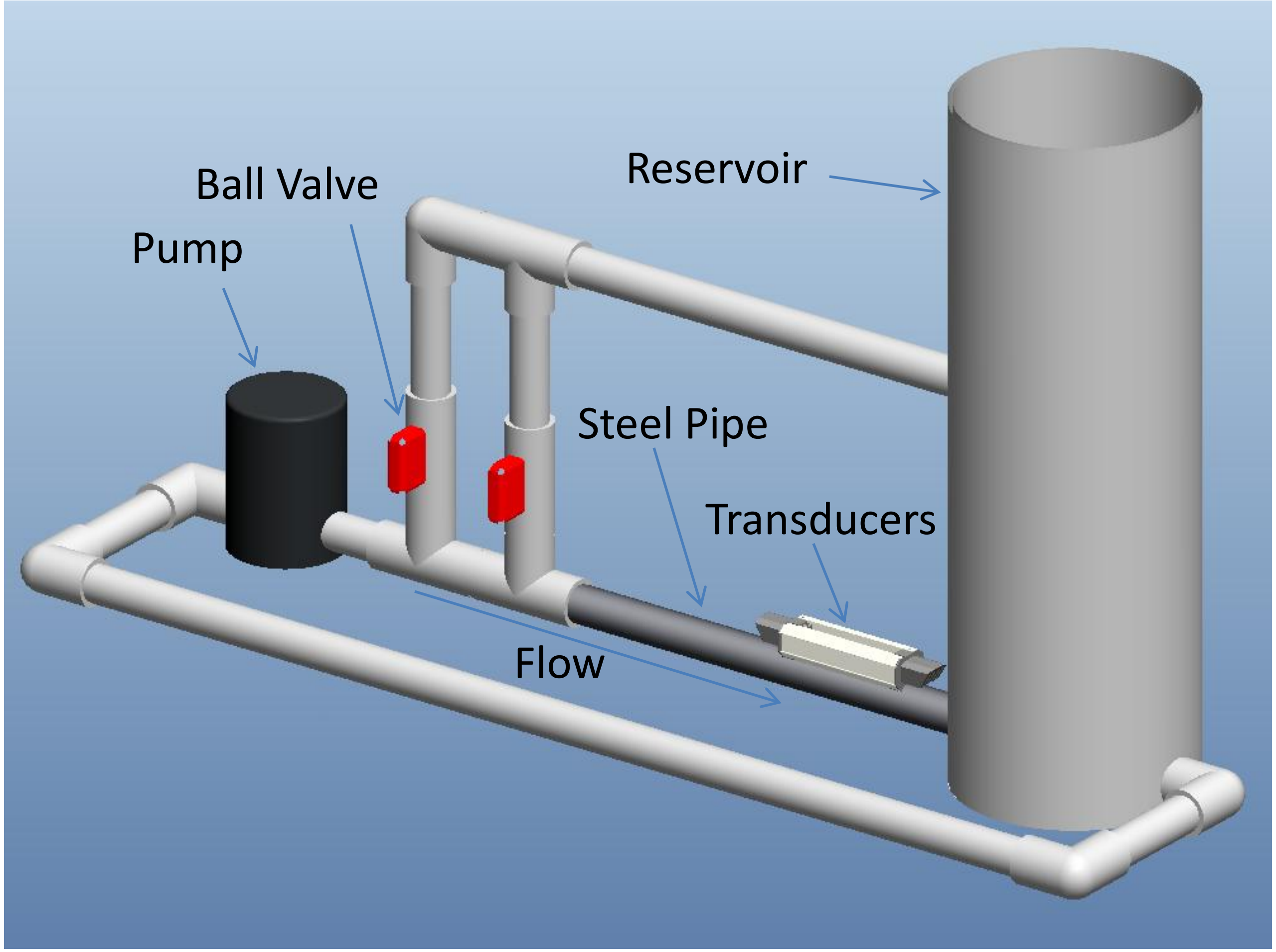


```c
#include<unistd.h>
#include<sys/mman.h>
#include<fcntl.h>
#include<stdio.h>
#include<stdlib.h>


#define DIOBASE 0xe8000008
#define CLK (1 << 5)
#define MOSI (1 << 3)
#define MISO (1 << 1)

#define RO *(dioptr + 0x05/sizeof(unsigned char))
#define RW *(dioptr + 0x09/sizeof(unsigned char))

volatile unsigned char *dioptr;

void init_dio() {
 int fd;
 fd = open("/dev/mem", O_RDWR|O_SYNC);
 dioptr = (unsigned char *)mmap(0, getpagesize(),
   PROT_READ|PROT_WRITE, MAP_SHARED, fd, DIOBASE);
 RW |= CLK;
}

unsigned char dio8(unsigned char c) {
 int i;
 unsigned char m0c0, m1c0, m0c1, m1c1;
 unsigned char ret = 0;

 m1c1 = RW | MOSI | CLK;
 m1c0 = m1c1 & ~CLK;
 m0c0 = m1c0 & ~MOSI;
 m0c1 = m1c1 & ~MOSI;
 for(i=0; i < 8; i++) {
   if (c & 0x80) {
     RW = m1c1;
     ret <<= 1;
     RW = m1c0;
     c <<= 1;
     if (~RO & MISO) ret |= 1;
   } else {
     RW = m0c1;
     ret <<= 1;
     RW = m0c0;
     c <<= 1;
     if (~RO & MISO) ret |= 1;
   }
 }
 RW = m0c1;
 return ret;
}

unsigned int dio32(unsigned int c) {
 int i;
 unsigned char m0c0, m1c0, m0c1, m1c1;
 unsigned int ret = 0;

 m1c1 = RW | MOSI | CLK;
 m1c0 = m1c1 & ~CLK;
```

Flow Test Apparatus

- Need a test apparatus for development, calibration, and demonstration of instrumentation and phone app
  - Consistent conditions needed to develop the entire system
  - Ability to adjust flow will aid calibration process
- Our instrument ultimately needs to measure flow of natural gas (air)
- Main goal is to prove the concept
- We will start with a setup to measure water flow
  - Our transducers are tuned to operate at the higher frequency needed for liquids (due to higher speed of sound in liquid medium)

Flow Test Apparatus



Flow Test Apparatus

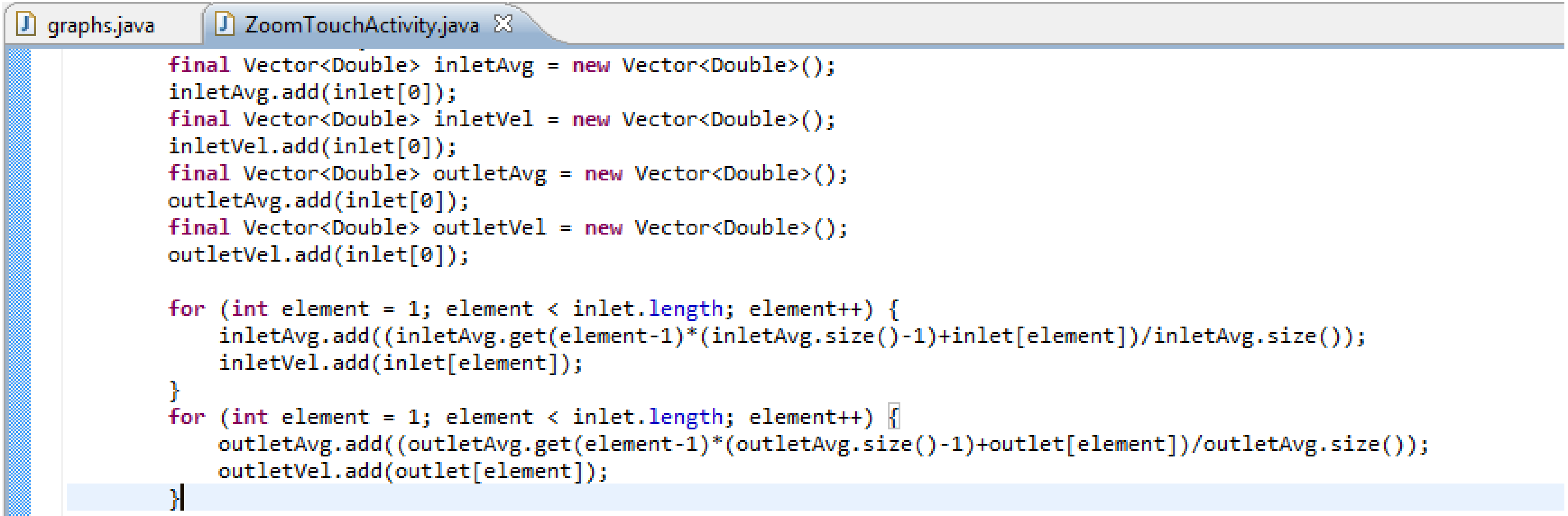
- Once concept is proven, we can attempt measurement in air
  - Speed of sound in air is lower (1117 ft/s, vs. 4814 ft/s for water)
  - May or may not require transducers that operate at lower frequency
  - Since SBPC can handle the higher frequencies needed for liquid measurements, it can easily handle any good frequency for air

Mobile Application

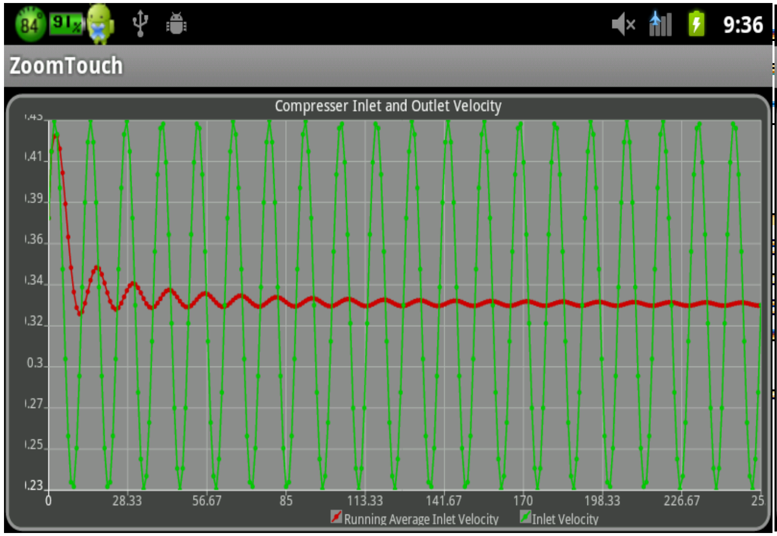
- The velocity inside the compressor pipes are expected to fluctuate by some small amount but we are only interested in large fluctuations
- Need to make a running average algorithm for our incoming velocity data to normalize and eliminate small fluctuations in the data
- Derived formula for the average of all points thus far at any given point

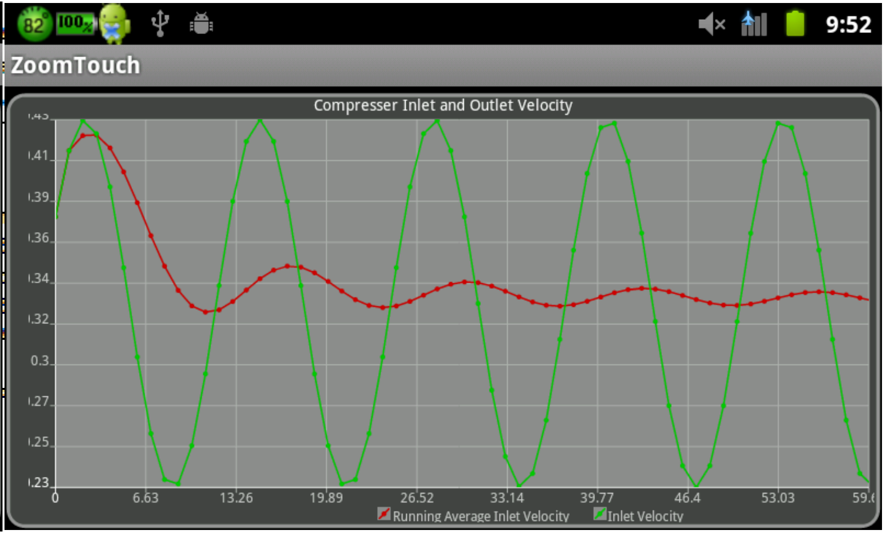
$$Current_{avg} = \frac{(Prev_{avg} \times Prev_{total}) + new\ point}{Current_{total}}$$

Mobile Application



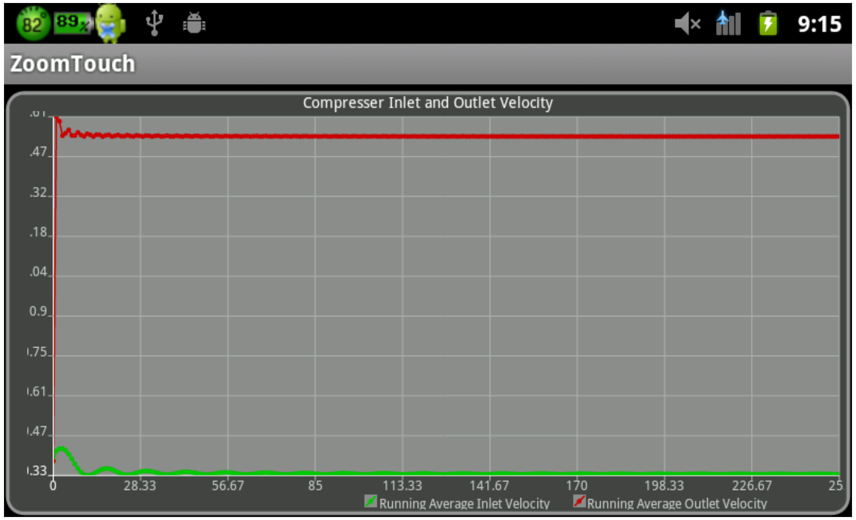
```
        final Vector<Double> inletAvg = new Vector<Double>();
        inletAvg.add(inlet[0]);
        final Vector<Double> inletVel = new Vector<Double>();
        inletVel.add(inlet[0]);
        final Vector<Double> outletAvg = new Vector<Double>();
        outletAvg.add(inlet[0]);
        final Vector<Double> outletVel = new Vector<Double>();
        outletVel.add(inlet[0]);

        for (int element = 1; element < inlet.length; element++) {
            inletAvg.add((inletAvg.get(element-1)*(inletAvg.size()-1)+inlet[element])/inletAvg.size());
            inletVel.add(inlet[element]);
        }
        for (int element = 1; element < inlet.length; element++) {
            outletAvg.add((outletAvg.get(element-1)*(outletAvg.size()-1)+outlet[element])/outletAvg.size());
            outletVel.add(outlet[element]);
        }
```

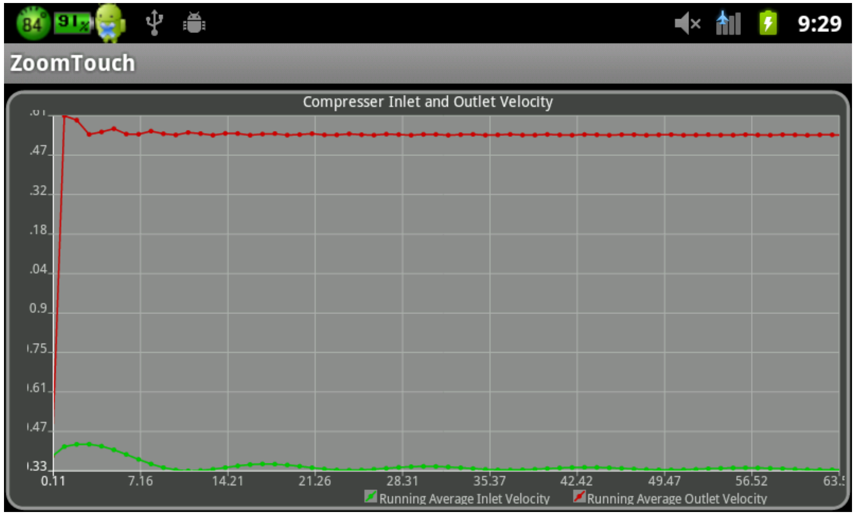




Mobile Application

- Realistic Graph of what we expect our velocity measurements to look like
- Includes inlet and outlet velocities
- Graph also has built in zooming function which changes the x-axis
- Future work will include the y-axis (i.e. full zooming into a specific point)


9:15
ZoomTouch
Compresser Inlet and Outlet Velocity
.47
.32
.18
.04
0.9
.75
.61
.47
.33
0
28.33
56.67
85
113.33
141.67
170
198.33
226.67
25
Running Average Inlet Velocity
Running Average Outlet Velocity

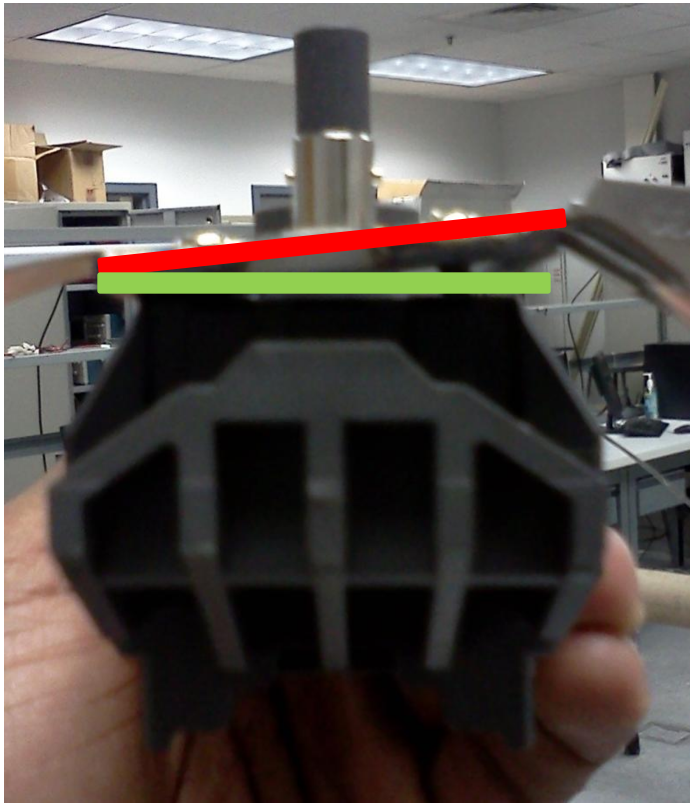

9:29
ZoomTouch
Compresser Inlet and Outlet Velocity
.47
.32
.18
.04
0.9
.75
.61
.47
.33
0.11
7.16
14.21
21.26
28.31
35.37
42.42
49.47
56.52
63.
Running Average Inlet Velocity
Running Average Outlet Velocity

Mobile Application

- Next steps: working on setting up laptop to send data to phone to use as testing tool
- Will use this to test reception of the data and proper storage into the database of phone
- Have been encountering difficulties with type mismatching while coding. Foresee issues with data type being stored in database and type needed for androidplot graphing functions.
- Might have to modify database to store in specific data type to simplify type-conversions during the calculation and graphing stages.

Results from Testing the Mounting System

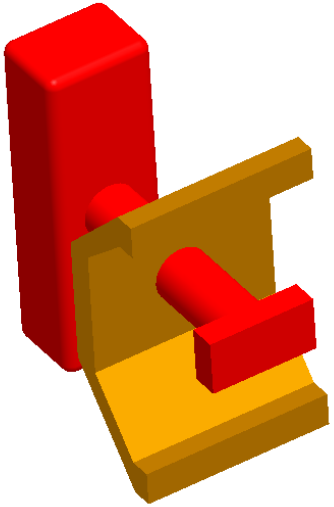
- Satisfied second customer need
- Discovered modifications needed for proper fitting Mounting System
- Developed procedures for easier mounting



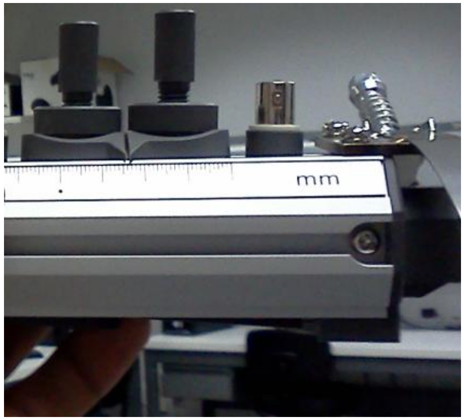
Unlevel attachment

Housing Unit Progress

- Method for attaching the housing unit has been selected
- Seen in figure 2, the design features four oute track grips with Keys
- Dimensions for the housing unit are to be determined

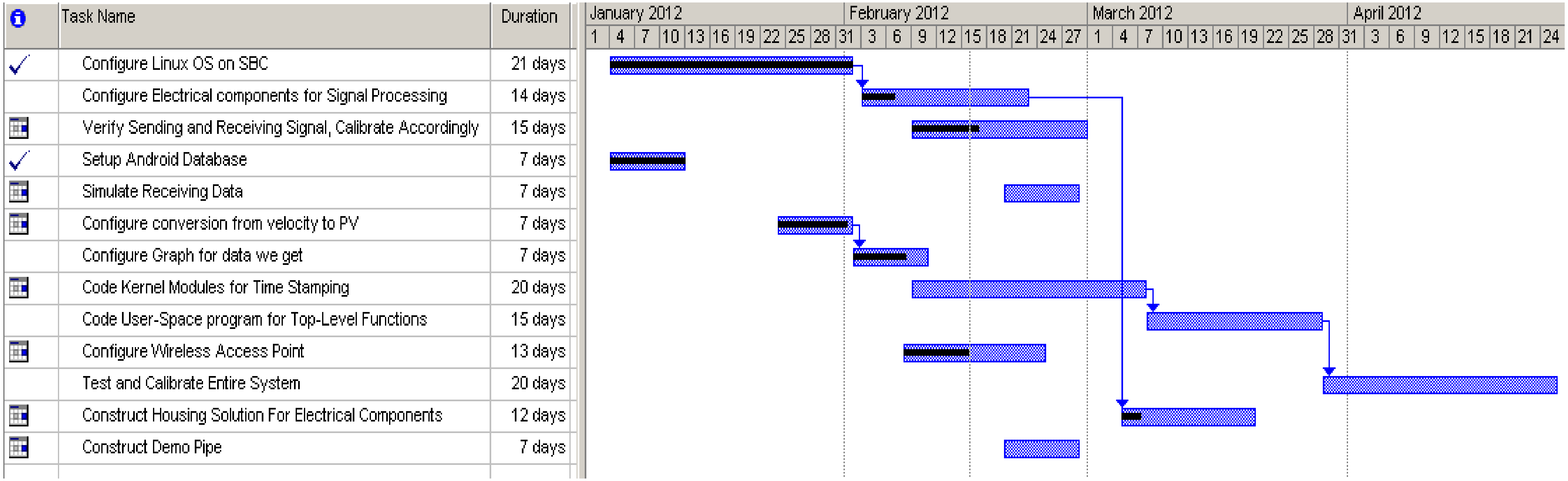


Outer Track Grips



Track of Transducers

Project Plan



ℹ	Task Name	Duration
✓	Configure Linux OS on SBC	21 days
	Configure Electrical components for Signal Processing	14 days
▦	Verify Sending and Receiving Signal, Calibrate Accordingly	15 days
✓	Setup Android Database	7 days
▦	Simulate Receiving Data	7 days
▦	Configure conversion from velocity to PV	7 days
	Configure Graph for data we get	7 days
▦	Code Kernel Modules for Time Stamping	20 days
	Code User-Space program for Top-Level Functions	15 days
▦	Configure Wireless Access Point	13 days
	Test and Calibrate Entire System	20 days
▦	Construct Housing Solution For Electrical Components	12 days
▦	Construct Demo Pipe	7 days

Questions

Image Provided by: http://www.datingadvice4christiansingles.com/image-files/askaquestion.jpg