

# On-Site Hotbox Calibration System

---

ECE445 – Team 2  
Final Report

**Team Members:**

**Pourya Assem**

**Paul Lupas**

**TA: Rajarshi Roy**

**Prof. A. C. Singer**

**12/12/2012**

## Contents

Introduction.....	3
Title.....	3
Problem Statement.....	3
Objectives .....	4
Wheels' Timing Calculations .....	5
High-Level System Block Diagram.....	6
Amplifiers Low-Level System Block Diagram .....	6
Transducer A/B - Signal Processing.....	7
Fuses + Transformers.....	7
Amplifiers / Clippers.....	10
Level Restorer / Slicer / Logic Converter .....	12
IR Sensor - Signal Processing.....	14
Fuses + Active Filter.....	14
TX Microprocessor.....	17
Bootloading Atmega328 – Fuse Configuration .....	17
Programming.....	17
Power Supply TX/RX.....	19
TX XBEE.....	22
Network ID Configuration.....	22
RX XBEE .....	24
Network ID Configuration.....	24
Multiplexer.....	24
GLCD.....	26
SD Card.....	28
Keypad.....	29
RX Microprocessor.....	30
Bootloading Atmega328 – Fuse Configuration .....	30
Programming.....	30
PCB Design.....	30
Field Compatibility.....	33
Cost Analysis .....	33

Labor.....	33
Parts .....	33
Grand Total .....	34
Testing Results/Accomplishments.....	34
Improvements .....	34
Difficulties / Uncertainties.....	36
Conclusion .....	36
Wrap-up and future work.....	36
Ethical considerations .....	36
References –need to work on.....	37
Appendix A – TX Unit Mother Board.....	38
Appendix B – RX Unit .....	38
Appendix C – TX Unit Signal Board.....	39
Appendix D – TX Processor Program .....	39
Appendix E – RX Processor Program.....	40
Appendix F – Testing.....	41

# Introduction

## Title

This project was chosen because of a high demand and immediate need in railroad industry to shift to an efficient and accurate heat sensor calibration procedure. Frequent inspection of train wheels and bearings is enforced by Federal Laws. This is to prevent any malfunction that can lead to a train accident because of overheated wheels and bearings. A massive network of reliable data acquisition system is essential to carry out this daily inspection. A reliable system needs a standardized reliable calibration procedure. This is the missing tile, and we decided to make a system that aids the industry to ease this process. We are excited to make a significant improvement in the calibration procedure. We are convinced that this product will be very marketable because it is a necessary piece of equipment, which has not been implemented before. Furthermore, CN Railroad, which we are in contact with, has showed a high interest to buy this device upon successful completion with rated specifications.



Fig. 1 Hotbox unit on track [1]

## Problem Statement

Figure.1 shows an ordinary heat sensor system - known as HotBox - setup on a North American track. This system is to be tuned, employing the calibration device we have proposed. The HotBox system consists of two gating transducers which wake the system's processing unit of wheel/bearing presence in the sight of IR heat sensors. The sensors are designated to read the heat signature of the wheel/bearing perimeters as they pass over them.

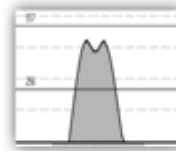


Fig.2 Standard Wheel Signature [1]

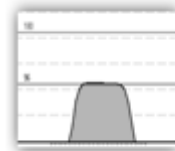
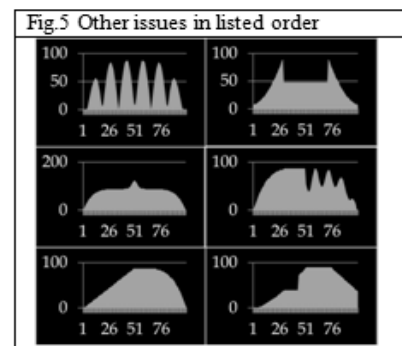
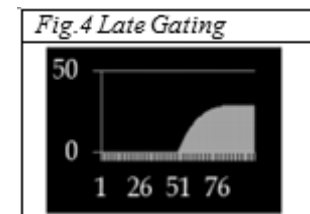


Fig.3 Standard Bearing Signature [1]

Figure.2 shows a standard heat signature of a normal wheel. Figure.3 shows a standard heat signature of a normal bearing. As shown, the heat levels are below the red alarm line.

The HotBox system goes out of calibration very often. The mean calibration lifetime is less than a month. Sensors and transducers units are continuously under hammering shocks exerted by the passing trains, which displace the sensing/timing units from their designated positions. A very common issue is that when the displaced gating transducers awake early/ late the processing units, which results in loss of a hefty portion of heat signature. Among other common issues:

- The heat sensors have internal faults.
- The housing connections are loose which cause abruptions in the transmission of heat signatures.
- The heat sensors are shifted and looking at the wrong sight.
- The sensors are looking into sun.



These are the most critical everyday issues that railroad industry is suffering from. The proposed calibration device will be used as an aid to visualize the heat signature signals, which are collected by the sensors in the field to diagnose all the issues addressed above. The basic idea is to fetch/log the heat signatures read by the IR sensors when the gating transducers are activated during the valid scanning time window. Note that the current technology is all mechanical and does not have the ability to fetch and generate such plots.

## Objectives

### Goals

- Shift the railroad industry from all mechanical to all electrical sensor calibration devices
- Guide servicemen through the calibration process by visualizing the heat signatures
- Faster and more accurate calibration process compared to the current mechanical methods
- Create a universal standard for integrating the calibration process

### Functions

- Acquire data from gate opening and closing transducers
- Acquire data from the IR heat sensors
- Ensure proper timing synchronization between transducers and IR heat sensors
- Ensure proper signal level of transducers and IR heat sensors
- Ensure proper waveform read by the IR heat sensors.

### Benefits

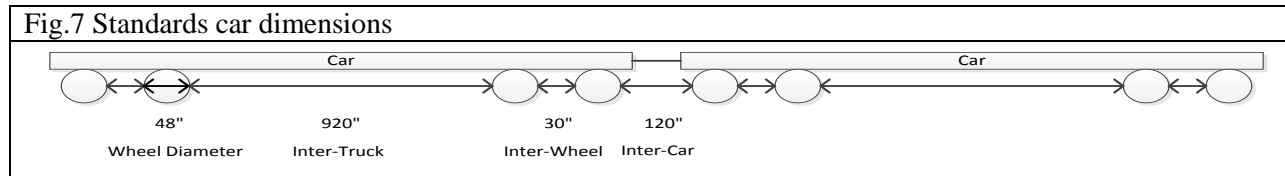
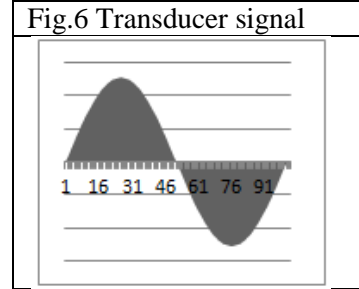
- Service time efficiency due to on-site data analysis and fast feedback
- Service cost efficiency due to time efficient method and higher calibration quality
- Less data traffic on servers because servicemen do not need to collect data from the servers
- Relatively cheap device ~\$200, compared to mechanical calibration tool kit
- Wireless data transmission allows service man to keep a safe distance from traveling train
- Wireless data transmission allows the service man to collect data in critical weather conditions and unreachable bungalow locations

### Features

- Wireless data transmission
- SD card data storage
- Graphical interface
- Battery powered
- Small handheld unit

## Wheels' Timing Calculations

In the Hotbox system, transducer blocks are responsible for generating a gating signal, which indicates/times the presence of the train wheel/bearing on top of the IR heat sensors. The transducer box is a simple electromagnetically activated hammer, which generates/sends a short analog pulse - one sine period - when a wheel passes over them. The transducers' gating pulse activates a shutter covering the IR sensors windows. To the right is an example of transducer signal as the wheel passes by.



The picture above shows the extreme dimensions of a standard train car operated in Northern America. There are five major sizes that characterize the metrics used in designing the system [1].

- Inter-Truck Distance → 920 inches
- Inter-Car Distance → 120 inches
- Inter-Wheel Distance → 30 inches
- Wheel Diameter → 48 inches
- Closing-Opening Transducer Spacing → 50 inches

The transducers will trigger when the center of the wheel is aligned with the center of transducers. The calculations below are carried out to find the shortest time span between the triggering pulses coming from the transducers. The time span is a function of the train's speed and the spacing addressed above. The two speed extremes are 20 mph for the lower and 60 mph for the upper bound.

The shortest time is calculated for the train going at the fastest speed, when a wheel passes a closing transducer and the next wheel passes the opening transducer. This distance is the axle to axle spacing of the two closest wheels minus the transducers spacing, which is  $24 + 30 + 24 - 50 = 28$  inches. The top speed is  $60 \text{ mph} = 1056 \text{ in/sec}$ , which means the minimum transducer timing is 26.5ms. The longest time is calculated for the train going at the slowest speed, when the two furthest (truck to truck) wheels pass over the transducers. The longest distance is  $24 + 920 + 24 = 968$  inches. The train's slowest speed is  $20 \text{ mph} = 352 \text{ in/sec}$ . This results in 2.8sec.

As calculated, the minimum time was 26.5ms. The algorithm developed in the TX Unit should be fast enough to distinguish transducer pulses 26.5ms apart from each other. The microprocessor used in the TX Unit is capable of processing or reading digital pulses as short as a tenth of a millisecond.

The transducer spacing is set to 50 inches, which results in wheel scan time of 47ms for the train going at 60 mph. A minimum number of 12 samples are needed for each wheel scan, which requires minimum read time of 4ms for each sample. The ADC should be fast enough to handle such conversion rate. The microprocessor ADC has conversion rate of 10 kilo samples per second which satisfies the requirement.

## High-Level System Block Diagram

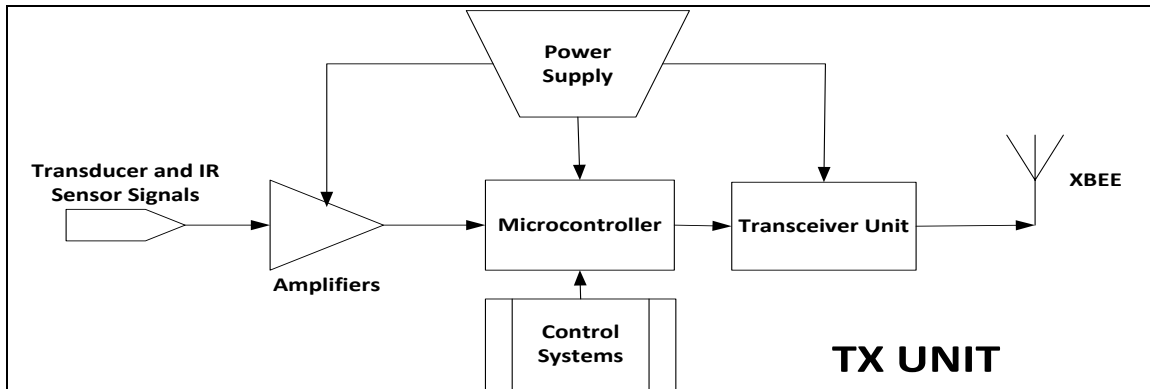


Fig.8 TX unit, look at appendix A for circuit diagram.

Note: This block minus the Amplifiers is implemented as the Mother board.

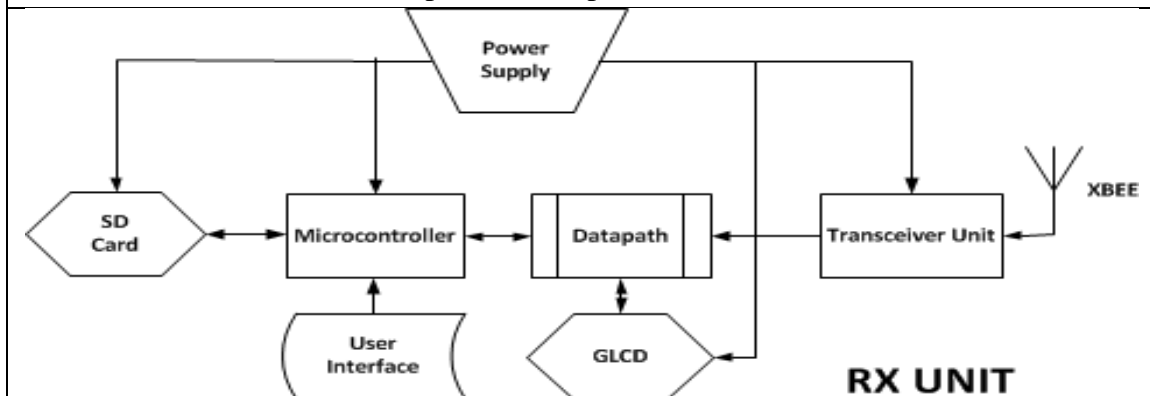


Fig.9 RX unit, look at appendix B for circuit diagram.

Note: This block is implemented as the RX board.

## Amplifiers Low-Level System Block Diagram

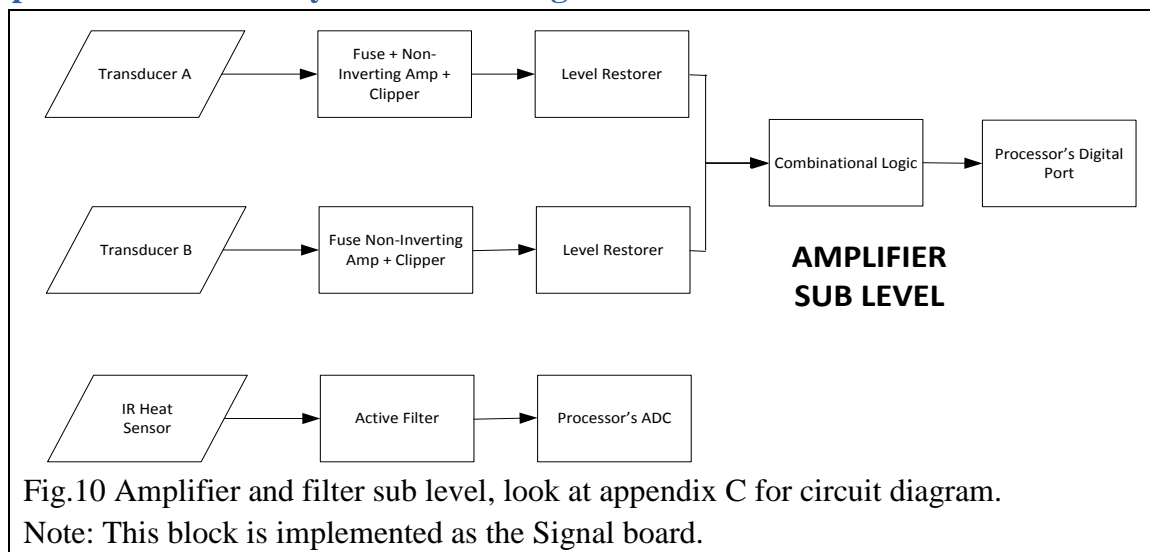


Fig.10 Amplifier and filter sub level, look at appendix C for circuit diagram.

Note: This block is implemented as the Signal board.

## Transducer A/B - Signal Processing

The mere purpose of this block is to convert the transducer signals into an acceptable waveform than can be seen/processed by the microcontroller's digital ports. The transducer signals will be passed through a 1:1 transformer to meet the requirements specified by the factory, which is perfect electrical isolation by the third party user [1]. Then the signal is passed through an amplifier which has a dual function of both amplifying and clipping the unwanted part of the signal. The next stage consists of a slicing and concatenating the opening and closing transducer signal. This stage also reshapes the ramps on the output of amplifiers to a perfect TTL logic. The processed signals are then send to the processor to indicate a change of state in either of the transducers.

It is a very common practice to use a low power fuse when sensitive electronic device is connected to equipment with direct contact to the tracks, because of:

- Danger of lightning
- Electrical charge produced by the passing trains
- High voltage signals sent through the tracks accidentally by the crewman

Our system is highly sensitive to high voltage spikes due to usage of integrated circuits in the main blocks of data acquisition structure. Therefore, we inserted a 0.5A fuse interconnecting the transformers to the amplifier/clipper blocks.

Note: The heat signature circuit does not include a transformer and fuse, because it is implemented within the breakout circuit inside the heat scanners.

## Fuses + Transformers

[Equation](#) / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / [Quantitative Results and Graphs](#) / [Interference with rest of the project](#)

It was decided to replace the dual line - Twisted Wire Toroid - transformer with a single line high impedance audio transformer in order to:

- Avoid handmade component and tolerance due to inaccurate crafting process. Avoid large magnetic field produced by the Toroid, which can tolerate nearby electronics. Reduce size.
- Provide larger input impedance at low frequencies to minimize the current drain from DUT. Prevent signal distortion and loading effect due to large current drain.
- Provide large impedance on output node, so the transformer can be seen as a current source.
- Provide large coupling factor, and minimum phase shift.

The audio transformer has a rated 10K $\Omega$ :10K $\Omega$  input/output impedance at desired operating frequency, and capable of handling analog data within the range of:

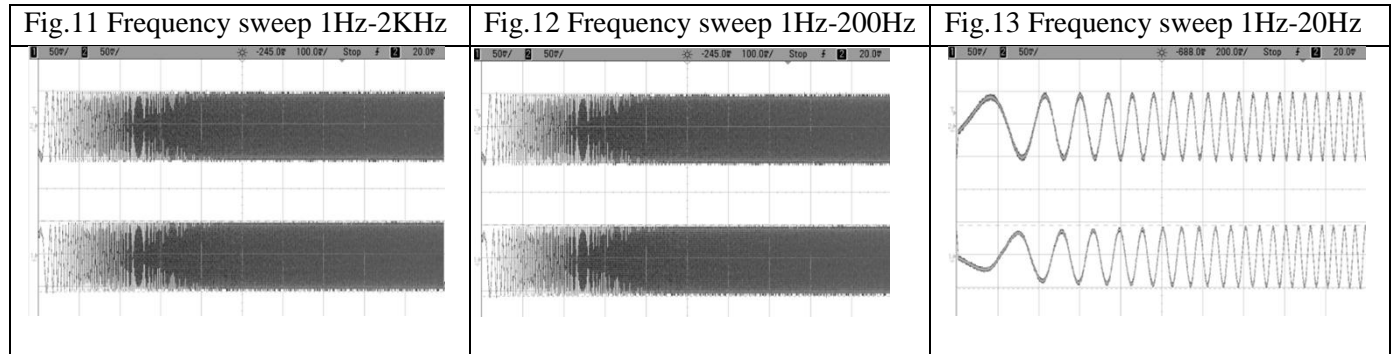
- 5Hz-3500Hz (this system operates on 10Hz-500Hz)
- 10mVpp-10Vpp (this system operates on 50mVpp-5Vpp)

These ratings are well fitted in the operating range of this system. The loss factor and coupling strength is proportional to input frequency range and voltage swing.



Transformers were tested for qualitative performance. The test setup is as follows:

- Use two function generators (uncoupled at random phase) to feed the transformers @ 50mVpp Sine - 50Ω terminal. Use the following setting, and trace the output signals to check for voltage transfer ratio + phase factor.



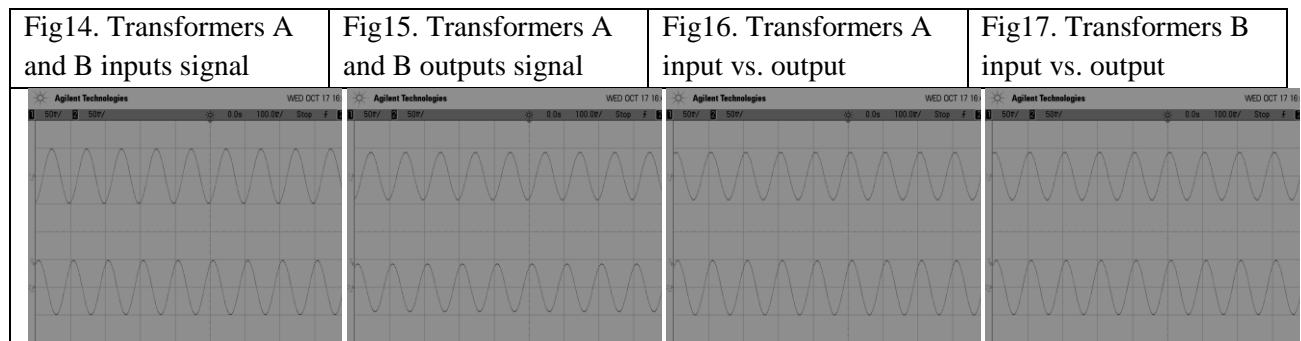
- As the measurements reflect, the coupling factor drops for frequencies below 5Hz. The drop is not significantly large (10% at 1Hz), and follows in a slow linear trend. Even though the system does not operate at such low rates, but it adds to the immunity of the system performance. The coupling factor is nearly 100% for frequencies above 5Hz, so the transformer voltage spice model can be approximated as (At all frequencies, due to large relative input/output inductance, and consequently impedance compare to driver and load):

$$X_L = j\omega L \gg R_{Load} \& R_{Source} \rightarrow |V_{in}| = |V_{out}|$$

- Surprisingly the phase shift of the output signal (due to inductive behavior of transformers), is significantly small compare to driving frequency. Additionally, this can be verified by driving the transformer using a square wave or pulse burst, to check for input/output delay (Note: in the measurements above, channel 2 is inverted and the signals are 180° out of phase).

$$V_{in} \approx V_{out}$$

Following are some more measurements results showing the transformer under same setting @ 10Hz (system's typical operating frequency):



The transformers are:

- Fed directly by the Gating Transducer output's transformer (with no common ground as required by manufacturer).
- Driving the amplifiers/clipper (fused by 0.5A glass tube) to immune the amplifier/clipper circuit to high voltage spikes.

Note: the transducers output signal is no more than 250mVpp [1], so the amplifiers' input never pass the available headroom ( $3.3Vp - 1.3Vp = 2Vp \rightarrow 4Vpp$  limit).

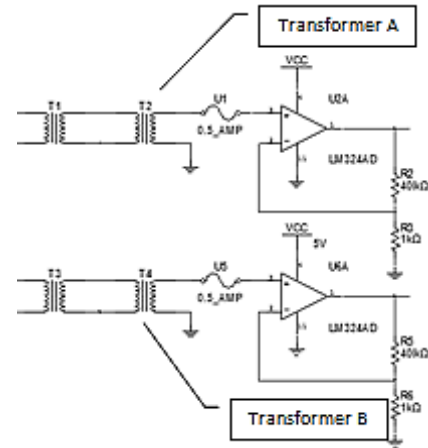


Fig18. Transformers in the system

<p>Fig19. Transducer signal simulator (100mVpp)</p>	<p>Fig20. Transformer's output vs. amplified/clipped signal (10Hz, 100mVpp)</p>
<p><b>Verification</b></p> <ol style="list-style-type: none"> <li>1. Wrap the twisted wire transformer (choke) as specified by the calculations. Make sure each leg of the twisted pair is of the same size, 1 meter. There should be exactly 50 turns on each transformer and make sure the wires are tightly wrapped on the core with a minimum of 30 degrees spacing between the terminals.           <p>Replaced with 10KΩ: 10KΩ Audio Transformer.</p> </li> <li>2. Connect the input of the transformer to a function generator @ 5Vpp - 10Hz sinusoid and trace the output with the resolution set on averaging. This signal should be @ 3.83mVpp – 10Hz sinusoid with maximum phase shift of 6 degrees w.r.t function generator.           <p>Passed, with new additional testing procedures. See testing procedure.</p> </li> <li>3. Set the 2<sup>nd</sup> channel of the oscilloscope to trace the input signal of the transformer. This signal should be @ 3.83mVpp – 10Hz sinusoid. Use the Math function to subtract the input and output signals to ensure a perfect match. This indicates a high coupling factor.           <p>Passed, with new additional testing procedures. See testing procedure.</p> </li> <li>4. Increase the function generator frequency to 20Hz and repeat step 3 (this time @ 7.65mVpp) to check for coupling factor.           <p>Passed, with new additional testing procedures. See testing procedure.</p> </li> <li>5. Repeat step 3 for frequencies up to 500Hz.           <p>Passed, with new additional testing procedures. See testing procedure.</p> </li> <li>6. If the coupling factor differs or decays over the frequency span, repeat step 1 with a core (better low frequency handling characteristics) and ensure the twisted wires are tightly wrapped.           <p>Replaced with 10KΩ: 10KΩ Audio Transformer.</p> </li> <li>7. Use two DMMs to monitor the current and voltage on the input of transformer and find their</li> </ol>	

product to ensure the drain power is below 400mW.

Passed.

8. If the specs on step 7 are not met, increase the number of turns and repeat steps 1-7 (iteratively and calculate new voltage values on the output using the equations provided).

Replaced with 10KΩ: 10KΩ Audio Transformer.

9. If the above steps are satisfied, make a second transformer.

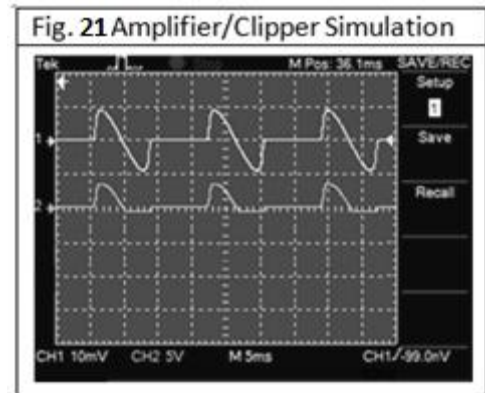
Replaced with 10KΩ: 10KΩ Audio Transformer.

10. For more accurate measurements, use a Network Analyzer to check for all the specs.

11. Connect the fuse to the output of the transformer and trace the output @ 10Hz and 20Hz to ensure the values match with the voltages provided in steps 3 and 4. If not, change the fuse and repeat step 11.

## Amplifiers / Clippers

The voltage directed to this block is a purely AC signal coming from the transformer output. However, this block operates on a DC level signal, because the amplifier is biased to process the signals ranging from 0 to 3.3V. By this mean, the negative going of the transducer signal (half sine wave) will be clipped and the positive going portion of the signal will be amplified near saturation regime (for some signal levels clipped on top) to reshape the positive going (half sine wave) of the transducer pulse into a nearly square shape signal. The amplifier is implemented using LM324 Quad Op-Amp chip in the non-inverting configuration with gain approximately 41. The circuit is tested for quality and proper signal level using a Function Generator to generate different input signals as specified in the Verification procedures and the measured output is observed on Oscilloscope.



Above is the simulation of the transducer amplifiers. The top waveform is the signal coming from transformer (20mVpp 27Hz sinusoid signal), input signal of amplifier. The bottom waveform is the output of the transducer amplifier. It is observed that the negative going of the input signal is clipped and the positive going is amplified to saturation. The same results are expected from measurements.

Fig.22 Amplifiers circuit diagram

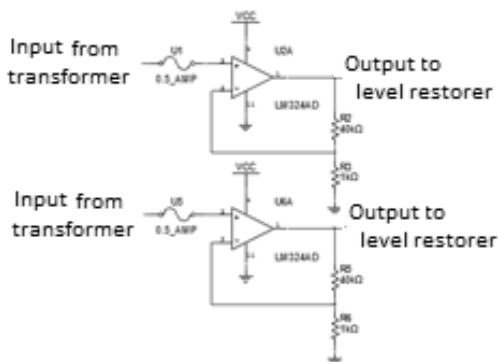


Fig. 23 Output of amplifiers with 30% saturation

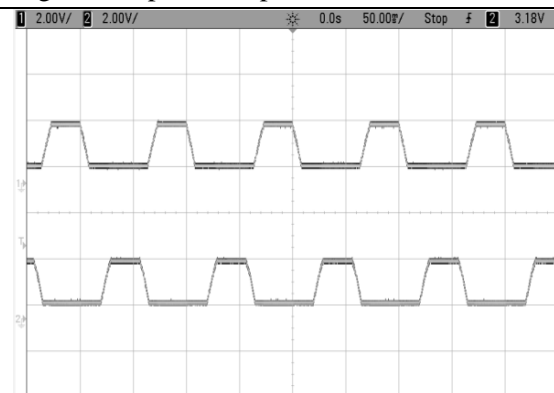


Fig.23 Output Noise with grounded input

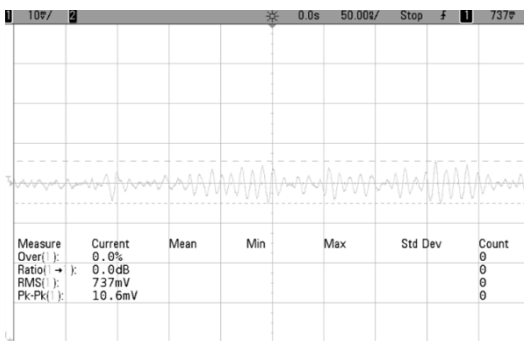
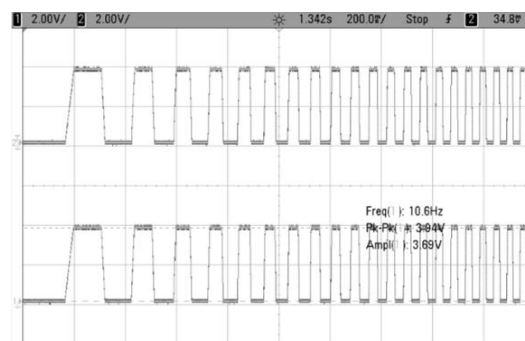


Fig.24 Frequency Sweep 1Hz-10Hz



$$Gain = 1 + \frac{40K\Omega}{1K\Omega} = 41$$

The saturation limit of the amplifier output is 2.3V. The amplifiers are tuned to have a 30% saturation on the output for 100mVpp 10Hz sinusoidal input signal which is the typical transducer signal used by North American Railroad Companies. Figure above shows that the amplified signal is 30% in saturation. This means that the amplifier cannot amplify the signal any higher than 2V. The rail voltage is 3.3V. It can be observed that the LM324 uses around 1.3V internally since the rail voltage is 3.3V and the amplification saturation is at 2V. To have signal amplitude of 3.3V, the output of this stage will be fed into a level restorer circuit to be further amplified and create a proper digital signal. The signal fed to the level restorer circuit will not exceed 2V because of the saturation limit of 2V.

## Verification

- Construct the amplifier as shown in figure and use a DMM to monitor the current being drained by the Quad Amplifier Chip (LM324).  
The resistors were implemented using 1MΩ potentiometer to adjust the gain more easily. The current drained by the Quad Amplifier Chip measured with the DMM is 8.1mA when circuit is driving.
- Set the potentiometer to obtain a gain of 200 and check the input/output response as described in the next step.
- Set the function generator @ 20mVpp - 10Hz sinusoid. Amplify the signal using the amplifier built and tuned in step 1-2 and trace the output on the oscilloscope to have the right frequency and non-inverted amplitude of 2Vpp - clipped with maximum phase shift of 6 degrees.  
The output signal is 30% in saturation. The amplitude is 2Vpp.
- Repeat step 3 with 20Hz (amplifier output @ 4Vpp - clipped).
- Change the gain to 41 and repeat steps 3 and 4, and check for output voltages of 4.01Vpp and approximately 4.70Vpp correspondingly.
- Note: The amplifier power rails are biased from 0 to 5VDC, and signal coming from generator is purely AC. So the signal will be clipped (rectified) in steps 4 and 5.
- If the voltage levels are not within the mentioned values in step 5, increase the gain above 41.  
The power supply was changed to 3.3VDC. There is no need to raise the gain because the output

signal is already 30% in saturation.

8. Consistently, during all these steps, check for the current drained by the amplifier to ensure it does not exceed 30mA.

A DMM was used throughout these steps and the current did not exceed 30mA.

9. Build the second branch of amplifier for the second choke and repeat steps 1-8.
10. In steps 1-9, the amplifier is driving a  $1K\Omega$  resistor (simulating a TTL Buffer).
11. In steps 1-9, check the output signal of the amplifiers when no input signal is injected to the input terminals (tied to ground). The output signal should have a minimum noise level. Preferably for TTL logic safety, the noise level should be less than 0.3V.

The RMS value of the noise level is 10.7mV which is less than 0.3V.

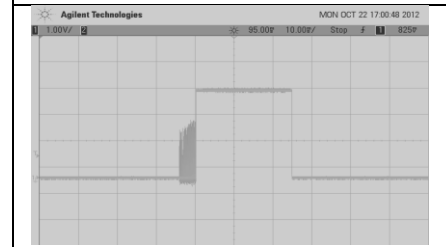
12. If any of the specs are not met, change the Amplifier Chip is a lower power noise version. Additionally if the gain values can be redefined to meet the specifications.

## Level Restorer / Slicer / Logic Converter

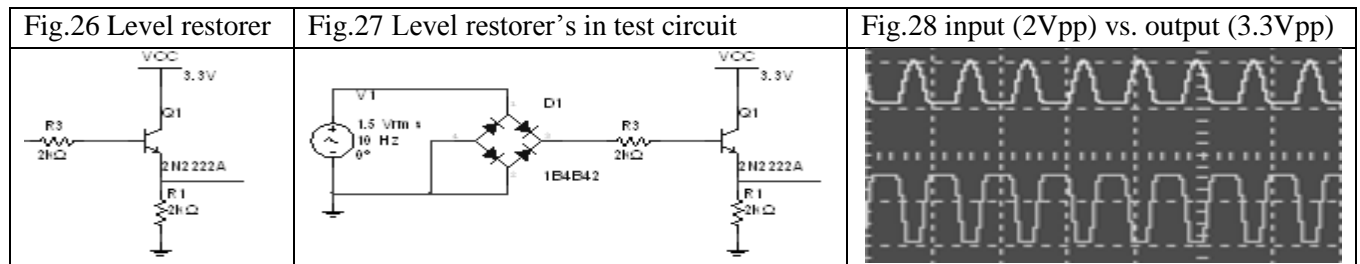
Equation / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / [Quantitative Results and Graphs](#) / [Interference with rest of the project](#)

As discussed in the amplifier/clipper section, the operational amplifier biased at 3.3VDC (power rail) uses approximately 1.3V for internal operation and limits the output voltage swing to maximum of 2Vpp (positive voltage with reference to ground). Unfortunately, this voltage level does not fall within the valid region (0VDC-0.2VDC for LOW, and 3VDC-3.3VDC for HIGH) of the TTL chip biased at 3.3VDC. TTLs' narrow noise margins coupled with this issues causes glitches and in some cases oscillatory behavior on the output of TTL logics. To the right is wave presenting such a behavior.

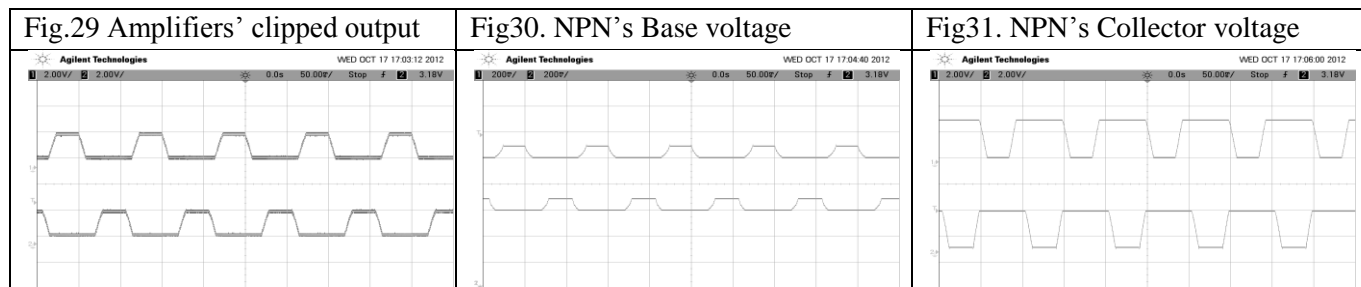
Fig25. NOT-NAND (OR) TTL output, driven by 2Vpp signal



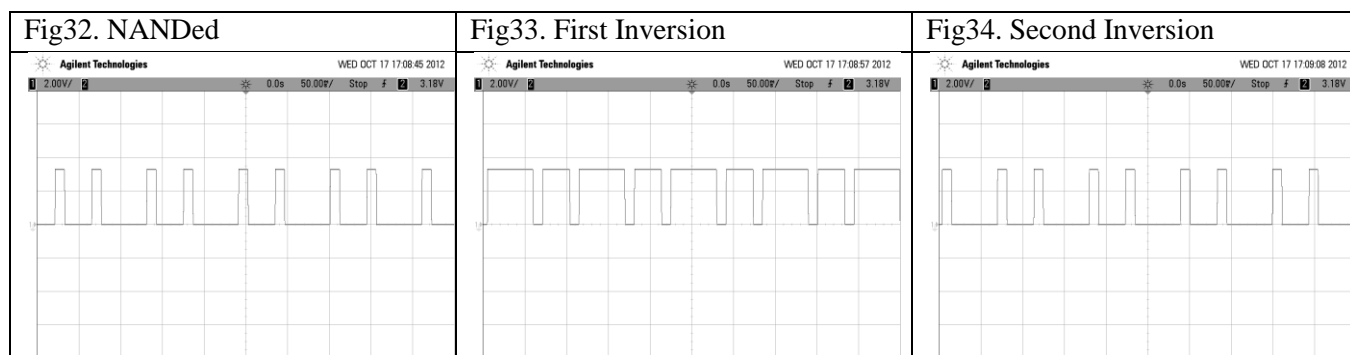
To overcome this issue a special configuration (no bias circuit) of NPN Common Collector-Amplifier is employed as a voltage restorer (and signal inversion). This circuit pumps up and inverts the voltage swing to a visible range (0VDC-0.2VDC for LOW, and 3VDC-3.3VDC for HIGH) by the TTL logics (biased at 3.3VDC). Below is a schematic of voltage restorer being derived by a costume signal simulation. As the simulation shows, the signal is leveled and inverted properly. The slight cut-off at the bottom of the output signal (blue) is due to high voltage swing of the input signal (yellow). By lowering the input signal voltage (to approximately 1.9Vpp) or increasing the input resistance (to  $2.2K\Omega$ ) or increasing the pull down resistance (to  $3K\Omega$ ) cut-off region will go away (as tested and verified by the circuit measurements).



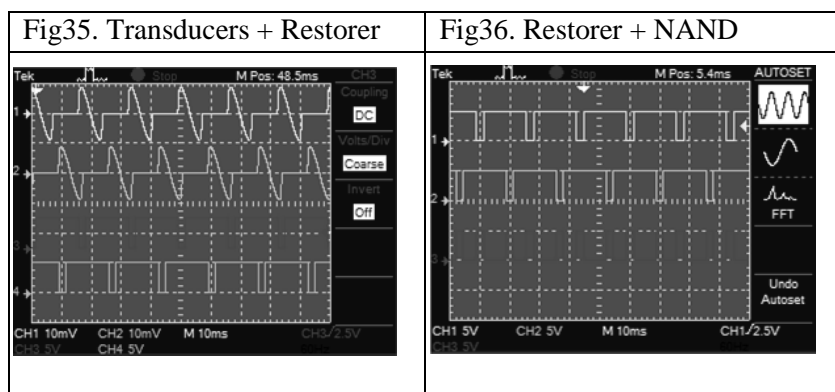
At this stage the voltage restorers are cascaded on the amplifiers'/clippers' output, and tested for rated performance. As shown in the wave captures below, the amplified /clipped signals are properly restored and inverted from 2Vpp to 3.3Vpp. The wave captures show both channels, where one is delayed from the other to simulate the time span between transducers gating interval.



Then, this signal is fed to the TTL buffer (slicers) to trim-off the slow ramp (rise and fall) edges. At this stage, the inverted signals are Nanded to implement the OR function. As the level restores were used to invert the signals, there are leftover TTL gates on the Quad NAND chip, from which two can be cascaded to implement a buffer. This buffer increases the overall gain of the slicer, which adds to the circuit noise margin immunity (one shot, two birds).



As shown above, the sinusoidal transducers' gating signals are converted to perfectly readable digital signals. This signal is fed directly to the TX microprocessor for timing the IR sensors' wave capture. These experimental results are also consistent with the simulation results.



Note: In the wave captures of output signal, the smaller interval between the gating signal pulses represents the duration for which the microprocessor captures the heat signature data.

## Verification

1. Use the Quad NAND gate chip (74LS00) to implement the combinational logic.  
**Modified to fit new circuit, Passed.**
2. Use a 1M $\Omega$  resistor on the output of the combinational logic to simulate the microprocessor digital port impedance.  
**Passed.**
3. Use a DMM on the power line connecting to the chip to monitor the drain current. It should not be more than 2mA. If it is not the case, replace the TTL chip with a lower power version.  
**Passed.**
4. Connect one of the inputs of the combinational logic (OR) to a function generator set at 10Hz square wave (0-5V swing) and the other input to ground. Trace the output signal to ensure full swing and the pulse shape (perfect square wave) matches the input.  
**Modified to 3.3V to fit new circuit, Passed.**
5. Repeat step 4, this time with 10Hz square wave at a lower swing (0-4V). This test should be performed because the signals coming from the amplifier stage were calculated to have similar values for the rated frequency.  
**Modified to 3.3V to fit new circuit, Passed.**
6. If step 5 is not passed then reconfigure the amplifier gain (to a higher value), so the amplifier output swing is within the acceptable TTL range.  
**Modified to fit level restorer, Passed.**
7. Increase the frequency to 20Hz and repeat steps 4-6.  
**Passed.**
8. In all steps, trace the noise level (tolerance) of the input signal and ensure it is not more than 0.3V<sub>pp</sub>  
**Passed.**

## IR Sensor - Signal Processing

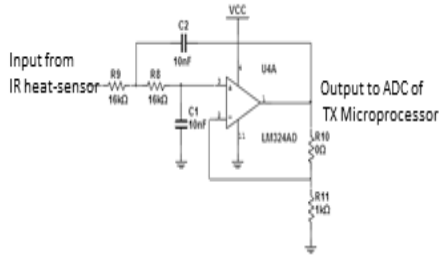
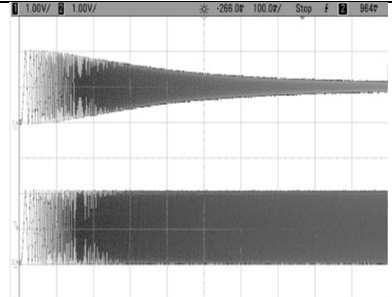
### Fuses + Active Filter

[Equation](#) / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / [Quantitative Results and Graphs](#) / [Interference with rest of the project](#)

An active filter was used to filter and amplify the signal coming from the IR heat sensors. The gain of amplifier should be adjustable in order to provide an appropriate range to the ADC of the microprocessor which ranges from 0 to 3.3VDC. The filter is a 2<sup>nd</sup> type low pass configuration with cutoff frequency at 1 KHz, providing 40dB of attenuation. The active part was implemented utilizing a non-inverting Op-Amp borrowed from the LM324 (used in the transducers branch). The cutoff frequency and gain calculations are provided in the table below. The circuit was built and tested to evaluate for the rated specifications. The two crucial test vectors are:


- 1Hz-3KHz sine sweep to check for VTC.
- 100Hz sine, which represents the most common bandwidth of heat signature signals, to check for VTC and phase shift [1].

As the measurement results indicate, the VTC response matches with the designed parameters. It has a flat response over the critical region of 0-500Hz and cut-off frequency at 1KHz. The 3dB bandwidth is 800Hz and falls to a 40dB at 3KHz.

Fig.37 Circuit Diagram	Fig38. Equations	Fig39. Frequency Sweep 1Hz-3KHz
	$f_{cut-off} = \frac{1}{2\pi \sqrt{\frac{1}{R8 \times R9 \times C1 \times C2}}}$ $f_{cut-off} = 1 \text{ KHz}$ $Gain = 1 + \frac{R_{10}}{R_{11}} = 1$	

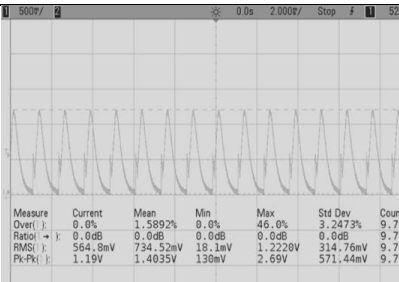
As the measurement results indicate, the phase shift is nearly zero, when the 100Hz sine (common bandwidth of heat signature signal) wave is injected. So there would be no delay to desynchronize the timing between the transducers' gating signal and heat signatures. The circuit was also tested for noise handling by injecting a 800Hz square wave (odd harmonics at 800Hz, 2.4KHz, 4KHz, ...). The active filter attenuates all harmonics above 1KHz and the fundamental harmonic was passed to the output (800Hz sine).

### Fig40. VTC and Phase for 100Hz sine



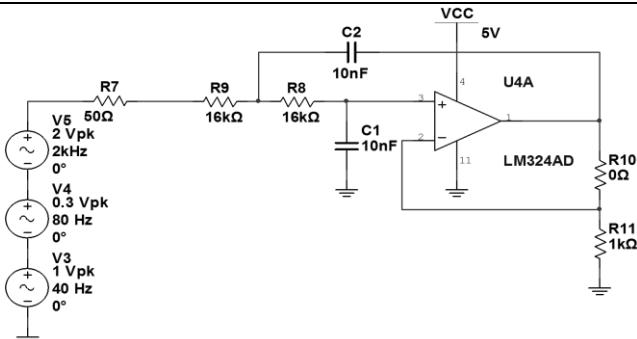
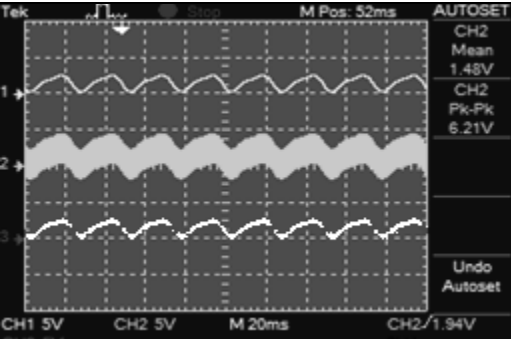
1.00V/ 1.00V/ 0.8s 10.00P/ Auto f 963P

### Fig41. Noise handling, 800 square wave



Measure	Current	Mean	Min	Max	Std Dev	Count
Overl	0.0%	1.5892%	0.0%	46.0%	3.2473%	9.701k
Ratio	0.0dB	0.0dB	0.0dB	0.0dB	0.0dB	9.702k
RMS	564.8mV	734.52mV	18.1mV	1.2220V	314.76mV	9.701k
Pk-Pk	1.19V	1.4035V	130mV	2.69V	571.44mV	9.700k

The experimental results are consistent within a close bound to the simulations, where a 2KHz sine (noise) is mixed with the heat signature. The 2KHz sine is filtered out, and heat signature is recovered.

Fig42. Simulation test circuit	Fig43. Simulations wave capture																																			
	 <table><tr><th>Measure</th><th>Current</th><th>Mean</th><th>Min</th><th>Max</th><th>Std Dev</th><th>Count</th></tr><tr><td>Overl</td><td>0.0%</td><td>1.5892%</td><td>0.0%</td><td>46.0%</td><td>3.2473%</td><td>9.701k</td></tr><tr><td>Ratio</td><td>0.0dB</td><td>0.0dB</td><td>0.0dB</td><td>0.0dB</td><td>0.0dB</td><td>9.702k</td></tr><tr><td>RMS</td><td>564.8mV</td><td>734.52mV</td><td>18.1mV</td><td>1.2220V</td><td>314.76mV</td><td>9.701k</td></tr><tr><td>Pk-Pk</td><td>1.19V</td><td>1.4035V</td><td>130mV</td><td>2.69V</td><td>571.44mV</td><td>9.700k</td></tr></table>	Measure	Current	Mean	Min	Max	Std Dev	Count	Overl	0.0%	1.5892%	0.0%	46.0%	3.2473%	9.701k	Ratio	0.0dB	0.0dB	0.0dB	0.0dB	0.0dB	9.702k	RMS	564.8mV	734.52mV	18.1mV	1.2220V	314.76mV	9.701k	Pk-Pk	1.19V	1.4035V	130mV	2.69V	571.44mV	9.700k
Measure	Current	Mean	Min	Max	Std Dev	Count																														
Overl	0.0%	1.5892%	0.0%	46.0%	3.2473%	9.701k																														
Ratio	0.0dB	0.0dB	0.0dB	0.0dB	0.0dB	9.702k																														
RMS	564.8mV	734.52mV	18.1mV	1.2220V	314.76mV	9.701k																														
Pk-Pk	1.19V	1.4035V	130mV	2.69V	571.44mV	9.700k																														



## Verification

1. Build the active filter as shown in figure. Connect a  $1\text{M}\Omega$  resistor as the load to simulate the microprocessor ADC.  
**The circuit was built on breadboard, Passed.**
2. Use a DMM to monitor the current drain by the amplifier chip. It should be less than  $30\text{mA}$ , while all three (two from transducers + active filter amplifier) are loaded and running.  
**The maximum current measured using the DMM is  $8.1\text{mA}$ , Passed.**
3. Use the function generator to generate a  $100\text{Hz}$  sinusoid with  $1\text{Vpp}$  and  $0.5\text{VDC}$  bias to simulate the heat signature signal as read by IR sensor. Make sure the function generator is set to  $50\Omega$  mode to simulate the IR sensor terminal impedance. Set the amplifier gain to 1 by choosing the feedback resistance to be  $0\Omega$ . This resistance (consequently the gain) is controlled by a potentiometer (log tape  $0\text{-}500\text{K}\Omega$ ).  
**In addition, sweep function was employed, Passed.**
4. Read the signal level on the input of active filter. This signal should not be less than  $0.9\text{V}$ . Otherwise, the feedback loop or the filter's resistive network has a bad connection. Double check the active filter to ensure the right components is used.  
**Passed.**
5. If step 4 is passed, use the function generator's sweep mode to sweep the signal injected into the active filter from  $10\text{Hz}$  to  $3\text{KHz}$ . Trace the output using an oscilloscope to ensure flat response over the critical region  $10\text{Hz}$  to  $500\text{Hz}$  and  $40\text{dB}$  attenuation over  $1\text{KHz}$  cutoff point. If the signal is not attenuated by the rated value at  $1.2\text{KHz}$  then shift the cutoff point toward a lower frequency (not lower than  $800\text{Hz}$ ). The  $800\text{Hz}$  was chosen to satisfy the bandwidth of heat-signature signal.  
**There is a flat response over the frequency range from  $0\text{-}500\text{Hz}$ , Passed.**
6. Adjust the function generator to generate a  $100\text{Hz}$  sinusoid of  $0.5\text{Vpp}$  (a common heat-signature peak value) with  $0.25\text{VDC}$  bias. Change the gain (increase) using the potentiometer and trace the output signal of the active filter to bring up the peak voltage level value to approximately  $2\text{V}$ . The  $3.3\text{V}$  ADC limit is not used to provide headroom for the heat-signature to prevent clipping in case the input signal exceeds  $2\text{Vpp}$  limit.  
**Passed.**
7. Inject the signal described in step 6 + a square wave with fundamental frequency at  $1.5\text{KHz}$ , and trace the output signal to ensure the additive noise (in this case square wave) is well attenuated.  
**The input signal is  $800\text{Hz}$  square wave  $1\text{Vpp}$  with  $0.5\text{VDC}$  offset. The output signal observed presents an almost sinusoidal signal because the higher harmonics of the square wave are filtered out which indicates that additive noise is attenuated, Passed.**
8. If step 7 is not passed, repeat steps 3-7 to adjust the gain and cutoff frequency to a reasonable value so that the output signal has a proper signal level (not higher than  $4.5\text{V}$ ) and noise level.  
**Specs are met, Passed.**
9. Note: For reconfiguring the cutoff frequency it is easier to change the active filter resistors' values (utilizing a pot eases the tuning process).

## TX Microprocessor

### Bootloading Atmega328 – Fuse Configuration

[Equation](#) / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / [Quantitative Results and Graphs](#) / [Interference with rest of the project](#)

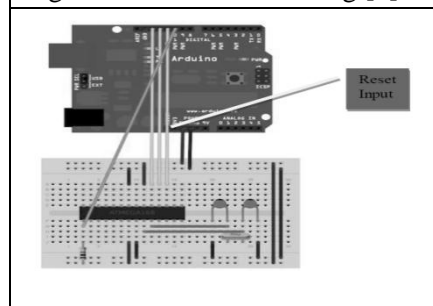
#### Requirements [2]:

- ☐ Arduino Duemilanove board with DIP ATMEGA328/328P + Arduino IDE (latest version)
- ☐ A computer with USB port and Microsoft OS
- ☐ ATMEGA328/328P microcontroller
- ☐ Small Breadboard + Pieces of jumper wires
- ☐ One 110 $\Omega$  to 128 $\Omega$  resistor + One 1K $\Omega$  resistor
- ☐ Three LEDs with 330 $\Omega$  to 1K $\Omega$  current limiting resistors
- ☐ Two 15PF to 22PF capacitors + One 1MHz to 20 MHz Crystal

#### Connect/Install Arduino Board [2]:

1. Download Arduino IDE from Arduino official website: <http://arduino.cc/>
2. Connect Arduino board to computer using USB cable.
3. Wait for windows to install the board
4. Start Arduino IDE, and select the board and serial port from the Tools menu that corresponds to the board.
5. Upload the ArduinoISP sketch onto Arduino board.
6. Wire up Arduino board and microcontroller as shown in the diagram, and connect the 110 $\Omega$  to 128 $\Omega$  resistor between Arduino reset input and 5V.
7. In the Arduino IDE folder, navigate to: hardware\tools\avr\etc
8. Open **avrdude.conf** file, and find ATMEGA328/328P
9. Identify the microcontrollers' signature and change it to:
  - ☐ ATMEGA328: 0x1e 0x95 0x14 [3]
  - ☐ ATMEGA328P: 0x1e 0x95 0x0F [3]
10. In the Arduino IDE folder, navigate to: \hardware\arduino\bootloaders\atmega
11. Copy all contents this folder to: \hardware\arduino\bootloaders
12. Restart Arduino IDE, and run the bootloader from Tools menu with Arduino as ISP.
13. Bootloading may take up to a minute. After you successfully burned the bootloader, change signature back to 0x1e 0x95 0x0F.

Fig44. Arduino Bootloading [2]



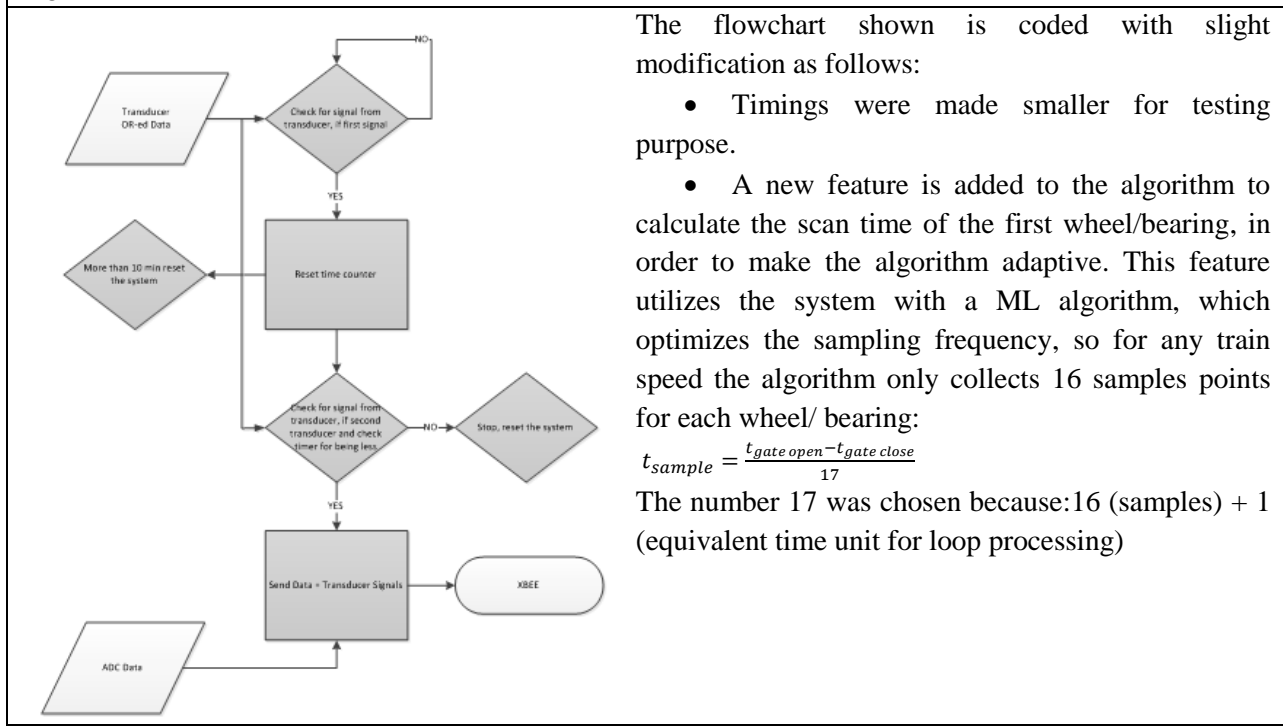
### Programming

[Equation](#) / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / [Quantitative Results and Graphs](#) / [Interference with rest of the project](#)

The Atmega328 was bootloaded and programmed to implement the proposed flowchart. Slight modifications are applied to expand, optimize and add new features to this the systems. The following algorithm was tested individually, and in-system. For both cases the expected result was fetched and analyzed by the algorithm. In addition, an external hard reset circuitry is implemented to reset the

processor upon successful completion of data acquisition cycle. Look at appendix D for TX processor program.

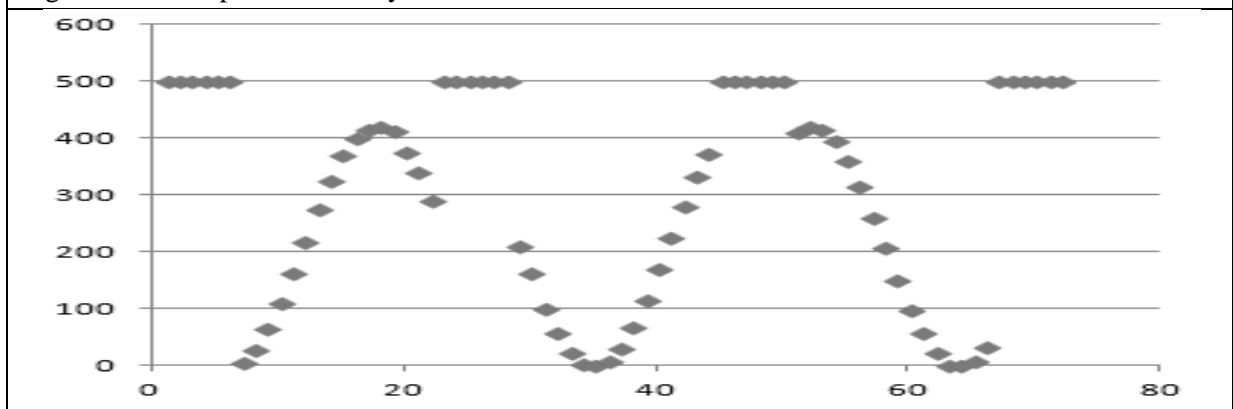
Fig45. Flowchart



The TX microprocessor was inserted in the TX system (in cascade with Amplifiers, XBEE, ...) and test vectors were injected to the system to evaluate the systems functionality. Below is the data stream captured/analyzed by the algorithm (sent over the wireless channel and captured on the serial port of RX XBEE). The heat signature is a 5Hz triangle wave, and the transducer signals are set to simulate a train going 35mph (10Hz) as calculated. As shown, there are only 16 samples in each period (wow, it works).

Note: Test vectors were simulated for trains going at different speeds (equivalent to transducers frequency of 1Hz to 20Hz) and all processed data contained only 16 sample points per wheel/bearing.

Fig46. TX microprocessor analysis



Note: The sample points at “500” represent the transducer signals at 30% duty cycle.

## Verification

1. Ensure the connections from TX microprocessor to TX XBEE are correct. TX of the microprocessor goes to the RX of the XBEE and the RX of the microprocessor goes to the TX of XBEE.  
**Passed.**
2. There are several triggering algorithms running in parallel which are controlled by a watchdog timer. The watchdog timer should be reset at the beginning of each iteration. Failing to do so, will prevent the algorithm from being reset.  
**Passed.**
3. The main functioning trigger is the transducer detection algorithm. If in case of any malfunction, this algorithm should be tested individually (in the first step of troubleshooting procedure) to evaluate its proper timing and correlation to incoming transducer signals. The most important body of this algorithm is the most frontal stage responsible to detect the first transducer followed by the second transducer within a predefined period (as defined in the flowchart and calculations). This stage tells the TX processor to wake up and start fetching next incoming transducer pulses along with the heat-signature. Use a function generator to feed the transducer's digital port reserved on the TX microcontroller. Change the duty cycle until the low time of the square wave is less than the specified time span mentioned in the calculations and use the serial terminal to send checkpoints if the algorithm detected and triggers the wake-up signal. Decreases the duty cycle until the low time is larger than the time span specified in the calculations. Use checkpoints to be sent to serial terminal to ensure the algorithm does not trigger the wake-up signal.  
**Passed.**
4. The next layer of troubleshooting the TX microprocessor is to feed the data acquisition's trigger manually to ensure the transducers and sampled heat-signatures are mixed and saved to an array properly. Feed the newly created array (transducers + heat-signatures) into the serial terminal (to a computer) and verify if the signals going into the microprocessor match with data obtained on the serial terminal.  
**Passed.**
5. Ensure the serial transmission rate is fast enough so the data traffic will not prevent data samples to get lost or delayed. This can be done by setting the serial baud rate to values higher than 9600.  
**Passed.**
6. If steps 1-5 are passed, connect XBEE to the TX processor and connect RX XBEE to the computer. Feed the TX chip with simulated signals and read the data transmitted to the computer.  
**Coded part of RX microprocessor algorithm to fetch the data.**

## Power Supply TX/RX

In the beginning stages of the design, the power supply was selected to be regulated at 5VDC. However, some later modifications and additional modules required a 3.3VDC on-board power supply as well as the 5VDC regulator. TX/RX XBEE units, the SD Card module, combinational logic, amplifiers, filters and level restorers are powered by a 3.3VDC supply. The 5VDC power supply is designated to run the microprocessors, GLCD and small number of combinational logics. This voltage is regulated by the circuits shown below. These circuits are capable of delivering 3.3VDC @ maximum drive current of 500mA, 5VDC @ maximum drive current of 1A. The input voltage – driving the regulators - can be

chosen from a wide range of 6V to 32V, which is suitable for running the system by a 9V battery, and 24 batteries used in HotBox systems.

Fig.47 The 5VDC power supply

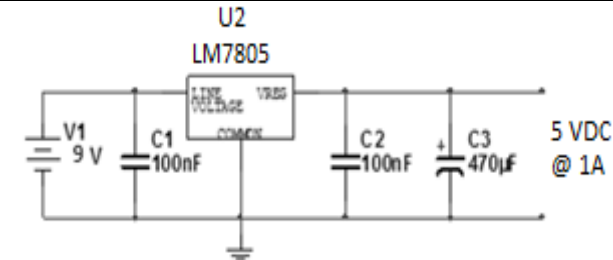
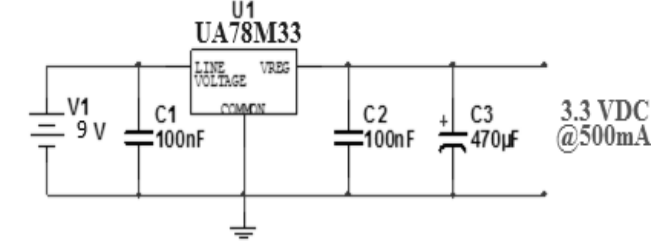


Fig.48 The 3.3VDC power supply



- TX unit at sleep mode  $\rightarrow 5\text{mA (Xbee)} + 10\text{mA (all three amplifiers)} + 2\text{mA (TTL chip)} + 4\text{mA (Level Restorer)} = 21\text{mA @ } 3.3\text{VDC} = 70\text{mW}$
- TX unit at sleep mode  $\rightarrow 5\text{mA (processor)} = 5\text{mA @ } 5\text{VDC} = 25\text{mW}$
- TX unit at active mode  $\rightarrow 35\text{mA (Xbee)} + 18\text{mA (all three amplifier)} + 2\text{mA (TTL chip)} + 7\text{mA (Level Restorer)} = 62\text{mA @ } 3.3\text{VDC} = 204\text{mW}$
- TX unit at active mode  $\rightarrow 5\text{mA (processor)} = 5\text{mA @ } 5\text{VDC} = 25\text{mW}$
- RX unit at active mode  $\rightarrow 45\text{mA (Xbee)} + 8\text{mA (SD Card)} = 53\text{mA @ } 3.3\text{VDC} = 175\text{mW}$
- RX unit at active mode  $\rightarrow 5\text{mA (processor)} + 65\text{mA} = 70\text{mA @ } 5\text{VDC} = 350\text{mW}$

The power consumption when TX circuit is active (transmitting data) is  $204\text{mW @ } 3.3\text{VDC}$ . The required drive current of the TX unit in active mode is  $62\text{mA}$  which is below the maximum allowable drive current of the  $3.3\text{VDC}$  regulator at  $500\text{mA}$ . The  $3.3\text{VDC}$  regulator is the main power supplier of the TX unit, while the RX unit is mainly supplied by the  $5\text{VDC}$  regulator at  $350\text{mW}$ . The critical tests are ensuring that the variations in the output voltage are within the specified limits. These tests are presented below.

Fig.49 Unloaded TX 3.3VDC power supply

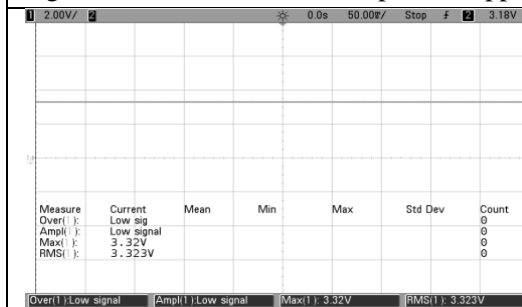
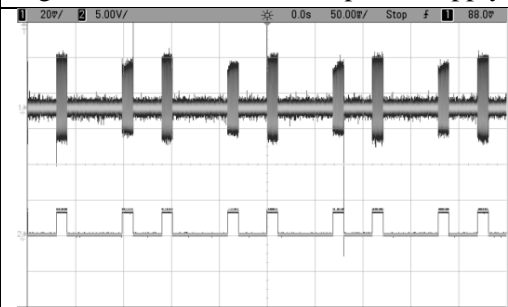
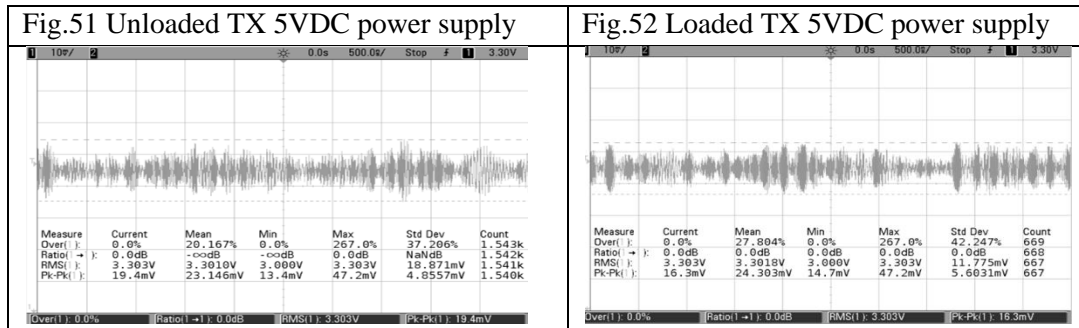


Fig.50 Loaded TX 3.3VDC power supply





As shown above, the largest voltage deviations peaks are at max 50mVpp biased at 3.23VDC, when the power supply is driving the TX circuit at full power. These peaks occur when the transducer amplifiers are activated, as the in phase periodic peaks match the transducer signal. The voltage variation on the 5VDC power supply is negligible due to small driving loads.

The tolerance of the power supply when driving the circuit is  $\frac{50mV}{3.32V} \times 100 = 1.5\%$ . This deviation is within the 2% allowable range of tolerance.

The RX unit has similar power distribution system, except the fact that 5VDC is the main power supplier of the circuit. The limitations on power line tolerations are softer for RX unit, because of the nature of components/blocks used in the structure of this unit. In other words, the RX unit does not have any analog signal processing unit that requires precision on the power line.

### Verification

1. Load both voltage regulators by a proper resistive load (10-12Ω) to ensure drive current of 400mA.  
**Specs are met.**
2. Use the function generator to couple the input power supply of the voltage regulators with the 60Hz sinusoid 0.5Vpp. Monitor the regulated signal on the output. If the voltage variation on the output of voltage regulator is more than 0.005Vpp, then change the bypass capacitors so that the voltage variation on the output is less than 0.005Vpp. This procedure is critical because the variation on the power supply will modulate the amplifier stage output signal.  
**Adding the 60Hz 0.5Vpp source to the input of the power supply does not alter the output voltage in any major way. The measured voltage variation is 24.303mVpp-23.146mVpp = 1.157mVpp which is well below 5mVpp.**
3. If step 2 is passed then load the voltage regulators by the actual circuit. Again, monitor the voltage variations on the regulator's output and the drained current to ensure the rated specs are met.  
**The measured tolerance is 1.5% when the power supply is driving the circuit.**
4. If the power supply tolerance is not met, change the bypass capacitors. If the drain current is not within the specs, change the voltage regulator to a higher power version.  
**Specs are met.**
5. If the current drain problem is not resolved, connect each stage of the circuit individually to the voltage regulator in order to find which stage is causing the problem (draining more than what is supposed to) to troubleshoot.

## TX XBEE

### Network ID Configuration

The TX XBEE is responsible for transmitting the serial data generated by the processor block from the TX Unit. The XBEE wireless module employs the Zigbee protocol [4] to communicate to the receiver module. The XBEE chips used in this project are Series 1 XBEE meaning that no coordinator is required to setup the wireless communication and both modules can act as transmitter/receiver. Arduino Uno was used as the serial adapter to configure the XBEE modules.

Fig.53 Wiring diagram for XBEE module configuration

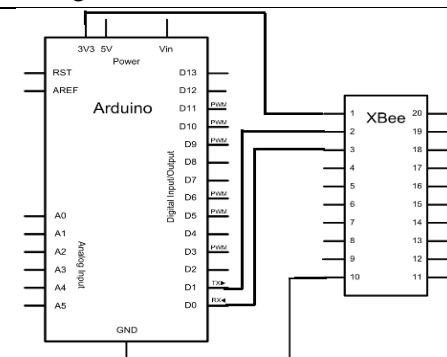


Fig.54 Identify XBEE using X-CTU Serial Terminal

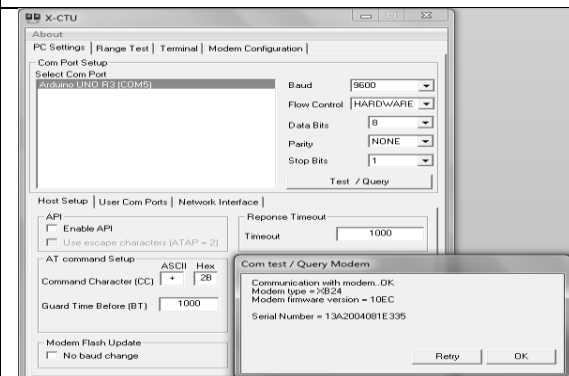
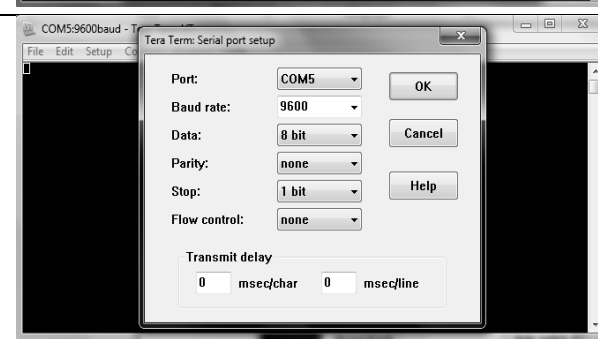
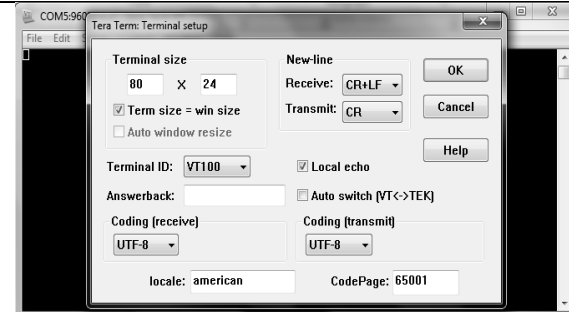
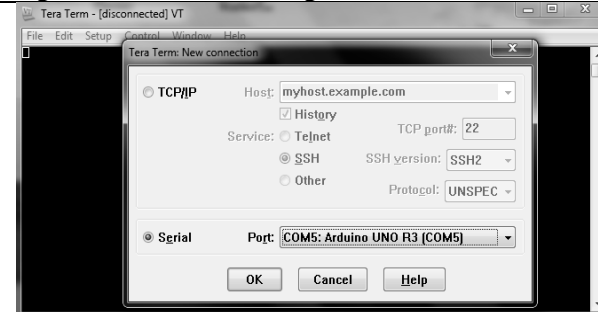


Fig.55 Network ID configuration menu





Program Arduino Uno with the program Empty.ino. This enables the Arduino Uno board to act as USB to Serial converter to configure the XBEE. After uploading the program, with the Arduino powered off, connect the XBEE as shown in the figure.

Identify the XBEE module [4]:

1. Plug the Arduino USB cable in the computer
2. Open X-CTU terminal
3. Click on Test/Query button
4. Ensure that the Serial Number in the window that pops up agrees with the Serial Number on the TX XBEE (ATDH: 0013A200 and ATDL: 4081E32A). The Serial Number is on the back of the XBEE module.

Configure XBEE [4]:

1. Plug the Arduino USB cable in the computer
2. Open the Tera Term terminal
3. Select the Serial port and click OK.
4. Navigate to Setup->Terminal, insert the following parameters and click OK.
5. Navigate to Setup tab ->Serial port, insert the following parameters and click OK.
6. Type “+++” which makes the XBEE go into command mode
7. Wait for OK. This ensures that the XBEE is responding
8. The PAN ID (Personal Area Network ID) used in this project is 2012. In the terminal type “atid2012” and press enter. Now type “+++” and wait for OK. Type again “atid” and press enter and the response should be “2012” which indicates the configured PAN ID is 2012.
9. ATDH is “Destination Address High”. This is the same for both TX and RX XBEEs. In terminal type “atdh13a200” and press enter. Type “+++” and wait for OK response. Now type “atdh” and press enter. The response should be “13a200”.
10. ATDL is “Destination Address Low”. This will be different for both XBEEs. The TX will be configured with the ATDL of RX and the RX will be configured with the ATDL of TX. This configuration enables the RX to receive information from TX and viceversa. In terminal type “atdl4081e335” and press enter. Type “+++” and wait for OK. Now type “atdl” and the response should be “4081e335”.
11. To ensure the configuration is not lost when the XBEE is powered off, type “atwr” and press enter. This command writes the configuration to firmware.

Verification
<ol style="list-style-type: none"><li>1. Connect RS232 serial cable to XBEE units on two separate computers and use the terminal port to send and receive specific test vectors. <b>One Arduino was programmed to send character “D” through the serial port connected to TX XBEE. Another Arduino was programmed to receive data from RX XBEE over the serial port. The data received by the RX XBEE was monitored and transmission contained no errors.</b></li><li>2. If the transmitted data matches the received data, then program two microcontrollers to send and receive a known test vector of ASCII characters over a wired channel. Check for accuracy of received data (with 0 bit error rate).</li></ol>



One Arduino was programmed to send transducer and heat-signature data signals to TX XBEE. Another Arduino was programmed to read the serial port data coming from RX XBEE. The data received was monitored on the serial monitor. Transducer signals along with the heat-signature data were received with absolutely no errors.

3. Connect the XBEE units to the programmed microcontroller from step 2 and check for accuracy of data received (ideally 0 bit error rate).

Test was performed in step 2 above.

## RX XBEE

### Network ID Configuration

The RX XBEE is configured the same way as TX XBEE with the exception of the following step:

- In the Configure XBEE section, in the Tera-Term Terminal in step 10, the command is “ATDL4081E32A”. All the other steps are followed in the same order [4].

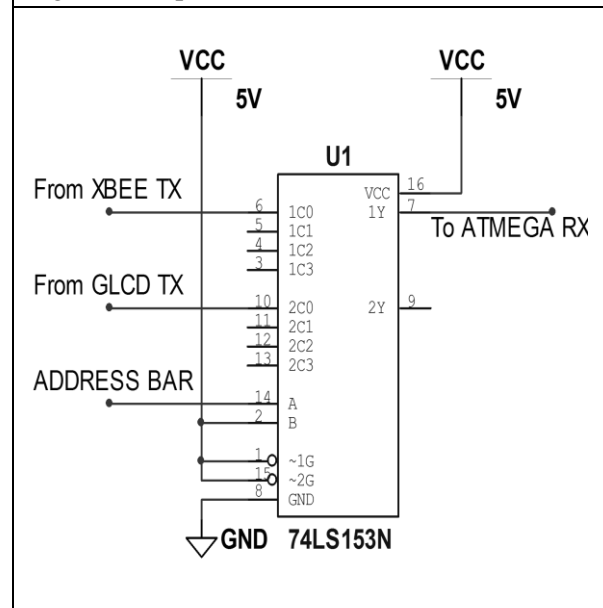
## Multiplexer

The ATMEL microprocessor - ATMEGA328 - provides us with only one set of embedded hardware serial terminal. Having only one set of serial terminal available along with multiple serial devices trying to talk to the processor becomes problematic. There are multiple solutions to this problem such as software serial [5] or providing an external hub. With both choices on the table, we decided to use an external hub. Having an external hub is beneficial in many aspects, which are:

- Software serial uses a pair of pins for each additional terminal, which is not a good design strategy to use of digital pins for a single protocol.
- Software serial driver is not as accurate as hardware serial, as we tested and analyzed the results.
- Having a hub provides the option of multiplexing/packaging data, so the processor can talk to multiple devices at the same time using only two data pins + address bus (This is simulating an I2C bus using serial terminal). The number of serial terminals can be extended to more number of pins that the processor has.
- It can run at high data rates accurately than the software serial, and more importantly it consumes less paging and CPU clock to run multiple levels of code to simulate a software serial than basically reading off from the hardware terminals.
- Give the designer the opportunity to add more devices to later revisions with minimal hardware modification.

The hub was implemented using 74LS153 which is a TTL Dual 4to1 multiplexer. In the case of our project, we needed to connect GLCD and XBEE wireless unit through the serial terminal to the microprocessor. The XBEE has a TX connecting to RX of the processor, and GLCD has both RX and TX lines going to TX and RX of the processor. We figured out that the GLCD RX can be connected to the processor's TX at all time with no problem. So, the multiplexer is used to switch the data path from the XBEE to GLCD whenever needed. To the right is the schematic showing the multiplexer connections. The GLCD is one line when the address bar is pulled HIGH, and the XBEE connection is established when the address bar is pulled low. Note that the GLCD and XBEE do not need to be initialized more than once.

Fig.56 Multiplexer connections



### Verification

1. Test that the correct input is selected. Connect two LEDs to the output of the Dual 4 to 1 Mux. For each pair of addressing bits (00, 01, 11, 10) input a HIGH Signal followed by a LOW Signal to the corresponding input and check that the LEDS light up accordingly.  
**Passed.**
2. Test data integrity. Refer to Figure above for connections. Input a test vector using an Arduino Uno Serial Port to the input "From XBEE TX". Select address 00. Observe the output "To ATMEGA RX" on a different Arduino Uno using the Serial Terminal. Ensure that the Serial Terminal data matches 100% with the test vector values.  
**Passed**
3. Repeat step 2, using the same procedure, but now test the remaining inputs that correspond to the address bits 01, 11, 10.  
**Passed**
4. Connect the RX XBEE and GLCD and observe that after data transmission process has ended the GLCD algorithm goes through the right sequence. This shows that the MUX successfully switches the data line between RX XBEE and GLCD module.  
**Passed.**

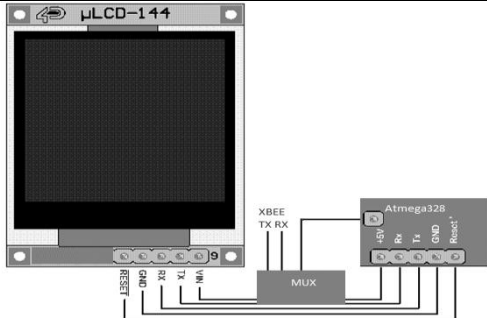
## GLCD

Equation / [Schematic with Components Diagram](#) / [Simulation](#) / [Circuit](#) / [Measurements](#) / Quantitative Results and Graphs / [Interference with rest of the project](#)

The system utilizes a Graphical Color LCD (128x128) as the main user interface module. This LCD runs on SGC firmware [5 6], which enables microcontroller to control the graphical content displayed on the screen through serial communication. The optimal serial baud rate, as suggested by the manufacturer [5 7], is at 115200 (for fast page refreshing). However it can be run at lower rate such as 9600 for older processors. The LCD driver is taken from the open source project and slightly modified (to work with pass by reference) to fit this application.

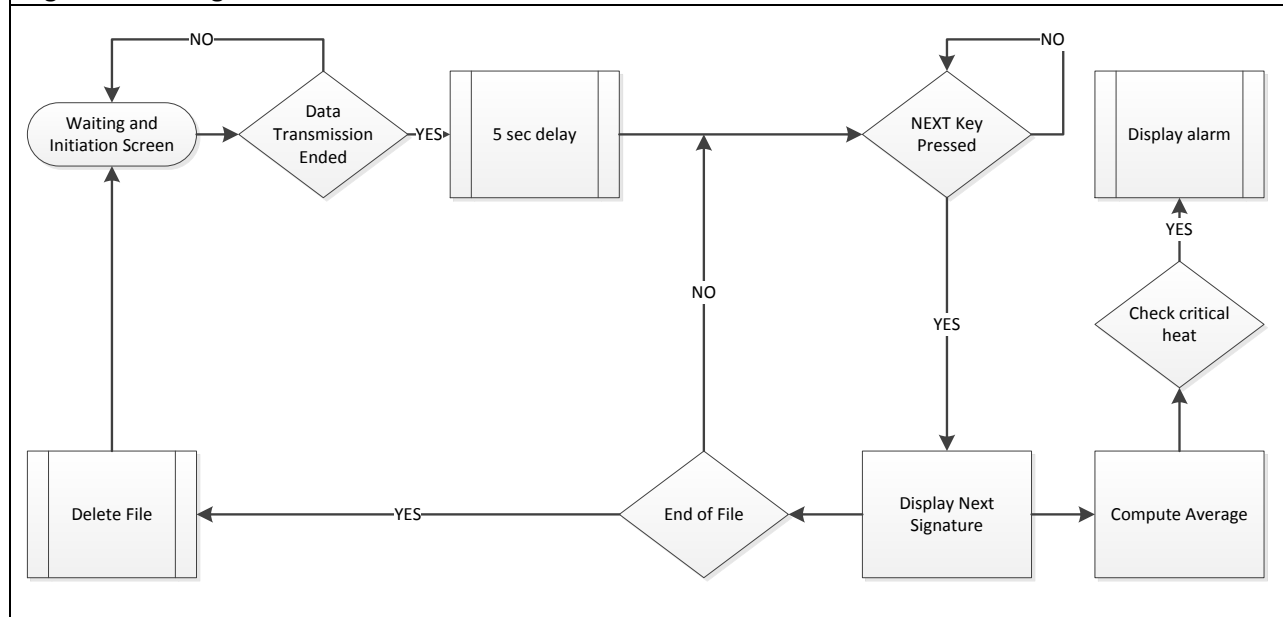
Note: Atmega328 has only one serial port, which should be shared between XBEE and the LCD module. So, a multiplexer unit should be used to connect the proper module to microprocessor as needed. The default is set for XBEE connection. The lines switch over, when the data acquisition process is over and the algorithm provides the user access to SD Card. In addition, the microprocessor should be reset after the connection is established (before any data transmission).

Below is the block diagram, showing the LCD's connections to microprocessor through the data multiplexer. Below are the programs used to test the LCD's functionality.

GLCD test program + Waveform Simulation	Fig57. GLCD144G2 SGC in system
<pre> #include &lt;displayshield4d.h&gt; // necessary library DisplayShield4d lcd; // create an instance of the LCD void setup() {   Serial.begin(115200); // LCD speed is very high   lcd.Init(); // wake up LCD   delay(20); } int data[26]={387,292,199,105,12,76,169,263,356,449,544, 639,731,804,710,616,523,429,336,239,147,54,33,126,220,314}; void loop() {   lcd.Clear(); // clear LCD   delay(20);   lcd.setfontmode(OLED_FONT_TRANSPARENT);   delay(20);   for(int i=0;i&lt;26;i++)   {     lcd.putpixel(((i*5)-1),(128-(data[i]/12)),lcd.RGB(255,0,0));     lcd.putpixel(i*5,(128-(data[i]/12)),lcd.RGB(0,255,0));     lcd.putpixel(((i*5)+1),(128-(data[i]/12)),lcd.RGB(0,0,255));     lcd.putpixel(i*5,(129-(data[i]/12)),lcd.RGB(0,255,255));     lcd.putpixel(i*5,(127-(data[i]/12)),lcd.RGB(255,255,0));   }   delay(2000); } </pre>	 <p>LCD with SGC firmware installed on its graphic processor. Reinstall the SGC firmware in case of any bad segment error. The highlighted code to the right is inserted in the SD Card algorithm to communicate with the LCD. The data values passed from the SD Card are scaled (mapped to 128x128), so they fit within the scope of the screen.</p>

The following state diagram presents GCLD state transition process. This algorithm is triggered and directed by the main structure of RX microprocessor's algorithm.

Fig. 58. GLCD Algorithm



The LCD was individually tested, and passed all requirements. It was also tested in system, while being driven by a portion of RX microprocessor algorithm responsible for communicating SD Card - using <SD.h> library [8] - to the LCD, and simulated waveforms (sent by TX) were accurately read and displayed on the screen (Demoed).

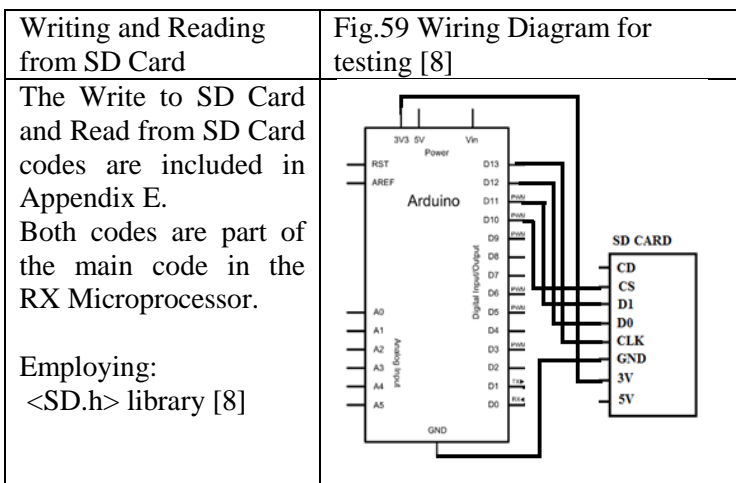
## Verification

1. The GLCD driver is only compatible with Arduino Compiler 002X.  
**Passed.**
2. Make sure the TX of the microprocessor is connected to RX of GLCD, and the RX of microprocessor is connected to the TX of GLCD.  
**Passed.**
3. The datasheet and GLCD drivers can be downloaded from the manufacturer website (included in the references section).  
**Passed.**
4. Make sure when the initialization process is done in the microprocessor the baud rate is set to 115200.  
**Passed.**
5. Make sure the transparent mode is activated during initialization otherwise the GLCD will remain black all the time.  
**Passed.**
6. Make sure the RESET of GLCD is connected only to the RESET of microprocessor and there are no other external analogs interfering with the RESET pin.  
**Passed.**
7. The alphabetical print command is passed by reference type and passing strings by value will not display anything on the screen.  
**Passed.**

8. The algorithm that fetches the data from SD card and responsible for ASCII to integer conversion is embedded within the GLCD algorithm. Any type of conversion (down sampling) should be performed with respect to GLCD screen size with minimum loss of data.  
**Implemented in SD Card algorithm, Passed.**
9. The most number of pixels shown on the horizontal scale is 128 so the data points for each heat-signature which is usually in the range of roughly 300-500 should be down sampled by a factor of 2 to 4. The algorithm should calculate the exact down conversion rate (for example 2.34 for the case of 300) so the down sampling conversion is optimized for minimum data loss.  
**This algorithm was implemented on the TX microprocessor for 26 points. Instead these algorithms copies/overlays each waveform four times with different colors to provide some volume.**
10. After all the train wheels are scanned (detected by the microprocessor when a delay of 3s is seen between the transducer pulses), the user gets access to read the SD card and activate the GLCD algorithm. If GLCD and interference keys are not locked during that time, the writing process gets interrupted and data samples will be lost. Therefore, check proper triggering of GLCD algorithm controlled by the main running algorithm on RX microprocessor.  
**To be implemented in RX algorithm.**
11. If the data on the screen has missing components that could be caused by static arrays which should be avoided and instead utilize heap memory.  
**Passed.**
12. The algorithm structure should allow a minimum of 5ms for each write instruction as well as clear and refreshing signals. So displaying a full heat signature can take up to 0.6 seconds.  
**Checked using millis() function, Passed.**
13. Reset the GLCD and microprocessor if they do not hand shake and random pixels are displayed on the screen.  
**Tested.**

## SD Card

The memory unit of RX module consists of a SD card (secure digital flash memory). It is powered from a 3.3V supply. The SD card is responsible for storing the data sent by the RX Microprocessor via SPI [8] (Serial Peripheral Interface) bus. The data will be stored in a text file. Each sample from the wheel signature will be separated by a comma and the data between every wheel will be delimited by the # character. This format makes it easier for the RX algorithm to read the data for every wheel and display it on the GLCD.



### Verification

1. Connect the SD card to an Arduino microcontroller as shown in Fig. 7. Upload the 'card info' code and ensure the SD card is in the FAT16 or FAT32 file format. Also ensure that the SD card memory does not exceed 8GB.  
**The SD card shows the correct file format:FAT32 and the memory is 8GB.**
2. Using the same configuration, upload the 'write to file' code and check that it takes 2ms at 115200 baud rate using the serial monitor. Repeat this process 6 times to ensure full functionality.  
**The text file shows that it takes 2ms to write a byte on the SD card. The test was performed 6 times.**
3. Using the same code as in step 2, write a predefined array of 20 digits to an empty text file. Power off the microcontroller. Take the SD card out and check with a card reader that the 20 elements of the array were written only once. Repeat this step 6 times to ensure full functionality.  
**The test was performed 6 times. Each time the data was written only once and the array elements were separated by comma and different arrays were separated by # character.**
4. To check the reading functionality, upload the 'read from file' code and use the serial monitor to read serially all the data on the file. Repeat this process 6 times to ensure full functionality.  
**The test was performed 6 times. Each time, all the array elements included in the text file were read successfully and the reading algorithm clearly distinguished different arrays.**

## Keypad

This block consists of a keypad which will be used for going through the data collected for the train wheels/bearing. This is connected to the Main Processor through the GUI processor. A pushbutton, NEXT, is used to access the next heat signature data. A RESET pushbutton is also implemented to reset the processor and GLCD unit if needed. The RESET can be used to refresh the program and delete the contents of SD card.

### Verification

1. Connect the pushbutton in series with a 1KOhm resistor and an LED. Power up the circuit by 5V power supply. Press the pushbutton and observe the LED light up.  
**Passed.**
2. Test the debouncer algorithm. Connect the pushbutton to digital pin 4 on Arduino Uno. Load the program Debouncer.ino. Open the Serial Terminal. For a short press on the pushbutton only one character should be observed. When the pushbutton is continuously held down, a string of the same character should be observed as long as the pushbutton is kept pressed.  
**Passed.**

## RX Microprocessor

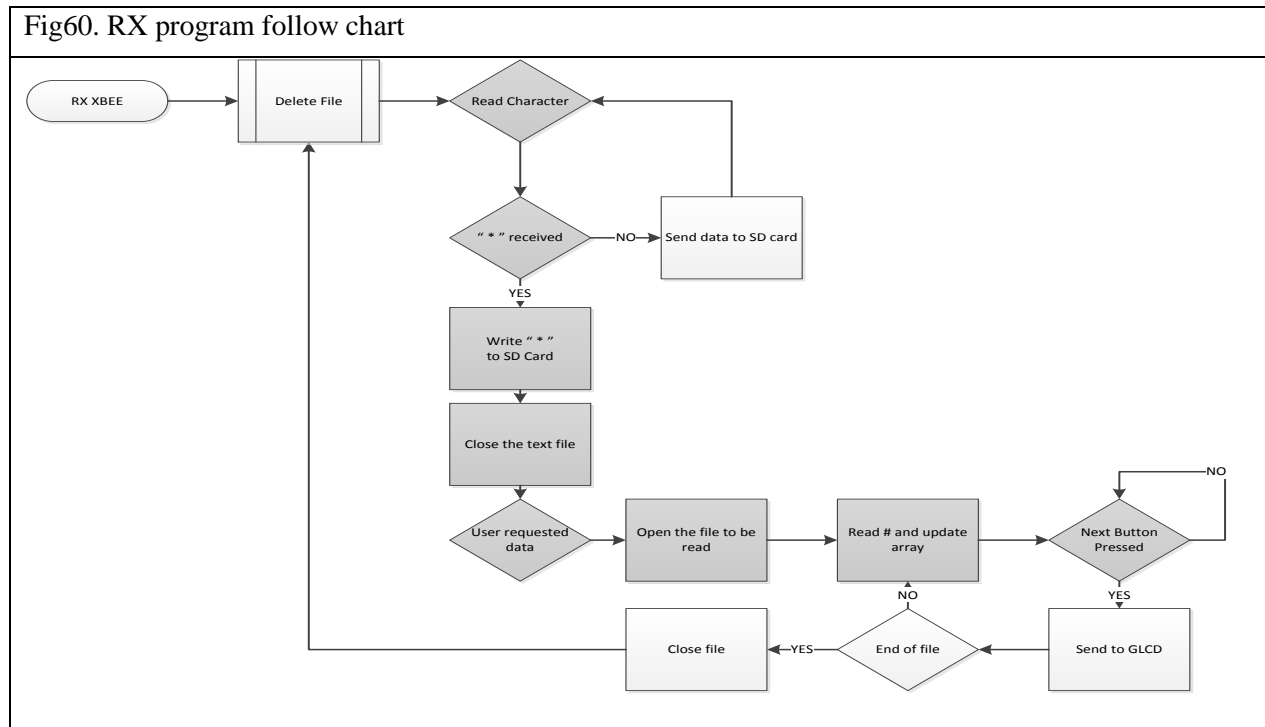
### Bootloading Atmega328 – Fuse Configuration

Follow the procedures from TX unit.

### Programming

This block uses the ATMEL328 chip to analyze the data received by the RX block. It has an algorithm that searches for the transducer pulses. As soon as a transducer pulse is observed, a counter is triggered which interrupts data analysis if the time between valid transducer pulses is greater than 3 seconds. Another algorithm works in parallel to check for data between the transducer pulses and send it to the memory block. If the time between the two transducer signals is greater than 3 seconds, the user gets the permission to access the data from the SD card. The 3 second time interval is calculated for a train traveling at 20mph. So, if the time interval is greater than 3 seconds, it means that the train passed over the site and all the wheels/bearings are scanned. Look at appendix E for RX processor program.

Fig60. RX program follow chart



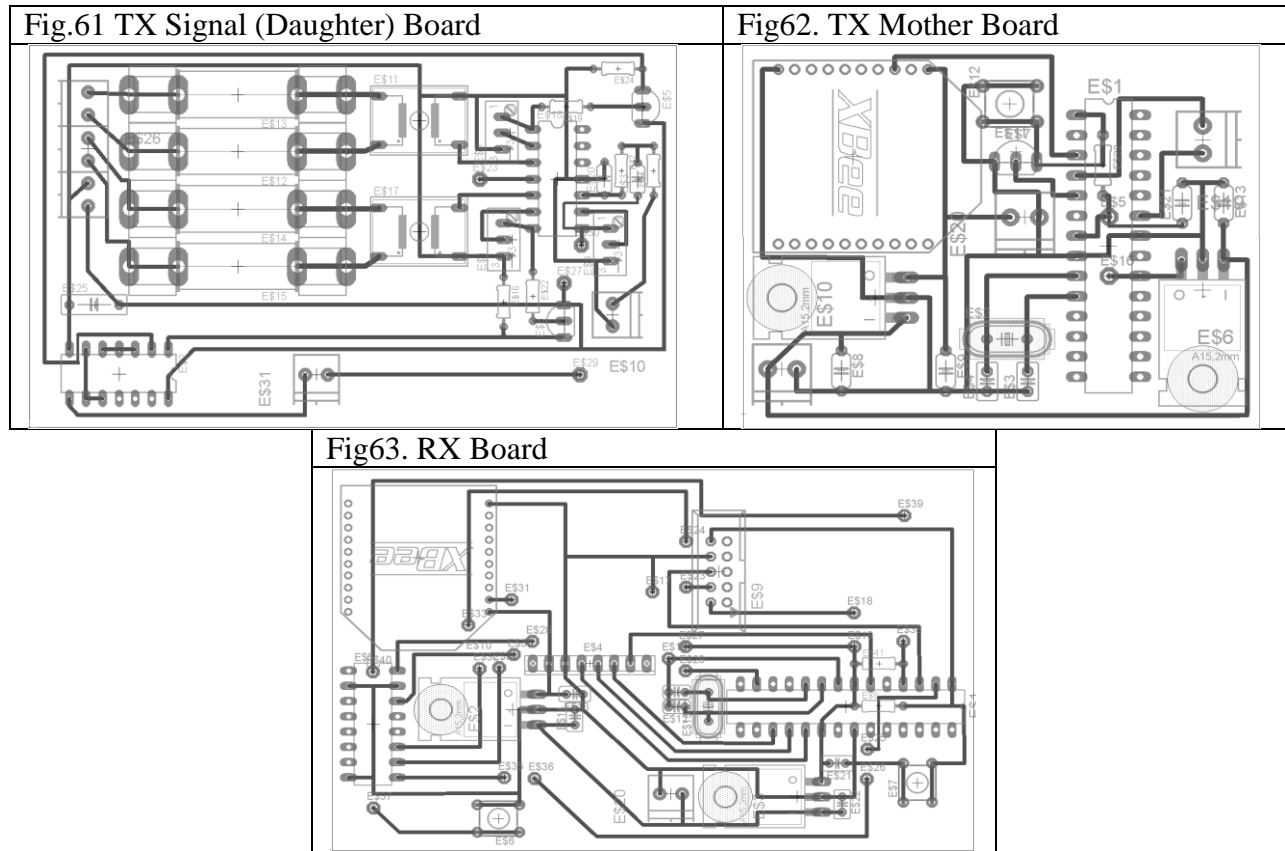
### PCB Design

Eagle design tool kit [9] was used to prepare the PCB layout of this project. The design rules followed in the design of PCBs are as follows:

- Design for one layer.
- Design for minimum area, and compact configurations.
- Design mother/daughter boards, so latter adjustments and revisions do not require change of all parts.

- Provide test-points for easy troubleshooting.
- Use appropriate routing size for specific currents, to increase reliability of operation in extreme current drives.
- Placement of general purpose block near and in one board, such as power supplies, processors and wireless units.

The boards' layouts are shown as follows:



The boards were milled, components were soldered, and the sub-circuits were tested modularly - block by block - to ensure proper operation/quality as they were functioning on the bread board design. The following images show the finalized version of our design.



Fig.64 TX Mother mounted on top of Signal Board, top view.

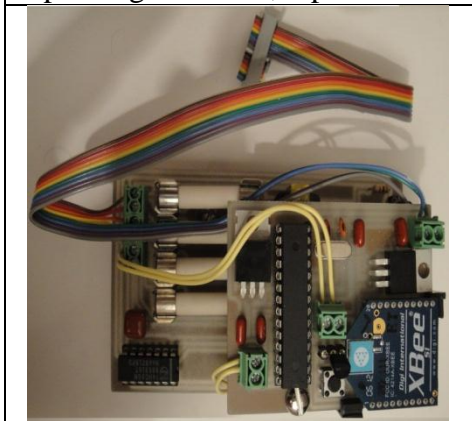


Fig.65 TX Mother mounted on top of Signal Board, side view.

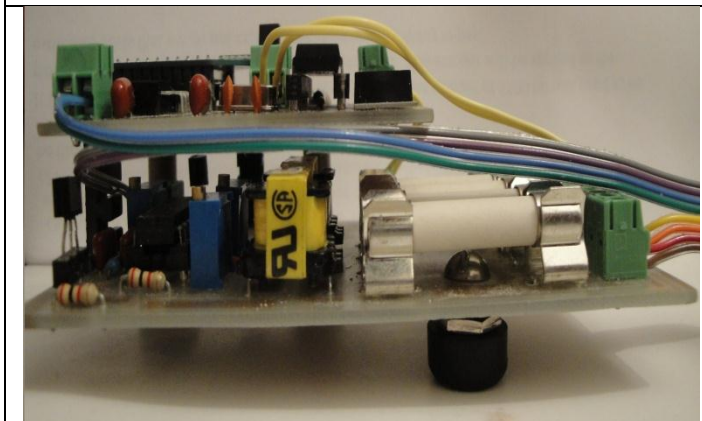


Fig.66 TX Mother mounted on top of Signal Board, bottom view.

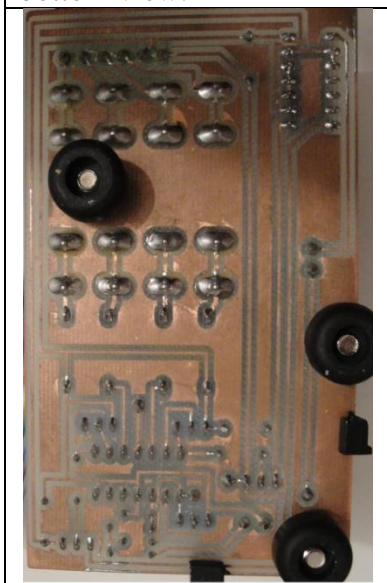


Fig.67 RX Board, Top view.

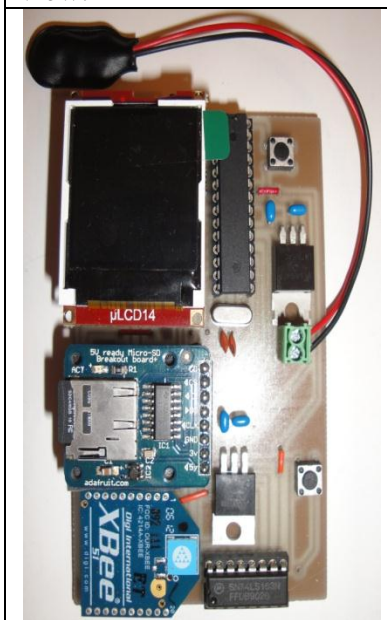
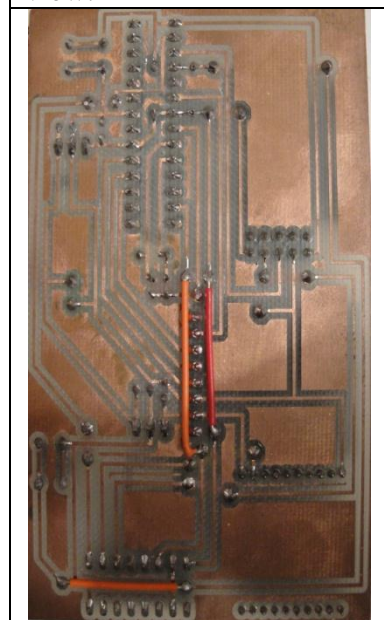
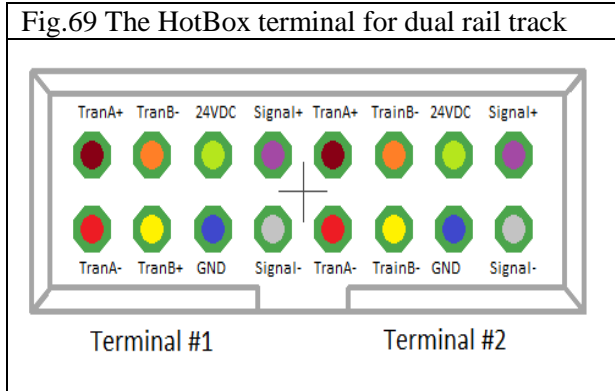


Fig.68 RX Board, bottom view.



## Field Compatibility

The TX unit header socket (female input connector) was designed to match the SDC2000 Hotbox series terminals [1]. In other word, this system is ready for field operation simply by plugging the header socket to the communication board of the Hotbox system. As the figure below indicates, there are two available terminals with the described connections on the communication board of Hotbox system. Terminal #1 is connected to the north/west bound track, and Terminal#2 is connected to the south/east track. Upon construction of two TX units with their specific wireless network ID, both tracks can be monitored using the designed RX unit.



## Cost Analysis

### Labor

Member	\$/hour	# of weeks	hours/week	Total of hours	Subtotal	Multiplier (x2.5)
Pourya Assem	40	13	14	182	\$7280	\$18200
Paul Lupas	35	13	14	182	\$6370	\$15925
Grand Total						\$34125

### Parts

Parts	Quantity	Cost/unit	Total
Atmel-Atmega328 Microprocessor	3(one burned)	\$6.00	\$18.00
Digi-XBEE 1mW with Chip Antenna Transceiver	3(one burned)	\$24.00	\$72.00
SD Card Slot	2(one burned)	\$19.00	\$38.00
SD Card 8GB	1	\$8.00	\$8.00
GLCD	2(one burned)	\$29.00	\$68.00
High impedance Audio Transformer (Choke)	2	\$7.00	\$14.00
0.5A Fuse + Holder	3	\$2.00	\$6.00
TTL Quad NAND 74LS00	1	\$1.00	\$1.00
Resistors	11	\$0.10	\$1.10
Capacitors	14	\$0.05	\$0.70
20MHZ XTAL + Isolator	2	\$1.50	\$3.00
Potentiometers	3	\$0.50	\$1.50
Connectors	8	\$0.10	\$0.80

Transistors	3	\$0.15	\$0.45
IC Sockets	6	\$0.05	\$0.30
74HC00	1	\$1.00	\$1.00
74HC153	1	\$1.00	\$1.00
LM7805	2	\$2.00	\$4.00
UA78M33	2	\$2.00	\$4.00
LM324	1	\$2.00	\$2.00
Push buttons	2	\$0.50	\$1.00
PCB	2	\$15.00	\$30.00
Screws, Rubber feets, Spacers, Wires	-	-	\$4.00
Grand Total			\$279.85

### Grand Total

Labor	Parts	Grand Total
\$34,125	\$279.85	\$34,404.85

## Testing Results/Accomplishments

Up to this point, we have the system fully working and tested as specified in the Design Requirements/Verification section. The crucial tasks of this project were successfully completed:

- Identify the transducer pulses for different train speeds (20mph to 65mph)
- Collect uncorrupted heat signature data by using the Active Filter
- Process the collected information efficiently
- Create a wireless link to facilitate data acquisition in unreachable locations or critical weather conditions
- Store the heat signature on the SD Card for further data analysis
- Create a user interface and graphical display to allow on-site diagnosis
- Make a portable RX Unit by choosing a 9V battery as the power supply
- PCB Mother/Daughter boards design to facilitate later revisions

Refer to Appendix F, for further testing results.

## Improvements

The following design changes were made to improve the overall system functionality in terms of power consumption, processing algorithms, and integrality with future sub-systems. These changes are the following:

- In the TX unit: The main power was drained by the component which could be run at lower voltages, such as amplifiers and combinational logics. Thus, we decided cut the power consumption by adding a 3.3VDC regulator to supply the power hungry blocks. Most ICs were replaced with their lower power package equivalent. These improvements cut the power consumption by nearly 40%.
- Used a level restorer, so if the incoming transducer signal is low, and the amplifier does not amplify and clip it properly, we still get a good signal level driving the combinational logic. It also prevents possible glitches of the TTL logic family used in the design, by leveling up the signal from 2V to 3.3V, because 2V is close to 1.8V threshold which is the known range of this TTL family logic. This threshold can be tolerated if the power supply is not capable of handling the circuit's drain current or the transducers' signals are critically low. This improvement guarantees the quality of operation at extreme limits caused by the varying signal level of transducers from location to location. By having a precise statistic about the ultimate bounds of the transducer signal level, the biasing network of the level restorer can be adjusted to guarantee proper functioning of the device
- Added a chain of inverters to increase the noise margins of combinational logic used to concatenate/merge transducer signals into one, which increases noise immunity of the system, and clears most of the possible glitches (noises propagated from the transducers). This is like a digital filtering of the tolerable noises.
- A waveform analysis was added to the GLCD driver. So it calculated the plotted signal average and compares it with the critical value. It also shows a conditional message stating result of this comparison.
- Most of data processing was shifted to the TX processor to cut the data traffic on the wireless module. This results in lower power consumption, less error rate and more accurate processing. The TX processor calculates the train speed as the first wheel passes by, and times the sampling time so only 16 samples are collected per wheel/bearing.
- Toroids replaced by transformers, so the frequency response is flatter over a wider range and phase delay, which is critical to proper operation of the circuit, is cut to almost 0 degrees. Furthermore, the drain current from the bungalow was tremendously reduced, because of high AC impedance of the transformers.
- The software serial was replaced by the hardware serial, to support more devices and increase the performance accuracy. This implementation utilizes a mux to overcome the task.
- Deleting file on SD card at the beginning of each stage, guarantees proper data logging.
- The socket connector configuration was redesigned to match the connections of the HotBox interference. The device is ready to be directly installed in the bungalow.
- Lowered the power on the RX and do most of the processing on the TX
- Developed data transmission protocol/technique instead of using heavy programs or libraries to package/unpackage data. This makes the circuit faster and robust.
- Going to lower use 3.3V for microcontroller as well, but the chip were already bootloaded; On the other hand 3.3V only supports up to 8MHz.
- Added an automatic hard reset on the TX unit processor. So the timer are refreshed and prevents the chip to reset their timer cycle in middle of processing a train, which can mess up the calculations.

## Difficulties / Uncertainties

A difficulty that came along the way was finding a way to simulate the transducer pulses of the train system. After many considerations, we have chosen the function generator as the transducer signal simulator. This works as follows: the square wave is set at a certain frequency that corresponds to the train's speed (3Hz for 20mph and 10Hz for 65mph). To generate an active pulse (showing a passing train) the duty cycle is set between 25% - 40%.

## Conclusion

### Wrap-up and future work

All the progress stated in the presented report is considered fully tested (under extreme conditions, to provide reliability) and functional. Some items/units are added to increase the compatibility of the systems, and some to resolve unpredicted design flaws.

The overall system was designed to allow upgrades and easy replacement if components are not functional. As it is the case with electronic systems, this project can also benefit from extra features as follows:

- Increase the wireless range (if necessary) by using a higher power TX XBEE
- Upgrade the user interface by adding a touchscreen which also replaces the GLCD
- Time stamp the data collected for each train, using a time stamp IC/Algorithm
- Save a multitude of train signatures on the SD Card and navigate through them
- Collect data from multiple tracks using only one TX unit by extending the daughter board
- Implement a Bluetooth Module to connect to an Android App and analyze the data on a mobile device as the train passes by

## Ethical considerations

We adhered to the statements of the IEEE Code of Ethics in designing and testing each subsystem as follows:

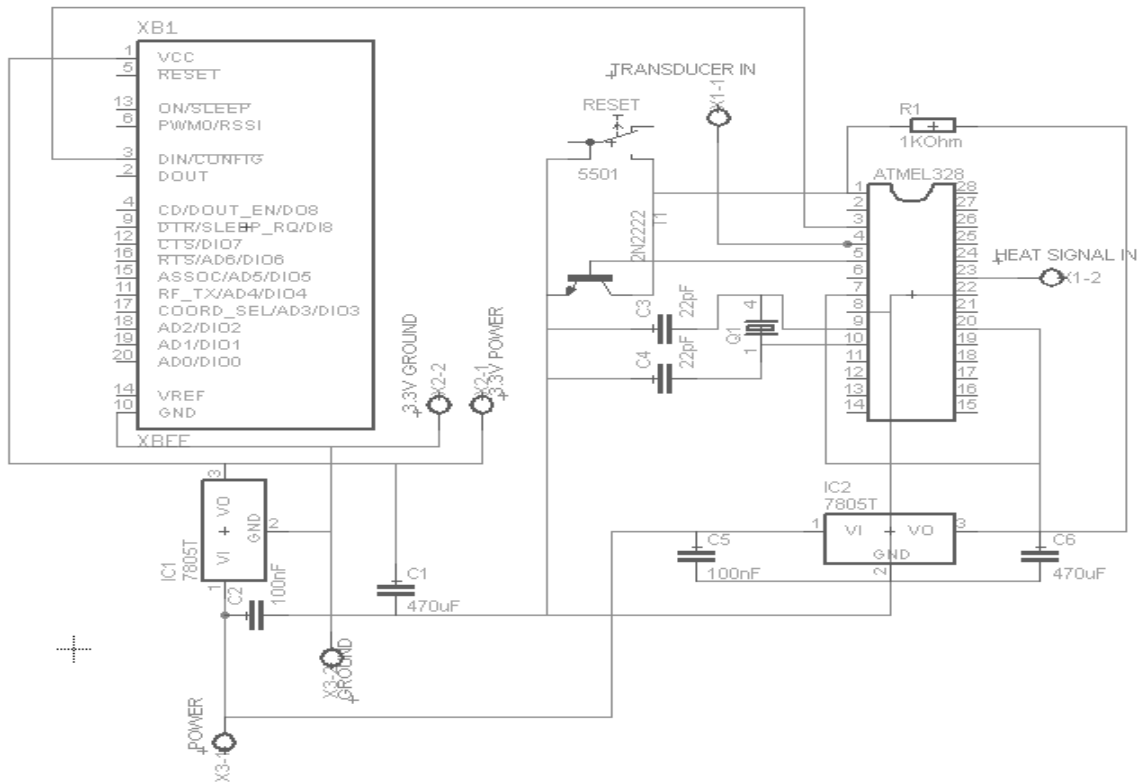
- *“3.to be honest and realistic in stating claims or estimates based on available data”*  
We ensured that all calculations are accurate by consulting the referenced books. All the conclusions drawn from experimental procedures are supported by data calculations and graphs obtained from measurements.
- *“6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;”*  
We applied our analytical and technical skills learned thus far to the best of our abilities in creating a product that will promote railroad safety. Moreover, we sought professional help from TAs and instructor when our experience was limited for a certain task. We continuously upgraded and refined the system, based on what we learned.
- *“7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.”*

In this group project, each person communicated his ideas on improving the design and provided feedback on the other person's work in terms of quality and things that needed improvement. We cited and credited all outside sources. We developed and own the IP, it is a new idea and there is no similar comparable work available on the market. We modified and updated our design based on the guidelines provided by our advisers to improve functionality, rationality, safety, and reliability.

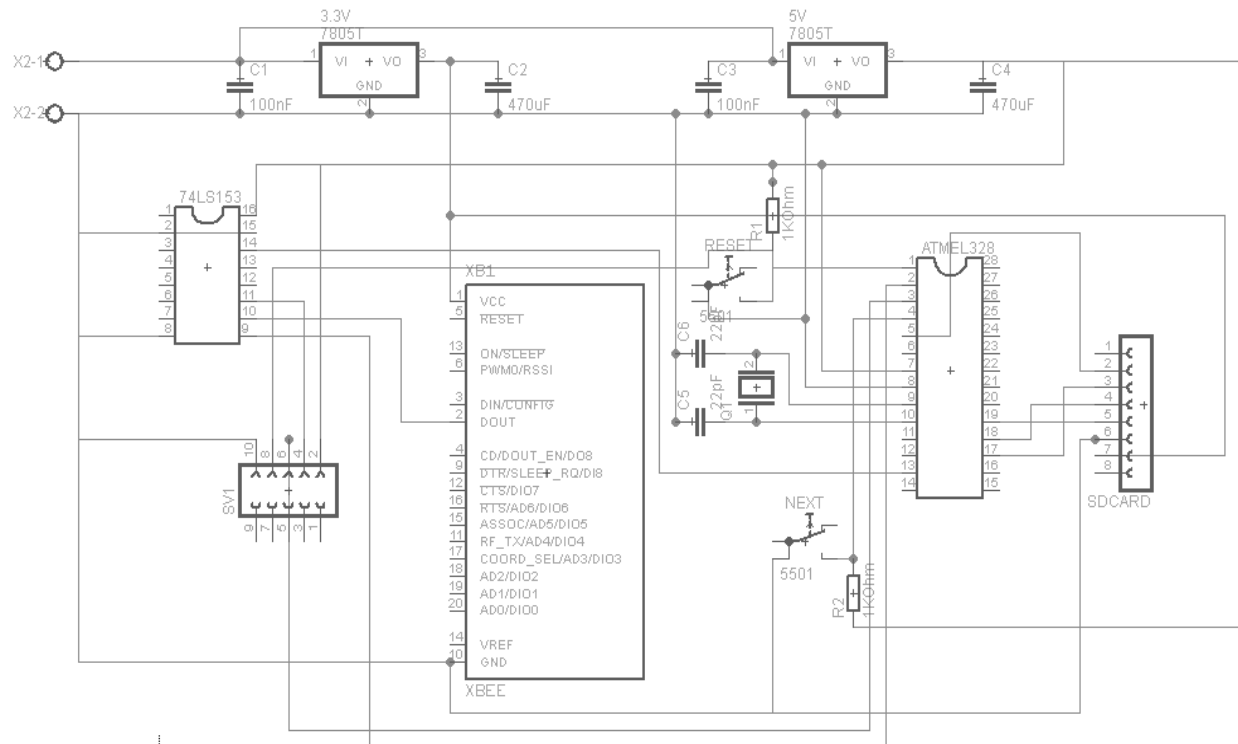
## References –need to work on

- [1] Courtesy of CN RR authorized by Pourya Assem
- [2] Boot loading procedures for the ATMEL328 microprocessor: Courtesy of CN RR, authored by Pourya Assem
- [3] "ATmega328." - *Atmel Corporation*. N.p., n.d. Web. 23 Oct. 2012.  
<<http://www.atmel.com/devices/atmega328.aspx>>.
- [4] Faludi, Robert. *Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing*. Farnham: O'Reilly, 2010. Print.
- [5] "TRONIXSTUFF." *TRONIXSTUFF*. N.p., n.d. Web. 23 Oct. 2012.  
<<http://tronixstuff.wordpress.com/2011/02/18/tutorial-arduino-and-tft-lcd/>>.
- [6] "ÂµLCD-144(SGC)." *ÂµLCD-144(SGC)*. N.p., n.d. Web. 23 Oct. 2012.  
<<http://www.4dsystems.com.au/prod.php?id=121>>.
- [7] "Downloads - Displayshield4d - Arduino Library for the 4Display-Shield by 4D Systems - Google Project Hosting." *Downloads - Displayshield4d - Arduino Library for the 4Display-Shield by 4D Systems - Google Project Hosting*. N.p., n.d. Web. 23 Oct. 2012. <<http://code.google.com/p/displayshield4d/downloads/list>>.
- [8] "Micro SD Card Tutorial - Using SD Cards with an Arduino!" *Micro SD Card Tutorial - Using SD Cards with an Arduino!* N.p., n.d. Web. 29 Sept. 2012. <<http://www.ladyada.net/products/microsd/>>.
- [9] "Downloads." N.p., n.d. Web. 09 Dec. 2012. <<http://www.cadsoftusa.com/download-eagle/?language=en>>.

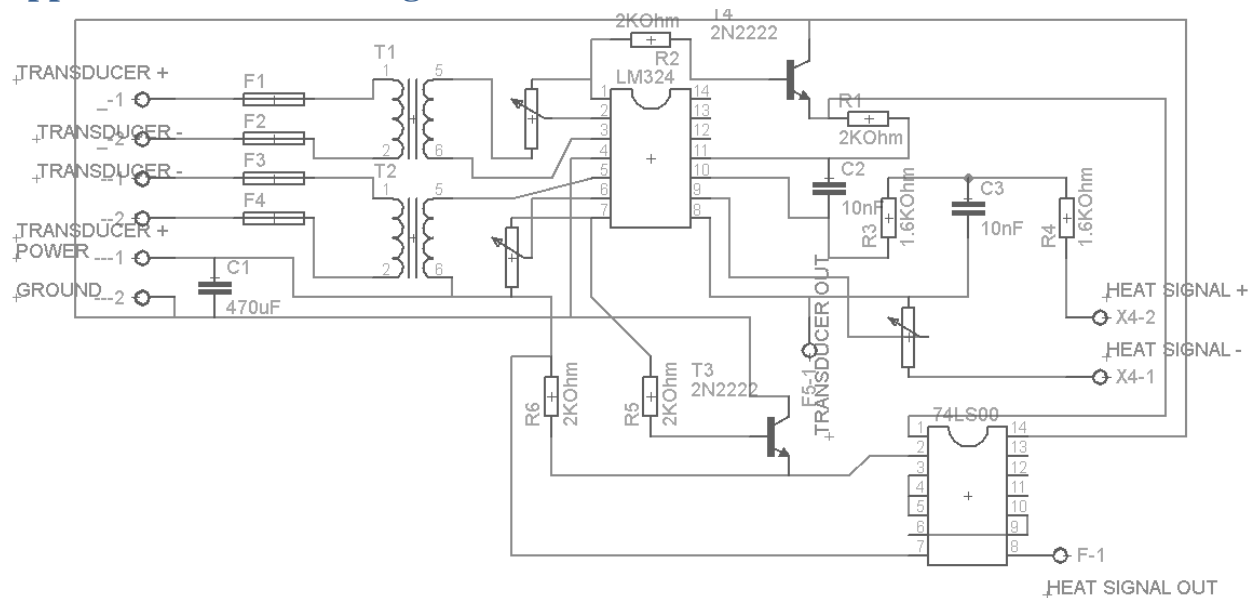
## Appendix A – TX Unit Mother Board



## Appendix B – RX Unit



## Appendix C – TX Unit Signal Board



## Appendix D – TX Processor Program

Program	Continued	Continued
<pre> unsigned long timer = 0; unsigned long halt = 0; boolean flag = true; int counter = 0; int gate_open = 0; int gate_close = 0; unsigned long time = 0; int array[16] = {   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,   0 }; void setup() {   pinMode(2,INPUT);   pinMode(3,OUTPUT);   Serial.begin(115200); } void loop() {   digitalWrite(3,LOW);   if(digitalRead(2) &amp;&amp; (counter &lt; 2))   {     timer = millis();     while(digitalRead(2))     {     }     if((millis() - timer) &gt; 15)     {       counter++;     }   } </pre>	<pre>     if(counter == 1)     {       gate_open = millis();     }     if(counter == 2)     {       gate_close = millis();     }   }   else   {     counter = 0;   } } if(counter == 2) {   int sample_time = (gate_close - gate_open) / 17;   for(;;)   {     while(digitalRead(2))     {       if(flag)       {         halt = millis();         flag = false;       }     }   } } </pre>	<pre>     }     time = millis() - halt;     if((time &lt;= 7))     {       Serial.print('*');       delay(1000);       digitalWrite(3,HIGH);     }     else     {       flag = true;       for(int i = 0; i &lt; 16; i++)       {         array[i] = analogRead(A0);         delay(sample_time);       }       for(int i = 0; i &lt; 16; i++)       {         Serial.print(array[i]);         Serial.write(',');       }       Serial.write('#');       //Serial.println(' ');       flag = true;     }   } } } </pre>



## Appendix E – RX Processor Program

[illegible]

<pre> lcd.setFontmode(OLED_FONT_TRANSPARENT); delay(50); lcd.drawstringblock(5, 5, 0, lcd.RGB(255, 255, 255), 1, 1, "HOTBOX CAL SYS"); delay(50); lcd.drawstringblock(5, 18, 0, lcd.RGB(0, 255, 0), 1, 1, "VERSION 1.0"); delay(50); lcd.drawstringblock(5, 31, 0, lcd.RGB(255, 0, 255), 1, 1, "ECE445 DEMO"); delay(50); lcd.drawstringblock(5, 44, 0, lcd.RGB(255, 0, 0), 1, 1, "Design Team:"); delay(50); lcd.drawstringblock(5, 57, 0, lcd.RGB(255, 0, 0), 1, 1, " *Pourya Assem"); delay(50); lcd.drawstringblock(5, 70, 0, lcd.RGB(255, 0, 0), 1, 1, " *Paul Lupas"); delay(50); lcd.drawstringblock(5, 93, 0, lcd.RGB(255, 0, 255), 1, 1, "GIVE US A+"); delay(50); </pre>	<pre> if ((millis() - time) &gt; debounce) {     switchstate = reading;     if (switchstate == LOW)         ready = true; //next key has been pressed     }     previous = reading; }  //Display the next wheel signature  total = total / (15 * 8); //Serial.println("here"); lcd.Clear(); // clear LCD delay(50); lcd.drawstringblock(5, 5, 0, lcd.RGB(255, 255, 255), 1, 1, "HEAT SIGNATURE"); delay(50); lcd.drawstringblock(5, 18, 0, lcd.RGB(255, 0, 255), 1, 1, "ALARM STATUS:"); delay(50); </pre>	<pre> //Serial.println("remove"); //delay(1000); } </pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------

## Appendix F – Testing

Below is a sample data logged by the system into the SD card. As shown, the window's length is 16 for all heat profiles. Each profile is distinguished by '#' and each sample point within the profile is distinguished by a ','. The file is ended by '\*'. These are the handles which RX unit uses to fetch, access and analyze the data received from the TX unit. As shown on the right, the heat

```

301,237,176,120,72,35,10,0,2,19,50,91,143,
201,264,329,#417,47542,563,565,546,508,4
53,387,313,237,164,100,48,14,0,4,29,#198,2
73,349,420,481,528,557,567,556,525,477,41
5,343,267,193,123,#3,0,17,52,105,170,244,3
20,394,460,512,548,566,563,539,495,#298,2
22,151,88,41,10,0,8,37,83,144,215,291,366,
435,491,#566,551,517,465,400,327,251,177,
111,57,19,0,2,24,64,118,#336,409,472,521,5
54,567,559,532,487,427,356,281,205,135,76
,31,#12,45,95,159,231,307,381,449,504,543,
564,564,544,506,451,382,#162,98,47,14,0,5,
30,74,133,203,277,353,424,484,530,555,#52
3,475,412,264,190,121,65,24,2,0,18,55,108,
174,247,#*

```

