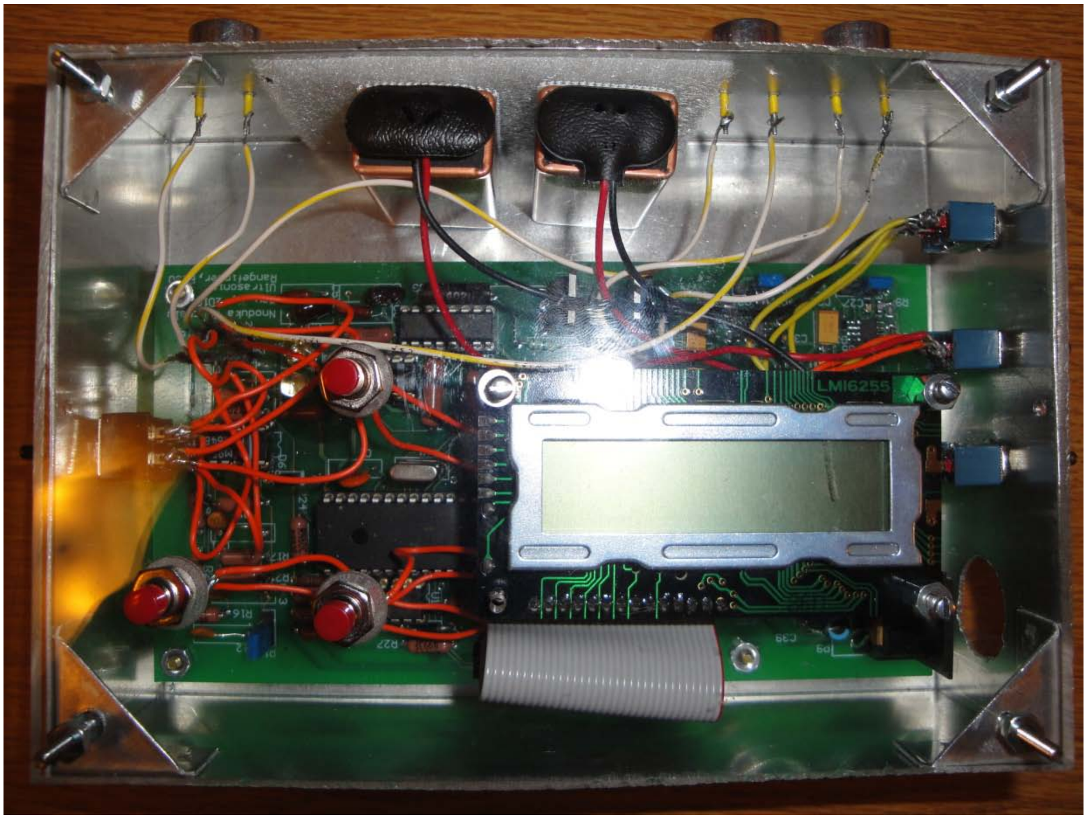
Ultrasonic Rangefinder Project
EE90, Spring 2010



Nnoduka C. Eruchalu
MSC 271, Caltech
(626)-660-4830
nnodukae@caltech.edu

Ultrasonic Rangefinder Project

Nnoduka C. Eruchalu
MSC 271, Caltech
(626)-660-4830
nnodukae@caltech.edu

Table of Contents

Getting Started

Introduction and Project Overview

User Manual

Detailed Project Description

Commented Software

General Safety Summary

To avoid fire, personal injury, or even worse total destruction of the rangefinder:

Use Proper Power Cord. Use only the power cord specified for this project and certified for use in the U.S.A. This power supply should be stable to avoid erratic behaviors of the rangefinder.

Connect and Disconnect Properly. Do not just pull out the power cord. This could break solder joints, destroy the connectors or break the board.

Do Not Operate With Suspected Failures. If you suspect there is damage to this product, have it inspected by a qualified past EE9x student, preferably the maker of this particular project, Nnoduka C. Eruchalu.

Do Not Operate at Extreme Temperatures. Room temperature is best. It is easiest to appreciate the art of range finding at decent temperatures.

Handle Rangefinder with Caution. While it is a sturdy product careful treatment will increase its lifetime. So do not move it around in quick and jerky motions, and store it in a safe location.

Keep Out of the Reach of Children. But do not deprive them of the joy of seeing a quality rangefinder in action.

Do Not Operate in Wet/Damp Conditions.

Keep Ultrasonic Rangefinder Clean and Dry.

Introduction and Project Overview

Motivation

The idea behind this project was to get some practical experience at building analog circuits. I had the theoretical background required for the task at hand, and now I got a chance to put what I had learnt about analog electronics to practice. This gives me an opportunity to go through an entire product development cycle- design, construction, debugging, and packaging in a professional manner.

Project Description and Specification

Project Description

The project for this term was the design and construction of an Ultrasonic Rangefinder. The project was implemented as an analog/digital hybrid.

The design requirements were to get a range of at least 2m with distance measurement accuracy up to at least a few centimeters from the rangefinder. With proper design and testing I was able to get my maximum range up to at least 8m. Getting readings beyond 8m is possible but hard and requires a very wide space so as to minimize bouncing off the walls. Also the distance measurement accuracy is to 1cm. This accuracy had to be achieved because in the short range mode which can get ranges down to 3cm, it is important to be able to distinguish 3cm from 4cm. Of importance also is the final project packaging and its ranging abilities in its finished form.

The project features, packaging and dimensions are discussed in detail:

Project Features

The ultrasonic rangefinder has the following noteworthy features:

1. Handheld -The size of the rangefinder and the battery power option makes it both handheld and portable
2. Mountable on any surface - The rangefinder has rubber feet to keep it fairly well mounted on most surfaces, even rugs.
3. Sturdy - The sheet aluminum metal casing with a 1/4-inch plexi-class makes for a sturdy rangefinder. The rubber feet/stands also help protect the rangefinder from many impulsive forces
4. Power selection - The user can choose to use either wall power or battery power.
5. Range Modes - The user should start off taking measurements in long range mode for an ability to measure distances up to at least 8m. For distances less than 40cm the user should switch out of the long range mode and go into the short range mode to get measurements down to 3cm.
6. Accuracy - The rangefinder is accurate to 1cm and this is especially important for the short range mode where 3cm should be distinct from 4cm.
7. Units - The rangefinder is accurate to 1cm, however if the user wants to get the equivalent nearest inch, all he has to do is push the UNITS change button. This will toggle the distance measurements between cm and inches. The measurements will still be accurate to 1cm but will be converted to the nearest inch if the display units is inches, and so it will appear that the measurements are only accurate to 1 inch. However the user can always go back to the accurate cm measurement by pushing the UNITS change button again.

Project Packaging

The circuitry is all done on a printed circuit board (pcb) with the only external peripherals being:

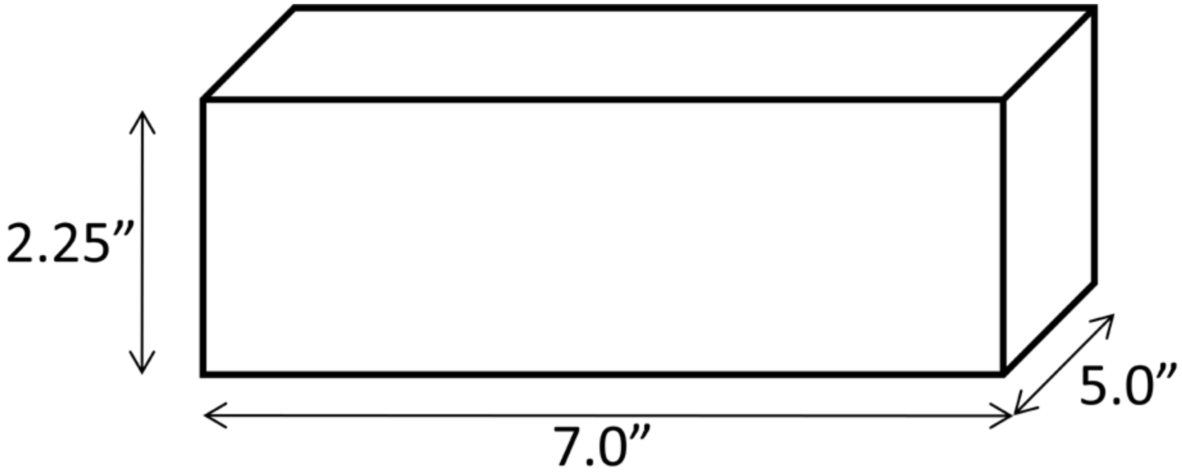
-Two 9V batteries and a wall power pack.

- Switches
- Transducers
- An LCD Display

Using the materials available to me in the Caltech EE stockroom I decided on using an aluminum sheet metal box with a plexi-glass lid. This way the curious user can get a good view of the internal circuitry. The power switches and DIN-5 connector slot are all on one side of the box, and the mode select switch is on the other side of the box. The control switches are all on the top of the box, protruding through the plexi-glass.

Project Dimensions

Below is a block image of the dimensions of the final product.



This does not take into account the switches on either side which protrude by 0.75” on both sides, and the transducer which protrude by 0.4” on the front panel.

The height of 2.25” actually includes the 0.25” thickness of the plexi-glass lid and the 2” height of the aluminum sheet-metal enclosure.

Characterization of Electrical Performance

Below is a table characterizing the electrical performance of the various important components of the Ultrasonic Rangefinder

Parameter	Minimum	Typical	Maximum	Unit
Battery Voltages	7	9	35	V
Oscillator Frequency	39.7	40.0	40.3	KHz
Transmitted Sine Wave Amplitude	-	26	-	V
Input Current from +ve battery/ wall supply	300	400	500	mA
Input current from –ve battery	50	70	100	mA

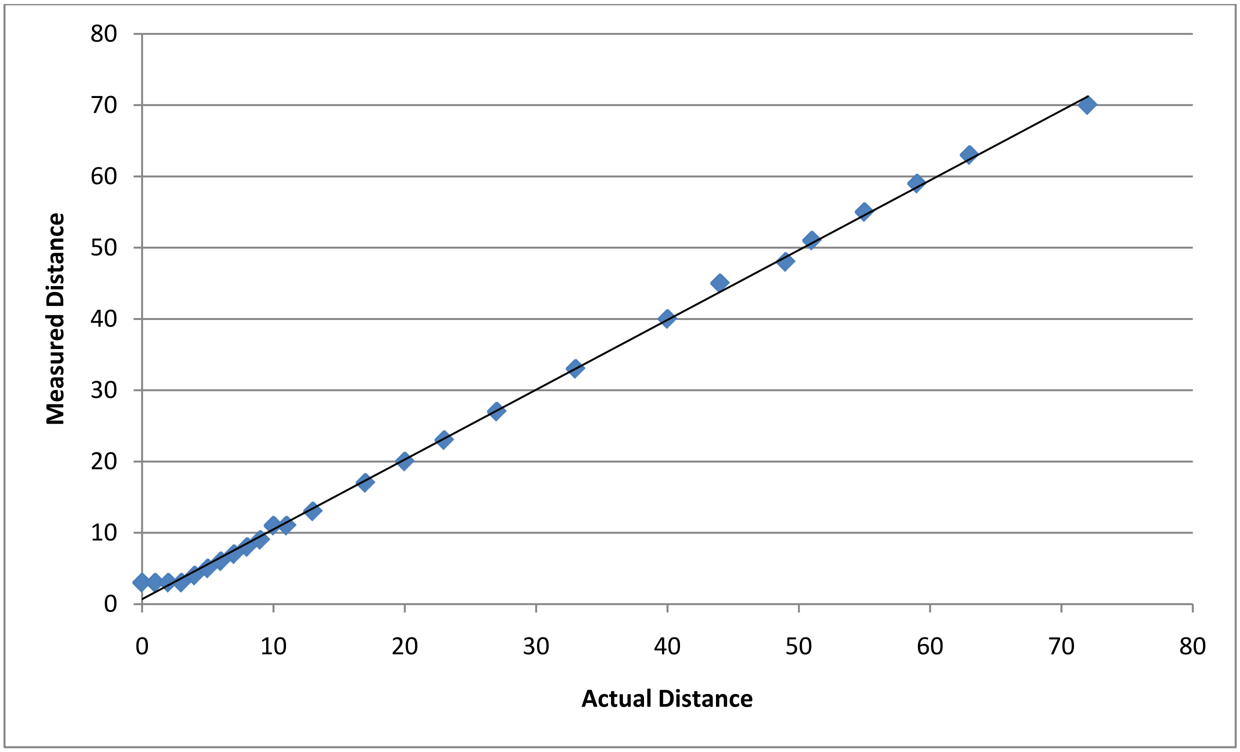
Characterization of Product Performance

Below is a table characterizing the product performance of the Ultrasonic Rangefinder.

Parameter	Minimum	Typical	Maximum	Unit
Short ranges	30	-	800+	cm
Long ranges	3	-	70	cm
Accuracy	1	1	1	cm

Note that Short ranges refer to the ranges detected in the short range mode and Long ranges refer to ranges detected in the long range mode.

Below is a plot which shows the accuracy of the Ultrasonic Rangefinder at various distances and shows the correlation between measured and actual distances when the system is in the **short range mode**.

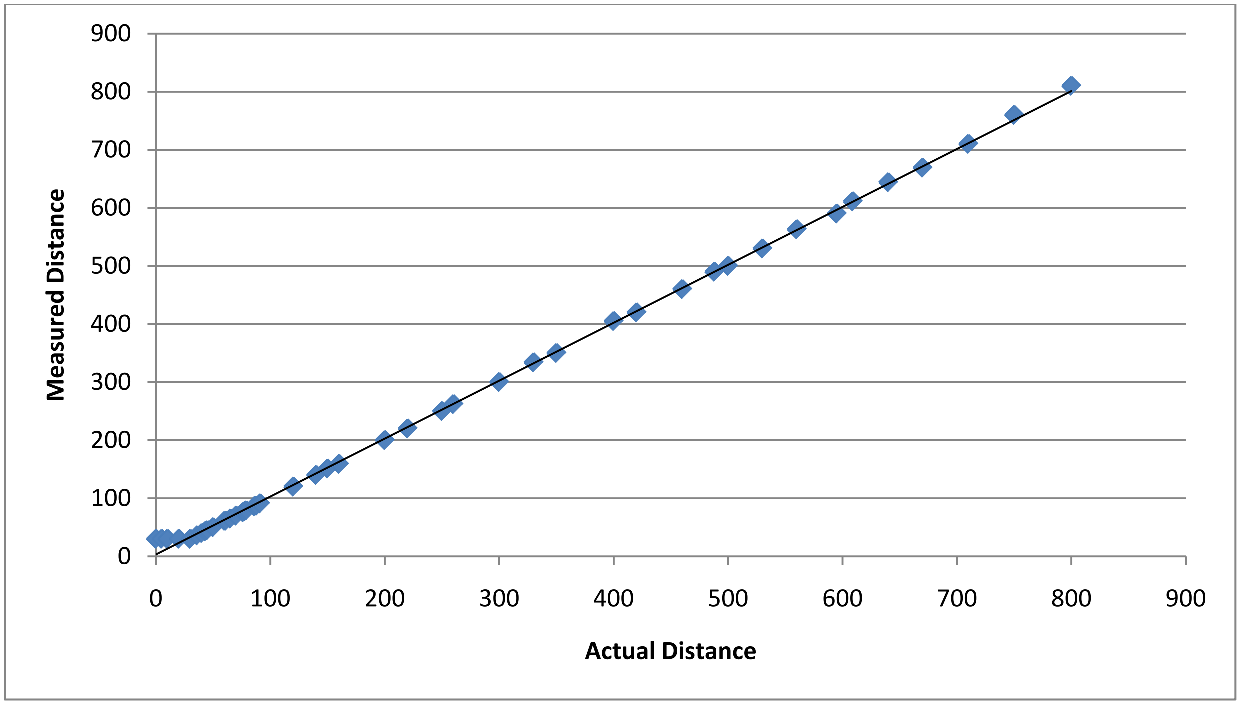


The correlation of this plot is: 0.999386

The equation of the regression line is: y = 0.979551x + 0.689366

where y = measured distance and x = actual distance

Below is a plot which shows the accuracy of the Ultrasonic Rangefinder at various distances and shows the correlation between measured and actual distances when the system is in the **long range mode**.



The correlation of this plot is: 0.999626

The equation of the regression line is: y = 0.99731x + 3.185185

where y = measured distance and x = actual distance

Challenges faced

The **biggest circuit challenge faced was getting the boost and inverter converters to work**. I went through a lot of different DC/DC converter chips before getting to one that could work for my high current drawing circuitry. Not many boost converter chips could take in the input current necessary to provide the high output current needed by the application circuitry. So this usually led to their blowing up or a drop in their output voltage which is to be expected by the conservation of energy.

When I finally got to the MAX629 chips and realized that they could meet the current requirements of my chip, I had to figure out how to use them safely. That is I had to figure out how to not suddenly start and stop them and how to sequence their turning on and off. The necessity for sequencing the power supplies is something that came up during testing. It seemed that trying to switch on both the boost and inverter converters on at the same time was not possible and was for some reason surpassing the specs of the switch. So I had to separate them into multiple switches. And with that I still had to turn on the inverter converter then the boost converter for the both of them to work properly. This is yet to be fully understood but I do know that it has something to do with the instability of the inverter converter which generates -27V with very low frequency oscillations.

For these reasons I considered the boost and inverter converters to be fragile circuitry and only used them to drive the two power op amps of the transmitter circuitry.

Lessons Learned

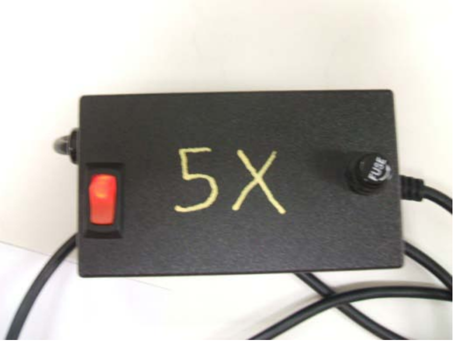
The biggest thing I learned here was how to design and complete a project within a given time frame. I learned of the importance of creating weekly milestones and diligently following them. This project also improved my confidence my abilities to put my analog theory to practice while getting to interface my analog circuitry with digital components that were also new to me. I had only used microprocessors and FPGAs before this class and I decided to use a microcontroller this time around so as to learn some new digital EE even in an analog project class.

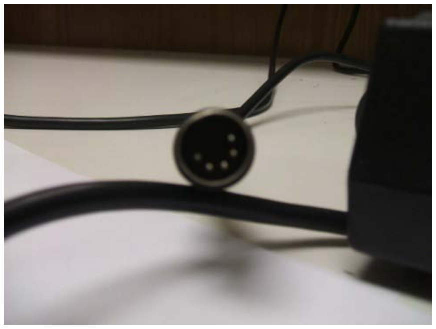
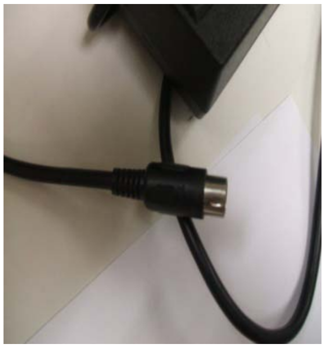
Of course I was also forced to learn how to use a drill press and other tools in the EE machining shop so as to be able to create my packaging. It was a tough and painful experience but now with a finished and well packaged product it feels well worth it.

User Manual

Connecting Power

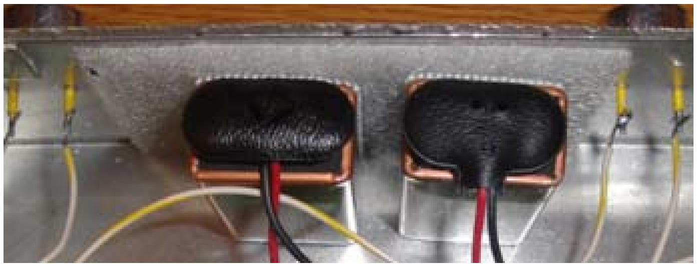
The power for the boost converter is either two 9V batteries or a wall supply unit. The wall supply unit is either an Elpac or a compatible power supply, which fits into the 5-pin DIN connector. Below is a sample of the best option for the wall power supply unit (one with a light indiating on/off status)



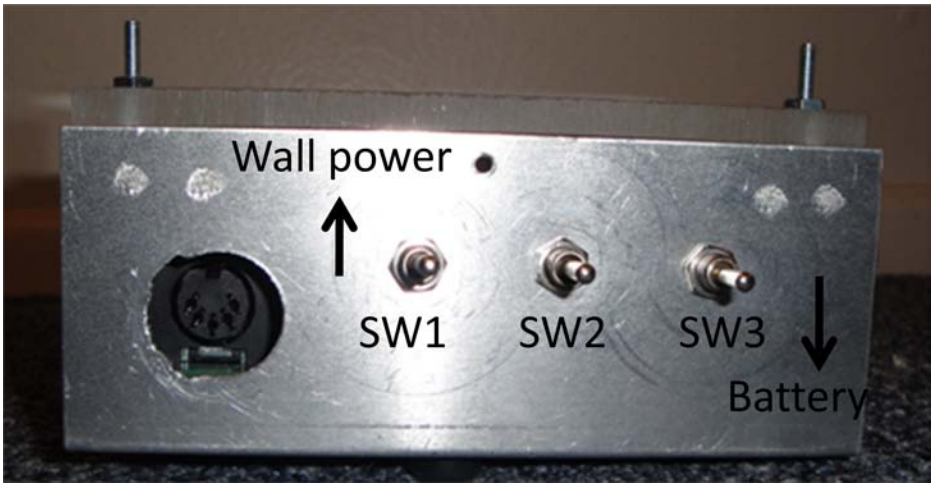
The batteries are connected inside the product. To use new batteries the user has to attach Velcro strips to one side of each battery and attach them to the Velcro surface between the transducer leads.

Below is a picture showing the batteries attached to the Velcro surface.



Controlling and Operating the Rangefinder

Now that we know how to connect the appropriate power supplies, the next step is to show how to sequence the power supplies and how to select battery or wall power mode. Below is a picture showing the power switches and the DIN-5 connector on a side of the product.



The switches are labeled SW1, SW2, SW3

They have to be turned on in the order SW1 then SW2 then SW3 and turned off in the reverse order SW3 to SW2 to SW1.

To use wall power, turn on all the switches by pushing them up as shown in the diagram and to use battery power turn on all the switches by pushing them down. These are SPDT switches so their off state is when they are in the horizontal position as shown above.

Now moving on to discussing the inputs and outputs:

Inputs:

Below is a table with the input devices and their descriptions

Input Device	Description
CALCULATE switch	This push button switch measures how far away the object the rangefinder is pointed at is. It returns the results on the LCD. This switch is located on the top of the product, on the plexi-glass.
UNITS switch	This push button switch toggles the units of range measurement

	between cm and inches. This switch is located on the top of the product, on the plexi-glass.
RESET switch	This push button switch resets the system. Really it resets the microcontroller in the system which is effectively resetting the system. This switch is located on the top of the product, on the plexi-glass.
MODE_SEL switch	This is a toggle switch used for selecting either long range or short range mode. When it is pushed up, then the system is in long range mode and when it is pushed down the system is in the short range mode. This switch is located on the left side of the box, thus on the opposite side from the power switches.

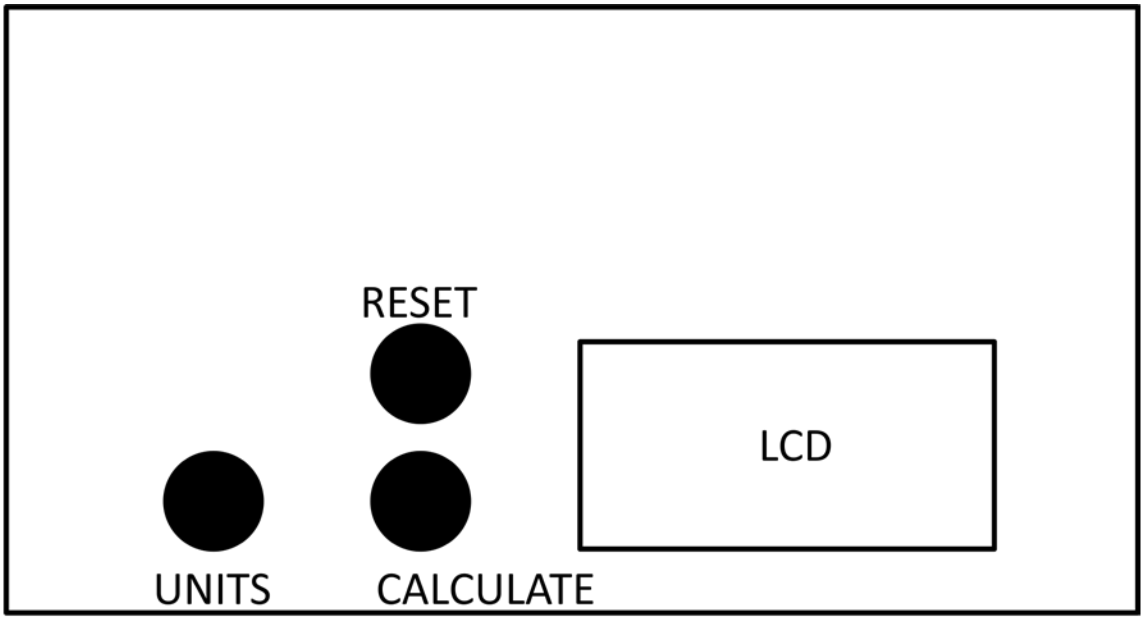
Outputs:

Below is a table with the output devices and their descriptions

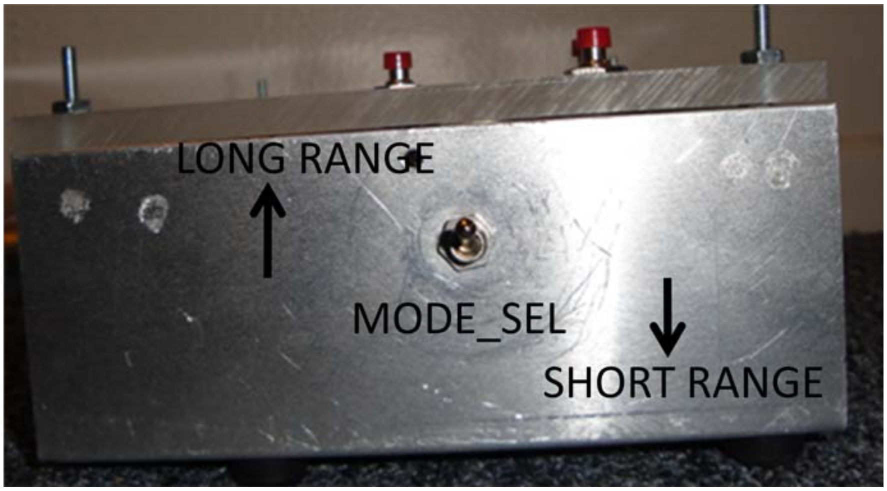
Output Device	**Description**
Display	This is a 2-line 16 character per line LCD. It will display the most recent distance measurement in either inches or cm. The default measurement unit is in cm. So the system starts up with the ranging being done in cm. The Display is located on the top of the product, right below the plexi-glass.

User Interface

Below is a depiction of the top of the box showing the user interface:



Below is an image showing the MODE_SEL switch’s placement and operation:



Example Images

Below are some images of the final product.

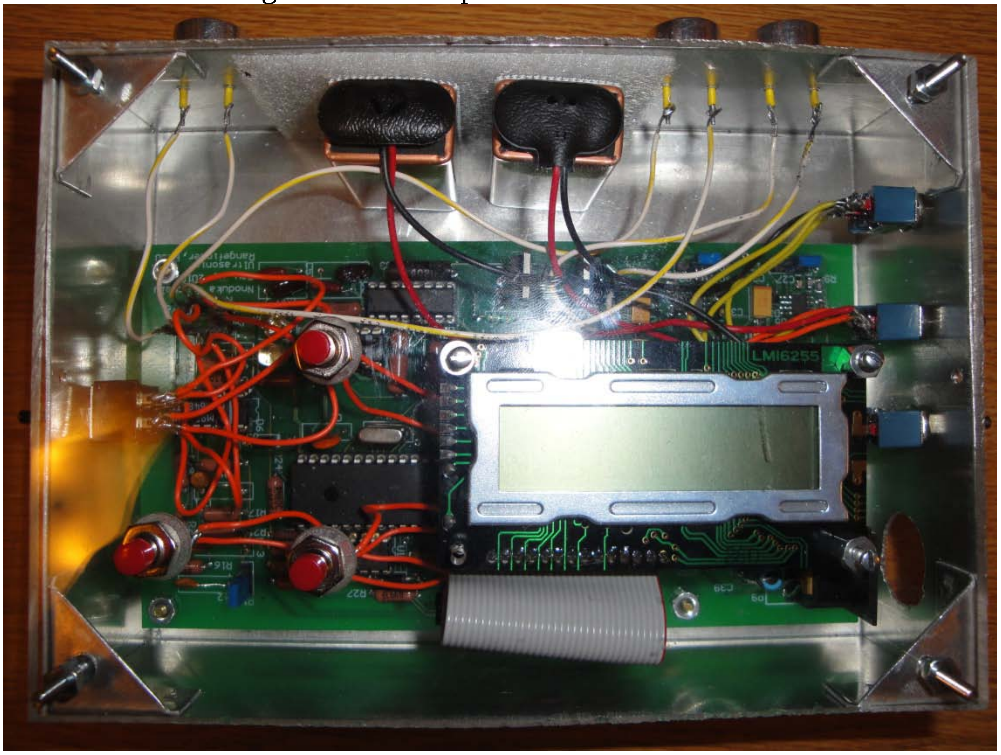


Figure 1- Top View

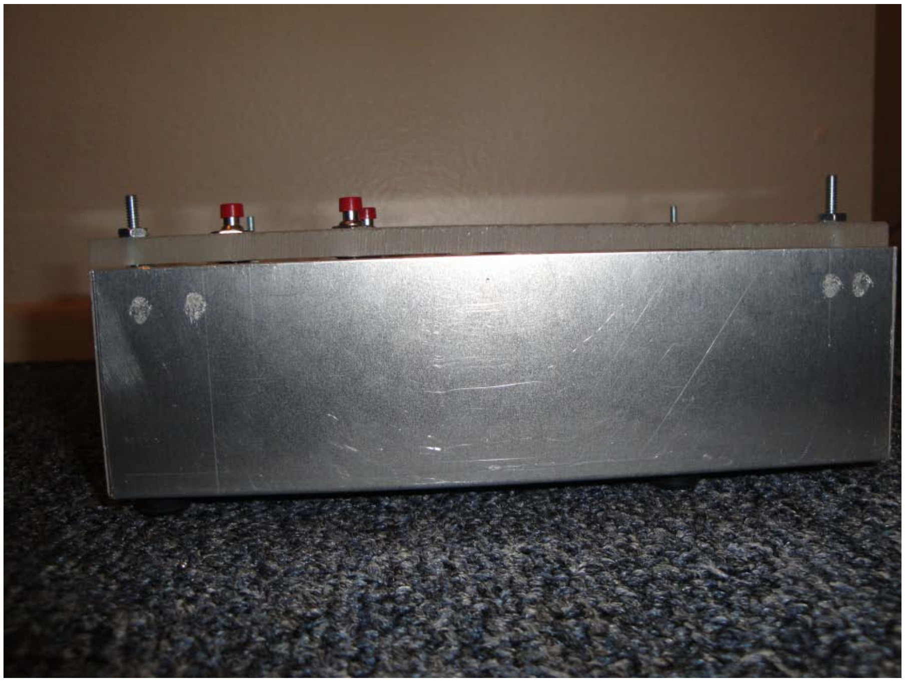


Figure 2- Back view

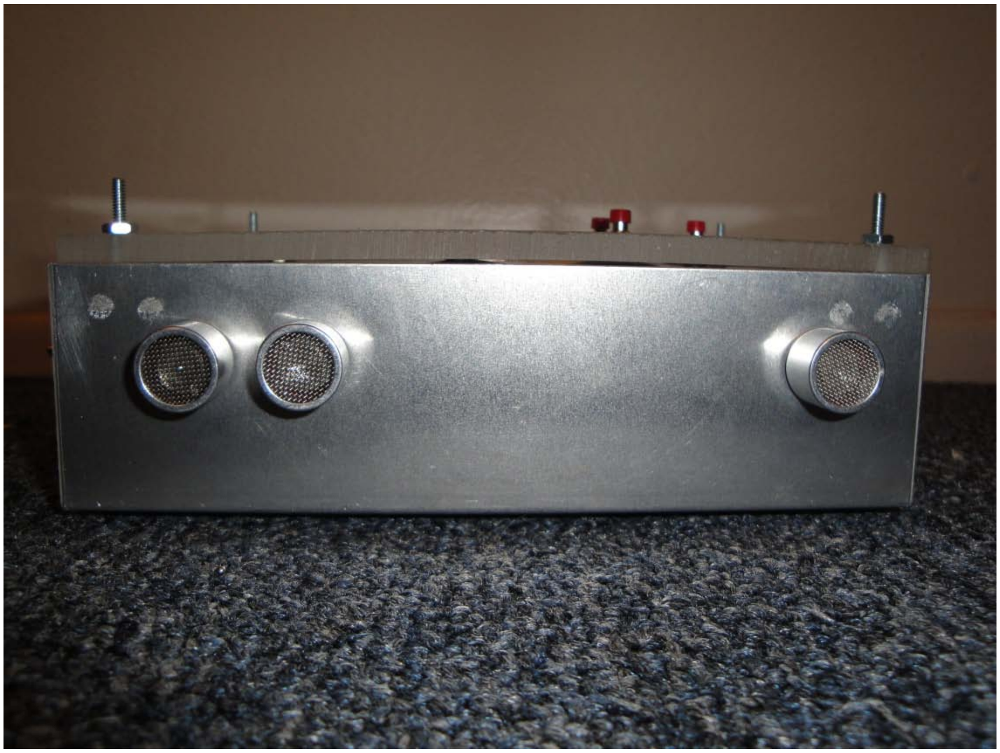


Figure 3- Front view

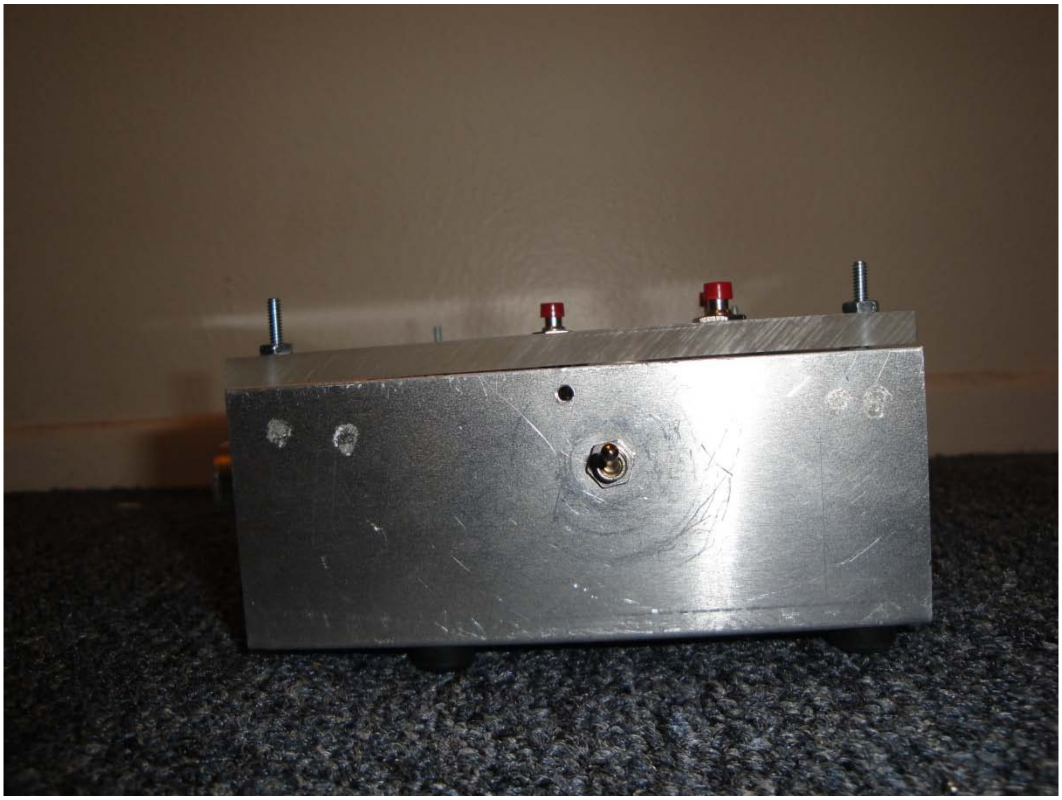


Figure 4- Left side view

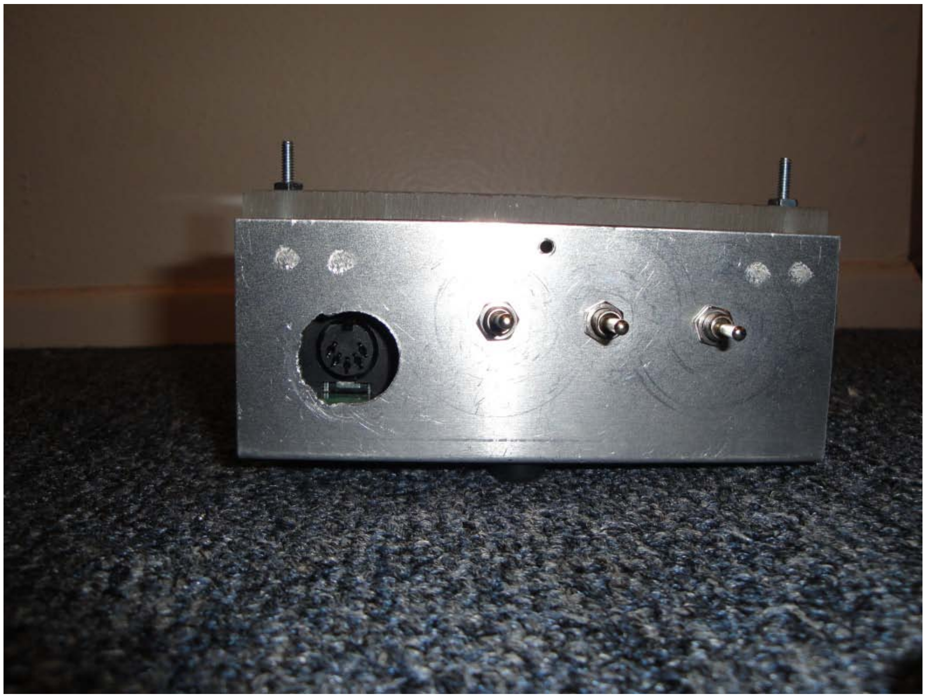


Figure 5- Right side view

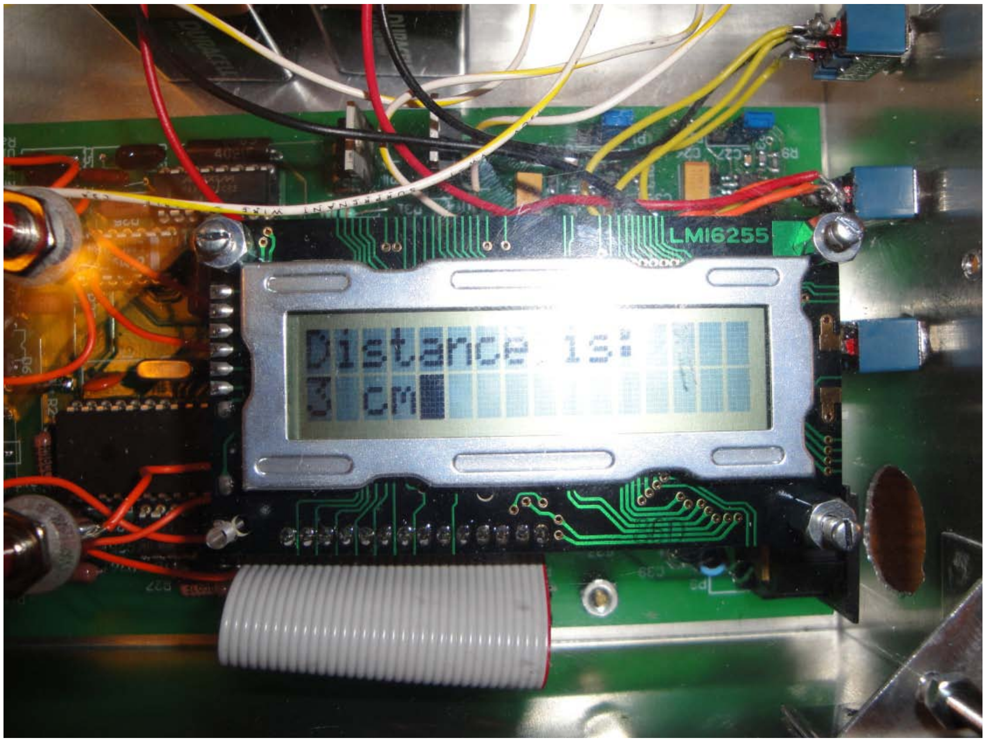


Figure 6- Ranging example

Detailed Project Description

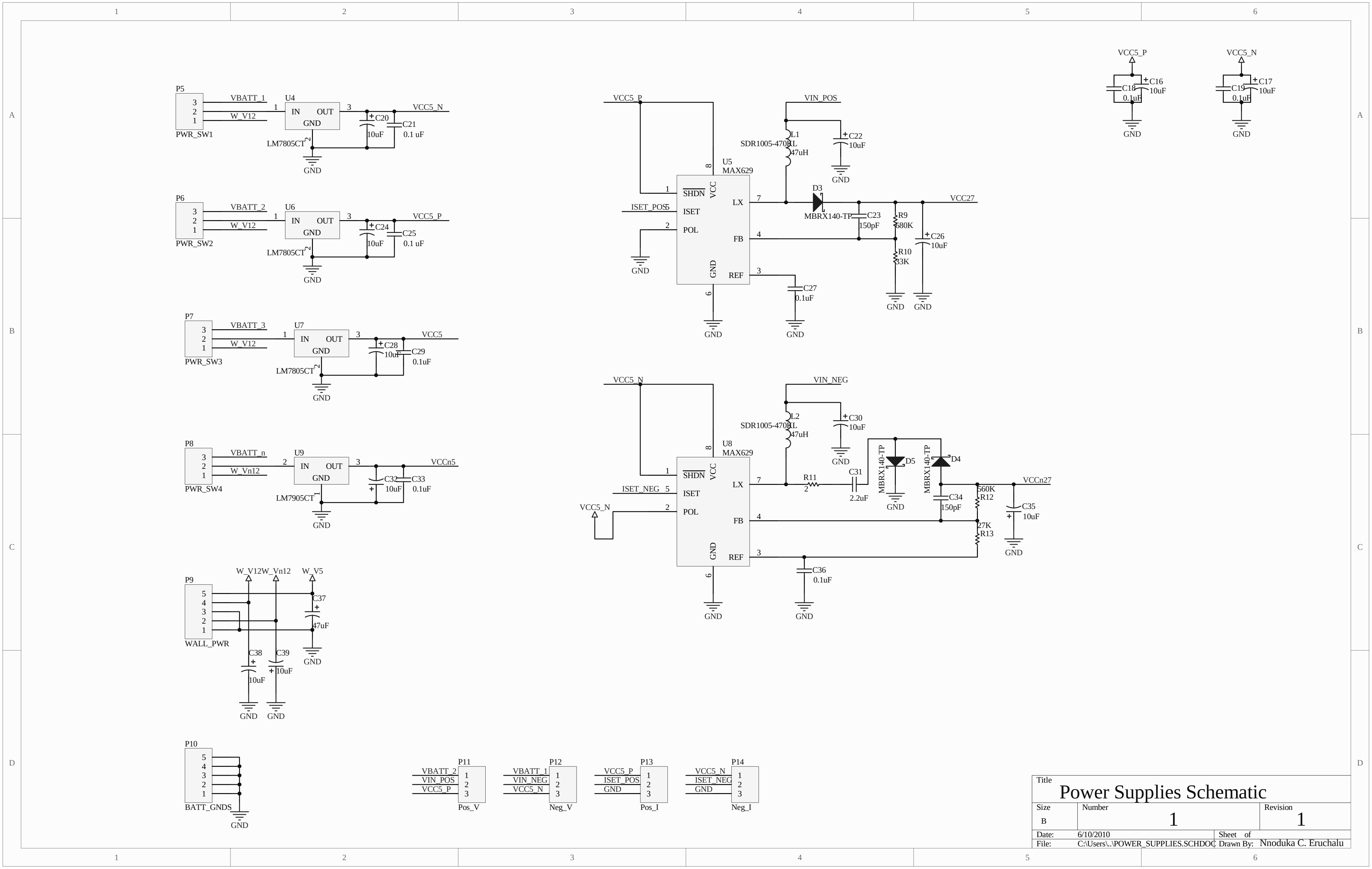
Component Listing

Below is a table showing the Bill of Materials used:

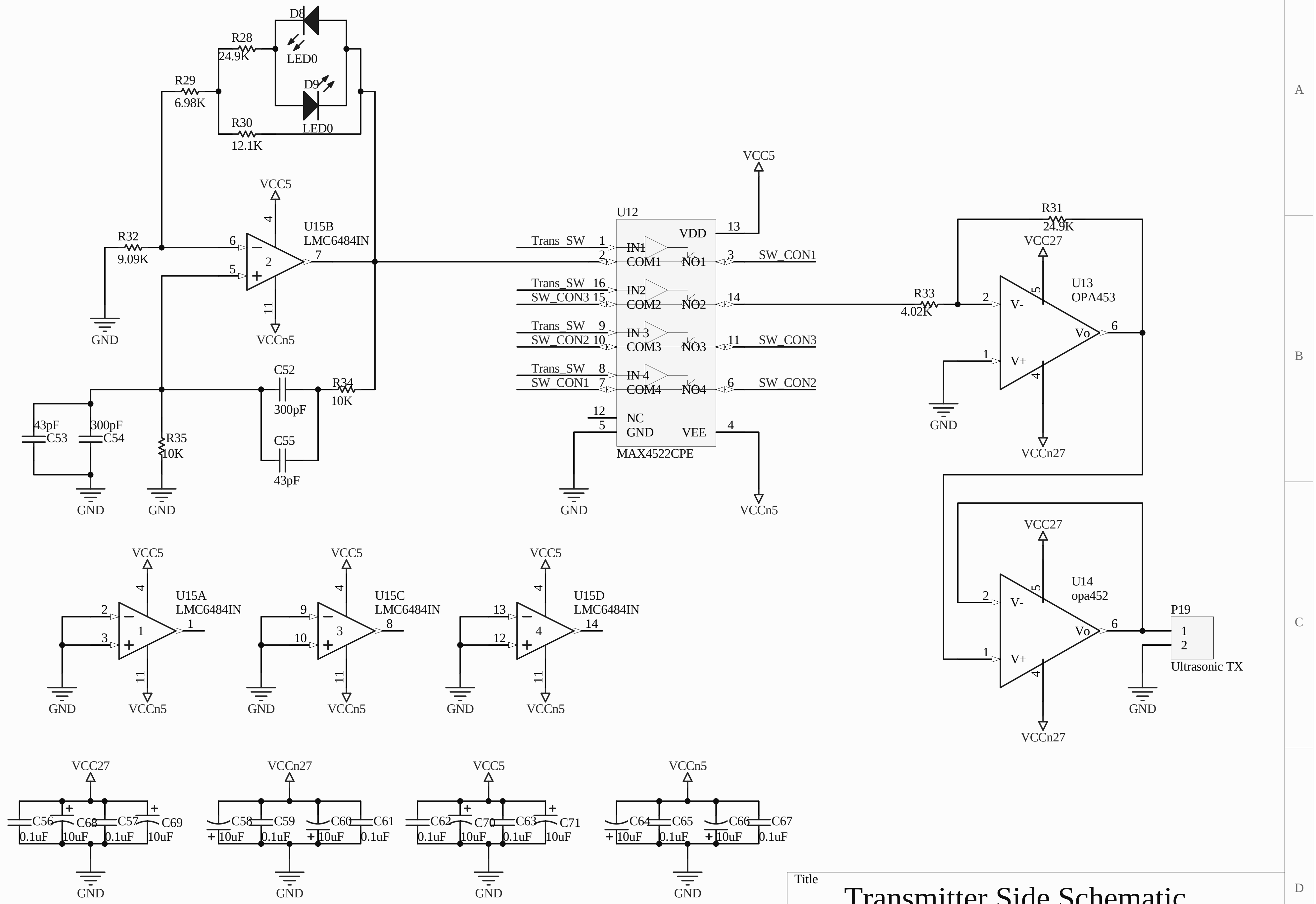
Comment	Description	Designator	Footprint	LibRef	Quantity
22pF	Polarized Capacitor (Radial)	C1, C3	RAD-0.3	Cap Pol1	2
10uF	Polarized Capacitor (Radial)	C2	RAD-0.3	Cap Pol1	1
1uF	Polarized Capacitor (Radial)	C4, C5	RAD-0.3	Cap Pol1	2
10uF	Polarized Capacitor (Radial)	C6, C7, C8, C9, C10, C16, C17, C20, C22, C24, C26, C28, C30, C32, C35, C44, C45, C48, C50, C58, C60, C64, C66, C68, C69, C70, C71	TC7343-2917	Cap Pol1	27
0.1uF	Capacitor	C11, C12, C13, C14, C15, C18, C19, C27, C29, C33, C36, C46, C47, C49, C51, C56, C57, C59, C61, C62, C63, C65, C67	C0805	Cap	23
0.1 uF	Capacitor	C21, C25	C0805	Cap	2
150pF	Capacitor	C23, C34	C0805	Cap	2
2.2uF	Capacitor	C31	C1210	Cap	1
47uF	Polarized Capacitor (Radial)	C37	RB.2/.4X	Cap Pol1	1
10uF	Polarized Capacitor (Radial)	C38, C39	RB.1/.2	Cap Pol1	2
2nF	Capacitor	C40, C41, C42, C43	RAD-0.4	Cap	4
330pF	Capacitor	C52, C54	RAD-0.3	Cap	2
5pF	Capacitor	C53, C55	RAD-0.3	Cap	2
Diode 1N4148	1 Amp General Purpose Rectifier	D1, D2	DIO7.4-4x2	Diode 1N4007	2
MBRX140-TP	Schottky Rectifier	D3, D4, D5	SOD123	Diode 10TQ035	3
1N5819RL	Schottky Rectifier	D6, D7	DIO10.46-5.3x2.8	Diode 10TQ035	2
LED0	Typical INFRARED GaAs LED	D8, D9	RAD-0.2	LED0	2
SDR1005-470KL	Inductor	L1, L2	SDR1005_v2	Inductor	2
Pickit2 connector	Header, 6-Pin	P1	HDR1X6	Header 6	1
LCD connector	Header, 17-Pin	P2	HDR1X17	Header 17	1
P_Meter_V	Header, 2-Pin	P3	HDR1X2	Header 2	1
P_Meter_I	Header, 2-Pin	P4	HDR1X2	Header 2	1
PWR_SW1	Header, 3-Pin	P5	HDR1X3	Header 3	1
PWR_SW2	Header, 3-Pin	P6	HDR1X3	Header 3	1
PWR_SW3	Header, 3-Pin	P7	HDR1X3	Header 3	1
PWR_SW4	Header, 3-Pin	P8	HDR1X3	Header 3	1
WALL_PWR	Header, 5-Pin	P9	DIN5	Header 5	1
BATT_GNDS	Header, 5-Pin	P10	HDR1X5	Header 5	1
Pos_V	Header, 3-Pin	P11	HDR1X3	Header 3	1

Neg_V	Header, 3-Pin	P12	HDR1X3	Header 3	1
Pos_I	Header, 3-Pin	P13	HDR1X3	Header 3	1
Neg_I	Header, 3-Pin	P14	HDR1X3	Header 3	1
NEAR_SEL	Header, 3-Pin	P15	HDR1X3	Header 3	1
NEAR Rx	Header, 2-Pin	P16	HDR1X2	Header 2	1
RX_MODE_SW	Header, 3-Pin, Dual row	P17	HDR2X3	Header 3X2	1
FAR Rx	Header, 2-Pin	P18	HDR1X2	Header 2	1
Ultrasonic TX	Header, 2-Pin	P19	HDR1X2	Header 2	1
10K	Resistor	R1, R2, R3, R4, R5, R6, R7, R8, R34, R35	AXIAL-0.4	Res1	10
700K	Resistor	R9	RESC3216N	Res1	1
36K	Resistor	R10, R13	RESC3216N	Res1	2
2	Resistor	R11	C0805	Res1	1
600K	Resistor	R12	RESC3216N	Res1	1
40.2K	Resistor	R14, R15, R16	AXIAL-0.4	Res1	3
6.98K	Resistor	R17, R29	AXIAL-0.4	Res1	2
4.02K	Resistor	R18, R19, R20, R21	AXIAL-0.4	Res1	4
2K	Resistor	R22, R25	AXIAL-0.4	Res1	2
402K	Resistor	R23, R26	AXIAL-0.4	Res1	2
499K	Resistor	R24, R27	AXIAL-0.4	Res1	2
24.9K	Resistor	R28	AXIAL-0.4	Res1	1
12.1K	Resistor	R30	AXIAL-0.6	Res1	1
24.9K	Resistor	R31	AXIAL-0.6	Res1	1
9.09K	Resistor	R32	AXIAL-0.4	Res1	1
4.02K	Resistor	R33	AXIAL-0.6	Res1	1
SW-PB RESET	Switch	S1	HDR1X2	SW-PB	1
CALCULATE	Switch	S2	HDR1X2	SW-PB	1
EXTRA_BUTTON	Switch	S3	HDR1X2	SW-PB	1
PIC18F4520-I/P	Enhanced Flash Microcontroller with 10-Bit A/D and nanoWatt Technology, 32K Flash, 40-Pin PDIP, Industrial Temperature Range	U1	PDIP600-P40	PIC18F4520-I/P	1
DAC0830LCN	8-Bit µP-Compatible, Double-Buffered Digital-to-Analog Converter	U2	N20A	DAC0830LCN	1
LMC6484IN	CMOS Quad Rail-to-Rail Input & Output Operational Amplifier	U3, U10, U15	N14A	LMC6484IN	3
LM7805CT	Series 3-Terminal Positive Regulator	U4, U6, U7	T03B	LM7805CT	3
MAX629	DC/DC Converter	U5, U8	SOIC8	MAX629	2
LM7905CT	3-Terminal Negative Regulator	U9	T03B	LM7905CT	1
LM393N	Low-Power Low Offset Voltage Dual Comparator	U11	N08E	LM393N	1
MAX4522CPE	Quad, Low-Voltage, SPST Analog Switch	U12	PE16A	MAX4522CPE	1
OPA453	Power Op Amp	U13	DFM-T7/X1.6V	opa452	1

OPA452	Power Op Amp	U14	DFM-T7/X1.6V	opa452	1
XTAL-20MHz	Crystal Oscillator	Y1	BCY-W2/E4.7	XTAL	1



Title			
Power Supplies Schematic			
Size B	Number 1		Revision 1
Date:	6/10/2010	Sheet of	
File:	C:\Users\..\POWER_SUPPLIES.SCHDOC	Drawn By:	Nnoduka C. Eruchalu



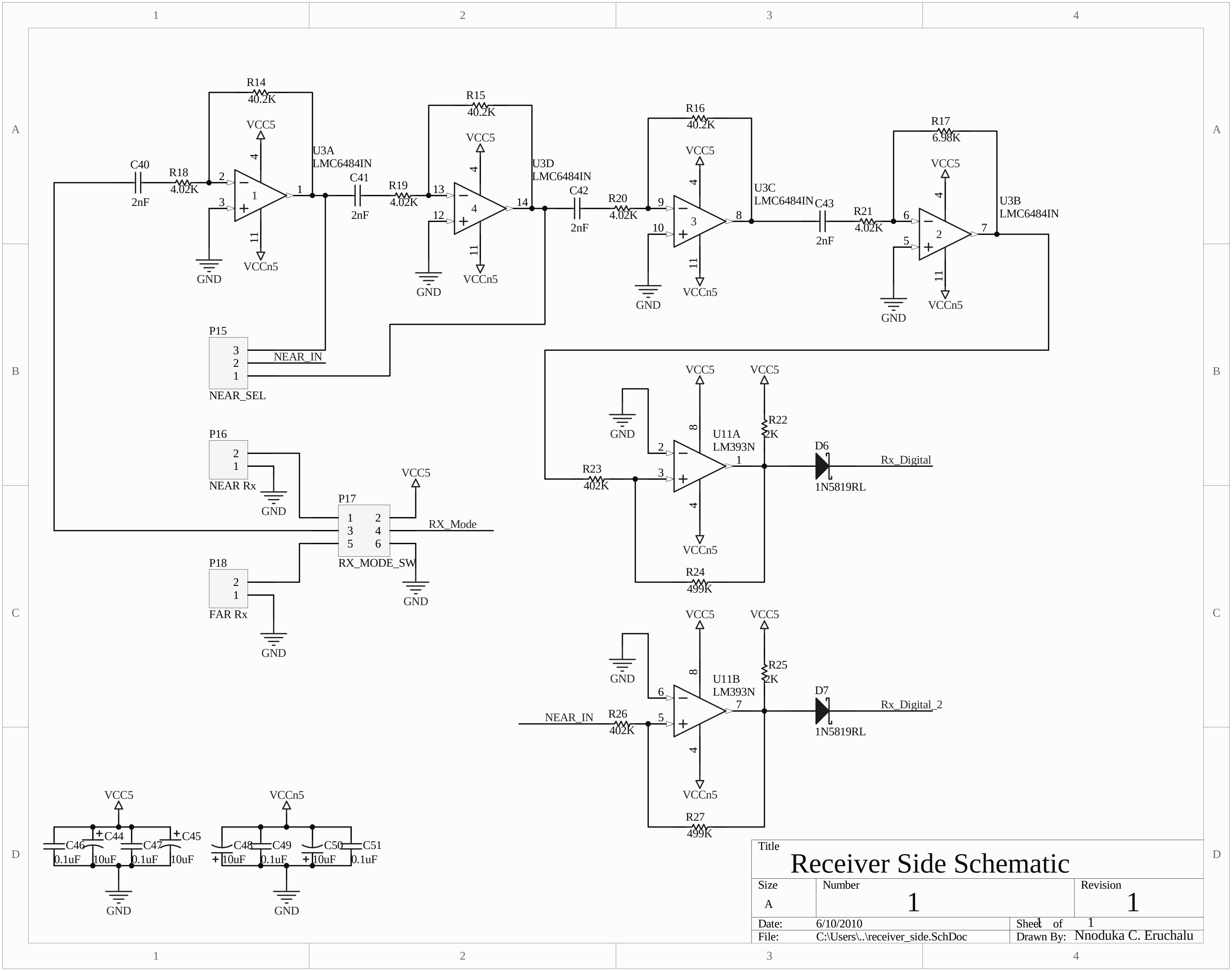
Title

Transmitter Side Schematic

Size	Number	Revision
A	1	1

Date: 6/10/2010 Nnoduka C. Eruchalu Sheet 1 of 1

File: C:\Users\..\TRANSMITTER_SIDE.SCHDOC Drawn By:

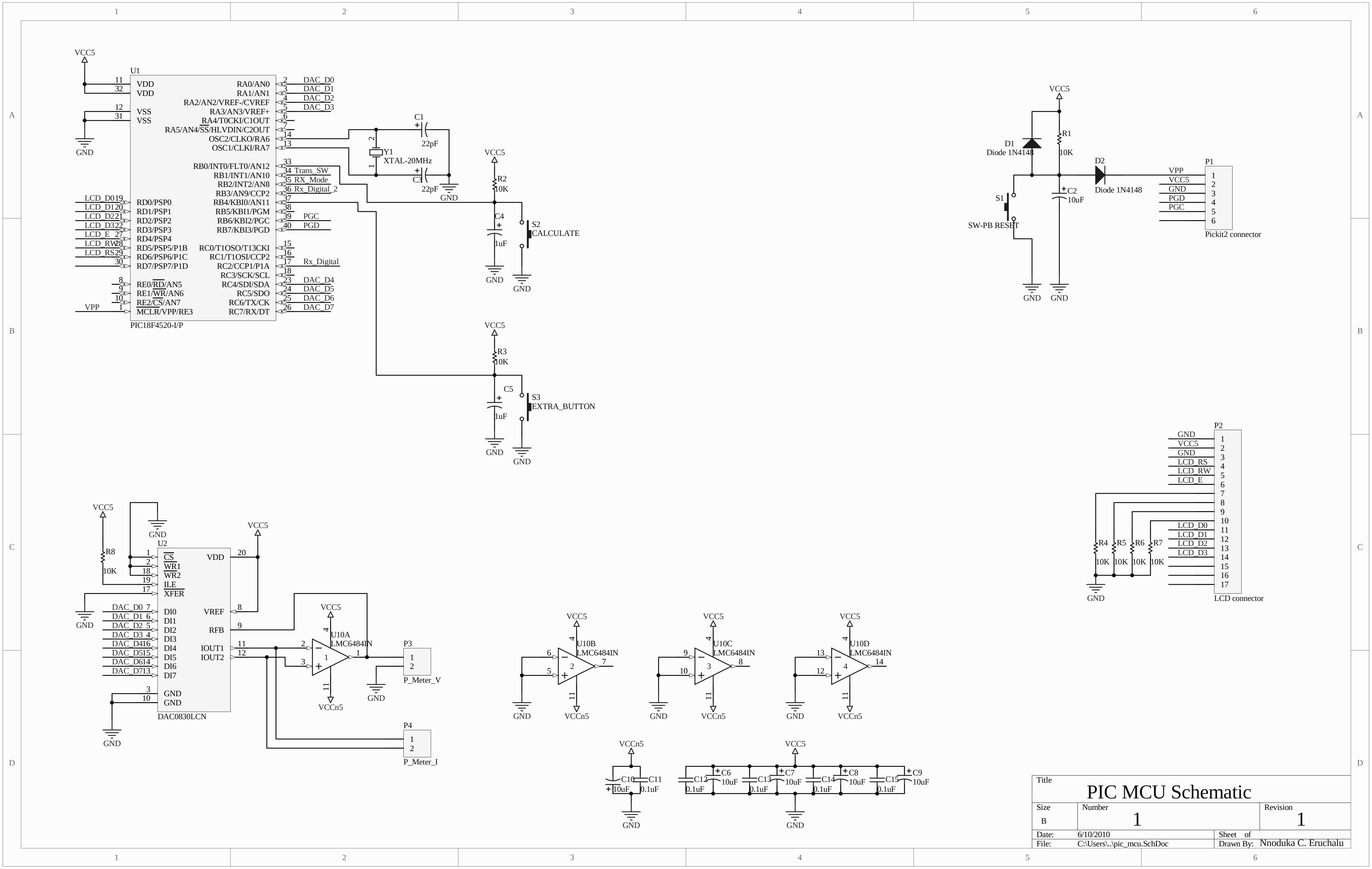


Title

Receiver Side Schematic

Size	Number	Revision
A	1	1

Date: 6/10/2010
Sheet 1 of 1
File: C:\Users\..\receiver_side.SchDoc
Drawn By: Nnoduka C. Eruchalu



Title		
PIC MCU Schematic		
Size B	Number 1	Revision 1
Date: 6/10/2010		Sheet of
File: C:\Users\..\pic_mcu.SchDoc		Drawn By: Nnoduka C. Eruchalu

----

DESCRIPTION OF THE POWER SUPPLIES SCHEMATIC

The power supplies section is made up of:
1. Input Power Sources (Using Battery or Wall Power)
2. Voltage Regulators (Using U4, U6, U7- all LM7805s and U9- LM7905)
3. Boost Converter (Using U5, MAX629)
4. Inverting Converter (Using U8, MAX629)

Description of the Input Power Sources:

The Input Power Sources could be two 9V batteries, or a wall supply.

The option to use 9V batteries is a matter of portability and the option to use a wall supply is also provided incase the user prefers that or does not have batteries at hand.

The user is presented with a sequence of switches which have to be turned on in a certain sequence. If the user toggles them all in the down direction he will be using battery power and if the user toggles them all in the upwards direction he will be using wall power.

The power to the system is sequenced because during testing it was discovered that for proper system operation the Inverting Converter(SW1) has to come on first followed by the Boost Converter's power and the General Positive power lines on the rest of the circuit (SW2). The last switch to be turned on is that for the General Negative power lines on the rest of the circuit (SW3).

This sequence makes sense as we would want to turn on the power op amps before sending in signals into them. The power op amps are powered by the Boost Converter and the Inverter Converter

NOTE that the power switches are turned off in a reverse direction from the turn-on sequence. So the SYSTEM OFF sequence is: SW3 to SW2 to SW1.

The SYSTEM OFF state is when all 3 power switches are in a horizontal position.

Description of the Voltage Regulators:

There are three 5V voltage regulators. One of those is used to power the Boost Converter, another is used to power the Inverter Converter, and the third is used as the general 5V supply for the rest of the Rangefinder circuit.

The input to these 3 voltage regulators is either one of the 9V batteries or +12V from the wall (depending on toggle switch position).

There is also a -5V voltage regulator which is only used for the general -5V supply for the rest of the Rangefinder circuit.

The input to this voltage regulator is the second 9V battery hooked up with its terminals reversed so as to generate -9V, or -12V from the wall (depending on the toggle switch location).

Description of the Boost Converter:

This is a DC/DC Voltage Converter which takes a 5V voltage input and generates a 27V voltage output.

The MAX629 chip being used is a low-power DC-DC converter which features an internal N-channel MOSFET switch and programmable current limiting. It has a

high switching frequency (up to 300kHz) which required that this be placed on a PCB with tiny surface-mount components for proper operation. Also it should be noted that on the actual PCB the placement of the components of this circuit was of extreme importance. Good placement is essential to proper operation.
The MAX629 (U5) pins and the reasons for their current connections are explained below:

/SHDN -Active-Low Shutdown Input. This is pulled high because I always want to generate +27V when the system is turned on.

POL -This is set to GND because this chip will ONLY be used for positive voltage generation

REF -This is the 1.25V reference output that only needs to be bypassed with a 0.1uF capacitor.

FB -This is the feedback input for setting output voltage. This is connected to an external voltage divider which sets the output voltage. The formula for setting the output voltage in this positive voltage configuration circuit is:
Vout = [(R9/R10) + 1]* Vref = (680K/33K + 1) * 1.25 = 27V

ISET -This is connected to VCC5 for a 500mA LX current limit. This high current limit is needed because of the large current draw by the system.

GND -This is connected to the system Ground. Note the use of the starfish connection and not a single ground plane. This helps with isolation of different components on the system. It is important that separate components be isolated.

LX - Internal N-Channel DMOS Switch Drain.

VCC - Power-Supply Input which is connected to a regulated 5V.

Note the use of bypass capacitors in the circuit. The most important being the output bypass capacitor had to be of a low ESR so as to minimize the output ripple. So to keep things safe, all capacitors were chosen to be of as low ESR as was practical for this project.
Also it should be noted that the resistors used were of 1% tolerance. Precision was of extreme importance as the power section of this project was indeed the most challenging.

Description of the Inverting Converter:

This is a DC/DC Voltage Converter which takes a 5V voltage input and generates a -27V voltage output.
The MAX629 chip being used is a low-power DC-DC converter which features an internal N-channel MOSFET switch and programmable current limiting. It has a high switching frequency (up to 300kHz) which required that this be placed on a PCB with tiny surface-mount components for proper operation. Also it should be noted that on the actual PCB the placement of the components of this circuit was of extreme importance. Good placement is essential to proper operation.

The MAX629 (U8) pins and the reasons for their current connections are explained below:

/SHDN -Active-Low Shutdown Input. This is pulled high because I always want to generate -27V when the system is turned on.

POL -This is set to VCC5 because this chip will ONLY be used for negative voltage generation

REF -This is the 1.25V reference output that needs to be bypassed with a 0.1uF capacitor while also connected to the lower potential end of the voltage divider used for setting output voltage.

FB -This is the feedback input for setting output voltage. This is connected to an external voltage divider which sets the output voltage. The formula for setting the output voltage in this negative voltage configuration circuit is:
|Vout| = (R12/R13)* Vref = (560K/27K) * 1.25 = 26V which comes out to 27V during implementation

ISET -This is connected to VCC5 for a 500mA LX current limit. This high current limit is needed because of the large current draw by the system.

GND -This is connected to the system Ground. Note the use of the starfish connection and not a single ground plane. This helps with isolation of different components on the system. It is important that separate components be isolated.

LX - Internal N-Channel DMOS Switch Drain.

VCC - Power-Supply Input which is connected to a regulated 5V.

Note the use of bypass capacitors in the circuit. The most important being the output bypass capacitor had to be of a low ESR so as to minimize the output ripple. So to keep things safe, all capacitors were chosen to be of as low ESR as was practical for this project.
Also it should be noted that the resistors used were of 1% tolerance. Precision was of extreme importance as the power section of this project was indeed the most challenging.

DESCRIPTION OF THE TRANSMITTER SIDE SCHEMATIC

The transmitter side is made up of:
1. A Wien Bridge Oscillator (Using U15, LMC6484)
2. An Analog Switch (Using U12, MAX4522)
3. An inverting amplifier (Using U13, OPA453)
4. A voltage follower (Using U14, OPA452)
5. Ultrasonic Transmitting Transducer

Description of the Wien Bridge Oscillator:

This is an electronic oscillator used to generate the sine wave needed to drive the Ultrasonic Tx Transducer.

The amplitude of the sine wave is 4.0V. Thus the sine wave is 8VVpp(peak to peak). This value was chosen so as to be able to use an analog switch power supplies +/- 5V. This way there would be no clipping of the sine wave after passing through the analog switch chip.

The frequency of the sine wave is 40KHz.

The frequency of Oscillation is given by f= 1/(2PiRC)
Where in this case R =R34 =R35 =10KOhms and C =C52||C55 =C53||C54 =343pF.

The calculated f = 46.4KHz but with pcb capacitances this comes down to 40KHz (as desired) when implemented.

Also during design, D1 and D2 were 1N4747 diodes. However during implementation I observed that I wasnt getting any oscillations unless I touched both diodes thus changing the circuit capacitances.
This led me to testing out using regular LEDs and with those I finally got oscillations.

From experience it was discovered that while it is relatively well known how to set the oscillator frequency, the oscillator amplitude calculations were relatively hard to understand.
However with enough research I was able to control and set my amplitude to fall within a desired range.
For more information on how I was able to control and set my amplitude refer to:
http://cnx.org/content/m32489/latest/

Description of the Analog Switch:
This is a MAX4522 which is a quad-analog switch chip. This is controlled by the PIC MCU which decides whether this should ouput the input sine wave or no signal 0V.
The Power lines of this chip are +/-5V which is why the Sine Wave oscillator was designed to be of amplitude 4V, to prevent clipping.
This chip makes it possible to pulse the Sine Wave.

During testing it was noted that using just one of the analog switches, the

"OFF" state really had a ripple of a 0.1Vpp
So to get rid of this ripple all 4 analog switches are cascaded and controlled by the same PIC MCU input TRANS_SW. The turn-on time of this chip is very low so the small delays have no effect on system operation.

Description of the Inverting Amplfier:

This is an amplifier circuit needed to convert the Sine Wave of amplitude 4V to a Sine wave of amplitude 26V (which comes to a Vrms of approximately 18.4V, well below the Transducer maximum specification of 20V).
The reason for wanting a Vrms of 18.4V is because this is close to the maximum accepted by the Ultrasonic Tx transducer to be used in the project, and this works with the somewhat fragile boost converter circuit.

By sending a sine wave of maximal Vrms I will be getting closer to my goal of transmitting a sine wave of maximal power. However this goal also requires that the current sent into the ultrasonic transducer be substantial hence the need for a Voltage Follower Section.

The OPA453 used for 136 does the actual signal amplification using a an inverting amplifier configuration with gain = -27K/4.02K = -6.7

This gives an output sine wave of 26V amplitude (52Vpp)

Description of the Voltage Follower:

It was stated in the last section that to drive the Tx Transducer with maximal power it is essential that the Transducer can source as much current as it needs from the Sine Wave supply circuit. Thus the need for this Voltage Follower using the unity gain stable OPA452, U14 which can deliver ouput currents of up to 50mA.

The OPA452 used for U14 is configured to serve as a unity gain buffer amplifier which can drive the Tx transducer.
This works because the input-impedance of the OPA452 is high and as a result this stage does not load down the last amplifier stage and does not draw any current from it. The ouput impedance of this stage is low and so it drives the Tx transducer as a good voltage source.

Also this serves the purpose of isolating the Amplifier Section from the Ultrasonic Transmitter which helps a good deal in battling the possible capacitive loading effects of the Tx Transducer on the system.

Description of the Ultrasonic Transmitting Transducer:
This is supplied a High Power Sine wave so as to help in the overall system design goal of long range detection.

DESCRIPTION OF THE RECEIVER SIDE SCHEMATIC

The receiver side is made up of:
1. Ultrasonic Receiving Transducers
2. Amplification Stages (Active High-Pass Filters) (Using U3, LMC6484)
3. Schmitt Triggers (Using U11, LM393)
4. Half-Wave Recitifiers (Using D6 and D7, 1N5819RL)

Description of the Ultrasonic Receiving Transducers:
There are two ultrasonic receiving transducers. One is for long range measurements and another is for short range measurements.

Ideally the user only attempts to put the Rangefinder in short range mode when the user is actually within the defined limits of what constitutes a "short range" (0 to 40cm).

This detection of nearer vs farther objects is simply a matter of taking certain outputs of this 4-gain stages setup. The higher the gain stage the farther the distance that can be detected. This comes with more noise and so the Rx tranducer will have to be placed further away from the TX transducer to compensate for this noise increase.

The short range Rx transducer is placed very close to the Tx transducer so as to be able to detect nearer objects (downt to 3cm!)
On the other hand the long range Rx transducer is placed a sizeable distance from the Tx transducer so as to be able to detect farther objects (over 8m!)

The DPDT switch RX_MODE_SW(P17) is used to decide which Rx Transducer's output will be fed into the Amplification Stages.
This switch also tells the PIC MCU which mode the system is in so that the PIC can control the system appropriately.

Description of the Amplification Stages:
This is composed of 4 cascaded Active High-Pass Filters made using a quad OP AMP chip, LMC6484, U3.
They all have the same cut-off frequency of: fc = 1/[2pi(R18)(C40)] = 19.8KHz.
This guarantees that low frequency noise is eliminated.
There was no need for designing actual Band Pass Filters because the Rx Transducers already have those.

However these high pass filters were just to throw in a little extra noise reduction and were relatively easy to implement considering that the OP AMPs were already in inverting amplifier configurations.

The first 3 amplifier stages have fixed gains of (-10), while the final amplifier amplifier stage has a gain of -1.74.
These provide for a total gain of 1740.

The output of the second gain stage (at a total gain of 100) is fed into a separate Schmitt Trigger from that which receives the output from the final gain stage. No need to consider selecting the ouput of the third gain stage.

At a gain of over 100, the system is detecting over a meter and that is by no means a "low range measurement".

Description of Schmitt Triggers:
This stage comprises of two Schmitt Triggers which generate square wave pulses that will be passed unto the Half-Wave Rectifiers before being sent into the PIC microcontroller. Thes pulses will be triggers for the PIC indicating that the echo of the Transmitted Sine Wave has been received.

So the ideal situation will be to use comparators and when the amplified voltage output of the Rx Transducers is greater than 0, output 5V (a logic 1), else output 0V (a logic 0). However this approach assumes noise of 0Vpp which is unrealistic.

So to combat this noise a Schmitt Trigger is used.
A Schmitt Trigger is a comparator circuit that incorporates positive feedback.
When a Schmitt Trigger is in the NON-INVERTING configuration, when the input is higher than a certain chosen threshold, the output is high; when the input is below a different (lower) chosen threshold, the output is low; when the input is between the two, the output retains its value.

These Schmitt Triggers are implemented in their simplest form using an LM393, U11 (a dual comparator chip)
This depends on railing to function.
The values of the thresholds are given by:
+/-(R23/R24)*VCC5 = +/-(R26/R27)*VCC5 = +/- 4V
R22 and R25 are used to ensure that the output state is at a 1 by default.0
R24 >> R22 and R27 >> R25 is a requirement!

The outputs of this stage are square waveforms with rails +/-5V so these have to go through Half-Wave Rectifiers before going to the PIC MCU.

Description of Half-Wave Recitifiers:
This stage is just two Schottky Diodes which Half-Wave Rectify the two Schmitt Trigger outputs. Schottky Diodes are used because they have low forward voltages and as such can generate appropriate digital logic 0s(0V) and 1s(5V) needed by the PIC microcontroller.
The outputs of this stage serve as triggers for the PIC that indicate the receipt of a transmitted sine wave pulse.

DESCRIPTION OF THE PIC MICROCONTROLLER UNIT SCHEMATIC

The pic microcontroller unit is made up of:
1. PIC microntroller (Using U1, PIC18F4520)
2. System Control Interface (Switches)
3. LCD Display
4. Analog Panel Meter Display (unimplemented)

Description of the PIC microntroller:
This schematic shows how the PIC microcontroller is setup for in-circuit serial programming (icsp) and for general use in this project.
Diode D6 stops high Vpp causing contention.

The icsp is done using a pickit2.

The pins on the pickit2 connector (in order from 1 to 6) are:

```
VPP     - Programming voltage (using 13V)
VCC5    - 5V power supply
GND     - Ground
PGD     - Data
PGC     - Clock
PGM     - LVP enable (left unconnected)
```

Note that PGC and PGD are left as dedicated programming pins so as to not interfere with the rest of the circuit during in-circuit programming

The PIC microcontroller is used to control the LCD by writing to the DataBus and the Control signals, and it receives user instructions via the switches.

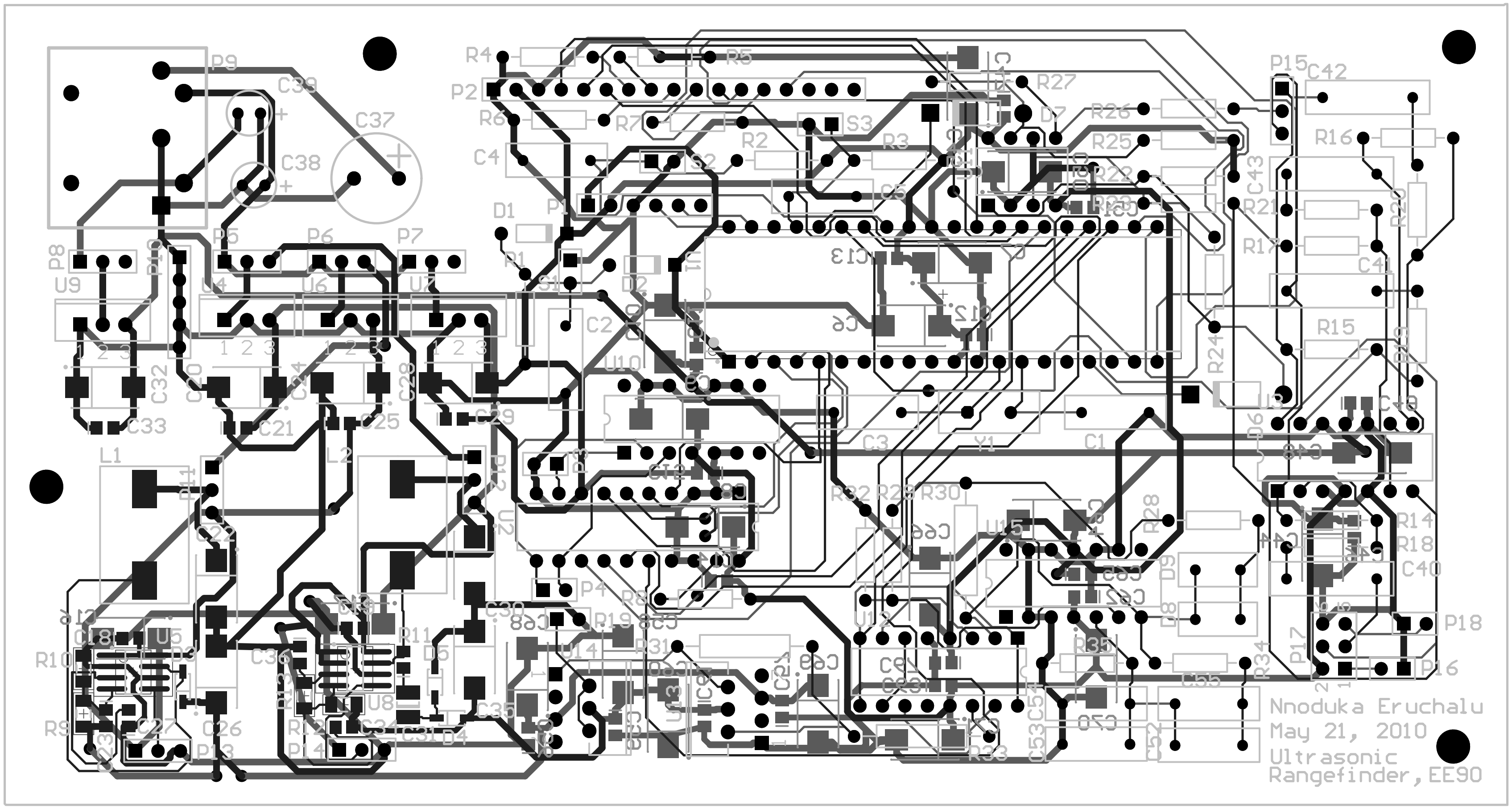
Description of the System Control Interface (Switches):
There are 3 Push-button Switches used in controlling the system's UI.

```
CALCULATE    -  calculate the distance of object the rangefinder is pointed at
CHANGE UNITS -  change the units of the currently displayed distance, which is
                the last distance calculated. This button can only toggle
                between "cm" and "inches"
RESET        -  reset the system and restart the PIC with the introductory
                messages being displayed again.
```

Description of the LCD interface:
The LCD interface is relatively straightforward.
The LCD databus is all of Port D. This implementation uses a 4-bit databus.
Thus the unused 4-bits of the databus are pulled low via 10KOhm resistors. The reason for pulling these lines low is so as to prevent them from being floating inputs and potentially leading to a sharp current spike. The necessity of the 10KOhm resistor comes about because these unused databus lines could also be outputs and we do not want to be shorting a digital logic "1" to ground.

Also the choice of a 4-bit databus helps minimize the number of wires with ever-changing data bits. This surely helps in reducing sustem noise.

```
--------------------------------------------------------------------------------
```



Nnoduka Eruchalu
May 21, 2010
Ultrasonic
Rangefinder, EE90
P9
C39
C37
C38
P8
P10
P5
P6
P7
U9
U4
U6
U7
C32
C33
C20
C21
C24
C25
C28
C29
L1
L2
R14
C22
C16
C18
U5
R10
R9
C23
C27
C26
P13
C36
R11
D5
C30
C35
C34
C31
D4
U8
P14
R12
R4
R5
P2
R6
R7
C4
D1
P1
S1
D2
C2
U10
S2
S3
R2
R3
C5
C13
C6
C12
U1
C3
Y1
C1
R27
D7
R26
R25
R22
R23
C10
C11
R32
R29
R30
C66
U15
R28
D9
D8
C62
C63
R35
C55
U12
C60
C54
C53
C52
R33
P4
R8
R19
U14
R31
C68
C69
U13
P15
C42
R16
C43
R20
R21
R17
C41
R15
R18
R24
U2
U16
C46
C48
R14
R18
C40
C44
P18
P16
P17
R34

```
/*
 * ----------------------------------------------------------------------------
 * -----                           RANGEF.C                               -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is the main file for the EE90 RangeFinder Project.
 *  The theory of operation behind this rangefinder is sending a pulse and
 *  measuring the time it takes for an echo to return.
 *
 * Assumptions:
 *  None
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Jun. 2, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>
#include <pic18.h>
#include <stdio.h>
#include "lcd.h"
#include "adc.h"
#include "ccp.h"
#include <math.h>       /*sqrt() needs this*/

/* chip settings
 *  for more info on these, look at section 23.1 (Configuration Bits) of the
 *  PIC184520 datasheet.
 *  As of today, this datasheet can be found here:
 *      http://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf
 */

__CONFIG(1,0x0200);
__CONFIG(2,0X1E1F);
__CONFIG(3,0X8000);         /* CCP2 input/output is multiplexed with RB3    */
__CONFIG(4,0X00C1);
__CONFIG(5,0XC00F);

#define CALCULATE RB0       /* define the calculate button's pinout         */
#define TRANS_SW LATB1      /* define the transmit switch's LAT pinout      */
#define NUM_PULSES 40       /* number of sine wave pulses to transmit       */
#define WAIT_PULSES 40      /* measuring wait time in terms of pulses       */
#define NUM_PULSES_S 2      /* for short range mode                         */
#define WAIT_PULSES_S 2
#define SAMPLE_SIZE 10      /* number of sample measurements to take        */
#define MIN_LEVEL 483       /* the minimum detectable CCPR value            */
#define NO_ECHO_LIMIT 5     /* how many non-echoes does it take to stop?    */
#define HALF_TRANS_SPC 42   /* square of half the dist b/w transducers (cm) */

volatile double distance = 0;           /* last measured distance           */
volatile unsigned int temp = 0;         /* temporary storage variable       */
volatile unsigned int cur_max = 0;      /* current max measured distance    */
volatile unsigned int num_no_echo = 0;  /* number of echoless transmissions */
volatile unsigned int wait_4_ccp = 0;   /* flag to indicate waiting for echo*/
volatile unsigned int rec_ccp = 0;      /* flag to indicate received signal */
volatile int i = 0;                     /* index into samples array         */
volatile int update = 0;                /* flag to update displayed distance*/
volatile int rx_mode = 0;               /* indicates receiver mode          */
volatile int units = 0;                 /*0 = cm, 1 = inches*/
volatile int unit_mode = 0;
volatile int unit_pend = 0;

/* simple delay routine */
void wait(unsigned int delay)
{
    for(;delay;delay--)
        __delay_us(100);
}


void main(void)
{
    unsigned int n;
    /* Let the Module start up */
    wait(100);

    /*Setup for TRANS_SW*/
    TRISB1 = 0;     /* always an output                                     */
    TRANS_SW = 0;   /* dont want to be transmitting a sine wave just yet    */

    /* Initialize the LCD Module */
    lcd_init();
    /* Initialize the CCP module */
    ccp_init();

    /* Setup the Receiver mode switch */
   TRISB |= 0x04;   /* B2 is always and input               */

   /* setup the units change button */
    TRISB4 = 1;     /* always an input                      */

    /* setup interrupts for Calculate button press          */
    TRISB0 = 1;     /* always an input                      */
    GIEH = 1;       /* enable interrupts                    */
    GIEL = 1;
    INT0IF = 0;     /* will be using INT0 for calculate     */
    INT0IE = 1;
    INTEDG0 = 0;    /* a switch, so trigger on falling edge */


    /* Clear the Module */
    lcd_clear();

    /* A Welcome message */
    lcd_write_str("Rangefinder!");
    lcd_cursor(40);     /* Then write on line 2 */
    lcd_write_str("By...");
    wait(1000);

    /* Display My name on a New Screen */
    lcd_clear();
    lcd_write_str("Nnoduka C.");    /* Have a long name so  */
    lcd_cursor(40);                 /* split over 2 lines   */
    lcd_write_str("Eruchalu");
    wait(1000);

    /* Give usage instructions */
    lcd_clear();
    lcd_write_str("Hit CALCULATE ");
    lcd_cursor(40);
    lcd_write_str("to range...");
    wait(1000);

    /* Always Display the current range */
    while(1)
    {
        unit_mode = PORTB;
        unit_mode &= 0x10;
        if ((unit_mode == 0) && (unit_pend == 0)) /* a units change has been */
        {                                   /* requested and none is pending*/
            units ^= 0xFF; /*invert the units flag*/
            unit_pend = 1;
            update = 1;
            wait(100);       /* debounce this switch*/
        }

        if (update == 1)    /* only write to the LCD if a distance has been */
        {                   /* calculated                                   */
            GIEH = 0;       /* disable interrupts                           */
            GIEL = 0;
            lcd_clear();    /* write a new distance                         */
            lcd_write_str("Distance is: ");
            lcd_cursor(40);
            if (units == 0) {
            lcd_write_int((int) distance);
            lcd_write_str(" cm");}
            else {
            lcd_write_int ((int) (distance/2.54));
            lcd_write_str(" inches");}
            unit_pend = 0;  /* units have finally been changed              */
            update = 0;     /* no more need for an update                   */
            GIEH = 1;       /* enable interrupts                            */
            GIEL = 1;
        }
        if (wait_4_ccp == 1)
        {
            wait(100);      /* if the program gets into this loop while the */
            if (wait_4_ccp == 1)    /* system is still waiting for an Rx    */
            {                       /* interrupt, then wait for a while to  */
                CCP1IE = 0;         /* give the system a chance to generate */
                wait_4_ccp = 0;     /* the Rx trigger. If after waiting     */
                if (num_no_echo == NO_ECHO_LIMIT)   /* the trigger is yet   */
                {                   /* to occur then disable the Rx INT     */
                    lcd_clear();    /* and check if this has occured at the */
                    lcd_write_str("Can't Detect");  /* max acceptable count */
                    num_no_echo = 0;
                }
                else                /* if the echoless limit has not been   */
                {                   /* hit then retransmit the waveform     */
                    num_no_echo++;
                    rx_mode = PORTB;
                    rx_mode &= 0x04;    /* extract bit info at bit 2, RB2   */

                    if (rx_mode == 0)
                    {
                      TRANS_SW = 1;     /* Pulse the TRANS_SW line          */
                      TMR1 = 0;         /* clear tmr1                       */
                      __delay_us(NUM_PULSES); /* do __delay_us(1) to get a  */
                      TRANS_SW = 0;           /* delay of 25us              */
                      __delay_us(WAIT_PULSES);  /* wait for transmitter to  */
                      INT0IF=0;         /* stop vibrating. Clear the flag   */
                      wait_4_ccp = 1;   /* now waiting on the CCP line      */
                      rec_ccp = 0;
                      CCP1IF = 0;
                      CCP1IE = 1;
                    }
                    else
                    {
                      TRANS_SW = 1;/* Pulse the TRANS_SW line               */
                      TMR1 = 0;   /* clear tmr1                             */
                      __delay_us(NUM_PULSES_S); /* do __delay_us(1) to get  */
                      TRANS_SW = 0;             /* a delay of 25us          */
                      __delay_us(WAIT_PULSES_S);  /* wait for transmitter   */
                      INT0IF=0;       /* to stop vibrating. Clear the flag  */
                      wait_4_ccp = 1; /* now waiting on the CCP line        */
                      rec_ccp = 0;
                      CCP2IF = 0;
                      CCP2IE = 1;
                    }
                }
            }
        }
    }

    /* Loop Forever, shouldnt get here */
    while(1) continue;
}


void interrupt isr(void)
{
    if((CCP1IE)&&(CCP1IF))  /*interrupt from long Range Rx has occured      */
    {
        temp = CCPR1;
        CCP1IE = 0;
        wait_4_ccp = 0;         /* no longer waiting for ccp                */
        rec_ccp = 1;

        if (i < SAMPLE_SIZE)    /* there is a max number of samples to take */
        {
            if (temp > cur_max) /* and only want to retain the max value    */
            {                   /* of all the samples                       */
              cur_max = temp;
            }
            i++;

            TRANS_SW = 1;/* Pulse the TRANS_SW line  so as to get new data  */
            TMR1 = 0;   /* clear tmr1                                       */
            __delay_us(NUM_PULSES);
            TRANS_SW = 0;
            __delay_us(WAIT_PULSES);    /*wait till Tx stops ringing        */
            CCP1IF = 0;     /* you need to enable this interrupt to detect  */
            CCP1IE = 1;     /* the next pulse                               */
        }
        else        /* have all sample data points, so now get the distance */
        {
            distance = ((double) cur_max) /2500 * 170;  /*convert to cm     */
            distance = distance*distance - HALF_TRANS_SPC;/*use pythagoras' */
            distance = sqrt(distance);/*theorem due to transducer spacing   */
            distance = distance - 2;  /*system delays during long range mode*/
            i = 0;                    /* reset index                        */
            cur_max = 0;              /* reset cur_max                      */
            update = 1;               /* a new calculation has been done    */
        }
        CCP1IF=0;                     /* Clear the flag                     */
    }


    if((CCP2IE)&&(CCP2IF))      /*interrupt from short range CCP has occured*/
    {
        temp = CCPR2;
        CCP2IE = 0;
        wait_4_ccp = 0;         /* no longer waiting for ccp                */
        rec_ccp = 1;

        if (i < SAMPLE_SIZE)
        {
            if (temp > cur_max)
            {
              cur_max = temp;
            }
            i++;

            TRANS_SW = 1;       /* Pulse the TRANS_SW line to get new data  */
            TMR1 = 0;           /* clear tmr1                               */
            __delay_us(NUM_PULSES_S);
            TRANS_SW = 0;
            __delay_us(WAIT_PULSES_S);  /*wait for Tx to stop ringing       */
            CCP2IF = 0;     /* you need to enable this interrupt to detect  */
            CCP2IE = 1;     /* the next pulse                               */
        }
        else        /* have all sample data points, so now get the distance */
        {
            distance = ((double) cur_max) /2500 * 170;  /*convert to cm     */
            distance = distance - 4; /*system delays during short range mode*/
            i = 0;                   /* reset index                         */
            cur_max = 0;             /*reset cur_max                        */
            update = 1;              /* a new calculation has been done     */
        }
        CCP2IF=0;                   /* Clear the flag                       */
    }

if((INT0IE)&&(INT0IF))      /* interrupt from CALCULATE has occured         */
    {
        cur_max = 0;        /* reset current maximum distance               */
        i = 0;              /* reset samples' index                         */
        rx_mode = PORTB;
        rx_mode &= 0x04;    /*extract bit info at bit 2, RB2                */

        if (rx_mode == 0)   /* if in long range mode                        */
        {
          TRANS_SW = 1;     /* Pulse the TRANS_SW line                      */
          TMR1 = 0;         /* clear tmr1                                   */
          __delay_us(NUM_PULSES); /* do __delay_us(1) to get a delay of 25us*/
          TRANS_SW = 0;
          __delay_us(WAIT_PULSES);  /*wait for transmitter to stop vibrating*/
          INT0IF=0;       /* Clear the flag                                 */
          wait_4_ccp = 1; /* now waiting on some Rx Trigger                 */
          rec_ccp = 0;
          CCP1IF = 0;
          CCP1IE = 1;
        }
        else                /* if in short range mode                       */
        {
          TRANS_SW = 1;     /* Pulse the TRANS_SW line                      */
          TMR1 = 0;         /* clear tmr1                                   */
          __delay_us(NUM_PULSES_S); /*do __delay_us(1) to get delay of 25us */
          TRANS_SW = 0;
          __delay_us(WAIT_PULSES_S);/* wait for Tx to stop vibrating        */
          INT0IF=0;                 /* Clear the flag                       */
          wait_4_ccp = 1;           /* now waiting on the CCP line          */
          rec_ccp = 0;
          CCP2IF = 0;
          CCP2IE = 1;
        }
    }
}
```

```c
/*
 * -----------------------------------------------------------------------------
 * -----                           LCD.C                                   -----
 * -----                           EE90                                    -----
 * -----------------------------------------------------------------------------
 *
 * File Description:
 *  This is a library of functions for interfacing the PIC18F4520 with a
 *  16 by 2 LCD in its 4-bit databus mode.
 *
 * Table of Contents:
 *  lcd_write       - write a byte to the lcd.
 *  wait_lcd_BF     - wait for LCD BF to clear.
 *  lcd_init        - initialize the LCD
 *  lcd_clear       - clear the LCD and home the cursor.
 *  lcd_write_str   - write a string of chars to the LCD
 *  lcd_write_int   - write an integer to the LCD
 *  lcd_write_db    - write a double to the LCD
 *  lcd_cursor      - move the cursor to a specified location on the LCD.
 *
 * Assumptions:
 *  The code assumes the actual hardware is hooked up as follows:
 *  PORTD bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 21, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>
#include <stdio.h>
#include "lcd.h"



/*
 * lcd_write
 * Description:
 *  This procedure simply writes a byte to the lcd. It does not return
 *  until the LCD's busy flag is cleared.
 *
 * Operation:
 *  Send the high nibble, then send the low nibble.
 *  After that wait for the BF to be cleared.
 *
 * Revision History:
 *  Apr. 22, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_write(unsigned char c)
{
   unsigned char hn, ln, temp;

   hn = c >> 4;         /* put the high nibble of c in the low nibble of hn */
   ln = c & 0x0F;       /* put the low nibble of c in the low nibble of ln */

   /* send high nibble */
   temp = (LCD_DATA_LAT & 0xF0)|(hn);
   LCD_DATA_LAT = temp;
   LCD_STROBE();

   /* send low nibble */
   temp = (LCD_DATA_LAT & 0xF0)|(ln);
   LCD_DATA_LAT = temp;
   LCD_STROBE();

   /*dont exit until the LCD is no longer busy */
   wait_lcd_BF();
}



/*
 * wait_lcd_BF
 * Description:
 *  This procedure simply loops until the LCD is not busy
 *  This clears the RS bit.
 *
 * Operation:
 *  Keep on looping and reading the LCD busy flag. When it indicates that the
 *  LCD is not busy then exit.
 *
 * Revision History:
 *  Apr. 21, 2010       Nnoduka Eruchalu        Initial Revision
 */
void wait_lcd_BF(void)
{
   unsigned char busy, status=0x00, temp;

   /*when reading a port change it to an input*/
   LCD_DATA_TRIS |= 0x0F;

   /* put the LCD in read mode*/
   SET_RW();        /* in read mode when R/W\ is set */
   CLEAR_RS();      /* will read in the value at the LCDbase */

   __delay_us(0.5);   /* letting the RW/RS lines stabilize, tAS on timing diag */

   do
   {
        SET_E();        /* during reads the E has to be active */
        __delay_us(0.5);  /* wait tDA for data to become available */

        status = LCD_DATA_PORT;  /* read in the value at LCDbase */
        status = status << 4;   /* remember that this is a 4bit interface
                                 * so the data comes a nibble at a time */
        __delay_us(0.5);

        CLEAR_E();      /* pull E low effectively pulsing E 4 a nibble read */
```

```c
        __delay_us(1);    /* tEL */

        SET_E();        /* during reads the E has to be active */
        __delay_us(0.5);  /* wait tDA for data to become available */

        temp = LCD_DATA_PORT;   /* read in the 2nd nibble at LCDbase */
        temp &= 0x0F;           /* and mask off the important data */

        status = status|temp;   /* now have the complete status byte */
        busy = status & 0x80;   /* busy flag is the highest status bit */

        __delay_us(0.5);

        CLEAR_E();      /* pull E low effectively pulsing E 4 a nibble read */
        __delay_us(1);    /* tEL */
   } while(busy);

   /* put the LCD in write mode*/
   CLEAR_RW();        /* in write mode when R/W\ is set */

   /* now the lcd data port should be an output again */
   LCD_DATA_TRIS &= 0xF0;
}



/*
 * lcd_init
 * Description:
 *  This initializes the LCD. This must be called before the LCD can be used.
 *  Thus this function has to be called before calling any other LCD-interface
 *  functions.
 *  The LCD is set to the following specifications:
 *      4-bit mode, 2-line display, 5x10 font
 *      cursor INCs, display doesn't shift
 *      cursor visible, cursor blinking
 *
 * Operation:
 *  Really just follows standard LCD initialization sequence.
 *  This function starts by appropriately setting up the LCD's function setting
 *  by writing the appropriate command to the LCDBase. Note that in writing the
 *  function set command, there are required minimum wait times between
 *  successive writes. These are achieved using the delay_ms macro.
 *  After that the function turns the display off. While the display is turned
 *  off, it is cleared and the entry mode is set. These are done by again
 *  writing the appropriate values to the LCDBase.
 *  Then the display is turned on again by writing the appropriate value to
 *  LCDBase.
 *  Before each of the writes from display off to display on again, the function
 *  checks the busy flag and waits till it is cleared to do the write operation.
 *
 * Revision History:
 *  Apr. 21, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_init(void)
```

```
{
    /* after power on wait some time for the LCD to startup*/
    __delay_ms(30);

    /* setup LCD IO ports as outputs */
    LCD_DATA_TRIS &= 0xF0;      /* LCD data bus uses low 4 bits */
    LCD_E_TRIS &= (~(1<<LCD_E_BIT));
    LCD_RW_TRIS &= (~(1<<LCD_RW_BIT));
    LCD_RS_TRIS &= (~(1<<LCD_RS_BIT));

    /* set all outputs low */
    LCD_DATA_LAT &= 0xF0;
    CLEAR_E();
    CLEAR_RS();
    CLEAR_RW();

    /* now initialize the LCD for 4-bit mode */
    __delay_us(0.3);    /* tAS */

    /* write the function set command for an 8-bit interface 3 times */
    LCD_DATA_LAT|=(0b00000011);
    LCD_STROBE();

    __delay_ms(5);
    LCD_STROBE();

    __delay_us(120);
    LCD_STROBE();

    /* wait until Busy Flag is cleared */
    wait_lcd_BF();

    /* now write the function set command to set to 4-bit mode */
    LCD_DATA_LAT &= 0xF0;
    LCD_DATA_LAT|=(0b00000010);
    LCD_STROBE();

    /* wait for the LCD to execute the FunctionSet command */
    wait_lcd_BF();

    /* now the LCD is in 4-bit mode */
    lcd_write(0x28);    /* 4-bit mode, 2-line display, 5x10 font */
    lcd_write(0x08);    /* display off */
    lcd_write(0x01);    /* clear display */
    lcd_write(0x06);    /* entry mode- cursor INCs, display doesn't shift */
    lcd_write(0x0F);    /* display on, cursor visible, cursor blinking */
}


/*
 * lcd_clear
 * Description:
 *  This function clears the LCD and return the cursor to the starting position
 *  This function doesnt return until the LCD's BF is cleared.
 *
 * Operation:
 *  Choose the base register of the LCD then write the clear command to it.
 *
 * Revision History:
 *  Apr. 22, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_clear(void)
{
    CLEAR_RS();
    lcd_write(0x01);
}



/*
 * lcd_write_str
 * Description:
 *  This function writes a string of characters to the LCD
 *  This function doesnt return until the LCD's BF is cleared.
 *
 * Operation:
 *  Choose the data register of the LCD then write each char of the string to
 *  the LCD.
 *
 * Revision History:
 *  Apr. 22, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_write_str(const char *str)
{

    while(*str != '\0')
        {
        SET_RS();       /* wait_lcd_BF clears RS */
        lcd_write(*str++);
        };
}



/*
 * lcd_write_int
 * Description:
 *  This function writes an integer to the lcd
 *  This function doesnt return until the LCD's BF is cleared.
 *
 *
 * Operation:
 *  Convert the integer to a string then write it to the LCD.
 *
 * Revision History:
 *  Apr. 28, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_write_int(unsigned int num)
{
    char buffer[STR_BUF_SIZE];  /*buffer to hold int string*/
    int i;
    sprintf(buffer, "%u", num);
    for(i=0 ; buffer[i]!='\0'; i++)
    {
        SET_RS();       /* wait_lcd_BF clears RS */
        lcd_write(buffer[i]);
    };
}



/*
 * lcd_cursor
 * Description:
 *  This function moves the cursor to a specific position
 *  This function doesnt return until the LCD's BF is cleared.
 *
 * Operation:
 *  Choose the base register of the LCD then update the cursor location.
 *
 * Revision History:
 *  Apr. 22, 2010       Nnoduka Eruchalu        Initial Revision
 */
void lcd_cursor(unsigned char loc)
{
    CLEAR_RS();
    lcd_write(0x80 + loc);
}
```

```c
/*
 * ----------------------------------------------------------------------------
 * -----                            CCP.C                                 -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is a library of functions for interfacing with the PIC18F4520's
 *  CCP1 interface.
 *
 * Table of Contents:
 *  ccp_init        - initialize the CCP module
 *
 * Assumptions:
 *  The code assumes the use of CCP1 (RC2)
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 29, 2010       Nnoduka Eruchalu        Initial Revision
 */


 #include <htc.h>
 #include "ccp.h"



/*
 * ccp_init
 * Description:
 *  This initializes the CCP2 module.
 *
 * Operation:
 *  This first sets up Timer 1 and also has to set up T3CON.
 *  Next it sets up the CCP2 pin (RB3) as an input.
 *  This sets up CCP2CON appropriately and clears CCPR2 so as to reset it to a
 *  state of no time-elapsed. It clears CCP2IF and disables interrupts by
 *  clearing CCP2IE
 *
 * Assumptions:
 *  The configuration bit CC2PMX = 0 so as to multiplex CCP2 with RB3
 *
 * Revision History:
 *  Apr. 29, 2010       Nnoduka Eruchalu        Initial Revision
 */
void ccp_init(void)
{
    T3CON = 0b00000001;     /* Setup T3CON to choose Timer 1 for CCP2 */
    TMR3 = 0x0000;               /* Load initial value into timer3 */
    TMR3ON = 1;

    T1CON = 0b00000001;     /* since using Timer 1, have to set ut up */
    TMR1 = 0;               /* Load initial value into timer1 */
    TMR1ON = 1;

    TRISC |= 0x04;          /* RC2 is IO pin for CCP1 */
    TRISB |= 0x08;          /* RB3 is IO pin for CCP2 */
    GIEH = 1;
    GIEL = 1;



    CCP1CON = 0x04;         /* now actually initialize CCP1 */
    CCPR1 = 0;              /* to trigger on falling edge */
    CCP1IF = 0;             /* use 0x05 for rising edge */
    CCP1IE = 0;             /* use 0x04 for falling edge*/

    CCP2CON = 0x04;         /* now actually initialize CCP2 */
    CCPR2= 0;              /* to trigger on falling edge */
    CCP2IF = 0;             /* use 0x05 for rising edge */
    CCP2IE = 0;             /* use 0x04 for falling edge*/
}
```

```
/*
 * ----------------------------------------------------------------------------
 * -----                            ADC.C                                 -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is a library of functions for interfacing with the PIC18F4520's ADC
 *  interface
 *
 * Table of Contents:
 *  adc_init        - initialize the ADC channel
 *  adc_read        - returns 10bit number indicative of the input analog level
 *
 * Assumptions:
 *  The code assumes the actual hardware is hooked up as follows:
 *  The input analog signal is connected to ADC_CHANNEL which is 0 to 3 only!
 *  The input analog signal is connected at PORTA.
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 28, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>
#include "adc.h"



/*
 * adc_init
 * Description:
 *  This initializes the ADC interface. This must be called before the ADC
 *  interface can be used.
 *  Thus this function has to be called before calling adc_read()
 *
 * Operation:
 *  Really just follows standard ADC initialization sequence.
 *  Check the datasheet.
 *
 * Revision History:
 *  Apr. 28, 2010       Nnoduka Eruchalu        Initial Revision
 */
void adc_init(void)
{
    TRISA |= (1<<ADC_CHANNEL);      /* make the ADC channel an INPUT */

    /* CONFIGURE THE A/D MODULE */

    /* configure analog pins, voltage reference, and digital I/O (ADCON1)
     * AN3:0- analog, AN12:4- digital, Vref- = Vss, Vref+ = Vdd */
    ADCON1 = 0b00001011;
    /* Select A/D input channel (ADCON0) */
    ADCON0 = ADC_CHANNEL;

    /* Select A/D acquisition time (ADCON2)
     * Select A/D conversion clock (ADCON2)
     * A/D Result Format =  Right Justified
     * Acquisition Time = 2T(AD), Conversion clock = Fosc/2 */
    ADCON2 = 0b10001000;


    ADON = 1;       /* Turn on A/D Module (ADCON0) */
    ADIE = 0;       /* Not interrupt driven */
    ADIF = 0;       /* Reset the ADC interrupt bit */
    ADRESL  =   0;  /* Reset the ADRES value register */
    ADRESH  =   0;
}



/*
 * adc_read
 * Description:
 *  Read the analog input on ADC_CHANNEL and return the converted 10 bit
 *  level indictar
 *
 * Operation:
 *  Really just follows standard ADC reading sequence. Check the datasheet.
 *
 * Revision History:
 *  Apr. 28, 2010       Nnoduka Eruchalu        Initial Revision
 */
unsigned int adc_read(void)
{
    volatile unsigned int adc_res;

    CLRWDT();          /* Clear the Watch Dog to allow time to convert b4 reset */
    GODONE = 1;        /* Start the ADC process */
    while(GODONE) continue;
    ADIF = 0;

    adc_res = ADRES;
    return (adc_res); /* return the converted value */
}
```

```
/*
 * ----------------------------------------------------------------------------
 * -----                            LCD.H                                 -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is the header file for lcd.c, the library of functions for interfacing
 *  the PIC18F4520 with a  16 by 2 LCD in its 4-bit databus mode.
 *
 * Assumptions:
 *  The code assumes the actual hardware is hooked up as follows:
 *  PORTD bits 0-3 are connected to the LCD data bits 4-7 (high nibble)
 *  PORTC bit 5 is connected to the LCD RS bit
 *  PORTC bit 6 is connected to the LCD RW bit
 *  PORTC bit 7 is connected to the LCD E bit
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 21, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>

#define _XTAL_FREQ 20000000UL       /* the PIC clock frequency is 20MHz */


#ifndef _LCD_H
#define _LCD_H



/*HARDWARE CONNECTIONS*/
#define LCD_DATA_LAT    LATD        /* use LAT for writing to i/o ports */
#define LCD_E_LAT       LATD
#define LCD_RS_LAT      LATD
#define LCD_RW_LAT      LATD

#define LCD_DATA_TRIS   TRISD       /* set a bit to 1 to make it an input*/
#define LCD_E_TRIS      TRISD       /* set a bit to 0 to make it an output */
#define LCD_RS_TRIS     TRISD
#define LCD_RW_TRIS     TRISD

#define LCD_E_BIT   4       /* PORTD bit 4 */
#define LCD_RW_BIT  5       /* PORTD bit 5 */
#define LCD_RS_BIT  6       /* PORTD bit 6 */



#define LCD_DATA_PORT PORTD         /* use PORT for reading from i/o ports*/



/*LCD CONTROL BIT MANIPULATIONS*/
/* setting a bit is simply ORing in a 1 to that bit and 0s to others */
```

```
#define SET_E()     (LCD_E_LAT  |= (1<<LCD_E_BIT))
#define SET_RW()    (LCD_RW_LAT |= (1<<LCD_RW_BIT))
#define SET_RS()    (LCD_RS_LAT |= (1<<LCD_RS_BIT))

/* clearing a bit is simply ANDing in a 0 to that bit and 1s to others */
#define CLEAR_E()   (LCD_E_LAT  &= (~(1<<LCD_E_BIT)))
#define CLEAR_RW()  (LCD_RW_LAT &= (~(1<<LCD_RW_BIT)))
#define CLEAR_RS()  (LCD_RS_LAT &= (~(1<<LCD_RS_BIT)))



/*MACROS*/
#define LCD_STROBE()  SET_E(); __delay_us(1); CLEAR_E(); __delay_us(1)
#define STR_BUF_SIZE    20  /* size of temporary strings */

/*FUNCTION PROTOTYPES*/
/* initialize the LCD to some documented specs */
extern void lcd_init(void);

/* write a byte to the LCD in 4-bit mode */
extern void lcd_write(unsigned char c);

/* clear the LCD and return the cursor to the starting position */
extern void lcd_clear(void);

/* write a string to the lcd */
extern void lcd_write_str(const char *str);

/* write an integer to the lcd */
extern void lcd_write_int(unsigned int num);

/* move the cursor to a specific location */
extern void lcd_cursor(unsigned char loc);

/* wait on the LCD busy flag */
extern void wait_lcd_BF(void);




#endif
```

```
/*
 * ----------------------------------------------------------------------------
 * -----                            CCP.H                                 -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is the header file for lcd.c, the library of functions for interfacing
 *  with the PIC18F4520's CCP2 interface.
 *
 * Assumptions:
 *  The code assumes the use of CCP2
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 29, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>


/*HARDWARE CONNECTIONS*/

/*MACROS*/

/*FUNCTION PROTOTYPES*/
/* initialize the LCD to some documented specs */
extern void ccp_init(void);
```

```
/*
 * ----------------------------------------------------------------------------
 * -----                            ADC.H                                 -----
 * -----                            EE90                                  -----
 * ----------------------------------------------------------------------------
 *
 * File Description:
 *  This is the header file for adc.c, the library of functions for interfacing
 *  the PIC18F4520's ADC interface.
 *
 * Assumptions:
 *  The code assumes the actual hardware is hooked up as follows:
 *  The input analog signal is connected to ADC_CHANNEL which is 0 to 3 only!
 *  The input analog signal is connected at PORTA.
 *
 * Compiler:
 *  HI-TECH C Compiler for PIC18 MCUs (http://www.htsoft.com/)
 *
 * Revision History:
 *  Apr. 28, 2010       Nnoduka Eruchalu        Initial Revision
 */

#include <htc.h>

#ifndef _ADC_H
#define _ADC_H

/*HARDWARE CONNECTIONS*/
#define ADC_CHANNEL 0       /* this has to be on PORTA and should be 0 to 3 */


/*FUNCTION PROTOTYPES*/
/* initialize the ADC interface to some documented specs */
extern void adc_init(void);

/* read the digital conversion of the Analog input on ADC_CHANNEL */
extern unsigned int adc_read(void);

#endif
```