Vehicular Location Via Ultrasonic Triangulation

Report By:
Dan Krill

Embedded System Design
ECE4534
4/28/2007

Table of Contents

Overview

The Goal of this project is to locate several moving vehicles, giving them a Cartesian set of coordinates in an acoustically and electrically noisy environment. Each of these moving targets should be positively identified and assigned a consistent address value that the control system can repeatedly use and retransmit to other devices for other external tasks.

This process will be executed using ultrasonic receivers placed strategically around the track. Each train will have its own transmitter, and when told, will transmit a brief pulse. Since all the receivers are in different locations around the track, they will all receive that pulse at different times. The order in which these pulses are received will be used to determine the origin of the pulse thus the coordinates of the train can be calculated.

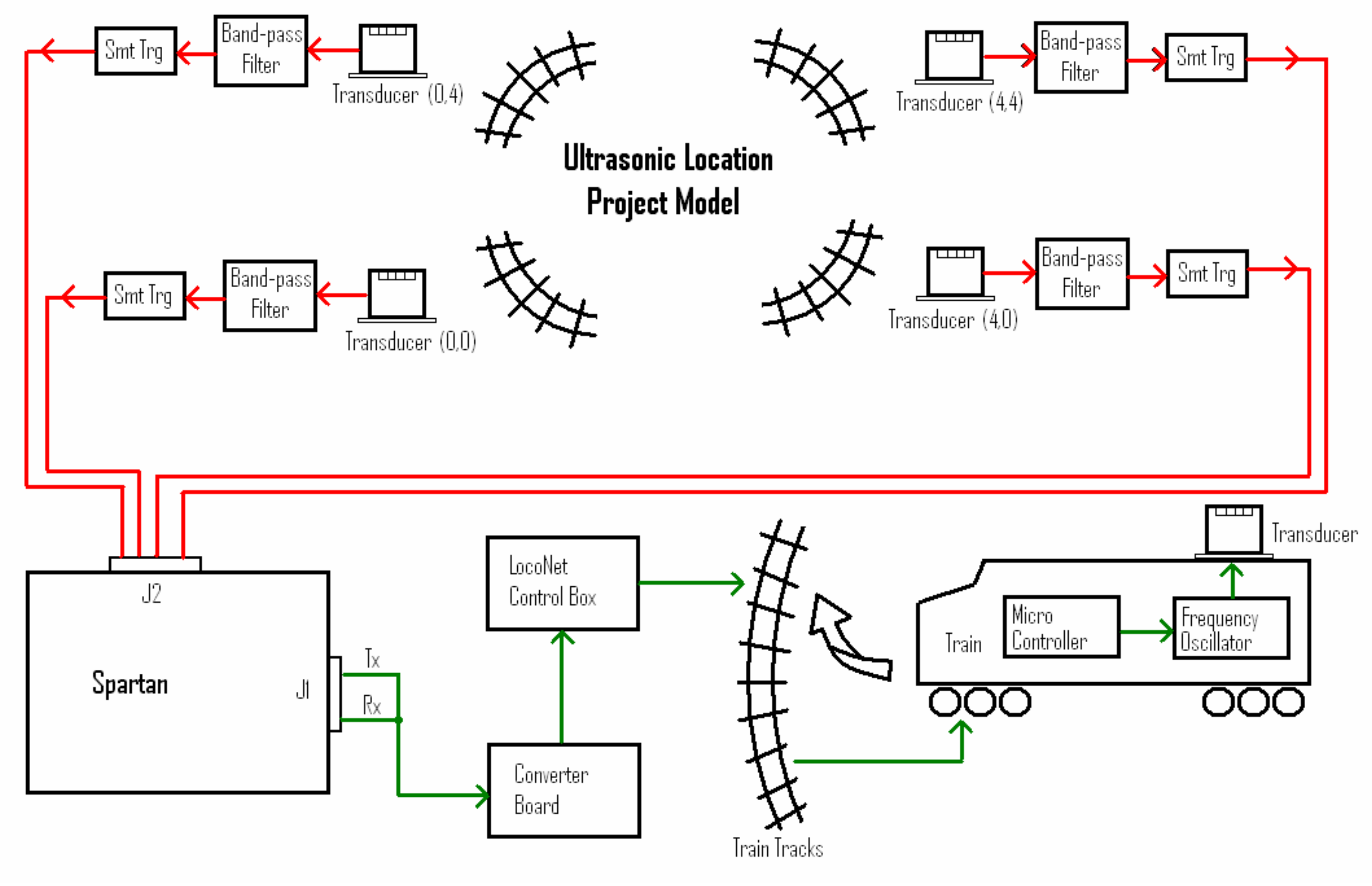


Figure 1) Overall Project Model

Project Achievements

After much trial and error using an oscilloscope as the primary tool, a working receiver and transmitter model were ascertained. The original schematics for these devices were good only for their structural assembly, as their pre-determined values proved to not be as effective in the real world. With the working models constructed on breadboards, actual permanent PCBs were able to be assembled with radio shack perforated through-hole boards, ebay components, and a soldering iron. The breadboard models acted as a master copy for the PCB cloning.

After much physical labor, 1 transmitter board, and 2 receiver boards were produced. Rigorous trials and extensive testing deemed them suitable for their designed application; they passed the QC.

The next working portion of this project came from receipt of an updated bit-stream. With that, I was able to enable a set of interrupts for the j4 jumper header. This we done so that the receivers could be directly connected to the Spartan board, and interrupts are the best way to ensure that the program will handle the data as fast as possible. Due to time constraints and several programming road blocks, however, the position determining algorithm was never fully realized. At its conclusion, the software can only notify the user via the on-screen prompt and LEDs which side of the track the train is closest to.

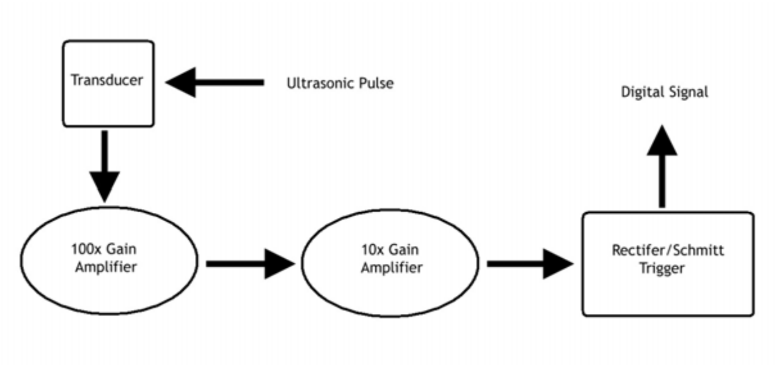


Figure 2) Receiver Process Flow Chart

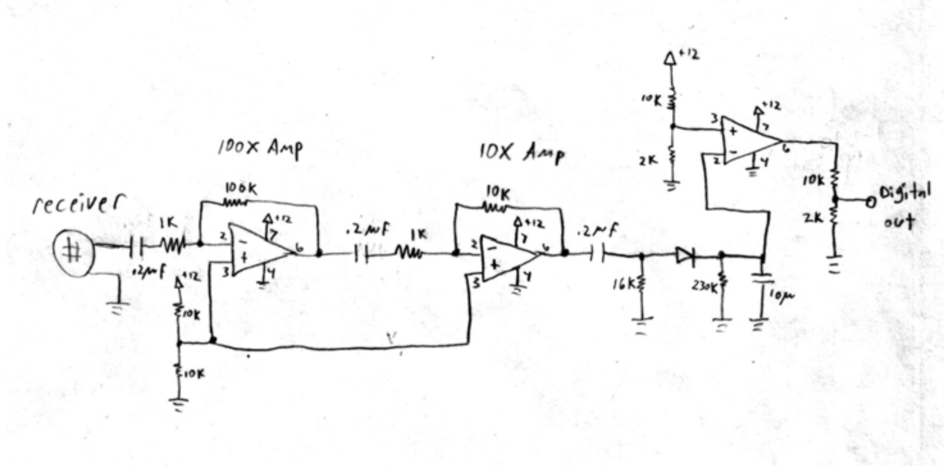


Figure 3) Receiver Electrical Schematic

Design and Implementation

The heaviest design portion of this project came from the receivers. The initial requirements were to have it pick up it's acoustic surroundings, filter out everything except the desired frequencies, amplify the remaining signal, and output a digital pulse if the incoming frequency is what's being looked for.

Given this set of requirements, fundamental circuits were chosen for integration to achieve the final device. The design involved an R-C passive filter, an op-amp based amplifier, and a Schmitt trigger. Values were chosen and calculated for peak efficiency around operation at 25KHz.

It was not until the prototype was assembled, that it was found that the original design was not sufficient. Also, on the opposite side of the coin, it was apparent that the R-C filter was not even necessary. As it turned out, the transducers had a highly narrow frequency range of operation; they wouldn't even work at other frequencies, so that filtering network was cut out. The other problems were that the heard signals were not strong enough. There was not enough amplification to reach the minimum threshold of the Schmitt trigger. The resolution to this was to include a second op-amp for a 2 stage amplifier. At 100x and 10x, the gain was now 1000x, which was sufficient for minimum threshold pickup. Once the wave's amplitude was large enough, it was observed that the wave was actually slow enough for the Schmitt triggering op-amp network to pick up all the oscillations, therefore its binary output was also oscillatory. This was alleviated by introducing yet another fundamental circuit network; this time a DC rectifier. This would allow only one side of the waveform to pass, level it off at the top by charging up a capacitor, thus ultimately holding a pretty constant voltage when the wave is present. This would allow for a better operation of the triggering mechanism. The design of this portion of the circuit came up with the discovery that once the capacitor was charged, it remained charged. That is, once a wave is detected, the whole receiver would signal its presence, but once the signal was removed, it still marked it as being there. It was found that this was caused by the high impedance on the op-amp; the capacity was never given the opportunity to discharge. Putting a resistor parallel to the capacitor and giving it just the right resistance forms a time constant. In this case, the cap will discharge now in only .1ms after the signal is removed.

Finally, at the last stage of the receiver's operation, where the digital output is generated, there was one more little problem that needed to be overcome. The rail to rail voltage operation of the Schmitt trigger was 1volt to 11 volts, and the operating range of the Spartan board and processor is maxed at 3.3. This was easily resolved by placing a voltage divider between the receivers digital out and ground. The Spartan could now be safely interface to the receiver across the lower of the 2 resistors in that divider network.

The transmitter portion of this project was, for the most part, pretty straight forward. The transmitter, in its most basic of forms, is merely a 555 timer oscillator circuit powered at its highest tolerance, and coupled to the transducer at the chip's output pin. The circuit chosen for this task was capable of swinging its output from 6 volts to -6 volts with the use of a virtual ground. This allowed for the transducer to be directly coupled to the circuit with no added drivers in between and still be able to produce a nicely formed sinusoidal waveform.

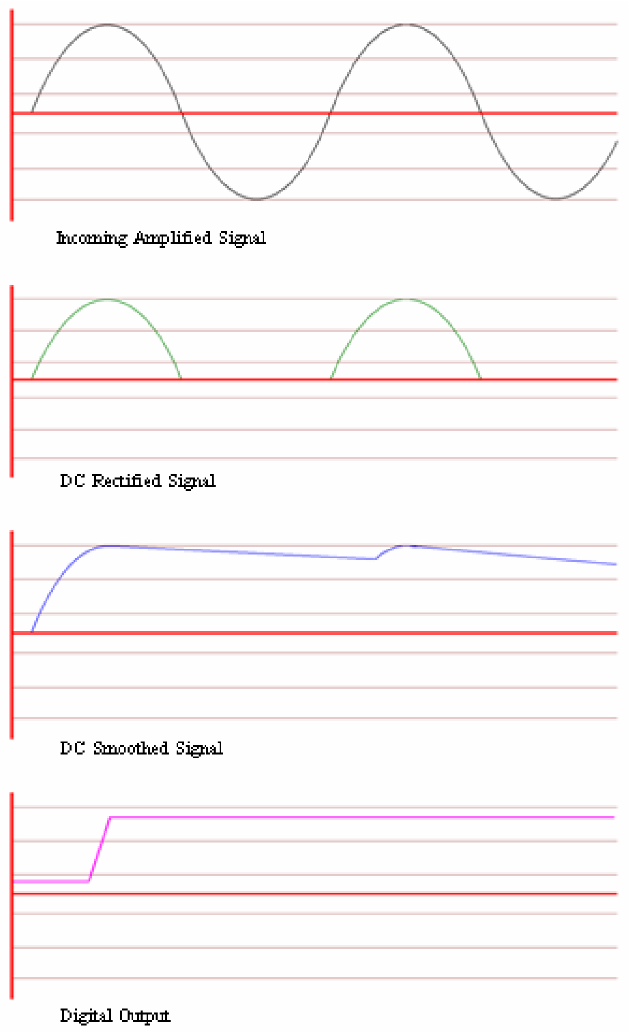


Figure 4) Receiver Wave Forms

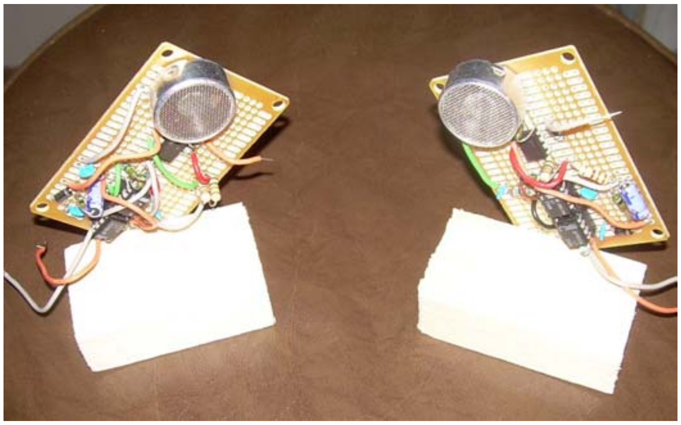


Figure 5) Receiver Modules

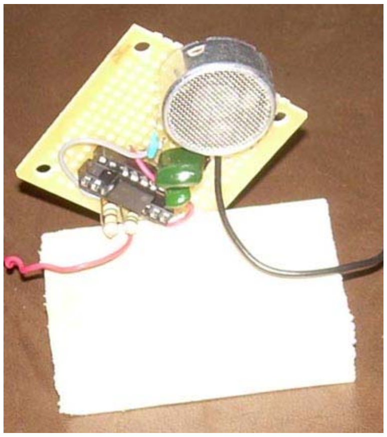


Figure 6) Transmitter Module

The software design was probably the least developed portion of this project due solely to the lack of time available to bring everything to a working closure. So while the original plans and expectations for the software were never met, it was at least brought to a small level of functionality. At present, the Spartan board and software can wait in a low power mode, and then when the receivers send their data to the board, the software wakes up the board, and has the board illuminate a set of indicator LEDs to show which side it heard first. In addition to the LEDs, the Spartan board will also write a set of prompts to the screen providing the operator with a more detailed description of what is going on.

The software is operating at an interrupt level, so it is fairly fast. The core of the code is actually based on the button press software used at the beginning of the course. The inputting was easily pointed to the J4 jumpers by modifying a few of the constants used in the interrupt handler initialization. So while the software does very little, it at least does it well.

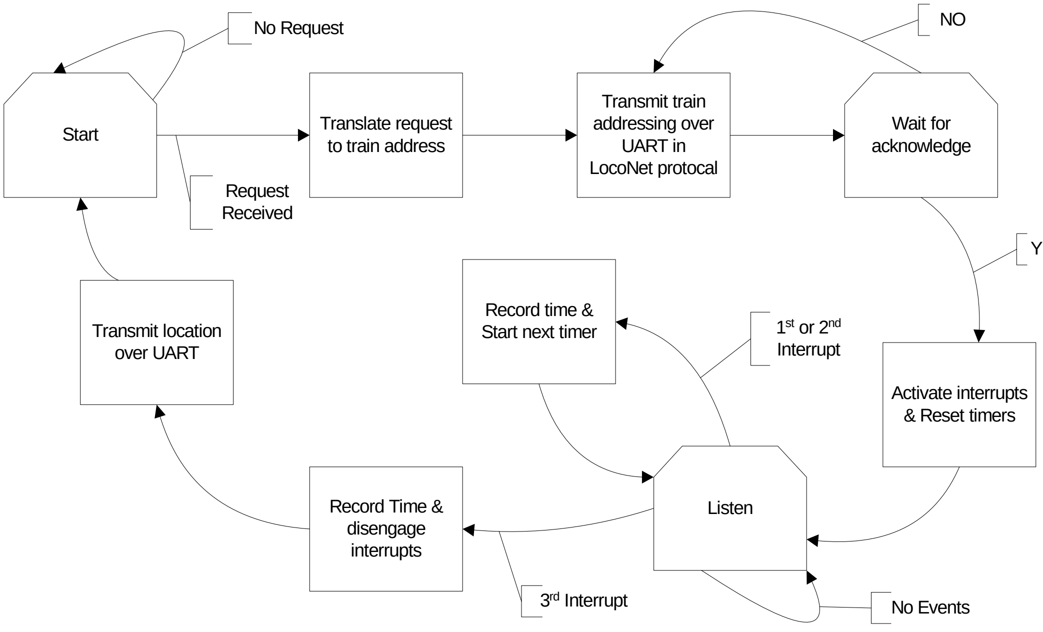

No Request
Start
Request Received
Translate request to train address
Transmit train addressing over UART in LocoNet protocal
NO
Wait for acknowledge
YES
Activate interrupts & Reset timers
Record time & Start next timer
1st or 2nd Interrupt
Transmit location over UART
Listen
No Events
Record Time & disengage interrupts
3rd Interrupt

Figure 7) Original Software State Flow Chart

Design Analysis

It took a lot of testing and tweaking to get the transmitter and receivers to play nicely. After undergoing several design changes, the two were finally in a state of readiness. In the final design, either receiver would signal the presence of the ultra-sonic pulse approximately 2ms after the transmitter's activation. This brief pause is due to the nature of the transmitter circuit, and not the receivers. Because of the number of capacitors on the transmitter, it takes roughly 2ms before it can fully reach steady state. During the transient period prior to the steady state, the frequency is working its way up to the 25 kHz mark. This lag does not affect the final results of the system because it is consistent and uniform upon receipt at either receiver site. In software, if ranging calculations are being conducted, it just needs to be taken into consideration that the pulse was actually received 2ms less than actually timed.

Something else that popped up later on in the testing process was something involving the sensitivity of the receivers. Even though both were made from the exact same parts, and assembled basically the exact same way, it was plainly obvious that one receiver was much more sensitive than the other. They both seemed to work fine, but one was so sensitive, that it was actually difficult to shield the transmitter's transmission from it. With one, it was easily disrupted when sliding a box or hard surface in front of its transducer, while the other, could be put under the table, inside a closed box, or whatever, and it still heard the transmitter. While this does not really affect the outcome of the project overall, it did ruin the prospect of simulation control and project consistency.

Conclusions

At the end of this project, there was 1 transmitter module, 2 receiver modules, a computer power supply providing them with power, and the computer linked Spartan board connected to the modules via its J4 header. The software was able to tell which side of the track the train was closest to. Though it doesn't sound like much, many areas of specialization were involved. Just some of the areas I covered in this project were, high level embedded OS programming, hardware manipulation, ultrasonics, amplifiers, acoustic transmitters, system integration, UART data transmission, and more. Very few projects are able to offer such a wide variety of areas while still maintaining feasibility.

While I have worked with, or at least seen, all of those areas before, there were still some concepts utilized in this project that were, for the most part, new to me.

Designing of the transmitter and receivers began initially from circuit concepts I already knew, but the fine details associated with them I had to learn from either my research or from actual trial and error. I discovered when using a dc powered op-amp to amplify ac based signals, there is a special way of biasing the op-amp so you do not loose the below-zero portion of the wave form. To do this, you have to create a virtual ground for the positive input. By creating a "ground" higher than its source grounds, the ac outputted signal is now dc biased and scooted up from the true ground, thus allowing the amplified signal to continue to swing both above and below the x-axis. The virtual ground can be made with a simple voltage divider across the main vcc and ground.

As far as the software is concerned, for the longest time, I was still kind of unclear as to how the interrupts were actually initialized. With more reading and scanning through the documentation and actual code, I was able to see that most of the interrupts are set up already and that the OS points to various registers to decide which interrupts to use. In the software, these registers are pre loaded with values that are defined as constants in the code. To control which interrupts run the interrupt handlers, one must only change these constants. It was from this newly acquired knowledge that I was so easily able to modify my button pressing code to work with the J4 jumper header instead.

There are a few things that could have been done differently that would have made this project turn out even better than it already had. For one, it would have been nice if the circuit boards used for the transmitter and receivers were actually manufactured instead of "home-made" as mine were from radio shack parts. This would have better ensured consistency among the receivers. The current ones are identical, or so they seem, yet one is more powerful than the other. I feel if these boards were designed in time and sent off to be professionally made, they would have turned out a little better.

Over all, I feel this project was a very good example of design work in the real world. Also, using the trains as our project medium, gives a better feel of the dangers and financial mishaps that can occur with poor engineering and just how important it is to analyze your design from every angle possible especially when human life is at stake.

References

"555 Timer IC." Wikipedia; The Free Encyclopeida.
15 February 2007. <http://en.wikipedia.org/wiki/555_timer_IC>

"Operational Amplifier." Wikipedia; The Free Encyclopeida.
28 January 2007. <http://en.wikipedia.org/wiki/Op-amp>

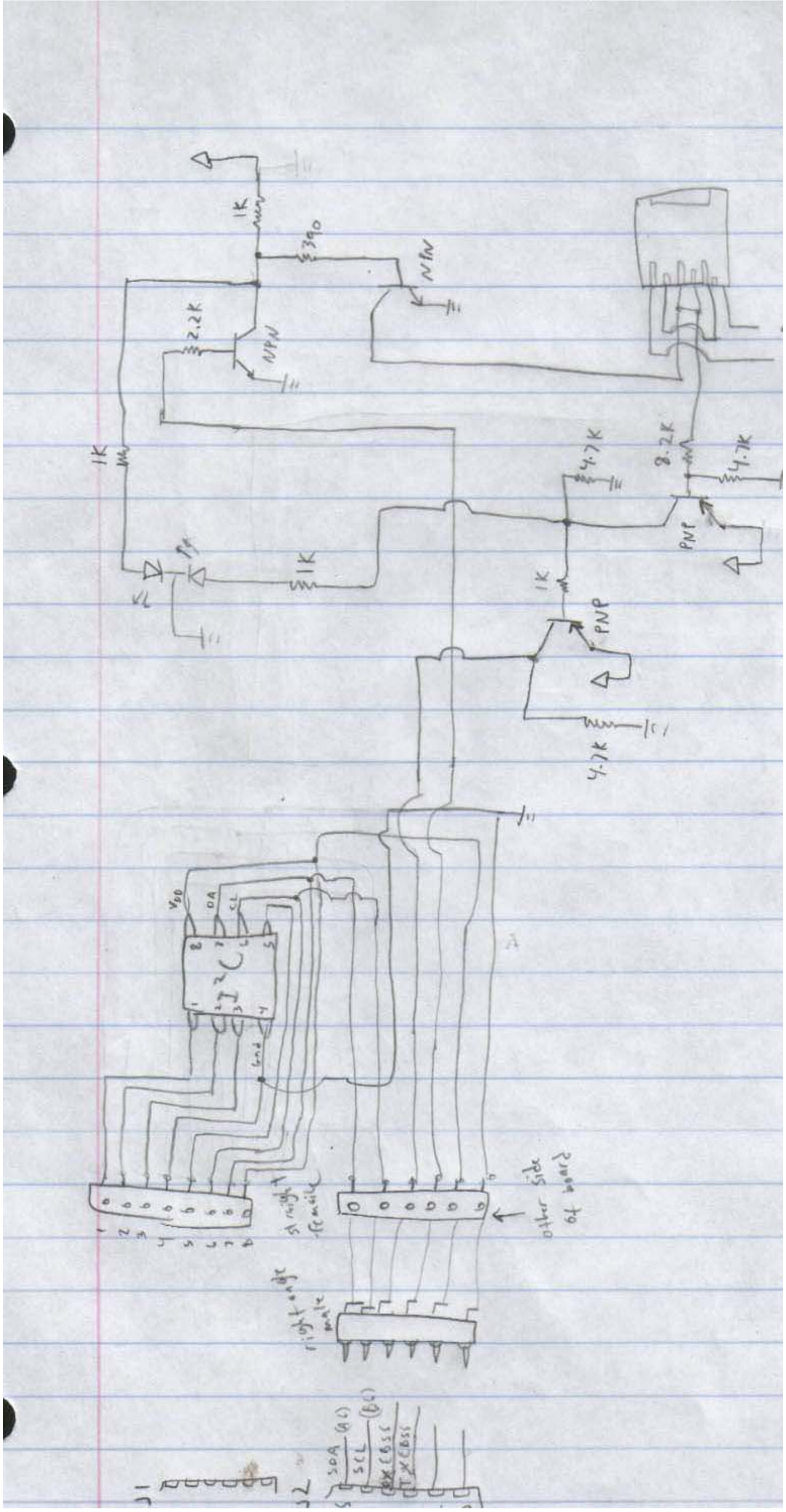
Lazar, Tomaz. "Ultrasonic Dog Whistle." MIT Free Servers.
23 February 2007. <http://www.mitedu.freeserve.co.uk/Circuits/Misc/whistle.htm>

Bowden, Bill. "Simple Op-Amp Radio." Bowden's Hobby Circuits.
23 February 2007.
<http://ourworld.compuserve.com/homepages/Bill_Bowden/page6.htm#fm.gif>

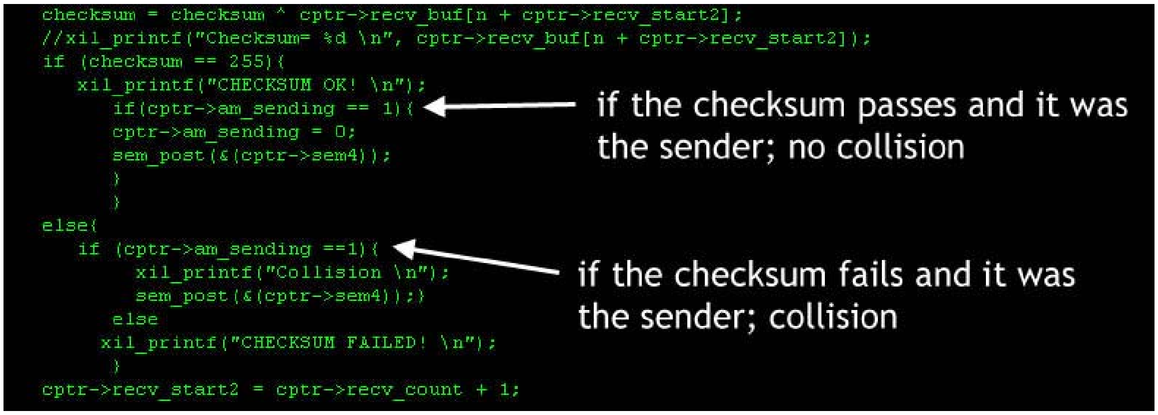
Appendix

Contents

Appendix: Loconet Breakout Board Schematic



Appendix: Collision Detection Handling

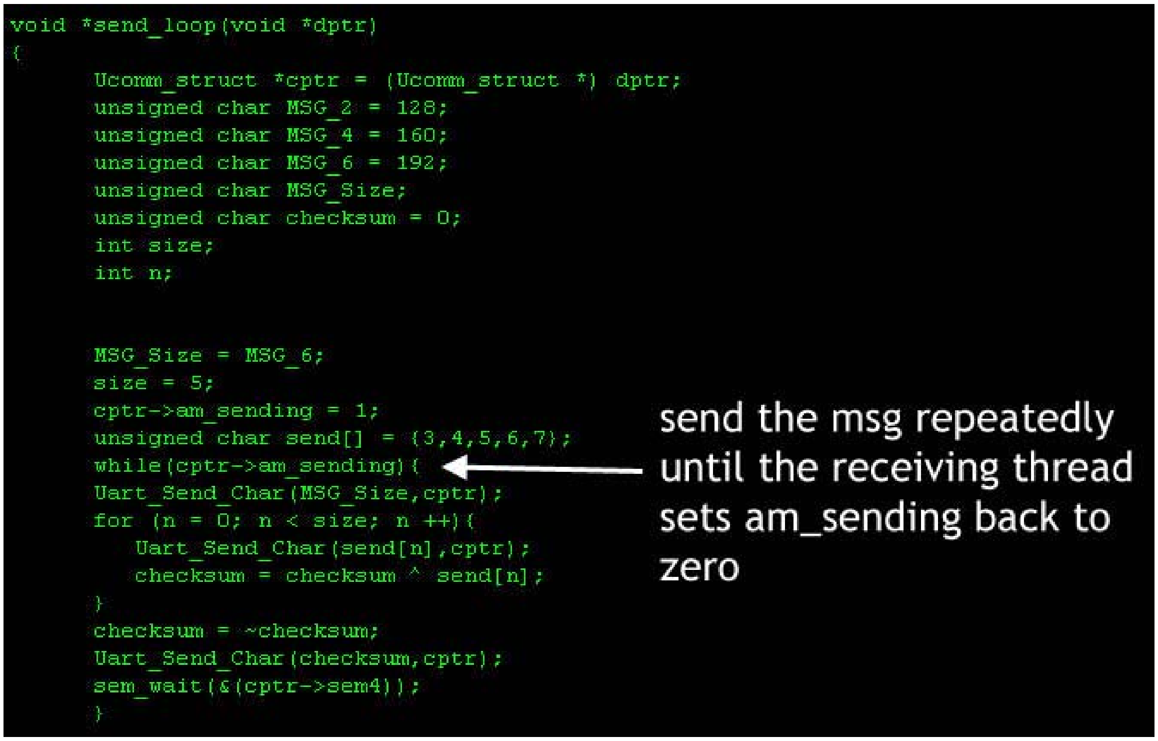


```
checksum = checksum ^ cptr->recv_buf[n + cptr->recv_start2];
//xil_printf("Checksum= %d \n", cptr->recv_buf[n + cptr->recv_start2]);
if (checksum == 255){
   xil_printf("CHECKSUM OK! \n");
      if(cptr->am_sending == 1){
      cptr->am_sending = 0;
      sem_post(&(cptr->sem4));
      }
      }
else{
   if (cptr->am_sending ==1){
        xil_printf("Collision \n");
        sem_post(&(cptr->sem4));}
      else
     xil_printf("CHECKSUM FAILED! \n");
      }
cptr->recv_start2 = cptr->recv_count + 1;
```

if the checksum passes and it was the sender; no collision

if the checksum fails and it was the sender; collision

Receiving Thread; Primary Loop



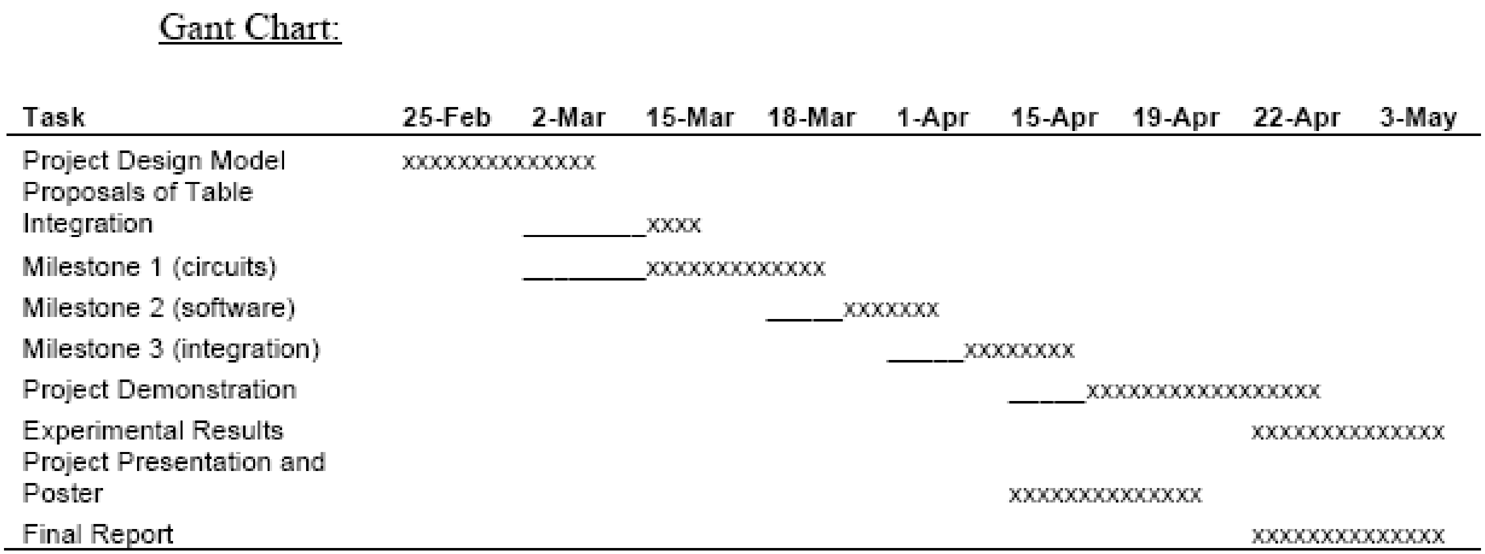
```
void *send_loop(void *dptr)
{
       Ucomm_struct *cptr = (Ucomm_struct *) dptr;
       unsigned char MSG_2 = 128;
       unsigned char MSG_4 = 160;
       unsigned char MSG_6 = 192;
       unsigned char MSG_Size;
       unsigned char checksum = 0;
       int size;
       int n;


       MSG_Size = MSG_6;
       size = 5;
       cptr->am_sending = 1;
       unsigned char send[] = {3,4,5,6,7};
       while(cptr->am_sending){
       Uart_Send_Char(MSG_Size,cptr);
       for (n = 0; n < size; n ++){
          Uart_Send_Char(send[n],cptr);
          checksum = checksum ^ send[n];
       }
       checksum = ~checksum;
       Uart_Send_Char(checksum,cptr);
       sem_wait(&(cptr->sem4));
       }
```

send the msg repeatedly until the receiving thread sets am_sending back to zero

Sending Loop; Primary Loop

Appendix: Original Project Time Line



Gant Chart:

Task	25-Feb	2-Mar	15-Mar	18-Mar	1-Apr	15-Apr	19-Apr	22-Apr	3-May
Project Design Model	xxxxxxxxxxxxxx								
Proposals of Table Integration		_________xxxx							
Milestone 1 (circuits)		_________xxxxxxxxxxxxx							
Milestone 2 (software)				_____xxxxxxx					
Milestone 3 (integration)					_____xxxxxxxx				
Project Demonstration						_____xxxxxxxxxxxxxxxxx			
Experimental Results								xxxxxxxxxxxxxx	
Project Presentation and Poster						xxxxxxxxxxxxxx			
Final Report								xxxxxxxxxxxxxx	

Appendix: Original Team Proposal

1) Participation:

There will be two separate demonstrations. We were going to do one, but two of the individual projects conflict with each other.

Demonstration1:	Dan Krill -	Train location via ultrasonic triangulation
	Jordan Mizak -	Train Control over RF
Demonstration2:	Taylor Green -	Train Control with homemade DCS
	Chris Thorwarth -	Acceleration data stream

2) Demonstration Descriptions:

Demonstration1: We will demonstrate using two Spartan S3E boards to send Loconet messages via a RF link to an on-train controller which will control the train's speed, direction, and ultrasonic triangulation emitter device. The one Spartan board will be solely for the train’s operation, while the other will be for triangulation calculations and telling the train when to fire the ultrasonic pulse. We will demonstrate multiple train speeds as controlled wirelessly, and a computer screen printout of the trains coordinates when prompted.

Demonstration2: We will demonstrate an accelerometer data logging system with train commands sent through a DCS50 replacement board, and accelerometer data transmitted over a train-mounted RF device. We will demonstrate one locomotive operating at a pre-defined number of speed steps.

Appendix: Fun Ultrasonic Facts

The speed of sound at this elevation is estimated to be roughly:

700 miles per hour or

1,026 feet per second.

Taking the inversion, it takes sound:

0.000974 seconds to travel one foot or

0.974 ms/foot.

At 50 MHz in 0.000974 seconds 48,700 clock pulses will pass.