Work Completed:

This week I began programming the platform microcontroller. I started by programming and testing the A/D conversion algorithm for the two load cell amplifier inputs. Using potentiometers, I found that the microcontroller actually calculated the voltage about four counts higher than when I tested it with my multimeter. This within an acceptable margin of error as long as the error is not nonlinear, which seems to be a valid assumption from the data I collected. AN0 and AN1 were selected as the primary load cell A/D ports in the original code, however, since the wireless received signal strength A/D port on the display occupies AN0, I may change the load cell inputs so that the received signal strength input is on the same port of the platform and the display. Several additional A/D ports are available for future expansion, such as additional load cells on each rail.

I began work on a serial send and receive program that would be common to both the platform and the display. While to program incorporated the outputs necessary to drive the wireless transceiver properly, that component has not yet been fully prototyped and thus testing can only proceed with a wired connection. One issue encountered when setting up the serial communication was a problem with the internal oscillator speed. Although the chips were intended to run at 1 MHz, a simple test program that just pulsed one of the general I/O ports resulted in a 4 microsecond instruction execution time, which would correspond to a 250 kHz clock cycle. This speed may adversely affect the baud rate generator of the serial communications program. However, as long as both processors have the same baud rate settings, the rates should be equal and serial communication should proceed as expected.

The serial transmission program was fairly simple to program. First a simple check of the byte being sent is performed in order to calculate the parity bit which is important for the wireless transmission within a noisy environment. Next, the parity bit for the transmission is set prior to sending because transmission automatically starts once the transmit register is loaded. Finally, the transmit register is loaded with data and the transceiver is reset to receive mode.

Writing the code necessary for serial receive was significantly more involved than serial send. While send instructions can happen at discrete points in the users code, data can be received at any point in the users code. Since waiting too long to read the received data could result in data loss, interrupts must be implemented to halt the currently executing code to process the received data. Also since the wireless transceiver draws nearly 15 mA of current when operating in receive mode, serial receive enable and disable instructions must be implemented to conserve power. After spending a while configuring the interrupts properly to enable the serial receive interrupt, disable the additional interrupts, and service the serial interrupt, the serial receive routine should function properly.

One issue that still must be addressed is the delay timing that is necessary to power-up the wireless transceiver from sleep mode and to switch between send and
receive. If this issue is not resolved, proper communication using the wireless chips will not be possible.

```assembly
serialsend
bsf RCSTA,SPEN
bsf wirelesspwd
bcf wirelessRXenable
bsf wirelessTXenable

; must add timer for switching

bitadcycle
bcf STATUS,C
rrf serialsendtemp,F
movf serialsendtemp,W
xorwf paritycalc,F
decfsz bitaddloop,F
goto bitadcycle

txbit
btfsc paritycalc,0
call setTX9
btfss paritycalc,0
call clearTX9

MOVFF serialsendbyte, serialsendbyte

 TXwait
bsf STATUS,RP0
btfss TXSTA,TRMT
goto TXwait
bcf STATUS,RP0

maintainwirelessenable
bsf maintainwirelessenable

disablewirelesstransmit
return

disablewirelesstransmit
bcf wirelesspwd
bcf RCSTA,SPEN
return

setTX9
bsf STATUS,RP0
bsf TXSTA,0
bcf STATUS,RP0
return

clearTX9
bsf STATUS,RP0
bcf TXSTA,0
bcf STATUS,RP0
return
```

Current Status:

Testing of the serial communication routines involved entering a key on the key pad and verifying the result of the serial receive bit on the platform processor. After some issues with the interrupts not being serviced, I found that the serial continuous receive bit
had been disabled and was preventing reception. When the two processors are wired together, the communication protocols appear to be functioning correctly. Following extensive testing, only one transmission out of thirty appeared incorrect, but I do not believe that this indicates a problem with any of the serial communications algorithms.

This figure represents the TX data line following a key press on the display keypad.

Future Work:

Next week the serial communication program should be verified with the wireless transceiver operational. The averaging program to determine the average A/D input over a given time interval should be complete along with the transmission programs necessary to communicate this information with the display. Once these aspects are properly functioning, a second platform processor will be prototyped to correspond with the second rail. Both need to be functioning in order to ensure that transmission between the rails and the display proceeds without interference.

Project Review:

Assuming we want to order the PCB by the week prior to Spring Break, I have a significant amount of work remaining. After valid data communication of the A/D converter values from the two rails is confirmed, the data manipulation algorithms must be implemented to change the binary value into an actual weight. A menu system must be derived in order to zero the scale, select between different units, subtract a given weight, and access and retrieve memory information. I expect to be able to finish these algorithms within two weeks, but they must be completed in time for Julie to finish the PCB work and for Chris to review the boards prior to sending the order for fabrication.

Hours Worked: 12 Hours