E.E. 3333-001

Project 1

Remote Event Timer

Kyle Romero

IEEE# 80306278

Texas Tech University

April 29, 2008

Instructor: Dr. Dickens

Advisor: Dr. Mitra

Group Members:

Will Gray, Aaron Adcock, Ronjan Mathur

Abstract

The purpose of this paper is to describe in detail the design and operation of a remote event sensing system that has been built as a Texas Tech project lab project. The system is essentially a timer that is activated or deactivated when an event at one of its transmitters occurs. The timer is accurate to 30+mins. Multiple events can be tracked by using an integrated RFID tag reader system and RFID tags with ID numbers loaded on them. The system implements a power amplifier to increase the usable range of the transmitters.

Contents

Figures

Tables

Introduction

The subject of this paper is a system that was developed to measure the time between two remote events. It contains two remote transmitters and a central receiver. Events occurring at either of the transmitters will activate or deactivate a timer within the central receiver. The timer must be accurate to the second for at least 15 minutes. The following figure contains a overview of the system [4]:

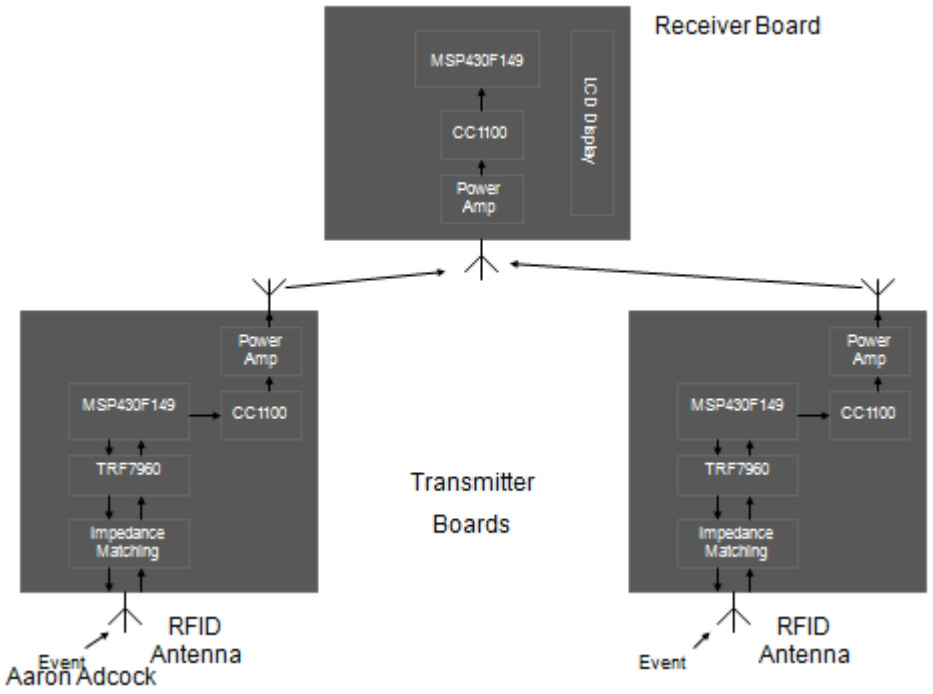


Figure 1- System Diagram

The transmitters and receivers are based around the TI CC1100 sub-1GHz transceiver board. This is a very compact and useful package that will be described in more detail later in the paper. Several CC1100s are connected to custom built microcontroller boards specifically designed for this project.

CC1100

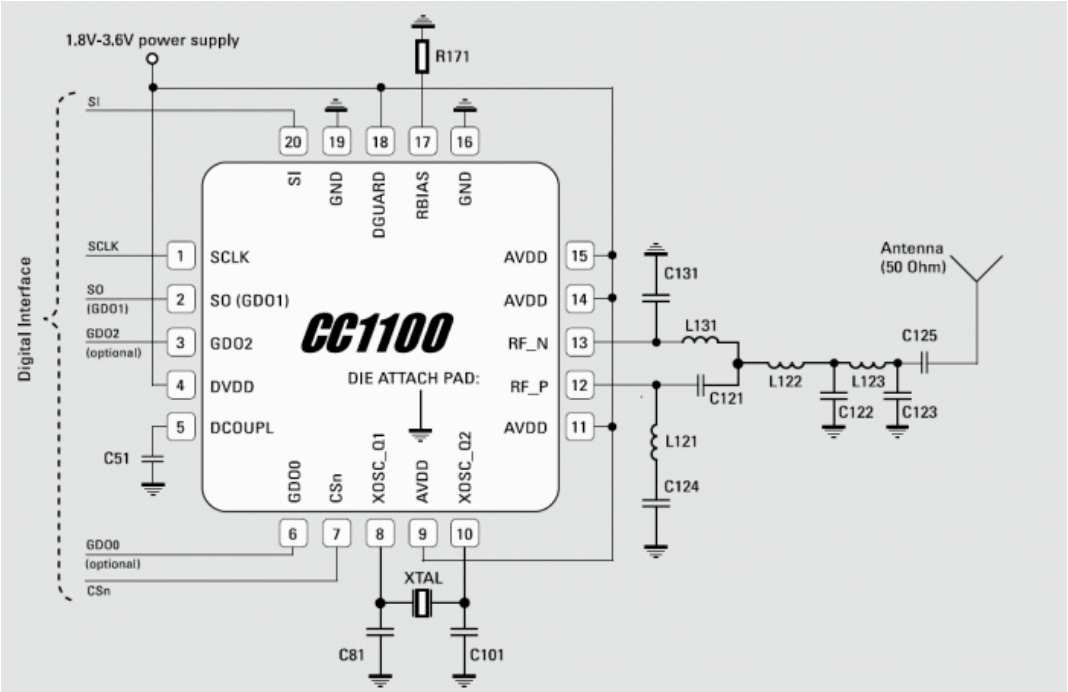


Figure 2-CC1100 [2]

According to the CC100 datasheet, "The CC1100 is a low cost true single chip UHF transceiver designed for very low power wireless applications. The circuit is mainly intended for the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency bands at 315, 433, 868 and 915MHz, but can easily be programmed for operation at other frequencies in the 300- 348MHz, 400-464MHz and 800-928MHz bands."[1].

Modulation

The CC1100 supports several different modulation techniques for its RF transmissions. The supported modulation techniques are:

Table 1: Modulation Techniques

ASK
OOK
2-FSK
GFSK
MSK

For detailed information about each modulation technique please refer to Appendix A.

Originally, it was planned to allow the receiver board to dynamically switch between multiple modulation techniques to attempt to get a longer range. However, due to time constraints this feature was never implemented. As of the end of the development cycle of this project, the modulation technique has been set to the default MSK, transmitting at a power level defined by the programmer at run time, and at 433 MHz.

Packet Construction

For purposes of this project, all that is needed is a very simple packet. A packet is the grouping of information to be transmitted. The following figure illustrates the official

packet setup provided by the CC1100:

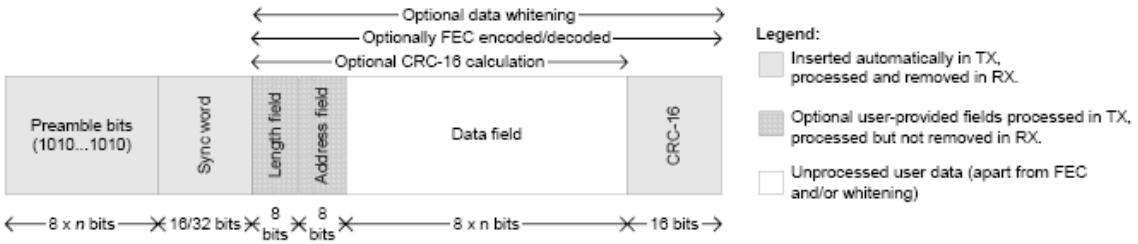


Figure 3- Packet Format

The timer is activated/deactivated when a ping from one of the transmission boards is detected. This allows the receiver board to record the time between two events. The actual packet transmitted is simply full of 1's because the contents are irrelevant. No data is transmitted within the packet, the packet itself is the thing that activates the receiver board timer.

Programming

The CC1100 is programmed by being connected to a separate MCU. In this case the MCU is the MSP microprocessors contained on the custom built microcontroller boards designed for this project. The CC1100 contains several important registers as well as command strobes that are important to the timing systems' operation.

Command Strobes

Command strobes are instructions to the CC1100 that control internal sequences of operations within the CC1100. The important command strobes with descriptions are as follows:

Table 2: Command Strobes

SRX	Enable RX. Perform calibration first if coming from IDLE and MCSM0.FS_AUTOCAL=1.
STX	In IDLE state: Enable TX. Perform calibration first if MCSM0.FS_AUTOCAL=1.If in RX state and CCA is enabled: Only go to TX if channel is clear.
SIDLE	Exit RX / TX, turn off frequency synthesizer and exit Wake-On-Radio mode if applicable.
SWOR	Start automatic RX polling sequence (Wake-on-Radio) as described in Section 19.5 if WORCTRL.RC_PD=0.

The command strobes allow easy state change of the CC1100 and greatly speed of the programming of the chip. They also insure everything is correctly setup for a given state, and take away a lot of the fallibility that comes inherent with direct programming.

Power Settings

The CC1100 provides two registers two control the power output level of the transmitter, PATABLE and the PA_POWER bit of FREND0. PATABLE is a special register that can hold up to eight user specified output power settings. The PA_POWER bit of FREND0 selects which PATABLE entry to use. If the current modulation technique requires power ramping, then more than one value should be entered into the PATABLE and the highest desired power output entry should be selected with the PA_POWER bit. Otherwise,

only one PATABLE entry should be entered. If the power output should be changed, simply rewrite the PATABLE with a new single entry. The following figure illustrates the recommended power settings.

	433 MHz	
Output Power [dBm]	Setting	Current Consumption, Typ. [mA]
-30	0x04	11.5
-20	0x17	12.1
-15	0x1C	12.7
-10	0x26	14.0
-5	0x57	13.7
0	0x60	15.6
5	0x85	19.1
7	0xC8	24.2
10	0xC0	29.2

Figure 4- Power Settings

The Receiver Board

The receiver is the microcontroller board that has been created to run the timer program.

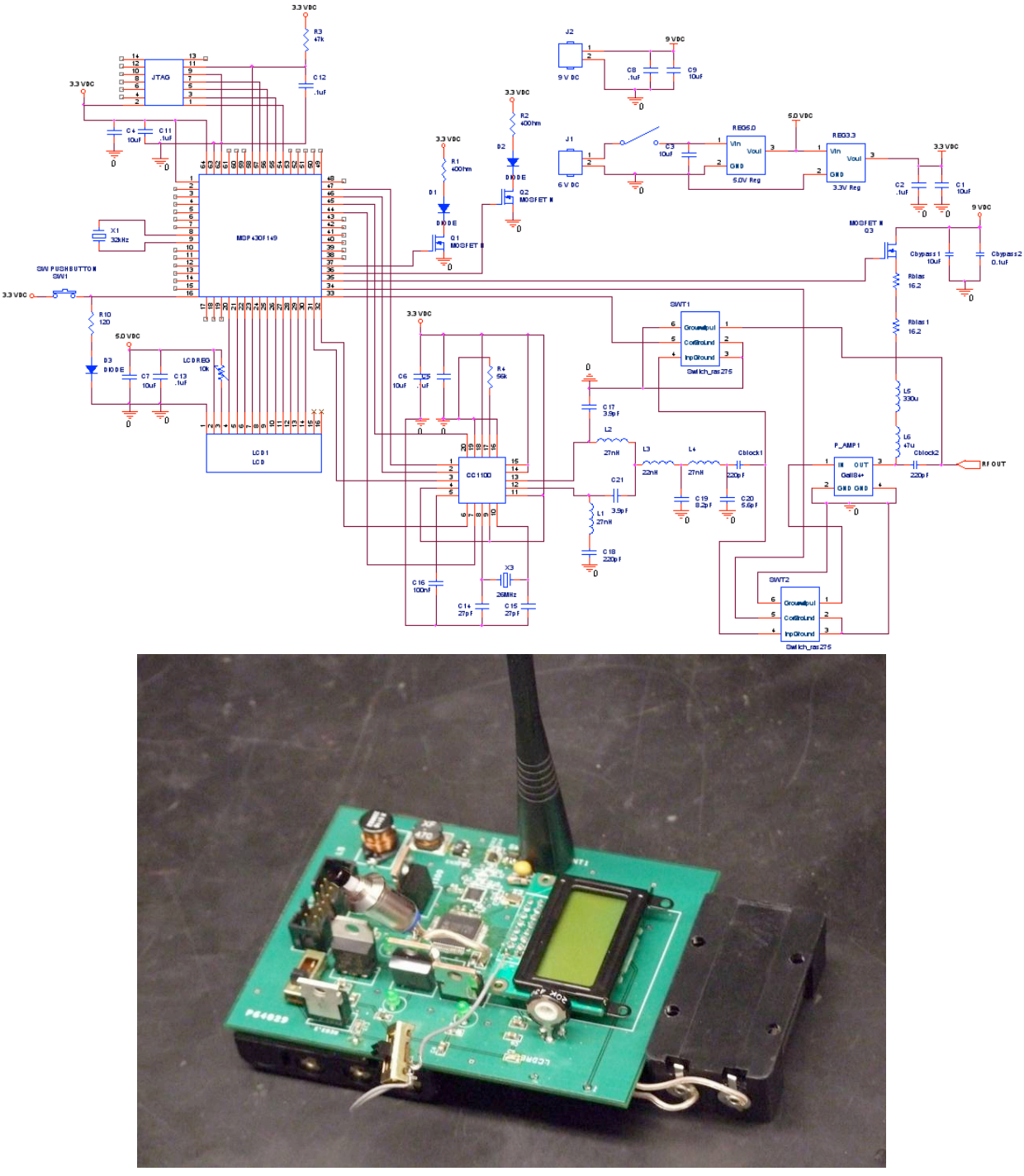


Figure 5- Receiver Board [3]

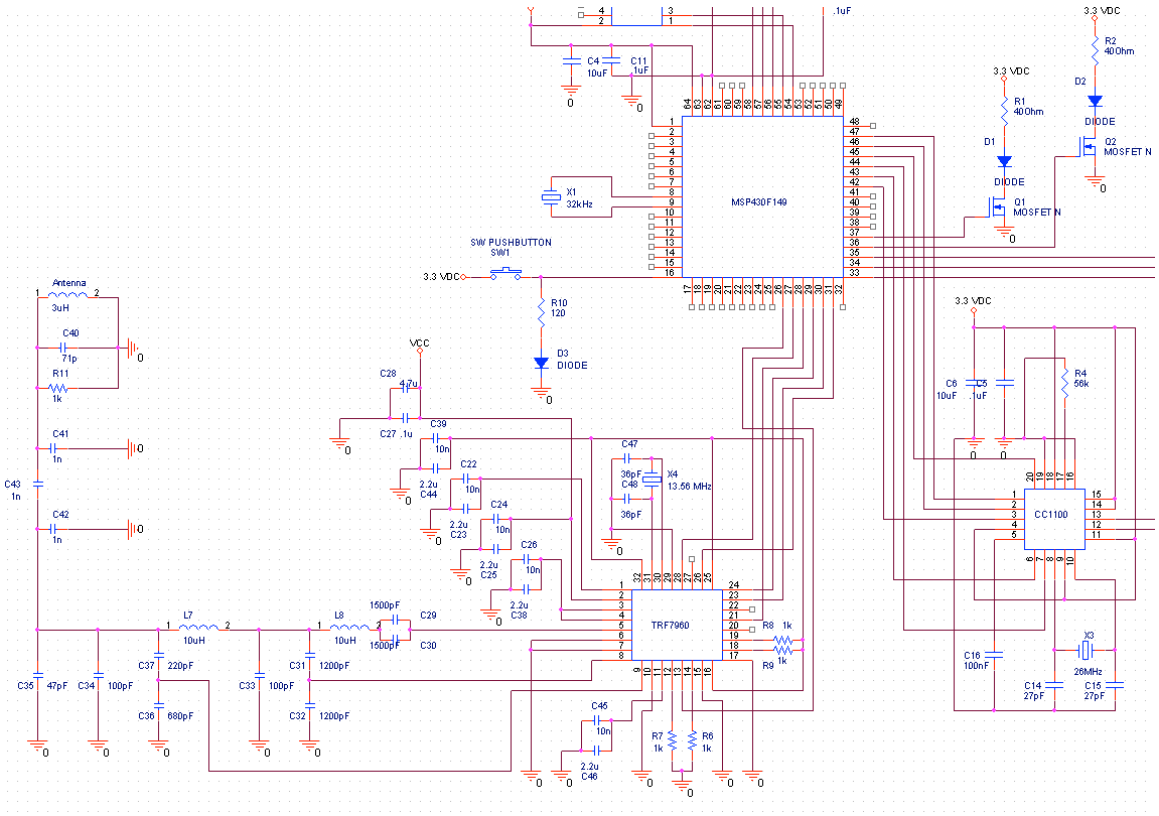
There are several main features of the Receiver Board. It contains a MSP430F149 microprocessor, an LCD screen, and integrated CC1100. The main reason for the LCD screen is to display the current state of the Timer. The board is connected to the computer using a JTAG parallel port -> serial connection, and programmed using the IAR workbench programming suite.

The LCD is made by Crystal Fontz, and is a simple 8 character by 2 lines display. The LCD contains 3 command pins and 8 data pins. The 3 command pins are RS, which controls whether the data sent is a character (High) or a command (Low), R/W, which is always low because low causes data to be sent to the LCD, and R_EN, which when dropped low causes the LCD to read the data pins. One final note: The LCD was connected to Port 2 and Port 3 on the MSP.

The CC1100 is connected to Port 4 of the MSP, and is interfaced with SPI bit-banging.

The Transmitter Board

The transmitter board similar to the receiver board, however, it contains a RFID antenna in place of the LCD. It also has an integrated TRF7960 RFID reader. Its main purpose is to interface and transmit data from the sensor so as to trigger the timer on the receiver board. It also contains an interface to connect a power amplifier to increase transmission ranges.



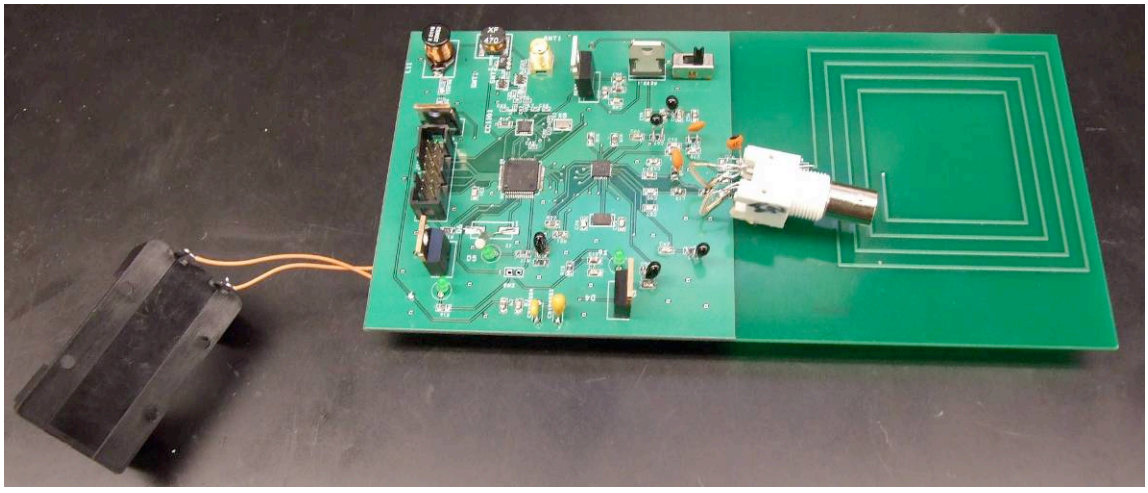


Figure 6- Transmitter Board [3]

The RFID antenna was designed by Aaron Adcock, and, along with the TRF7960, is designed to work in the 13Mhz – 14Mhz range.

The Power Amplifier

The following schematic represents the power amplifier chosen to be attached to the transmitter boards.

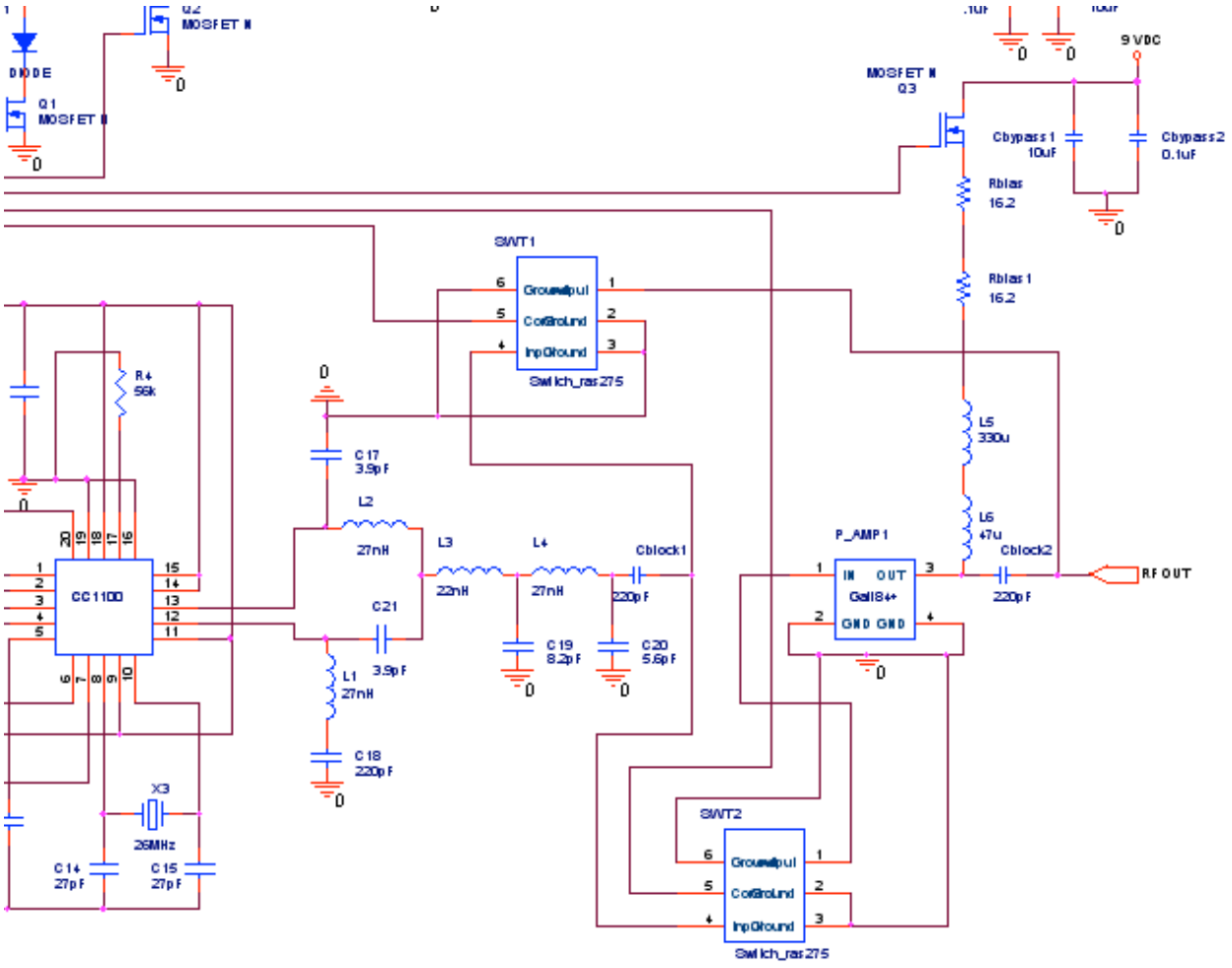


Figure 7- Power Amplifier [5]

The power amplifier is designed to boost the TX output power to allow a greater operational distance. Its maximum output power is 23dBm. There is a software switch inside the transmitter board that must be activated in order to turn on the power amplifier. If this is neglected then transmission will not occur.

The Software

The programming for the different boards was written in the IAR Workbench programming suite. The code was written in the C programming language, and deals mostly with setting up the registers and SPI on the different boards. The rest of the code mainly deals with the interrupts and the timer program.

Low Level Programming

Registers

Registers are a specific memory location. Therefore, the easiest way to program them is to first define their names as an alias for the memory location via a compile time macro.

For example:

```
#define Register 0x5F
```

SPI

Once all of the registers are setup in this fashion, and stored in a header file, you must then setup the SPI, which stands for Serial Peripheral Interface. The steps for interfacing via SPI are illustrated in the following figure:

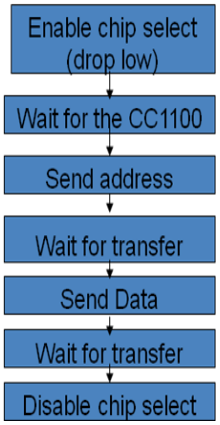


Figure 8- SPI [4]

The SPI was implemented using the bit-banging technique. This means that the whole process is coded manually. An example of bit banging follows [7]:

```
        const unsigned char bitMask8[] = {
   0x80,  // binary 10000000
   0x40,  // binary 01000000
   0x20,  // binary 00100000
   0x10,  // binary 00010000
   0x08,  // binary 00001000
   0x04,  // binary 00000100
   0x02,  // binary 00000010
   0x01   // binary 00000001
};

// This will send data in bit7~0, updating the clock each bit
void send_8bit_serial_data(unsigned char data)
{
   unsigned char x;

   output_high(SD_CS);     // lets select the device

   // Loop through all the bits, 7...0
   for(x = 0; x < 8; x++)
   {
       if(data & bitMask8[x])
       {
           output_high(SD_DI);       // we have a bit, make high
       }
       else
       {
           output_low(SD_DI);        // no bit, make low
       }

       output_low(SD_CLK);           // update clock
       output_high(SD_CLK);          // update clock
   }

   output_low(SD_CS);     // lets DE-select the device
}
```

LCD

Also, as mentioned earlier, the LCD requires a specific method, which is basically bit-banging, to get commands into its controller. The following code illustrates how to send commands to the LCD.

```
/*
 * Write a command to the LCD.
 */
void LcdWriteCmd(unsigned char cmd)
{
      P2OUT = LCD_RWEN; //E
      P2OUT |= cmd<<3;
        P3OUT = cmd >> 5;
        P3OUT+= POWERAMP;
      P2OUT ^=LCD_RWEN; //
    Delay(100);//waitUntilDone();
} // LcdWriteCmd
```

The command has to be bitwise shifted left by 3 bits to correctly output on port 2 and has to be be shifted 5 bits right to be outputted on Port 3. This is because the 5 data pins are on port 2 and 3 are on port 3. LCD_RWEN is the enable pin and is dropped low to send the command.

TRF7960

The TRF also requires bitbanging to configure it. The following code is the code used to bitbang the TRF:

```
void bitbangtrf(unsigned char data){
 unsigned char x;

 P3OUT = 0x01;

    for(x = 0; x < 8; x++)
 {
     if(data & bitMask8[x])
     {
         P3OUT |= 0x02;
     }
     else
     {
         P3OUT |= 0x00;;
     }

     P3OUT |= 0x08;
     P3OUT ^= 0x08;              }

 P3OUT ^= 0x01;
}
```

The TRF requires an initialization routing upon power-up, much like the LCD screen on the receiver board. The following commands must be sent to the TRF to get it to activate:

```
//1st 01 0C 00 03 04 10 00 21 01 13 00 00

//2nd 01 09 00 03 04 F0 00 00 00

//3rd 01 09 00 03 04 F1 FF 00 00
```

These commands were taken from the TRF7960 datasheet. However, after much work, the TRF was still not operating properly. The main reason for this is time restraints as the transmitter board was not populated until 1 day before the project was due.

The Timer

The Timer is a combination of variable handling and a Timer A interrupt. The main part of the timer is insuring that the variables used to track seconds, minutes, and hours overflow correctly. For example, seconds overflow at 60, minutes overflow at 60, and hours do not overflow. This was implemented using a simple function:

```
void tick(){
  //overflow seconds
  timestring[7]+=1;  //tick seconds

  if(timestring[7]>'9'){ //overflow seconds digit 1
   timestring[7]='0';
   timestring[6]+=1;
  }
  if(timestring[6]=='6'){ //overflow seconds digit 2
    timestring[7]='0';
   timestring[6]='0';
   timestring[4]+=1;
  }
  if(timestring[4]>'9'){ //overflow minutes digit 1
   timestring[4]='0';
   timestring[3]+=1;
  }
   if(timestring[3]=='6'){ //overflow minutes digit 2
    timestring[3]='0';
   timestring[4]='0';
   timestring[1]+=1;
  }
  if(timestring[1]>'9'){ //overflow hours digit 1
   timestring[1]='0';
   timestring[0]+=1;
  }
}
```

The other main part of the timer was the interrupt. It is based off of Timer a:

```
CCTL0 = CCIE;                             // CCR0 interrupt enabled
CCR0 = 32768;
TACTL = TASSEL_1 + MC_1;                  // INCLK, upmode

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{

  #ifdef RECEIVER
      lcdhome();
     LcdWriteString("Event 1");
     LcdSecondLine();

    tick();
    char * time = timestring;
    LcdWriteString(time);
  #endif
```

The receiver board has a 32,768 Hz crystal, Therefore the CCR0 register was set to the clock speed so that it will overflow every second. The interrupt is set to actiavte upon CCR0 overflow, so therefore it is a 1 second recurring interrupt. It is used to tick the clock and update the LCD.

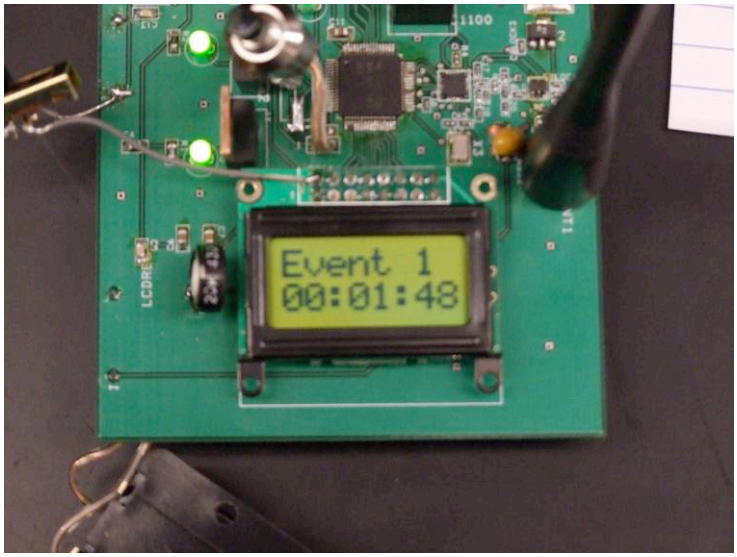


Figure 9 - LCD with Timer

Testing and Problems

Due to last minute population and programming, some of the testing is not as complete as desired. The main problem areas were the TRF7960, the integrated CC1100 on the receiver board, and the power amplifier.

As previously mentioned, the TRF7960 was not successfully programming. The TRF had several pins of importance, including an enable pin, and a clock pin. The enable pin was basically the power on pin, and the TRF would only work while this pin is high. The clock pin was connected to a clock source. Oscilloscope testing confirmed that the MSP was enabling the correct pins on the TRF, and that its clock source was oscillating. It was unable to be determined why the TRF would not initialize correctly, it was probably a software problem. A similar problem was encountered with the LCD because a delay was not being followed between each command being loaded. However, the TRF datasheet made no mention of a requisite delay.

The integrated CC1100 on the receiver board was another problem area. The GDO0 pin, which is the pin that flags for received data, would never go high. This was probably due to a cold solder joint, but there was insufficient time to correct this problem.

The power amplifier turned out to be another problem area. Initial testing confirmed that it did in fact work to amplify the signal.

Figure 10- Power Amplifier Power Spectrum [5]

However, upon attempting to use the power amplifier, no transmissions were picked up on the spectrum analyzer. It was suspected that the RF switches were malfunctioning and they were removed. On subsequent removal, and after bypassing the power amplifier, the CC110s on both transmitter boards were functioning correctly.

Another area that was heavily tested before programming commenced was the RFID antenna. As you can see in the next shot, the RFID tags are successfully coupling to the antenna,

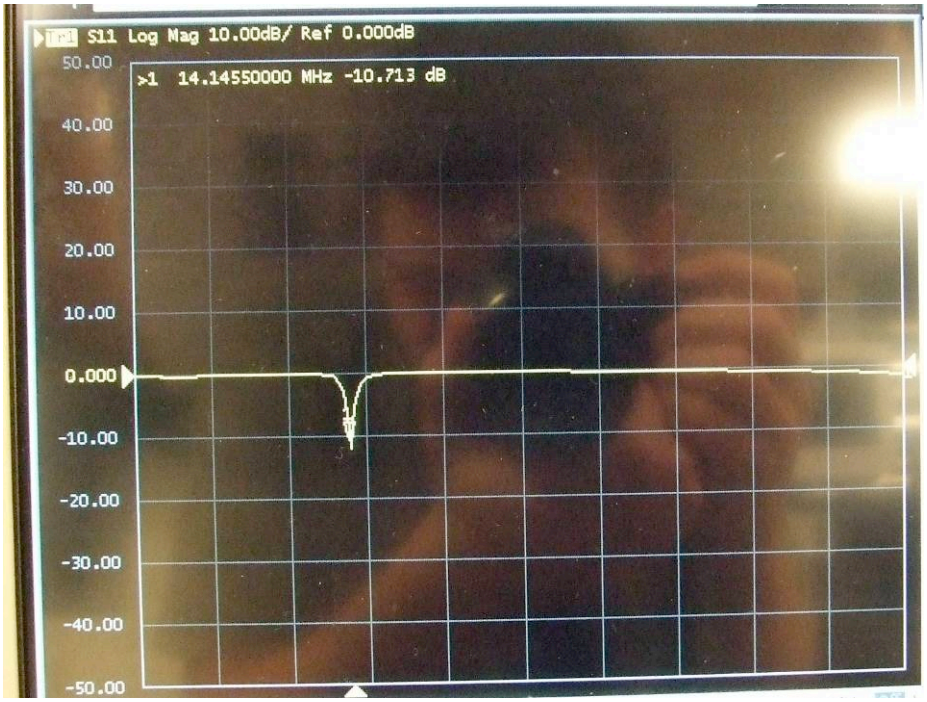


Figure 11- RFID Tag [5]

However, since the TRF7960 was not successfully programmed, the antenna was of no use.

Uncompleted

Due to time restraints and delays in board design and population, coding was left until the last minute and remains incomplete. The original design for the software is illustrated in the following diagram:

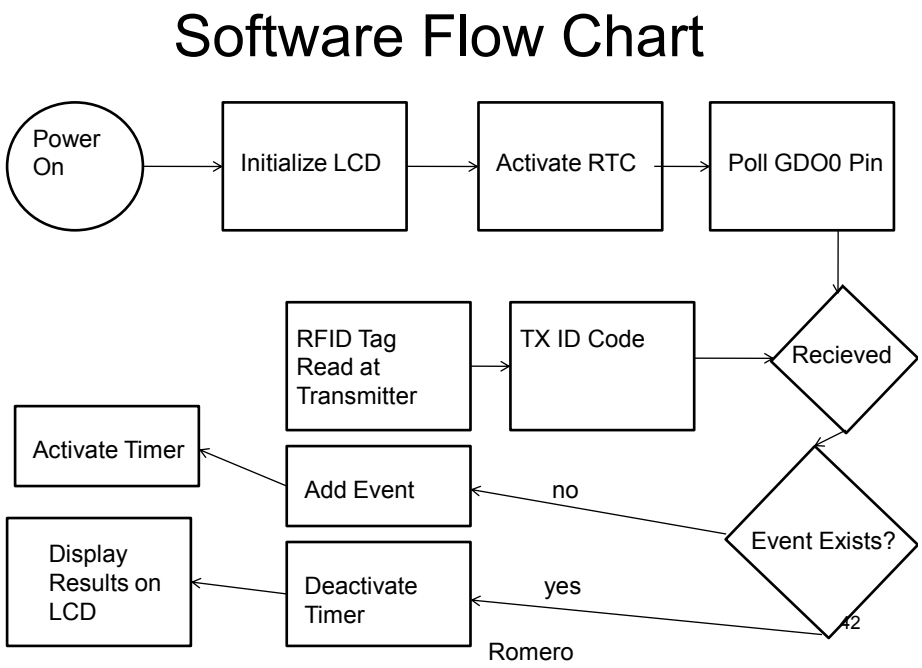


Figure 12 - Software Flow Chart

The TRF7960 and antenna would have been used to read an id number from a RFID tag that is passed over this. The transmitter board would then transmit the id number in a packet to the receiver board. The receiver board would check for the existence of an even with this id number. If there is none, it would activate new event. If it did exist, it would stop the timer for this event. When all events are deactivated, the results will be displayed on the LCD at 5s intervals. This method would have allowed many events to be tracked simultaneously, and would have made full use of the boards' capabilities.

Conclusion

At the time this project s due, it shows great potential. However, it still has much work to be done to it to make it a functioning system. The two transmitter boards successfully can communicate between each other, which mean that the CC1100s are populated and connected correctly. The problems on the receiver board are most likely just due to soldering problems.

The timer works, and is accurate to the second for over an hour in testing conditions, and is successfully displayed on the LCD. If more time were available, the main things that would be corrected are:

1.) The TRF7960 must be programmed correctly

2.) The receiver board's errors must be corrected, i.e. the CC1100 must work.

3.) The RF switches must be replaced and operational for the power amplifier to work.

The main reason why this project fell behind was shifting levels of complexity. The initial project description was quite simple. However, when it was discovered that the preliminary requirements could be completed quickly, new things were added to make the project more interesting and complex. This caused indecision among group members as to how to proceed, and wasted time.

In the end however, everything except the RFID worked, and the project demonstrated that a remote event timer could in fact be implemented, so this project is a partial success.

Works Cited

1. CC1100.pdf, “CC1100 Datasheet”, TI, 3/2/2008
2. http://www.tdc.co.uk/index.php?key=chipcon, “TDC.CO.UK, Chipcon Products”
3. Will Gray, “Weekly Presentations” , 4/29/2008
4. Aaron Adcock, “Weekly Presentations”, 4/29/2008
5. Ronjan Mathur, “Weekly Presentations”, 4/29/2008
6. Http://www.wikipedia.com , “Wikiedia”, 3/2/2008
7. http://wikipedia.com/, bit-banging, 4/29/08

Appendix A: Pin Outs

MSP430 Pin-out Assignments – Receiver Board

MSP430 Pin#	MSP430 Port#	Device	Description	Device Pin#
1.	DVcc	POWER	+3.3V	N/A
2.	P6.3	N/A	N/A	N/A
3.	P6.4	N/A	N/A	N/A
4.	P6.5	N/A	N/A	N/A
5.	P6.6	N/A	N/A	N/A
6.	P6.7	N/A	N/A	N/A
7.	Vref+	N/A	N/A	N/A
8.	XIN	32KHz XTAL	XTAL	XTAL_IN
9.	XOUT	32KHz XTAL	XTAL	XTAL_OUT
10.	VeRef+	N/A	N/A	N/A
11.	Vref-/VeRef-	N/A	N/A	N/A
12.	P1.0	N/A	N/A	N/A
13.	P1.1	N/A	N/A	N/A
14.	P1.2	N/A	N/A	N/A
15.	P1.3	N/A	N/A	N/A
16.	P1.4	Pushbutton	+3.3V input	N/A
17.	P1.5	N/A	N/A	N/A
18.	P1.6	N/A	N/A	N/A
19.	P1.7	N/A	N/A	N/A
20.	P2.0	LCD	RS	LCD_4
21.	P2.1	LCD	R/W	LCD_5
22.	P2.2	LCD	R/W_EN	LCD_6
23.	P2.3	LCD	DB0	LCD_7
24.	P2.4	LCD	DB1	LCD_8
25.	P2.5	LCD	DB2	LCD_9
26.	P2.6	LCD	DB3	LCD_10
27.	P2.7	LCD	DB4	LCD_11
28.	P3.0	LCD	DB5	LCD_12
29.	P3.1	LCD	DB6	LCD_13
30.	P3.2	LCD	DB7	LCD_14
31.	P3.3	N/A	N/A	N/A
32.	P3.4	N/A	N/A	N/A
33.	P3.5	RFSW_1	P_AMP ON/OFF	RFSW_1_CTL
34.	P3.6	RFSW_2	REG_TX ON/OFF	RFSW_2_CTL
35.	P3.7	P_AMP	Power ON/OFF	N/A
36.	P4.0	POWER_LED_1	LED_1 ON/OFF	N/A
37.	P4.1	POWER_LED_2	LED_2 ON/OFF	N/A
38.	P4.2	N/A	N/A	N/A

39.	P4.3	N/A	N/A	N/A
40.	P4.4	N/A	N/A	N/A
41.	P4.5	N/A	N/A	N/A
42.	P4.6	CC1100	GDO2	CC1100_3
43.	P4.7	CC1100	GDO0	CC1100_6
44.	P5.0	CC1100	CSn	CC1100_7
45.	P5.1	CC1100	SI	CC1100_20
46.	P5.2	CC1100	SO/GDO1	CC1100_2
47.	P5.3	CC1100	SCLK	CC1100_1
48.	P5.4	N/A	N/A	N/A
49.	P5.5	N/A	N/A	N/A
50.	P5.6	N/A	N/A	N/A
51.	P5.7	N/A	N/A	N/A
52.	XT2OUT	N/A	N/A	N/A
53.	XT2IN	N/A	N/A	N/A
54.	TDO/TDI	JTAG	TDO/TDI	JTAG_1
55.	TDI/TCLK	JTAG	TDI/TCLK	JTAG_3
56.	TMS	JTAG	TMS	JTAG_5
57.	TCK	JTAG	TCK	JTAG_7
58.	RST/NMI	JTAG	RST/NMI	JTAG_11
59.	P6.0	N/A	N/A	N/A
60.	P6.1	N/A	N/A	N/A
61.	P6.2	N/A	N/A	N/A
62.	AVss	GND	GND	JTAG_9
63.	DVss	GND	GND	N/A
64.	AVcc	POWER	+3.3V	N/A

Appendix B: Registers

	Write		Read		
	Single byte	Burst	Single byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x00	IOCFG2				R/W configuration registers, burst access possible
0x01	IOCFG1				
0x02	IOCFG0				
0x03	FIFOTHR				
0x04	SYNC1				
0x05	SYNC0				
0x06	PKTLEN				
0x07	PKTCTRL1				
0x08	PKTCTRL0				
0x09	ADDR				
0x0A	CHANNR				
0x0B	FSCTRL1				
0x0C	FSCTRL0				
0x0D	FREQ2				
0x0E	FREQ1				
0x0F	FREQ0				
0x10	MDMCFG4				
0x11	MDMCFG3				
0x12	MDMCFG2				
0x13	MDMCFG1				
0x14	MDMCFG0				
0x15	DEVIATN				
0x16	MCSM2				
0x17	MCSM1				
0x18	MCSM0				
0x19	FOCCFG				
0x1A	BSCFG				
0x1B	AGCCTRL2				
0x1C	AGCCTRL1				
0x1D	AGCCTRL0				
0x1E	WOREVT1				
0x1F	WOREVT0				
0x20	WORCTRL				
0x21	FREND1				
0x22	FREND0				
0x23	FSCAL3				
0x24	FSCAL2				
0x25	FSCAL1				
0x26	FSCAL0				
0x27	RCCTRL1				
0x28	RCCTRL0				
0x29	FSTEST				
0x2A	PTEST				
0x2B	AGCTEST				
0x2C	TEST2				
0x2D	TEST1				
0x2E	TEST0				
0x2F					
0x30	SRES		SRES	PARTNUM	Command Strobes, Status registers (read only) and multi byte registers
0x31	SFSTXON		SFSTXON	VERSION	
0x32	SXOFF		SXOFF	FREQEST	
0x33	SCAL		SCAL	LQI	
0x34	SRX		SRX	RSSI	
0x35	STX		STX	MARCSTATE	
0x36	SIDLE		SIDLE	WORTIME1	
0x37	SAFC		SAFC	WORTIME0	
0x38	SWOR		SWOR	PKTSTATUS	
0x39	SPWD		SPWD	VCO_VC_DAC	
0x3A	SFRX		SFRX	TXBYTES	
0x3B	SFTX		SFTX	RXBYTES	
0x3C	SWORRST		SWORRST		
0x3D	SNOP		SNOP		
0x3E	PATABLE	PATABLE	PATABLE	PATABLE	
0x3F	TX FIFO	TX FIFO	RX FIFO	RX FIFO	

Table 31: SPI Address Space

Appendix C-Budget and Gantt

Estimated Budget (1/18/2008)

Labor Costs

Name	Hourly Pay Rate	Hours Worked	Total Pay
Ronjan Mathur	$15.00	210	$3,150.00
Aaron Adcock	$15.00	210	$3,150.00
Will Gray	$15.00	210	$3,150.00
Kyle Romero	$15.00	210	$3,150.00
		Total Labor Cost	$12,600.00
		75% Overhead	$9,450.00
		Total Labor and Overhead	$22,050.00

Parts Costs

Part Name	Cost	Quantity	Total Cost
Misc. Parts	$50.00	NA	$0.00
		Total Parts Cost	$0.00

Equipment Rental Costs

Equipment Name	Purchase Cost	Day Rental Rate	Days Rented	Total Cost
CC1100 x 3	$300.00	$0.60	60	$36.00
Development Board	?		60	
Experimenters Board	$100.00	$0.20	60	$12.00
Oscope	$8,000.00	$16.00	60	$960.00
			Total Rental Costs	$1,008.00

Total Estimated Budget $23,058.00

Weekly Budget	(4/24/2008)			
Labor Costs				
Name	**Hourly Pay Rate**	**Hours Worked**	**Total Pay**	
Ronjan Mathur	$15.00	150	$2,250.00	
Aaron Adcock	$15.00	150	$2,250.00	
Will Gray	$15.00	150	$2,250.00	
Kyle Romero	$15.00	150	$2,250.00	
		Total Labor Cost	$9,000.00	
		75% Overhead	$6,750.00	
		Total Labor and Overhead	$15,750.00	
Part Name	**Cost**	**Quantity**	**Total Cost**	
Board Printing	$33.00	3	$99.00	
Transmitter Board	$15.77	2	$31.54	
Receiver Board	$22.79	1	$22.79	
General Board Parts	$151.82	1	$151.82	
		Total Parts Cost	$305.15	
Equipment Rental Costs				
Equipment Name	**Purchase Cost**	**Day Rental Rate**	**Days Rented**	**Total Cost**
CC1100 x 3	$300.00	$0.60	98	$58.80
Experimenters Board	$100.00	$0.20	98	$19.60
Oscope	$8,000.00	$16.00	98	$1,568.00
			Total Rental Costs	$1,646.40
Budget Total	**$17,701.55**			

ID	Task Name	Start	Finish	Duration	% Complete	Resource Names
1	**Week 1**	**01/10/08**	**01/16/08**	**5d?**	**100**	
2	Gain understanding of the CC1100	01/10/08	01/16/08	5d?	100	Mathur
3	Research into Packet/Modulation Techniques	01/10/08	01/16/08	5d?	100	Adcock
4	Look into the Programming Suite (SmartRF)	01/10/08	01/16/08	5d?	100	Romero
5	Began work with the Experimenter Board	01/10/08	01/16/08	5d?	100	Gray
6	Researched FCC Bands	01/10/08	01/10/08	1d?	100	Gray
7	**Week 2**	**01/17/08**	**01/23/08**	**5d?**	**0**	
8	Attempt SmartRF communication with Experime...	01/17/08	01/23/08	5d?	0	Romero
9	Research Packet Creation	01/17/08	01/23/08	5d?	0	Adcock
10	Work on Transmitter Board	01/17/08	01/23/08	5d?	0	Gray
11	Review Sensor Setups	01/17/08	01/23/08	5d?	0	Mathur
12	**Week 3**	**01/24/08**	**01/30/08**	**5d?**	**0**	
13	Look into CC1100 Registers / MSP430	01/24/08	01/30/08	5d?	0	Romero
14	Begin Creating Custom Packets	01/24/08	01/30/08	5d?	0	Adcock
15	Begin Transmitting with Transmitter Board	01/24/08	01/30/08	5d?	0	Gray
16	Order Sensors	01/24/08	01/30/08	5d?	0	Mathur
17	**Week 4**	**01/31/08**	**02/06/08**	**5d?**	**0**	
18	Begin Timing Algorithm	01/31/08	02/06/08	5d?	0	Romero
19	Continue Working with Packets	01/31/08	02/06/08	5d?	0	Adcock
20	Begin working with MSP/Sensor Interface	01/31/08	02/06/08	5d?	0	Gray
21	Testing of Sensors	01/31/08	02/06/08	5d?	0	Mathur

ID	Task Name	Start	Finish	Duration	% Complete	Resource Names
22	**Week 5**	**02/07/08**	**02/13/08**	**5d?**	**0**	
23	Research Sensor ID # Techniques	02/07/08	02/13/08	5d?	0	Romero
24	Testing Sensors with Transmission	02/07/08	02/13/08	5d?	0	Mathur
25	Transfer Sensor Data to packets	02/07/08	02/13/08	5d?	0	Gray
26	Finalize Packet Setup	02/07/08	02/13/08	5d?	0	Adcock
27	**Week 6**	**02/14/08**	**02/20/08**	**5d**	**0**	
28	Begin testing packet Data	02/14/08	02/20/08	5d	0	Adcock
29	Begin Building Sensor/Transmitter Package	02/14/08	02/20/08	5d	0	Mathur
30	Begin Interpreting Packet Data	02/14/08	02/20/08	5d	0	Romero
31	Begin Power algorithm	02/14/08	02/20/08	5d	0	Gray
32	**Week 7**	**02/21/08**	**02/27/08**	**5d**	**0**	
33	Begin Modulation Tests	02/21/08	02/27/08	5d	0	Adcock
34	Continue Building Sensor/Transmitter Package	02/21/08	02/27/08	5d	0	Mathur
35	Continue interpreting packet data	02/21/08	02/27/08	5d	0	Romero
36	Continue Power Algorithm	02/21/08	02/27/08	5d	0	Gray
37	**Week 8**	**02/28/08**	**03/05/08**	**5d**	**0**	
38	Decide Modulation method	02/28/08	03/05/08	5d	0	Adcock
39	Finish Sensor/Transmitter Package	02/28/08	03/05/08	5d	0	Mathur
40	Finish Timing Algorithm	02/28/08	03/05/08	5d	0	Romero
41	Finish Power Algorithm	02/28/08	03/05/08	5d	0	Gray
42	**Week 9**	**03/06/08**	**03/12/08**	**5d**	**0**	
43	Test Packet Transmission (Separate)	03/06/08	03/12/08	5d	0	Adcock
44	Test Power Algorithm (Separate)	03/06/08	03/12/08	5d	0	Gray
45	Test Timing Algorithm (Separate)	03/06/08	03/12/08	5d	0	Romero
46	Design Full System Test	03/06/08	03/12/08	5d	0	Mathur
47	**Week 10**	**03/13/08**	**03/26/08**	**10d**	**0**	
48	Test Packet Transmission (Full System)	03/20/08	03/26/08	5d	0	Adcock
49	Test Power Algorithm (Full System)	03/20/08	03/26/08	5d	0	Gray
50	Test Timing/Reciever Algorithms (Full System)	03/20/08	03/26/08	5d	0	Romero
51	Test Sensors (Full System)	03/20/08	03/26/08	5d	0	Mathur
52	**Week 11**	**03/13/08**	**03/19/08**	**5d**	**0**	
53	Spring Break!	03/13/08	03/19/08	5d	0	All
54	**Week 12**	**03/27/08**	**04/02/08**	**5d**	**0**	
55	Fix Packet Problems	03/27/08	04/02/08	5d	0	
56	Fix Power Problems	03/27/08	04/02/08	5d	0	
57	Fix Reciever Problems	03/27/08	04/02/08	5d	0	
58	Fix Sensor Problems	03/27/08	04/02/08	5d	0	
59	**Week 13**	**04/03/08**	**04/09/08**	**5d**	**0**	
60	Continue System Testing	04/03/08	04/09/08	5d	0	Adcock
61	Continue System Testing	04/03/08	04/09/08	5d	0	Gray
62	Continue System Testing	04/03/08	04/09/08	5d	0	Romero
63	Continue System Testing	04/03/08	04/09/08	5d	0	Mathur
64	**Week 14**	**04/10/08**	**04/16/08**	**5d**	**0**	
65	Prepare Demonstration	04/10/08	04/16/08	5d	0	Romero
66	Make Poster For Demonstration	04/10/08	04/16/08	5d	0	Gray
67	Work out transmission errors	04/10/08	04/16/08	5d	0	Adcock
68	Work out Mechanical errors	04/10/08	04/16/08	5d	0	Mathur
69	**Week 15**	**04/17/08**	**04/23/08**	**5d**	**0**	
70	Demo Day!	04/17/08	04/23/08	5d	0	All
71	**Week 16**	**04/24/08**	**04/30/08**	**5d**	**0**	
72	Final Presentation/Report	04/24/08	04/30/08	5d	0	All

Mathur