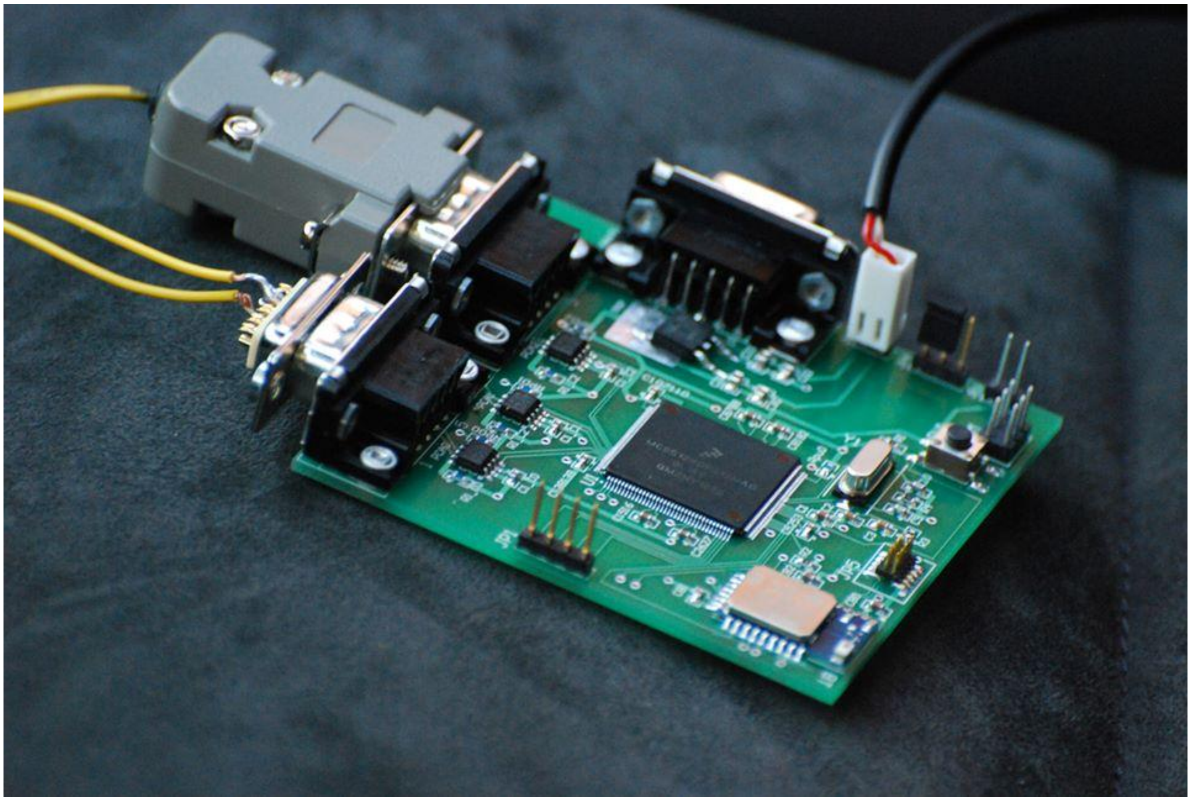
VW CAN

A BLE-enabled Android App for Sending CAN Commands



ME 218D – Smart Product Design Practice, Fall 2013

Stanford

Maxwell Wu, Erin Watson, Cliff Bargar

Executive Summary

This report describes our project for Smart Product Design Practice, also known as ME 218D, completed in the fall quarter of 2013.

As outlined in the following pages, we have created a complete system allowing a user to control various aspects of their car using their smartphone over Bluetooth. In particular, our system consists of an Android application which can run on any Bluetooth 4.0-enabled device and a fully-integrated PCB containing a Bluetooth module, microcontroller, voltage regulator, and CAN transceivers.

Acknowledgements

First and foremost we'd like to thank Professor Ed Carryer of the Stanford University Mechanical Engineering department for his guidance through not only this project but a full four quarters of mechatronics at Stanford.

We would also like to thank the Volkswagen ERL for sponsoring our project and Henry Chen in particular for providing lots of support and guidance.

Additionally, the team at Bay Area Circuits in Fremont, CA was very generous in quickly manufacturing our PCB prototypes.

Table of Contents

List of Figures

List of Tables

Problem Definition

The growing adoption of Bluetooth 4.0, which includes the standard known as Bluetooth Smart or Bluetooth Low Energy, allows for low power wireless communication between all sorts of devices. The BLE standard now makes it possible to create a wireless vehicle communication system which is always listening for devices without causing a substantial drain on the vehicle's battery.

Figure 1: Bluetooth Smart logo

Goal

The goal of the project is to create a system to communicate over Bluetooth Low Energy and control car functions through the CAN bus.

Specifications

To be effective, such a device must meet a number of specifications for both capabilities and power consumption.

Power Consumption

The device must be able to take a variable 9V to 14.4V input voltage from the car. Additionally it must draw 100µA or less while in standby mode.

Hardware

All components must be automotive grade and there must be a minimum of two CAN channels available for communications with the vehicle's various CAN buses. Additionally, there is an optional specification for an SD card which would contain the necessary CAN commands to be sent.

Communications

The device must be capable of sending commands at the various CAN bus speeds of 500, 250, and 100kbps, depending on which system it is communicating with. It also must take advantage of the Bluetooth Low Energy protocol.

Software

The device may use any microcontroller and can interface with Android, iOS, or both.

Communication Subsystems

As shown in Figure 2, Android communicates with the Bluetooth module over Bluetooth, the BT module then sends commands to the microprocessor through SPI, and the microprocessor communicates with the vehicle through the on-board CAN transceivers.

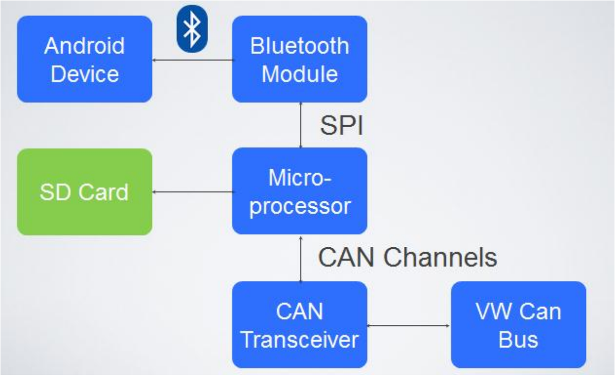


Figure 2: Communications block diagram

Hardware Selection

From the specifications, the primary concerns in sourcing hardware are low power consumption and being automotive grade. The major components that had to be determined are the microprocessor, the Bluetooth module, the voltage regulator, and the CAN transceivers and controllers.

For a full component listing refer to Appendix A: Final PCB - Bill of Materials.

Bluetooth Module

There are a number of Bluetooth Low Energy enabled modules on the market. We selected the Bluegiga BLE112 because it is automotive grade, has a 900nA sleep mode, and has a fairly easy to use API.

Figure 3: Bluegiga BLE112 Bluetooth module

Microprocessor

Several microprocessor families were considered, including those from Microchip, Atmel, and Texas Instruments. The processors in the Freescale MC9S12 family were the only ones found that have multiple CAN controllers integrated as peripherals on the chip. After testing with an evaluation board, we designed our PCB around the MC9S12XDP512 chip, which has 5 CAN controllers, 2 SPI channels, and is automotive grade. Additionally, it can be run at both 5V and 3.3V, which allows easier interfacing with the BLE112, which runs at 3.3V. Finally the XDP512 also has a low power mode which draws only 60nA of current.

CAN Transceiver

The Texas Instruments SN65HVD234DR was selected because it operates with 3.3V logic and has an enable pin, allowing for lower power consumption when not in use.

Voltage Regulator

After initially selecting a switching regulator, the TI TPS7A6533-Q1 was chosen instead for its smaller PCB footprint and lower power consumption. The regulator has a 3.3V output voltage when given an input between 3.6V and 40V, which is well within the specification. Additionally, the regulator draws only 25μA of quiescent current.

Prototype Development

The device went through three major rounds of prototyping. The first round was primarily using evaluation boards while the second and third rounds consisted of integrated custom PCBs. This strategy allowed the team to test the viability of each component while developing the necessary software for a full solution.

Initial Prototype

The main components of the initial prototype were the EVB9S12XDP512, the evaluation board for the microprocessor, and the BLE112 Development Kit; the full system is shown in Figure 4.

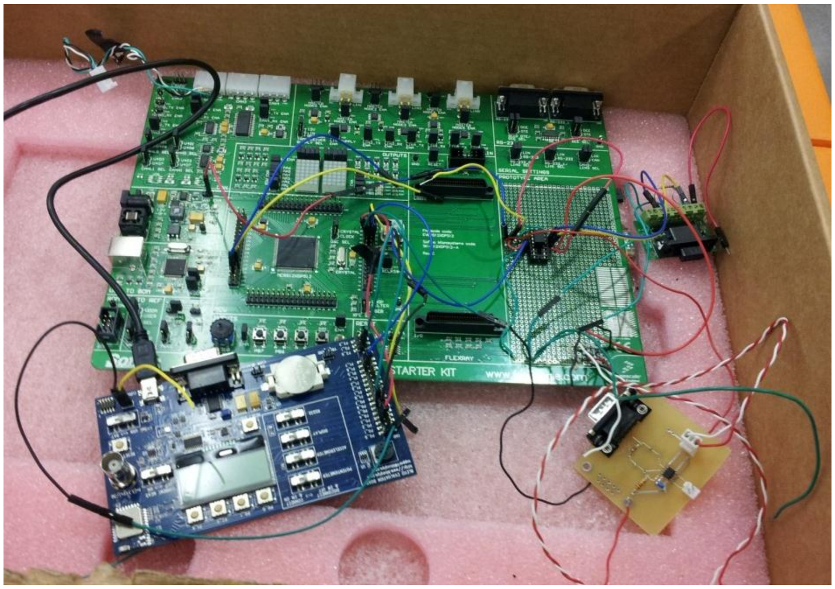


Figure 4: Initial prototype

Incremental PCB Design

While validating the initial prototype, we also designed PCBs for testing the BLE112 chip on its own as well as for testing the CAN transceivers connected directly to the DB9 header. The BLE112 test board is shown in Figure 5, while the CAN transceiver board is in the lower right corner of Figure 4.

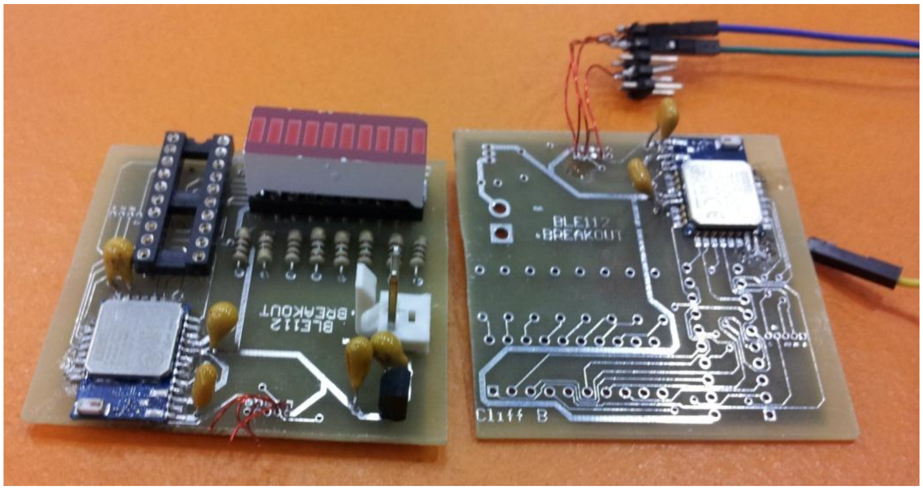


Figure 5: BLE112 test board

First PCB Iteration

The first PCB iteration combined all of the subsystems, along with a switching voltage regulator, into a package measuring roughly 4in by 3in. While testing this PCB it was determined that the switching regulator was difficult to use and took up too much PCB space. Additionally, there were a few misrouted pins, necessitating an additional round of PCB prototyping. The PCB is shown in Figure 6, while the schematic is in Appendix D: First PCB – Schematic and the circuit layout is in Appendix E: First PCB – Circuit Layout.

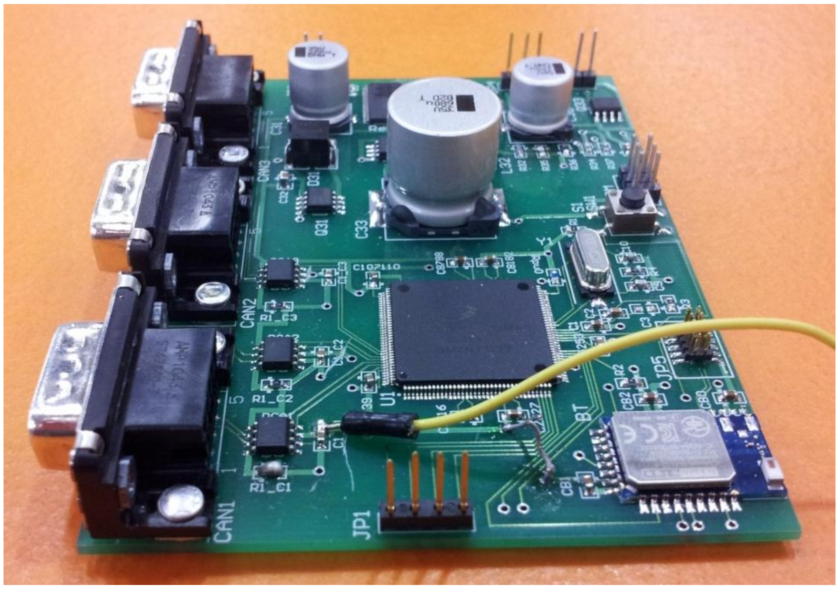


Figure 6: First PCB iteration

Second PCB Iteration

The second and final PCB iteration fixed the issues discovered in the first iteration while also including a low-dropout linear voltage regulator, cutting the final size down to a 3in by 3in footprint. The schematic and circuit layout are shown Appendix B: Final PCB – Schematic and Appendix C: Final PCB – Circuit Layout respectively. The manufactured PCB is shown in Figure 7.

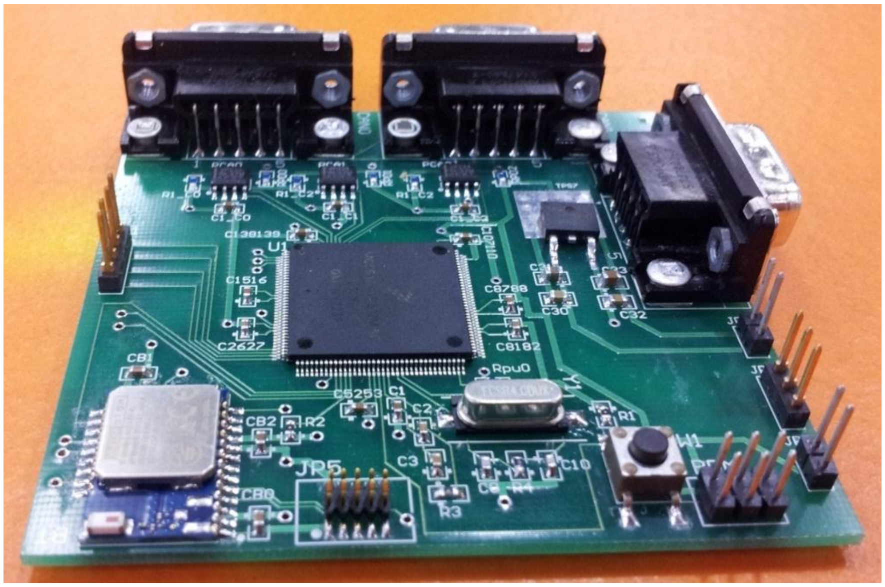


Figure 7: Second PCB iteration

Software Design

The software was split into three main areas – two sets of firmware running on the microcontroller and Bluetooth module and an application for the Android operating system.

MC9S12XDP512 Firmware

The microcontroller firmware is written in C using CodeWarrior, which is provided by Freescale. The firmware is capable of sending and receiving CAN messages at different speeds over different CAN buses. With a 4 MHz crystal, we were able to choose a multiplier with certain time quanta specified by the CAN protocol to make the XDP512 relay CAN messages at exactly 500, 250, and 100 kbps.

It is an SPI subordinate of the BLE112 and thus only reacts when it receives a message from the Bluetooth module. It sends CAN commands as instructed and relays relevant data back to the Android device via the BLE112. With every update and command message from the BLE112, the XDP512 returns any information that has changed since the last update.

The full listing is provided in Appendix F: MC9S12XDP512 Firmware Listing and is based on the EVB9S12XDP512's provided MSCAN sample code.

BLE112 Firmware

The Bluetooth module's firmware is written in BGScript, which is Bluegiga's scripting language combined with XML. The module is configured to communicate over Bluetooth with the Android device and send/receive SPI messages to/from the XDP512 as a master.

The BLE112 also automatically goes to sleep when not connected over Bluetooth, sending a message to the XDP512 to go to sleep as well. The Bluetooth module checks periodically for new devices; when a new connection is established it leaves sleep mode and sends an interrupt instructing the XDP512 to wake up, too. This process is further outlined in Minimizing Power Consumption on page 16.

To communicate over BLE, we also had to create a custom GATT profile that contained attributes specific to the CAN bus application. Each attribute was either read or write, depending on if it was designated for commands from the user to the CAN bus, or updates from the CAN bus to the user. The UUIDs of the profiles and attributes were chosen with no true significance.

Android Application

The Android app is written in Java using the Android SDK. Its graphical user interface displays data obtained from the car and allows the user to send commands to the BLE112. Two screenshots are shown in Figure 8.

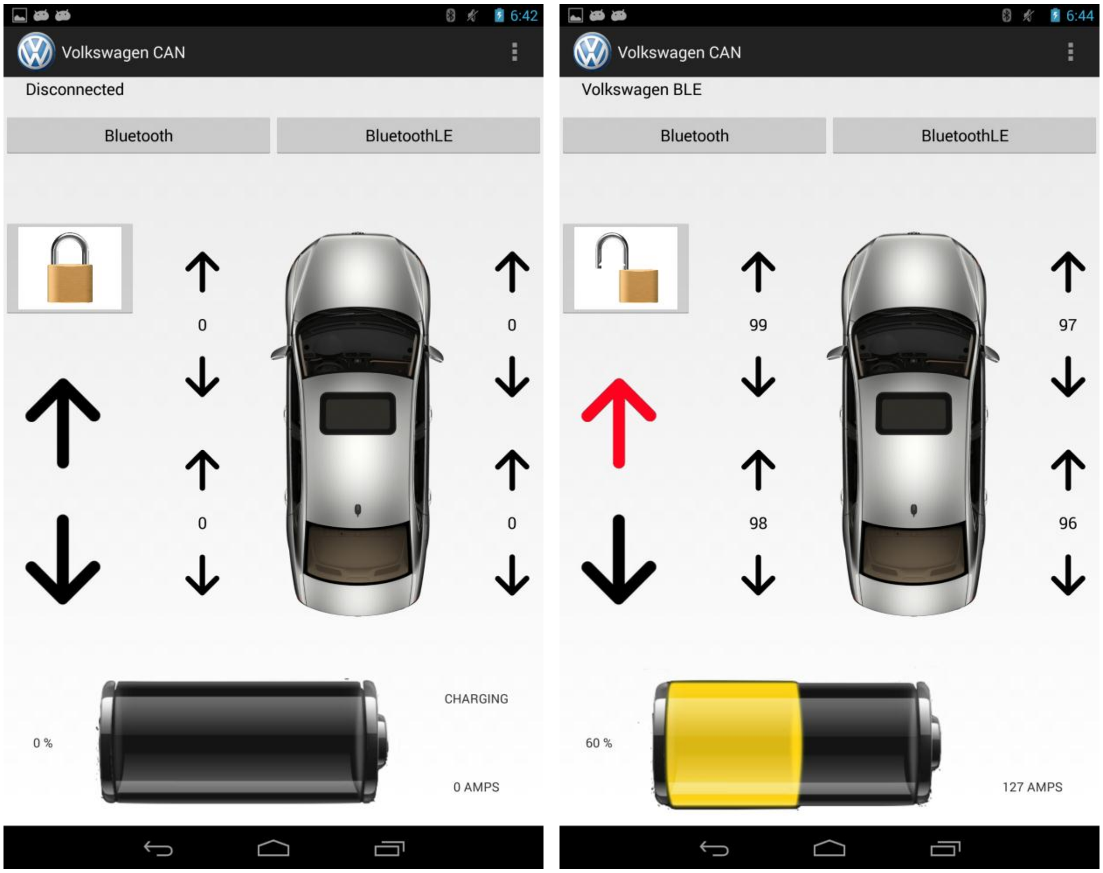


Figure 8: Android App screenshots

Commands and Data

The system is capable of relaying three main types of commands and data – lock/unlock doors, open/close windows, and battery status. As shown in Figure 8, the app is capable of commanding the system to lock or unlock all of the car’s doors, it can read the battery charge level and tell whether it’s charging and how much current is being drawn, and it can send commands to open/close either all four windows or one at a time while also relaying back the amount each window is open.

Minimizing Power Consumption

In order to minimize the system's power consumption and meet the current draw specification each individual component had to be selected and configured appropriately.

The BLE112 can be placed in "Power Mode 2" when asleep. In this mode it wakes up sporadically to check for nearby devices using its own sleep timer and draws only 900nA.

When it receives a "go to sleep" message, the MC9S12XDP512 has been programmed to go into "Stop Mode," which stops the clock and waits for an XIRQ/IRQ wake up interrupt. In this mode the XDP512 draws only 40µA of current. When the BLE112 connects to a device over Bluetooth it pulls up the XDP512's IRQ line to reawaken it.

Both the voltage regulator and the CAN Transceivers have been selected to minimize current draw. The TPS7A6533 provides 3.3V while drawing only 25µA of quiescent current. The HVD243DR CAN Transceiver can be put into sleep mode, drawing only 50nA of current.

Table 1: Current consumption

Device	**Max Current (active)**	**Current (sleep mode)**	**Implementation**
XDP512	110mA	<40µA	SPI message -> STOP command IRQ interrupt WakeUp
BLE112	27mA	0.9µA	Setting in API
HVD234DR (CAN Transceiver)	5mA (recessive), 45mA (dominant)	2µA	Pull enable pin low
TPS7A6533 (3.3V regulator)	4.85mA	25µA	Automatic (light loads, 8-18V input)
Total (predicted)	150-200mA	70µA	
Total (measured)	20mA	60µA	

Appendix A: Final PCB - Bill of Materials

Table 2: Bill of materials

	Part Name	Quantity	Designators	Footprint	Digikey Link
1	IC Microcontroller MC9S12XDP512	1	U1	QFP144	http://www.mouser.com/ProductDetail/Freescale-Semiconductor/MC9S12XDP512MAG/?qs=sGAEpiMZZMt7FrWooXVB15rynH6g0RpM
2	BLE112-A	1	BT	Custom	http://www.mouser.com/ProductDetail/Bluegiga-Technologies/BLE112-A/?qs=sGAEpiMZZMt1Kg3wsyvxe2VNUu4igQjl
3	DB9 Male	3	CAN0, CAN1, CAN2	DB9	http://www.digikey.com/product-detail/en/5747840-3/A32091-ND/808646
4	.1 inch headers (male)	2	BDM(2x3), JP4(1x3), JP1(1x4), JP2(1x2), JP3(1x2)	.1 inch spaced	http://www.digikey.com/product-detail/en/5-146868-1/A105161CT-ND/3440485
5	CC Debugger .05 pitch connector for BLE112	1	JP5	.05 inch pitch spaced SMT	http://www.digikey.com/product-detail/en/20021121-00010T4LF/609-3729-ND/2209075
6	4Mhz crystal	1	Y1	2 pad SMT	http://www.digikey.com/product-detail/en/ECS-40-20-5PXDU-TR/XC1524CT-ND/1693693
9	65HVD234DR CAN Transceiver	3	PCA0, PCA1, PCA2	SOT-8	http://www.digikey.com/product-detail/en/SN65HVD234DR/296-27991-1-ND/2451275
10	Pushbutton switch	1	SW1	4 pad SMT	http://www.digikey.com/product-detail/en/FSM4JSMA/450-1129-ND/525821
14	Capacitor C0G 10pF 0603	2	C9, C10	603	http://www.digikey.com/product-detail/en/CGA3E2NP02A100D080AA/445-12346-1-ND/3954012
16	Capacitor X7R 2.2nF 0603	1	C2	603	http://www.digikey.com/product-detail/en/CGA3E2X7R1H222K080AA/445-5660-1-ND/2443700
17	Capacitor X7R 22nF 0603	1	C3	603	http://www.digikey.com/product-detail/en/CGA3E2X7R1H223K080AD/445-8833-1-ND/3248141

20	Capacitor X7R .22uF 0603	3	C1516, C8788, C1	603	http://www.digikey.com/product-detail/en/CGA3E3X7R1V224K080AB/445-12551-1-ND/3954217
21	Capacitor X7R .1uF 0603	8	C1_C0, C1_C1, C1_C2, C2627, C5253, C107110, C138139, C8182	603	http://www.digikey.com/product-detail/en/CGA3E2X7R1E104K080AA/445-5667-1-ND/2443707
22	Capacitor X7R 1uF 0603	3	CB0, CB1, CB2	603	http://www.digikey.com/product-detail/en/CGA3E1X7R1E105K080AC/445-6931-1-ND/2672949
28	Resistor 10M 0603	1	R4	603	http://www.digikey.com/product-detail/en/CRCW060310M0JNEA/541-10MGCT-ND/1179465
29	Resistor 100k 0603	1	Rpu0, RPD0, PRD1, RPD2	603	http://www.digikey.com/product-detail/en/CRCW0603100KJNEAHP/541-100KSACT-ND/2222823
35	Resistor 1.8k 0603	3	R1, R2, R3	603	http://www.digikey.com/product-detail/en/ESR03EZPJ182/RHM1.8KDCT-ND/4053734
36	Resistor 100 0603	3	R1_C0, R1_C1, R0_C2	603	http://www.digikey.com/product-detail/en/MCT0603MC1000FP500/MCT0603-100-MFCT-ND/3883968
37	Capacitor 0.1 uF	2	C30, C32	805	http://www.digikey.com/product-detail/en/CL21B104KCC5PNC/1276-2447-1-ND/3890533
38	Capacitor 10uF 0805	1	C31	805	http://www.digikey.com/product-detail/en/JMK212B7106KG-T/587-2396-1-ND/2179009
39	Capacitor 4.7uF 0805	1	C33	805	http://www.digikey.com/product-detail/en/CGA4J1X7R1E475M125AC/445-12761-1-ND/3954427
40	3.3 Volt Regulator TPS7A6533	1	TPS7	T0-252-3	http://www.digikey.com/product-detail/en/TPS7A6533QKVURQ1/296-36857-2-ND/3305529
	Total	44			

Appendix B: Final PCB – Schematic

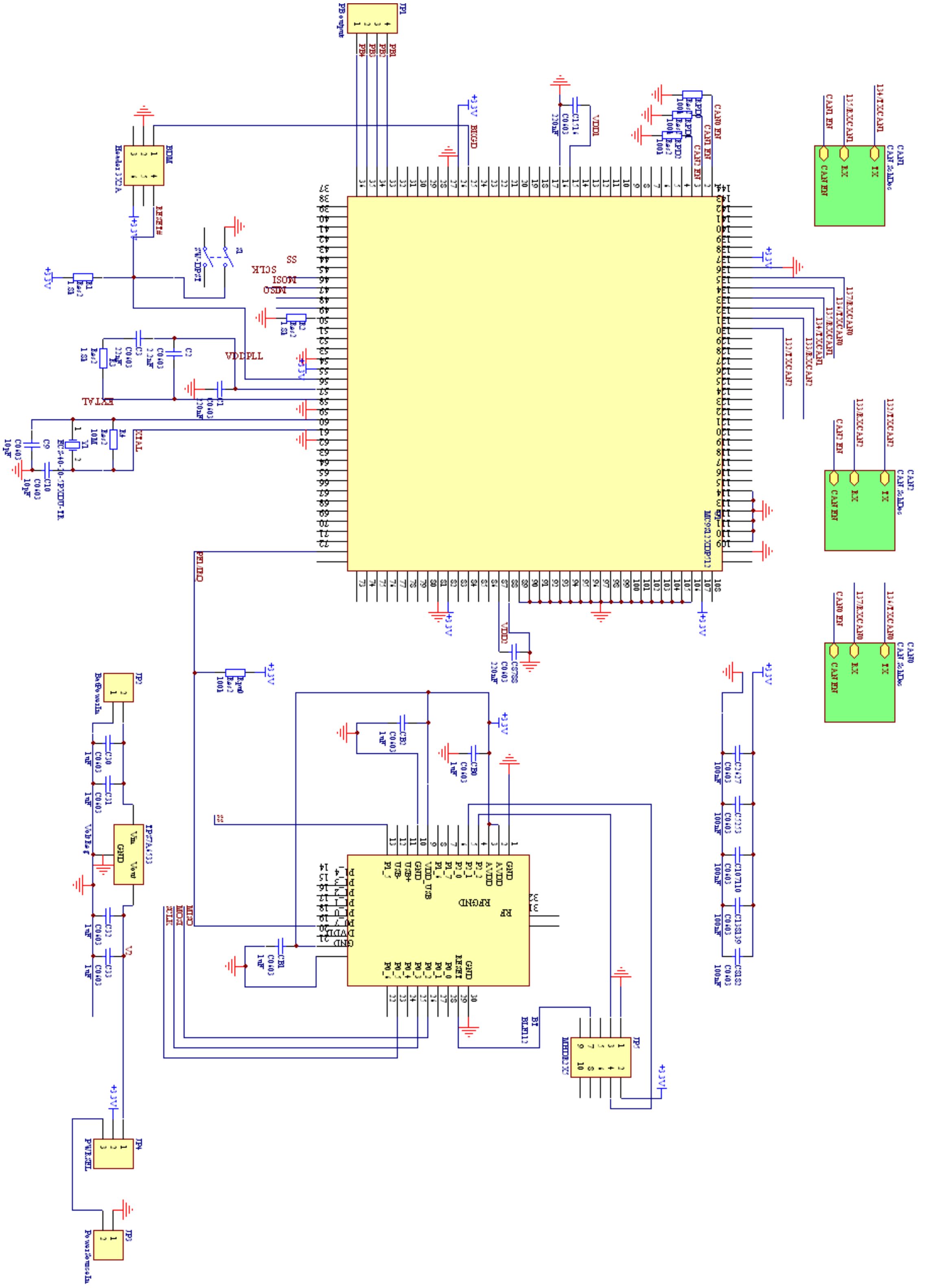


Figure 9: Schematic for final prototype

Appendix C: Final PCB – Circuit Layout

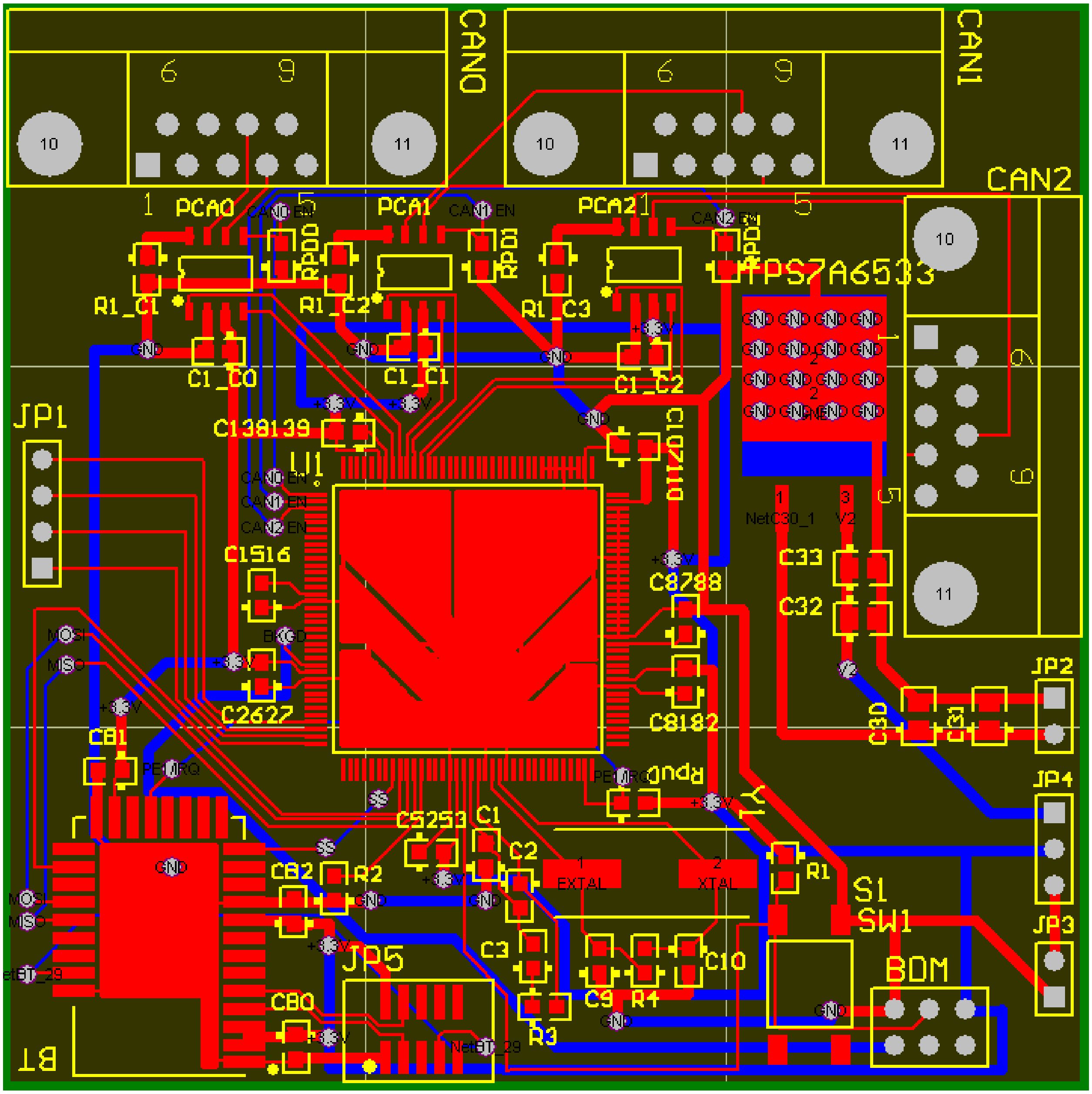


Figure 10: Circuit layout for final prototype

Appendix D: First PCB – Schematic

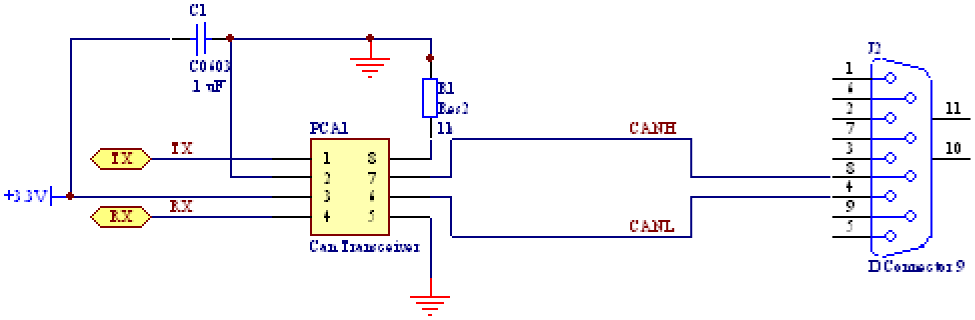


Figure 11: CAN Transceiver circuit

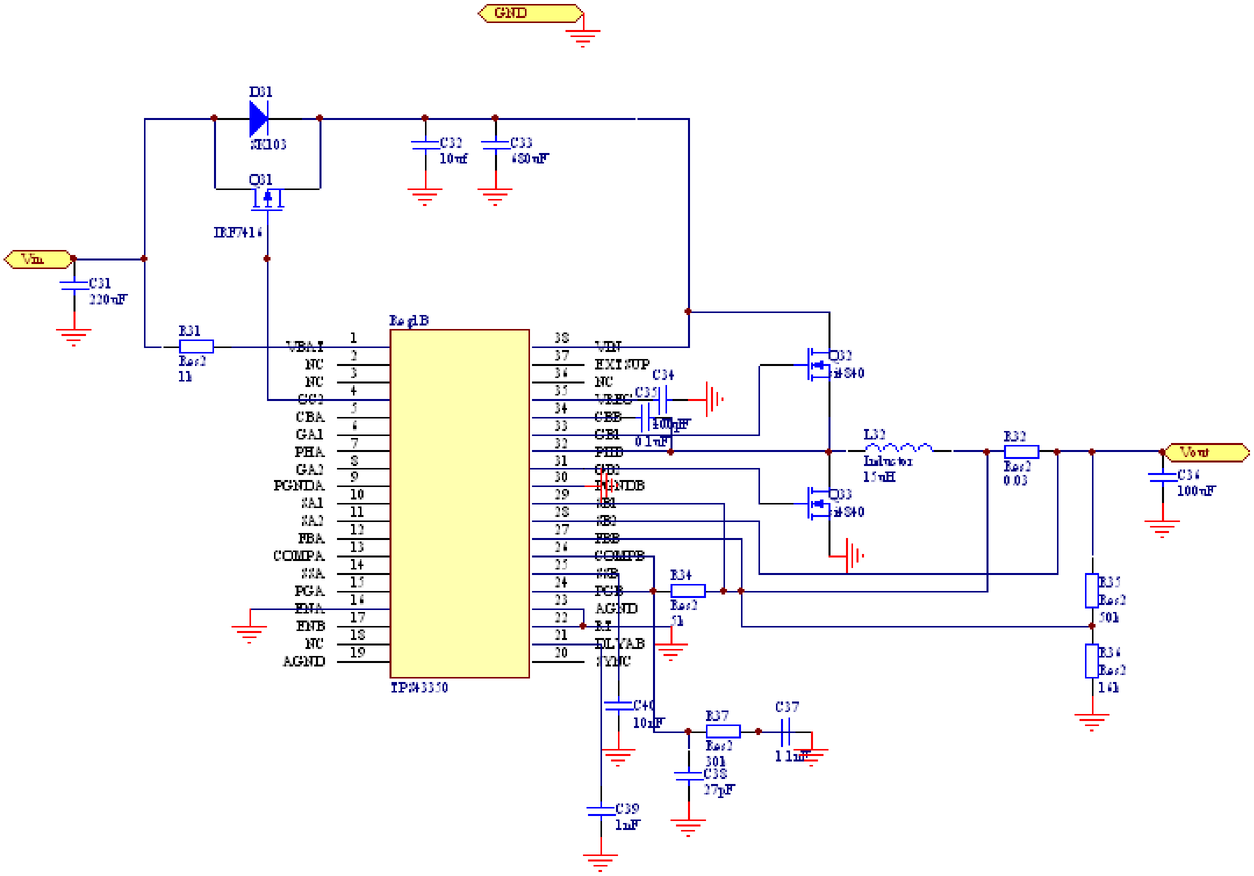


Figure 12: Switching regulator circuit

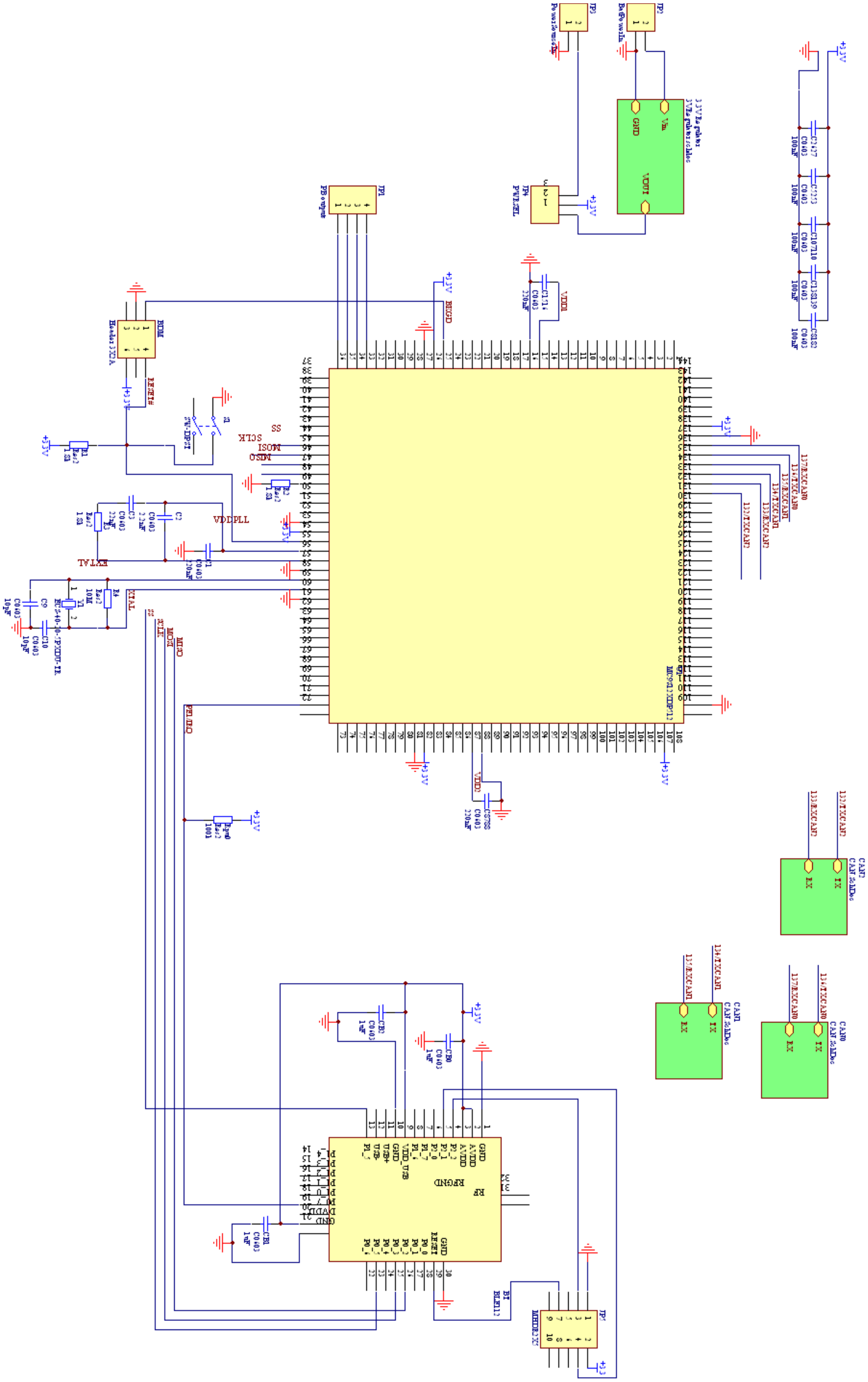


Figure 13: First PCB schematic

Appendix E: First PCB – Circuit Layout

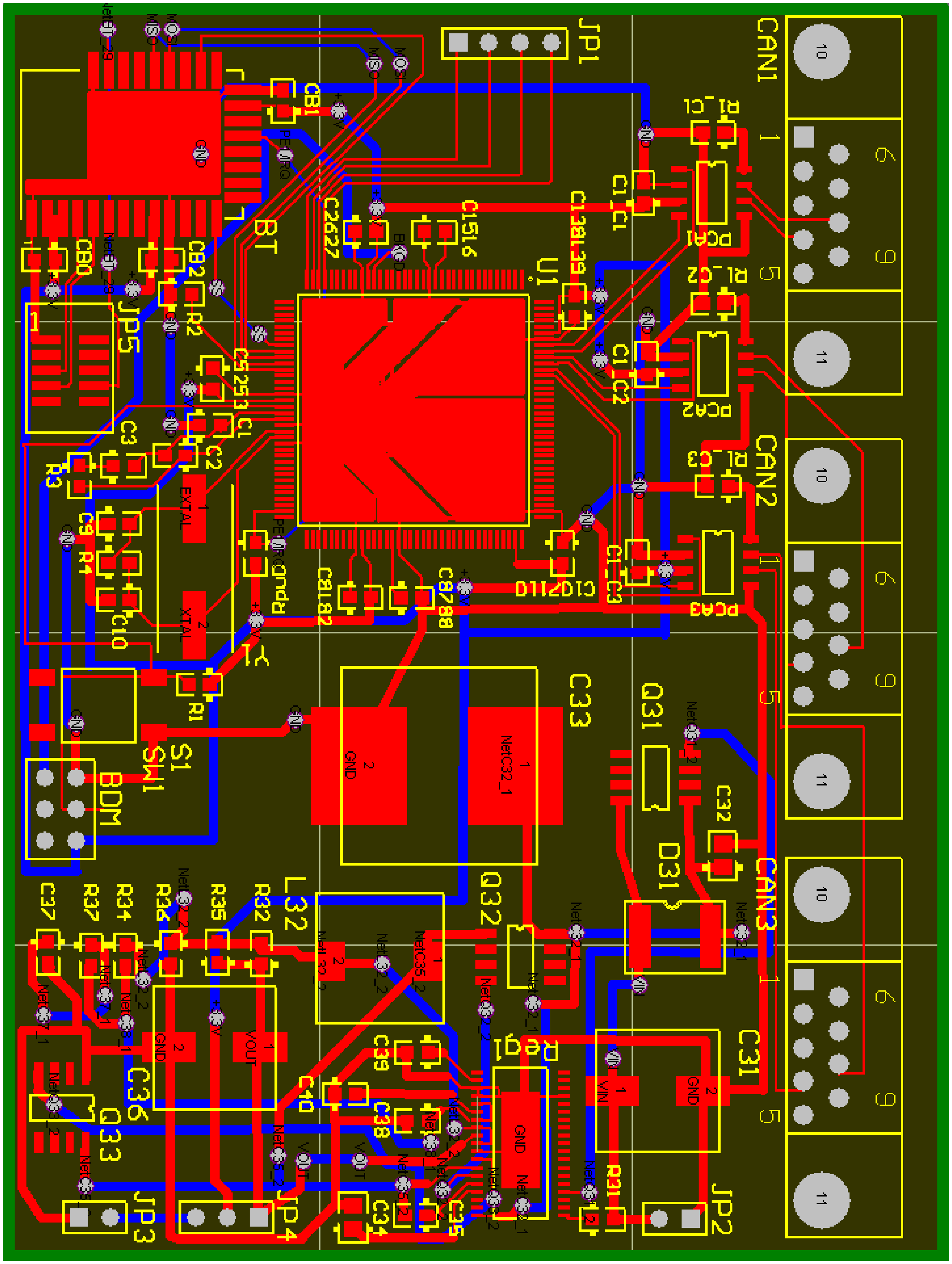


Figure 14: Circuit layout for first PCB

Appendix F: MC9S12XDP512 Firmware Listing

main.c

```
#include <hidef.h>
#include "mc9s12xdp512.h"
#include "mscan.h"
#include "SPI.h"
#include "stdio.h"
#include "BITDEFS.H"
#pragma LINK_INFO DERIVATIVE "mc9s12xdp512"

////////////////////////////////////////////////////////////////////////////////
// Defines and variables
////////////////////////////////////////////////////////////////////////////////

#define NUM_COLS          10
#define CAN_MSG_ID_DF       0x04
#define CAN_MSG_ID_DR       0x05
#define CAN_MSG_ID_PF       0x06
#define CAN_MSG_ID_PR       0x07
#define CAN_MSG_CH_LV       0x08
#define CAN_MSG_CH_ST       0x09

unsigned char LED_display_col;
unsigned char LED_matrix_data[NUM_COLS];
unsigned char potentiometer_value;
unsigned char can_delay = 10;

static int windowDF, windowDR, windowPF, windowPR, chargeLevel, chargeState, voltage, current;

struct can_msg WinOpen;
struct can_msg WinClose;
struct can_msg Lock;
struct can_msg Unlock;
struct can_msg DFU;
struct can_msg DRU;
struct can_msg PFU;
struct can_msg PRU;
struct can_msg DFD;
struct can_msg DRD;
struct can_msg PFD;
struct can_msg PRD;

struct can_msg CanMessages[12];
```

```
static void send_message(struct can_msg);
static void stopController(void);


////////////////////////////////////////////////////////////////////////////////
// Peripheral Initialization
////////////////////////////////////////////////////////////////////////////////

void WakeUpInit(void)
{
   IRQCR |= IRQCR_IRQE_MASK; //Sets INT with bits 7 and 8 high to enable IRQ interrupt and detect falling edges
   IRQCR |= IRQCR_IRQEN_MASK;
}

void MsgInit(void)
{
   WinOpen.RTR = WinClose.RTR = Lock.RTR = Unlock.RTR
   = DFU.RTR = DFD.RTR = DRU.RTR = DRD.RTR
   = PFU.RTR = PFD.RTR = PRU.RTR = PRD.RTR = FALSE;

   WinOpen.id = WinClose.id
   = DFU.id = DFD.id = DRU.id = DRD.id
   = PFU.id = PFD.id = PRU.id = PRD.id = 0x01;
   Lock.id = Unlock.id = 0x02;

   WinOpen.len = WinClose.len = Unlock.len = 1;
   DFU.len = DFD.len = DRU.len = DRD.len
   = PFU.len = PFD.len = PRU.len = PRD.len = 1;
   Lock.len = 1;

   WinOpen.prty = WinClose.prty = Lock.prty = Unlock.prty
   = DFU.prty = DFD.prty = DRU.prty = DRD.prty
   = PFU.prty = PFD.prty = PRU.prty = PRD.prty = 0;

   WinOpen.data[0] = 0x03;

   WinClose.data[0] = 0x02;

   Lock.data[0] = 0x01;

   Unlock.data[0] = 0x00;

   DFU.data[0] = 0x01;
   DFD.data[0] = 0x02;
   PFU.data[0] = 0x04;
   PFD.data[0] = 0x08;
   DRU.data[0] = 0x10;
```

```
    DRD.data[0] = 0x20;
    PRU.data[0] = 0x40;
    PRD.data[0] = 0x80;

    CanMessages[0] = WinOpen;
    CanMessages[1] = WinClose;
    CanMessages[2] = Lock;
    CanMessages[3] = Unlock; // = {WinOpen, WinClose, Lock, Unlock};
    CanMessages[4] = DFU;
    CanMessages[5] = DRU;
    CanMessages[6] = PFU;
    CanMessages[7] = PRU;
    CanMessages[8] = DFD;
    CanMessages[9] = DRD;
    CanMessages[10] = PFD;
    CanMessages[11] = PRD;
}

void PeriphInit(void)
{
    // Configures PA[7..0] port as output
    DDRA = 0xFF;
    PORTA = 0x00;

    // Configures PB[7..0] as output
    DDRB = 0xFF;
    PORTB = 0x00;

    // Enables pull-ups on PB port
    //PUCR |= 0x02;

    // Configures PC[7..0] port as output
    DDRC = 0xFF;
    PORTC = 0x00;


    // Configures PD[7..0] port as output
    DDRD = 0xFF;
    PORTD = 0x00;

    // Configures PE[7..2] port as output
    DDRE = 0xFC;
    PORTE = 0x00;

    // Configures PH[3..0] port as output
    DDRH = 0x0F;
    PTH = 0x00;
```

```
 // Configures PJ[7..0] port as output
 DDRJ = 0xFF;
 PTJ = 0x00;

 // Configures PK[7..0] port as output
 DDRK = 0xFF;
 PORTK = 0x00;

 // Configures PM[7..6] port as output
 DDRM = 0xC0;
 PTM = 0x00;

 // Configures PP[7..0] port as output
 DDRP = 0xFF;
 PTP = 0x0E;

 // Configures PS[7..0] port as output
 DDRS = 0xFF;
 PTS = 0x00;

 // Configures PT[7..0] port as output
 DDRT = 0xFF;
 PTT = 0x00;

 ATD0DIEN = 0xFF;
 ATD1DIEN0 = 0xFF;
 ATD1DIEN1 = 0xFF;


 // Configures the ATD peripheral
 // (16 conversions per sequence, 8 bit resolution, wrap around channel, continuous conversion)
/* ATD1CTL3 = 0x38;
 ATD1CTL4 = 0x80;
 ATD1CTL0 = 0x05;
 ATD1CTL2 = 0x80;
 ATD1CTL5 = 0x32;
   */
 // Configures the PIT (Periodic Interrupt Timer) to generate a periodic interrupt of 500us
 // (Interrupt on channel 0)
 PITCE = 0x01;
 PITINTE = 0x01;
 PITLD0 = 1000;
 PITCFLMT = 0xA0;



 MSCANInit(MSCAN_0);
 MSCANInit(MSCAN_1);
```

```
    MSCANInit(MSCAN_2);

    SPIInit();

    EnableInterrupts;
}

//////////////////////////////////////////////////////////////////////////////
// Outputs a graphical pattern on the displays
//////////////////////////////////////////////////////////////////////////////

/*void disp_light_pattern(unsigned char value)
{
    unsigned char i;

    for(i=0; i<5; i++)
    {
        LED_matrix_data[i] = light_table[value][i];
        LED_matrix_data[5 + i] = light_table[value][5 - i - 1];
    }
}    */

//////////////////////////////////////////////////////////////////////////////
// Entry point
//////////////////////////////////////////////////////////////////////////////

void main(void)
{
    static int count = 0;
    struct can_msg msg_get;
    unsigned char SPI_msg;
    static int lastWindowDF, lastWindowDR, lastWindowPF, lastWindowPR, lastChargeLevel, lastChargeState,
lastCurrent;

    int voltage = 0;

    windowDF = windowDR = windowPF = windowPR = chargeLevel = chargeState = current = 0;
    lastWindowDF = lastWindowDR = lastWindowPF = lastWindowPR = 0;
    PeriphInit();
    MsgInit();
    WakeUpInit(); //testing


    // Low voltage interrupt disable
    VREGCTRL &= ~VREGCTRL_LVIE_MASK;

    IRQCR &= ~IRQCR_IRQEN_MASK;
    PORTB |= BIT1HI;
```

```
  PORTB |= BIT2HI;
  PORTB &= ~BIT3HI;
  PORTB &= ~BIT4HI;

  stopController();
 // send_message(CanMessages[1]);

  while(TRUE)
  {

      // Checks if a message is received from MSCAN1 and send out over spi
     if(MSCANCheckRcvdMsg(MSCAN_0))
     {
        if(MSCANGetMsg(MSCAN_0, &msg_get))
        {
           if(msg_get.id == CAN_MSG_ID_DF){
             windowDF = msg_get.data[2]/2;
             //PORTB |= 0x04;

           }

           if(msg_get.id == CAN_MSG_ID_DR){
             windowDR = msg_get.data[2]/2;
             //PORTB |= 0x04;
           }

           if(msg_get.id == CAN_MSG_ID_PF){
             windowPF = msg_get.data[2]/2;
            // PORTB |= 0x04;
           }

           if(msg_get.id == CAN_MSG_ID_PR){
             windowPR = msg_get.data[2]/2;
             //PORTB |= 0x04;
           }


        }

     }

     if(MSCANCheckRcvdMsg(MSCAN_1))
     {
        if(MSCANGetMsg(MSCAN_1, &msg_get))
        {
           if(msg_get.id == CAN_MSG_CH_LV){
             chargeLevel = ((int)msg_get.data[7]*2)/5;
```

```
    }

    if(msg_get.id == CAN_MSG_CH_ST){
     current = ((msg_get.data[1] & 0xF0) >> 4) | (msg_get.data[2] << 4);
     current /= 4;
     current -= 511;
     if (current > 127) current = 127;
     if (current < -127) current = -127;


     chargeState = msg_get.data[4];
     chargeState >>= 4;
     chargeState &= 0x07;
    }

  }

}


//check if message is received from SPI1
if(SPI2SR & SPI2SR_SPIF_MASK)
{
 SPI_msg = SPI2DR;

 switch (SPI_msg) {

 case 0x56:
  stopController();
  break;

  PORTB ^= BIT3HI;
 case 0x80:
  if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
  {
    if (windowDF != lastWindowDF) {
     SPI2DR = windowDF;
     lastWindowDF = windowDF;
    } else {
     SPI2DR = 0xFF;
     //SPI2DR = 0x01;
    }
  }

  break;

 case 0x81:
  if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
```

```
     {
       if (windowDR != lastWindowDR) {
        SPI2DR = windowDR;
        lastWindowDR = windowDR;
       } else {
        SPI2DR = 0xFF;
        //SPI2DR = 0x02;
       }
     }
     break;

    case 0x82:
     if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
     {
       if (windowPF != lastWindowPF) {
        SPI2DR = windowPF;
        lastWindowPF = windowPF;
       } else {
        SPI2DR = 0xFF;
        //SPI2DR = 0x03;
       }
     }
     break;

    case 0x83:

     if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
     {
       if (windowPR != lastWindowPR) {
        SPI2DR = windowPR;
        lastWindowPR = windowPR;
       } else {
        SPI2DR = 0xFF;
        //SPI2DR = 0x04;
       }
     }
     break;

    case 0x84:

     if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
     {
       if (chargeLevel != lastChargeLevel) {
        SPI2DR = chargeLevel;
        lastChargeLevel = chargeLevel;
       } else {
        SPI2DR = 0xFF;
        //SPI2DR = 60;
```

```
           }
         }
         break;

        case 0x85:

         if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
         {
           if (chargeState != lastChargeState) {
            SPI2DR = chargeState;
            lastChargeState = chargeState;
           } else {
            SPI2DR = 0xFF;
            //SPI2DR = 0x05;
           }
         }
         break;

        case 0x86:

         if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
         {

           if (current != lastCurrent) {

            //If negative, make sure bit 8 is high
            if (current < 0) {
             current *= -1;
            }
            current &= ~BIT8HI;

            SPI2DR = current;
            lastCurrent = current;
           } else {
            SPI2DR = 0xFF;
            //SPI2DR = 127;
           }
         }
         break;

        case 0x87:
         if((SPI2SR & SPI2SR_SPTEF_MASK) != 0)
         {
           SPI2DR = 0x00;
         }

         break;
```

```
     default:
      break;

     }


     if( SPI_msg >= 0x02 && SPI_msg <= 0x10)
     {
      //disp_light_pattern((unsigned char)(SPI_msg));

      if (!can_delay) {

       send_message(CanMessages[(int)SPI_msg-2]);
       send_message(CanMessages[(int)SPI_msg-2]);
       send_message(CanMessages[(int)SPI_msg-2]);
      }
     }
    }
  }
}

static void send_message(struct can_msg message) {
 (void)MSCANSendMsg(MSCAN_0, message);
// (void)MSCANSendMsg(MSCAN_1, message);
 //(void)MSCANSendMsg(MSCAN_2, message);
 can_delay = 10;
}

static void stopController(void) {
 PTP = 0x00;
 PORTB &= ~(BIT0HI | BIT1HI);
 IRQCR |= IRQCR_IRQE_MASK; //Sets INT with bits 7 and 8 high to enable IRQ interrupt and detect falling edges
 IRQCR |= IRQCR_IRQEN_MASK;
 asm ANDCC #0x6F;  //ANDCC = Logical AND with CCR - S X H I N Z V C - Clears S and I
 asm STOP;

}

//////////////////////////////////////////////////////////////////////
// PIT0 Interrupt Service Routine
//////////////////////////////////////////////////////////////////////


#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt void IRQ_ISR(void)
{
  static int flag;
```

```
    PTP = 0x0E;
    //if(PORTB & BIT1HI == 1) PORTB &= ~ BIT1HI;
    //else
    if(flag == 1) {
      PORTB &= ~BIT2HI;
      flag = 0;
    } else {
      flag = 1;
      PORTB |= BIT2HI;
    }
   // IRQCR &= ~IRQCR_IRQEN_MASK;  testing

}

#pragma CODE_SEG __NEAR_SEG NON_BANKED
interrupt void PIT0_ISR(void)
{
    EnableInterrupts;
    PORTB ^= BIT1HI;

    if(can_delay > 0)
      --can_delay;

    PITTF = 0x01;
    return;
}



#pragma CODE_SEG DEFAULT
```

mscan.h

```
////////////////////////////////////////////////////////////////////////////////
//
// Sample for SofTec Microsystems SK-S12XDP512-A Starter Kit
// (Freescale code: EVB9S12XDP512)
//
// ----------------------------------------------------------------------------
//
// Copyright (c) 2005 SofTec Microsystems
// http://www.softecmicro.com/
//
////////////////////////////////////////////////////////////////////////////////

#include <hidef.h>

////////////////////////////////////////////////////////////////////////////////
```

```
// Defines
//////////////////////////////////////////////////////////////////////////////

#define MAX_TX_BUFFERS      3
#define MAX_RX_BUFFERS      5

#define MaskOR(A)          (0x01<<A)

#define MSCAN_0            0
#define MSCAN_1            1
#define MSCAN_2            2
#define MSCAN_3            3
#define MSCAN_4            4

#define CANCTL0            0x00
#define CANCTL0_INITRQ_MASK 0x01
#define CANCTL0_SYNCH_MASK  0x10

#define CANCTL1            0x01
#define CANCTL1_INITAK_MASK 0x01

#define CANBTR0            0x02
#define CANBTR1            0x03
#define CANRFLG            0x04
#define CANRFLG_RXF_MASK    0x01

#define CANRIER            0x05
#define CANTFLG            0x06
#define CANTIER            0x07
#define CANTARQ            0x08
#define CANTAAK            0x09
#define CANTBSEL           0x0A
#define CANIDAC            0x0B
#define CANMISC            0x0D
#define CANRXERR           0x0E
#define CANTXERR           0x0F
#define CANIDAR_1B         0x10    // First bank: 4 registers
#define CANIDMR_1B         0x14    // First bank: 4 registers
#define CANIDAR_2B         0x18    // Second bank: 4 registers
#define CANIDMR_2B         0x1C    // Second bank: 4 registers
#define CANRXIDR           0x20    // 4 registers
#define CANRXDSR           0x24    // 8 registers
#define CANRXDLR           0x2C
//#define TBPR             0x2D    // Not available for receive buffers
#define CANRXTSRH          0x2E
#define CANRXTSRL          0x2F
#define CANTXIDR           0x30    // 4 registers
#define CANTXDSR           0x34    // 8 registers
```

```
#define CANTXDLR          0x3C
#define CANTXTBPR         0x3D
#define CANTXTSRH         0x3E
#define CANTXTSRL         0x3F

struct can_msg {
   unsigned int id;
   Bool RTR;
   unsigned char data[8];
   unsigned char len;
   unsigned char prty;
};

////////////////////////////////////////////////////////////////////////////
// Functions
////////////////////////////////////////////////////////////////////////////

void MSCANInit(unsigned char can_num);
Bool MSCANSendMsg(unsigned char can_num, struct can_msg msg);
Bool MSCANGetMsg(unsigned char can_num, struct can_msg *msg);
Bool MSCANCheckRcvdMsg(unsigned char can_num);
```

mscan.c

```
#include "mc9s12xdp512.h"
#include "mscan.h"

////////////////////////////////////////////////////////////////////////////
// Variables
////////////////////////////////////////////////////////////////////////////

unsigned char *can_periph[5] = {
   &CAN0CTL0,
   &CAN1CTL0,
   &CAN2CTL0,
   &CAN3CTL0,
   &CAN4CTL0
};

////////////////////////////////////////////////////////////////////////////
// MSCAN Peripheral Initialization
////////////////////////////////////////////////////////////////////////////

void MSCANInit(unsigned char can_num)
{
   unsigned char *can_pt;
```

```
can_pt = can_periph[can_num];

// If MSCAN peripheral is not in Initialization Mode, enables the Inizialization Mode Request
if(!(can_pt[CANCTL1]&CANCTL1_INITAK_MASK))
    {
    can_pt[CANCTL0] = CANCTL0_INITRQ_MASK;
    while(!(can_pt[CANCTL1]&CANCTL1_INITAK_MASK))
        ;
    }

// Enables MSCAN peripheral and chooses Oscillator Clock, Loop Disabled and Normal Operation
can_pt[CANCTL1] = 0x80; //was 0x80


if (can_num == MSCAN_1){
  // Configures SJW = 3Tq and Prescaler = 1
  // 500 kbps
  can_pt[CANBTR0] = 0x80;

  // Configures One Sample, Time Segment 1 = 5Tq and Time Segment 2 = 2Tq
  can_pt[CANBTR1] = 0x14;

} else {
  // Configures SJW = 3Tq and Prescaler = 3
  // 100 kbps
  can_pt[CANBTR0] = 0x83;

  // Configures One Sample, Time Segment 1 = 6Tq and Time Segment 2 = 3Tq
  can_pt[CANBTR1] = 0x25;

}


// Disables all the Filters
can_pt[CANIDMR_1B+0] = 0xFF;
can_pt[CANIDMR_1B+1] = 0xFF;
can_pt[CANIDMR_1B+2] = 0xFF;
can_pt[CANIDMR_1B+3] = 0xFF;
can_pt[CANIDMR_2B+0] = 0xFF;
can_pt[CANIDMR_2B+1] = 0xFF;
can_pt[CANIDMR_2B+2] = 0xFF;
can_pt[CANIDMR_2B+3] = 0xFF;

// Restarts MSCAN peripheral and waits for Initialization Mode exit
can_pt[CANCTL0] = 0x00;
while(can_pt[CANCTL1]&CANCTL1_INITAK_MASK)
    ;
    // Waits for MSCAN synchronization with the CAN bus
    while(!(can_pt[CANCTL0]&CANCTL0_SYNCH_MASK))
        ;
}

////////////////////////////////////////////////////////////////////////////
// MSCAN Send Message Routine
////////////////////////////////////////////////////////////////////////////

Bool MSCANSendMsg(unsigned char can_num, struct can_msg msg)
{
    unsigned char n_tx_buf = 0, i;
    unsigned char *can_pt;
    int txcount;

    can_pt = can_periph[can_num];
    if(msg.len > 8)
        return(FALSE);
    if(!(can_pt[CANCTL0]&CANCTL0_SYNCH_MASK))
        return(FALSE);
    txcount = 0;
    while(!(can_pt[CANTFLG]&MaskOR(n_tx_buf)) && txcount++ < 30) {
        n_tx_buf = (n_tx_buf == MAX_TX_BUFFERS)? 0: (unsigned char)(n_tx_buf + 1);
    }
    can_pt[CANTBSEL] = MaskOR(n_tx_buf);
    can_pt[CANTXIDR+0] = (unsigned char)(msg.id>>3);
    can_pt[CANTXIDR+1] = (unsigned char)(msg.id<<5);
    if(msg.RTR)
        can_pt[CANTXIDR+1] |= 0x10;
    for(i = 0; i < msg.len; i++)
        can_pt[CANTXDSR+i] = msg.data[i];
    can_pt[CANTXDLR] = msg.len;
    can_pt[CANTXTBPR] = msg.prty;
    can_pt[CANTFLG] = MaskOR(n_tx_buf);
    return(TRUE);
}

////////////////////////////////////////////////////////////////////////////
// MSCAN Get Message Routine
////////////////////////////////////////////////////////////////////////////

Bool MSCANGetMsg(unsigned char can_num, struct can_msg *msg)
{
    unsigned char i;
    unsigned char *can_pt;

    can_pt = can_periph[can_num];
    if(!(can_pt[CANRFLG]&CANRFLG_RXF_MASK))
    return(FALSE);
  if(can_pt[CANRXIDR+1]&0x08)
    return(FALSE);
  msg->id = ((can_pt[CANRXIDR+0]<<3)&0x0700) | (unsigned char)(can_pt[CANRXIDR+0]<<3) | (unsigned char)(can_pt[CANRXIDR+1]>>5);
  if(can_pt[CANRXIDR+1]&0x10)
    msg->RTR = TRUE;
  else
    msg->RTR = FALSE;
  msg->len = can_pt[CANRXDLR];
  for(i = 0; i < msg->len; i++)
    msg->data[i] = can_pt[CANRXDSR+i];
  can_pt[CANRFLG] = CANRFLG_RXF_MASK;
  return(TRUE);
}

////////////////////////////////////////////////////////////////////////////
// MSCAN Check for Received Message Routine
////////////////////////////////////////////////////////////////////////////

Bool MSCANCheckRcvdMsg(unsigned char can_num)
{
  unsigned char *can_pt;

  can_pt = can_periph[can_num];
  if(can_pt[CANRFLG]&CANRFLG_RXF_MASK)
    return(TRUE);
  return(FALSE);
}
```

spi.h

```
#include <hidef.h>

////////////////////////////////////////////////////////////////////////////
// Functions
////////////////////////////////////////////////////////////////////////////
void SPIInit( void);
Bool SPICheckReceive( void);
unsigned char SPIGetMsg( void);
```

spi.c

```
#include "mc9s12xdp512.h"
#include "BITDEFS.h"
```

```
#include "SPI.h"


///////////////////////////////////////////////////////////////////////////
// SPI Peripheral Initialization
///////////////////////////////////////////////////////////////////////////

//initializes SPI1 to the desired settings for communicating as a slave to the BLE112 module
void SPIInit()
{
  //SPI control reg 1
  //SPI1CR1 |= SPI1CR1_SPIE_MASK; //turn on SPI interrupt enable

  /*SPI1CR1 |= SPI1CR1_SPE_MASK; //turn on SPI enable
  SPI1CR1 &= ~SPI1CR1_MSTR_MASK; //set as a slave
  //clock polarity: p.24 of the BLE112 developer's guide has clock polarity as positive
  SPI1CR1 &= ~SPI1CR1_CPOL_MASK; //set active high
  //clock phase: p.24 of the BLE112 developer's guide has clock phase as 1
  SPI1CR1 |= SPI1CR1_CPHA_MASK; //set to odd edges
  //bit order: p.24 has endianness = MSB
  SPI1CR1 &= ~SPI1CR1_LSBFE_MASK; //MSB first (?)

  //SPI1BR = 0x65;

  //SPI control reg 2
  SPI1CR2 &= ~SPI1CR2_SPC0_MASK; //not bidirectional

  //per Erin's soldering, use PH0-4 as MISO1, MOSI1, SCK1, SS1#
  //note: not using the soldered ones --> PP0-4
  MODRR |= MODRR_MODRR5_MASK; //from table 22-38   **SPI1 - PH4-PH7**

//  MODRR |= MODRR_MODRR5_MASK; //from table 22-38 **SPI1 - PH0-PH3**
  DDRH |= BIT0HI; //set PH0 to output (MISO1)
  DDRH &= BIT1LO; //set PH1 to input (MOSI1)
  DDRH &= BIT2LO; //set PH2 to input (SCK1)
  DDRH &= BIT3LO; //set PH3 to input (SS1)
  //DDRP |= BIT0HI; //set PP0 to output (MISO1)
  //DDRP &= BIT1LO; //set PP1 to input (MOSI1)
  //DDRP &= BIT2LO; //set PP2 to input (SCK1)
  //DDRP &= BIT3LO; //set PP3 to input (SS1)
   */

  MODRR =0x60;
  SPI2CR1 |= SPI2CR1_SPE_MASK; //turn on SPI enable
  SPI2CR1 &= ~SPI2CR1_MSTR_MASK; //set as a slave
  SPI2CR1 &= ~SPI2CR1_CPOL_MASK; //set active high
  SPI2CR1 |= SPI2CR1_CPHA_MASK; //set to odd edges
  SPI2CR1 &= ~SPI2CR1_LSBFE_MASK; //MSB first (?)
  SPI2CR2 &= ~SPI2CR2_SPC0_MASK; //not bidirectional




}

////////////////////////////////////////////////////////////////////////////////
// MSCAN Get Message Routine
////////////////////////////////////////////////////////////////////////////////
unsigned char SPIGetMsg()
{
  return SPI1DR;
}


////////////////////////////////////////////////////////////////////////////////
// SPI Check for Received Message Routine
////////////////////////////////////////////////////////////////////////////////

Bool SPICheckReceive()
{
  if(SPI1SR & SPI1SR_SPIF_MASK)
    return(TRUE);
  return(FALSE);
}
```

isr_vectors.c

```
extern void near _Startup(void);       /* Startup routine */
extern void near IRQ_ISR(void);

#pragma CODE_SEG __NEAR_SEG NON_BANKED /* Interrupt section for this module. Placement will be in NON_BANKED area. */
__interrupt void UnimplementedISR(void)
{
   /* Unimplemented ISRs trap.*/
   asm BGND;
}

typedef void (*near tIsrFunc)(void);
const tIsrFunc _vect[] @0xFF80 = {     /* Interrupt table */
        UnimplementedISR,              /* vector 63 */
        UnimplementedISR,              /* vector 62 */
        UnimplementedISR,              /* vector 61 */
        UnimplementedISR,              /* vector 60 */
```

```
    UnimplementedISR,           /* vector 59 */
    UnimplementedISR,           /* vector 58 */
    UnimplementedISR,           /* vector 57 */
    UnimplementedISR,           /* vector 56 */
    UnimplementedISR,           /* vector 55 */
    UnimplementedISR,           /* vector 54 */
    UnimplementedISR,           /* vector 53 */
    UnimplementedISR,           /* vector 52 */
    UnimplementedISR,           /* vector 51 */
    UnimplementedISR,           /* vector 50 */
    UnimplementedISR,           /* vector 49 */
    UnimplementedISR,           /* vector 48 */
    UnimplementedISR,           /* vector 47 */
    UnimplementedISR,           /* vector 46 */
    UnimplementedISR,           /* vector 45 */
    UnimplementedISR,           /* vector 44 */
    UnimplementedISR,           /* vector 43 */
    UnimplementedISR,           /* vector 42 */
    UnimplementedISR,           /* vector 41 */
    UnimplementedISR,           /* vector 40 */
    UnimplementedISR,           /* vector 39 */
    UnimplementedISR,           /* vector 38 */
    UnimplementedISR,           /* vector 37 */
    UnimplementedISR,           /* vector 36 */
    UnimplementedISR,           /* vector 35 */
    UnimplementedISR,           /* vector 34 */
    UnimplementedISR,           /* vector 33 */
    UnimplementedISR,           /* vector 32 */
    UnimplementedISR,           /* vector 31 */
    UnimplementedISR,           /* vector 30 */
    UnimplementedISR,           /* vector 29 */
    UnimplementedISR,           /* vector 28 */
    UnimplementedISR,           /* vector 27 */
    UnimplementedISR,           /* vector 26 */
    UnimplementedISR,           /* vector 25 */
    UnimplementedISR,           /* vector 24 */
    UnimplementedISR,           /* vector 23 */
    UnimplementedISR,           /* vector 22 */
    UnimplementedISR,           /* vector 21 */
    UnimplementedISR,           /* vector 20 */
    UnimplementedISR,           /* vector 19 */
    UnimplementedISR,           /* vector 18 */
    UnimplementedISR,           /* vector 17 */
    UnimplementedISR,           /* vector 16 */
    UnimplementedISR,           /* vector 15 */
    UnimplementedISR,           /* vector 14 */
    UnimplementedISR,           /* vector 13 */
    UnimplementedISR,           /* vector 12 */
    UnimplementedISR,          /* vector 11 */
    UnimplementedISR,          /* vector 10 */
    UnimplementedISR,          /* vector 09 */
    UnimplementedISR,          /* vector 08 */
    UnimplementedISR,          /* vector 07 */
    IRQ_ISR,               /* vector 06 */
    UnimplementedISR,          /* vector 05 */
    UnimplementedISR,          /* vector 04 */
    UnimplementedISR,          /* vector 03 */
    UnimplementedISR,          /* vector 02 */
    UnimplementedISR,          /* vector 01 */
    _Startup               /* Reset vector */
 };
```

Appendix G: BLE112 Firmware Listing

hardware.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<hardware>
  <sleeposc enable="true" ppm="30" />
  <sleep enable="true" max_mode="2"/>
  <usb enable="false" />
  <txpower power="5" bias="5" />
  <usart channel="0" mode="spi_master" alternate="1" polarity="negative" phase="1" endianness="msb" baud="9600" endpoint="none" />
  <script enable="true" />
  <pmux regulator_pin="7" />
  <slow_clock enable="true"/>

</hardware>
```

gatt.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>

  <service uuid="1800">
   <description>Generic Access Profile</description>

   <characteristic uuid="2a00">
    <properties read="true" const="true" />
    <value>Volkswagen BLE</value>
   </characteristic>

   <characteristic uuid="2a01">
    <properties read="true" const="true" />
    <value type="hex">0002</value>
   </characteristic>
  </service>

  <service uuid="00000000-0000-1000-8000-000888888888">
    <description>CAN Command</description>
    <characteristic uuid="00000000-0000-0000-0000-111100001111" id="WindowControl">
      <properties write ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110000" id="DoorLockControl">
      <properties write ="true" />
      <value length ="1"/>
    </characteristic>

    <characteristic uuid="00000000-0000-0000-0000-000011110001" id="WindowDF">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110010" id="WindowDR">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110011" id="WindowPF">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110100" id="WindowPR">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110101" id="BatteryLevel">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110110" id="BatteryChargeStatus">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>
    <characteristic uuid="00000000-0000-0000-0000-000011110111" id="Current">
      <properties read ="true" />
      <value length ="1"/>
    </characteristic>


  </service>

</configuration>
```

vwble.bgs

```
#dim tmp(10)
dim connected # 1 = connected, 0 = disconnected to android device
dim Command     # command received from android device
dim MCUState
const PinToggleTime = 33 #time = 33/(32768Hz) = 1ms
const PinToggleTimer = 1 #handle

const UpdateTimer = 2
#dim channel
#dim data_len
#dim result
#dim SPI_data # data recieved from S12

dim ret_result
dim ret_channel
dim ret_data_len
dim ret_SPI_data(10)

dim toggle

##### PROCEDURES

procedure WakeUpMCU(hex)
        #set P1_0 low (S12 falling edge interrupt - PE1)
        call hardware_io_port_write(1, 1, 0)
        call hardware_set_soft_timer(0, UpdateTimer, 1) #Safety turn off for timer
        call hardware_set_soft_timer(PinToggleTime, PinToggleTimer, 1)
end


procedure StopMCU(hex)
        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x56") # message to stop controller
        call hardware_io_port_write(1, 32, 32) # SPI slave select line
        call hardware_io_port_write(1, 32, 0)
end

procedure sendMessage(hex) #UNTESTED
        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,hex)
        call hardware_io_port_write(1, 32, 32) # SPI slave select line
end

procedure getInformation(hex)
        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x80")(ret_result, ret_channel, ret_data_len, ret_SPI_data(0))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x81")(ret_result, ret_channel, ret_data_len, ret_SPI_data(1))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line
        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x82")(ret_result, ret_channel, ret_data_len, ret_SPI_data(2))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x83")(ret_result, ret_channel, ret_data_len, ret_SPI_data(3))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x84")(ret_result, ret_channel, ret_data_len, ret_SPI_data(4))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x85")(ret_result, ret_channel, ret_data_len, ret_SPI_data(5))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x86")(ret_result, ret_channel, ret_data_len, ret_SPI_data(6))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line

        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x87")(ret_result, ret_channel, ret_data_len, ret_SPI_data(7))
        call hardware_io_port_write(1, 32, 32) # SPI slave select line
end



##### EVENTS

#init gap mod, bonding and start freerunning times on system boot
event system_boot(major ,minor ,patch ,build ,ll_version ,protocol_version ,hw )


        toggle = 0
    # configure P1_0 + 1_5 as output (port, direction bitmask) bit1,5 = output (FIGURE OUT HOW TO NOT PULL LOW, switch order??)
    call hardware_io_port_config_direction(1, $35)
    call hardware_io_port_write(1, 35, 3)


    # set P1_0 pin high (port, mask, data)
    #call hardware_io_port_write(1, 1, 1)

    # configure P2_0 as output (port, direction bitmask) - testing
    call hardware_io_port_config_direction(2, 3)

    # set P2_0 pin low (port, mask, data) - testing
    call hardware_io_port_write(2, 1, 0)

    call hardware_io_port_write(2, 2, 0)
    #call hardware_spi_transfer(0,1,"\x57") # check MCU state

    call gap_set_adv_parameters(2000, 3000, 7)
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)


end


#Connection Listener
event connection_status(connection, flags, address, address_type, conn_interval, timeout, latency, bonding)
    if connected = 0 then
        #Device connected to BLE112
        connected = 1
        call WakeUpMCU(1)


    end if

end

event hardware_soft_timer(handle)
    # set P1_0 high
    if handle = PinToggleTimer then
        call hardware_io_port_write(1, 1, 1)
        call hardware_set_soft_timer(32768, UpdateTimer, 0)
    end if
    if handle = UpdateTimer then
        call hardware_io_port_write(1, 32, 0) # SPI slave select line
        call hardware_spi_transfer(0,1,"\x01")
        call hardware_io_port_write(1, 32, 32) # SPI slave select line !! should be (1,32,1) (port, mask, data)

        call getInformation(1)

        #Currently, somehow 0 - 83 1 - 84 2 - 80 3 - 81 4 - 82
        if ret_SPI_data(0:1) != $FF then
            #Should be 0x00
        end if

        if ret_SPI_data(1:1) != $FF then
            call attributes_write(WindowDF,0,1,ret_SPI_data(1:1))
        end if

        if ret_SPI_data(2:1) != $FF then
            call attributes_write(WindowDR,0,1,ret_SPI_data(2:1))
        end if

        if ret_SPI_data(3:1) != $FF then
            call attributes_write(WindowPF,0,1,ret_SPI_data(3:1))
        end if

        if ret_SPI_data(4:1) != $FF then
            call attributes_write(WindowPR,0,1,ret_SPI_data(4:1))
        end if

        if ret_SPI_data(5:1) != $FF then
            call attributes_write(BatteryLevel,0,1,ret_SPI_data(5:1))
        end if

        if ret_SPI_data(6:1) != $FF then
            call attributes_write(BatteryChargeStatus,0,1,ret_SPI_data(6:1))
        end if

        if ret_SPI_data(7:1) != $FF then
            call attributes_write(Current,0,1,ret_SPI_data(7:1))
        end if

        if connected = 0 then
            call hardware_io_port_write(1, 32, 0)
        end if

    end if
end

#Disconnect Listener
event connection_disconnected(handle, result)

    if connected = 1 then
        #Device disconnected from BLE112
        connected = 0
        call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
        call hardware_set_soft_timer(0, UpdateTimer, 1)
        call StopMCU(1)
    end if

end

event attributes_value(connection, reason, handle, offset, value_len, value_data)
    call hardware_io_port_write(2, 1, 1)
    if handle = WindowControl then
        call hardware_io_port_write(2, 1, 1)
        if value_len < 2 then
            Command=value_data(0:1)
```

```
if Command = 1 then #up
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x02")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line !! should be (1,32,1) (port, mask, data)

    call getInformation(1)
    #call WakeUpMCU(1)
end if
if Command = 2 then #down
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x03")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line

    call getInformation(1)


end if
if Command = 3 then #DFU
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x06")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line

    call getInformation(1)


end if
if Command = 4 then #DRU
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x07")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line

    call getInformation(1)


end if
if Command = 5 then #PFU
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x08")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line

    call getInformation(1)


end if
if Command = 6 then #PRU
    call hardware_io_port_write(1, 32, 0) # SPI slave select line
    call hardware_spi_transfer(0,1,"\x09")
    call hardware_io_port_write(1, 32, 32) # SPI slave select line
```

```
            call getInformation(1)


        end if
        if Command = 7 then #DFD
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x0a")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)


        end if
        if Command = 8 then #DRD
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x0b")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)


        end if
        if Command = 9 then #PFD
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x0c")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)


        end if
        if Command = $a then #PRD
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x0d")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)


        end if
    end if
end if
if handle = DoorLockControl then
    call hardware_io_port_write(2, 1, 1)
    if value_len < 2 then
        Command=value_data(0:1)
        if Command = 1 then
```

```
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x04")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)

        end if
        if Command = 2 then
            call hardware_io_port_write(1, 32, 0) # SPI slave select line
            call hardware_spi_transfer(0,1,"\x05")
            call hardware_io_port_write(1, 32, 32) # SPI slave select line

            call getInformation(1)


        end if
    end if
end if

#Currently, somehow 0 - 83 1 - 84 2 - 80 3 - 81 4 - 82
if ret_SPI_data(0:1) != $FF then
    #Should be 0x00

end if

if ret_SPI_data(1:1) != $FF then
    call attributes_write(WindowDF,0,1,ret_SPI_data(1:1))
end if

if ret_SPI_data(2:1) != $FF then
    call attributes_write(WindowDR,0,1,ret_SPI_data(2:1))
end if

if ret_SPI_data(3:1) != $FF then
    call attributes_write(WindowPF,0,1,ret_SPI_data(3:1))
end if

if ret_SPI_data(4:1) != $FF then
    call attributes_write(WindowPR,0,1,ret_SPI_data(4:1))
end if

if ret_SPI_data(5:1) != $FF then
    call attributes_write(BatteryLevel,0,1,ret_SPI_data(5:1))
end if

if ret_SPI_data(6:1) != $FF then
    call attributes_write(BatteryChargeStatus,0,1,ret_SPI_data(6:1))
    end if

    if ret_SPI_data(7:1) != $FF then
        call attributes_write(Current,0,1,ret_SPI_data(7:1))
    end if


end
```

attributes.txt

```
WindowControl 8
DoorLockControl 10
WindowDF 12
WindowDR 14
WindowPF 16
WindowPR 18
BatteryLevel 20
BatteryChargeStatus 22
Current 24
```

variable_memory_usage.txt

```
38
0:0 BatteryChargeStatus
0:0 BatteryLevel
0:0 Current
0:0 DoorLockControl
0:0 GAP_AD_FLAG_BREDR_NOT_SUPPORTED
0:0 GAP_AD_FLAG_GENERAL_DISCOVERABLE
0:0 GAP_AD_FLAG_LIMITED_DISCOVERABLE
0:0 GAP_AD_FLAG_MASK
0:0 GAP_AD_FLAG_SIMULTANEOUS_LEBREDR_CTRL
0:0 GAP_AD_FLAG_SIMULTANEOUS_LEBREDR_HOST
0:0 GAP_SCAN_HEADER_ADV_DIRECT_IND
0:0 GAP_SCAN_HEADER_ADV_DISCOVER_IND
0:0 GAP_SCAN_HEADER_ADV_IND
0:0 GAP_SCAN_HEADER_ADV_NONCONN_IND
0:0 GAP_SCAN_HEADER_CONNECT_REQ
0:0 GAP_SCAN_HEADER_SCAN_REQ
```

```
0:0 GAP_SCAN_HEADER_SCAN_RSP
0:0 WindowControl
0:0 WindowDF
0:0 WindowDR
0:0 WindowPF
0:0 WindowPR
0:0 attclient_attribute_value_type_indicate
0:0 attclient_attribute_value_type_indicate_rsp_req
0:0 attclient_attribute_value_type_notify
0:0 attclient_attribute_value_type_read
0:0 attclient_attribute_value_type_read_blob
0:0 attclient_attribute_value_type_read_by_type
0:0 attributes_attribute_change_reason_write_command
0:0 attributes_attribute_change_reason_write_request
0:0 attributes_attribute_change_reason_write_request_user
0:0 attributes_attribute_status_flag_indicate
0:0 attributes_attribute_status_flag_notify
0:0 connection_completed
0:0 connection_connected
0:0 connection_encrypted
0:0 connection_parameters_change
0:0 gap_ad_type_flags
0:0 gap_ad_type_localname_complete
0:0 gap_ad_type_localname_short
0:0 gap_ad_type_none
0:0 gap_ad_type_services_128bit_all
0:0 gap_ad_type_services_128bit_more
0:0 gap_ad_type_services_16bit_all
0:0 gap_ad_type_services_16bit_more
0:0 gap_ad_type_services_32bit_all
0:0 gap_ad_type_services_32bit_more
0:0 gap_ad_type_txpower
0:0 gap_address_type_public
0:0 gap_address_type_random
0:0 gap_adv_policy_all
0:0 gap_adv_policy_whitelist_all
0:0 gap_adv_policy_whitelist_connect
0:0 gap_adv_policy_whitelist_scan
0:0 gap_broadcast
0:0 gap_directed_connectable
0:0 gap_discover_generic
0:0 gap_discover_limited
0:0 gap_discover_observation
0:0 gap_general_discoverable
0:0 gap_limited_discoverable
0:0 gap_non_connectable
0:0 gap_non_discoverable
0:0 gap_scan_policy_all
```

```
0:0 gap_scan_policy_whitelist
0:0 gap_scannable_connectable
0:0 gap_undirected_connectable
0:0 gap_user_data
0:0 sm_bonding_key_addr_public
0:0 sm_bonding_key_addr_static
0:0 sm_bonding_key_csrk
0:0 sm_bonding_key_edivrand
0:0 sm_bonding_key_irk
0:0 sm_bonding_key_ltk
0:0 sm_bonding_key_masterid
0:0 sm_io_capability_displayonly
0:0 sm_io_capability_displayyesno
0:0 sm_io_capability_keyboarddisplay
0:0 sm_io_capability_keyboardonly
0:0 sm_io_capability_noinputnooutput
0:0 system_endpoint_api
0:0 system_endpoint_script
0:0 system_endpoint_test
0:0 system_endpoint_uart0
0:0 system_endpoint_uart1
0:0 system_endpoint_usb
0:4 connected
4:4 Command
8:4 MCUState
12:0    PinToggleTime
12:0    PinToggleTimer
12:0    UpdateTimer
12:4    ret_result
16:4    ret_channel
20:4    ret_data_len
24:10   ret_SPI_data
34:4    toggle
```

Appendix H: Android Code Listing

CANUI.java

```
package com.example.VolkswagenCAN;

import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattService;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.util.Log;
import android.view.Menu;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.bluetoothcan.R;



@SuppressLint("NewApi")
public class CANUI extends Activity {
```

```
// Private class variables
private int lockState = 0; // lockState = 0 for locked, 1 for unlocked
private int lockState2 = 0; // lockState2 = 0 for locked, 1 for unlocked
private boolean connected = false;

// timer variables
private static int windowParam;
private static int testcount = 0;
private static boolean WindowFlag = false;


// Static variables for regular Bluetooth message handler
public static final int MESSAGE_READ = 1;
public static final int MESSAGE_DEVICE_NAME = 2;
public static final int MESSAGE_TOAST = 3;

// gatt characteristics
public static final int ALLD = 1;
public static final int ALLU = 2;
public static final int DFU = 3;
public static final int DRU = 4;
public static final int PFU = 5;
public static final int PRU = 6;
public static final int DFD = 7;
public static final int DRD = 8;
public static final int PFD = 9;
public static final int PRD = 10;


public static final String TOAST = "toast";

private static final String TAG = "CANUserInterface";

// Static variables for intent extras and returns to know what we just connected from
private static final int CONNECT_BLUETOOTH = 1;
private static final int CONNECT_BLUETOOTH_LE = 2;

// Static strings for intent extras and data packets
public static final String EXTRAS_DEVICE_NAME = "LEDevice";
public static final String EXTRAS_DEVICE_ADDRESS = "LEAddress";

//Predefined UUIDs for GATT Profile used in BLE with BLE112
private static final UUID serviceUUID =
        UUID.fromString("00000000-0000-1000-8000-008888888888");
private static final UUID windowsUUID =
        UUID.fromString("00000000-0000-0000-0000-111100001111");
private static final UUID locksUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110000");
private static final UUID windowDFUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110001");
private static final UUID windowDRUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110010");
private static final UUID windowPFUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110011");
private static final UUID windowPRUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110100");
private static final UUID batteryLevelUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110101");
private static final UUID batteryChargeStateUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110110");
private static final UUID currentLevelUUID =
        UUID.fromString("00000000-0000-0000-0000-000011110111"); // !!



private String connectedDeviceName = null;
private String leDeviceName;
private String leDeviceAddress;

private BluetoothAdapter bluetooth = null;
private BluetoothManager mBluetoothManager = null;
private BluetoothLeService mBluetoothLeService = null;

private BluetoothGatt btGatt = null;
private BluetoothGattCharacteristic windows = null;
private BluetoothGattCharacteristic locks = null;
private BluetoothGattCharacteristic windowDF = null;
private BluetoothGattCharacteristic windowDR = null;
private BluetoothGattCharacteristic windowPF = null;
private BluetoothGattCharacteristic windowPR = null;
private BluetoothGattCharacteristic batteryLevel = null;
private BluetoothGattCharacteristic batteryChargeState = null;
private BluetoothGattCharacteristic currentLevel = null;

private Timer updateTimer;
private TimerTask timerTask;
private int timerCounter = 0;


@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_canui);
```

```
        bluetooth = BluetoothAdapter.getDefaultAdapter();

        if (bluetooth == null) {
            Toast.makeText(this, "No bluetooth?!", Toast.LENGTH_LONG).show();
            finish();
            return;
        }

        updateTimer = new Timer();
        timerTask = new myTimerTask();
        updateTimer.schedule(timerTask, 0, 50);


        OnTouchUp(findViewById(R.id.windowALLU), ALLU);
        OnTouchDown(findViewById(R.id.windowALLD), ALLD);
        OnTouchUp(findViewById(R.id.windowDFU), DFU);
        OnTouchDown(findViewById(R.id.windowDFD), DFD);
        OnTouchUp(findViewById(R.id.windowPFU), PFU);
        OnTouchDown(findViewById(R.id.windowPFD), PFD);
        OnTouchUp(findViewById(R.id.windowDRU), DRU);
        OnTouchDown(findViewById(R.id.windowDRD), DRD);
        OnTouchUp(findViewById(R.id.windowPRU), PRU);
        OnTouchDown(findViewById(R.id.windowPRD), PRD);


        //This is for regular bluetooth, includes a messageHandler to return messages to
        mBluetoothManager = new BluetoothManager(this, messageHandler);
        //mBluetoothManager.listen();

        //Binds BLE service so this activity can use its functions
        Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);
        if(bindService(gattServiceIntent, mServiceConnection, BIND_AUTO_CREATE)) {
            Log.v(TAG, "Bound to service");
        } else {
            Log.v(TAG, "Unsuccessful at bind");
        }


    }



    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());
    }
```

```
@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(mGattUpdateReceiver);
}

@Override
protected void onDestroy() {
    super.onDestroy();
    mBluetoothManager.stop();
    mBluetoothLeService.close();
    unbindService(mServiceConnection);
    mBluetoothLeService = null;
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.canui, menu);
    return true;
}

/*
 * lockClicked - Response to lock button press.
 * Flips image to opposite (unlocked<->locked)
 * Uses BLE to set BluetoothGattCharacteristic value to 2 for unlocked and 1 for locked
 */

public void sendWindow() {
    if (windows != null) {
        //Set windows GattCharacteristic to 2 for window up
        Log.v(TAG, "send window top");
        windows.setValue(windowParam, BluetoothGattCharacteristic.FORMAT_UINT8, 0);
        btGatt.writeCharacteristic(windows);
        Log.v(TAG, "send window bottom");
    }
}

public boolean lockClicked(View view) {
    ImageButton lock = (ImageButton) findViewById(R.id.lockImage);
    if (lockState == 0) {
        lock.setImageResource(R.drawable.padlockunlocked);
        if (locks != null) { //Checks to make sure we're connected to BLE
            locks.setValue(2, BluetoothGattCharacteristic.FORMAT_UINT8, 0);
            btGatt.writeCharacteristic(locks);
        }
        lockState = 1;
    } else if (lockState == 1) {
        lock.setImageResource(R.drawable.padlocklocked);
        if (locks != null) { //Checks to make sure we're connected to BLE
            locks.setValue(1, BluetoothGattCharacteristic.FORMAT_UINT8, 0);
            btGatt.writeCharacteristic(locks);
            //windows.setValue(1, BluetoothGattCharacteristic.FORMAT_UINT8, 0);
            //btGatt.writeCharacteristic(windows);

        }
        lockState = 0;
    }
    return false;
}


// Returns from connection activities with identifiers
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    Log.v(TAG, "onActivityResult with" + resultCode);
    TextView btDevice = (TextView) findViewById(R.id.bluetoothDevice);
    if (data==null)
    {
        Log.v(TAG, "data is null");
    }

    switch (requestCode) {
    case CONNECT_BLUETOOTH:
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, true);
            btDevice.setText(data.getExtras().getString("device_mac_address"));
        }
        if (resultCode == Activity.RESULT_CANCELED) {

        }
        break;
    case CONNECT_BLUETOOTH_LE:
        if (resultCode == Activity.RESULT_OK) {
            //Connect to device if one is chosen from BluetoothLEDevice finding
            leDeviceName = data.getStringExtra(EXTRAS_DEVICE_NAME);
            leDeviceAddress = data.getStringExtra(EXTRAS_DEVICE_ADDRESS);
            Log.v(TAG, leDeviceName);
            mBluetoothLeService.connect(leDeviceAddress);

        }
        if (resultCode == Activity.RESULT_CANCELED) {

        }
        break;
    }
}
```

```
//Regular bluetooth connect function
private void connectDevice(Intent data, boolean secure) {
    String address = data.getExtras().getString("device_mac_address");
    BluetoothDevice device = bluetooth.getRemoteDevice(address);
    Log.v(TAG, "connecting to " + address);
    mBluetoothManager.connect(device);
}

//Response to regular bluetooth button, starts bluetooth device finding activity
public void bluetoothDeviceActivity(View view){
    Intent intent = new Intent(this, BluetoothDeviceList.class);
    startActivityForResult(intent, CONNECT_BLUETOOTH);
}

//Response to BLE button, starts BLE device finding activity
public void bluetoothLEDeviceActivity(View view){
    Intent intent = new Intent(this, BluetoothLEDevice.class);
    startActivityForResult(intent, CONNECT_BLUETOOTH_LE);
}

//This message handler is for regular bluetooth returns
private final Handler messageHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            //TextView testingText = (TextView) findViewById(R.id.testingText);
            //testingText.setText(readMessage);
            break;
        case MESSAGE_DEVICE_NAME:
            // save the connected device's name
            connectedDeviceName = msg.getData().getString("device_name");
            Toast.makeText(getApplicationContext(), "Connected to "
                          + connectedDeviceName, Toast.LENGTH_LONG).show();
            break;
        case MESSAGE_TOAST:
            Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                          Toast.LENGTH_SHORT).show();
            break;
        }
    }
};

//mGattUpdateReceiver receives information concerning BLE connection/GATT
```

```
private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final String action = intent.getAction();
        if (BluetoothLeService.ACTION_GATT_CONNECTED.equals(action)) {
            //Sets connected device text
            TextView btDevice = (TextView) findViewById(R.id.bluetoothDevice);
            btDevice.setText(leDeviceName);

            Toast.makeText(getApplicationContext(), "Connected to "
                    + leDeviceName, Toast.LENGTH_LONG).show();

            connected = true;

            invalidateOptionsMenu();
        } else if (BluetoothLeService.ACTION_GATT_DISCONNECTED.equals(action)) {
            mBluetoothLeService.connect(leDeviceAddress);

            //Sets disconnected device text
            TextView btDevice = (TextView) findViewById(R.id.bluetoothDevice);
            btDevice.setText("Disconnected");

            Toast.makeText(getApplicationContext(), "Disconnected from "
                    + leDeviceName, Toast.LENGTH_LONG).show();

            connected = false;

            invalidateOptionsMenu();
        } else if (BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)) {
            //This is the return when the Gatt Service Discovery finishes
            //The code is specifically coded only for our BLE112 Gatt Profile
            List<BluetoothGattService> serviceList =
                    mBluetoothLeService.getSupportedGattServices();

            //Gets last bluetooth gatt service uuid and prints it (debugging)
            BluetoothGattService btGattService= serviceList.get(serviceList.size() - 1);
            Log.v(TAG, btGattService.getUuid().toString());
            Log.v(TAG, Integer.toString(btGattService.getCharacteristics().size()));
            //Sets characteristics for programs
            locks = btGattService.getCharacteristic(locksUUID);
            windows = btGattService.getCharacteristic(windowsUUID);
            windowDF = btGattService.getCharacteristic(windowDFUUID);
            windowDR = btGattService.getCharacteristic(windowDRUUID);
            windowPF = btGattService.getCharacteristic(windowPFUUID);
            windowPR = btGattService.getCharacteristic(windowPRUUID);
            batteryLevel = btGattService.getCharacteristic(batteryLevelUUID);
            batteryChargeState = btGattService.getCharacteristic(batteryChargeStateUUID);
            currentLevel = btGattService.getCharacteristic(currentLevelUUID);
            btGatt = BluetoothLeService.getBluetoothGatt();
            Log.v(TAG, "locks " + locks.getUuid().toString());
            Log.v(TAG, "windows " + windows.getUuid().toString());
            Log.v(TAG, "battery level " + batteryLevel.getUuid().toString());
        } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
            String charUUID = intent.getStringExtra(BluetoothLeService.INTENT_CHARACTERISTIC);
            processData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA), UUID.fromString(charUUID));
        }
    }
};



private void processData(String data, UUID charUUID) {

    int intData = 0;
    char c = data.charAt(0);
    intData = (int) c;
    intData &= 0xFF;
    //if (data != null) intData = Integer.parseInt(data);
    Log.v(TAG,intData + " " + charUUID.toString());

    TextView windowDFText = (TextView) findViewById(R.id.frontLeftText);
    TextView windowDRText = (TextView) findViewById(R.id.backLeftText);
    TextView windowPFText = (TextView) findViewById(R.id.frontRightText);
    TextView windowPRText = (TextView) findViewById(R.id.backRightText);
    TextView batteryLevelText = (TextView) findViewById(R.id.battLevelText);
    TextView currentStatusText = (TextView) findViewById(R.id.currentLevelText);
    TextView batteryChargeStatusText = (TextView) findViewById(R.id.battStatusText);

    if (charUUID.toString().equals(windowDFUUID.toString())) {
        intData = 100-intData;
        windowDFText.setText(Integer.toString(intData).trim());
    }
    if (charUUID.toString().equals(windowDRUUID.toString())) {
        intData = 100-intData;
        windowDRText.setText(Integer.toString(intData).trim());
    }
    if (charUUID.toString().equals(windowPFUUID.toString())) {
        intData = 100-intData;
        windowPFText.setText(Integer.toString(intData).trim());
    }
    if (charUUID.toString().equals(windowPRUUID.toString())) {
        intData = 100-intData;
        windowPRText.setText(Integer.toString(intData).trim());
    }
    if (charUUID.toString().equals(batteryLevelUUID.toString())) {
            batteryLevelText.setText(Integer.toString(intData).trim() + " %");
            ImageView batteryImage = (ImageView) findViewById(R.id.battery);
            if (intData > 65) {
                batteryImage.setImageResource(R.drawable.greenbatt);
            } else if (intData > 35) {
                batteryImage.setImageResource(R.drawable.yellowbatt);
            } else if (intData > 5) {
                batteryImage.setImageResource(R.drawable.redbatt);
            } else {
                batteryImage.setImageResource(R.drawable.emptybatt);
            }
        }
        if (charUUID.toString().equals(batteryChargeStateUUID.toString())) {
            if (intData == 3 || intData == 4 || intData == 6) {
                batteryChargeStatusText.setText("CHARGING");
            } else {
                batteryChargeStatusText.setText(" ");
            }

        }
        if (charUUID.toString().equals(currentLevelUUID.toString())) {

            currentStatusText.setText(Integer.toString(intData).trim() + " AMPS");
        }
    }

// Code to manage Service life cycle.
    private final ServiceConnection mServiceConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName componentName, IBinder service) {
            mBluetoothLeService = ((BluetoothLeService.LocalBinder) service).getService();
            if (!mBluetoothLeService.initialize()) {
                Log.e(TAG, "Unable to initialize Bluetooth");
                finish();
            }
            // Automatically connects to the device upon successful start-up initialization.
            mBluetoothLeService.connect(leDeviceAddress);
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            mBluetoothLeService = null;
        }
    };

    private static IntentFilter makeGattUpdateIntentFilter() {
        final IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_CONNECTED);
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_DISCONNECTED);
        intentFilter.addAction(BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED);
        intentFilter.addAction(BluetoothLeService.ACTION_DATA_AVAILABLE);
        return intentFilter;
    }

    private class myTimerTask extends TimerTask{
        @Override
        public void run() {

            if (connected && windowDF != null && windowPF != null && windowDR != null && windowPR !=
null && batteryLevel != null && batteryChargeState != null && mBluetoothLeService != null) {
                if (timerCounter == 0) mBluetoothLeService.readCharacteristic(windowDF);
                if (timerCounter == 2) mBluetoothLeService.readCharacteristic(windowDR);
                if (timerCounter == 4) mBluetoothLeService.readCharacteristic(windowPF);
                if (timerCounter == 6) mBluetoothLeService.readCharacteristic(windowPR);
                if (timerCounter == 8) mBluetoothLeService.readCharacteristic(batteryLevel);
                if (timerCounter == 10) mBluetoothLeService.readCharacteristic(batteryChargeState);
                if (timerCounter == 12) mBluetoothLeService.readCharacteristic(currentLevel);

            }
            if (timerCounter % 2 == 1)  {
                    if (WindowFlag == true) sendWindow();
                }

            if (++timerCounter > 13) timerCounter = 0;

            // handles window movement
            /*        if(WindowFlag == true){

                sendWindow();
                if (windows != null) {
                    //Set windows GattCharacteristic to 2 for window up
                    windows.setValue(windowParam, BluetoothGattCharacteristic.FORMAT_UINT8, 0);
                    btGatt.writeCharacteristic(windows);
                }
            }*/


        }
    }

    private boolean OnTouchUp(View win, int WinParam){
        final int parameter = WinParam;
        win.setOnTouchListener(new View.OnTouchListener() {

            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_UP){
                    ((ImageView) v).setImageResource(R.drawable.uparrowreleased);
                    Log.v(TAG, "window action up up");
                    WindowFlag = false;
                    return true;
                }

                if(event.getAction() == MotionEvent.ACTION_DOWN){
                    ((ImageView) v).setImageResource(R.drawable.uparrowselected);
                    Log.v(TAG, "window action up down");
                    WindowFlag = true;
                    windowParam = parameter;
                    return true;
                }
                return false;
            }
        });
        return false;
    }


    private boolean OnTouchDown(View win, int WinParam){
        final int parameter = WinParam;
        win.setOnTouchListener(new View.OnTouchListener() {

            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_UP){
                    ((ImageView) v).setImageResource(R.drawable.downarrowreleased);
                    Log.v(TAG, "window action down up");
                    WindowFlag = false;
                    return true;
                }

                if(event.getAction() == MotionEvent.ACTION_DOWN){
                    ((ImageView) v).setImageResource(R.drawable.downarrowselected);
                    Log.v(TAG, "window action down down");
                    WindowFlag = true;
                    windowParam = parameter;
                    return true;
                }
                return false;
            }
        });
        return false;
    }
}
```

Appendix I: Team Photo

Figure 15: Team photograph - from left to right, Erin Watson, Cliff Bargar, and Maxwell Wu