Low-Level Design

Smart Windows

3/4/2010
EE Senior Design 2010
Professor Schafer

Kelley Daniels
Tommy Haunert
David Shilling
Andy Spangler

Smart Windows
Let your blinds work for you!

Table of Contents

1 Introduction

Exciting energy-saving technologies such as the “smart” grid have captured the cultural imagination. Meanwhile, simple household practices are increasingly being used to reduce energy demand. Products like the compact fluorescent lamp (CFL) have shown the commercial viability of these everyday energy-savers.

As identified by the Department of Energy, one simple energy-saving solution is the proper operation of window treatments like blinds and shades. These window treatments are capable of reducing the load on household air conditioning units, thereby reducing energy consumption. In addition, properly operated window treatments offer other benefits, such as security, to the homeowner. However, to maximize the return on these window treatments, an automated system is needed to ensure the timely opening and closing of the window treatments. Through the utilization of current technology paired with innovative engineering design, Smart Windows solves this problem of window treatment automation.

While the Smart Windows system will be adaptable to many types of window treatments, the Smart Window is designed for Solar Shades polarizing window treatments. Solar Shades, a start-up company operated out of Innovation Park at the University of Notre Dame, seeks to revolutionize the design of window treatments. The Smart Windows system will be the brains behind these windows, giving the homeowner central control and powerful automation of the Solar Shades.

2 Problem Statement and Proposed Solution

In collaboration with Solar Shades, Smart Windows will create a bridge between Notre Dame EE Senior Design and Notre Dame’s Innovation Park. For the first time, undergraduate electrical engineers will be faced with real world business challenges.

In addition, Smart Windows seeks to solve a very important technical problem. In a world faced with rising energy demand and depleting energy supply, the next generation of technology must take into account energy usage at every level. According to the United States Energy Information Administration’s (EIA) Residential Energy Consumption Survey (RECS), air conditioning consumes a significant and growing portion of US electric power. From 1997 to 2005, the RECS shows that the percent of US residential electric power used for air conditioning rose from 14% to 20.2%.[1]

To fight rising energy usage, the US Department of Energy’s Energy Savers program recommends reducing the stress on air conditioners. One important Department of Energy guideline is to use window treatments like blinds and shades to reduce the thermal gain through the windows due to radiant solar energy: “Install window shades or other window treatments and close the shades. Shades will help block out not only direct sunlight, but also radiated heat from the outdoors, and insulated shades will reduce the conduction of heat into your home through your windows." In fact, the Department of Energy says that reflective shades can reduce heat gain up to 45%.[2]

In addition to reducing air conditioning energy consumption, window treatments have other functions. For example, the September 23, 2009 edition of the University of Notre Dame and Saint Mary's College newspaper *The Observer* recommends window treatments as a crime prevention tool. In an article entitled "Burglars Target Off-Campus Housing," South Bend Police Captain Phil Trent notices, "There's people with their front windows right open and I can see a 50-inch plasma screen from the street. You can see someone with the lights on in their house and they're working on a laptop computer…A burglar can do an assessment of what they can steal just by walking down the street looking in the windows." The article states, "To prevent burglaries, students should keep their windows and curtains closed."[3]

These benefits of window treatments are only effective, however, if the homeowner is diligent in opening and closing them. To access the energy benefits of window treatments, the homeowner must constantly monitor sunlight exposure. To access the security benefits, the homeowner must close every treatment prior to leaving the house. Since few homeowners can afford to give this level of attention to their window treatments, the benefits of properly operated window treatments are rarely utilized. For these benefits to be tapped into, the windows must operate automatically in the homeowner's stead. It is this problem of maximizing window treatment utility through automation and electronic intelligence that the Smart Windows design addresses.

The Smart Windows system is centered on a PC Control Unit (PCU). The user will be able to issue commands to On-Window Units (OWUs) from this PCU and from a wireless remote control unit (RCU). Both the PCU and the RCU will communicate with the OWUs through a wireless interface. The PCU will consist of a custom-designed PC application with graphical user interface (GUI) which allows users the choice to manually control individual windows, to implement sensor control, or operate all windows in unison. These choices are defined as modes.

The OWU, operating in one of these three modes, will drive the window treatment with an appropriate motor. In sensor-controlled mode, or Green Mode, the OWU will use a light sensor to decide if the Solar Shade should be opened or closed to maximize the efficiency of the household HVAC. In Safe Mode, all the window treatments in the house are closed in unison, typically at night or during working hours. In manual mode, the user operates the windows individually from the PC application, remote control, or on-window buttons. Users will choose from opened, closed, or half opened Solar Shades. The OWU will derive its intelligence from an embedded microprocessor.

[1] EIA online RECS 2005 Status Report. <http://www.eia.doe.gov/emeu/recs/contents.html>.
[2] DOE Energy Savers.
<www.energysavers.gov/your_home/space_heating_cooling/index.cfm/mytopic=12353>.
[3] Mervosh, Sarah. "Burglars Target Off-Campus Housing." *The Observer*. 23 Sept 2009.

<http://media.www.ndsmcobserver.com/media/storage/paper660/news/2009/09/23/News/Burglars.Target.OffCampus.Housing-3780142.shtml>.

3 System Description and Block Diagram

The Smart Windows system will consist of a motor block and three main units, the PC control unit (PCU), the on-window unit (OWU), and the remote control (RCU). Each unit will be broken down into several subsystems and the interfaces between them. These units will communicate through two-way RF communication using ZigBee protocol.

Each unit will require a printed circuit board (PCB) containing a microcontroller, a ZigBee transceiver circuit, and peripherals. Some peripherals will be mounted directly on the main board, but others will be mounted on secondary boards. Since each unit will require many of the same basic peripherals, a single PCB will be designed. This board will be referred to as the main board. The main board will be capable of accepting all peripherals in use in this project. However, the main board for any particular unit will only have the necessary peripherals attached. All main boards will contain a microcontroller, ZigBee transceiver circuit, a microcontroller programmer circuit, and a 20-MHz ceramic oscillator circuit supplying the microcontroller with its clock.

The PCU will consist of the PC application and a main board. The PC will connect to the microcontroller on the board through a USB interface. The ZigBee circuitry will connect to the microcontroller through a standard serial parallel interface (SPI) synchronous serial interface. The PCU main board will have three peripherals attached, a real-time clock, a serial EEPROM chip, and a FTDI serial USB/UART device. The real-time clock will interface with the microcontroller via through SPI and will keep the current time and date. The serial EEPROM will act as non-volatile data memory and will connect to the microcontroller through SPI. The FTDI serial USB/UART device will allow the microcontroller UART to communicate through standard asynchronous serial communication with the PC through USB. The PCU main board will receive its power from the USB connection to the PC.

The OWU will consist of a main board with a variety of peripherals. These peripherals will be a real-time clock, a serial EEPROM, a DC-input regulation circuit, a DC/DC voltage converter, a limit switch jumper, a button card, and a motor board, and a light sensor board. The DC-input circuit will accept a voltage from a battery stack and regulate it down to the required 3.3V. The DC/DC converter will step up this 3.3V signal to the 5.0V signal used by several of the peripherals. The limit switch jumper will contain the pull-down resistors necessary to operate the motor limit switches. The button card will contain six buttons and related circuitry and will connect to the main board through jumper wires. The motor board will contain the h-bridge circuit needed to run the DC motor and will connect to the main board through jumper wires. The light sensor board will contain a phototransistor that produces a voltage proportional to ambient light levels. It will also interface to the main board through jumper wires.

The RCU will also consist of a main board with several peripherals. These peripherals will be a DC-input circuit, LCD screen, a DC/DC voltage converter, a button card, and a serial EEPROM chip. The DC input circuit will accept a battery voltage and regulate it down to a usable value. The LCD screen will display the name of the currently selected window. It will connect to the microcontroller through an SPI interface. The DC/DC converter will provide the 5V signal needed by the LCD. The button card will hold the buttons that cycle through selected window and command the selected window to open or close. The serial EEPROM will store the names of all connected windows.

The motor block will consist of a DC motor, window blinds, two limit switches, two ID hubs, a rubber spider coupling, and a steel rod. The DC motor will drive the window blinds, causing the shaft on the blinds to rotate. The ID hubs and spider coupling will couple the window blind shaft to a bent steel rod. As the shaft turns the steel rod will come into contact with the limit switches. A press of a limit switch will indicate that the window is fully opened or closes.

A high level illustration of the system is shown in **Figure 3.1**. An in-depth illustration of the PCU is shown in **Figure 3.2**. The OWU is shown in **Figure 3.3**, and the RCU is shown in **Figure 3.4**.

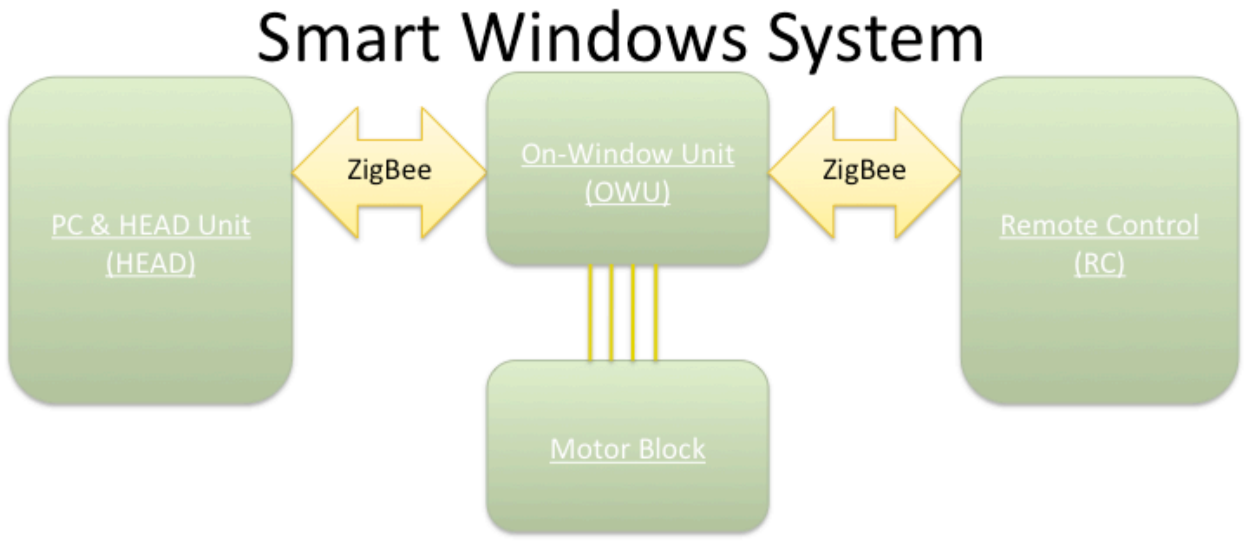


Figure 3.1. Block Diagram of the Smart Windows system.

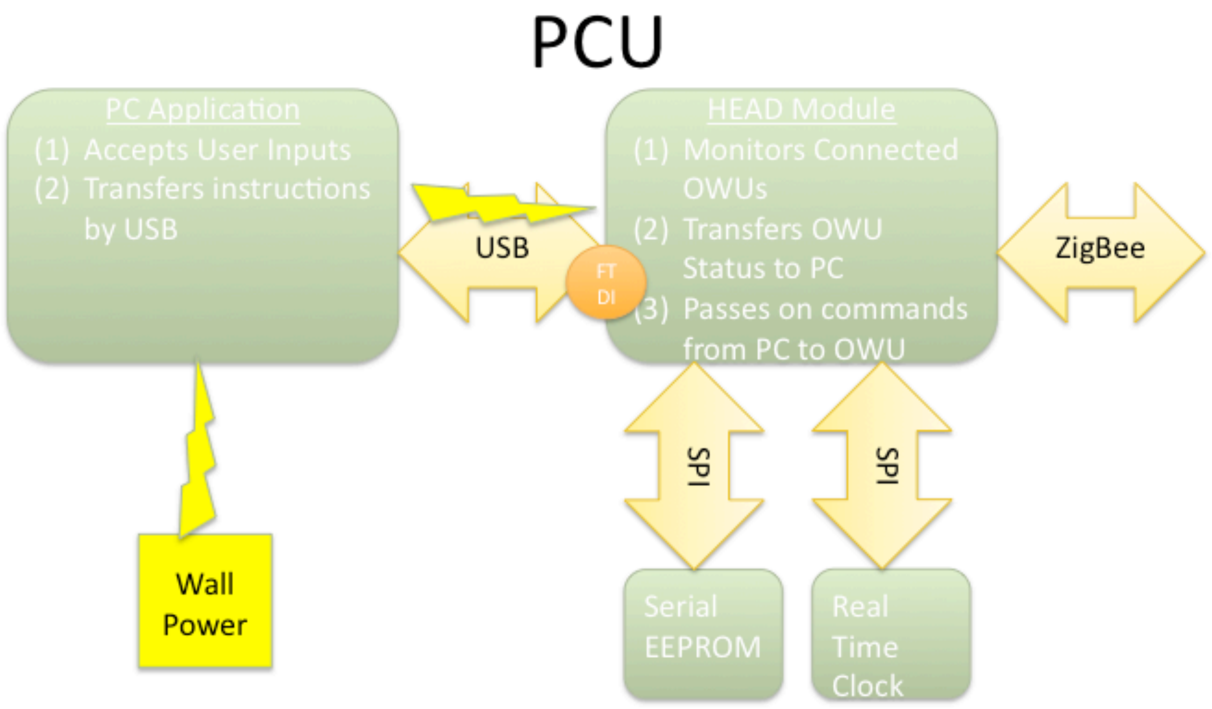

PCU
PC Application
(1) Accepts User Inputs
(2) Transfers instructions by USB
USB
FT DI
HEAD Module
(1) Monitors Connected OWUs
(2) Transfers OWU Status to PC
(3) Passes on commands from PC to OWU
ZigBee
SPI
SPI
Wall Power
Serial EEPROM
Real Time Clock

Figure 3.2. Block diagram of the PCU.

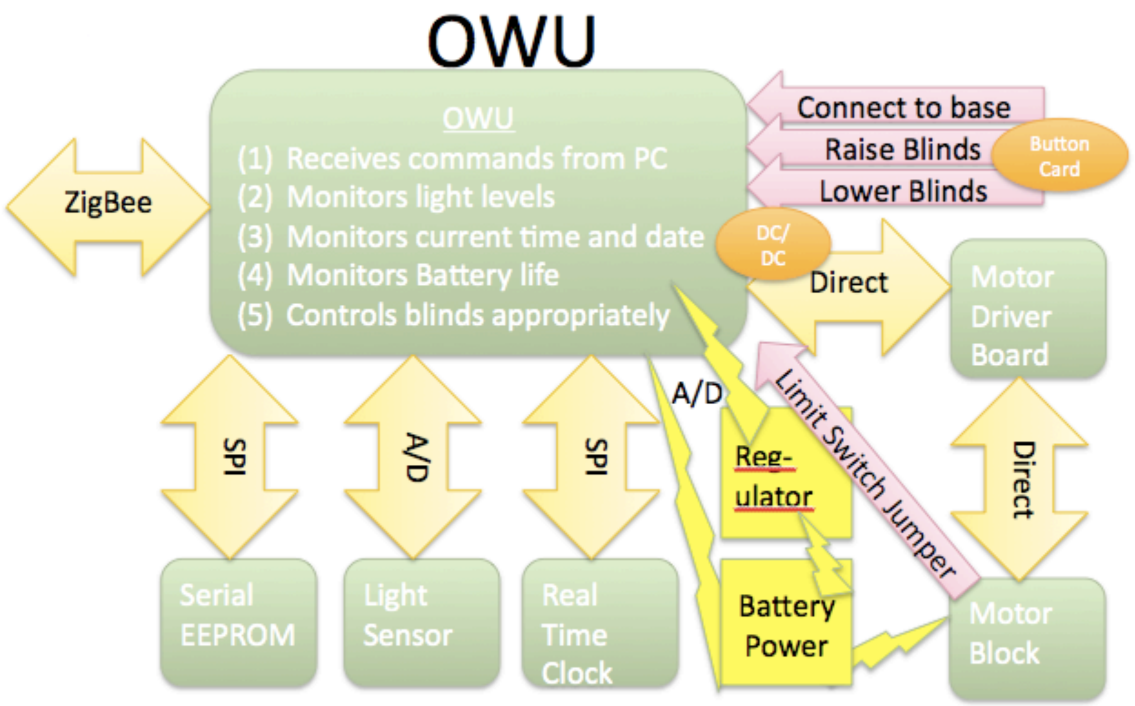


Figure 3.3. Block diagram of the OWU.

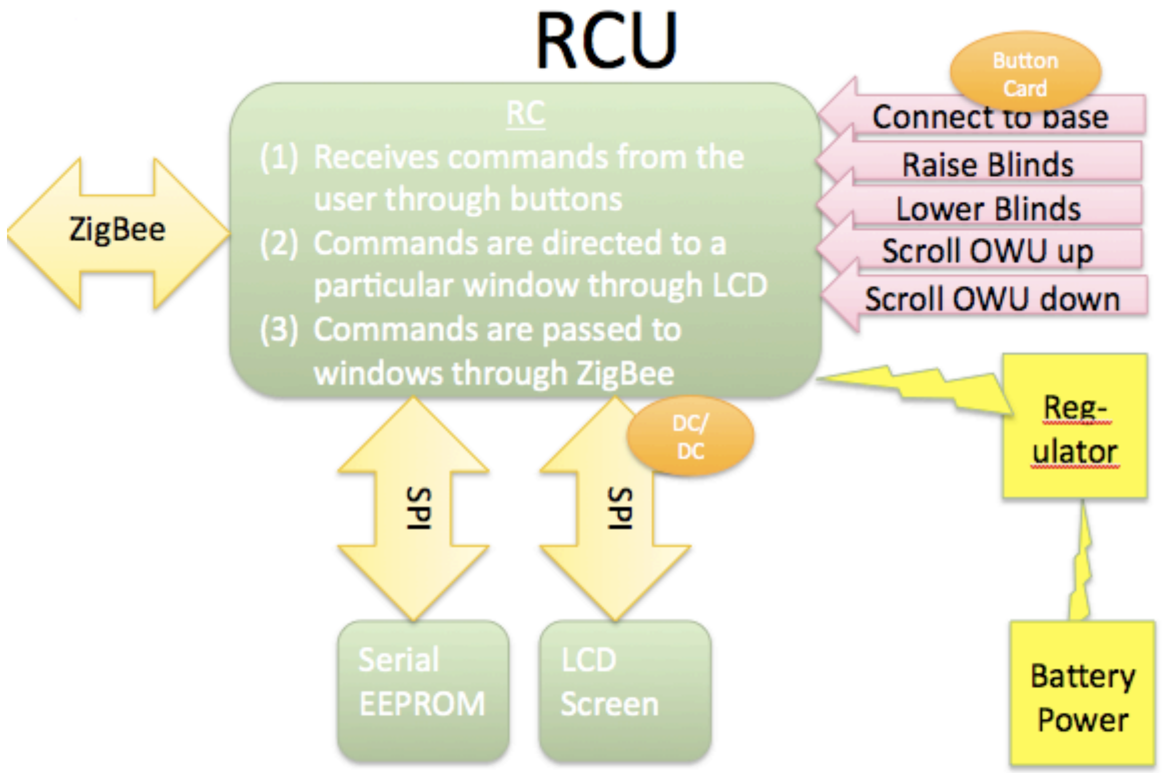

RCU
RC
(1) Receives commands from the user through buttons
(2) Commands are directed to a particular window through LCD
(3) Commands are passed to windows through ZigBee
ZigBee
Button Card
Connect to base
Raise Blinds
Lower Blinds
Scroll OWU up
Scroll OWU down
Reg-ulator
DC/DC
SPI
SPI
Serial EEPROM
LCD Screen
Battery Power

Figure 3.4. Bock diagram of the RCU

4 System Requirements

4.1 Overall System Requirements

Overall Requirements	
General Purpose	Must be capable of remotely operating window blinds from a PC and a remote control
User Implementation	Must be able to be assembled and run without specialized knowledge or atypical household tools Must operate whether or not the software application is running
Wireless Requirements	Must be able to work with window modules within 100 feet Must support up to 20 window treatments at once
Safety	Must be safe in homes Must not carry risk of electrical fires
Power	PC-based portion must draw power through USB Other portions must draw from rechargeable batteries lasting at least 2 weeks between charges
FCC Compliance	Must not cause harmful interference with other household wireless devices
Expected Life of Product	Batteries will not hold charge after 2 years of normal use. Other products guaranteed for 4 years.
Cost	System prototype must cost less than $500 to design and produce

4.2 Subsystem and Interface Requirements

Subsystem and Interface Requirements	
On Window Unit (OWU)	
General	Must be able to control window treatments intelligently Must continue to operate when wireless communication is broken
Size	Must have total dimensions less than or equal to 8" x 5" x 4"
Weight	Must be less than 5 pounds Must be able to be hung from our specific Lowe's window treatment
Power	Must use 4-8 rechargeable AA batteries lasting a minimum of 14 days
Microcontroller Software	Must use a reasonable amount of program memory Must operate the drive the motor appropriately when necessary Must periodically monitor the light sensor and manual buttons Must decode received wireless messages Must enter power-saving mode when possible
Motor	Must be a DC motor capable of at least 20 oz-in of torque Must be geared to turn less than 100 rotations per minute Must have a safety clutch to protect the window treatment Must make minimal noise when operating Must meet power requirements (see "Power") above
Light Sensor	Must be capable of differentiating a sunny day from a cloudy day Must ignore light coming from inside the house Must be report light levels to microcontroller using minimal I/O pins Must meet power requirements (see "Power") above
Wireless Transceiver	Must send and receive messages at an indoor distance of 100 feet Must be able to address messages to a particular target Must not create interference with other household items Must be able to interface with a microcontroller quickly and reliably Must meet power requirements (see "Power") above
Manual Buttons	Must reliably control the system when used Must be easily accessible Must let the user open or close the treatment one increment
EEPROM Memory	Must be able to hold 1000 bytes of non-volatile memory Must maintain memory for at least a week without power Must interface with a microcontroller
Real Time Clock	Must be capable of accepting current time from microcontroller Must keep time and date accurately from that point forward as long as power is connected.
PC Control Unit (PCU)	
General	Must give the user the highest amount of control over the system Must have control over every connected window treatment in the house
Power	Must be able to draw power from the PC USB connection
PC Software	Must be able to interface to a microcontroller through USB Must have an intuitive graphical user interface Must be capable of placing each window into one of the three modes Must be capable of controlling individual windows when in manual mode

PC Software	Must be able to interface to a microcontroller through USB Must have an intuitive graphical user interface Must be capable of placing each window into one of the three modes Must be capable of controlling individual windows when in manual mode Must store at least Wake-up/Work/Return-from-work/Sleep times locally
EEPROM Memory	Must be able to hold 1000 bytes of non-volatile memory Must maintain memory for at least a week without power Must interface with a microcontroller
Real Time Clock	Must be capable of accepting current time from microcontroller Must keep time and date accurately from that point forward as long as power is connected.
Microcontroller Software	Must decode messages received through the USB connection Must pass these messages to the wireless transceiver
Wireless Transceiver	Must send and receive messages at an indoor distance of 100 feet Must be able to address messages to a particular target Must not create interference with other household items Must be able to interface with a microcontroller quickly and reliably Must meet power requirements (see "Power") above
Remote Control Unit	
General	Must give the user remote control over a particular window treatment Must be capable of converting a particular window to manual mode Must be able to select an active window treatment to control
Power	Must use 2-4 rechargeable AA batteries lasting a minimum of 14 days
Microcontroller Software	Must monitor the user input buttons Must pass user commands to the wireless transceiver Must conserve power when possible
Manual Buttons	Must reliably control the system when used Must be easily accessible Must let the user open or close the treatment one increment Must let the user select the active window to communicate with Must interface with the microcontroller directly through 8 or less I/O pins
Wireless Transceiver	Must send and receive messages at an indoor distance of 100 feet Must be able to address messages to a particular target Must not create interference with other household items Must be able to interface with a microcontroller quickly and reliably Must meet power requirements (see "Power") above
EEPROM Memory	Must be able to hold 1000 bytes of non-volatile memory Must maintain memory for at least a week without power Must interface with a microcontroller
LCD Screen	Must display the actively selected window on-screen Must be consistent with Power requirements above (See "Power")
Wireless Interface	
Distance	Must connect the various units at a distance of 100 feet

4.3 Future Enhancement Requirements

Future Enhancement Requirements	
Software OS	Must be able to update software to support Mac OS X Leopard, Mac OS X Snow Leopard, and simple command-line Unix and Linux
Additional PC Software Features	Must be able to update to include the ability to automatically recognize major holidays and treat them as weekends Must be able to update to include the ability to download RSS feeds including weather predictions and use them Must be able to update to include the ability to control windows remotely over the internet Must be able to update to include the ability to control product through a Desktop Widget
Wireless Range	Must be able to increase to the full size of large homes
Smart Window	Must be able to include confirmation when a window changes state
Power	Must be able to include solar recharging ability
Interfacing	Must be able to add an iPhone application which can control window treatments
Security	Must be able to add strong encryption to stop window treatment opening by malicious hackers.
Window Blinds	Should be able to work with many kinds of window blinds Should be able to raise and lower blinds in addition to opening and closing

5 Low Level Design

A low-level design of each unit, subsystem, software routine, and interface is given below. The parts and schematics necessary to accomplish each subsystem are given. Reasons are given for choosing each part. A testing plan is given for each of these subsystems. Distributers for each of these parts are given in Section 6, Bill of Materials.

5.1 PC Control Unit (PCU)

5.1.1 PC application

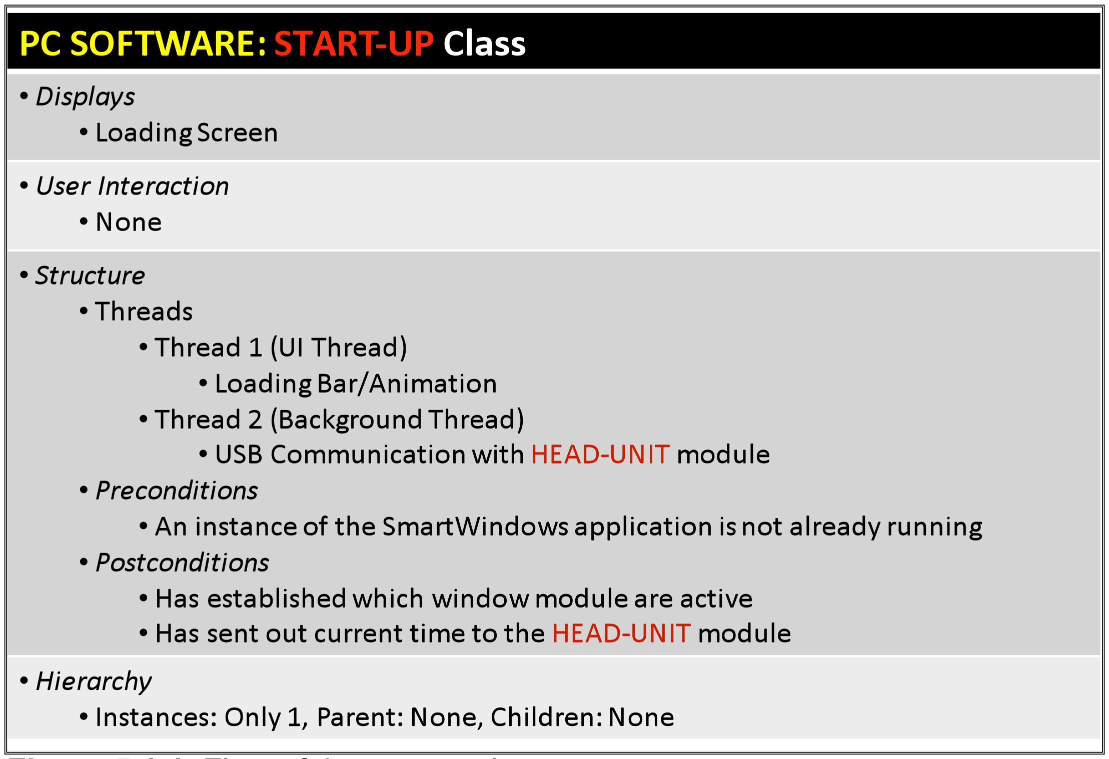
For the development of our SmartWindow PC interface, we chose to use the language Python, with PyQT and PyUSB libraries. Our decision began with how to best develop a professional looking GUI that we could rapidly develop and that would be powerful enough for our task. (We did not consider USB communication at first, assuming this could be done in every major application language.) Towards this goal, we considered the several languages and graphics plugins. The pros and cons of each considered language are shown in **Table 5.1**. Ultimately, we decided on Python as our language for this project.

Table 5.1. Summary of considered languages.

Language (Graphics Libraries)	**Pros**	**Cons**
C++ (with QT)	Professional and powerful. Cross-platform. Free.	QT with C++ is probably overkill and has steep learning curve.
C++ with C, assembly and native Win32 API calls.	Very efficient and clean. Free.	For Windows platform only, and way overkill for an application that will never actually be sold.
C++/C# (with Microsoft Visual Studio libraries)	Relatively easy to code. MFC classes are professional grade.	Deployable only on Windows. MFC classes have licensing fee. May require user to install Visual Studio DLL's. (Extra hoops in end deployment.)
JAVA (with standard Swing GUI libraries)	Professional and cross-platform. Free.	Hard to compile (without extra tools and significant effort). May require user to install Java Virtual Machine.
Python (with QT, called PyQT)	As professional looking as C++ with QT, but slightly easier and faster to program. Cross-platform. Free.	Usually an interpreted language, but we can use external tools (such as py2exe) to easily make executable.

After we decided to use Python with PyQT, we searched for USB libraries. We initially found “UsbLib” and then later “PyUsb,” a simpler thin wrapper around the “UsbLib” API. As far as open-source libraries, these two were the only options we found.

The PC application will operate in three classes. The first class occurs during startup and initializes the application by loading a settings file stored locally on the PC. The second class represents standard operation. The third class consists of child windows that allow the user to input additional information about a specific window. **Figure 5.1.1** documents the flow of the startup class. **Figure 5.1.2** shows a graphical illustration of the program creating the settings file. **Figure 5.1.3** documents the flow of the main class. **Figure 5.1.4** documents the flow of the window-child class.



PC SOFTWARE: START-UP Class

- *Displays*
  - Loading Screen
- *User Interaction*
  - None
- *Structure*
  - Threads
    - Thread 1 (UI Thread)
      - Loading Bar/Animation
    - Thread 2 (Background Thread)
      - USB Communication with HEAD-UNIT module
  - *Preconditions*
    - An instance of the SmartWindows application is not already running
  - *Postconditions*
    - Has established which window module are active
    - Has sent out current time to the HEAD-UNIT module
- *Hierarchy*
  - Instances: Only 1, Parent: None, Children: None

Figure 5.1.1. Flow of the startup class.

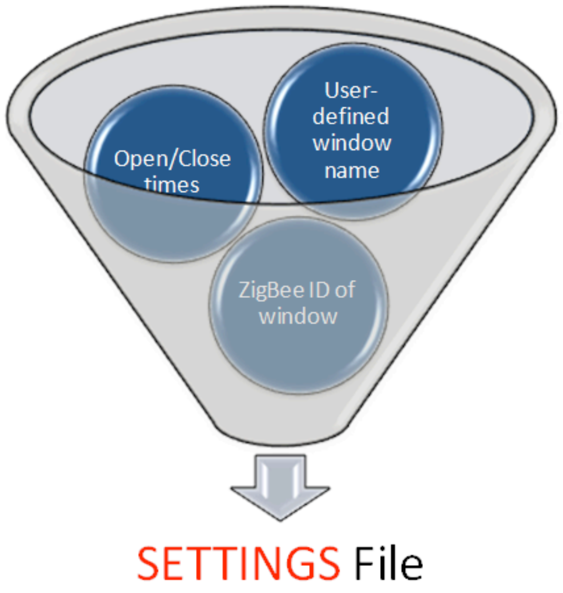
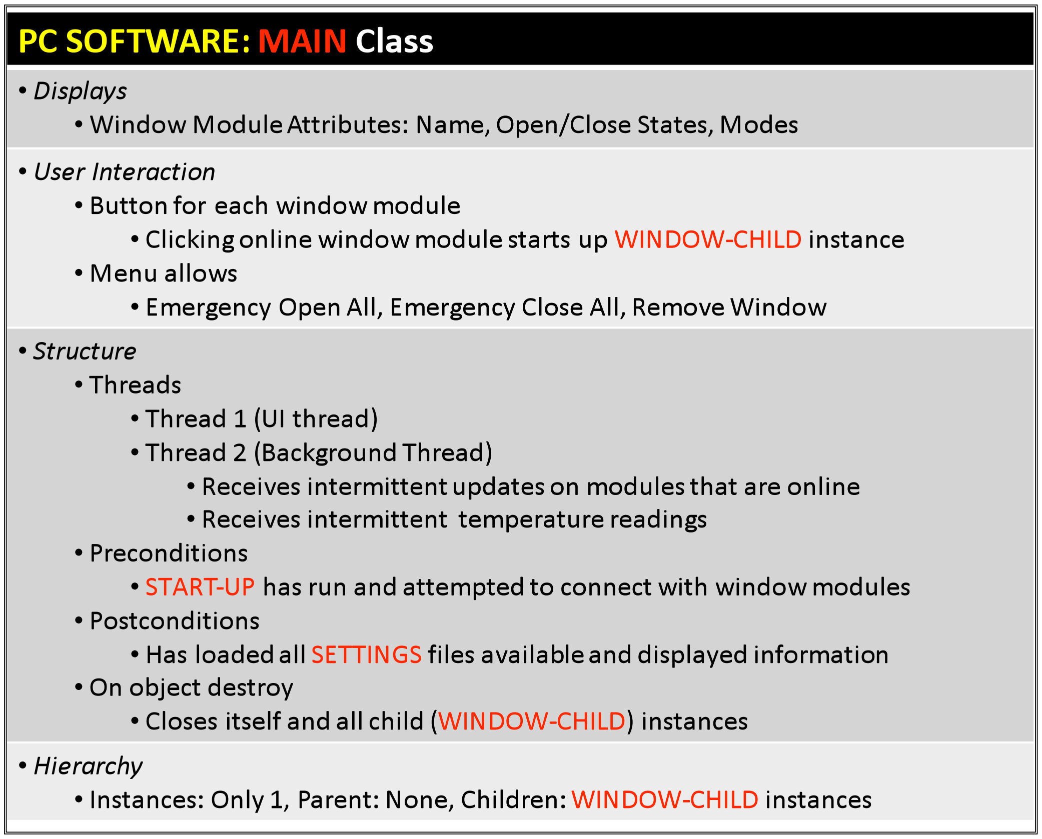


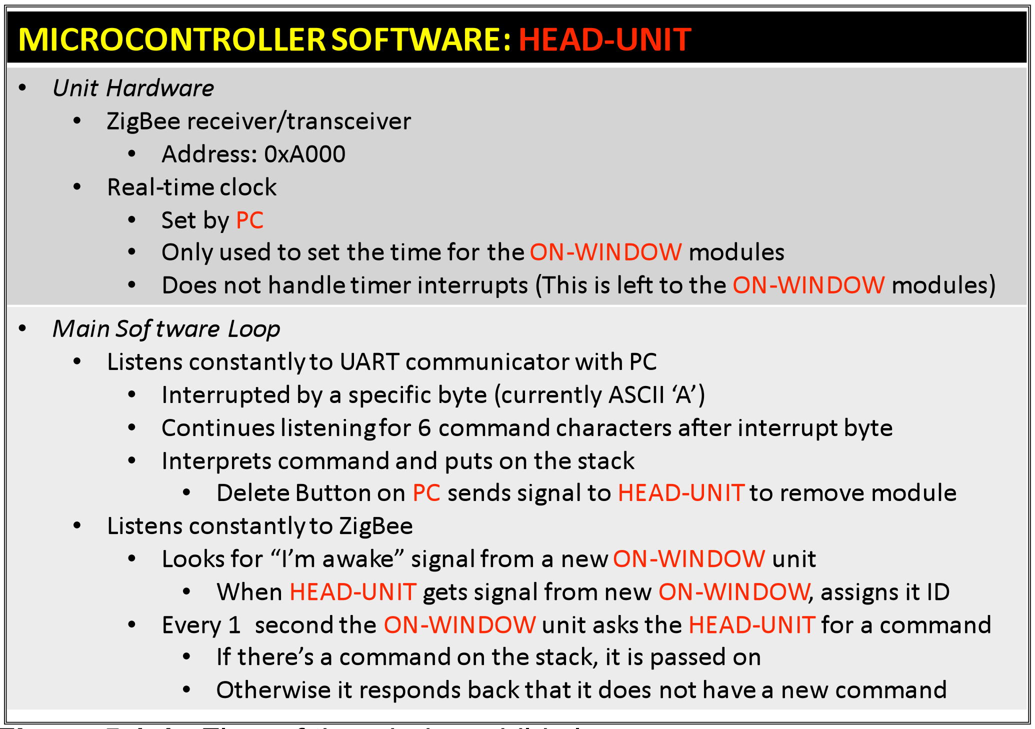
Figure 5.1.2. Settings file data diagram.



PC SOFTWARE: MAIN Class

- *Displays*
  - Window Module Attributes: Name, Open/Close States, Modes
- *User Interaction*
  - Button for each window module
    - Clicking online window module starts up WINDOW-CHILD instance
  - Menu allows
    - Emergency Open All, Emergency Close All, Remove Window
- *Structure*
  - Threads
    - Thread 1 (UI thread)
    - Thread 2 (Background Thread)
      - Receives intermittent updates on modules that are online
      - Receives intermittent temperature readings
  - Preconditions
    - START-UP has run and attempted to connect with window modules
  - Postconditions
    - Has loaded all SETTINGS files available and displayed information
  - On object destroy
    - Closes itself and all child (WINDOW-CHILD) instances
- *Hierarchy*
  - Instances: Only 1, Parent: None, Children: WINDOW-CHILD instances

Figure 5.1.3. Flow of the main class.



MICROCONTROLLER SOFTWARE: HEAD-UNIT

- *Unit Hardware*
  - ZigBee receiver/transceiver
    - Address: 0xA000
  - Real-time clock
    - Set by PC
    - Only used to set the time for the ON-WINDOW modules
    - Does not handle timer interrupts (This is left to the ON-WINDOW modules)
- *Main Sof tware Loop*
  - Listens constantly to UART communicator with PC
    - Interrupted by a specific byte (currently ASCII ‘A’)
    - Continues listening for 6 command characters after interrupt byte
    - Interprets command and puts on the stack
      - Delete Button on PC sends signal to HEAD-UNIT to remove module
  - Listens constantly to ZigBee
    - Looks for “I’m awake” signal from a new ON-WINDOW unit
      - When HEAD-UNIT gets signal from new ON-WINDOW, assigns it ID
    - Every 1 second the ON-WINDOW unit asks the HEAD-UNIT for a command
      - If there’s a command on the stack, it is passed on
      - Otherwise it responds back that it does not have a new command

Figure 5.1.4. Flow of the window-child class.

Outside of the USB communication parts of our PC Program there are two major types of code we will check over: deterministic commands and highly variable commands. Out of necessity, we will use two different methods for testing these two types of code.

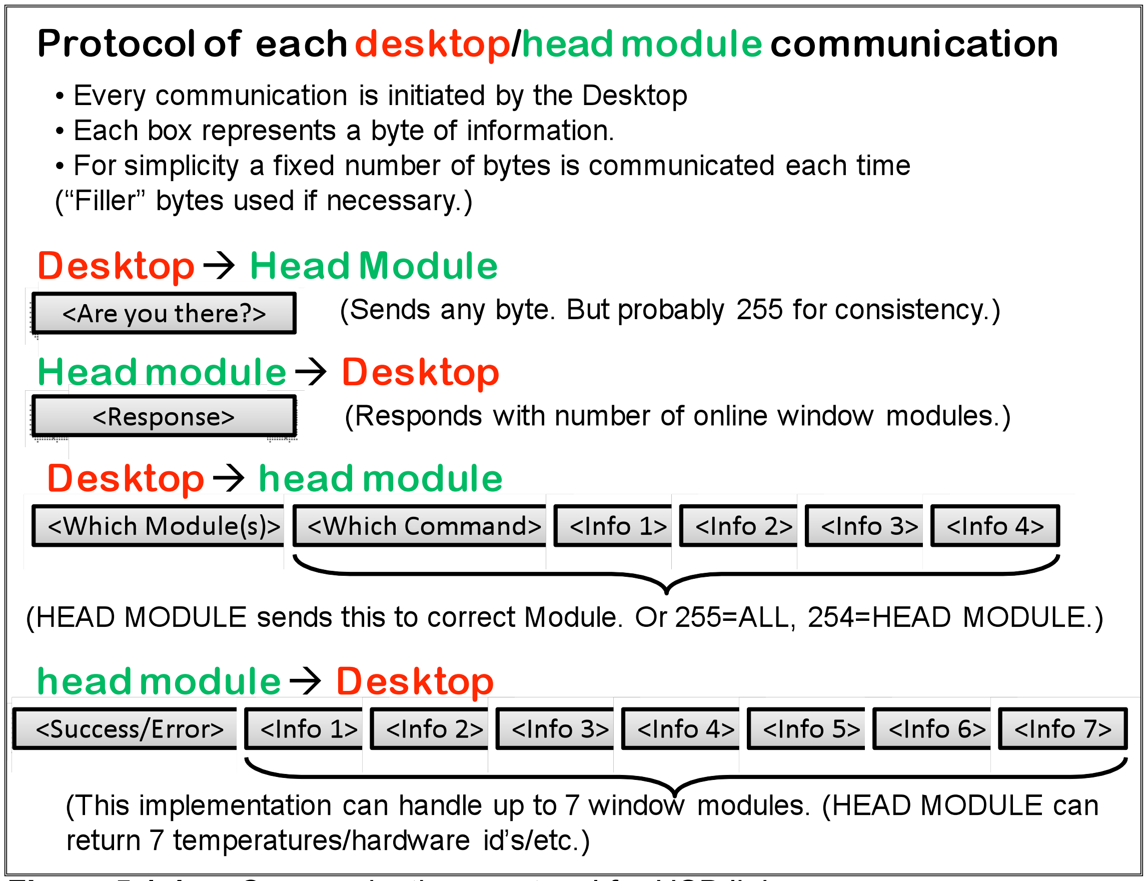
Deterministic commands, such as “change mode to ECO,” “delete a window,” or “change the name of a window” will essentially be tested using comprehensive testing. Every possible command will be sent to the microcontroller or PC and we will ensure that each command is successfully executed.

Other commands, such as the custom timing for the “Automatic Mode” of our windows, is customizable to the point of being nearly infinitely variable. Thus, we will have to settle for the systematic testing as extensively as our patience allows. For instance, after verifying that all eight open/close times on our form are programmatically identical, we will enter extensive day and times and take note of the output bytes. By using certain built-in objects, such as the python time-entry widget, we can be confident that it will not be possible to enter in unintelligible times (such as 25:61 o’clock), as many other programmers have verified this is not possible with the widget. Whenever possible we have used reliable blocks such as this.

In addition, at least two of our team will complete an inspection and walkthrough of the source code together, in keeping with the principles of pair programming. A combination of clean programming, carefully checked over, and extensive testing of major cases will be sufficient in our case. Writing computer testing routines is not feasible in our timeframe.

5.1.2 USB Interface

The USB interface will consist of asynchronous serial communication between the PC and the main board. The PC will interact with the USB channel through a virtual COM port. This USB signal will be converted into simple asynchronous serial communication through the FTDI part described in Section 5.1.3.11. The microcontroller will send and receive bytes from the FTDI part through its on-board USART. **Figure 5.1.4 a-g** shows a summary of the communication protocol between the microcontroller and the PC.



Protocol of each desktop/head module communication

- Every communication is initiated by the Desktop
- Each box represents a byte of information.
- For simplicity a fixed number of bytes is communicated each time

("Filler" bytes used if necessary.)

Desktop → Head Module

<Are you there?> (Sends any byte. But probably 255 for consistency.)

Head module → Desktop

<Response> (Responds with number of online window modules.)

Desktop → head module

<Which Module(s)> $\underbrace{\text{<Which Command> <Info 1> <Info 2> <Info 3> <Info 4>}}$

(HEAD MODULE sends this to correct Module. Or 255=ALL, 254=HEAD MODULE.)

head module → Desktop

<Success/Error> $\underbrace{\text{<Info 1> <Info 2> <Info 3> <Info 4> <Info 5> <Info 6> <Info 7>}}$

(This implementation can handle up to 7 window modules. (HEAD MODULE can return 7 temperatures/hardware id's/etc.)

Figure 5.1.4.a. Communications protocol for USB link.

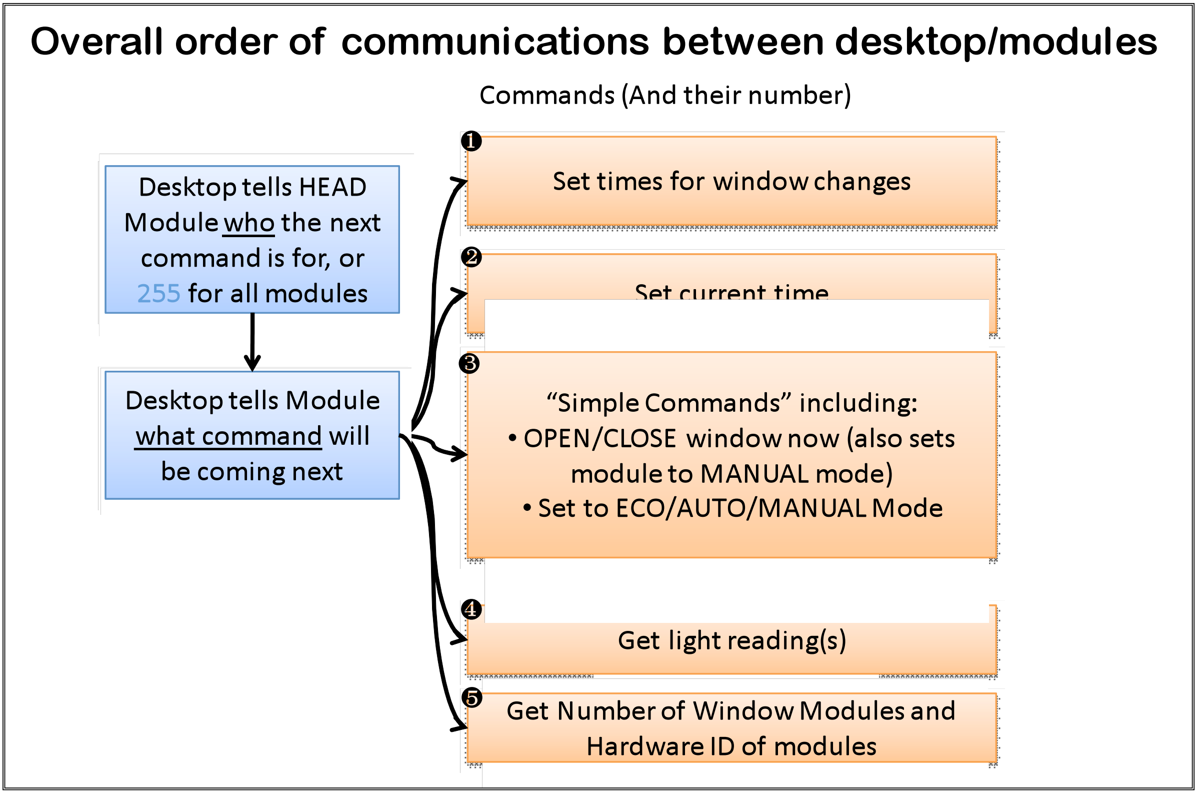
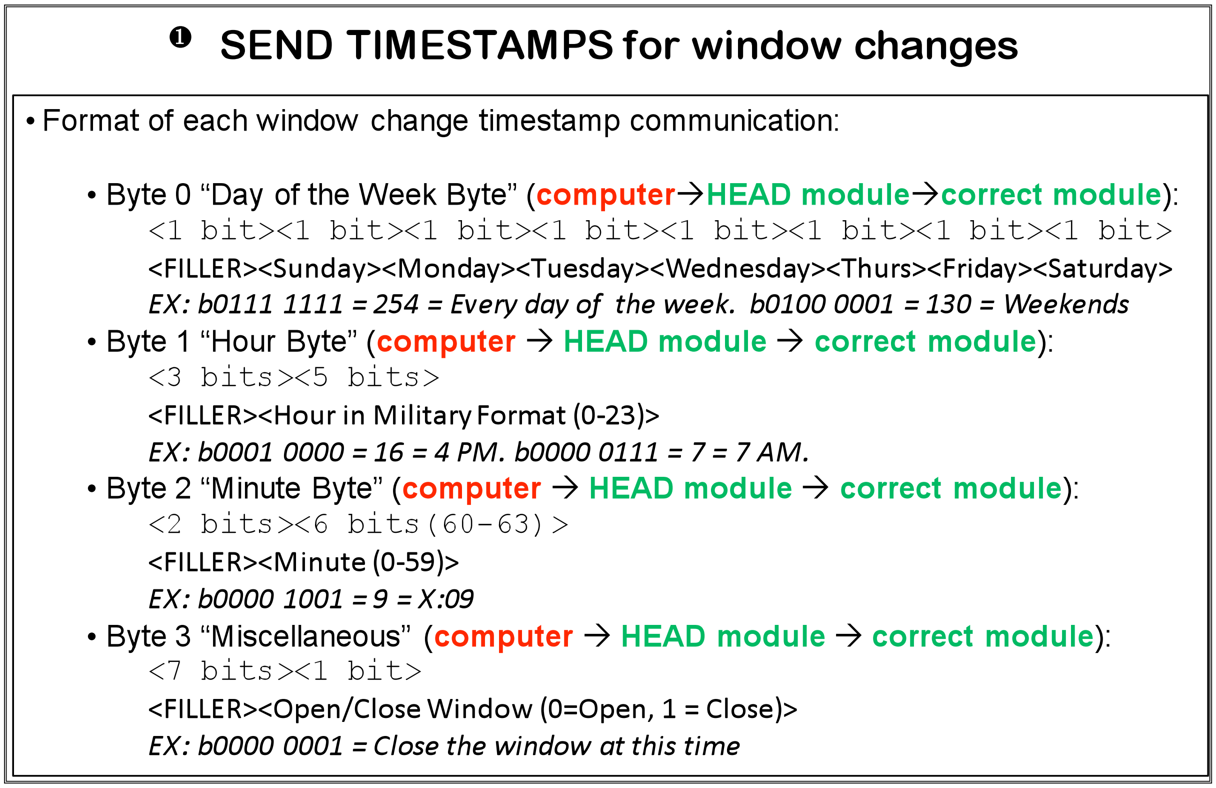

Overall order of communications between desktop/modules
Commands (And their number)
Desktop tells HEAD Module who the next command is for, or 255 for all modules
Desktop tells Module what command will be coming next
❶ Set times for window changes
❷ Set current time
❸ "Simple Commands" including:
• OPEN/CLOSE window now (also sets module to MANUAL mode)
• Set to ECO/AUTO/MANUAL Mode
❹ Get light reading(s)
❺ Get Number of Window Modules and Hardware ID of modules

Figure 5.1.4.b. Communications protocol for USB link.



❶ SEND TIMESTAMPS for window changes

- Format of each window change timestamp communication:
  - Byte 0 "Day of the Week Byte" (computer→HEAD module→correct module):
    `<1 bit><1 bit><1 bit><1 bit><1 bit><1 bit><1 bit><1 bit>`
    <FILLER><Sunday><Monday><Tuesday><Wednesday><Thurs><Friday><Saturday>
    EX: b0111 1111 = 254 = Every day of the week. b0100 0001 = 130 = Weekends
  - Byte 1 "Hour Byte" (computer → HEAD module → correct module):
    `<3 bits><5 bits>`
    <FILLER><Hour in Military Format (0-23)>
    EX: b0001 0000 = 16 = 4 PM. b0000 0111 = 7 = 7 AM.
  - Byte 2 "Minute Byte" (computer → HEAD module → correct module):
    `<2 bits><6 bits(60-63)>`
    <FILLER><Minute (0-59)>
    EX: b0000 1001 = 9 = X:09
  - Byte 3 "Miscellaneous" (computer → HEAD module → correct module):
    `<7 bits><1 bit>`
    <FILLER><Open/Close Window (0=Open, 1 = Close)>
    EX: b0000 0001 = Close the window at this time

Figure 5.1.4.c. Communications protocol for USB link.

❷ SET WINDOW MODULE DAY AND TIME

• Based on PC time. (PC time/date must accurate.)
• The window module's times will be set when modules are added and at periodic intervals thereafter.
• Only the day of the week will be told to the window module. (The actual date/year is irrelevant to all features)

• Byte 0 (**computer** → **HEAD module** → **correct module**):
  <Current Day of the Week> (0 = Sunday, 1 = Monday… 6 = Saturday)
  `<3 bits unsigned> (7→"get time fail", others ignored)`
  <Hour in Military Format> (0-23)
  `<5 bits unsigned> (24-31 → invalid)`

Byte 1 (**computer** → **HEAD module** → **correct module**):
  <FILLER = `2 bits>`
  <Current Minute> (0-59)
  `<6 bits unsigned> (60-63 → invalid)`

Byte 2 (**computer** → **HEAD module** → **correct module**):
  <FILLER = `2 bits>`
  <Seconds> (0-59)
  `<6 bits unsigned> (60-63 → invalid)`

Byte 3 (**computer** → **HEAD module** → **correct module**):
  <FILLER = 8 `bits>`

Figure 5.1.4.d. Communications protocol for USB link.

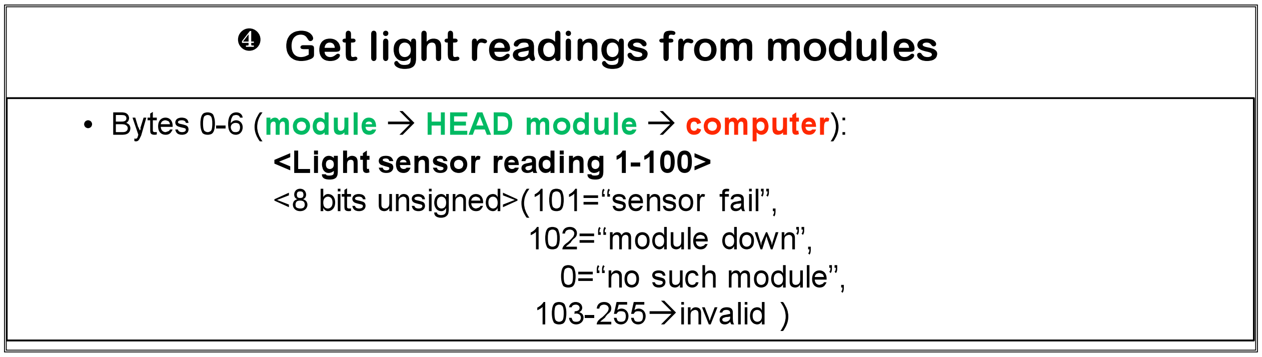
❸ "SIMPLE COMMANDS"

• Each of these commands sends one byte of information to the module.

• Byte 0 (**computer** → **HEAD module** → **correct module**):
  <Command>
  <8 bits unsigned>
  0 → Open Window (Also switch to MANUAL mode)
  1 → Close Window (Also switch to MANUAL mode)
  2 → Switch to ECO mode
  3 → Switch to AUTO mode
  4 → Switch to MANUAL mode

• Bytes 1,2,3 (**computer** → **HEAD module** → **correct module**):
  <FILLER = 8 bits>

Figure 5.1.4.e. Communications protocol for USB link.

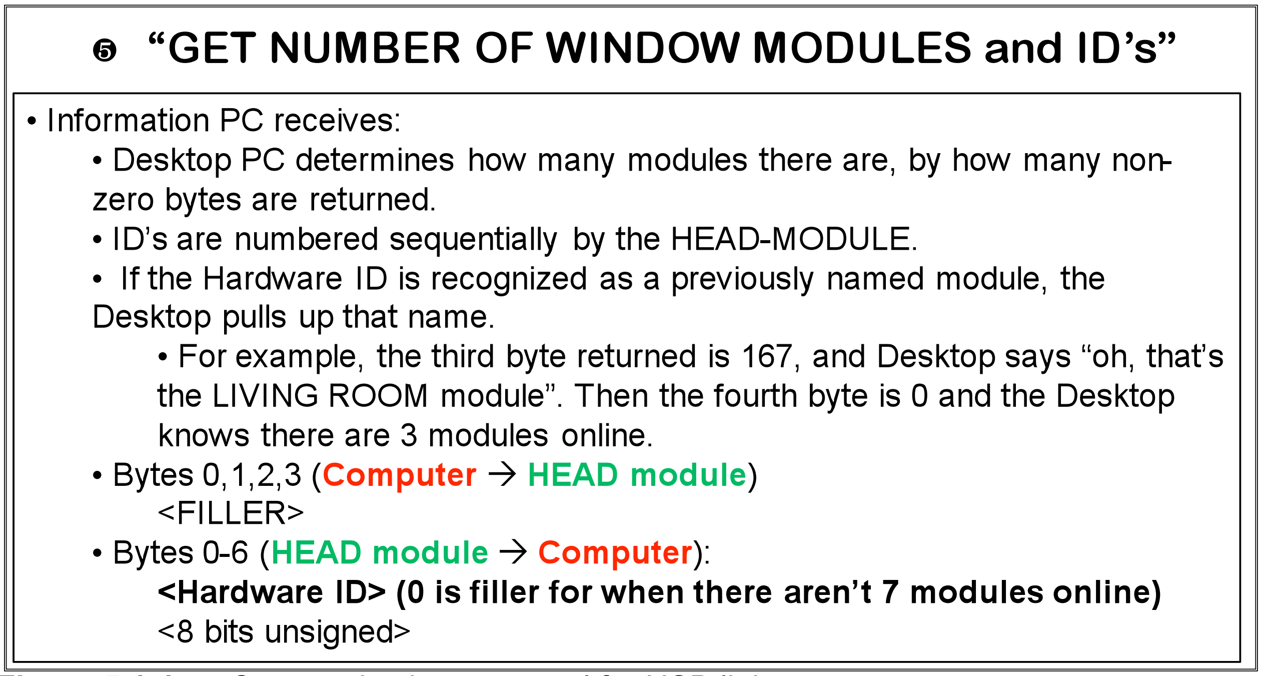


❹ Get light readings from modules

- Bytes 0-6 (**module → HEAD module → computer**):

  <Light sensor reading 1-100>
  <8 bits unsigned>(101=“sensor fail”,
  102=“module down”,
  0=“no such module”,
  103-255→invalid)

Figure 5.1.4.f. Communications protocol for USB link.



❺ “GET NUMBER OF WINDOW MODULES and ID’s”

- Information PC receives:
  - Desktop PC determines how many modules there are, by how many non-zero bytes are returned.
  - ID’s are numbered sequentially by the HEAD-MODULE.
  - If the Hardware ID is recognized as a previously named module, the Desktop pulls up that name.
    - For example, the third byte returned is 167, and Desktop says “oh, that’s the LIVING ROOM module”. Then the fourth byte is 0 and the Desktop knows there are 3 modules online.
  - Bytes 0,1,2,3 (**Computer → HEAD module**)
    <FILLER>
  - Bytes 0-6 (**HEAD module → Computer**):
    <Hardware ID> (0 is filler for when there aren’t 7 modules online)
    <8 bits unsigned>

Figure 5.1.4.g. Communications protocol for USB link.

To test this USB protocol, a variety of bytes will be sent from the microcontroller to a terminal. If all these bytes can be received correctly, than microcontroller transmission and PC reception have been verified. Then, a variety of bytes will be sent from the PC program to a microcontroller and displayed on an LCD screen. If these bytes are received correctly, then microcontroller reception and PC transmission will have been verified. Finally, the microcontroller and the PC program will be connected together for two-way communication. If successful, the USB protocol will pass this test.

5.1.3 Main Board

The main board will be designed as a printed circuit board. As described in Section 3 above, this main board will have certain standard parts and unit-specific peripherals. Both the standard parts and the peripherals of the board are described below. The main board testing will consist of testing each of the individual subsystems and interfaces that make up the board. These test plans are also described below.

5.1.3.1 Microcontroller

The microcontroller used as the embedded intelligence for this project must have 33 I/O pins, a universal asynchronous receiver transmitter (UART), non-volatile memory, and capable of 3.3V operation. Since the microcontroller software is written with the BoostC compiler, a Microchip PIC18 model microcontroller was selected. The lowest cost PIC18 that meets our design requirements is the PIC18LF4620. **Figure 5.1.5** shows a schematic of this microcontroller with the pins connected to I/O signals. The naming conventions applied to these I/O signals will be applied throughout the document.

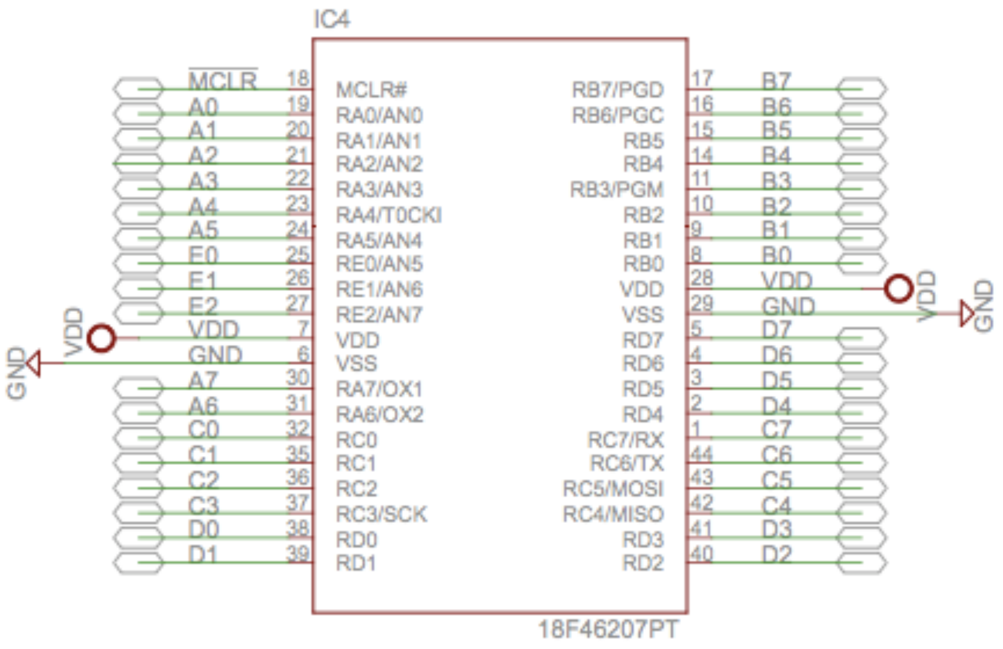
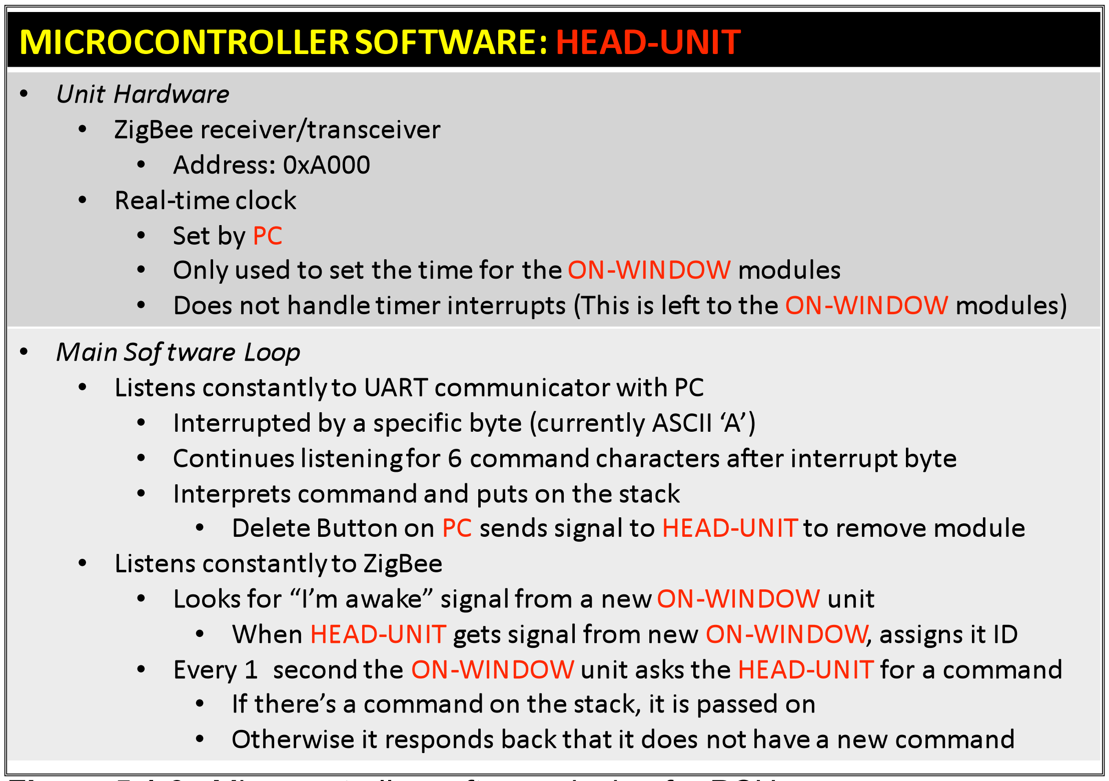


Figure 5.1.5. Microcontroller Schematic.

Since each subsystem relies on proper microcontroller operation, testing the operation of this microcontroller will consist of many implicit tests along the way. To specifically test the soldering connections of the microcontroller, the ports can be configured as inputs and connected to V_{DD} and ground successively. An LCD screen can display the values on the ports and insure they are operating as expected.

5.1.3.2 Microcontroller Software

The design for the microcontroller software for the PCU is shown in **Figure 5.1.6**. The software will consist of a main loop that listens for UART communication from the PC and ZigBee communication. When a PC instruction is received, it is placed on a stack indicating which window the instruction is for. Once a communication link is achieved with that window, the instructions are removed from the stack and sent to the window. If the window requests the join a network, the PCU orchestrates this process and stores the window's address.



MICROCONTROLLER SOFTWARE: HEAD-UNIT

- *Unit Hardware*
  - ZigBee receiver/transceiver
    - Address: 0xA000
  - Real-time clock
    - Set by PC
    - Only used to set the time for the ON-WINDOW modules
    - Does not handle timer interrupts (This is left to the ON-WINDOW modules)
- *Main Software Loop*
  - Listens constantly to UART communicator with PC
    - Interrupted by a specific byte (currently ASCII 'A')
    - Continues listening for 6 command characters after interrupt byte
    - Interprets command and puts on the stack
      - Delete Button on PC sends signal to HEAD-UNIT to remove module
  - Listens constantly to ZigBee
    - Looks for "I'm awake" signal from a new ON-WINDOW unit
      - When HEAD-UNIT gets signal from new ON-WINDOW, assigns it ID
    - Every 1 second the ON-WINDOW unit asks the HEAD-UNIT for a command
      - If there's a command on the stack, it is passed on
      - Otherwise it responds back that it does not have a new command

Figure 5.1.6. Microcontroller software design for PCU.

To test the microcontroller code, the code will be run many times with a variety of inputs and scenarios. All attempts will be made to create unusual situations for the program to handle. Program crashes will be monitored and corrected.

5.1.3.3 ZigBee Circuit

The ZigBee circuit must contain an integrated circuit capable of performing ZigBee radio frequency (RF) communication. This IC must be able to interface to the microcontroller through an SPI interface. To perform these functions, we chose the ATMEL ZigBee part AT86R231. The ZigBee circuit also contains an antenna, an oscillator, and associated impedance-matching traces. These ZigBee circuit components were designed by Professor Mike Schafer of the University of Notre Dame and are reproduced here exactly. The schematic for this ZigBee circuit is shown in **Figure 5.1.7**.

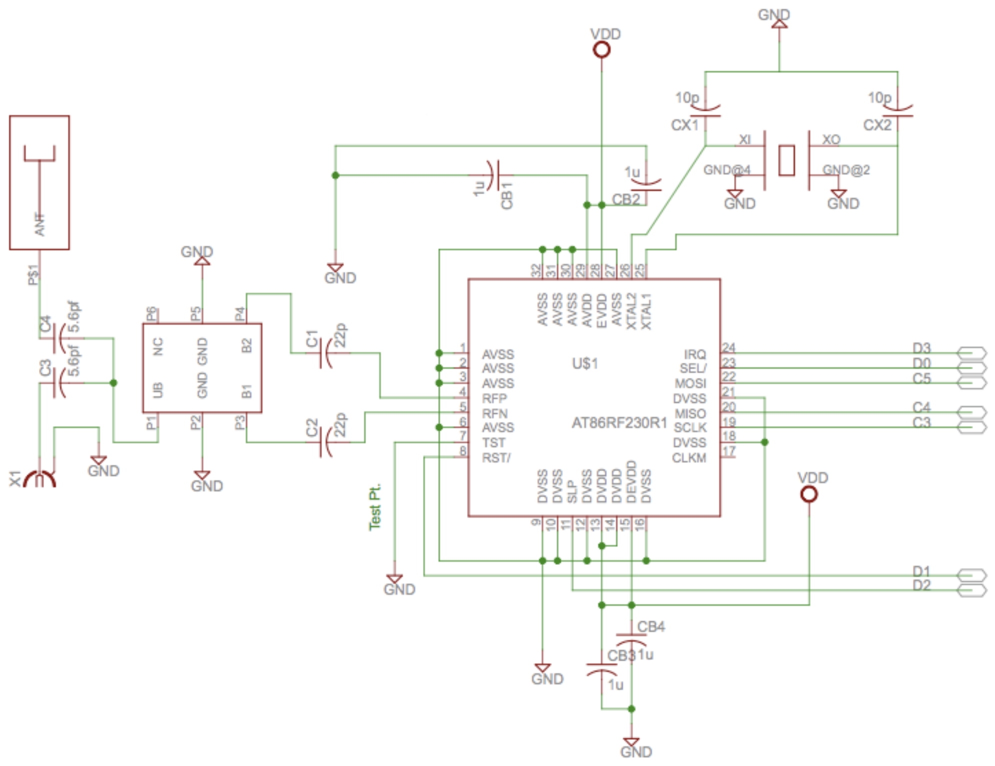


Figure 5.1.7. ZigBee Circuit Schematic.

To test this ZigBee circuit, a two-step process will be used. First, the SPI interface will be tested by writing and reading to registers on the ATMEL chip. Once this can be confirmed, the ZigBee transmission will be tested. Messages will be sent over ZigBee to a packet sniffer. This will confirm transmission. Then, packets will be received from a packet sender to test reception. Finally, two ZigBee devices will be connected in two-way communication.

5.1.3.4 ZigBee Wireless Interface

The ZigBee circuit has two interfaces. The SPI interface connects the ATMEL ZigBee IC to the microcontroller. This interface is a standard protocol SPI interface. Standards describing SPI operation and the sending of bytes through SPI are readily available from a variety of sources.

The ATMEL ZigBee IC communicates with the ZigBee units on other main boards through a ZigBee protocol RF wireless interface. The ZigBee protocol we are using is compliant with the IEEE 802.15.4 standard. ZigBee units connected to the network are given a unique two-byte short address by the PCU. The ZigBee modules are set to extended mode, given them the capability for automatic address filtering, automatic message acknowledgement, and automatic retry until acknowledgment. In any particular message, the first byte represents the type of instruction and subsequent bytes represent additional information associated with that instruction. These messages follow the protocol set forward for USB communication set forth in **Figure 5.1.4** above.

The ZigBee interface will be tested as part of the ZigBee circuit test in the previous section.

5.1.3.5 Ceramic Resonator

To save power, our microcontrollers will be set to sleep mode in between computation bursts. To decrease power usage, the time per instruction must be minimal so as to increase the sleep duty cycle. To ensure efficient operation of the microcontroller, we will use an external ceramic resonator to provide a 20MHz clock to the microcontroller. The appropriate oscillator is available from Murata through Digi-Key. The part number and ordering information are provided in the Bill of Materials below. The oscillator circuit is shown in **Figure 5.1.8**.

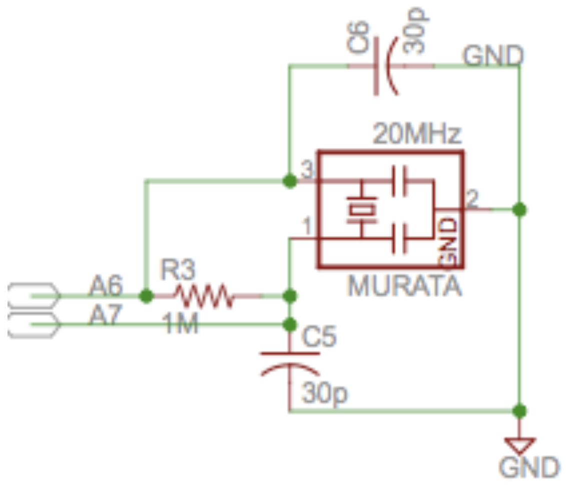


Figure 5.1.8. Oscillator Circuit.

This circuit will not need to be tested explicitly since correct microcontroller operation will indicate successful clock generation.

5.1.3.6 Programmer Circuit

The programmer circuit allows the Melabs programmer card to interface to our PIC microcontroller. This circuit was designed by Professor Mike Schafer of the University of Notre Dame and is included in **Figure 5.1.9** with permission.

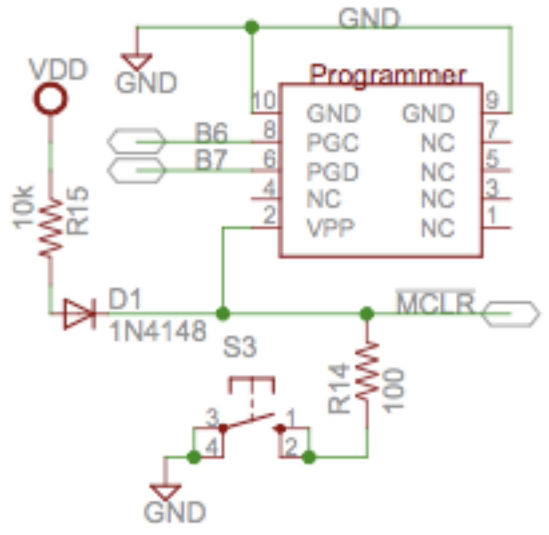


Figure 5.1.9. Programmer Circuit

This circuit will not need to be tested explicitly since correct microcontroller operation will indicate successful programming operation.

5.1.3.7 Serial EEPROM

The main board must be able to maintain information about connect window units when power is disconnected. While the PIC18 microcontroller has limited non-volatile EEPROM memory, our needs may exceed the limited size of this memory. Therefore, a external serial EEPROM is included in the design to provide additional memory if necessary. The serial EEPROM chip chosen is the Microchip 25LC640, which provides an additional 64kbits of memory. This chip connects to the microcontroller through an SPI interface. The circuit connecting this EEPROM chip to our microcontroller is shown in **Figure 5.1.10**.

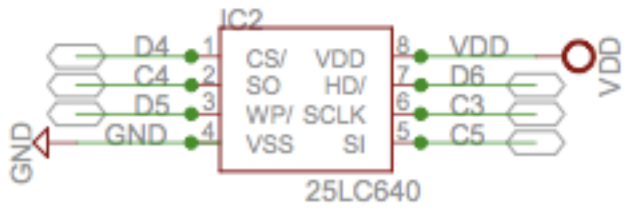


Figure 5.1.10. Serial EEPROM schematic.

The EEPROM device will be tested by simply writing to and reading from the device a variety of times. Then, data will be written to the device and the power will be shut off. The power will be turned back on several days later and the data existence will be confirmed.

5.1.3.8 Serial EEPROM SPI Interface

The serial EEPROM chip connects to the microcontroller through a standard protocol SPI interface. Standards describing SPI operation and the sending of bytes through SPI are readily available from a variety of sources. The SPI interface will be tested as part of the EEPROM device test listed above.

5.1.3.9 Real Time Clock

When in automatic mode, the OWUs will be asked to open and close at certain times and days of the week. In order to do this, they will need an IC capable of keeping track of the current date and time. Therefore, each OWU will be equipped with a Dallas DS1305 real time clock device. Since the PCU is responsible for syncing these real time clocks to the PC clock, the OWU will also need a real time clock chip. The DS1305 connects to the microcontroller through an SPI interface.

Figure 5.1.11 shows the schematic that governs the real time clock operation. This real time clock requires a 32.768 kHz crystal oscillator. The SER3205 has been used for this purpose.

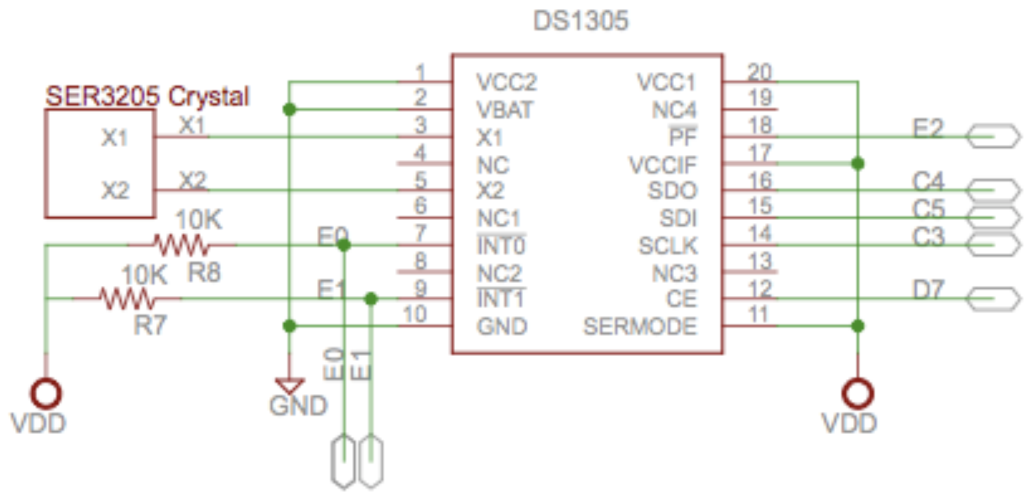


Figure 5.1.11. Real Time Clock Schematic.

The real time clock will be tested by placing the current time on it through the SPI interface. Then, the device will be allowed to run for a significant interval of time. Meanwhile, the elapsed time will be measured with an external clock. After the interval of time has passed, the time will be read from the device and compared to the known actual time.

5.1.3.10 Real Time Clock SPI Interface

The real time clock connects to the microcontroller through an SPI interface. Standards describing SPI operation and the sending of bytes through SPI are readily available from a variety of sources. The SPI interface will be tested as part of the real time clock device test listed above.

5.1.3.11 USB/UART FTDI Circuit

To convert the asynchronous serial messages transmitted by the microcontroller into the standard USB signals transmitted by the PC through the COM port, an intermediary is needed. The device we have chosen to do this is the FT232RI from FTDI. Professor Mike Schafer of the University of Notre Dame designed the circuit governing the operation of this device. The circuit is used with permission and shown in **Figure 5.1.12**.

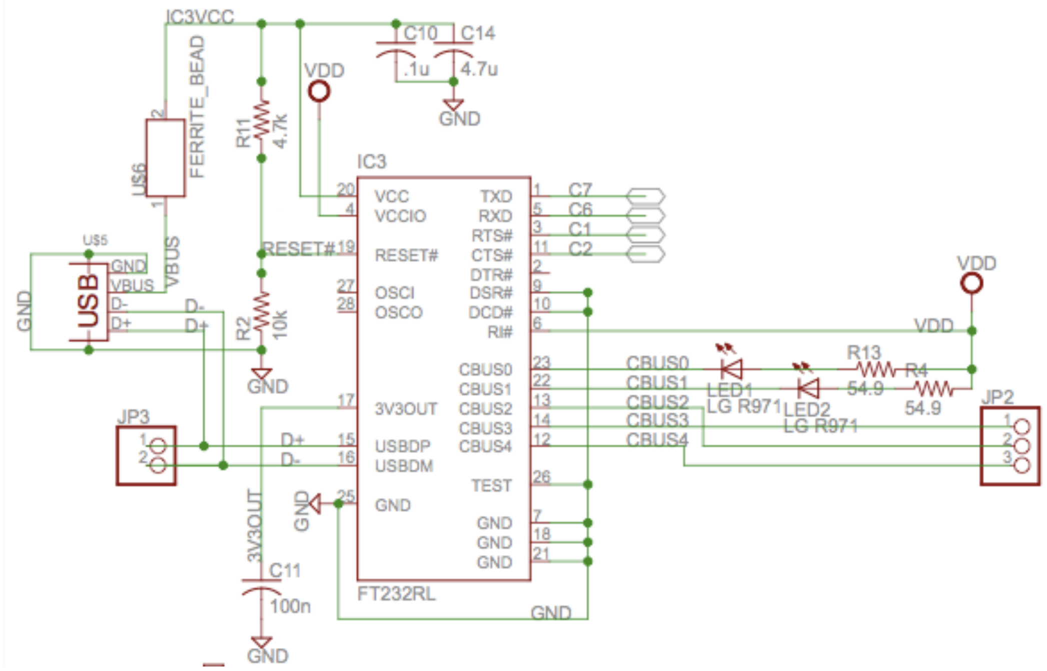


Figure 5.1.12. FT232RL schematic.

This USB signal from the PC also carries a 5V signal along with it. This 5V signal will be used to power the PCU main board. This 5V voltage is run directly into a 3.3V regulator, as shown in **Figure 5.1.13**.

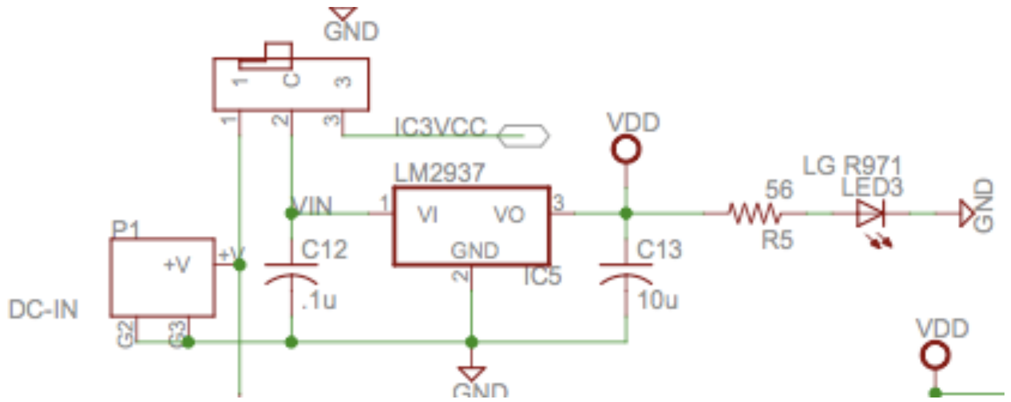


Figure 5.1.13. Regulator circuit.

For testing the communications properties of this circuit, bytes will be sent and received to the PC. This process will be part of the USB interface test already described above. To test the power generation portion of the circuit, we will simply check to make sure the board is receiving power.

5.2 On-Window Unit (OWU)

5.2.1 Motor Block

The motor block will consist of a DC motor, window blinds, two limit switches, two ID hubs, a rubber spider coupling, and a steel rod. The DC motor will drive the window blinds, causing the shaft on the blinds to rotate. The ID hubs and spider coupling will couple the window blind shaft to a bent steel rod. As the shaft turns the steel rod will come into contact with the limit switches. A press of a limit switch will indicate that the window is fully opened or closes. To test this circuit, the motor will be operated several hundred times. For a successful test the motor must start and stop exactly 100% of the time. No motor faults can be tolerated.

5.2.2 Main Board

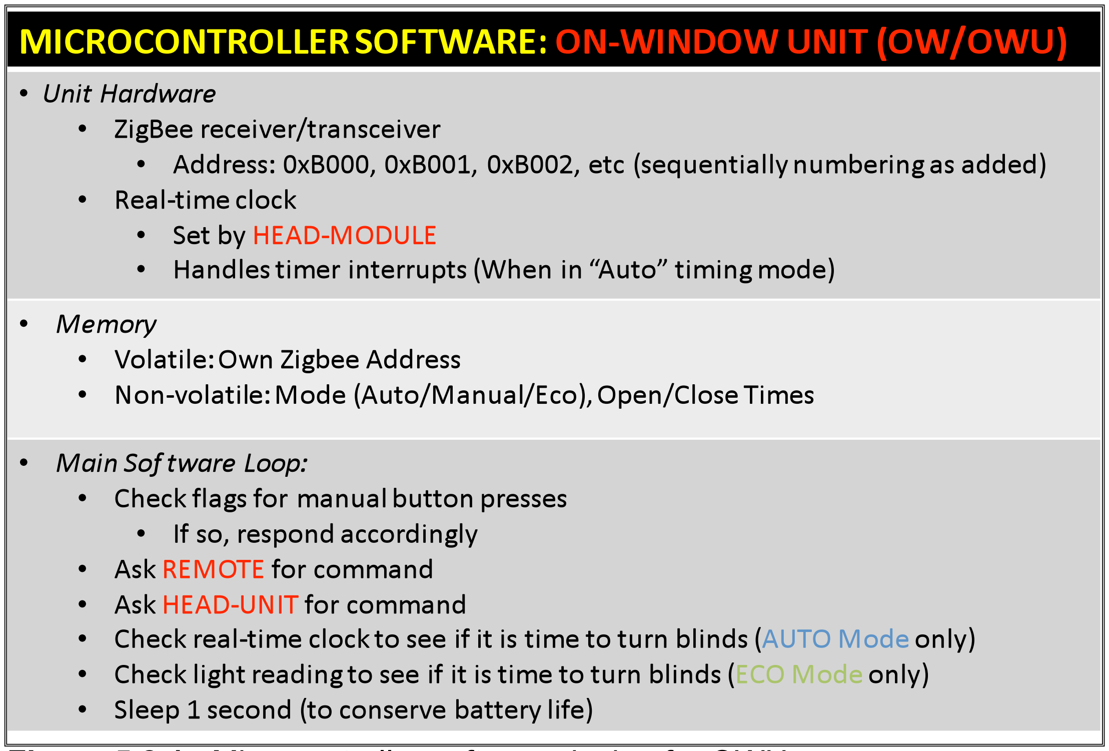
The design and testing of this system is identical to Section 5.1.3. OWU-specific peripherals will be attached as described below.

5.2.2.1 Microcontroller

Identical to Section 5.1.3.1

5.2.2.2 Microcontroller Software

The design for the microcontroller software for the OWU is shown in **Figure 5.2.1**.



MICROCONTROLLER SOFTWARE: ON-WINDOW UNIT (OW/OWU)

- *Unit Hardware*
  - ZigBee receiver/transceiver
    - Address: 0xB000, 0xB001, 0xB002, etc (sequentially numbering as added)
  - Real-time clock
    - Set by HEAD-MODULE
    - Handles timer interrupts (When in “Auto” timing mode)
- *Memory*
  - Volatile: Own Zigbee Address
  - Non-volatile: Mode (Auto/Manual/Eco), Open/Close Times
- *Main Software Loop:*
  - Check flags for manual button presses
    - If so, respond accordingly
  - Ask REMOTE for command
  - Ask HEAD-UNIT for command
  - Check real-time clock to see if it is time to turn blinds (AUTO Mode only)
  - Check light reading to see if it is time to turn blinds (ECO Mode only)
  - Sleep 1 second (to conserve battery life)

Figure 5.2.1. Microcontroller software design for OWU.

To test the microcontroller code, the code will be run many times with a variety of inputs and scenarios. All attempts will be made to create unusual situations for the program to handle. Program crashes will be monitored and corrected.

5.2.2.3 ZigBee Circuit

Identical to Section 5.1.3.3

5.2.2.4 ZigBee Wireless Interface

Identical to Section 5.1.3.4

5.2.2.5 Ceramic Resonator

Identical to Section 5.1.3.5

5.2.2.6 Programmer Circuit

Identical to Section 5.1.3.6

5.2.2.7 Serial EEPROM

Identical to Section 5.1.3.7

5.2.2.8 Serial EEPROM SPI Interface

Identical to Section 5.1.3.8

5.2.2.9 Real Time Clock

Identical to Section 5.1.3.9

5.2.2.10 Real Time Clock SPI Interface

Identical to Section 5.1.3.10

5.2.2.11 DC Input Circuit

Because there is no USB connection to the OWU, they must derive their power from another source. Connecting each window to a wall outlet would use a disproportionate number of outlets. Therefore, rechargeable batteries must power the system. Due to their combination of safety and longevity, we have chosen NiMH AA batteries. To reach the voltage required by the DC motor, we will need 8 batteries in series.

To regulate these voltages down to the 3.3V required by the main board, a voltage regulator will be used. A voltage regulator capable of this conversion is the LM2937ES-3.3. To monitor battery voltage, the battery voltage must be scaled down to a safe level and run into the microcontroller analog to digital converter. To achieve this level conversion, a 4:1 resistor divider is used.

This DC input circuit is shown in **Figure 5.2.2**.

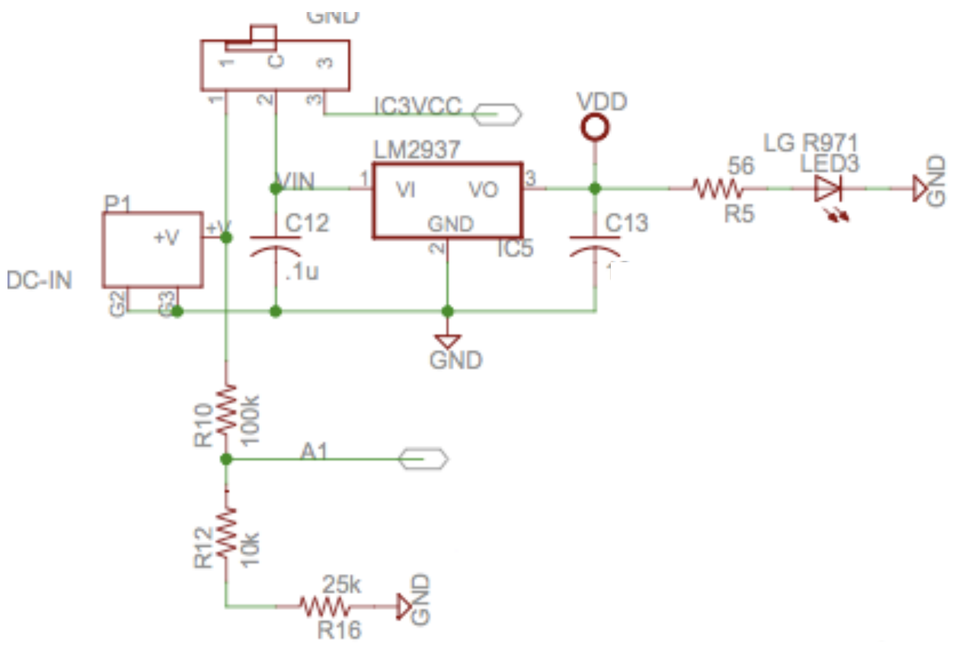


Figure 5.2.2. DC input circuit.

Testing the DC input circuit will be straightforward. A voltmeter will be used to insure that the board is receiving power.

5.2.2.12 Light Sensor Board

When the OWU is in eco mode, it must be able to respond to changing light levels. Therefore, a small light sensing board has been designed. This board will be fixed to the window surface. Jumper wires connect this board to the main board.

The light sensor board contains a TEMT6000 phototransistor. This transistor uses light levels in place of its gate. The fully designed light sensor board is shown **Figure 5.2.3**. The jumper on the main board that connects to the light sensor board is shown in **Figure 5.2.4**.

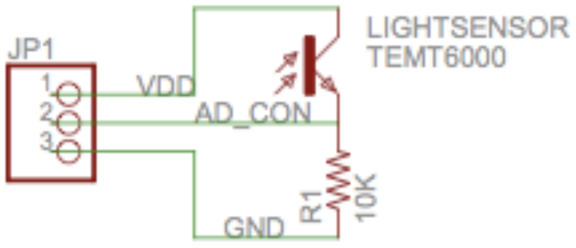


Figure 5.2.3. Light sensor board.

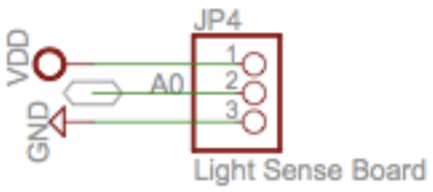


Figure 5.2.4. Main board light sensor jumper.

To test the light sensor board, the voltage output will be converted to a digital signal using the A/D converter. This digital reading will be displayed on the LCD screen. The light levels on the sensor will be varied by changing the room lighting conditions. We will verify that the light readings on the screen agree with the light levels on the sensor.

5.2.2.13 Button Card

The OWU will need three buttons. One button will be used to connect the OWU to the base station. The other two buttons will be used open and close the window blinds. Since these buttons must be in a place accessible to the user, they cannot be attached to the main board. Instead, the buttons are placed on a separate button card. This card supports up to 6 buttons, but only three will be needed for the OWU. **Figure 5.2.5** contains the schematic for the button board, and **Figure 5.2.6** contains the main board jumper to which this button card can be attached.

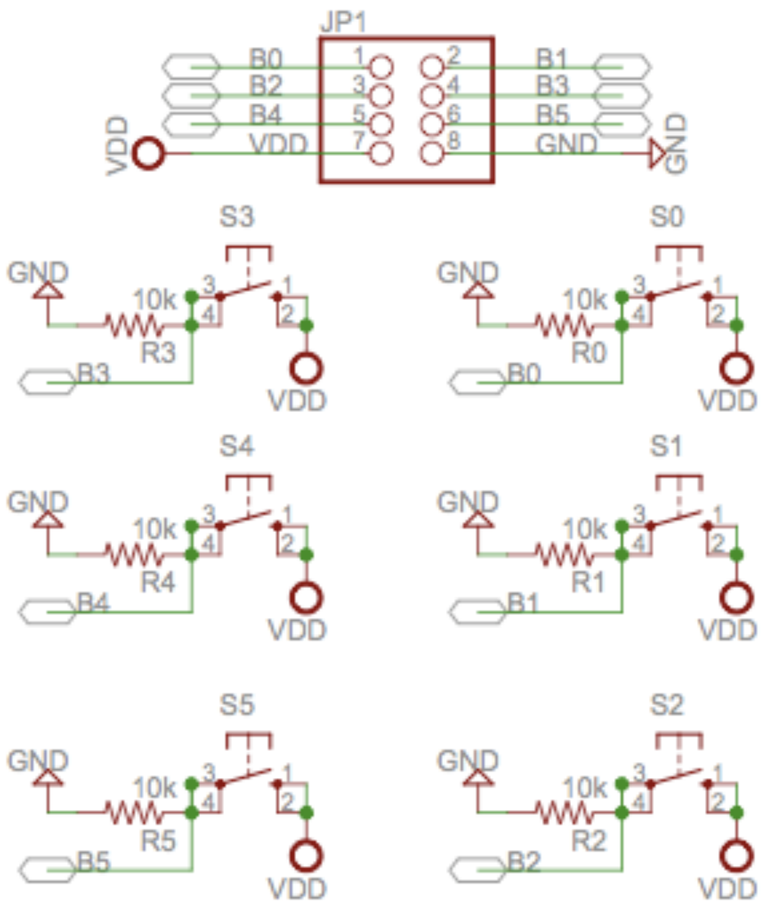


Figure 5.2.6. Button Card.

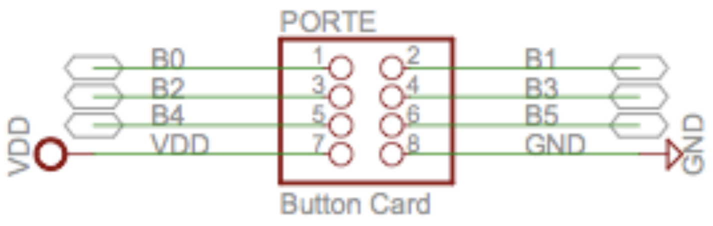


Figure 5.2.7. Button Card Main Board Jumper.

To test the button card, we will display the value on the Port B on the LCD screen. We will press various combinations of the buttons to ensure that they all work as expected.

5.2.2.14 DC/DC Converter

The motor control board requires a 5V signal. To convert the main board 3.3V signal to a 5.0V signal, a DC-DC converter device is necessary. A suitable device is the MCP1252 charge pump. The schematic governing the operation of this device was designed by Professor Mike Schafer at the University of Notre Dame and is reproduced here with permission as **Figure 5.2.8**.

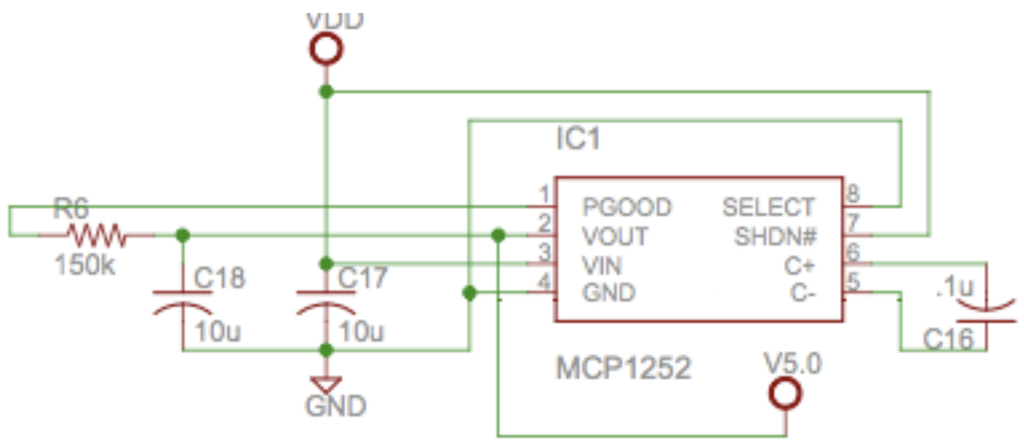


Figure 5.2.8. DC/DC converter circuit.

To test the charge pump DC/DC converter, a voltmeter will be used to make sure the appropriate 5V signal is being generated.

5.2.2.15 Motor Control Board

To control the DC motor from the microcontroller, an h-bridge and related protection circuitry are needed. The h-bridge converts logic outputs from the microcontroller into the voltages needed to drive a motor. While many h-bridges are available, a suitable h-bridge is the LM298. Because the motor deals with high voltage signals that are dangerous to the microcontroller, this motor driving circuit is placed on a physically separate board. **Figure 5.2.9** shows the motor board. The protection circuitry around the h-bridge was designed using the datasheet for the LM298. The jumper connection that interfaces to the motor control board is shown in **Figure 5.2.10**.

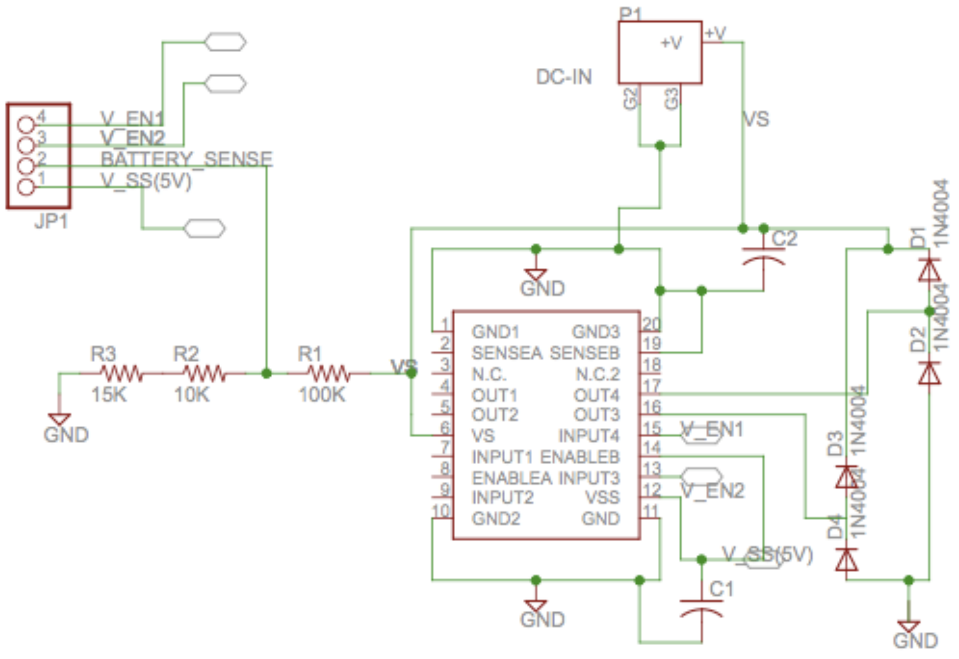


Figure 5.2.9. Motor control board.

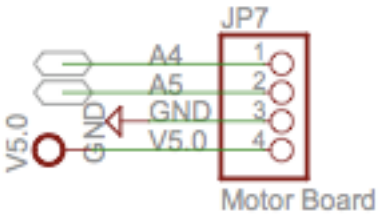


Figure 5.2.10. Motor control board to main board jumper connection.

No specific test will be needed for the h-bridge circuit since the operation of the motor is being tested above in the motor block test. If the motor block is able to respond appropriately with certainty to microcontroller signals, it can be assumed that the driver board is working correctly.

5.2.2.16 Limit Switch Jumper Circuit

When operating the motor, a limit switch is depressed when the motor when the window becomes fully opened or fully closed. The microcontroller will read this voltage to control motor movement. Therefore, pull-down resistors are necessary to create a working switch circuit. The necessary resistors are included on the main board and connected to the switches through a jumper. The circuit is shown in **Figure 5.2.11**.

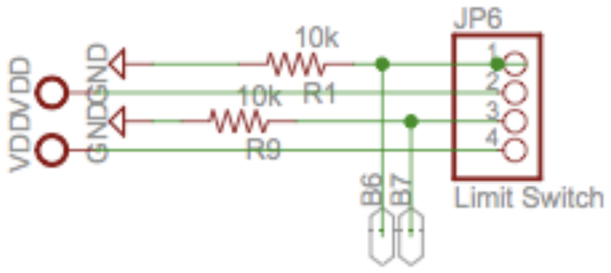


Figure 5.2.11. Limit switch jumper circuit.

No specific test will be needed for the limit switch circuit since the operation of the motor is being tested above in the motor block test. If the motor block is able to stop appropriately when it reaches the limits of its motion, it can be assumed that the limit switch circuit is working correctly.

5.3 Remote Control

5.3.1 Main Board

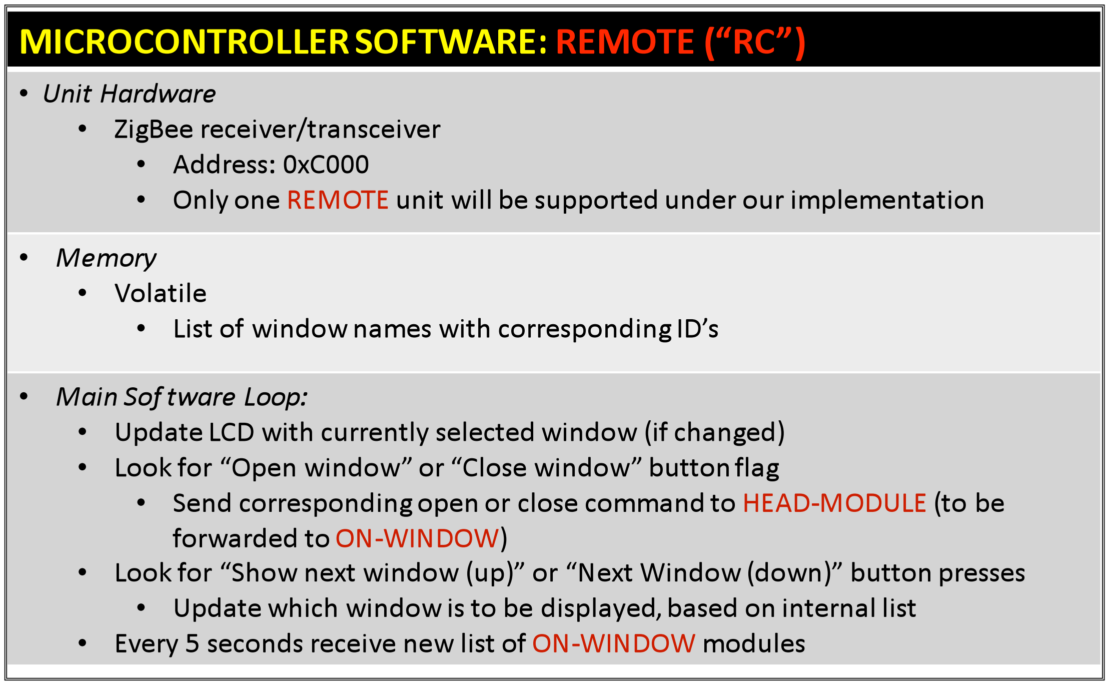
Identical to Section 5.1.3

5.3.1.1 Microcontroller

Identical to Section 5.1.3.1

5.3.1.2 Microcontroller Software

The design for the microcontroller software for the RCU is shown in **Figure 5.3.1**.



MICROCONTROLLER SOFTWARE: REMOTE (“RC”)

- *Unit Hardware*
    - ZigBee receiver/transceiver
        - Address: 0xC000
        - Only one REMOTE unit will be supported under our implementation
- *Memory*
    - Volatile
        - List of window names with corresponding ID’s
- *Main Software Loop:*
    - Update LCD with currently selected window (if changed)
    - Look for “Open window” or “Close window” button flag
        - Send corresponding open or close command to HEAD-MODULE (to be forwarded to ON-WINDOW)
    - Look for “Show next window (up)” or “Next Window (down)” button presses
        - Update which window is to be displayed, based on internal list
    - Every 5 seconds receive new list of ON-WINDOW modules

Figure 5.3.1. Microcontroller software design for RCU.

To test the microcontroller code, the code will be run many times with a variety of inputs and scenarios. All attempts will be made to create unusual situations for the program to handle. Program crashes will be monitored and corrected.

5.3.1.3 ZigBee Circuit

Identical to Section 5.1.3.3

5.3.1.4 ZigBee Wireless Interface

Identical to Section 5.1.3.4

5.3.1.5 Ceramic Resonator

Identical to Section 5.1.3.5

5.3.1.6 Programmer Circuit

Identical to Section 5.1.3.6

5.3.1.7 Serial EEPROM

Identical to Section 5.1.3.7

5.3.1.8 Serial EEPROM SPI Interface

Identical to Section 5.1.3.8

5.3.1.9 DC Input Circuit

Identical to Section 5.2.2.11

5.3.1.10 Button Card

Identical to Section 5.2.2.13

5.3.1.11 DC/DC Converter

Identical to Section 5.2.2.14

5.3.1.12 LCD Screen

The remote control can only address one OWU at a time. To display the currently addressed window to the user, an LCD screen must be included on the remote control. For this purpose will use the 2x16 character display offered by New Haven Displays. This display uses a one-directional SPI interface to display characters on the screen. **Figure 5.3.2** shows the schematic for the main board jumper that enables connection to the LCD screen. The LCD screen requires a 5V signal for power. This signal is obtained from the DC/DC converter in Section 5.4.1.11.

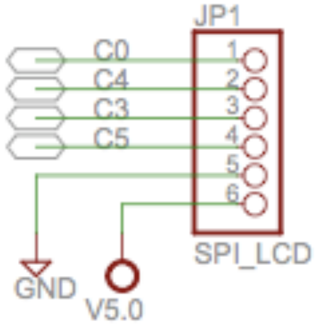


Figure 5.3.2. LCD screen jumper.

To test the LCD screen, a variety of characters will be sent to the screen through the SPI interface. If these characters can be displayed correctly, we will send commands such as clear the screen and advance one line. If these also work correctly, then the LCD has passed the test.

5.3.1.13 LCD Screen SPI Interface

The SPI screen communicates with the microcontroller through a one-way SPI protocol. This protocol operates in the same manner as the standard SPI protocol available in a variety of places. The difference is that for one-way communication the slave (the LCD screen) does not return a byte after the master sends one. Instead, the master sends bytes corresponding to the ASCII codes to display on the screen. The SPI interface will be tested with the LCD screen test listed above.

5.4 Preliminary Board Design

As a preliminary board design for our main project board, all the circuit schematics from the previous pages are reproduced here together in **Figure 5.4.1 a-c**.

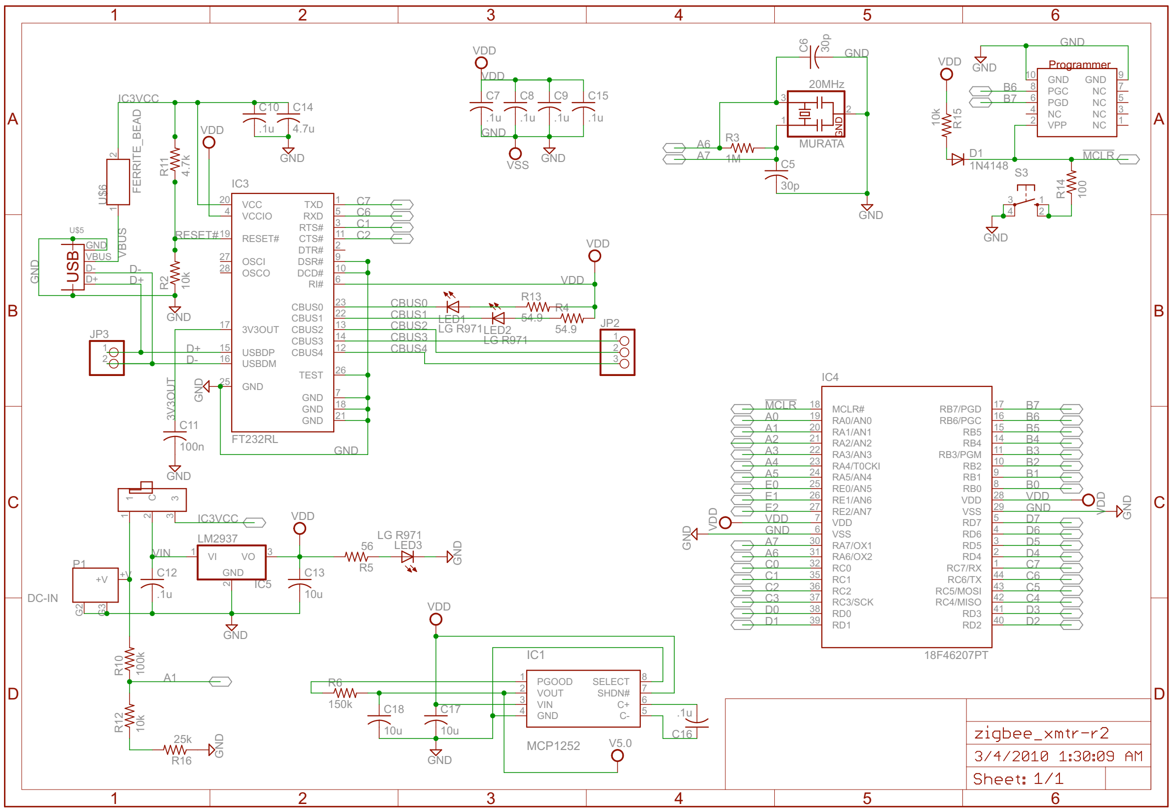
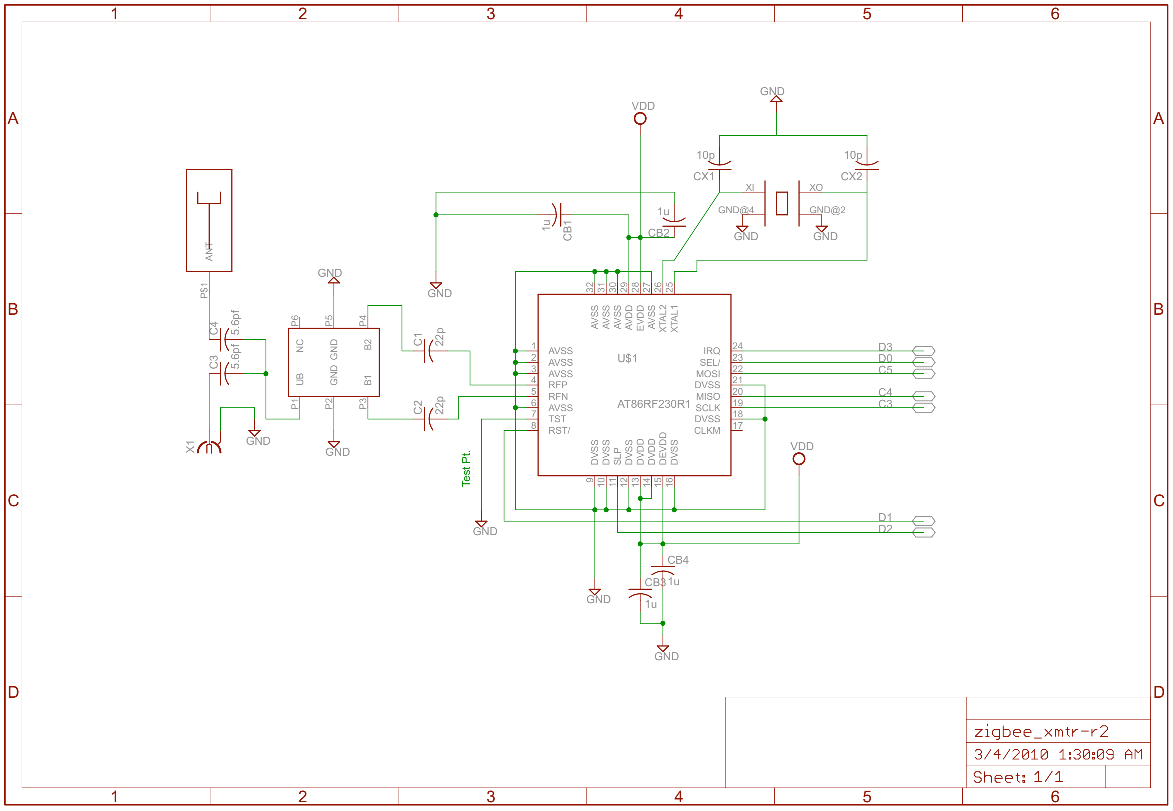
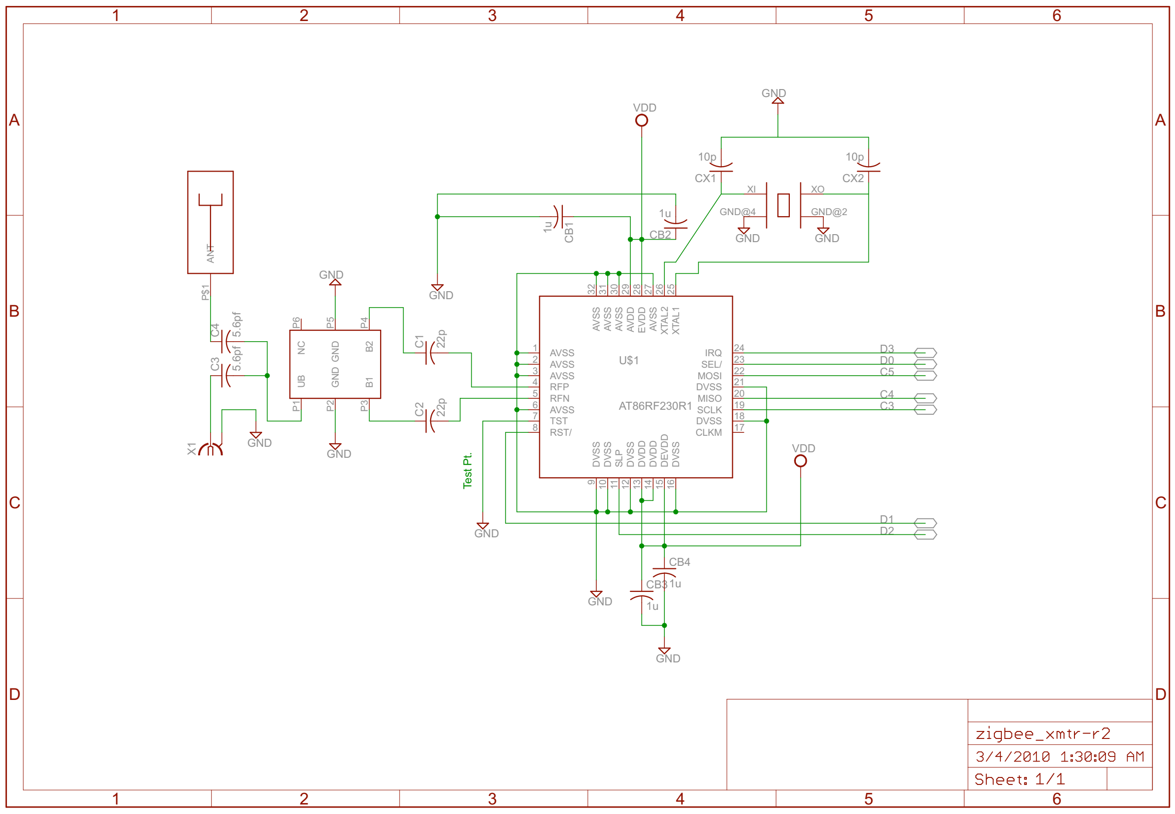


Figure 5.4.1.a. Page one of the preliminary board design.



3/4/2010 1:32:27 AM N:\Public\SmartWindows\Eagle\MainBoard\SmartWindowsMB\zigbee_xmtr-r2.sch (Sheet: 1/1)

Figure 5.4.1.b. Page two of the preliminary board design.



3/4/2010 1:32:27 AM N:\Public\SmartWindows\Eagle\MainBoard\SmartWindowsMB\zigbee_xmtr-r2.sch (Sheet: 1/1)

Figure 5.4.1.c. Page three of the preliminary board design.

6 Bill of Materials

Subsystem	**Part Description**	**Distributer**	**Part #**	**Quanti ty**	**Price/Unit**
Motor	H-bridge	Digikey	497-3624-1-ND	1	$5.55
Motor	DC Power Connector	Stock	Stock	1	Stock
Motor	Header Pins	Stock	Stock	4	Stock
Motor	1N4004 Diode	Stock	Stock	4	Stock
Motor	100 nF Capacitor	Digikey	PCC1853CT-ND	2	$0.26
Motor Block	Motor	Solarbotic	GM3	1	$7.95
Motor Block	Steal Rod	Lowes	Lowes	1	
Motor Block	Microswitch	Digikey	CKN9940-ND	2	$2.27
Motor Block	Levolor 23" x 42" White Mini Blind	Lowes	168349	1	$8.96
Motor Block	Rubber Spider	Jamecom	162000	1	$1.55
Motor Block	.125 ID Hub	Jamecom	162288	1	$1.49

Motor Block	.197 ID Hub	Jamecom	161998	1	$1.19
Motor Block	8 Battery Holder	Digikey	BH48AAW-ND	1	$1.91
Motor Block	24 pack Battery	Amazon	AA-NIMH-2600-24	1	$29.99
Motor Block	Battery Charger	Stock	Stock	1	Stock
Controller					
Controller	Push Button Switch	Digikey	450-1655-ND	6	$0.27
Controller	10K resistor	Stock	Stock	6	Stock
Voltage Divider					
Main	15K Resistor	Stock	Stock	1	Stock
Main	10K Resistor	Stock	Stock	1	Stock
Main	100K Resistor	Stock	Stock	1	Stock
Voltage Step Up					
Main	DC/DC Charge Pumper	Digikey	MCP1252-33X50I/MS-ND	1	$1.64
Main	10 uF Capacitor	Digikey	445-1372-1-ND	2	$1.43
Main	100 nF Capacitor	Digikey	PCC1853CT-ND	2	$0.26
Main	150K Resistor	Stock	Stock	1	Stock
MicroController					
Main	MicroController	Digikey	PIC18LF4620-I/PT-ND	1	$8.20
Programmer					
Main	10K resistor	Stock	Stock	1	Stock
Main	Push Button Switch	Digikey	450-1655-ND	1	$0.27
Main	100 Resistor	Stock	Stock	1	Stock
Main	1N4148	Digikey	1N4148WSFSCT-ND	1	$0.33
Main	100 Resistor	Stock	Stock	1	Stock
Voltage Regulator 3.3					
Main	DC Power Connector	Stock	Stock	1	Stock
Main	100 nF Capacitor	Digikey	PCC1853CT-ND	2	$0.26
Main	10 uF Capacitor	Digikey	445-1372-1-ND	2	$1.43
Main	3.3V Voltage Regulator	Digikey	LM2937ES-3.3-ND	1	$2.33
Main	Slide Switch	Digikey	EG1903-ND	1	$0.73
Main	56 resistor	Stock	Stock	1	Stock
Main	Red LED	Digikey	475-1410-1-ND	1	$0.11
SPI Screen					
Controller	New Haven Serial Display	New Haven	NHD-0216K3Z-NSW-BBW	1	$20.75

Transceiver					
Main	Antenna (LINX24ANT)		Need more information	1	
Main	Transceiver Card	Digikey	AT86RF230-ZU-ND	1	$3.70
Main	Female SMA Connect (bu-sma-v)		Need more information	1	
Main	BALLIN 0805		Need more information	1	
Main	ABRACKY		Need more information	1	
Main	1uF Capacitor	Digikey	490-1700-2-ND	4	$0.05
Main	10pF Capacitor	Digikey	490-1590-1-ND	2	$0.22
Main	22pF Capacitor	Digikey	490-1591-1-ND	2	$0.22
Main	5.6pF Capacitor	Digikey	478-1301-1-ND	2	$0.23
USB Input					
Main	USB - Serial UART	Digikey	768-1007-1-ND	1	$4.50
Main	Ferrite Bead		Need More Information	1	
Main	LED	Digikey	475-1410-1-ND	2	$11.00
Main	USB - Type B	Digikey	151-1121-ND	1	$0.93
Main	4.7k Resistor	Stock	Stock	1	Stock
Main	10k Resistor	Stock	Stock	1	Stock
Main	54.9 Resistor	Stock	Stock	2	Stock
Main	100nF Capacitor	Digikey	PCC1853CT-ND	1	$0.26
Main	0.1uF Capacitor	Digikey	490-1723-1-ND	1	$0.05
Main	4.7uF Capacitor	Digikey	490-3901-1-ND	1	$0.25
Parallel CAP					
Main	0.1uF Capacitor	Digikey	490-1723-1-ND	4	$0.05
JP6					
Main	54.9k Resistor	Stock	Stock	2	Stock
Real Time Clock					
Main	Crystal	Digikey	SER3205-ND	1	$0.32
Main	Real Time Clock	Digikey	DS1305EN/T&R-ND	1	$2.19
Main	10k Resistor	Stock	Stock	2	Stock
EEPROM					
Main	EEPROM	Digikey	25LC640A-I/ST-ND	1	$0.84
Light Sensor					
Main	10k Resistor	Stock	Stock	1	Stock
Main	Photodiode	Digikey	751-1055-1-ND	1	$1.44

7 Conclusions

This project brings to the forefront a series of design issues and serious engineering challenges. However, a solution exists to each of these various problems. In discovering and implementing these solutions, we will gain a great deal of technical knowledge as well as problem solving strategies.

In an effort to best minimize costs and maximize functionality, this design has been carefully considered and revised. A wide variety of sources and experts were consulted in constructing this design. To this end, several members of the Notre Dame engineering faculty, especially Professor Schafer, and the resources at Solar Shade deserve thanks for their valuable tools and guidance.

The Smart Windows system, as it is designed here, will be a widely applicable and adaptable system. However, it will also be targeted as a valuable marketing tool for the Solar Shades window treatment.

8 References

Quoted Resources:

[1] EIA online RECS 2005 Status Report. <http://www.eia.doe.gov/emeu/recs/contents.html>.
[2] DOE Energy Savers. <www.energysavers.gov/your_home/space_heating_cooling/index.cfm/mytopic=12353>.
[3] Mervosh, Sarah. “Burglars Target Off-Campus Housing.” *The Observer*. 23 Sept 2009. <http://media.www.ndsmcobserver.com/media/storage/paper660/news/2009/09/23/News/Burglars.Target.OffCampus.Housing-3780142.shtml>.

Relevant Device Spec Sheets (Attached to the electronic copy):
1N4004- Diode
1N4148- Diode
AT86RF230 – Zigbee transceiver chip
MURATA – Ceramic Resonator
MCP1252/3 - Charge Pumper
FT232RL - USB - Serial UART
25AA640A/25LC640A -EEPROM
L298 – H-bride Driver
TEMT6000X01 –Photodiode
LM2937-3.3 Voltage Regulator
CKN9940-ND –Microswitch
NHD-0216K3Z-NSW-BBW – Serial Display
PIC18F2525/2620/4525/4620 – Microcontroller
DS1305 –Serial Alarm Real Time Clock