A-

Senior Design Proposal:

Lights on, Lights off!
A wireless approach

EECS 129B
Professor Klefstad

Aakash Desai
ID:
Jeffrey Diomampo
ID:
Francisco Diaz
ID:

March 22nd, 2007

Abstract

Our group intends to create a cheap and easy-to-use home-based wireless lighting network system. This system will allow the user to control their home lighting through one easy to use wireless interface. The system will consist of a master remote/controller that is connected to a manual light switch and a slave switch which will control the transmitter.

Table of Contents

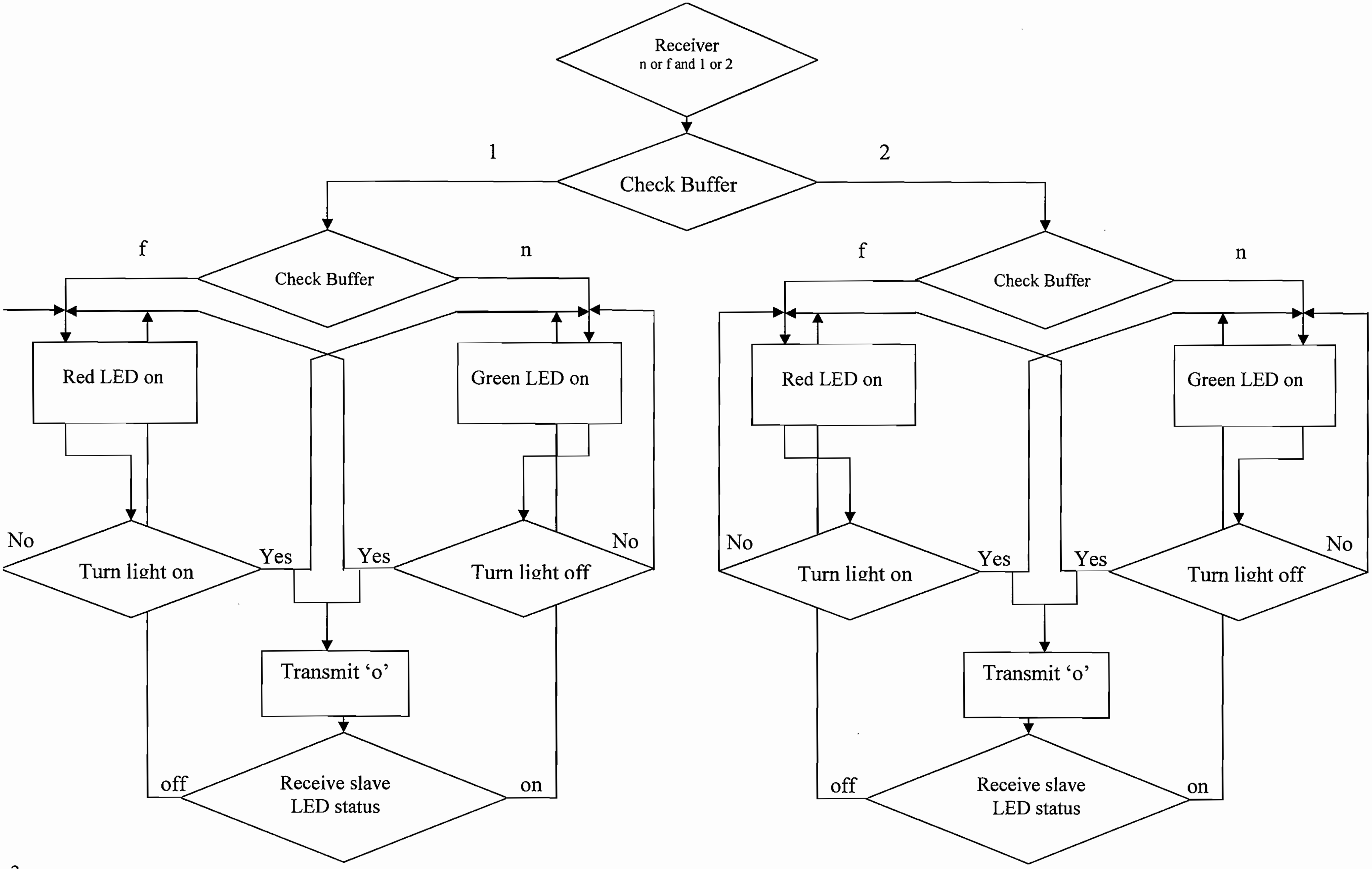
Project Description

Project Goals:

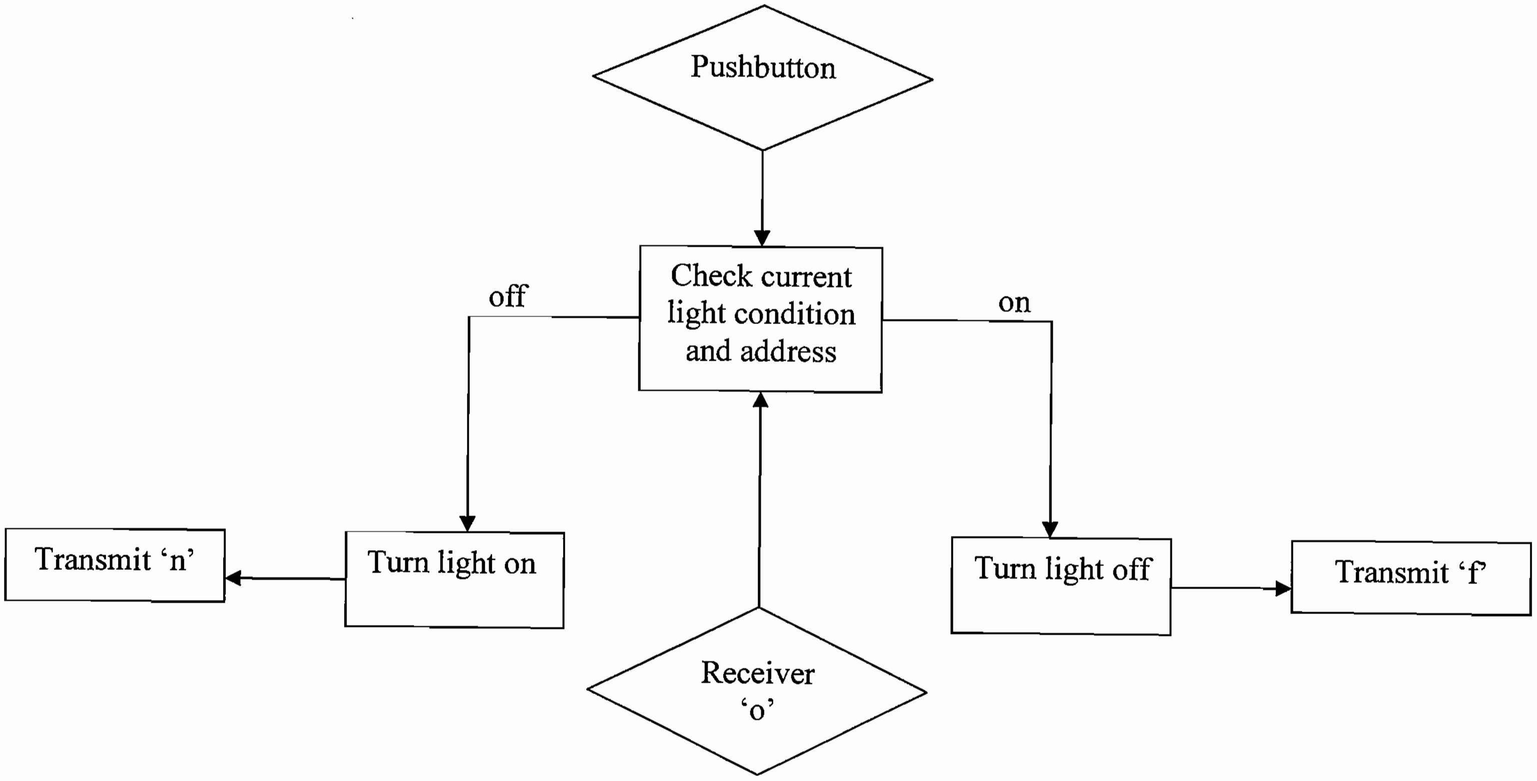
Light On, Lights Off is a simple wireless lighting solution. A Master Remote operates a series of Slave lights and displays status as well as the option to control and operate individual lights. Each individual component is based on an Arduino microcontroller that utilizes an XBee RF Wireless chip. This enables simple bi-directional communication between the Master and the Slave. Communication is the key to the entire network, it is through this communication that status on the remote is updated as well as command directives are carried out. Ultimately, the design of the circuits will support these goals and establish the wireless network.

Creating the Master and the Slave circuits proved to be an intricate process of design, labor and testing. Several obstacles were encountered along the way. More often than not, acute hardware issues such as programming the microcontroller or communication between the PC and the microcontroller itself became increasingly frustrating tasks. Additionally, mechanical and electrical issues arose in the actual testing phases of the design. Occasionally, it seemed that the entire design or code was scrapped and started anew. The project was troubled with burned out chips and several unresponsive microcontrollers. Not only did this delay progress, it also made the group question our design and implementation methods. The greatest hurdle seems to be in assessing the power output necessary to power the relay and in effect, control the lights. Notwithstanding these issues, the entire project represents significant experience in designing, understanding and troubleshooting electrical circuits.

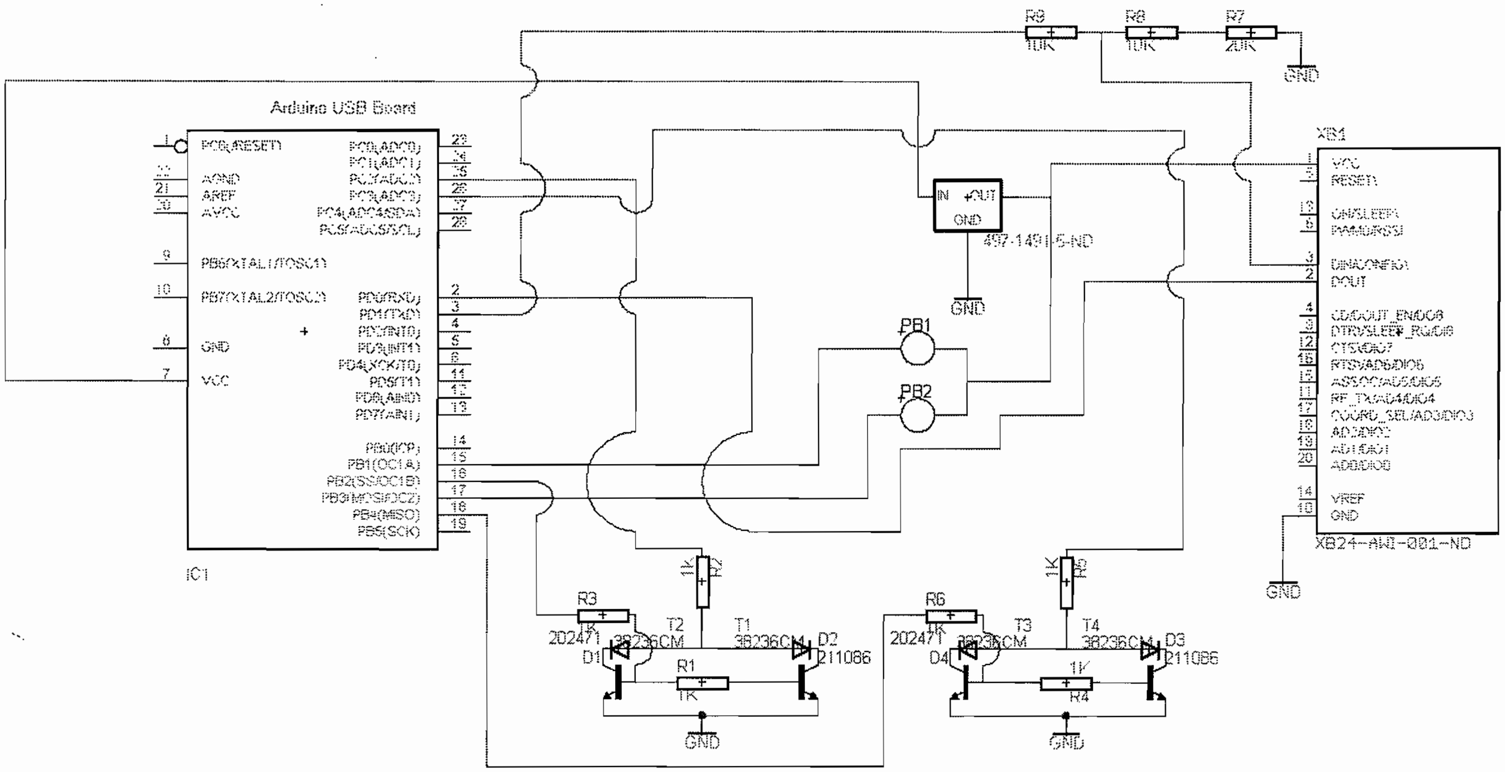
Master Switch Flow Chart



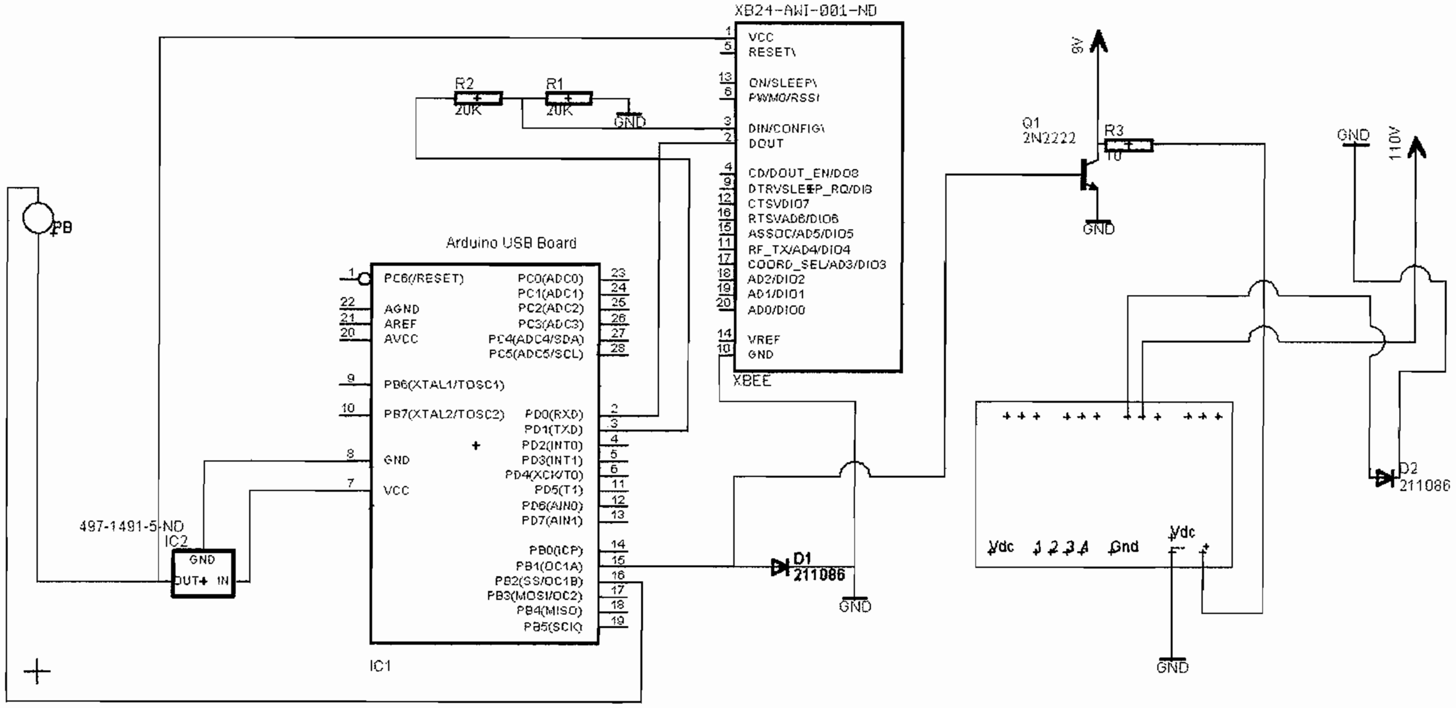
Slave Switch
Flow Chart



Project Schematic
for Master Remote/Controller



Project Schematic
for Slave/Switched Receivers



Software Description

The software for the Lights On, Lights Off Project facilitates wireless communication through status updates between the Master Remote and Slave Switches. The initialization of the circuits is done in the initial setup() of the code and the main work of the code is performed in the loop method.

Remote:

The Master Remote will set the PAN (Personal Area Network) ID as well as place the hardware into a broadcast mode for all the slaves to hear it. It sends the address of the slave it wishes to communicate with and the commands needed. In order to ensure there isn't any outside interference from other slaves, a block is placed in the code to stop the remote from receiving anything until the communication with the specific slave is finished.

Initial:

+ POWER on → setup() → Turn on status LED's → Send "s" → Wait 500ms

Transmit:

+ Pushbutton One pressed → switchOneAction()

+ Pushbutton Two pressed → switchTwoAction()

Receive:

+ Receive "1" or "2" → set controlPin to either ControlOnePin or ControlTwoPin → block communication from other slaves

+ Receive "n" or "f" → if msg not equal to readStatus() of LED's, then ledAction() → Open communication from other slaves

Methods:

setup(): Sets up xBee module with Broadcast Communication between it and slaves

- Set Baud Rate to 9600 → Send "+++" for Command Mode → Send corresponding commands needed ("ATID8000,MY8055,DH0,DLFFFF,CN")

ledAction(): Switches the controlPin of its statusLEDs depending on the previous state

- Read controlPin → if HIGH, write LOW; else write HIGH

switchOneAction(): Turn on/off controlOnePin and send on/off command to slave one

- Send "1" → Wait 250ms → Send "o" → ledAction() → Send readStatus() → Wait 500ms

switchTwoAction():Turn on/off switchTwoLED and send on/off command to slave one

- Send “2” → Wait 250ms → Send “o” → ledAction() → Send readStatus() → Wait 500ms

readStatus(): Returns the current status of the light; “f” is off and “n” is on

- Read lightPin → If HIGH, return “n”; else return “f”

Slave:

The slave sets up its PAN ID with a unary connection to the master remote and tells the remote when it has been plugged in or restarted by sending characters as status updates. From there, the hardware waits for an event to occur, the light is either manually or wirelessly switched, after the initial setup. Once there is an event, the light turns on or off and sends it’s address and new status to the remote as a report.

Initial:

+ POWER on → setup() → sendStatus()

Transmit:

+ PushButton pressed → lightOperation() → sendStatus()

Receive:

+ Receive “s” → sendStatus()

+ Receive appropriate slave address “1” or “2” → Allow for incoming messages → Receive “o” → Switch lightPin to turn on/off lightbulb → Wait 250ms → Receive “n” or “f” → Check if status message equals to readStatus() → Change lightPin if not

Methods:

lightOperation(): Switches the lightPin depending on its previous state

- Read lightPin → if HIGH, write LOW; else write HIGH

setup(): Sets up xBee module with Unary Communication between it and master remote

- Set Baud Rate to 9600 → Send “+++” for Command Mode → Send corresponding commands needed ("ATID8000,MY8008 or MY 8047,DH0,DL8055,CN")

readStatus(): Returns the current status of the light; “f” is off and “n” is on

- Read lightPin → If HIGH, return “n”; else return “f”

sendStatus() : Sends the address and status messages in a fixed time interval

- Send slave address “1” or “2” → Wait 250ms → readStatus() → Send status to Remote → Wait 500 ms

System Test Plan

System-Wide and Module Testing:

Remote:

1. Power the slave unit on.
2. Press the slave pushbutton once to turn on the light bulb.
3. Power the remote unit on.
4. Wait 3 seconds.
5. Check if the corresponding green status LED on the remote is on.
6. Press the remote pushbutton once to turn off the slave light bulb.
   a. It works properly if the light bulb turns off and the red status LED on the remote is on.
7. Power both units off.

Slave:

1. Power the slave unit on.
2. Press the slave pushbutton twice to check if the light bulb reacts.
3. Power the remote unit on.
4. Press the slave pushbutton once to check if the corresponding status LED's on the remote react (Green for on, Red for off).
5. Power the slave unit off and check if the corresponding status LED's on the remote react (Green for on, Red for off).

Parts/ Materials
Final Costs - Numbers

Slave Switch

Digikey
- (1) Xbee OEM RF Module (XB24-AWI-001-ND) - $23.20
- (1) 3.3V Voltage Regulator (497-1491-5-ND) - $0.77

Jameco
- (2) 3" x 5" Breadboard (194271) - $15.95
- (2) 10k *Resistors* (107676) - $0.02
- (1) 20k *Resistor* (107596) - $0.01

Sparkfun
- (1)Arduino USB Board (Arduino-USB) - $31.95

Master Remote/Controller

Jameco
- PC Board - $1.99
- (2) 10k *Resistors* (107676) - $0.02
- (1) 20k *Resistor* (107596)- $0.01
- (3) Green LED (211086) - $0.15
- (3) Red LED (202471) - $0.15
- (1) White LED (backlight) (320531)- $0.05

Digikey
- (1) Xbee OEM RF Module (XB24-AWI-001-ND) - $23.20
- (1) 3.3V Voltage Regulator (497-1491-5-ND) - $0.77

Sparkfun
- (1)Arduino USB Board (Arduino-USB)- $31.95
- (2) Push Buttons (315580CM) – $1.65

Misc.
- (1) 9V Battery - $3.95

General
- 60 ft. black, red and yellow wiring (36792, 36856, 36920) - $5.49

Total Kit Cost (2 Slaves, 1 Master) = 71.90 + 71.90 + 69.38 = $213.18

Summary

Lights On, Lights Off creates a network of circuits that operate through wireless communication. A microcontroller is the brain of the each circuit and controls the RF transmissions. In addition, a relay is implemented in order to properly configure and output the correct voltage and signal to the corresponding light switch. This enables the autonomous use of the consumer or the business applicant to manage the proper use of the entire network. Moreover, by creating a wireless circuit built into each independent circuit, this allows for easy implementation into existing power and electrical systems and backwards compatible with pre-wired electrical outlets.

The results of the experiment demonstrate suitable prototype of the entire project. Wireless communication between the master and the slaves occurs effectively and as expected. Power management, a big concern considering environmental and financial effects, is resolved using the relay as well as fewer resistors that initially anticipated. By allowing a small voltage to be outputted by the microcontroller, the entire circuit can be easily powered through a 9V batter or regulated through some higher voltage. Master and slave synchronization occurs through system software setup. This allows for both the master and the slave to return the proper status broadcasts as well as to determine operation through a series of protected broadcasts through the source code.

The biggest challenges in the undertaking of the project were hardware difficulties as well as understanding just how critical power management would be in the entire process. Most of the troubleshooting would come in the form of either triggering the relay using such a smaller voltage or actually controlling the 110V outlet power that was necessary to power and shut off the corresponding light. There also existed several instances when technical difficulties involving the microcontroller programming were encountered.

The problems stumbled upon through the development of the project were mainly technical and hardware related. Most were solved by simply isolating problems and trying to break down problems into smaller ones. However, many times the microcontroller would not respond to the commands of the programs and represented a burned-out chip. The only feasible solution was to replace the PC boards themselves. Also, there arose some trouble in actually triggering the relay to switch. However, this issue was resolved using circuit analysis of the relay.

Challenges

In completing this project, we ran into many problems and were unsuccessful in accomplishing all of our goals.

xBee RF Module:

When it came to setting up a network with 3 xBee RF Module, our setup would just not work. The xbee located on the master remote would not setup correctly to communicate with our two slave switches. We contacted the manufacturer and asked them on the properties of the xbee in addition to how many times it could be set up. We were told that there were no time constraints in how fast you could setup the xbee or in how many times you could do it. We were even given some code to use by the representative but to no avail. Therefore, we had to alter our setup properties when communicating with the slave switches and were able to overcome this problem.

Relay:

When it came to our relay, we had a difficult time in getting it to work with a 9V power supply. We asked our t.a., Mark Roland, for help, however, no positive outcome came about. Then suddenly the next time we had lab, our t.a. Mark was present to see that with no alterations of any kind, the relay began to function with the given power supply. However, a new problem arose in that the relay would not function by pulsing the inputs with high and low values. To resolve this problem, we would need to purchase another relay.

Arduino Boards:

In completing this project, we had the awful experience of burning out 2 of our boards. This was due to the fact that when we would upload code to our boards, we would leave the pushbutton connected to the 3.3V regulator which would create a direct connection with our board at the time we uploaded our code. Furthermore, to correct this problem, we bought the chips that we destroyed, same description and part number. However, once we placed them on the boards and were ready to upload again, no power was being read by the board. We contacted the Arduino seller and were told that those chips we purchased were not the correct ones even though they had the same part number and were told to ship back the complete units because no more arduino boards would be available for the month of February. In shipping the boards, they were lost of their way back to us.

Appendix

Timeline

Week 1

Design/Planning/Master Remote-Controller Hardware development

Week 2

Planning/Material Gathering for Slave Switch Hardware development

Week 3

MASTER Remote Prototype Circuit Building

Week 4

Slave Switch Prototype Circuit Building and Verification

and

Master Remote/Controller Verification

Week 5

Relay Switch Soldering and Construction

Week 6

Test Relay functionality and control systems

Week 7

Relay response to Arduino Small Signal Voltage

Week 8

Relay control of 110V power supply

And

Integration of Relay into MASTER and SLAVE operations

Week 9

Code Verification for Complete System Functionality

Week 10

Testing/Product Shell Creation

Project Task Assignments

Aakash Desai	Jeffrey Diomampo	Francisco Diaz
• Week 1 - Master Remote/Controller Design • Week 2 – Master Remote Coding • Week 3 – Slave Coding	• Week 1 — Master Remote/Controller Design 1 • Week 2 – Improved Hardware Schematics • Week 3 – XBee Wireless Communication Testing	• Week 1 – - Master Remote/Controller Design 1 • Week 2 – Hardware wiring • Week 3 – Model Testing of wiring techniques for accurate response of voltage/current leads from transmitter
• Week 4 – Arduino circuit MASTER Remote Code to Hardware Testing • Week 5 – Arduino circuit SLAVE Code to Hardware Testing • Week 6 – Isolated Arduino Wireless Communication Testing between MASTER and SLAVE	• Week 4 – Construction of 110V Relay switching • Week 5 – Testing of Relay switch • Week 6 – Determine the function of the control signal as well as application and implementation with the Arduino Small Signal Voltage.	• Week 4 – Arduino circuit construction • Week 5 – Validate hardware construction • Week 6 – Test communication between Master Remote/Controller and slave switch
• Week 7 – REMOTE Code fixed • Week 8 – SLAVE Code fixed • Week 9 – Final Debugs on Communications • Week 10 – Casing Construction	• Week 7 – Relay Debugging • Week 8 – Final debugs on MASTER • Week 9 – Final debugs on SLAVE • Week 10 – Casing construction	• Week 7 – Short Wave testing • Week 8 – Long wave testing • Week 9 – Final debugs on Transmitter • Week 10 – Casing construction