ECE 110 Honor Project Report

Chang He, Xiaowei Zhang, Yang Yuan

Computer controlled circuits for household electrical appliances

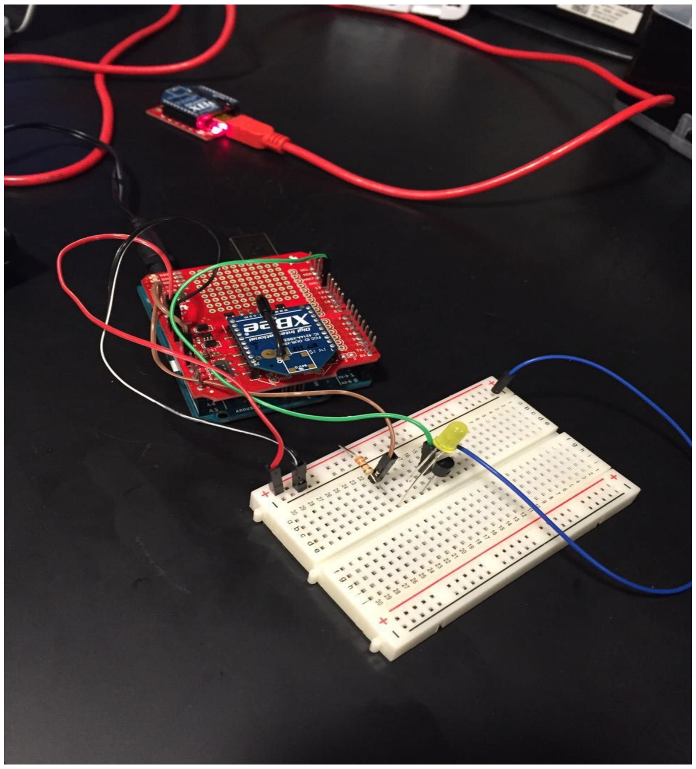
I.Introduction

The basic problem our project trying to solve is to figure out a convenient way to control the electric appliances remotely, which is part of home intelligentization. With the help of our project, when far away from home, people won't worry about the power wasting caused by their carelessness to turn off the electric appliances. They could solve the problem just by turning on a computer and click 'turn off 'on the interface. Another goal we want to achieve is to observe the power consumption of the electric appliances at any time we want.

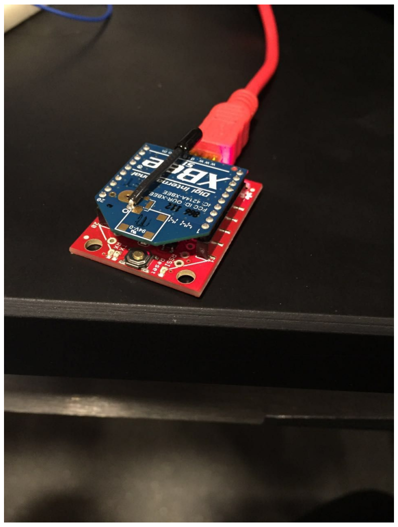
Basically we worked to make our project looks like a black box, and it could be plugged in any socket we want to control. We used Xbees to complete the remote information transmission which is pretty convenient and applicable.The most important part in our project is the code, basically we used Java to make everything happen.

II.Design

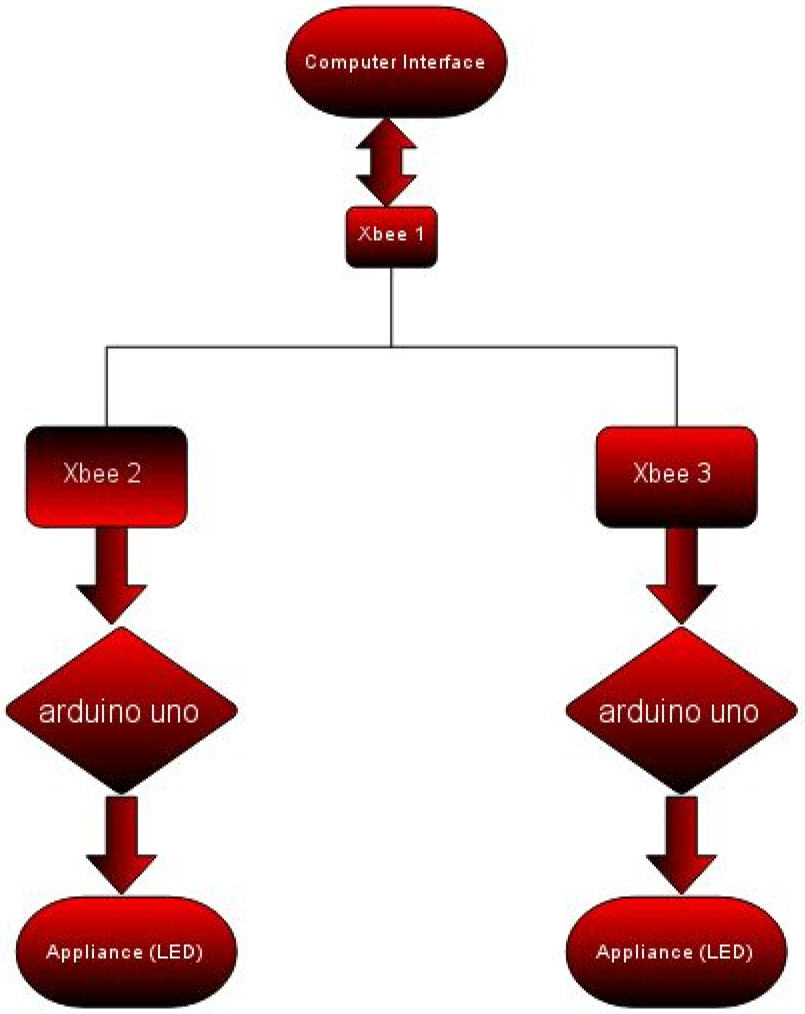
A. Written Descriptions of Blocks



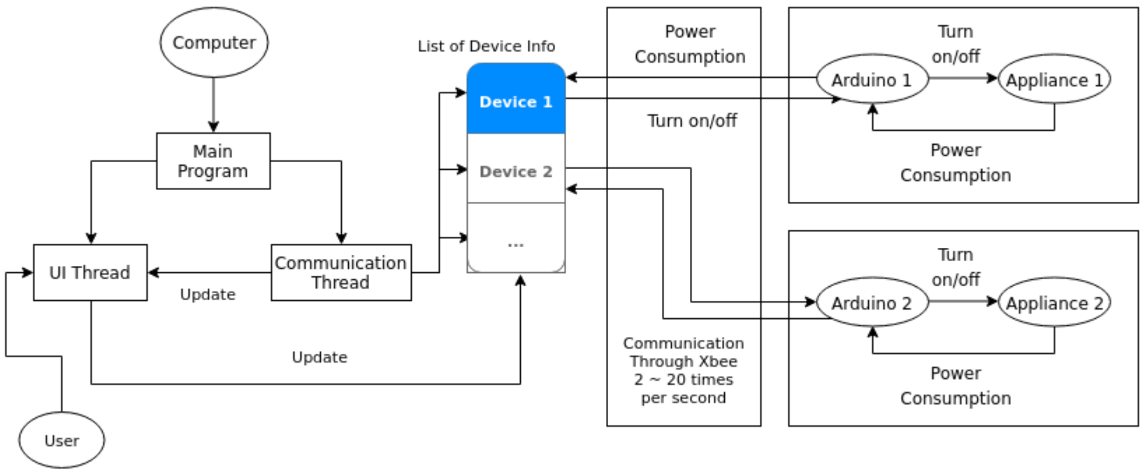
In this lab, we applied 2 xbees and 1 arduino Uno to complete our task. (ideally shown 3 xbees and 2 arduino Uno below). Computer interface is the GUI we coded for users to wirelessly control the appliances. Xbee 1 is the command transmitter as well as receiver who receives the signal from the computer through a USB port and then send that signal to Xbee 2 and 3. Xbee 2 and 3 works primarily as a receiver who receive the signal from Xbee 1 and pass that signal to arduino Uno through Xbee shields (not included in the block diagrams). Arduino Unos decode the signal and turn on/off the appliances depending on the code.



B. Block Diagrams



D. (If Applicable) Flow chart of software



E.(If Applicable) Code Used (Java 8, Rxtx Library 2.1.7, Digi Xbee Java Library, Xbee Arduino Library): See Appendix A

III.Result

In our demo, the LED is the only electric appliance. The basic present result is that we could control the LED to be on or off just by clicking ‘ on’ or’ off’ on a computer far away and the power consumption could be displayed on the screen correctly. The result is obvious just about the LED’s on and off and the display of the power consumption so we could clearly see that the basic code and the circuit we applied was correct. There is actually no quantitative analysis we could get from the result because the result of LED’s on and off was not about any quantity. For the power consumption, the result is the same as what we measured so we could say everything is on the right track.

IV. Future Work

Currently there is limitation for the distance between the controlling computer and the appliances because of the Xbee we chose. Later to make things like controlling the home electric appliances in the office, we need to replace current Xbee with a higher quality one to make sure the transmission over miles' happen. We also need to minimize current circuit to make it being small enough to be plug into socket. And we need to reconsider the situation when there are more than one electric appliance because the circuit and the code will also change a lot in the new situations.

V. Conclusion

Obviously we've achieved our basic goal to control an electric appliance remotely and observe the power consumption of it. However we haven't make that happen when there are lots of electric appliances and we haven't make our project to really work by be plugged into a socket. During this interesting process of working on the project, we learned the importance of computer science because the code is actually the core of this whole project. We also learned a lot of knowledge about things like IRs,Xbees, and relays during the process of reading and explorations. Working on this project was generally a great process for all of us.

VI. Appendix

A. code used:

Main.java:

```
package sample;

import gnu.io.CommPortIdentifier;
import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.beans.binding.Bindings;
import javafx.beans.property.LongProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.util.Enumeration;

import static sample.Controller.*;

public class Main extends Application {
    final TableView<Device> tableView = new TableView<>();
    final TextField addDeviceMAC = new TextField();
    final TextField addDeviceRemark = new TextField();
    final Button addButton = new Button("add");
    final Button removeButton = new Button("remove");
    final Button toggleButton = new Button("turn on/off");
    final Button findPortButton = new Button("Refresh Ports");
    final ChoiceBox<String> choiceBox = new ChoiceBox<>();
    final Button setPortButton = new Button("Set Serial Port");
    final Label portLabel = new Label();


    {
        addDeviceMAC.setPromptText("Enter the MAC address of your device");
        addDeviceRemark.setPromptText("Enter the name of your device");
    }

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Power Control");

        GridPane pane = new GridPane();
        ColumnConstraints c1 = new ColumnConstraints(), c2 = new ColumnConstraints();
        c1.setPercentWidth(50);
        c2.setPercentWidth(50);
        pane.getColumnConstraints().addAll(c1, c2);

        Scene scene = new Scene(pane);
        pane.setPrefSize(800, 600);

        pane.add(tableView, 0, 0, 1, 3);
        pane.setHgap(10);
        pane.setVgap(10);

        tableView.setEditable(false);
        tableView.setItems(devices);

        TableColumn<Device, String> macColumn = new TableColumn<>();
        macColumn.setText("MAC Address");
        macColumn.setCellValueFactory(param ->
                Bindings.createStringBinding(() ->
Long.toUnsignedString(param.getValue().getMac(), 16).toUpperCase(),
                        param.getValue().macProperty()));

        TableColumn<Device, String> remarkColumn = new TableColumn<>();
```

```java
        remarkColumn.setText("Remark");
        remarkColumn.setCellValueFactory(new PropertyValueFactory<>("remark"));

        TableColumn<Device, String> statusColumn = new TableColumn<>();
        statusColumn.setText("Status");
        statusColumn.setCellValueFactory(param ->
                Bindings.createStringBinding(() ->
                        param.getValue().getIsOn() ? "on" : "off",
param.getValue().isOnProperty()));

        TableColumn<Device, String> powerColumn = new TableColumn<>();
        powerColumn.setText("Power");
        powerColumn.setCellValueFactory(param -> Bindings.createStringBinding(() ->
String.format("%.4f", param.getValue().getPower()), param.getValue().powerProperty()));

        tableView.getColumns().addAll(macColumn, remarkColumn, statusColumn, powerColumn);


        addButton.setOnAction(this::handleAddButton);
        removeButton.setOnAction(this::handleRemoveButton);
        toggleButton.setOnAction(this::handleToggleButton);
        findPortButton.setOnAction(this::handleFindPort);

        VBox vBox = new VBox(10);
        HBox hBox = new HBox(20);
        hBox.getChildren().addAll(addButton, removeButton);
        vBox.getChildren().addAll(addDeviceMAC, addDeviceRemark, hBox);
        pane.add(vBox, 1, 0);
        pane.add(toggleButton, 1, 1);

        HBox hBox2 = new HBox(20);
        VBox vBox2 = new VBox(10);
        hBox2.getChildren().addAll(choiceBox, findPortButton);
        vBox2.getChildren().addAll(hBox2, setPortButton);

        VBox vBox3 = new VBox(10);
        vBox3.getChildren().addAll(new Label("Current Port: "), portLabel);

        portLabel.textProperty().bind(port);

        setPortButton.setOnAction(event -> {
            String p = choiceBox.getValue();
            port.setValue(p);
        });

        pane.add(vBox2, 1, 2);
        pane.add(vBox3, 0, 3);

        findPortButton.fire();
        Task<Integer> task = new Task<Integer>() {
            @Override
            protected Integer call() throws Exception {
                int iterations;
                for (iterations = 0; true; iterations++) {
                    try {
                        Thread.sleep(delay);
                        update();
                    } catch (InterruptedException e) {
                        return iterations;
                    }
                }
            }
        };

        new Thread(task).start();

        final LongProperty lastUpdate = new SimpleLongProperty(0);
        AnimationTimer timer = new AnimationTimer() {
            @Override
            public void handle(long now) {
                if (now - lastUpdate.get() > 1000) {
                    try {
                        if (updates.isEmpty()) return;
                        Controller.Update update = updates.take();
                        update.oldValue.setPower(update.newValue.getPower());
                    } catch (InterruptedException e) {
                        System.err.println("interrupted");
                        return;
                    }
                }
                lastUpdate.set(now);
            }
        };

        timer.start();

        primaryStage.setOnCloseRequest(event -> task.cancel());

        primaryStage.setScene(scene);
        primaryStage.show();
    }

    void handleAddButton(ActionEvent event) {
        long macAddr = 0;
        String addr = addDeviceMAC.getText();
        try {
            if (addr.length() > 16) throw new NumberFormatException();
            macAddr = Long.parseUnsignedLong(addr, 16);
        } catch (NumberFormatException e) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("java.lang.NumberFormatException");
            alert.setHeaderText(addr + " is not a legal hexadecimal string of 64 bits!");
            alert.show();
            return;
        }
        String remark = addDeviceRemark.getText();
        devices.add(new Device(macAddr, remark, false, 0.0f));
        addDeviceMAC.clear();
        addDeviceRemark.clear();
    }

    void handleRemoveButton(ActionEvent event) {
        Device selected = tableView.getSelectionModel().getSelectedItem();
        if (selected == null) return;
        if (selected.getIsOn()) {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Error");
            alert.setHeaderText("You must turn your device off before removing it!");
            alert.show();
            return;
        }

        devices.remove(selected);
        tableView.getSelectionModel().clearSelection();
    }

    void handleToggleButton(ActionEvent event) {
        Device selected = tableView.getSelectionModel().getSelectedItem();
        if (selected == null) return;
        selected.setIsOn(!selected.getIsOn());
        //update(selected);
    }

    void handleFindPort(ActionEvent event) {
        choiceBox.getItems().clear();
        Enumeration<CommPortIdentifier> ids = CommPortIdentifier.getPortIdentifiers();
        while (ids.hasMoreElements()) {
            CommPortIdentifier id = ids.nextElement();
            if (id.getPortType() == CommPortIdentifier.PORT_SERIAL)
choiceBox.getItems().add(id.getName());
        }
    }
}
```

Controller.java:

```
package sample;

import com.digi.xbee.api.RemoteXBeeDevice;
import com.digi.xbee.api.XBeeDevice;
```

```java
import com.digi.xbee.api.exceptions.XBeeException;
import com.digi.xbee.api.models.XBee64BitAddress;
import com.digi.xbee.api.models.XBeeMessage;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

public class Controller {

    static long delay = 50;
    static ObservableList<Device> devices =
FXCollections.synchronizedObservableList(FXCollections.observableArrayList());
    static BlockingQueue<Update> updates = new ArrayBlockingQueue<Update>(10000);
    static ObservableList<String> ports = FXCollections.observableArrayList();
    static SimpleStringProperty port = new SimpleStringProperty("");

    static void update() {
        if (port.get().isEmpty()) return;
        XBeeDevice xbee = new XBeeDevice(port.get(), 9600);
        for (Device device : devices) {
            RemoteXBeeDevice dest = new RemoteXBeeDevice(xbee, new
XBee64BitAddress(Long.toUnsignedString(device.getMac(), 16)));
            try {
                xbee.open();
                xbee.sendData(dest, device.getIsOn() ? new byte[]{1} : new byte[]{0});
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // nothing
                }

                XBeeMessage response = xbee.readDataFrom(dest, 5000);
                Float power =
ByteBuffer.wrap(response.getData()).order(ByteOrder.LITTLE_ENDIAN).getFloat();
                if (power == device.getPower()) return;
                Device copy = new Device(device);
                copy.setPower(power);
                updates.offer(new Update(device, copy));
            } catch (XBeeException e) {
                e.printStackTrace();
            } finally {
                xbee.close();
            }
        }
    }
    @Deprecated
    static void update(Device device) {
        XBeeDevice xbee = new XBeeDevice(port.get(), 9600);
        RemoteXBeeDevice dest = new RemoteXBeeDevice(xbee, new
XBee64BitAddress(Long.toUnsignedString(device.getMac(), 16)));
        try {
            xbee.open();
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
            }
            xbee.sendData(dest, device.getIsOn() ? new byte[]{1} : new byte[]{0});
        } catch (XBeeException e) {
            e.printStackTrace();
        } finally {
            xbee.close();
        }
    }

    static class Update {
        Device oldValue;
        Device newValue;

        Update(Device oldValue, Device newValue) {
            this.oldValue = oldValue;
            this.newValue = newValue;
        }
    }
}
```

Device.java:

```
package sample;

import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.property.SimpleFloatProperty;
import javafx.beans.property.SimpleLongProperty;
import javafx.beans.property.SimpleStringProperty;

/**
 * Created by apple on 5/1/16.
 */
public class Device implements Cloneable {

    private final SimpleLongProperty mac;
    private final SimpleStringProperty remark;
    private final SimpleBooleanProperty isOn;
    private final SimpleFloatProperty power;

    Device() {
        mac = new SimpleLongProperty(0);
        remark = new SimpleStringProperty(null);
        isOn = new SimpleBooleanProperty(false);
        power = new SimpleFloatProperty(0.0f);
    }

    Device(long mac, String remark, boolean isOn, float power) {
        this.mac = new SimpleLongProperty(mac);
        this.remark = new SimpleStringProperty(remark);
        this.isOn = new SimpleBooleanProperty(isOn);
        this.power = new SimpleFloatProperty(power);
    }

    Device(Device device) {
        this(device.getMac(), device.getRemark(), device.getIsOn(), device.getPower());
    }

    public long getMac() {
        return mac.get();
    }

    public void setMac(long mac) {
        this.mac.set(mac);
    }

    public SimpleLongProperty macProperty() {
        return mac;
    }

    public String getRemark() {
        return remark.get();
    }

    public void setRemark(String remark) {
        this.remark.set(remark);
    }

    public SimpleStringProperty remarkProperty() {
        return remark;
    }

    public boolean getIsOn() {
        return isOn.get();
    }

    public void setIsOn(boolean isOn) {
        this.isOn.set(isOn);
    }

    public SimpleBooleanProperty isOnProperty() {
        return isOn;
    }

    public float getPower() {
        return power.get();
    }

    public void setPower(float power) {
        this.power.set(power);
    }

    public SimpleFloatProperty powerProperty() {
        return power;
    }
}
```

sample.fxml:

```
<?import javafx.scene.layout.GridPane?>
<GridPane xmlns:fx="http://javafx.com/fxml"
          fx:controller="sample.Controller" alignment="center" hgap="10" vgap="10">
</GridPane>
```

Arduino Code:

```
#include <Printers.h>
#include <XBee.h>

XBee xbee = XBee();
Rx64Response rx64 = Rx64Response();
bool isOn = false;
double power = 0;

void setup() {
  // put your setup code here, to run once:
  delay(1000);
  Serial.begin(9600);
  xbee.setSerial(Serial);
  pinMode(13, OUTPUT);
  pinMode(A1, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  int in = analogRead(A1);
  power = (5 - 5 * ((double) in / 1023.0))/330 * 5;

  union {double in; unsigned char out[4]; } convert;
  convert.in = power;

  xbee.readPacket();
```

```
    if (xbee.getResponse().isAvailable()) {
      //Serial.print("getResponse!\n");
      if (xbee.getResponse().getApiId() == RX_64_RESPONSE) {
        xbee.getResponse().getRx64Response(rx64);
        isOn = rx64.getData(0);
        digitalWrite(13, isOn ? HIGH : LOW);

        //delay(1000);
        Tx64Request tx64(rx64.getRemoteAddress64(), &convert.out[0], 4);
        xbee.send(tx64);
      }
    }else if (xbee.getResponse().isError()) {
        Serial.print("Error!\n");
    }
  }
```