California Polytechnic State University,
San Luis Obispo

Senior Project

Bluemote!

by
Philip Kaye

Winter Quarter - March 2012

Senior Project Advisor: Dr. John Seng

Table of Contents

1.0 Executive Summary……3
2.0 Design & Implementation……..5
3.0 Hardware & Software Configuration……...9
3.1 Base Station……..10
3.2 Android Application……12
4.0 Conclusion……..16
5.0 Future Expansion……..16

Appendix A – Arduino Sketch Code……18
Appendix B – BluetoothChat Class……...47
Appendix C – BluetoothChatService Class……...58
Appendix D – DeviceListActivity Class……68

List of Figures and Tables

Figure 1: Block Diagram of my Senior Project……..7
Figure 2: Photo of Base Station Assembled……...…….8
Figure 3: Modulated Pin Layout for the Base Station…….11
Figure 4: Pulse Width Modulation of the Sony SIRC Protocol……..12
Figure 5: Pin Schematic of Base Station……...12
Figure 6: Flowchart Diagram of Bluemote Android Application……...15

Table 1: Hardware Specifications for Arduino Uno……10

Executive Summary

1.0 Executive Summary

In today's society, our mobile phones are an extremely useful device when we are on the go and away from our homes; however, when we return home, our smart phone becomes just a communication tool again. Home automation is inevitable, and it will find its way into every house. The first person to commercialize it to the point where it is as easy to set-up and as affordable as a home router will end up breaking this slow trend. The key to this success is the smart phone; this should end up being the most expensive piece of hardware for this system and with the everyone owning a smart phone these days, that future is in sight.

The objective of this senior project is to demonstrate the capabilities of your mobile phone in your home. The full design of this is idea involves one home base station that the user connects to via Bluetooth, in which that base station is connected to any number of peripherals via the Zigbee protocol, allowing the user to manipulate a wide range of tasks from the convenience of their phone. For example; dimming the lights, closing the garage door, or controlling ones entertainment system.

The current status of this project is a scaled down version of the total idea. At this stage, the user is able to connect to a base station via an Android app (Bluemote!) that I developed. The base station is an Arduino Uno Development Board fitted with a Bluetooth module and an IR LED, which is used for Pulse Width Modulation (PWM) to control a Sony TV via the SIRC Protocol.

Design & Implementation

2.0 Design & Implementation

The design of the current system is a scaled down version of the larger idea. The larger idea consists of a single base station controller that is connectable via the Bluetooth on a mobile phone, in which the user will connect and be able to control a wide range of peripherals around their home. The base station is the only thing the user has to deal with. The base manages the peripherals via the Zigbee protocol, which is a low power, low cost, alternative to Wifi or Bluetooth. Using the Zigbee protocol allows the peripherals to talk in a mesh network, allowing applications and sensors to communicate with each other. For example; if you use your phone to turn the A/C on, the base station can check to see if any windows are open, and communicate to the user to close it before turning on the air conditioner.

For the scaled down, proof-of-concept version of this idea, I decided to implement the base station and one single peripheral node, an IR remote to control your home entertainment system. One design challenge I encountered was my initial intentions to use the controller from a third-party universal remote and be able to simulate button presses in order to use all the preconfigured pulse width modulation (PWM) codes for all the current possible TVs, set-top boxes, etc; however, I was unsuccessful in hacking the controller because they make them tamper proof. I surrendered the ability to make it a universal remote, so as it currently stands, it can only control a Sony TV that uses the 12 bit SIRC Protocol.

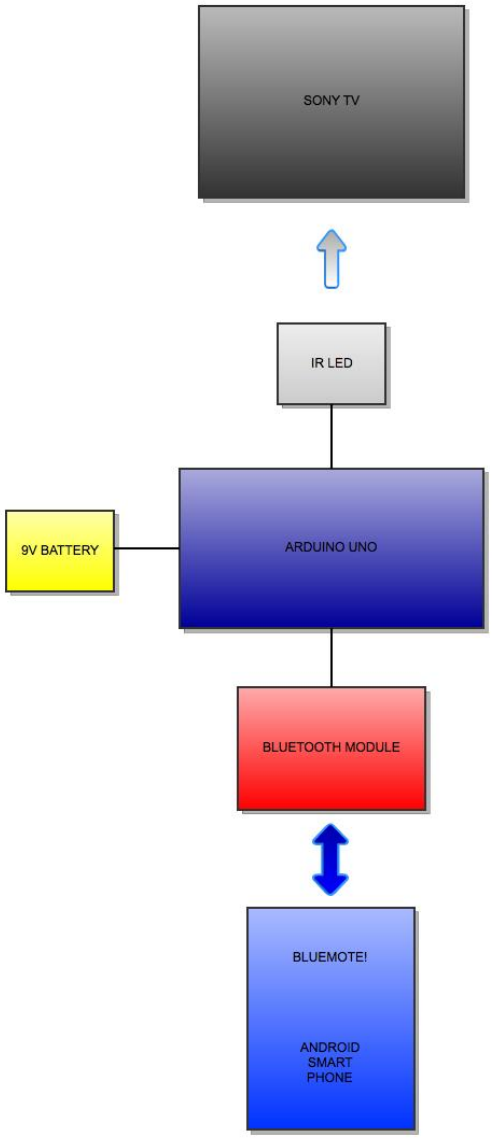

Figure 1: Block Diagram of my Senior Project

I used an Arduino Uno Development Board as the base station because it's very simple to use and is great for rapid prototyping. A single 9 V battery is used to power the board and make it portable in order to allow the user to set it up to face their equipment, never having to worry about pointing their remote at their equipment again. I connected a Bluetooth module to its serial port and an IR LED to an output pin. I configured the Arduino Uno to output to the IR LED a pulse width modulation (PWM) at 38kHz for the

SIRC protocol. Its input comes from the serial port (pin 0), where a switch statement reads the bytes in and determines which signal to output to the TV.

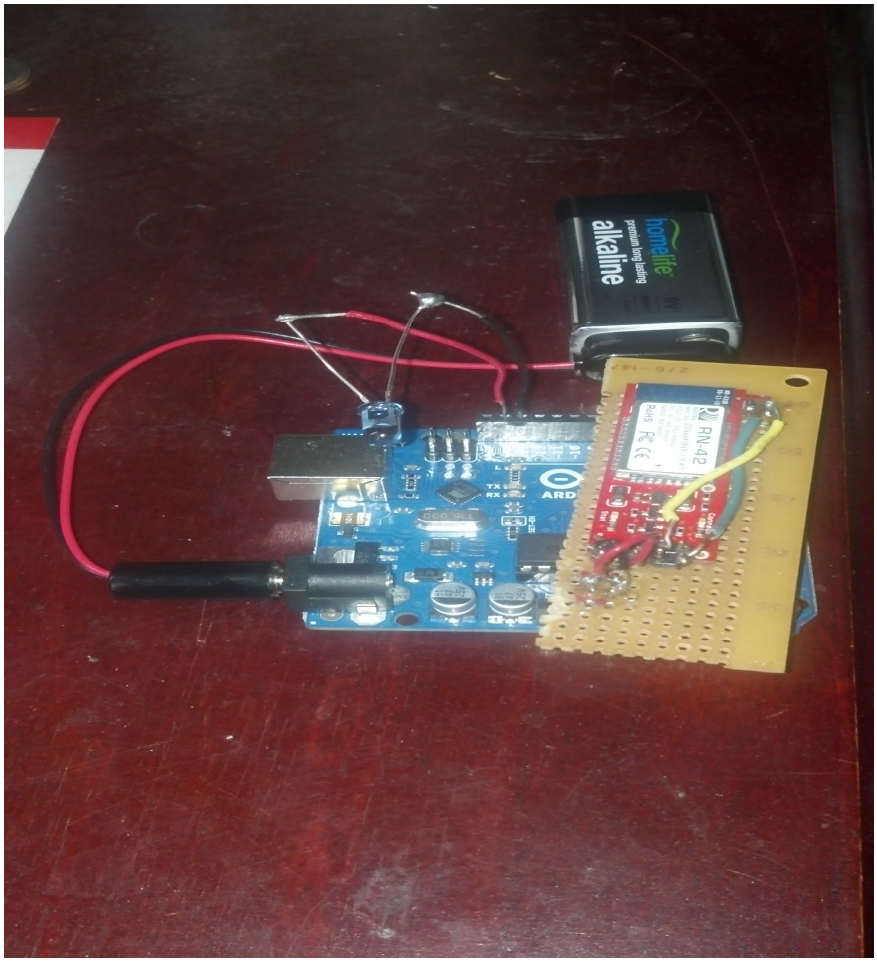

Figure 2: Photo of Base Station Assembled

I decided to write an Android app to connect to the Arduino Uno because I wanted to teach myself Android OS and because I just recently purchased a Droid Razr and that would be the phone I would use to demo.

When a user clicks on the app the first thing it does is check to see if Bluetooth is enabled on the phone and if not, it requests permission to turn it on. Once on the main screen, the user clicks the menu button, which brings up a small screen to connect to the Arduino Uno. When the app has finished connecting, all the buttons are activated and the user is free to control their Sony TV as they please.

Hardware & Software Configuration

3.0 Hardware & Software Configuration

3.1 Base Station

The base station is built around the Arduino Uno. The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Table 1: Hardware Specifications for Arduino Uno

Software is uploaded via a USB port. Uploading software to the board is referred to as "burning sketches." The sketches include two functions: a setup function, which runs only once at startup, and a loop function that repeats continuously during runtime. My setup function, located in Appendix A, sets up pin 13 as an output because this is where the IR LED is placed. Also, I had to set the serial baud rate to the same baud rate of my Bluetooth module because if it wasn't I wouldn't receive the data properly. My Sparkfun Electronics Bluetooth Mate Silvers' default baud rate is 115200. The loop function waits for data to be received from the serial port and then uses a switch statement to determine which button press (pulse width modulation) to output.

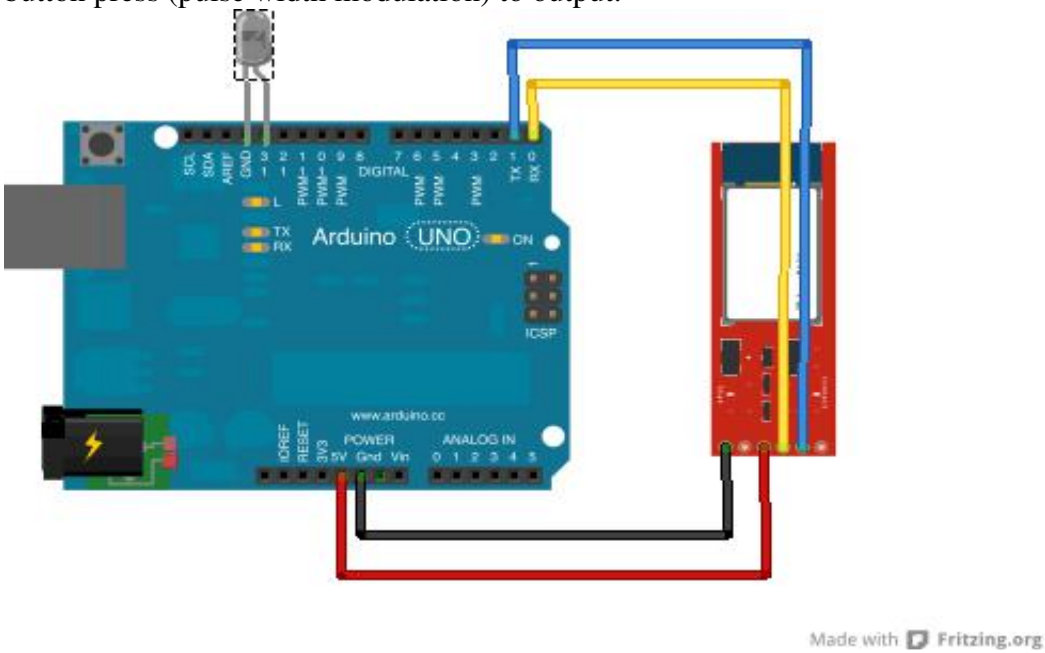

Figure 3: Modulated Pin Layout for the Base Station

As stated before, the IR LED's cathode goes into pin 13 and its anode into ground. The protocol used for the Sony TV is SIRC 12-bit protocol. The pulse width modulation oscillates at 38kHz. The 12-bit protocol is broken down into 3 sections. A 2.4ms start command, followed by a 7 bit command, and a 5 bit address. The command tells the electronic which button is pressed and the address field determines which type of electronic should receive the command; TV, VCR, etc.

QuickTime™ and a
decompressor
are needed to see this picture.

Figure 4: Pulse Width Modulation of the Sony SIRC Protocol

The Bluetooth Mates' Vcc pin goes to the 5 V power pin on the Arduino, while ground goes to ground. The Serial port (Tx, Rx) on the Bluetooth Mate have to go on their opposite pins on the Arduino Uno. The Tx pin goes to the Rx pin and vise versa. To power the base station wirelessly I used a 9 V battery.

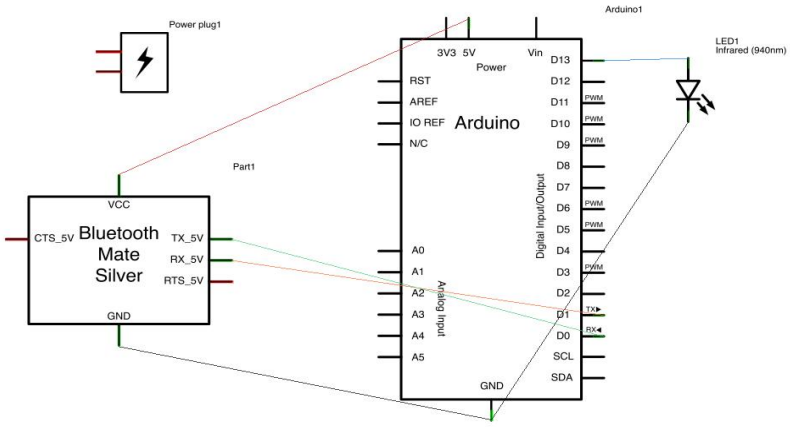

Figure 5: Pin Schematic of Base Station

3.2 Android Application

The Android Application is comprised of three classes: BluetoothChat (Appendix B), BluetoothChatService (Appendix C), and DeviceListActivity (Appendix D).

The DeviceListActivity class is an activity in charge of connecting to a Bluetooth device. The class is used to scan for new devices or connect

to a previously paired device. The user accesses this sub menu by clicking the menu button on the phone. It holds an array of paired devices and newly scanned devices. At the current stage of the app, you need to already be paired with the home base station in order to connect.

The BluetoothChatService class handles the communication between the two devices. It contains two sub classes connectThread and connectedThread. The connectThread is used while attempting to make an outgoing connection with a device and the connectedThread is used while connected to the device. Both are threads because each contain a call that blocks, so a thread is used so the application isn't blocked as well. The BluetoothChatService controls both of these threads, while updating the current status of the connection. The one key note about this class is it manages the handshake between the two devices. In order to connect to the Bluetooth Mate module, I needed to pick a serial port Bluetooth profile. This is achieved via the UUID that my application uses to connect to the base station.

```
private static final UUID MY_UUID =
                    UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
```

There are other profiles available, but this is the one needed to connect to the Bluetooth Mate module.

The BluetoothChat class is the main Activity of the application. It manages the main.xml UI layout that the user controls the TV with. Upon entering the application, BluetoothChat checks to see if Bluetooth is enabled on the phone, and if it isn't, requests the user to turn it on or exits

upon failure. The top right of the layout tells the user the current connection status. All the buttons are deactivated until connected. Pressing the menu button on the phone pops up a menu of devices to connect to. Once a successful connection has been achieved, the buttons are activated, the status is changed to connected, and the user is free to control their Sony TV.

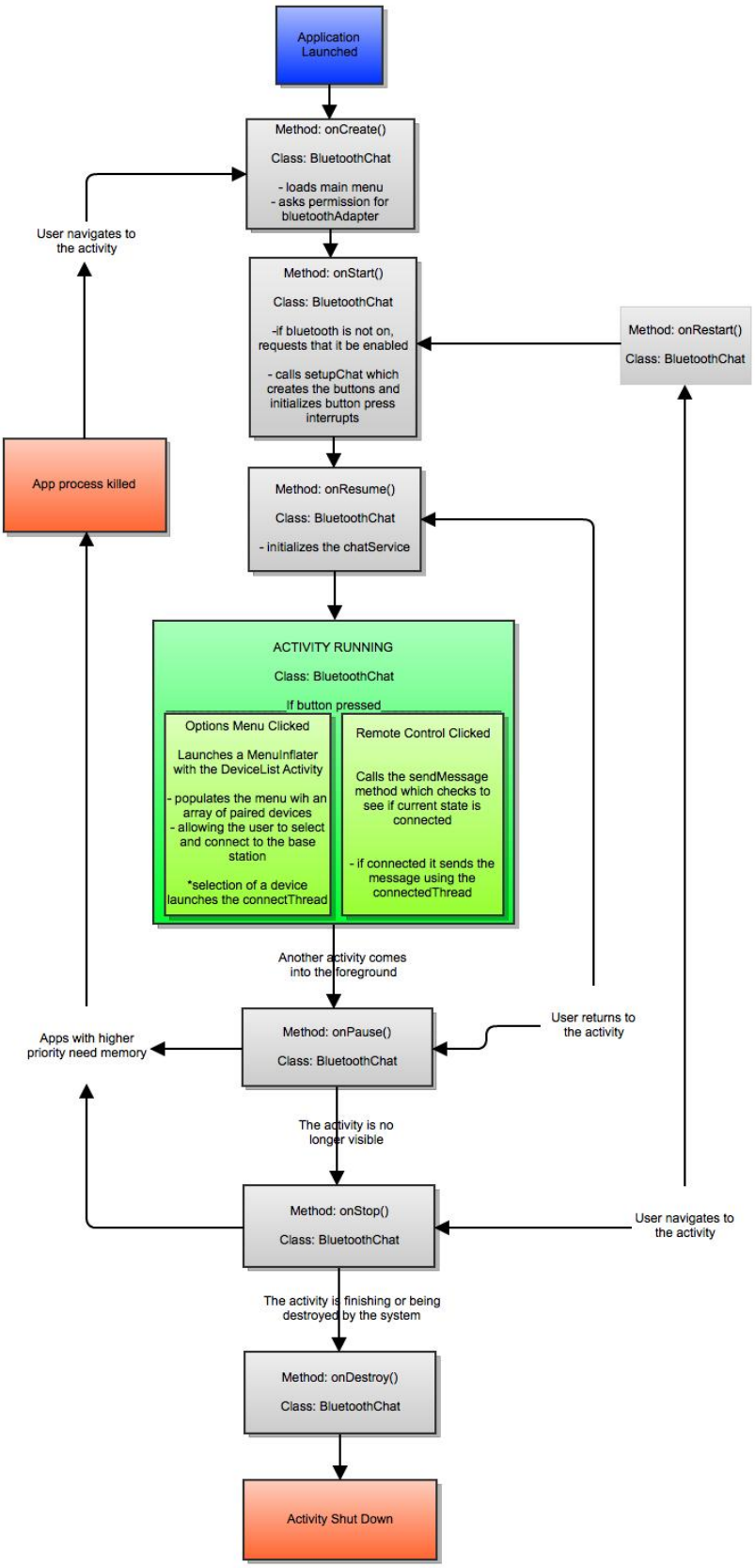

Figure 6: Flowchart Diagram of Bluemote Android Application

Conclusion & Future Expansion

4.0 Conclusion

This project showed a proof-of-concept for a much broader idea. It is intended to prove the capabilities of your mobile phone in your home and a step towards home automation. A user with an android mobile phone is able to control their Sony TV by connecting to a base station, controlled by an Arduino Uno Development Board modulated with a Bluetooth Mate Silver and an IR LED.

Working on this project required knowledge on how Pulse Width Modulation works (specifically the Sony SIRC Protocol). I had to teach myself how to program an Arduino Uno and I had to teach myself the Android OS (specifically Gingerbread). Obvious knowledge of Java and C were necessary. Overall, the project combined my CPE curriculum evenly between the Computer Science software side and the Electrical Engineering hardware side. The project was completed successfully, except for the one drawback of it not being a universal remote.

5.0 Future Expansion

In the future, I hope to expand the remote control peripheral to be capable of controlling every IR electronic in the home, making it truly universal. I hope to move the remote from the base station and make it a single node that has its own controller that communicates to the base station via the Zigbee Protocol. Also, further expansion will encompass more peripherals that would expand the applications ability to control more than just a TV; for example; controlling the lights in the home or managing the security system.

Appendix A

```
int IRledPin =  13;    // IR LED connected to digital pin 13
int inByte = 0;

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the IR digital pin as an output:
  pinMode(IRledPin, OUTPUT);

  Serial.begin(115200);
}

void loop()
{
 // Serial.println("Sending IR signal");

  // read from port 0
  if (Serial.available()) {
    //Serial.println("Got data");
    int inByte = Serial.read();
    //Serial.println(inByte, BYTE);

    switch (inByte) {
  case 'p':
  //  Serial.println("power");
    SonyOn();
    break;
  case '-':
  //  Serial.println("Volume down");
    SonyVolumeDown();
    break;
  case '+':
  //  Serial.println("Volume up");
    SonyVolumeUp();
    break;
  case 'd':
//    Serial.println("Channel down");
    SonyChannelDown();
    break;
  case 'e':
//    Serial.println("Enter");
    SonyEnter();
    break;
  case 'i':
//    Serial.println("Input");
```

```
    SonyInput();
    break;
  case 'j':
//    Serial.println("Jump");
    SonyJump();
    break;
  case 'm':
//    Serial.println("Mute");
    SonyMute();
    break;
  case 's':
//    Serial.println("Sleep");
    SonySleep();
    break;
  case 'u':
//    Serial.println("Channel up");
    SonyChannelUp();
    break;
  case '0':
//    Serial.println("0");
    Sony0();
    break;
  case '1':
//    Serial.println("1");
    Sony1();
    break;
  case '2':
//    Serial.println("2");
    Sony2();
    break;
  case '3':
//    Serial.println("3");
    Sony3();
    break;
  case '4':
//    Serial.println("4");
    Sony4();
    break;
  case '5':
//    Serial.println("5");
    Sony5();
    break;
  case '6':
//    Serial.println("6");
    Sony6();
    break;
```

```
  case '7':
//    Serial.println("7");
    Sony7();
    break;
  case '8':
//    Serial.println("8");
    Sony8();
    break;
  case '9':
//    Serial.println("9");
    Sony9();
    break;
  }

  }

}

// This procedure sends a 38KHz pulse to the IRledPin
// for a certain # of microseconds. We'll use this whenever we need to send codes
void pulseIR(long microsecs) {
          // we'll count down from the number of microseconds we are told to wait

          cli();  // this turns off any background interrupts

          while (microsecs > 0) {
            // 38 kHz is about 13 microseconds high and 13 microseconds low
            digitalWrite(IRledPin, HIGH);  // this takes about 4 microseconds to happen
            delayMicroseconds(9);         // hang out for 9 microseconds
            digitalWrite(IRledPin, LOW);   // this also takes about 4 microseconds
            delayMicroseconds(9);         // hang out for 9 microseconds

            // so 26 microseconds altogether
            microsecs -= 26;
  }

  sei();  // this turns them back on
}

//The Sony SIRC Protocol requires you send the signal twice

void SonyOn() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
```

```
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);

delay(45);

  pulseIR(2400);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyVolumeDown(){
  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyVolumeUp() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);
```

```
    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony1() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony2() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
```

```
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);

delay(45);

  pulseIR(2400);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
```

```
  pulseIR(600);
}

void Sony3() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony4() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
```

```
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony5() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony6() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
```

```
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}
```

```
void Sony7() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony8() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
```

```
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony9() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void Sony0() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
```

```
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyChannelUp(){
```

```
pulseIR(2400);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);

delay(45);

  pulseIR(2400);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
 }

 void SonyChannelDown() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
```

```
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyMute() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyJump() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
```

```
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyEnter() {

  pulseIR(2400);
```

```
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);

delay(45);

  pulseIR(2400);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(600);
delayMicroseconds(600);
pulseIR(1200);
delayMicroseconds(600);
pulseIR(600);
```

```
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonySleep() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);

  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
```

```
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}

void SonyInput() {

  pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
```

```
  delay(45);

    pulseIR(2400);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(1200);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
  delayMicroseconds(600);
  pulseIR(600);
}
```

Appendix B

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.BluetoothChat;


import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

/**
 * This is the main Activity that displays the current chat session.
 */
public class BluetoothChat extends Activity {
    // Debugging
    private static final String TAG = "Remote!";
```

```
    private static final boolean D = true;

    // Message types sent from the BluetoothChatService Handler
    public static final int MESSAGE_STATE_CHANGE = 1;
    public static final int MESSAGE_READ = 2;
    public static final int MESSAGE_WRITE = 3;
    public static final int MESSAGE_DEVICE_NAME = 4;
    public static final int MESSAGE_TOAST = 5;

    // Key names received from the BluetoothChatService Handler
    public static final String DEVICE_NAME = "device_name";
    public static final String TOAST = "toast";

    // Intent request codes
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int REQUEST_ENABLE_BT = 3;

    // Layout Views
    private TextView mTitle;
    private ListView mConversationView;
   // private EditText mOutEditText;
   // private Button mSendButton;
    private Button mPowerButton;
    private Button mMuteButton;
    private Button mJumpButton;
    private Button mChupButton;
    private Button mChdownButton;
    private Button mVolupButton;
    private Button mVoldownButton;
    private Button mEnterButton;
    private Button mInputButton;
    private Button mSleepButton;
    private Button m0Button;
    private Button m1Button;
    private Button m2Button;
    private Button m3Button;
    private Button m4Button;
    private Button m5Button;
    private Button m6Button;
    private Button m7Button;
    private Button m8Button;
    private Button m9Button;


    // Name of the connected device
    private String mConnectedDeviceName = null;
```

```
    // Array adapter for the conversation thread
    private ArrayAdapter<String> mConversationArrayAdapter;
    // Local Bluetooth adapter
    private BluetoothAdapter mBluetoothAdapter = null;
    // Member object for the chat services
    private BluetoothChatService mChatService = null;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if(D) Log.e(TAG, "+++ ON CREATE +++");

        // Set up the window layout
        requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
        setContentView(R.layout.main);
        getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE,
R.layout.custom_title);

        // Set up the custom title
        mTitle = (TextView) findViewById(R.id.title_left_text);
        mTitle.setText(R.string.app_name);
        mTitle = (TextView) findViewById(R.id.title_right_text);

        // Get local Bluetooth adapter
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        // If the adapter is null, then Bluetooth is not supported
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, "Bluetooth is not available",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
    }

    @Override
    public void onStart() {
        super.onStart();
        if(D) Log.e(TAG, "++ ON START ++");

        // If BT is not on, request that it be enabled.
        // setupChat() will then be called during onActivityResult
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
    // Otherwise, setup the chat session
    } else {
      if (mChatService == null) setupChat();
    }
}

@Override
public synchronized void onResume() {
    super.onResume();
    if(D) Log.e(TAG, "+ ON RESUME +");

    // Performing this check in onResume() covers the case in which BT was
    // not enabled during onStart(), so we were paused to enable it...
    // onResume() will be called when ACTION_REQUEST_ENABLE activity returns.
    if (mChatService != null) {
      // Only if the state is STATE_NONE, do we know that we haven't started already
      if (mChatService.getState() == BluetoothChatService.STATE_NONE) {
        // Start the Bluetooth chat services
        mChatService.start();
      }
    }
}

private void setupChat() {
    Log.d(TAG, "setupChat()");

    // Initialize the array adapter for the conversation thread
    mConversationArrayAdapter = new ArrayAdapter<String>(this, R.layout.message);
    mConversationView = (ListView) findViewById(R.id.in);
    mConversationView.setAdapter(mConversationArrayAdapter);

    mPowerButton = (Button) findViewById(R.id.power);
    mPowerButton.setOnClickListener(new OnClickListener(){
      public void onClick(View v){
            sendMessage("p");
      }
    });

    mVolupButton = (Button) findViewById(R.id.volup);
    mVolupButton.setOnClickListener(new OnClickListener(){
      public void onClick(View v){
            sendMessage("+");
      }
    });
```

```
mVoldownButton = (Button) findViewById(R.id.voldown);
mVoldownButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("-");
    }
});

mChupButton = (Button) findViewById(R.id.chup);
mChupButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("u");
    }
});

mChdownButton = (Button) findViewById(R.id.chdown);
mChdownButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("d");
    }
});

mMuteButton = (Button) findViewById(R.id.mute);
mMuteButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("m");
    }
});

mJumpButton = (Button) findViewById(R.id.jump);
mJumpButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("j");
    }
});

mEnterButton = (Button) findViewById(R.id.enter);
mEnterButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("e");
    }
});

mInputButton = (Button) findViewById(R.id.input);
mInputButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("i");
```

```
    }
});

mSleepButton = (Button) findViewById(R.id.sleep);
mSleepButton.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("s");
    }
});

m0Button = (Button) findViewById(R.id.b0);
m0Button.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("0");
    }
});

m1Button = (Button) findViewById(R.id.b1);
m1Button.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("1");
    }
});

m2Button = (Button) findViewById(R.id.b2);
m2Button.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("2");
    }
});

m3Button = (Button) findViewById(R.id.b3);
m3Button.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("3");
    }
});

m4Button = (Button) findViewById(R.id.b4);
m4Button.setOnClickListener(new OnClickListener(){
    public void onClick(View v){
            sendMessage("4");
    }
});

m5Button = (Button) findViewById(R.id.b5);
```

```
    m5Button.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
                sendMessage("5");
        }
    });

    m6Button = (Button) findViewById(R.id.b6);
    m6Button.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
                sendMessage("6");
        }
    });

    m7Button = (Button) findViewById(R.id.b7);
    m7Button.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
                sendMessage("7");
        }
    });

    m8Button = (Button) findViewById(R.id.b8);
    m8Button.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
                sendMessage("8");
        }
    });

    m9Button = (Button) findViewById(R.id.b9);
    m9Button.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
                sendMessage("9");
        }
    });

    // Initialize the BluetoothChatService to perform bluetooth connections
    mChatService = new BluetoothChatService(this, mHandler);

}

@Override
public synchronized void onPause() {
    super.onPause();
    if(D) Log.e(TAG, "- ON PAUSE -");
}

@Override
```

```
public void onStop() {
    super.onStop();
    if(D) Log.e(TAG, "-- ON STOP --");
}

@Override
public void onDestroy() {
    super.onDestroy();
    // Stop the Bluetooth chat services
    if (mChatService != null) mChatService.stop();
    if(D) Log.e(TAG, "--- ON DESTROY ---");
}

/**
 * Sends a message.
 * @param message  A string of text to send.
 */
private void sendMessage(String message) {
    // Check that we're actually connected before trying anything
    if (mChatService.getState() != BluetoothChatService.STATE_CONNECTED) {
        Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
        return;
    }

    // Check that there's actually something to send
    if (message.length() > 0) {
        // Get the message bytes and tell the BluetoothChatService to write
        byte[] send = message.getBytes();
        mChatService.write(send);

    }
}

// The Handler that gets information back from the BluetoothChatService
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
            switch (msg.arg1) {
            case BluetoothChatService.STATE_CONNECTED:
                mTitle.setText(R.string.title_connected_to);
                mTitle.append(mConnectedDeviceName);
                mConversationArrayAdapter.clear();
                break;
```

```
            case BluetoothChatService.STATE_CONNECTING:
                mTitle.setText(R.string.title_connecting);
                break;
            case BluetoothChatService.STATE_NOTCONNECTED:
            case BluetoothChatService.STATE_NONE:
                mTitle.setText(R.string.title_not_connected);
                break;
            }
            break;
        case MESSAGE_WRITE:
            byte[] writeBuf = (byte[]) msg.obj;
            // construct a string from the buffer
            String writeMessage = new String(writeBuf);
            mConversationArrayAdapter.add("Me:  " + writeMessage);
            break;
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            mConversationArrayAdapter.add(mConnectedDeviceName+":  " +
readMessage);
            break;
        case MESSAGE_DEVICE_NAME:
            // save the connected device's name
            mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
            Toast.makeText(getApplicationContext(), "Connected to "
                           + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
            break;
        case MESSAGE_TOAST:
            Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                           Toast.LENGTH_SHORT).show();
            break;
        }
    }
};

public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(D) Log.d(TAG, "onActivityResult " + resultCode);
    switch (requestCode) {
    case REQUEST_CONNECT_DEVICE:
        // When DeviceListActivity returns with a device to connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data);
        }
        break;
    case REQUEST_ENABLE_BT:
```

```
            // When the request to enable Bluetooth returns
            if (resultCode == Activity.RESULT_OK) {
                // Bluetooth is now enabled, so set up a chat session
                setupChat();
            } else {
                // User did not enable Bluetooth or an error occured
                Log.d(TAG, "BT not enabled");
                Toast.makeText(this, R.string.bt_not_enabled_leaving,
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    private void connectDevice(Intent data) {
        // Get the device MAC address
        String address = data.getExtras()
            .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
        // Get the BLuetoothDevice object
        BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
        // Attempt to connect to the device
        mChatService.connect(device);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.option_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent serverIntent = null;
        switch (item.getItemId()) {
        case R.id.connect:
            // Launch the DeviceListActivity to see devices and do scan
            serverIntent = new Intent(this, DeviceListActivity.class);
            startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);
            return true;
        }
        return false;
    }

}
```

Appendix C

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.BluetoothChat;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

/**
 * This class does all the work for setting up and managing Bluetooth
 * connections with other devices. It has a thread that listens for
 * incoming connections, a thread for connecting with a device, and a
 * thread for performing data transmissions when connected.
 */
public class BluetoothChatService {
    // Debugging
    private static final String TAG = "BluetoothChatService";
    private static final boolean D = true;

    // Name for the SDP record when creating server socket
   // private static final String NAME_SECURE = "BluetoothChatSecure";
```

```
    // Unique UUID for this application
    private static final UUID MY_UUID =
        UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");

    // Member fields
    private final BluetoothAdapter mAdapter;
    private final Handler mHandler;
    private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;
    private int mState;

    // Constants that indicate the current connection state
    public static final int STATE_NONE = 0;       // we're doing nothing
    public static final int STATE_NOTCONNECTED = 1;     // now listening for incoming
connections
    public static final int STATE_CONNECTING = 2; // now initiating an outgoing
connection
    public static final int STATE_CONNECTED = 3;  // now connected to a remote
device

    /**
     * Constructor. Prepares a new BluetoothChat session.
     * @param context  The UI Activity Context
     * @param handler  A Handler to send messages back to the UI Activity
     */
    public BluetoothChatService(Context context, Handler handler) {
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        mState = STATE_NONE;
        mHandler = handler;
    }

    /**
     * Set the current state of the chat connection
     * @param state  An integer defining the current connection state
     */
    private synchronized void setState(int state) {
        if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
        mState = state;

        // Give the new state to the Handler so the UI Activity can update
        mHandler.obtainMessage(BluetoothChat.MESSAGE_STATE_CHANGE, state, -
1).sendToTarget();
    }

    /**
```

```
     * Return the current connection state. */
    public synchronized int getState() {
        return mState;
    }

    /**
     * Start the chat service. Specifically start AcceptThread to begin a
     * session in listening (server) mode. Called by the Activity onResume() */
    public synchronized void start() {
        if (D) Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}

        setState(STATE_NOTCONNECTED);

    }

    /**
     * Start the ConnectThread to initiate a connection to a remote device.
     * @param device  The BluetoothDevice to connect
     * @param secure Socket Security type - Secure (true) , Insecure (false)
     */
    public synchronized void connect(BluetoothDevice device) {
        if (D) Log.d(TAG, "connect to: " + device);

        // Cancel any thread attempting to make a connection
        if (mState == STATE_CONNECTING) {
            if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread =
null;}
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}

        // Start the thread to connect with the given device
        mConnectThread = new ConnectThread(device);
        mConnectThread.start();
        setState(STATE_CONNECTING);
    }
```

```
    /**
     * Start the ConnectedThread to begin managing a Bluetooth connection
     * @param socket  The BluetoothSocket on which the connection was made
     * @param device  The BluetoothDevice that has been connected
     */
    public synchronized void connected(BluetoothSocket socket, BluetoothDevice
            device, final String socketType) {
        if (D) Log.d(TAG, "connected, Socket Type:" + socketType);

        // Cancel the thread that completed the connection
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread =
null;}

        // Start the thread to manage the connection and perform transmissions
        mConnectedThread = new ConnectedThread(socket, socketType);
        mConnectedThread.start();

        // Send the name of the connected device back to the UI Activity
        Message msg =
mHandler.obtainMessage(BluetoothChat.MESSAGE_DEVICE_NAME);
        Bundle bundle = new Bundle();
        bundle.putString(BluetoothChat.DEVICE_NAME, device.getName());
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        setState(STATE_CONNECTED);
    }

    /**
     * Stop all threads
     */
    public synchronized void stop() {
        if (D) Log.d(TAG, "stop");

        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }

        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }
```

```
    setState(STATE_NONE);
}

/**
 * Write to the ConnectedThread in an unsynchronized manner
 * @param out The bytes to write
 * @see ConnectedThread#write(byte[])
 */
public void write(byte[] out) {
    // Create temporary object
    ConnectedThread r;
    // Synchronize a copy of the ConnectedThread
    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mConnectedThread;
    }
    // Perform the write unsynchronized
    r.write(out);
}

/**
 * Indicate that the connection attempt failed and notify the UI Activity.
 */
private void connectionFailed() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(BluetoothChat.TOAST, "Unable to connect device");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    // Start the service over to restart listening mode
    BluetoothChatService.this.start();
}

/**
 * Indicate that the connection was lost and notify the UI Activity.
 */
private void connectionLost() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(BluetoothChat.TOAST, "Device connection was lost");
    msg.setData(bundle);
    mHandler.sendMessage(msg);
```

```
    // Start the service over to restart listening mode
    BluetoothChatService.this.start();
}


/**
 * This thread runs while attempting to make an outgoing connection
 * with a device. It runs straight through; the connection either
 * succeeds or fails.
 */
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    private String mSocketType;

    public ConnectThread(BluetoothDevice device) {
        mmDevice = device;
        BluetoothSocket tmp = null;

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {

                tmp = device.createRfcommSocketToServiceRecord(MY_UUID);

        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "create() failed", e);
        }
        mmSocket = tmp;
    }

    public void run() {
        Log.i(TAG, "BEGIN mConnectThread SocketType:" + mSocketType);
        setName("ConnectThread" + mSocketType);

        // Always cancel discovery because it will slow down a connection
        mAdapter.cancelDiscovery();

        // Make a connection to the BluetoothSocket
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            mmSocket.connect();
        } catch (IOException e) {
            // Close the socket
```

```
            try {
                mmSocket.close();
            } catch (IOException e2) {
                Log.e(TAG, "unable to close() " + mSocketType +
                        " socket during connection failure", e2);
            }
            connectionFailed();
            return;
        }

        // Reset the ConnectThread because we're done
        synchronized (BluetoothChatService.this) {
            mConnectThread = null;
        }

        // Start the connected thread
        connected(mmSocket, mmDevice, mSocketType);
    }

    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "close() of connect " + mSocketType + " socket failed", e);
        }
    }
}

/**
 * This thread runs during a connection with a remote device.
 * It handles all incoming and outgoing transmissions.
 */
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket, String socketType) {
        Log.d(TAG, "create ConnectedThread: " + socketType);
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the BluetoothSocket input and output streams
        try {
            tmpIn = socket.getInputStream();
```

```
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, "temp sockets not created", e);
        }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }

    public void run() {
        Log.i(TAG, "BEGIN mConnectedThread");
        byte[] buffer = new byte[1024];
        int bytes;

        // Keep listening to the InputStream while connected
        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);

                // Send the obtained bytes to the UI Activity
                mHandler.obtainMessage(BluetoothChat.MESSAGE_READ, bytes, -1,
buffer)
                        .sendToTarget();
            } catch (IOException e) {
                Log.e(TAG, "disconnected", e);
                connectionLost();
                break;
            }
        }
    }

    /**
     * Write to the connected OutStream.
     * @param buffer  The bytes to write
     */
    public void write(byte[] buffer) {
        try {
            mmOutStream.write(buffer);

            // Share the sent message back to the UI Activity
            mHandler.obtainMessage(BluetoothChat.MESSAGE_WRITE, -1, -1, buffer)
                    .sendToTarget();
        } catch (IOException e) {
            Log.e(TAG, "Exception during write", e);
        }
```

```
        }

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "close() of connect socket failed", e);
            }
        }
    }
}
```

Appendix D

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.BluetoothChat;

import java.util.Set;


import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

/**
 * This Activity appears as a dialog. It lists any paired devices and
 * devices detected in the area after discovery. When a device is chosen
 * by the user, the MAC address of the device is sent back to the parent
 * Activity in the result Intent.
```

```
 */
public class DeviceListActivity extends Activity {
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.device_list);

        // Set result CANCELED incase the user backs out
        setResult(Activity.RESULT_CANCELED);

        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });

        // Initialize array adapters. One for already paired devices and
        // one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);
```

```
        // Find and set up the ListView for newly discovered devices
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);

        // Register for broadcasts when a device is discovered
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);

        // Register for broadcasts when discovery has finished
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);

        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();

        // Get a set of currently paired devices
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

        // If there are paired devices, add each one to the ArrayAdapter
        if (pairedDevices.size() > 0) {
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices = getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        // Make sure we're not doing discovery anymore
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }

        // Unregister broadcast listeners
        this.unregisterReceiver(mReceiver);
    }

    /**
```

```
     * Start device discover with the BluetoothAdapter
     */
    private void doDiscovery() {
        if (D) Log.d(TAG, "doDiscovery()");

        // Indicate scanning in the title
        setProgressBarIndeterminateVisibility(true);
        setTitle(R.string.scanning);

        // Turn on sub-title for new devices
        findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

        // If we're already discovering, stop it
        if (mBtAdapter.isDiscovering()) {
            mBtAdapter.cancelDiscovery();
        }

        // Request discover from BluetoothAdapter
        mBtAdapter.startDiscovery();
    }

    // The on-click listener for all devices in the ListViews
    private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
            // Cancel discovery because it's costly and we're about to connect
            mBtAdapter.cancelDiscovery();

            // Get the device MAC address, which is the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Create the result Intent and include the MAC address
            Intent intent = new Intent();
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

            // Set result and finish this Activity
            setResult(Activity.RESULT_OK, intent);
            finish();
        }
    };

    // The BroadcastReceiver that listens for discovered devices and
    // changes the title when discovery is finished
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
```

```
            String action = intent.getAction();

            // When discovery finds a device
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // If it's already paired, skip it, because it's been listed already
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                }
            // When discovery is finished, change the Activity title
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
                    String noDevices = getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };

}
```