WIFI Controlled AC Switch
System Documentation

By:
Chris Maciel
Reed Reese-Steiner
Shaun Akers

Table of Contents:

Engineering Requirements and Video Links for WIFI Controlled AC Switch

Base Engineering Requirements

1. **Customer Requirement:** The switch needs to work for household lights.
   Engineering Requirement: The system will turn off and on at least 2 independent household lamps with up to 100W incandescent bulbs and report current watts delivered by each channel.
   Video: https://drive.google.com/open?id=1EzzBayjD8HZTX3Ef7fvozKCWXt7P_GBz
2. **Customer Requirement:** The system needs to have a simple application
   Engineering Requirement: 9 out of 10 users will be able to turn the switches off and on from a mobile phone in less than 10 seconds without any training or having previously seen the interface.
   Video: TO DO
3. **Customer Requirement:** The system must be safe.
   Engineering Requirement: The system will use only US standard plugins for connecting to external devices and will not allow any object with a diameter greater than 1 mm to enter the enclosure, and will be disabled if more than 5A is drawn from the wall power.
   Video: https://drive.google.com/open?id=1EzzBayjD8HZTX3Ef7fvozKCWXt7P_GBz
4. **Customer Requirement:** The system must have a timer
   Engineering Requirement: The system must turn off the output after a time of up to 1 hour ± 1 minute when enabled.
   Video: https://drive.google.com/open?id=1EzzBayjD8HZTX3Ef7fvozKCWXt7P_GBz
5. **Customer Requirement:** The system must be wireless
   Engineering Requirement: The system will be able to accept commands from a mobile phone over 20 feet away from the plugins.
   Video: TO DO

Custom Engineering Requirements

6. **Customer Requirement:** System must provide power consumption history
   Engineering Requirement: Inside the UI application, the system will provide a plot of at least 24 hours of power consumption for each outlet, sampled at least 6 times per hour
   Video: https://drive.google.com/open?id=1EzzBayjD8HZTX3Ef7fvozKCWXt7P_GBz
7. **Customer Requirement:** System should have a display that informs users on device status
   Engineering Requirement: System will have two indicating LEDs per channel mounted on the outlet cover plate, one to indicate if an outlet is powered, and one to indicate if the timer is enabled.
   Video: https://drive.google.com/open?id=1EzzBayjD8HZTX3Ef7fvozKCWXt7P_GBz

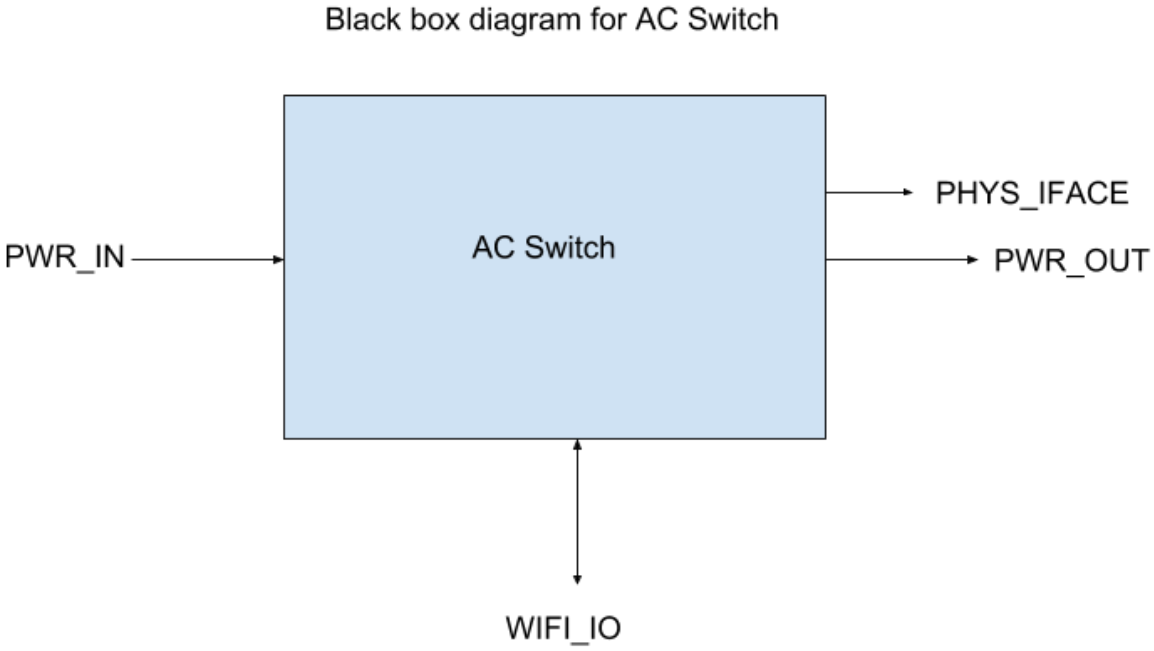
Usability Survey

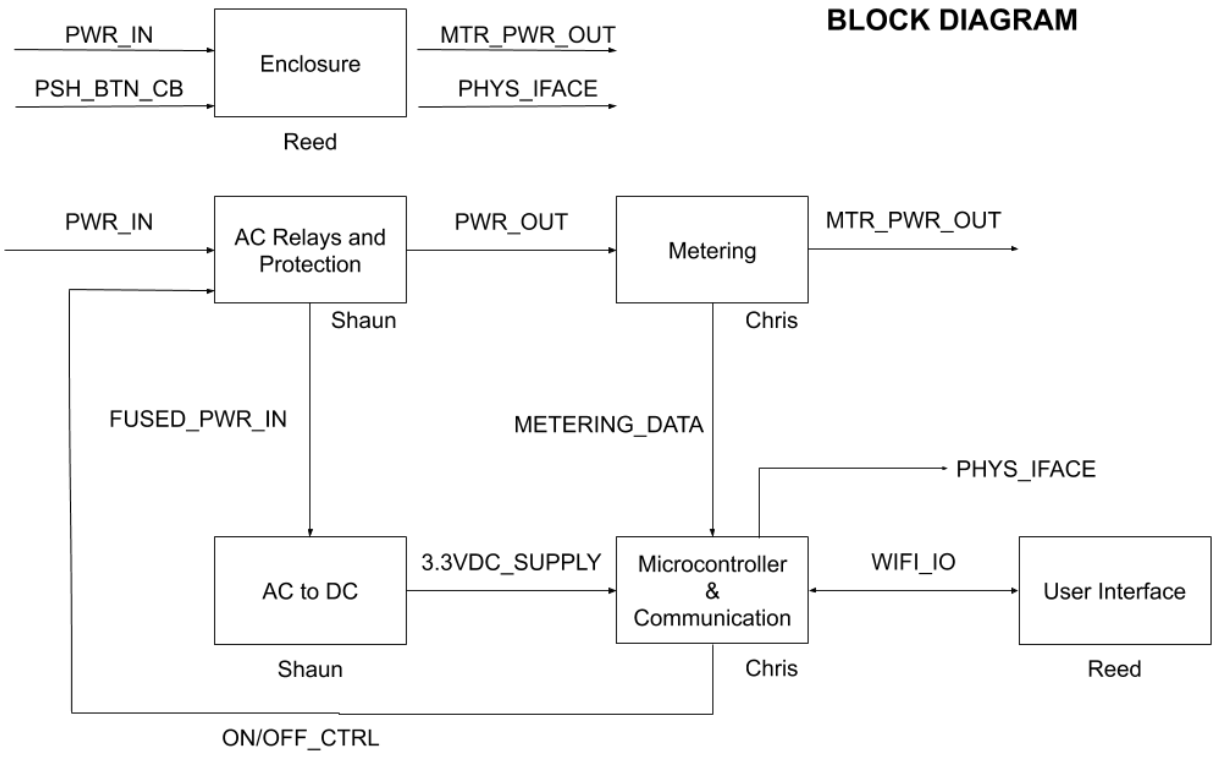
WiFi-Controlled AC Switch
Thunder Eagles

Are you able to turn the switches on and off within 10 seconds of seeing the Android app?

Yes/No	Name	Signature
Yes	Vladimir Vesely	Vladimir Vesely
Yes	Ian Bunch	Ian Bunch
Yes	Zan Paul	
yes	Aditya Kothari	Adit Kothari
yes	Jiecong Chen	
Yes	Jordan Colmenero	Jordan Colmenero
Yes	Juan Angeles	
Yes	Jacob Cook	Jacob
Yes	Andrew Leichner	Andrew Leichner
Yes	Aaron Walder	

Block Diagram



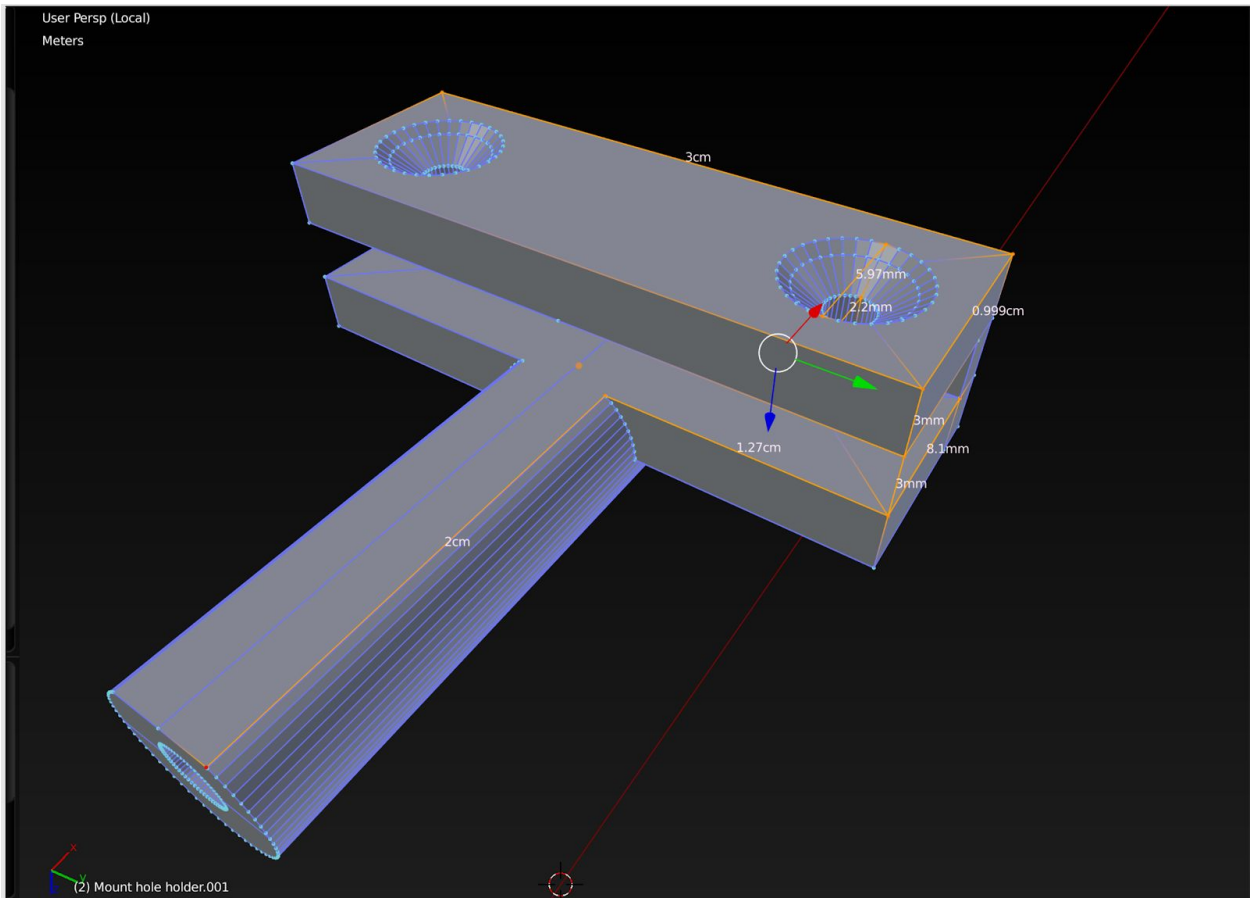


Interface Definitions

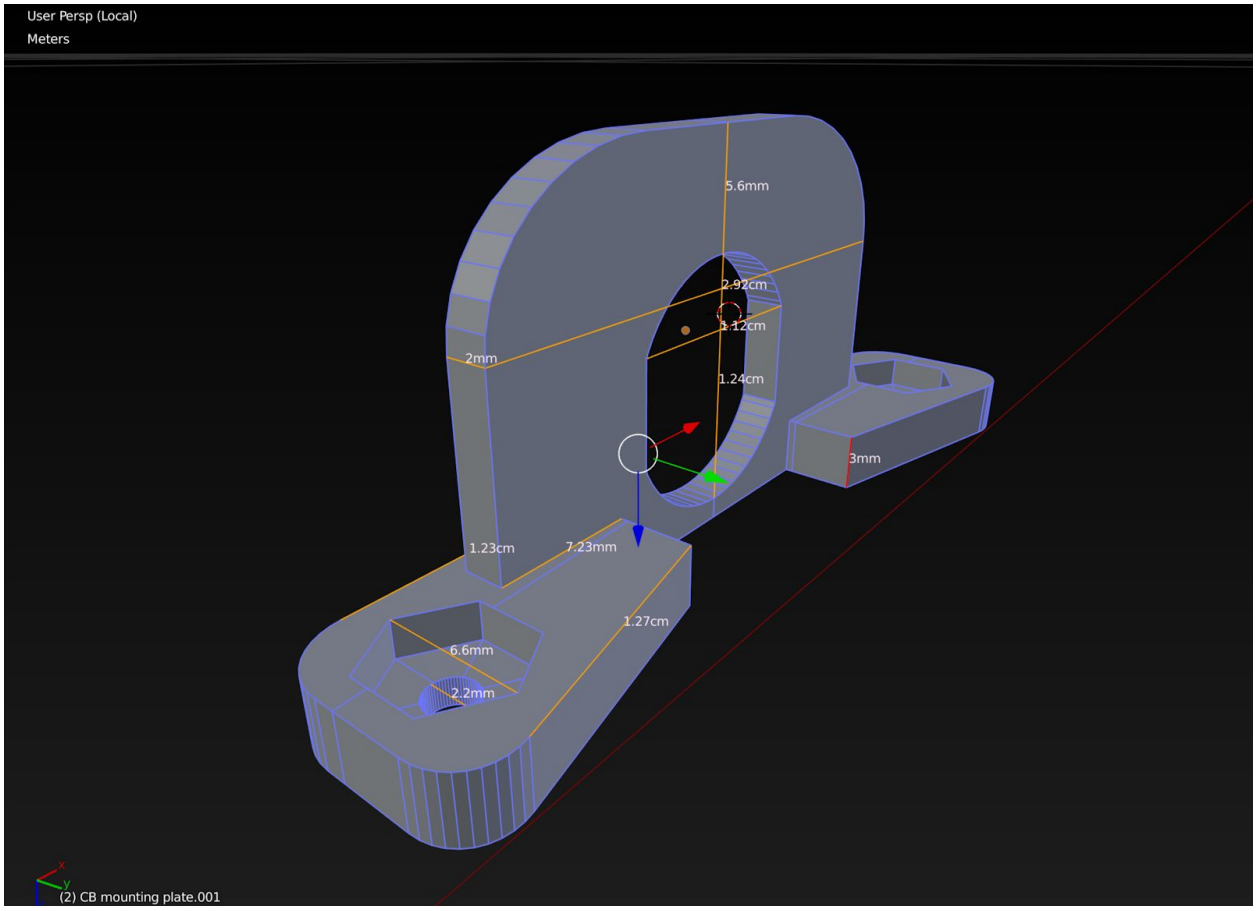
Interface	Description
PWR_IN	AC power supply from mains Vrms = 120V Ipeak = 5A f = 60Hz
PWR_OUT	Current-limited, switched AC supply, unmetered Vrms = 120V Ipeak = 5A f = 60Hz
MTR_PWR_OUT	Same as PWR_OUT, after its voltage and current have been measured Vrms = 120V Ipeak = 5A f = 60Hz
FUSED_POWER_IN	AC power supply from mains, protected by a circuit breaker Input to the AC→DC converter Vrms = 120V Ipeak = 1A f = 60Hz
3.3VDC_SUPPLY	3.3V DC supply to power the microcontroller V = 3.3V Pmax = 3.3W
ON/OFF_CTRL	Low-voltage control signal from the microcontroller to control relays that switch PWR_OUT V = 3.3V Digital control signal
WIFI_IO	Wireless WiFi link between the microcontroller and a user's device or a local area network f = 2.4GHz or 5GHz Inputs: User switch commands, timer settings, commands from other devices Outputs: Current and voltage data, switch state, timer information
METERING_DATA	Current data, gathered from PWR_OUT using current sensor IC, Signals will be processed by an ADC onboard the microcontroller Vmax = 3.3V
PSH_BTN_CB	A button to reset the main circuit breaker for the device. Fits on the cover plate of a duplex receptacle. Should be accessible with devices plugged into both receptacles.
PHYS_IFACE	The physical user interface on the outside of the device, Including: - One LED indicator light per channel to indicate on/off status - One LED indicator light per channel for timer status

Enclosure Drawing

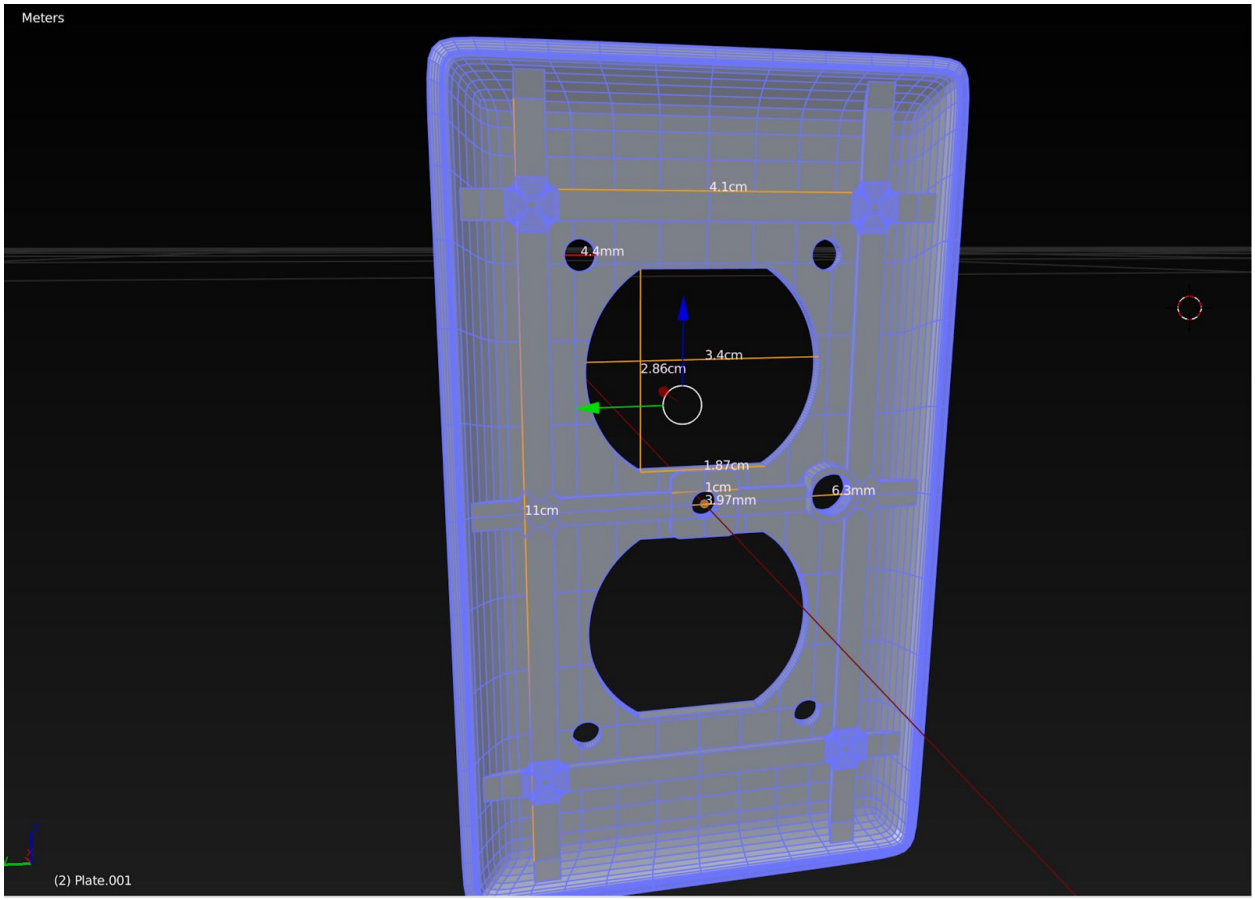
Receptacle mounting hole adapter



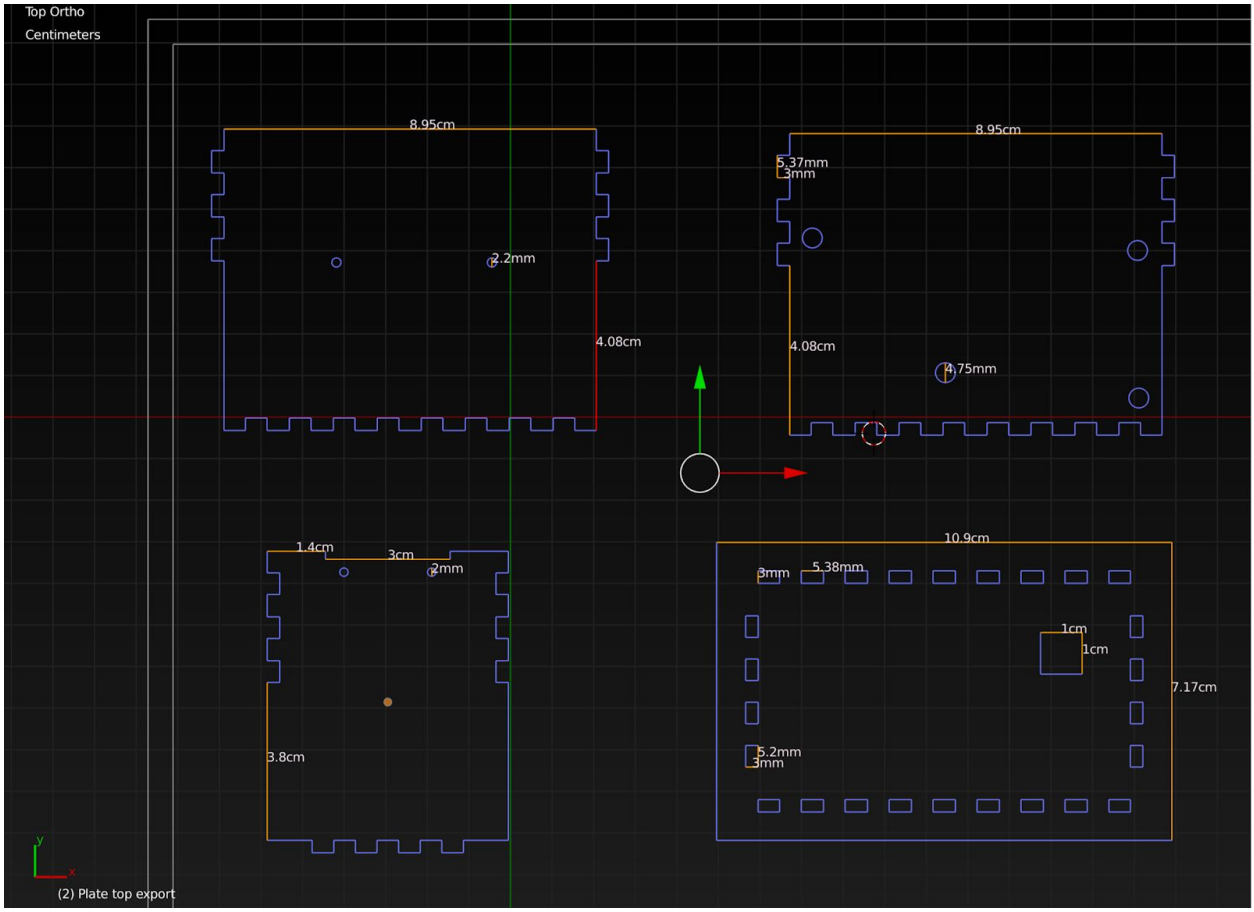
Circuit Breaker Mounting Adapter



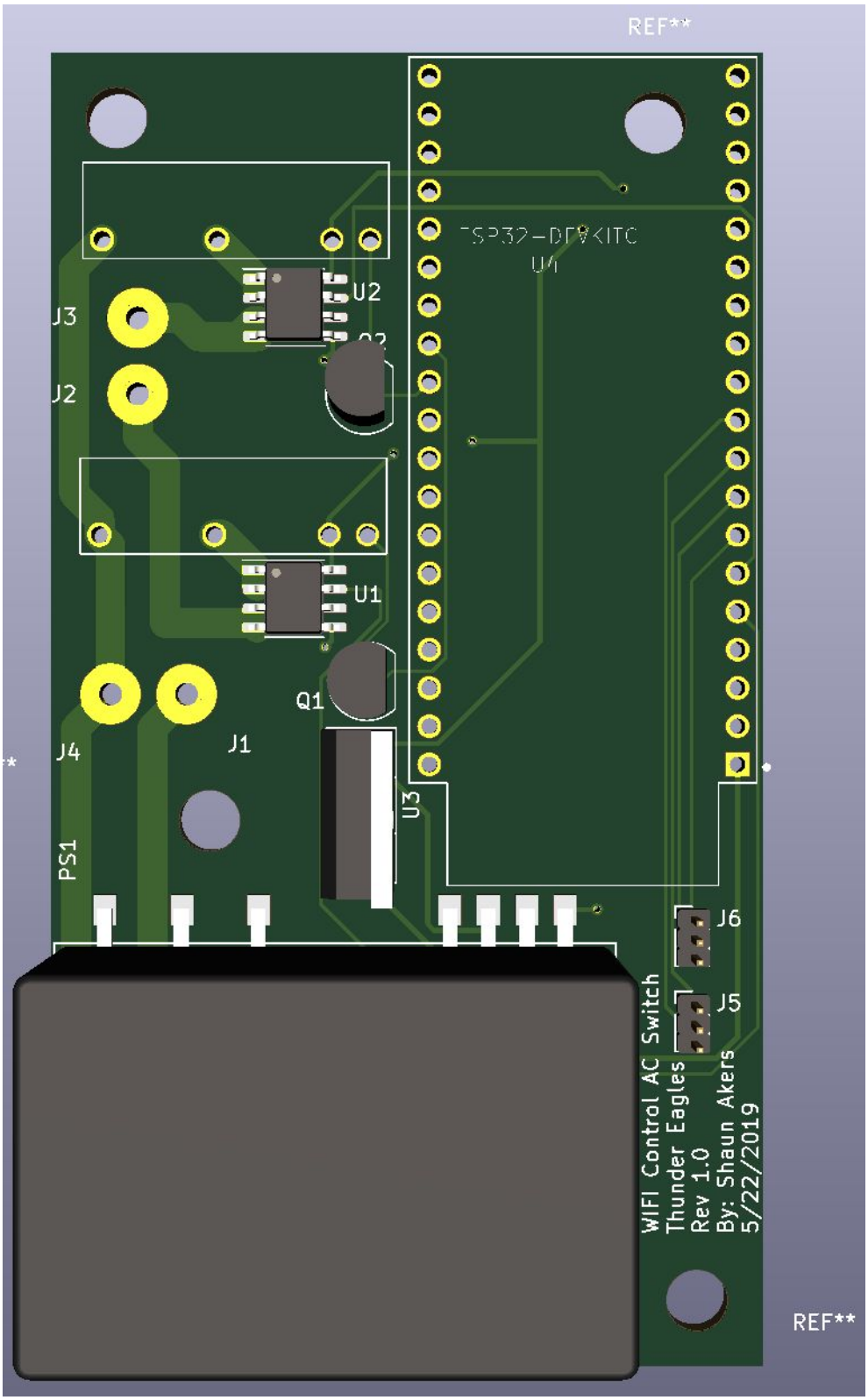
Receptacle Faceplate with LED and Circuit Breaker Cutouts



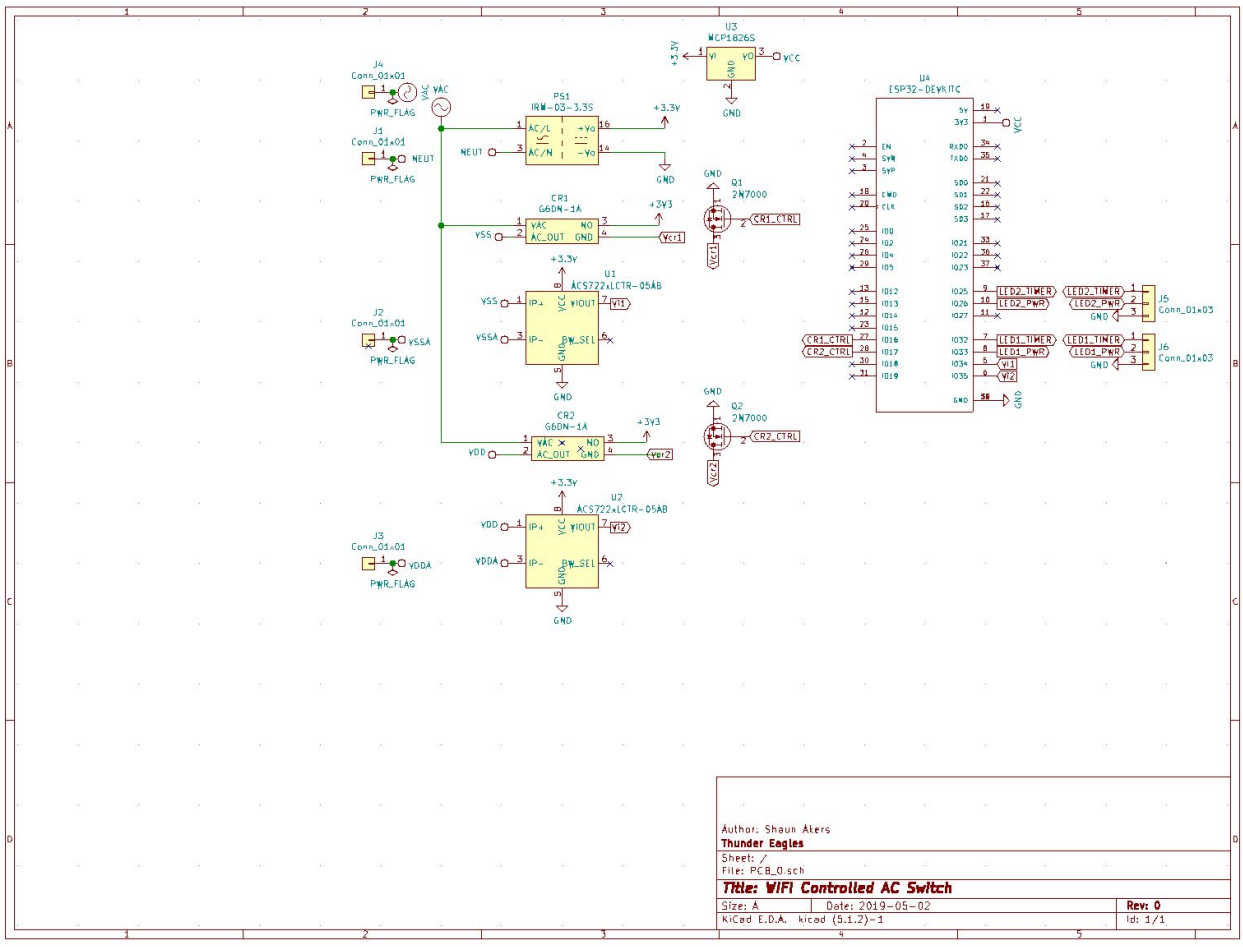
Enclosure Walls (top left then clockwise: left side, right side, top and bottom (identical, backplate)



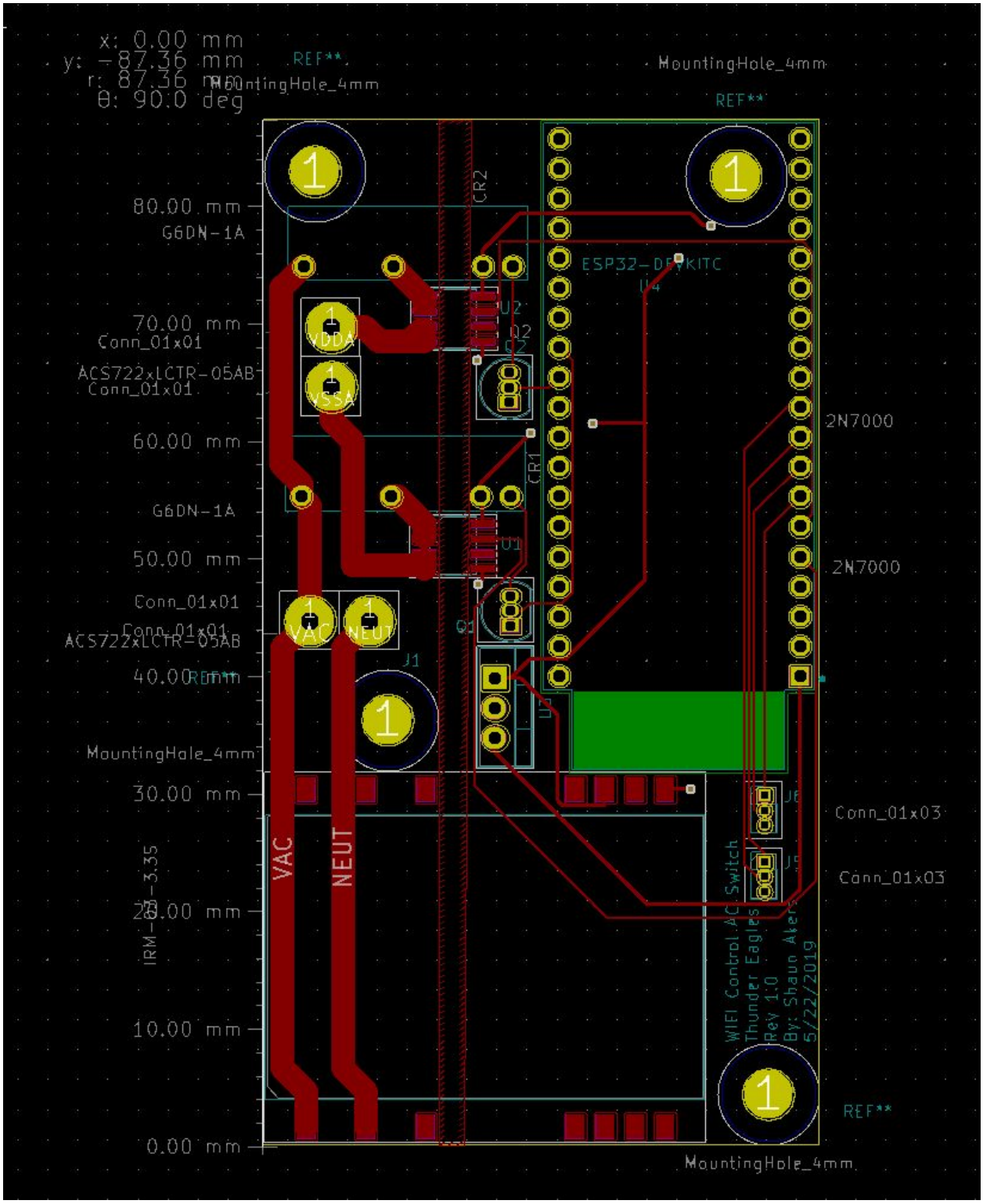
PCB Drawing


REF**
U2
J3
J2
U1
Q1
J4
J1
U3
PS1
J6
J5
WIFI Control AC Switch
Thunder Eagles
Rev 1.0
By: Shaun Akers
5/22/2019
REF**

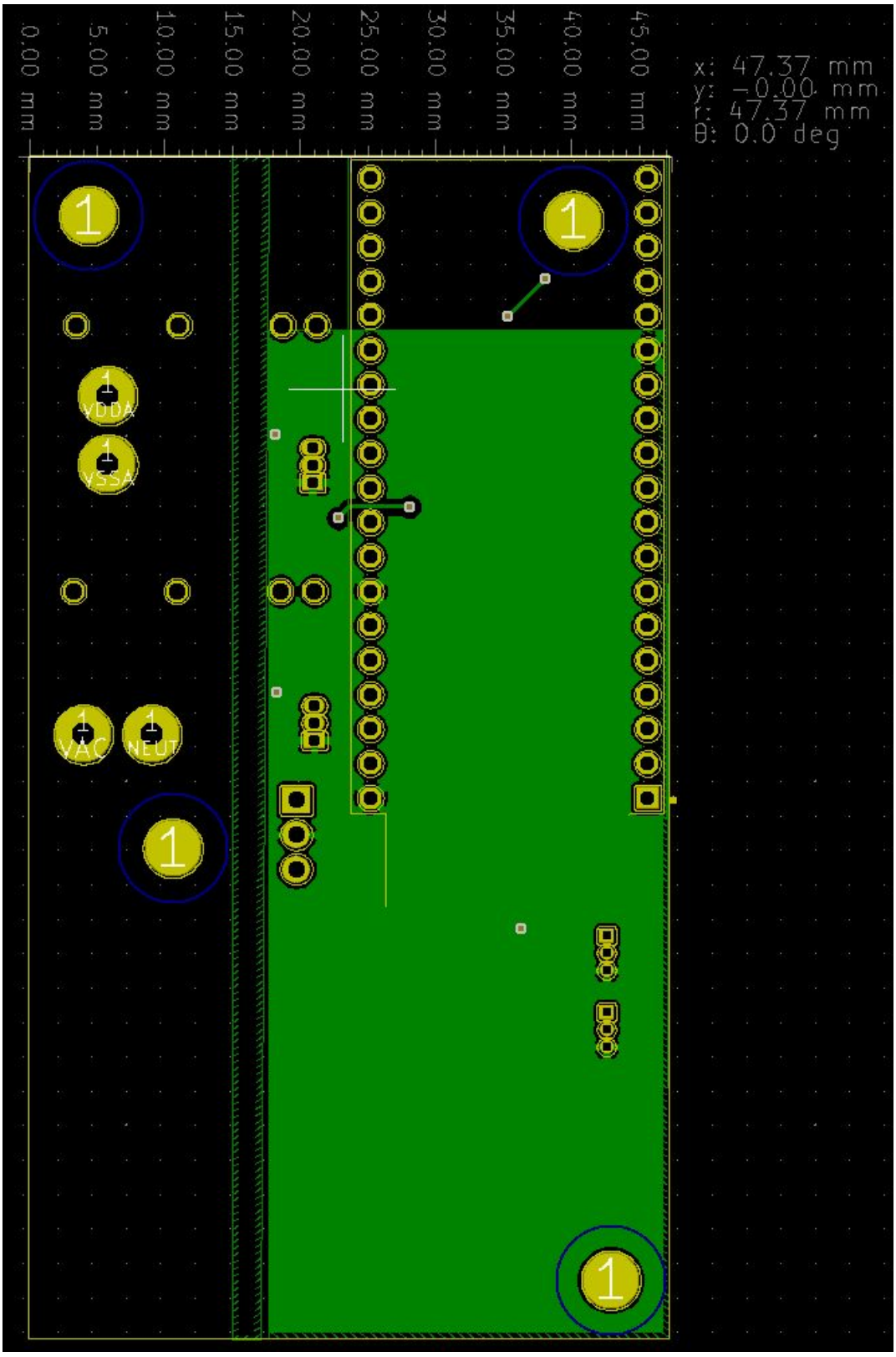
PCB Schematic



PCB Layers
Front



Back


0.00 mm
5.00 mm
10.00 mm
15.00 mm
20.00 mm
25.00 mm
30.00 mm
35.00 mm
40.00 mm
45.00 mm
x: 47.37 mm
y: -0.00 mm
r: 47.37 mm
θ: 0.0 deg
VDDA
VSSA
VAC
NEUT

Arduino Code

WifiSwitch.ino Page 1

```
/** @author Christopher Maciel @brief Wifi
Controlled AC Switch */#include <WiFi.h>
#include <WebServer.h> #include
<driver/adc.h> #include <FreeRTOS.h>

/* Put your SSID & Password */ const char* ssid = "Thunder Eagles"; // Enter
SSID here const char* password = "wifiswitch"; //Enter Password here

/* Put IP Address details */ IPAddress
local_ip(192, 168, 1, 1); IPAddress gateway(192,
168, 1, 1); IPAddress subnet(255, 255, 255, 0);
WebServer server(80);

const int samplingRate = 1; //Period of currentsensor data in minutes const int
CURRENTSENSOR1 = 34; const int CURRENTSENSOR2 = 35;

const int PWRLED1 = 33; const
int PWRLED2 = 26;

const int RELAY1 = 16; bool
RELAY1status = LOW; const int
RELAY2 = 17; bool RELAY2status
= LOW;

const int TIMERLED1 = 32; bool
TIMER1status = LOW; const int
TIMERLED2 = 25; bool
TIMER2status = LOW;
```

```
int iref1; int iref2;
TaskHandle_t Data;
void setup() {
Serial.begin(115200);
    pinMode(PWRLED1, OUTPUT);
    pinMode(PWRLED2, OUTPUT);
    pinMode(RELAY1, OUTPUT);
    pinMode(RELAY2, OUTPUT);
    pinMode(TIMERLED1, OUTPUT);
    pinMode(TIMERLED2, OUTPUT);
    WiFi.softAP(ssid, password); WiFi.softAPConfig(local_ip,
    gateway, subnet); delay(100);
    server.on("/", handle_OnConnect); server.on("/outlet1on",
    handle_outlet1on); server.on("/timer1set", handle_timer1set);
    server.on("/outlet1off", handle_outlet1off);
    server.on("/outlet2on", handle_outlet2on);
    server.on("/timer2set", handle_timer2set);
    server.on("/outlet2off", handle_outlet2off);
    server.onNotFound(handle_NotFound);
    server.begin(); Serial.println("HTTP server started");
    xTaskCreatePinnedToCore( DataCode, /* Function to implement
    the task */ "Data", /* Name of the task */ 10000, /* Stack size in
    words */ NULL, /* Task input parameter */ 1, /* Priority of the task
    */
```

WifiSwitch.ino Page 2

```
&Data, /* Task handle. */ 0); /* Core where the task should run */ }void
```

```
DataCode(void * parameter) { iref1 = analogRead(CURRENTSENSOR1);
iref2 = analogRead(CURRENTSENSOR2); double voltage1 =
analogRead(CURRENTSENSOR1); double voltage2 =
analogRead(CURRENTSENSOR2); double current1 = (abs(voltage1 − iref1)
* 264)/1000; double current2 = (abs(voltage2 − iref2) * 264)/1000; while
(true) { voltage1 = analogRead(CURRENTSENSOR1); voltage2 =
analogRead(CURRENTSENSOR2); current1 = (abs(voltage1 − iref1) *
264)/1000; current2 = (abs(voltage2 − iref2) * 264)/1000;
Serial.print("Current through Outlet 1 is: "); Serial.println(current1);
Serial.print("Current through Outlet 2 is: "); Serial.println(current2);
Serial.println('\n'); Serial.print("Voltage read at Outlet 1 is: ");
Serial.println(voltage1); Serial.print("Voltage read at Outlet 2 is: ");
Serial.println(voltage2); delay(samplingRate * 10000); } }void
timerDuration(int minutes, int outlet) { unsigned long StartTime = millis();
unsigned long CurrentTime = millis(); while ((CurrentTime − StartTime) <
minutes) { CurrentTime = millis(); delay(1000); } if (outlet == 1) {
digitalWrite(RELAY1, LOW); digitalWrite(PWRLED1, LOW);
digitalWrite(TIMERLED1, LOW); } else if (outlet == 2) {
digitalWrite(RELAY2, LOW); digitalWrite(PWRLED2, LOW);
digitalWrite(TIMERLED2, LOW); } }void loop() { server.handleClient();

    if (RELAY1status) { digitalWrite(RELAY1,
    HIGH); digitalWrite(PWRLED1, HIGH); }
    else { digitalWrite(RELAY1, LOW);
    digitalWrite(PWRLED1, LOW);
    digitalWrite(TIMERLED1, LOW); } if
    (RELAY2status) { digitalWrite(RELAY2,
    HIGH); digitalWrite(PWRLED2, HIGH); }
```

```
    else { digitalWrite(RELAY2, LOW);
    digitalWrite(PWRLED2, LOW);
    digitalWrite(TIMERLED2, LOW); }
```

WifiSwitch.ino Page 3

```
if (TIMER1status) { digitalWrite(TIMERLED1, HIGH); } else { digitalWrite(TIMERLED1, LOW); } if (TIMER2status) {
digitalWrite(TIMERLED2, HIGH); } else { digitalWrite(TIMERLED2, LOW); } }void handle_OnConnect() {
RELAY1status = LOW; RELAY2status = LOW; Serial.println("GPIO16 Status: OFF | GPIO17 Status: OFF");
server.send(200, "text/html", SendHTML(RELAY1status, RELAY2status)); }void handle_outlet1on() { RELAY1status
= HIGH; Serial.println("GPIO16 Status: ON"); server.send(200, "text/html", SendHTML(true, RELAY2status));
}void handle_outlet1off() { RELAY1status = LOW; Serial.println("GPIO16 Status: OFF"); server.send(200,
"text/html", SendHTML(false, RELAY2status)); }void handle_timer1set() { WiFiClient client; // Listen for incoming
clients if (client.available()) { // if there's bytes to read from the client, int minutes = client.read();
Serial.println(minutes); timerDuration(minutes, 1); } TIMER1status = HIGH; Serial.println("GPIO32 Status: ON");
}void handle_outlet2on() { RELAY2status = HIGH; Serial.println("GPIO17 Status: ON"); server.send(200,
"text/html", SendHTML(RELAY1status, true)); }void handle_outlet2off() { RELAY2status = LOW;
Serial.println("GPIO17 Status: OFF"); server.send(200, "text/html", SendHTML(RELAY1status, false)); }void
handle_timer2set() { WiFiClient client; // Listen for incoming clients if (client.available()) { // if there's bytes to
read from the client, int minutes = client.read(); Serial.println(minutes); timerDuration(minutes, 2); } TIMER2status
= HIGH; Serial.println("GPIO25 Status: ON"); }void handle_NotFound() { server.send(404, "text/plain", "Not
found"); }String SendHTML(uint8_t outlet1stat, uint8_t outlet2stat) {
```

WifiSwitch.ino Page 4

```
String ptr = "<!DOCTYPE html> <html>\n"; ptr += "<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale= 1.0, user-scalable=no\">\n"; ptr += "<title>Wifi Controlled
Switch</title>\n"; ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px a uto;
text-align: center;}\n"; ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {co
lor: #444444;margin-bottom: 50px;}\n"; ptr += ".button {display: block;width: 80px;background-color:
#3498db;border: none ;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px a
```

```
uto 35px;cursor: pointer;border−radius: 4px;}\n"; ptr += ".button−on {background−color: #3498db;}\n"; ptr += ".button−on:active {background−color: #2980b9;}\n"; ptr += ".button−off {background−color: #34495e;}\n"; ptr += ".button−off:active {background−color: #2c3e50;}\n"; ptr += "p {font−size: 14px;color: #888;margin−bottom: 10px;}\n"; ptr += "</style>\n"; ptr += "</head>\n"; ptr += "<body>\n"; ptr += "<h1>Thunder Eagles Web Server</h1>\n";

if (outlet1stat) { ptr += "<p>Outlet 1 Status: ON</p><a class=\"button button−off\" href=\"/outlet1 off\">OFF</a>\n"; } else { ptr += "<p>Outlet 1 Status: OFF</p><a class=\"button button−on\" href=\"/outlet1 on\">ON</a>\n"; } if (outlet2stat) { ptr += "<p>Outlet 2 Status: ON</p><a class=\"button button−off\" href=\"/outlet2 off\">OFF</a>\n"; } else { ptr += "<p>Outlet 2 Status: OFF</p><a class=\"button button−on\" href=\"/outlet2 on\">ON</a>\n"; }

ptr += "</body>\n"; ptr += "</html>\n"; return ptr; }
```

UI Code

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.ThdrEglsACSwitch">
  <uses-permission android:name="android.permission.INTERNET"/>

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".DisplayMessageActivity"
          android:parentActivityName=".MainActivity">
      <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
    </activity>

    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

SimpleDrawingView.java

```
package com.example.ThdrEglsACSwitch;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import java.util.Random;


class SimpleDrawingView extends View {
  private String TAG = "SimpleDrawingView";
  private final int paintColor = Color.BLACK;
  // defines paint and canvas
  private Paint drawPaint;
  private Paint drawAxisLabel;
  private Paint drawAxes;
  private Paint drawGridMajor;
  Random rng = new Random();
  private Path graphPath = new Path();
  private Path xAxisLine = new Path();
  private Path yAxisLine = new Path();
  private Path xGrid = new Path();
  private Path yGrid = new Path();

  public SimpleDrawingView(Context context, AttributeSet attrs) {
     super(context, attrs);
     setFocusable(true);
     setFocusableInTouchMode(true);
     setupPaint();
  }

  // Setup paint with color and stroke styles
  private void setupPaint() {
     drawPaint = new Paint();
     drawPaint.setColor(Color.RED);
     drawPaint.setAntiAlias(true);
     drawPaint.setStrokeWidth(3);
     drawPaint.setStyle(Paint.Style.STROKE);
     drawPaint.setStrokeJoin(Paint.Join.ROUND);
     drawPaint.setStrokeCap(Paint.Cap.ROUND);

     drawGridMajor = new Paint();
     drawGridMajor.setColor(Color.GRAY);
     drawGridMajor.setAntiAlias(true);
     drawGridMajor.setStrokeWidth(1);
     drawGridMajor.setStyle(Paint.Style.STROKE);
     drawGridMajor.setStrokeJoin(Paint.Join.ROUND);
```

```
    drawGridMajor.setStrokeCap(Paint.Cap.ROUND);

    drawAxisLabel = new Paint();
    drawAxisLabel.setTextSize(40);

    drawAxes = new Paint();
    drawAxes.setColor(Color.BLACK);
    drawAxes.setStyle(Paint.Style.STROKE);
    drawAxes.setStrokeWidth(5);
    drawAxes.setStrokeCap(Paint.Cap.ROUND);
}

public void setPath(Path newPath) {
    this.graphPath = newPath;

}

protected float graphTransformX(float input, int widthGraph, float maxX, int axisPaddingX){
    float result = input * widthGraph/maxX + axisPaddingX;
    return result;
}

protected float graphTransformY(float input, float maxY, int heightGraph, int axisPaddingY) {
    String TAG = "graphTransformY";
    float result = ((maxY - input)) * heightGraph/maxY + axisPaddingY;
    //float result = (1 - input) * heightGraph + axisPaddingY;
    return result;
}

@Override
protected void onDraw(Canvas canvas) {
    graphPath.reset();
    float maxCurrent = (float) 2.5; // Amps
    float voltage = 120; // Volts
    float gridIntervalXMaj = 3*60*60; // 3 hours
    float gridIntervalYMaj = 100; // Watts
    int sizeAxisLabel = 40; // pixels??
    int widthView = this.getWidth();
    int heightView = this.getHeight();
    int axisPaddingY = (int) (0.1 * heightView);
    int axisPaddingX = (int) (0.1 * widthView);
    int originX = axisPaddingX;
    int originY = heightView - axisPaddingY;
    float randomOffset = rng.nextFloat();

    int widthGraph = widthView - 2*axisPaddingX;
    int heightGraph = heightView - 2*axisPaddingY;

    float minX = 0;
    float maxX = 24*60*60; // 1 day in seconds
    float minY = 0;
    float maxY = maxCurrent*voltage;

    float x;
```

```
    float x1;
    float x2;
    float y;
    float y1;
    float y2;
    float some_data;

    for (float i = minX; i <= maxX; i += gridIntervalXMaj) {
      x = graphTransformX(i, widthGraph, maxX, axisPaddingX);
      y1 = graphTransformY(minY, maxY, heightGraph, axisPaddingY);
      y2 = graphTransformY(maxY, maxY, heightGraph, axisPaddingY);
      xGrid.moveTo(x, y1);
      xGrid.lineTo(x, y2);
      canvas.drawText(String.valueOf(i/60/60), (float) (x - 0.8*sizeAxisLabel), (float) (y1 + 1*sizeAxisLabel),
drawAxisLabel);
    }

    for (float i = minY; i <= maxY; i += gridIntervalYMaj) {
      y = graphTransformY(i, maxY, heightGraph, axisPaddingY);
      x1 = graphTransformX(minX, widthGraph, maxX, axisPaddingX);
      x2 = graphTransformX(maxX, widthGraph, maxX, axisPaddingX);
      yGrid.moveTo(x1, y);
      yGrid.lineTo(x2, y);
      canvas.drawText(String.valueOf(i), x1 - 3*sizeAxisLabel, y, drawAxisLabel);
    }


    for (float i = 0; i < maxX; i = (float) (i + 60.0*60.0/4)) {
      some_data = (float) (maxY*(0.5*Math.cos(2*Math.PI/(6*60*60) * i + randomOffset * 100) + 0.5)); // A sine wave
transformed from 0 to 1 with a period of 6 hours

      // Transform time-power pairs into screen space
      x = graphTransformX(i, widthGraph, maxX, axisPaddingX);
      y = graphTransformY(some_data, maxY, heightGraph, axisPaddingY);

      if (i == 0) {
        graphPath.moveTo(x, y);
      } else {
        graphPath.lineTo(x, y);
      }
    }


    drawAxisLabel.setTextSize(sizeAxisLabel);

    canvas.drawText("Time (h)", (widthView - axisPaddingX) - 3 * sizeAxisLabel, (float) ((heightView -
2*axisPaddingY) + 5*sizeAxisLabel), drawAxisLabel);
    canvas.drawText("Power (W)", axisPaddingX, axisPaddingY, drawAxisLabel);

    xAxisLine.moveTo(0, heightView - axisPaddingY);
    xAxisLine.lineTo(widthView - axisPaddingX, heightView - axisPaddingY);
    yAxisLine.moveTo(axisPaddingX, heightView);
    yAxisLine.lineTo(axisPaddingX, axisPaddingY);
```

```
        canvas.drawPath(xAxisLine, drawAxes);
        canvas.drawPath(yAxisLine, drawAxes);
        canvas.drawPath(graphPath, drawPaint);
        canvas.drawPath(xGrid, drawGridMajor);
        canvas.drawPath(yGrid, drawGridMajor);

    }


}
```

MainActivity.java

```
package com.example.ThdrEglsACSwitch;


import android.os.AsyncTask;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Random;


public class MainActivity extends AppCompatActivity {
    public static final String EXTRA_MESSAGE = "com.example.ThdrEglsACSwitch.MESSAGE";
    private static final String TAG = "MainActivity";
    FragmentPagerAdapter adapterViewPager;
    boolean channel1State = false;
    boolean channel2State = false;
    Random rgen = new Random();
    int nChannels = 2;
    int nChannelParams = 3;
    // parameters: titleID, statusID, buttonID
    Object[][] powerChannelDefinitions = {
            {R.id.ch1_title, R.id.ch1_status, R.id.ch1_button},
            {R.id.ch2_title, R.id.ch2_status, R.id.ch2_button},
```

```
};
int nTimes = 2;
int nTimParams = 6;
int[][] timChannelDefinitions = {
      {R.id.tim1_off, R.id.tim1_15min, R.id.tim1_30min, R.id.tim1_60min, R.id.tim1_state, R.id.tim1_time},
      {R.id.tim2_off, R.id.tim2_15min, R.id.tim2_30min, R.id.tim2_60min, R.id.tim2_state, R.id.tim2_time},
};
String deviceURL = "http://192.168.4.1:80";


public void updateGraph (View v) {
   View graph1View = findViewById(R.id.simpleDrawingView1);
   View graph2View = findViewById(R.id.simpleDrawingView2);
   graph1View.invalidate();
   graph2View.invalidate();
}

public void updateChannelData(int channelID, int channelParam, String newText) {
   TextView v = (TextView) findViewById((int) powerChannelDefinitions[channelID][channelParam]);
   v.setText(newText);
}

public void updateTim1State(View v) {
   int tim_status_box_id = timChannelDefinitions[0][4];
   TextView tim_status_box  = findViewById(tim_status_box_id);
   String new_status_text = "You shouldn't see this";
   Http_GET_Information command = new Http_GET_Information();
   //Http_POST_Information command = new Http_POST_Information();
   int timerValue = 0;


   // off
   if (v.getId() == timChannelDefinitions[0][0]) {
      //command.url = (deviceURL + "/tim1_off");
      new_status_text = "off";

   }
   else if (v.getId() == timChannelDefinitions[0][1]) {
      //command.url = (deviceURL + "/tim1_15min");
      new_status_text = "15 minutes";
      timerValue = 15;
   }
   else if (v.getId() == timChannelDefinitions[0][2]) {
      //command.url = (deviceURL + "/tim1_30min");
      new_status_text = "30 minutes";
      timerValue = 30;
   }
   else if (v.getId() == timChannelDefinitions[0][3]) {
      //command.url = (deviceURL + "/tim1_60min");
      new_status_text = "60 minutes";
      timerValue = 60;
   }
   command.url = (deviceURL + "/timer1set");
   //command.dataPOST = String.valueOf(timerValue);
```

```
    new GetUrlContentTask().execute(command);
    //new HttpPOSTData().execute(command);
    tim_status_box.setText(new_status_text);
}

public void updateTim2State(View v) {
    int tim_status_box_id = timChannelDefinitions[1][4];
    TextView tim_status_box  = findViewById(tim_status_box_id);
    String new_status_text = "You shouldn't see this";
    Http_GET_Information command = new Http_GET_Information();
    //Http_POST_Information command = new Http_POST_Information();
    int timerValue = 0;


    // off
    if (v.getId() == timChannelDefinitions[1][0]) {
        //command.url = (deviceURL + "/tim1_off");
        new_status_text = "off";

    }
    else if (v.getId() == timChannelDefinitions[1][1]) {
        //command.url = (deviceURL + "/tim1_15min");
        new_status_text = "15 minutes";
        timerValue = 15;
    }
    else if (v.getId() == timChannelDefinitions[1][2]) {
        //command.url = (deviceURL + "/tim1_30min");
        new_status_text = "30 minutes";
        timerValue = 30;
    }
    else if (v.getId() == timChannelDefinitions[1][3]) {
        //command.url = (deviceURL + "/tim1_60min");
        new_status_text = "60 minutes";
        timerValue = 60;
    }
    command.url = (deviceURL + "/timer2set");
    //command.dataPOST = String.valueOf(timerValue);

    new GetUrlContentTask().execute(command);
    //new HttpPOSTData().execute(command);
    tim_status_box.setText(new_status_text);
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ViewPager vpPager = (ViewPager) findViewById(R.id.main_ui_pager);
    adapterViewPager = new MyPagerAdapter(getSupportFragmentManager());
    vpPager.setAdapter(adapterViewPager);
}
```

```
/*public void sendMessage(View view) {
    // FIXME: do something here
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.editText);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}*/

private class Http_GET_Information {
    String url;
    int view_id = -1;
    String data_to_send = "This means HTTP didn't work";
    String suffix = "";
    String prefix = "";
}

private class Http_POST_Information {
    String url;
    String dataPOST;
    // Maybe some kind of timestamp?
}

Http_GET_Information httpgeti = new Http_GET_Information();


private class GetUrlContentTask extends AsyncTask<Http_GET_Information, Integer, Http_GET_Information> {
    public static final String TAG = "GetUrlContentTask";
    protected Http_GET_Information doInBackground(Http_GET_Information... inputURL) {
        try {
            URL url = new URL(inputURL[0].url);
            Log.v(TAG, "We have URL " + url + " Trying to open connection.");
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            Log.v(TAG, "Connection open. Trying to connect.");
            connection.setRequestMethod("GET");
            connection.setDoInput(true);
            connection.setConnectTimeout(5000);
            connection.setReadTimeout(5000);
            connection.connect();
            Log.v(TAG, "Connected.");
            int http_code = connection.getResponseCode();
            Log.v(TAG, "got HTTP response code " + http_code);
            InputStreamReader inr = new InputStreamReader(connection.getInputStream());
            BufferedReader rd = new BufferedReader(inr);
            Log.v(TAG, "Reading input lines.");
            String content = "", line;
            while ((line = rd.readLine()) != null) {
                content += line + "\n";
            }
            inputURL[0].data_to_send = content;

            httpgeti.url = inputURL[0].url;
            httpgeti.data_to_send = inputURL[0].data_to_send;
            httpgeti.view_id = inputURL[0].view_id;
```

```
            httpgeti.prefix = inputURL[0].prefix;
            httpgeti.suffix = inputURL[0].suffix;


            return httpgeti;
        }
        catch (Exception e) {
            Log.v(TAG, "an exception happened: " + e.getMessage() + " " + e.getCause() + " " + e.getClass() );
            return null;
        }
    }

    protected void onProgressUpdate(Integer... progress) {}

    protected void onPostExecute (Http_GET_Information result) {
        result = httpgeti; // FIXME: Oh god why
        Log.v(TAG, "from the http server: " + result.data_to_send);
        int view_id_to_change = result.view_id;

        if (view_id_to_change >= 0) {
            TextView tmp_view = findViewById(result.view_id);
            tmp_view.setText(result.prefix + result.data_to_send + result.suffix);
        }
        //updateChannelData(0, 1, result);
    }
}

private class HttpPOSTData extends AsyncTask<Http_POST_Information, Integer, String> {
    public static final String TAG = "HttpPostData";
    protected String doInBackground(Http_POST_Information... params) {
        try {
            URL url = new URL(params[0].url);
            String dataPOST = params[0].dataPOST;


            HttpURLConnection connection = (HttpURLConnection) url.openConnection();



            //connection.setRequestMethod("POST");
            connection.setDoOutput(true);
            connection.setChunkedStreamingMode(0);
            connection.setRequestProperty("Accept-Encoding", "identity");
            //connection.setConnectTimeout(5000);
            //connection.setReadTimeout(5000);
            connection.connect();

            //int http_code = connection.getResponseCode();

            OutputStream out = new BufferedOutputStream(connection.getOutputStream());
            OutputStreamWriter outr = new OutputStreamWriter(out);
            //InputStreamReader inw = new InputStreamReader(connection.getInputStream());
            //BufferedWriter wr = new BufferedWriter(outr);
```

```
            Log.v(TAG, "Writing string " + dataPOST + " to http server");
            outr.write(dataPOST, 0, dataPOST.length());
            outr.close();
            Log.v(TAG, "Finished sending data to server.");

            InputStreamReader inr = new InputStreamReader(connection.getInputStream());
            BufferedReader rd = new BufferedReader(inr);
            Log.v(TAG, "Reading input lines during POST request.");
            String content = "", line;
            while ((line = rd.readLine()) != null) {
               content += line + "\n";
            }
            inr.close();
            Log.v(TAG, "Recieved string \"" + content + "\"from POST request");

            connection.disconnect();

            //return content;
            return "";
         }
         catch (Exception e) {
            Log.v(TAG, "an expection happened: " + e.getMessage() + " " + e.getCause() + " " + e.getClass() );
         }

         return "This function doesn't actually return anything";
      }

      protected void onProgressUpdate(Integer... progress) {}

      protected void onPostExecute (String result) {
         //FIXME: Do some kind of verification / UI update here
      }
   }

   public void updateChannel1State(View v) {
      String TAG = "updateChanel1State";
      Log.v(TAG, String.valueOf(v.getId()));
      int view_id_current = R.id.ch1_status;
      String str_from_http = "Oops, the code didn't work";

      Http_GET_Information command_toggle = new Http_GET_Information();
      Http_GET_Information command_current = new Http_GET_Information();


      command_current.url = deviceURL + "/ch1_current";
//       command_current.view_id = view_id_current;
//       command_current.prefix = "Current: ";
//       command_current.suffix = " A";


      if (channel1State) { // channel on
         command_toggle.url = deviceURL + "/outlet1off";
         updateChannelData(0, 1, "Channel off");
```

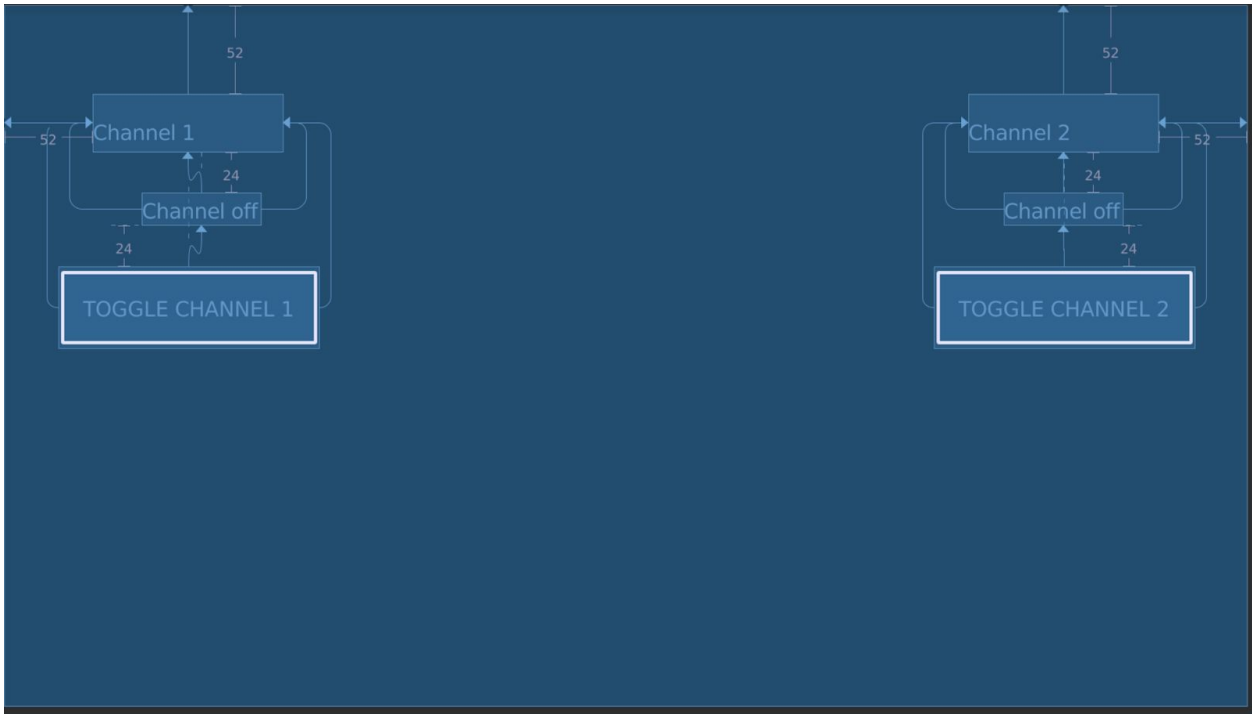
```
        }
        else { //channel off
            command_toggle.url = deviceURL + "/outlet1on";
            updateChannelData(0, 1, "Channel on");
        }

        new GetUrlContentTask().execute(command_toggle);
        //new GetUrlContentTask().execute(command_current);

        channel1State = !channel1State;
    }

    public void updateChannel2State(View v) {
        String TAG = "updateChanel1State";
        Log.v(TAG, String.valueOf(v.getId()));
        int view_id_current = R.id.ch2_status;
        String str_from_http = "Oops, the code didn't work";

        Http_GET_Information command_toggle = new Http_GET_Information();
        Http_GET_Information command_current = new Http_GET_Information();


        command_current.url = deviceURL + "/ch1_current";
//      command_current.view_id = view_id_current;
//      command_current.prefix = "Current: ";
//      command_current.suffix = " A";


        if (channel2State) { // channel on
            command_toggle.url = deviceURL + "/outlet2off";
            updateChannelData(1, 1, "Channel off");
        }
        else { //channel off
            command_toggle.url = deviceURL + "/outlet2on";
            updateChannelData(1, 1, "Channel on");
        }

        new GetUrlContentTask().execute(command_toggle);
        //new GetUrlContentTask().execute(command_current);

        channel2State = !channel2State;
    }

//  public void updateChannel2State(View v) {
//      String newText;
//
//      TextView current = (TextView) findViewById(R.id.ch2_status);
//      if (channel2State) {
//          newText = "Current: " + rgen.nextInt(4) + "A";
//      }
//      else {
//          newText = "Channel off";
//      }
//
```

```
//      current.setText(newText);
//      channel2State = !channel2State;
//
//      Http_POST_Information poststuff = new Http_POST_Information();
//      poststuff.url = deviceURL + "/echo";
//      poststuff.dataPOST = newText;
//
//      new HttpPOSTData().execute(poststuff);
//  }

  public static class MyPagerAdapter extends FragmentPagerAdapter {
    private static int NUM_ITEMS = 3;

    public MyPagerAdapter(FragmentManager fragmentManager) {
      super(fragmentManager);
    }


    @Override
    public int getCount() {
      return NUM_ITEMS;
    }

    @Override
    public Fragment getItem(int position) {
      switch (position) {
        case 0:
          return FragmentMainPowerUI.newInstance(0, "some title");
        case 1:
          return FragmentTimerUI.newInstance(1, "some other title");
        case 2:
          return FragmentGraphUI.newInstance(2, "do these titles actually do anything?");
        default:
          return null;
      }
    }

    @Override
    public CharSequence getPageTitle(int position) {
      return "Page " + position;
    }
  }
}
```

Fragment_main_power_ui.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
    android:id="@+id/ch2_title"
    style="@style/ChannelTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="52dp"
    android:layout_marginEnd="52dp"
    android:layout_marginRight="52dp"
    android:text="Channel 2"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

  <Button
    android:id="@+id/ch1_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:onClick="updateChannel1State"
    android:text="Toggle channel 1"
    app:layout_constraintEnd_toEndOf="@+id/ch1_title"
    app:layout_constraintStart_toStartOf="@+id/ch1_title"
    app:layout_constraintTop_toBottomOf="@+id/ch1_status" />

  <TextView
    android:id="@+id/ch1_title"
    style="@style/ChannelTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="52dp"
    android:layout_marginLeft="52dp"
    android:layout_marginTop="52dp"
    android:text="Channel 1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

  <TextView
    android:id="@+id/ch1_status"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="24dp"
    android:text="Channel off"
```

```
    app:layout_constraintEnd_toEndOf="@+id/ch1_title"
    app:layout_constraintStart_toStartOf="@+id/ch1_title"
    app:layout_constraintTop_toBottomOf="@+id/ch1_title" />

  <TextView
    android:id="@+id/ch2_status"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:text="Channel off"
    app:layout_constraintEnd_toEndOf="@+id/ch2_title"
    app:layout_constraintStart_toStartOf="@+id/ch2_title"
    app:layout_constraintTop_toBottomOf="@+id/ch2_title" />

  <Button
    android:id="@+id/ch2_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:onClick="updateChannel2State"
    android:text="Toggle Channel 2"
    app:layout_constraintEnd_toEndOf="@+id/ch2_title"
    app:layout_constraintStart_toStartOf="@+id/ch2_title"
    app:layout_constraintTop_toBottomOf="@+id/ch2_status" />

</android.support.constraint.ConstraintLayout>
```



Bill of Materials

Part Name	Part Number	Quantity	Description	Datasheet	Cost/1	Cost/10	Cost/100
G6DN-1A Slim Power Relay for 5A switching	CR1, CR2	2	4.5VDC supply, 250 VAC, 5A max, 24.4mA coil current	Datasheet	$1.78	$16.65	$133.20
IRM-03-3.3S AC to DC pwr supply	PS1	1	85-305VAC Input, 3.3VDC, 900mA Output, 68% efficiency, 100mV ripple	Datasheet	$7.10	$67.20	$630.00
W57-XB7A4A10-5 5A circuit breaker	CB1	1	5A, 240 VAC reset only, push button	Datasheet	$2.11	$19.60	$174.00
ESP32 Microcontrollers	U4	1	ESP32-DevKitC (ESP-WROOM-32) Rev1 Development Board, WiFi, Bluetooth, Ultra-Low Power Consumption, Dual Core	Datasheet	$10.99	$109.90	$1,099.00
ACS722LLCTR-05AB-T	U1, U2	2	Current Sensor 5A 1 Channel Hall Effect, Open Loop Bidirectional 8-SOIC (0.154", 3.90mm Width)	Datasheet	$5.54	$43.16	$301.28
2N7000 MOSFET	Q1, Q2	2	Small Signal MOSFET 200mA, 60V	Datasheet	$0.47	$3.70	$25.57
MCP1826S-2502E/AB-ND	U3	1	2.5Vout, 1A out, fixed voltage regulator 0.4V drop	Datasheet	$1.01	$10.10	$76.22
645-558-0301-001F Yellow snap in LED	L1, L2	2	2.1V drop, 10mA, 3mm LED, 4mm mounting hole	Datasheet	$1.26	$7.88	$63.00
645-558-0101-001F Red snap in LED	L3, L4	2	2V drop, 10mA, 3mm LED, 4mm mounting hole	Datasheet	$1.26	$7.88	$63.00
796636-2 power wire connector, PCB	J1, J2, J3, J4	2	2X1 connections, shrouded on 4 sides, 15A max, 5.08 pitch, male header pins, vertical connection	Datasheet	$0.95	$5.91	$44.49
284041-2 power connector, wire	J1, J2, J3, J4	2	2X1 connections, terminal block into male pin header, horizontal wire entry, vertical pin connect, 5.08 pitch, 12-30 AWG wires	Datasheet	$1.78	$13.46	$109.45
530470310 LED connector, PCB	J5, J6	2	3X1 connections, male pin header, 1.25 pitch, shrouded on 4 sides, vertical connection	Datasheet	$0.24	$2.27	$15.73
510210300 LED connector, wire	J5, J6	2	3X1 connections, female pin housing, 1.25mm pitch,	Datasheet	$0.22	$2.09	$14.51
3723-F		4	2-56 x 1/2" stainless steel flat head Phillips machine screw		$0.33	$3.30	$33.00
3731-E		2	2-56 x 3/8" stainless steel flat head Phillips machine screw		$0.30	$3.00	$30.00
3733-G		6	2-56 nut		$0.40	$4.00	$40.00
				Total cost for batch	$52.53	$417.12	$3,524.00
				Total cost/device	$52.53	$45.60	$39.52