**ensino lúdico da física usando a eletrônica**

**Geração Eólica (documentação para docentes)**

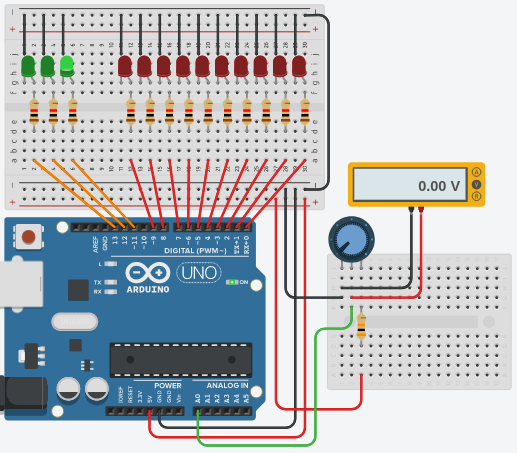
1. **Simulação**

Link para acesso à simulação online do projeto:

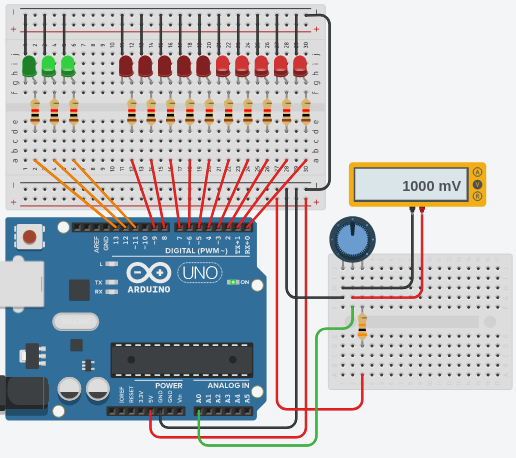
<https://www.tinkercad.com/things/645YoLXwXta-copy-of-turbina/editel?sharecode=6CCoQWjvi1xjBCJG0TjRc8_t75oPBdFEUQtCkK-A4XE>

O potenciômetro serve, nesse arquivo, para simular a tensão gerada pela hélice. Girando o potenciômetro para o sentido horário, o valor da tensão aumenta e, com isso, mais LEDs acendem.

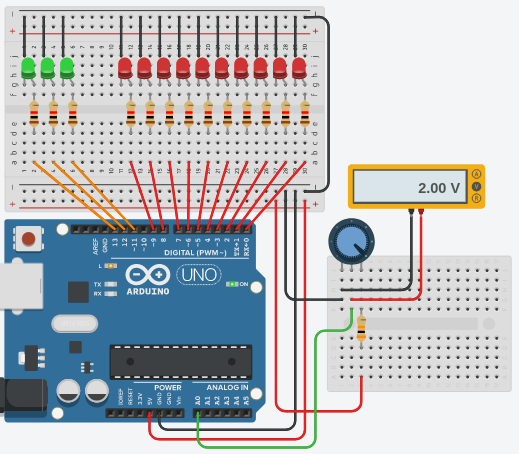
Resultado esperado com o valor mínimo do potenciômetro:



Resultado esperado com valor médio do potenciômetro:

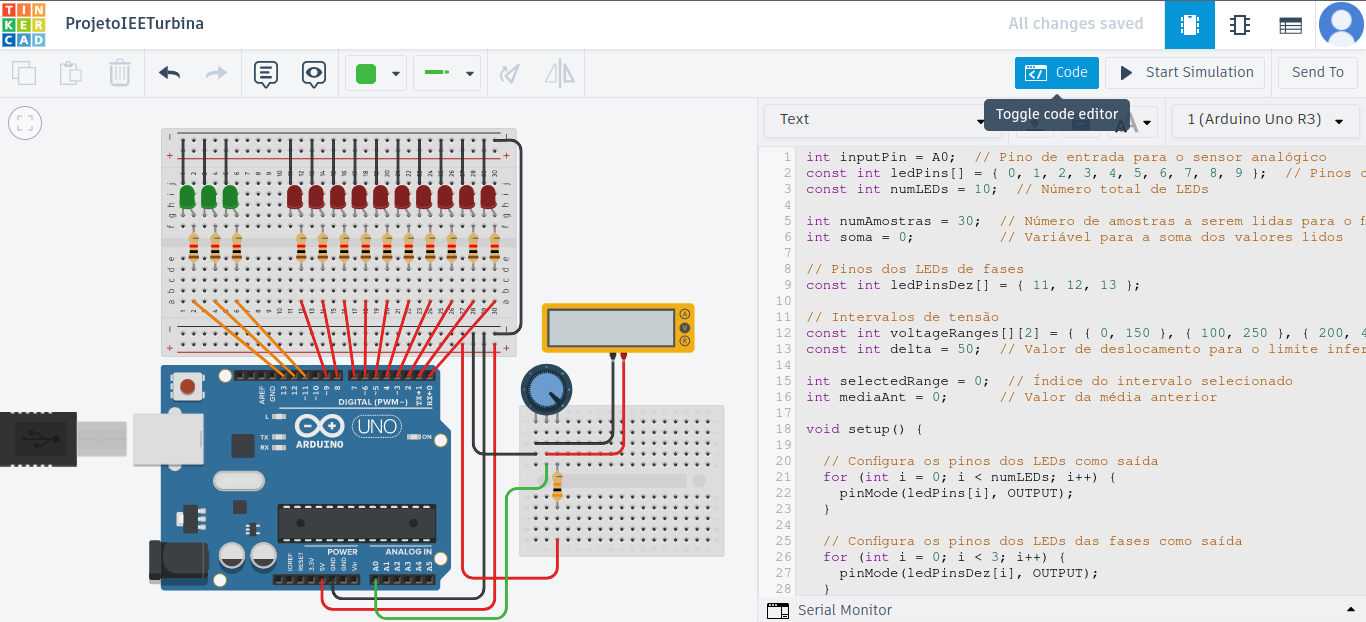


Resultado esperado com valor máximo do potenciômetro:



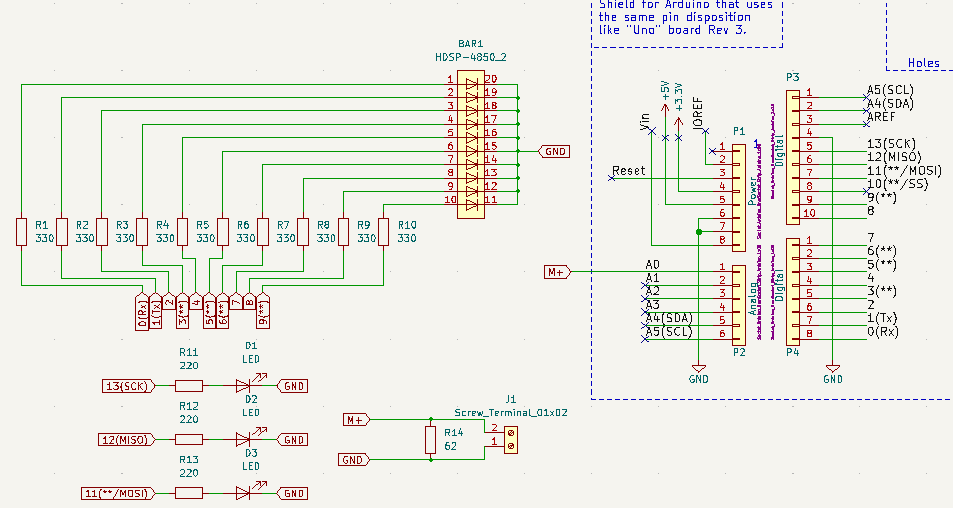
Os três LEDs verdes agrupados à esquerda são os marcadores das fases. Os LEDs vermelhos, agrupados à direita, são os indicadores de nível dentro de cada fase (simulam o *bargraph* da placa).

Na parte direita do simulador é possível visualizar o código do projeto com a explicação comentada.



1. **Esquemático**

Após a conclusão da simulação, foram realizados testes do circuito em *protoboard* e em placa de circuito impresso ilhada. Como verificou-se o funcionamento também nesses suportes, partiu-se para o projeto de uma placa de circuito impresso definitiva. O esquemático e o leiaute da placa foram elaborados através do *software* KiCad, levando-se em consideração o circuito apresentado no link de simulação mencionado anteriormente. Com base nisso, o esquemático resultou da seguinte forma:

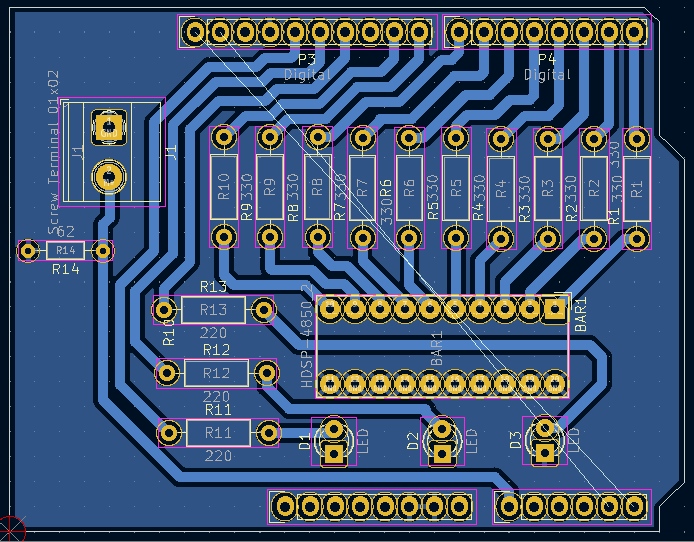
****

No esquemático, podemos observar a inclusão de um modelo pré-definido do Arduino Uno, fornecido pelo próprio fabricante. Ele é representado no canto superior direito, dentro de um retângulo pontilhado. Esse modelo inclui todos os pinos do Arduino e garante que o *shield*, uma placa que se conecta ao microcontrolador, esteja em conformidade com o padrão utilizado pelo Arduino.

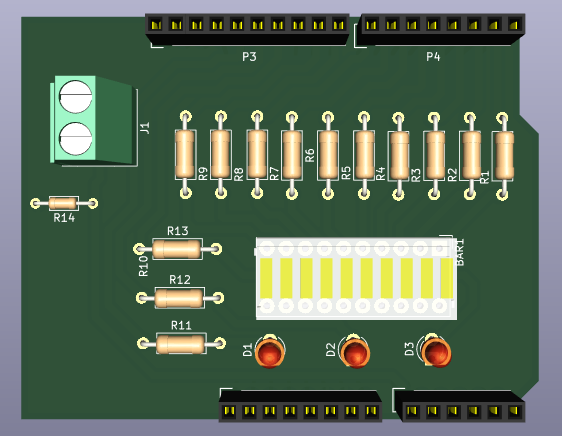
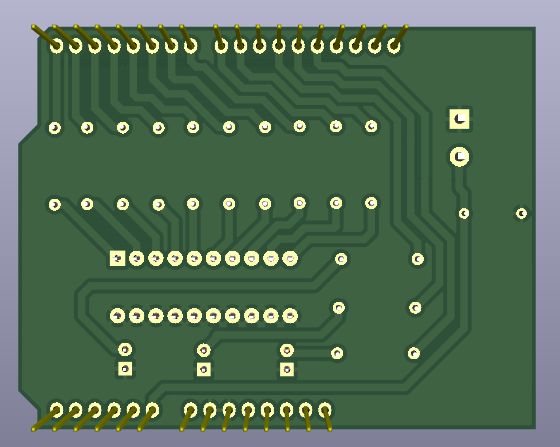
Os demais elementos do esquemático foram criados e montados com base na simulação e nos testes realizados em *protoboard* e em placa de circuito impresso ilhada. Todos os componentes usados neste esquema, e consequentemente na placa foram:

* 1 Bargraph (barra de LEDs);
* 10 Resistores de 1k5 Ω;
* 3 Resistores de 220 Ω;
* 1 Resistor de 62 Ω;
* 1 Conector KRE;
* 3 LEDs verdes;
* Barras de pinos fêmea (uma de 10 pinos, duas de 8 pinos e uma de 6 pinos).

Após a elaboração do esquemático, o leiaute dos componentes e trilhas foi organizado para a fabricação da placa de circuito impresso. Após a conclusão desse processo, a placa ficou com o seguinte aspecto:

****

A partir dessa representação da placa, o software oferece uma visualização em 3D da mesma, conforme mostrado abaixo, sendo a primeira imagem referente a parte superior da placa e a segunda da parte inferior.

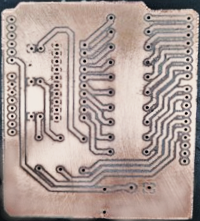
** **

1. **Fabricação**

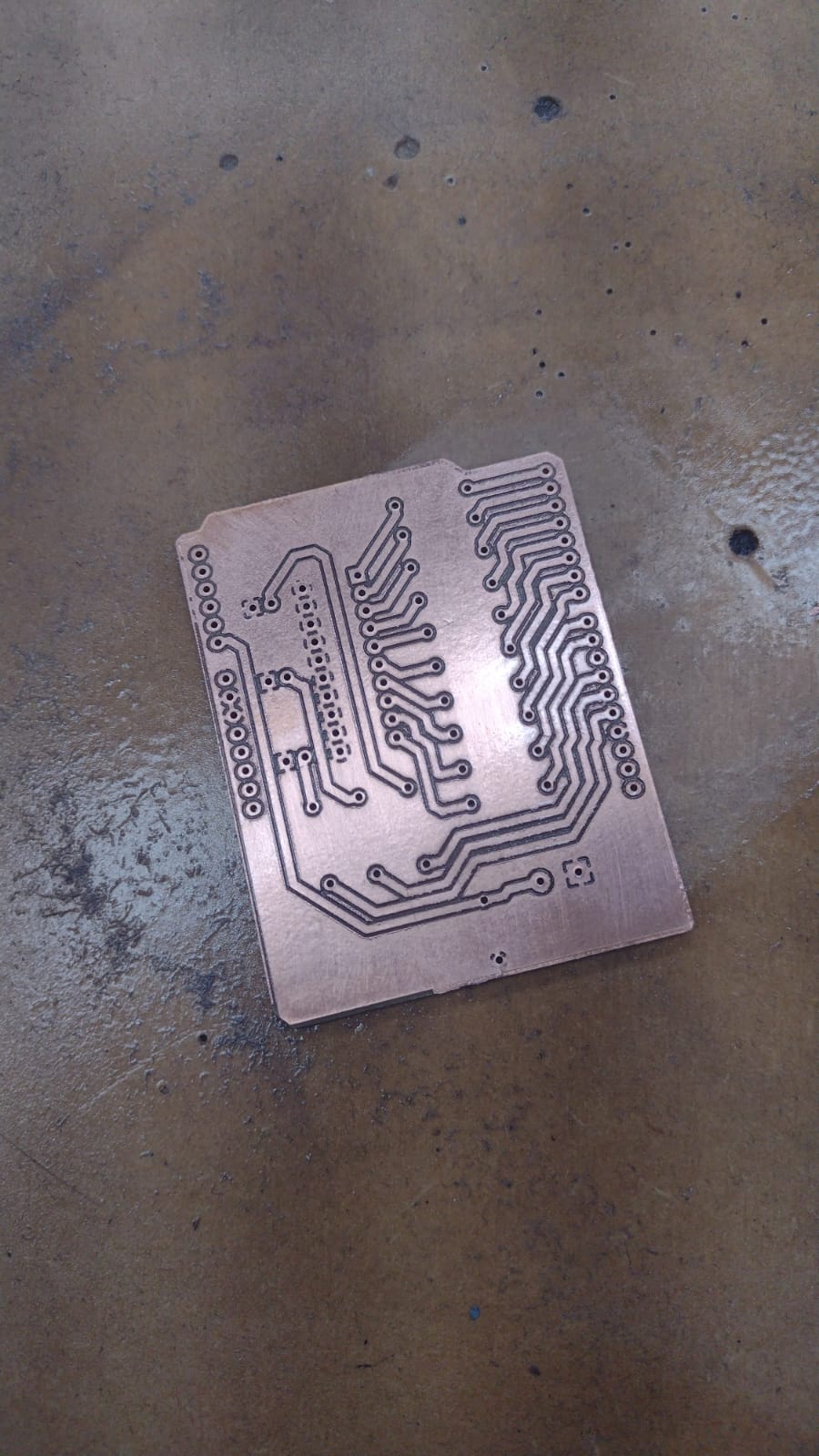
A fabricação foi feita a partir de uma máquina fresadora CNC, construída no IFSC. A máquina foi utilizada para usinar, furar e cortar a placa.

Para realizar esse processo, se fez uso dos arquivos Gerber de trilhas e de furação disponíveis como resultado do projeto no KiCad. Foi então utilizado o *software* FlatCAM para criar o código necessário para a máquina, especificando as ferramentas necessárias para usinar, furar e cortar. Com esses arquivos, o software BCNC foi utilizado para controlar a máquina CNC. Um pequeno vídeo demonstrando o funcionamento da máquina fabricando a placa pode ser acessado através do seguinte [link](https://drive.google.com/file/d/1iGi9JybQoS5oYUsmaBZtkz6HqUjvkt6K/view?usp=sharing).

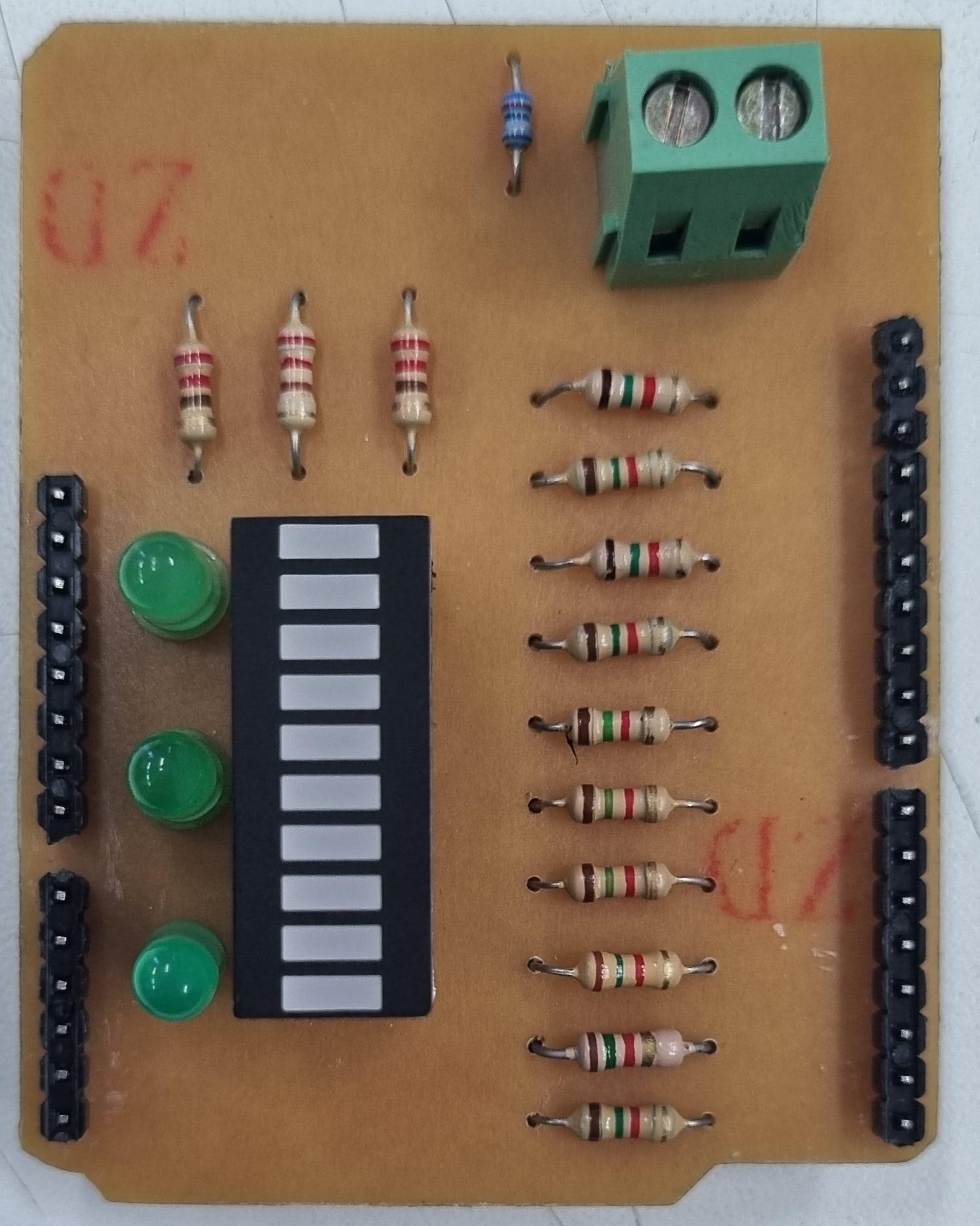
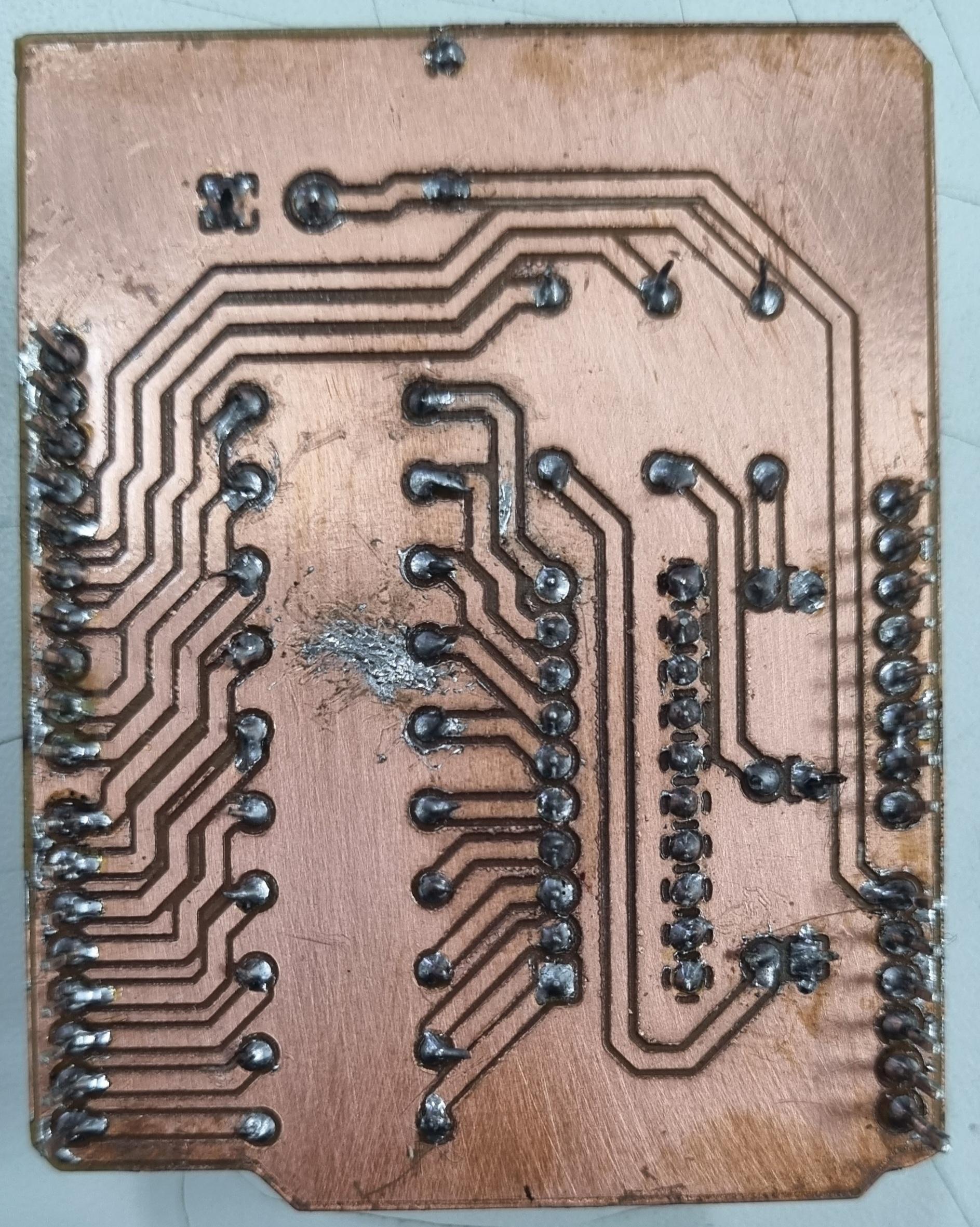
Dessa forma, a placa foi fabricada e, após sair da máquina, apresentava o seguinte aspecto:



Após a fabricação da placa, o próximo passo foi lixar as bordas da placa e, com a esponja de aço, polir o cobre da placa. Em seguida, foi aplicado o verniz para conservação da placa, como mostrado na figura abaixo:



A última etapa foi soldar os componentes e testar seu funcionamento. Com sucesso, a placa finalizada possui os seguintes aspectos, conforme as imagens abaixo:

**Apêndice A - Código do Projeto**

int inputPin = A0; // Pino de entrada para o sensor analógico

const int ledPins[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }; // Pinos dos LEDs

const int numLEDs = 10; // Número total de LEDs

int numAmostras = 30; // Número de amostras a serem lidas para o filtro de ruído

int soma = 0; // Variável para a soma dos valores lidos

// Pinos dos LEDs de fases

const int ledPinsDez[] = { 11, 12, 13 };

// Intervalos de tensão

const int voltageRanges[][2] = { { 0, 150 }, { 100, 250 }, { 200, 400 } }; // Matriz de intervalos de tensões para as fases

const int delta = 50; // Valor de deslocamento para o limite inferior do intervalo

int selectedRange = 0; // Índice do intervalo selecionado

int mediaAnt = 0; // Valor da média anterior

void setup() {

// Configura os pinos dos LEDs como saída

for (int i = 0; i < numLEDs; i++) {

pinMode(ledPins[i], OUTPUT);

}

// Configura os pinos dos LEDs das fases como saída

for (int i = 0; i < 3; i++) {

pinMode(ledPinsDez[i], OUTPUT);

}

}

void loop() { // Filtro de ruído

soma = 0; // Zera a variável de soma para a próxima leitura

// Lê várias amostras do sensor analógico e soma os valores

for (int i = 0; i < numAmostras; i++) {

int volume = analogRead(inputPin); // Lê o valor analógico do pino de entrada

soma += volume; // Adiciona o valor lido à variável de soma

delay(20); // Aguarda antes de ler a próxima amostra

}

int media = soma / numAmostras; // Calcula a média dos valores lidos

media = (media + mediaAnt)/2; // Suaviza a média com o valor anterior

// Determina a fase

if ((media >= voltageRanges[0][0]) && (media <= voltageRanges[0][1])) {

selectedRange = 0;

} else if ((media >= voltageRanges[1][0]) && (media <= voltageRanges[1][1])) {

selectedRange = 1;

} else if ((media >= voltageRanges[2][0]) && (media <= voltageRanges[2][1])) {

selectedRange = 2;

}

// Calcula os limites inferior e superior do intervalo selecionado

int lowerBound = voltageRanges[selectedRange][0];

if ( selectedRange > 0 ) {

lowerBound = lowerBound + delta;

}

int upperBound = voltageRanges[selectedRange][1];

// Mapeia a média para o brilho dos LEDs usando os limites do intervalo

int brightness = map(media, lowerBound, upperBound, 0, numLEDs);

// Ajusta o brilho para o caso de uma leitura alta

if (media > 400) {

brightness = 10;

}

// Controla os LEDs de acordo com o brilho calculado

for (int i = 0; i < numLEDs; i++) {

if (i < brightness) {

digitalWrite(ledPins[i], HIGH);

} else {

digitalWrite(ledPins[i], LOW);

}

}

// Controla os LEDs das fases de acordo com o intervalo selecionado

for (int i = 0; i < 3; i++) {

if (i <= selectedRange) {

digitalWrite(ledPinsDez[i], HIGH);

} else {

digitalWrite(ledPinsDez[i], LOW);

}

}

mediaAnt = media; // Armazena o valor da média atual para uso na próxima iteração

}