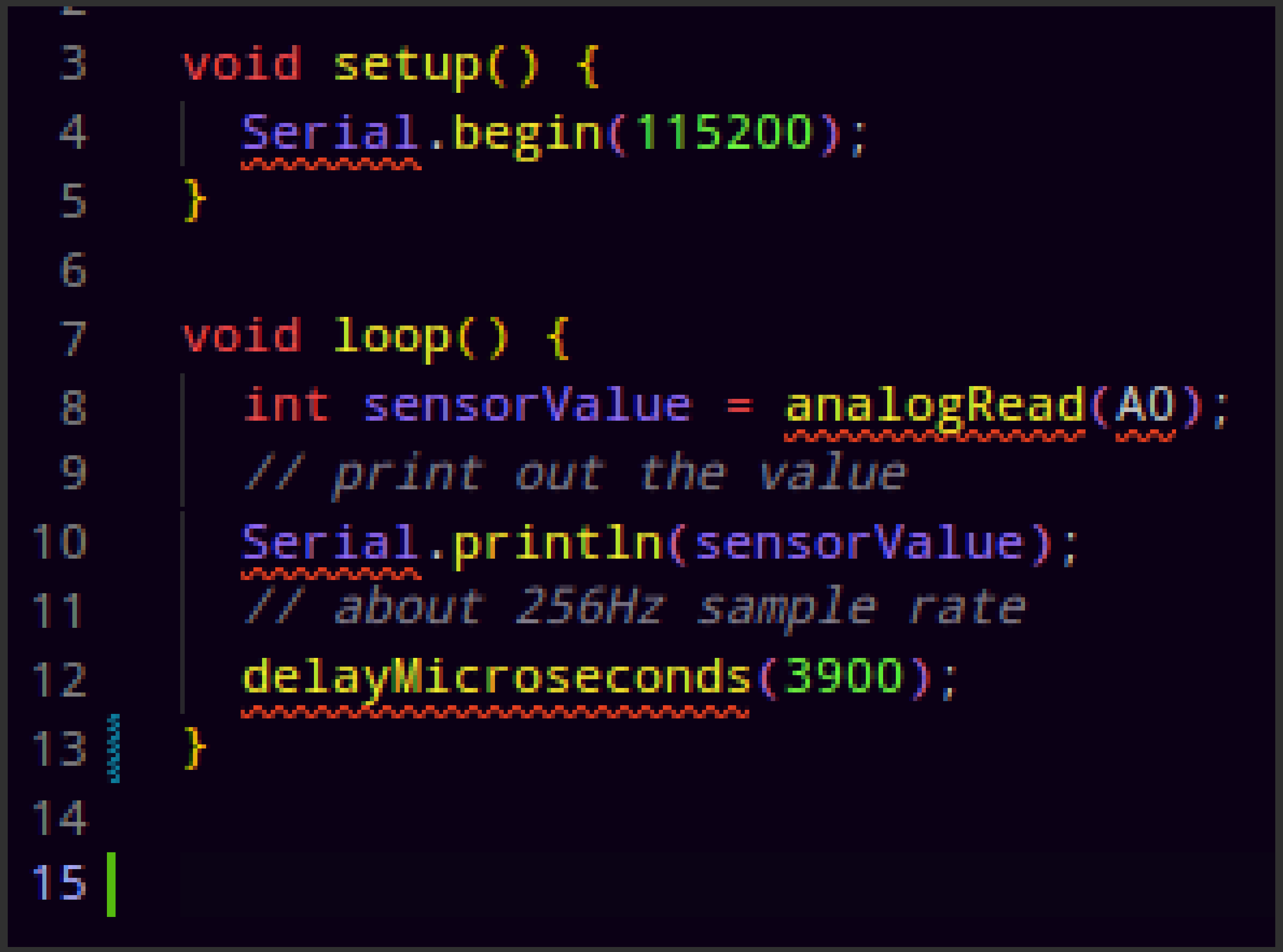
# Como funciona

1 Coleta do sinal

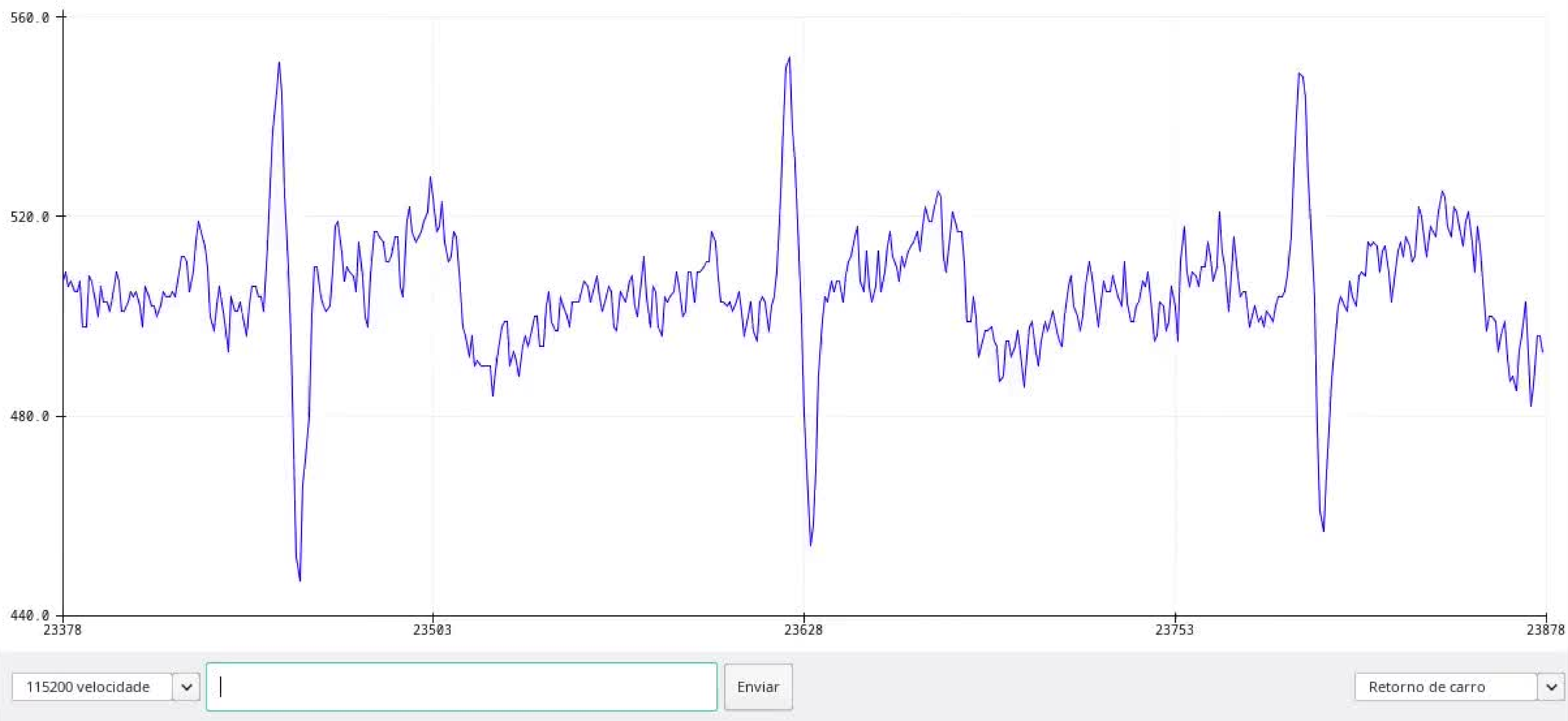
2 MODWT

3 Gráfico do sinal

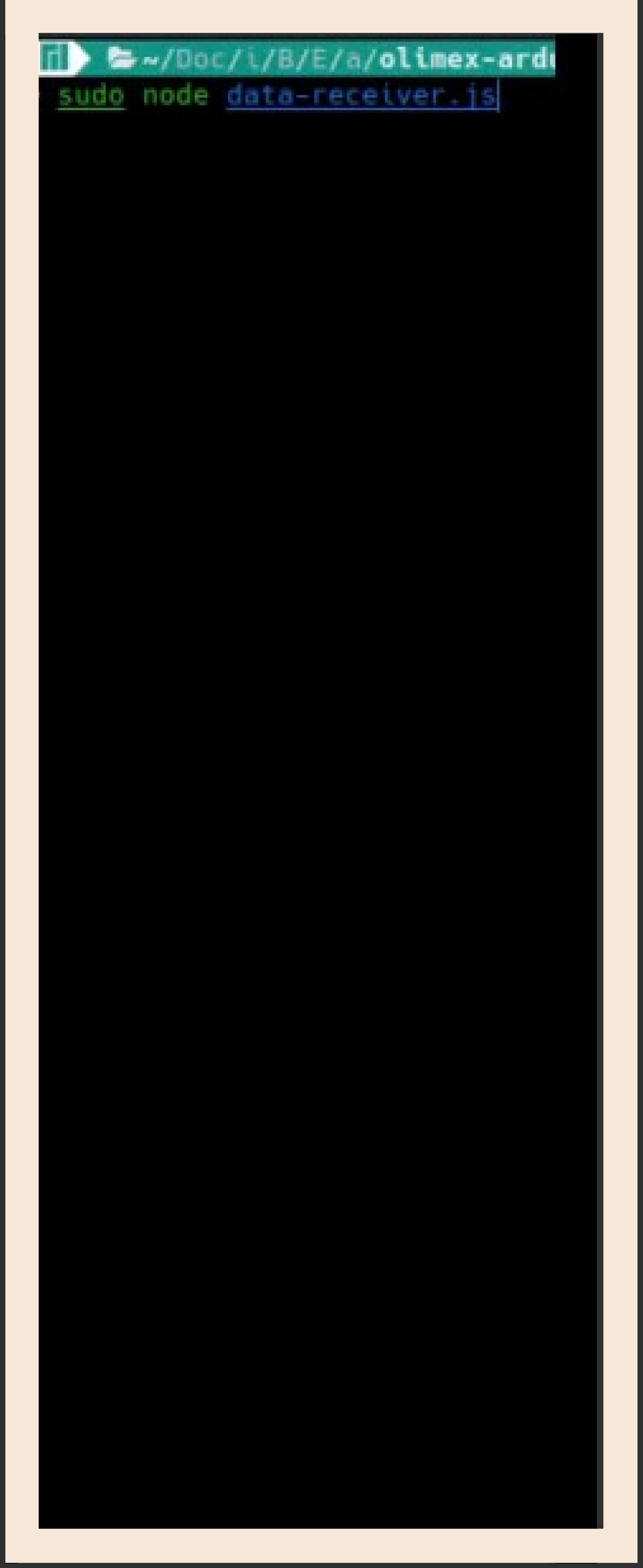
# Coleta dos dados



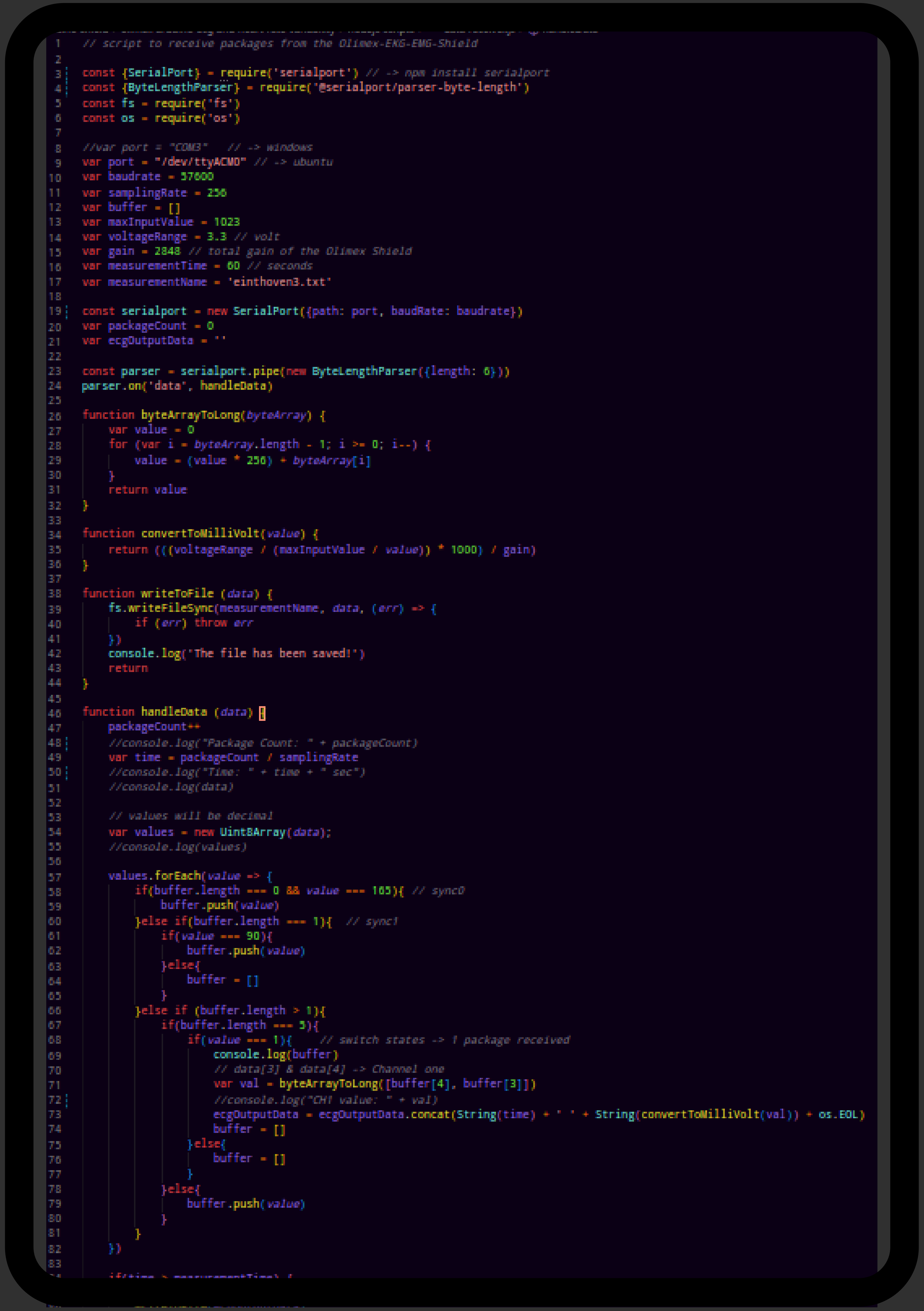
```
void setup() {
  Serial.begin(115200);
}

void loop() {
  int sensorValue = analogRead(A0);
  // print out the value
  Serial.println(sensorValue);
  // about 256Hz sample rate
  delayMicroseconds(3900);
}

```



# Coleta dos dados



```
~/Doc/i/B/E/a/olimex-ardu
sudo node data-receiver.js
```

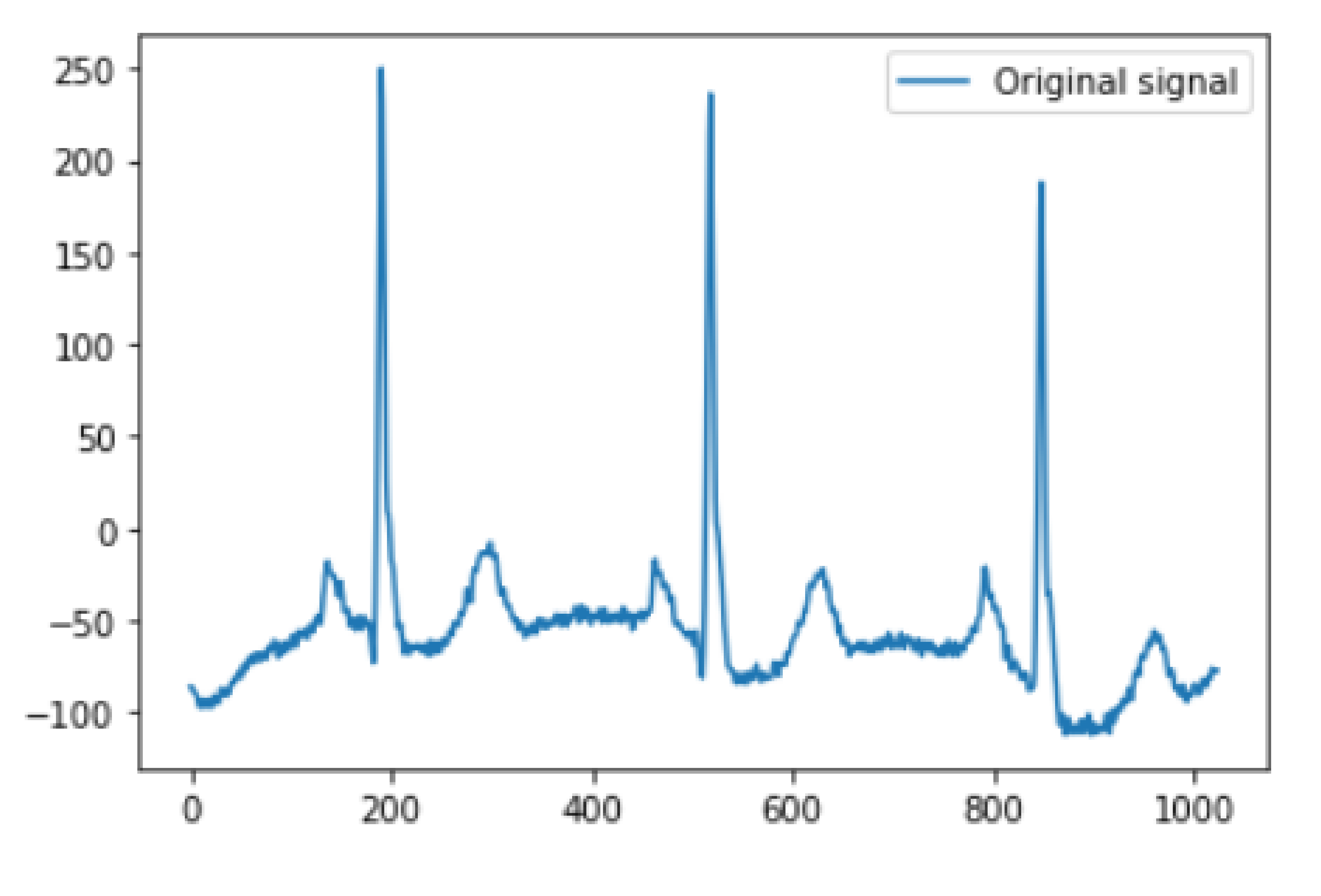


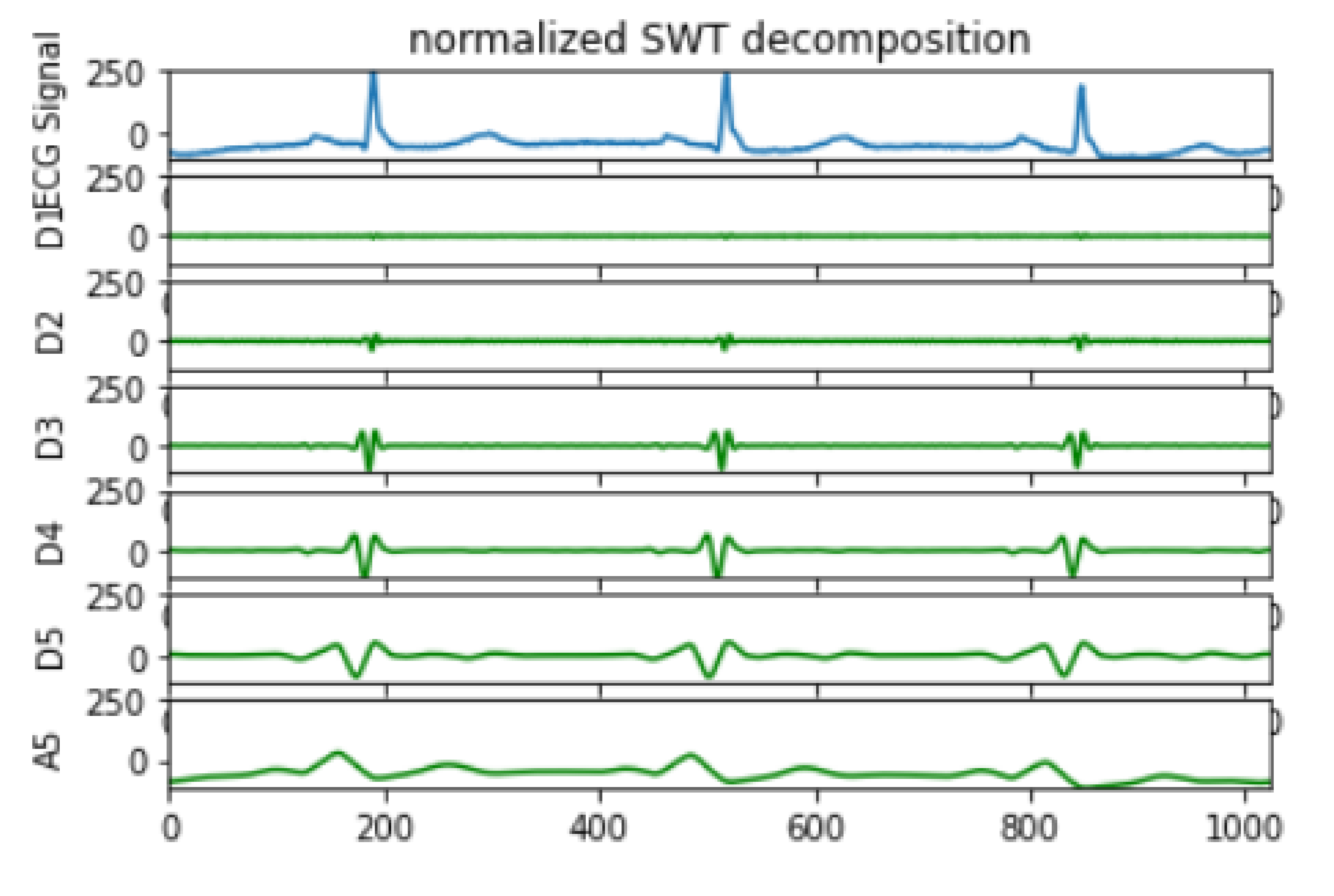
```javascript
// script to receive packages from the Olimex-EKG-EMG-Shield

const {SerialPort} = require('serialport') // -> npm install serialport
const {ByteLengthParser} = require('@serialport/parser-byte-length')
const fs = require('fs')
const os = require('os')

//var port = "COM3"   // -> windows
var port = "/dev/ttyACM0" // -> ubuntu
var baudrate = 57600
var samplingRate = 256
var buffer = []
var maxInputValue = 1023
var voltageRange = 3.3 // volt
var gain = 2848 // total gain of the Olimex Shield
var measurementTime = 60 // seconds
var measurementName = 'einthoven3.txt'

const serialport = new SerialPort({path: port, baudRate: baudrate})
var packageCount = 0
var ecgOutputData = ''

const parser = serialport.pipe(new ByteLengthParser({length: 6}))
parser.on('data', handleData)

function byteArrayToLong(byteArray) {
    var value = 0
    for (var i = byteArray.length - 1; i >= 0; i--) {
        value = (value * 256) + byteArray[i]
    }
    return value
}

function convertToMilliVolt(value) {
    return (((voltageRange / (maxInputValue / value)) * 1000) / gain)
}

function writeToFile (data) {
    fs.writeFileSync(measurementName, data, (err) => {
        if (err) throw err
    })
    console.log('The file has been saved!')
    return
}

function handleData (data) {
    packageCount++
    //console.log("Package Count: " + packageCount)
    var time = packageCount / samplingRate
    //console.log("Time: " + time + " sec")
    //console.log(data)

    // values will be decimal
    var values = new Uint8Array(data);
    //console.log(values)

    values.forEach(value => {
        if(buffer.length === 0 && value === 165){ // sync0
            buffer.push(value)
        }else if(buffer.length === 1){  // sync1
            if(value === 90){
                buffer.push(value)
            }else{
                buffer = []
            }
        }else if (buffer.length > 1){
            if(buffer.length === 5){
                if(value === 1){    // switch states -> 1 package received
                    console.log(buffer)
                    // data[3] & data[4] -> Channel one
                    var val = byteArrayToLong([buffer[4], buffer[3]])
                    //console.log("CH1 value: " + val)
                    ecgOutputData = ecgOutputData.concat(String(time) + ' ' + String(convertToMilliVolt(val)) + os.EOL)
                    buffer = []
                }else{
                    buffer = []
                }
            }else{
                buffer.push(value)
            }
        }
    })

    if(time > measurementTime) {
```

# MODWTI

## Referencial

**MODWT: Maximal overlap discrete wavelet transform**

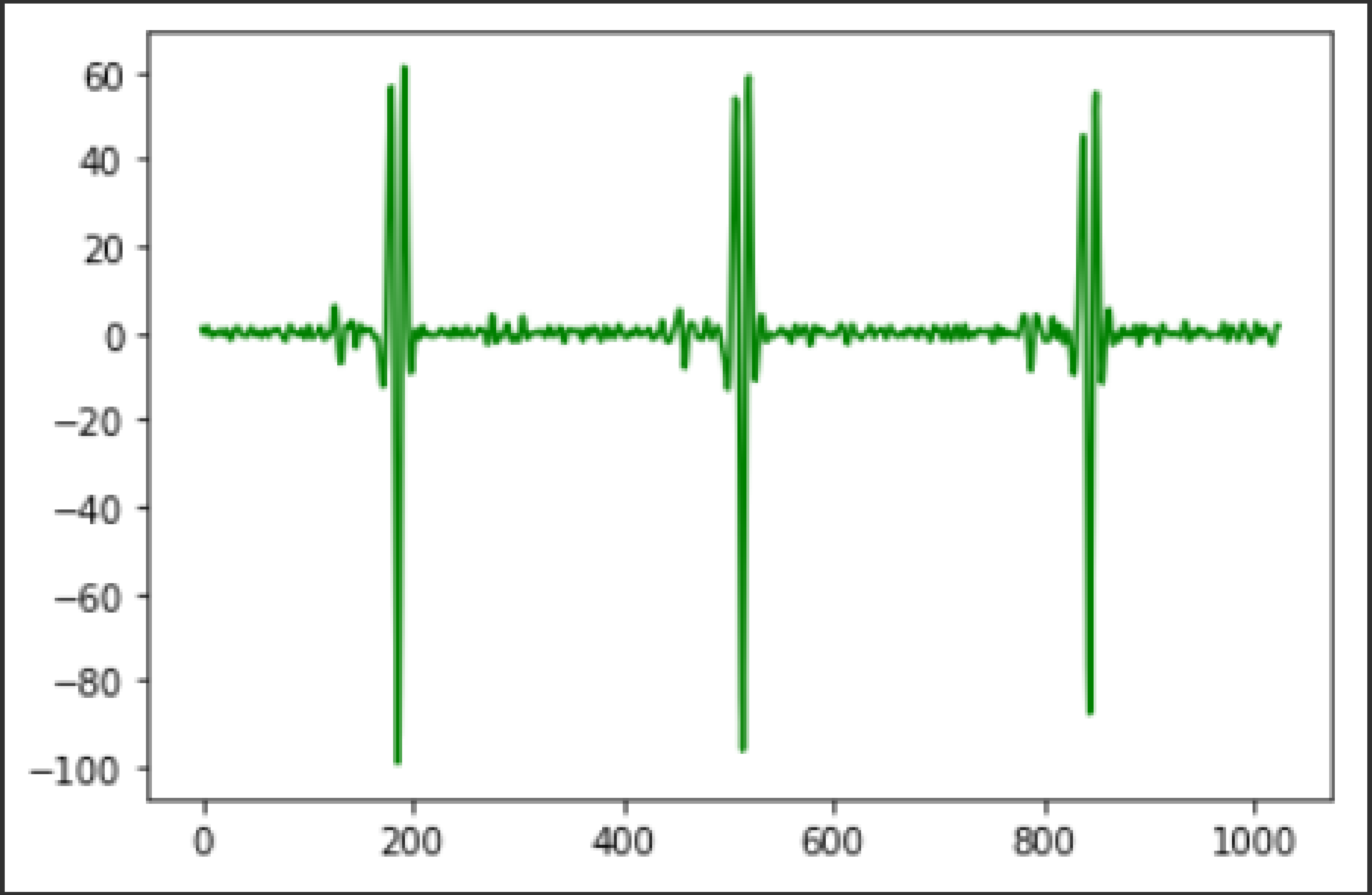




# MODWTI

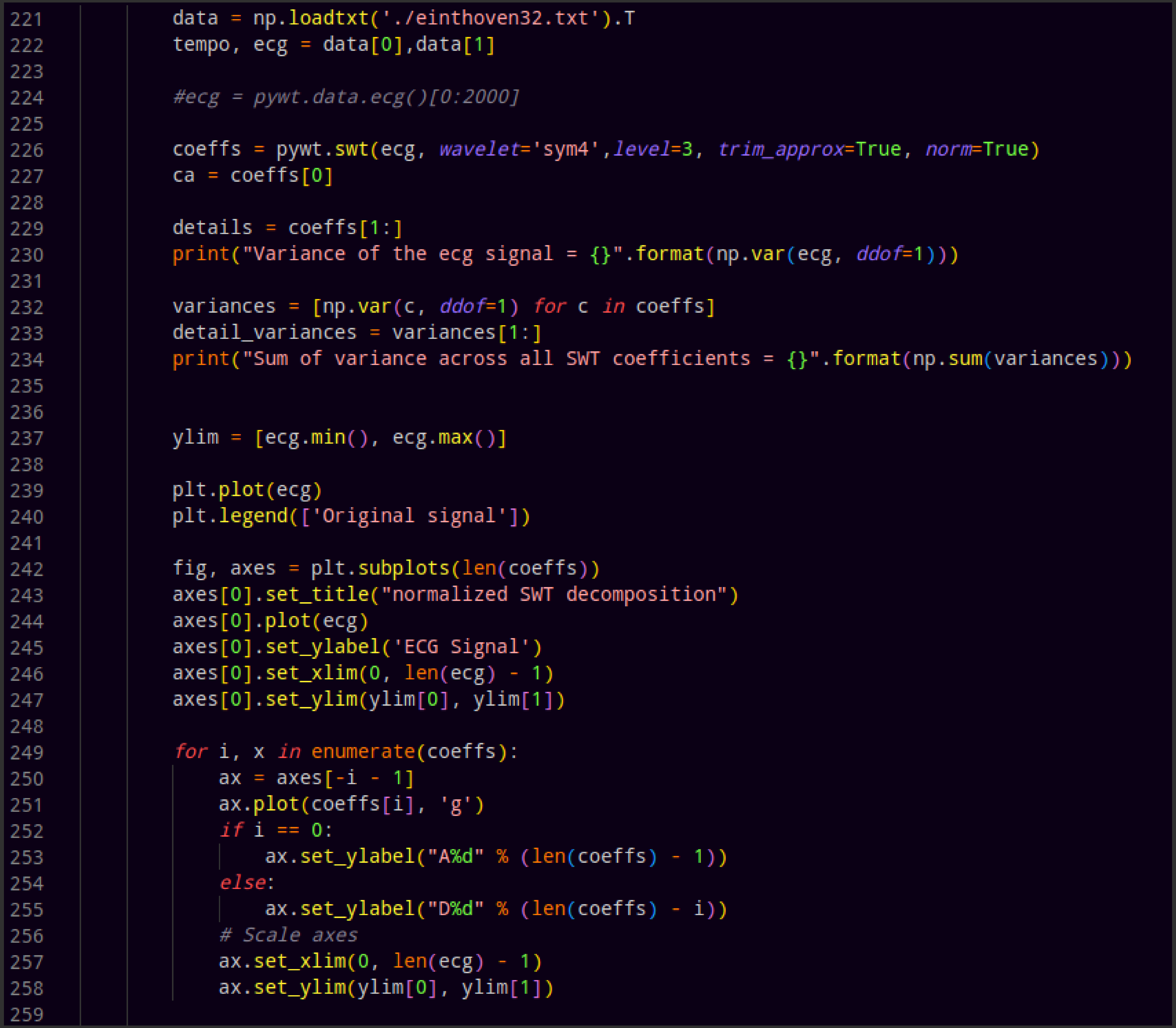
## MODWT: Maximal overlap discrete wavelet transform

Referencial



Função do Matlab: https://www.mathworks.com/help/wavelet/ref/modwt.html

# Gráfico dos resultados



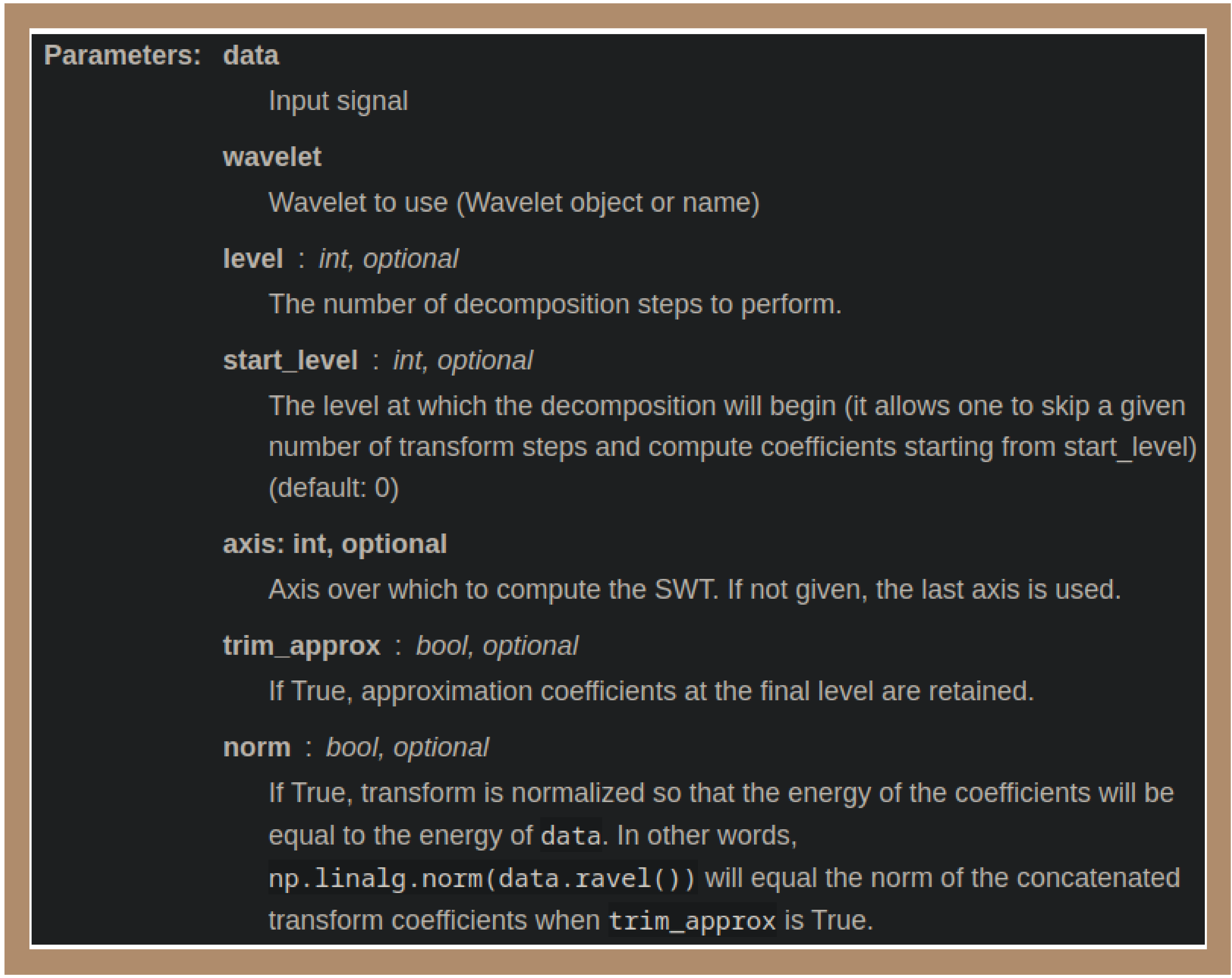
```
        data = np.loadtxt('./einthoven32.txt').T
        tempo, ecg = data[0],data[1]

        #ecg = pywt.data.ecg()[0:2000]

        coeffs = pywt.swt(ecg, wavelet='sym4',level=3, trim_approx=True, norm=True)
        ca = coeffs[0]

        details = coeffs[1:]
        print("Variance of the ecg signal = {}".format(np.var(ecg, ddof=1)))

        variances = [np.var(c, ddof=1) for c in coeffs]
        detail_variances = variances[1:]
        print("Sum of variance across all SWT coefficients = {}".format(np.sum(variances)))


        ylim = [ecg.min(), ecg.max()]

        plt.plot(ecg)
        plt.legend(['Original signal'])

        fig, axes = plt.subplots(len(coeffs))
        axes[0].set_title("normalized SWT decomposition")
        axes[0].plot(ecg)
        axes[0].set_ylabel('ECG Signal')
        axes[0].set_xlim(0, len(ecg) - 1)
        axes[0].set_ylim(ylim[0], ylim[1])

        for i, x in enumerate(coeffs):
            ax = axes[-i - 1]
            ax.plot(coeffs[i], 'g')
            if i == 0:
                ax.set_ylabel("A%d" % (len(coeffs) - 1))
            else:
                ax.set_ylabel("D%d" % (len(coeffs) - i))
            # Scale axes
            ax.set_xlim(0, len(ecg) - 1)
            ax.set_ylim(ylim[0], ylim[1])

```

# Gráfico dos resultados

## PyWavelets
## Wavelet Transforms in Python

**Estacionária Wavelet Transform (SWT) , também conhecida como Undecimated wavelet transform ou Algorithme à trous , é uma modificação de invariância de tradução da Transformada Wavelet Discreta que não dizima os coeficientes em todos os níveis de transformação.**

**Quando usado com norm=True, essa transformação está intimamente relacionada ao DWT de sobreposição múltipla (MODWT) popularizado para análise de séries temporais, embora a implementação subjacente seja ligeiramente diferente daquela publicada em [1] . Especificamente, a implementação usada aqui requer um sinal que seja múltiplo de 2**levelcomprimento.**



**Parameters:** **data**

Input signal

**wavelet**

Wavelet to use (Wavelet object or name)

**level** : *int, optional*

The number of decomposition steps to perform.

**start_level** : *int, optional*

The level at which the decomposition will begin (it allows one to skip a given number of transform steps and compute coefficients starting from start_level) (default: 0)

**axis: int, optional**

Axis over which to compute the SWT. If not given, the last axis is used.

**trim_approx** : *bool, optional*

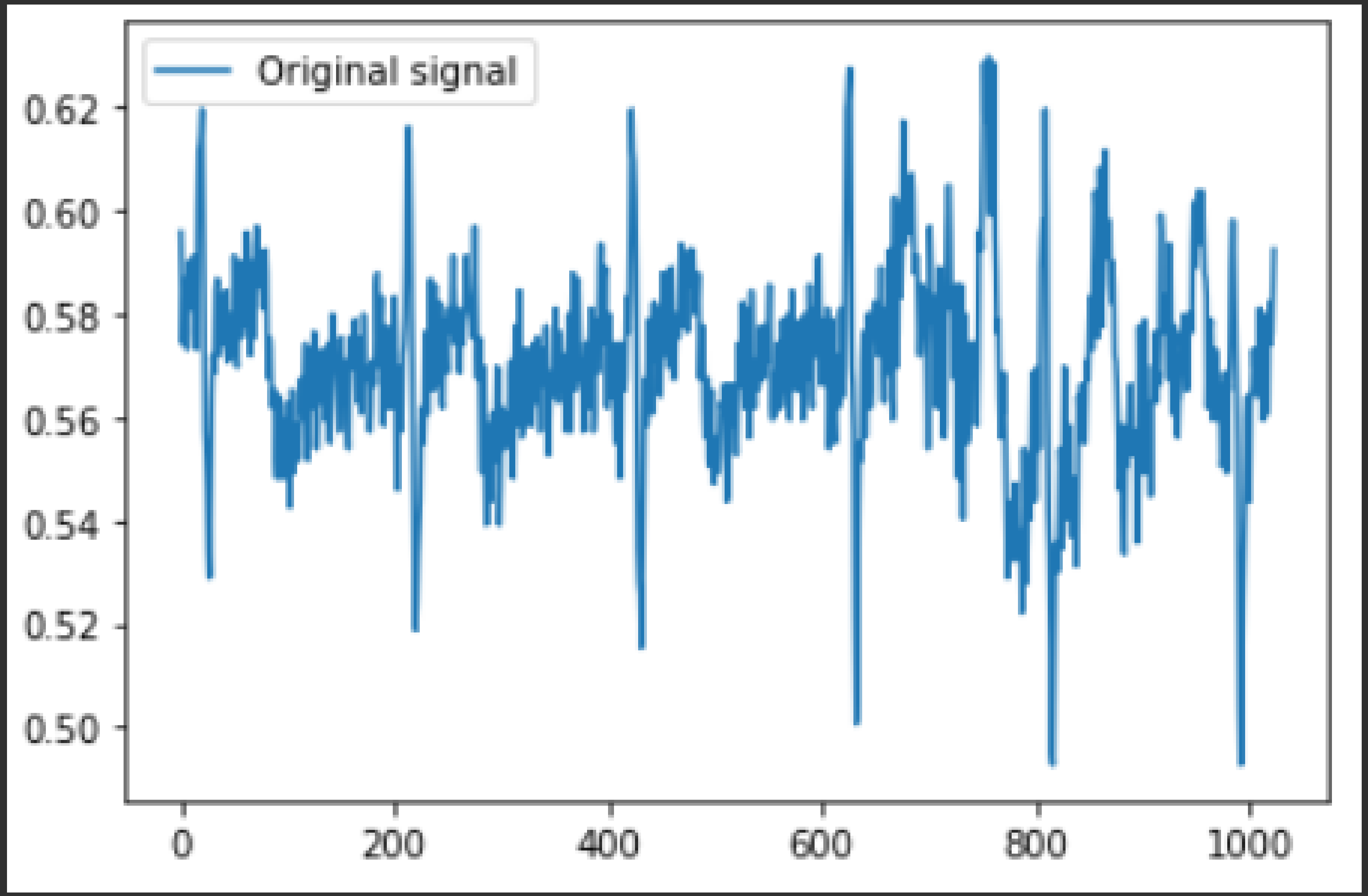
If True, approximation coefficients at the final level are retained.

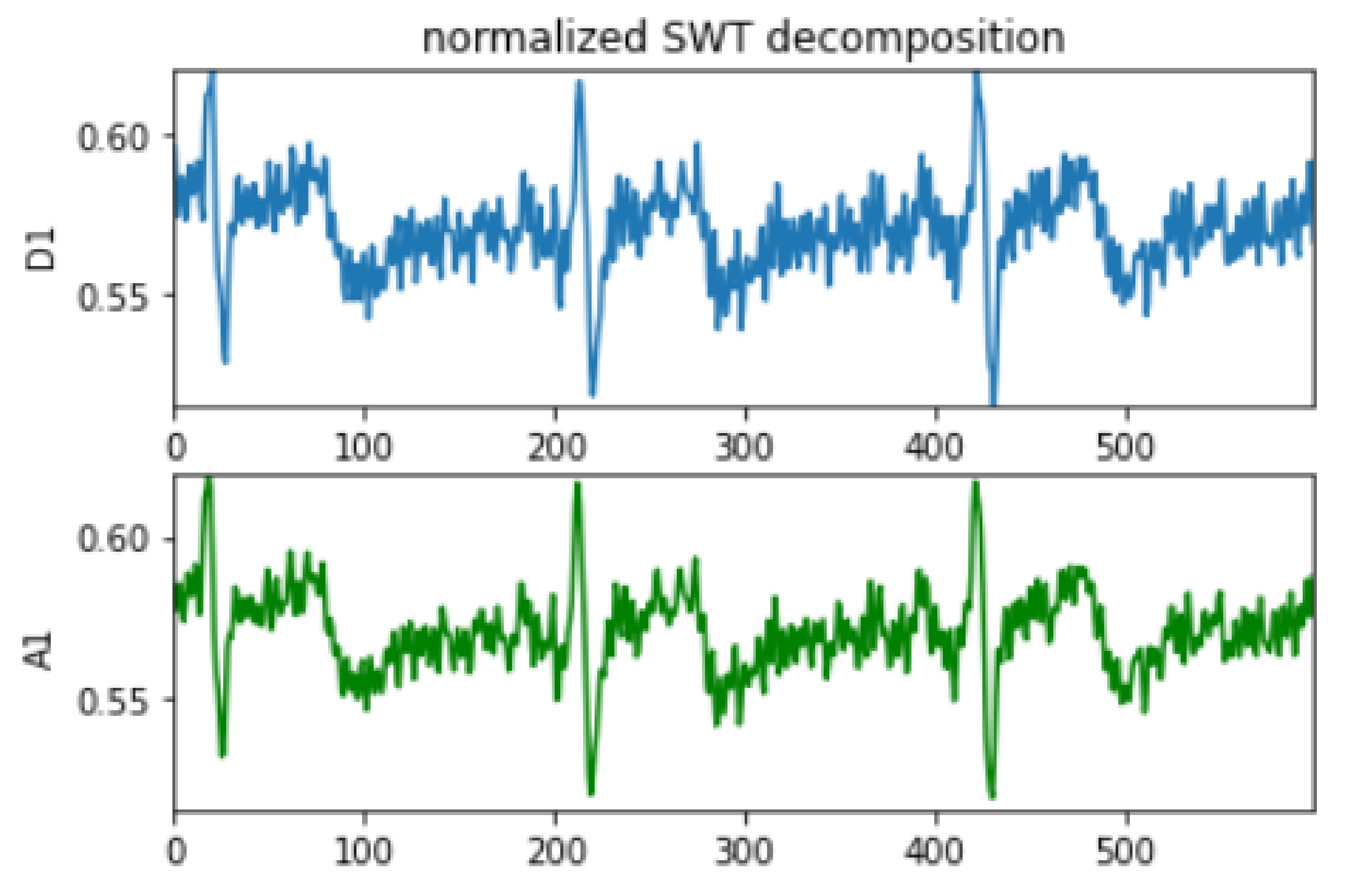
**norm** : *bool, optional*

If True, transform is normalized so that the energy of the coefficients will be equal to the energy of `data`. In other words, `np.linalg.norm(data.ravel())` will equal the norm of the concatenated transform coefficients when `trim_approx` is True.

# Gráfico dos resultados

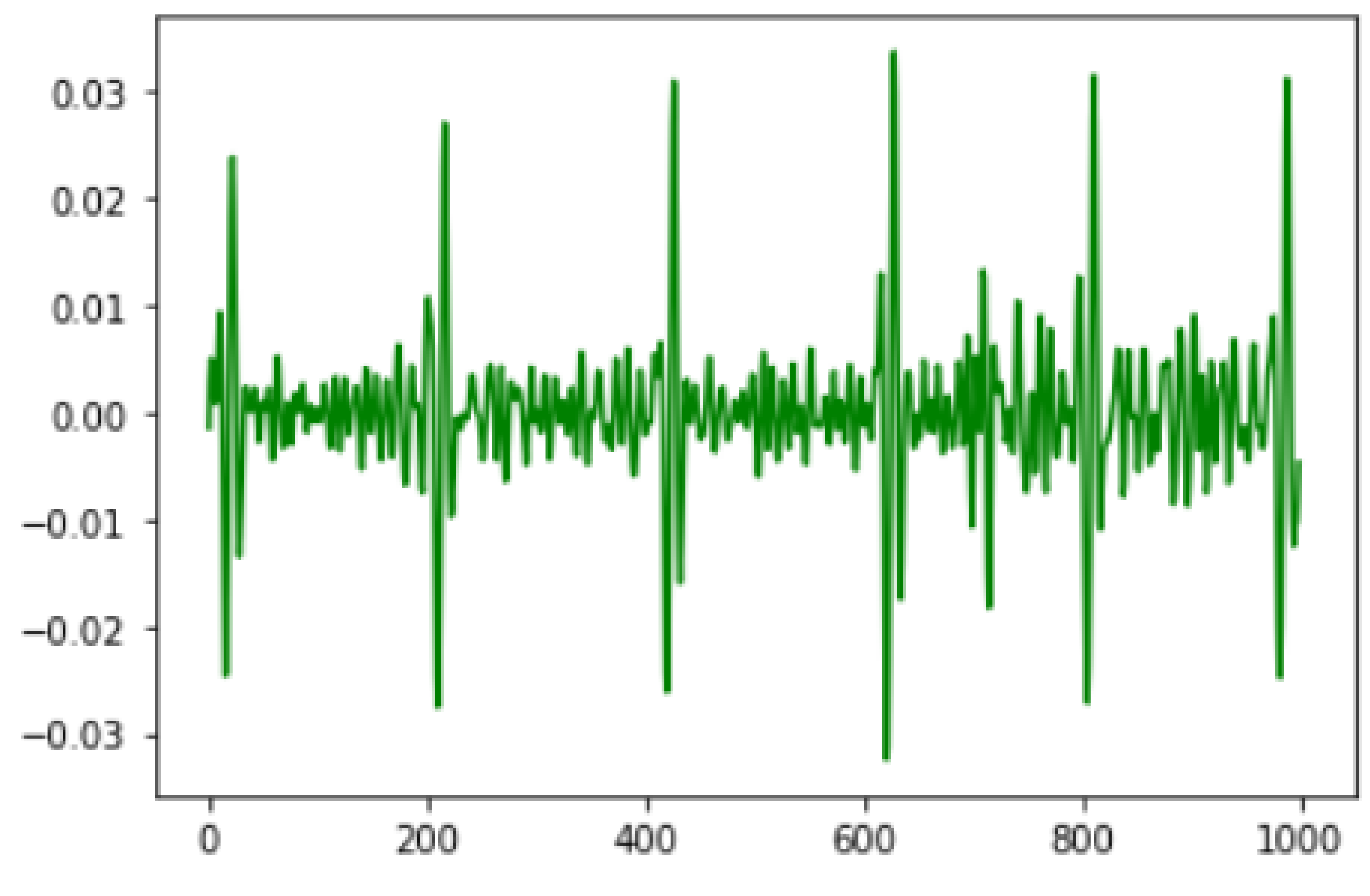
## MODWTl em level1

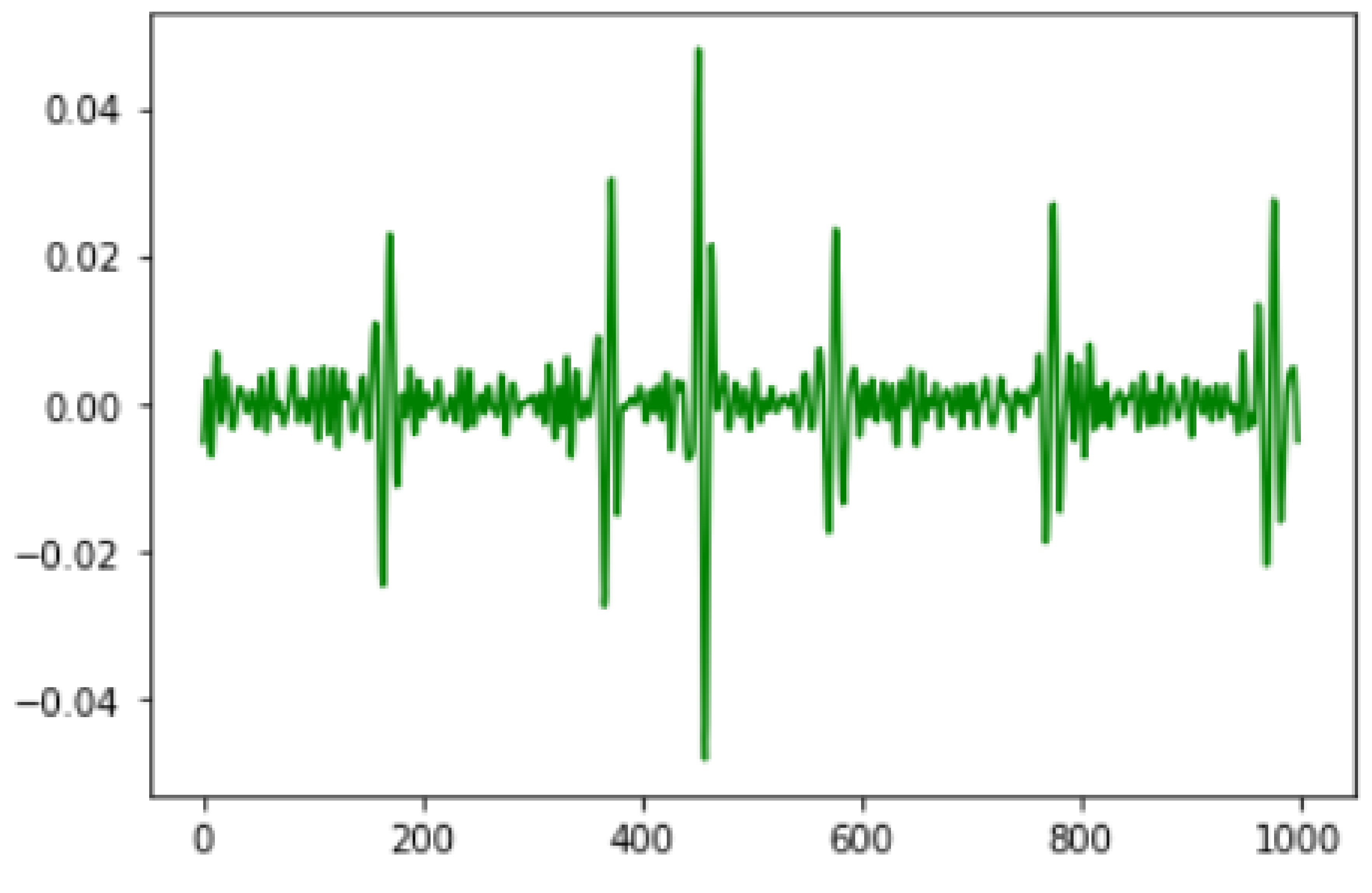




# Gráfico dos resultados

## MODWTI em level 3





**Obrigado pela atenção!**

**Duvidas ?**