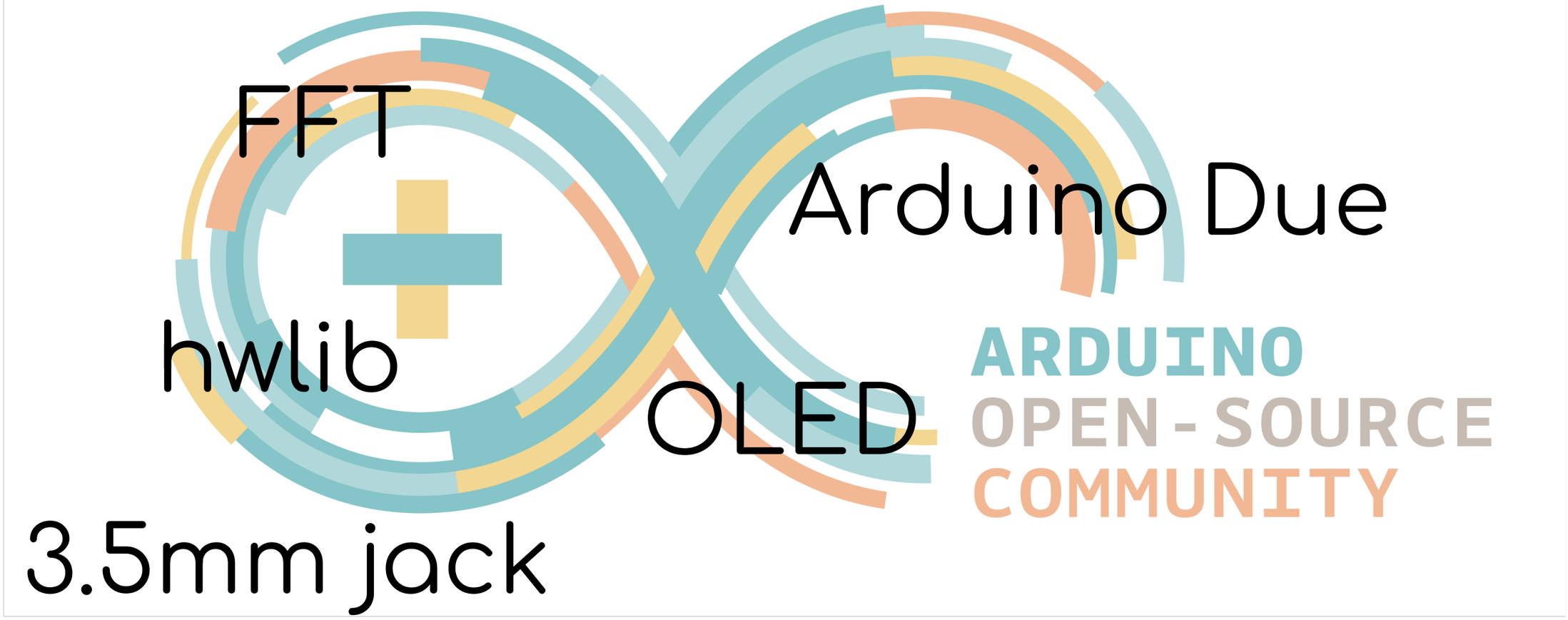
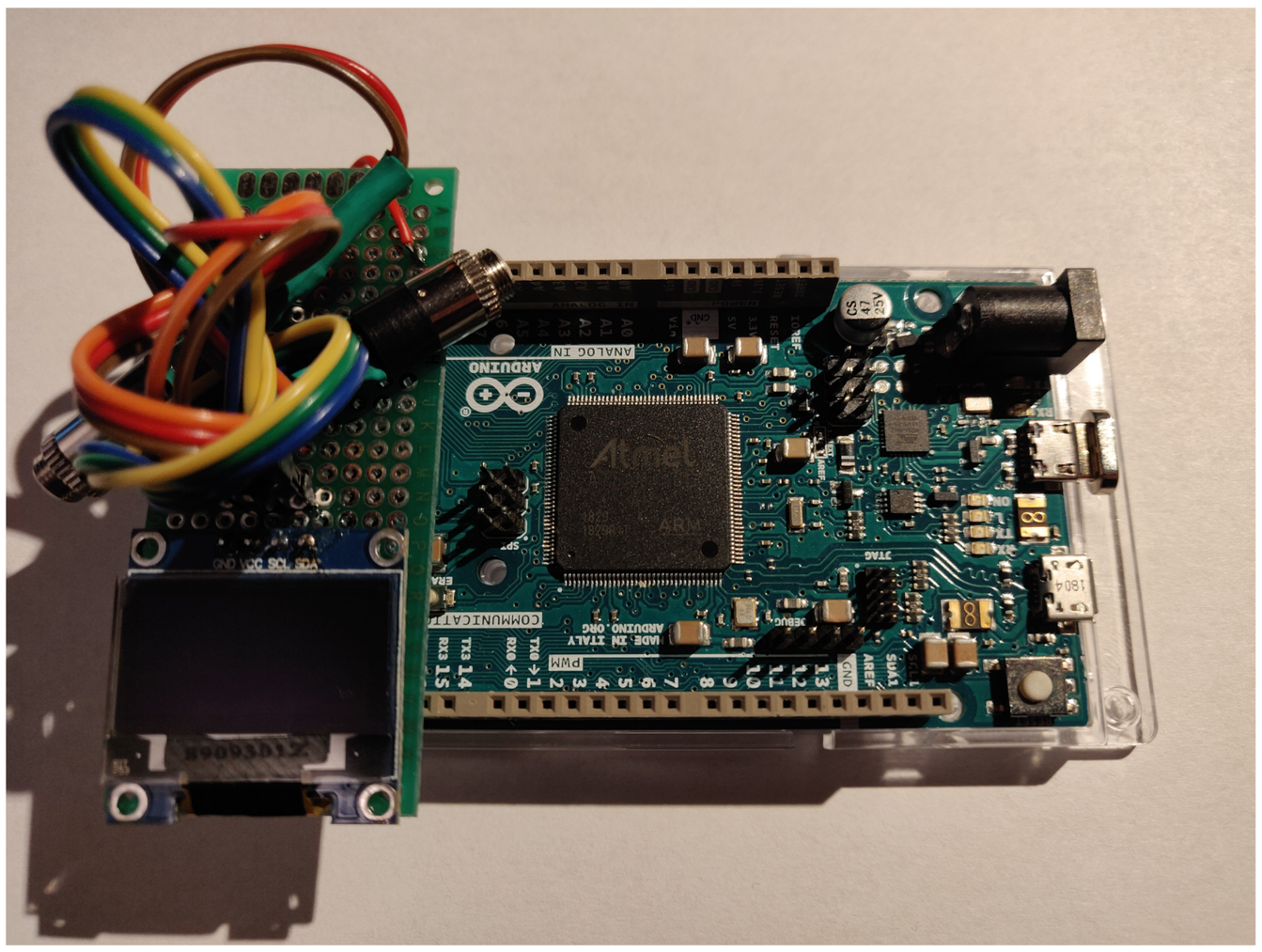
Arduino Audiovisializer

Met Fast Fourier Transformaties





```
class complex {
public:
        double imag, real;

        complex(const double &r, const double &i = 0):
                imag(i),
                real(r)
                {}

        complex& operator+=(const complex &rhs){
                real += rhs.real;
                imag += rhs.imag;
                return *this;
        }
        complex& operator-=(const complex &rhs){
                real -= rhs.real;
                imag -= rhs.imag;
                return *this;
        }
        complex& operator*=(const complex &rhs){
                real *= rhs.real;
                imag *= rhs.imag;
                return *this;
        }
};

complex polar(const double &rho, const double &theta);
complex operator+(const complex &lhs, const complex &r);
complex operator-(const complex &lhs, const complex &r);
complex operator*(const complex &lhs, const complex &r);
```

```
CArray slice(CArray &arr, const unsigned int &start, const unsigned int &end, const unsigned int &step) {
        CArray carr;
        for (unsigned int i = start; i < end; i+=step) {
                carr.push_back(arr[i]);
        }
        return carr;
}
```

```
void fft(CArray &arr) {
        const size_t N = arr.size();
        if (N <= 1) return;

        // std::slice replaced with own function -- functionality roughly equal
        // new function implemented in fft_new.cpp
        CArray even = slice(arr, 0, N/2, 2);
        CArray  odd = slice(arr, 1, N/2, 2);

        fft(even);
        fft( odd);

        for (size_t i = 0; i < N/2; i++) {
                // std::polar replaced with own function -- functionality the same
                // new function implemented in complex.cpp
                complex t = polar(1.0, -2 * PI * i / N) * odd[i];
                arr[i] = even[i] + t;
                arr[i+N/2] = even[i] - t;
        }
}
```

```
float remap(float value, float low1, float high1, float low2, float high2) {
        if ((high1 - low1) != 0) {
                // maths
                return low2 + (high2 - low2) * ((value - low1) / (high1 - low1));
        }
        else {
                // 0-division error; return og value
                return value;
        }
}
```


Auteur

Lennard Duinkerken
1746360
lennard@cooledomeinnaam.nl