

```
1 #include "EC12P.h"
2
3 namespace Hardware
4 {
5     /*! Constructor*/
6     EC12P::EC12P()
7     {
8         // Init Rotary button
9         Button.SetDirection(GPIO::Input);
10        Button.SetEdge(GPIO::Rising);
11
12        // Init Encoder
13        Rotary.set_period(100000000L);
14
15        // Init Encoder color
16        R.SetDirection(GPIO::Output);
17        B.SetDirection(GPIO::Output);
18        G.SetDirection(GPIO::Output);
19        SetPixelColor(None);
20
21        threadRunning = false;
22    }
23
24    /*! De-constructor*/
25    EC12P::~EC12P() { }
26
27    /*! Set the shaft color
28    \param value as Color enumerator
29    */
30    void EC12P::SetPixelColor(Color value)
31    {
32        switch (value)
33        {
34            case Hardware::EC12P::Red:
35                R.SetValue(GPIO::High);
36                B.SetValue(GPIO::Low);
37                G.SetValue(GPIO::Low);
38                break;
39            case Hardware::EC12P::Pink:
40                R.SetValue(GPIO::High);
41                B.SetValue(GPIO::High);
```

```
42         G.SetValue(GPIO::Low);
43         break;
44     case Hardware::EC12P::Blue:
45         R.SetValue(GPIO::Low);
46         B.SetValue(GPIO::High);
47         G.SetValue(GPIO::Low);
48         break;
49     case Hardware::EC12P::SkyBlue:
50         R.SetValue(GPIO::Low);
51         B.SetValue(GPIO::High);
52         G.SetValue(GPIO::High);
53         break;
54     case Hardware::EC12P::Green:
55         R.SetValue(GPIO::Low);
56         B.SetValue(GPIO::Low);
57         G.SetValue(GPIO::High);
58         break;
59     case Hardware::EC12P::Yellow:
60         R.SetValue(GPIO::High);
61         B.SetValue(GPIO::Low);
62         G.SetValue(GPIO::High);
63         break;
64     case Hardware::EC12P::White:
65         R.SetValue(GPIO::High);
66         B.SetValue(GPIO::High);
67         G.SetValue(GPIO::High);
68         break;
69     case Hardware::EC12P::None:
70         R.SetValue(GPIO::Low);
71         B.SetValue(GPIO::Low);
72         G.SetValue(GPIO::Low);
73         break;
74     }
75     PixelColor = value;
76 }
77
78 /*! Loops through all the colors except of as a thread */
79 void EC12P::RainbowLoop(int sleeperperiod)
80 {
81     this->sleeperperiod = sleeperperiod;
```

```
82     this->threadRunning = true;
83     if (pthread_create(&thread, NULL, colorLoop, this)) { throw Exception::FailedToCreateThreadException(); }
84 }
85
86 /*! The thread function that runs trough all the colors*/
87 void *colorLoop(void *value)
88 {
89     int i = 0;
90     EC12P *ec12p = static_cast<EC12P*>(value);
91     EC12P::Color pcolor;
92     while (ec12p->threadRunning)
93     {
94         pcolor = static_cast<EC12P::Color>(i);
95         ec12p->SetPixelColor(pcolor);
96         usleep(ec12p->sleeppperiod);
97         i++;
98         if (i == 6) { i = 0; }
99     }
100     return ec12p;
101 }
102 }
```