Namn och CID på gruppmedlemmar:

Blend Ahmed Omar (blend) Ebbe Ledin (ebbel) Albin Östling (ostlinga)

In [19]:
```python
import numpy as np
import matplotlib.pyplot as plt
plt.style.use("ggplot")
```

Uppgift 1 - 3D-bio

(a) Skriv en kort funktion $J_{proj}(\alpha)$, som ger Jones-matrisen som projicerar E-fältets komponenter på ett koordinatsystem vridet vinkeln α

In [20]:
```python
# Byt ut np.nan värdena så att det blir rätt!
def J_proj(alpha):
    return np.array([[np.cos(alpha), np.sin(alpha)],
                     [-np.sin(alpha), np.cos(alpha)]])
```

(b) Skriv de två korta funktionerna $J_{pol}(\alpha)$, som genererar Jones-matrisen för en roterad polarisator med transmissionsriktningen vinkeln α från x-axeln, samt $J_{ret}(\alpha, \varphi)$ som ger Jones-matrisen för en roterad "retarder" med fasförskjutningen φ radianer (en kvartsvågsplatta har alltså $\varphi = \pi/2$) och eo-riktningen vinkeln α från x-axeln. Utnyttja funktionen $J_{proj}(\alpha)$.

In [21]:
```python
def J_pol(alpha): # J_pol med rotation med vinkel alpha.
    return J_proj(-alpha)@np.array([[1, 0],[0, 0]])@J_proj(alpha)

def J_ret(alpha, phi): # J_ret med rotation med vinkel alpha och fasförskjutning phi.
    return np.matmul(J_proj(-alpha), np.array([[np.exp(1j*phi), 0],[0, 1]]))@J_proj(alpha)
```

(c) Relatera E^{ut} till E^{in} för de fyra olika fallen. Beräkna även I^{ut} och visa att inget ljus kommer fram i fall (ii) och (iii), medan hela E^{in}:s amplitud finns kvar vid observatörens öga i fall (i) och (iv).

In [22]:
```python
# (i) R→R
E1_ut = J_pol(0)@J_ret(-np.pi/4,np.pi/2)@J_ret(np.pi/4,np.pi/2)@np.array([1,0])
I1_ut = np.linalg.norm(E1_ut)**2
print(I1_ut, E1_ut)
print("Intensiteten är 1 så allt ljus går igrnom.")
```

```
1.0000000000000009 [1.11022302e-16+1.j 0.00000000e+00+0.j]
Intensiteten är 1 så allt ljus går igrnom.
```

In [23]:
```python
# (ii) R→L
E2_ut = J_pol(0)@J_ret(np.pi/4,np.pi/2)@J_ret(np.pi/4,np.pi/2)@np.array([1,0])
I2_ut = np.linalg.norm(E2_ut)**2
print(I2_ut, E2_ut)
print("Intensiteten är ungefär 0 så inget ljus går igenom.")
```

```
2.4651903288156624e-32 [-1.11022302e-16+1.11022302e-16j  0.00000000e+00+0.00000000e+00j]
Intensiteten är ungefär 0 så inget ljus går igenom.
```

In [24]:
```python
# (iii) L→R
E3_ut = J_pol(0)@J_ret(-np.pi/4,np.pi/2)@J_ret(-np.pi/4,np.pi/2)@np.array([1,0])
I3_ut = np.linalg.norm(E3_ut)**2
print(I3_ut, E3_ut)
print("Intensiteten är ungefär 0 så inget ljus går igenom.")
```

```
2.4651903288156624e-32 [-1.11022302e-16+1.11022302e-16j  0.00000000e+00+0.00000000e+00j]
Intensiteten är ungefär 0 så inget ljus går igenom.
```

In [25]:
```python
# (iv) L→L
E4_ut = J_pol(0)@J_ret(-np.pi/4,np.pi/2)@J_ret(np.pi/4,np.pi/2)@np.array([1,0])
I4_ut = np.linalg.norm(E4_ut)**2
print(I4_ut, E4_ut)
print("Intensiteten är 1 så allt ljus går igrnom.")
```

```
1.0000000000000009 [1.11022302e-16+1.j 0.00000000e+00+0.j]
Intensiteten är 1 så allt ljus går igrnom.
```

(d) Tyvärr är inte kvartsvåglängdsplattan våglängdsoberoende i verkligheten, utan fungerar bara perfekt för en våglängd i mitten av det synliga spektrumet. Antag att för en våglängd i kanten av det synliga området så avviker fasförskjutningen med 25% (välj själv åt vilket håll) från den perfekta $\pi/2$, för alla kvartsvågsplattorna. Hur mycket av den oönskade bilden – den som tidigare var svart – ser man nu?

In [26]:
```python
# (i) R→R
# (i) R→R
E1_ut = J_pol(0)@J_ret(-np.pi/4,5*np.pi/8)@J_ret(np.pi/4,5*np.pi/8)@np.array([1,0])
I1_ut = np.linalg.norm(E1_ut)**2
print(I1_ut, E1_ut)
```

```
1.0000000000000004 [-0.38268343+0.92387953j  0.        +0.j        ]
```

In [27]:
```python
# (ii) R→L
# (ii) R→L
E2_ut = J_pol(0)@J_ret(np.pi/4,5*np.pi/8)@J_ret(np.pi/4,5*np.pi/8)@np.array([1,0])
I2_ut = np.linalg.norm(E2_ut)**2
print(I2_ut, E2_ut)
```

```
0.14644660940672638 [0.14644661-0.35355339j 0.        +0.j        ]
```

In [28]:
```python
# (iii) L→R
E3_ut = J_pol(0)@J_ret(-np.pi/4,5*np.pi/8)@J_ret(-np.pi/4,5*np.pi/8)@np.array([1,0])
I3_ut = np.linalg.norm(E3_ut)**2
print(I3_ut, E3_ut)
```

```
0.14644660940672638 [0.14644661-0.35355339j 0.        +0.j        ]
```

In [29]:
```python
# (iv) L→L
E4_ut = J_pol(0)@J_ret(-np.pi/4,5*np.pi/8)@J_ret(np.pi/4,5*np.pi/8)@np.array([1,0])
I4_ut = np.linalg.norm(E4_ut)**2
print(I4_ut, E4_ut)
```

```
1.0000000000000004 [-0.38268343+0.92387953j  0.        +0.j        ]
```

Vi ökade fasförskjutningen 25% och fick att R->R och L->L har samma intensitet medan intensiteten ökar för L->R och R->L från 0 till ca. 0.15.

(e) En spökbild, som den oönskade bilden i (d), är inte alls bra för 3D-upplevelsen. Som tur är finns det en mirakulöst enkel lösning på detta problem, som visas i figuren nedan: rotera glasen vid papperslapparna 90º! Visa att detta system inte ger någon spökbild alls, trots en felaktig fasförskjutning så hög som 25%. Ett litet pris får man dock betala för detta, vilket?

In [30]:
```python
# kod
# (i) R→R
Ein_roterad = np.array([0,1])

E1_ut = J_pol(0)@J_ret(np.pi/4,5*np.pi/8)@J_ret(np.pi/4,5*np.pi/8)@Ein_roterad
I1_ut = np.linalg.norm(E1_ut)**2
print("R till R:", I1_ut, E1_ut)

# (ii) R→L
E2_ut = J_pol(0)@J_ret(np.pi/4,5*np.pi/8)@J_ret(-np.pi/4,5*np.pi/8)@Ein_roterad
I2_ut = np.linalg.norm(E2_ut)**2
print(I2_ut, E2_ut)

# (iii) L→R
E3_ut = J_pol(0)@J_ret(-np.pi/4,5*np.pi/8)@J_ret(np.pi/4,5*np.pi/8)@Ein_roterad
I3_ut = np.linalg.norm(E3_ut)**2
print(I3_ut, E3_ut)

# (iv) L→L
E4_ut = J_pol(0)@J_ret(-np.pi/4,5*np.pi/8)@J_ret(-np.pi/4,5*np.pi/8)@Ein_roterad
I4_ut = np.linalg.norm(E4_ut)**2
print(I4_ut, E4_ut)
```

```
R till R: 0.8535533905932742 [-0.85355339-0.35355339j  0.        +0.j        ]
6.216093914885078e-35 [-7.8842209e-18+0.j  0.0000000e+00+0.j]
6.216093914885078e-35 [7.8842209e-18+0.j 0.0000000e+00+0.j]
0.8535533905932742 [0.85355339+0.35355339j 0.        +0.j        ]
```

Att intensitetn inte är noll för R->L och L->R är inte önskvärt så denna metod löser det problemet, men priset man får betala för lösningen är att intensiteten för R->R och L->L sjunker till ca. 0.85 från 1.

Uppgift 2 - Bildskärmen

(a) Skriv en funktion som genererar Jones-matrisen för den n-te roterade tunna skivan, $J_{ret2}(\alpha_n, \delta, n_{eo,\theta}, n_O, \lambda)$ där $n_{eo,\theta} = n_{eo}(\theta)$ för den aktuella tiltvinkeln och λ är (vakuum-)våglängden för ljuset som passerat färgfiltret. Använd $J_{proj}(\alpha)$ från föregående uppgift.

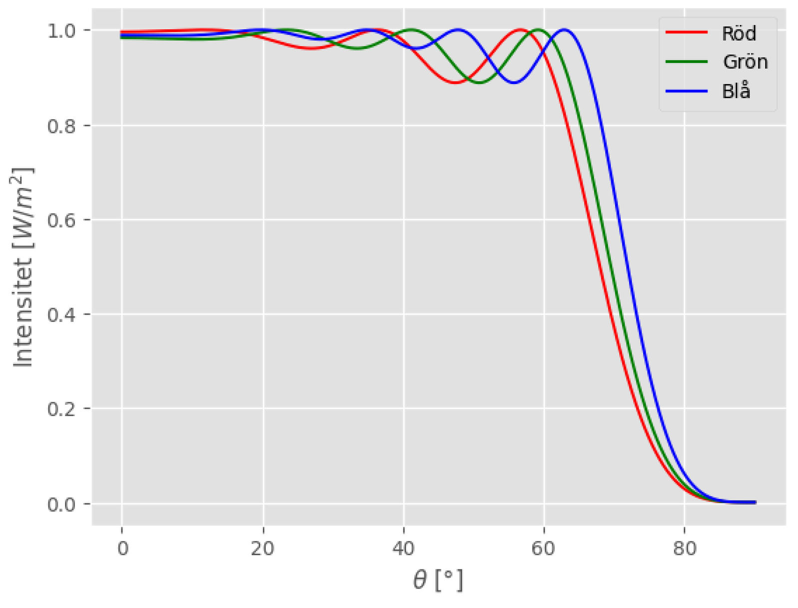
In [31]:
```python
def J_ret2(alpha, delta, ne0, n0, lam):
    return J_proj(-alpha)@np.array([[np.exp(1j*delta*(ne0-n0)*2*np.pi/lam), 0],[0, 1]])@J_proj(alpha)
```

(b) Med hjälp av funktionen $J_{ret2}(\alpha_n, \delta, n_{eo,\theta}, n_O, \lambda)$ samt $J_{pol}(\alpha)$ från föregående uppgift, skriv kod som beräknar Jones-matrisen som relaterar E^{ut} och E^{in} enligt figuren i uppgiftsbeskrivningen.

Gör detta för ett antal olika värden på tiltvinkeln θ och plotta intensiteten på det ljus som kommer utfrån pixeln (efter högra polarisatorn) som funktion av θ. I plotten, markera vilket värde på θ som svarar mot 0V spänning respektive hög spänning. Är det alltså möjligt att erhålla en kontinuerlig modulation av intensiteten mellan 0 och 100% för alla färger med denna LC skärm?

In [32]:
```python
n0 = 1.5
ne0 = 1.6
d = 20e-6
lam_r = 633e-9
lam_g = 549e-9
lam_b = 432e-9
N = 50

theta = np.arange(0, np.pi/2, 0.001)

def ne(theta, ne0, n0):
    return ne0*n0/(np.sqrt(n0**2*np.cos(theta)**2+ne0**2*np.sin(theta)**2))



def I_ut(theta, lam):
    E_in = np.array([1,0])
    for n in range(N):
        alpha_n = np.pi*n/(2*N)
        E_in = J_ret2(alpha_n, d/N, ne(theta, ne0, n0), n0, lam)@E_in

    E_ut = J_pol(alpha_n)@E_in
    return np.linalg.norm(E_ut)**2

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_r))

plt.plot(theta*180/np.pi, I_ut_matris, label="Röd", color="r")

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_g))

plt.plot(theta*180/np.pi, I_ut_matris, label="Grön", color="g")

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_b))

plt.plot(theta*180/np.pi, I_ut_matris, label="Blå", color="b")

plt.legend()
plt.xlabel(r'$\theta$ [°]')
plt.ylabel("Intensitet [$W/m^2$]")
print("Theta är 0 vid 0 spänning och 90 grader vid hög spänning. Och det är möjligt att erhålla en kontinuerlig modulation mellan 0 och 100% för varje våglängd.")
```

```
Theta är 0 vid 0 spänning och 90 grader vid hög spänning. Och det är möjligt att erhålla en kontinuerlig modulation mellan 0 och 100% för varje våglängd.
```



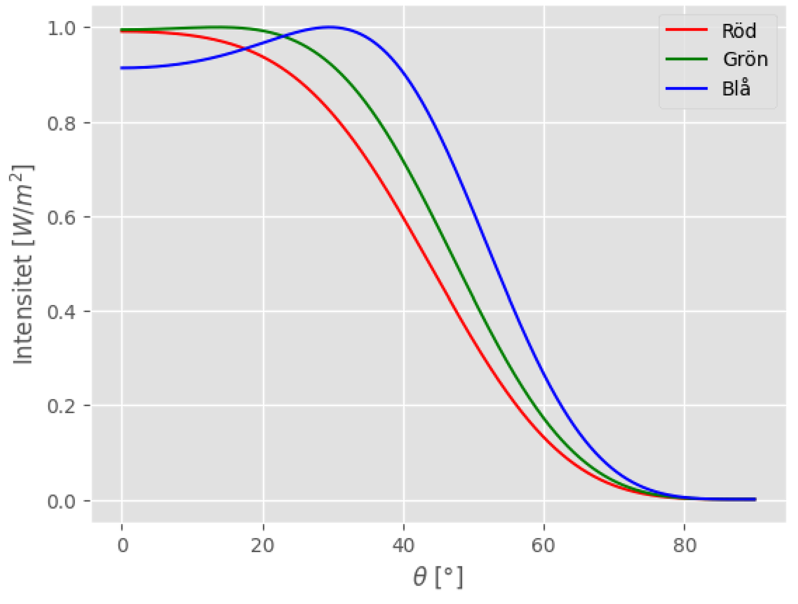
(c) Hur ser alltså en "död" pixel ut, d.v.s. en pixel som det, p.g.a. tillverkningsfel, inte går att lägga spänning över?

En "död" pixel har theta=0, vilket ger således att intensiteten för de olika färgerna på ljuset är alla 1 och den resulterande färgen blir vit.

(d) Hur tunt kan vätskekristallskiktet göras? Med andra ord, hur liten kan d vara under förutsättning att vi fortfarande vill kunna modulera utintensiteten mellan 0 och 100% genom att ändra tiltvinkeln mellan 0 och 90°. Att ha ett litet värde på d är viktigt eftersom detta ökar snabbheten i ändringen av tiltvinkel så att displayen kan visa snabba bildsekvenser.

In [33]:
```python
d=5.2e-6

def I_ut(theta, lam):
    E_in = np.array([1,0])
    for n in range(N):
        alpha_n = np.pi*n/(2*N)
        E_in = J_ret2(alpha_n, d/N, ne(theta, ne0, n0), n0, lam)@E_in

    E_ut = J_pol(alpha_n)@E_in
    return np.linalg.norm(E_ut)**2

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_r))

plt.plot(theta*180/np.pi, I_ut_matris, label="Röd", color="r")

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_g))

plt.plot(theta*180/np.pi, I_ut_matris, label="Grön", color="g")

I_ut_matris = []
for i in theta:
    I_ut_matris.append(I_ut(i, lam_b))

plt.plot(theta*180/np.pi, I_ut_matris, label="Blå", color="b")

plt.legend()
plt.xlabel(r'$\theta$ [°]')
plt.ylabel("Intensitet [$W/m^2$]")
```

Out[33]:
```
Text(0, 0.5, 'Intensitet [$W/m^2$]')
```



d = 5.2e-6 är den ungefärliga gränsen för att kontinuerlig modulation mellan 0 till 100% ska erhållas. Flera värden har testats.