Lab 4 Prep: Design and Manufacture Inductor

Lauren Xiong, Ian Eykamp, Melissa Kazazic

In [ ]:
```
# %%capture
# Imports and setup
from pint import UnitRegistry
import math
import numpy

# use pint
units = UnitRegistry()
units.default_format = "~P"
units.load_definitions('units_definition.txt')
```

Out[ ]:

```
{None: ParsedSource(parsed_source=PintRootBlock(opening=BOF(start_line=0, start_co
l=0, end_line=0, end_col=0, raw=None, content_hash=Hash(algorithm_name='blake2b',
hexdigest='fc8618d3ea1c0bbca70f75d4bd6141ec24df680bd120dd216c2b2e09280a971990a1fb7
1c275ca4cdca0f1a5c639dfc3886d3ee06244dc0a296617a0ff969879'), path=WindowsPath('c:/
dev/git/power-electronics/Lab 4/units_definition.txt'), mtime=1678082649.396907),
body=(UnitDefinition(name='tesla', defined_symbol='T', aliases=('teslas',), conver
ter=ScaleConverter(scale=1.0), reference=<UnitsContainer({'amp': -1, 'kilogram':
1, 'second': -2})>, start_line=1, start_col=0, end_line=1, end_col=60, raw='tesla
= kilogram * second ** (-2) * amp ** (-1) = T = teslas'), UnitDefinition(name='web
er', defined_symbol='Wb', aliases=('webers',), converter=ScaleConverter(scale=1),
reference=<UnitsContainer({'meter': 2, 'tesla': 1})>, start_line=2, start_col=0, e
nd_line=2, end_col=40, raw='weber = tesla * meter ** 2 = Wb = webers')), closing=E
OS(start_line=3, start_col=0, end_line=3, end_col=0, raw=None)), config=ParserConf
ig(non_int_type=<class 'float'>))}
```


Pt 1. Determining Turns and Gap Distance

Inductance and Permeance

In [ ]:
```
# Calculating for Inductance and Permeance

target_inductance = 20e-6 * units.henry
print(target_inductance)

perm_air = 4e-7*math.pi*(units.meter*units.kilogram*units.second**(-2)*units.amp**(
eq_area_core = 62.6 * units.millimeter**2
eq_length_core = 45.7 * units.millimeter
rel_perm_core = 3300

# changing variables
eq_length_gap = 0.009 * units.inch
n_turns = 7
```

```
# gap area
lg = eq_length_gap
g = 9 * units.millimeter

eq_area_wing_gap = (2 * ((7 * units.millimeter) + lg) * ((10.65 * units.millimeter)
    (((7 * units.millimeter) + lg) * (numpy.sqrt((g - lg)** 2 - (7 * units.millimet
    ((g - lg) ** 2) * (math.asin((7 * units.millimeter + lg) / (g - lg)))


area_circle = ((7.9 / 2) * units.millimeter + lg)**2 * math.pi
eq_area_gap = (area_circle + 2 * eq_area_wing_gap)/2

print("Circle area:", area_circle)
print("Wing gap area:", eq_area_wing_gap)
print("Total equivalent area of gap:", eq_area_gap)

P_gap = (perm_air*eq_area_gap)/eq_length_gap
P_core = (perm_air*rel_perm_core*eq_area_core)/eq_length_core
P = ((P_core)**(-1)+(P_gap)**(-1))**(-1)
L = P*(n_turns**2)
print("Calculated Permeance:", units.Quantity(P, units.microhenry))
print("Calculated Inductance:",units.Quantity(L, units.microhenry))
```

```
2×10⁻⁵ H
Circle area: 54.85440043768629 mm²
Wing gap area: 46.835775356850235 mm²
Total equivalent area of gap: 74.26297557569337 mm²
Calculated Permeance: 0.38086007700599384 µH
Calculated Inductance: 18.6621437732937 µH
```

Peak Current Density

In [ ]:

```
# Peak current density

awg_val = 20 # AWG
print(awg_val, "AWG")

awg_d = 0.005*92**((36-awg_val)/39)*units.inch
awg_area = math.pi*((awg_d/2)**2)

D = 0.5 # duty cycle
ipeak = 10*units.amp
irms = ipeak * (numpy.sqrt(D/3)) # current root mean square

# current density
current_density = irms/awg_area
print("Current density:", units.Quantity(current_density,units.A/units.millimeter**
```

```
20 AWG
Current density: 7.887038529387198 A/mm²
```

Peak Flux Density

In [ ]:

```
# Peak flux density
```

```
# givens
min_effective_area = 59.1 * units.millimeter**2
Bmax = 200*units.mT

# magnetic potential
mmf = n_turns*irms

# max flux density
flux_max = (mmf*P)/min_effective_area

print("Bmax:",Bmax)
print("Maximum Flux Density:",units.Quantity(flux_max,units.mT))
```

```
Bmax: 200 mT
Maximum Flux Density: 184.16215353729703 mT
```

Pt. 2: Real values

In [ ]:
```
# Recorded values
n_actual = 9.3 #turns
L_actual = 0.0189 * units.millihenry

# Permeance
P_actual = L_actual/(n_actual**2)

print("Measured Turns:", n_actual)
print("Measured Inductance:", units.Quantity(L_actual, units.uhenry))
print("Measured Permeance:", units.Quantity(P_actual, units.uhenry))
```

```
Measured Turns: 9.3
Measured Inductance: 18.9 µH
Measured Permeance: 0.218522372528616 µH
```

In [ ]:
```
# Peak flux density

flux_actual = n_actual*irms*P_actual

flux_density_actual = flux_actual/min_effective_area

print("Actual Maximum Flux Density:",units.Quantity(flux_density_actual,units.mT))
```

```
Actual Maximum Flux Density: 140.38339773605898 mT
```

So, despite needing more turns to meet the required inductance, this also meant that the permeance shrunk, and so the maximum flux density is also around the value that we wanted.

Peak Energy Stored

Using Calculated Values

In [ ]:
```
# Total delta Work
dW_total = (1/2)*(mmf**2)*P
```

```python
# Work of Core
dW_core = dW_total * (P/P_core)

# Work of Gap
dW_gap = dW_total * (P/P_gap)

# Print values
print("Estimated total delta Work", units.Quantity(dW_total,units.ujoule))
print("Core delta Work", units.Quantity(dW_core,units.ujoule))
print("Gap delta Work", units.Quantity(dW_gap,units.ujoule))
```

```
Estimated total delta Work 155.5178647774475 µJ
Core delta Work 10.427106244646026 µJ
Gap delta Work 145.09075853280146 µJ
```

Using Measured Value

In [ ]:

```python
# Total Work

dW_actual = (1/2)*(mmf**2)*P_actual
print("Actual delta Work", units.Quantity(dW_actual,units.ujoule))
```

```
Actual delta Work 89.22996878251821 µJ
```