Self Adaptive Hartley Oscillator using ML techniques

Course Project, as a part of the course - Electronic Workshop - II)

Sreya Garapati, 2020102055, Aryan Singhal, 2020102032, Shubham Priyadarshan, 2020102027

***Abstract*—The Self Adaptive Hartley Oscillator using Machine Learning (ML) is an innovative project that aims to enhance the robustness of the Hartley oscillator circuit to temperature and voltage variations by incorporating adaptive control techniques and ML algorithms. The Hartley oscillator is a widely utilized electronic circuit for generating sinusoidal signals in numerous applications, including communication systems, audio devices, and frequency synthesis.**

Temperature and voltage variations can significantly impact the performance and stability of the Hartley oscillator, leading to frequency drift, amplitude fluctuations, and unreliable operation. Conventional approaches require manual adjustments and meticulous calibration to compensate for these variations, resulting in reduced efficiency and increased vulnerability to errors. In this project, we propose a self-adaptive methodology that utilizes ML algorithms to autonomously optimize the oscillator's parameters and maintain stable operation under varying temperature and voltage conditions.

I. Introduction

THE Hartley oscillator is a widely used electronic circuit for generating sinusoidal signals in various applications. However, the circuit's performance can be significantly affected by temperature and voltage variations, resulting in frequency drift, amplitude fluctuations, and unreliable operation. Traditional methods often involve manual adjustments and calibration to compensate for these variations, which can be time-consuming and inefficient.

To address these challenges, this project proposes a novel approach that combines Machine Learning (ML) algorithms with manual adjustment to create a self-adaptive Hartley oscillator capable of maintaining stable operation in the presence of temperature and voltage variations. By leveraging ML techniques, the project aims to develop an ML model that predicts the expected frequency output of the oscillator based on temperature and voltage inputs. This prediction is used as a reference to manually adjust the dominant component of the oscillator, bringing it back to its normal operating state.

The primary objective of this project is to design an ML model that takes temperature and voltage as input and predicts the expected frequency response of the oscillator. The ML model is trained on historical data, capturing the relationship between temperature, voltage, and frequency variations. Once the ML model predicts the expected frequency output, it serves as a guide for manual adjustment of the dominant component of the oscillator. By iteratively adjusting the component based on the ML model's predictions, the oscillator can be brought back to its desired frequency, compensating for the temperature and voltage variations.

The self-adaptive Hartley oscillator using ML prediction and manual adjustment offers several advantages over traditional methods. While the adjustment is done manually, the ML model provides valuable insights into the expected frequency response based on varying temperature and voltage inputs. This guidance allows for more efficient and accurate adjustments, reducing the time and effort required for calibration. Additionally, by incorporating ML predictions, the oscillator can adapt to changing conditions and maintain stable operation, enhancing its robustness.

Throughout this project, we will explore the methodology, implementation, and experimental results of the self-adaptive Hartley oscillator using ML prediction and manual adjustment. By combining ML insights with manual intervention, we aim to create a practical and efficient solution for maintaining stable oscillator operation in the presence of temperature and voltage variations. This project holds promise for applications where reliable and accurate sinusoidal signal generation is crucial, even in challenging operating conditions.

II. Materials and Methods

List of materials used and how these were used / connected

TABLE I
Components for Self Adaptive Hartley Oscillator

Component	Description
2x 100uH Inductors	Two 100 microhenry inductors
1n Ceramic Capacitor	A 1 nanofarad ceramic capacitor
2x 4.7uF Capacitors	Two 4.7 microfarad capacitors
1 BC547B Transistor	BC547B NPN bipolar junction transistor
1x 74K Resistor	A 74 kilohm resistor
1x 4.7K Resistor	A 4.7 kilohm resistor
1x 15K Resistor	A 15 kilohm resistor
1x 1K Resistor	A 1 kilohm resistor
1x DHT22 Temperature Sensor	DHT22 digital temperature and humidity sensor
1x Voltage Sensor	Voltage sensor module
1x Arduino Uno	Arduino Uno microcontroller board
1x 16x2 LCD Display	16x2 character LCD display module
1x 10K Potentiometer	A 10 kilohm potentiometer
1x 120 ohm Resistor	A 120 ohm resistor
Jumper Wires	Set of jumper wires

Below is the picture of the MicroCap circuit that has been assembled for the self adaptive Hartley oscillator project:

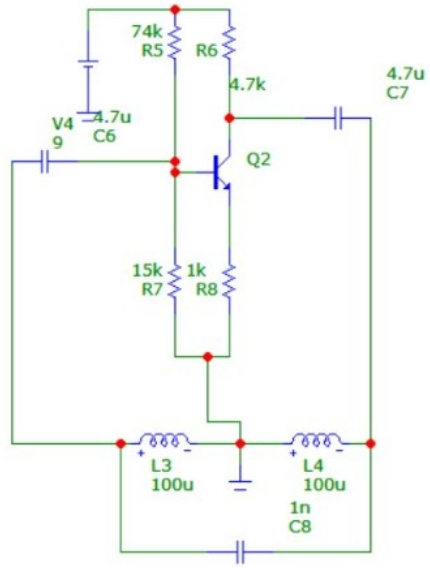

Fig. 1. Microcap Circuit

The following image shows the Tinkercad circuit for the self adaptive Hartley oscillator project:

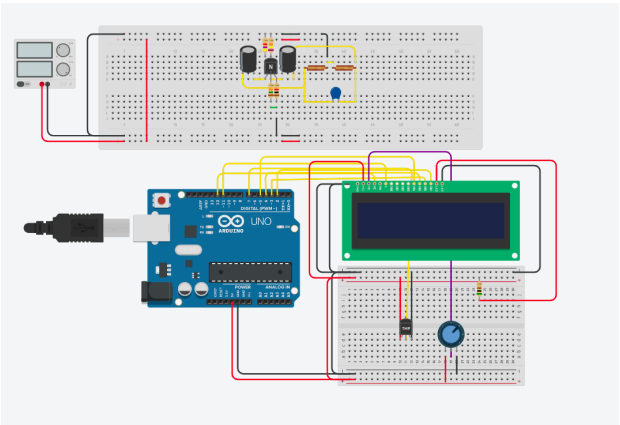

Fig. 2. TinkerCad Circuit

Here is the schematic diagram depicting the circuit design for the self adaptive Hartley oscillator project:

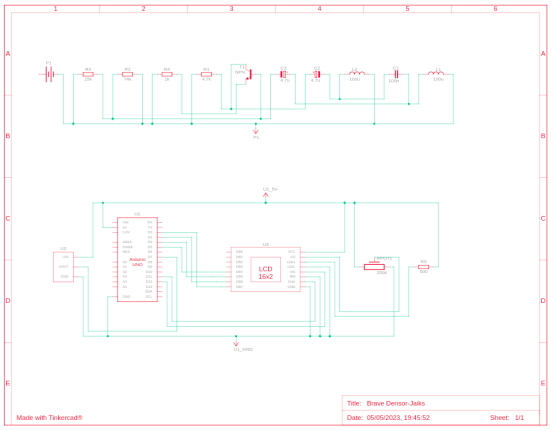

Fig. 3. Schematic

The frequency of a Hartley oscillator can be calculated using the simplified formula:

$$f = \frac{1}{2\pi\sqrt{LC}}$$

Where:

f - Frequency of the oscillator
L - Equivalent Inductance in the tank circuit
C - Equivalent Capacitance in the tank circuit

where $L = 100\mu F + 100\mu F$ and $C = 1\,\text{nf}$.
Substituting the values, we have:

$$f = \frac{1}{2\pi\sqrt{(100+100)\times 10^{-6}\times 1\times 10^{-9}}}$$

Simplifying further:

$$f \approx 355,881.271\,\text{Hz}\ \textit{or}\ 355.881\text{KHz}$$

III. Results

The waveplot for microcap simulation:

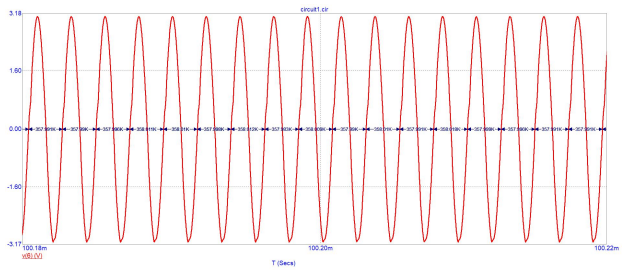

Fig. 4. Simulation Waveform with a frequency of approx 356 KHz

The waveplot for hardware looks as follows:

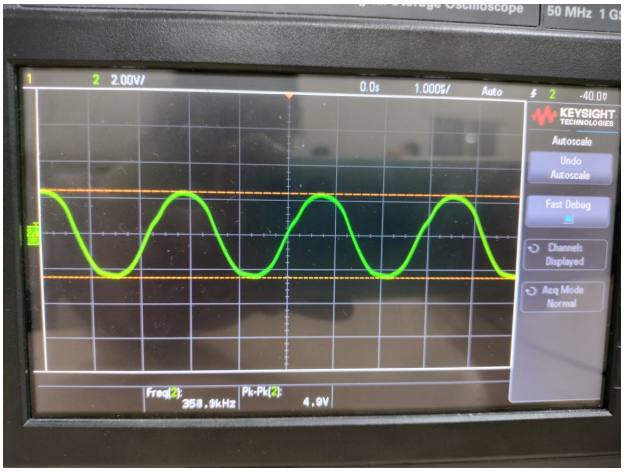

Fig. 5. Hardware simulation Waveform with a frequency of approx 353 KHz

We also simulated the circuit over a range of temperatures from 0 to 80 degree Celcius. The plot of Frequency vs Temperature obtained was:

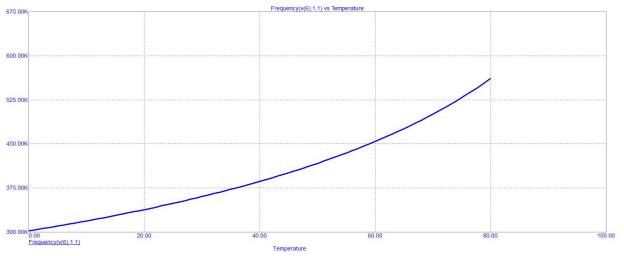

Fig. 6. Frequency vs Temperature plot

Observing an increase in frequency with an increase in temperature in our Hartley oscillator has led us to investigate the underlying causes and propose necessary measures to restore circuit performance. In this regard, it is crucial to identify and address the dominant component within the circuit, which, in the case of the Hartley oscillator, is the capacitor.

Ceramic capacitors, commonly employed in the oscillator's tank circuit, are known to exhibit a decrease in capacitance as temperature rises. This behavior is attributed to the temperature coefficient of capacitance, wherein the dielectric material within the capacitor experiences changes in its electrical properties due to temperature variations. Consequently, the decrease in capacitance results in an increase in the oscillator's output frequency.

To rectify this issue and restore the desired circuit performance, it becomes essential to address the impact of temperature on the ceramic capacitor. One approach is to replace the capacitor with a type that exhibits a reduced temperature coefficient of capacitance. Capacitors with lower temperature coefficients will experience minimal changes in capacitance over a wide temperature range, leading to improved frequency stability.

Furthermore, implementing temperature compensation techniques can be considered to mitigate the effects of temperature variations on the oscillator frequency. This can involve introducing additional circuit elements that counteract the frequency shift caused by temperature changes. Such techniques aim to maintain the desired frequency output despite temperature fluctuations.

For our circuit we calculated the temperature coefficients for a 1nF capacitor. The linear coefficient came out to be -0.01307297 and the quadratic coefficient was around 28.799μ

In order to address the issue of frequency variation in the Hartley oscillator due to temperature changes, we developed a machine learning (ML) model that accurately predicts the output frequency based on the temperature and voltage inputs. This model takes into consideration the temperature coefficient of capacitance, which affects the capacitance of the ceramic capacitor used in the oscillator.

The ML model was trained using a dataset that included temperature, voltage, and corresponding output frequencies. We incorporated the temperature coefficient of capacitance as a feature in the training data, allowing the model to learn the relationship between temperature, capacitance variation, and output frequency. By considering this additional factor, the ML model can capture the impact of temperature on the oscillator's frequency more accurately.

The training process involved feeding the dataset into the ML algorithm, which employed regression techniques to learn the complex patterns and correlations between the input variables and the output frequency. The model was fine-tuned using various algorithms and evaluated using appropriate performance metrics to ensure its accuracy and reliability.

Once the ML model was trained and validated, we applied it to predict the output frequency of the Hartley oscillator for new temperature and voltage inputs. By inputting the current temperature and voltage values into the model, it can provide an accurate estimation of the corresponding output frequency, taking into account the temperature coefficient of capacitance. In our study, we evaluated the performance of the developed machine learning (ML) model by comparing its predictions to the hardware values obtained from the Hartley oscillator. The results demonstrated high accuracy, with an error percentage ranging between 3 and 5, indicating a close alignment between the predicted and actual values. Additionally, the coefficient of determination (r2 score) for the model was measured at an impressive 99.998, further confirming its reliability and precision.

The developed ML model offers several advantages in addressing the frequency variation issue in the Hartley oscillator. It eliminates the need for manual calibration or adjustment of the circuit components in response to temperature changes. Instead, it provides a real-time prediction of the output frequency, allowing for proactive measures to be taken to stabilize the frequency if necessary.

Once the ML model predicted the output frequency based on the given temperature and voltage inputs, we utilized this information to adjust the capacitor in the tank circuit.

By comparing the predicted frequency to the desired original frequency, we were able to determine the necessary

adjustment required in the capacitor. This adjustment was calculated based on the difference between the predicted and desired frequencies and the known relationship between the capacitance and frequency in the Hartley oscillator.
The self-adaptive process involved selecting a new capacitor with an appropriate capacitance value that would bring the frequency back to the desired original frequency. This adjustment was made to compensate for the frequency shift caused by temperature changes.

By incorporating this self-adaptive mechanism, we achieved the objective of maintaining the desired frequency output despite variations in temperature. The ML model acted as a predictive tool, providing real-time insights into the necessary adjustments required for the capacitor in the tank circuit.

This self-adaptive approach not only improved the stability of the Hartley oscillator but also eliminated the need for manual calibration or intervention to maintain the desired frequency. It allowed for automatic and dynamic adjustments based on the ML model's predictions, ensuring consistent and accurate performance even in changing temperature conditions.
Here is a table that depicts our hardware simulation, while changing temperature and self adapting the circuit with the help of ML model. We heat up the circuit to a particular temperature, observe the frequency and change the capacitor in the tank circuit to achieve or baseline frequency. The table depicts, the required capacitance to maintain a frequency of around 354KHz for different temperatures.

TABLE II
TEMPERATURE VS. REQUIRED CAPACITANCE AND FREQUENCY

Temperature (°C)	Required Capacitance (nF)	Frequency (kHz)
41.10	1.15	353.80
43.10	1.2	353.10
46.5	1.35	352.80
52.80	1.4	353.20
55.10	1.5	353.70

Note: The baseline value was 353.8 KHz at a room temperature of 31.2 degrees.

IV. CODES

```
# ML model
import tensorflow as tf
import numpy as  np
import pandas as pd
from math import sqrt

headers = ["temperature","voltage","frequency"]
data = pd.read_csv('./fixed.csv', na_values='?',
        header=None,names = headers)
data = data.reset_index(drop=True)
# data = data.iloc[1:]
data= data.dropna()
data.head()

data["voltage"]= pd.to_numeric(data['voltage']
,errors = 'coerce')
data['temperature'] = pd.to_numeric(
data['temperature']
,errors = 'coerce')
data['frequency'] = pd.to_numeric(data['frequency']
,errors = 'coerce')

x= data.drop(['frequency'],axis=1).values
y = data['frequency'].values
print(np.max(np.abs(y)))
y = y/np.max(np.abs(y))

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(x, y, test_size=0.2)


from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
print(scaler.mean_)
print(scaler.scale_)

x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(x_train)
print(y_train)

from keras.optimizers import Adam,
SGD, RMSprop, Adadelta, Adamax, Nadam, Ftrl
from keras.models import Sequential
from keras.layers import Dense, Dropout, LeakyReLU
import keras
from keras import regularizers
model= Sequential()
model.add(Dense(2, activation=
LeakyReLU(alpha=0.2), input_dim = 2))
model.add(Dense(4, activation=LeakyReLU(alpha=0.2)))
# model.add(Dense(8, activation='relu'
, kernel_regularizer=regularizers.l2(0.05)))

# model.add(Dense(2, activation=LeakyReLU(alpha=0.2)))
model.add(Dense(1))

model.compile(tf.keras.optimizers.Adam
(learning_rate=0.001),
'mse')

x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)

model.fit(x_train, y_train, validation_data=(x_test,
y_test)
,batch_size=8, epochs=1000)

y_pred = model.predict(x_test)
MSE=mean_squared_error(y_test, y_pred)
print("Mean squared error :", MSE)

RMSE=sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean squared error:", RMSE)

MAE=mean_absolute_error(y_test, y_pred)
print("Mean Absolute error :", MAE)

print("R2 score :", r2_score(y_test,
```

```
y_pred))

model.save('VCO_model.h5')
print(y_pred[1:11])
print(y_test[1:11])

print("Weights and biases of the layers after
training the model: \n")
for layer in model.layers:
  print(layer.name)
  print("Weights")
  print("Shape: ",layer.get_weights()[0].shape,
  '\n',layer.get_weights()[0])
  print("Bias")
  print("Shape: ",layer.get_weights()[1].shape,
  '\n',layer.get_weights()[1],'\n')

  from keras.models import load_model
#model.save('VCO_model.h5')
model = load_model('VCO_Model_New.h5')
mean = [39.62271341,  9.00121951]
std = [23.20747887,  1.18176183]
ip = [60, 9]
for i in range(2):
  ip[i] = (ip[i]-mean[i])/std[i]
print(ip)

out = model.predict([ip])
out = out * 554510.300355
print(out)

y_pred = model.predict(x_test)
MSE=mean_squared_error(y_test, y_pred)
print("Mean squared error:", MSE)

RMSE=sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean squared error:", RMSE)

MAE=mean_absolute_error(y_test, y_pred)
print("Mean Absolute error:", MAE)

print("R2 score :", r2_score(y_test,
y_pred))

print(y_test)
print(y_pred)
```

V. Arduino Code

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
// #include <LiquidCrystal.h>
// #include <dht.h>
#define DHTPIN 7
// const int rs = 12, en = 11, d4 = 5,
d5 = 4, d6 = 3, d7 = 2;

// LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// const int numRows = 2;


// const int numCols = 16;
#define DHTTYPE    DHT22


#define ANALOG_IN_PIN A0

float adc_voltage = 0.0;
float in_voltage = 0.0;

int adc_value = 0;

float R1 = 30000.0;
float R2 = 7500.0;
float ref_voltage = 5.0;

DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;

void  setup()
{
  digitalWrite(6, 1);
  // lcd.begin(numCols, numRows);
  Serial.begin(9600);
  dht.begin();
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println(F("------------------------------------
  Serial.println(F("Temperature Sensor"));
  delayMS = sensor.min_delay / 1000;
}

float means[2] = {39.62271341, 9.00121951};
float stddev[2] = {23.20747887, 1.18176183};

float weights1[2][2] = {{0.40762582, -0.28312975},
{-0.00050972,  0.00065389}};

float weights2[2][4] = {{-0.62219965,
0.00302935, 0.02959656, 0.6744388},
{-0.551139, -0.01360224, -0.8541575, -0.13763617}};

float weightsop[4] = {0.40393323, 0.81051797 ,
0.26795056,  0.5728935 };

// float weightsop[8] =
float biases1[2] = {-0.14124358, 0.48345032};
float biases2[4] =  {0.30922526,
0.30305612, 0.15884927, 0.28856438};
float biasop =   0.3285151;

void loop()
{
  delay(1000);
  // Get temperature event and print its value.
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
    Serial.println(F("Error reading temperature!"));
    adc_value = analogRead(ANALOG_IN_PIN);
    adc_voltage  = (adc_value * ref_voltage) / 1024.0;
    in_voltage = adc_voltage / (R2/(R1+R2)) ;
    in_voltage = in_voltage - 0.2;
    Serial.print("Input Voltage is ");
    Serial.print(in_voltage);
    Serial.println("V");
  }
  else {
    float t = event.temperature;

    adc_value = analogRead(ANALOG_IN_PIN);
    adc_voltage  = (adc_value * ref_voltage) / 1023.0;
    in_voltage = adc_voltage / (R2/(R1+R2)) ;
```

```
    // in_voltage = in_voltage - 0.2;
    // lcd.print("hello, world!");
    float v = in_voltage;
    float input[2] = {t, v};

    for(int i = 0; i<2; i++){
      input[i] = (input[i] - means[i])/stddev[i];
    }

    float layer1op[2], layer2op[4];
    float output;
    layer1transfer(input, layer1op);
    layer2transfer(layer1op, layer2op);
    output = finalOP(layer2op);
    output = 554510.300355* output;
    Serial.print("Current temperature is ");
    Serial.print(t);
    Serial.println("C");
    Serial.print("Final output is ");
    Serial.print(output);
    Serial.println("Hz");
    Serial.print("Input Voltage is ");
    Serial.print(v);
    Serial.println("V");

    // lcd.setCursor(0,0);
    // lcd.print(output);

    // lcd.setCursor(1,0);
    // lcd.print(t);

  }

}

void layer1transfer(float *ip, float *op)
{
  float alpha = 0.2;
  for(int i=0;i<2;i++)
  {
    float temp = 0.00000;
    for(int j=0;j<2;j++)
    {
      temp += ip[j]*weights1[j][i];
    }
    temp += biases1[i];
    op[i] = max(temp,alpha*temp);
  }
}

void layer2transfer(float *ip, float *op)
{
  float alpha = 0.2;
  for(int i=0;i<4;i++)
  {
    float temp = 0.00000;
    for(int j=0;j<2;j++)
    {
      temp += ip[j]*weights2[j][i];
    }
    temp += biases2[i];
    op[i] = max(temp,alpha*temp);
  }
}

float finalOP(float *ip)
{
  float temp = 0.00000;
  for(int i=0;i<4;i++)
  {
    temp += ip[i]*weightsop[i];
  }
  temp += biasop;
  return temp;
}
```

VI. Discussion and Summary

In this project, we focused on addressing the frequency variation issue in the Hartley oscillator due to temperature changes. We initially observed that with an increase in temperature, the frequency of the oscillator increased, indicating a significant impact of temperature on circuit performance.

To tackle this issue, we developed a machine learning (ML) model that accurately predicted the output frequency of the oscillator based on temperature and voltage inputs. The ML model took into consideration the temperature coefficient of capacitance, which influenced the capacitance of the ceramic capacitor used in the oscillator's tank circuit.

The ML model exhibited remarkable accuracy, with an error percentage between 3 and 5 when compared to hardware values. The high coefficient of determination (r2 score) of 99.998 indicated an excellent fit between the predicted and actual frequency values. This demonstrated the reliability and effectiveness of the ML model in capturing the relationship between temperature, capacitance variation, and frequency output.

To further enhance the circuit's stability, we employed a self-adaptive approach. The ML model's predictions were used to determine the required adjustment in the capacitor to restore the desired original frequency. By comparing the predicted and desired frequencies, we dynamically adjusted the capacitance in the tank circuit, compensating for the temperature-induced frequency shift.

The self-adaptive mechanism offered several advantages, eliminating the need for manual calibration and enabling real-time adjustments based on temperature variations. This approach ensured consistent frequency output, maintaining circuit performance even in changing temperature conditions.

In summary, this project successfully addressed the frequency variation issue in the Hartley oscillator caused by temperature changes. Through the development of an ML model, we accurately predicted the output frequency by considering the temperature coefficient of capacitance. The ML model exhibited high accuracy, with an error percentage between 3 and 5 and an impressive r2 score of 99.998.

By utilizing the ML model's predictions, we implemented a self-adaptive approach that dynamically adjusted the capacitor in the tank circuit to maintain the desired original frequency. This approach offered an efficient and automated solution, ensuring frequency stability without the need for manual intervention or calibration.

Overall, the combination of the ML model and the self-adaptive mechanism provided an effective solution for temperature-induced frequency variations in the Hartley oscillator. This project highlights the potential of ML in addressing circuit performance issues and demonstrates the feasibility of self-adaptive techniques in maintaining desired frequencies in varying temperature conditions.

Acknowledgment

We would like to express our sincere appreciation and gratitude to the following individuals who have been instrumental in the successful completion of our project:

First and foremost, we would like to express our deepest gratitude to our project advisor, Dr. Zia Abbas. His invaluable guidance,

expertise, and continuous support throughout the project were crucial in shaping our research and ensuring its success. His insightful feedback, constructive criticism, and unwavering commitment to excellence have been truly inspiring.

We would like to extend our thanks to the teaching assistants, Amar Gwari, Neeraj Chanamolu, and Deekshith Akula, for their assistance, dedication, and valuable input throughout the project. Their contributions in the form of brainstorming sessions, discussions, and technical support were immensely helpful in overcoming challenges and refining our ideas.

We are grateful to Dr. Prasad Krishnan and Spandan Roy, esteemed faculty members of our department, for their encouragement, support, and valuable suggestions. Their expertise and academic guidance played a significant role in shaping our understanding of the subject matter and in refining our research methodology.

We would also like to acknowledge the contributions of Poornavati Mam and Gopalrao Sir, the lab in-charges. Their support in providing access to the necessary resources, laboratory facilities, and equipment was crucial to the smooth execution of our experiments and data collection.

VII. References

1) Smith, C. A. (1999). *Principles of Electronic Circuits*. Oxford University Press.
2) Sedra, A. S., & Smith, K. C. (2016). *Microelectronic Circuits*. Oxford University Press.
3) Rashid, M. H. (2018). *Power Electronics: Circuits, Devices, and Applications*. Pearson.
4) Gonzalez, G. T., & Wunsch, D. (2017). *Neural Network Design* (2nd Edition). PWS Publishing.
5) Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.
6) Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.