INTERNSHIP REPORT

Name: Tarun H

Year/Dept: III/CSE

College Name: Sri Sairam Institute of Technology

Guide Name: Dr. G. R. Kanagachidambaresan, Associate Professor, Department of Computer Science and Engineering

Internship Organization: Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai

Student Signature **Internship Guide Signature**

Offer Letter

10/17/21, 7:21 PM Sairam Institutions - TAP CELL Mail - Internship offer

YOGESH K <sit19cs094@sairamtap.edu.in>

Internship offer

Asso. Prof., CSE Vel Tech, Chennai <drgrkanagachidambaresan@veltech.edu.in> Mon, Sep 27, 2021 at 4:20 PM
To: rajalakshmi.cse@sairamit.edu.in
Cc: sit19cs032@sairamtap.edu.in, sit19cs118@sairamtap.edu.in, sit19cs116@sairamtap.edu.in, sit19cs094@sairamtap.edu.in, Sit19cs044@sairamtap.edu.in

Dear Prof,

Greetings, Thanks for your request. This is for your kind information that the following candidates are welcome to do 15 days internship in Expert systems lab under a DBT funded project on Aquaculture monitoring and machine learning. The offers do not have any financial commitment, however collaborative publication and copyrights will be done with both the students and faculty members involved in this project.

Mr. Balakumar C
9677247953
sit19cs032@sairamtap.edu.in
Mr. Laxminarayanan V
9789926846
sit19cs118@sairamtap.edu.in
Mr. Karthick M
sit19cs116@sairamtap.edu.in
9487856337
YOGESH K
sit19cs094@sairamtap.edu.in
9003275291
Mr. Tarun h
Sit19cs044@sairamtap.edu.in
7358009475

With thanks and regards
Dr. G. R. Kanagachidambaresan,
Associate Professor, CSE,
Vel Tech Rangarajan Dr Sagunthala R&D Institute of Science and Technology,
Avadi, Chennai.

Acknowledgement

I would like to thank Dr. K. Palanikumar Principal, Dr. B. Sreedevi Head of the Department Computer Science and Engineering and staff members of Sri Sairam Institute of Technology, Chennai for giving me the opportunity to have a practical exposure on research projects.

I deeply express my gratitude to my guide Dr. G. R. Kanagachidambaresan, Associate Professor, Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai and staffs at the institute for their patience and assistance during my internship. Through his guidance, I was able to gain real time industrial insights on full-scale projects. I thank him for his proffessional and compassionate approach towards us.

I also like to take this moment to thank and appreciate my fellow co-interns who were kind enough to share their knowledge and resources during the length of the program aiding ourselves in completing the projects

Table of Contents

Work Experience

Shrimp Aquaculture Project

The prawn Industry represents intensive farming systems capable of returning high yields of protein for human consumption. Prawn ponds are dynamic, complex systems. Their performance is affected by a complex mix of starting condition, management practices, and multiple environmental factors. For example, two ponds can have significantly different yields, and yet be located next to each other. In order to understand how pond dynamics is changing over time, the collection of on-farm data is considered as being critical by the industry.

The objective of this project is to design a working system that collects data from the dummy ponds and to grahically represent it and a set of predicted values for the forthcoming length of time. We deployed sensors for real-time monitoring of important water quality parameters. Alongside, it is also important to get some insight from the data (collected by sensors) and interactive visualisation tools that can be used for management decision making. Hence, we also explored the effectiveness of predictive analytics and plot-based visual analytics tools.

This real-time data was fed into the temporary google sheet database where a new row of values were appended every 3 seconds. We successfully implemented An API system using google cloud platform for the flow of data from google sheets to our python script

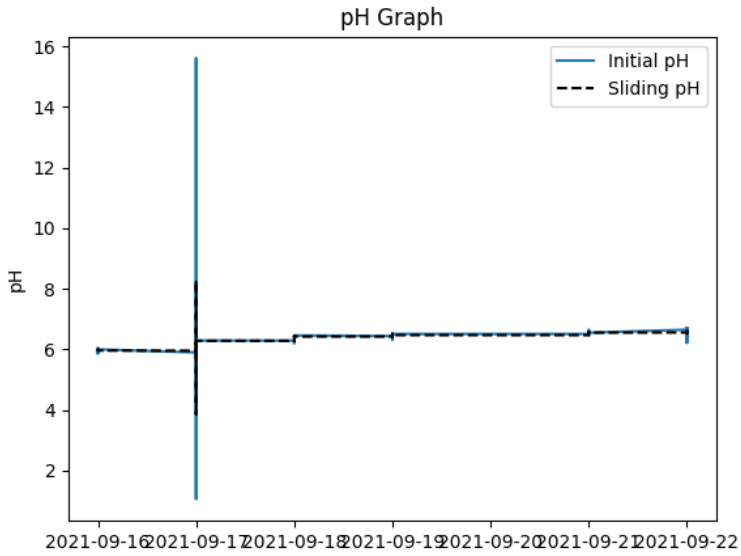
Our first task was to foolproof our data because the sensors can be faulty and is not always reliable for the entire duration , hence we deployed a sliding value algorithm which takes the average of N values from our database over S seconds of time to provide us with a sliding graph for the values Ph, Turbidity, DO, Temperature, Humidity

Following that we Apply a simple Linear Regression machine learning model to predict the future values using the past data values in the database as our training set.

Source code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file = pd.read_excel("sowmiya_ph.xlsx")
initial_pH = file["pH"]
date = file["Date"]
sliding_pH = []
sliding_value, temp = 100, 0
      for i in range(sliding_value):
          temp += initial_pH[i]
      temp /= sliding_value
      sliding_pH.append(temp)
      for i in range(1, len(initial_pH)):
          temp = 0
          if i+sliding_value > len(initial_pH):
              break
          c = 0
          while c < sliding_value:
              temp2 = i+c
              temp += initial_pH[temp2]
              c+=1
          temp /= sliding_value
          sliding_pH.append(temp)
      diff = len(initial_pH) - len(sliding_pH)
      for i in range(diff):
          sliding_pH.append(np.nan)
      file.insert(9, "SpH", sliding_pH)
      #print(file.head(5))
      FFT = np.array(sliding_pH, float)
      plt.plot(date,initial_pH,date,sliding_pH,"--k")
      plt.title("pH Graph")
      plt.ylabel("pH")
      plt.legend(['Initial pH', 'Sliding pH'])
      plt.show()

```

sliding plot

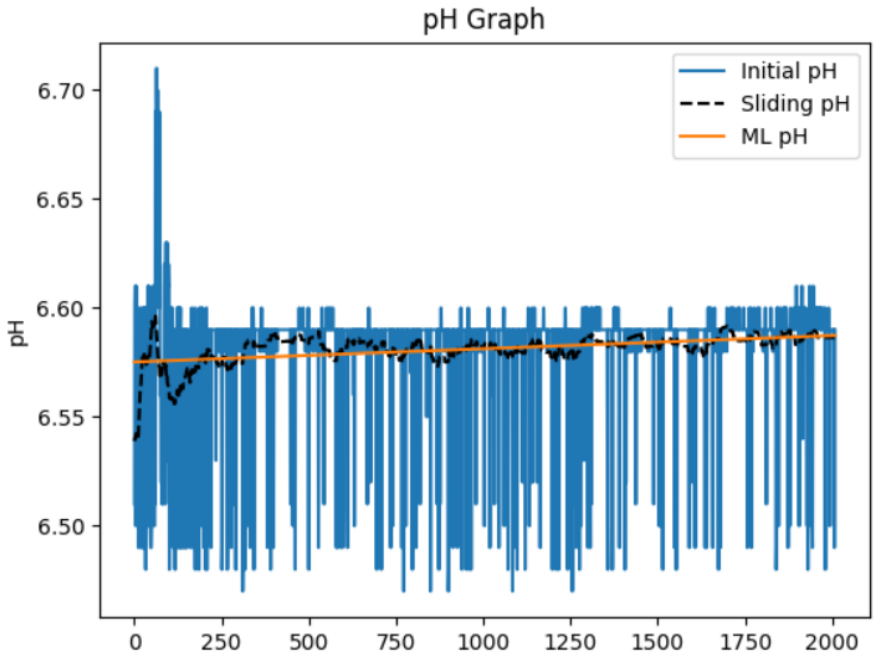


Later I trained a machine learning model to predict the pH/DO value for a given time, to compare it with the actual censor value to check if there is any anomaly or a fault in censor. This helps in getting accurate results. Linear Regression was the algorithm used here to achieve the output using *sklearn* package.

ML Model Predicting the pH Value:

```
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:\Users\Yogesh\Desktop\Internship\ML_Own.py ============
Training Test Splitted.
Model Training...
Model Predicting value for Id: 21
Predicted Value:  [7.15188679]
>>>
```

Final output



Supply Chain Management Application Project

Supply chain management is the management of the flow of goods and services and includes all processes that transform raw materials into final products. It involves the active streamlining of a business's supply-side activities to maximize customer value and gain a competitive advantage in the marketplace.

Typically, SCM attempts to centrally control or link the production, shipment, and distribution of a product. By managing the supply chain, companies can cut excess costs and deliver products to the consumer faster. This is done by keeping tighter control of internal inventories, internal production, distribution, sales, and the inventories of company vendors.

Our objective was to present a working UI program which displays the number of products manufactured by each sub-industry and classify it into defective and working sections. The program will have a report generate button to generate a report after every shift and send a automatic report mail to the designated address

Working and presentation

The UI for the application is made using the *tkinter* package for python. Various modules are created to undergo individual tasks. Firstly, there is a timer module which countdowns the time in decrements and notifies when the Shift has ended. Once the Shift has ended, an automatic

email is sent to respective authorities. The email consists of an excel file which contains the Shift details. The Automatic Mail sending module is made possible using the *smtplib* and *email* package available in python.

In case, the report of the shift is required in between the shift/before the shift ends, there is a Generate Report button. This button calls the module which is responsible for generating the report at the instance when it is clicked. The shift details are stored in an excel file using the *pandas* package. The excel file is named as the date and time at which the report was generated.

In order to get the live updates of the good and bad count from the Manufacturing Unit to the Assembly Unit, the values are published to a MQTT server online. In this case, we have used HiveMQ MQTT server. Using the *paho-mqtt* package in python, we would be able to call the necessary functions and publish the required values to the server. Simultaneously, the Assembly Unit would be receiving the values by subscribing to the respective topic in which it is getting published. Again, these values are displayed in the Assembly Unit application with *tkinter* being its base UI.

Publisher Source Code:

```
1. from tkinter import *
2. from tkinter import ttk
3. import tkinter as tk
4. import time
5. import random
6. import pandas as pd
7. import xlsxwriter
8. from datetime import date, datetime
9. from datetime import datetime
10. import getpass
11. import os
12. import smtplib
13. from email.message import EmailMessage
14. import paho.mqtt.client as paho
15. from paho import mqtt
16.
17. qwe = [0,0,0,'0']
18.
19. def on_connect(client, userdata, flags, rc, properties=None):
20.     print("Connection Success! Received with code %s." % rc)
21.
22. # with this callback you can see if your publish was successful
23. def on_publish(client, userdata, mid, properties=None):
24.     print("mid: " + str(mid))
25.
26. # print which topic was subscribed to
```

```
def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

#Sends Automatic Mail once the Shift gets over
def automaticEmail():
    sender_email= "mayoisnicee@gmail.com"
    sender_pass="Mayonnaise24"
    receiver_email="balakumarbk03@gmail.com"

    msg = EmailMessage()
    msg['Subject'] = "REPORT"
    msg['From'] = sender_email
    msg['To'] = receiver_email
    msg.set_content('Shift Report Has Been Attached...') #to be filled


    #User Name
    user = getpass.getuser()

    file_directory=["C:/Users/"+ user + "/Documents/" + qwe[3] +  ".xlsx"]

    for file in file_directory:
        with open(file, 'rb') as f:
            file_data=f.read()
            file_name=qwe[3]+'.xlsx'
            print("filename",file_name)
        msg.add_attachment(file_data, maintype='application',
        subtype='octet-stream', filename=file_name)


    with smtplib.SMTP_SSL("smtp.gmail.com", 465) as smtp:
        smtp.login(sender_email,sender_pass)
        smtp.send_message(msg)
    Label(root, text='Report Generated!', font = 'arial 15 bold').place(x=800,
   y=450)
    print("Mail Sent!")

def vals(count,good,bad):
    qwe[0] = count; qwe[1] = good; qwe[2] = bad

#Generates the current report when the Generate Report Button is clicked.
def generateReport(): #(t,number,name,count,good,bad):
    #Date Time
    now = datetime.now()
    dt_string = now.strftime("%B %d %Y,%H-%M-%S")
    qwe[3] = dt_string
    #User Name
    user = getpass.getuser()

    count_val,good_val,bad_val,shift_dt = [0],[0],[0],[0]

    count_val[0] = qwe[0]
```

```
    good_val[0] = qwe[1]
    bad_val[0] = qwe[2]
    shift_dt[0] = dt_string

    df = pd.DataFrame({'Shift Time':shift_dt[0],
                       'Shift Duration(secs)':t.get(),
                       'Shift Number':number.get(),
                       'Supervisor':name.get(),
                       'Production Count':count_val,
                       'Good Products':good_val,
                       'Defected Products':bad_val})

    writer = pd.ExcelWriter(r"C:\Users\\"+ user + "\Documents\\" + dt_string
  +  ".xlsx")

    # Convert the dataframe to an XlsxWriter Excel object.
    df.to_excel(writer, sheet_name='Sheet1',index = False)
    workbook  = writer.book
    worksheet = writer.sheets['Sheet1']
    worksheet.set_column('A:G',25 )
          writer.save()
          print("Report Generated Successfully on "+dt_string)


      #Shift Runtime
      def shiftRun():
          l = []
          time_sec = int(t.get())
          count = 0; good = 0; bad = 0
          while time_sec:
              mins, secs = divmod(time_sec, 60)
              timeformat = '{:02d}:{:02d}'.format(mins, secs)
              #print(timeformat, end='\r')
              l.append(timeformat)
              #print(l[-1])
              count += 1
              if random.randint(0,1):
                  good += 1
              else:
                  bad += 1
              Label(root, text=l[-1], font = 'arial 15 bold').place(x=800,
  y=50)
              Label(root, text="Production Count: "+str(count), font = 'arial
  20 bold').place(x=20, y=360)
              Label(root, text="Good: "+str(good), font = 'arial 20
  bold').place(x=20, y=400)
              Label(root, text="Defects: "+str(bad), font = 'arial 20
  bold').place(x=20, y=440)
              client.publish("test/Bosch_Good",payload=str(good),qos=0)
              time.sleep(1)
              time_sec -= 1
              root.update()

              vals(count,good,bad)
```

```
131.           Label(root, text='Shift Ended!', font = 'arial 15
   bold').place(x=800, y=80)
132.           generateReport()
133.           automaticEmail()
134.
135.       try:
136.           client = paho.Client(client_id="", userdata=None,
   protocol=paho.MQTTv5)
137.           client.on_connect = on_connect
138.
139.           # enable TLS
140.           client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
141.
142.           client.username_pw_set("industry4.0", "Qwerty@123")
143.           client.connect("1d1587d792b6466fa8f2db432d9d3db2.s1.eu.hivemq.cloud"
   , 8883)
144.
145.           client.on_message = on_message
146.           client.on_subscribe = on_subscribe
147.           client.on_publish = on_publish
148.
149.           root = Tk()
150.           root.geometry('1000x500')
151.           #production_name = input("Production Name(Eg:
   Bulb/Indicator/Housing/Electrical): ")
152.           Label(root, text = 'Indicator Production' , font = 'arial 20
   bold').pack()
153.
154.           #Starting Protocols
155.           Label(root, font ='arial 15 bold', text = 'Set Time').place(x = 20
   ,y = 50)
156.           t = Entry(root,width=15)
157.           t.place(x=180, y=55)
158.           Label(root, font ='arial 15 bold', text = 'Shift Number').place(x =
   20 ,y = 80)
159.           number = Entry(root,width=15)
160.           number.place(x=180, y=85)
161.           Label(root, font ='arial 15 bold', text = 'Supervisor').place(x = 20
   ,y = 110)
162.           name = Entry(root,width=15)
163.           name.place(x=180,y=115)
164.
165.           #Start Button
166.           Button(root, text='START', bd ='5', command = shiftRun, font =
   'arial 10 bold').place(x=80, y=160)
167.
168.           #Generate Report Button
169.           Button(root, text='Generate Report', bd ='5', command =
   generateReport, font = 'arial 10 bold').place(x=800, y=400)
170.
171.
172.           #Quit Button
173.           ttk.Button(root, text="Quit",
   command=root.destroy).place(x=450,y=450)
174.           root.mainloop()
```

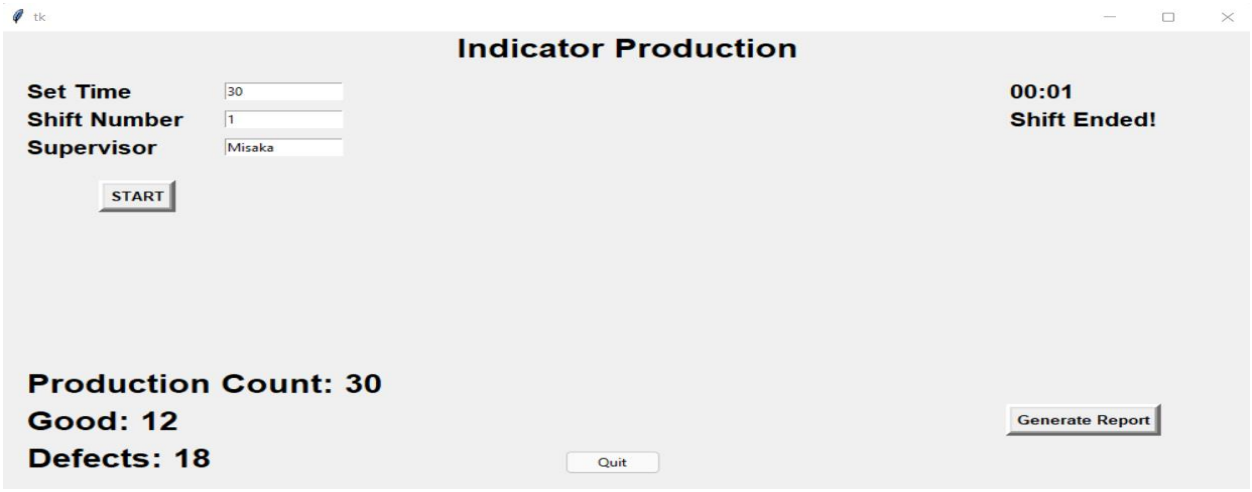
```
175.          print("Press 'Ctrl+C' to close the application.")
176.          client.loop_forever()
177.
178.
179.      except KeyboardInterrupt:
180.              print("Byeeee!!")
181.              pass
```

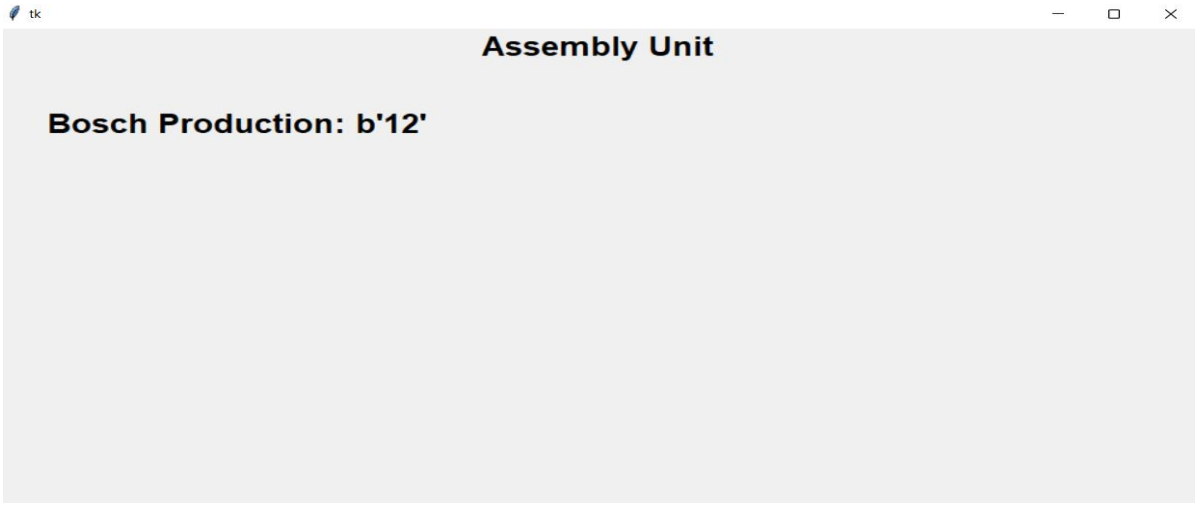
Subscriber Source Code:

```
1.  import time
2.  import paho.mqtt.client as paho
3.  from paho import mqtt
4.  from tkinter import *
5.  from tkinter import ttk
6.  import tkinter as tk
7.
8.  def on_connect(client, userdata, flags, rc, properties=None):
9.      print("Connection Success! Received with code %s." % rc)
10.
11. # with this callback you can see if your publish was successful
12. def on_publish(client, userdata, mid, properties=None):
13.     print("mid: " + str(mid))
14.
15. # print which topic was subscribed to
16. def on_subscribe(client, userdata, mid, granted_qos, properties=None):
17.     print("Subscribed: " + str(mid) + " " + str(granted_qos))
18.
19. # print message, useful for checking if it was successful
20. def on_message(client, userdata, msg):
21.     print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
22.     topic = msg.topic
23.     payload = msg.payload
24.     if 'Crompton_Good' in topic:
25.         #print("Update Crompton Label: ",payload)
26.         Label(root, text = 'Crompton Production: '+str(payload) , font =
    'arial 20 bold').place(x=40,y=200)
27.
28.     elif 'Bosch_Good' in topic:
29.         #print("Update Bosch Label: ",payload)
30.         Label(root, text = 'Bosch Production: '+str(payload), font = 'arial 20
    bold').place(x=40,y=80)
31.     root.update()
32.
33. client = paho.Client(client_id="", userdata=None, protocol=paho.MQTTv5)
34. client.on_connect = on_connect
35.
36. # enable TLS for secure connection
37. client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
38. # set username and password
39. client.username_pw_set("industry4.0", "Qwerty@123")
40. # connect to HiveMQ Cloud on port 8883 (default for MQTT)
```

```
client.connect("1d1587d792b6466fa8f2db432d9d3db2.s1.eu.hivemq.cloud", 8883)

# setting callbacks, use separate functions like above for better visibility
client.on_subscribe = on_subscribe
client.on_message = on_message
client.on_publish = on_publish

# subscribe to all topics of encyclopedia by using the wildcard "#"
client.subscribe("sensor1/#", qos=0)

root = Tk()
root.geometry('1000x500')
Label(root, text = 'Assembly Unit' , font = 'arial 20 bold').pack()

client.loop_forever()
```

Output





Rice Leaf Disease Classification

Rice disease has serious negative effects on crop yield, and the correct diagnosis of rice diseases is the key to avoid these effects. However, the existing disease diagnosis methods for rice are neither accurate nor efficient, and special equipment is often required. In this study, an automatic diagnosis method was developed and implemented in a smartphone app.

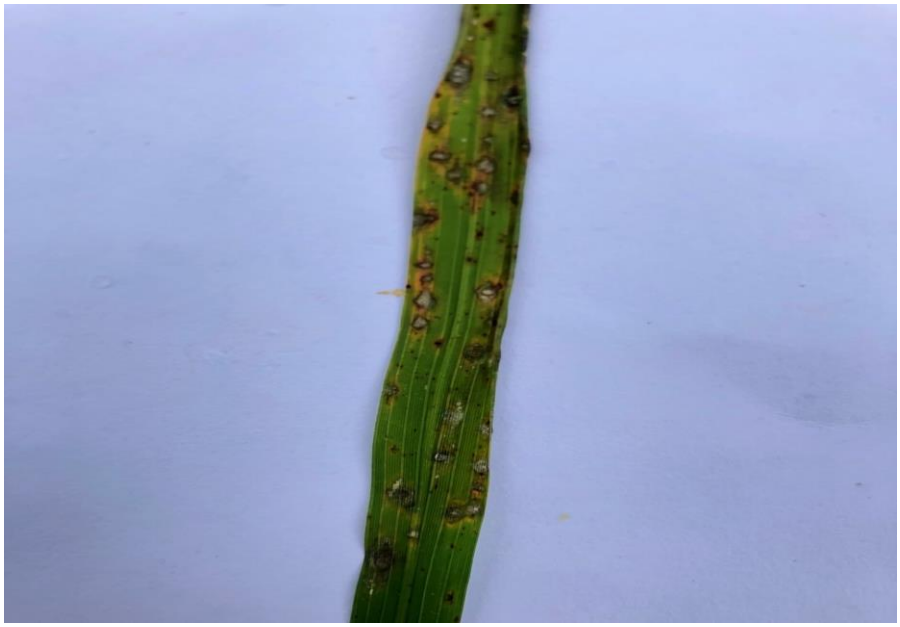
The front-end objective of this project was to develop a mobile app which will seamlessly work with our database and processing server. Its functions include taking a photograph, sending the photograph to our server and retreiving and displaying the predicted value from our script.

Backend includes integration of our app with the server and database, the database stores every photograph taken by the app and runs the script for the photograph as the argument. The python script predicts the disease of the leaf and returns it to the database which is retrieved and displayed to the user by the app using our API credentials

The app was developed using MIT App Inventor as it was a open-source developing platform with drag and drop features. The choice of temporary database was google drive since we were already familiar with the google drive API. Database option can be scaled to Firebase or AWS databases for future implementation. The script employs a basic combination of StackingCVClassifier & KneighborsClassifier models for accurate prediction of the leaf disease.

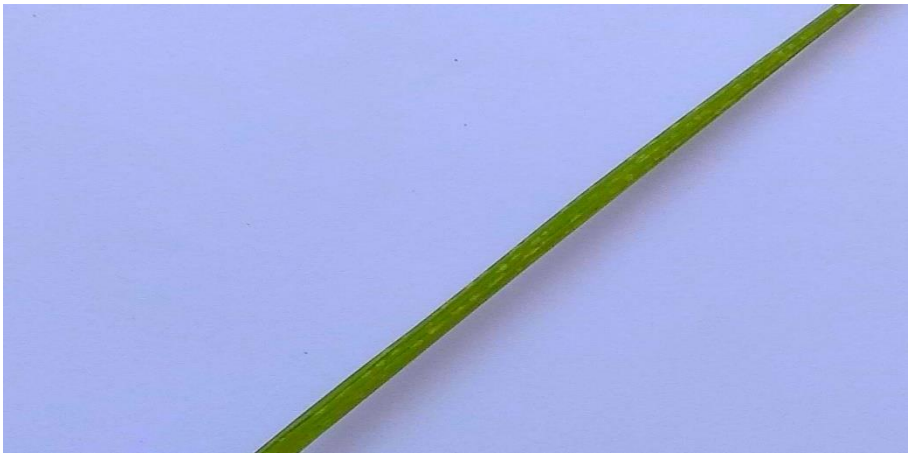
The dataset was sourced from kaggle , it includes three of the major rice leaf diseases and a healthy leave set, for our convinience I had split the dataset into training and testing data set

- Leaf blast



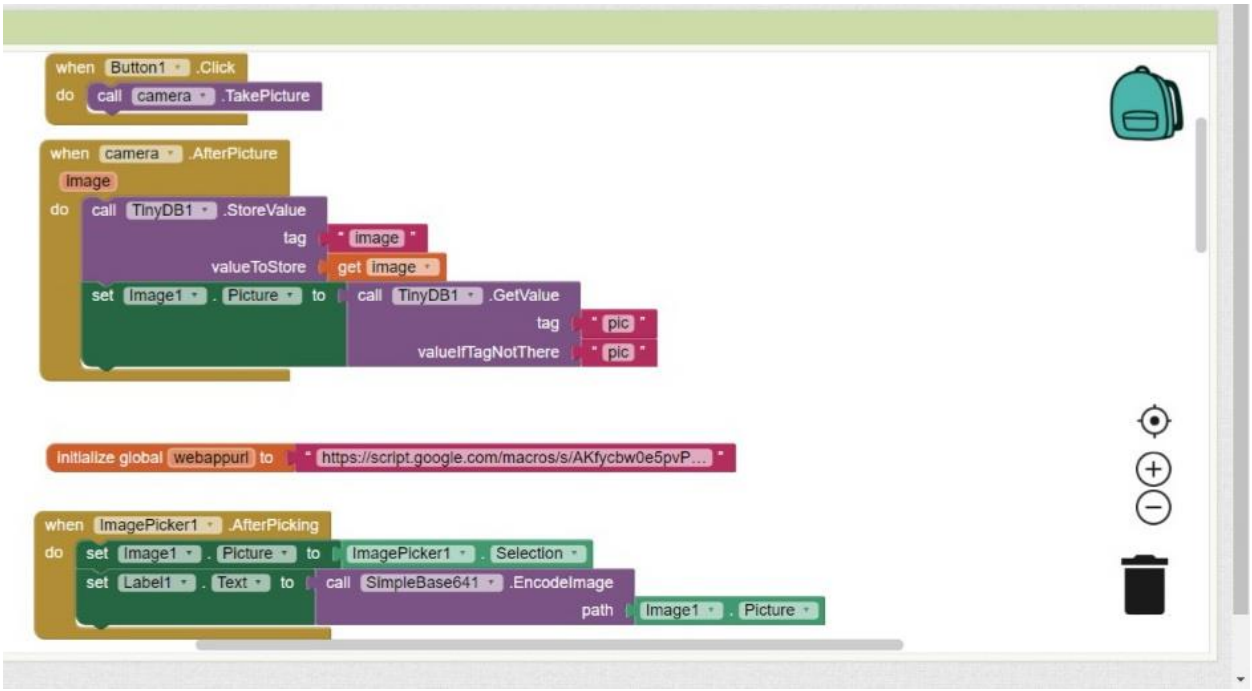
- Blight

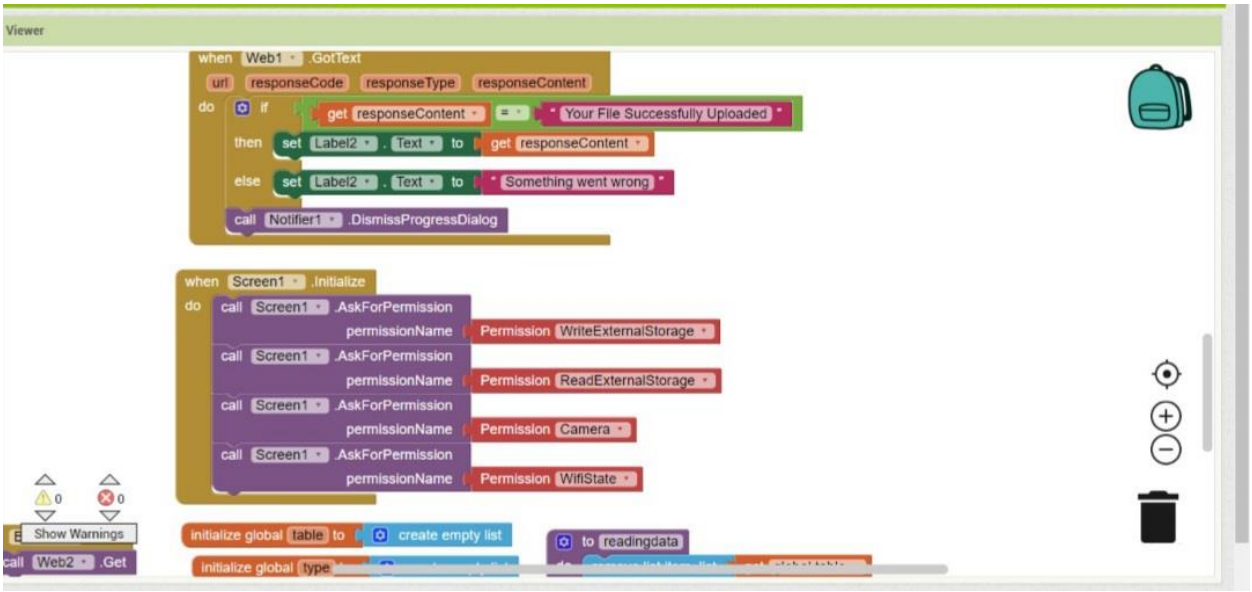
- Hispa



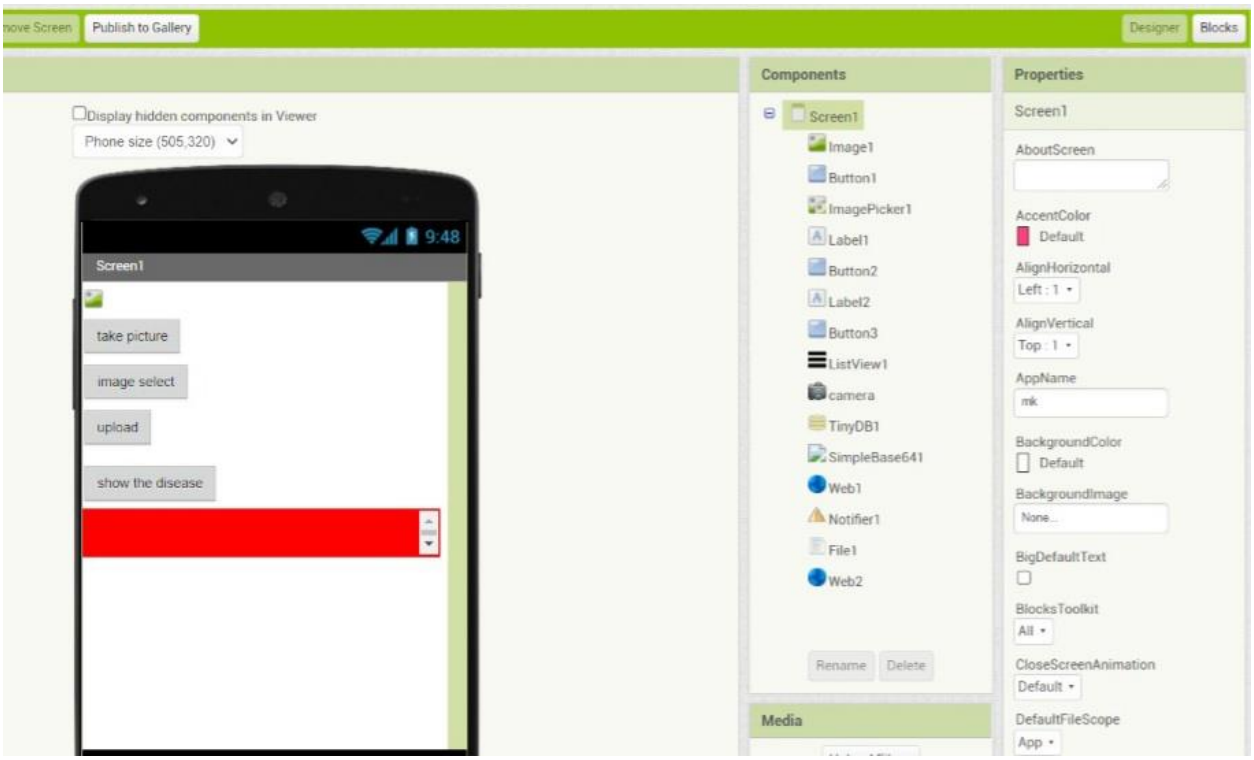
- healthy

Mobile App Architecture





Mobile app screenshot



Source code

```
import os

import pandas as pd

for dirname, _, filenames in
os.walk('/home/mayo/Downloads/labelled/Labelled'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


import operator
import os
import pickle
import sys

import numpy as np
import PIL
import tensorflow as tf
from keras.preprocessing import image
from keras.preprocessing.image import (ImageDataGenerator, img_to_array,
                                       load_img)
from mlxtend.classifier import StackingCVClassifier
```

```
from PIL import Image
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from tensorflow import keras

print("[INFO] Loading Training dataset images...")
DIRECTORY = "/home/mayo/Downloads/labelled/Labelled"
CATEGORIES = ["BrownSpot", "Healthy", "Hispa", "LeafBlast"]
data = []
clas = []
print("[INFO] Preprocessing...")
for category in CATEGORIES:
    print(category)
    path = os.path.join(DIRECTORY, category)
    print(path)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        img = image.load_img(img_path, target_size=(128, 128))
        img = image.img_to_array(img)
        img = img / 255
        data.append(img)
        clas.append(category)

    print("[INFO] Features Extraction completed")

x_train = np.array(data)
x_train = x_train.reshape(len(x_train), -1)
y_train = np.array(clas)

print("[INFO] Training Stacking Classifier")
clf_knn = KNeighborsClassifier(n_neighbors=4)
clf_svm = SVC(C=50, degree=1, gamma="auto", kernel="rbf", probability=True)
classifier = StackingCVClassifier(classifiers=[clf_knn, clf_svm],
                                  shuffle=False,
                                  use_probas=True,
                                  cv=5,

meta_classifier=SVC(probability=True))
classifier.fit(x_train, y_train)
with open('./Stack.model', 'wb') as f:
    pickle.dump(classifier, f)
print("[INFO] Training Stacking Classifier created successfully..!")



import numpy as np
from keras.preprocessing import image

#Prediction Code
test_img =[]
```

```
testimage = "/home/mayo/Downloads/699.jpg"
img3 = image.load_img(testimage, target_size=(128, 128))
img3 = image.img_to_array(img3)
img3 = img3 / 255
test_img.append(img3)
Xtest = np.array(test_img)
Xtest = Xtest.reshape(len(Xtest), -1)

model = open('./Stack.model', 'rb')
clf_cnn = pickle.load(model)
predicted = clf_cnn.predict(Xtest)
result = predicted[0]
print("Prediction Result is :",result)
```

Sample output

```
Prediction Result is : Tungro
```

Industry 4.0 Prognostic Module

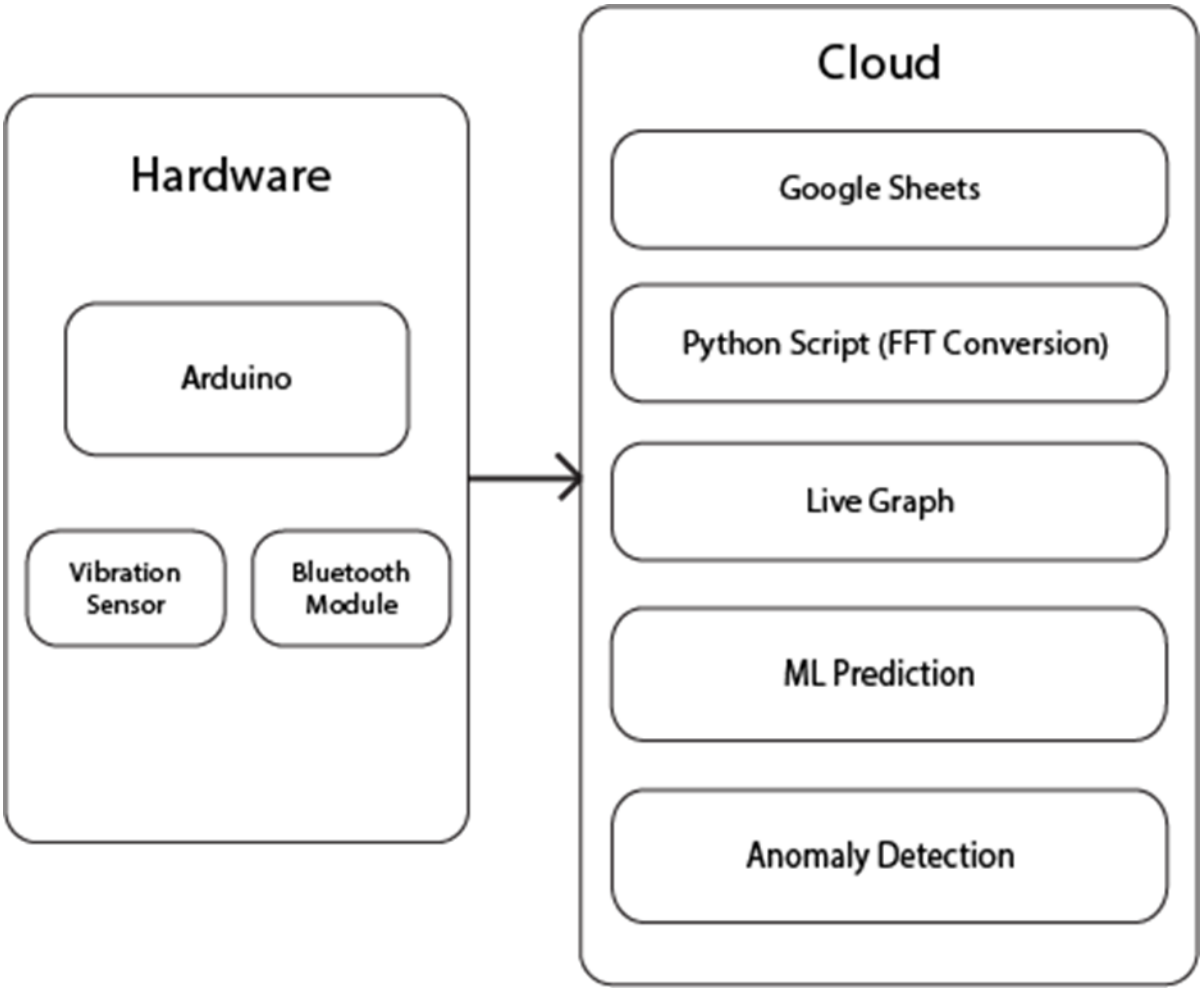
Prognostic maintenance has been a goal in many industries for many years. Now it's on the horizon. Prognostic maintenance (also known as condition-based maintenance, predictive maintenance, or simply prognostics1) is the ability to know the condition of equipment, and to plan and perform maintenance accordingly before a critical failure.

Data-driven prognostics require massive sensor data on the condition of the system at hand and on the performance of the system. This data is continuously collected and run through machine learning algorithms in hopes of detecting correlations and using these to make prognoses of future failures.

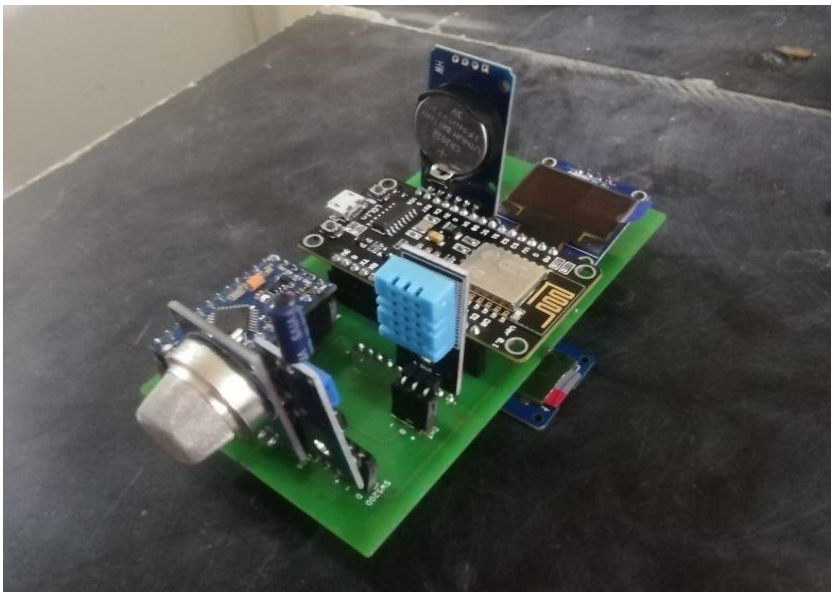
The objective of this project was to develop a sensor module that can detect the vibration signals from an equipment and pass it through the onboard FTT Algorithm that we implemented to get a frequency plot for us to infer and find any defect in the Industrial Equipment

The sensor module consisted of a Vibration sensor and Bluetooth Node both interfaced with the Arduino Nano Board

Architecture Diagram



Module



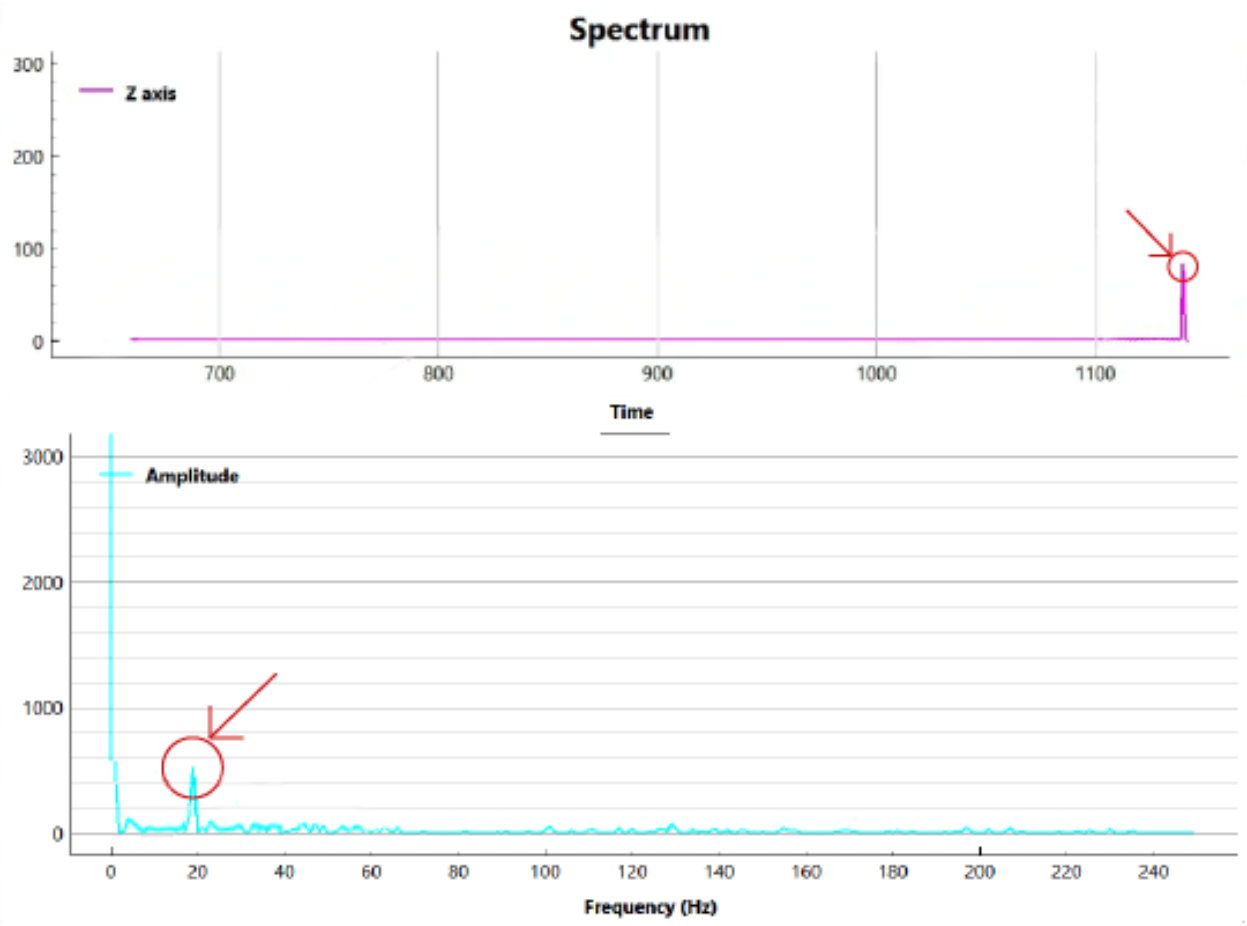
Source code

```
1. # -*- coding: utf-8 -*-
2. import sys
3. import glob
4. import serial
5. import pyqtgraph as pg
6. from pyqtgraph.Qt import QtCore, QtGui
7. import numpy as np
8.
9.
10.      #port_num = input("Enter the port number: ")
11.      usb = 'COM7'
12.
13.      #usb = input('Select the serial port: ')
14.
15.      arduinoData = serial.Serial(usb, 9600)  # 115200
16.      arduinoData.flush()
17.
18.      class ReadLine:
19.          def __init__(self, s):
20.              self.buf = bytearray()
21.              self.s = s
22.
23.          def readline(self):
24.              i = self.buf.find(b"\n")
25.              if i >= 0:
26.                  r = self.buf[:i+1]
27.                  self.buf = self.buf[i+1:]
28.                  return r
29.              while True:
30.                  i = max(1, min(2048, self.s.in_waiting))
31.                  data = self.s.read(i)
32.                  i = data.find(b"\n")
33.                  if i >= 0:
34.                      r = self.buf + data[:i+1]
35.                      self.buf[0:] = data[i+1:]
36.                      return r
37.                  else:
38.                      self.buf.extend(data)
39.
40.      acelz = []
41.
42.      freq = 500                  # 1/T   -> T is the arduino delay in seconds
43.      guarda = 500                 # Buffer for the FFT, higher value equals
   better resolution
44.      r = range(0, int(freq/2+1), int(freq/guarda))
```

```
45.
46.       win = pg.GraphicsWindow()
47.       win.setWindowTitle('Spectrum')
48.       pg.setConfigOption('foreground', 'w')
49.
50.       p1 = win.addPlot(
51.           title="<span style='color: #ffffff; font-weight: bold; font-
   size:20px'>Spectrum</span>")
52.       linha1 = pg.mkPen((0, 255, 0), width=2)  # style=QtCore.Qt.DashLine)
53.       linha2 = pg.mkPen((0, 0, 255), width=2)  # style=QtCore.Qt.DashLine)
54.       linha3 = pg.mkPen((255, 0, 0), width=2)  # style=QtCore.Qt.DashLine)
55.       p1.addLegend(offset=(10, 5))
56.
57.       curve1 = p1.plot(acelz,
58.                        pen=linha1,
59.                        name="<span style='color: #ffffff; font-weight: bold;
   font-size: 12px'>Z axis</span>")
60.
61.       p1.setRange(yRange=[-4, 300])
62.       p1.setLabel('bottom',
63.                   text="<span style='color: #ffffff; font-weight: bold; font-
   size: 12px'>Time</span>")
64.       p1.showGrid(x=True, y=False)
65.
66.       win.nextRow()
67.       p2 = win.addPlot()
68.       linha4 = pg.mkPen((255, 0, 0), width=2)
69.       p2.addLegend(offset=(10, 5))
70.
71.       curve2 = p2.plot(acelz,
72.                        pen=linha4,
73.                        name="<span style='color: #ffffff; font-weight: bold;
   font-size: 12px'>Amplitude</span>")
74.
75.       p2.setRange(yRange=[0,200000], xRange=[0, int(freq/2)])
76.       p2.setLabel('bottom',
77.                   text="<span style='color: #ffffff; font-weight: bold; font-
   size: 12px'>Frequency (Hz)</span>")
78.       p2.showGrid(x=False, y=True)
79.
80.       i = 0
81.
82.       def update():
83.           global i
84.           frequencia = np.fft.fftfreq(guarda, d=1/freq)
85.           '''if frequencia > 150000:
86.               print("EXCEEDING LIMIT")
87.               serial.write(40)'''
```

```
88.         try:
89.
90.             arduinoString = ReadLine(arduinoData)
91.             dataArray = arduinoString.readline().decode("utf-8").split(',')
92.
93.             acelerometroz = int(dataArray[0])/8 # 16384, 8192, 4096, 2048
    for accelerometer set 2, 4, 8, 16g
94.
95.             acelz.append(acelerometroz)
96.
97.             #np.savetxt("Accelerometer-FFT---Real-time\Data\Data.csv",
    acelz, delimiter=",")
98.
99.             if i > guarda:
100.                try:
101.                    data = np.fft.fft(acelz[-guarda:])
102.                except IOError:
103.                    pass
104.                curve2.setData(frequencia[:int(guarda/2)],
    abs(np.real(data[:int(guarda/2)]))**2)
105.
106.            curve1.setData(acelz)
107.            i += 1
108.            curve1.setPos(i, 0)
109.            if i > guarda:
110.                del acelz[0:1]
111.
112.        except ValueError:
113.            pass
114.
115.    timer = pg.QtCore.QTimer()
116.    timer.timeout.connect(update)
117.    timer.start(0)
118.
119.    '''if __name__ == '__main__':
120.        import sys
121.        if (sys.flags.interactive != 1) or not hasattr(QtCore,
    'PYQT_VERSION'):
122.            QtGui.QApplication.instance().exec()'''
```

Live FTT Graph



The prototype for this project was placed in a water jet cutting machine from which the above graph was obtained. In the above the graph, the arrow mark present under the spectrum denotes the peak at which the water comes in contact with the material from the water jet. Subsequently we are able to observe the drastic change in the graph of the frequency, which is its respective FFT converted graph.

Learning

During the course of the internship I had the opportunity to meet a lot of exciting and skillfull people who motivated me to learn more . The IOT laboratory and staff and fellow interns and junior research staffs imparted me with knowledge on the field as IOT was completely new to me.

I also learnt a hefty amount of intricacies of the Python programming language, Machine Learning and its concepts, Data Manupulation and Analysis. I was introduced to the data science aspect of python with the help of various scientific modules and communication Protocols like

- Numpy
- Pandas
- Sklearn
- Gspread
- Pyplot
- Keras
- Tensorflow
- Serial
- Pyqt
- MQTT
- SMTP
- Tkinter
- Scikit-learn

I had the opportunity to work with arduino boards for the first time which grew a lot on me. The working of the boards and their seamless interaction with each other fascinated me. Lastly I gained a plethora of working experience on the LINUX environment. Applied the uses of linux ports, kernal, terminal and its simple interfacing with sensors and boards.

Conclusively , I had a great experience and a very brief introductory and application oriented learning curve to the field of IOT which I enjoyed very much and hope to pursue this interest further in my career.