

## Beta.py

```
1  import time
2  import threading
3  import RPi.GPIO as GPIO
4  import Adafruit_ADS1x15
5  from PyQt5.QtWidgets import *
6  from PyQt5 import QtGui
7  from PyQt5.QtCore import QRect, Qt, QTimer, QSize
8  from PyQt5 import QtCore, QtWidgets
9  from PyQt5.QtGui import QPixmap, QIcon
10 import sys
11 import Adafruit_DHT
12 from matplotlib.backends.backend_qt5agg import FigureCanvas
13 from matplotlib.figure import Figure
14 from matplotlib.backends.backend_qt5agg import (NavigationToolbar2QT as NavigationToolbar)
15 import datetime
16 import serial
17 import csv
18
19
20
21 adc = Adafruit_ADS1x15.ADS1115()
22
23 GAIN = 4
24 count = 50
25 maxVal = 0
26 amp = 0.001
27 prevAmp = 0.001
28
29 global sourceState
30 global maxTime
31 global maxCurrent
32
33 sourceState = 0
34 maxTime = 30
35 maxcurrent = 15
36
37 global currentArray
38 global currentTime
39 global timeArray
40 currentArray = []
41 timeArray = []
42
43 global stop_threads
44
45 global myFont
46 myFont = "Linux Biolinum O"
47 global lineList
48 global currentFontItem
49 currentFontItem = 11
50
51 global tempUnit
52 tempUnit = "C"
53
54 global timeFormat
55 timeFormat = "%I:%M:%S %p"
56 global currentTimeItem
57 currentTimeItem = 0
```

```
58
59 global dateFormat
60 dateFormat = "%a %d-%m-%Y"
61 global currentDateItem
62 currentDateItem = 0
63
64 global tripTime
65 tripTime = ""
66
67 GPIO.setwarnings(False)
68 GPIO.setmode(GPIO.BCM)
69 GPIO.setup(21, GPIO.OUT)
70 GPIO.output(21, GPIO.LOW)
71 GPIO.setup(26, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
72
73 class Window(QMainWindow):
74
75
76     def __init__(self):
77         super().__init__()
78
79         global myFont
80         global timeFormat
81
82         title = "Relay Test Unit"
83         left = 2
84         top = 30
85         width = 796
86         height = 450
87
88         self.setWindowTitle(title)
89         self.setGeometry(left, top, width, height)
90         self.setFixedSize(width, height)
91
92         wid = QtWidgets.QWidget(self)
93         self.setCentralWidget(wid)
94         self.setStyleSheet("background-color: white;")
95
96         mainVBox = QVBoxLayout()
97         topBox = QHBoxLayout()
98         button1Box = QHBoxLayout()
99         button2Box = QHBoxLayout()
100         bottomBox = QHBoxLayout()
101
102         self.labelTemp = QLabel(self)
103         self.labelTemp.setAlignment(Qt.AlignLeft)
104         self.labelTemp.setAlignment(Qt.AlignTop)
105         self.labelTemp.setFixedHeight(40)
106         self.labelTemp.setText("Temp: 36 °C")
107         self.labelTemp.setFont(QtGui.QFont(myFont))
108         self.labelTime = QLabel(self)
109         self.labelTime.setAlignment(Qt.AlignRight)
110         self.labelTime.setFixedHeight(40)
111         self.labelTime.setText("Time")
112         self.labelTime.setFont(QtGui.QFont(myFont))
113
114         self.btn1 = QPushButton()
115         self.btn1.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
116         self.btn1.setFixedSize(110,100)
117         self.btn1.setText("MCB Test")
118         self.btn1.setFont(QtGui.QFont(myFont, 9))
```

```
119 self.btn2 = QPushButton()
120 self.btn2.setStyleSheet(Qt.ToolButtonTextUnderIcon)
121 self.btn2.setFixedSize(110,100)
122 self.btn2.setText("OCR Test")
123 self.btn2.setFont(QtGui.QFont(myFont, 9))
124 self.btn3 = QPushButton()
125 self.btn3.setStyleSheet(Qt.ToolButtonTextUnderIcon)
126 self.btn3.setFixedSize(110,100)
127 self.btn3.setText("Logged Data")
128 self.btn3.setFont(QtGui.QFont(myFont, 9))
129 self.btn4 = QPushButton()
130 self.btn4.setStyleSheet(Qt.ToolButtonTextUnderIcon)
131 self.btn4.setFixedSize(110,100)
132 self.btn4.setText("Settings")
133 self.btn4.setFont(QtGui.QFont(myFont, 9))
134 self.btn5 = QPushButton()
135 self.btn5.setStyleSheet(Qt.ToolButtonTextUnderIcon)
136 self.btn5.setFixedSize(110,100)
137 self.btn5.setText("How To Use")
138 self.btn5.setFont(QtGui.QFont(myFont, 9))
139 self.btn6 = QPushButton()
140 self.btn6.setStyleSheet(Qt.ToolButtonTextUnderIcon)
141 self.btn6.setFixedSize(110,100)
142 self.btn6.setText("About")
143 self.btn6.setFont(QtGui.QFont(myFont, 9))
144 self.btn7 = QPushButton()
145 self.btn7.setFixedSize(60,55)
146 pixmapMCB = QPixmap("MCB.jpg")
147 pixmapcurrentRelay = QPixmap("Current Relay.jpg")
148 pixmapVoltRelay = QPixmap("Voltage Relay.jpg")
149
150 pixmapEmpty = QPixmap("Empty.jpg")
151
152 pixmapDataLog = QPixmap("DataLog.jpg")
153 pixmapSettings = QPixmap("Settings.jpg")
154 pixmapUManual = QPixmap("User Manual.jpg")
155 pixmapAbout = QPixmap("About.jpg")
156 pixmapClose = QPixmap("Close.png")
157
158 self.btn1.setIcon(QIcon(pixmapMCB))
159 self.btn1.setIconSize(QSize(106,80))
160 self.btn2.setIcon(QIcon(pixmapcurrentRelay))
161 self.btn2.setIconSize(QSize(106,80))
162 self.btn3.setIcon(QIcon(pixmapDataLog))
163 self.btn3.setIconSize(QSize(106,80))
164 self.btn4.setIcon(QIcon(pixmapSettings))
165 self.btn4.setIconSize(QSize(106,80))
166 self.btn5.setIcon(QIcon(pixmapUManual))
167 self.btn5.setIconSize(QSize(106,80))
168 self.btn6.setIcon(QIcon(pixmapAbout))
169 self.btn6.setIconSize(QSize(106,80))
170 self.btn7.setIcon(QIcon(pixmapClose))
171 self.btn7.setIconSize(QSize(50,50))
172
173 self.btn1.clicked.connect(self.MCBTest)
174 self.btn2.clicked.connect(self.relayTest)
175 self.btn4.clicked.connect(self.settings)
176 self.btn5.clicked.connect(self.howToUse)
177 self.btn6.clicked.connect(self.about)
178 self.btn7.clicked.connect(self.closeFunction)
179
```

```
180         topBox.addWidget(self.labelTemp)
181         topBox.addWidget(self.labelTime)
182
183         button1Box.addWidget(self.btn1)
184         button1Box.addWidget(self.btn2)
185         button1Box.addWidget(self.btn3)
186         button2Box.addWidget(self.btn4)
187         button2Box.addWidget(self.btn5)
188         button2Box.addWidget(self.btn6)
189         bottomBox.addWidget(self.btn7)
190
191         mainVBox.addLayout(topBox)
192         mainVBox.addLayout(button1Box)
193         mainVBox.addLayout(button2Box)
194         mainVBox.addLayout(bottomBox)
195
196         clockTimer = QTimer(self)
197         clockTimer.timeout.connect(self.showTime)
198         clockTimer.start(1000)
199         self.showTime()
200
201
202
203
204
205
206
207
208         wid.setLayout(mainVBox)
209         self.show()
210
211
212     def showTime(self):
213         now = datetime.datetime.now()
214         now = now.strftime(dateFormat+"\n"+timeFormat)
215         self.labelTime.setText(now)
216
217     def MCBTest(self):
218         self.w = mcbWindow()
219         self.w.show()
220
221     def relayTest(self):
222         self.w = relayWindow()
223         self.w.show()
224
225     def settings(self):
226         self.w = settingsWindow()
227         self.w.show()
228         self.close()
229
230     def howToUse(self):
231         self.w = howtoWindow()
232         self.w.show()
233
234     def about(self):
235         self.w = aboutWindow()
236         self.w.show()
237
238     def closeFunction(self):
239         self.close()
240
```

```

241 class mcbWindow(QMainWindow):
242     global sourceState
243     global maxTime
244     global maxCurrent
245     global myFont
246
247     global timeFormat
248     global dateFormat
249
250
251     sourceState = 0
252     maxTime = 30
253     maxcurrent = 15
254     def __init__(self):
255         super().__init__()
256         self.setWindowTitle("MCB Test")
257         left = 2
258         top = 30
259         width = 796
260         height = 450
261         self.setGeometry(left, top, width, height)
262         self.setFixedSize(width, height)
263
264         wid = QtWidgets.QWidget(self)
265         self.setCentralWidget(wid)
266         self.setStyleSheet("background-color: white;")
267
268         mainVBox = QVBoxLayout()
269         topBox = QHBoxLayout()
270         bottomBox = QHBoxLayout()
271         graphVBox = QVBoxLayout()
272         settingsVBox = QVBoxLayout()
273         buttonHBox = QHBoxLayout()
274
275         #Designing TopBox
276         self.labelTemp = QLabel(self)
277         self.labelTemp.setAlignment(Qt.AlignLeft)
278         self.labelTemp.setAlignment(Qt.AlignTop)
279         self.labelTemp.setFixedHeight(40)
280         self.labelTemp.setText("Temp: 36 °C")
281         self.labelTemp.setFont(QtGui.QFont(myFont))
282         self.labelTime = QLabel(self)
283         self.labelTime.setAlignment(Qt.AlignRight)
284         self.labelTime.setFixedHeight(40)
285         self.labelTime.setText("Time")
286         self.labelTime.setFont(QtGui.QFont(myFont))
287
288         topBox.addWidget(self.labelTemp)
289         topBox.addWidget(self.labelTime)
290
291         clockTimer = QTimer(self)
292         clockTimer.timeout.connect(self.showTime)
293         clockTimer.start(1000)
294         self.showTime()
295
296         #Designing Graph anf Buttons
297         self.plotMCB = MatplotlibWidget()
298         self.plotMCB.setFixedSize(550, 350)
299         graphVBox.addWidget(self.plotMCB)
300
301         self.backButton = QPushButton()

```

```
302 self.backButton.setFixedSize(60,30)
303 self.backButton.clicked.connect(self.closeFunction)
304 self.onButton = QPushButton()
305 self.onButton.setFixedSize(60,30)
306 self.onButton.setText("On")
307 self.onButton.setFont(QtGui.QFont(myFont))
308 self.onButton.clicked.connect(self.onButtonFunction)
309 self.onTimeButton = QPushButton()
310 self.onTimeButton.setFixedSize(90,30)
311 self.onTimeButton.setText("On+Time")
312 self.onTimeButton.setFont(QtGui.QFont(myFont))
313 self.onTimeButton.clicked.connect(self.onTimeButtonFunction)
314 self.offButton = QPushButton()
315 self.offButton.setFixedSize(60,30)
316 self.offButton.setText("Off")
317 self.offButton.setFont(QtGui.QFont(myFont))
318 #print(myFont)
319 self.offButton.clicked.connect(self.offButtonFunction)
320
321 pixmapBack = QPixmap("Back.png")
322 self.backButton.setIcon(QIcon(pixmapBack))
323 self.backButton.setIconSize(QSize(60,35))
324
325 buttonHBox.addWidget(self.backButton)
326 buttonHBox.addWidget(self.onButton)
327 buttonHBox.addWidget(self.onTimeButton)
328 buttonHBox.addWidget(self.offButton)
329 graphVBox.addLayout(buttonHBox)
330
331 #Designing Settings
332 currentLabel = QLabel(self)
333 currentLabel.setText("Maximum Current")
334 currentLabel.setFont(QtGui.QFont(myFont, 12))
335 currentLabel.setAlignment(Qt.AlignCenter | Qt.AlignTop)
336 self.currentSelector = QComboBox()
337 self.currentSelector.setFont(QtGui.QFont(myFont, 12))
338 self.currentSelector.addItem("2.5 Amps")
339 self.currentSelector.addItem("5 Amps")
340 self.currentSelector.addItem("7.5 Amps")
341 self.currentSelector.addItem("10 Amps")
342 self.currentSelector.addItem("12.5 Amps")
343 self.currentSelector.addItem("15 Amps")
344 self.currentSelector.addItem("17.5 Amps")
345 self.currentSelector.addItem("20 Amps")
346 self.currentSelector.addItem("22.5 Amps")
347 self.currentSelector.addItem("25 Amps")
348 self.currentSelector.addItem("27.5 Amps")
349 self.currentSelector.addItem("30 Amps")
350 self.currentSelector.addItem("32.5 Amps")
351 self.currentSelector.addItem("35 Amps")
352 self.currentSelector.addItem("37.5 Amps")
353 self.currentSelector.addItem("40 Amps")
354 self.currentSelector.addItem("42.5 Amps")
355 self.currentSelector.addItem("45 Amps")
356 self.currentSelector.setCurrentIndex(5)
357 self.currentSelector.currentIndexChanged.connect(self.currentChange)
358
359 timeLabel = QLabel(self)
360 timeLabel.setText("Maximum Time")
361 timeLabel.setFont(QtGui.QFont(myFont, 12))
362 timeLabel.setAlignment(Qt.AlignCenter)
```

```
363 self.timeSelector = QComboBox()
364 self.timeSelector.setFont(QtGui.QFont(myFont, 12))
365 self.timeSelector.addItem("5 Sec")
366 self.timeSelector.addItem("10 Sec")
367 self.timeSelector.addItem("15 Sec")
368 self.timeSelector.addItem("20 Sec")
369 self.timeSelector.addItem("25 Sec")
370 self.timeSelector.addItem("30 Sec")
371 self.timeSelector.addItem("35 Sec")
372 self.timeSelector.addItem("40 Sec")
373 self.timeSelector.addItem("45 Sec")
374 self.timeSelector.addItem("50 Sec")
375 self.timeSelector.addItem("55 Sec")
376 self.timeSelector.addItem("60 Sec")
377 self.timeSelector.addItem("65 Sec")
378 self.timeSelector.addItem("70 Sec")
379 self.timeSelector.addItem("75 Sec")
380 self.timeSelector.addItem("80 Sec")
381 self.timeSelector.addItem("85 Sec")
382 self.timeSelector.addItem("90 Sec")
383 self.timeSelector.setCurrentIndex(5)
384 self.timeSelector.currentIndexChanged.connect(self.timeChange)
385
386 self.timeVal = QLabel(self)
387 self.currentVal = QLabel(self)
388 self.timeVal.setText("Time: ----- Sec")
389 self.currentVal.setText("Current: ----- Amps")
390 self.timeVal.setFont(QtGui.QFont(myFont, 12))
391 self.timeVal.setAlignment(Qt.AlignCenter)
392 self.currentVal.setFont(QtGui.QFont(myFont, 12))
393 self.currentVal.setAlignment(Qt.AlignCenter)
394
395 updateGraphButton = QPushButton()
396 updateGraphButton.setText("Update Graph")
397 updateGraphButton.setFont(QtGui.QFont(myFont))
398 updateGraphButton.clicked.connect(self.updateGraph)
399
400 logButton = QPushButton()
401 logButton.setText("LogData")
402 logButton.setFont(QtGui.QFont(myFont))
403 logButton.setFixedSize(113,30)
404 logButton.clicked.connect(self.logData)
405
406 resetButton = QPushButton()
407 resetButton.setText("Reset")
408 resetButton.setFont(QtGui.QFont(myFont))
409 resetButton.setFixedSize(113,30)
410 resetButton.clicked.connect(self.resetFunction)
411
412 self.status = QLabel(self)
413 self.status.setText("Status: MCB Test Interface Initiated")
414 self.status.setStyleSheet("color: black")
415 self.status.setAlignment(Qt.AlignLeft)
416 self.status.setFont(QtGui.QFont(myFont, 12))
417
418 settingsVBox.addWidget(currentLabel)
419 settingsVBox.addWidget(self.currentSelector)
420 settingsVBox.addWidget(timeLabel)
421 settingsVBox.addWidget(self.timeSelector)
422 settingsVBox.addWidget(self.currentVal)
423 settingsVBox.addWidget(self.timeVal)
```

```

424 settingsVBox.addWidget(updateGraphButton, alignment=Qt.AlignCenter)
425 settingsVBox.addWidget(logButton, alignment=Qt.AlignCenter)
426 settingsVBox.addWidget(resetButton, alignment=Qt.AlignCenter)
427 settingsVBox.addWidget(self.status)
428
429
430
431 bottomBox.addLayout(graphVBox)
432 bottomBox.addLayout(settingsVBox)
433 mainVBox.addLayout(topBox)
434 mainVBox.addLayout(bottomBox)
435
436
437 wid.setLayout(mainVBox)
438 self.show()
439
440 def showTime(self):
441     now = datetime.datetime.now()
442     now = now.strftime(dateFormat+"\n"+timeFormat)
443     self.labelTime.setText(now)
444
445 def onButtonFunction(self):
446     global sourceState
447     global amp
448     GPIO.output(21, GPIO.HIGH)
449     sourceState = 1
450     self.timeVal.setText("Time: ----- Sec")
451     self.status.setText("Status: Source Turned On Manually")
452     self.status.setStyleSheet("color: black")
453     amp = 0.001
454     while(True):
455         QtCore.QTimer.singleShot(10, self.currentSensing)
456
457         time.sleep(0.05)
458         if(sourceState == 0):
459             self.status.setText("Status: Source Turned Off Manually")
460             self.status.setStyleSheet("color: black")
461             break
462         if(amp > maxcurrent):
463             self.status.setText("Status: Maximum Current Reached ")
464             self.status.setStyleSheet("color: red")
465             GPIO.output(21, GPIO.LOW)
466             sourceState = 0
467             self.currentVal.setText("Current: {:.2f} Amps".format(amp))
468             break
469             self.currentVal.setText("Current: {:.2f} Amps".format(amp))
470             QtCore.QCoreApplication.processEvents()
471             amp = 0.001
472
473 def run(self):
474     global currentTime
475     global stop_threads
476     start = time.time()
477     while True:
478         if stop_threads:
479             break
480         #print('thread running')
481         end = time.time()
482         currentTime = end-start
483         self.timeVal.setText("Time: {:.2f} Sec".format(currentTime))
484         time.sleep(0.09)

```



```

485
486     def onTimeButtonFunction(self):
487         global sourceState
488         global maxTime
489         global maxcurrent
490         global stop_threads
491         global currentTime
492         global amp
493         global prevAmp
494         global tripTime
495         amp = 0.001
496         prevAmp = 0.001
497         GPIO.output(21, GPIO.HIGH)
498         sourceState = 1
499         self.status.setText("Status: Source Turned On Manually")
500         self.status.setStyleSheet("color: black")
501
502         stop_threads = False
503         t1 = threading.Thread(target = self.run)
504         t1.start()
505         while(True):
506             QtCore.QTimer.singleShot(0, self.currentSensing)
507             if(amp > 0):
508                 if (prevAmp - amp <= 2):
509                     prevAmp = amp
510             if(amp <= 0):
511                 self.status.setText("Status: MCB Tripped Successfully")
512                 now = datetime.datetime.now()
513                 now = now.strftime(dateFormat+" "+timeFormat)
514                 tripTime = now
515                 self.status.setStyleSheet("color: red")
516                 GPIO.output(21, GPIO.LOW)
517                 sourceState = 0
518                 amp = 0.001
519                 break
520             if(sourceState == 0):
521                 self.status.setText("Status: Source Turned Off Manually")
522                 self.status.setStyleSheet("color: black")
523                 break
524             if(amp > maxcurrent):
525                 self.status.setText("Status: Maximum Current Reached ")
526                 self.status.setStyleSheet("color: red")
527                 GPIO.output(21, GPIO.LOW)
528                 sourceState = 0
529                 self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
530                 break
531             if(currentTime >= maxTime):
532                 self.status.setText("Status: Maximum Time Reached ")
533                 self.status.setStyleSheet("color: red")
534                 GPIO.output(21, GPIO.LOW)
535                 sourceState = 0
536                 self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
537                 break
538
539             QtCore.QCoreApplication.processEvents()
540             self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
541             amp = 0.001
542             #prevAmp = 0.001
543             stop_threads = True
544             t1.join()
545             print('thread killed')

```

```

546
547
548 def currentSensing(self):
549     global GAIN
550     global count
551     global maxVal
552     global amp
553     global prevAmp
554     series = [0]*count
555     value = 0
556     for i in range (count):
557         try:
558             value = adc.read_adc(0, gain=GAIN)
559             time.sleep(0.001)
560             series[i] = value
561         except:
562             print("ADC Read Error")
563             QtCore.QCoreApplication.processEvents()
564     maxVal = max(series)
565     #amp = 0.0014*maxVal + 0.4
566     amp = -0.00000000000003*(maxVal**3) + 0.0000000023*(maxVal**2) + 0.0014*maxVal + 0.4
567     amp = round(amp,1)
568
569     if(amp < 0.3): amp = 0
570
571     QtCore.QCoreApplication.processEvents()
572
573 def offButtonFunction(self):
574     global sourceState
575     GPIO.output(21, GPIO.LOW)
576     self.currentVal.setText("Current: ----- Amps")
577     self.timeVal.setText("Time: ----- Sec")
578     sourceState = 0
579
580 def updateGraph(self):
581     global currentArray
582     global timeArray
583     global currentTime
584     global prevAmp
585     currentArray.append(prevAmp)
586     timeArray.append(currentTime)
587     print(currentArray)
588     print(timeArray)
589     self.plotMCB.axes.cla()
590     self.plotMCB.axes.set_ylim( bottom = 0, top = 50)
591     self.plotMCB.axes.set_xlim( left = 0, right = 50)
592     self.plotMCB.axes.set_title("TCC Curve")
593     self.plotMCB.axes.set_ylabel("Time (Sec)")
594     self.plotMCB.axes.set_xlabel("Current (Amps)")
595     self.plotMCB.axes.plot(currentArray,timeArray)
596     self.plotMCB.draw()
597
598
599
600 def currentChange(self,i):
601     global maxcurrent
602     maxcurrent = (i+1)*2.5
603     #print((i+1)*2.5)
604
605 def timeChange(self,i):
606     global maxTime

```

```

607         maxTime = (i+1)*5
608         #print((i+1)*5)
609
610     def resetFunction(self):
611         self.timeVal.setText("Time: ----- Sec")
612         self.currentVal.setText("Current: ----- Amps")
613         self.status.setText("Status: MCB Test Interface Initiated")
614         self.status.setStyleSheet("color: black")
615         self.plotMCB.axes.cla()
616         self.plotMCB.axes.set_ylim( bottom = 0, top = 50)
617         self.plotMCB.axes.set_xlim( left = 0, right = 50)
618         self.plotMCB.axes.set_title("TCC Curve")
619         self.plotMCB.axes.set_ylabel("Time (Sec)")
620         self.plotMCB.axes.set_xlabel("Current (Amps)")
621         self.plotMCB.draw()
622
623     def logData(self):
624         global prevAmp
625         global currentTime
626         global tripTime
627
628         print(tripTime)
629
630
631
632
633     def closeFunction(self):
634         global sourceState
635         global prevAmp
636         global amp
637         GPIO.output(21, GPIO.LOW)
638         sourceState = 0
639         global currentArray
640         global timeArray
641         currentArray = []
642         timeArray = []
643         prevAmp = 0.001
644         amp = 0.001
645         self.close()
646
647 class relayWindow(QMainWindow):
648     global sourceState
649     global maxTime
650     global maxCurrent
651     global myFont
652
653     global dateFormat
654     global timeFormat
655
656     sourceState = 0
657     maxTime = 30
658     maxcurrent = 15
659     def __init__(self):
660         super().__init__()
661         self.setWindowTitle("Relay Test")
662         left = 2
663         top = 30
664         width = 796
665         height = 450
666         self.setGeometry(left, top, width, height)
667         self.setFixedSize(width, height)

```

```
668
669     wid = QtWidgets.QWidget(self)
670     self.setCentralWidget(wid)
671     self.setStyleSheet("background-color: white;")
672
673     mainVBox = QVBoxLayout()
674     topBox = QHBoxLayout()
675     bottomBox = QHBoxLayout()
676     graphVBox = QVBoxLayout()
677     settingsVBox = QVBoxLayout()
678     buttonHBox = QHBoxLayout()
679
680     #Designing TopBox
681     self.labelTemp = QLabel(self)
682     self.labelTemp.setAlignment(Qt.AlignLeft)
683     self.labelTemp.setAlignment(Qt.AlignTop)
684     self.labelTemp.setFixedHeight(40)
685     self.labelTemp.setText("Temp: 36 °C")
686     self.labelTemp.setFont(QtGui.QFont(myFont))
687     self.labelTime = QLabel(self)
688     self.labelTime.setAlignment(Qt.AlignRight)
689     self.labelTime.setFixedHeight(40)
690     self.labelTime.setText("Time")
691     self.labelTime.setFont(QtGui.QFont(myFont))
692
693     topBox.addWidget(self.labelTemp)
694     topBox.addWidget(self.labelTime)
695
696     clockTimer = QTimer(self)
697     clockTimer.timeout.connect(self.showTime)
698     clockTimer.start(1000)
699     self.showTime()
700
701     #Designing Graph anf Buttons
702     self.plotMCB = MatplotlibWidget()
703     self.plotMCB.setFixedSize(550,350)
704     graphVBox.addWidget(self.plotMCB)
705
706     self.backButton = QPushButton()
707     self.backButton.setFixedSize(60,30)
708     self.backButton.clicked.connect(self.closeFunction)
709     self.onButton = QPushButton()
710     self.onButton.setFixedSize(60,30)
711     self.onButton.setText("On")
712     self.onButton.setFont(QtGui.QFont(myFont))
713     self.onButton.clicked.connect(self.onButtonFunction)
714     self.onTimeButton = QPushButton()
715     self.onTimeButton.setFixedSize(90,30)
716     self.onTimeButton.setText("On+Time")
717     self.onTimeButton.setFont(QtGui.QFont(myFont))
718     self.onTimeButton.clicked.connect(self.onTimeButtonFunction)
719     self.offButton = QPushButton()
720     self.offButton.setFixedSize(60,30)
721     self.offButton.setText("Off")
722     self.offButton.setFont(QtGui.QFont(myFont))
723     self.offButton.clicked.connect(self.offButtonFunction)
724
725     pixmapBack = QPixmap("Back.png")
726     self.backButton.setIcon(QIcon(pixmapBack))
727     self.backButton.setIconSize(QSize(60,35))
728
```

```
729 buttonHBox.addWidget(self.backButton)
730 buttonHBox.addWidget(self.onButton)
731 buttonHBox.addWidget(self.onTimeButton)
732 buttonHBox.addWidget(self.offButton)
733 graphVBox.addLayout(buttonHBox)
734
735 #Designing Settings
736 currentLabel = QLabel(self)
737 currentLabel.setText("Maximum Current")
738 currentLabel.setFont(QtGui.QFont(myFont, 12))
739 currentLabel.setAlignment(Qt.AlignCenter | Qt.AlignTop)
740 self.currentSelector = QComboBox()
741 self.currentSelector.setFont(QtGui.QFont(myFont, 12))
742 self.currentSelector.addItem("2.5 Amps")
743 self.currentSelector.addItem("5 Amps")
744 self.currentSelector.addItem("7.5 Amps")
745 self.currentSelector.addItem("10 Amps")
746 self.currentSelector.addItem("12.5 Amps")
747 self.currentSelector.addItem("15 Amps")
748 self.currentSelector.addItem("17.5 Amps")
749 self.currentSelector.addItem("20 Amps")
750 self.currentSelector.addItem("22.5 Amps")
751 self.currentSelector.addItem("25 Amps")
752 self.currentSelector.addItem("27.5 Amps")
753 self.currentSelector.addItem("30 Amps")
754 self.currentSelector.addItem("32.5 Amps")
755 self.currentSelector.addItem("35 Amps")
756 self.currentSelector.addItem("37.5 Amps")
757 self.currentSelector.addItem("40 Amps")
758 self.currentSelector.addItem("42.5 Amps")
759 self.currentSelector.addItem("45 Amps")
760 self.currentSelector.setCurrentIndex(5)
761 self.currentSelector.currentIndexChanged.connect(self.currentChange)
762
763 timeLabel = QLabel(self)
764 timeLabel.setText("Maximum Time")
765 timeLabel.setFont(QtGui.QFont(myFont, 12))
766 timeLabel.setAlignment(Qt.AlignCenter)
767 self.timeSelector = QComboBox()
768 self.timeSelector.setFont(QtGui.QFont(myFont, 12))
769 self.timeSelector.addItem("5 Sec")
770 self.timeSelector.addItem("10 Sec")
771 self.timeSelector.addItem("15 Sec")
772 self.timeSelector.addItem("20 Sec")
773 self.timeSelector.addItem("25 Sec")
774 self.timeSelector.addItem("30 Sec")
775 self.timeSelector.addItem("35 Sec")
776 self.timeSelector.addItem("40 Sec")
777 self.timeSelector.addItem("45 Sec")
778 self.timeSelector.addItem("50 Sec")
779 self.timeSelector.addItem("55 Sec")
780 self.timeSelector.addItem("60 Sec")
781 self.timeSelector.addItem("65 Sec")
782 self.timeSelector.addItem("70 Sec")
783 self.timeSelector.addItem("75 Sec")
784 self.timeSelector.addItem("80 Sec")
785 self.timeSelector.addItem("85 Sec")
786 self.timeSelector.addItem("90 Sec")
787 self.timeSelector.setCurrentIndex(5)
788 self.timeSelector.currentIndexChanged.connect(self.timeChange)
789
```

```
790 self._toggle = True
791 self.nOpenCheck = QCheckBox("Normally Open",self)
792 self.nOpenCheck.setFont(QtGui.QFont(myFont))
793 self.nOpenCheck.setChecked(self._toggle)
794 self.nCloseCheck = QCheckBox("Normally Close",self)
795 self.nCloseCheck.setFont(QtGui.QFont(myFont))
796 self.nCloseCheck.setChecked(not self._toggle)
797 self.nCloseCheck.clicked.connect(self.toggle)
798 self.nOpenCheck.clicked.connect(self.toggle)
799
800
801 self.timeVal = QLabel(self)
802 self.currentVal = QLabel(self)
803 self.timeVal.setText("Time: ----- Sec")
804 self.currentVal.setText("Current: ----- Amps")
805 self.timeVal.setFont(QtGui.QFont(myFont, 12))
806 self.timeVal.setAlignment(Qt.AlignCenter)
807 self.currentVal.setFont(QtGui.QFont(myFont, 12))
808 self.currentVal.setAlignment(Qt.AlignCenter)
809
810
811 updateGraphButton = QToolButton()
812 updateGraphButton.setText("Update Graph")
813 updateGraphButton.setFont(QtGui.QFont(myFont))
814 updateGraphButton.clicked.connect(self.updateGraph)
815
816 logButton = QToolButton()
817 logButton.setText("LogData")
818 logButton.setFont(QtGui.QFont(myFont))
819 logButton.setFixedSize(113,30)
820
821 resetButton = QToolButton()
822 resetButton.setText("Reset")
823 resetButton.setFont(QtGui.QFont(myFont))
824 resetButton.setFixedSize(113,30)
825 resetButton.clicked.connect(self.resetFunction)
826
827 self.status = QLabel(self)
828 self.status.setText("Status: Relay Test Interface Initiated")
829 self.status.setStyleSheet("color: black")
830 self.status.setAlignment(Qt.AlignLeft)
831 self.status.setFont(QtGui.QFont(myFont, 12))
832
833 settingsVBox.addWidget(currentLabel)
834 settingsVBox.addWidget(self.currentSelector)
835 settingsVBox.addWidget(timeLabel)
836 settingsVBox.addWidget(self.timeSelector)
837 settingsVBox.addWidget(self.nOpenCheck, alignment=Qt.AlignCenter)
838 settingsVBox.addWidget(self.nCloseCheck, alignment=Qt.AlignCenter)
839 settingsVBox.addWidget(self.currentVal)
840 settingsVBox.addWidget(self.timeVal)
841 settingsVBox.addWidget(updateGraphButton, alignment=Qt.AlignCenter)
842 settingsVBox.addWidget(logButton, alignment=Qt.AlignCenter)
843 settingsVBox.addWidget(resetButton, alignment=Qt.AlignCenter)
844 settingsVBox.addWidget(self.status)
845
846
847
848 bottomBox.addLayout(graphVBox)
849 bottomBox.addLayout(settingsVBox)
850 mainVBox.addLayout(topBox)
```

```

851         mainVBox.addLayout(bottomBox)
852
853
854         wid.setLayout(mainVBox)
855         self.show()
856
857     def showTime(self):
858         now = datetime.datetime.now()
859         now = now.strftime(dateFormat+"\n"+timeFormat)
860         self.labelTime.setText(now)
861
862     def onButtonFunction(self):
863         global sourceState
864         global amp
865         GPIO.output(21, GPIO.HIGH)
866         sourceState = 1
867         self.timeVal.setText("Time: ----- Sec")
868         self.status.setText("Status: Source Turned On Manually")
869         self.status.setStyleSheet("color: black")
870         amp = 0.001
871         while True:
872             QtCore.QTimer.singleShot(10, self.currentSensing)
873
874             time.sleep(0.05)
875             if(sourceState == 0):
876                 self.status.setText("Status: Source Turned Off Manually")
877                 self.status.setStyleSheet("color: black")
878                 break
879             if(amp > maxcurrent):
880                 self.status.setText("Status: Maximum Current Reached ")
881                 self.status.setStyleSheet("color: red")
882                 GPIO.output(21, GPIO.LOW)
883                 sourceState = 0
884                 self.currentVal.setText("Current: {:.2f} Amps".format(amp))
885                 break
886             self.currentVal.setText("Current: {:.2f} Amps".format(amp))
887             QtCore.QCoreApplication.processEvents()
888             amp = 0.001
889
890     def run(self):
891         global currentTime
892         global stop_threads
893         start = time.time()
894         while True:
895             if stop_threads:
896                 break
897             #print('thread running')
898             end = time.time()
899             currentTime = end-start
900             self.timeVal.setText("Time: {:.2f} Sec".format(currentTime))
901             time.sleep(0.09)
902
903     def onTimeButtonFunction(self):
904         global sourceState
905         global maxTime
906         global maxcurrent
907         global stop_threads
908         global currentTime
909         global amp
910         global prevAmp
911         amp = 0.001

```

```

912     prevAmp = 0.001
913     GPIO.output(21, GPIO.HIGH)
914     sourceState = 1
915     self.status.setText("Status: Source Turned On Manually")
916     self.status.setStyleSheet("color: black")
917
918     stop_threads = False
919     t1 = threading.Thread(target = self.run)
920     t1.start()
921     while(True):
922         QtCore.QTimer.singleShot(0, self.currentSensing)
923         if(amp > 0):
924             if (prevAmp - amp <= 2):
925                 prevAmp = amp
926         if(self.nOpenCheck.isChecked()):
927             if(GPIO.input(26) == GPIO.HIGH):
928                 self.status.setText("Status: Relay Tripped Successfully")
929                 self.status.setStyleSheet("color: red")
930                 GPIO.output(21, GPIO.LOW)
931                 sourceState = 0
932                 amp = 0.001
933                 break
934         if(self.nCloseCheck.isChecked()):
935             if(GPIO.input(26) == GPIO.LOW):
936                 self.status.setText("Status: Relay Tripped Successfully")
937                 self.status.setStyleSheet("color: red")
938                 GPIO.output(21, GPIO.LOW)
939                 sourceState = 0
940                 amp = 0.001
941                 break
942
943         if(sourceState == 0):
944             self.status.setText("Status: Source Turned Off Manually")
945             self.status.setStyleSheet("color: black")
946             break
947         if(amp > maxcurrent):
948             self.status.setText("Status: Maximum Current Reached ")
949             self.status.setStyleSheet("color: red")
950             GPIO.output(21, GPIO.LOW)
951             self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
952             sourceState = 0
953             break
954         if(currentTime >= maxTime):
955             self.status.setText("Status: Maximum Time Reached ")
956             self.status.setStyleSheet("color: red")
957             GPIO.output(21, GPIO.LOW)
958             self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
959             sourceState = 0
960             break
961
962         QtCore.QCoreApplication.processEvents()
963         self.currentVal.setText("Current: {:.2f} Ams".format(prevAmp))
964     amp = 0.001
965     stop_threads = True
966     t1.join()
967
968     def currentSensing(self):
969         global GAIN
970         global count
971         global maxVal
972         global amp

```



```

973     global prevAmp
974     series = [0]*count
975     value = 0
976     for i in range (count):
977         try:
978             value = adc.read_adc(0, gain=GAIN)
979             time.sleep(0.001)
980             series[i] = value
981         except:
982             print("ADC Read Error")
983             QtCore.QCoreApplication.processEvents()
984     maxVal = max(series)
985     #amp = 0.0014*maxVal + 0.4
986     amp = -0.00000000000003*(maxVal**3) + 0.0000000023*(maxVal**2) + 0.0014*maxVal + 0.4
987     amp = round(amp,1)
988
989     if(amp < 0.3): amp = 0
990
991     QtCore.QCoreApplication.processEvents()
992
993     def offButtonFunction(self):
994         global sourceState
995         self.status.setText("Status: Source Turned Off Manually")
996         self.status.setStyleSheet("color: black")
997         GPIO.output(21, GPIO.LOW)
998         self.currentVal.setText("Current: ----- Amps")
999         self.timeVal.setText("Time: ----- Sec")
1000         sourceState = 0
1001
1002     #@pyqtSlot()
1003     def toggle(self):
1004         self._toggle = not self._toggle
1005         self.nOpenCheck.setChecked(self._toggle)
1006         self.nCloseCheck.setChecked(not self._toggle)
1007
1008     def updateGraph(self):
1009         global currentArray
1010         global timeArray
1011         global currentTime
1012         global prevAmp
1013         currentArray.append(prevAmp)
1014         timeArray.append(currentTime)
1015         print(currentArray)
1016         print(timeArray)
1017         self.plotMCB.axes.cla()
1018         self.plotMCB.axes.set_ylim( bottom = 0, top = 50)
1019         self.plotMCB.axes.set_xlim( left = 0, right = 50)
1020         self.plotMCB.axes.set_title("TCC Curve")
1021         self.plotMCB.axes.set_ylabel("Time (Sec)")
1022         self.plotMCB.axes.set_xlabel("Current (Amps)")
1023         self.plotMCB.axes.plot(currentArray,timeArray)
1024         self.plotMCB.draw()
1025
1026
1027
1028     def currentChange(self,i):
1029         global maxcurrent
1030         maxcurrent = (i+1)*2.5
1031         #print((i+1)*2.5)
1032
1033     def timeChange(self,i):

```

```

1034     global maxTime
1035     maxTime = (i+1)*5
1036     #print((i+1)*5)
1037
1038     def resetFunction(self):
1039         self.timeVal.setText("Time: ----- Sec")
1040         self.currentVal.setText("Current: ----- Amps")
1041         self.status.setText("Status: MCB Test Interface Initiated")
1042         self.status.setStyleSheet("color: black")
1043         self.plotMCB.axes.cla()
1044         self.plotMCB.axes.set_ylim( bottom = 0, top = 50)
1045         self.plotMCB.axes.set_xlim( left = 0, right = 50)
1046         self.plotMCB.axes.set_title("TCC Curve")
1047         self.plotMCB.axes.set_ylabel("Time (Sec)")
1048         self.plotMCB.axes.set_xlabel("Current (Amps)")
1049         self.plotMCB.draw()
1050
1051
1052     def closeFunction(self):
1053         global sourceState
1054         global prevAmp
1055         global amp
1056         GPIO.output(21, GPIO.LOW)
1057         sourceState = 0
1058         global currentArray
1059         global timeArray
1060         currentArray = []
1061         timeArray = []
1062         prevAmp = 0.001
1063         amp = 0.001
1064         self.close()
1065
1066     class MatplotlibWidget(FigureCanvas):
1067         def __init__(self, parent=None, xlim = 50,ylim = 50, hold=False):
1068             super(MatplotlibWidget, self).__init__(Figure())
1069             self.setParent(parent)
1070             self.figure = Figure(figsize=(5, 1),dpi = 95)
1071             self.canvas = FigureCanvas(self.figure)
1072
1073             self.axes = self.figure.add_subplot(111)
1074
1075             self.axes.set_title("TCC Curve")
1076             self.axes.set_ylabel("Time (Sec)")
1077             self.axes.set_xlabel("Current (Amps)")
1078             self.axes.set_xbound(lower = 0, upper = xlim)
1079             self.axes.set_ylim( bottom = 0, top = 50)
1080
1081     class settingsWindow(QMainWindow):
1082
1083         def __init__(self):
1084             global lineList
1085             global myFont
1086             global dateFormat
1087             global timeFormat
1088             global currentTimeItem
1089             global currentDateItem
1090             global currentFontItem
1091
1092             super().__init__()
1093             self.setWindowTitle("Settings")
1094             left = 2

```

```

1095     top = 30
1096     width = 796
1097     height = 450
1098     self.setGeometry(left, top, width, height)
1099     self.setFixedSize(width, height)
1100     wid = QtWidgets.QWidget(self)
1101     self.setCentralWidget(wid)
1102     self.setStyleSheet("background-color: white;")
1103
1104     mainVBox = QVBoxLayout()
1105
1106     #Designing TopBox
1107     topBox = QHBoxLayout()
1108     self.labelTemp = QLabel(self)
1109     self.labelTemp.setAlignment(Qt.AlignLeft)
1110     self.labelTemp.setAlignment(Qt.AlignTop)
1111     self.labelTemp.setFixedHeight(40)
1112     self.labelTemp.setText("Temp: 36 °C")
1113     self.labelTemp.setFont(QtGui.QFont(myFont))
1114     self.labelTime = QLabel(self)
1115     self.labelTime.setAlignment(Qt.AlignRight)
1116     self.labelTime.setFixedHeight(40)
1117     self.labelTime.setText("Time")
1118     self.labelTime.setFont(QtGui.QFont(myFont))
1119
1120     topBox.addWidget(self.labelTemp)
1121     topBox.addWidget(self.labelTime)
1122
1123     clockTimer = QTimer(self)
1124     clockTimer.timeout.connect(self.showTime)
1125     clockTimer.start(1000)
1126     self.showTime()
1127
1128     fontBox = QHBoxLayout()
1129     fontLabel = QLabel()
1130     fontLabel.setText("Select Font:      ") #5 Spaces After
1131     fontLabel.setFont(QtGui.QFont(myFont))
1132     fontList = QComboBox()
1133     with open("fontList") as f:
1134         lineList = [line.rstrip('\n') for line in open("fontList")]
1135     fontList.addItem(lineList)
1136     fontList.setFont(QtGui.QFont(myFont))
1137     fontList.setCurrentIndex(currentFontItem)
1138     fontList.currentIndexChanged.connect(self.fontChange)
1139     fontBox.addWidget(fontLabel, alignment=Qt.AlignRight)
1140     fontBox.addWidget(fontList, alignment=Qt.AlignLeft)
1141
1142     speedBox = QHBoxLayout()
1143     speedLabel = QLabel()
1144     speedLabel.setText("Set Fan Speed:      ")
1145     speedLabel.setFont(QtGui.QFont(myFont))
1146     speedSlider = QSlider(Qt.Horizontal)
1147     speedSlider.setFocusPolicy(Qt.StrongFocus)
1148     speedSlider.setTickPosition(QSlider.TicksBothSides)
1149     speedSlider.setFixedSize(150,30)
1150     speedSlider.setMinimum(50)
1151     speedSlider.setMaximum(100)
1152     speedSlider.setValue(70)
1153     speedSlider.setTickInterval(5)
1154     speedSlider.setSingleStep(5)
1155     #speedSlider.valueChanged.connect(self.valuechange)

```

```
1156 speedBox.addWidget(speedLabel, alignment=Qt.AlignRight)
1157 speedBox.addWidget(speedSlider, alignment=Qt.AlignLeft)
1158
1159 tempBox = QHBoxLayout()
1160 tempLabel = QLabel()
1161 tempLabel.setText("Temperature Unit: ")
1162 tempLabel.setFont(QtGui.QFont(myFont))
1163 tempList = QComboBox()
1164 tempList.addItem("°C")
1165 tempList.addItem("°F")
1166 tempList.setFont(QtGui.QFont(myFont))
1167 tempList.currentIndexChanged.connect(self.tempChange)
1168 tempBox.addWidget(tempLabel, alignment=Qt.AlignRight)
1169 tempBox.addWidget(tempList, alignment=Qt.AlignLeft)
1170
1171 dateBox = QHBoxLayout() #DateFormat dd/mm/yyyy
1172 dateLabel = QLabel()
1173 dateLabel.setText("Date Format: ")
1174 dateLabel.setFont(QtGui.QFont(myFont))
1175 dateList = QComboBox()
1176 dateList.setFont(QtGui.QFont(myFont))
1177 dateList.addItem("dd/mm/yyyy")
1178 dateList.addItem("dd/mm/yy")
1179 dateList.addItem("dd/mmmm/yyyy")
1180 dateList.addItem("dd/mmmm/yy")
1181 dateList.addItem("mm/dd/yyyy")
1182 dateList.addItem("mm/dd/yy")
1183 dateList.addItem("mmmm/dd/yyyy")
1184 dateList.addItem("mmmm/dd/yy")
1185 dateList.setCurrentIndex(currentDateItem)
1186 dateList.currentIndexChanged.connect(self.dateChange)
1187 dateBox.addWidget(dateLabel, alignment=Qt.AlignRight)
1188 dateBox.addWidget(dateList, alignment=Qt.AlignLeft)
1189
1190 timeBox = QHBoxLayout() #TiemFormat 12/24Hr
1191 timeLabel = QLabel()
1192 timeLabel.setText("Time Format: ")
1193 timeLabel.setFont(QtGui.QFont(myFont))
1194 timeList = QComboBox()
1195 timeList.setFont(QtGui.QFont(myFont))
1196 timeList.addItem("12hr")
1197 timeList.addItem("24hr")
1198 timeList.setCurrentIndex(currentTimeItem)
1199 timeList.currentIndexChanged.connect(self.timeChange)
1200 timeBox.addWidget(timeLabel, alignment=Qt.AlignRight)
1201 timeBox.addWidget(timeList, alignment=Qt.AlignLeft)
1202
1203 saveButton = QPushButton()
1204 saveButton.setText("Save Settings")
1205 saveButton.setFont(QtGui.QFont(myFont))
1206 saveButton.setFixedSize(113,30)
1207 saveButton.clicked.connect(self.saveSettings)
1208
1209
1210
1211
1212
1213
1214 mainVBox.addLayout(topBox)
1215 mainVBox.addLayout(fontBox)
1216 mainVBox.addLayout(speedBox)
```

```
1217         mainVBox.addLayout(tempBox)
1218         mainVBox.addLayout(timeBox)
1219         mainVBox.addLayout(dateBox)
1220         mainVBox.addWidget(saveButton)
1221
1222
1223         wid.setLayout(mainVBox)
1224
1225         self.show()
1226
1227     def showTime(self):
1228         now = datetime.datetime.now()
1229         now = now.strftime(dateFormat+"\n"+timeFormat)
1230         self.labelTime.setText(now)
1231
1232     def fontChange(self,i):
1233         global myFont
1234         global lineList
1235         global currentFontItem
1236
1237         #print(lineList[i])
1238         myFont = lineList[i]
1239         currentFontItem = i
1240
1241     def tempChange(self,i):
1242         global tempUnit
1243
1244         if(i==0): tempUnit = "C"
1245
1246         if(i==1): tempUnit = "F"
1247
1248     def dateChange(self,i):
1249         global dateFormat
1250         global currentDateItem
1251
1252         if(i==0): dateFormat="%a %d-%m-%Y"
1253         if(i==1): dateFormat="%a %d-%m-%y"
1254         if(i==2): dateFormat="%a %d-%b-%Y"
1255         if(i==3): dateFormat="%a %d-%b-%y"
1256         if(i==4): dateFormat="%a %m-%d-%Y"
1257         if(i==5): dateFormat="%a %m-%d-%y"
1258         if(i==6): dateFormat="%a %b-%d-%Y"
1259         if(i==7): dateFormat="%a %b-%d-%y"
1260
1261
1262         currentDateItem= i
1263
1264
1265
1266     def timeChange(self,i):
1267         global timeFormat
1268         global currentTimeItem
1269
1270         if(i==0): timeFormat = "%I:%M:%S %p"
1271
1272         if(i==1): timeFormat = "%H:%M:%S"
1273
1274         currentTimeItem = i
1275
1276     def saveSettings(self):
1277
```

```

1278         self.w = Window()
1279         self.w.show()
1280         self.close()
1281
1282     class howtoWindow(QScrollArea):
1283
1284         def __init__(self):
1285
1286             global myFont
1287             super().__init__()
1288             self.setWindowTitle("How To Use")
1289             left = 2
1290             top = 30
1291             width = 796
1292             height = 450
1293             self.setGeometry(left, top, width, height)
1294             self.setFixedSize(width, height)
1295             self.setStyleSheet("background-color: white;")
1296
1297             widget = QWidget()
1298             layout = QVBoxLayout(widget)
1299             layout.setAlignment(Qt.AlignTop)
1300
1301             Title = QLabel("How To Use")
1302             Title.setFont(QtGui.QFont(myFont, 26, QtGui.QFont.Bold))
1303             layout.addWidget(Title, alignment=Qt.AlignCenter)
1304
1305             Title0 = QLabel("Power Supply:")
1306             Title0.setFont(QtGui.QFont(myFont, 16, QtGui.QFont.Bold))
1307             layout.addWidget(Title0)
1308
1309             description0 = QLabel("Following conditions should be met before powering the equipment\n"\
1310                                   "• Supply Voltage is reliable 220V (+-10%)\n"\
1311                                   "• Supply Frequency is reliable 50Hz (+-2%)\n"\
1312                                   "• Voltage Stabilizer should be used in case of unreliable supply\n"\
1313                                   "• Supply should be free of harmonics (Do not operate on UPS)")
1314             description0.setFont(QtGui.QFont(myFont, 13))
1315             layout.addWidget(description0)
1316
1317             Title1 = QLabel("Terminals Detail:")
1318             Title1.setFont(QtGui.QFont(myFont, 16, QtGui.QFont.Bold))
1319             layout.addWidget(Title1)
1320
1321             description = QLabel("There are 4 terminals on back and 6 terminals on front which are marked\n"\
1322                                   "accordingly\n"\
1323                                   ">>> Back Terminals\n"\
1324                                   "    • Two back terminals are used to power the variac\n"\
1325                                   "    • Two back terminals are used to get output from variac\n"\
1326                                   ">>> Front Terminals\n"\
1327                                   "    • Two front terminals are rated at 40 Amps\n"\
1328                                   "    • Two front terminals are rated at 15 Amps\n"\
1329                                   "    • Two front terminals are used to get state of relay contacts\n")
1329             description.setFont(QtGui.QFont(myFont, 13))
1330             layout.addWidget(description)
1331
1332             Title2 = QLabel("MCB Test:")
1333             Title2.setFont(QtGui.QFont(myFont, 16, QtGui.QFont.Bold))
1334             layout.addWidget(Title2)
1335
1336             description2 = QLabel("MCBs upto 40 Amps can be tested with this equipment (1/2 Pole)\n"\
1337                                   "• Make connections with the MCB\n")

```

```

1338         "\n• Turn MCB on while keeping variac at 0\n"\
1339         "\n• Go to MCB Test Window\n"\
1340         "\n• Turn on the injection by pressing \"On\" button\n"\
1341         "\n• Set desired current by increasing variac voltage\n"\
1342         "\n• Turn off the injection after achieving desired current\n"\
1343         "\n• Wait for a second\n"\
1344         "\n• Again turn on the injection by pressing \"On + Time\" button\n"\
1345         "\n• Wait for the MCB to trip\n"\
1346         "\n• Get results")
1347     description2.setFont(QtGui.QFont(myFont,13))
1348     layout.addWidget(description2)
1349
1350     Title3 = QLabel("Relay Test:")
1351     Title3.setFont(QtGui.QFont(myFont,16,QtGui.QFont.Bold))
1352     layout.addWidget(Title3)
1353
1354     description3 = QLabel("Single Phase Relays upto 40 Amps can be tested with this
equipment\n\n")
1355         "\n• Keep Variac at 0\n"\
1356         "\n• Make connections with the Relay\n"\
1357         "\n• Go to Relay Test Window\n"\
1358         "\n• Set the Initial state of relay i.e. NO/NC\n"\
1359         "\n• Turn on the injection by pressing \"On\" button\n"\
1360         "\n• Set desired current by increasing variac voltage\n"\
1361         "\n• Turn off the injection after achieving desired current\n"\
1362         "\n• Wait for a second\n"\
1363         "\n• Again turn on the injection by pressing \"On + Time\" button\n"\
1364         "\n• Wait for the Relay to trip\n"\
1365         "\n• Get results")
1366     description3.setFont(QtGui.QFont(myFont,13))
1367     layout.addWidget(description3)
1368
1369     howto1L = QLabel(self)
1370     howto1P = QPixmap('howto1.PNG')
1371     howto1L.setPixmap(howto1P)
1372     howto1L.setFixedSize(488,400)
1373     layout.addWidget(howto1L,alignment=Qt.AlignCenter)
1374
1375     Title4 = QLabel("Current/Time Settings:")
1376     Title4.setFont(QtGui.QFont(myFont,16,QtGui.QFont.Bold))
1377     layout.addWidget(Title4)
1378
1379     description4 = QLabel("There are various current and time ranges given in the equipment to
protect the device being tested\n\n")
1380         "\n• Select the Maximum current you want to inject\n"\
1381         "\n• Select the Maximum time for test operation\n"\
1382         "\n• The equipment will automatically abort test if current value is
increased from preset value\n"
1383         "\n• The equipment will automatically abort test if test time is increased
from preset value")
1384
1385     description4.setFont(QtGui.QFont(myFont,13))
1386     layout.addWidget(description4)
1387
1388     howto2L = QLabel(self)
1389     howto2P = QPixmap('howto2.PNG')
1390     howto2L.setPixmap(howto2P)
1391     #howto2L.setFixedSize(488,400)
1392     layout.addWidget(howto2L,alignment=Qt.AlignCenter)
1393
1394     Title5 = QLabel("Graph and Data Logging:")
1395     Title5.setFont(QtGui.QFont(myFont,16,QtGui.QFont.Bold))

```

```

1396         layout.addWidget(Title5)
1397
1398         description5 = QLabel("The equipment has plotting and data logging capabilities \n\n\"
1399             \"• After successful conduction of test, press \"Update Graph\" button to
show values on graph\n\"
1400             \"• The plotting feature is available for both Relays and MCB\n\"
1401             \"• Press \"Log Data\" button to log the most resent result\n\"
1402             \"• All the logged results can be viewed in \"Logged Data\" window on
Main Screen\n\"
1403             \"• Reset button is used to clear the Test Screen\n\"
1404             \"• \"Status\" shows the most recent state of equipment\")
1405
1406         description5.setFont(QtGui.QFont(myFont,13))
1407         layout.addWidget(description5)
1408
1409         howto3L = QLabel(self)
1410         howto3P = QPixmap('howto3.PNG')
1411         howto3L.setPixmap(howto3P)
1412         #howto2L.setFixedSize(488,400)
1413         layout.addWidget(howto3L,alignment=Qt.AlignCenter)
1414
1415         self.backButton = QPushButton()
1416         self.backButton.setFixedSize(60,30)
1417         self.backButton.clicked.connect(self.closeFunction)
1418         pixmapBack = QPixmap("Back.png")
1419         self.backButton.setIcon(QIcon(pixmapBack))
1420         self.backButton.setIconSize(QSize(60,35))
1421         layout.addWidget(self.backButton)
1422
1423         self.setWidget(widget)
1424         self.show()
1425
1426         def closeFunction(self):
1427             self.close()
1428
1429         class aboutWindow(QScrollArea):
1430
1431             def __init__(self):
1432
1433                 global myFont
1434                 super().__init__()
1435                 self.setWindowTitle("About")
1436                 left = 2
1437                 top = 30
1438                 width = 796
1439                 height = 450
1440                 self.setGeometry(left, top, width, height)
1441                 self.setFixedSize(width, height)
1442                 self.setStyleSheet("background-color: white;")
1443
1444                 widget = QWidget()
1445                 layout = QVBoxLayout(widget)
1446                 layout.setAlignment(Qt.AlignTop)
1447
1448                 descriptionTitle = QLabel("Description")
1449                 descriptionTitle.setFont(QtGui.QFont(myFont,26,QtGui.QFont.Bold))
1450                 layout.addWidget(descriptionTitle ,alignment=Qt.AlignCenter)
1451
1452                 description = QLabel("Most test units currently used are based on power electronics circuitry
and hence are very expensive. Most\n\"
1453                 \"industries cannot afford such very expensive equipment and hence they
hire third parties for relay testing \n\"

```



```

1454         "which is also expensive. This relay testing unit is economical, user
friendly and can test over current relays, \n"
1455         "earth fault relays and reverse power relays along with the wide range
of miniature circuit breakers. The test set \n"
1456         "is designed to perform the secondary injection testing by artificially
injecting the fault currents in controlled \n"
1457         "manner and find out the tripping time of protective relays and
miniature circuit breakers.\nFeatures of equipment are; \n\n"
1458         "• User Friendly Graphical User Interface\n"
1459         "• Rigid Equipment Design\n"
1460         "• Reliable for use in Industrial Settings\n"
1461         "• Manual Cooling System\n"
1462         "• Built-in Protection System\n"
1463         "• TCC Curve Plotting\n"
1464         "• Report Generator\n"
1465         "• Wide Range of Settings\n")
1466
1467
1468     description.setFont(QtGui.QFont(myFont,13))
1469     layout.addWidget(description)
1470
1471     specificationTitle = QLabel("Specifications")
1472     specificationTitle.setFont(QtGui.QFont(myFont,26,QtGui.QFont.Bold))
1473     layout.addWidget(specificationTitle ,alignment=Qt.AlignCenter)
1474     specificationTable = QTableWidgetItem()
1475     specificationTable.setRowCount(11)
1476     specificationTable.setColumnCount(2)
1477     specificationTable.setFixedSize(312,350)
1478     specificationTable.setHorizontalHeaderLabels(("Specification","Value"))
1479     specificationTable.horizontalHeaderItem(0).setFont(QtGui.QFont(myFont,13,QtGui.QFont.Bold))
1480     specificationTable.horizontalHeaderItem(1).setFont(QtGui.QFont(myFont,13,QtGui.QFont.Bold))
1481
1482     specificationTable.setItem(0,0,QTableWidgetItem("Rated Supply"))
1483     specificationTable.setItem(0,1,QTableWidgetItem("220 V"))
1484     specificationTable.setItem(1,0,QTableWidgetItem("Operating Temperature (Min)"))
1485     specificationTable.setItem(1,1,QTableWidgetItem("0 °C"))
1486     specificationTable.setItem(2,0,QTableWidgetItem("Operating Temperature (Max)"))
1487     specificationTable.setItem(2,1,QTableWidgetItem("70 °C"))
1488     specificationTable.setItem(3,0,QTableWidgetItem("Max Current Output"))
1489     specificationTable.setItem(3,1,QTableWidgetItem("40 Amps"))
1490     specificationTable.setItem(4,0,QTableWidgetItem("Max Terminal Voltage"))
1491     specificationTable.setItem(4,1,QTableWidgetItem("24 Volts"))
1492     specificationTable.setItem(5,0,QTableWidgetItem("Operating Time (0-5 Amps)"))
1493     specificationTable.setItem(5,1,QTableWidgetItem("30 Mins"))
1494     specificationTable.setItem(6,0,QTableWidgetItem("Operating Time (5-10 Amps)"))
1495     specificationTable.setItem(6,1,QTableWidgetItem("20 Mins"))
1496     specificationTable.setItem(7,0,QTableWidgetItem("Operating Time (10-15 Amps)"))
1497     specificationTable.setItem(7,1,QTableWidgetItem("15 Mins"))
1498     specificationTable.setItem(8,0,QTableWidgetItem("Operating Time (15-20 Amps)"))
1499     specificationTable.setItem(8,1,QTableWidgetItem("10 Mins"))
1500     specificationTable.setItem(9,0,QTableWidgetItem("Operating Time (20-30 Amps)"))
1501     specificationTable.setItem(9,1,QTableWidgetItem("5 Mins"))
1502     specificationTable.setItem(10,0,QTableWidgetItem("Operating Time (above 30 Amps)"))
1503     specificationTable.setItem(10,1,QTableWidgetItem("1 Min"))
1504     specificationTable.resizeColumnsToContents()
1505     specificationTable.resizeRowsToContents()
1506     specificationTable.verticalHeader().setVisible(False)
1507     layout.addWidget(specificationTable,alignment=Qt.AlignCenter)
1508
1509     groupTitle = QLabel("\nManufacturer Details")
1510     groupTitle.setFont(QtGui.QFont(myFont,26,QtGui.QFont.Bold))
1511     layout.addWidget(groupTitle ,alignment=Qt.AlignCenter)

```

```
1512
1513     descriptionGroup = QLabel("This test set is manufactured by Group-44  Batch 2015-16\n\n"
1514                               "• Omama Zaheen (EE-15136)\n"\
1515                               "• Muhamamd Raaïd khan (EE-15141)\n"\
1516                               "• Hammad Junaïd (EE-15146)\n"\
1517                               "• Uzair Ali Khan (EE-15156)\n\n"\
1518                               "Under the supervision of:\n\n"\
1519                               "• Internal Examiner: Muhammad Farooq Siddiqui\n"\
1520                               "• External Examiner: Muhammad Humaid Saeed\n"\
1521                               )
1522
1523
1524     descriptionGroup.setFont(QtGui.QFont(myFont,13))
1525     layout.addWidget(descriptionGroup)
1526
1527     self.backButton = QPushButton()
1528     self.backButton.setFixedSize(60,30)
1529     self.backButton.clicked.connect(self.closeFunction)
1530     pixmapBack = QPixmap("Back.png")
1531     self.backButton.setIcon(QIcon(pixmapBack))
1532     self.backButton.setIconSize(QSize(60,35))
1533     layout.addWidget(self.backButton)
1534
1535     self.setWidget(widget)
1536     self.show()
1537
1538     def closeFunction(self):
1539         self.close()
1540
1541 if __name__ == "__main__":
1542
1543     App = QApplication(sys.argv)
1544     print("Window Initiated")
1545     window = Window()
1546     sys.exit(App.exec())
1547
```