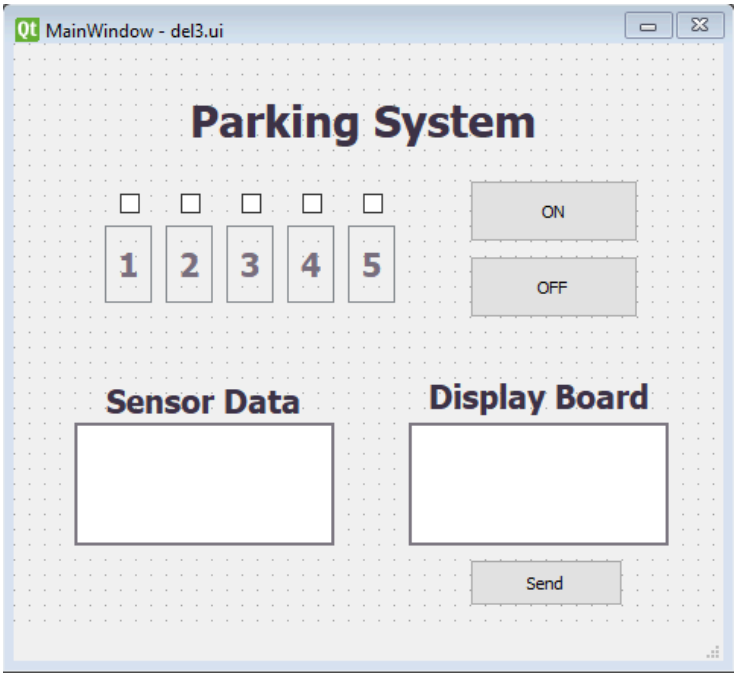
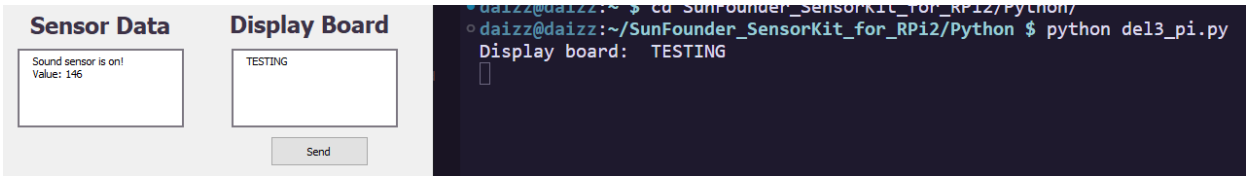
Name: Daizzah Botoy
NSID: dcb772
Deliverable: 3

User Interface Design:



Sending code from display board:



PC code to run the GUI:

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QLineEdit
import paho.mqtt.client as mqtt
from PyQt5.QtCore import QTimer
import json

class Ui_MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.sensor_data = ""
        self.parking_data = {}
        self.setupUi(self)
```

```
        self.off_button.clicked.connect(self.off_button_click)
        self.on_button.clicked.connect(self.on_button_click)
        self.send_button.clicked.connect(self.send_button_click)
        self.checkboxes = [self.checkBox, self.checkBox_2, self.checkBox_3,
self.checkBox_4, self.checkBox_5]
        for checkbox in self.checkboxes:
            checkbox.setEnabled(False)

        # Create a QTimer to update the text at regular intervals
        self.update_timer = QTimer(self)
        self.update_timer.timeout.connect(self.update_sensor_text)
        self.update_timer.start(1000)  # Set the interval in milliseconds
(e.g., 1000 milliseconds = 1 second)
        self.mqttBroker = "broker.hivemq.com"

        # Subscribe to sensor and parking data
        self.subscribe_sensor_mqtt()
        self.subscribe_parking_mqtt()

        self.light_data = "OFF"
        self.client = mqtt.Client("ui_client")
        self.client.connect(self.mqttBroker)


    def on_sensor_message(self, client, userdata, message):
        self.sensor_data = message.payload.decode("utf-8")
        # print("Received data from RPi:", self.sensor_data)

    def subscribe_sensor_mqtt(self):
        sensor_client = mqtt.Client("sensor_client")
        sensor_client.connect(self.mqttBroker)

        sensor_client.on_message = self.on_sensor_message
        sensor_client.subscribe("sensor_dcb772")  # Subscribe to sensor data
        sensor_client.loop_start()

    def update_sensor_text(self):
        # Update the text in the QTextEdit widget
        self.sensor_textbox.setText(self.sensor_data)

    def update_checkboxes(self):
        for checkbox, (space, availability) in zip(self.checkboxes,
self.parking_data.items()):
            checkbox.setChecked(availability == 0)

    def on_parking_message(self, client, userdata, message):
        self.parking_data = json.loads(message.payload.decode("utf-8"))
```

```
        self.update_checkboxes()

    def subscribe_parking_mqtt(self):
        parking_client = mqtt.Client("parking_client")
        parking_client.connect(self.mqttBroker)

        parking_client.on_message = self.on_parking_message
        parking_client.subscribe("parking_dcb772")  # Replace with your
actual parking topic
        parking_client.loop_start()

    def send_button_click(self):
        # Sends the text input from the GUI to RPi terminal
        text_from_display = self.display_textbox.toPlainText()
        self.client.publish("display_dcb772", text_from_display)

    def off_button_click(self):
        # When clicked, the warning light/LED turns off
        # print("Off Button Clicked!")
        self.light_data = "OFF"
        self.client.publish("light_dcb772", self.light_data)

    def on_button_click(self):
        # When clicked, the warning light/LED should flash continuously
        # print("On Button Clicked!")
        self.light_data = "ON"
        self.client.publish("light_dcb772", self.light_data)

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(467, 407)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.frame_2 = QtWidgets.QFrame(self.centralwidget)
        self.frame_2.setGeometry(QtCore.QRect(180, 120, 31, 51))
        self.frame_2.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_2.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame_2.setLineWidth(10)
        self.frame_2.setObjectName("frame_2")
        self.display_textbox = QtWidgets.QTextEdit(self.centralwidget)
        self.display_textbox.setGeometry(QtCore.QRect(260, 250, 171, 81))
        self.display_textbox.setStyleSheet("QTextEdit {\n"
            "    border: 2px solid rgb(125,122,134);\n"
            "    border -radius: 10px;\n"
            "    color: #000;\n"
            "    padding-left: 10px;\n"
            "    padding-right: 10px;\n"
```

```
            "    background-color: rgb(255,255,255);\n"
            "}")
        self.display_textbox.setObjectName("display_textbox")
        self.checkBox = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox.setGeometry(QtCore.QRect(70, 90, 16, 31))
        self.checkBox.setText("")
        self.checkBox.setObjectName("checkBox")
        self.checkBox_3 = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox_3.setGeometry(QtCore.QRect(190, 90, 16, 31))
        self.checkBox_3.setText("")
        self.checkBox_3.setObjectName("checkBox_3")
        self.sensor_textbox = QtWidgets.QTextEdit(self.centralwidget)
        self.sensor_textbox.setGeometry(QtCore.QRect(40, 250, 171, 81))
        self.sensor_textbox.setStyleSheet("QTextEdit {\n"
            "    border: 2px solid rgb(125,122,134);\n"
            "    border -radius: 10px;\n"
            "    color: #000;\n"
            "    padding-left: 10px;\n"
            "    padding-right: 10px;\n"
            "    background-color: rgb(255,255,255);\n"
            "}")

self.sensor_textbox.setWordWrapMode(QtGui.QTextOption.WrapAtWordBoundaryOrAny
where)  # Enable word wrap if needed
        self.sensor_textbox.setReadOnly(True)  # Set the QTextEdit to be
read-only
        self.sensor_textbox.setObjectName("sensor_textbox")
        self.checkBox_4 = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox_4.setGeometry(QtCore.QRect(150, 90, 16, 31))
        self.checkBox_4.setText("")
        self.checkBox_4.setObjectName("checkBox_4")
        self.frame_4 = QtWidgets.QFrame(self.centralwidget)
        self.frame_4.setGeometry(QtCore.QRect(100, 120, 31, 51))
        self.frame_4.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_4.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame_4.setLineWidth(10)
        self.frame_4.setObjectName("frame_4")
        self.checkBox_5 = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox_5.setGeometry(QtCore.QRect(230, 90, 16, 31))
        self.checkBox_5.setText("")
        self.checkBox_5.setObjectName("checkBox_5")
        self.checkBox_2 = QtWidgets.QCheckBox(self.centralwidget)
        self.checkBox_2.setGeometry(QtCore.QRect(110, 90, 16, 31))
        self.checkBox_2.setText("")
        self.checkBox_2.setObjectName("checkBox_2")
        self.frame_5 = QtWidgets.QFrame(self.centralwidget)
        self.frame_5.setGeometry(QtCore.QRect(60, 120, 31, 51))
```

```
        self.frame_5.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_5.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame_5.setLineWidth(10)
        self.frame_5.setObjectName("frame_5")
        self.on_button = QtWidgets.QPushButton(self.centralwidget)
        self.on_button.setGeometry(QtCore.QRect(300, 90, 111, 41))
        self.on_button.setObjectName("on_button")
        self.frame_3 = QtWidgets.QFrame(self.centralwidget)
        self.frame_3.setGeometry(QtCore.QRect(140, 120, 31, 51))
        self.frame_3.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame_3.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame_3.setLineWidth(10)
        self.frame_3.setObjectName("frame_3")
        self.off_button = QtWidgets.QPushButton(self.centralwidget)
        self.off_button.setGeometry(QtCore.QRect(300, 140, 111, 41))
        self.off_button.setObjectName("off_button")
        self.sensor_label = QtWidgets.QLabel(self.centralwidget)
        self.sensor_label.setGeometry(QtCore.QRect(50, 210, 151, 31))
        self.sensor_label.setObjectName("sensor_label")
        self.display_label = QtWidgets.QLabel(self.centralwidget)
        self.display_label.setGeometry(QtCore.QRect(250, 210, 181, 31))
        self.display_label.setObjectName("display_label")
        self.frame = QtWidgets.QFrame(self.centralwidget)
        self.frame.setGeometry(QtCore.QRect(220, 120, 31, 51))
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Plain)
        self.frame.setLineWidth(10)
        self.frame.setObjectName("frame")
        self.parking_label = QtWidgets.QLabel(self.centralwidget)
        self.parking_label.setGeometry(QtCore.QRect(110, 30, 241, 41))
        self.parking_label.setObjectName("parking_label")
        self.send_button = QtWidgets.QPushButton(self.centralwidget)
        self.send_button.setGeometry(QtCore.QRect(300, 340, 101, 31))
        self.send_button.setObjectName("send_button")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.on_button.setText(_translate("MainWindow", "ON"))
        self.off_button.setText(_translate("MainWindow", "OFF"))
```

```
        self.sensor_label.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" font-size:18pt;
font-weight:600; color:#3c3648;\">Sensor Data</span></p></body></html>"))
        self.display_label.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" font-size:18pt;
font-weight:600; color:#3c3648;\">Display Board</span></p><p
align=\"center\"><br/></p></body></html>"))
        self.parking_label.setText(_translate("MainWindow",
"<html><head/><body><p align=\"center\"><span style=\" font-size:pt;
font-weight:600; color:#3c3648;\">Parking System</span></p></body></html>"))
        self.send_button.setText(_translate("MainWindow", "Send"))


if __name__ == "__main__":
    import sys
    app = QApplication(sys.argv)
    mainWindow = Ui_MainWindow()
    mainWindow.show()
    sys.exit(app.exec_())
```

RPi code:

```
# This is the script we run from the RPi
import RPi.GPIO as GPIO
import PCF8591 as ADC
import time
import paho.mqtt.client as mqtt
import json

BtnPin = 11
Gpin   = 12
Rpin   = 13
light_flash = 0
sensor_on = False


# Dictionary to easily iterate through each of them
parking_pins = {
    "P1": 35,
    "P2": 37,
    "P3": 36,
    "P4": 38,
    "P5": 40
```

```
}

# To setup all components
def setup():
      GPIO.setmode(GPIO.BOARD)
      GPIO.setup(Gpin, GPIO.OUT)
      GPIO.setup(Rpin, GPIO.OUT)
      GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

      # Sets up each pins for the parking space
      for pin in parking_pins.values():
            GPIO.setup(pin, GPIO.IN)

      ADC.setup(0x48)

      # Starts the LED as OFF
      GPIO.output(Gpin, GPIO.LOW)
      GPIO.output(Rpin, GPIO.LOW)

def read_space_availabilities():
      # This creates a dictionary to check the availability of each parking
space
      # Should look like this {'P1': 1, 'P2': 1, 'P3': 0, 'P4': 0, 'P5': 0}
    space_availabilities = {}
    for space, pin in parking_pins.items():
        availability = GPIO.input(pin)
        space_availabilities[space] = availability
    return space_availabilities


# This is ran when the ON/OFF button on the GUI is clicked
# It sets light_flash to 1 and flashes the LED
def light_on_message(client, userdata, message):
      global light_flash
      light_data = message.payload.decode("utf-8")
      print("Light data from client: ", light_data)

      if (light_data == "ON"):
            light_flash = 1
      else:
            light_flash = 0

# This is ran when the SEND button on the GUI is clicked
def display_on_message(client, userdata, message):
      display_data = message.payload.decode("utf-8")
      print("Display board: ", display_data)
```

```
def check_button_press():
	global sensor_on
	if GPIO.input(BtnPin) == 0:
		sensor_on = True
	if GPIO.input(BtnPin) == 1:
		sensor_on = False

def flash_green():
	# Flashes LED when light_flash is 1
	if light_flash == 1:
		GPIO.output(Gpin, GPIO.HIGH)
		time.sleep(0.2)
		GPIO.output(Gpin, GPIO.LOW)
		time.sleep(0.2)
		GPIO.output(Gpin, GPIO.HIGH)
		time.sleep(0.2)
		GPIO.output(Gpin, GPIO.LOW)


def read_sound_sensor():
	sensor_data = ""
	if sensor_on:
		sensor_data += "Sound sensor is on!\n"
		voice_value = ADC.read(0)
		if voice_value:
			sensor_data += "Value: " + str(voice_value) + " "
			if voice_value < 70:
				sensor_data += "Voice In!! "
	else:
		sensor_data += "Sound sensor is off! Hold button to turn sensor
on. "

	return sensor_data

def publish():
	client.publish("sensor_dcb772", read_sound_sensor())

	json_data = json.dumps(read_space_availabilities())
	client.publish("parking_dcb772", json_data)

# Loop to run for the button/sensor and publishing
def loop():
	while True:
		check_button_press()
		flash_green()
		publish()
```

```
            time.sleep(0.5)

# Turns off LEDs at the end
def destroy():
      GPIO.output(Gpin, GPIO.LOW)
      GPIO.output(Rpin, GPIO.LOW)
      GPIO.cleanup()


if __name__ == '__main__':
      mqttBroker = "broker.hivemq.com"

      client = mqtt.Client("pi_client")
      client.connect(mqttBroker)

      light_client = mqtt.Client("light_client")
      light_client.connect(mqttBroker)

      light_client.on_message = light_on_message
      light_client.subscribe("light_dcb772")
      light_client.loop_start()

      display_client = mqtt.Client("display_client")
      display_client.connect(mqttBroker)

      display_client.on_message = display_on_message
      display_client.subscribe("display_dcb772")
      display_client.loop_start()

      try:
            setup()
            loop()
      except KeyboardInterrupt:
            destroy()
```