QRCODE 1

import cv2

import numpy as np

# set up camera object

from picamera2 import Picamera2

import time

picam2 = Picamera2()

picam2.start(show\_preview=True)

while True:

time.sleep(1)

image = picam2.capture\_image("main")

cv2\_img = np.array(image)

opencvImage = cv2.cvtColor(cv2\_img, cv2.COLOR\_RGB2BGR)

qcd = cv2.QRCodeDetector()

retval, decoded\_info, points, straight\_qrcode = qcd.detectAndDecodeMulti(opencvImage)

print(retval)

print(decoded\_info)

QRCODE 2

import cv2

import numpy as np

from picamera2 import Picamera2

from pyzbar.pyzbar import decode

import time

picam2 = Picamera2()

picam2.start(show\_preview=True)

while True:

time.sleep(1)

image = picam2.capture\_image("main")

cv2\_img = np.array(image)

detectedBarcodes = decode(cv2\_img)

for barcode in detectedBarcodes:

(x, y, w, h) = barcode.rect

cv2.rectangle(cv2\_img, (x, y), (x + w, y + h), (255, 0, 0), 5)

qrcode = str(barcode.data)

qrcode = qrcode.replace("'","")

qrcode = qrcode.lstrip("b")

print(qrcode)

QRGUI

import cv2

import numpy as np

from picamera2 import Picamera2,Preview

from pyzbar.pyzbar import decode

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QApplication, QWidget , QTableWidget , QTableWidgetItem

from libcamera import Transform

picam2 = Picamera2()

picam2.start\_preview(Preview.QTGL, x=10, y=10, width=200, height=200,

transform=Transform(hflip=1))

picam2.start()

class ThreadRFID(QtCore.QThread):

signal = QtCore.pyqtSignal(str, str, int)

successSignal = QtCore.pyqtSignal()

def \_\_del\_\_(self):

self.wait()

def run(self):

#Initialize RFID Reader

while True:

#if \_id == 955738786217: #Validate card is company ID

print(" scan card success")

\_id=0

self.successSignal.emit()

self.signal.emit("ID#:"+str(\_id), str(\_id), 10)

class MainWindow(QtWidgets.QMainWindow):

def \_\_init\_\_(self, parent=None):

super(MainWindow, self).\_\_init\_\_(parent)

self.title = ' '

self.left = 420

self.top = 106

self.width = 800

self.height = 600

self.setFixedSize(500,300)

self.setWindowFlags( QtCore.Qt.CustomizeWindowHint ) #Turns off Min/Max/Close buttons

self.lbl = QtWidgets.QLabel('', self)

self.lbl.setGeometry(10, 10, 321,41)

# creating a QTableWidget

table = QTableWidget(self)

table.setRowCount(2)

table.setColumnCount(3)

table.setGeometry(10 , 50, 350 , 150)

self.initUI()

self.myThread = ThreadRFID(self)

self.myThread.start()

# self.myThread.successSignal.connect(self.PunchDialog.show)

self.myThread.signal.connect(self.onPunch)

@QtCore.pyqtSlot(str, str, int)

def onPunch(self, strA, strB, int1):

self.lbl.setText(strA)

def initUI(self):

self.setWindowTitle(self.title)

self.setGeometry(self.left, self.top, self.width, self.height)

def closeEvent(self, event):

GPIO.cleanup()

super(MainWindow, self).closeEvent(event)

if \_\_name\_\_ == '\_\_main\_\_':

import sys

app = QtWidgets.QApplication(sys.argv)

MW = MainWindow()

MW.show()

sys.exit(app.exec\_())