PEOPLE'S EDUCATION SOCIETY

SIDDHARTH COLLEGE OF ARTS, SCIENCE & COMMERCE

(BUDDHA BHAVAN)

(Affiliated to University of Mumbai)

PURSHOTTAMDAS THAKURDAS MARG, OUTRAM ROAD, FORT, MUMBAI, MAHARASHTRA 400001

DEPARTMENT OF BACHELOR OF ARTS

CERTIFICATE

Class: **T.Y.B.A** Year: **2022 - 2023**

This is to certify that the project entitled **Psychology** in this journal is the work of **Mr. Ritik Suresh Poojari** of **T.Y.B.A** Roll No. **A-220193** has satisfactorily completed the required number of practicals and worked for **Semester 5** of the academic year 20**22** – 20**23** in the college as laid down by the university.

_____	_____	_____
Coordinator	**External Examiner**	**Internal Examiner**

_____

Date: College Seal

INDEX

Sr.No.	Practical Name	Page No.	Date	Remark
01.	Displaying Different LED patterns using Raspberry Pi	3-5	09/08/2022	
02.	Displaying Time Over 4-Digit-7 Segment display using Raspberry Pi	6-8	16/08/2022	
03.	Raspberry Pi based Oscilloscope	9-10	27/08/2022	
04.	Fingerprint sensor interfacing with Raspberry Pi	11-17	13/09/2022	
05.	Raspberry Pi GPS module Interface	18-24	20/09/2022	
06.	visitor Monitoring with Raspberry Pi & Pi Camera	25-26	27/09/2022	
07.	Interfacing Raspberry Pi with RFID	27-31	11/10/2022	

Practical 1: Displaying Different LED patterns using Raspberry Pi

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(12,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(16,GPIO.OUT)
GPIO.setup(20,GPIO.OUT)
```

```
GPIO.setup(21,GPIO.OUT)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(24,GPIO.OUT)
GPIO.setup(25,GPIO.OUT)

switch0 = 7
switch1 = 8

switchPressCount = 0

GPIO.setup(switch0, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(switch1, GPIO.IN, pull_up_down=GPIO.PUD_UP)

pins = [12,13,16,20,21,23,24,25]


def blink():
 for j in range(0,8):
 GPIO.output(pins[j],False)
 time.sleep(0.5)
 for j in range(0,8):
```



```
GPIO.output(pins[j],True)
 time.sleep(0.5)

def toggle():
 for j in range(0,8):
 if j%2 == 0:
 GPIO.output(pins[j],False)  else:
 GPIO.output(pins[j],True)
time.sleep(0.5)
```

```
 for j in range(0,8):
 if j%2 == 1:
 GPIO.output(pins[j],False)  else:
 GPIO.output(pins[j],True)
time.sleep(0.5)

def incremental():
 for j in range(0,8):
 GPIO.output(pins[j],False)
 for j in range(0,8):
 GPIO.output(pins[j],True)
 time.sleep(0.5)

print('LED Pattern')
while True:
 try:
 switchPressed = GPIO.input(switch0)  if
switchPressed == False:
 switchPressCount += 1
 switchPressCount %= 3
 if switchPressCount == 0:
```



```
 blink();
 if switchPressCount == 1:
toggle();
 if switchPressCount == 2:
incremental();
 except KeyboardInterrupt as e:
print('Cleaning GPIO')
GPIO.cleanup()
 print('Cleaning GPIO')
 GPIO.cleanup()
```

Practical 2: Displaying Time Over 4-Digit-7 Segment display using Raspberry Pi

```
import RPi.GPIO as GPIO
import time, datetime
now = datetime.datetime.now()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
#GPIO ports for the 7seg pins
segment8 = (26,19,13,6,5,11,9,10)
for segment in segment8:
```

```
 GPIO.setup(segment, GPIO.OUT)
 GPIO.output(segment, 0)
#Digit 1
GPIO.setup(7, GPIO.OUT)
GPIO.output(7, 0) #Off initially
#Digit 2
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, 0) #Off initially
#Digit 3
GPIO.setup(25, GPIO.OUT)
GPIO.output(25, 0) #Off initially
#Digit 4
GPIO.setup(24, GPIO.OUT)
GPIO.output(24, 0) #Off initially
null = [0,0,0,0,0,0,0]
zero = [1,1,1,1,1,1,0]
one = [0,1,1,0,0,0,0]
two = [1,1,0,1,1,0,1]
three = [1,1,1,1,0,0,1]
four = [0,1,1,0,0,1,1]
five = [1,0,1,1,0,1,1]
```



```
six = [1,0,1,1,1,1,1]
seven = [1,1,1,0,0,0,0]
eight = [1,1,1,1,1,1,1]
nine = [1,1,1,1,0,1,1]
def print_segment(charector):
 if charector == 1:
 for i in range(7):
 GPIO.output(segment8[i], one[i])  if
charector == 2:
 for i in range(7):
```

```
 GPIO.output(segment8[i], two[i])  if
charector == 3:
 for i in range(7):
 GPIO.output(segment8[i], three[i])  if
charector == 4:
 for i in range(7):
 GPIO.output(segment8[i], four[i])  if
charector == 5:
 for i in range(7):
 GPIO.output(segment8[i], five[i])  if
charector == 6:
 for i in range(7):
 GPIO.output(segment8[i], six[i])  if
charector == 7:
 for i in range(7):
 GPIO.output(segment8[i], seven[i])  if
charector == 8:
 for i in range(7):
 GPIO.output(segment8[i], eight[i])  if
charector == 9:
 for i in range(7):
 GPIO.output(segment8[i], nine[i])  if
charector == 0:
```



```
 for i in range(7):
 GPIO.output(segment8[i], zero[i])
return;
 while 1:
 now = datetime.datetime.now()
 hour = now.minute
 minute = now.second
 h1 = hour/10
```

```
 h2 = hour % 10
 m1 = minute /10
 m2 = minute % 10
 print (h1,h2,m1,m2)
 delay_time = 0.001 #delay to create virtual effect

 GPIO.output(7, 1) #Turn on Digit One
print_segment (h1) #Print h1 on segment
time.sleep(delay_time)
 GPIO.output(7, 0) #Turn off Digit One
GPIO.output(8, 1) #Turn on Digit One
print_segment (h2) #Print h1 on segment
GPIO.output(10, 1) #Display point On
time.sleep(delay_time)
  GPIO.output(10, 0) #Display point Off
GPIO.output(8, 0) #Turn off Digit One
GPIO.output(25, 1) #Turn on Digit One
 print_segment (m1) #Print h1 on segment
time.sleep(delay_time)
 GPIO.output(25, 0) #Turn off Digit One
GPIO.output(24, 1) #Turn on Digit One
print_segment (m2) #Print h1 on segment
time.sleep(delay_time)
 GPIO.output(24, 0) #Turn off Digit One
#time.sleep(1)
```


Practical 3: Raspberry Pi based Oscilloscope

```
import time
import matplotlib.pyplot as plt
#import numpy
from drawnow import *
# Import the ADS1x15 module.
```

```
import Adafruit_ADS1x15
# Create an ADS1115 ADC (16-bit) instance.
adc = Adafruit_ADS1x15.ADS1115()

GAIN = 1
val = [ ]
cnt = 0
plt.ion()
# Start continuous ADC conversions on channel 0 using the previous gain
value. adc.start_adc(0, gain=GAIN)
print('Reading ADS1x15 channel 0')
#create the figure function
def makeFig():
 plt.ylim(-5000,5000)
 plt.title('Osciloscope')
 plt.grid(True)
 plt.ylabel('ADC outputs')
 plt.plot(val, 'ro-', label='Channel 0')
 plt.legend(loc='lower right')
while (True):
 # Read the last ADC conversion value and print it out.
 value = adc.get_last_result()
 print('Channel 0: {0}'.format(value))
 # Sleep for half a second.
 time.sleep(0.5)
```



```
 val.append(int(value))
drawnow(makeFig)
plt.pause(.000001)  cnt
= cnt+1
 if(cnt>50):
 val.pop(0)
```

Practical 4: Fingerprint sensor interfacing with Raspberry Pi

```
import binascii

import time


class GT511C3:
```

```
def __init__(self, serial):
self.serial = serial

def writeCMD(self, paramBytes, cmdBytes):
paramBytes.reverse()
cmdBytes.reverse()
checksum = 256
data = []
i = 0;
data.append(0x55)
data.append(0xAA)
data.append(0x01)
data.append(0x00)
for param in paramBytes:
data.append(param & 0xFF)
checksum += (param & 0xFF)
for cmd in cmdBytes:
data.append(cmd & 0xFF)
checksum += (cmd & 0xFF)
data.append(checksum & 0xFF)
data.append(checksum>>8 & 0xFF)
#print(data)
self.serial.write(data)

def getACK(self):
```



```
data = bytearray(12)
self.serial.readinto(data)
resp = ""
for c in data:
resp += str(c) + " "
#sprint(resp)
```

```
return data

def isACK(self, response):
return response[8] == 0x30

def open(self):
self.writeCMD([0,0,0,0], [0, 1])
return self.isACK(self.getACK())

def cmosLED(self, turnOn):
param = [0,0,0,0]
if turnOn:
param[3] = 1
self.writeCMD(param, [0, 0x12])
return self.isACK(self.getACK())

def isPressFinger(self):
self.writeCMD([0,0,0,0], [0, 0x26])
resp = self.getACK()
if self.isACK(resp):
respParam = resp[4]+resp[5]+resp[6]+resp[7]
return respParam == 0
return False

def captureFinger(self, goodQuality=False):
print('Put Finger On Sensor')
```



```
while not self.isPressFinger():
time.sleep(0.01)
param = [0,0,0,0]
if goodQuality:
param[3] = 1
```

```
 self.writeCMD(param, [0, 0x60])
return self.isACK(self.getACK())
 def enrollStart(self, id):
 if id == -1:
 param = [0xFF, 0xFF, 0xFF, 0xFF]  else:
 param = [0, 0, 0, id & 0xFF]
self.writeCMD(param, [0, 0x22])
resp = self.getACK()
 if not self.isACK(resp):
 return resp[6]<<8 + resp[7]
return 0xFFFF

 def enrollFirst(self):
 return self.enroll(0x23)

 def enrollSecond(self):
 return self.enroll(0x24)

 def enrollThird(self):
 return self.enroll(0x25)

 def enroll(self, count):
 param = [0,0,0,0]
 self.writeCMD(param, [0, count&0xFF])
resp = self.getACK()
```



```
if not self.isACK(resp):
 return resp[6]<<8 + resp[7]
 return 0xFFFF

 def identify(self):
 self.writeCMD([0,0,0,0], [0, 0x51])
```

```
 resp = self.getACK()
 if self.isACK(resp):
 respParam = resp[4]
 return respParam
 return -1

 def deleteAll(self):
 self.writeCMD([0,0,0,0], [0, 0x41])
 return self.isACK(self.getACK())
FingerPrintTest
import FingerPrintGT511C3
import serial
import time

serialPort = serial.Serial("/dev/ttyUSB0", baudrate=9600,
timeout=1) if not serialPort.isOpen():
 serialPort.open()
print('Serial Port Opend!')
fps = FingerPrintGT511C3.GT511C3(serialPort)


def enroll():
 print('Enter ID(0-199) for storing fingerprint')
 id = input('Please Enter Unused ID')
 print('Enrolling at ' + str(id))
 fps.cmosLED(True)
```



```
fps.enrollStart(int(id))
 fps.captureFinger(True)
fps.enrollFirst()
 print('Remove Finger')
```

```
while fps.isPressFinger():
time.sleep(0.1)
 fps.captureFinger(True)
fps.enrollSecond()
 print('Remove Finger')
while fps.isPressFinger():
time.sleep(0.1)
 fps.captureFinger(True)
fps.enrollThird()
 print('Remove Finger')
while fps.isPressFinger():
time.sleep(0.1)
 print('Enroll Success ' + str(id))
fps.cmosLED(False)
 print(' ')

def openFPS():
 print('Opening FPS')
 if fps.open():
 print('FPS Open Sucess')
else:
 print('FPS Open Failed')
raise Exception('')

def blinkLED():
 if fps.cmosLED(True):
 print('FPS LED ON')
 time.sleep(2)
```



```
 fps.cmosLED(False)
 else:
 print('FAILED: FPS LED ON')
```

```
 raise Exception('Error While Blinking LED')

def search():
 print('Searching...')
 fps.cmosLED(True)
 fps.captureFinger(True)
 foundAt = fps.identify()
 if not foundAt == -1:
 print('Match Foundd at ' + str(foundAt))
else:
 print('Not Found')
 fps.cmosLED(False)
 print('')


def deleteAll():
 print('Deleting All Previous Records')
fps.deleteAll()
 time.sleep(00.0001)

openFPS()

#blinkLED()

while(True):
 print('Select Operation To Perform :')
print('1. Delete All Existing Records')
print('2. Enroll New Fingerprint')
 print('3. Search Fingerprint')
```



```
 print('4. Exit')
 option = input()
```

```
print(option)
if option == 1:
deleteAll()
if option == 2:
enroll()
if option == 3:
search()
if option == 4:
break
print('Exiting program...')
```


Practical 5: Raspberry Pi GPS module Interface

```
import time
```

```
import serial
import string
import pynmea2
import RPi.GPIO as GPIO

# Refer GPIO PIN Numbers
LCD_RS = 7
LCD_E = 8
LCD_D4 = 21
LCD_D5 = 23
LCD_D6 = 24
LCD_D7 = 25

LCD_WIDTH = 16
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0X80
LCD_LINE_2 = 0XC0

E_PULSE = 0.0005
E_DELAY = 0.0005

GPIO.setmode(GPIO.BCM)
GPIO.setup(LCD_E, GPIO.OUT)
GPIO.setup(LCD_RS, GPIO.OUT)
GPIO.setup(LCD_D4, GPIO.OUT)
GPIO.setup(LCD_D5, GPIO.OUT)
GPIO.setup(LCD_D6, GPIO.OUT)
GPIO.setup(LCD_D7, GPIO.OUT)
```



```
def lcd_init():
  lcd_byte(0X33,LCD_CMD)
```

```
lcd_byte(0X32,LCD_CMD)
lcd_byte(0X28,LCD_CMD)
lcd_byte(0X0C,LCD_CMD)
lcd_byte(0X06,LCD_CMD)
lcd_byte(0X01,LCD_CMD)

def lcd_string(msg):
 msg = msg.ljust(LCD_WIDTH, ' ')
for i in range(LCD_WIDTH):
lcd_byte(ord(msg[i]),LCD_CHR)

def lcd_byte(bits,mode):
   GPIO.output(LCD_RS,   mode)
GPIO.output(LCD_D4,      False)
GPIO.output(LCD_D5,      False)
GPIO.output(LCD_D6,      False)
GPIO.output(LCD_D7, False)
 if bits&0x10==0x10:
 GPIO.output(LCD_D4, True)  if
bits&0x20==0x20:
 GPIO.output(LCD_D5, True)  if
bits&0x40==0x40:
 GPIO.output(LCD_D6, True)  if
bits&0x80==0x80:
 GPIO.output(LCD_D7, True)
time.sleep(E_DELAY)
 GPIO.output(LCD_E, True)
time.sleep(E_PULSE)
 GPIO.output(LCD_E, False)
```



```
 time.sleep(E_DELAY)

 GPIO.output(LCD_D4, False)
```

```
GPIO.output(LCD_D5, False)
GPIO.output(LCD_D6, False)
GPIO.output(LCD_D7, False)

if bits&0x01==0x01:
GPIO.output(LCD_D4, True)
if bits&0x02==0x02:
GPIO.output(LCD_D5, True)
if bits&0x04==0x04:
GPIO.output(LCD_D6, True)
if bits&0x08==0x08:
GPIO.output(LCD_D7, True)
time.sleep(E_DELAY)
GPIO.output(LCD_E, True)
time.sleep(E_PULSE)
GPIO.output(LCD_E, False)
time.sleep(E_DELAY)


lcd_init()
lcd_byte(LCD_LINE_1, LCD_CMD)

#create a serial object
ser = serial.Serial("/dev/ttyUSB0", baudrate = 9600, timeout = 0.5)

while 1:
try:
data = ser.readline()
print("Reading ..." + data)
```



```
#wait for the serial port to churn out data
if data[0:6] == '$GPGGA': # the long and lat data are always contained in the GPGGA string of the NMEA
```

```
data
 msg = pynmea2.parse(data)
 #parse the latitude and print
 latval = msg.lat
 concatlat = "lat:" + str(latval)
 print(concatlat)
 lcd_byte(LCD_LINE_1, LCD_CMD)
 lcd_string(concatlat)

 #parse the longitude and print
 longval = msg.lon
 concatlong = "long:"+ str(longval)
 print(concatlong)
 lcd_byte(LCD_LINE_2, LCD_CMD)
 lcd_string(concatlong)

 time.sleep(0.5)#wait a little before picking the next data.
 finally:
 GPIO.cleanup()
```

LCD

```
import RPi.GPIO as GPIO
import time

# Refer GPIO PIN Numbers
LCD_RS = 7
LCD_E = 8
LCD_D4 = 21
LCD_D5 = 23
LCD_D6 = 24
```



```
LCD_D7 = 25
```

```
LCD_WIDTH = 16
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0X80
LCD_LINE_2 = 0XC0

E_PULSE = 0.0005
E_DELAY = 0.0005

def main():
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(LCD_E, GPIO.OUT)
        GPIO.setup(LCD_RS,  GPIO.OUT)
        GPIO.setup(LCD_D4, GPIO.OUT)
        GPIO.setup(LCD_D5, GPIO.OUT)
        GPIO.setup(LCD_D6, GPIO.OUT)
        GPIO.setup(LCD_D7, GPIO.OUT)

        lcd_init()

        lcd_byte(LCD_LINE_1,
        LCD_CMD) lcd_string("Future
        Chip")
        lcd_byte(LCD_LINE_2,
        LCD_CMD)
        lcd_string("Technologies")
        time.sleep(3)

def lcd_init():
        lcd_byte(0X33,LCD_CMD)
        lcd_byte(0X32,LCD_CMD)
        lcd_byte(0X28,LCD_CMD)
```

```
 lcd_byte(0X0C,LCD_CMD)
        lcd_byte(0X06,LCD_CMD)
        lcd_byte(0X01,LCD_CMD)

def lcd_string(msg):
        msg = msg.ljust(LCD_WIDTH, ' ')
        for i in range(LCD_WIDTH):
                lcd_byte(ord(msg[i]),LCD_CHR)

def lcd_byte(bits,mode):
        GPIO.output(LCD_RS, mode)
        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)

        if bits&0x10==0x10:
                GPIO.output(LCD_D4, True)
        if bits&0x20==0x20:
                GPIO.output(LCD_D5, True)
        if bits&0x40==0x40:
                GPIO.output(LCD_D6, True)
        if bits&0x80==0x80:
                GPIO.output(LCD_D7, True)
        time.sleep(E_DELAY)
        GPIO.output(LCD_E, True)
        time.sleep(E_PULSE)
        GPIO.output(LCD_E, False)
        time.sleep(E_DELAY)

        GPIO.output(LCD_D4, False)
        GPIO.output(LCD_D5, False)
```

```
        GPIO.output(LCD_D6, False)
        GPIO.output(LCD_D7, False)

        if bits&0x01==0x01:
                GPIO.output(LCD_D4, True)
        if bits&0x02==0x02:
                GPIO.output(LCD_D5, True)
        if bits&0x04==0x04:
                GPIO.output(LCD_D6, True)
        if bits&0x08==0x08:
                GPIO.output(LCD_D7, True)
        time.sleep(E_DELAY)
        GPIO.output(LCD_E, True)
        time.sleep(E_PULSE)
        GPIO.output(LCD_E, False)
        time.sleep(E_DELAY)
if __name__=='__main__':
 try:
 print("Running LCD Code...")  main()
 finally:
 print("Cleanning GPIO")
 GPIO.cleanup()
```

Practical 6: visitor Monitoring with Raspberry Pi & Pi Camera

```
import RPi.GPIO as GPIO
import time
import picamera
import datetime

buzzer = 6
switch0 = 7
readyLED = 8

MOTOR_P = 12
MOTOR_M = 13

GPIO.setmode(GPIO.BCM)
GPIO.setup(buzzer,GPIO.OUT)
GPIO.setup(readyLED,GPIO.OUT)
GPIO.setup(MOTOR_P,GPIO.OUT)
GPIO.setup(MOTOR_M,GPIO.OUT)

GPIO.setup(switch0, GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.output(MOTOR_P, False)
GPIO.output(MOTOR_M, False)
GPIO.output(buzzer, False)
GPIO.output(readyLED, True)

camera = picamera.PiCamera()
print('Running Visitor Monitoring ..')
while True:
 try:
 switchPressed = GPIO.input(switch0)
```

```
 if switchPressed == False:
 GPIO.output(readyLED, False)
 GPIO.output(buzzer, True)
 while GPIO.input(switch0) == False:
 time.sleep(0.001)
 GPIO.output(buzzer, False)
 camera.start_preview()
 time.sleep(2)
 camera.stop_preview()
 timestamp = datetime.datetime.now().strftime("%m_%d_%Y_%H_%M_%S")
imagename = "images/visitor_" + timestamp + ".jpg"
 print('Capturing Image ' + imagename)
 camera.capture(imagename)
 print('Opening Door...')
 GPIO.output(MOTOR_P, True)
 GPIO.output(MOTOR_M, False)
 time.sleep(0.8)
 GPIO.output(MOTOR_P, False)
 GPIO.output(MOTOR_M, False)
 time.sleep(1.5)
 GPIO.output(MOTOR_P, False)
 GPIO.output(MOTOR_M, True)
 print('Closing Door...')
 time.sleep(0.81)
 GPIO.output(MOTOR_P, False)
 GPIO.output(MOTOR_M, False)

 GPIO.output(readyLED, True)
 except Exception as e:
 print('Cleaning GPIO')
 GPIO.cleanup()
GPIO.cleanup()
```

Practical 7: Interfacing Raspberry Pi with RFID

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#
# Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#
# This file is part of MFRC522-Python
# MFRC522-Python is a simple Python implementation for
# the MFRC522 NFC Card Reader for the Raspberry Pi.
#
# MFRC522-Python is free software: you can redistribute it and/or modify # it
under the terms of the GNU Lesser General Public License as published by #
the Free Software Foundation, either version 3 of the License, or # (at your
option) any later version.
#
# MFRC522-Python is distributed in the hope that it will be useful, # but
WITHOUT ANY WARRANTY; without even the implied warranty of #
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the #
GNU Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public License
# along with MFRC522-Python. If not, see <http://www.gnu.org/licenses/>. #

'''

| Name | Pin # | Pin name |
|:------:|:-------:|:------------:|
| SDA | 24 | GPIO8 |
| SCK | 23 | GPIO11 |
```



```
| MOSI | 19 | GPIO10 |
```

```
| MISO | 21 | GPIO9 |
| IRQ | None | None |
| GND | Any | Any Ground |
| RST | 22 | GPIO25 |
| 3.3V | 1 | 3V3 |

'''

import RPi.GPIO as GPIO
import MFRC522
import signal
import time

continue_reading = True

buzzer = 33 #13
errorLED = 35 #19
readyLED = 37 #26

GPIO.setmode(GPIO.BOARD)
GPIO.setup(buzzer,GPIO.OUT)
GPIO.setup(readyLED,GPIO.OUT)
GPIO.setup(errorLED,GPIO.OUT)

GPIO.output(buzzer, False)
GPIO.output(errorLED, False)
GPIO.output(readyLED, False)

# Capture SIGINT for cleanup when the script is
aborted def end_read(signal,frame):
 global continue_reading
```

```
 print ("Ctrl+C captured, ending read.")
continue_reading = False
 GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
 MIFAREReader = MFRC522.MFRC522()

# Welcome message
print("Welcome to the MFRC522 data read example")
print("Press Ctrl-C to stop.")

orignalID = ""

def getHexChar(d):
 if d > 9:
 if d == 10:
 return "A"
 if d == 11:
 return "B"
 if d == 12:
 return "C"
 if d == 13:
 return "D"
 if d == 14:
 return "E"
 if d == 15:
 return "F"
 else:
 return str(d)
```

```
def getHex(no):
 LSB = no & 0x0F
 MSB = (no>>4) & 0x0F
 return getHexChar(MSB) + getHexChar(LSB)



# This loop keeps checking for chips. If one is near it will get the UID and authenticate
while continue_reading:

 # Scan for cards
 (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

 # If a card is found
 if status == MIFAREReader.MI_OK:
 print("Card detected")

 # Get the UID of the card
 (status,uid) = MIFAREReader.MFRC522_Anticoll()

 # If we have the UID, continue
 if status == MIFAREReader.MI_OK:
 GPIO.output(buzzer, True)
 # Print UID
 rfid = getHex(uid[0]) + " " + getHex(uid[1]) + " " + getHex(uid[2]) + " " + getHex(uid[3])
print("Card read UID: " + rfid)
 if len(orignalID) < 2:
 orignalID = rfid
 if rfid == orignalID:
 print('Approved')
 GPIO.output(readyLED, True)
 else:
```

```
print('Rejected')


GPIO.output(errorLED, True)
time.sleep(1)
GPIO.output(readyLED, False)
GPIO.output(errorLED, False)
GPIO.output(buzzer, False)



'''
# This is the default key for authentication
key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

# Select the scanned tag
MIFAREReader.MFRC522_SelectTag(uid)

# Authenticate
status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)

# Check if authenticated
if status == MIFAREReader.MI_OK:
MIFAREReader.MFRC522_Read(8)
MIFAREReader.MFRC522_StopCrypto1()
else:
print("Authentication error")
```