**CODING:**

#!/usr/bin/python

import time

import RPi.GPIO as GPIO

import time

import os, sys

from urllib.parse import urlparse import paho.mqtt.client as paho

GPIO.setmode (GPIO.BOARD)

GPIO.setwarnings (False)

#define pin for lcd.

# Timing constants

E\_PULSE = 0.0005

E\_DELAY 0.0005

delay = 1

# Define GPIO to LCD mapping

LCD\_RS = 7

LCD\_E = 11

LCD\_D4 = 12

LCD\_D5 = 13

LCD\_D6 = 15

LCD D7 = 16

slot1\_Sensor = 29

slot2\_Sensor = 31

GPIO.setup(LCD\_E, GPIO.OUT) # E

GPIO.setup(LCD\_RS, GPIO.OUT) # RS

GPIO.setup(LCD\_D4, GPIO.OUT) # DB4 GPIO.setup(LCD\_D5, GPIO.OUT) # DB5

GPIO.setup(LCD\_D6, GPIO.OUT) # DB6

GPIO.setup(LCD\_D7, GPIO.OUT) # DB7 GPIO.setup(slot1\_Sensor, GPIO.IN)

GPIO.setup(slot2\_Sensor, GPIO.IN)

# Define some device constants

LCD\_WIDTH = 16 # Maximum characters per line

LCD\_CHR = True

LCD\_CMD= False

LCD\_LINE\_1 = 0x80 # LCD RAM address for the 1st line.

LCD\_LINE\_2 = 0xC0 # LCD RAM address for the 2nd line

LCD\_LINE\_3 = 0x90# LCD RAM address for the 3nd line

def on\_connect(self, mosq, obj, rc):

self.subscribe("Fan", 0)

def on\_publish(mosq, obj, mid):

print("mid: " + str(mid)) mqttc paho.Client()

object declaration

# Assign event callbacks mqttc.on\_connect on connect mqttc.on\_publish on publish

url\_str=os.environ.get('CLOUDMQTT\_URL', 'tcp://broker.emqx.io: 1883') url =

urlparse(url\_str) mqttc.connect(url.hostname, url.port)

Function Name :lcd\_init()

Function Description: this function is used to initialized lcd by sending the different

def lcd\_init():

# Initialise display

lcd\_byte(0x33, LCD\_CMD)

# 110011 Initialise

lcd\_byte(0x32, LCD\_CMD)

# 110010 Initialise

lcd\_byte(0x06, LCD\_CMD)

# 000110 Cursor move

direction

lcd\_byte(0x0C, LCD\_CMD)

# 001100 Display On,

Cursor Off, Blink off

lcd\_byte(0x28, LCD\_CMD) #

101000 Data length,

number of lines, font size

lcd\_byte(0x01, LCD\_CMD) #

000001 Clear display

time.sleep(E\_DELAY)

Function Name: lcd\_byte(bits,mode)

Fuction Name: the main purpose of this function to convert the byte data into bit and send to

lcd port

def lcd\_byte(bits, mode):

# Send byte to data pins

# bits = data

# mode = True for character

#False for command

GPIO.output (LCD\_RS,

mode) # RS

# High bits

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)

# Toggle 'Enable' pin lcd\_toggle\_enable()

# Low bits

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)

# Toggle 'Enable' pin lcd\_toggle\_enable()

Function Name: lcd\_toggle\_enable()

Function Description: basically this is used to toggle Enable pin

def lcd\_toggle\_enable():

# Toggle enable

time.sleep(E\_DELAY)

GPIO.output (LCD\_E, True)

time.sleep(E\_PULSE)

GPIO.output (LCD\_E, False)

time.sleep(E\_DELAY)

Function Name : lcd\_string(message, line) Function Description: print the data on lcd

def lcd\_string (message, line):

# Send string to display

message= message.1just

(LCD\_WIDTH," ")

lcd\_byte(line, LCD\_CMD)

for i in range(LCD\_WIDTH):

lcd\_byte(ord(message[i]),

LCD\_CHR)

lcd\_init()

lcd\_string("welcome",LCD\_LINE\_1)

time.sleep(0.5)

lcd\_string("Car Parking ", LCD\_LINE\_1)

lcd\_string("System ", LCD\_LINE\_2) time.sleep(0.5)

lcd\_byte(0x01, LCD\_CMD) # 000001 Clear display

# Define delay between readings delay = 5

while 1:

# Print out results

rc = mqttc.loop()

slot1\_status = GPIO. input

(slot1\_Sensor)

time.sleep(0.2)

slot2\_status = GPIO.input

(slot2\_Sensor)

time.sleep(0.2)

if (slot1\_status==False):

lcd\_string("Slot1 Parked ',

LCD\_LINE\_1)

mqttc.publish("slot1", "1")

time.sleep(0.2)

else:

lcd\_string("Slot1 Free ",

LCD\_LINE\_1)

mqttc.publish("slot1","0")

time.sleep(0.2)

if (slot2\_status == False):

lcd\_string("Slot2 Parked ",

LCD\_LINE\_2)

mqttc.publish("slot2", "1")

time.sleep(0.2)

else:

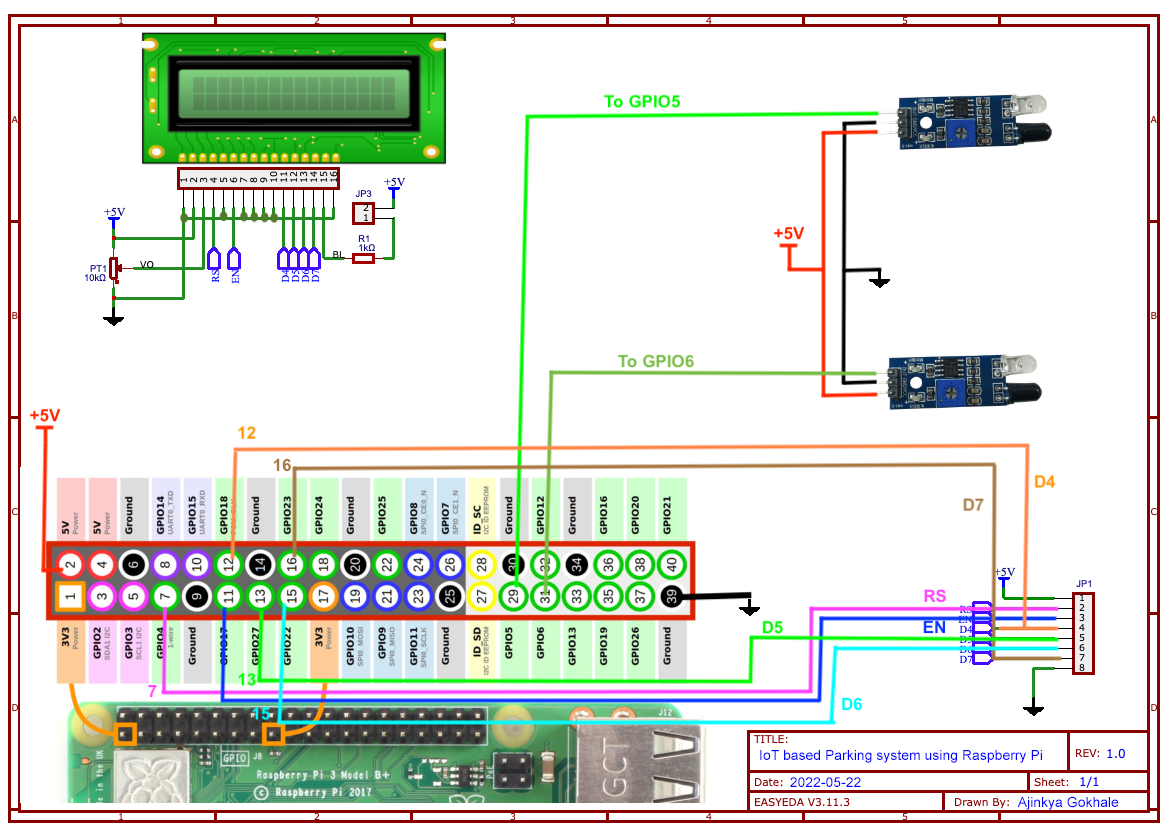
lcd\_string("Slot2 Free ",

LCD\_LINE\_2)

mqttc.publish("slot2","0")

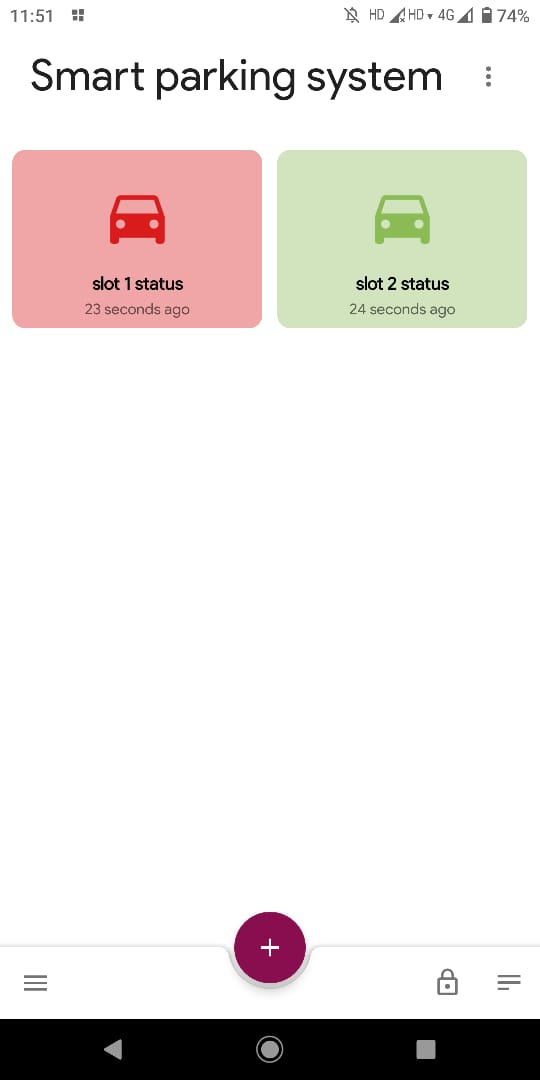
time.sleep(0.2)

**CONNECTION:**



**OUTPUT:**

Output 1:



Output 2:

