**DOKUMEN DESAIN PERANGKAT LUNAK**

**GATE PARKIR**

SOURCE CODE SISTEM GATE PARKIR

*LSKK-SW-SC-01-GP*



**PT. LANGGENG SEJAHTERA KREASI KOMPUTASI**

**©2019**

Dokumen Desain Perangkat Lunak, GATE PARKIR

Source Code Sistem Gate Parkir

PT. Langgeng Sejahtera Kreasi Komputasi ©2019

# Deskripsi Aplikasi

Sistem ini dirancang untuk mendukung bisnis proses parkir di sekolah atau tempat-tempat parkir lainnya. Sistemnya dirancang dan diimplementasi dengan konsep IOT sehingga proses implementasinya dapat dilakukan dengan sangat cepat.

# Sistem Server

|  |
| --- |
| import time  import datetime  import json  import os  import uuid  import mysql.connector  import paho.mqtt.client as mqtt  # define client  db = mysql.connector.connect(  host='192.168.16.99',  user='parkir',  passwd='parkir',  database="parkir\_smk"  )  cursor = db.cursor()  topicPublish = "/parking.response"  def entry\_gate(id\_card):  print('entry\_gate function')  print(f'ini id card {id\_card}')  count = 0  words = id\_card.replace('#', '')  print(f'ini words {words}')  sql = f"SELECT \* FROM card WHERE id\_card = '{words}'"  cursor.execute(sql)  result = cursor.fetchall()  for res in result:  count += 1  if count > 0:  if res[1] == 'active':  print('Open Gate')  check\_parkir(res[0])  # kirim mqtt  mqttc.publish('gate/open\_entry', '1')  else:  print(f'card dengan id {words} tidak valid')  else:  print(f'card dengan id {words} tidak ditemukan')  def check\_parkir(id\_card):  print('check\_parkir function')  count = 0  sql = f"SELECT \* FROM parkir WHERE id\_card = '{id\_card}'"  cursor.execute(sql)  result = cursor.fetchall()  for results in result:  count += 1  if count > 0:  print('update data parkir')  update\_parkir(id\_card)  else:  print('insert data parkir')  insert\_parkir(id\_card)  def insert\_parkir(id\_card):  print('insert\_parkir function')  id\_parkir = f"P-{uuid.uuid4().hex}"  timestamp = datetime.datetime.now()  sql = "INSERT INTO parkir (id\_parkir, id\_card, timestamp\_masuk, timestamp\_keluar) VALUES (%s, %s, %s, %s)"  values = (f"{id\_parkir}", f"{id\_card}", f"{timestamp}", "")  cursor.execute(sql, values)  db.commit()  print(cursor.rowcount, "record inserted")  def update\_parkir(id\_card):  print('update\_parkir function')  timestamp = datetime.datetime.now()  sql = f"UPDATE parkir SET timestamp\_masuk = '{timestamp}' WHERE id\_card = '{id\_card}'"  cursor.execute(sql)  db.commit()  print(cursor.rowcount, "record(s) affected")  def exit\_gate(data):  print('exit\_gate function')  count = 0  datas = data.split('#')  id\_card = datas[0].lower()  timestamp\_out = datetime.datetime.now()  sql = f"UPDATE parkir SET timestamp\_keluar = '{timestamp\_out}' WHERE id\_card = '{id\_card}'"  cursor.execute(sql)  db.commit()  print(cursor.rowcount, "record(s) affected")  sql = f"SELECT \* FROM parkir WHERE id\_card = '{id\_card}'"  cursor.execute(sql)  result = cursor.fetchall()  for results in result:  count += 1  if count > 0:  temp\_timestamp = results[2]  words = temp\_timestamp.split(' ')  timestamp\_masuk = words[1].split('.')  id\_parkir = results[0]  mqttc.publish(topicPublish, f'Keluar#{timestamp\_masuk[0]}#{id\_parkir}')  # kirim mqtt ke server  else:  print('Gagal keluar parkir')  mqttc.publish(topicPublish, 'Keluar#failed')  def login(data):  print('login function')  counter = 0  splits = data.split('#')  str\_json = json.loads(splits[1])  username = str\_json['Username']  password = str\_json['Password']  sql = f"SELECT \* FROM user WHERE username = '{username}' AND password = '{password}'"  cursor.execute(sql)  result = cursor.fetchall()  for results in result:  nama = result[4]  counter += 1  if counter > 0:  print('Login Success')  mqttc.publish(topicPublish, f'Login#success#{nama}')  else:  print('Login Failed')  mqttc.publish(topicPublish, f'Login#failed')  def log\_transaksi(data):  datas = data.split('#')  id\_transaksi = f'T-{uuid.uuid4().hex}'  waktu\_transaksi = datetime.datetime.now()  username = datas[0]  nopol = datas[1]  id\_parkir = datas[2]  duration = datas[3]  total = datas[4]  sql = "INSERT INTO detail\_transaksi (id\_transaksi, waktu\_transaksi, username, nopol\_kendaraan, id\_parkir, lama\_parkir, total\_bayar) VALUES (%s, %s,%s, %s,%s, %s, %s)"  val = (f'{id\_transaksi}', f'{waktu\_transaksi}', f'{username}',  f'{nopol}', f'{id\_parkir}', f'{duration}', f'{total}')  cursor.execute(sql, val)  db.commit()  print(cursor.rowcount, "record inserted.")  mqttc.publish('gate/close\_gate', '1')  # Define event callbacks  def on\_connect(client, userdata, flags, rc):  print("Connected to server " + str(rc))  print("----------------------------------------")  print("------lpay services on progress-------")  print("--------press CTRL+C to close---------")  print("----------------------------------------")  print("########################################")  mqttc.subscribe("gate/rfid", 0)  mqttc.subscribe("/parking.request", 0)  mqttc.subscribe("gate/payment", 0)  def on\_message(client, obj, msg):  print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))  if msg.topic == 'gate/rfid':  entry\_gate(str(msg.payload, 'utf-8'))  elif msg.topic == '/parking.request':  msgs = str(msg.payload, 'utf-8')  section = msgs.split("#")  if section[0] == "Login":  login(str(msg.payload, 'utf-8'))  elif section[0] == "Keluar":  exit\_gate(msgs)  elif msg.topic == "gate/payment":  print("masuk payment")  msgs = str(msg.payload, 'utf-8')  log\_transaksi(msgs)  def on\_publish(client, obj, mid):  print("mid: " + str(mid))  def on\_subscribe(client, obj, mid, granted\_qos):  print("Subscribed: " + str(mid) + " " + str(granted\_qos))  def on\_log(client, obj, level, string):  print(string)  mqttc = mqtt.Client()  # Assign event callbacks  mqttc.on\_message = on\_message  mqttc.on\_connect = on\_connect  mqttc.on\_publish = on\_publish  mqttc.on\_subscribe = on\_subscribe  # Uncomment to enable debug messages  #mqttc.on\_log = on\_log  # Connect  mqttc.connect('192.168.16.99', 1883)  # Start subscribe, with QoS level 0  # mqttc.subscribe(topic, 0)  # Publish a message  # mqttc.publish(topic, "my message")  # Continue the network loop, exit when an error occurs  rc = 0  ui\_counter = 0  # main loop  try:  while True:  # -------------------------------------------------------------------------  # Coding Disini  # -------------------------------------------------------------------------  rc = mqttc.loop()  ui\_counter += 1  if ui\_counter > 30:  # clear console  os.system('cls||clear')  # print  print("----------------------------------------")  print("------lpay services on progress-------")  print("--------press CTRL+C to close---------")  print("----------------------------------------")  print("########################################")  ui\_counter = 0  # recoonnect  if rc != 0:  mqttc.connect('192.168.16.141', 1883)  # ---------------------------------------------------------------------------------  # Keyboard Interrupt (ok)  # ---------------------------------------------------------------------------------  except KeyboardInterrupt:  pass  print("rc: " + str(rc))  # ---------------------------------------------------------------------------------  # Keyboard Interrupt Executed, Exit in 1s (ok)  # ---------------------------------------------------------------------------------  # print  print("------------------------------------^C")  print("--------------close in 1s---------------")  print("^C------------------------------------^C")  print("########################################")  print("")  # timer sleep 1s  time.sleep(1)  # --------------------------------------------------------------------------------- |

# Sistem Gate In

|  |
| --- |
| import RPi.GPIO as GPIO  import paho.mqtt.client as mqtt  import time  import os  # -------------------------------------------------------------------------  # initialization pin raspberry  # -------------------------------------------------------------------------  pinOutUp = 23  pinOutDown = 24  pinOutStop = 18  pinInUpLimit = 17  pinInDownLimit = 27  pinInLoop = 22  varStatusLoop = 0  varStatusUpLimit = 0  varStatusDownLimit = 0  # Define event callbacks  def on\_connect(client, userdata, flags, rc):  print("rc: " + str(rc))  def on\_message(client, obj, msg):  print("Data Dari MQTT : " + str(msg.payload))  # logic for open the gate  if str(msg.payload) == "1" and varStatusLoop == 1:  GPIO.output(pinOutUp, GPIO.LOW)  print("OPEN GATE MQTT 1")  time.sleep(0.5)  GPIO.output(pinOutUp, GPIO.HIGH)  if str(msg.payload) == "O":  GPIO.output(pinOutUp, GPIO.LOW)  print("OPEN GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutUp, GPIO.HIGH)  elif str(msg.payload) == "C":  GPIO.output(pinOutDown, GPIO.LOW)  print("CLOSE GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutDown, GPIO.HIGH)  elif str(msg.payload) == "S":  GPIO.output(pinOutStop, GPIO.LOW)  print("STOP GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutStop, GPIO.HIGH)  def on\_publish(client, obj, mid):  print("mid: " + str(mid))  def on\_subscribe(client, obj, mid, granted\_qos):  print("Subscribed: " + str(mid) + " " + str(granted\_qos))  def on\_log(client, obj, level, string):  print(string)  # -------------------------------------------------------------------------  # disable warning on console  # -------------------------------------------------------------------------  GPIO.setwarnings(False)  # -------------------------------------------------------------------------  # set mode to bcm  # -------------------------------------------------------------------------  GPIO.setmode(GPIO.BCM)  # -------------------------------------------------------------------------  # setup pin  # -------------------------------------------------------------------------  GPIO.setup([pinOutUp, pinOutDown, pinOutStop], GPIO.OUT, initial=GPIO.HIGH)  GPIO.setup([pinInUpLimit, pinInDownLimit, pinInLoop],  GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)  client = mqtt.Client()  # Assign event callbacks  client.on\_message = on\_message  client.on\_connect = on\_connect  client.on\_publish = on\_publish  client.on\_subscribe = on\_subscribe  client.connect("192.168.16.99", 1883)  client.subscribe("gate/open\_entry", 0)  rc = 0  os.system('cls||clear')  while rc == 0:  rc = client.loop()  # Read Digital Input  varStatusLoop = GPIO.input(pinInLoop)  varStatusUpLimit = GPIO.input(pinInUpLimit)  varStatusDownLimit = GPIO.input(pinInDownLimit)  print("Loop: "+str(varStatusLoop)+" Up Limit: " +  str(varStatusUpLimit)+" Down Limit: "+str(varStatusDownLimit))  # logic for close the gate  if varStatusLoop == 0 and varStatusUpLimit == 1 and varStatusDownLimit == 0:  time.sleep(3)  GPIO.output(pinOutDown, GPIO.LOW)  print("CLOSE GATE DARI WHILE")  time.sleep(0.5)  GPIO.output(pinOutDown, GPIO.HIGH)  print("rc: " + str(rc)) |

# Sistem Gate Out

|  |
| --- |
| import RPi.GPIO as GPIO  import paho.mqtt.client as mqtt  import time  import os  # -------------------------------------------------------------------------  # initialization pin raspberry  # -------------------------------------------------------------------------  pinOutUp = 23  pinOutDown = 24  pinOutStop = 18  pinInUpLimit = 17  pinInDownLimit = 27  pinInLoop = 22  varStatusLoop = 0  varStatusUpLimit = 0  varStatusDownLimit = 0  # Define event callbacks  def on\_connect(client, userdata, flags, rc):  print("rc: " + str(rc))  def on\_message(client, obj, msg):  print("Data Dari MQTT : " + str(msg.payload))  # logic for open the gate  if str(msg.payload) == "1" and varStatusLoop == 1:  GPIO.output(pinOutUp, GPIO.LOW)  print("OPEN GATE MQTT 1")  time.sleep(0.5)  GPIO.output(pinOutUp, GPIO.HIGH)  if str(msg.payload) == "O":  GPIO.output(pinOutUp, GPIO.LOW)  print("OPEN GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutUp, GPIO.HIGH)  elif str(msg.payload) == "C":  GPIO.output(pinOutDown, GPIO.LOW)  print("CLOSE GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutDown, GPIO.HIGH)  elif str(msg.payload) == "S":  GPIO.output(pinOutStop, GPIO.LOW)  print("STOP GATE MQTT")  time.sleep(0.5)  GPIO.output(pinOutStop, GPIO.HIGH)  def on\_publish(client, obj, mid):  print("mid: " + str(mid))  def on\_subscribe(client, obj, mid, granted\_qos):  print("Subscribed: " + str(mid) + " " + str(granted\_qos))  def on\_log(client, obj, level, string):  print(string)  # -------------------------------------------------------------------------  # disable warning on console  # -------------------------------------------------------------------------  GPIO.setwarnings(False)  # -------------------------------------------------------------------------  # set mode to bcm  # -------------------------------------------------------------------------  GPIO.setmode(GPIO.BCM)  # -------------------------------------------------------------------------  # setup pin  # -------------------------------------------------------------------------  GPIO.setup([pinOutUp, pinOutDown, pinOutStop], GPIO.OUT, initial=GPIO.HIGH)  GPIO.setup([pinInUpLimit, pinInDownLimit, pinInLoop],  GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)  client = mqtt.Client()  # Assign event callbacks  client.on\_message = on\_message  client.on\_connect = on\_connect  client.on\_publish = on\_publish  client.on\_subscribe = on\_subscribe  client.connect("192.168.16.99", 1883)  client.subscribe("gate/close\_gate", 0)  rc = 0  os.system('cls||clear')  while rc == 0:  rc = client.loop()  # Read Digital Input  varStatusLoop = GPIO.input(pinInLoop)  varStatusUpLimit = GPIO.input(pinInUpLimit)  varStatusDownLimit = GPIO.input(pinInDownLimit)  print("Loop: "+str(varStatusLoop)+" Up Limit: " +  str(varStatusUpLimit)+" Down Limit: "+str(varStatusDownLimit))  # logic for close the gate  if varStatusLoop == 0 and varStatusUpLimit == 1 and varStatusDownLimit == 0:  time.sleep(3)  GPIO.output(pinOutDown, GPIO.LOW)  print("CLOSE GATE DARI WHILE")  time.sleep(0.5)  GPIO.output(pinOutDown, GPIO.HIGH)  print("rc: " + str(rc)) |

# *Release Notes*

Berikut *release notes* aplikasi untuk sistem Gate Parkir.

### *GP-SC, b1912, v1.0*

* Versi rilis pertama.

# *Approval*

Dibuat oleh Tim *S/W* *Engineer* pada 17 Desember 2019.

Dicek dan disetujui pada 17 Desember 2019, oleh:

Mochamad Vicky Ghani Aziz