SMART PARKING

PHASE 3 : DEVELOPMENT PART 1

ABSTRACT :

- One of the most common problems today is a saturation of parking spaces.
- Vehicles continue to outnumber existing parking spaces, thus clogging roads.
- Incidences of violence over occupancy, deformed cars due to a space crunch, and overcharging for parking are some problems that result.
- An IR proximity sensor works by applying a voltage to a pair of IR light emitting diodes (LED's) which in turn, emit infrared light.
- This light propagates through the air and once it hits an object it is reflected back towards the sensor.
- If the object is close, the reflected light will be stronger than if the object is further away.
- The sensing unit, in the form of an integrated circuit (IC), detects the reflected infrared light, and if its intensity is strong enough, the circuit becomes active.

PYTHON SCRIPT:

```
Import random

Import time

data (0 for empty, 1 for occupied)

Parking_spots = [0, 0, 0, 0, 0]

Def get_parking_status():

Return [random.choice([0, 1]) for _ in range(len(parking_spots))]

While True:

Parking_spots = get_parking_status()

# Send parking_spots data to the cloud (simulated)

Print("Sending data to the cloud:", parking_spots)

Time.sleep(10)  # Simulated data update every 10 seconds
```

RASPBERRY PI INTEGRATION:

```
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
E_PULSE = 0.0005
E_DELAY = 0.0005
Delay = 1
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)
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()                    declaration
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)
Def lcd_init()
  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)
```

```
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, )
  Lcd_toggle_enable()
  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:
```

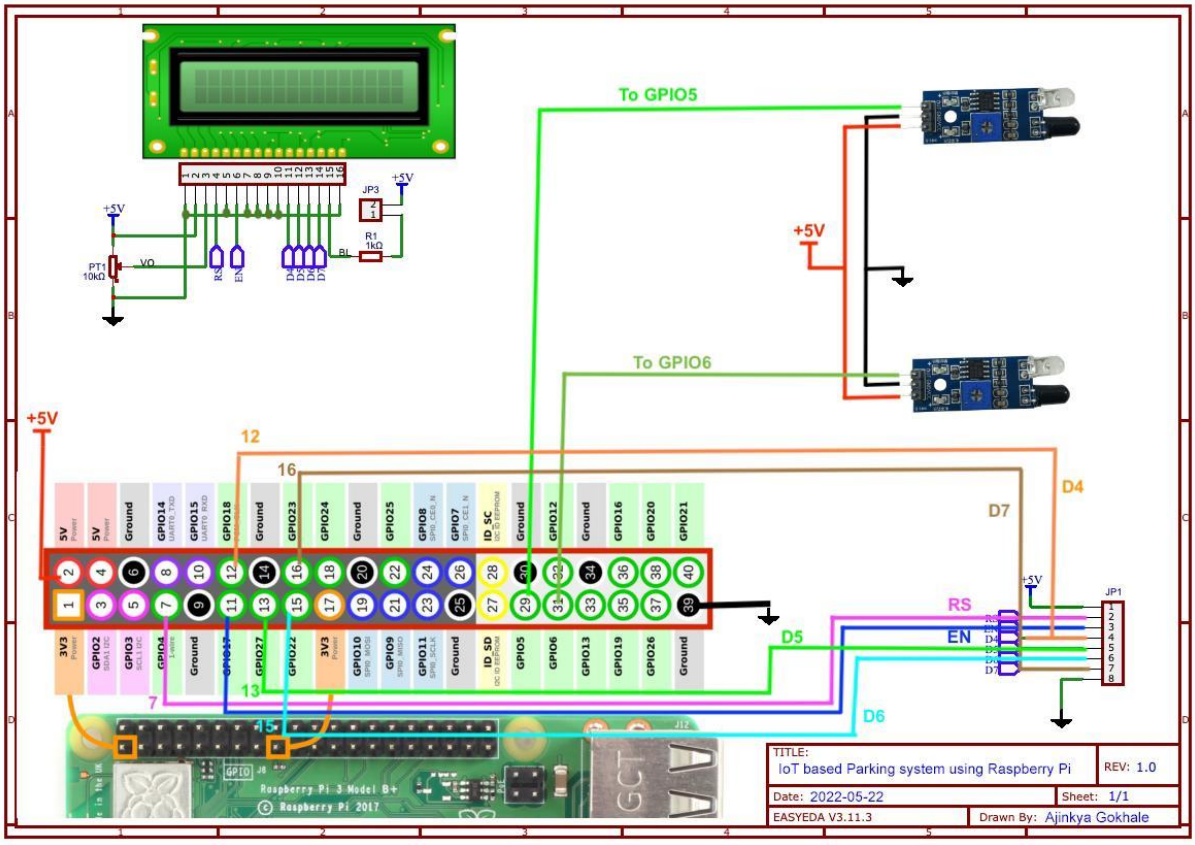
```
    GPIO.output(LCD_D7, True)
  Lcd_toggle_enable()
Def lcd_toggle_enable():
  Time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  Time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  Time.sleep(E_DELAY)
Def lcd_string(message,line):
  Message = message.ljust(LCD_WIDTH," ")
   Lcd_byte(line, LCD_CMD)
   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
Delay = 5
While 1:
  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)
```

IOT DEVICE:



THESE CODE AND THEORY ARE INCLUDED IN PHASE 3 : SMART PARKING

BY TEAM MATES :

AKASH.G

AKASH.S

ARUN KUMAR. A

ASHIKA ANGEL.J

DEEPALAKSHMI.E