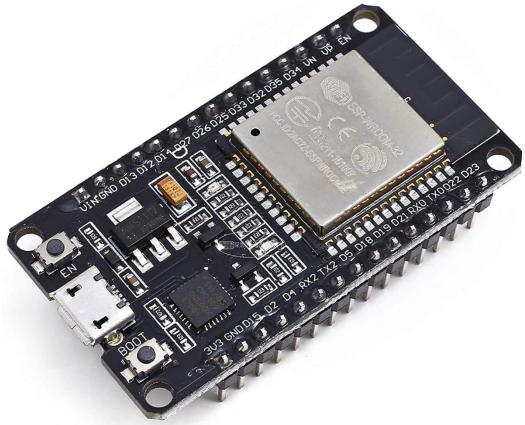
SMART PARKING SYSTEM USING IOT

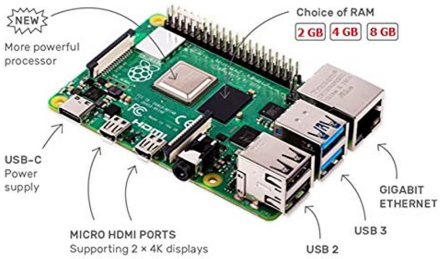
IOT DEVICE:

ESP32 DEVELOPMENT BOARD



The ESP32 is a popular microcontroller and that includes Wi-Fi and Bluetooth connectivity, making it suitable for a wide range of applications.

RASPBERRY PI:



The Raspberry Pi is a series of small, affordable, single-board computers developed by the Raspberry Pi Foundation. They are widely used for various projects and educational purposes, thanks to their versatility and low cost. Raspberry Pi boards come in different models with varying features and capabilities, allowing users to build everything from basic hobbyist projects to more complex applications like media centers and IoT devices.

ULTRASONIC SENSOR:

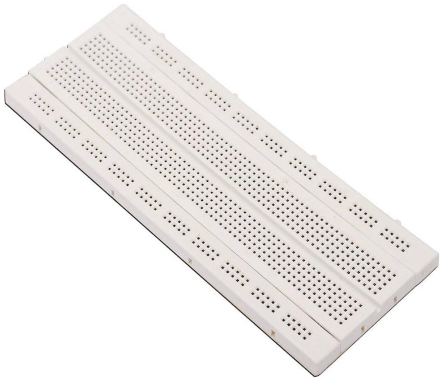
An ultrasonic sensor is a device that uses high-frequency sound waves to detect the distance to or presence of objects in its vicinity. It works on the principle of sending out an ultrasonic pulse and measuring the time it takes for the pulse to bounce back after hitting an

object. The sensor typically consists of a transmitter that emits the ultrasonic pulse and a receiver that listens for the returning echo.

JUMPER WIRES:

Jumper wires are simple, flexible wires with connectors at each end that are used in electronics and electrical projects to create connections between different components or points on a breadboard, circuit board, or microcontroller. They are an essential tool for prototyping, experimenting, and connecting various electronic components like sensors, LEDs, and microcontrollers.

BREADBOARD:



A breadboard is a fundamental tool used in electronics for prototyping and building temporary circuits. It's a rectangular plastic board with a grid of holes that allows you to insert and connect electronic components and wires without soldering. Breadboards are commonly used by students, hobbyists, and engineers for testing and designing circuits before finalizing them on a printed circuit board (PCB).

PROGRAMMING:

```
#!/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

commands
```

```
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)
```