SMART PARKING

Submitted By

M.PRADEEP

Department of Electronics and Communication Engineering

Anna University Regional Campus Coimbatore

LIST OF SENSORS

- **Active state conditions :**

1. Ultrasonic sensor
2. Occupancy sessor
3. Video Camera

- **Passive state conditions :**

4. RFID sensor
5. Entry and exit gate sensor

➢ *ACTIVE STATE CONDITIONS :*

1.ULTRASONIC SESNOR :

These sensors use sound waves to detect the presence of vehicles in parking spaces. They are often mounted on the ceiling or walls of parking structures and can provide information about whether a parking spot is occupied or vacant.

PYTHON SCRIPT FOR ULTRASONIC SENSOR :

```
import RPi.GPIO as GPIO
import time
# Set the GPIO pins for the ultrasonic sensor
trigger_pin = 23
echo_pin = 24
# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(trigger_pin, GPIO.OUT)
GPIO.setup(echo_pin, GPIO.IN)
try:
    while True:
    # Trigger the ultrasonic sensor
   GPIO.output(trigger_pin,GPIO.HIGH)
   time.sleep(0.00001)
   GPIO.output(trigger_pin, GPIO.LOW)
  pulse_start = time.time()
   pulse_end = time.time()
  # Wait for the echo response
  while GPIO.input(echo_pin) == 0:
  pulse_start
```

2.OCCUPANCY SENSOR :

These sensors are often used in conjunction with lighting systems in parking structures. They can detect the presence of vehicles and adjust lighting accordingly to save energy.

PYTHON SCRIPT FOR OCCUPANCY SENSOR:

```
import RPi.GPIO as GPIO
 # Import the GPIO library (for
   Raspberry Pi)
# Define the GPIO pin where the PIR
   sensor is connected
PIR_SENSOR_PIN = 17
# Initialize the GPIO settings
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_SENSOR_PIN,
   GPIO.IN)
try:
    print("Occupancy sensor is active.
   Waiting for motion...")
 while True:
 If GPIO.input(PIR_SENSOR_PIN):
print("Motion detected Occupancy
detected.")
else:
print("No motion detected. Space is
vacant.")
except KeyboardInterrupt:
 print("Occupancy sensor
script   terminated.")
 GPIO.cleanup()
```

3.VIDEO CAMERAS :

Video cameras, including both standard and specialized license plate recognition cameras, can be used to monitor parking spaces and identify available spots.

PYTHON SCRIPT FOR VIDEO CAMERAS :

```
import cv2
import time
# Initialize the video capture (0 is the default camera)
cap = cv2.VideoCapture(0)
# Define the region of interest (ROI) for parking space detection
# These coordinates represent the top-left and bottom-right corners of the ROI
roi_x1, roi_y1, roi_x2, roi_y2 = 100, 100, 300, 300
# Function to monitor parking spaces
def monitor_parking_spaces():
    while True:
        ret, frame = cap.read()
 if not ret:
            break
 # Crop the frame to the defined ROI
  roi = frame[roi_y1:roi_y2, roi_x1:roi_x2]
 # Perform image processing to detect occupancy (e.g., using computer vision techniques)
 # For demonstration purposes, we'll simulate occupancy detection based on color
 # You would typically use more advanced techniques like object detection
         blue_color_threshold  = 100
        if roi[:, :, 0].mean() > blue_color_threshold:
        print("Parking space occupied.")
        else:
        print("Parking space vacant.")
    cv2.rectangle(frame,  (roi_x1, roi_y1), (roi_x2, roi_y2), (0, 0, 255), 2)  # Draw ROI rectangle
     cv2.imshow('Smart Parking', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
         break
    cap.release()
    cv2.destroyAllWindows()
if _name_  == "_main_":
    try:
        print("Smart Parking system is active. Press 'q' to exit.")
        monitor_parking_spaces()
    except KeyboardInterrupt:
        print("Smart Parking script terminated.")
```

➢ PASSIVE STATE CONDITIONS

4.RFID Sensors:

Radio-Frequency Identification (RFID) sensors can be used for tracking vehicles as they enter and exit parking facilities. This data is helpful for occupancy and pricing decisions.

PYTHON SCRIPT FOR RFID SENSOR:

```
import RPi.GPIO as GPIO
import MFRC522
import time
# Initialize the RFID sensor
reader = MFRC522.MFRC522()
# Define GPIO pins for barrier control
barrier_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(barrier_pin, GPIO.OUT)
# Define a list of authorized RFID card IDs
authorized_cards = [
    [0x01, 0x23, 0x45, 0x67, 0x89],
    # Add more card IDs as needed]
try:
    while True:
        # Scan for RFID cards
        (status, TagType) =
     reader.MFRC522_Request(reader.PICC_REQIDL)
     if status == reader.MI_OK:
            # A card is detected, now try to read it
            (status, uid) = reader.MFRC522_Anticoll()
    if status == reader.MI_OK:
        card_id = uid[:4]  # Extract the first 4 bytes of the   card's
UID
                if card_id in authorized_cards:
                    print("Access granted!")
                    # Open the barrier
                    GPIO.output(barrier_pin, GPIO.HIGH)
                    time.sleep(5)  # Keep the barrier open for   5 seconds
                    GPIO.output(barrier_pin, GPIO.LOW)  #   Close the
barrier
                else:
                    print("Access denied!")
    except KeyboardInterrupt:
    GPIO.cleanup()
```

5.ENTRY AND EXIT GATE SENSOR :

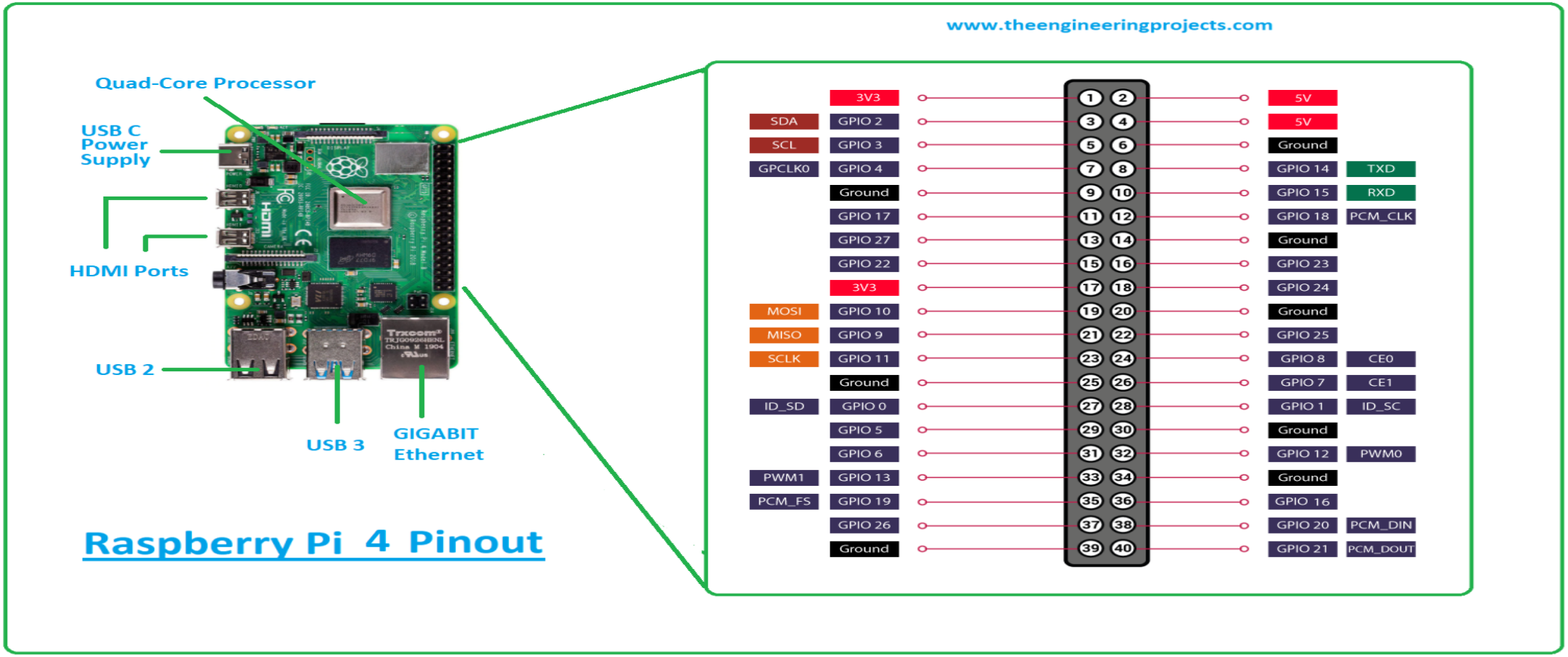
Sensors at entry and exit points of parking facilities can track vehicles coming and going. This data is valuable for understanding the flow of vehicles and determining pricing adjustments.

PYTHON SCRIPT FOR ENTRY AND EXIT GATE SENSOR :

```
import RPi.GPIO as GPIO
import time
# Set GPIO pins for entry and exit sensors
entry_sensor_pin = 23
exit_sensor_pin = 24
entry_gate_pin = 17
exit_gate_pin = 18
# Initialize GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(entry_sensor_pin, GPIO.IN)
GPIO.setup(exit_sensor_pin, GPIO.IN)
GPIO.setup(entry_gate_pin, GPIO.OUT)
GPIO.setup(exit_gate_pin, GPIO.OUT)
# Initial gate states (closed)
entry_gate_open = False
exit_gate_open = False
try:
    while True:
        # Check entry sensor
        if GPIO.input(entry_sensor_pin) == GPIO.HIGH:
            if not entry_gate_open:
                print("Entry gate opened")
                GPIO.output(entry_gate_pin, GPIO.HIGH)
                entry_gate_open = True
        else:
            if entry_gate_open:
                print("Entry gate closed")
                GPIO.output(entry_gate_pin, GPIO.LOW)
                entry_gate_open = False
        # Check exit sensor
        if GPIO.input(exit_sensor_pin) == GPIO.HIGH:
            if not exit_gate_open:
                print("Exit gate opened")
                GPIO.output(exit_gate_pin, GPIO.HIGH)
                exit_gate_open = True
        else:
            if exit_gate_open:
                print("Exit gate closed")
                GPIO.output(exit_gate_pin, GPIO.LOW)
                exit_gate_open = False
        time.sleep(0.1)  # Check sensor states every 0.1 seconds
except KeyboardInterrupt:
    GPIO.cleanup()
```

CONTROLLER

RASPBERRY PI 4 :



THANK YOU