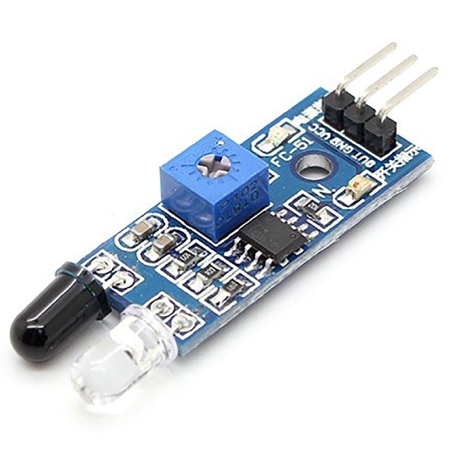
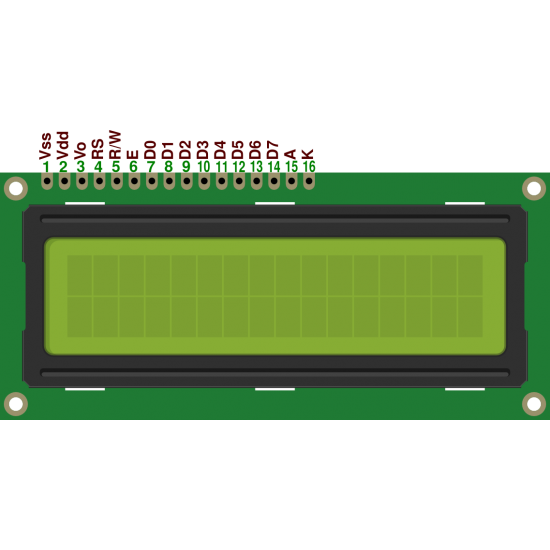
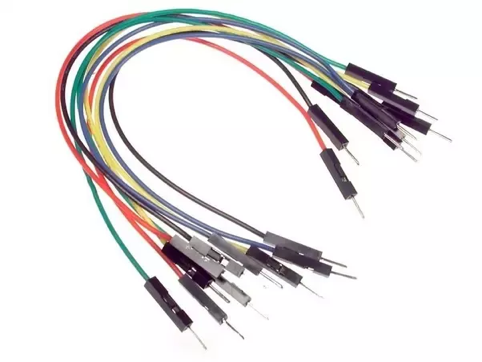
**Smart Car Parking System**

A smart car parking system using Raspberry Pi is a project that combines Raspberry Pi's capabilities with various sensors and components to create an automated and efficient parking system. The system can provide real-time information about parking space availability, automate the entry and exit processes, and improve overall parking management.

**Equipment Required:**

* Raspberry Pi 3 B+
* Two IR Sensors
* One 16\*2 LCD Screen
* Two Servo Motors
* One Potentiometer
* Connecting Wires

**Specifications:**

Raspberry Pi 3 B+

The Raspberry Pi 3 Model B+ is a single-board computer that offers a range of features and specifications. Here are the key specifications of the Raspberry Pi 3 Model B+:

* Processor: Broadcom BCM2837B0, Cortex-A53 64-bit quad-core SoC clocked at 1.4 GHz.
* Memory: 1GB LPDDR2 SDRAM.
* Wireless Connectivity: Built-in 2.4 GHz and 5 GHz 802.11b/g/n/ac Wi-Fi, supporting wireless LAN and Bluetooth 4.2/BLE.
* Ethernet: 10/100/1000 Mbps Ethernet port.
* USB Ports: 4 USB 2.0 ports.
* Video Output: Full-size HDMI port supporting resolutions up to 1080p.
* Audio Output: 3.5mm stereo audio/composite video output.
* GPIO Pins: 40 GPIO pins, compatible with previous Raspberry Pi models.
* Camera Interface: CSI camera port for connecting a Raspberry Pi camera module.
* Display Interface: DSI display port for connecting a Raspberry Pi touchscreen display.
* Storage: MicroSD card slot for operating system and data storage.
* Power: Micro USB power input for powering the Raspberry Pi.
* Operating System: The Raspberry Pi 3 Model B+ supports a range of operating systems, including Raspbian (the official Raspberry Pi operating system), Ubuntu, Windows 10 IoT Core, and various Linux distributions.
* Dimensions: The board dimensions are approximately 85mm x 56mm x 17mm.

IR Sensor:

* Detection Method: Infrared (IR) emission and reflection.
* Operating Voltage: Typically 3.3V or 5V (depends on the specific model).
* Output: Digital output (HIGH or LOW) indicating the presence or absence of an object within its detection range.
* Detection Range: Varies based on the specific model but typically a few centimeters to a few meters.
* Pin Configuration: Generally has three pins - VCC (power supply), GND (ground), and OUT (digital output).

Servo Motor:

* Type: Usually a standard DC servo motor.
* Operating Voltage: Typically 5V.
* Control Interface: Pulse Width Modulation (PWM) signal is used to control the motor's position.
* Rotation Range: Can rotate from 0 to 180 degrees (standard servo motor).
* Pin Configuration: Generally has three pins - VCC (power supply), GND (ground), and SIGNAL (PWM control signal).

16x2 LCD Display:

* Display Size: 16 characters per line, 2 lines.
* Resolution: 16 columns x 2 rows.
* Communication: Parallel interface (using multiple GPIO pins) or I2C interface (using only a couple of GPIO pins).
* Backlight: Most models include an LED backlight for better visibility.
* Controller: Commonly used controllers include Hitachi HD44780 or compatible controllers.
* Pin Configuration: Varies depending on the specific model and interface used (parallel or I2C).

**Code for the project:**

import RPi.GPIO as IO

import time

import Adafruit\_CharLCD as LCD

IO.setmode(IO.BCM)

lcd\_rs = 12

lcd\_en = 16

lcd\_d4 = 25

lcd\_d5 = 24

lcd\_d6 = 23

lcd\_d7 = 18

lcd\_backlight = 2

lcd\_columns =16

lcd\_rows = 2

lcd = LCD.Adafruit\_CharLCD(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows, lcd\_backlight)

Entry\_pin = 19

Exit\_pin =21

servo\_pin=2

IO.setmode(IO.BCM)

IO.setwarnings(False)

IO.setup(Entry\_pin,IO.IN)

IO.setup(Exit\_pin,IO.IN)

IO.setup(servo\_pin,IO.OUT)

pwm=IO.PWM(2, 50)

pwm.start(0)

def SetAngle(angle):

duty = angle / 18 + 2

IO.output(2, True)

pwm.ChangeDutyCycle(duty)

time.sleep(1)

IO.output(2, False)

pwm.ChangeDutyCycle(0)

available\_space = 5

try:

while True:

if IO.input(Entry\_pin) ==IO.HIGH:

if available\_space > 0:

available\_space -= 1

print("Vehicle has entered, Available Spaces:", available\_space)

lcd.message('vacantspace ' + str(available\_space))

time.sleep(2)

lcd.clear()

SetAngle(0)

time.sleep(0.5)

SetAngle(90)

time.sleep(0.1)

else:

print("No Available Spaces. Entry Denied")

lcd.message('NO SPACE')

time.sleep(2)

lcd.clear()

time.sleep(1)

if IO.input(Exit\_pin) ==IO.HIGH:

if available\_space < 5:

available\_space += 1

print("Vehicle has exited, Available Spaces:", available\_space)

lcd.message('vacantspace ' + str(available\_space))

time.sleep(2)

lcd.clear()

else:

print("No Cars Exited")

time.sleep(2)

except KeyboardInterrupt:

print("Interrupted By USER")

finally:

IO.cleanup()

**Results:**

The implementation of a smart car parking system using Raspberry Pi can yield several beneficial results. Here are some potential outcomes of a well-designed and functioning smart car parking system:

* Improved Parking Efficiency: The system can optimize parking space utilization by providing real-time information about available parking spaces. Drivers can quickly identify vacant spots, reducing the time spent searching for parking and overall congestion in the parking area.
* Enhanced User Experience: Drivers can easily find available parking spaces through the user interface, reducing frustration and stress. The system can provide directions to the nearest vacant spot, simplifying the parking process.
* Reduced Traffic Congestion: With a more efficient parking system, traffic congestion around parking areas can be reduced. Drivers spend less time circling around searching for parking, leading to smoother traffic flow.
* Increased Revenue: Smart car parking systems can generate additional revenue streams through features like reservation systems, premium parking options, or payment integration. Efficient space allocation and improved customer satisfaction can attract more users.
* Better Parking Management: The system provides valuable data on parking occupancy, duration, and patterns. This information can be analysed to optimize parking lot design, pricing strategies, and resource allocation. It enables parking management to make data-driven decisions.
* Environmental Benefits: By minimizing the time spent searching for parking, the system helps reduce carbon emissions and fuel consumption. It promotes sustainable transportation practices and contributes to a greener environment.
* Improved Security: Smart car parking systems can incorporate surveillance cameras or sensors to enhance security. Monitoring the parking area can deter theft, vandalism, or unauthorized access. Additionally, the system can track and record entry and exit events for security purposes.
* Cost Savings: By automating certain processes, such as entry and exit control, the system can reduce the need for manual labour and associated costs. It can also help in optimizing maintenance and resource utilization, resulting in cost savings for parking operators.
* Real-time Monitoring and Alerts: The system allows parking operators to monitor the status of parking spaces, detect issues, and receive alerts in real-time. This enables prompt response to maintenance requirements, malfunctioning sensors, or parking violations.