IOT\_Phase5

Hardware Components:

Raspberry pi model B

Modulo IP Transceiver

Adafruit Standard LCD -16 x 2 white on blue

Software apps and online services

MQTT

Raspberry Pi Raspbian

Hand tools and fabrication machines:

Soldering iron (generic)

solder wire, Lead free

Solder Flux, Soldering

Introduction

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.

Most cities propose increasing parking spaces to combat the problem. Parks and vacant plots are used as potential parking spaces and multi-level facilities are being built, irrespective of the limited land space and resources. But their exists a silly problem. People enter the parking and then came to know that it's full. Should n't it automated ? Don't you think, we should already know if the parking has space for us or not ? Yeah, isn't it a good thought ? This project will help us show the availability of car slots to park the vehicle. This is implemented by using Raspberry Pi 3 B+ and IR Modules.

IR Proximity Module

If you want to make your own IR Module, then here is the schemetic for it.

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. When the sensing unit becomes active, it sends a corresponding signal to the output terminal which can then be used to activate any number of devices. For the purpose of this exercise, a small LED will turn on when the sensor becomes active. Click here to download Gerber File for IR Module

MQTT

MQTT stands for Message Queuing Telemetry Transport. It is a communication standard for IoT devices. This protocol is an open standard, which means anyone can access it and implement it in a networked application.

The system of MQTT is not a software package but it can be used by the developers of new software as a standard to send messages to IoT devices and receive responses back. This opens up the possibility of writing new control instruction systems and also creating IoT device monitoring tools.

The MQTT system was first created in 1999, so it has been around for more than two decades. It was written as a standard to communicate with oil pipeline equipment. As an industrial communication system, MQTT was written within the framework of the Supervisory Control and Data Acquisition (SCADA) framework. The SCADA architecture is designed to communicate with shopfloor equipment in factories and sensors and controllers in process control.

Hardware Implementation

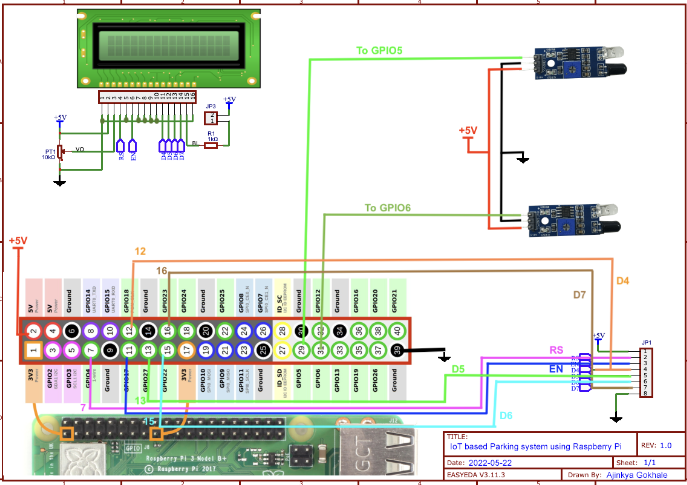
MQTT APP SETUP

Step 1: Download MQTT Dashboard App from Playstore

Step 2: Add broker with the following parameters in your MQTT Dashboard App

Step 3: Add Toggle to the Project and rename it as slot1 and select ON value as '1' and OFF value as '0'. Select Red color for ON and Green Color for OFF with car icon.

Step 4: Everything is done, here are the examples of how it works.



A real-time parking availability system using Python and Raspberry Pi benefits drivers by helping them quickly find parking spots and reduces parking-related issues. Ultrasonic sensors detect available parking spaces, and Python scripts process the data. Real-time updates are stored in a database and presented through a mobile app or web interface. This system saves time, reduces stress, and enhances city planning by improving parking space utilization and reducing traffic congestion.

IOT\_Phase5 Code

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.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

# 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)