# Innovative Smart Parking

Team Leader:

S. Yogul Kishore

211521104187

Team Members:

Veniyan U

211521104176

Sai Niranjan M S

211521104131

K S. Viknesh

211521104179

S V. Sowmiya Narayanan

211521104154

# Table Of Contents

1. **Introduction**
2. **Driver Code**
3. **Parts Diagram**
4. **Output**
5. **Conclusion**

# Introduction

The following documentation provides an overview of a comprehensive hardware setup for a simulated environment, configured using JSON data for the Wokwi simulation platform. This setup includes various electronic components, their connections, and positions, allowing for experimentation, testing, and prototyping within a virtual environment. In this documentation, we will explore the configuration, components, and connections that make up this virtual hardware ecosystem, enabling developers and enthusiasts to simulate and test their electronic projects with ease. The provided JSON data defines the composition of this virtual hardware setup and can serve as a valuable resource for individuals interested in electronics and hardware simulation.

# DRIVER CODE

from smbus2 import SMBus

from RPLCD.i2c import CharLCD

import RPi.GPIO as GPIO

import time

servo\_pin = 3

ir1\_pin = 2

ir2\_pin = 4

smoke\_detector\_pin = 6

buzzer\_pin = 7

slot = 4

flag1 = False

flag2 = False

last\_lcd\_update = 0

lcd\_update\_interval = 1 # Update the LCD every 1 second

GPIO.setmode(GPIO.BCM)

GPIO.setup(ir1\_pin, GPIO.IN)

GPIO.setup(ir2\_pin, GPIO.IN)

GPIO.setup(smoke\_detector\_pin, GPIO.IN)

GPIO.setup(buzzer\_pin, GPIO.OUT)

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

lcd = CharLCD('PCF8574', 0x27)

lcd.clear()

lcd.backlight\_enabled = True

def display\_message(line1, line2):

lcd.clear()

lcd.cursor\_pos = (0, 0)

lcd.write\_string(line1)

lcd.cursor\_pos = (1, 0)

lcd.write\_string(line2)

pwm = GPIO.PWM(servo\_pin, 50)

pwm.start(0)

display\_message(" RASPBERRY", " PARKING SYSTEM")

try:

while True:

if GPIO.input(ir1\_pin) == GPIO.LOW and not flag1:

if slot > 0:

flag1 = True

if not flag2:

pwm.ChangeDutyCycle(7)

slot -= 1

else:

display\_message(" SORRY :( ", " Parking Full ")

if GPIO.input(ir2\_pin) == GPIO.LOW and not flag2:

flag2 = True

if not flag1:

pwm.ChangeDutyCycle(7)

slot += 1

if flag1 and flag2:

time.sleep(1)

pwm.ChangeDutyCycle(2.5)

flag1 = False

flag2 = False

if time.time() - last\_lcd\_update >= lcd\_update\_interval:

if GPIO.input(smoke\_detector\_pin) == GPIO.HIGH:

display\_message(" WARNING! ", " Smoke Detected ")

GPIO.output(buzzer\_pin, GPIO.HIGH)

else:

display\_message(" WELCOME! ", "Slot Left: " + str(slot))

GPIO.output(buzzer\_pin, GPIO.LOW)

last\_lcd\_update = time.time()

except KeyboardInterrupt:

pass

pwm.stop()

GPIO.cleanup()

# PARTS DIAGRAM

{

"version": 1,

"author": "Shraddha Trivedi",

"editor": "wokwi",

"parts": [

{ "type": "wokwi-arduino-uno", "id": "uno", "top": 10, "left": -112, "attrs": {} },

{ "type": "wokwi-servo", "id": "servo1", "top": -154.2, "left": -95.59, "attrs": {} },

{ "type": "wokwi-ir-receiver", "id": "ir1", "top": -99.35, "left": 279.95, "attrs": {} },

{ "type": "wokwi-ir-receiver", "id": "ir2", "top": 129.39, "left": 354.11, "attrs": {} },

{

"type": "wokwi-lcd1602",

"id": "lcd1",

"top": 264.82,

"left": 323.78,

"attrs": { "pins": "i2c" }

},

{

"type": "wokwi-buzzer",

"id": "bz1",

"top": -215.97,

"left": 154.63,

"attrs": { "volume": "0.1" }

},

{

"type": "wokwi-pir-motion-sensor",

"id": "pir1",

"top": 295.88,

"left": -227.65,

"attrs": {}

}

],

"connections": [

[ "servo1:PWM", "uno:3", "green", [ "h211.26", "v36.1" ] ],

[ "ir1:DAT", "uno:2", "green", [ "v19.37", "h-211.08" ] ],

[ "ir2:DAT", "uno:4", "green", [ "v-195.46", "h-239.7" ] ],

[ "ir1:GND", "uno:GND.2", "black", [ "v0" ] ],

[ "ir2:GND", "uno:GND.3", "black", [ "v-116.8", "h-11.16", "v2", "h-0.67", "v-3.33" ] ],

[ "ir1:VCC", "uno:5V", "red", [ "v207.21", "h-278.74" ] ],

[ "ir2:VCC", "uno:5V", "red", [ "v-128.13", "h-13.43", "v124", "h-276" ] ],

[ "lcd1:VCC", "uno:5V", "red", [ "h-4.56", "v-99.59", "h-275.86" ] ],

[ "lcd1:SDA", "uno:A4", "blue", [ "h0" ] ],

[ "lcd1:SCL", "uno:A5", "green", [ "h0" ] ],

[ "servo1:V+", "uno:5V", "red", [ "h420.11", "v244.21", "h-274.13" ] ],

[ "servo1:GND", "uno:GND.3", "black", [ "h410.97", "v81.11" ] ],

[ "lcd1:GND", "uno:GND.2", "black", [ "h-269.94", "v-95.52" ] ],

[ "bz1:1", "uno:GND.3", "cyan", [ "v337.36", "h-114.55" ] ],

[ "pir1:VCC", "uno:5V", "orange", [ "v6.98", "h240.35", "v-193.34" ] ],

[ "pir1:GND", "uno:GND.3", "#8f4814", [ "v-4.85", "h210.37" ] ],

[ "pir1:OUT", "uno:6", "green", [ "v-411.27", "h249.03" ] ],

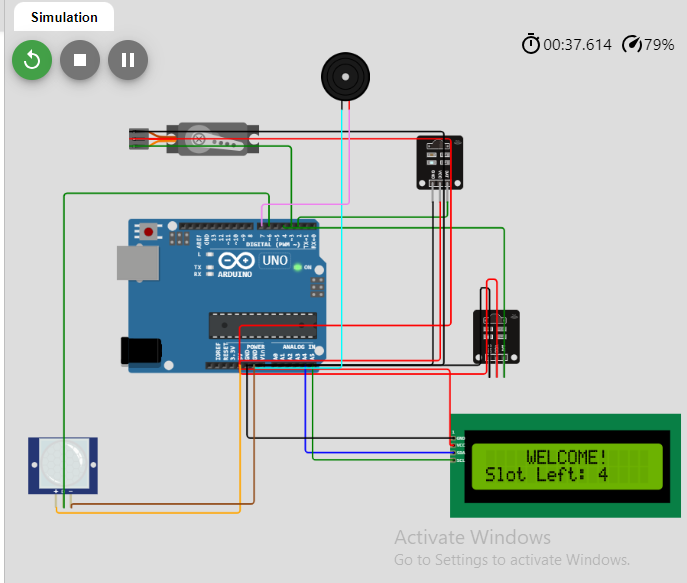
[ "bz1:2", "uno:7", "violet", [ "v123.05", "h-119.07" ] ]

],

"dependencies": {}

}

**OUTPUT**



# CONCLUSION

In this documentation, we have explored the design and implementation of an Arduino-based Smart Parking System using various components, including servos, IR receivers, an LCD display, a buzzer, and a PIR motion sensor. This system provides a comprehensive solution for parking management, occupancy detection, and security.

Throughout the documentation, we covered the setup, code explanation, and interactions with the components. The system demonstrates the ability to monitor parking spaces in real-time and provide information to users through the LCD display.

As technology continues to advance, innovative solutions like this Smart Parking System offer the potential to enhance parking management, reduce congestion, and improve the overall parking experience. By implementing such systems, we can contribute to a more efficient and environmentally friendly urban environment.

Feel free to reach out if you have any questions or need further assistance in using or modifying this system. We hope this documentation has been valuable to you in your exploration of Smart Parking solutions.