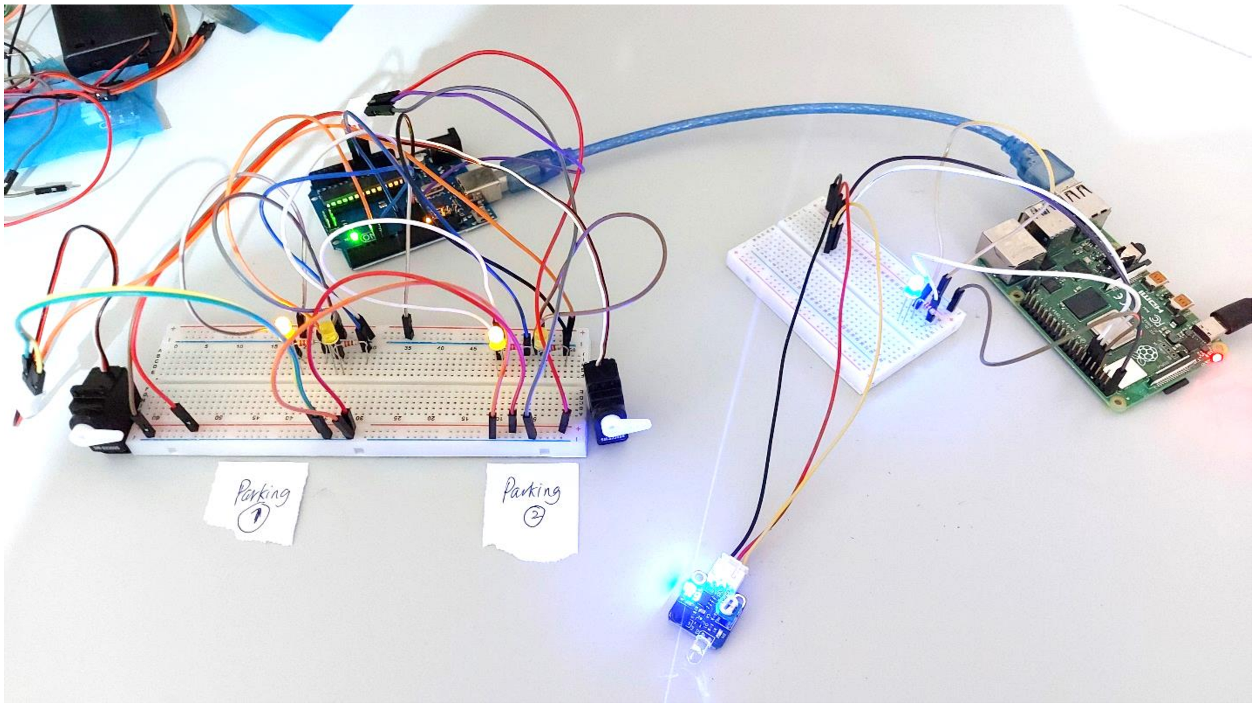
Smart Parking using Raspberry Pi 4 and Arduino

Internet of Things, Web & Mobile



JANUARY 5, 2023

HEMANT RAMPHUL & LAV SINGH RAMESSUR

Table of contents

1. Create a daemon that launches the program when you start the Raspberry Pi 4.

1.1. Create the Raspberry Pi 4 Python program which will start on boot

After creating the Python program **Smart Parking** which is in the **/home/pi/hraspberry** as the home repository. The program filename is "*SmartParking.py*"

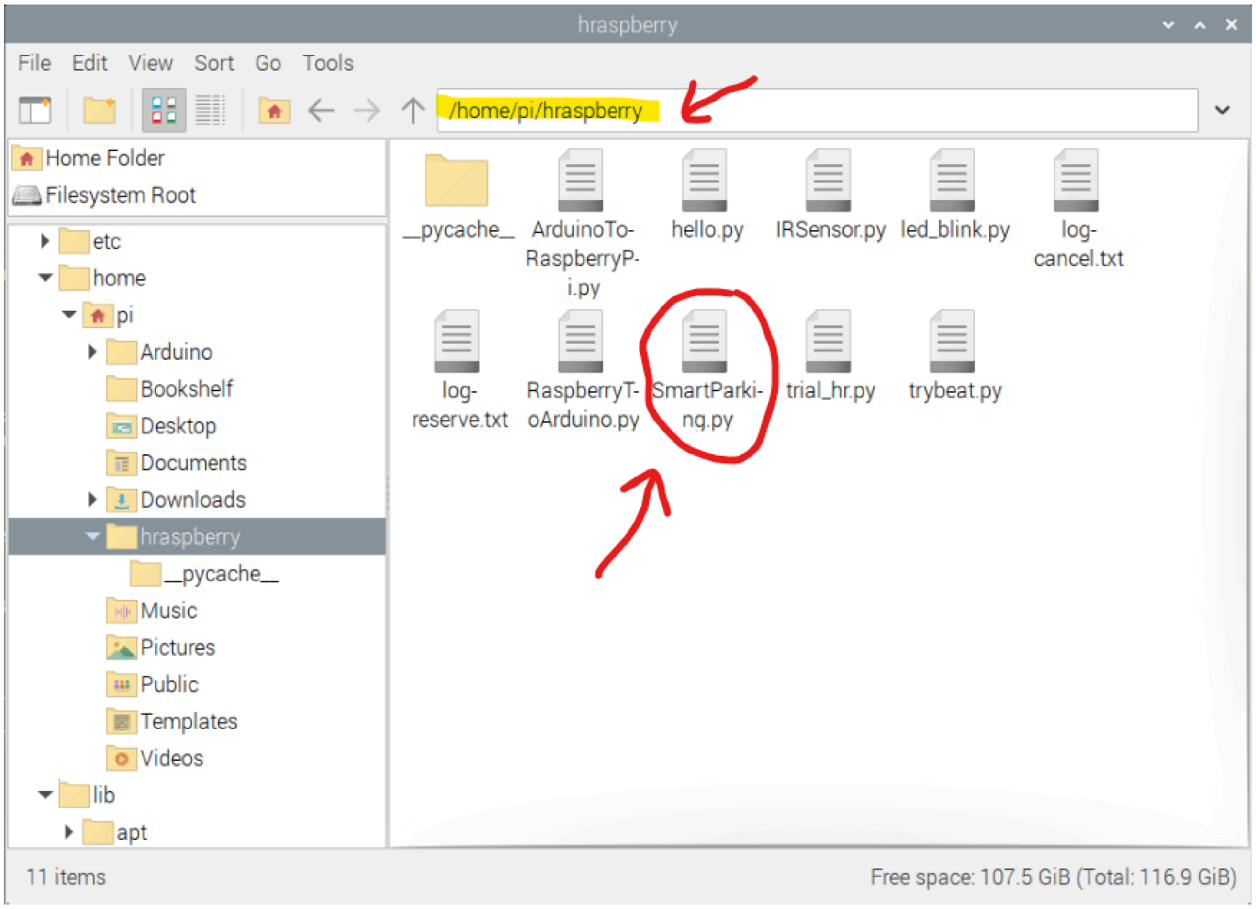


Fig 1. Home repository with python program

1.2. Add a systemd service

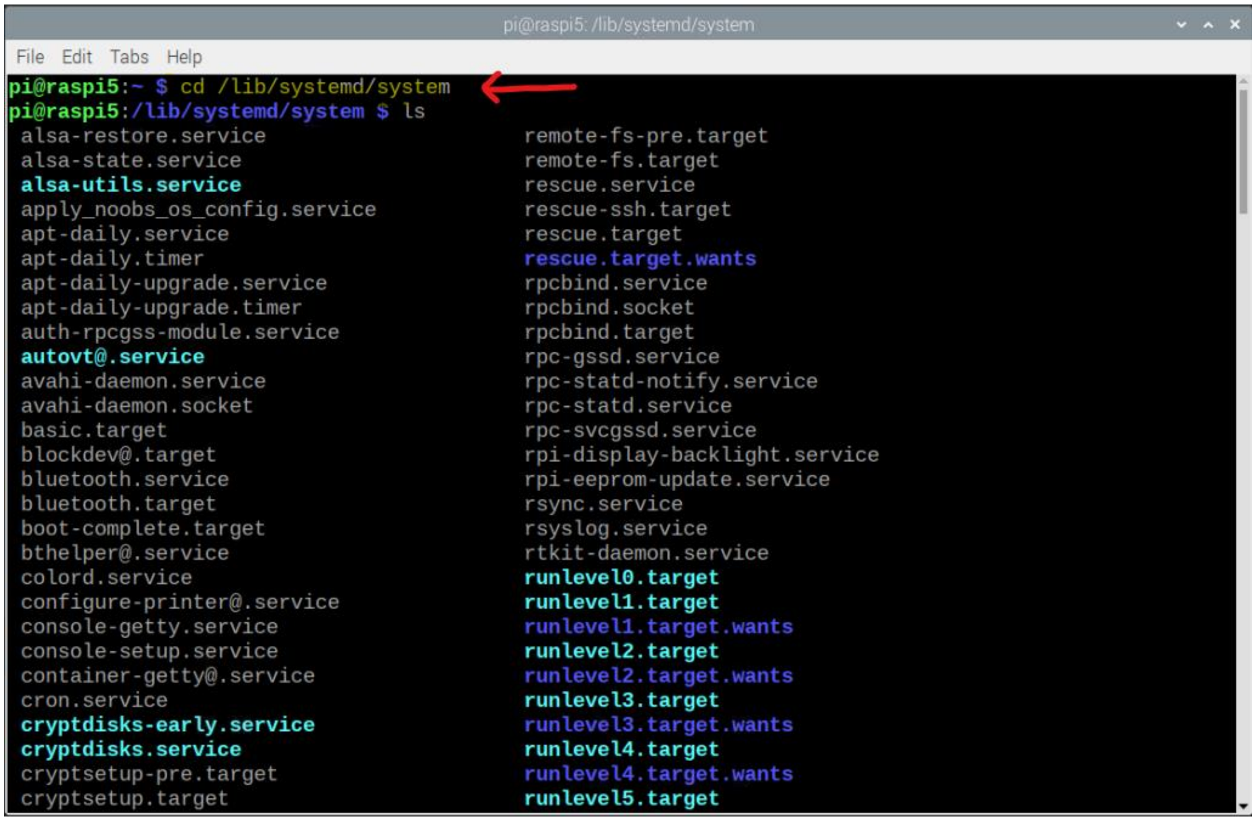
Now that we have a Python program script, and we know it's working correctly, let's create a new **systemd service**.

Step to follow:

1. Open the **Terminal**

2. Go to "*/lib/systemd/system*". All services will be in the "*/lib/systemd/system*" folder. On boot, **systemd** will look after all enabled services and start them.

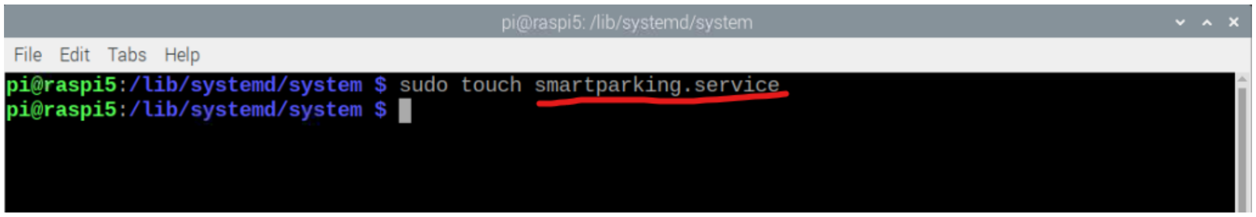
```
cd /lib/systemd/system
```



```
pi@raspi5:~ $ cd /lib/systemd/system
pi@raspi5:/lib/systemd/system $ ls
 alsa-restore.service                      remote-fs-pre.target
 alsa-state.service                        remote-fs.target
 alsa-utils.service                        rescue.service
 apply_noobs_os_config.service             rescue-ssh.target
 apt-daily.service                         rescue.target
 apt-daily.timer                           rescue.target.wants
 apt-daily-upgrade.service                 rpcbind.service
 apt-daily-upgrade.timer                   rpcbind.socket
 auth-rpcgss-module.service                rpcbind.target
 autovt@.service                           rpc-gssd.service
 avahi-daemon.service                      rpc-statd-notify.service
 avahi-daemon.socket                       rpc-statd.service
 basic.target                              rpc-svcgssd.service
 blockdev@.target                          rpi-display-backlight.service
 bluetooth.service                         rpi-eeprom-update.service
 bluetooth.target                          rsync.service
 boot-complete.target                      rsyslog.service
 bthelper@.service                         rtkit-daemon.service
 colord.service                            runlevel0.target
 configure-printer@.service                runlevel1.target
 console-getty.service                     runlevel1.target.wants
 console-setup.service                     runlevel2.target
 container-getty@.service                  runlevel2.target.wants
 cron.service                              runlevel3.target
 cryptdisks-early.service                  runlevel3.target.wants
 cryptdisks.service                        runlevel4.target
 cryptsetup-pre.target                     runlevel4.target.wants
 cryptsetup.target                         runlevel5.target
```

3. Create the new system service namely "**smartparking.service**".

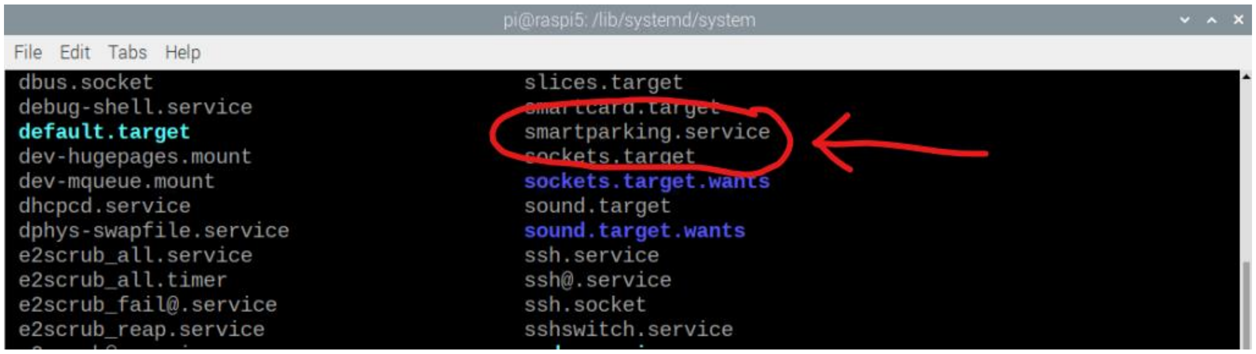
```
sudo touch smartparking.service
```



```
pi@raspi5:/lib/systemd/system $ sudo touch smartparking.service
pi@raspi5:/lib/systemd/system $ 
```

4. Verify if service "**smartparking.service**" created successfully. Type `ls` to the Terminal

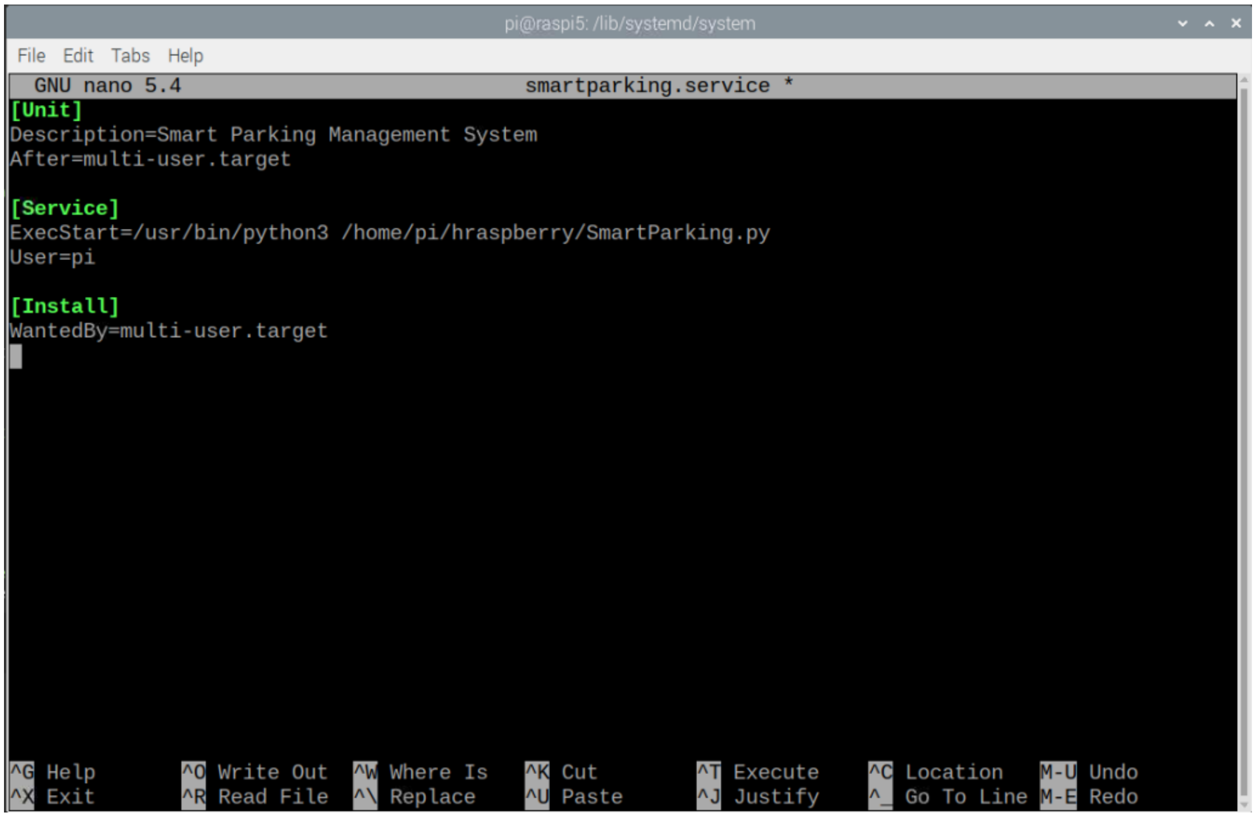
```
ls
```



```
dbus.socket                              slices.target
debug-shell.service                      smartcard.target
default.target                           smartparking.service
dev-hugepages.mount                      sockets.target
dev-mqueue.mount                         sockets.target.wants
dhcpcd.service                           sound.target
dphys-swapfile.service                   sound.target.wants
e2scrub_all.service                      ssh.service
e2scrub_all.timer                        ssh@.service
e2scrub_fail@.service                    ssh.socket
e2scrub_reap.service                     sshswitch.service
```

5. Now, edit this file (with **sudo** nano smartparking.service) and write the following:

```
[Unit]
Description=Smart Parking Management System
After=multi-user.target

[Service]
ExecStart=/usr/bin/python3 /home/pi/hraspberry/SmartParking.py
User=pi

[Install]
WantedBy=multi-user.target
```



```
  GNU nano 5.4                          smartparking.service *
[Unit]
Description=Smart Parking Management System
After=multi-user.target

[Service]
ExecStart=/usr/bin/python3 /home/pi/hraspberry/SmartParking.py
User=pi

[Install]
WantedBy=multi-user.target

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line  M-E Redo
```

Fig 2. Systemd service script

Important:

This is a very basic template for writing a **systemd** service. Here's what each field means:

- **Description**: not important, just write anything you want here about the project.
- **After**: this controls when the service should be triggered. Basically, *"multi-user.target"* means that the service will be triggered once the multi-user environment is available. Note that this will not wait until the login or desktop screen appears.
- **ExecStart**: the actual command line to execute. Here we start the Python script that we've created in the home repository. Note that it's important to use an absolute path for the script path.
- **User**: if we don't specify the user, then the program will be executed as root user. What could be the problem here? Well, the Python script we execute is creating a file. If the root user launches the Python script, then the file will be created by the root user, not by the "*pi*" user. So, the "pi" user might encounter permission issues when trying to modify/remove this script.
- **WantedBy**: as we specified "*multi-user.target*" before, we need to also write it here.

1.3. Enable the systemd service

Need to enable the service so systemd will run it on boot.

Step to follow:

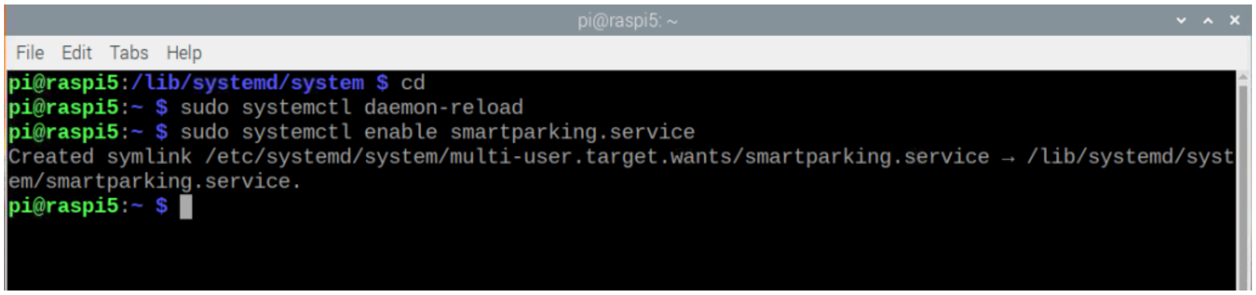
1. We need to tell systemd to recognize our service, so enter:

```
sudo systemctl daemon-reload
```

* Note that you will need to enter this command every time you change your ***.service*** file, as **systemd** needs to know it has been updated.

2. Tell systemd that the service needs to start on boot.

```
sudo systemctl enable smartparking.service
```



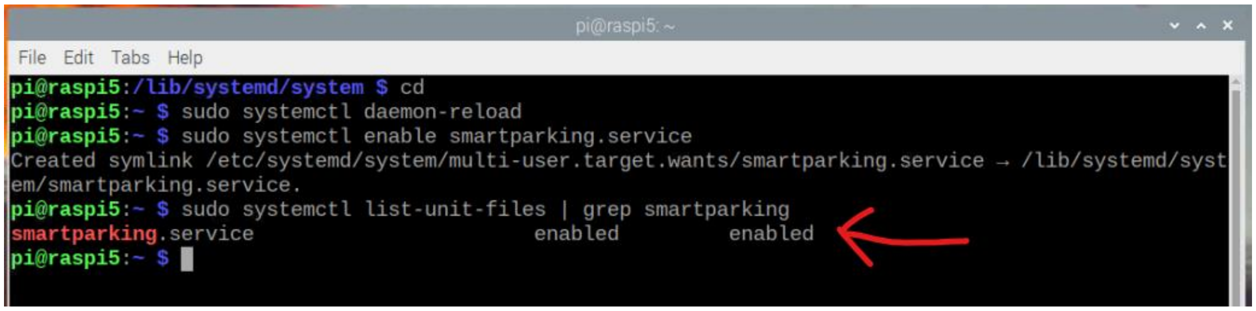
```
pi@raspi5:/lib/systemd/system $ cd
pi@raspi5:~ $ sudo systemctl daemon-reload
pi@raspi5:~ $ sudo systemctl enable smartparking.service
Created symlink /etc/systemd/system/multi-user.target.wants/smartparking.service → /lib/systemd/syst
em/smartparking.service.
pi@raspi5:~ $
```

* Reboot with **sudo reboot** to verify that the program works. The LED should begin to turn ON after the Pi boots!

3. To check if the service is enabled, use this command:

```
sudo systemctl list-unit-files | grep smartparking
```

* This tells that 1. The "smartparking" service exists, and 2. it is enabled.



```
pi@raspi5:/lib/systemd/system $ cd
pi@raspi5:~ $ sudo systemctl daemon-reload
pi@raspi5:~ $ sudo systemctl enable smartparking.service
Created symlink /etc/systemd/system/multi-user.target.wants/smartparking.service → /lib/systemd/syst
em/smartparking.service.
pi@raspi5:~ $ sudo systemctl list-unit-files | grep smartparking
smartparking.service                      enabled         enabled
pi@raspi5:~ $
```

1.4. Disable the systemd service

1. Need to disable the service, simply do:

```
sudo systemctl disable smartparking.service
```

2. Code Explanation

2.1. Python code “SmartParking.py”

```
"""
Assignment Title: Gestion de parking intelligent
Purpose          : Implementation of Smart Parking
Language         : Python
Author           : Hemant Ramphul
Github           : https://github.com/hemantramphul/Smart-Parking/
Date             : 05 January 2023

Université des Mascareignes (UdM)
Faculty of Information and Communication Technology
Master Artificial Intelligence and Robotics
Official Website: https://udm.ac.mu
"""

import RPi.GPIO as GPIO  # Importing the library of RPi.GPIO
import time  # Importing the library of time
import serial  # Importing the library of time
import string  # Importing the library string
import random  # Importing the library random
import os  # Importing the library os
from datetime import datetime  # Importing the library datetime

# Format to store date
fmt = '%Y-%m-%d %H:%M:%S'

# Filename
fileLogReserve = "log-reserve.txt"  # Log all reserved parking info
fileLogCancel = "log-cancel.txt"  # Log all cancellation parking info

# [True] -> Reservation mode
# [False] ->  Unlock mode
modeUnlockOrReserve = True

# Parking
parkingOneAvailable = True
parkingTwoAvailable = True

led = 17  # Declaring GPIO 17 of Raspberry Pi
sensor = 27  # Declaring GPIO 27 of Raspberry Pi

GPIO.setmode(GPIO.BCM)  # Declaring the BCM mode of pins
GPIO.setwarnings(False)  # Disabled warning
GPIO.setup(sensor, GPIO.IN)  # Set the behaviour of sensor as input
GPIO.setup(led, GPIO.OUT)  # Set the behaviour of led as output
```

```
def secretCode(length=4):
    """
    Function to generate a secret code
    :param length: 4 is the default number
    :return: String [secret code]
    """
    # Choose from all digits 0123456798
    letters = string.digits
    # Generate secret code
    code = ''.join(random.choice(letters) for i in range(length))

    return code  # Generated secret code


def sensorCheck():
    """
    Function to check if a car is near sensor
    :return: boolean [True or False]
    """
    try:
        while GPIO.input(sensor):  # Checking input on sensor
            GPIO.output(led, False)  # Led turned on
            while GPIO.input(sensor):  # Checking input on sensor again
                time.sleep(0.2)  # Time delay of 0.2 seconds
        GPIO.output(led, True)  # Led turned off if there is no input on
sensor
    except KeyboardInterrupt:  # If any key is pressed on keyboard terminate
the program
        GPIO.cleanup()  # Cleanup the GPIO pins for any other program use

    return GPIO.input(sensor)


def splitLog(line):
    """
    Function to split line contains ";"
    :param line:
    :return: array
    """
    return line.split(';')


def CheckParkingExpiredTime(reserved_time):
    """
    Function to check expired date with 15 seconds
    :param reserved_time:
    :return: boolean [True or False]
    """
    # Check expiration time
    t = datetime.now()
    expired = datetime.strftime(t, fmt) - datetime.strptime(reserved_time,
fmt)
    # Returns the difference of the time
    minutes = divmod(expired.seconds, 60)

    # Check if time have pass 15 seconds
    return True if minutes[1] > 15 else False


def readResponse():
    """
    Function to read Arduino response
    :return:
    """
    while arduino.inWaiting() == 0: pass  # Wait for Arduino to answer if any
    if arduino.inWaiting() > 0:
        answer = arduino.readline()  # Read answer
        print("{}".format(answer))  # Display answer


def writeToFile(code, parking):
    """
    Function to write to a log file [Reserve]
    :param code: secret code
    :param parking: Parking number
    :return:
    """
    fileToWrite = open(fileLogReserve, "a")  # Append mode
    # Save record like "code,parking_number,reserved,datetime"
    fileToWrite.write(code + ";" + parking + ";Reserved;" +
str(datetime.strftime(datetime.now(), fmt)) + "\n")
    fileToWrite.close()  # Close file connection


def writeToFileCancel(line):
    """
    Function to write to a log file [Cancellation]
    :param line: log
    :return:
    """
    fileToWrite = open(fileLogCancel, "a")  # Append mode
    slotInfo = splitLog(line)  # Split line to takes some info
    fileToWrite.write(
        slotInfo[0] + ";" + slotInfo[1] + ";Canceled;" +
str(datetime.strftime(datetime.now(), fmt)) + "\n")
    fileToWrite.close()  # Close file connection


def deleteLine(code):
    """
    Function to delete a line in log file
    :param code: secret code
    :return: boolean, String
    """
    log = ""
    deleted = False
    with open(fileLogReserve, "r") as input:  # Read from file
        with open("temp.txt", "w") as output:  # Use a temp file
            # Iterate all lines from file
            for line in input:
                if line.strip("\n").startswith(code):
                    log = line
                    writeToFileCancel(line)
                    deleted = True
                # If line starts with substring 'code' then don't write it in
temp file
                if not line.strip("\n").startswith(code):
                    output.write(line)

    # Replace file with original name
    os.replace('temp.txt', fileLogReserve)

    return deleted, log


def searchLog(code):
    """
    Function to search line in log file
    :param code: secret code
    :return: String
    """
    log = ""
    with open(fileLogReserve, "r") as input:
        for line in input:
            # If line contain Secret Code
            if line.strip("\n").startswith(code):
                log = line

    return log


# Main application start
if __name__ == '__main__':
    print('Smart Parking System. \nStatus [Running]. Press CTRL-C to
exit.\n')  # App info
    GPIO.output(led, True)  # Led turned on

    arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=1)  # Connect to
Arduino via specific port
    arduino.reset_input_buffer()

    time.sleep(0.1)
    if arduino.isOpen() and not sensorCheck():
        print("{} connected!".format(arduino.port))  # Check Arduino if is
well connected

        try:
            # Show message for mode option
            print(
                '[*] Type "R" to enter reservation mode. \n[*] Type "U" to
enter unlock mode. \n[*] Type "C" to enter '
                'cancelation mode.\n')

            while True:
                # User input according to option available
                userInput = input('Enter MODE "R" or "U" or "C": ')

                # Parking canceled
                if userInput.upper() == 'C':
                    # Enter secret code
                    userCode = input('Enter Secret Code: ')
                    deleted, log = deleteLine(userCode)  # Call function
deleteLine to cancel reservation
                    slotInfo = splitLog(log)  # Split line to take some info
                    if deleted:
                        print(slotInfo[1], 'canceled.\n')
                        if slotInfo[1] == 'Parking 1':  # Check which car
Parking to cancel according to the secret code
                            arduino.write(b'1P1')  # Send Arduino information
to perform

                            parkingOneAvailable = not parkingOneAvailable
                        else:
                            arduino.write(b'1P2')  # Send Arduino information
to perform

                            parkingTwoAvailable = not parkingTwoAvailable
                    else:  # If secret code not found
                        print('Invalid secret code!\n')

                # Parking Reservation
                elif userInput.upper() == 'R':
                    # Enter Parking number
                    print('[*] Type "1" for Parking 1. \n[*] Type "2" for
Parking 2. \n')
                    userParkingInput = input('Enter PARKING "1" or "2": ')
                    code = secretCode()  # Generate secret code
                    available = False
                    saveMsg = "Parking " + userParkingInput
                    msg = ""

                    if userParkingInput == "1" and parkingOneAvailable:
                        parkingOneAvailable = not parkingOneAvailable
                        available = True
                        arduino.write(b'2P1')  # Send Arduino information to
perform

                        readResponse()  # Read Arduino response
                        writeToFile(code, saveMsg)  # Write to log file
                    elif userParkingInput == "2" and parkingTwoAvailable:
                        parkingTwoAvailable = not parkingTwoAvailable
                        available = True
                        arduino.write(b'2P2')  # Send Arduino information to
perform

                        readResponse()  # Read Arduino response
                        writeToFile(code, saveMsg)  # Write to log file
                    elif not parkingOneAvailable and not parkingTwoAvailable:
                        # Check if parking available or full
                        msg = "Parking full!"
                    else:
                        # Check selected parking
                        msg = "Selected parking is not available."

                    print(msg)
                    print("Secret Code: " + code if available else "")
                    print('----------------------------------\n')

                    modeUnlockOrReserve = True
                # Parking Unlock
                elif userInput.upper() == 'U':
                    # Enter secret code
                    userCode = input('Enter Secret Code: ')
                    log = searchLog(userCode)  # Search information about the 
Parking reserved with the secret code

                    if log != "":
                        available = False
                        # Get information in array
                        slotInfo = splitLog(log)

                        # Check expired date
                        if CheckParkingExpiredTime(slotInfo[3].strip()):
                            print('Secret Code expired!\n')
                        else:
                            if slotInfo[1] == 'Parking 1':
                                arduino.write(b'3P1')  # Send Arduino 
information to perform
                            else:
                                arduino.write(b'3P2')  # Send Arduino 
information to perform

                        readResponse()
                        deleted, log = deleteLine(userCode)  # Delete log in 
log file

                    modeUnlockOrReserve = False
                else:
                    print('Invalid option!\n')
                    print('----------------------------------\n')

        except KeyboardInterrupt:
            print("KeyboardInterrupt has been caught.")
```

2.2. Arduino Code "SmartParking.ino"

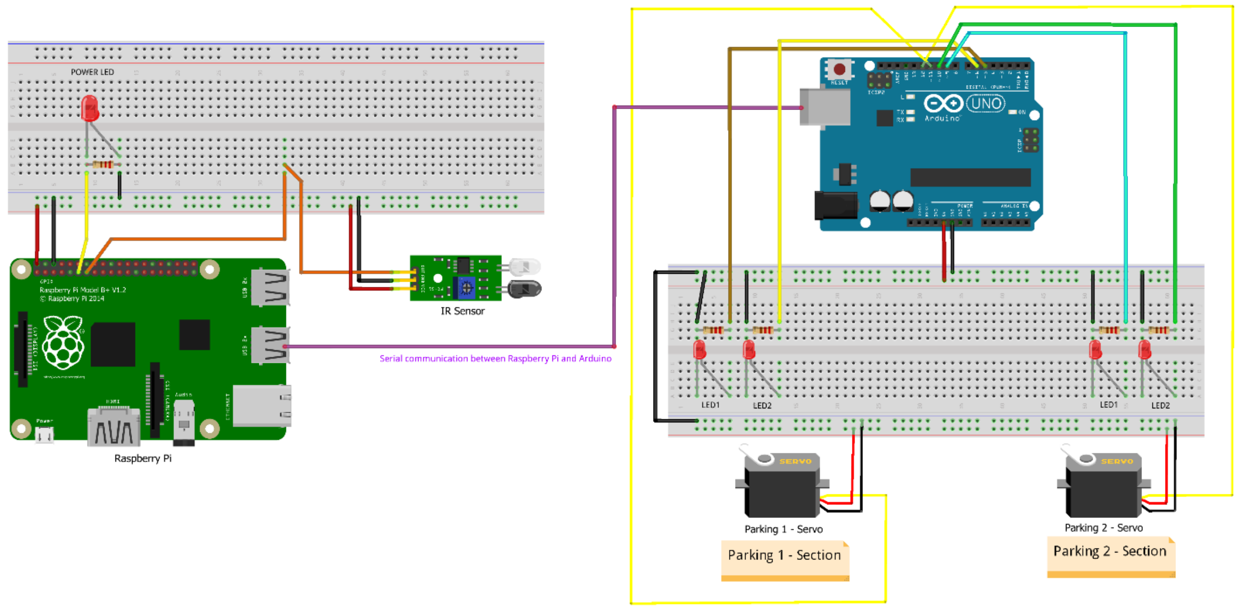
```
#include <Servo.h>
// Connection from Raspberry Pi 4 to Arduino

Servo P1_SERVO; // Create servo object to control Parking 1 servo
Servo P2_SERVO; // Create servo object to control Parking 2 servo
int P1_LED1_PIN = 5; // Assign as YELLOW Led 1 (Parking 1)
int P1_LED2_PIN = 6; // Assign as YELLOW Led 2 (Parking 1)
int P2_LED1_PIN = 9; // Assign as YELLOW Led 1 (Parking 2)
int P2_LED2_PIN = 10; // Assign as YELLOW Led 2 (Parking 2)

int P1_SERVO_PIN = 11; // Assign servo (Parking 1)
int P2_SERVO_PIN = 12; // Assign servo (Parking 1)

String MSG; // Read Input
bool P1_AVAILABLE = true; // Disponibility of Parking 1
bool P2_AVAILABLE = true; // Disponibility of Parking 2
```

```
int pos = 0; // Variable to store the servo position

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(P1_LED1_PIN, OUTPUT);
  pinMode(P1_LED2_PIN, OUTPUT);
  pinMode(P2_LED1_PIN, OUTPUT);
  pinMode(P2_LED2_PIN, OUTPUT);
  P1_SERVO.attach(11);  // Attaches the servo on pin 11 to the servo object
  P2_SERVO.attach(12);  // Attaches the servo on pin 12 to the servo object

  P1_SERVO.write(map(0, 0, 1023, 0, 180)); // Status: Upper (default) - Parking 1
  P2_SERVO.write(map(0, 0, 1023, 0, 180)); // Status: Upper (default) - Parking 2

  delay(10);
  ParkingAvailable();
}

void loop() {
  // Read serial port
  ReadSerialPort();

  // Available: Parking 1 and Parking 2
  if (MSG == "1P1") { // Available: P1_LED1_PIN ON, P1_LED2_PIN OFF
    digitalWrite(P1_LED1_PIN, HIGH); // (Parking 1)
    digitalWrite(P1_LED2_PIN, LOW); // (Parking 1)
    P1_SERVO.write(map(0, 0, 1023, 0, 180)); // Status: Upper (default) - Parking
1
    Serial.println("Parking 1 available!");
  } else if (MSG == "1P2") { // Available: P2_LED2_PIN ON, P2_LED2_PIN OFF
    digitalWrite(P2_LED1_PIN, HIGH); // (Parking 2)
    digitalWrite(P2_LED2_PIN, LOW); // (Parking 2)
    P2_SERVO.write(map(0, 0, 1023, 0, 180)); // Status: Upper (default) - Parking
2
    Serial.println("Parking 2 available!");
  }

  // Reserved: Parking 1 and Parking 2
  if (MSG == "2P1") { // Reserved: P1_LED1_PIN ON, P1_LED2_PIN ON
    digitalWrite(P1_LED1_PIN, HIGH); // (Parking 1)
    digitalWrite(P1_LED2_PIN, HIGH); // (Parking 1)
    Serial.println("Parking 1 reserved!");
  } else if (MSG == "2P2") { // Reserved: P2_LED1_PIN ON, P2_LED2_PIN ON
```

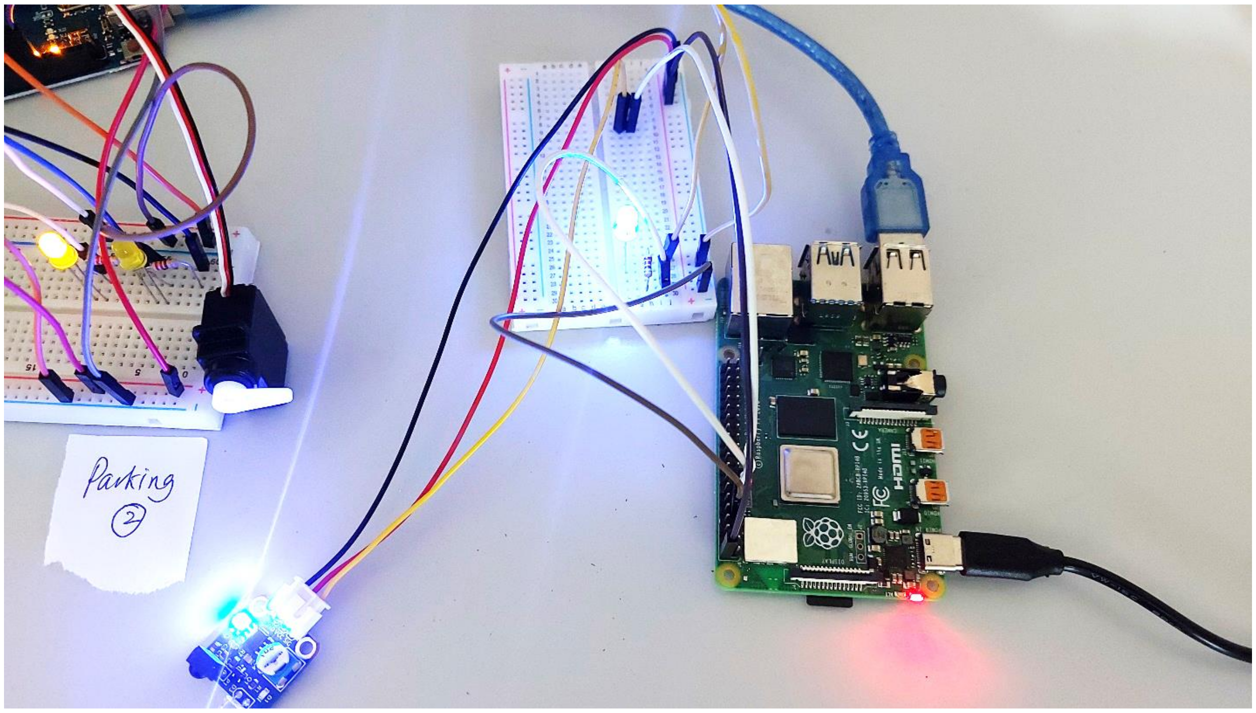
```
    digitalWrite(P2_LED1_PIN, HIGH); // (Parking 2)
    digitalWrite(P2_LED2_PIN, HIGH); // (Parking 2)
    Serial.println("Parking 2 reserved!");
  }

  // Reserved and Parked: Parking 1 and Parking 2
  if (MSG == "3P1") { // Reserved and Parked: P1_LED1_PIN OFF, P1_LED2_PIN ON
    digitalWrite(P1_LED1_PIN,LOW); // (Parking 1)
    digitalWrite(P1_LED2_PIN,HIGH); // (Parking 1)
    P1_SERVO.write(map(0, 0, 1023, 180, 180)); // Status: Lower/Close - Parking 1
    Serial.println("Parking 1 reserved and parked!");
  } else if (MSG == "3P2") { // Reserved and Parked: P1_LED1_PIN OFF, P1_LED2_PIN
ON
    digitalWrite(P2_LED1_PIN,LOW); // (Parking 2)
    digitalWrite(P2_LED2_PIN,HIGH); // (Parking 2)
    P2_SERVO.write(map(0, 0, 1023, 180, 180)); // Status: Lower/Close (default) -
Parking 2
    Serial.println("Parking 2 reserved and parked!");
  }

  // Not Reserved and Parked: Parking 1 and Parking 2
  if (MSG == "4P1") { // Reserved and Parked: P1_LED1_PIN OFF, P1_LED2_PIN
ON/BLINK
    digitalWrite(P1_LED1_PIN,LOW); // (Parking 1)
    BlinkLedParking1(); // (Parking 1)
    Serial.println("Parking 1 reserved and parked!");
  } else if (MSG == "4P2") { // Reserved and Parked: P1_LED1_PIN OFF, P1_LED2_PIN
ON/BLINK
    digitalWrite(P2_LED1_PIN,LOW); // (Parking 2)
    BlinkLedParking2(); // (Parking 2)
    Serial.println("Parking 2 reserved and parked!");
  }

  delay(100); // Wait
}

void ReadSerialPort() {
  MSG = "";
  // Check availability
  if (Serial.available()) {
    delay(10);
    while (Serial.available() > 0) {
      MSG += (char)Serial.read();
    }
    Serial.flush();
  }
}

void ParkingAvailable() {
  digitalWrite(P1_LED1_PIN, HIGH); // (Parking 1)
  digitalWrite(P1_LED2_PIN, LOW); // (Parking 1)
  digitalWrite(P2_LED1_PIN, HIGH); // (Parking 2)
  digitalWrite(P2_LED2_PIN, LOW); // (Parking 2)
}

void BlinkLedParking1() {
  while (Serial.available() < 2) {
    //Serial.println(Serial.available());
    digitalWrite(P1_LED2_PIN, HIGH); // (Parking 1)
    delay(500);
    digitalWrite(P1_LED2_PIN, LOW); // (Parking 1)
    delay(500);
  }
}

void BlinkLedParking2() {
  while (Serial.available() < 2) {
    //Serial.println(Serial.available());
    digitalWrite(P2_LED2_PIN, HIGH); // (Parking 2)
    delay(500);
    digitalWrite(P2_LED2_PIN, LOW); // (Parking 2)
    delay(500);
  }
}
```

3. Setup Raspberry Pi 4 and Arduino

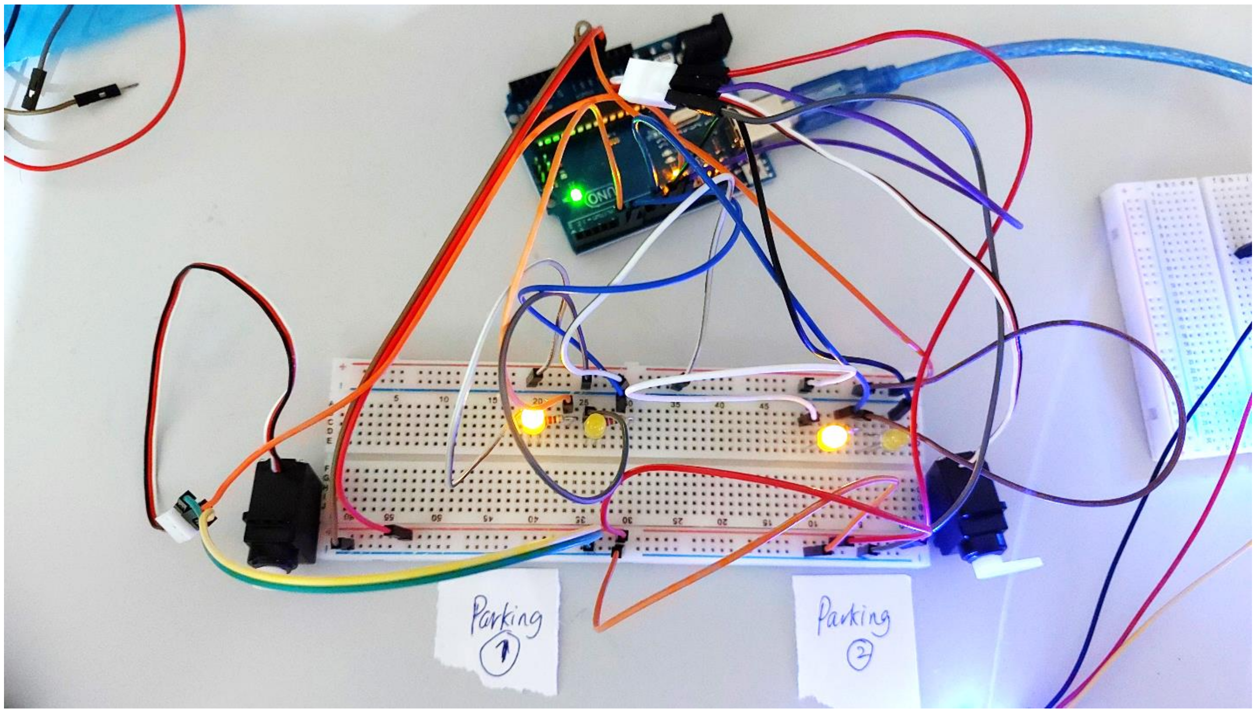
3.1. Wiring



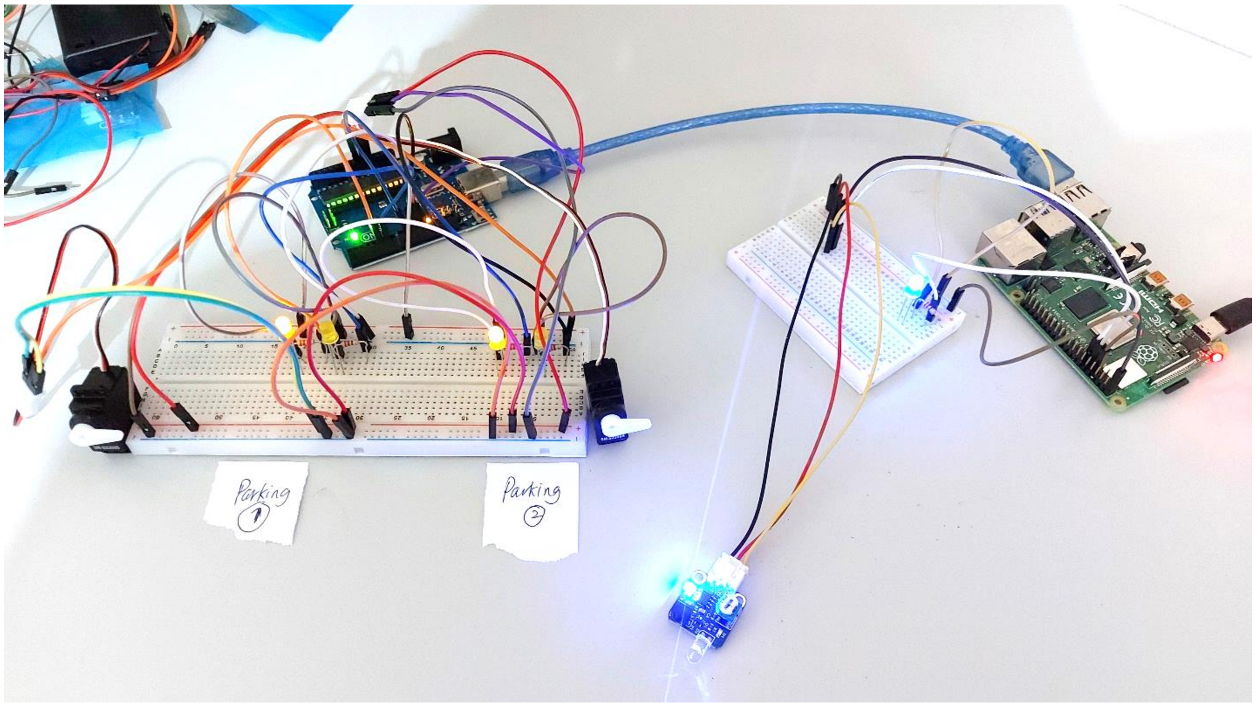
3.2. Raspberry Pi4 Setup



3.3. Arduino Setup



3.4. Raspberry Pi 4 and Arduino together



4. Source code

Full **source code** and other files are available in Github which is 100% executable, and each line is explained.

4.1. Download link.

Download link:

https://github.com/hemantramphul/Smart-Parking