Nicholas Lopez
ID: 80608107
May 2023

Smart Parking System

I) Introduction:

It has been reported that 30% of traffic is a factor not being able to efficiently find parking [1]. Many smart cities such as San Francisco, Malaga, London, Moscow, and Los Angeles have deployed smart parking systems to ease the flow of traffic. Some benefits of these smart parking systems, such as reducing traffic, reducing wasted fuel, increasing safety, and being more environmentally sound [2]. There are many approaches taken in smart parking systems with machine learning being incorporated into the design. There are three approaches that can be taken such as relying on sensors, camera-based observations, and crowdsensing [3]. Weight and proximity sensors are typically used in smart parking systems to detect when a car is in a particular spot. Some garages have even implemented lights at the top of each spot to indicate if a spot is taken or available. One such example that I have experienced is with the Winchester Lot in San Jose, CA. The figure below displays some features of the parking structure:

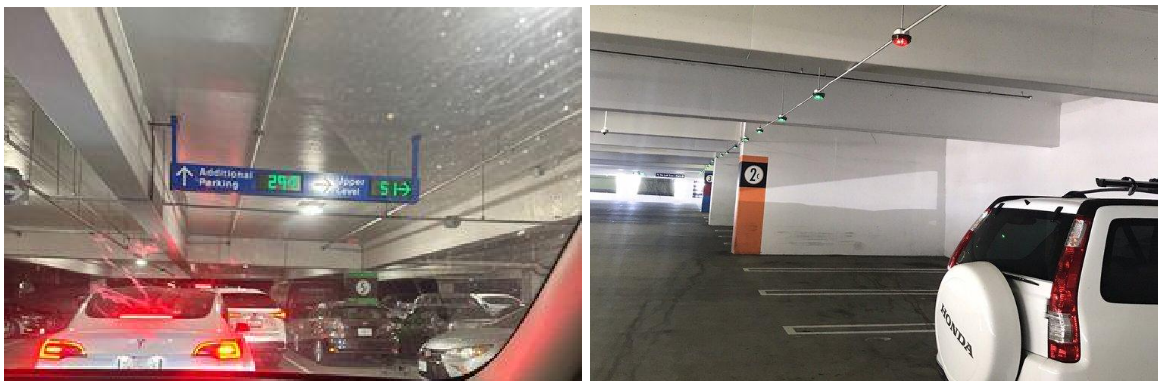

Figure 1: Winchester Smart Parking Lot

It can be seen from Figure 1 how the count is displayed on signs throughout each level to show how many parking spaces are available in a given area and above/below that level. The count display and availability displays were a great help during weekends, especially when it would become crowded in the evening. While sensors are useful, it can be impractical to place a sensor in every single spot. Parking structures can exceed up to a number of parking spaces
Sensors are reliable, but can be an expensive approach to smart parking. A camera-based approach incorporating computer vision can be more cost-effective in designing the smart parking system. It can also be incorporated with smart surveillance to keep track of activities within a parking lot/garage. This has recently been implemented by using computer vision to detect license plates. While a camera-based approach may be more cost efficient, there comes the issue of requiring a high bandwidth. As an alternative, crowdsensing is often used to determine peak hours of smart parking lots and to estimate the number of spots available.

While it would be ideal to have a combined approach, this project will combine aspects of IoT such as sensors, actuators, and protocols to implement a prototype smart parking system. For this design, RFID will be used as the main protocol to allow entry and exit of drivers in the parking lot. With the presence of a tag, the RFID module will be used with a servo motor to automate entry and exit. For the sensor end of the project, IR sensors will be used to detect the presence or absence of a vehicle and display the number of spots available. This prototype will be used to simulate the sensor approach used in smart parking systems. With this implementation, users will have the simplicity of a parking system that reduces traffic while using ideas from IoT.

II) Tools Required:

- Raspberry Pi
- Python (to code the raspberry pi)
- IR/Proximity Sensors
- RFID Module with tag and programmable card (MFRC-522)
- Toy Cars (to model vehicles)
- Motors (to control gate entry/exit)
- LCD Display (to display count)

III) Implementation Details with Code:

The initial part of the project was to test all the hardware and see how each module functions. The project began with testing the IR proximity sensors and the LCD Display. The LCD display was set up to incorporate the parking space count that takes input from the sensors. The figure below displays the pin configuration of the LCD display and the IR sensors:

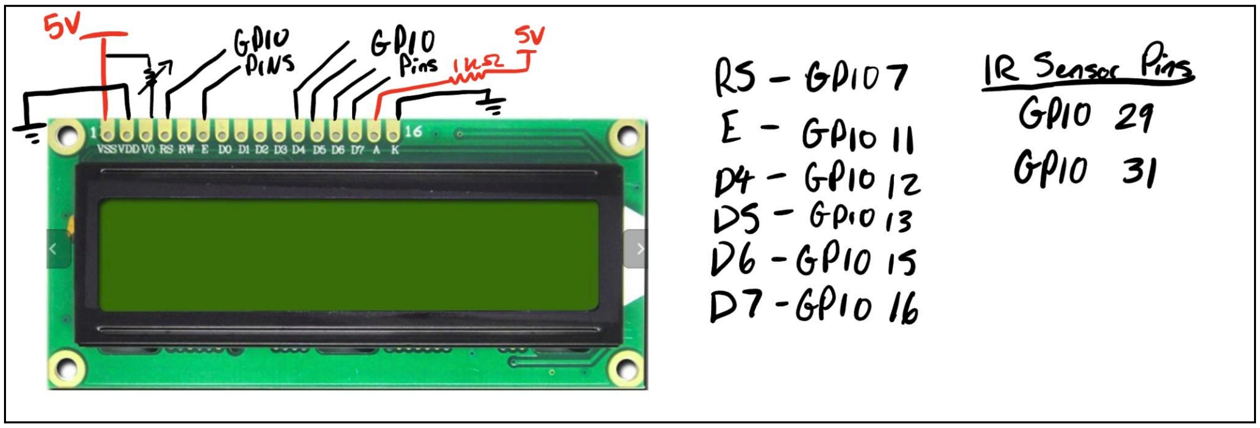

Figure 2: LCD Pin Configuration

Once the LCD display was correctly set up, the IR sensors were tested to make sure that they could detect objects in close proximity. There are three main connections for each IR sensor. The sensors require 3.3V power supply, a GPIO connection to the raspberry pi, and a connection to ground. Once the IR sensors were connected, they were tested to make sure that they could detect objects. When an object was brought into a range of about 5cm, a light would go off to signal that an object was detected.

The next part involved setting up the RFID module to function with a servo motor. The RFID module had the purpose in validating users who enter and exit the gate of the garage. The RFID module prompts users to bring the tag near the reader. Once brought near, the data is written and the servo motor moves to act as an opening gate. There is a three second delay to allow users to enter, and the motor moves back to the original position. The code keeps checking until the tag is brought near the reader again so that the user can exit. Upon exiting, the Python shell reads "Have a nice day!" The figure below displays the pin configuration of the RFID module.

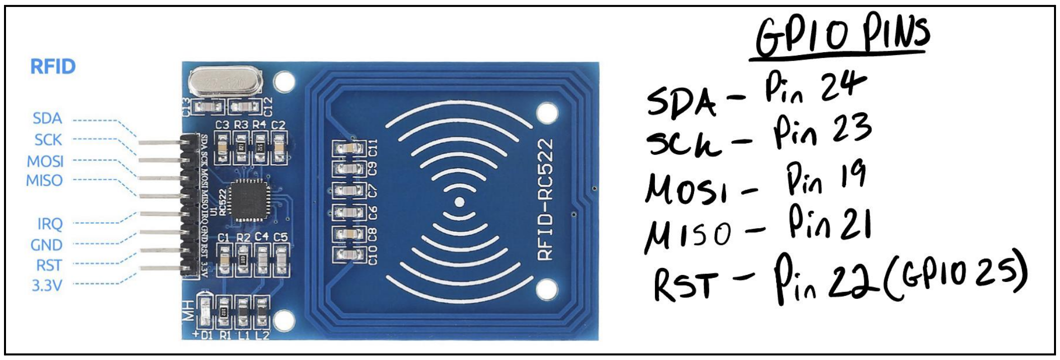

Figure 3: RFID Pin Configuration

The servo motor was set up on pin 37 of the Raspberry pi, and draws power from the 5V supply and is connected to the ground rail.

A) Setup Details:

Once each module was tested and working, the hardware could be integrated to perform as required by the design. The figure below displays the overall hardware components connected on the breadboard.

Nicholas Lopez
ID: 80608107
May 2023

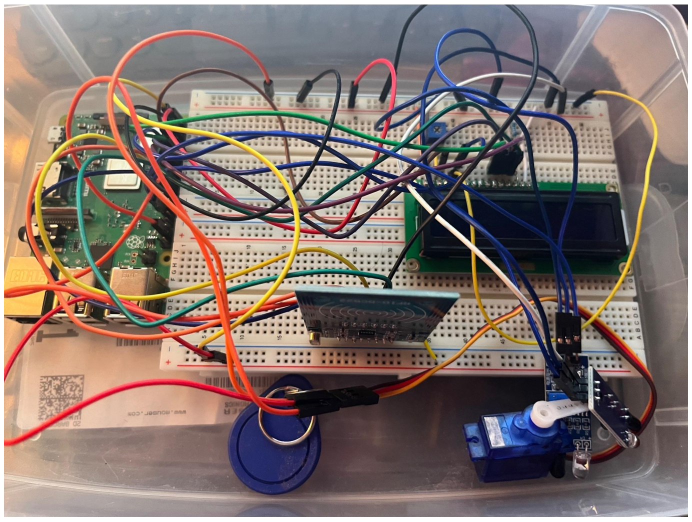
Figure 4: Hardware Implementation

The figure above shows a rough layout of the overall system. It can be seen how there are lots of wires involved, but having the pin configuration helped with keeping track of pins once implemented on the software. Once everything was set up, the RFID module and servo motor were integrated to display the count. The code below displays how the modules were integrated together to create the gate of the parking system.

Code 1:

```
reader = Simple MFRC522()
GPIO.setup(37,GPIO.OUT)

####Sets Servo to pin 37 with frequency of 50 Hz
pwm=GPIO.PWM(37,50)

###Function to move based on angle input
def SetAngle(angle):
        duty = angle/18+2
        GPIO.output(37,True)
        pwm.ChangeDutyCycle(duty)
        sleep(1)
        GPIO.output(37,False)
        pwm.ChangeDutyCycle(0)

###Prompt users to bring tag near reader for entry
text = input("Enter Name:")
print("Place tag to enter")
```

```
Try:
        ##Read data from tag and write
        reader.write(text)
        print("Written")

###Open gate for car to enter, close 2 seconds after
        pwm.start(0)
        SetAngle(90)
        sleep(2)
        SetAngle(0)

##Prompt users to bring tag near reader for exit
        print("place tag to exit\n")
        Id,text = reader.read()
        print(id)
        print("Have a nice day!")
        print(text)

##Open gate for car to exit, close 2 seconds after
        SetAngle(90)
        sleep(2)
        SetAngle(0)
        pwm.stop()
finally:
        GPIO.cleanup()
```

The next portion of the project involved integrating the IR sensors to display the count on the LCD display. The table below displays logic used to implement the count One thing that should be noted is that the IR sensors are in an active low state. Therefore, a false in the code would indicate the presence of a car and a true would represent absence.

Sensor 1	Sensor 2	Count
True	True	2
True	False	1
False	True	1
False	False	0

Table 1: Logic for Implementing Count

It can be seen from the table how having both cars absent would provide a count of two, one car would be at count 1, and both cars being present would signify that the parking lot is full. The code below displays how the IR sensors were implemented with the logic to display the count to the LCD.

Code 2:

```
####Initialize GPIO pins for Sensors and LCD data pins

##Functions to initialize, toggle, and clear LCD Display

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)
```

```
while 1:
#   lcd_string("Slots Open",LCD_LINE_1)
  slot1_status = GPIO.input(slot1_Sensor)
  time.sleep(0.2)
  slot2_status = GPIO.input(slot2_Sensor)
  time.sleep(0.2)

  if((slot1_status == True) and (slot2_status==True)):
   count=2
   lcd_string("2 ",LCD_LINE_2)

  if((slot1_status == True) and (slot2_status==False)):
```

```
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
# Define delay between readings
delay = 1
```

```
   count=1
   lcd_string("1 ",LCD_LINE_2)

  if((slot1_status == False) and (slot2_status==True)):
   count=1
   lcd_string("1 ",LCD_LINE_2)

  if((slot1_status == False) and
(slot2_status==False)):
   count=0
   lcd_string("Full ",LCD_LINE_2)

  print("Slots Open:",count)
```

It can be seen from the code how the initial display message is “Welcome to the Car Parking System.” Furthermore, messages were programmed to display on a specific line. The last portion of the project involved integrating all parts of the system together. The code below displays a general overview of how the project integrated all modules.

Code 3 (Integrated):

```
########Define the reader and the pin for servo motor
reader = SimpleMFRC522()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(38,GPIO.OUT)

############50 represents  frequency of 50Hz
pwm=GPIO.PWM(38,50)

#####Function to convert entered angle to required
duty cycles for servo motor
def SetAngle(angle):
    duty=angle/18+2
    GPIO.output(38,True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
    GPIO.output(38,False)
    pwm.ChangeDutyCycle(0)


######Prompts user to enter their name and to bring
tag near reader
text = input("Enter Name: ")
print("Place tag to enter")

try:
    ####Once tag is brought near, name is written to
RFID module
    reader.write(text)
```

```
# Send string to display

  ####Function to display "Full" when the count is 0

  #####Function to display "1 Space Open" when the
count is 1

   #####Function to display "1 Space Open" when the
count is 2


###Function used in testing to see if count could be
held
def lcd_count(count,line):
  # Send string to display
  count= count.ljust(LCD_WIDTH," ")
  lcd_byte(line, LCD_CMD)
  for i in range(LCD_WIDTH):
      lcd_byte(ord(count[i]),LCD_CHR)
##############Function to check the spots available
and to determine count
def check_spots(slot1_status,slot2_status):
  #########This is the condition if both spots are
available (Count is 2)
    if((slot1_status == True) and (slot2_status==True)):
        count=2
        time.sleep(0.2)
        count2()
    print("Written")


    ######Use Angle function to open gate and close
after 2 seconds
    pwm.start(0)
    SetAngle(90)
    sleep(2)
    SetAngle(0)


    ######Wait for tag to be brought near reader again
    print("place tag to exit\n")
    id,text = reader.read()

    #####Print tag ID and name, along with message
    print(id)
    print("Have a nice day!")
    print(text)

    ####Open and close gate again
    SetAngle(90)
    sleep(2)
    SetAngle(0)
    pwm.stop()

finally:
    GPIO.cleanup()

os.system('python3
/home/pi/Downloads/iot_finalproj-main/lcd_sens.py')
```

```
        lcd_byte(0x01,LCD_CMD)
        delay=5


#########This is the condition if one spot is available
(Count is 1)
    if((slot1_status == True) and
(slot2_status==False)):
        count=1
        time.sleep(0.2)
        count1()
        lcd_byte(0x01,LCD_CMD)
        delay=5


#########This is the condition if one spot is available
(Count is 1)
    if((slot1_status == False) and
(slot2_status==True)):
        count=1
        time.sleep(0.2)
        count1()
        lcd_byte(0x01,LCD_CMD)
        delay=5

#########This is the condition if there are no spots
available (Count is 0)
    if((slot1_status == False) and
(slot2_status==False)):
        count=0
        time.sleep(0.2)
        count0()
        lcd_byte(0x01,LCD_CMD)
        delay=5

while 1:

    #####Check the sensor status and clear display
every 0.1 seconds
    slot1_status = GPIO.input(slot1_Sensor)
    time.sleep(0.1)
    slot2_status = GPIO.input(slot2_Sensor)
    time.sleep(0.1)
    check_spots(slot1_status,slot2_status)
    time.sleep(0.1)
    lcd_byte(0x01,LCD_CMD)
    delay =5
```

It can be seen from the code how the code for the RFID is run first to allow the cars to enter the parking lot. The gate activates whenever the tag is brought near the RFID reader. The code then moves onto the while loop so that the IR sensors can consistently determine the count and be displayed on the LCD screen. The figures below display the final outcome of the smart parking system:

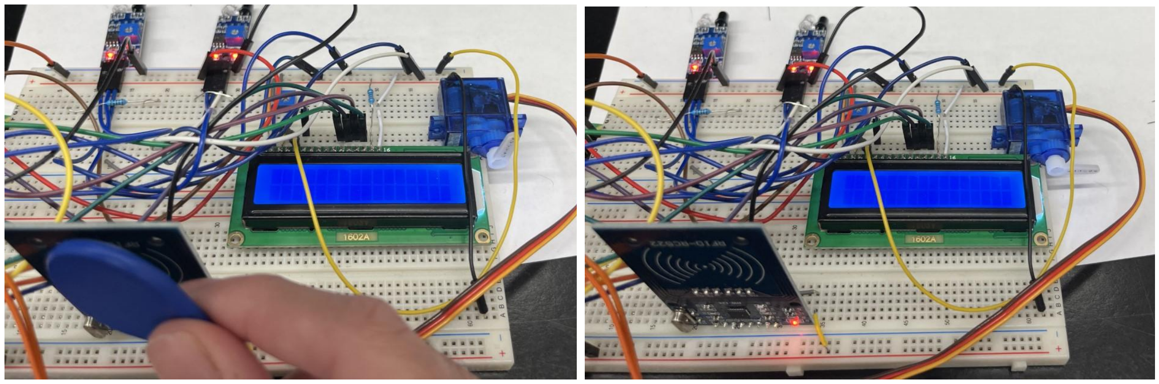

Figure 5: Entry/Exit Module with RFID Reader with servo at 90° and 0°

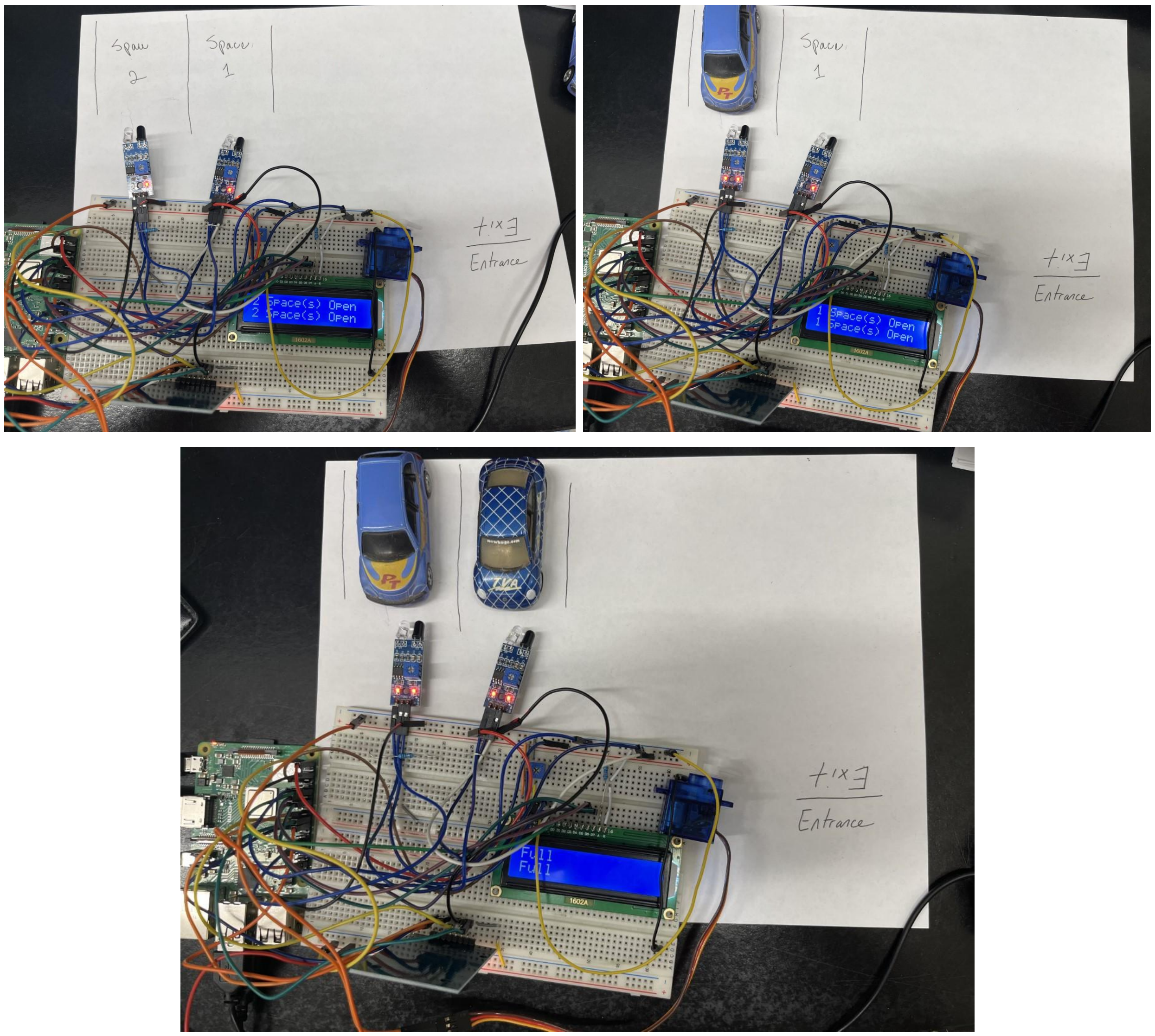

Figure 5: Count from IR Sensors

Figure 4 displays the initial stage when the tag is near the RFID reader and the servo motor points up at 90 degrees. After a couple seconds, the servo motor is brought back down to 0 degrees to close the gate. Figure 5 shows the count for the parking lot, showing when both spots are open, when one spot is open, and when the spaces are full. The parking system prototype functions as expected in the design. The full demo can be found in the zip folder of the final submission.

IV) Project Progress Details:

Week 1 (April 3rd-8th): Submit project description and get proximity sensors/RFID module started.
*Week 2 (April 9th-15th): Setup LCD display, test each individual module, and get started with creating GUI.
*Week 3 (April 16th-22nd): Finish setting up login GUI/credit card page, start with encryption of GUI, and submit first project interim report.
Week 4 (April 23rd-29th): Integrate RFID module with servo motor, integrate count with proximity sensor, and submit second project interim report.
Week 5 (April 30th-May 6th): Integrate all modules, add comments to code.
Week 6 (May 7th-10th): Verify that hardware works as expected and submit the final report.

Project Timeline

*Indicate weeks where the Cybersecurity end of project was the main focus, working more on the GUI and encryption involved with the security aspect of the project.

During Week 1, each module was tested to see how the individual hardware components function. There was also some setting up involved, such as soldering the pins for the MFRC522 module. During Week 2, the LCD display was tested with the IR sensors. However, the LCD display was just tested to see if the presence/absence of a car could be displayed. During Week 4, the RFID module was integrated with the servo motor to create the gate entrance/exit of the parking lot. Week 5 included integration of all aspects of the project, but also troubleshooting errors in the LCD display.

V) Pitfalls and Challenges:

One of the major issues was integrating the servo motor with the RFID reader. Originally, there were going to be two gates: one designated for entrance, and the other designated for exit of the parking lot. However, the MFRC-522 takes a lot of GPIO pins on the Raspberry Pi and the servo motors also pull a lot of current. As a result, the project was modified to have one designated gate for entry and exit. I read that it was best to have a driver or separate power supply for the

Nicholas Lopez
ID: 80608107
May 2023

motor. As the project was under time constraints, this was considered as an alternative option. The servo motor does have a bit of noise and jitter, but it does function as expected.

As mentioned in the progress details, one of the main issues was with the LCD display. When integrated with the IR sensors, it would display the right data for a couple of cycles, but would then go haywire when holding count. A lot of the issue had to do with timing in the while loop. To make the code easier to debug, functions were created for the count and the display. After many attempts, the LCD was still acting up upon entering the while loop. While it was not a complete fix, I was able to hold the wording in place and properly display the count. This was done by putting the initialize function for the LCD at the beginning of the while loop. This allows the LCD to hold the count determined by the sensors.

Despite the challenges, this project was one of the most fun I've had working on. It was interesting to see how the sensors interact with other parts of the project, along with how the RFID protocol could be implemented. IoT is an interesting course, and I'm glad that we got to see everything from theory to implementation. Having this project is a great way to get experience with IoT and to apply skills learned throughout the course.

References:

1. Bura, Harshitha et. al. An Edge Based Smart Parking Solution Using Camera Networks and Deep Learning (nsf.gov). 2018 IEEE International Conference on Cognitive Computing
2. Ancion, Kai. "What are the benefits of smart parking for drivers?" Parkeagle.com. March 19, 2019.
3. Barker, Jonathan and Rehman, Sabih ur. "Investigating the use of Machine Learning for Smart Parking Applications. IEEE Xplore Full-Text PDF:
4. Using Infrared (IR) Sensor with Raspberry Pi - donskytech.com
5. Use an RFID reader with the Raspberry Pi. – Howto Raspberry Pi
6. How to Setup an LCD on the Raspberry Pi and Program It With Python - Circuit Basics
7. Medjuck, Brett. "How to Set up the RC522 to Work with a Raspberry Pi." April 23, 2022. https://youtu.be/eUNZdDf70E4
8. Parvat Computer Technology. "How To Create Login Form for Apps Using Python | Tkinter Project." March 26, 2022. https://youtu.be/X9reTI_Mckk
9. Jadhav, Rahul. "Car parking System using Raspberry Pi." January 23, 2022. https://youtu.be/azCb5Zj76UQ