EMBEDDED SYSTEM AND DESIGN

EXPERIMENT NO. 5

Parking Lot Counter

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE160P-4/A1

Group No.:

Date of Performance: 09/09/2024
Date of Submission: 09/14/2024

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

In the first part of the experiment, is designing on how to improvise the parking lot counter. My idea in mind is to mimic a real-life application of this experiment which a parking lot system. In this experiment I have used 2 servo motors that will represent as the gates of the parking lot, I also used a ultrasonic senor for the detection of the cars entering and exiting the parking lot, and for my out put I have used a 4 digit seven segment display which displays the real time number of cars inside the parking lot, and show the number of slots that are available inside the parking lot. In Addition, the first two digits from the left of the display will show the available slots inside, while the last two digits on the right will show the number of cars inside the parking. I have used two breadboards which is necessary for the circuit to manage the cables to make it cleaner, and at the same time to use it to create a small road between the two breadboards as a path to enter and exit the parking lot.

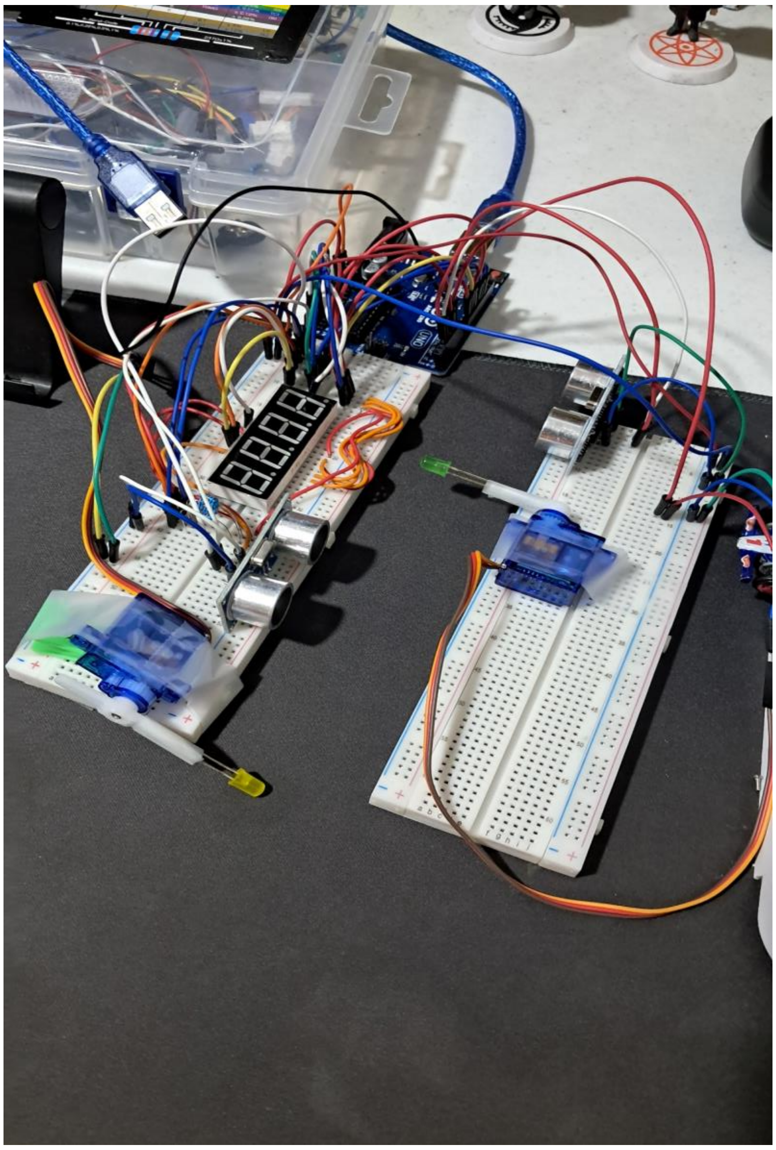


Figure 1: *The connection of the 4-digit seven segment display, ultrasonic sensors, and servo motors to the Arduino board.*

To visualize more clearly in my head, I tried the circuit with my little car toys in order to get the feel of a real parking lot in this experiment. So, basically to extend the reach of the shaft of the two servo motors I taped some LEDs on it. The gates will only open once the ultrasonic sensor detects an object within its set range. As you see in the figure the ultrasonic sensors are placed before the gates.

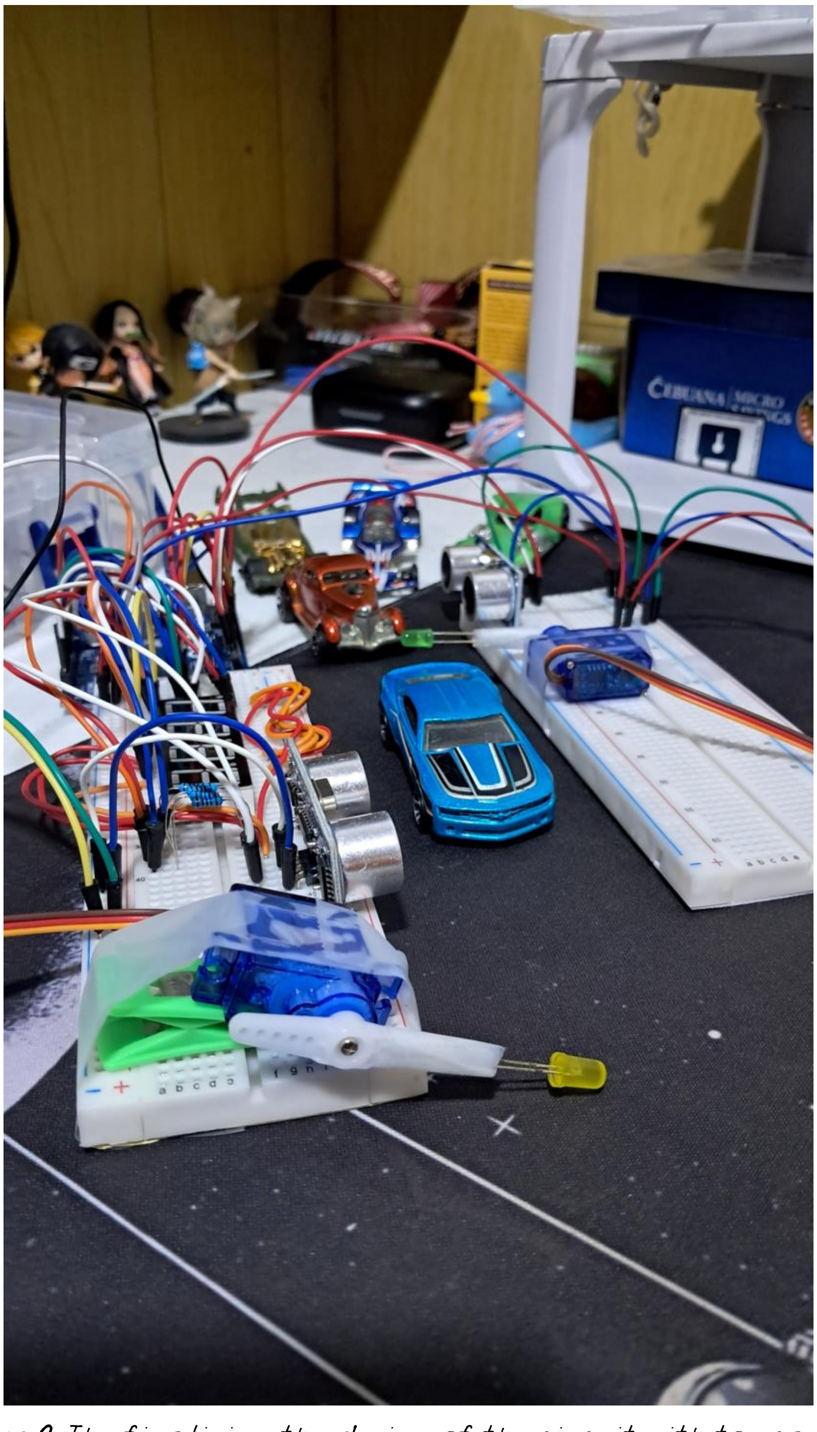
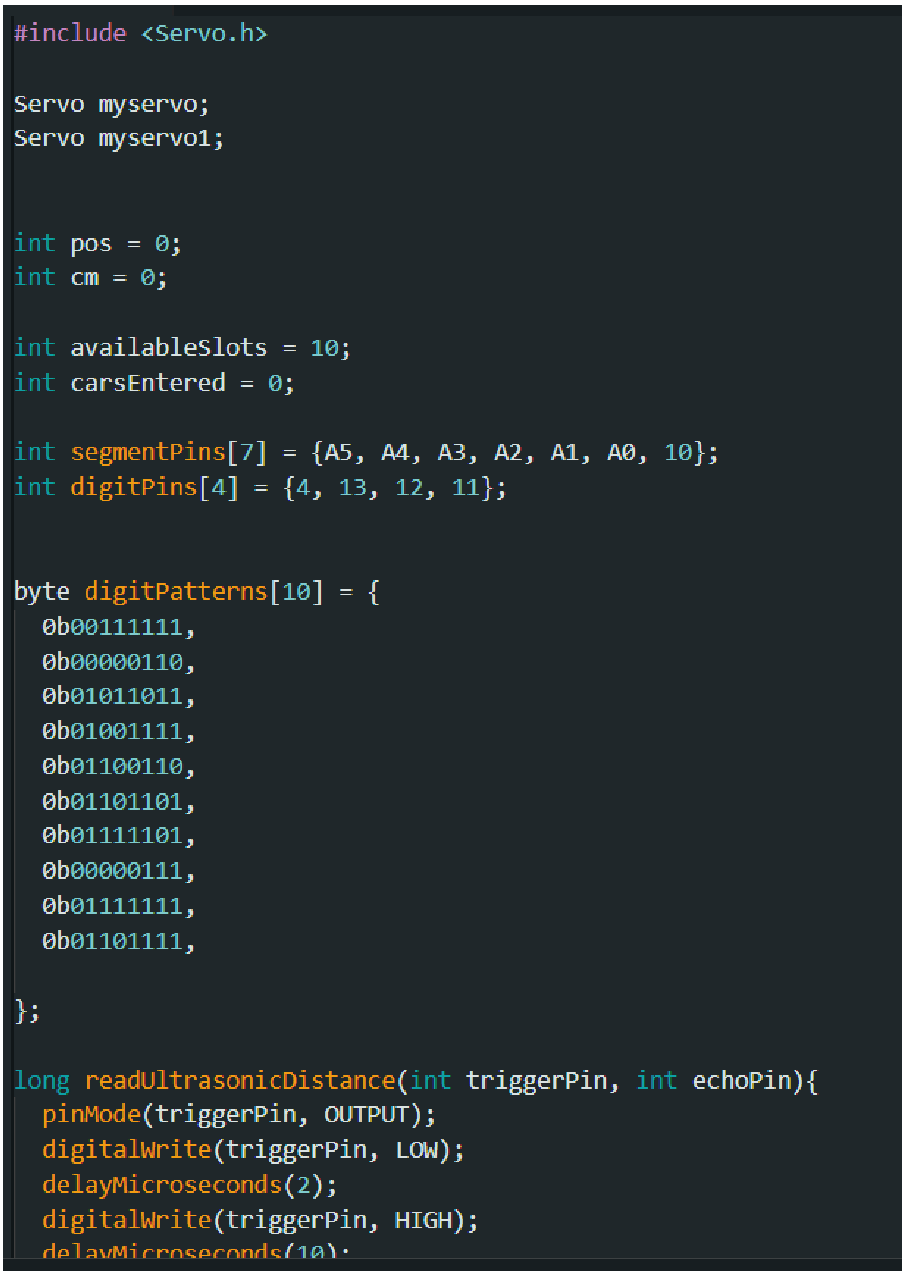


Figure 2: The finalizing the design of the circuit with toy cars.

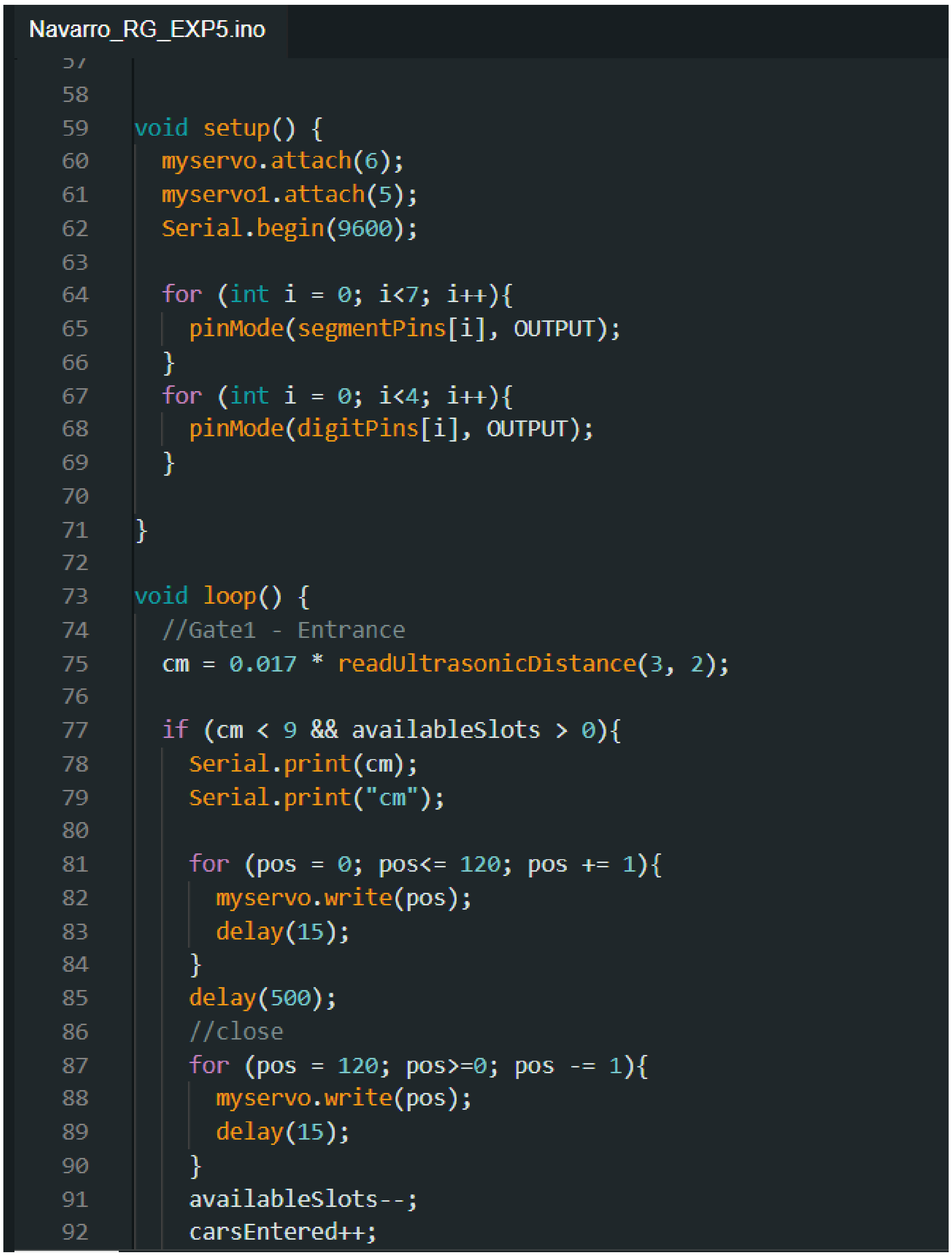
For the coding, I have only used one library which is the Servo.h library in order to control the servo motor. After initializing the library, I also initialized variables that I need in the program like the pos for the position angle, the cm as the range of the detection of the ultrasonic sensor, and many more as well as the segment and digit pins, and the digit patterns of the display.



```
#include <Servo.h>

Servo myservo;
Servo myservo1;


int pos = 0;
int cm = 0;

int availableSlots = 10;
int carsEntered = 0;

int segmentPins[7] = {A5, A4, A3, A2, A1, A0, 10};
int digitPins[4] = {4, 13, 12, 11};


byte digitPatterns[10] = {
  0b00111111,
  0b00000110,
  0b01011011,
  0b01001111,
  0b01100110,
  0b01101101,
  0b01111101,
  0b00000111,
  0b01111111,
  0b01101111,

};

long readUltrasonicDistance(int triggerPin, int echoPin){
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
```

Figure 3: *The insertion of the library and other variables needed in the program.*

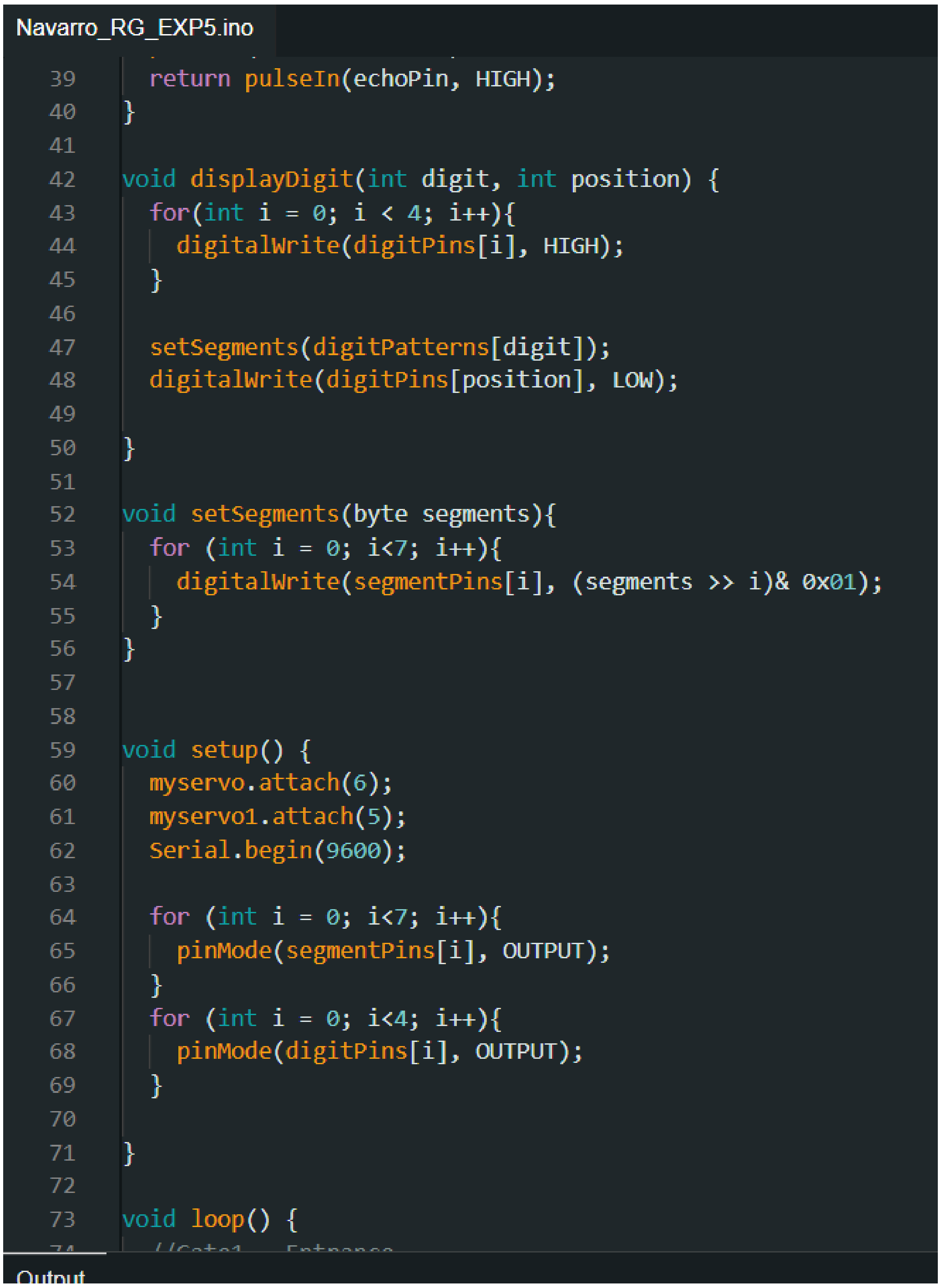
In this experiment, the code controls a parking system with an entrance and exit gate, as well as a seven-segment display to show the number of available parking slots and cars that have entered. The two servo motors, myservo and myservo1, are used to open and close the gates. Ultrasonic sensors measure the distance to detect cars entering and exiting. If a car is detected within a certain range, the entrance gate opens, the available parking slots decrease, and the cars entered count increases. Similarly, when a car leaves, the exit gate opens, the available slots increase, and the cars entered count decreases.


Navarro_RG_EXP5.ino

```

void setup() {
  myservo.attach(6);
  myservo1.attach(5);
  Serial.begin(9600);

  for (int i = 0; i<7; i++){
    pinMode(segmentPins[i], OUTPUT);
  }
  for (int i = 0; i<4; i++){
    pinMode(digitPins[i], OUTPUT);
  }

}

void loop() {
  //Gate1 - Entrance
  cm = 0.017 * readUltrasonicDistance(3, 2);

  if (cm < 9 && availableSlots > 0){
    Serial.print(cm);
    Serial.print("cm");

    for (pos = 0; pos<= 120; pos += 1){
      myservo.write(pos);
      delay(15);
    }
    delay(500);
    //close
    for (pos = 120; pos>=0; pos -= 1){
      myservo.write(pos);
      delay(15);
    }
    availableSlots--;
    carsEntered++;
```

Figure 4: *The initiation of local variables, as well as the conditions to control the gates.*

The seven-segment display is controlled using segmentPins for the segments and digitPins for the four digits. The display shows the number of available slots and cars entered. A multiplexing technique is used in the multiplexDisplay() function, where each digit is refreshed quickly in a loop to prevent flickering. The displayDigit() and setSegments() functions work together to control which segments light up for each number. This ensures that the current state of the parking system is clearly displayed in real time.



```
  return pulseIn(echoPin, HIGH);
}

void displayDigit(int digit, int position) {
  for(int i = 0; i < 4; i++){
    digitalWrite(digitPins[i], HIGH);
  }

  setSegments(digitPatterns[digit]);
  digitalWrite(digitPins[position], LOW);

}

void setSegments(byte segments){
  for (int i = 0; i<7; i++){
    digitalWrite(segmentPins[i], (segments >> i)& 0x01);
  }
}


void setup() {
  myservo.attach(6);
  myservo1.attach(5);
  Serial.begin(9600);

  for (int i = 0; i<7; i++){
    pinMode(segmentPins[i], OUTPUT);
  }
  for (int i = 0; i<4; i++){
    pinMode(digitPins[i], OUTPUT);
  }

}

void loop() {
```

Figure 5: The configuration of the display.

CONCLUSION

The experiment successfully demonstrated the creation of a basic parking lot counter using two servo motors, ultrasonic sensors, and a 4-digit seven-segment display. The servo motors acted as gates that opened and closed based on car detection, which was achieved using the ultrasonic sensors. The system accurately tracked and displayed the number of available parking slots and cars inside the lot. This practical setup, including the use of toy cars, helped mimic a real-life parking system, making the concept easy to understand and visualize.

The coding part of the experiment ensured that the system worked efficiently by controlling the gates and updating the display in real-time. The seven-segment display used multiplexing to show the current parking status without flickering, making it easy to track available slots and cars entered. The experiment demonstrated how basic components can be integrated into a functional parking lot system, providing a clearer understanding of the hardware and software requirements involved in such a experiment.