Programming Embedded Systems
CS4111
Fall 2023

Instructor: Dr. Zain Balfaqih

Students:
Hanin Alzaher - S20106145
Reem Alsharabi - S20106353
Razan Almahdi - S20106649
Hager Abas - S20106427

Smart Parking System

1. Introduction

In recent years, the rapid growth of urbanization and the increasing number of vehicles on the road have led to a significant challenge in finding available parking spaces efficiently. Traditional parking systems often result in time-consuming and frustrating experiences for drivers, causing congestion and unnecessary fuel consumption. Additionally, the ever-present risk of fire accidents in parking areas further highlights the need for enhanced safety measures.

To address these challenges, we present our project, the "Smart Parking System." Our aim is to develop an advanced embedded system that utilizes Arduino microcontrollers and sensors to provide an intelligent parking solution with integrated fire detection and prevention capabilities.

2. Hardware

The hardware components used in this project are as follows:

A. 1 breadboard
B. 1 smoke/gas sensor
C. 2 ultrasonic distance sensors
D. 1 Push button
E. 1 potentiometer
F. 1 Piezo
G. LCD display screen

The breadboard: The breadboard serves as a platform for prototyping and connecting various electronic components. It provides a convenient way to create and modify circuits without the need for soldering. In this project, the breadboard acts as the central hub for connecting and organizing the components.

- **Input Devices:**
  The smoke/gas sensor: The smoke/gas sensor is designed to detect the presence of smoke or fire in the surroundings. It uses optical or ionization techniques to sense smoke particles in the air. In the Smart Parking System, the smoke sensor is utilized to detect the presence of smoke or fire within the parking area. When smoke or fire is detected, it triggers the fire safety protocols, including activating alarms and fire suppression systems. It is also used to detect the presence of hazardous gasses in the environment. It can identify various types of gasses, such as carbon monoxide (CO), methane (CH4), or propane (C3H8). In the Smart Parking System with Fire Safety Integration, the gas sensor is utilized to detect

the presence of potentially dangerous gasses, such as leaked fuel or combustible gasses within the parking area. If a hazardous gas is detected, appropriate actions can be taken, such as activating ventilation systems or triggering alarms.

The ultrasonic distance sensor: Ultrasonic distance sensors use ultrasonic waves to measure the distance between the sensor and an object. These sensors emit ultrasonic pulses and measure the time it takes for the pulses to bounce back after hitting an object. By analyzing the time delay, the sensor can calculate the distance to the object. In the parking system, two ultrasonic distance sensors are employed to detect the availability of parking spaces. They measure the distance between the sensor and the vehicles in the parking area, allowing the system to determine whether a parking space is occupied or vacant.

The potentiometer: potentiometers, also known as variable resistors, are adjustable electrical components used to control the voltage or current in a circuit. They have a knob or slider that allows the user to vary the resistance. In this project, potentiometers are used as input devices to adjust certain parameters, such as sensitivity levels or thresholds. For example, a potentiometer can be used to adjust the sensitivity of the smoke sensor or the distance threshold for parking space detection.

The Piezo: A piezo transducer or buzzer is an electronic component that converts electrical energy into sound waves. In the Smart Parking System, the piezo is used as an audio output device to generate audible alarms or alerts. When a fire or hazardous gas is detected, the piezo can be activated to emit a loud sound, notifying people in the vicinity about the potential danger.

- **Output Devices:**
  LCD Display Screen: It is used to display information or data from the Arduino. It can show text, numbers, symbols, or even graphical elements on the screen. The Arduino sends signals to control the display and update its content based on the desired output. In this project, we use the LCD display screen to display the welcoming message, distances detected by the sensors, number of free parking slots, and emergency warning in the case of smoke/gas detection.

3. Software

3.1 Sketch:

The code used to program the sketch for the circuit is as follows:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
int Smoke = 0;
int r = 0;

#define t1 10
#define t2 9
#define t3 8

int distanceThreshold = 100;
bool alarmActive = false;
int buttonState = 0;
int oldButton = 0;

void setup() {
  pinMode(A0, INPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(11, INPUT);
}

void loop() {
  checkSmoke();
  checkDistance();
  handleButton();
}

void checkSmoke() {
  Smoke = analogRead(A0);
  Serial.println(analogRead(A0));

  if (Smoke >= 25 && oldButton != 1) {
    activateAlarm();
    displaySmokeInfo();
  } else {
    deactivateAlarm();
    displayGoodDay();
  }
```

```
  delay(10); // Delay a little bit to improve simulation performance
}

void activateAlarm() {
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  tone(13, 523); // play tone 60 (C5 = 523 Hz)
  alarmActive = true;
}

void deactivateAlarm() {
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  lcd.clear();
  noTone(13);
  alarmActive = false;
}

void handleButton() {
  buttonState = digitalRead(11);
  if (buttonState != oldButton && buttonState == HIGH) {
    if (alarmActive) {
      deactivateAlarm(); // Turn off the alarm when the button is pressed
      oldButton = buttonState;
    }
    delay(500); // Debouncing delay
  }else if(oldButton == buttonState && oldButton == HIGH){
    oldButton = 0;
    delay(500);
    }
}

void displaySmokeInfo() {
  lcd.setCursor(0, 0);
  lcd.print("Emergency exit is right to the elevator");
  delay(100);
  lcd.setCursor(0, 1);
  lcd.print("Fire and Rescue Dial 101 immediately");
  lcd.setCursor(1, 0);

  for (r = 0; r < 36; r++) {
    lcd.scrollDisplayLeft();
    delay(100);
  }
```

```
}

void displayGoodDay() {
  lcd.clear();
  lcd.setCursor(6, 0);
  lcd.print("GOOD");
  lcd.setCursor(6, 1);
  lcd.print("DAY:)");
  delay(1000);
}
// distance

long readDistance(int triggerPin, int echoPin) {
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
}
void displayFreeSlots(int slots) {
  lcd.setCursor(0, 0);
  Serial.print(slots);
  Serial.println(" Slots Free");
  lcd.print(slots);
  lcd.print(" Slots Free");
  lcd.setCursor(0, 1);

  if (slots == 3) {
    lcd.print("Slot 1 2 3 Free");
  } else if (slots == 2) {
    if (readDistance(t1, t1) > distanceThreshold && readDistance(t2, t2) >
distanceThreshold) {
      lcd.print("Slot 1 & 2 Free");
    } else if (readDistance(t1, t1) > distanceThreshold && readDistance(t3, t3) >
distanceThreshold) {
      lcd.print("Slot 1 & 3 Free");
    } else {
      lcd.print("Slot 2 & 3 Free");
    }
  } else if (slots == 0) {
    Serial.print("0");
```

```
    Serial.println(" Slots Free");
    lcd.print("No Slot Free");
    lcd.setCursor(0, 1);
    lcd.print("Parking Full");
  } else if (slots == 1) {
    if (readDistance(t1, t1) > distanceThreshold) {
      lcd.print("Slot 1 is Free");
    } else if (readDistance(t2, t2) > distanceThreshold) {
      lcd.print("Slot 2 is Free");
    } else {
      lcd.print("Slot 3 is Free");
    }
  }

  delay(500);
}

void checkDistance() {
  float d1 = 0.01723 * readDistance(t1, t1);
  float d2 = 0.01723 * readDistance(t2, t2);
  float d3 = 0.01723 * readDistance(t3, t3);

  Serial.println("d1 = " + String(d1) + "cm");
  Serial.println("d2 = " + String(d2) + "cm");
  Serial.println("d3 = " + String(d3) + "cm");

  if (d1 > distanceThreshold && d2 > distanceThreshold && d3 > distanceThreshold) {
    displayFreeSlots(3);
  } else if ((d1 > distanceThreshold && d2 > distanceThreshold) || (d2 >
distanceThreshold && d3 > distanceThreshold) || (d3 > distanceThreshold && d1 >
distanceThreshold)) {
    displayFreeSlots(2);
  } else if (d1 < distanceThreshold && d2 < distanceThreshold && d3 <
distanceThreshold) {
    displayFreeSlots(0);
  } else if ((d1 < distanceThreshold && d2 < distanceThreshold) || (d2 <
distanceThreshold && d3 < distanceThreshold) || (d3 < distanceThreshold && d1 <
distanceThreshold)) {
    displayFreeSlots(1);
  }
  delay(100);
}
```

3.2 Code explanation

1. **The LiquidCrystal Library:**

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

The code begins by including the "LiquidCrystal" library, which is used for controlling the LCD display. The LiquidCrystal object lcd is then initialized with the pins to which the LCD is connected (RS, E, D4, D5, D6, D7).

2. **Variable and Constant Declarations:**

```
int Smoke = 0;
int r = 0;

#define t1 10
#define t2 9
#define t3 8

int distanceThreshold = 100;
bool alarmActive = false;
int buttonState = 0;
int oldButton = 0;

```

This section declares variables and constants that will be used in the code. It includes variables for smoke detection (Smoke), a loop counter (r), pin assignments for ultrasonic sensors (t1, t2, t3), a distance threshold for parking space detection (distanceThreshold), a flag to indicate if the alarm is active (alarmActive), and variables to handle a button press (buttonState and oldButton).

3. **Setup Functions:**

```
void setup() {
  pinMode(A0, INPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(11, INPUT);
}
```

The setup() function is called once when the Arduino board starts. In this section, pin modes are set for the analog input pin A0, digital pins 13, 12, and 11. The Serial communication is initiated with a baud rate of 9600. The LCD display is initialized with the dimensions of 16 columns and 2 rows.

4. **Main loop:**

```
void loop() {
  checkSmoke();
  checkDistance();
  handleButton();
}

```

The loop() function is the main program loop that runs continuously. It calls three functions in sequence: checkSmoke(), checkDistance(), and handleButton().

5. **Smoke Detection:**

```
33 void checkSmoke() {
34   Smoke = analogRead(A0);
35   Serial.println(analogRead(A0));
36
37   if (Smoke >= 25 && oldButton != 1) {
38     activateAlarm();
39     displaySmokeInfo();
40   } else {
41     deactivateAlarm();
42     displayGoodDay();
43   }
44
45   delay(10); // Delay a little bit to improve
46 }
47
```

The checkSmoke() function reads the analog value from the smoke sensor connected to pin A0. If the smoke level is above or equal to 25 and the button is not already pressed (oldButton != 1), it calls the activateAlarm() function to activate the alarm and displays smoke-related information on the LCD using the displaySmokeInfo() function. If the smoke level is below 25 or the button is pressed, it calls the deactivateAlarm() function to deactivate the alarm and displays a "Good Day" message on the LCD using the displayGoodDay() function.

6. **Alarm Activation and Deactivation:**

```
48 void activateAlarm() {
49   digitalWrite(13, HIGH);
50   digitalWrite(12, LOW);
51   tone(13, 523); // play tone 60 (C5 = 523 Hz)
52   alarmActive = true;
53 }
54
55 void deactivateAlarm() {
56   digitalWrite(13, LOW);
57   digitalWrite(12, HIGH);
58   lcd.clear();
59   noTone(13);
60   alarmActive = false;
61 }
```

These functions control the alarm activation and deactivation. activateAlarm() turns on the LED on pin 13, turns off the LED on pin 12, and plays a tone at 523 Hz using the tone() function. It sets the alarmActive flag to true. deactivateAlarm() turns off the LED on pin 13, turns on the LED on pin 12, clears the LCD display, and stops the tone using noTone(). It sets the alarmActive flag to false.

7. **Button Handling:**

```
void handleButton() {
  buttonState = digitalRead(11);
  if (buttonState != oldButton && buttonState == HIGH) {
    if (alarmActive) {
      deactivateAlarm(); // Turn off the alarm when the button is pressed
      oldButton = buttonState;
    }
    delay(500); // Debouncing delay
  }else if(oldButton == buttonState && oldButton == HIGH){
    oldButton = 0;
    delay(500);
    }
}
```

The handleButton() function reads the state of the button connected to pin 11 using digitalRead(). If the button state has changed from the previous state and it is currently pressed (buttonState == HIGH), it checks if the alarm is active. If the alarm is active, it calls the deactivateAlarm() function to turn off the alarm and update the oldButton variable. It also includes a debouncing delay of 500 milliseconds to prevent multiple readings due to button bounce. Additionally, if the button is continuously held down (oldButton == buttonState == HIGH), it resets the oldButton variable to 0 after a delay of 500 milliseconds.

8. Display Functions:

```
void displaySmokeInfo() {
  lcd.setCursor(0, 0);
  lcd.print("Emergency exit is right to the elevator");
  delay(100);
  lcd.setCursor(0, 1);
  lcd.print("Fire and Rescue Dial 101 immediately");
  lcd.setCursor(1, 0);

  for (r = 0; r < 36; r++) {
    lcd.scrollDisplayLeft();
    delay(100);
  }
}

void displayGoodDay() {
  lcd.clear();
  lcd.setCursor(6, 0);
  lcd.print("GOOD");
  lcd.setCursor(6, 1);
  lcd.print("DAY:)");
  delay(1000);
}
```

These functions are responsible for displaying information on the LCD. displaySmokeInfo() prints a message indicating the emergency exit location and a notification to dial 101 in case of fire. It also includes a scrolling animation by using the lcd.scrollDisplayLeft() function. displayGoodDay() displays a "Good Day" message on the LCD for 1 second.

9. Distance Measurement:

```
long readDistance(int triggerPin, int echoPin) {
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
}
void displayFreeSlots(int slots) {
  lcd.setCursor(0, 0);
  Serial.print(slots);
  Serial.println(" Slots Free");
  lcd.print(slots);
  lcd.print(" Slots Free");
  lcd.setCursor(0, 1);

  if (slots == 3) {
    lcd.print("Slot 1 2 3 Free");
  } else if (slots == 2) {
    if (readDistance(t1, t1) > distanceThreshold && readDistance(t2, t2) > distanceThreshold) {
      lcd.print("Slot 1 & 2 Free");
    } else if (readDistance(t1, t1) > distanceThreshold && readDistance(t3, t3) > distanceThreshold) {
      lcd.print("Slot 1 & 3 Free");
    } else {
      lcd.print("Slot 2 & 3 Free");
    }
  } else if (slots == 0) {
    Serial.print("0");
    Serial.println(" Slots Free");
    lcd.print("No Slot Free");
    lcd.setCursor(0, 1);
    lcd.print("Parking Full");
```

The readDistance() function is responsible for measuring the distance using an ultrasonic sensor. It takes two parameters: triggerPin (the pin connected to the trigger pin of the ultrasonic sensor) and echoPin (the pin connected to the echo pin of the ultrasonic sensor). Inside the function, the trigger pin is set as an output and is initially set to low. Then, a short delay of 2 microseconds is introduced. Next, a high pulse of 10 microseconds is sent to the trigger pin, which triggers the ultrasonic sensor to send out an ultrasonic wave. After that, the trigger pin is set back to low. Finally, the echo pin is set as an input, and the function returns the duration of the pulse received by the echo pin using the pulseIn() function.

10. Parking Space Detection:

```
  } else if (slots == 1) {
    if (readDistance(t1, t1) > distanceThreshold) {
      lcd.print("Slot 1 is Free");
    } else if (readDistance(t2, t2) > distanceThreshold) {
      lcd.print("Slot 2 is Free");
    } else {
      lcd.print("Slot 3 is Free");
    }
  }

  delay(500);
}

void checkDistance() {
  float d1 = 0.01723 * readDistance(t1, t1);
  float d2 = 0.01723 * readDistance(t2, t2);
  float d3 = 0.01723 * readDistance(t3, t3);

  Serial.println("d1 = " + String(d1) + "cm");
  Serial.println("d2 = " + String(d2) + "cm");
  Serial.println("d3 = " + String(d3) + "cm");

  if (d1 > distanceThreshold && d2 > distanceThreshold && d3 > distanceThreshold) {
    displayFreeSlots(3);
  } else if ((d1 > distanceThreshold && d2 > distanceThreshold) || (d2 > distanceThreshold &&
    displayFreeSlots(2);
  } else if (d1 < distanceThreshold && d2 < distanceThreshold && d3 < distanceThreshold) {
    displayFreeSlots(0);
  } else if ((d1 < distanceThreshold && d2 < distanceThreshold) || (d2 < distanceThreshold &&
    displayFreeSlots(1);
  }

  delay(100);
}
```

The checkDistance() function is responsible for measuring distances using the ultrasonic sensors and displaying the number of free parking slots on the LCD. Inside the function, the distances are measured using the readDistance() function for three ultrasonic sensors connected to pins t1, t2, and t3. The measured distances (d1, d2, and d3) are then displayed on the serial monitor for debugging purposes.

The number of free parking slots is determined based on the measured distances. If all three distances are greater than the distanceThreshold (which is set to 100 in the code), it means that all parking slots are free. If that is the case, the displayFreeSlots() function is called with a parameter value of 3 to display a message indicating that all slots are free.

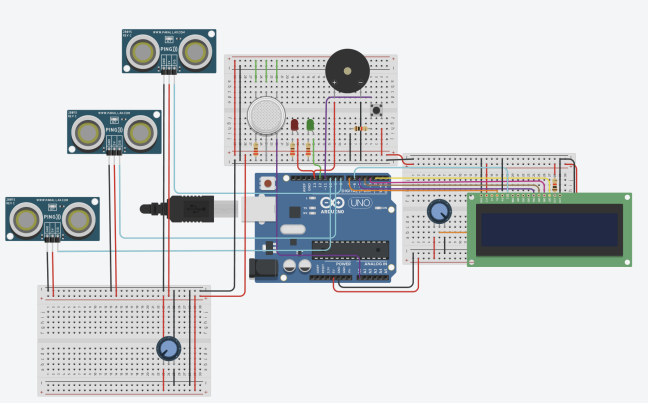
If two distances are greater than the distanceThreshold, it checks which two slots are free by comparing the distances of each slot. The appropriate message is then displayed using the displayFreeSlots() function with a parameter value of 2.

If all distances are below the distanceThreshold, it means that no slots are free, and the displayFreeSlots() function is called with a parameter value of 0 to display a "Parking Full" message.

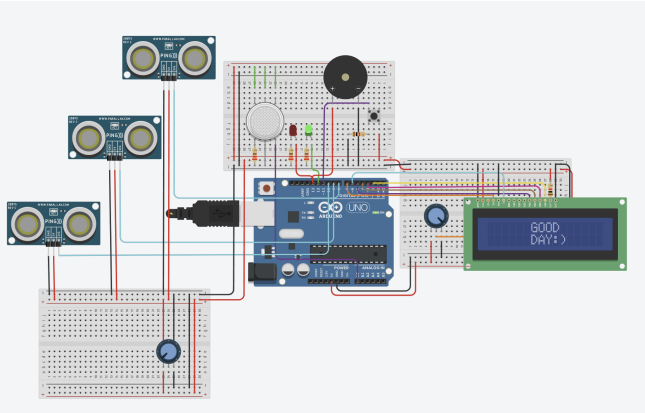
If only one distance is greater than the distanceThreshold, it determines which slot is free by comparing the distances of each slot. The displayFreeSlots() function is called with a parameter value of 1 and displays the appropriate message indicating the free slot.

The displayFreeSlots() function displays the number of free parking slots on the LCD based on the parameter value passed to it. It also includes conditional statements to display specific messages based on the number of free slots or the combination of free slots.

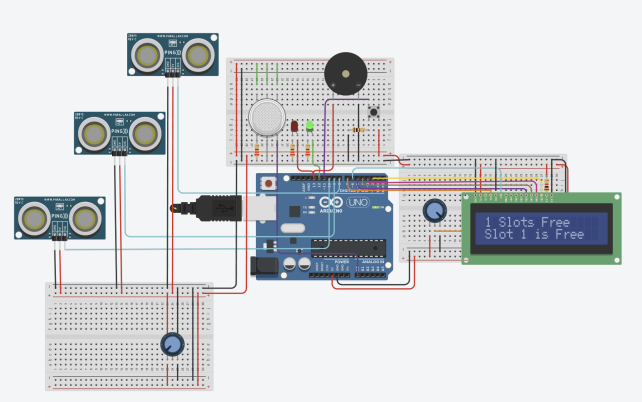
Initial circuit design (turned off):



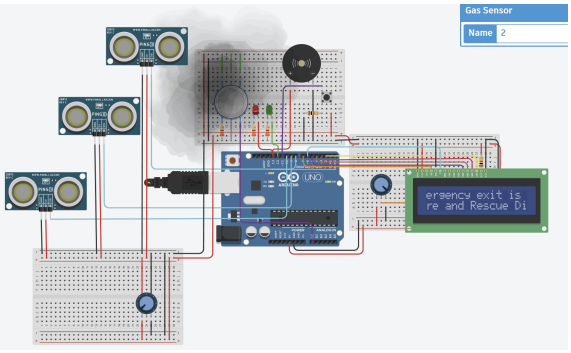
Initial circuit design (turned On):

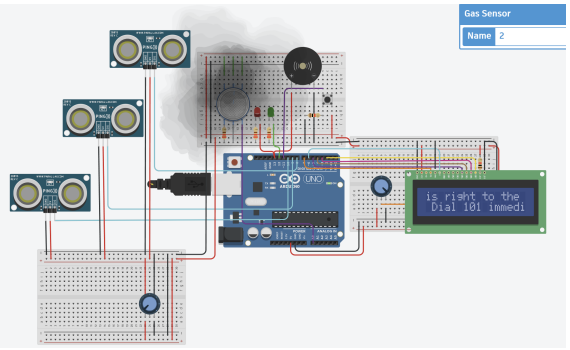


- If the smoke reading is below 25 or the oldButton state is 1, the alarm will be deactivated, the "GOOD DAY:)" message will be displayed on the LCD, and the alarm LED will turn off.



- The distances (d1, d2, d3) are calculated using the readDistance() function, which utilizes ultrasonic sensors connected to the trigger and echo pins (t1, t2, t3).
- The distances are converted to centimeters (cm) using a conversion factor of 0.01723.
- The calculated distances are then compared to the distanceThreshold value (set to 100 cm in the code) to determine the number of free parking slots.
- The number of free slots is displayed on the LCD, and the corresponding message is printed on the Serial Monitor.





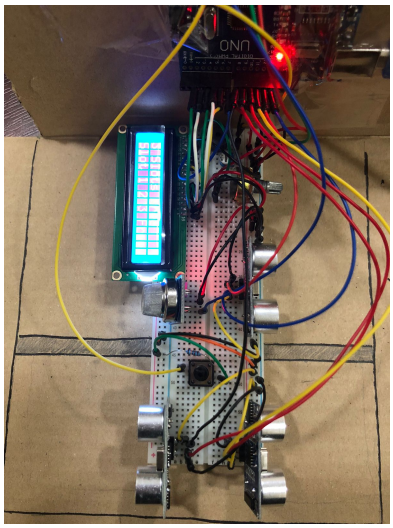
- If the smoke reading from analog pin A0 is greater than or equal to 25 and the oldButton state is not 1, the alarm will be activated, and the "Emergency exit is right to the elevator" message along with the "Fire and Rescue Dial 101 immediately" message will be displayed on the LCD. The alarm LED (pin 13) will turn on, and a tone will be played.

The serial monitor Output:

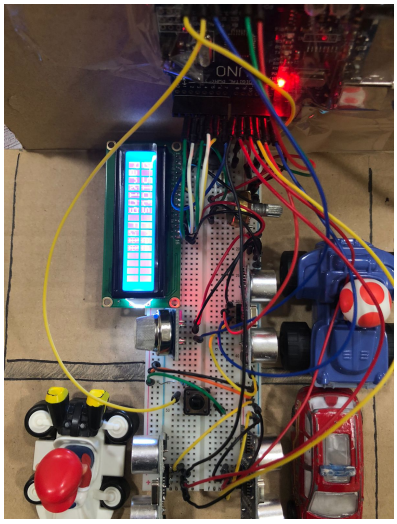
Serial Monitor

```
d1 = 47.97cm
d2 = 192.18cm
d3 = 95.97cm
1 Slots Free
92
d1 = 47.93cm
d2 = 192.25cm
d3 = 96.16cm
1 Slots Free
92
```

Physical Circuit:



The sensors do not have a target in range, and the smoke/gas sensor does not detect any smoke or gas, thus the LCD screen displays that 3 parking slots are empty.



The sensors are alerted with the presence of 3 cars in the respective parking slots, thus the screen displays that 0 slots are free.

4. Related Work

The concept of using distance measurement and parking space detection systems has been explored by various researchers and developers in the field of smart parking solutions. While it is challenging to provide an exhaustive list of all related works, this section discusses a few notable examples and highlights their similarities and differences to the described concept.

1. **Smart Parking Systems with Ultrasonic Sensors:**
    - Similarities: Several existing smart parking systems utilize ultrasonic sensors for distance measurement, similar to the concept described. These systems typically employ multiple sensors to monitor the availability of parking spaces.
    - Differences: The specific implementation details, such as the number and placement of sensors, may vary among different systems. Additionally, the algorithms used to detect parking space availability and the user interface designs can differ.
2. **Computer Vision-based Parking Space Detection:**
    - Similarities: Some advanced parking systems employ computer vision techniques for parking space detection. These systems use cameras to capture images or video footage and employ image processing algorithms to analyze the data and identify vacant parking spaces.

- Differences: In contrast to the described concept, which relies on ultrasonic sensors, computer vision-based systems offer the advantage of being able to detect parking space occupancy based on visual information. These systems often require more computational resources and may have different accuracy levels depending on the image processing algorithms employed.

3. **Internet of Things (IoT) Enabled Parking Systems:**
   - Similarities: IoT-based parking systems share the goal of providing real-time information about parking space availability. These systems often integrate various sensors, including ultrasonic sensors, to measure distance and detect parking space occupancy.
   - Differences: IoT-enabled parking systems typically rely on a network of interconnected devices, such as sensors, gateways, and a cloud-based platform, to collect and process data. They may incorporate additional features, such as mobile applications for users to access parking information, and advanced analytics for predicting parking availability.
4. **Mobile Applications for Parking Space Detection:**
   - Similarities: Some mobile applications provide parking space detection functionality. These apps use smartphone sensors, such as GPS and accelerometers, to detect parking events and determine parking space availability.
   - Differences: Unlike the described concept, which focuses on hardware-based distance measurement using ultrasonic sensors, mobile application-based systems rely on smartphone sensors and algorithms to infer parking space occupancy. These apps often provide additional features, such as navigation to available parking spaces and integration with parking payment systems.

In summary, while the concept of using distance measurement and parking space detection systems has been explored by others, each implementation may have its unique characteristics and approaches. The described concept utilizes ultrasonic sensors for distance measurement and provides a straightforward and cost-effective solution for monitoring parking space availability. By considering the differences and similarities with related works, developers can gain insights and inspiration for further improvements and innovations in the field of smart parking systems.

5. Conclusion

Our project aimed to develop a Smart Parking System, utilizing Arduino microcontrollers and sensors. By combining efficient parking space management with advanced fire detection and prevention capabilities, we envision a future where parking becomes hassle-free, safe, and environmentally friendly.

- **Member Tasks**

Report preparation: Razan.
Circuit design: Reem.
Physical circuit implementation: Hanin.
Presentation design: Hager.
Video demonstration: Hanin.

- **Challenges**

While the project showcases a functional proof-of-concept, several challenges were encountered during its development, in terms of accuracy, scalability, and incorporating advanced technologies. In terms of future work, the project could explore the integration of wireless communication capabilities to enable real-time data transmission and remote monitoring.

- **Future Work**

A feature that would enhance the project would be to enable users to access parking space availability information through mobile applications or web interfaces, enhancing user convenience and optimizing parking resource utilization. Moreover, machine learning algorithms can be used to optimize parking allocation and provide intelligent recommendations. While the project demonstrates a functional parking space detection system using distance measurement and ultrasonic sensors, there are opportunities for improvement and expansion. Overcoming the challenges related to accuracy, scalability, and integration with advanced technologies like computer vision and machine learning could enhance the system's performance and usability, making it a more robust and intelligent solution for optimizing parking management in diverse scenarios.