EasyPark – Smart Parking System based on Internet of Things technology

Abstract
In this era of widespread usage of automobiles, people often come across the problem of finding the right parking space in public, where their vehicle is safe and can be legally parked. Mostly, people tend to park their light motor vehicles on the roadsides or at the footpaths, thereby causing traffic jams and other inconveniences to the general public. As a result, the Governments of all prosperous countries across the world are rapidly developing multi-level parking spaces located close to popular spots such as markets, gardens, apartments etc. In these large multi-floored parking lots, drivers find themselves troubled when they have to find out just the right empty spot for themselves, where they can leave their vehicles safely. This is because, at the entry point, the driver of vehicle does not have any knowledge about the location of these free parking spaces within the parking lot. Also, most parking lots do not have any safety mechanism to handle emergency situations such as fire and smoke inside the parking lot, earthquakes etc. Va;et facility may not always be provided at parking lots too. This project aims to help the cause of making it easier for a driver to find just the right empty spot inside the premises of the parking lot, using Internet of Things (IoT) technology.

Keywords - *Parking System, Internet of Things, Sensors, Communication Protocol.*

I. INTRODUCTION

The Internet of Things (IoT) technology is the latest development in the era of Computing and Networking. It is used for interfacing of various devices such as Liquid Crystal Display (LCD), Servo Motor, Temperature Sensor, Light Emitting Diodes (LED) etc, with a programmable microcontroller such as Arduino UNO R3, so that the functioning of these devices can be controlled using a program that has been stored in the memory of the microcontroller. This technology forms the basis of this EasyPark Project.

A microcontroller is capable of having its pin slots set up to obtain input from an external device or provide output to it. Also, a microcontroller pin slot can work with three different signals namely, Analog, Pulse Width Modulated and Digital Signals. This project is an attempt at automating a parking lot with 8 parking slots and 2 floors. It also incorporates a safety emergency system for the safety of passengers and vehicles. The EasyPark System is a Smart Parking System that utilizes the IoT technology described above. It is simulated using TinkerCAD [1] software.

It uses 3 Arduino UNO R3 Microcontrollers, which are interfaced with each other using the Inter Integerated Circuits (I²C) Serial Communication Protocol and utilize the all the above mentioned IoT devices to realize the purpose of providing a convenient way of obtaining a parking slot in a parking while taking care of the safety and requirements of all the concerned passengers. I²C Protocol is a two wire serial communication protocol that establishes connection between a master and a slave arduino, so that master can provide specific commands to slave arduino to work appropriately, or slave can request a command from its master, as applicable. The connection thus established using this protocol is Half Duplex. Communication between 2 Arduino's can be established by connecting their Serial Clock (SCL) and Serial Data (SDA) pins with each other, or using two analog pins of both Arduino to act as SCL and SDA pins - to provide the bus connection for I²C Protocol. Also, both the Arduinos must share the same ground pin. Various IoT sensors have been used in the development of this project. Infrared (IR) sensors have been used to detect the presence of vehicles at the entrance and exit of parking lot as well as the slots in which they are to be parked; Gas Sensors and Temperature Sensors have been used in the safety mechanism offered by the system, to detect the presence of Fire and Smoke respectively, inside the parking.

II. METHODOLOGY ADOPTED

Three Arduino UNO R3 microcontrollers interfaced with each other using the I²C communication protocol as defined above, are used to handle and automate most aspects of this parking system.

Let's assume that an Arduino 1 called "Master" handles the safety sub-system and provides information about the same, to its two slaves – Arduino 2 and Arduino 3, referred to as "Slave 1" and "Slave 2" respectively, henceforth.

The entire Smart Parking System is divided into 4 subsystems that are handled simultanously, as explained below. Each Sub-System has a definite priority assigned to it.

Parking System [Priority 3]

When the vehicle enters into the parking lot, the state of parking lot is examined. If the parking lot has at least one empty spot and the parking lot is NOT in an emergency situation (such as smoke or fire), the entrance gate is opened up and the assigned slot is shown by the LCD at the entry point. Inside the parking lot, if a vehicle is being parked at a certain slot, then the parking of that vehicle is detected using a PIR sensor installed at the forefront of that slot and thus, the LED installed at outside wall of the slot glows up to show that this slot is currently occupied. As slot 3 and slot 4 lie on first floor of parking lot, there are trays present on the ground floor on which the vehicle to be parked at one of those slots is placed. As soon as the presence of this vehicle is detected on a tray, a pair of motors are switched ON to carry the tray to first floor. After a small delay, the direction of rotation of the motors is reversed and the tray is brought down back to the ground floor. At the exit gate, if a vehicle moves closer to the exit gate sensor, it's position is detected and parking operator has to enter the vehiclenumber. Based on duration of parking of this vehicle, the bill amount is calculated and displayed on the LCD installed at the exit point. Valets can be called at entrance to park a vehicle inside parking lot or to retrieve one from it.

Safety System [Priority 1]
If any emergency situation such as: smoke (detected by gas sensor) or fire (detected by temperature sensor) is found to be present inside the parking lot, then:

(a) All the LEDs associated with the Slots are switched ON to increase visibility inside parking lot.
(b) Emergency Exit Gate is opened.
(c) Emergency Ramp is unfolded so that vehicles parked on first floor can be brought down manually.
(d) Parking Entry and Exit operations are suspended.
(e) "Parking Unavailable" message is shown by the Entry LCD (aka LCD1) while "Use Emergency Gate" message is shown by the Exit LCD (LCD2).
(f) Operation of Car Finder button and Help switch continues, as usual.

Help System [Priority 2]
It features the Car Finder switch which is used for searching for the slot assigned to a certain vehicle using its vehicle number, and the Help Switch which can be used to request assistance from Parking Staff and is especially useful in times of emergency situations.

Maintenance System [Priority 4]
It is used to revert the actions of Safety System that were taken due to detection of an emergency situation inside parking lot. The Parking Operator can automatically re-roll the emergency ramp, shut down the emergency exit and remove emergency messages from the entry and exit LCDs. The parking operator can also mark all parking slots as “available” or “unavailable”. The Parking Operator can also reset or disable the safety system if required.

Slave 1 is able to handle the entire parking system, and it is able to respond to the events reported to it by the Master Arduino or Slave 2, using the I²C bus.

Table 1 – Pin-Component Connection Map of Slave 1

Component (Quantity)	**Associated Pins**	**Purpose**
16 x 2 LCD (2)	(13,12,11, 10,9,6) and (13,4,11,10 11,9,6)	LCD1 installed at entrance shows slot assigned to vehicle at entry gate and LCD2 installed at exit shows the final bill amount when a vehicle is present at exit gate. Output Pin Configuration.
Servo Motors (2)	3 and 5	Used to open and close both entry and exit gates. Output Pin Configuration.
PushButton (1)	2	To find the slot assigned to a vehicle, that's parked inside the parking lot. Input Pin Configuration.
LED(2)	7 and 8	Each LED installed at every oarking slot shows whether parking slot is occupied or not. Output Pin Configuration.
PIR Sensor (4)	A0, A1. A2 and A3	To detect the presence of a vehicle at entry or exit gate, and at one of the two assigned parking slots. Input Pin Configuration.
I²C Bus	A4, A5 and GND	The SDA and SCL pins required for I²C bus connection with Master and Slave 2. Half Duplex.

For all the algorithms incorporated by the Slave 1, the following three data structures are used.

```
struct slot
{
    int vno, inTime; /*Vehicle Number and the Time of Allotment are the two parameters stored by slot's structure*/
};

slot s[8]; /*Since 8 slots are to be handled by this parking system*/
int reservedslots[8]; /*Currently reserved slots, unavailable for assignment.*/

int nov = 0; /*Number of vehicles currently inside parking lot*/

byte rxd = 0; /*If rxd = 1, then system is in unsafe state,
rxd = 0 implies system is safe; rxd = 2 implies safety system is switched off.*/
```

Following algorithm was used for slot assignment to a vehicle currently at the entrance or parking slot:

```
int assignSlot()
{
 //Choosing the slot which is closest to the entrance.
 for(int i = 0; i < 8; i++)
 {
  if(s[i].inTime==0)
  {
   return i;
  }
 }
 return -1; //If no empty slot is available, then return -1.
}
```

Time Complexity of above algorithm is O(n).

Following algorithm was used for searching for the slot assigned to a vehicle using its vehicle number.

```
int searchSlot(int vno)
{
 Serial.println(nov);
 if(nov==0)
 {
  return -1;
 }
 else
 {
 for(int i = 0; i < nov; i++)
 {
  if(s[i].vno == vno)
  {
   int slot = i + 1;
   return slot;
  }
 }
  return -1;
 }
}
```

Time Complexity of this algorithm is O(n) too, as maximum value of nov can be 8.

Algorithm used to develop the billing system is as given below.

```
int billingSystem(int vno)
{
  int spot = searchSlot(vno);  /*To find the assigned slot*/
  if(spot==-1)
return -1;  /*If this vehicle is not found inside parking lot, return with value -1*/
```

```
      else
      {
        int outTime = millis();  /*Arduino Library function to
obtain up time of system in milliseconds.*/
       int duration = outTime – inTime;
       int bill = 0;
       if(duration<=10)//Chosen criterion for determining bill
       bill = 20;
       else if((duration>10)&&(duration<=20))
       bill = 40;
       else if(duration>20)
       bill = 100;
       s[spot].vno = 0;
       s[spot].inTime = 0;
       }
   return bill;
}
```

Time Complexity of this algorithm is O(1).

To reserve a parking slot, following algorithm was developed:

```
int reserveSlot(bool r, int spot) /*if r = true, then slot is to be
reserved, otherwise, slot is made available again.*/
{  if(r)
   { if(reservedslots[spot]!=1)
       { s[spot].inTime = millis(); /*At this time, the slot has
been reserved.*/
        reservedslots[spot] = 1; /*To indicate that the slot has
been reserved.*/
        Return 1;
       }
      else
        Return -1;
    }
    else
    { { if(reservedslots[spot]!=0)
           { s[spot].inTime = millis(); /*At this time, the slot
has been made available.*/
            reservedslots[spot] = 0; /*To indicate that the slot
has been made available.*/
           Return 1;
          }
         else
           Return -1;
       }
     }
 }
```

Time Complexity of this algorithm is also O(1).

In the void setup() function, we set up the pins through which all components are connected to arduino, as input or output pins. Further, we switch off all the LEDs and close the gates of parking system.

We seek to connect Slave 1 to the I²C bus (half duplex link) and for that purpose, we have to add the following code. [2]

```
#include<Wire.h>
byte rxd = 0;
void setup()
{
      Wire.begin(1); /*Argument '1' refers to address of this
Slave, which is 1.*/
       Wire.onReceive(receiveEvent);  /*receiveEvent() is
invoked when a byte is received.*/
 }
void receiveEvent(int howMany)
{
    rxd = Wire.read();
}
```

Finally, the void loop() function contains the code that is to be re-iterated until power is switched off. Various tasks are fulfilled by it, as per their priorities, as depicted by Figure 1.

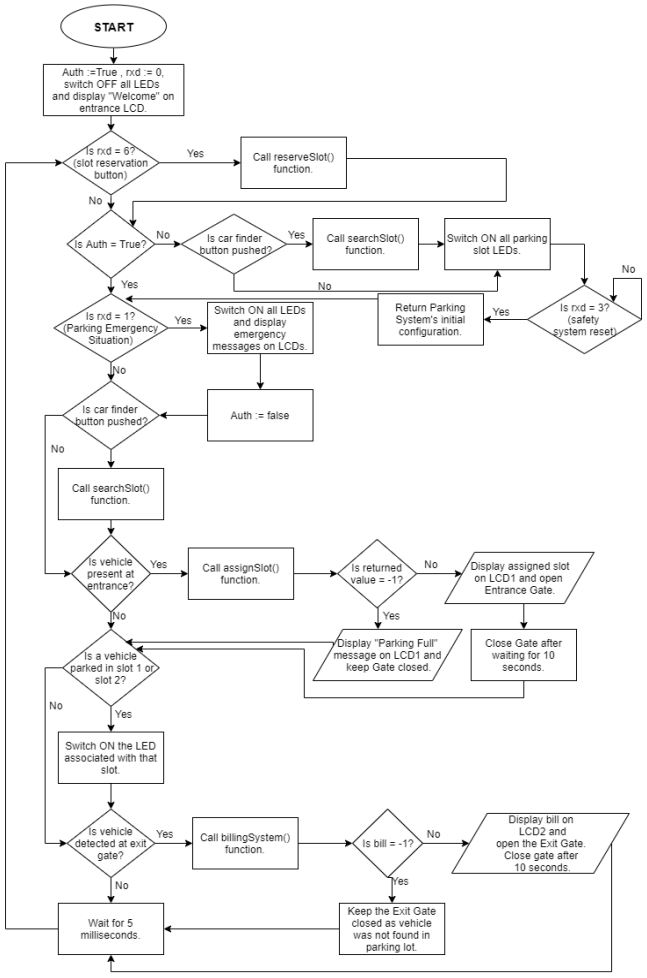


Figure 1 - Flowchart illustrating functionality of Slave 1.

Master is able to handle the entire safety system along with two additional slots on first floor, and it was able to transmit byte value 1 or 0 on I²C bus to indicate to its slaves whether the parking lot is currently in unsafe or safe state respectively.Table 2 shows configurations of pins of Master.

Algorithm for byte transmission to Slave 1 is described next.

```
#include<Wire.h>
byte sentbyte = 0;
void setup()
{
  Wire.begin();
}
void sendSafetyInformation()
{
   Wire.beginTransmission(1);
   Wire.write(sentbyte);
   Wire.endTransmission();
}
```

To find out the amount of smoke inside parking lot using gas sensor, following algorithm was used.

```
void detectSmoke()
{
  int value = map(analogRead(smokedetector), 300, 750, 0, 100);
  /*If excessive smoke is detected then, associated LED is switched ON, otherwise the LED is OFF.*/
 if(value>=80)
  {
   digitalWrite(smokeled, HIGH);
   safe = 1;
   Serial.println(safe);
  }
  else if(value<80)
  {
   digitalWrite(smokeled, LOW);
  }
 }
```

For checking whether an active fire is currently present inside the parking lot, the current temperature of the parking lot is used as a parameter, which could be detected by a temperature sensor, using the following algorithm.

```
void detectFire()
{
 int tmp = analogRead(firedetector);
 float voltage = (tmp * 5.0)/1024;
 float milliVolt = voltage * 1000;
 float tmpCel =  (milliVolt-500)/10 ;
 if(tmpCel>=70.0)
 {
 emergencygate.write(75); /*Open emergency gate, attached using emergencygate.attach(analog_pin) in void setup.*/
   tone(firealarm,22); /*Raise the fire alarm (piezo alarm).*/
   delay(50);
   noTone(firealarm);
   safe = 1;
 }
 else
 {
   emergencygate.write(5); //Keep emergency gate closed.
   delay(10);
 }
}
```

That completes the definition of the safety system installed in this project. Now, moving on to slot management in case of Master Arduino, the following algorithm has been developed.

```
void manageSlot()
{
 if(analogRead(slotn)>500) //slotn is Slot IR sensor pin.
 { /*turning ON the associated pair of motors*/
   digitalWrite(motorenablen, HIGH);
   delay(200);
   digitalWrite(motor_ip1n, HIGH);
   digitalWrite(motor_ip2n, LOW);
   delay(100);
   /*waiting until vehicle is placed into slot at first floor.*/
   digitalWrite(motor_ip1n, LOW);
   digitalWrite(motor_ip2n, LOW);
   delay(200);
   /*reversing direction of rotation of motors*/
   digitalWrite(motor_ip1n, LOW);
   digitalWrite(motor_ip2n, HIGH);
   delay(100);
   digitalWrite(motorenablen, LOW);
   /*switching ON the LED associated with parking slot.*/
   digitalWrite(ledslotn, HIGH);
   delay(400);
 }
 else if(analogRead(slotn)<=500)
 {
   digitalWrite(motorenablen, LOW);
   digitalWrite(motor_ip1n, LOW);
   digitalWrite(motor_ip2n, LOW);
   delay(1);
   digitalWrite(ledslotn, LOW);
 }
}
```

n can be replaced by 3 or 4, depending upon the slot which is to be managed.

Table 2 – Pin-Component Connection Map of Master

Component (Quantity)	Associated Pins	Purpose
PIR Sensor (2)	A3 and A2	Installed at every Parking Tray to detect presence of a vehicle parked upon them. Input Pin Configuration.
LED(2)	7,8 and 12	When a vehicle is successfully parked at either slot 3 or slot 4 then the slot-associated LED is switched on. Last LED glows if too much smoke is detected by sensor. Output Pin Configuration.
H-Bridge Motor Driver (2)	(1,6,5) and (4,11,10)	To switch on the pair of motors associated with every tray and control the direction of rotation of these motors. Output Pin Configuration.
Servo Motor (1)	9	For opening and closing the emergency exit gate, as required. Output Pin Configuration.
Slider Switch (1)	2	Acts as main switch for the Master, if it's turned off then the Master does not perform its work. Input Pin Configuration.
Gas Sensor (1)	A1	To detect the amount of smoke inside the parking lot. Input Pin Configuration.
Temperature Sensor (1)	A0	To check the temperature of the Parking Lot. Used in detecting active fires in the lot. Input Pin Configuration.
Piezo Alarm (1)	13	If fire is detected inside the Parking Lot then this alarm is raised as a Warning signal. Output Pin.
I²C Bus	A4 ,A5 and GND	For transferring data to Slave 1 and Slave 2. Half Duplex Connection is established.

Finally, the Slave 2 is added only to extend the number of Parking Slots to 8. Due to limitation of slots available with Slave 1, it also serves the purpose of contacting Slave 1 if its "Slot Reservation Button" is pressed by Parking Operator for slot reservation.

The slot management algorithm used in case of Slave 2 is as follows:

```
void manageSlot()
{
 if(analogRead(slotn)>500)
   {
    digitalWrite(ledn,HIGH);
    delay(1);
   }
   else
   {
    digitalWrite(ledn, LOW);
    }
}
```

In the above code, *n* can be replaced by 5,6,7 or 8 - whichever slot has to be managed.

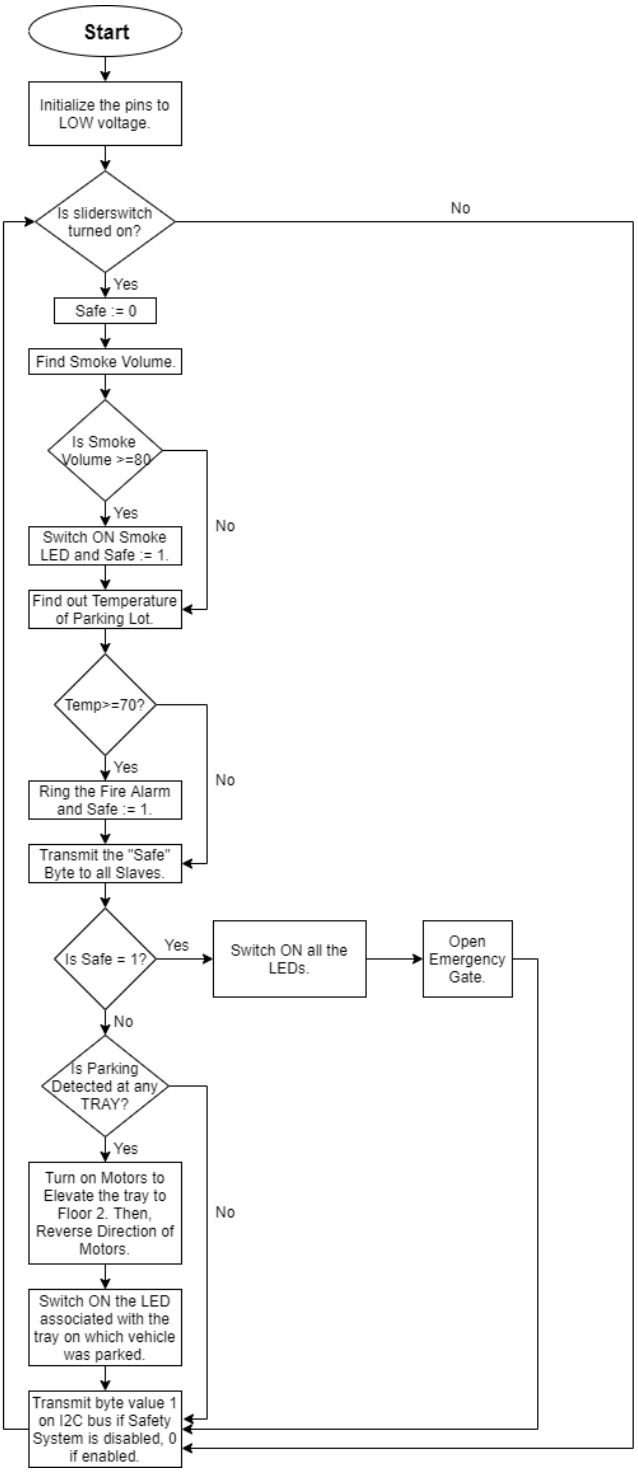


Figure 2 - Flowchart illustrating functionality of Master.

The event generated by the Slot Reservation Push Button can be handled using following code snippet:

```
int val = digitalRead(reserveslotbutton);
 if(val==0)
 {
 val = val + 6;
  Wire.beginTransmission(1); /*Informing Slave 1 that Slot Reservation Button was pressed.*/
 Wire.write(val);
 Wire.endTransmission();}
```

Note that, in order to use the I²C bus for communication with Slave 1, the Slave 2 must register itself on the I²C network using its own ID as 2. Also, the main switch of Master must be turned on, to establish communication with all the links.

Similarly, Help Button input is handled as follows.

```
int help = digitalRead(helpswitch);
if(help==0)
{
 Serial.println("Customer needs help!!!");
}
```

Valet System is implemented by Slave 2 as follows:

```
if(rxd==11) //Slave 1 asks slave 2 to ring valet buzzer.
  {
   while(digitalRead(valetbutton)!=0)
       {
        tone(valetbuzzer,55);
        delay(20);
        noTone(valetbuzzer);
       }
   Wire.beginTransmission(1);
   Wire.write(12); //Slave 2 tells Slave 1 - valet has come.
   Wire.endTransmission();
  }
```

Table 3 – Pin Component Connection Map for Slave 2

Component (Quantity)	Associated Pins	Purpose
LED (5)	5,6,7,8 and 9	Four of these are associated with Parking Slots and show its current status. Last LED shows whether Safety System of Master is currently enabled (glows) or disabled. Output Pins.
H-Bridge Motor Driver (1)	11,12 and 13	To roll or unfold the emergency ramp using a DC motor, whose direction is controlled by this driver. Output Pins.
PIR Sensor (4)	A0, A1, A2 and A3	Detect the presence of a vehicle inside the associated parking slot. Input Pins.
PushButton (3)	2, 3 and 4	First one displays Help Message on Serial Monitor and Second one sends Slot Reservation request to Slave 1. Also used by valet to answer a call, upon arrival at entry or exit gate. Input Pins.
Piezo (1)	10	Used as a Valet Calling Device (Buzzer). Output Pin.
I²C Bus	A4, A5 and GND	To connect Slave 2 onto the common I²C bus connected with both the Master and the Slave 1. Half Duplex Links.

Table 4 – Description of all Bytes transferred using I2C Bus

Byte Value	Sender	Receiver(s)	Inference
0	Master	Slave 1, Slave 2	Parking is in SAFE state.
1	Master	Slave 1, Slave 2	Parking is in UNSAFE state.
3	Master	Slave 1, Slave 2	Safety System has been RESET.
6	Slave 2	Slave 1	Parking Slot Reservation is requested.
11	Slave 1	Slave 2	Ring the Valet's buzzer.
12	Slave 2	Slave 1	Valet has answered the call to buzzer.

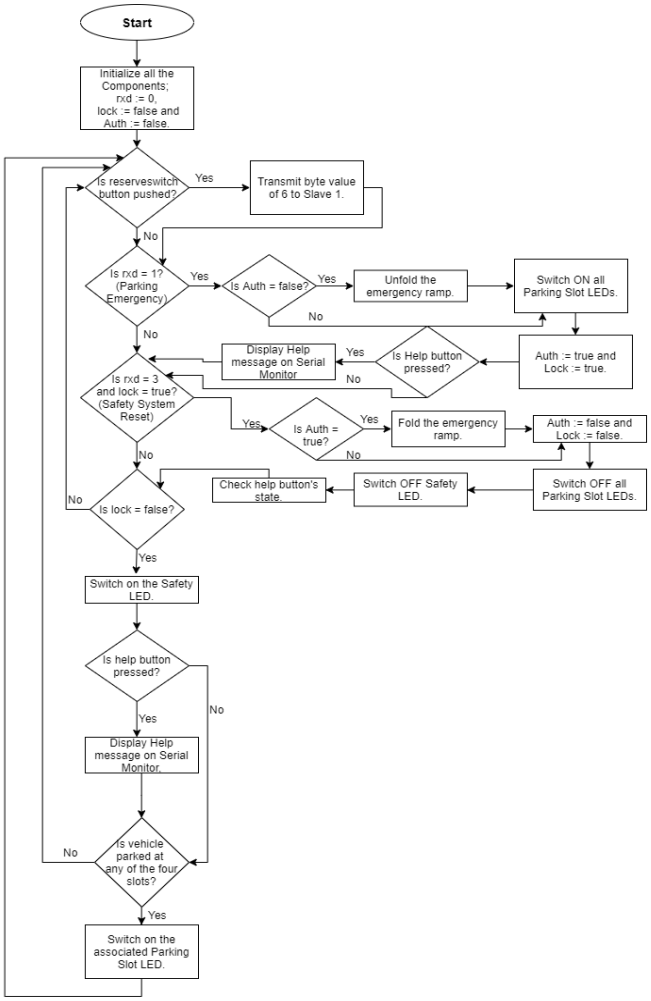


Figure 3 - Flowchart illustrating functionality of Slave 2.

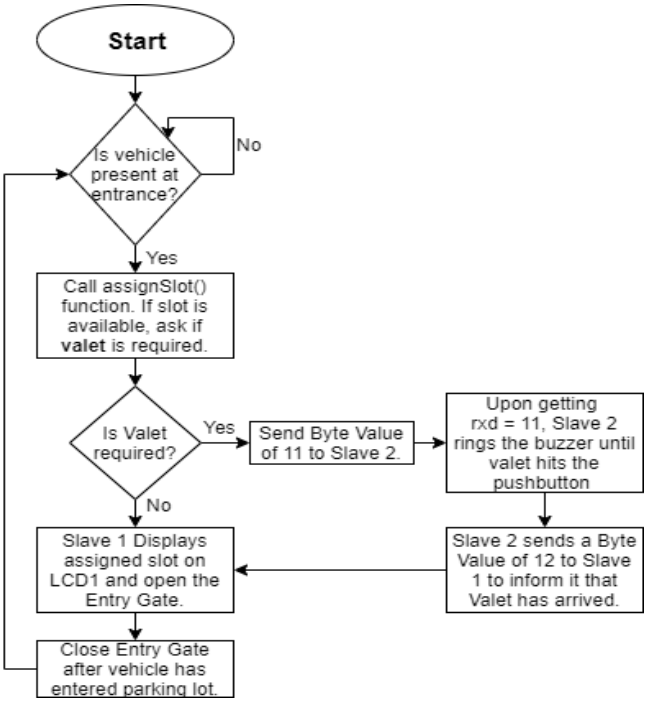


Figure 4 - Algorithm to call a valet at entrance gate using I2C Protocol.

III. RESULTS AND DISCUSSION

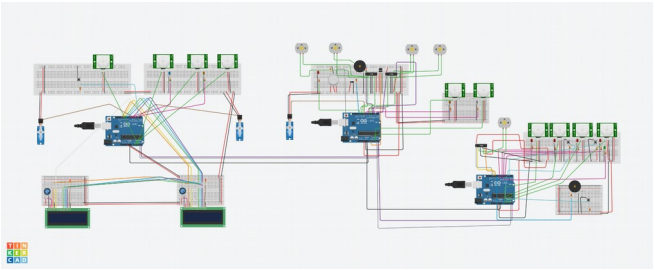
The above project seeks to provide an effective approach towards management of a small parking lot with 8 parking slots.

The EasyPark - Smart Parking System project was originally validated against the following requirements:

1. Provision of at least 8 parking slots in the parking lot, distributed between two floors.
2. Automation of the Entry Gate as well as the Exit Gate of the parking lot.
3. Incorporation of a Billing System, that accepts duration of parking of a vehicle inside lot as a parameter and generates the bill based on any chosen criterion.
4. Allocation a free parking slot, which is closest to the Entrance of the lot – at any given time, to the vehicle drivers at the entrance.
5. Installation of LEDs on the outside wall of each parking slot to indicate whether a vehicle is parked in that slot or not.
6. Presence of a Login system for the parking operator to authorize him/her before the above mentioned functionality of the lot can be utilized.
7. Provision of “Help” switches at every parking spot, so that any handicapped people can be easily taken care of by the parking staff.
8. Provision of a safety system featuring smoke detection, fire alarms, emergency gate and emergency ramp.
9. Search for a particular vehicle using its vehicle number inside the list of currently parked vehicles, to obtain its slot number.
10. Usage of an LCD at Entrance to print the assigned slot number and an LCD at the Exit to display the bill amount.
11. Provision of Valets for convenient car parking.

Number of slots can be increased by interfacing more arduino microcontrollers with the master, and the safety system can be expanded to include more temperature and gas sensors.

The EasyPark Smart Parking System was originally simulated on the TinkerCAD software [1], and the simulated version of it is depicted by Figure 5.



Th e Arduino in the middle is the Master and Arduino's on its left and right are Slave 1 and Slave 2 respectively.

Figure 5 - Hardware Assembly of EasyPark - Smart Parking System.

IV. REFERENCES

[1] The circuit design and simulation of the same could be made possible by using the TinkerCAD simulator available on the website: www.tinkercad.com.
[2] Properties of all the components used in this project were referenced from the official website of Arduino community, www.arduino.cc.