COMPUTATIONAL METHODS AND MODELLING FOR ENGINEERING APPLICATIONS (GENG-8030)

GROUP NUMBER - 1

FINAL PROJECT REPORT

SMART VEHICLE PARKING MANAGEMENT USING MATLAB

SUBMITTED BY:

CANICE JERRY-AKPAN-105115621

ADITYA SUBRAMANIAN-110062220

SUBMITTED TO:

PROFESSOR DR. YASSER M. ALGINAHI

DATE: 25 JULY 2022

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The smart parking management system is an efficient system that automatically calculates the space availability and manages the parking process without human intervention. An Arduino is used as the core controller of this parking system; instructions are given to hardware levers to grant or deny access to the parking space, and further assistance is given to the user by the LCD and LEDs.

In any confined space, there is space only for a finite number of cars. This value is defined and fed into the system. The system keeps a record of the number of cars inside the parking space and appropriately decides whether to allow more vehicles into the parking space. This data is transparently available to the user on the LCD. Upon the parking space being fully consumed, a prompt is provided to let the user know.

To enter or exit the premise, a user button is provided for each action; upon interaction with these buttons, appropriate LEDs guide the user to enter or exit the premise.

In a world with an increasing automobile presence, it is important to have smart systems to save time and effort. Thus, a smart parking facility serving all use cases and minimizing human intervention is proposed and put in place.

1.1 PROJECT OBJECTIVE

The project employs the Arduino board with the AT328 as the controller. LEDs, LCD, servo motors, and other peripherals are interfaced with the controller to achieve the smart project system.

Through this embedded system, an effective parking solution can be obtained. MATLAB and its compiler are employed to flash the code onto the controller.

The system keeps count of the available parking space and notifies the user of the availability. In addition to keeping count, the lever to let cars in or out is to be controlled by a servo motor which

is instructed by the controller upon satisfying certain conditions. Additionally, LEDs are deployed to safely enable entry or exit to the parking premises.

1.2 PROJECT OUTCOME

The outcome can be realized by observing the hardware outputs corresponding to the user inputs.

2. HARDWARE AND SOFTWARE USED

The following section gives an overview of the hardware and software packages used to implement the system.

2.1.HARDWARE

2.1.1. ARDUINO UNO

The Arduino board is powered by the Atmega 328P controller. It is an 8bit single-core controller with 32KB of flash and 2KB of RAM. It has 23 GPIOs capable of operating small-medium applications.

2.1.2. LIQUID CRYSTAL DISPLAY (LCD)

The system is interfaced with a 16x2 LCD. The 16 corresponds to the operational columns and 2 corresponds to the number of rows.

2.1.3. MISCELLANEOUS COMPONENTS

Miscellaneous components include servo motors for the parking barriers, user buttons, ultrasonic sensors to detect obstacles, and LED indicators, along with wires, potentiometers, and power sources.

2.2.SOFTWARE

MATLAB, its build, and compile system are used to program the controller. Prints and other debug information can be viewed on the MATLAB terminal. Additional libraries are used to interface MATLAB and the Arduino.

2.3.HARDWARE CONNECTION DIAGRAM

The following section demonstrates the hardware architecture of the project. This is followed by an explanation of the architecture.

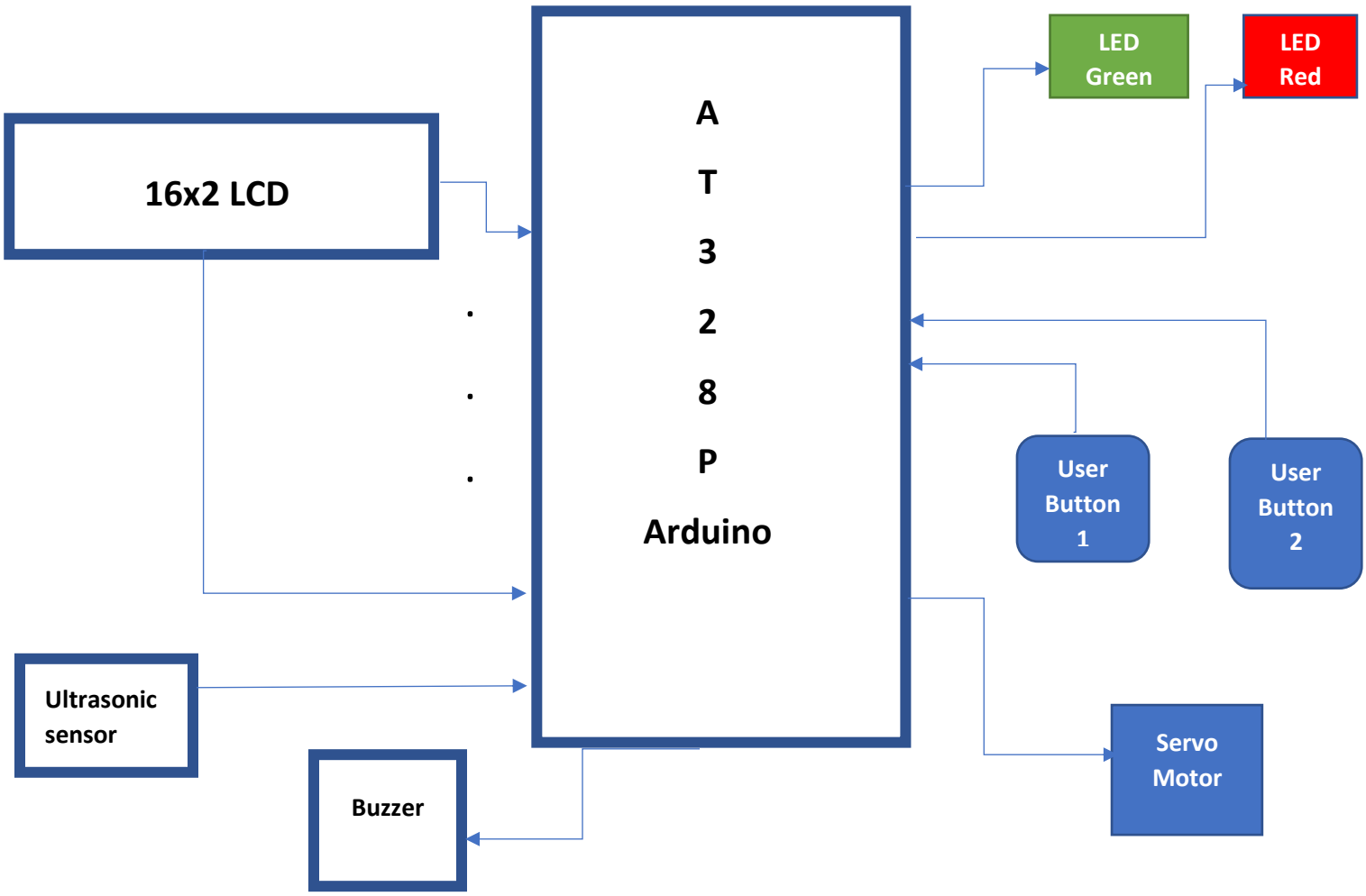

Figure 1 - Hardware blueprint for the smart vehicle parking system

The Arduino is the master controller of the project. There are mainly 6 modules connected to the Arduino, namely: LCD, User Buttons, LED, Ultrasonic Sensor, Buzzer, and Servo Motors. The LCD is to be run in 8-bit mode to improve speed (sends 8 bits of data to the LCD driver at once) [1], and hence 8 lines of the LCD are connected to the Arduino. Two pins namely trigger, and echo pins are used to operate the ultrasonic sensor which serves as input to perform certain functionality. Two pins are dedicatedly used to collect user data via push buttons This serves as input to the rest of the modules. Four pins are dedicated output pins to mainly the LEDs, buzzer, and servo motor.

2.4.SOFTWARE ARCHITECTURE

The following section describes the software architecture of the system. A brief run-through of the system startup is also depicted within the architecture to better understand how user code gets control of the processor. Appropriate servo motor, ultrasonic, and LCD libraries are invoked to use the hardware using APIs provided by these libraries [1] [2]. In addition, data gathered from the following system is sent to the cloud for data analytics. Appropriate web APIs and libraries are included in Matlab to use this feature.

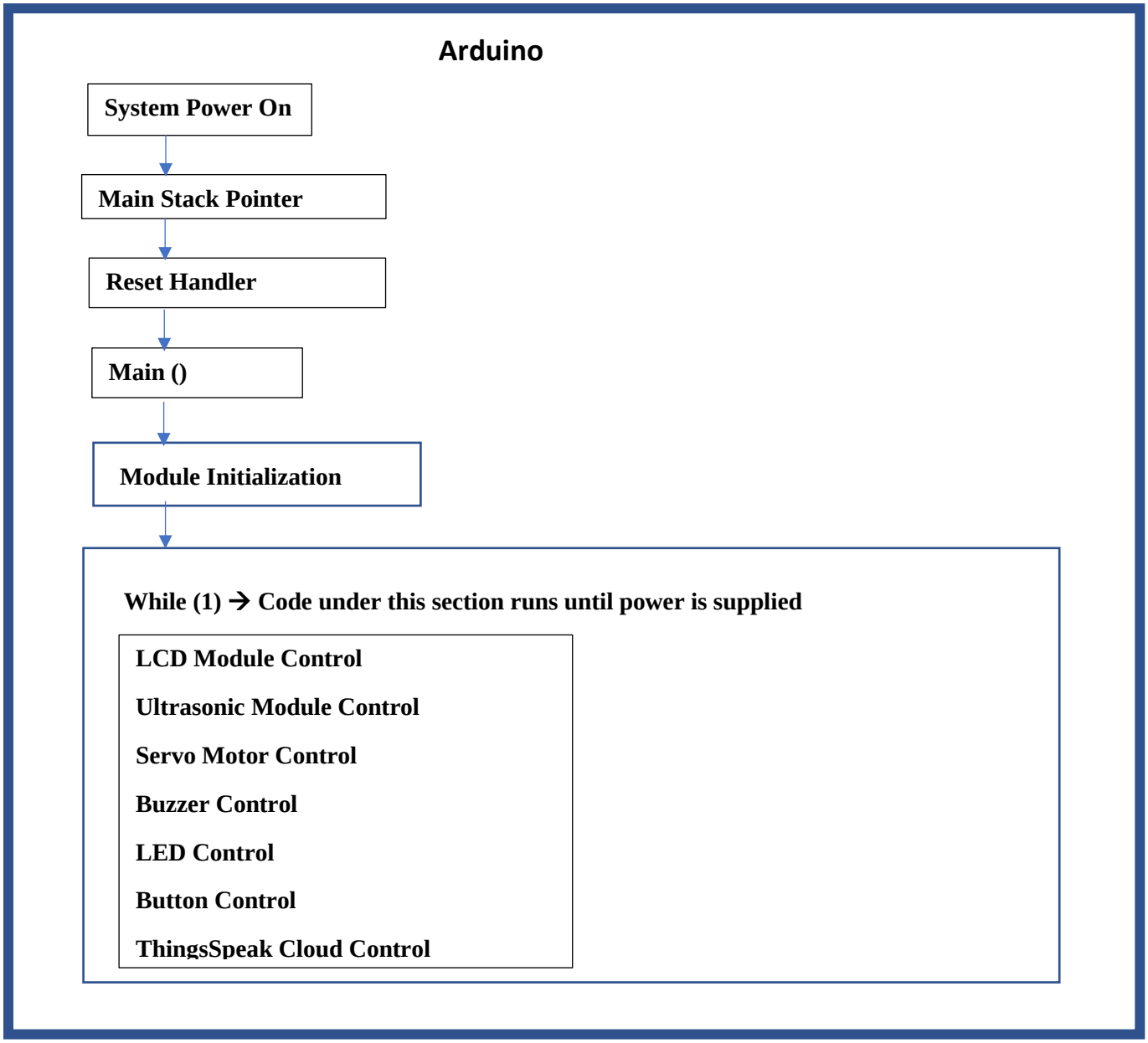

Figure 2 – Software blueprint for the smart parking management system.

3. METHODOLOGY

The following section shows the logical flow of the control system.

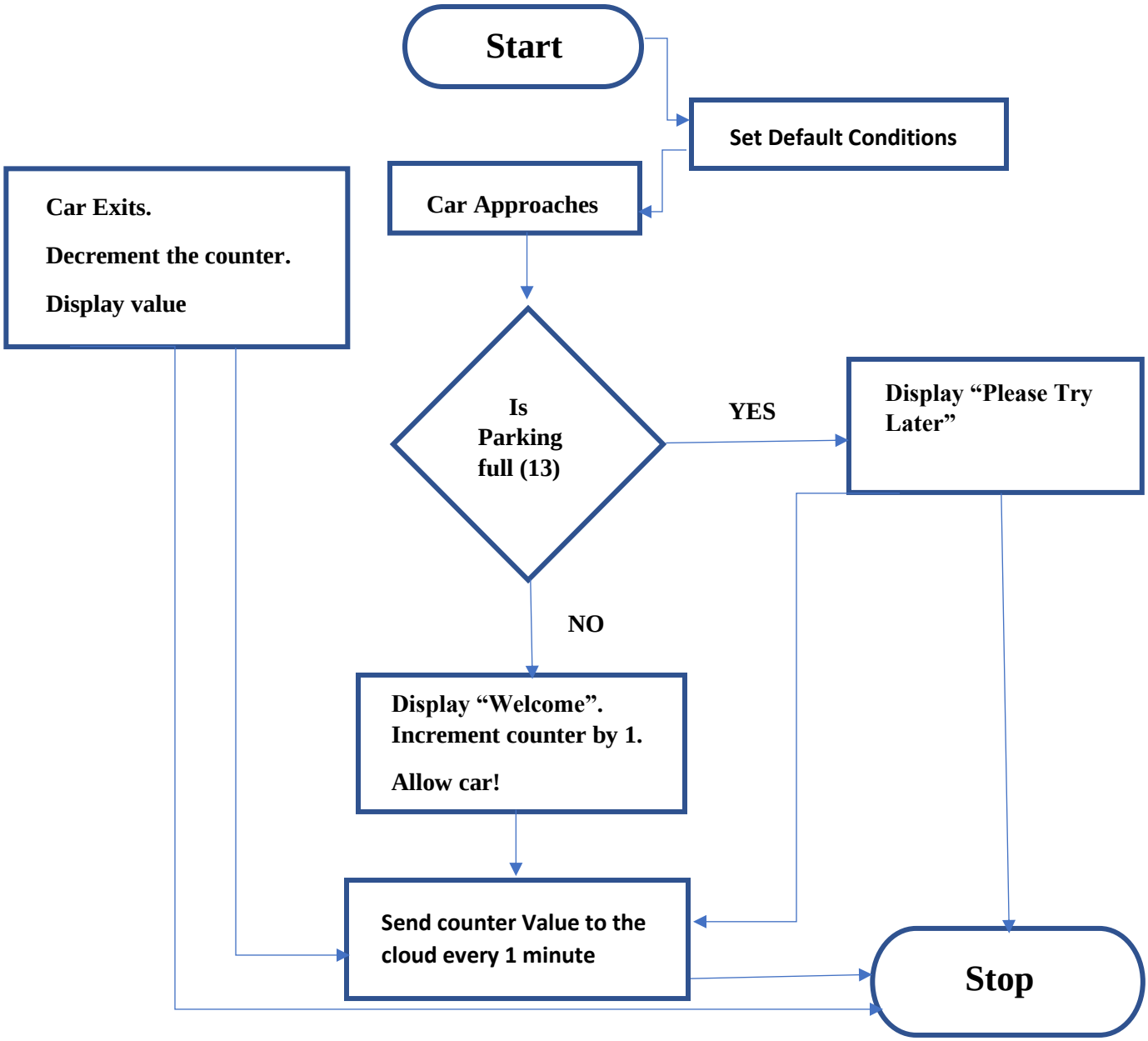

Figure 3 - Logical flow of the control system

4. PRELIMINARY PSEUDOCODE

The following section contains the preliminary pseudocode for the system.

```
/*****************************************************************************
*****                      SMART PARKING SYSTEM                          *****
*****************************************************************************/
/*!
***     \file       Smart_Parking.c
***
***     \author     Aditya Subramanian
***                 Jerry Akpan
***
***     \brief      This file controls the working of a smart parking system
***     \par        File_description
***
***
*\n*/
/*****************************************************************************/
#include <stdio.h>
/* Library for LCD */
#include <LiquidCrystal.h>
/* Library for Servo Mortor */
#include <servo.h>

/* Global counter variable that holds value of current parking count */
int counter = 0;
/* Global variable that holds value of max parking count */
int park_max = 13;
/* LED GREEN connected as output PIN. Assign appropriate PIN */
int led_Green = OUTPUT(PIN NUMBER);
/* LED RED connected as output PIN. Assign appropriate PIN */
int led_Red = OUTPUT(PIN NUMBER);
/* User Button to ENTER . Set as INPUT PIN */
int button_Enter = INPUT(PIN NUMBER);
/* User Button to EXIT . Set as INPUT PIN */
int button_Exit = INPUT(PIN NUMBER);
/* Delay Variable */
int delay = 65535;
void main()
{
    /* Run User Code indefenitely until system is powered */
    while(1)
    {
        /*  By Default the RED_LED will be turned ON */
        led_Red(HIGH);
        led_Green(LOW);
        if( counter == 0)
        {
            LCD_DISPLAY(WELCOME);
            LCD_DISPLAY(park_max);
        }
        /* Check if user Enter button has been pressed */
        while( button_Enter == HIGH );
```

Figure 4 – The pseudocode for the smart parking system.

```
void main()
{
    /* Run User Code indefenitely until system is powered */
    while(1)
    {
        /*  By Default the RED_LED will be turned ON */
        led_Red(HIGH);
        led_Green(LOW);
        if( counter == 0)
        {
            LCD_DISPLAY(WELCOME);
            LCD_DISPLAY(park_max);
        }
        /* Check if user Enter button has been pressed */
        while( button_Enter == HIGH );
        {
           /* Come in if Enter button has been pressed atleast once */
           if(counter<=park_max)
           {
            counter++;
            Servo_OPEN();
            Led_Green(HIGH);
            Led_Red(LOW);
            /* wait for car to pass */
            while(delay--);
            DISPLAY(counter);
           }
           else
           {
            DISPLAY(PLEASE COME BACK LATER);
           }
           led_Red(HIGH);
           led_Green(LOW);
           delay = 65535;
        }
        /* Check if Exit Button has been pressed */
        while( button_Exit == HIGH);
        {
            counter --;
            Led_Green(HIGH);
            Led_Red(LOW);
            while(delay--);
           led_Red(HIGH);
           led_Green(LOW);
        }
    }
}
```

5. PROJECT SIMULATION

The project was simulated using a software called proteus. It is an open-source software commonly used to test and simulate embedded applications.

5.1. CONNECTION DIAGRAM

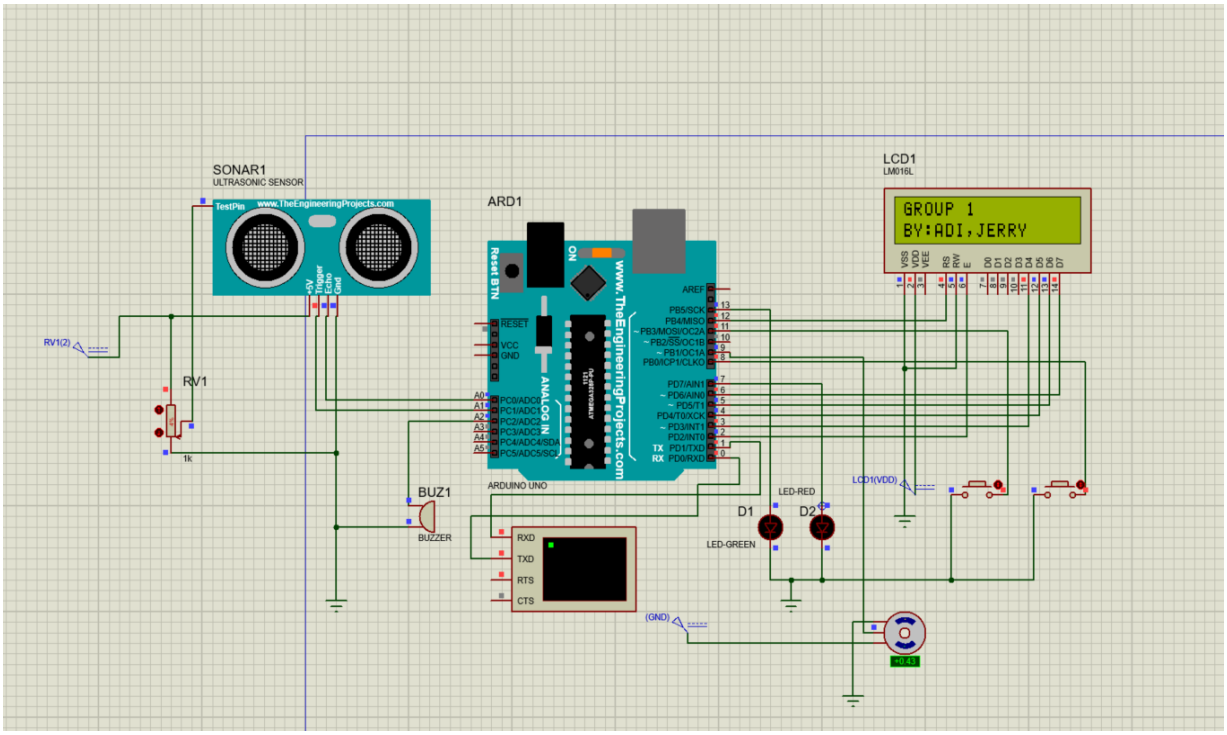

Figure 5- Simulated workspace in Proteus

The image above shows the simulated workspace in proteus. The Arduino is programmed using the Arduino IDE and later fed as binaries to the following Arduino board. The Arduino code for the following simulation is attached as images below.

5.2. ARDUINO CODE

The following section contains the Arduino code for the simulation.

```
/*****************************************************************************
*****                      SMART PARKING SYSTEM                          *****
*****************************************************************************/
/*!
***     \file       Smart_Parking.c
***
***     \author     Aditya Subramanian
***                 Jerry Akpan
***
***     \brief      This file controls the working of a smart parking system
***     \par        File_description
***
***
*\n*/
/*****************************************************************************/

/*============================================================================
=======                            INCLUDES                           =======
============================================================================*/
/*------ standard includes -------*/
#include "stdio.h"
#include "string.h"
/*------ project includes --------*/

/*------ module includes --------*/
#include <LiquidCrystal.h>
#include <Servo.h>
/*============================================================================
=======              DEFINES & MACROS FOR GENERAL PURPOSE             =======
============================================================================*/
/* MACROS */
#define LCD_D0 3
#define LCD_D1 4
#define LCD_D2 5
#define LCD_D3 6
#define LED_RED 7
#define LED_GREEN 13
#define BUTTON_ENTER 11
#define BUTTON_EXIT 8
#define SERVO_CONTROL 9
#define LCD_RS 12
#define LCD_EN 2
#define MAX_SPACE 13
#define NO_SPACE 0
#define BUTTON_PRESSED LOW
/* Local, Global variable declaration */
static int current_space = MAX_SPACE,servo_position,dist_check;
static bool Enter_ButtonState = HIGH,Exit_ButtonState=HIGH, safety_check = HIGH,Init_Button_State = HIGH;
long duration = 0;
/*============================================================================
```

Figure 6- Arduino Code Snippet

```
static bool Enter_ButtonState = HIGH,Exit_ButtonState=HIGH, Safety_Check = HIGH,Init_Button_State = HIGH;
long duration = 0;
/*============================================================================
=======                      CONSTANTS  &  TYPES                      =======
============================================================================*/
/* Initialize the LCD pins */
LiquidCrystal lcd(LCD_RS,LCD_EN,LCD_D0,LCD_D1,LCD_D2,LCD_D3);
Servo Servo_Arm;
/*============================================================================
=======                   PRIVATE FUNCTION DECLARATIONS               =======
============================================================================*/
static void parkingEntrySequence(void);
static void parkingExitSequence(void);
static void parkingNoSpaceSequence(void);
static void parkingDefaultSequence(void);
static void parkingHardwareInitSequence(void);
/*============================================================================
=======                             METHODS                           =======
============================================================================*/

void setup()
{
  /* Initialize Hardware Peripherals */
   parkingHardwareInitSequence();
}

void loop()
{

  /* Keep Reading Button states at the start of every loop cycle */
  Enter_ButtonState = digitalRead(BUTTON_ENTER);
  Exit_ButtonState  = digitalRead(BUTTON_EXIT);

  /* ENTRY BUTTON PRESSED */
  if (Enter_ButtonState == BUTTON_PRESSED)
  {
    /* Call Enter Parking Function */
    parkingEntrySequence();
  }

  /* EXIT BUTTON PRESSED */
  if(Exit_ButtonState == BUTTON_PRESSED)
  {
    /* Call Exit Sequence */
    parkingExitSequence();
  }

  if(current_space == NO_SPACE)
  {
    /* Call No Space Sequence */
```

```
/*============================================================================
=======                              METHODS                             =======
============================================================================*/

void setup()
{
  /* Initialize Hardware Peripherals */
   parkingHardwareInitSequence();
}

void loop()
{

  /* Keep Reading Button states at the start of every loop cycle */
  Enter_ButtonState = digitalRead(BUTTON_ENTER);
  Exit_ButtonState  = digitalRead(BUTTON_EXIT);

  /* ENTRY BUTTON PRESSED */
  if (Enter_ButtonState == BUTTON_PRESSED)
  {
    /* Call Enter Parking Function */
    parkingEntrySequence();
  }

  /* EXIT BUTTON PRESSED */
  if(Exit_ButtonState == BUTTON_PRESSED)
  {
    /* Call Exit Sequence */
    parkingExitSequence();
  }

  if(current_space == NO_SPACE)
  {
    /* Call No Space Sequence */
    parkingNoSpaceSequence();
  }
}

static void parkingHardwareInitSequence(void)
{
  Serial.begin(9600);
  /* Setup the LCD Module */
  lcd.begin(16,2);
  /* Setup PUSH Buttons */
  pinMode(BUTTON_ENTER,INPUT_PULLUP);
  pinMode(BUTTON_EXIT,INPUT_PULLUP);
  /* Setup LEDs */
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  /* Setup Servo Motor */
```

```
static void parkingHardwareInitSequence(void)
{
  Serial.begin(9600);
  /* Setup the LCD Module */
  lcd.begin(16,2);
  /* Setup PUSH Buttons */
  pinMode(BUTTON_ENTER,INPUT_PULLUP);
  pinMode(BUTTON_EXIT,INPUT_PULLUP);
  /* Setup LEDs */
  pinMode(LED_RED,OUTPUT);
  pinMode(LED_GREEN,OUTPUT);
  /* Setup Servo Motor */
  Servo_Arm.attach(SERVO_CONTROL);
  /* Setup Ultrasonic Sensor Pin Mode */
  pinMode(A1,OUTPUT);
  pinMode(A0,INPUT);
  /* Setup Buzzer */
  pinMode(A2,OUTPUT);
  /* For the very first time, set default condition */
  parkingDefaultSequence();
}

static void parkingDefaultSequence(void)
{
  lcd.setCursor(0,0);
  lcd.print("Welcome!!!");
  delay(5000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("GROUP 1 ");
  lcd.setCursor(0,1);
  lcd.print("BY:ADI,JERRY");
  while (Init_Button_State == HIGH)
  {
    /* Stay Here Until the button is pressed to initiate Main program*/
     Init_Button_State = digitalRead(BUTTON_ENTER);
     while(digitalRead(BUTTON_ENTER)== BUTTON_PRESSED);
  }
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Welcome!!!");
  lcd.setCursor(0,1);
  lcd.print(MAX_SPACE);
  digitalWrite(LED_RED,HIGH);
  Servo_Arm.write(0);
}

static void parkingNoSpaceSequence(void)
{
```

```
}

static void parkingEntrySequence(void)
{
  /*
   * IF Enter Button is pressed do the following
   * 1. Check if there is parking available
   * 2. If yes, turn servo motor by 90 degrees
   * 3. print Enter for 2 seconds and change it back to slot availability
   * 4. Enable Green LED
   * 5. With 3 second delay; push all configurations back to default.
   * 6. If no parking is available: Enter Parking FULL ; Please Try later
   * 7. Edit the availability and feed it back to the LCD
   */
   if(current_space == NO_SPACE)
   {
    /* The number of available slots are 0. Send Regret */
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Parking Full!");
    lcd.setCursor(0,1);
    lcd.print("Plz Come Later!");
    delay(2000);
   }
   else
   {
    /* Parking Available ; allow CAR */
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Enter!");
    digitalWrite(LED_RED,LOW);
    digitalWrite(LED_GREEN,HIGH);
    Servo_Arm.write(90);
    /*Delay for car to pass through */
    safety_check = HIGH;
    delay(3000);
    while (safety_check == HIGH)
    {
       digitalWrite(A1, LOW);
       delayMicroseconds(2);
       // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
       digitalWrite(A1, HIGH);
       delayMicroseconds(10);
       digitalWrite(A1, LOW);
       duration = (pulseIn(A0,HIGH));
       dist_check = (duration*0.034)/2;
       Serial.print("Distance: ");
       Serial.print(dist_check);
       Serial.println(" cm");
       if (dist_check < 10)
```

```
            }
            else
            {
             safety_check = LOW;
             digitalWrite(A2,LOW);
            }
          }
          current_space--;
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Welcome!!!");
          lcd.setCursor(0,1);
          lcd.print(current_space);
          /* Time to retract to default */
          digitalWrite(LED_GREEN,LOW);
          digitalWrite(LED_RED,HIGH);
          Servo_Arm.write(0);
         }
         /* To avoid multiple press or debouncing effect */
         while(digitalRead(BUTTON_ENTER)== BUTTON_PRESSED);
}

static void parkingExitSequence(void)
{
  /*   1. Allow the car to exit  if there is Car in spot
       2. Modify slot availability
       3. Close barrier and retract to default
    */
    if(current_space == MAX_SPACE)
    {
      /* Parking Empty
         Do Nothing */
    }
    else
    {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("ThankYou!");
      lcd.setCursor(0,1);
      lcd.print("Come Back Soon!");
      Servo_Arm.write(90);
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);
      current_space++;
      safety_check = HIGH;
 /* 2 second delay for car to exit */
      delay(2000);
      while (safety_check == HIGH)
      {
         digitalWrite(A1, LOW);
```

```
    }
    else
    {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("ThankYou!");
      lcd.setCursor(0,1);
      lcd.print("Come Back Soon!");
      Servo_Arm.write(90);
      digitalWrite(LED_RED,LOW);
      digitalWrite(LED_GREEN,HIGH);
      current_space++;
      safety_check = HIGH;
/* 2 second delay for car to exit */
      delay(2000);
      while (safety_check == HIGH)
      {
         digitalWrite(A1, LOW);
         delayMicroseconds(2);
         // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
         digitalWrite(A1, HIGH);
         delayMicroseconds(10);
         digitalWrite(A1, LOW);
         duration = (pulseIn(A0,HIGH));
         dist_check = (duration*0.034)/2;
         if (dist_check < 30)
         {
          safety_check = HIGH;
          digitalWrite(A2,HIGH);
          delay(500);
          digitalWrite(A2,LOW);
         }
         else
         {
          safety_check = LOW;
          digitalWrite(A2,LOW);
         }
      }
      Servo_Arm.write(0);
      digitalWrite(LED_GREEN,LOW);
      digitalWrite(LED_RED,HIGH);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Welcome!");
      lcd.setCursor(0,1);
      lcd.print(current_space);
       /* To avoid multiple press or debouncing effect */
      while(digitalRead(BUTTON_EXIT)== BUTTON_PRESSED);
      }
}
```

The code is modularized into multiple functions for better readability and efficiency.

The main methods/functions used in the project are

1. **Setup** → Takes care of setting up of hardware/Software Peripherals.
2. **ParkingHardwareInitSequenc**e → Definitions for hardware peripherals used.
3. **parkingDefaultSequence** → Governs default software condition before the main loop.
4. **Loop** → Continuous loop similar to while (1).
5. **parkingEntrySequence** → Governs the entry sequence of cars into the parking lot.
6. **parkingExitSequence** → Governs the exit sequence of cars leaving the parking lot.
7. **parkingNospaceSequence** → Governs the no-space sequence of the parking lot.

Each API has its specific functionality. Proteus then simulates this as if it were physical hardware.

6. CODE AND CIRCUIT EXPLANATION

6.1. CIRCUIT EXPLANATION

The main responsibility of the entire circuitry lies with the Arduino board. Different I/Os of the Arduino are employed to work with different hardware components of the project.

The following table shows the Arduino pins used against the hardware used.

Peripheral	Arduino Pinout
LCD (RS, EN, D4, D5, D6, D7)	D12, D2, D3, D4, D5, D6
Servo Motor	D9
PushButton1	D8
PushButton2	D11
LED1	D7
LED2	D11
Buzzer	A2
Ultrasonic (Trigger, Echo)	A1, A0

Table 1- Arduino Pins used against Hardware used

The peripherals are connected as mentioned in the table. The LCD, LED, BUZZER, and servo serve as output devices and pushbuttons and the ultrasonic sensor serves as input devices.

There are two input buttons: Enter Button and the Exit Button to allow for different operations. Based on these inputs, the circuit behaves in a certain way. Upon one of the buttons being pushed, the appropriate output peripherals are activated.

In addition, the ultrasonic sensor serves as a safety check feature in this system. In case a car is within close vicinity of the sensor, a buzzer is meant to ring, and further movement of the leaver attached to the servo motor is restricted.

6.2. SOFTWARE EXPLANATION

The entire project can be split into logical stages and each stage is explained in detail below.

6.2.1. STAGE 1

This is the Initialization of the Hardware, Software, and program Requirements stage. Different variables and hardware peripherals are initialized at this stage of the code. Servo, LCD, Ultrasonic, and other peripherals such as buttons and LEDs are initialized and the

Matlab code snippet for the same is shown in the image below.

```
%% Stage 1 : Initialization of Hardware, Software and Program Requirements
clc;
clear;
Enter_ButtonState   = 0;
Exit_ButtonState    = 0;
max_available_space = 13;
Init_button_status  = 1;
BUTTON_PRESSED      = 0;
current_space       = max_available_space;
NO_SPACE            = 0;
safety_check        = 1;
time_entry_index    = 1;
time_exit_index     = 1;
t_breaker           = 1;
old_date            = 0;
```

```
%% Initalize Hardware and Software peripherals
%{
1. Decide Arduino Hardware Pinout
2. Configure the pin as Output/Input
3. Initialize LCD, Servo and other peripherals as required
%}
% Create an object to control and work with the arduino; invoke the required libraries.
ard_control = arduino('COM8','Uno','Libraries',{'ExampleLCD/LCDAddon','Servo','Ultrasonic'},'ForceBuildOn',true);
% Create an object for servo control
servo_control = servo(ard_control,'D9');
% Create an object for LCD control
lcd_control = addon(ard_control,'ExampleLCD/LCDAddon','RegisterSelectPin','D12','EnablePin','D2','DataPins',{'D3','D4','D5','D6'})
% Create an object for ultrasonic sensor
ultrasonic_control = ultrasonic(ard_control,'A1','A0','OutputFormat','double');
% State pin usage
% Two input buttons
configurePin(ard_control,'D8','Pullup');
configurePin(ard_control,'D11','Pullup');
% Configure Pin for Buzzer
configurePin(ard_control,'A2','DigitalOutput');
%Two output LEDs red and green
configurePin(ard_control,'D7','DigitalOutput');
configurePin(ard_control,'D13','DigitalOutput');
initializeLCD(lcd_control);
```

6.2.2. STAGE 2

The init sequence is written to cover the project requirements to display the name and have appropriate display interfaces running smoothly. An additional feature has been included to have the user start the functioning of the main software; this is done with the help of a user button; once this button is pressed; the name tags are erased, and the functioning of the parking system starts. The Matlab code for the following can be found below.

```
%% Stage 2 : Init Sequence
%{
The init sequence should set stage for process to start.
cover project descriptions and all peripherals should be ready.
%}
writePosition(servo_control,0);
printLCD(lcd_control,'Welcome!!!');
pause(5);
clearLCD(lcd_control);
printLCD(lcd_control,'Group1');
printLCD(lcd_control,'By:ADI,JERRY');
while( Init_button_status == 1)
% % Stay here until button is pressed
Init_button_status = readDigitalPin(ard_control,'D11');
end
```

6.2.3. STAGE 3

This stage marks the beginning of the main program stage. The servo motor is set to its default position and the red LED is lit waiting for a car to enter. At this stage, the LCD indicates the Welcome message and the number of slots available currently in the parking lot. The Matlab code for the following can be found below.

```
%% Stage 3 Button is pressed; the actual program can start
%{
This stage is responsible to display the Welcome message, available
space in the parking lot and ensuring Red light is turned on.
%}
clearLCD(lcd_control);
printLCD(lcd_control,'Welcome!!!');
printLCD(lcd_control,strcat(num2str(max_available_space)));
% By default red ligth is turned on
writeDigitalPin(ard_control,'D7',1);
% Make sure servo is positioned at close
writePosition(servo_control,0);
```

6.2.4. STAGE 4

This is the main program loop where the code is continuously executed servicing the functionality within.

The entire stage 4 is logically written in a sequence of events that should occur in the parking system.

The Buttons are read to understand if a user is waiting to be serviced. If the entry button is pressed; the system checks to see if there is space available to service, the customer. If there is no space, a prompt indicates the lack of availability. If there is space, the customer is allowed into the parking space.

A safety feature has been implemented to prevent any accidents or mishaps in the parking space. A sensor is placed to detect if the car has passed through or not. If the car, for some reason, has not passed through, the lever does not shut, and a buzzer is activated indicating the user to move. Once the car has passed a safe distance; the system resets to the default state and the counter is decremented by 1.

The same applies to the exit condition. If there is a user wishing to exit; an appropriate greeting is displayed on the LCD and the leaver opens to let the car pass through. A similar safety feature is implemented on the way out, to let the car pass through safely. The car upon passing the safe distance is followed by the leaver closing and the system resetting to the default state. The counter is incremented by 1 indicating a freed-up slot in the parking space.

In case the system detects the parking lot to be full, the default message changes from welcome to "Parking Full, Plz Come Later". This prevents the user from pressing the enter button and even if done; no response is provided by the system.

During the system run, data is collected pertaining to entry time, exit time, and the number of cars parked at a given instance of time. This data is important to predict busy times of the day and generate statistics for business solutions. This data is pushed to the cloud in real-time and this can be monitored in a dashboard provided by Things Speak; the cloud and IoT vendor used for this project. This data can be made available to the public to know whether parking is available at this spot.

The Matlab code for the following feature is below.

```
%% Stage 4 : Main Program Loop : Stay in this loop till power shutdown
%{
Check Different button states and allow/deny entry of vehicles into the
parking lot.
%}
while(1)

    % Continuously check both the button statuses
    Enter_ButtonState  = readDigitalPin(ard_control,'D11');
    Exit_ButtonState   = readDigitalPin(ard_control,'D8');

    if Enter_ButtonState == BUTTON_PRESSED
        % Activate Entry Sequence
        %{
           /*
            *** IF Enter Button is pressed do the following***
            * 1. Check if there is parking space available
            * 2. If yes, turn servo motor by 90 degrees
            * 3. print Enter for 2 seconds and change it back to slot availability
            * 4. Enable Green LED
            * 5. With 3 second delay; push all configurations back to default.
            * 6. If no parking is available: Enter Parking FULL ; Please Try later
            * 7. Edit the availability and feed it back to the LCD
           */
        %}
        if current_space == NO_SPACE
           clearLCD(lcd_control);
```

```
        */
    %}
    if current_space == NO_SPACE
        clearLCD(lcd_control);
        printLCD(lcd_control,'Parking Full!');
        printLCD(lcd_control,'Plz Come Later');
        pause(2);
    else
        clearLCD(lcd_control);
        printLCD(lcd_control,'Enter!');
        time_entry(time_entry_index,1) = datetime('now');
        time_entry_index = time_entry_index+1;
        writeDigitalPin(ard_control,'D7',0);
        writeDigitalPin(ard_control,'D13',1);
        writePosition(servo_control,0.5);
        % Allow for car to pass
        safety_check = 1;
        pause(3);
        % To check whether the car has passed ; use the ultrasonic
        % sensor to measure distance.
        % if there is an obstacle within 10 cm of sensor. Assume car
        % hasnt passed and trigger buzzer.
        % stay on this loop until obstacle has cleared the way.
        while safety_check == 1
              dist_check_1 = readDistance(ultrasonic_control);
              dist_check = round(dist_check_1*100,2);
              if dist_check <= 30
```

```
        % hasnt passed and trigger buzzer.
        % stay on this loop until obstacle has cleared the way.
        while safety_check == 1
              dist_check_1 = readDistance(ultrasonic_control);
              dist_check = round(dist_check_1*100,2);
              if dist_check <= 30
                  safety_check = 1;
                  % Trigger Buzzer
                  writeDigitalPin(ard_control,'A2',1);
                  pause(1.5);
                  writeDigitalPin(ard_control,'A2',0);
              else
                  writeDigitalPin(ard_control,'A2',0);
                  safety_check = 0;
              end
        end
        current_space = current_space-1;
        writePosition(servo_control,0);
        writeDigitalPin(ard_control,'D13',0);
        writeDigitalPin(ard_control,'D7',1);
        clearLCD(lcd_control);
        printLCD(lcd_control,'Welcome!');
        printLCD(lcd_control,strcat(num2str(current_space)));
    end
    while (BUTTON_PRESSED == readDigitalPin(ard_control,'D11'))
    end
end
```

```
% Exit State
if Exit_ButtonState == BUTTON_PRESSED
    %{
     /*   1. Allow the car to exit if there is Car in spot
      *   2. Modify slot availability
      *   3. Close barrier and retract to default
      */
    %}
     safety_check = 1;
     if current_space == max_available_space
         % Do Nothing; no car is inside.
     else
       clearLCD(lcd_control);
       printLCD(lcd_control,'Thank You!');
       printLCD(lcd_control,'Come Back Soon!');
       time_exit(time_exit_index,1) = datetime('now');
       time_exit_index = time_exit_index+1;
       writeDigitalPin(ard_control,'D7',0);
       writeDigitalPin(ard_control,'D13',1);
       % Allow Car to pass through
       writePosition(servo_control,0.5);
       % Wait for Car to pass through
       pause(3);
       while safety_check == 1
             dist_check_1 = readDistance(ultrasonic_control);
             dist_check = round(dist_check_1*100,2);
             if dist_check <= 30
```

```
        writePosition(servo_control,0);
        writeDigitalPin(ard_control,'D13',0);
        writeDigitalPin(ard_control,'D7',1);
        clearLCD(lcd_control);
        printLCD(lcd_control,'Welcome!!!');
        printLCD(lcd_control,strcat(num2str(current_space)));
        while (BUTTON_PRESSED == readDigitalPin(ard_control,'D8'))
        end
    end
end

if current_space == NO_SPACE
    % No space sequence invoked
    clearLCD(lcd_control);
    printLCD(lcd_control,'Parking Full!');
    printLCD(lcd_control,'Plz Come Later');
end
% Send Data to the cloud every one minute; to track the stats of
% parking utilization.
c = clock;
new_date = c(3);
if t_breaker == 1
   ref_time = c(5);
   t_breaker = 0;
end
send_time = c(5) - ref_time;

if send_time >= 1 || old_date ~= new_date
```

```
    if send_time >= 1 || old_date ~= new_date
        % entry time buffer is full; send the data to the cloud and reset
        % the index.
        ref_time = c(5);
        ChannelId  = 1807419;
        writekey   = 'JN3VNGACM4THZBCH';
        userApikey = '923ZHT68B6COANY0';
        url = sprintf('https://api.thingspeak.com/channels/1807419/feeds.json?api_key=%s',userApikey);
        if old_date ~= new_date
            %flush the data after the day ends.
            response = webwrite(url,weboptions('RequestMethod','delete'));
            pause(1);
            old_date = new_date;
        end
        display('Sending Data to cloud');
        %data = timetable(time_entry,current_space)
        thingSpeakWrite(ChannelId,current_space,'WriteKey',writekey);
    end
end
```

The data from the cloud is cleared every day to avoid overlap of data and overhead of data management. This is done through an API provided by Things Speak through Matlab.

7. TESTING SCENARIOS

The following section covers different testing scenarios to test the robustness and corner cases of the system. This has been recorded in the table shown before.

Testing Scenarios	Status (PASS/FAIL)
When the parking lot is empty, 6 cars enter the parking lot sequentially.	PASS
1 car leaves the parking lot	PASS
8 cars enter the parking lot sequentially.	PASS
1 Car enters the parking lot	PASS
1 Car leaves the parking lot	PASS
1 car enters the parking lot	PASS
User Double presses button	No system actions. Software Handle. PASS
Obstacle placed in front of the sensor	Servo Motor does not shut. Buzzer screams. Software Handle. PASS
Exit button pressed when parking empty	No action from the system
Date change; results of flushing data in cloud	PASS. Software Handle.
Enter Button pressed when parking full	Appropriate response given. Software Handle. PASS

Table 2- Testing Scenarios

8. MATLAB CODE

The Matlab code for the entire project is placed below.

```
%%
%{
/*****************************************************************************
*
*****                    SMART PARKING SYSTEM                         *****
*****************************************************************************
/
/*!
***    \file       Smart_Parking.c
***
***    \author     Aditya Subramanian
***                Jerry Akpan
***
***    \brief      This file controls the working of a smart parking system
***    \par        File_description
***
***
*\n*/
/*****************************************************************************
/
%}
%% Stage 1 : Initialization of Hardware, Software and Program Requirements
clc;
clear;
Enter_ButtonState          = 0;
Exit_ButtonState           = 0;
max_available_space        = 13;
Init_button_status         = 1;
BUTTON_PRESSED             = 0;
current_space              = max_available_space;
NO_SPACE                   = 0;
safety_check               = 1;
time_entry_index           = 1;
time_exit_index            = 1;
t_breaker                  = 1;
old_date                   = 0;
%% Initalize Hardware and Software peripherals
%{
1. Decide Arduino Hardware Pinout
2. Configure the pin as Output/Input
3. Initialize LCD, Servo and other peripherals as required
%}
```

```
% Create an object to control and work with the arduino; invoke the required libraries.
ard_control =
arduino('COM8','Uno','Libraries',{'ExampleLCD/LCDAddon','Servo','Ultrasonic'},'ForceBuildO
n',true);
% Create an object for servo control
servo_control = servo(ard_control,'D9');
% Create an object for LCD control
lcd_control =
addon(ard_control,'ExampleLCD/LCDAddon','RegisterSelectPin','D12','EnablePin','D2','DataPin
s',{'D3','D4','D5','D6'});
% Create an object for ultrasonic sensor
ultrasonic_control = ultrasonic(ard_control,'A1','A0','OutputFormat','double');
% State pin usage
% Two input buttons
configurePin(ard_control,'D8','Pullup');
configurePin(ard_control,'D11','Pullup');
% Configure Pin for Buzzer
configurePin(ard_control,'A2','DigitalOutput');
%Two output LEDs red and green
configurePin(ard_control,'D7','DigitalOutput');
configurePin(ard_control,'D13','DigitalOutput');
initializeLCD(lcd_control);

%% Stage 2 : Init Sequence
%{
The init sequence should set stage for process to start.
cover project descriptions and all peripherals should be ready.
%}
writePosition(servo_control,0);
printLCD(lcd_control,'Welcome!!!');
pause(5);
clearLCD(lcd_control);
printLCD(lcd_control,'Group1');
printLCD(lcd_control,'By:ADI,JERRY');
while( Init_button_status == 1)
% % Stay here until button is pressed
Init_button_status = readDigitalPin(ard_control,'D11');
end

%% Stage 3 Button is pressed; the actual program can start
%{
This stage is responsible to display the Welcome message, available
space in the parking lot and ensuring Red light is turned on.
%}
clearLCD(lcd_control);
printLCD(lcd_control,'Welcome!!!');
```

```
printLCD(lcd_control,strcat(num2str(max_available_space)));
% By default red ligth is turned on
writeDigitalPin(ard_control,'D7',1);
% Make sure servo is positioned at close
writePosition(servo_control,0);

%% Stage 4 : Main Program Loop : Stay in this loop till power shutdown
%{
Check Different button states and allow/deny entry of vehicles into the
parking lot.
%}
 while(1)

    % Continuously check both the button statuses
    Enter_ButtonState  = readDigitalPin(ard_control,'D11');
    Exit_ButtonState   = readDigitalPin(ard_control,'D8');

    if Enter_ButtonState == BUTTON_PRESSED
       % Activate Entry Sequence
       %{
         /*
          *** IF Enter Button is pressed do the following***
          * 1. Check if there is parking space available
          * 2. If yes, turn servo motor by 90 degrees and do safety check
          * 3. print Enter for 2 seconds and change it back to slot availability
          * 4. Enable Green LED
          * 5. With 3 second delay; push all configurations back to default.
          * 6. If no parking is available: Enter Parking FULL ; Please Try later
          * 7. Edit the availability and feed it back to the LCD
         */
       %}
       if current_space == NO_SPACE
          clearLCD(lcd_control);
          printLCD(lcd_control,'Parking Full!');
          printLCD(lcd_control,'Plz Come Later');
          pause(2);
       else
          clearLCD(lcd_control);
          printLCD(lcd_control,'Enter!');
          time_entry(time_entry_index,1) = datetime('now');
          time_entry_index = time_entry_index+1;
          writeDigitalPin(ard_control,'D7',0);
          writeDigitalPin(ard_control,'D13',1);
          writePosition(servo_control,0.5);
          % Allow for car to pass
          safety_check = 1;
```

```
        pause(3);
        % To check whether the car has passed ; use the ultrasonic
        % sensor to measure distance.
        % if there is an obstacle within 10 cm of sensor. Assume car
        % hasnt passed and trigger buzzer.
        % stay on this loop until obstacle has cleared the way.
        while safety_check == 1
            dist_check_1 = readDistance(ultrasonic_control);
            dist_check = round(dist_check_1*100,2);
            if dist_check <= 30
                safety_check = 1;
                % Trigger Buzzer
                writeDigitalPin(ard_control,'A2',1);
                pause(1.5);
                writeDigitalPin(ard_control,'A2',0);
            else
                writeDigitalPin(ard_control,'A2',0);
                safety_check = 0;
            end
        end
        current_space = current_space-1;
        writePosition(servo_control,0);
        writeDigitalPin(ard_control,'D13',0);
        writeDigitalPin(ard_control,'D7',1);
        clearLCD(lcd_control);
        printLCD(lcd_control,'Welcome!');
        printLCD(lcd_control,strcat(num2str(current_space)));
    end
    while (BUTTON_PRESSED == readDigitalPin(ard_control,'D11'))
    end
end

% Exit State
if Exit_ButtonState == BUTTON_PRESSED
    %{
     /*   1. Allow the car to exit if there is Car in spot
      *   2. Modify slot availability
      *   3. Close barrier and retract to default
      */
    %}
     safety_check = 1;
     if current_space == max_available_space
        % Do Nothing; no car is inside.
     else
       clearLCD(lcd_control);
       printLCD(lcd_control,'Thank You!');
```

```
        printLCD(lcd_control,'Come Back Soon!');
        time_exit(time_exit_index,1) = datetime('now');
        time_exit_index = time_exit_index+1;
        writeDigitalPin(ard_control,'D7',0);
        writeDigitalPin(ard_control,'D13',1);
        % Allow Car to pass through
        writePosition(servo_control,0.5);
        % Wait for Car to pass through
        pause(3);
        while safety_check == 1
            dist_check_1 = readDistance(ultrasonic_control);
            dist_check = round(dist_check_1*100,2);
            if dist_check <= 30
                safety_check = 1;
                % Trigger Buzzer
                writeDigitalPin(ard_control,'A2',1);
                pause(1.5);
                writeDigitalPin(ard_control,'A2',0);
            else
                writeDigitalPin(ard_control,'A2',0);
                safety_check = 0;
            end
        end
        current_space = current_space+1;
        writePosition(servo_control,0);
        writeDigitalPin(ard_control,'D13',0);
        writeDigitalPin(ard_control,'D7',1);
        clearLCD(lcd_control);
        printLCD(lcd_control,'Welcome!!!');
        printLCD(lcd_control,strcat(num2str(current_space)));
        while (BUTTON_PRESSED == readDigitalPin(ard_control,'D8'))
        end
    end
end

if current_space == NO_SPACE
    % No space sequence invoked
    clearLCD(lcd_control);
    printLCD(lcd_control,'Parking Full!');
    printLCD(lcd_control,'Plz Come Later');
end
% Send Data to the cloud every one minute; to track the stats of
% parking utilization.
c = clock;
new_date = c(3);
if t_breaker == 1
```

```
        ref_time = c(5);
        t_breaker = 0;
    end
    send_time = c(5) - ref_time;

    if send_time >= 1 || old_date ~= new_date
        % entry time buffer is full; send the data to the cloud and reset
        % the index.
        ref_time = c(5);
        ChannelId  = 1807419;
        writekey   = 'JN3VNGACM4THZBCH';
        userApikey = '923ZHT68B6COANY0';
        url =
sprintf('https://api.thingspeak.com/channels/1807419/feeds.json?api_key=%s',userApikey);
        if old_date ~= new_date
            %flush the data after the day ends.
            response = webwrite(url,weboptions('RequestMethod','delete'));
            pause(1);
            old_date = new_date;
        end
        display('Sending Data to cloud');
        %data = timetable(time_entry,current_space)
        thingSpeakWrite(ChannelId,current_space,'WriteKey',writekey);
    end
end
```

**********END**********

9. BONUS SECTION

The bonus section covers details of all the additional features added to the system apart from the core requirements of the project. This can be found in the table below.

BONUS	PURPOSE
Added Ultrasonic Sensor	For safety reasons.
Buzzer	Added to increase the safety level of the system
Data Analytics using things speak	Sending acquired data to the cloud to create a dashboard view of parking statistics and this data can be made public to be informed on parking spot availability.
Time of Entry and Exit is captured	This can help predict busy parts of the day; using powerful MATLAB functions such as Interpolation (Interp1) and Extrapolation; predictive analysis can be applied to this data and fed into the system for users to plan their parking.
Proteus Project Simulation + Arduino Code	To have a soft prototype (Proof of Concept)

Table 3 – Additional features added to the system.

The dashboard view on Things speak is attached below.

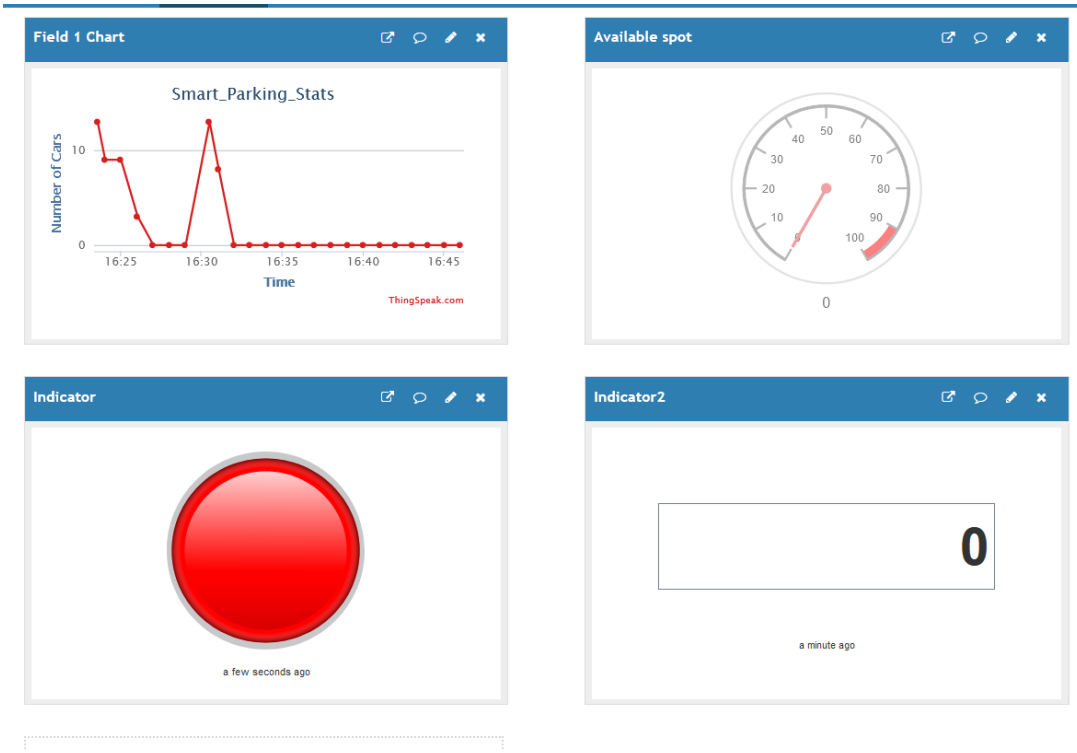

Figure 7 – Dashboard view on Things Speak.

This data is updated every minute and can be observed in real-time. This can help generate good statistics and informed business decisions can be taken. In addition, during the program; the time of entry and exit are recorded, serving as a database for predictive analysis.

10. Lessons Learned

The following lessons were taken away from implementing this system

1. MATLAB poses extensive restrictions on using the Arduino hardware to its full potential.
2. Interfacing MATLAB and Arduino is useful for data analytics purposes, but if the motive is not to capture too much data; Arduino standalone is convenient to use.
3. MATLAB does not flash the software into system flash; the entire program runs on RAM and data is fetched from the MATLAB in real-time.
4. Things Speak allows for convenient data analysis and easy integration to MATLAB.
5. The serial communication between MATLAB and Arduino is a useful channel to leverage.

11.PROJECT TIMELINE AND RISK ASSESSMENT

11.1. PROJECT TIMELINE

Date	Agenda	Status
June 13	Formed team and ordered an Arduino kit.	Completed
June 20	Investigated Different parking management projects as a case study.	Completed
June 27	Implementation of individual modules with hardware.	Completed
July 4	Testing and Finalizing all software components.	Completed
July 18	Final Integration of software and Hardware.	Completed
July 20	Simulation with Proteus	Completed
July 24	Final Integration to cloud with software and Hardware	Completed

Table 4 – Project timeline

11.2. RISK ASSESSMENT

Matlab and Arduino Interfacing proved challenging. In addition, using WebAPIs with MATLAB to push and retrieve data from Things Speak Cloud was considered a risk factor to the project.

12.CONCLUSIONS

The following features have been successfully implemented in the smart parking system

1. Arduino and MATLAB have been successfully interfaced.
2. Proteus simulation of the project has been completed indicating completeness of Proof Of Concept.
3. System allows for 13 parking spots and interactive hardware for users to interact and use the parking facility.
4. Safety features have been put in place to avoid any mishaps in the parking system.
5. Cloud integration has been added to facilitate better data analytics.

13. REFERENCES

[1] A. Team, "'Hello world!"," *Arduino*. [Online]. Available: https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld/. [Accessed: 26-Jun-2022].

[2] A. Team, "Servo Motor Basics with Arduino: Arduino documentation," *Arduino Documentation | Arduino Documentation*, 22-Jun-2022. [Online]. Available: https://docs.arduino.cc/learn/electronics/servo-motors. [Accessed: 26-Jun-2022].

[3] G. Finet, A. Negi, and N. Cueto, "How to add and simulate ultrasonic sensor library in Proteus?," *eTechnophiles*, 24-Jul-2022. [Online]. Available: https://www.etechnophiles.com/add-simulate-ultrasonic-sensor-proteus-2018-edition/. [Accessed: 25-Jul-2022].

[4] "The engineering projects," *The Engineering Projects*. [Online]. Available: https://www.theengineeringprojects.com/document/ultrasonic-sensor-library-proteus/1. [Accessed: 25-Jul-2022].

[5] "Getting started with thingsSpeak," *Get started with ThingSpeak*. [Online]. Available: https://www.mathworks.com/help/thingspeak/getting-started-with-thingspeak.html. [Accessed: 25-Jul-2022].