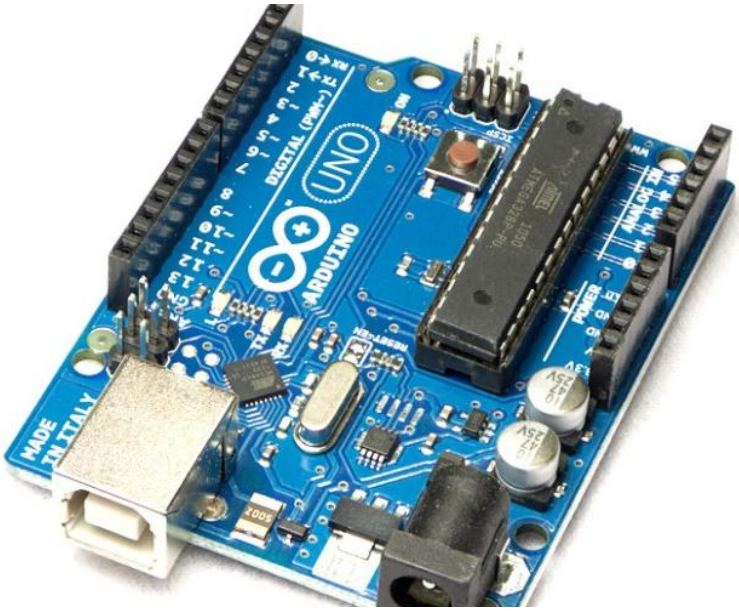


MICROPROCCESSORS AND PERIPHERALS

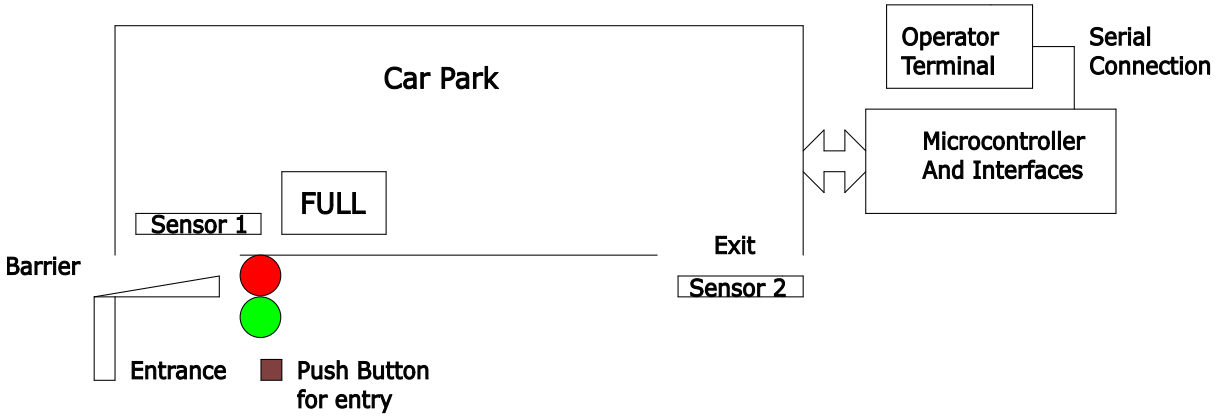
Arduino Code Assignment

Williams, Matthew

Table of Contents

Design Brief 2
Project Plan 3
Flow Chart 4
State Diagram 5
Algorithm 6
Code 9
The Structure of the code 19
Circuit Diagram 27
Test Plan 27
Full replica simulation 28
Test Results 28
Conclusions from Testing 31

Design Brief



The above represents a generic design for a car park with a single entrance controlled by a traffic light system and a barrier. The exit is free flow.
An Arduino microcontroller provides the control functionality. A terminal connected via a USB link provides an operator interface.

The car park is intended to operate as follows,

1. An operator sets the maximum available number of parking spaces (in the range 10 to 255.
2. Cars arriving at the entrance are initially prevented from entry by a barrier.
   A red light will also be illuminated
3. The driver of the car may request entry by pressing a push button.
4. If the car park is full then a full sign is illuminated and entry will be denied
5. If the car park is not full then the barrier will raise. When it has risen the red light will extinguish and a green lamp will become illuminated.
6. When the car has passed the barrier as detected by the car's presence at sensor 1 then the green lamp will extinguish, the red lamp will be illuminated and the barrier will lower. (Note there is an assumption that vehicles are clear of the barrier at this time).
7. Cars may leave the car park at any time. In doing so they activate sensor 2.
   (Billing is handled by a separate number plate reading system)
8. The operator has additional controls and indications namely :-
   a. The operator can view how many cars are parked at any time
   b. The operator can see the state of the barrier, red lamp, green lamp, Full lamp.
   c. The operator can cause the barrier to raise or lower
   d. The operator can cause the indicators (Red lamp, Green lamp, Full Lamp) to be ON or OFF.
   e. The operator can prevent more cars from entering the car park regardless of how many cars are actually in the car park.
      (Details of exactly how the operator will perform these actions is left to your design. You *may* wish to consider a Manual / Automatic mode of operation)

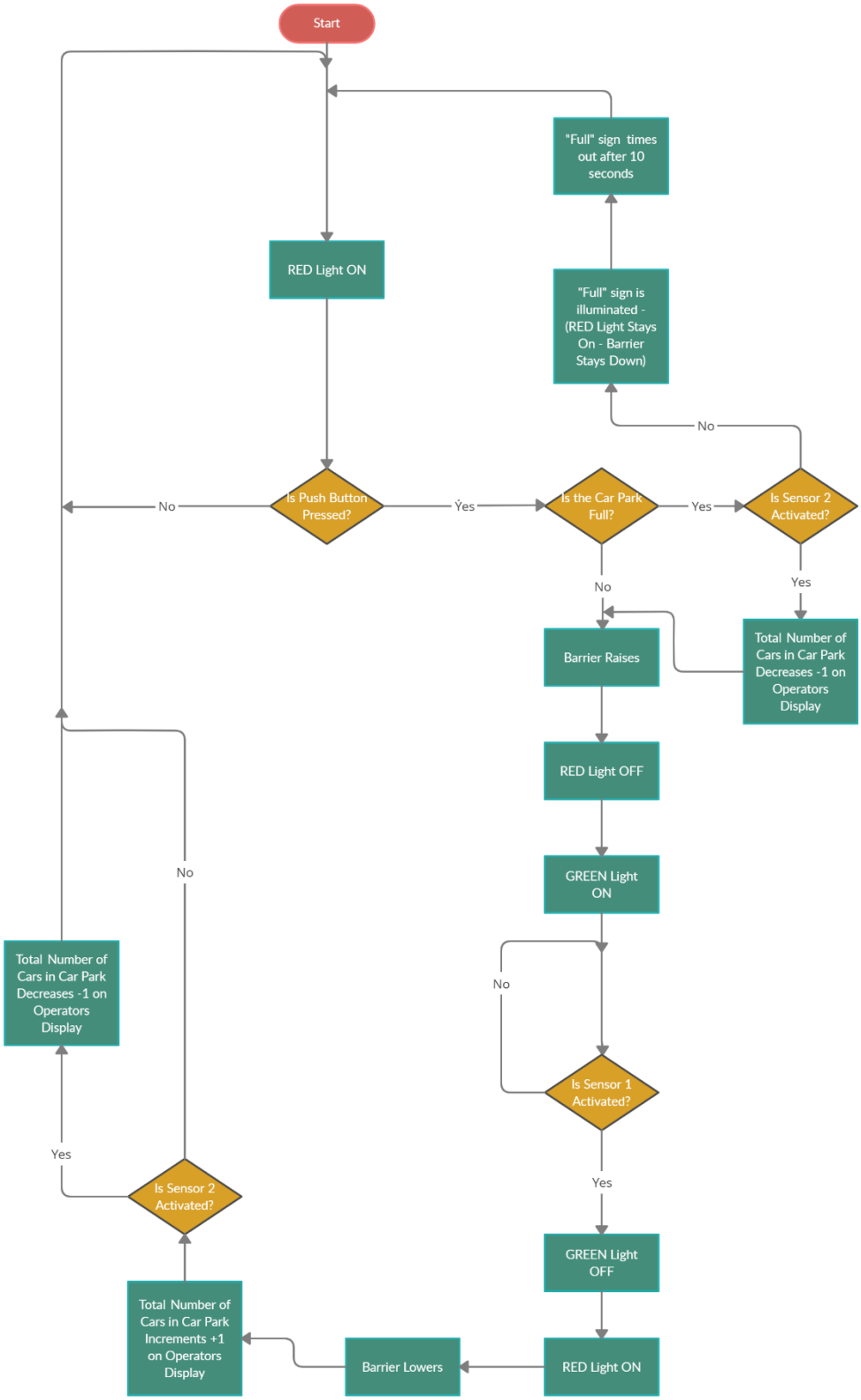
The microcontroller interface towards the car park is as follows:-

1. Red Lamp A digital output on Pin 6 Logic level 1 = Lamp ON
2. Green Lamp A digital output on Pin 5 Logic level 1 = Lamp ON
3. Full Lamp A digital output on Pin 7 Logic level 1 = Full Lamp ON
4. Barrier control an interface to a stepper motor connected on pins 8, 9, 10, 11 the stepper has 200 steps per revolution. 90° clockwise raises the barrier,
5. Sensor 1 A logic input on Pin 3 Logic level 1 = car present
6. Sensor 2 A logic input on Pin 4 Logic level 1 = car present
7. Push Button. A logic input on Pin 2 (derived from a mechanical switch) Logic level 1 = pressed

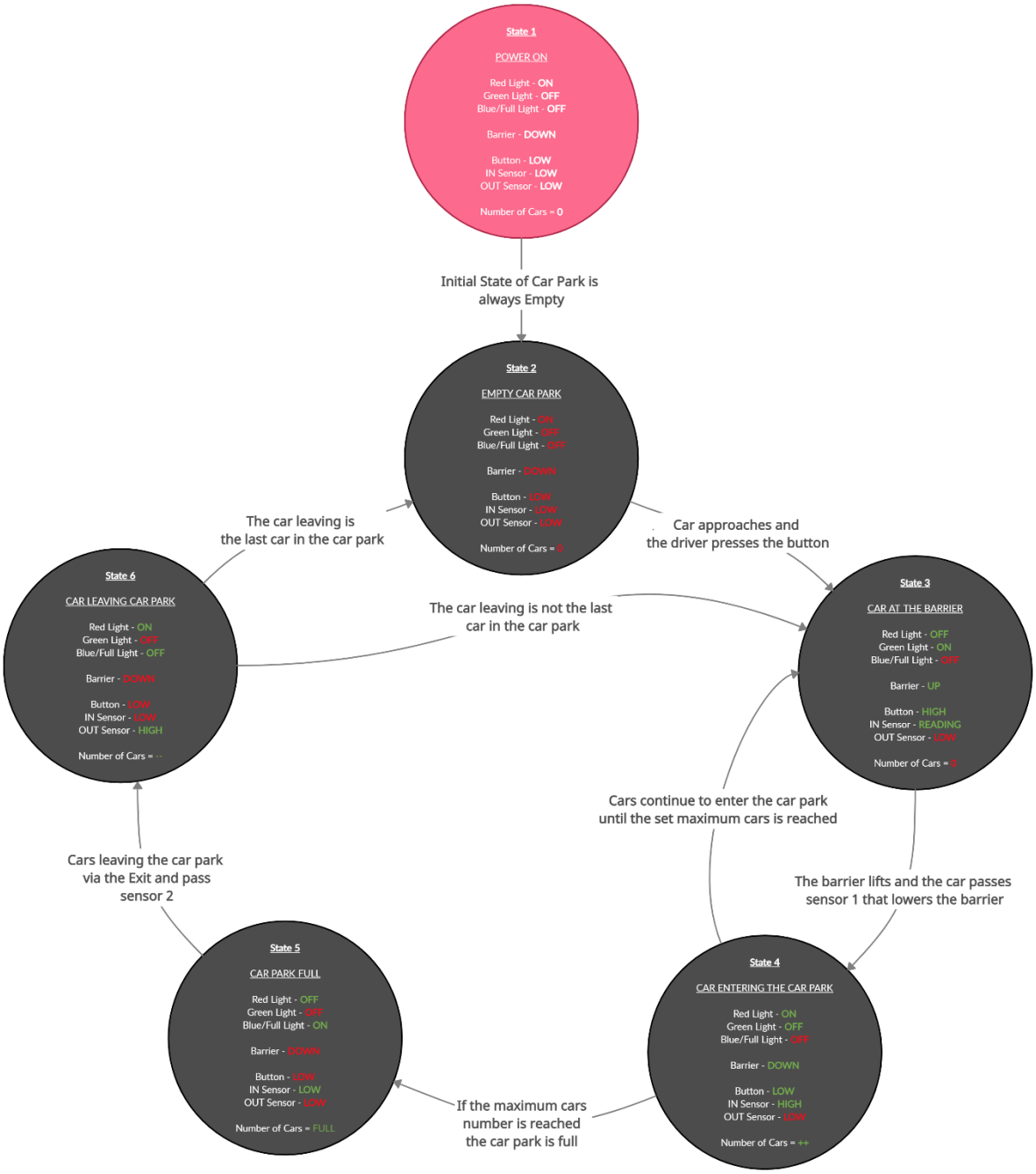
Project Plan

1) Create a plan of how the car park will work an Look
    a. A Flow Chart to work out the different individual states that the automated system might find itself in, and what decisions will have to be made.
    b. A State Diagram to identify the changes that happen when the program runs into a new state. I.e. what lights change when, when do the barriers raise and lower.
    c. Create an algorithm to structure the program in a logical way to achieve what the brief requires.
2) Program Code
    a. Start with assigning the pins on the Arduino
    b. Listing the variables that will be needed
    c. Code the void setup () with what the status of the Pin Modes should be
    d. Break down the void Loop () into small more manageable parts:
        i. Start with the smallest code parts needed
        ii. Write simple code for every state
        iii. Build on the simple code to enable the program to work better and in a more user friendly way.
        iv. Get the program to work in an automatic way first
        v. Add in the Manual override parts of the code.
        vi. Make tweaks to the code once it's tested.
3) Draw out a Circuit Diagram
4) Draw up a test plan
5) Create a simulation or Build the design if available
6) Test design according to plan
7) Record Test Results
8) Make any conclusions or note any changes I would have implemented if I was to do this again

Flow Chart


Start
"Full" sign times out after 10 seconds
RED Light ON
"Full" sign is illuminated - (RED Light Stays On - Barrier Stays Down)
No
No
Is Push Button Pressed?
Yes
Is the Car Park Full?
Yes
Is Sensor 2 Activated?
No
Yes
Barrier Raises
Total Number of Cars in Car Park Decreases -1 on Operators Display
RED Light OFF
GREEN Light ON
No
No
Total Number of Cars in Car Park Decreases -1 on Operators Display
Is Sensor 1 Activated?
Yes
Yes
Is Sensor 2 Activated?
GREEN Light OFF
Total Number of Cars in Car Park Increments +1 on Operators Display
Barrier Lowers
RED Light ON

State Diagram



Algorithm

- Include the stepper motor Library
- Set Button to pin 2
- Set Sensor 1 to pin 3
- Set Sensor 2 to pin 4
- Set the Green LED to pin 5
- Set the Red LED to pin 6
- Set the Full LED to pin 7
- Set the number of steps one full revolution of the Stepper Motor to 2048
- Set the stepper motor speed to 10
- Set the maximum number of cars allowed in the car park to 4
- Create a variable newFullPark to enable operator to set their own maximum number and set to 0
- Create a variable cars and set to 0
- Create a variable carsDailyTotal and set to 0
- Create a variable Barrier and set to 0
- Create a variable insense and set to 0
- Create a variable outsense and set to 0
- Create a variable storedSensor2 and set to 1
- Create a variable FullOn and set to LOW
- Create a variable GreenOn and set to LOW
- Create a variable RedOn and set to LOW
- Set Stepper Motor to pins 8, 10, 9 and 11

- Set the stepper motor speed to the variable motSpeed
- Initialise the Serial Port at 9600 baud rate
- Print a welcome message
- Print the number of cars allowed in the car park today
- Print a message to instruct how to launch the Operators Menu
- Set the Button as an INPUT
- Set Sensor 1 as an INPUT

- Set Sensor 2 as an INPUT
- Set the Green LED as an OUTPUT
- Set the Red LED as an OUTPUT
- Set the Full LED as an OUTPUT
- Set the Button to LOW
- Set Sensor 1 to LOW
- Set Sensor 2 to LOW
- Set the Red LED HIGH

- If the car park is full
    - Set the Red LED LOW
    - Set the Full/Blue LED HIGH
    - Print a message to tell the Operator
- If the car park is not full
    - Read the button to see if it is pressed
        - If the button is pressed raise the barrier
        - Print a message to say the barrier is lifting
        - Add one to the variable cars
        - Set the Red LED LOW
        - Set the Green LED HIGH
    - Read Sensor 1 to see if a car passes it
        - When a car passes Sensor 1 set the Red LED HIGH
        - Set the Green LED LOW
        - Lower the barrier
        - Print a message to say the barrier is coming down
        - Print a message to say how many cars are now in the car park
- If a car leaves the car park
    - Read Sensor 2 to see if a car passes it
        - If a car passes sensor 2 reduce one from the variable cars
        - Print a message to say a car has left
        - Print a message to say how many cars are now in the car park

- Set the letter ‘Q’ to pull up the Operators Menu on the serial monitor
- Print the Operators Menu
- Set the letter ‘B’ to turn the Full/Blue LED on/off
- Print the Full Light is on
- Set the letter ‘G’ to turn the Green LED on/off
- Print the Green Light is on
- Set the letter ‘R’ to turn the Red LED on/off
- Print the Red Light is on
- Set the ‘+’ to raise the Barrier
- Print the Barrier is up
- Set the ‘-’ to lower the Barrier
- Print the Barrier is down
- Set ‘O’ to show the status of the LED’s
- Print the status of the LED’s
- Set ‘Z’ to show the number of cars in the car park
- Print the number of cars currently in the car park
- Set ‘H’ to show the Barrier status
- Print the barrier status
- Set ‘D’ to show the number of cars that have been in the car park today
- Print the number of cars that have been in the car park since the program started
- Set ‘N’ to change the number of cars allowed in the car park at any one time
- Print the question “How many cars do you want?”
- Set the answer to newFullPark variable and override the cars variable
- Print the new number of cars allowed in the car park

Code

```
/*Arduino Microcontrolled Smart Car Parking System

 Designed by Matthew Williams

 April/May 2021

 HNC Microprocessors Assignment

*/


#include <Stepper.h>


int Button = 2;  //assign the push button to Pin 2

int sensor1 = 3; //assign the first sensor to Pin 3

int sensor2 = 4; //assign the second sensor to Pin 4

int Green = 5;   //assign the Green LED to Pin 5

int Red = 6;     //assign the Red LED to Pin 6

int Full = 7;    //assign the Blue/Full LED to Pin 7

int stepsPerRevolution = 2048; //the number of steps per on revolution of the stepper motor

int motSpeed=10;  //set the speed of the stepper motor at 10 rpm:

int fullPark =2;  //the number of cars that will make the car park full

int newFullPark= 0; //the variable that changes when the user defines how many cars the car park can hold: between 10 & 255

int cars = 0;     //the variable that stores the number of cars currently in the cat oark at any given time

int carsDailyTotal=0; //the variable that constantly adds the number of cars entering the car park (not taking into account cars leaving)

int barrier=0;  //variable assigned to calculate the current barrier possition

int insense=0;  //defying the 1st sensor, Entry Sensor

int outsense = 0;  //defying the 2nd sensor, Exit Sensor

int storedSensor2 =1; //defines the state of Sensor 2, Exit Sensor
```

```
int FullOn =LOW;  //variable for manualy switching Blue/Full LED light on/off

int GreenOn =LOW; //variable for manualy switching Green LED light on/off

int RedOn =LOW;   //variable for manualy switching Red LED light on/off


Stepper myStepper(stepsPerRevolution, 8,10,9,11); //initialize the stepper library on pins 8 through
11 in correct sequence


void setup()

{  //this is the setup code, this code runs once only at the start of your program.


myStepper.setSpeed(motSpeed); // initialize the set speed of the stepper motor from above

Serial.begin(9600);  //initialize the serial port at 9600 baud rate

Serial.println(""); //insert a blank row to keep the serial monitor looking tidy

Serial.println(" Welcome to Matthew's Car Park"); //print a welcome mesage on initial power up

Serial.println(""); //insert a blank row to keep the serial monitor looking tidy

Serial.print(" The Max number of cars allowed into the Car Park today is "); //print this message

Serial.print(fullPark); //takes the preset number from the variable

Serial.println(" cars!");//then prints this message

Serial.println(""); //insert a blank row to keep the serial monitor looking tidy

Serial.println(" To change this number or to access the OPERATORS MENU enter 'Q' into the Serial
Monitor at any time"); //print a prompt to enter the OPERATORS MENU

Serial.println(""); //insert a blank row to keep the serial monitor looking tidy



pinMode(Button, INPUT);//set the button as an input

pinMode(sensor1, INPUT);//set sensor 1 as an input

pinMode(sensor2, INPUT);//set sensor 2 as an input

pinMode(Green, OUTPUT);//set the Green LED as an output

pinMode(Red, OUTPUT);//set the Red LED as an output

pinMode(Full, OUTPUT);//set the Blue/Full LED as an output


digitalWrite(Button,LOW);//set the button to "off", so when driver presses it, it goes "on"
```

```
 digitalWrite(sensor1,LOW);//set the sensor 1 to "off", so when car passes it, it goes "on"
 digitalWrite(sensor2,LOW);//set the sensor 2 to "off", so when car passes it, it goes "on"

 digitalWrite(Red,HIGH);//turn the Red LED on

}

 void loop() {
{// this is your main code, anything in this part runs repeatedly until its interupted or infinity.

 if (cars == fullPark)//if variable cars is the same as variable fullPark, then do this next part
{
 delay(100);//small delay for stability (milliseconds)
 digitalWrite(Red, LOW);//turn Red LED off
 digitalWrite(Full, HIGH);//turn Blue/Full LED on
 Serial.println(" Sorry, The car park is full"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 delay(500);//small delay for stability (milliseconds)
}
}

 if(cars < fullPark) //if variable cars is below the variable fullPark, then do this next part
 if(digitalRead(Button)==1) //read the button until it is pressed, 1 denotes HIGH or pressed
{
 Serial.println(" Raising The Barrier"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 myStepper.step(stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution
anticlockwise
 delay(500);//small delay for stability (milliseconds)
 barrier ++;//this increments the barrier variable in order to determine its possition
 digitalWrite(Red,LOW);//turn the Red LED off
```

```
 digitalWrite(Green,HIGH);//turn the Green LED on


 do
{
 delay(100);//small delay for stability (milliseconds)
 insense = digitalRead(sensor1); // check sensor 1 for car to pass by
}


 while (insense==1); //when a car triggers the first sensor, do the following
 cars ++; //increments the number of cars in the car park +1
 carsDailyTotal++; //increments the number of cars that have entered the car park since initial state
+1, different to cars it doesnt decrease
 digitalWrite(Red,HIGH);//turns Red LED on
 digitalWrite(Green,LOW); //turns Green LED off
 Serial.println(" Lowering The Barrier"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 myStepper.step(-stepsPerRevolution/4); //turns the stepper motor 1/4 of a full revolution
clockwise
 delay(500);//small delay for stability (milliseconds)
 barrier --;//this reduces the barrier variable in order to determine its possition
 Serial.print(" The No. In The Car Park Is Now "); //print the " message " to the serial monitor
 Serial.print(cars);//followed by the variable cars
 Serial.println(" cars"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
{
 outsense = digitalRead(sensor2); //see if the car is leaving the car park
 if (outsense == 1)//read sensor 2 until it is activated, 1 denotes HIGH or activated
{
 if (storedSensor2 ==0)//the stored variable for sensor 2
{
 cars--;//reduces the number of cars in the car park -1
```

```
  Serial.println(" One Car Has Left The Car Park"); //print the " message " to the serial monitor

  Serial.println(""); //insert a blank row to keep the serial monitor looking tidy

  Serial.print(" Cars In The Car Park Now "); //print the " message " to the serial monitor

  Serial.println(cars);//followed by the variable cars

  Serial.println(""); //insert a blank row to keep the serial monitor looking tidy

  digitalWrite (Full, LOW);//turn Blue/Full LED off

  digitalWrite (Red, HIGH);//turn Red LED on

 }

 }

 }


  storedSensor2 = outsense;//the stored variable for sensor 2 denotes if sensor 2 is HIGH or LOW


  if (Serial.available() >0)//waits for an input to serial port from the user and does nothing until an
input is recieved

 {

  char c = Serial.read();//read the input as a charachter


  if (c == 'q' || c =='Q')//if you press q or Q, initiallizes the following

 {

  Serial.println(); //insert a blank row to keep the serial monitor looking tidy

  Serial.println(" OPERATORS MENU"); //print the " message " to the serial monitor

  Serial.println(); //insert a blank row to keep the serial monitor looking tidy

  Serial.println(" Q = Service Menu"); //print the " message " to the serial monitor

  Serial.println(" R = Red Light ON & OFF"); //print the " message " to the serial monitor

  Serial.println(" G = Green Light ON & OFF"); //print the " message " to the serial monitor

  Serial.println(" B = Full Light ON & OFF"); //print the " message " to the serial monitor

  Serial.println(" + = Barrier UP"); //print the " message " to the serial monitor

  Serial.println(" - = Barrier DOWN"); //print the " message " to the serial monitor

  Serial.println(" O = Currend Lights Status Report"); //print the " message " to the serial monitor

  Serial.println(" Z = Currend No. of Cars in Car Park Report"); //print the " message " to the serial
monitor
```

```
  Serial.println(" H = Currend Barrier Status Report"); //print the " message " to the serial monitor

  Serial.println(" D = Total No. Cars Through Car Park Today Report"); //print the " message " to the
serial monitor

  Serial.println(" N = Change the number of Cars allowed in the Car Park"); //print the " message " to
the serial monitor

  Serial.println(); //insert a blank row to keep the serial monitor looking tidy

  delay(1000);  //small delay for stability (milliseconds)

 }


 if (c == 'b'||c=='B')//if you press b or B, initiallizes the following

 {

 if (FullOn == HIGH)//asks itself if the Blue/Full LED is on or off

 {

 FullOn = LOW;//if its off do this

 digitalWrite (Full, FullOn);//turn Blue/Full LED on

 Serial.println(" Blue / Full Light OFF"); //print the " message " to the serial monitor

 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy

 }

  else

 {

 FullOn = HIGH;//unless, if its on do this

 digitalWrite (Full, FullOn);//turn Blue/Full LED off

 Serial.println(" Blue / Full Light ON"); //print the " message " to the serial monitor

 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy

 }

 }


 if (c == 'g'||c=='G')//if you press g or G, initiallizes the following

 {

 if (GreenOn == HIGH)//asks itself if the Green LED is on or off

 {

 GreenOn = LOW;//if its off do this
```

```
 digitalWrite (Green, GreenOn);//turn Green LED on
 Serial.println(" Green Light OFF"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else
{
 GreenOn = HIGH;//unless, if its on do this
 digitalWrite (Green, GreenOn);//turn Green LED off
 Serial.println(" Green ON"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
}
}

 if (c == 'r'||c=='R')//if you press r or R, initiallizes the following
{
 if (RedOn == HIGH)//asks itself if the Red LED is on or off
{
 RedOn = LOW;//if its off do this
 digitalWrite (Red, RedOn);//turn Red LED on
 Serial.println(" Red Light OFF"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
}
 else
{
 RedOn = HIGH;//unless, if its on do this
 digitalWrite (Red, RedOn);//turn Red LED off
 Serial.println(" Red ON"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

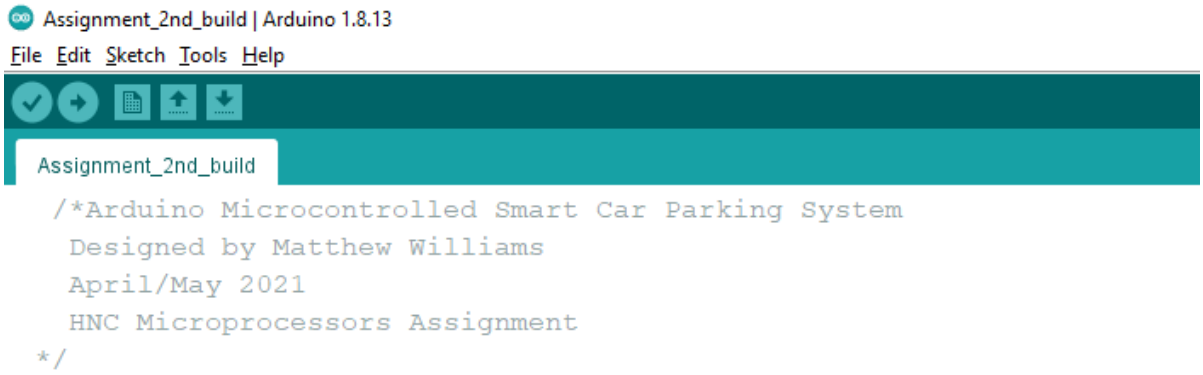
```
 if (c == '+')//if you press +, initiallizes the following
{
 myStepper.step(stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution
anticlockwise
 Serial.println(" Barrier UP"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}

 if (c == '-')//if you press -, initiallizes the following
{
 myStepper.step(-stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution clockwise
 Serial.println(" Barrier DOWN"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}

 if (c == 'o'||c=='O')//if you press o or O, initiallizes the following
{
 if (digitalRead (Red) == HIGH)//looks to see if the Red LED is on?
{
 Serial.println(" Red ON"); //print the " message " to the serial monitor
}
 else if (digitalRead (Red) ==LOW)//looks to see if the Red LED is off?
{
 Serial.println(" Red OFF"); //print the " message " to the serial monitor
}

 if (digitalRead (Full) == HIGH)//looks to see if the Blue/Full LED is on?
{
 Serial.println(" Blue ON"); //print the " message " to the serial monitor
}
 else if (digitalRead (Full) ==LOW)//looks to see if the Blue/Full LED is off?
```

```
{
 Serial.println(" Blue OFF"); //print the " message " to the serial monitor
}

 if (digitalRead (Green) == HIGH)//looks to see if the Green LED is on?
{
 Serial.println(" Green ON"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else if (digitalRead (Green) ==LOW)//read the Green pin to see if its LOW
{
 Serial.println(" Green OFF"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}

 if (c =='z'||c=='Z')//if you press z or Z, initiallizes the following
{
 Serial.print(" No. Cars in Car Park "); //print the " message " to the serial monitor
 Serial.println(cars);//followed by the variable cars
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}

{
 if (c =='d'||c=='D')//if you press d or D, initiallizes the following
{
 Serial.print(" Total No. Cars Through Car Park Today "); //print the " message " to the serial monitor
 Serial.println(carsDailyTotal);//followed by the variable carsDailyTotal
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

```
{
 if (c == 'h'||c=='H')//if you press h or H, initiallizes the following
{
 if (barrier == 1)//reads variable barrier to see if it is 1 or 0
{
 Serial.println(" Barrier UP"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else if (barrier == 0)//reads variable barrier to see if it is 1 or 0
{
 Serial.println(" Barrier DOWN"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
}
 if (c == 'n'||c=='N')//if you press n or N, user can change the maximum number of cars allowed into the car park
{
 Serial.println(" Enter a number between 10 and 255, to change max number of cars in car park: ");//print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 while (Serial.available () ==0) {}//waits for a reply from the user and does nothing until an input is recieved
 newFullPark = Serial.parseInt();//read the characters inputed, and put them into the variable newFullPark
 if ((newFullPark >= 10) && (newFullPark <= 255))//sets the parameters the user needs to input
{
 Serial.print(" New number of cars in car park is: "); //print the " message " to the serial monitor
 Serial.println(newFullPark);//followed by the new full car park total defined by  the user
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 fullPark = newFullPark; //the new maximum cars changes from the variable fullPark to the variable newFullPark
 if (fullPark > cars) //if fullPark is greater than the number of cars in the car park
```

```
{
 digitalWrite(Full, LOW);//turn the Blu/Full light off
}
}
 else
{
 Serial.print(" Invalid input, please pick a number between 10 and 255"); //print the " message " to the serial monitor
}
}
}
}
```

The Structure of the code



The above is a little description of what the code is and who the author is.

```
#include <Stepper.h>

int Button = 2;  //assign the push button to Pin 2
int sensor1 = 3; //assign the first sensor to Pin 3
int sensor2 = 4; //assign the second sensor to Pin 4
int Green = 5;   //assign the Green LED to Pin 5
int Red = 6;     //assign the Red LED to Pin 6
int Full = 7;    //assign the Blue/Full LED to Pin 7
int stepsPerRevolution = 2048; //the number of steps per on revolution of the stepper motor
int motSpeed=10;  //set the speed of the stepper motor at 10 rpm:
int fullPark =2;  //the number of cars that will make the car park full
int newFullPark= 0; //the variable that changes when the user defines how many cars the car park can hold: between 10 & 255
int cars = 0;     //the variable that stores the number of cars currently in the cat oark at any given time
int carsDailyTotal=0; //the variable that constantly adds the number of cars entering the car park (not taking into account cars leaving)
int barrier=0;  //variable assigned to calculate the current barrier possition
int insense=0;  //defying the 1st sensor, Entry Sensor
int outsense = 0;  //defying the 2nd sensor, Exit Sensor
int storedSensor2 =1; //defines the state of Sensor 2, Exit Sensor
int FullOn =LOW;  //variable for manualy switching Blue/Full LED light on/off
int GreenOn =LOW; //variable for manualy switching Green LED light on/off
int RedOn =LOW;   //variable for manualy switching Red LED light on/off

Stepper myStepper(stepsPerRevolution, 8,10,9,11); //initialize the stepper library on pins 8 through 11 in correct sequence
```

In this part of the code we have declared all the Pins that will be utilised from the Arduino. As well as all the variables that are needed to be declared before our main program will work. We have also included in this part of the code a library that we have downloaded that give us all the Stepper Motor files required to run the device in our program.

```
 void setup()
{  //this is the setup code, this code runs once only at the start of your program.

 myStepper.setSpeed(motSpeed); // initialize the set speed of the stepper motor from above
 Serial.begin(9600);  //initialize the serial port at 9600 baud rate
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 Serial.println(" Welcome to Matthew's Car Park"); //print a welcome mesage on initial power up
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 Serial.print(" The Max number of cars allowed into the Car Park today is "); //print this message
 Serial.print(fullPark); //takes the preset number from the variable
 Serial.println(" cars!");//then prints this message
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 Serial.println(" To change this number or to access the OPERATORS MENU enter 'Q' into the Serial Monitor at any time"); //print a prompt to enter the OPERATORS MENU
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy


 pinMode(Button, INPUT);//set the button as an input
 pinMode(sensor1, INPUT);//set sensor 1 as an input
 pinMode(sensor2, INPUT);//set sensor 2 as an input
 pinMode(Green, OUTPUT);//set the Green LED as an output
 pinMode(Red, OUTPUT);//set the Red LED as an output
 pinMode(Full, OUTPUT);//set the Blue/Full LED as an output

 digitalWrite(Button,LOW);//set the button to "off", so when driver presses it, it goes "on"
 digitalWrite(sensor1,LOW);//set the sensor 1 to "off", so when car passes it, it goes "on"
 digitalWrite(sensor2,LOW);//set the sensor 2 to "off", so when car passes it, it goes "on"

 digitalWrite(Red,HIGH);//turn the Red LED on

}
```

The next part of the code is the void setup ()

This part of the Arduino code gets utilised once at the start of the program. Code in this part is used to set the initial state of all peripherals that have been used in the program. For example in this code, it initialises the communication between the Arduino and the serial monitor, sends some txt to print on the serial monitor, sets buttons and sensors to low so when activated they send a signal HIGH to the microprocessor and also turns on the Red LED light.

```
void loop() {
{// this is your main code, anything in this part runs repeatedly until its interupted or infinity.

 if (cars == fullPark)//if variable cars is the same as variable fullPark, then do this next part
{
 delay(100);//small delay for stability (milliseconds)
 digitalWrite(Red, LOW);//turn Red LED off
 digitalWrite(Full, HIGH);//turn Blue/Full LED on
 Serial.println(" Sorry, The car park is full"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 delay(500);//small delay for stability (milliseconds)
}
}
```

Once the void loop () has completed its job, the main part of the program, the part that runs over and over until infinity or something causes an interrupt kicks in. This part of the code is called the void loop ().

Here we ask the question, if the number in the 'cars' variable is the same as the number in the 'fullPark' variable the car park is full and the Blue/Full LED comes on and a message is printed to the serial monitor.

```
 if(cars < fullPark) //if variable cars is below the variable fullPark, then do this next part
 if(digitalRead(Button)==1) //read the button until it is pressed, 1 denotes HIGH or pressed
{
 Serial.println(" Raising The Barrier"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 myStepper.step(stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution anticlockwise
 delay(500);//small delay for stability (milliseconds)
 barrier ++;//this increments the barrier variable in order to determine its possition
 digitalWrite(Red,LOW);//turn the Red LED off
 digitalWrite(Green,HIGH);//turn the Green LED on
```

Here we ask the question, if the number in the 'cars' variable is below the number in the 'fullPark' variable the car park is not full and cars are allowed into the car park.

Thereafter, if the button is pressed, the barrier lifts, the Red LED goes off and the Green LED comes on. Along with some txt being printed to the Serial Monitor.

```
 do
{
 delay(100);//small delay for stability (milliseconds)
 insense = digitalRead(sensor1); // check sensor 1 for car to pass by
}

 while (insense==1); //when a car triggers the first sensor, do the following
 cars ++; //increments the number of cars in the car park +1
 carsDailyTotal++; //increments the number of cars that have entered the car park since initial state +1, different to cars it doesnt decrease
 digitalWrite(Red,HIGH);//turns Red LED on
 digitalWrite(Green,LOW); //turns Green LED off
 Serial.println(" Lowering The Barrier"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 myStepper.step(-stepsPerRevolution/4); //turns the stepper motor 1/4 of a full revolution clockwise
 delay(500);//small delay for stability (milliseconds)
 barrier --;//this reduces the barrier variable in order to determine its possition
 Serial.print(" The No. In The Car Park Is Now "); //print the " message " to the serial monitor
 Serial.print(cars);//followed by the variable cars
 Serial.println(" cars"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
```

This ‘do while’ loop tells the program to read the sensor going into the car park, while incrementing the ‘cars’ variable, lowering the barrier once a car passes by. Turns the Green LED off and Red LED light on and prints some txt to the Serial Monitor.

```
{
 outsense = digitalRead(sensor2); //see if the car is leaving the car park
 if (outsense == 1)//read sensor 2 until it is activated, 1 denotes HIGH or activated
{
 if (storedSensor2 ==0)//the stored variable for sensor 2
{
 cars--;//reduces the number of cars in the car park -1
 Serial.println(" One Car Has Left The Car Park"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 Serial.print(" Cars In The Car Park Now "); //print the " message " to the serial monitor
 Serial.println(cars);//followed by the variable cars
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 digitalWrite (Full, LOW);//turn Blue/Full LED off
 digitalWrite (Red, HIGH);//turn Red LED on
}
}
}

 storedSensor2 = outsense;//the stored variable for sensor 2 denotes if sensor 2 is HIGH or LOW
```

This code reads the state of sensor 2 and when a car goes past, decreases the variable ‘cars’, turns the Blue/Full LED off and puts the Red LED on. There is also some text printed to the serial monitor.

```
 if (Serial.available() >0)//waits for an input to serial port from the user and does nothing until an input is recieved
{
 char c = Serial.read();//read the input as a charachter

 if (c == 'q' || c =='Q')//if you press q or Q, initiallizes the following
{
 Serial.println(); //insert a blank row to keep the serial monitor looking tidy
 Serial.println(" OPERATORS MENU"); //print the " message " to the serial monitor
 Serial.println(); //insert a blank row to keep the serial monitor looking tidy
 Serial.println(" Q = Service Menu"); //print the " message " to the serial monitor
 Serial.println(" R = Red Light ON & OFF"); //print the " message " to the serial monitor
 Serial.println(" G = Green Light ON & OFF"); //print the " message " to the serial monitor
 Serial.println(" B = Full Light ON & OFF"); //print the " message " to the serial monitor
 Serial.println(" + = Barrier UP"); //print the " message " to the serial monitor
 Serial.println(" - = Barrier DOWN"); //print the " message " to the serial monitor
 Serial.println(" O = Currend Lights Status Report"); //print the " message " to the serial monitor
 Serial.println(" Z = Currend No. of Cars in Car Park Report"); //print the " message " to the serial monitor
 Serial.println(" H = Currend Barrier Status Report"); //print the " message " to the serial monitor
 Serial.println(" D = Total No. Cars Through Car Park Today Report"); //print the " message " to the serial monitor
 Serial.println(" N = Change the number of Cars allowed in the Car Park"); //print the " message " to the serial monitor
 Serial.println(); //insert a blank row to keep the serial monitor looking tidy
 delay(1000);  //small delay for stability (milliseconds)
}
```

The bottom part of the code gives us the code for manually controlling the peripherals from the Serial Monitor. It basically says, if a character is entered into the monitor, do this.
In this screenshot we see that “Q” initialises the Operators Menu.

```
 if (c == 'b'||c=='B')//if you press b or B, initiallizes the following
{
 if (FullOn == HIGH)//asks itself if the Blue/Full LED is on or off
{
 FullOn = LOW;//if its off do this
 digitalWrite (Full, FullOn);//turn Blue/Full LED on
 Serial.println(" Blue / Full Light OFF"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
  else
{
 FullOn = HIGH;//unless, if its on do this
 digitalWrite (Full, FullOn);//turn Blue/Full LED off
 Serial.println(" Blue / Full Light ON"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

Pressing “B” switches the Blue/Full LED on.

```
 if (c == 'g'||c=='G')//if you press g or G, initiallizes the following
{
 if (GreenOn == HIGH)//asks itself if the Green LED is on or off
{
 GreenOn = LOW;//if its off do this
 digitalWrite (Green, GreenOn);//turn Green LED on
 Serial.println(" Green Light OFF"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else
{
 GreenOn = HIGH;//unless, if its on do this
 digitalWrite (Green, GreenOn);//turn Green LED off
 Serial.println(" Green ON"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
}
}
```

Pressing “G” switches the Green LED on.

```
 if (c == 'r'||c=='R')//if you press r or R, initiallizes the following
{
 if (RedOn == HIGH)//asks itself if the Red LED is on or off
{
 RedOn = LOW;//if its off do this
 digitalWrite (Red, RedOn);//turn Red LED on
 Serial.println(" Red Light OFF"); //print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
}
 else
{
 RedOn = HIGH;//unless, if its on do this
 digitalWrite (Red, RedOn);//turn Red LED off
 Serial.println(" Red ON"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

Pressing “R” switches the Red LED on.

```
 if (c == '+')//if you press +, initiallizes the following
{
 myStepper.step(stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution anticlockwise
 Serial.println(" Barrier UP"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
```

Pressing “+” raises the barrier.

```
 if (c == '-')//if you press -, initiallizes the following
{
 myStepper.step(-stepsPerRevolution/4);//turns the stepper motor 1/4 of a full revolution clockwise
 Serial.println(" Barrier DOWN"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
```

Pressing “-” lowers the barrier.

```
 if (c == 'o'||c=='O')//if you press o or O, initiallizes the following
{
 if (digitalRead (Red) == HIGH)//looks to see if the Red LED is on?
{
 Serial.println(" Red ON"); //print the " message " to the serial monitor
}
 else if (digitalRead (Red) ==LOW)//looks to see if the Red LED is off?
{
 Serial.println(" Red OFF"); //print the " message " to the serial monitor
}

 if (digitalRead (Full) == HIGH)//looks to see if the Blue/Full LED is on?
{
 Serial.println(" Blue ON"); //print the " message " to the serial monitor
}
 else if (digitalRead (Full) ==LOW)//looks to see if the Blue/Full LED is off?
{
 Serial.println(" Blue OFF"); //print the " message " to the serial monitor
}

 if (digitalRead (Green) == HIGH)//looks to see if the Green LED is on?
{
 Serial.println(" Green ON"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else if (digitalRead (Green) ==LOW)//read the Green pin to see if its LOW
{
 Serial.println(" Green OFF"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

Pressing “O” Gives a status report on the state of the three LED’s.

```
 if (c =='z'||c=='Z')//if you press z or Z, initiallizes the following
{
 Serial.print(" No. Cars in Car Park "); //print the " message " to the serial monitor
 Serial.println(cars);//followed by the variable cars
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}

{
```

Pressing “Z” Gives a status report on the number of cars currently in the Car Park.

```
{
 if (c =='d'||c=='D')//if you press d or D, initiallizes the following
{
 Serial.print(" Total No. Cars Through Car Park Today "); //print the " message " to the serial monitor
 Serial.println(carsDailyTotal);//followed by the variable carsDailyTotal
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
```

Pressing “D” Gives a report on the number of cars passed through the Car Park since the program started.

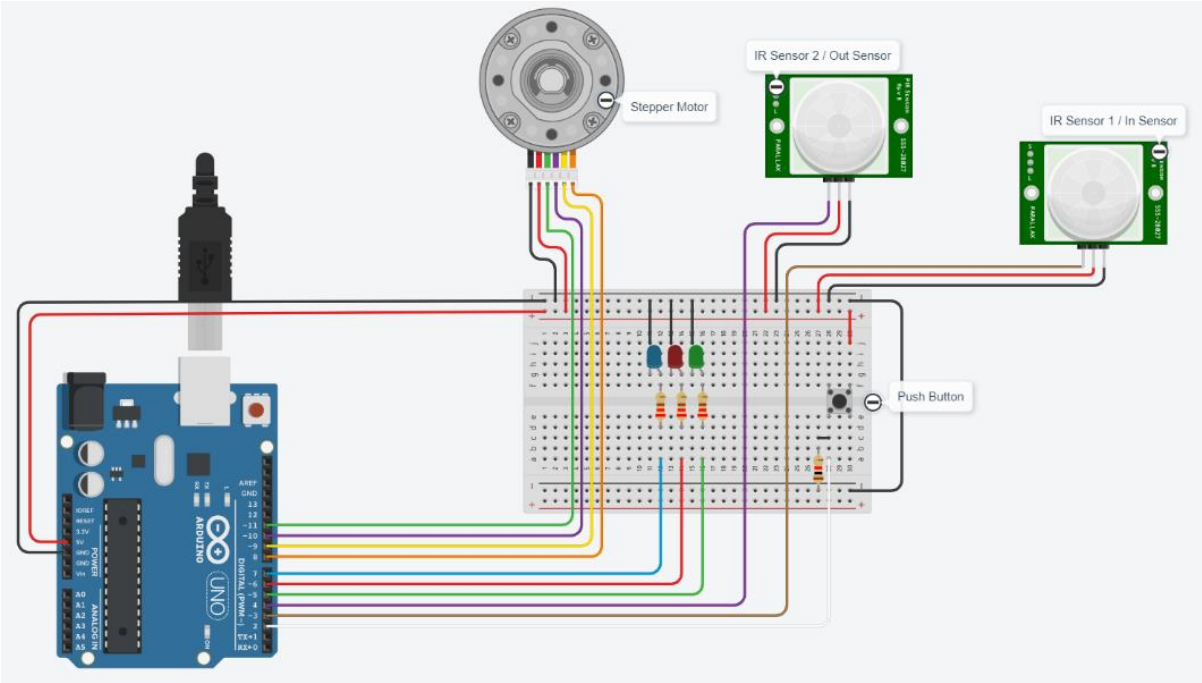
```
{
 if (c == 'h'||c=='H')//if you press h or H, initiallizes the following
{
 if (barrier == 1)//reads variable barrier to see if it is 1 or 0
{
 Serial.println(" Barrier UP"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
 else if (barrier == 0)//reads variable barrier to see if it is 1 or 0
{
 Serial.println(" Barrier DOWN"); //print the " message " to the serial monitor
 Serial.println("");  //insert a blank row to keep the serial monitor looking tidy
}
}
}
```

Pressing “H” Gives a status report on the state of the three barrier.

```
 if (c == 'n'||c=='N')//if you press n or N, user can change the maximum number of cars allowed into the car park
{
 Serial.println(" Enter a number between 10 and 255, to change max number of cars in car park: ");//print the " message " to the serial monitor
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 while (Serial.available () ==0) {}//waits for a reply from the user and does nothing until an input is recieved
 newFullPark = Serial.parseInt();//read the characters inputed, and put them into the variable newFullPark
 if ((newFullPark >= 10) && (newFullPark <= 255))//sets the parameters the user needs to input
{
 Serial.print(" New number of cars in car park is: "); //print the " message " to the serial monitor
 Serial.println(newFullPark);//followed by the new full car park total defined by  the user
 Serial.println(""); //insert a blank row to keep the serial monitor looking tidy
 fullPark = newFullPark; //the new maximum cars changes from the variable fullPark to the variable newFullPark
 if (fullPark > cars) //if fullPark is greater than the number of cars in the car park
{
 digitalWrite(Full, LOW);//turn the Blu/Full light off
}
}
 else
{
 Serial.print(" Invalid input, please pick a number between 10 and 255"); //print the " message " to the serial monitor
}
}
}
}
```

Pressing “N” Gives the operator the ability to change the maximum number of cars allowed into the car park at any one time.

Circuit Diagram



Test Plan

The states that require testing are:

1) Power On
2) Empty Car Park
3) Car at the Barrier when car park is not full and the button is pressed
4) Car goes into the car park and passes sensor 1
5) Car goes into the car park and passes sensor 1 when there is only one space left in the car park
6) Car at the barrier when car park is full and button is pressed
7) Car leaves the car park when the car park is not full
8) Car leaves the car park when the car park is full
9) Car leaves the car park at the same time as a car enters the car park
10) What happens when you press “Q” or “q”
11) What happens when you press “R” or “r”
12) What happens when you press “G” or “g”
13) What happens when you press “B” or “b”
14) What happens when you press “+”
15) What happens when you press “-”
16) What happens if you press "RGB" or "rgb" at the same time
17) What happens when you press “O” or “o”
18) What happens when you press “Z” or “z”
19) What happens when you press “H” or “h”
20) What happens when you press “D” or “d”
21) What happens when you press “N” or “n”

The program will be tested in two ways.

- A full replica will be built to simulate the car park (recorded with YouTube link)
- The program will be tested in small parts as described above to determine if actual behaviour of the program behaves in the expected way. Do inputs have the desired outputs that

Full replica simulation

https://www.youtube.com/watch?v=olUVqcp2CH4

Test Results

Arduino Microcontroller Smart Car Park System Testing Results		
State to test	**Predicted Outcome**	**Results**
Power On	Red LED - On	All predicted outcomes were observed without any unexpected outcomes
	Barrier - Down	
	Welcome message on Serial Monitor	
Empty Car Park	Red LED - On	All predicted outcomes were observed without any unexpected outcomes
	Barrier - Down	
Car at the Barrier when car park is not full and the button is pressed	Barrier - Raises	All predicted outcomes were observed without any unexpected outcomes
	Red LED - Off	
	Green LED - ON	
	Barrier Raising on Serial Monitor	
Car goes into the car park and passes sensor 1	Green LED - Off	The predicted outcomes are observed, however all at the same time and not in sequence
	Red LED - On	
	Barrier - Lowers	
	Barrier Lowering on Serial Monitor	
	No. Cars in the car park on the Serial Monitor +1	
Car goes into the car park and passes sensor 1 when there is only one space left in the car park	Green LED - Off	All predicted outcomes were observed without any unexpected outcomes
	Red LED - On	
	Barrier - Lowers	
	Blue / Full LED - On	
	Barrier Lowering on Serial Monitor	
	No. Cars in the car park on the Serial Monitor	
	The Car Park is Full on Serial Monitor	

Car at the barrier when car park is full and button is pressed	Blue / Full LED - Stays On	All predicted outcomes were observed without any unexpected outcomes
	Barrier Stays Down	
Car leaves the car park when the car park is not full	No. Cars in the car park on the Serial Monitor -1	All predicted outcomes were observed without any unexpected outcomes
Car leaves the car park when the car park is full	Blue / Full LED - Off	All predicted outcomes were observed without any unexpected outcomes
	Red LED - On	
	No. Cars in the car park on the Serial Monitor -1	
	Cars can then enter	
Car leaves the car park at the same time as a car enters the car park	No. Cars in the car park on the Serial Monitor -1	The program does not allow for a car to enter and leave at the same time
	Barrier - Raises	
	Red LED - Off	
	Green LED - ON	
	Barrier Raising on Serial Monitor	
What happens when you press “Q” or “q”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Operators Menu on Serial Monitor	
What happens when you press “R” or “r”	Both Upper and Lower case commands work	The predicted outcomes are observed, however if the Red LED is On when first inputted, you must type it again to make it go off
	Red LED goes off if on and vice versa	
	Red LED status displayed on Serial Monitor	
What happens when you press “G” or “g”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Green LED goes off if on and vice versa	
	Green LED status displayed on Serial Monitor	
What happens when you press “B” or “b”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Blue / Full LED goes off if on and vice versa	
	Blue / Full LED status displayed on Serial Monitor	
What happens when you press “+”	Barrier Raises	All predicted outcomes were observed without any unexpected outcomes
	Barrier Up on Serial Monitor	
What happens when you press “-”	Barrier Lowers	All predicted outcomes were observed without any unexpected outcomes
	Barrier Down on Serial Monitor	
What happens if you press "RGB" or "rgb"	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Green, Red, Blue LEDs goes off if on and vice versa	
	Green, Red, Blue LEDs status displayed on Serial Monitor	

What happens when you press “O” or “o”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Green, Red, Blue LEDs status displayed on Serial Monitor	
What happens when you press “Z” or “z”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Current number of cars in the car park displayed on Serial Monitor	
What happens when you press “H” or “h”	Both Upper and Lower case commands work	The status of the barrier is recorded, but only if it is not in a state. i.e. if the button is pressed and it is reading sensor 1
	Current status of the barrier displayed on Serial Monitor	
What happens when you press “D” or “d”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Number of cars through the car park today displayed on Serial Monitor	
What happens when you press “N” or “n”	Both Upper and Lower case commands work	All predicted outcomes were observed without any unexpected outcomes
	Question is asked, how many cars do you want in the Car Park?	
	You enter a number between 10 - 255	
	Serial Monitor confirms the new number of cars allowed in the car park	
	Error Message is displayed if the number is invalid	
	The new Maximum number of cars is stored in the program	

Key to Results
All outputs as expected
Mostly as expected
Program not responding as expected

Conclusions from Testing

On the whole, the software is functioning as designed and the inputs are causing the expected behaviour.

The four states that did throw up some unexpected outcomes were:

- When a car goes into the car park and passes sensor 1
- When a car leaves the car park at the same time as a car enters the car park
- When you press “R” or “r” to switch the Red LED on & off
- When you press “H” or “h” to see the status of the barrier

What can be done to overcome these discrepancies?

1) When a car goes into the car park and passes sensor 1
   The main issue with this state was that all the outputs occurred at the same time, and not in sequence. This could easily be overcome by putting in a small delay in between each output.

2) When a car leaves the car park at the same time as a car enters the car park
   The program does not allow for a car to enter and leave the car park at the same time. There could be several reasons for this, however the most probable cause is that the Arduino controller is only able to perform a single task at any one time. Therefore, to get the desired function of a car leaving at the same time a car enters, we would probably need to create and implement an interrupt sub-routine into the code.

3) When you press “R” or “r” to switch the Red LED on & off
   If the Red LED is on when you first input an “R” or “r”, you must type it in again to make it go off/on. This “blip” in the code is possibly due to the default setting of the Red LED is on. If the command to switch the LED on was put in after the button is pressed then this would possibly not happen.

4) When you press “H” or “h” to see the status of the barrier
   The status of the barrier is recorded, but only if it is not in a state. This should be overcome by reading the position during every function in then code.

On the whole, the functionality of the software is good with only minor discrepancies and I believe the Design Brief has been met.