



Microprocessors, computers and peripherals

CAR PARK CONTROL

MICHAEL JONES

Table of Contents

Introduction and assignment outline.....	3
Algorithm for car park	5
Flowchart	6
State Diagram	7
Writing the Program	8
Circuit Diagram	23
Testing	24
Final copy of the program	27
Conclusion.....	40
References	41

Introduction and assignment outline

For this assignment, I have been tasked with writing a code for a car park. The following parameters must be met in the program.

1. An operator sets the maximum available number of parking spaces in the range of 10 to 255.
2. Cars arriving at the entrance are initially prevented from entry by a barrier and 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 following inputs for the Arduino must also be correct.

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 degrees 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

Before attempting to write my code, I have decided to write an algorithm as I believe this will be the best place to start. In simple terms, 'an algorithm is the list of instructions and rules that a computer needs to do to complete a task' (Think Automation, 2021). The algorithm is usually written in plain English. This is so it is easy for anyone to follow and is clearly distinguishable from a piece of code.

Algorithm for car park

Set *push button* to pin 2

Set *sensor 1* to pin 3, and *sensor 2* to pin 4

Set *Green LED* to pin 5, *Red LED* to pin 6 and *Full sign* to pin 7

Set *stepper motor* to pins 8, 9, 10 and 11 and number of revolutions to 200

Set max number of cars to 10

Declare variables for number of *cars*, *change* and *barrier*

Turn the Red LED on

Check if the car park is full

If the car park is not full

 Check if the *push button* is pressed.

 If the button is pressed, raise the *barrier*, turn the *Red LED* off and turn the *Green LED* on

 While the barrier is raised read *sensor 1*

 If a car passes *sensor 1*, add 1 to the count, lower the barrier, turn *Green LED* off and *Red* on

 Check *sensor 2*

 If a car passes the *sensor 2*, subtract 1 from the count

If the car park is full

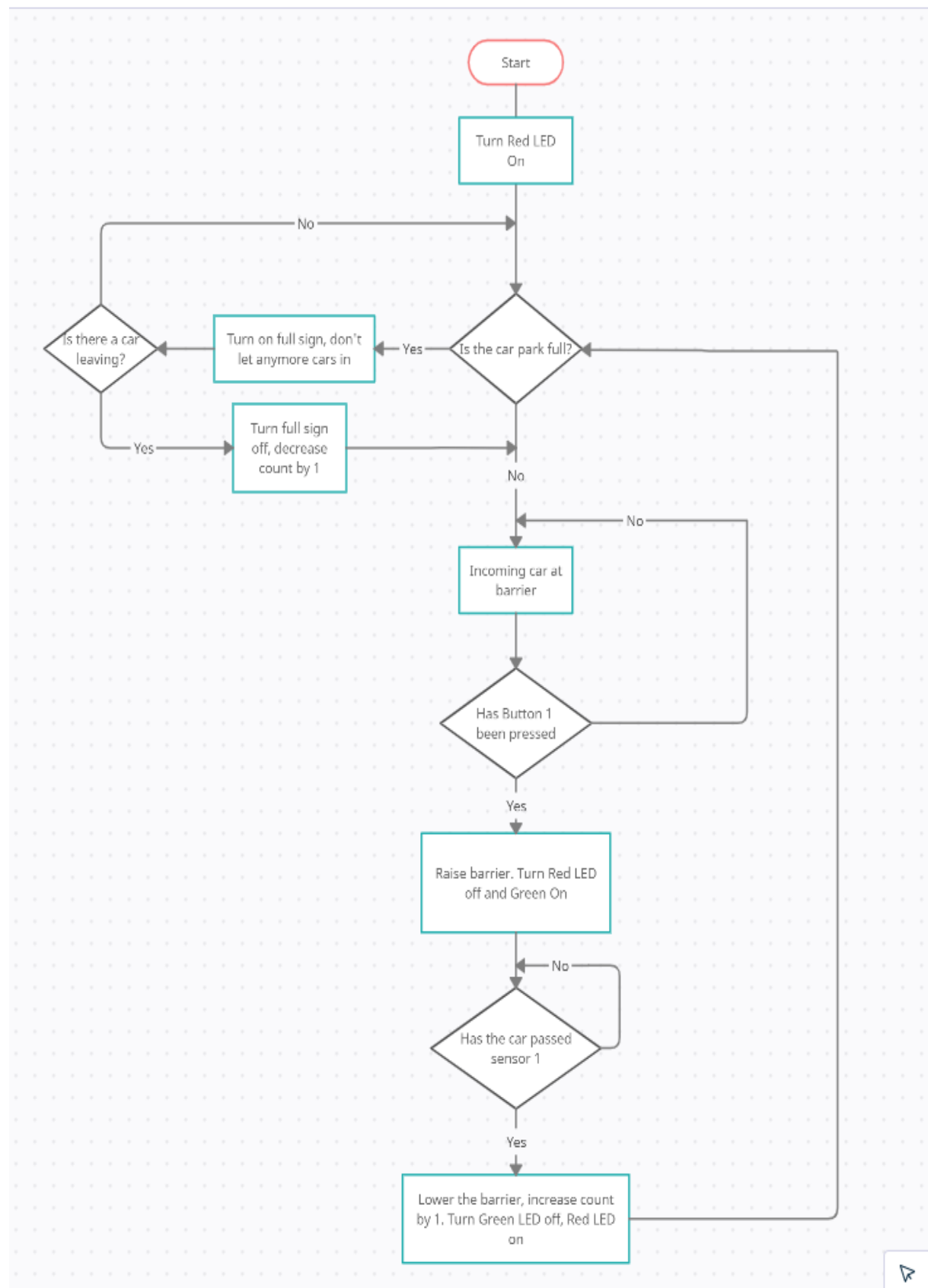
 Turn on the *Full sign*, and do not let any more cars enter

 If a car passes the *sensor 2*, subtract 1 from the count and turn off the full sign

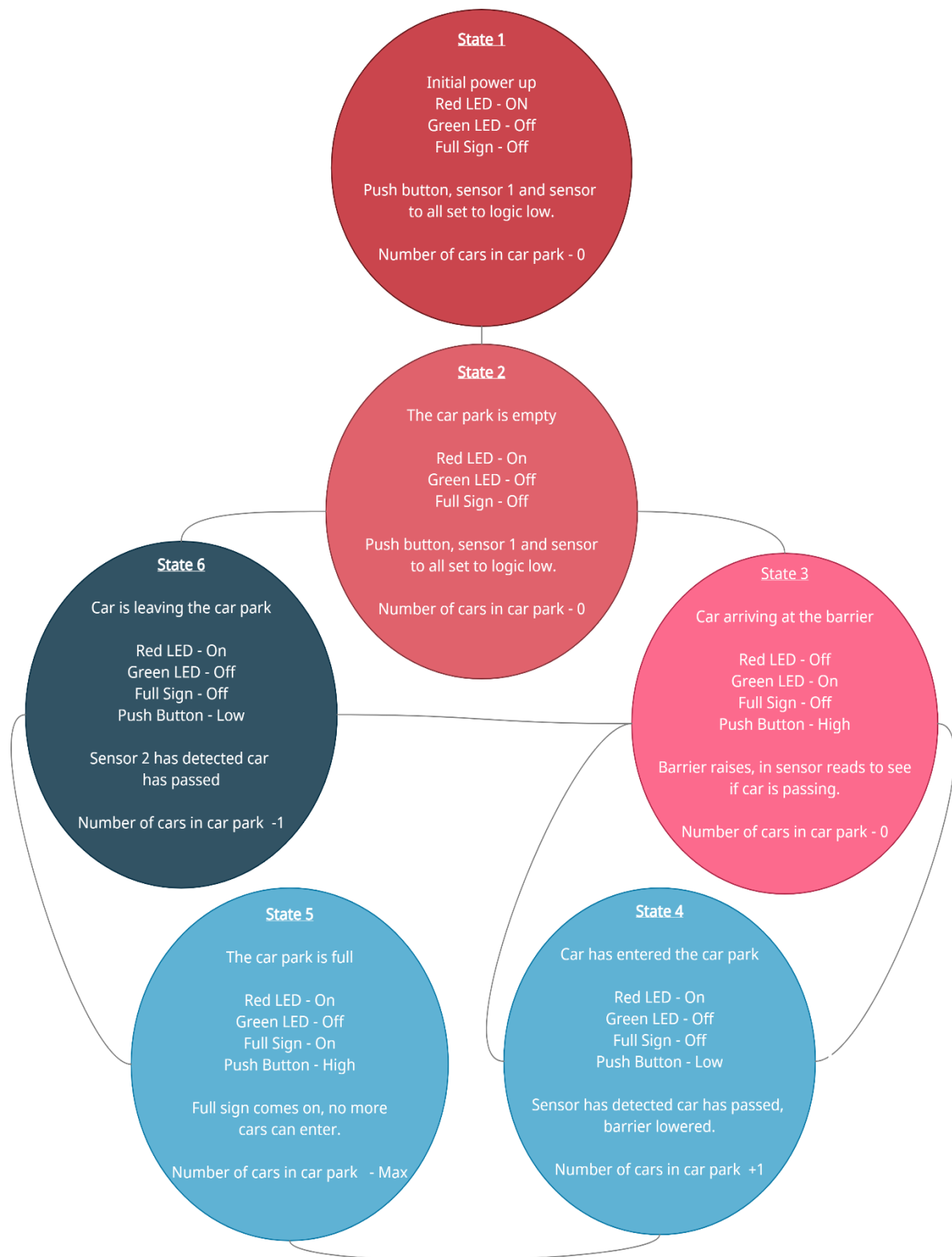
Enter Manual menu

From the above, we can see that the algorithm to start is very basic. It meets the points in the brief, and from here we can start to develop our program. On the next page is a flowchart which mirrors what I have written in the algorithm.

Flowchart



State Diagram



Writing the Program

The first thing I want to do when writing my program, is decide what variables I want to use, declare them at the start of the program and decide if any of the variables need to be assigned to pins. If they do, I can assign them at the start of the program using the 'const int' command. I will also need to include the stepper motor library which is labelled <Stepper.h> using the #include command. The stepper motor will be set to 200 steps per revolution because it asks for this in the brief. Below is the first part of my code.

```
#include <Stepper.h>

const int button = 2;    // assign the push button to pin 2
const int sensor1 = 3;   // assign sensor 1 to pin 3
const int sensor2 = 4;   // assign sensor 2 to pin 4
const int green = 5;     // assign the green LED to pin 5
const int red = 6;       // assign the red LED to pin 6
const int FULL_SIGN = 7; // assign the full sign display to pin 7

int redOn = HIGH;        // will be used to dictate the state of the red LED. Will start HIGH.
int greenOn = LOW;       // will be used to dictate the state of the green LED. Will start LOW.
int blueOn = LOW;        // will be used to dictate the state of the full sign. Will start LOW.
int change;              // variable that will be used to change the number of cars in the car park whilst the program is running
const int stepsPerRevolution = 200; // change this to fit the number of steps per revolution for your motor
int barrier;              // will be used to view the status of the barrier later in the program
unsigned int fullpark = 10; // max amount of cars
int cars = 0;             // holds number of cars currently in car park
int x = 0;                // will be used to check the state of the sensor 2
|
int outsense = 0;         // state of the outsense
int storedSensor2 = 1;    // state of stored sensor 2

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

Now that we have declared all variables and assigned the appropriate pins, we need to write our initial setup. Whatever we include in the setup will execute as soon as the Arduino is turned on.

```
void setup() {
  myStepper.setSpeed(60); // set the speed at 60 rpm:
  Serial.begin(9600);      // initialize the serial port:
  Serial.println("Press q to enter service menu."); // print the message to the serial monitor
  Serial.println("Press n to enter max number of cars allowed in garage."); // print the message to the serial monitor
  pinMode(button, INPUT); // configure the push button pin to be a digital input
  pinMode(sensor1, INPUT); // configure the sesnor 1 pin to be a digital input
  pinMode(sensor2, INPUT); // configure the sensor 2 pin to be a digital input
  pinMode(green, OUTPUT); // configure the green LED pin to be a digital output
  pinMode(red, OUTPUT); //configure the green LED pin to be a digital output
  pinMode(FULL_SIGN, OUTPUT); //configure the green LED pin to be a digital output
  digitalWrite(red, redOn); // Tell the red LED to adopt the status of redOn
  digitalWrite(green, greenOn); // Tell the green LED to adopt the status of greenOn
  digitalWrite(FULL_SIGN, blueOn); // Tell the full sign to adopt the status of blueOn
  digitalWrite(button, LOW); // button to start at logic LOW
  digitalWrite(sensor1, LOW); // sensor 1 to start at logic LOW
  digitalWrite(sensor2, LOW); // sensor 2 to start at logic LOW
}
```

As you can see, when we start the Arduino, the first thing that will happen is the stepper motor will be set to move at 60rpm. The serial monitor will also be initialised and will print the first two lines of code starting with 'Serial.println.' I have set the button and the two sensors as inputs, and the LED's and full sign as outputs. The LED's have also been told to adopt the state of redOn, greenOn and blueOn respectively. When we set the variables at the beginning, we set redOn to HIGH, which means that the red LED will start with the light illuminated. I have done this because it asks to do so in the assignment brief.

The next thing we need to do is to start a loop and execute the main body of our program. I will start by using the 'if' statement to check if the car park is full. The first part of the code attached below details what should happen if the car park is not full.

```
if (cars < fullpark){ // execute the following code if the car park is not full

if(digitalRead(button)==1){ // execute the following code if the button is pressed
    myStepper.step(stepsPerRevolution/4); // turn the motor 90 degrees clockwise
    digitalWrite(red, HIGH); // turn on the red LED
    Serial.print("Raising barrier\n"); // print the message 'raising barrier'
    delay(1000);
    digitalWrite(green, HIGH); // Turn green LED on
    digitalWrite(red, LOW); // Turn red LED off
    barrier == 1; // Set the barrier status to 1

do
{
    delay(50); // wait for sensors to stablize
    x = digitalRead(sensor1); // check sensor1
} while (x == 1); // when the car passess the sensor, do the following
    Serial.print("Lowering barrier\n"); // print that the barrier is lowering
    myStepper.step(-stepsPerRevolution/4); // rotate the motor 90 degrees anti clockwise
    digitalWrite(green, LOW); // turn green LED off
    digitalWrite(red, HIGH); // turn red LED on
    cars ++; // increase the number of cars in the car park by 1
    delay(500);
    Serial.print("The count is now "); // print the new value of the cars in the car park
    Serial.println(cars);
    barrier == 0; // set the barrier status to 0
}
}
{
outsense = digitalRead(sensor2); // see if a car is leaving

if(outsense == 1){ // if a car is leaving, do the following
    if (storedSensor2 == 0) {
        cars--; // decrease the number of cars by 1
        Serial.print("The count is now "); // print the new value of the cars in the car park
        Serial.println(cars);
        digitalWrite(FULL_SIGN, LOW); // turn off full sign if on
    }
}

storedSensor2 = outsense;
}
```

When the car park is not full, the first thing the code checks for is if the push button has been pressed. If it hasn't, it checks the sensor at the output. If there is a car leaving, it will subtract 1 from the count and print to the serial monitor the new value of cars. If the car park was previously full, it will now turn the full sign off too.

If the button has been pressed, then it raises the barrier, extinguishes the red LED and turns on the green LED to signal it is clear for the car to enter the car park. Once the barrier has raised, there is a very short delay of 50ms before the code checks if the car has passed sensor 1. When the car does pass sensor 1, the barrier lowers, adds 1 to the count and prints to the serial monitor how many cars are now in the car park. It also extinguishes the green LED and illuminates the red LED.

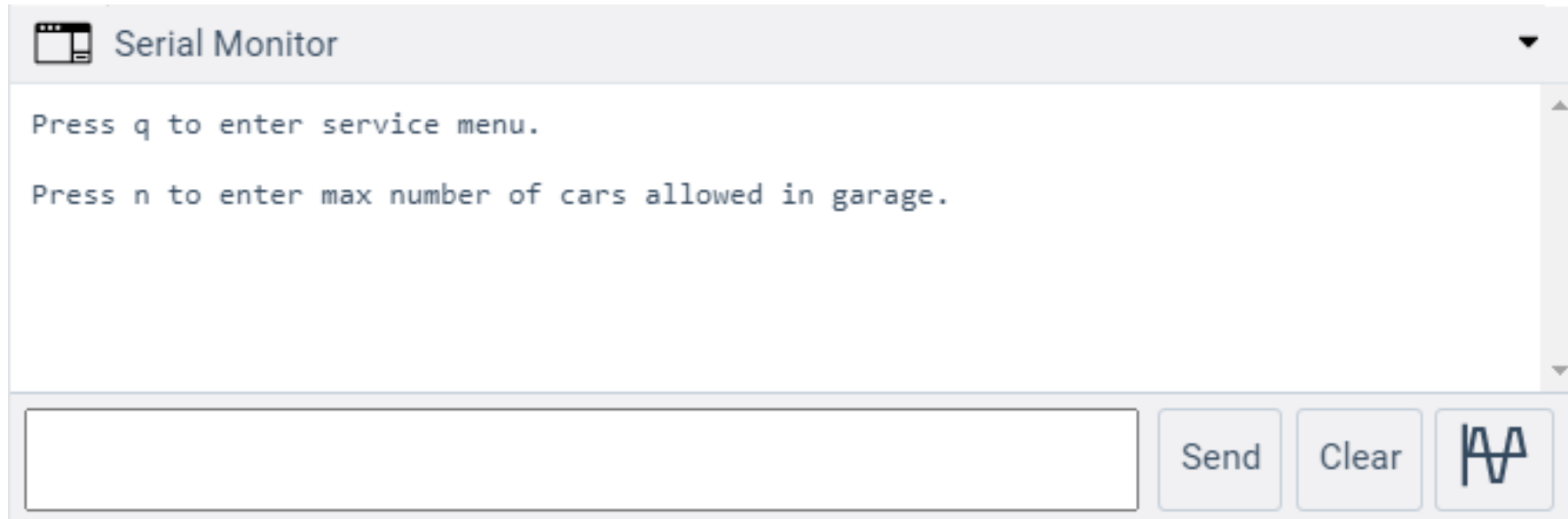
Note that when the barrier is raising and lowering, the lines 'myStepper.step(stepsPerRevolution/4);' and 'myStepper.step(-stepsPerRevolution/4);' are used respectively. The reason the steps per revolution is divided by 4 is because we only want the motor to execute a quarter turn in either direction.

When the car park becomes full, the below code is executed.

```
if (cars == fullpark){           // car park has reached maximum number of cars allowed
    delay(300);
    digitalWrite(FULL_SIGN, HIGH); // Turn on the full sign LED
    digitalWrite(red, HIGH);       // Turn on the red LED
    Serial.println("Sorry, The car park is full\n"); // display the message that the car park is full
    delay(300);
}
```

What happens first in the code is it detects that the maximum number of cars has been reached. When it reaches the maximum number, the red LED and full sign LED are turned on and no more cars will be allowed to enter until another car leaves.

Now that the main part of the code has been written, I need to look at a manual option to control all parts of the car park. On initial set up of the Arduino, the serial monitor displays the following message.

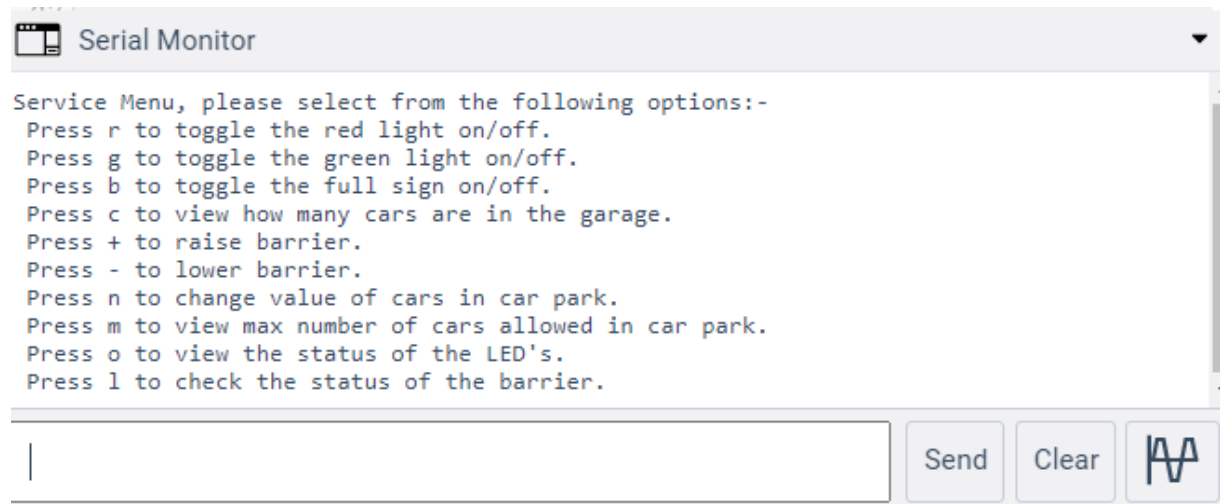


This says that the operator can do two of the following things. They can type the letter q, which will take them into a new menu with options to manually control all aspects of the car park. Or they can type the letter n and change the maximum number of cars that are allowed in the car park.

To get the serial monitor to display the service menu after entering the letter q, I had to write a piece of code which incorporated the 'Serial.available' function and the 'char' functions. The 'char' function takes whatever letter or word you input into the serial monitor, and from that will execute any code I have assigned each letter to. The below code shows us what actually happens when we enter the letter q.

```
if (Serial.available() > 0){ // check if the serial port available
char c = Serial.read();      // read the letter inputted to the serial monitor
if (c == 'q') {              // if the letter q is entered to the serial monitor then do the following
Serial.println("Service Menu, please select from the following options:-");
Serial.println(" Press r to toggle the red light on/off.");
Serial.println(" Press g to toggle the green light on/off.");
Serial.println(" Press b to toggle the full sign on/off.");
Serial.println(" Press c to view how many cars are in the garage.");
Serial.println(" Press + to raise barrier.");
Serial.println(" Press - to lower barrier.");
Serial.println(" Press n to change value of cars in car park.");
Serial.println(" Press m to view max number of cars allowed in car park.");
Serial.println(" Press o to view the status of the LED's.");
Serial.println(" Press l to check the status of the barrier.");
}
```

From this list we now have a number of further options. The next image is what the code looks like in the serial monitor once it has been executed.



The first three options allow us to control the state of the three LED's. With the code below I am able to toggle between the states.

```
// Toggle red LED on/off
if (c == 'r'){
    if(redOn == HIGH){
redOn = LOW;
        digitalWrite(red, redOn);
        Serial.println("Red Light OFF");
    } else {
        redOn = HIGH;
        digitalWrite(red, redOn);
        Serial.println("Red Light ON");
    }
}

// Toggle green LED on/off
if (c == 'g'){
    if(greenOn == HIGH){
greenOn = LOW;
        digitalWrite(green, greenOn);
        Serial.println("Green Light OFF");
    } else {
        greenOn = HIGH;
        digitalWrite(green, greenOn);
        Serial.println("Green Light ON");
    }
}

// Toggle blue LED/Full sign on/off
if (c == 'b'){
    if(blueOn == HIGH){
blueOn = LOW;
        digitalWrite(FULL_SIGN, blueOn);
        Serial.println("Full sign OFF ");
    } else {
        blueOn = HIGH;
        digitalWrite(FULL_SIGN, blueOn);
        Serial.println("Full sign ON ");
    }
}
```

The code here is firstly checking the state of the LED. At the beginning of the program I set three states for the LED's (redOn, greenOn and blueOn) instead of using the HIGH/LOW indicators. By using these states, I can now easily toggle between on and off. If, when you enter 'r' for the red LED, the code detects that redOn is high, then it will turn the LED off, and display the message "Red Light on." If it detects the red LED to be switched off, then it will change its state to high, and display the message "Red Light On." The code will do this for each LED.

By inputting the letter 'c' into the serial monitor I can check how many cars are currently in the car park.

```
// Check how many cars are currently in the garage
if (c == 'c'){
  Serial.print("The current number of cars in the garage is ");
  Serial.println(cars);
}
```

When the Arduino receives the letter 'c', it will display the message "The current number of cars in the garage is " followed by the number stored in the variable 'cars.'

The next thing we can do is manually raise and lower the barrier. The Arduino will receive the sign '+' to raise the barrier, and the sign '-' to lower the barrier. When it receives either one of these characters it executes the below code.

```

// Raise the barrier manually
if (c == '+'){
    myStepper.step(stepsPerRevolution/4); // turn the motor 90 degrees clockwise
    Serial.println("Raising barrier");
    digitalWrite(red, LOW);                // turn the red LED off
    digitalWrite(green, HIGH);             // turn the green LED on
    barrier =1;                            // set the variable 'barrier' to 1
}

// Lower the barrier manually
if (c == '-'){
    myStepper.step(-stepsPerRevolution/4); // turn the motor 90 degrees anti clockwise
    Serial.println("Lowering barrier");
    digitalWrite(red, HIGH);                // turn the red LED on
    digitalWrite(green, LOW);              // turn the green LED on
    barrier =0;                            // set the variable 'barrier' to 0
}

```

In both parts of the code we can see that the barrier is raised or lowered depending on the selection, and the state of the LED's match what happens when the barrier operates in the main part of the code. We also set the 'barrier' variable to 1 when the barrier is raised, and to 0 when the barrier is lowered. This will come in handy later when we need to check the status of the barrier.

The next option in the list is to change the maximum number of cars allowed in the car park at any given time. To do this we first need to enter the letter 'n' into the serial monitor.

```

// change the number of cars allowed in the car park
if (c == 'n'){
  Serial.println("Enter a number between 10 and 255, to change max number of cars in car park: ");
  while (Serial.available () == 1) {} // hold this state to allow operator to enter a new value
  change = Serial.parseInt(); // read the number entered into the serial monitor, and store in variable 'change'
  if ((change >= 10) && (change <= 255)){ // if the value of 'change' is more than or equal to 10, or less than or equal to 255, then do the following
    Serial.print("New number of cars in car park is: ");
    Serial.println(change);
    fullpark = change; // the variable fullpark will now take the value of the variable change
    if (fullpark > cars){
      digitalWrite(FULL_SIGN, LOW); // if the full sign was on but the new value is now greater than the full park then turn off the full sign
    }
  }
  else { // if the input is not in the range of '10 >= and <= 255 then print the next message
    Serial.print("Invalid input, please pick a number between 10 and 255");
  }
}
}

```

The first thing that happens is the message “Enter a number between 10 and 255, to change max number of cars in car park: “ is displayed. If we enter a number within that range, then the number of cars allowed in the car park will change to the number entered and stored in the variable ‘change.’ If the car park was previously full before making the change, then the full sign LED will also turn off.

If the number entered is not within the range, then you will receive the following error message saying, ‘Invalid input, please pick a number between 10 and 255’. On the next page is both results displayed in the serial monitor.

 Serial Monitor

Enter a number between 10 and 255, to change max number of cars in car park:
New number of cars in car park is: 22

Send

Clear



The number 22 was entered, and the maximum number of cars was changed as it was within the boundaries.

 Serial Monitor

Enter a number between 10 and 255, to change max number of cars in car park:
Invalid input, please pick a number between 10 and 255

Send

Clear

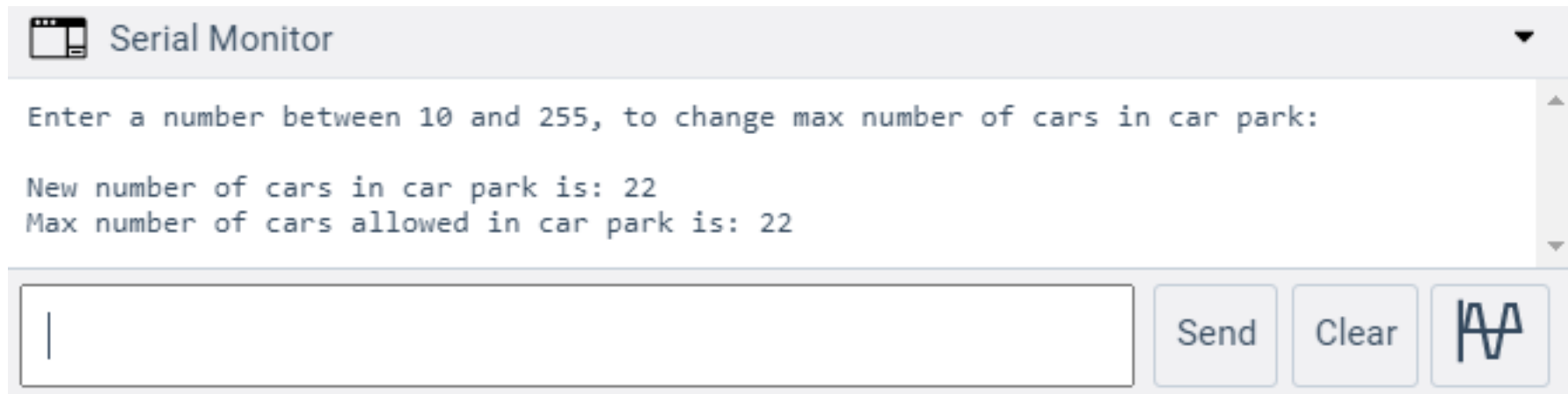


This time I entered the number 4, and the “Invalid input, please pick a number between 10 and 255” message was displayed as the number 4 was not within the boundaries.

Now that we have changed the number of cars allowed, we can check that number by entering the letter 'm'. This reads the value currently stored in the variable 'fullpark' and displays it on the serial monitor.

```
// check how many cars allowed in car park
if (c == 'm'){
    Serial.print("Max number of cars allowed in car park is: ");
    Serial.println(fullpark);
}
```

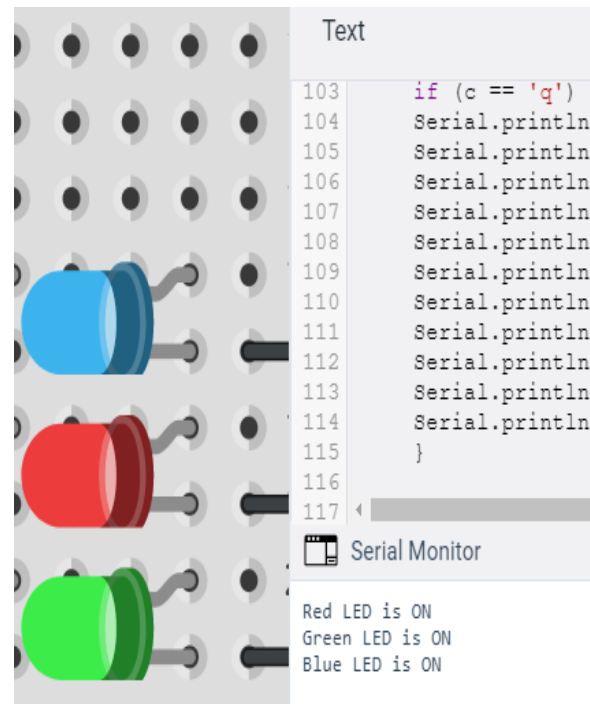
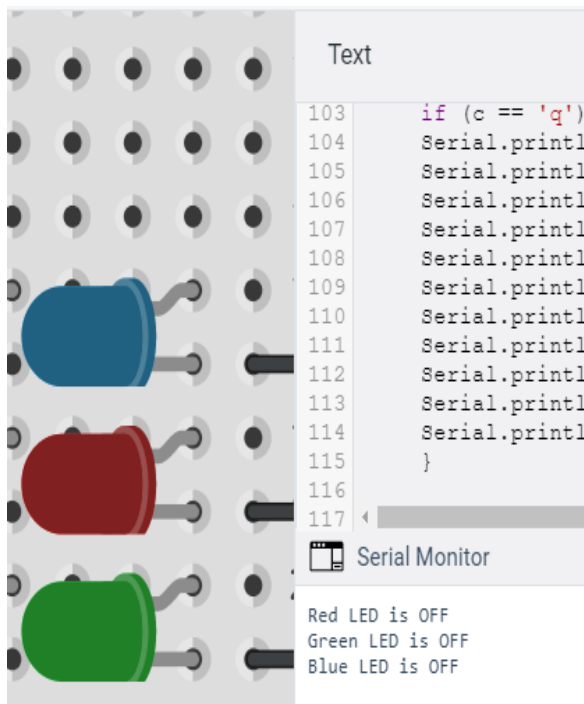
As we have changed the number to 22, we can prove it by showing what is displayed in the serial monitor below.



Next, we can check the current status of all the LED's by entering the letter 'o'. The code for this is below.

```
// check status of LED's
if (c == 'o'){
    if (digitalRead(red) == HIGH){
        Serial.println("Red LED is ON ");}
    else if (digitalRead(red) == LOW)
    {
        Serial.println("Red LED is OFF");
    }
    if (digitalRead(green) == HIGH){
        Serial.println("Green LED is ON ");}
    else if (digitalRead(green) == LOW)
    {
        Serial.println("Green LED is OFF");
    }
    if (digitalRead(FULL_SIGN) == HIGH){
        Serial.println("Blue LED is ON ");}
    else if (digitalRead(FULL_SIGN) == LOW)
    {
        Serial.println("Blue LED is OFF");
    }
}
```

This checks the individual status of each LED and prints a message depending on what state they are in. On the next pages will be an example of the circuit simulating in TinkerCad. One will have all LED's on and the correct message will be displayed, and the other image will have all LED's turned off with the correct message for these also.



The final condition we can check on the list is the status of the barrier. This is done by entering the letter 'l'.

```

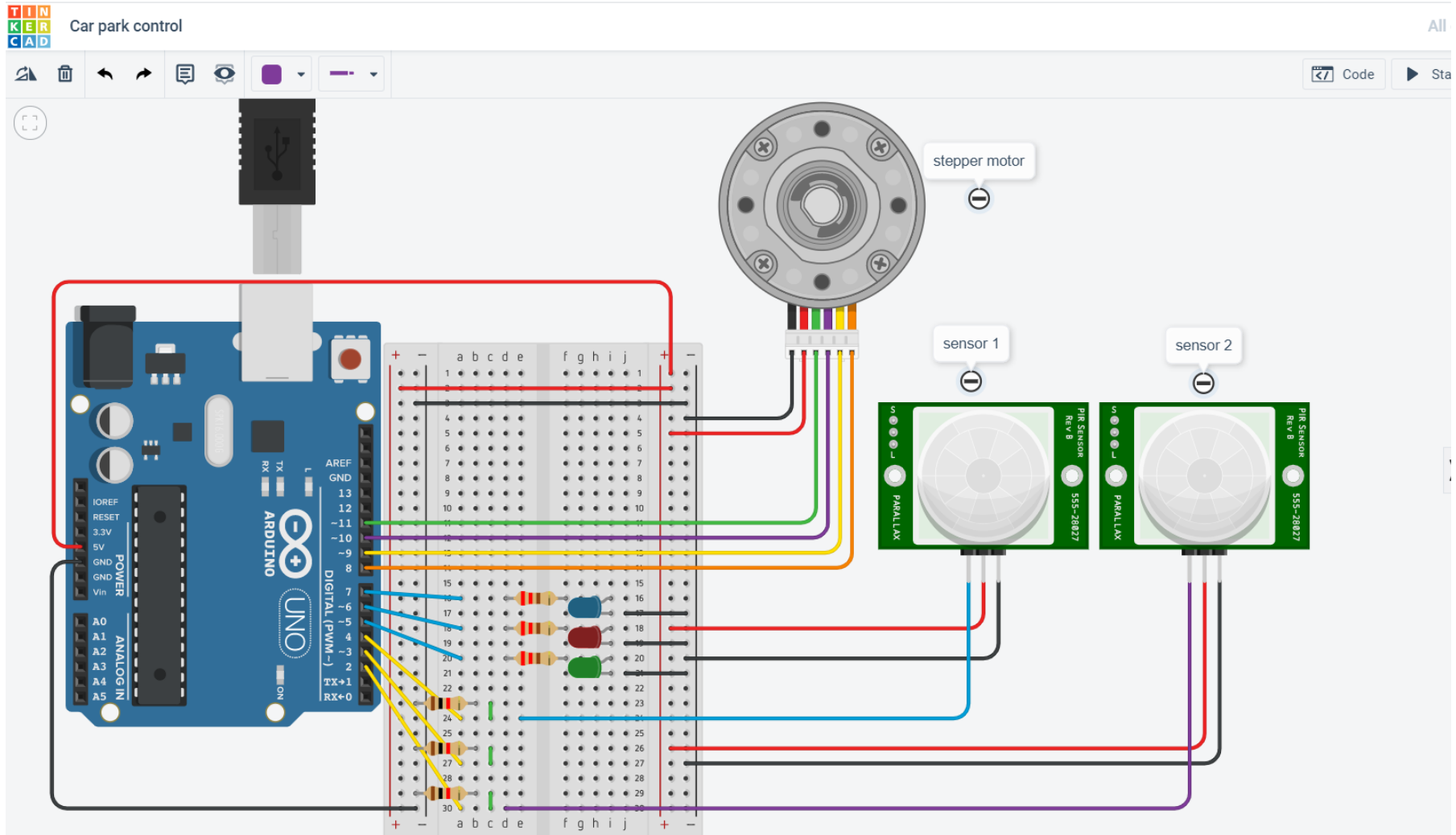
// check status of the barrier
if (c == 'l'){
    Serial.println("checking barrier status.... ");

    if (digitalRead(red) == HIGH && barrier == 0)
    {
        Serial.println("lowered");}
    else if (digitalRead(red) == LOW and barrier == 1) {
        Serial.println("raised");
    }
    else { Serial.println("lowered");
    }
}

```

If the barrier variable is set to 0, and the red LED is on, then it will display the message that the barrier is lowered. If the variable is set to 1, and the green LED is on, then it will display the barrier is raised.

Circuit Diagram



Testing

Now that I have written the entirety of my code, I need to conduct some testing. I have done this in the form of two testing tables. One for the main body of the code for the push button and the two sensors, and one for the manual part of the code.

Testing Table – Button and sensors			
Input	Expected result	Actual result	Desired Result? (Y/N)
Press 'Button'	Red LED should turn off, Green LED should come on, barrier raises.	Red LED off, Green LED on, barrier raised.	Y
Press 'Sensor 1'	Green LED should turn off, Red LED should come back on and barrier will lower and increase the count by 1	Green LED off, Red LED on, barrier lowered, count increased by 1	Y
Press 'Sensor 1' 1 before max limit	Green LED should turn off, Red LED should come back on and barrier will lower and increase the count by 1. Full sign should illuminate.	Green LED turned off, Red LED came on, Full sign LED also came on.	Y
Press 'Sensor 2' when car park not full	Count should decrease by 1	Count decreases by 1	Y
Press 'Sensor 2' when car park full	Count should decrease by 1 and the full sign should go off	Count decreases by 1 and the full light goes off	Y

From this first table I can see that all tests worked correctly and displayed the same results that I expected to see.

Testing Table – Service menu

Input	Expected result	Actual result	Desired Result? (Y/N)
Serial read 'r'	Red LED should toggle on/off	Red LED turned on, pressed again turned off	Y
Serial read 'g'	Green LED should toggle on/off	Green LED turned on, pressed again turned off	Y
Serial read 'b'	Full sign LED should toggle on/off	Full sign LED turned on, pressed again turned off	Y
Serial read 'o'	Display the status off the LED	Displayed that all LED's were off	Y
Serial read 'm'	Displays the maximum number of cars allowed in the car park	Displayed that the max number of cars allowed is 10	Y
Serial read 'n'	Change number of cars allowed in car park between 10 and 255	Asked to set new number of cars between 10 and 255. Entered 8, said number was invalid try again. Entered 259, said number was invalid try again. Entered 27, set new number of cars to 27	Y
Serial read 'l'	Display the status of the barrier.	Displayed that the barrier was lowered. Manually raised barrier, checked again, displayed that barrier was raised.	Y
Serial read 'c'	Display how many cars are currently in the car park	Displayed there is currently 0 cars in the car park	Y
Serial read '+'	Barrier will raise	Barrier raised	Y
Serial read '-'	Barrier will lower	Barrier lowered	Y

Similarly, all tests for the manual element of the code worked as expected too.

Whilst conducting some further tests, I found that the Capital letter alternative to each letter entered in the serial monitor did not give us the desired output. I have rectified this in my final code by using the OR function in Arduino (||).

Final copy of the program

Over the next number of pages is the full copy of my program.

```
#include <Stepper.h>

const int button = 2; // assign the push button to pin 2
const int sensor1 = 3; // assign sensor 1 to pin 3
const int sensor2 = 4; // assign sensor 2 to pin 4
const int green = 5; // assign the green LED to pin 5
const int red = 6; // assign the red LED to pin 6
const int FULL_SIGN = 7; // assign the full sign display to pin 7
int redOn = HIGH; // will be used to dictate the state of the red LED. Will start HIGH.
int greenOn = LOW; // will be used to dictate the state of the green LED. Will start LOW.
int blueOn = LOW; // will be used to dictate the state of the full sign. Will start LOW.
int change; // variable that will be used to change the number of cars stored in the car park
const int stepsPerRevolution = 200; // change this to fit the number of steps per revolution for your motor
int barrier; // will be used to view the status of the barrier
unsigned int fullpark = 10; // max amount of cars
```

```
int cars = 0; //holds number of cars currently in car park
```

```
int x = 0; // will be used to check the state of sensor 2
```

```
int outsense = 0;    // starting state of the outsense
```

```
int storedSensor2 = 1; // starting state of sensor 2
```

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

```
void setup() {
```

```
    myStepper.setSpeed(60); // set the speed at 60 rpm:
```

```
    Serial.begin(9600); // initialize the serial port:
```

```
    Serial.println("Press q to enter service menu. \n"); // print the message to the serial monitor
```

```
    Serial.println("Press n to enter max number of cars allowed in garage. \n"); // print the message to the serial monitor
```

```
    pinMode(button, INPUT); // configure the push button to be a digital input
```

```
    pinMode(sensor1, INPUT); // configure sensor 1 to be a digital input
```

```
    pinMode(sensor2, INPUT); // configure sensor 2 to be a digital input
```

```
    pinMode(green, OUTPUT); // configure the green LED to adopt the status of a digital output
```

```
    pinMode(red, OUTPUT); // configure the red LED to adopt the status of a digital output
```

```
    pinMode(FULL_SIGN, OUTPUT); // configure the full sign to adopt the status of a digital output
```

```
    digitalWrite(red, redOn); // Tell the red LED to adopt the status of redOn
```

```
    digitalWrite(green, greenOn); // Tell the green LED to adopt the status of greenOn
```

```

digitalWrite(FULL_SIGN, blueOn); // Tell the full sign to adopt the status of blueOn
digitalWrite(button, LOW); // button to start at logic LOW
digitalWrite(sensor1, LOW); // sensor 1 to start logic LOW
digitalWrite(sensor2, LOW); // sensor 2 to start logic LOW
}

void loop() {

    if (cars < fullpark){ // execute the following code if the car park is not full

        if(digitalRead(button)==1){ // execute the following code if the push button is pressed
            myStepper.step(stepsPerRevolution/4); // turn the motor 90 degrees clockwise
            digitalWrite(red, HIGH); // turn the red LED on
            Serial.print("Raising barrier\n"); // print the message 'raising barrier'
            delay(1000);
            digitalWrite(green, HIGH); // Turn the green LED on
            digitalWrite(red, LOW); // Turn the red LED off
            barrier == 1; // set barrier status to 1

        }

    }
}

```

```

    delay(50); // wait for sensors to stabilize
    x = digitalRead(sensor1); // check sensor1
} while (x < 1); // when the car passes sensor 1, do the following
    Serial.print("Lowering barrier\n"); // print that the barrier is lowering
    myStepper.step(-stepsPerRevolution/4); // turn the motor 90 degrees anti clockwise
    digitalWrite(green, LOW); // turn green LED off
    digitalWrite(red, HIGH); // turn red LED on
    cars++; //increase the number of cars in the car park
    delay(500);
    Serial.print("The count is now "); // print the new value for the cars in the car park
    Serial.println(cars);
    barrier == 0;

}
}
{
    outsense = digitalRead(sensor2); // see if car is leaving

    if(outsense == 1){ // if a car is leaving do the following
        if (storedSensor2 == 0) {
            cars--; // decrease the number of cars by 1
        }
    }
}

```

```

    Serial.print("The count is now "); // print the new value of cars in the car park
    Serial.println(cars);
    digitalWrite(FULL_SIGN, LOW);    // turn off full sign if on
}
}
storedSensor2 = outsense;
}

if (cars == fullpark){    // car park has reached maximum number of cars allowed
    delay(300);
    digitalWrite(FULL_SIGN, HIGH); // Turn on the full sign LED
    digitalWrite(red, HIGH);    // Turn on the red LED
    Serial.println("Sorry, The car park is full\n"); // display the message that the car park is full
    delay(300);
}

//the following code is for the service menu
//press q to activate service menu

if (Serial.available() > 0){ // check if the serial port available
    char c = Serial.read();    // read the letter inputted to the serial monitor

```

```

if (c == 'q') {           // if the letter q is entered to the serial monitor then do the following
  Serial.println("Service Menu, please select from the following options:-");
  Serial.println(" Press r to toggle the red light on/off.");
  Serial.println(" Press g to toggle the green light on/off.");
  Serial.println(" Press b to toggle the full sign on/off.");
  Serial.println(" Press c to view how many cars are in the garage.");
  Serial.println(" Press + to raise barrier.");
  Serial.println(" Press - to lower barrier.");
  Serial.println(" Press n to change value of cars in car park.");
  Serial.println(" Press m to view max number of cars allowed in car park.");
  Serial.println(" Press o to view the status of the LED's.");
  Serial.println(" Press l to check the status of the barrier.");
}

```

// Toggle red LED on/off

```

if (c == 'r' || c == 'R'){
  if(redOn == HIGH){
    redOn = LOW;
    digitalWrite(red, redOn);
    Serial.println("Red Light OFF");
  } else {

```

```
    redOn = HIGH;
    digitalWrite(red, redOn);
    Serial.println("Red Light ON");
  }
}
```

```
// Toggle green LED on/off
```

```
    if (c == 'g' || c == 'G'){
    if(greenOn == HIGH){
    greenOn = LOW;
    digitalWrite(green, greenOn);
    Serial.println("Green Light OFF");
    } else {
    greenOn = HIGH;
    digitalWrite(green, greenOn);
    Serial.println("Green Light ON");
    }
    }
```

```
// Toggle blue LED/Full sign on/off
```

```
    if (c == 'b' || c == 'B'){
```

```
if(blueOn == HIGH){  
  blueOn = LOW;  
  digitalWrite(FULL_SIGN, blueOn);  
  Serial.println("Full sign OFF ");  
} else {  
  blueOn = HIGH;  
  digitalWrite(FULL_SIGN, blueOn);  
  Serial.println("Full sign ON ");  
}  
}
```

// Check how many cars are currently in the garage

```
if (c == 'c' || c == 'C'){  
  Serial.print("The current number of cars in the garage is ");  
  Serial.println(cars);  
}
```

// Raise the barrier manually

```
if (c == '+'){  
  myStepper.step(stepsPerRevolution/4); // turn the motor 90 degrees clockwise  
  Serial.println("Raising barrier");  
}
```

```

digitalWrite(red, LOW);          // turn the red LED off
digitalWrite(green, HIGH);       // turn the green LED on
    barrier =1;                  // set the variable 'barrier' to 1
}

```

// Lower the barrier manually

```

if (c == '-') {
    myStepper.step(-stepsPerRevolution/4); // turn the motor 90 degrees anti clockwise
    Serial.println("Lowering barrier");
    digitalWrite(red, HIGH);              // turn the red LED on
    digitalWrite(green, LOW);             // turn the green LED on
    barrier =0;                           // set the variable 'barrier' to 0
}

```

// change the number of cars allowed in the car park

```

if (c == 'n' || c == 'N') {
    Serial.println("Enter a number between 10 and 255, to change max number of cars in car park: ");
    while (Serial.available () == 1) {} // hold this state to allow operator to enter a new value
    change = Serial.parseInt(); // read the number entered into the serial monitor, and store in variable 'change'
    if ((change >= 10) && (change <= 255)) { // if the value of 'change' is more than or equal to 10, or less than or equal to 255, then do the
following
        Serial.print("New number of cars in car park is: ");

```

```

Serial.println(change);

fullpark = change; // the variable fullpark will now take the value of the variable change
if (fullpark > cars){
    digitalWrite(FULL_SIGN, LOW); // if the full sign was on but the new value is now greater than the full park then turn off the full sign
}
}
else { // if the input is not in the range of '10 >= and <= 255 then print the next message
    Serial.print("Invalid input, please pick a number between 10 and 255");
}
}
}

```

// check how many cars allowed in car park

```

if (c == 'm' || c == 'M'){
    Serial.print("Max number of cars allowed in car park is: ");
    Serial.println(fullpark);
}

```

// check status of the barrier

```

if (c == 'l' || c == 'L'){
    Serial.println("checking barrier status.... ");
}

```

```
if (digitalRead(red) == HIGH && barrier == 0)
{
    Serial.println("lowered");}
else if (digitalRead(red) == LOW and barrier == 1) {
    Serial.println("raised");
}
else { Serial.println("lowered");
}
}
```

// check status of LED's

```
if (c == 'o' || c == 'O'){
    if (digitalRead(red) == HIGH){
        Serial.println("Red LED is ON ");}
    else if (digitalRead(red) == LOW)
    {
        Serial.println("Red LED is OFF");
    }
    if (digitalRead(green) == HIGH){
        Serial.println("Green LED is ON ");}
```

```
else if (digitalRead(green) == LOW)
{
    Serial.println("Green LED is OFF");
}
if (digitalRead(FULL_SIGN) == HIGH){
    Serial.println("Blue LED is ON ");}
else if (digitalRead(FULL_SIGN) == LOW)
{
    Serial.println("Blue LED is OFF");
}
}
}
```


Conclusion

In conclusion, the code works to the brief. However, when I carried out some further tests there were a couple of things I found that I could improve on if I were to do the same task again.

When there is a car entering the car park and leaving the car park at the same time there is a problem with the count. It appears that the Arduino can only process one of the actions at a time. To rectify this, I think that I would have to introduce an interrupt or sub routine to be able to perform both actions together.

I also took for granted that the upper case and lower-case characters would be read the same using the 'char' command. However, upon testing it turns out they didn't. This is because in the ASCII code the upper case and lower-case characters have separate codes. To get over this I used the OR command in Arduino for both upper- and lower-case characters.

The final problem I found was the count for 'cars - -' would allow the count of the cars to go into the negative numbers. I have found a solution to change it. The line 'if(outsense == 1){' would allow the count to go negative. By changing the line of code to 'if(outsense == 1 && cars >= 1) {', then the count will only count down to 0.

Below is a working link for my program simulating in Tinkercad:

<https://www.dropbox.com/s/0r6obtq4glrkyk/Michael%20Jones%20Microprocessor%20working%20example.mp4?dl=0>

References

Think Automation, 2021. *What is an algorithm? An 'in a nutshell' explanation*. [Online]
Available at: <https://www.thinkautomation.com/eli5/what-is-an-algorithm-an-in-a-nutshell-explanation/>
[Accessed 09 05 2021].