

Car Park Assignment

In this essay I will explain how I am going to produce a program for an Arduino microprocessor to control a functioning carpark.

Not finding a parking space for you sometimes is indeed a critical issue. An important appointment might be missed. Time will be wasted and more money will be spent. Criticism and inconvenience are more likely to happen. So, parking monitoring is an important solution.

It has many practical advantages. It can count the number of vehicles inside the parking area at all times, in addition it can tell you that the carpark is full which will display a red and yellow light so that no more cars can enter and the barrier will not open if the button to enter is pressed.

A Barrier will be installed which will rise when the button is pressed (if car park is not full) illuminating a green light so you know when the barrier has raised making it safe to enter.

A sensor on the entrance will let the Arduino know that a car has passed it, increasing the number with every car that comes in until the carpark is full. Once the car has passed the sensor a red light will illuminate then the barrier will lower.

Another sensor will be placed at the exit which will instantly tell the Arduino that a car has exited thus reducing the number of cars. If the carpark became full then as soon as a car left it would turn off the full light and reduce the number of cars by one meaning there is space for one more to enter, so it then allows the button to be pressed again and another car may enter.

Operator overrides will also be installed allowing all the processes to be seen and allowing them to operate any function as it is desired.

I think a barrier at the exit would be a great advantage so that cars are not able to come in that way, that is if it was tried.

Arduino Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Car Park Assignment

Arduino Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

Car Park Assignment

Programming

The Arduino Uno can be programmed with the Arduino software and other programming websites such as Tinkercad which I have used.

C is a compiled language. A compiler is a special tool that compiles the program and converts it into the object file which is machine readable. After the compilation process, the linker will combine different object files and creates a single executable file to run the program.

Nowadays, various compilers are available online, and you can use any of those compilers. The functionality will never differ and most of the compilers will provide the features required to execute both 'C' and 'C++' programs.

Programming Summary

- 'C' was developed by Dennis Ritchie in 1972.
- It is a robust language.
- It is a low programming level language close to machine language
- It is widely used in the software development field.
- It is a procedure and structure oriented language.
- It has the full support of various operating systems and hardware platforms.
- Many compilers are available for executing programs written in 'C'.
- A compiler compiles the source file and generates an object file.
- A linker links all the object files together and creates one executable file.
- It is highly portable.

Bibliography

www.farnell.com/datasheets/1682209.pdf

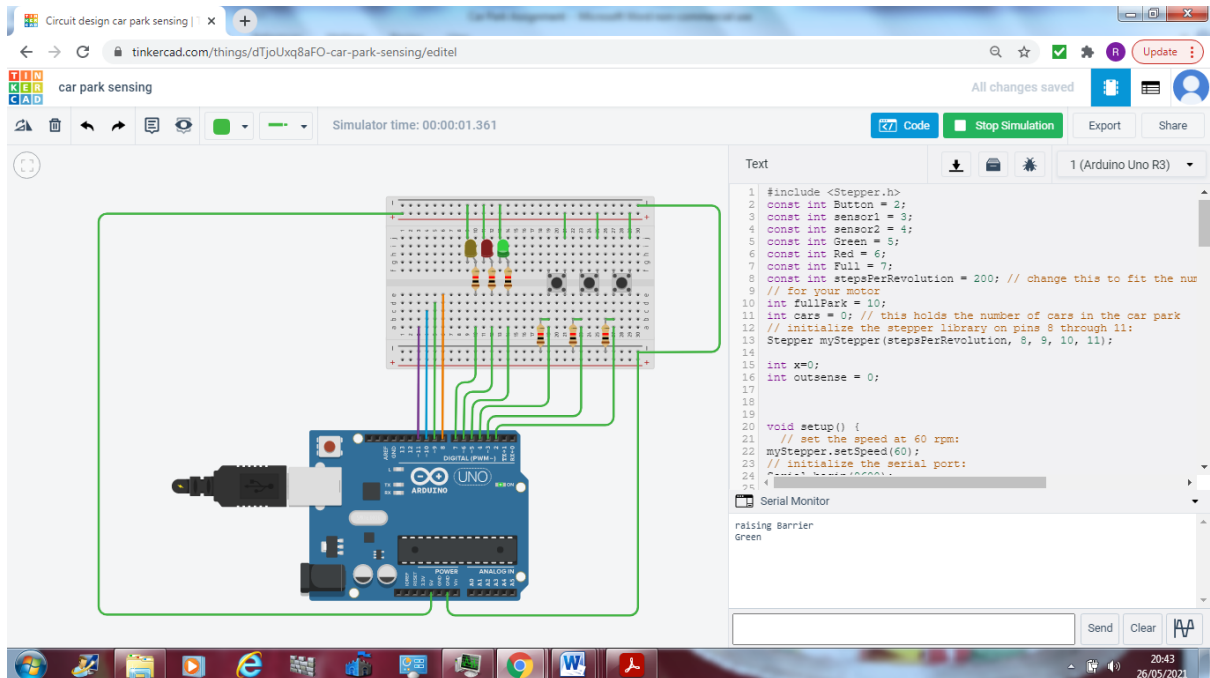
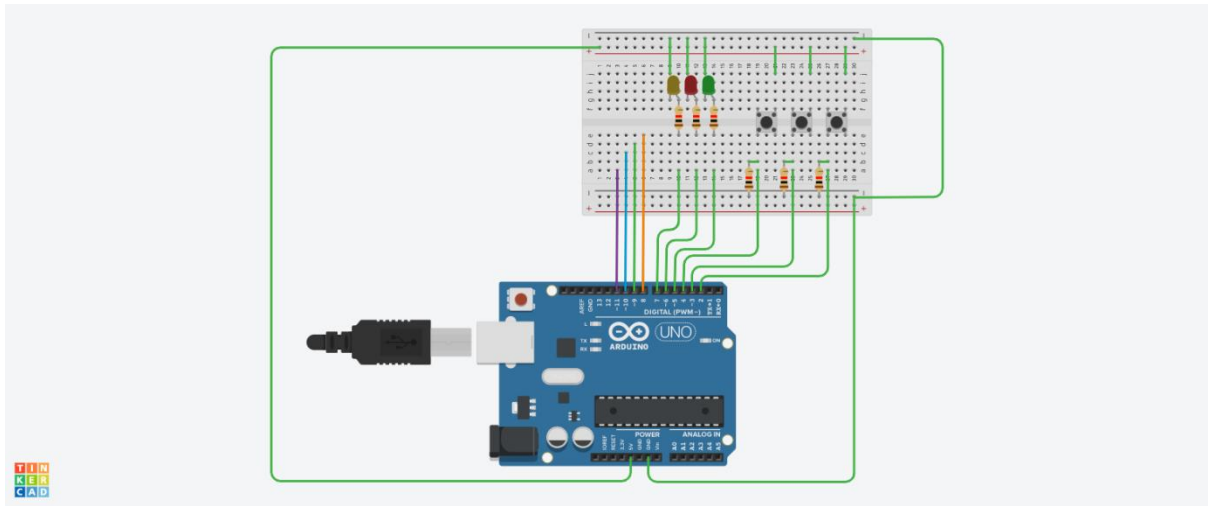
www.guru99.com/c-programming-language.html

Below is a flow chart of how the system works, pictures of my project from Tinkercad showing the Arduino uno connected to a Breadboard with one button on the right for entry and another two acting as sensors. The other pictures show the barrier raising with green light on, the red light on with barrier lowering, and the yellow & red light on when the carpark becomes full, and the code that I have produced.

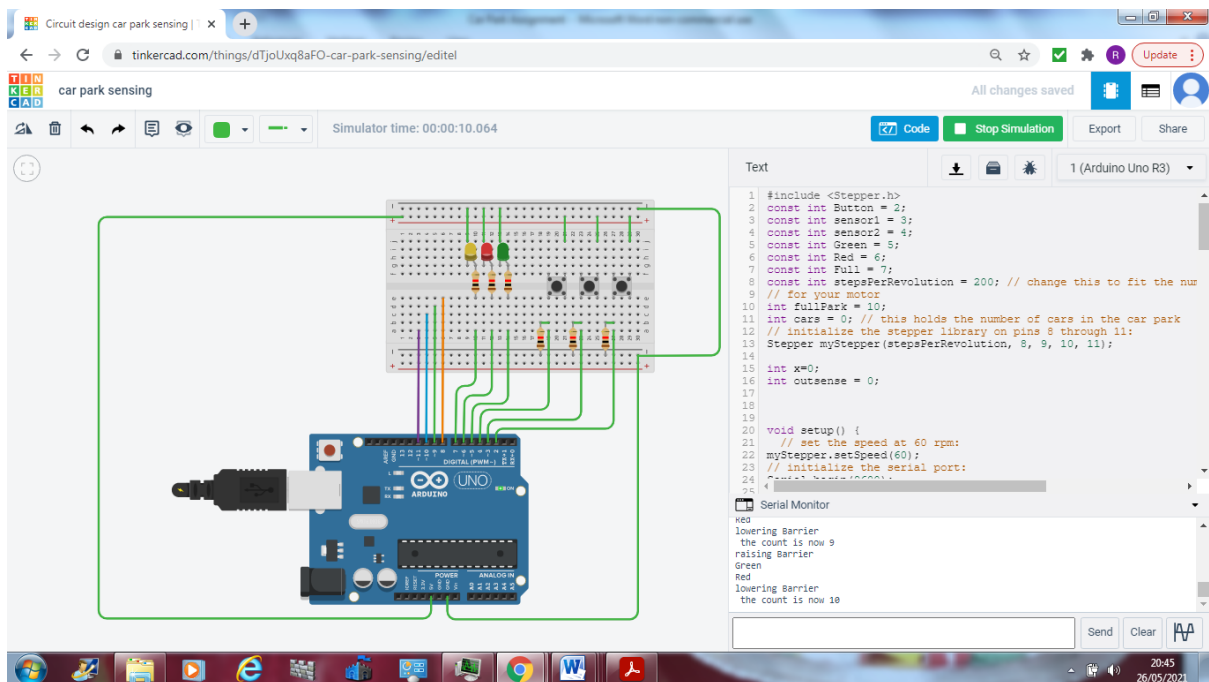
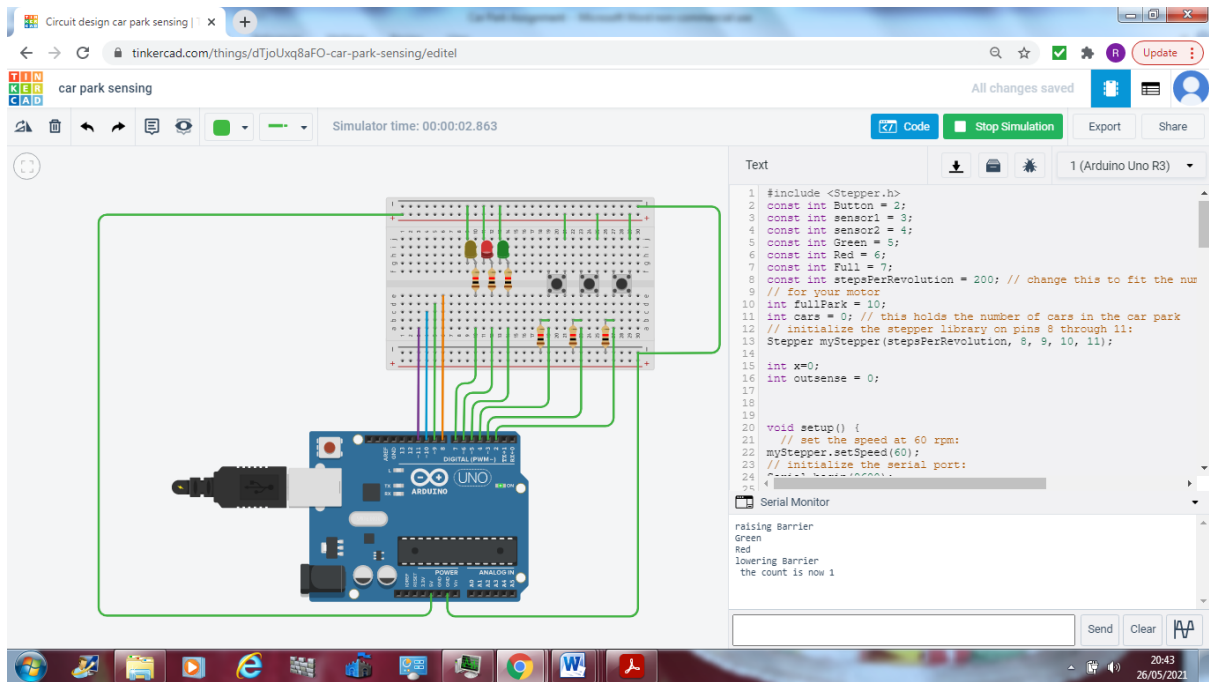
Flow Chart



Car Park Assignment



Car Park Assignment



Car Park Assignment

Car Park Code

```
#include <Stepper.h>

const int Button = 2;

const int sensor1 = 3;

const int sensor2 = 4;

const int Green = 5;

const int Red = 6;

const int Full = 7;

const int stepsPerRevolution = 200; // change this to fit the number of steps per revolution

// for your motor

int fullPark = 3;

int cars = 0; // this holds the number of cars in the car park

// initialize the stepper library on pins 8 through 11:

Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);


int x=0;

int outsense = 0;


void setup() {

// set the speed at 60 rpm:

myStepper.setSpeed(60);

// initialize the serial port:

Serial.begin(9600);

pinMode(Button, INPUT);

pinMode(sensor1, INPUT);

pinMode(sensor2, INPUT);

pinMode(Green, OUTPUT);

    digitalWrite(Green,LOW);

pinMode(Red, OUTPUT);

    digitalWrite(Red,LOW);

pinMode(Full, OUTPUT);

    digitalWrite(Full,LOW);
```

Car Park Assignment

```
}

void loop()

{

  if (cars >= fullPark) // Denys Entry

  {

    digitalWrite(Full, HIGH);

  }

  else

  {

    digitalWrite(Full, LOW);

  }

  digitalWrite(Green, LOW);

  if (cars >= fullPark) goto LetCarOut;

{

  LetCarIn:// Approves Entry

  {

    if (cars < fullPark){

    {

      if ( digitalRead(Button) == HIGH)

      {

        digitalWrite(Green, HIGH);// Green LED is switched on

      }

      else

      {

        digitalWrite(Green, LOW);

      }

    }

  }

  if(digitalRead(Button)==1){// Initialises Barrier

  myStepper.step(stepsPerRevolution/4);// Barrier raises

  Serial.println("raising Barrier ");
```

Car Park Assignment

```
Serial.println("Green ");

do

{

    // wait for sensors to stabilize

    x = digitalRead(sensor1); // check the sensors

    {

        if ( digitalRead(sensor1) == HIGH)

        {

            digitalWrite(Red, HIGH); // Red LED is switched on

        }

        else

        {

            digitalWrite(Red, LOW);

        }

    }

} while (x < 1); // Initialises Barrier

    myStepper.step(-stepsPerRevolution/4); //Barrier lowers

Serial.println("Red ");

Serial.println("lowering Barrier ");

cars++; //increments the number of cars in the car park by 1

Serial.print(" the count is now ");

Serial.println(cars); // shows the number of cars

}
```

Car Park Assignment

```
LetCarOut: // Lets car leave

outsense = digitalRead(sensor2); // see if car is leaving the car park

if (outsense == 1){

  delay(50); // Avoids switch bouncing

  {

    cars --; //decrements the number of cars leaving the car park by 1

    Serial.println("sensor2 at 1");

    Serial.println("reduce number ");

    Serial.println(" the count is now ");

    Serial.println(cars); // shows the number of cars

  }

}

}
```