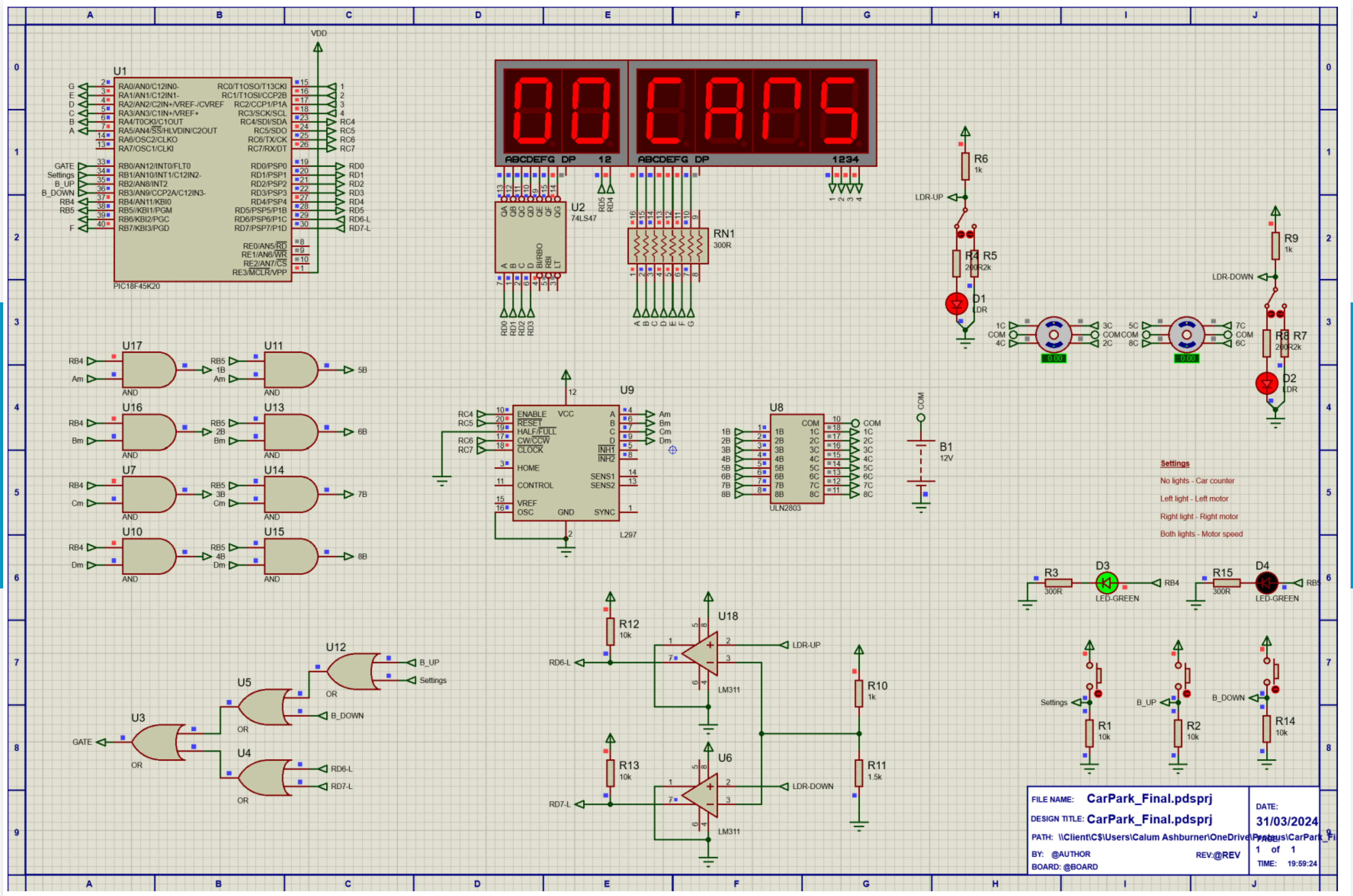
Car Park Project

- This project was given to me at University. We were required to design a car park using a PIC microcontroller that was programmed with Assembly Language.
- First, the car park was created in software simulation, then it was made in real life using electrical components.
- The Park needed to be demonstrated in front of my lecturers to show its function.
- I received a 100% score for this project.

Car Park Criteria

- The car park needed to have a maximum capacity of 20 cars.
- Two motors needed to be used to control gates that allow access in and out of the car park.
- LDRs would be used to check if a car has entered or exited.
- The number of cars will be displayed on a screen, with extra feedback being given for when the car park is full.
- The car park operator needs to be able to manually override the number of cars that is recorded in the system.
- The speed of the motors must be adjustable.

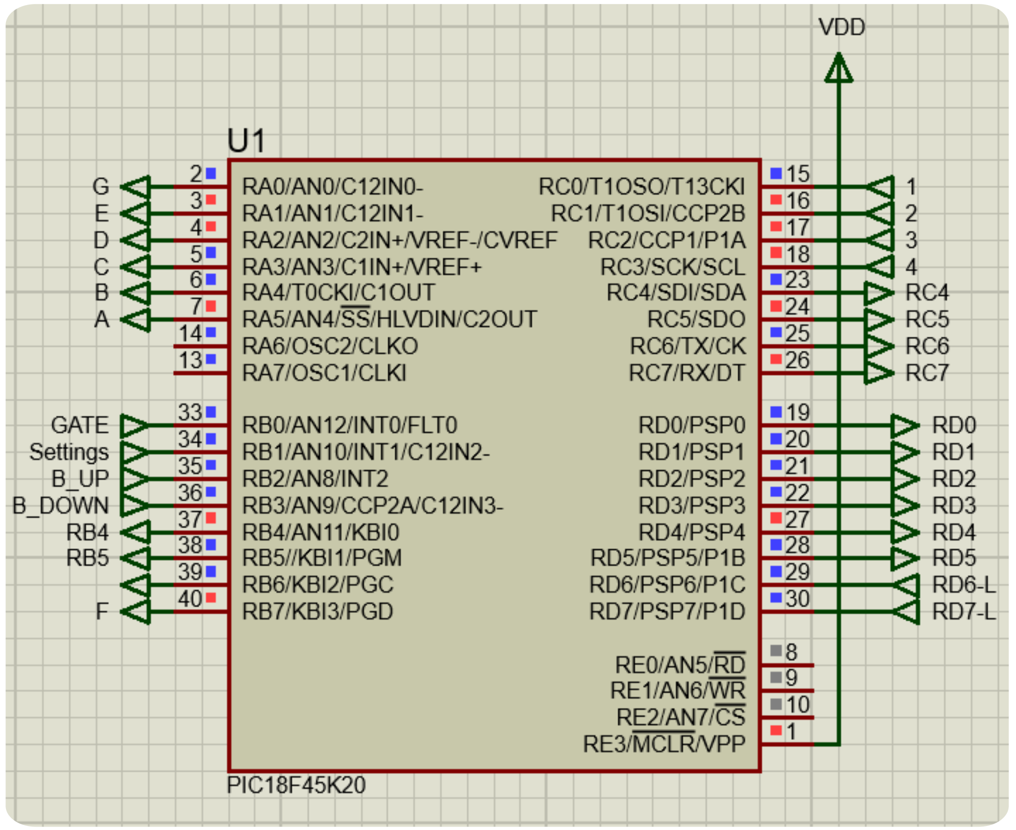
CIRCUIT SCHEMATIC



Settings
No lights - Car counter
Left light - Left motor
Right light - Right motor
Both lights - Motor speed

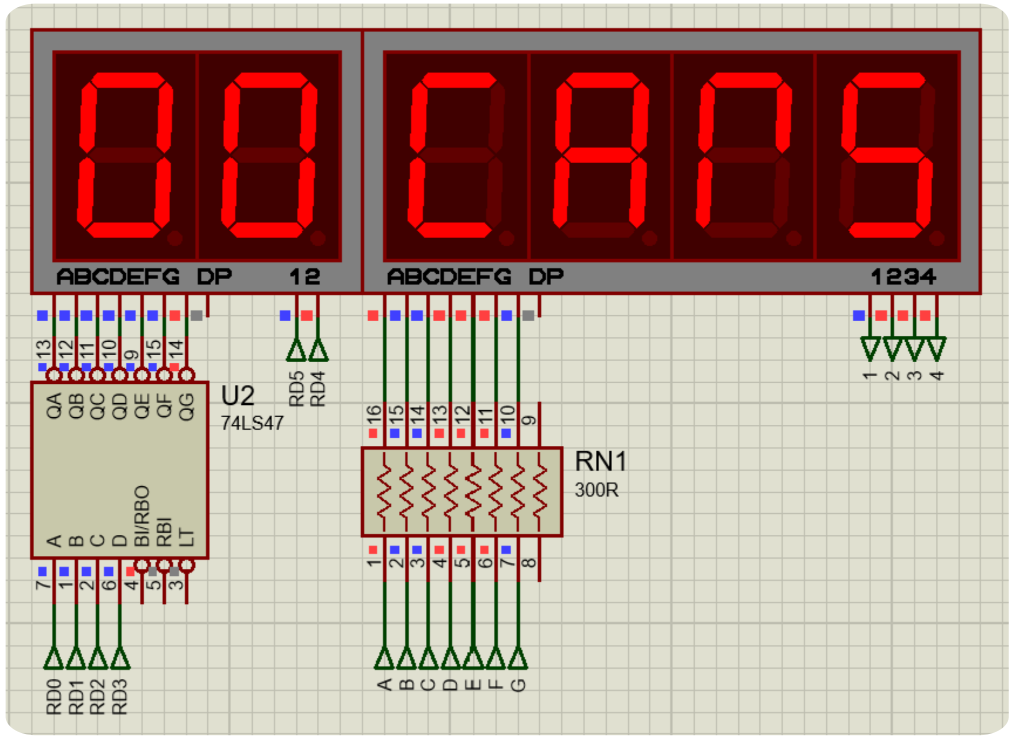
FILE NAME: CarPark_Final.pdsprj
DESIGN TITLE: CarPark_Final.pdsprj
PATH: \\Client\C$\Users\Calum Ashburner\OneDrive\Proteus\CarPark_Fi
BY: @AUTHOR REV:@REV
BOARD: @BOARD
DATE: 31/03/2024
PAGE: 1 of 1
TIME: 19:59:24

PIC18F45K20 Microcontroller

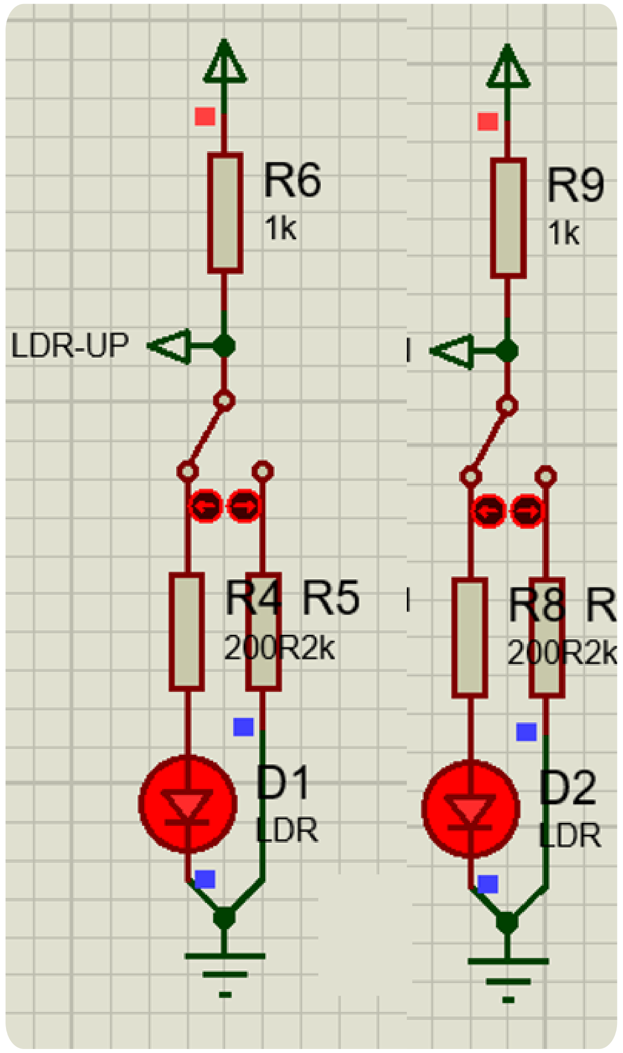


- The PIC18F45K20 was used for this project, as it was a simple 8-bit microcontroller with plenty of I/O pins.
- MPLAB X IDE was used to program and debug the chip, with the code being done purely in Assembly Language.
- We were provided a development board to plug the chip in and connect the inputs and outputs to a breadboard containing other components.

Multiplexed 7 Segment Display



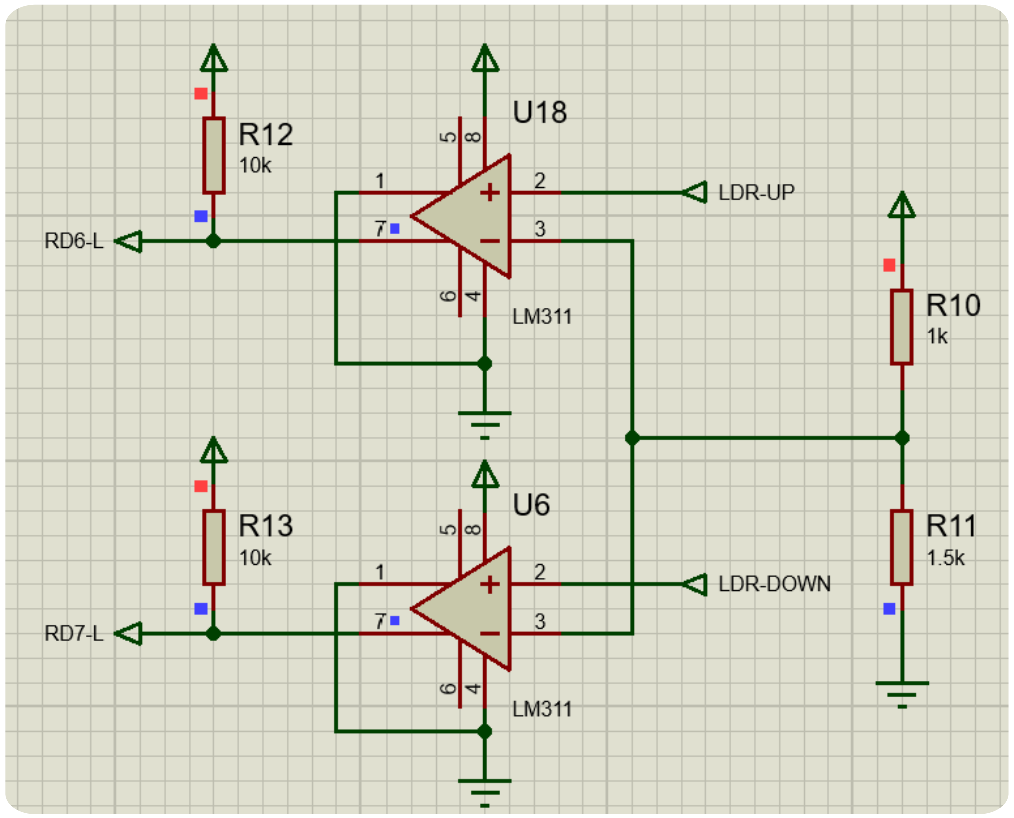
- The car park needed to display the number of cars in a car park and then provide feedback of when the car park is full.
- A binary to 7 segment display decoder was used to free up pins on the PIC.
- The screen was multiplexed, meaning each digit needed to be lit up one at a time at a very fast rate to look like all the digits are on at the same time. This allowed different numbers/letters to show on each digit.



Light Dependent Resistors

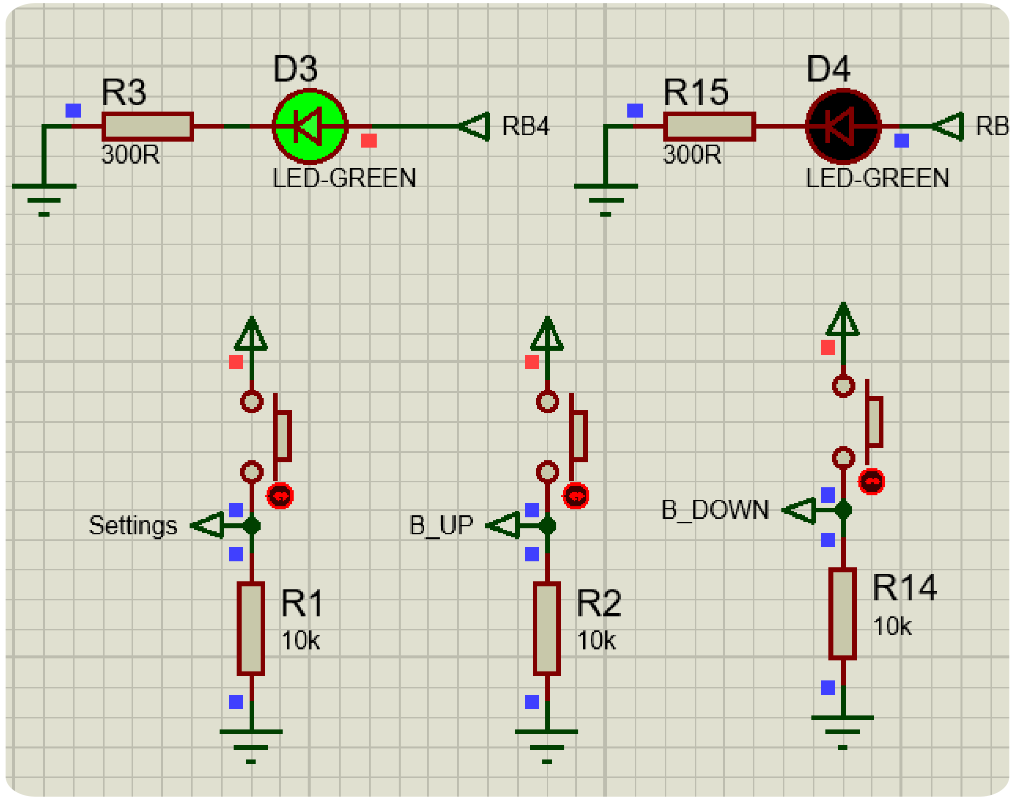
- The Car park used two light dependant resistors, which each had a laser pointing towards it. When the car would cross the gates, the laser would be blocked by the car, Telling the microcontroller that a car was entering/leaving the car park.
- The output of the LDR's each go to a comparator, to check that the light was blocked.
- The schematic shows a representation of the Laser, so you can see how the program reacts to the light being blocked/allowed through.

Comparators



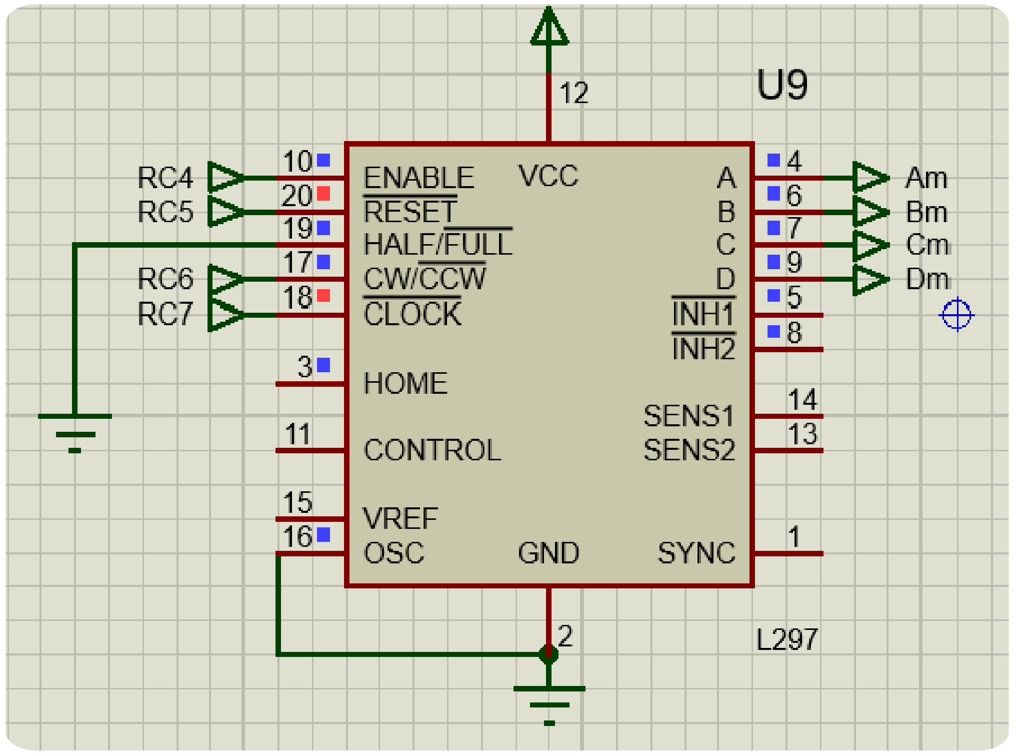
- The inputs comparators were connected to the outputs of the LDR's, and a reference voltage.
- When a LDR was blocked, the resistance goes higher and therefore the voltage that is sent is higher.
- The comparator checks to see if this voltage is higher than the reference voltage in order to send a signal to the microcontroller.

LED's and Buttons

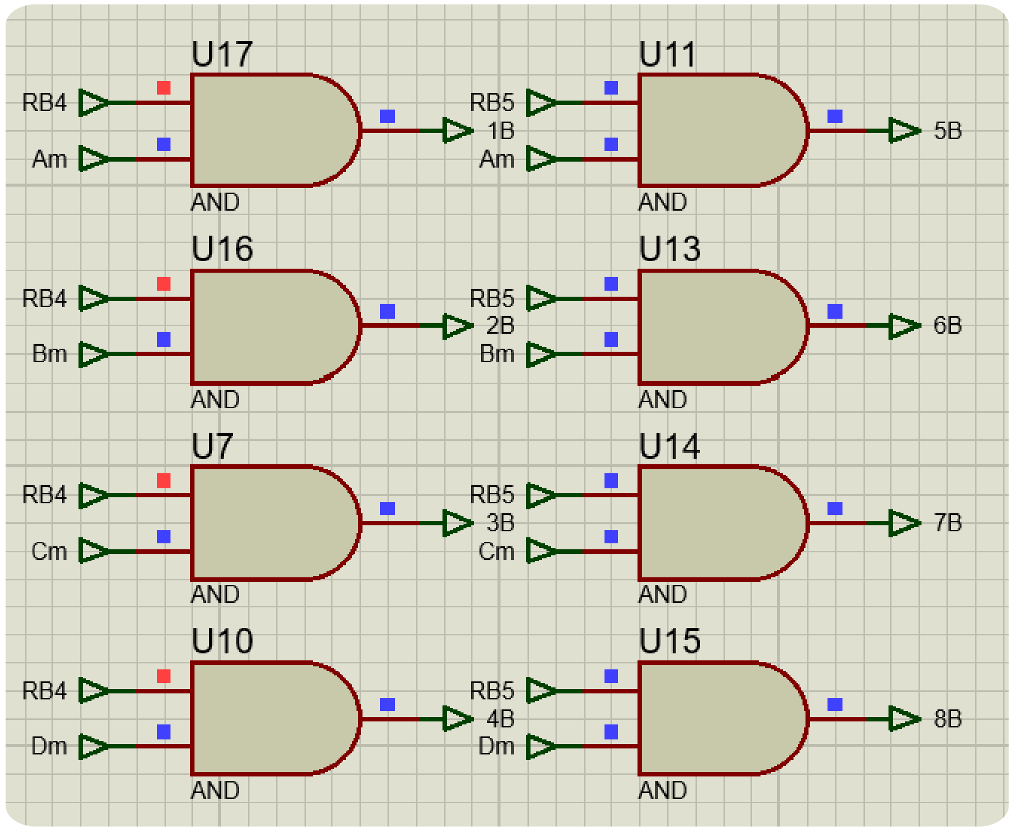


- The car park operator had access to three buttons.
- One cycled through settings, one did up, and one did down.
- LEDs indicated which setting the up/down buttons would change.
- The settings were: Motor one, Motor 2, Motor Speed and Car Counter Manual Override.

Stepper motor controller



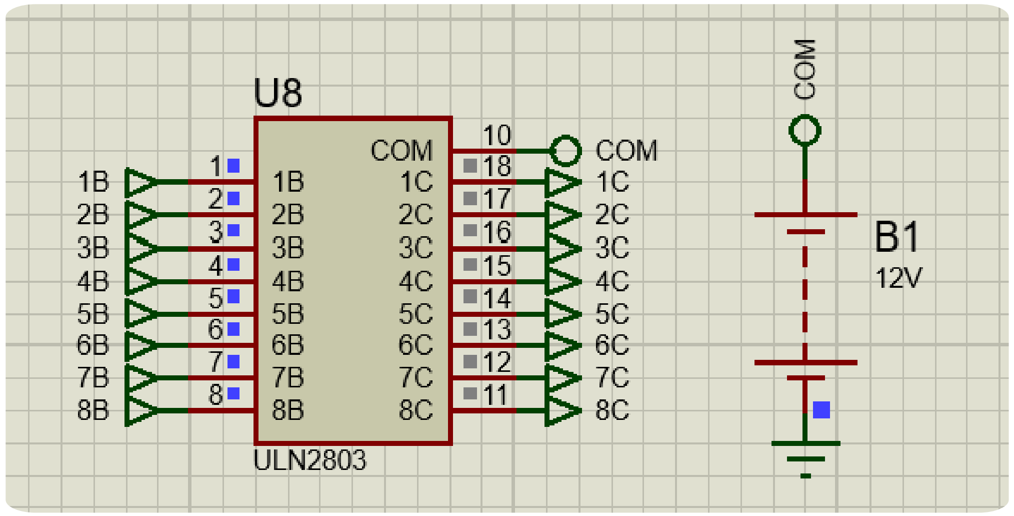
- As the project used stepper motors, they needed to be controlled by a motor controller.
- The microcontroller sends a clock signal to get the controller to change its output to turn the motors.
- The microcontroller also sent signals controlling the enable pin, the reset pin and motor direction.
- The pin that controls if the motor is doing a half step or full step is permanently shorted to ground to ensure it is always full stepping



AND Gates

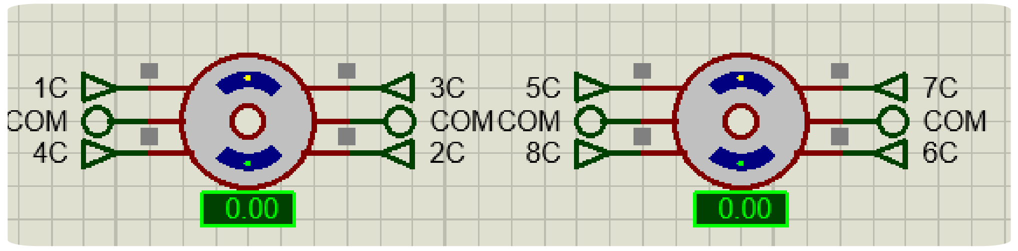
- As the car park has two motors but only one motor controller, AND gates are used.
- This is so that the microcontroller can choose where the output signals of the stepper motor controller ultimately go to.

Motor Driver

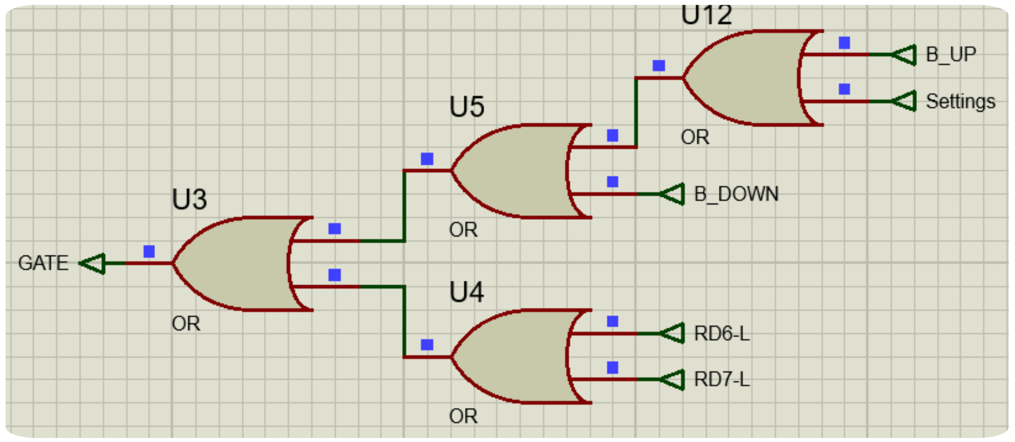


- Each AND gate output goes into the motor driver.
- The motor driver is connected to the two motors.
- As the motor needs relatively high voltage and current, a driver is needed as the AND gates can not produce the voltage or current.
- The Driver has diode protection to protect against back EMF.

Motors



- The two motors used were quad phase stepper motors.
- The stepper motor had a 7.5-degree per step rotation.
- This information was needed to work out how many steps was needed to get to the desired location.
- The motor speed is adjustable by the operator.



Universal interrupt gates

- There are multiple inputs which trigger an interrupt on this program.
- All these inputs were connected to the output of this gate array, meaning when any input changes the microcontroller knows there is an interrupt.
- Each of the input to the or gates were also connected to other input pins on the microcontroller.
- The program scans the inputs to see what caused the interrupt.

Timer Information

- As the PIC18F45K20 is a single core microcontroller, it is uncapable of parallel processing.
- This is a problem as the Car park has multiple tasks to be done at once.
- As a result, timers were used to mimic parallel processing.
- This was achieved by having timers triggering an interrupt, doing a task, and then returning from that interrupt.
- If the tasks from interrupts are short enough, it won't be noticeable that the microcontroller is only doing one task at once.