|  |  |
| --- | --- |
|  | Smart Park |
|  |  |
| 12/9/13 | Real-time traffic and congestion control. |
|  | By Jayson Francis and Karim Zahriya  Computer Engineering 185 – Fall 2013 |

Smart Park

Real-time traffic and congestion control.

Project Description

* This project is a simulation concept design of a parking garage at our university. Smart park system consists of a LCD display at the entrance of the garage, which displays the current capacity in real time. We are using sensors at the entrance that will send signals to our microcontroller (Arduino Uno) that are also operating a gate, which is operated by a servo. Every time motion is detected at the entrance, a pulse is sent to make the servo move to a 90-degree angle, opening the entrance for the gate.
* We have multiple sensors at specific parking spots inside the garage. When these sensors are “high”, it informs our microcontroller, and our program then recalculates the available capacity in the garage and updates our LCD display.
* Our long-term goal for this project will be to make the Arduino communicate data through the Internet using a Wi-Fi Shield, updating information on a SQL database. We will develop both Android and iOS applications to be able to communicate with the SQL database. The applications will then display the same information the LCD display will contain at the garage, but be able to do remotely through the Internet, making it easier for students to be updated on current traffic and congestion statuses in real time.

Part 1 – construction of the liquid crystal display

* The first part of our project was to construct our LCD display to operate correctly. This is where the main output will be in the beginning stages of our design. The arduino program that is written in the C programming language will first initiate a welcome message on the LCD, just to make sure everything is operating correctly. After a series of moving the cursor and printing out a message on the screen, the LCD screen is cleared. After the first phase of initialization in the setup() method, it then operates in a loop(), constantly watching for any changes in our parking design.

Void intro() 🡪 Method used to display a welcome message on the LCD display on startup

Void showcap() 🡪 Method used to display the available capacity current status of the parking lot.

Part 2 – construction of the sensor and servo control

* The second part of our experiment, we implement a servo controlled gate to operate the flow of incoming traffic into our parking lot. Using a sensor to detect pressure, when ever pressure is felt that the sensor value exceeds ‘69’, our program will send a pulse to the servo, making it operate a 90 degree angle, leaving it in a delay for a couple seconds, and then returning to the base case at 0, closing the gate. With this functionality, we construct a control of our parking system. When there are no spots available, the gate will not open, saving people time instead of wasting it looking for a parking spot that does not exist. It also helps traffic by not congesting the flow inside the garage with additional cars not able to get in until a spot is free.

Void servoopen() 🡪 servo1.write(90); delay(3000); servoclose();

Part 3 – construction of the parking flex sensors

* The third part of our construction we establish the ability to read data at the individual parking spots. Each parking spot will consist of a flex sensor, that will let the program know if there is a car at the position or not. If there is a flex in the sensor, it will let the program know by sending a pulse, the program then re evaluates the variable, ‘capacity’, while also updating our LCD display. Every time a vehicle leaves or enters the parking spot, our variables are reestablished, creating a real-time update of the parking situation.

if (flexpositionone > 800 && flexpositiontwo > 800){ capacity = 0; }

else if (flexpositionone > 800 || flexpositiontwo > 800){ capacity = 1; }

else { capacity = 2;}