**Car-Door-Alarm**

Project of David Burke and Matthew Drill

Topic 14.1 from "Programmable Microcontrollers With Applications" by Unsalan and Gurhan

**Abstract:**

For this project we employed the MSP430 for the assembly of a car door alarm. To this end we used code composer studio to write and execute the software (in C). The microcontroller was strapped to a breadboard which then was wired to simulate the car doors and lock, as per the directions in the textbook. ( This doesn’t take too long to get through, slow down).

**Background:**

Manufactured by Texas Instruments, the MSP 430 is a mixed- signal microcontroller built around a 16-bit CPU which can be used for low powered embedded devices. 25 MHz is the fastest CPU speed. The MSP430 may throttle for even lower power consumption. Its on-chip memory is up to 512 KB flash and 66 KB ram which is best suited for smaller applications. Combined with its USB support, this allows us to affix the MSP430 to a breadboard and build our circuit, as well as execute our C program.

**Methods:**

For this project, we wired the board to active high. We employed 5 LED’s and push buttons; four representing the car doors, one representing the car lock. 220 OHM resistors were used for the leads, buttons, and LEDs’, as opposed to the internal resistors of the MSP430 shown in the schematic. CCS was used to compile code and execute.

ALGORITHM:

If the car doors are all closed

If the alarm is pressed

        If is\_pressed is equal to 0

toggle the state of the car lock.

Set is\_pressed to 1

        Else

                Set is\_pressed to 0

If the car is locked

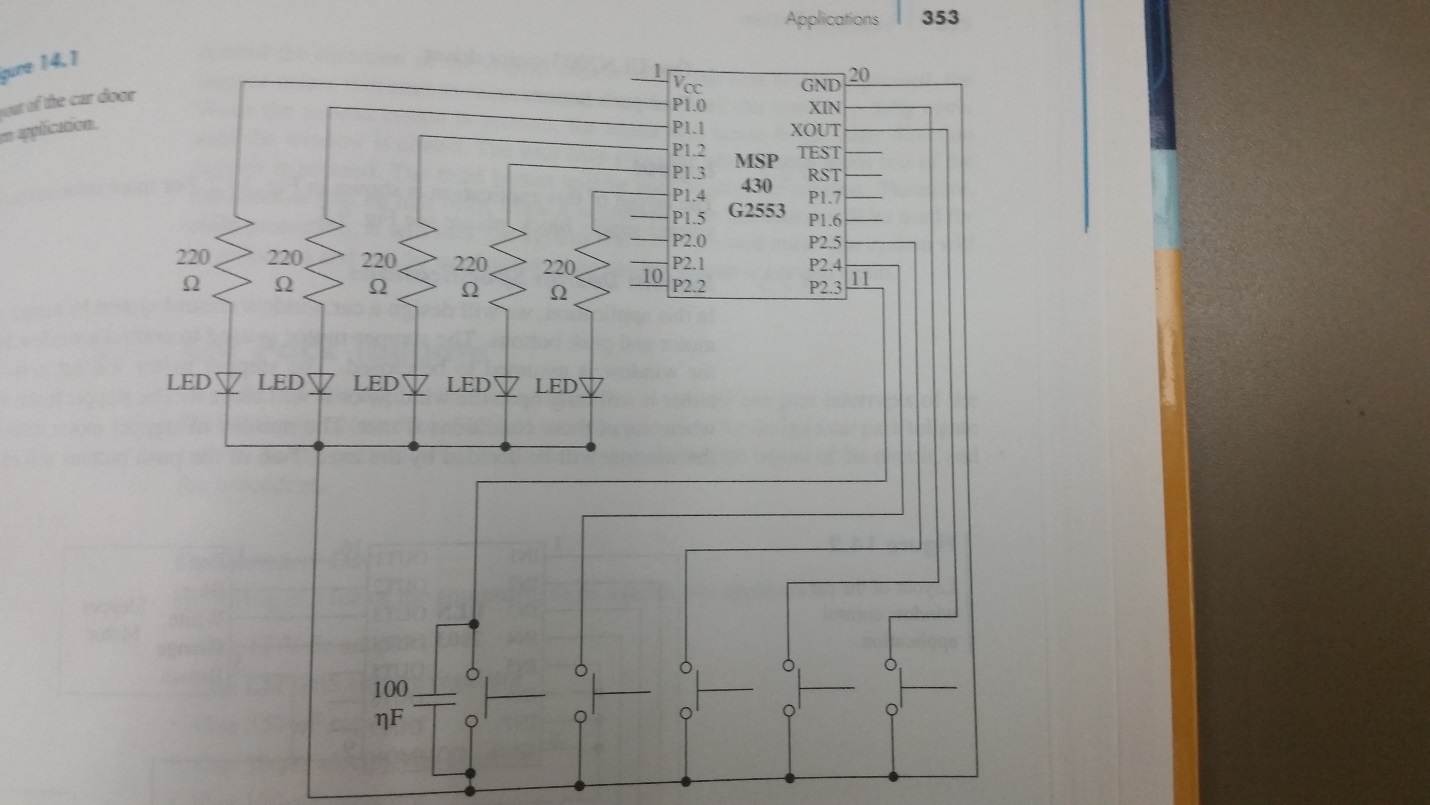
All the doors are closed

Else

the state of the doors is set to the state of the push buttons

Output the inverse of the car doors and car lock to the warning LEDs

**Schematic:**

****

We connected push buttons 2.0-2.4 as opposed to schematic which used 2.3-2.5 and XIN & XOUT. We also wired the circuit to active high rather than active low.

**Results:**

Screenshot 1:

Before code is run (Shots 1-2) look at P1OUT, P2IN, Car\_Lock

1. For P1OUT and P2IN, we are concerned only with bits 0-4. When all doors are open, we look at P2IN, the register holds all 0’s because none of the buttons have been pressed (bits 0-3) therefor the car door’s variable is set to 0. Because all of the car doors are locked, P2.5 is not set, because the car lock button is not pressed. Because neither of these are set, the car door lock variable is also 0. The is\_pressed variable is 0 because the car lock has not been toggled.

Screenshot 2:

2. All car doors have been closed but the car is not yet locked. When looking at P2IN, the first 4 bits are set. Because all 4 car door buttons have been pressed. Now the car doors variable is set to the first 4 bits of P2IN so the Car\_Doors variable (1ST 4) are set. First 4 bits of P1Out are 0 because the first 4 bits of the car doors are set and P1 outputs the inverse of the car doors. The car lock variable has not been toggled because the 5th bit of P2IN is 0. The 5th bit of P1OUT is set because it is the inverse of the car door variable

Screenshot 3:

3. Now all the doors are closed and the car is locked. So we can see when we look at P2IN all the bits are set, because all of the buttons have been pressed. So now Car\_Doors now equal to the first 4 bits of P2IN. Car\_Lock is toggled to 1 b/c the 5th bit of P2IN was set and is\_pressed was equal to 0. Now, is\_pressed is equal to one b/c after toggling Car\_Lock is\_pressed is set to 1 to ensure that Car\_Lock did not toggle. So now the first 5 bits of P1OUT are 0 because the first 4 bits of P1OUT are set to the inverse of car doors, and the 5th bit of P1OUT is set equal to the inverse of Car\_Lock.

**Discussion:**

At a glance, attempting to work from the schematic exactly as it was presented proved to be challenging. We ended up using this schematic as a template for our finished product. A few minor adjustments that we made include wiring the circuit to active high, using external resistors as opposed to those within the MSP430.

**Conclusion:**

The MSP430 was used in order to produce circuitry for a car door alarm. We used CCS to compile and execute the code for the microcontroller. The breadboard was wired to simulate the car doors and lock with LED’s for demonstration. The project has full functionality in its completed form. The project was completed with little complications.

**References:**

Ünsalan, C., & Gürhan, H. D. (2014). Programmable microcontrollers with applications: MSP430 LaunchPad with CCS and Grace. New York: McGraw-Hill Education.