Leonardo DaGraca
CS7680
HW3
Task C

1) How do you know that your Raspberry Pi GPIO pins are in safe states at the end of your code? Why is this important?

The Raspberry Pi GPIO pins are in a safe state at the end of the code because their modes are changed to input mode. This is the original state of the pins and setting the mode back to the input at the end of the program ensures that it returns to that state with no voltage coming out. It is important not to have the pins set to output after the completion of a program because volts could still be coming out of the pin. This can lead to the pin or Pi being damaged, a short-circuiting issue, or any components connected to the Pi being damaged.

2) Describe without writing code how you might adjust the traffic light configuration of Tasks A and B by introducing a road pressure plate (a device that signals to the system that a vehicle is waiting at the traffic signal for a change in direction). What might be needed to adjust the hardware circuit? What might be needed in the software to support this input signal? How might the traffic signal operational loop be modified?

Task A’s original configuration was to have the three LEDs flash in sequential order, where the red LED would flash 3 times, the yellow would flash 3 times, and then the green 3 times. Task B’s original configuration was to emulate a traditional intersection traffic signal where the red LED illuminated followed by the green and then the yellow.

By adding a road pressure plate we would have to change our hardware circuit so that it can receive input from the pressure plate. We would have to place pins/wires to and from the pressure plate so that the input signal is correctly picked up. This would mean that we would dedicate a GPIO pin for the pressure plate on the Pi cobbler (where it will connect to the Pi) and connect a wire to the plate to communicate the signal on our circuit. We would also add resistors where needed to regulate the voltage from the pins.

The software would then need to include this input value and be able to read the input. The software would also use the input to manage the current state it is in. If the input exists we would handle the LED lighting accordingly. For Task A, our loop would change by initially checking the signal state from the pressure plate. If there is one we’d let the traffic system know there is currently a vehicle present waiting on a change in direction and could prioritize the green LED to allow the vehicle to pass by or diminish the waiting time. If there is not any signal coming then our original traffic signal loop would take place. For Task B, we would be checking for the signal when the red LED is on. This would allow the traffic light system to detect that vehicles are waiting for a change in direction and that the change can occur. If the signal is detected then the system could diminish the time it spends on the red LED and transition to the green LED quicker. If there is no signal the loop will run in its original state. When the green LED is on vehicles will be passing by so keeping track of the pressure plate signal is not of importance to us.

3) Consider a problem solution where illuminating a light/LED would be useful. Design a different circuit that could be used and describe (not code) the software rules and flow that would be necessary to deliver that solution.

A problem solution where illuminating an LED would be useful is for an elevator button. The goal would be to have the elevator button's LED illuminate when it is pressed and after the button task is completed the LED would turn off.

To design such a circuit we would take an elevator button that has an integrated LED that lights up when the button is pressed. We would then connect one of the button's pins to a GPIO pin and connect the LED control pin to another GPIO pin that will control whether the LED is turned on or off. The pin that was not connected would then be connected to a ground pin on the circuit. Lastly, we would add resistors where needed to regulate the voltage from the pins.

In our software, we would initialize the button as an input pin and read in its current state. We would also configure the GPIO pin connected to the LED as an output pin to control when the LED turns on or off. The software would be written to then continuously read the state of the button input. If the button is not pressed then the LED will not be illuminated. Otherwise, if the button is pressed, our state will change and the LED will illuminate. Furthermore, within this state, we will keep track of the button's task, so that when the task is completed we can return to the initial state and turn the LED off.