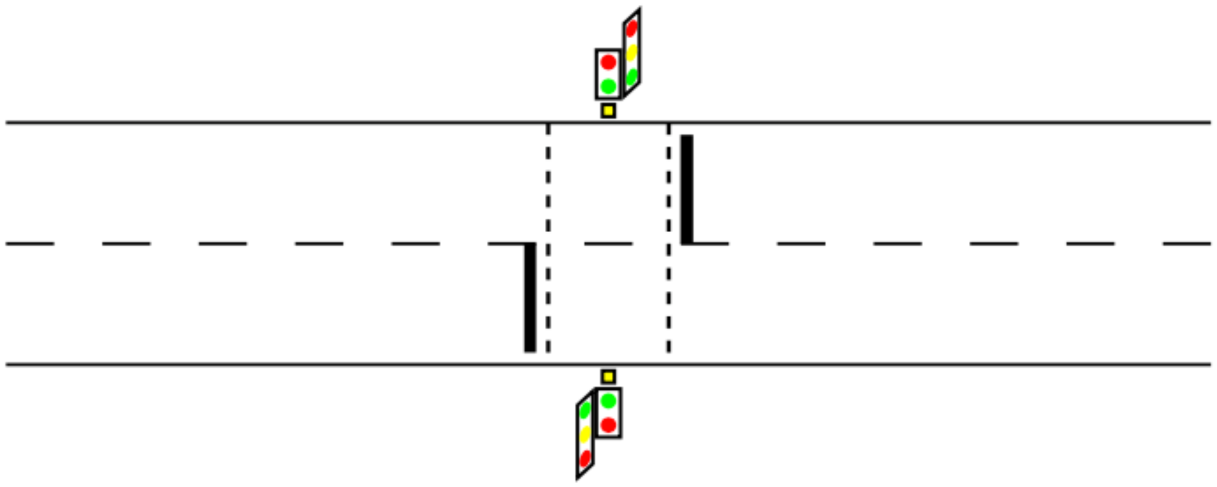
MENGLONG YUAN



Pedestrian crossing traffic light system

Prepared by:Menglong Yuan

6 March 2017

1. German Traffic Light Sequence and Delay

Traffic signals in Germany use the same red, yellow, and green lights found in the US and elsewhere. However, an extra indication is added: just before the light changes from red to green, the yellow signal comes on briefly in conjunction with the red. This means PREPARE FOR GREEN and is helpful if you are driving a manual transmission to give you a bit of warning to get into gear. [1]

The clearing time of the traffic lights ensures that pedestrians can cross the whole protected area at a speed between 1.0 and 1.5 m / s (usually 1.2 m / s) to the next safe area.[2] Assuming a single-lane road is 7 meters wide, the pedestrian crossing time set for the traffic light is about 6 seconds.

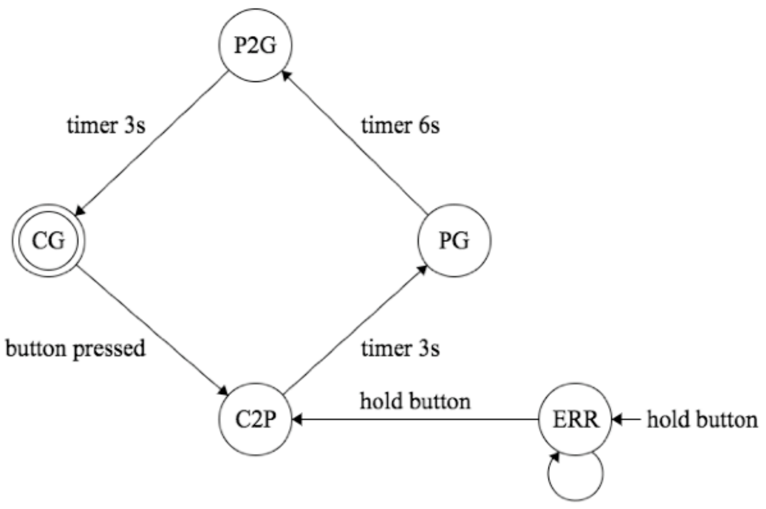
2. Implementation

The complete implementation can be found and run in the simulator:

https://circuits.io/circuits/4194604-pedestrian-crossing-traffic-light-system

The entire traffic light system can be describe in a state machine with the following 5 states:

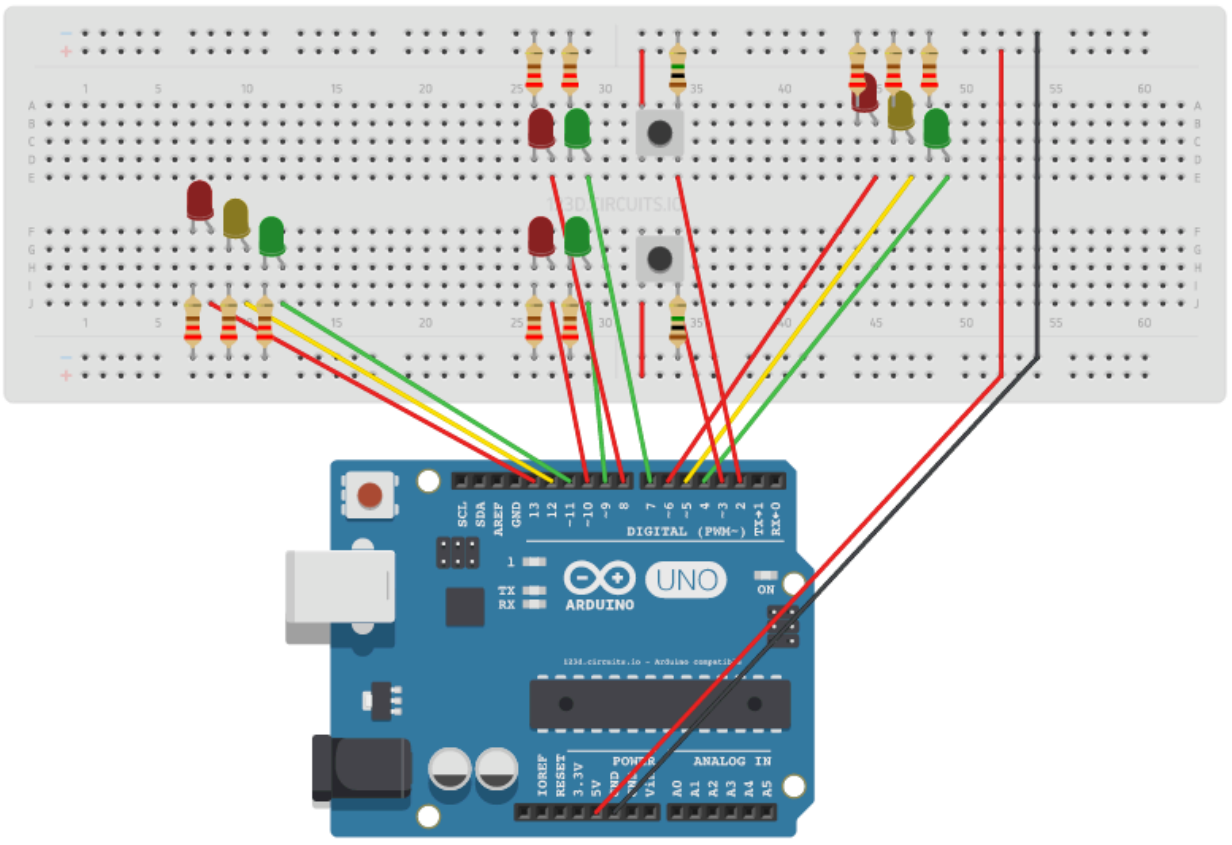
1. CG. Car goes, pedestrian stops. car_led green, ped_led red.
2. C2P. Pedestrian presses the button. car_led yellow, ped_led red. Last 3 seconds.
3. PG. Pedestrian goes, car stops. car_led red, ped_led green. Last 6 seconds.
4. P2C. Time runs out for pedestrians. car_led red and yellow, ped_led red. Last 3 seconds.
5. ERR. Error state. car_led yellow blinks.



[1] http://www.gettingaroundgermany.info/zeichen2.shtml#signals

[2] https://de.wikipedia.org/wiki/Ampel#Fu.C3.9Fg.C3.A4ngersignal

3. Set up the Hardware



4.a Expand this code to check your wiring and make sure all your LEDs blink as expected.

Check the corresponding source code under /src folder.

4.b Implement the Traffic light system as simply/quickly as you can.

Check the corresponding source code under /src folder.

5 Use the following techniques to implement the System in a neat and easy to read way:

- Use a timer rather than delay()s between each change, so that your run loop would be free to handle other tasks
- Use a Finite State Machine (or more if you like) to implement the system.

Check the corresponding source code under /src folder.

6 Even if the simulator does not allow this, please explain how a real timer would work rather than the delay() based one which you used in 5).

Check the corresponding source code under /src folder. Instead of using the delay() function, I include the Timer1.h library to utilize the 16 bit hardware timer on the chip. I set the Timer1 with a 0.1 second period and attach update_timer() as the timer overflow interruption. The hardware timer will updates the time every 0.1 second.

7 Even if the simulator does not allow this, please explain how a hardware Interrupt for the button would be used.

Check the corresponding source code under /src folder. Instead of checking the button pins every loop, 2 interruptions are attached to the button pins, so that whenever the button is pressed or unpressed, the interrupt will handle the corresponding button status.