In order to successfully manage the stop lights we must have the following:

1) **Logic for passiveness**

When no button is pressed after 120s we check to see what lights aren't green and we change them to green. We reset all variables, and the timer.

2) **Timer for cycle**

We create a timer which increments upwards to 120s from 0s, it is reset on either a successful button press, or by default through passiveness, an unsuccessful button or ignored button will, by default, be considered passive.

3) **Logic for button**

On user input, we first check if a user had successfully used the button last cycle. If they did, we do three things, first we set the **currentCycleWasPressed** variable to true, in order to prevent double button presses. Next we set the **lastCycleWasSuccessful** variable to false, in order to indicate to the next cycle that a button was not successful the previous. Lastly we get the current cycle time, and loop until it is greater than or equal to 120s, change the light, then reset all variables and the timer. If a user had not successfully used the button last cycle, which will be the case if no one pressed the button last cycle, or if someone had had an unsuccessful button press this cycle (we need to check because we reset the variable), we check if the user has pressed the button at all this cycle. If they did, we ignore their input. If they did not, we check if the current cycle time is greater than 60s. If it is, we return unsuccessful, set **currentCycleWasPressed** to true, get the current cycle time, and then loop until the timer reaches 120s. If the timer is less than 60s we check which lights are green. Wait 30s, change them accordingly, set **lastCycleWasSuccessful** to true, reset the timer and reset the loop.

Test Cases:

Wait one cycle
Wait two cycles

Press button before 60s
Press button after 60s

Wait one cycles, press button before 60s
Wait one cycles, press button after 60s

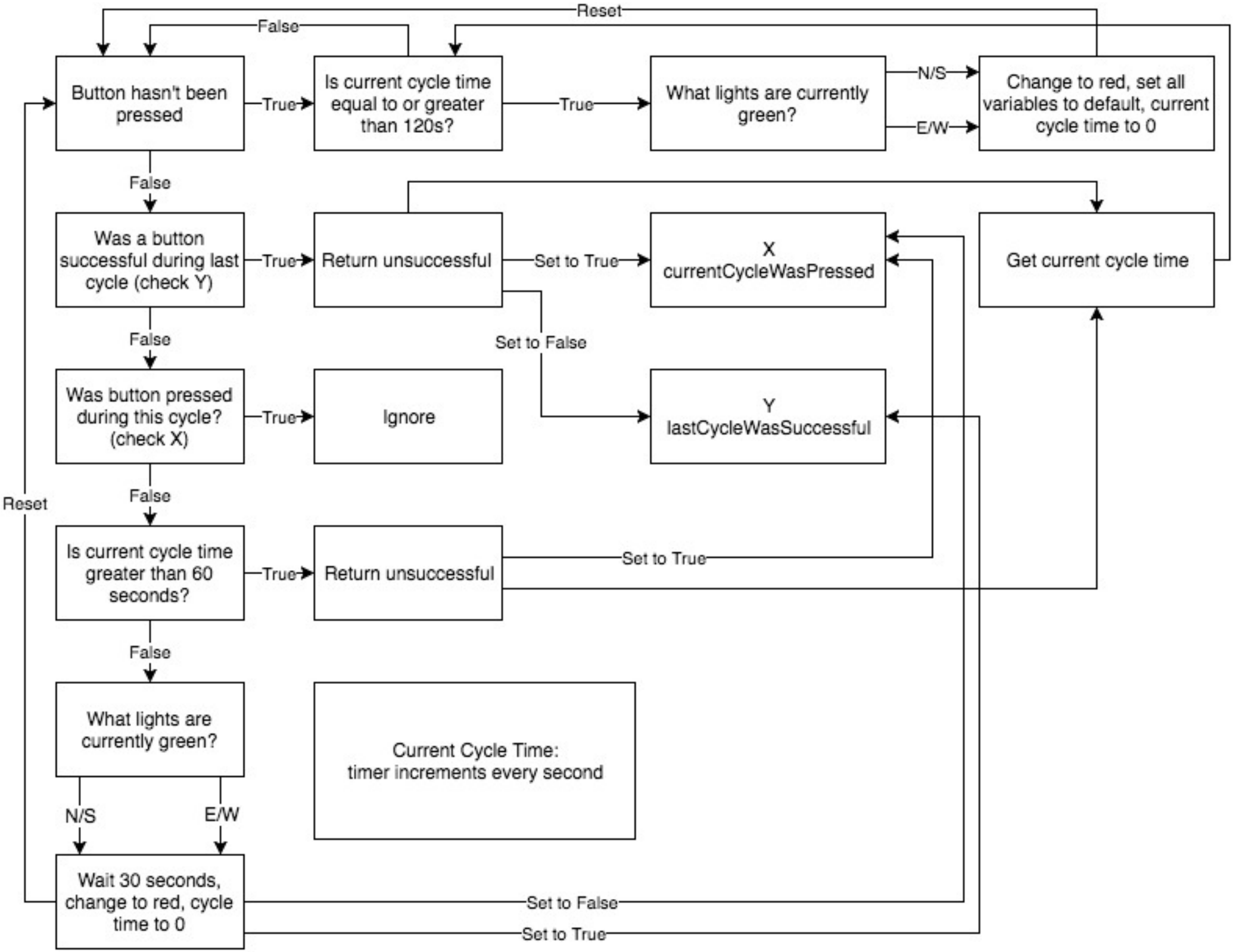
Wait two cycles, press button before 60s
Wait two cycles, press button after 60s

Press button twice before 60s
Press button twice after 60s

Press button before 60s, then press again after 60s

Press button before 60s, then press after 60s next cycle
Press button after 60s, then press before 60s next cycle
Press button before 60s, then press before 60s next cycle
Press button after 60s, then press after 60s next cycle

Press button before 60s, wait full cycle, press before 60s
Press button before 60s, wait full cycle, press after 60s
Press button after 60s, wait full cycle, press before 60s
Press button after 60s, wait full cycle, press after 60s


Reset
False
Button hasn't been pressed
True
Is current cycle time equal to or greater than 120s?
True
What lights are currently green?
N/S
E/W
Change to red, set all variables to default, current cycle time to 0
False
Was a button successful during last cycle (check Y)
True
Return unsuccessful
Set to True
X currentCycleWasPressed
Get current cycle time
False
Set to False
Was button pressed during this cycle? (check X)
True
Ignore
Y lastCycleWasSuccessful
Reset
False
Is current cycle time greater than 60 seconds?
True
Return unsuccessful
Set to True
False
What lights are currently green?
Current Cycle Time:
timer increments every second
N/S
E/W
Wait 30 seconds, change to red, cycle time to 0
Set to False
Set to True