Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
CSE233: Computer Organization

Lab #2 Report

Manual Pedestrian Traffic Lights

Names:

1. Mohamed Abdalla Yassen Mohamed (23010765)
2. Ahmed Mohamed Saied Mohamed (23011684)

1. Problem Statement

Implementation of Manual Pedestrian Traffic Lights using Arduino.

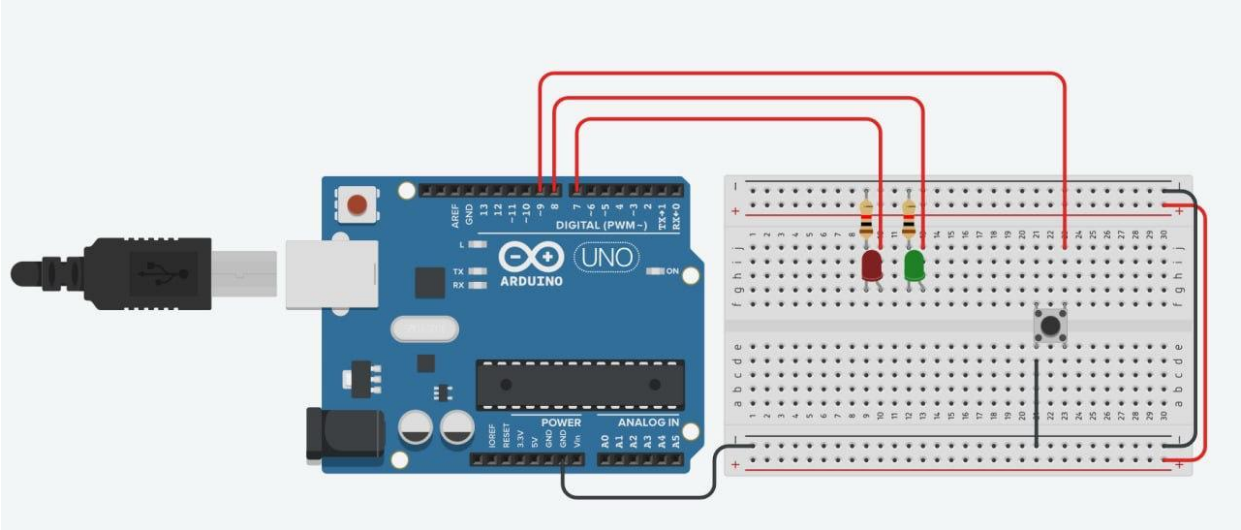
2. Code “GitHub Rebo”

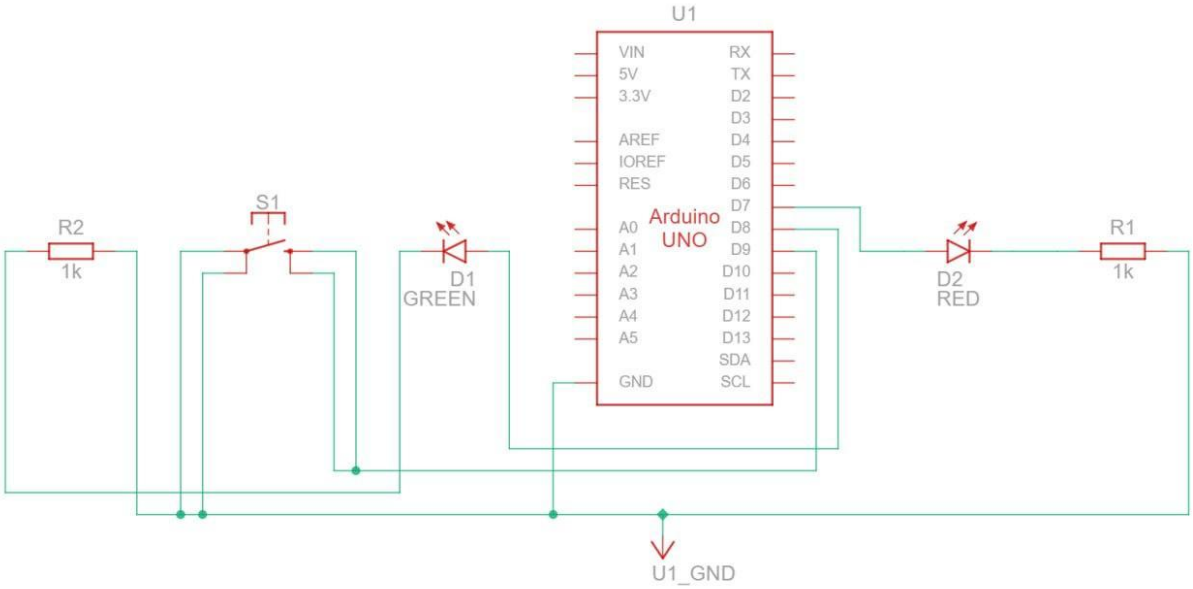
GitHub:
https://github.com/Mohamed-Abdalla-Yassen/Computer-Organization-Projects.git
Arduino web editor:
https://app.arduino.cc/sketches/5c5341c5-6647-4123-b875-c30207b01e50?view-mode=preview

3. Video “YouTube”

https://youtu.be/k-MxY7-eHX8

4. Schematic diagram





5. Description & Challenges

Button Debouncing:

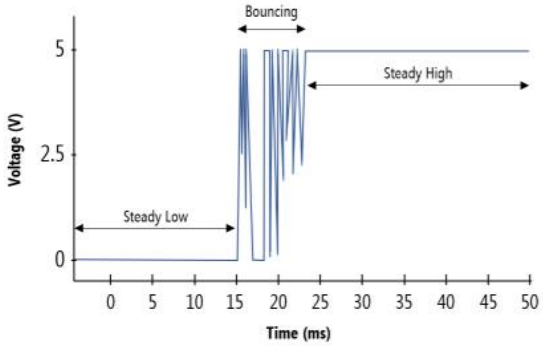
When you press normal pushbutton, two pieces of metal come into contact with each other. If these two tiny sheets of metal aren't perfectly flat or perfectly aligned (and they're not) then they can make and break contact a few times before being firmly enough squished together that they're always conducting.

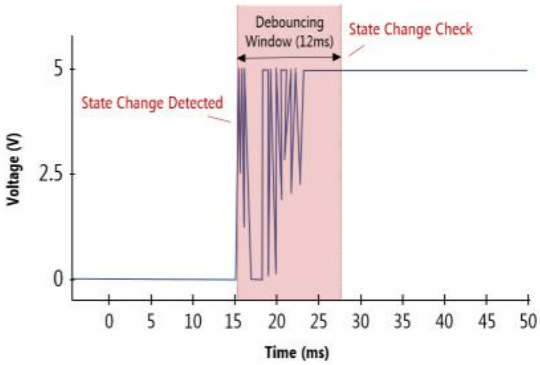
This causes the microcontroller to pick up multiple false inputs giving the hallucination that the button is flickering and the picture below demonstrates this problem. After we search we figured out this problem was two solutions; a Hardware solution and a software solution.

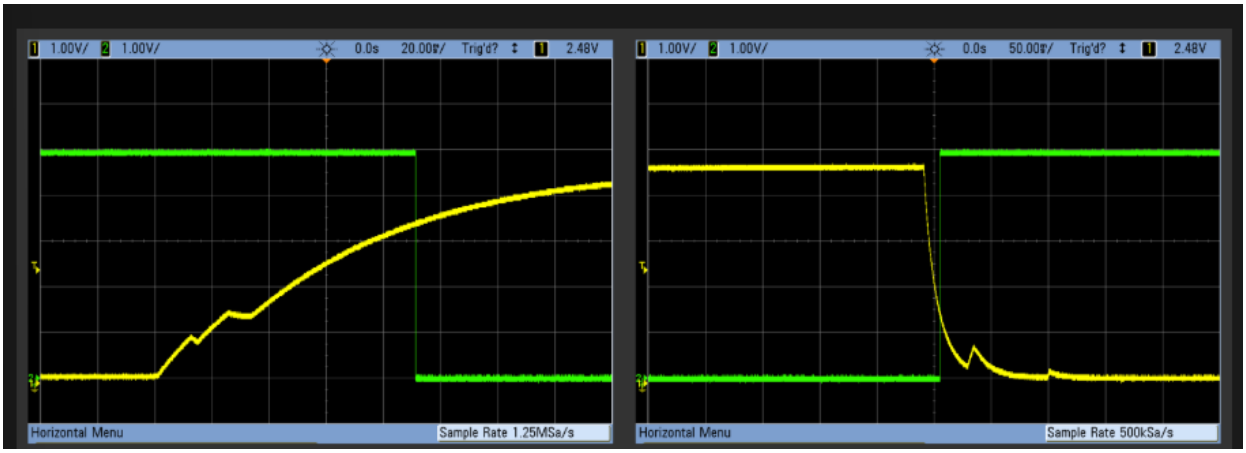
The Hardware solution:

We want to eliminate the False input in the debouncing window to do so we can introduce a Capacitor to the switch, Why?

The Capacitor will accumulate the voltage until it reaches 5 Volt its more like a smooth curve rather than as shown in photo(2).This will eliminate the debounce and we will have a fully functional circuit.







The Software solution:
As we can see that debouncing basically occurs when first clicking on the button, So to avoid taking and debouncing we made sure to keep track of the zero state (S_O) and the State when pressing the button (S_N), We only want to pick the Signal when the (Time of S_N – Time of S_O) is more than the debouncing window and we assumed it is 50ms and we only registered The signal when S_O is 0 and S_N is 1 indicating a change in the state. This approach will guarantee us a fully functional circuit as well.
We applied the software approach in this Lab

A floating state:
in Arduino means the input pin is not connected to anything — it's left "open."
When this happens, the pin doesn't know if it should read **HIGH (1)** or **LOW (0)**, so it randomly changes between them because of tiny electrical noise in the environment.
This can cause **unstable or unpredictable behavior** in your program — for example, a button input might trigger randomly even when not pressed.
To fix this, you use **pull-up** or **pull-down resistors**:

- **Pull-up resistor:** connects the pin to **Vcc (5V)** so it reads HIGH by default.
- **Pull-down resistor:** connects the pin to **GND** so it reads LOW by default.

In Arduino, you can easily enable an **internal pull-up resistor** using:
pinMode(pin, INPUT_PULLUP);
This keeps the pin stable without adding an external resistor.