

Polling Based Implementation:

Polling is a method where a device continuously checks the status of another device to determine if it needs attention. This involves sending requests or queries to the device, waiting for a response, and repeating this process in a loop until the desired condition or data is obtained.

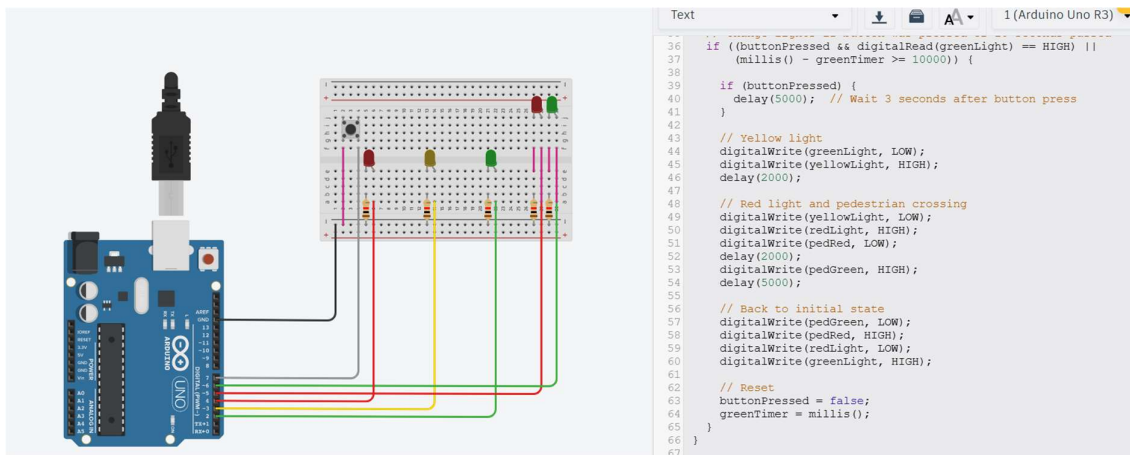
Advantages of polling include:

- **Simplicity:** It is a simpler mechanism to implement and provides straightforward control over the process.
- **Compatibility:** Polling can work on systems that do not support interrupt-driven mechanisms, making it widely applicable across various platforms.
- **Predictability:** Since the system checks for data at regular intervals, the behaviour is predictable and can be fine-tuned for specific performance requirements.

Disadvantages of polling include:

- **Resource Intensive:** Constantly checking for data or events consumes system resources, including CPU cycles, leading to inefficiency and wasted power.
- **Latency:** There can be a delay between the occurrence of an event and the system detecting it, especially if the polling interval is long, leading to higher latency in event processing.
- In real-time applications where immediate responses are critical, polling may not be suitable.

My Implementation:



- The button is working.
- When the system is not interrupted by a button, the green light remains on for about 10 seconds before going to yellow.
- Conversely, when it is pressed, the green light stays on for about 5 seconds and then goes to yellow.

Interrupt-based

- Interrupts are signals sent by devices to the CPU to indicate that they need attention. When a device has data ready or an event occurs, it sends an interrupt signal to the CPU, causing the CPU to stop its current activities and attend to the interrupting device.
- For example, a microcontroller can be interrupted by a sensor when it has a new value or by a serial port when it receives data.
- Interrupts can be either hardware (keyboard for example) or software based (components running on the processor).

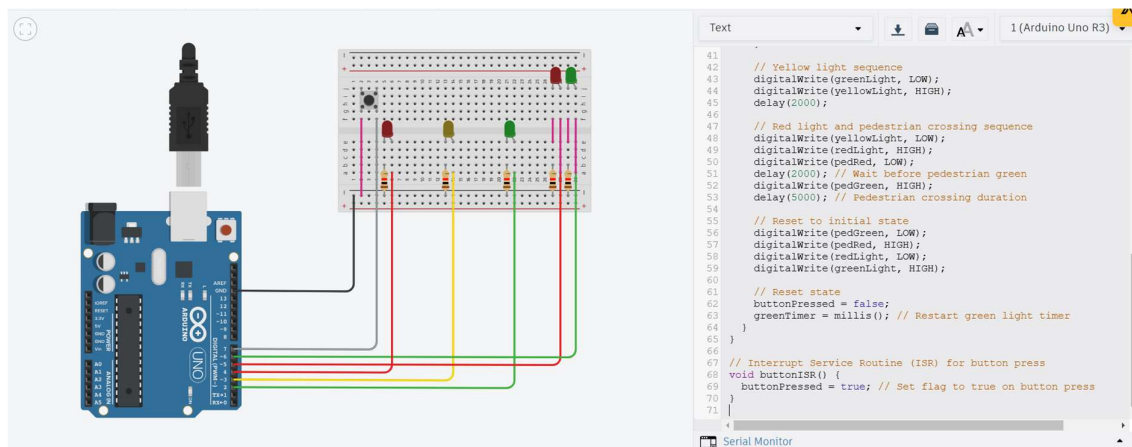
Advantages of interrupts include:

- **Efficiency:** Systems that employ interrupts are generally more efficient as the CPU can focus on other tasks while waiting for external events, rather than constantly polling.
- **Reduced Latency:** Devices can be serviced immediately when they have data to transfer, leading to faster response times, reducing latency.
- **Real-time Responsiveness:** Interrupts enable immediate responses to external events, making them essential for real-time applications where timely reactions are critical.
- **Prioritization:** Interrupts can be prioritized, allowing the system to respond to the most important events first, ensuring essential tasks are handled promptly.

Disadvantages of interrupts include:

- **Complexity:** Implementing and managing interrupts can be complex, involving intricate synchronization and handling mechanisms, especially in multi-device environments.
- **Task Interference:** In some cases, interrupts can restrict the simultaneous execution of tasks as the CPU's attention is frequently diverted to handle interrupt requests.

My implementation:



- The button doesn't work.
- Regardless of the button being pushed or not, the green button only goes to yellow after 10 seconds.