Title:

Part-2: Digital Timer project using millis() function to avoid the problems associated with delay().

Theory & Methodology:

The conventional delay() function in Arduino programming can be restrictive because it freezes the program's state, hindering input and output operations. To address this, the millis() function is utilized, which tracks the time in milliseconds since the Arduino started running. By storing this time in an unsigned long variable, the Arduino can effectively keep track of time for extended periods, up to nearly 50 days, without overflow issues. This allows for more flexible timing operations, such as triggering events at specific intervals, without halting the program's execution.

Moreover, the use of a tilt switch as a digital input adds a unique dimension to the project. Tilt switches, which function similarly to regular switches but detect orientation changes, are employed to initiate actions based on the device's tilt. In this case, the tilt switch is utilized to reset the timer when the device is turned over, triggering another cycle of LED activations. With six LEDs in the setup, the timer is designed to run for a total of six minutes, each LED representing a one-minute interval.

Overall, by leveraging millis() for time tracking and incorporating a tilt switch for input detection, the Arduino digital timer project gains flexibility and functionality. This approach enables precise timing operations and enhances user interaction through orientation-based triggering, making the timer more versatile and practical for various applications.

Overview of Arduino Uno Board:

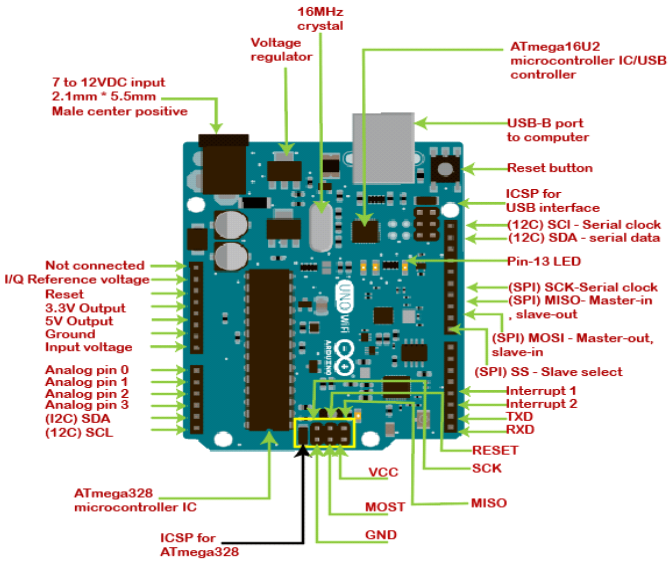

Figure 1: Pinout diagram of an Arduino Uno Board

Overview of Tilt Sensor:

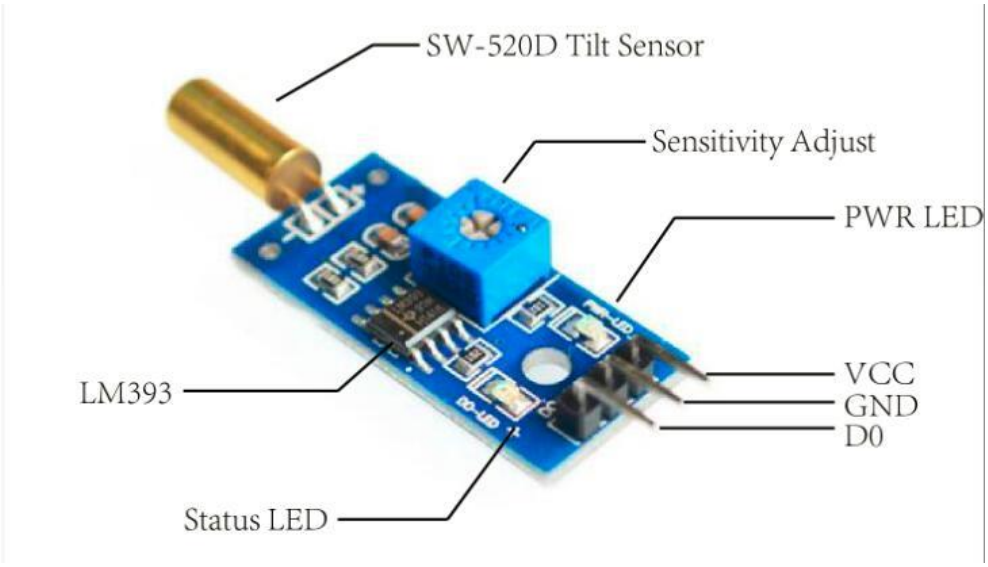

Figure 2: Pinout diagram of Tilt Sensor

Apparatus:

- Breadboard
- Arduino Uno/ Arduino Mega
- Tilt sensor.
- LED lights (Six)
- Resistors (One 10K and six 220 ohms)

Experimental Procedure:

To conduct the experiment for creating a digital timer using the millis() function on an Arduino Uno or Mega, we've collected essential components. Then, we connect the tilt sensor to one of the digital pins, such as pin 2, and wire six LED lights to digital pins 3 to 8, each through a 220-ohm resistor. Additionally, connect a 10K ohm resistor between the tilt sensor's output pin and ground to act as a pull-down resistor, ensuring stable readings. Once the hardware is configured, we proceed to code the digital timer in the Arduino IDE.

In the code, start by initializing variables to hold the pin numbers for the tilt sensor and LEDs, then set up the pinMode for each pin, specifying whether it's an input or output. After that, serial communication is initialized, and pin modes are configured in the setup function. Within the loop function, utilize the millis() function to create a non-blocking delay for reading the tilt sensor and toggling LEDs. Implement logic to detect changes in the tilt sensor's state and trigger LED toggling, accordingly, incorporating debouncing techniques to prevent false triggers.

Once we have written the code, verify it for any syntax errors and upload it to the Arduino board. Open the Serial Monitor to observe output messages and ensure proper functioning. We test the digital timer by tilting the sensor and observing the LEDs' behavior; each tilt should toggle the LEDs' state. Refine the code as necessary, adjusting timing parameters or sensor placements to optimize sensitivity and responsiveness. We ensured that the code is well-commented and organized for clarity and future reference.

Hardware Picture'(s):

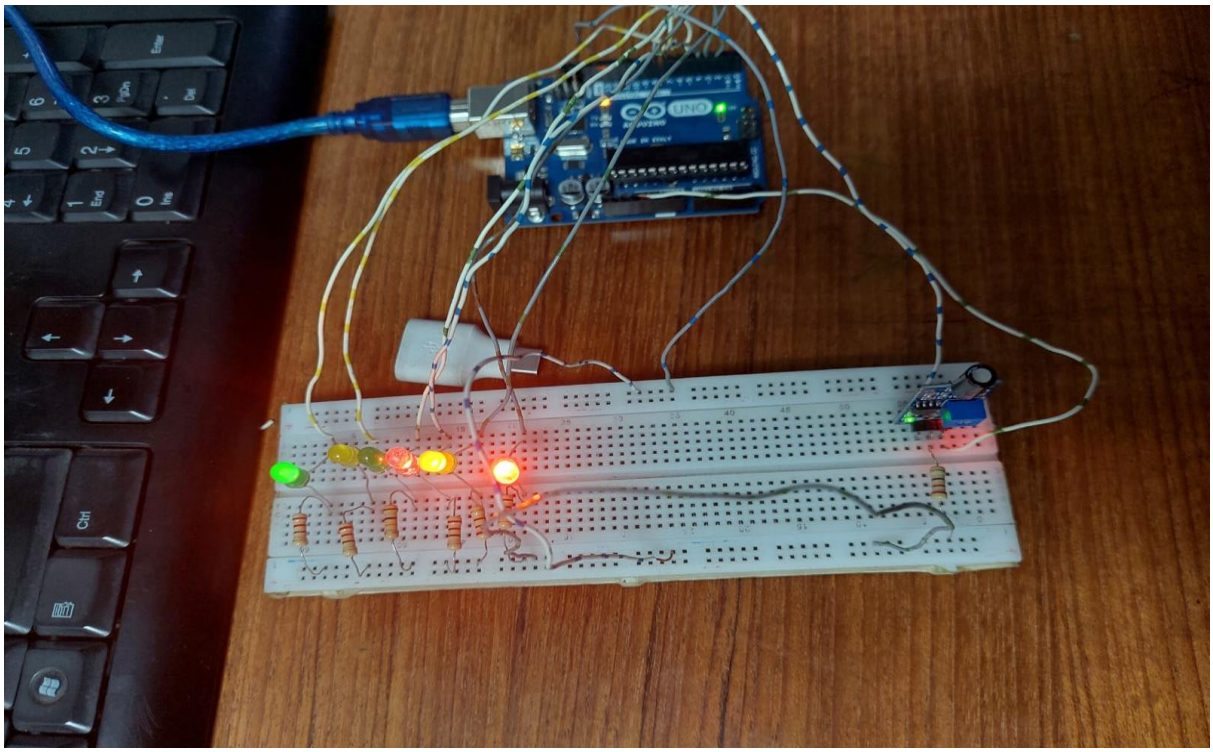

Figure 3: All LEDs turned ON.

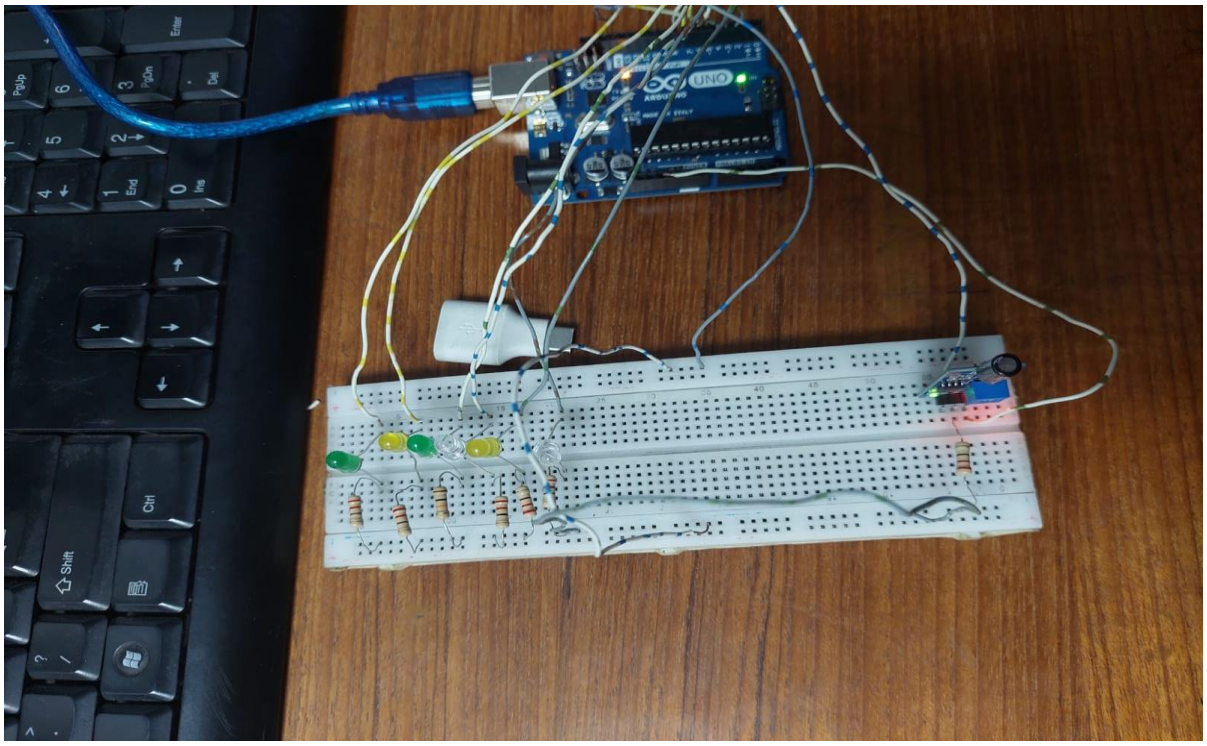

Figure 4: All LEDs turned OFF after giving vibration to the Tilt Switch

Discussion:

The development of a digital timer system using the millis() function represents a practical application of Arduino programming techniques. Through observations gathered from various sources, it was noted that the millis() function serves as a fundamental tool for tracking time intervals in Arduino projects, offering an alternative to the delay() function. One significant advantage highlighted was the non-blocking nature of millis(), allowing concurrent execution of other code while waiting for specified time intervals. However, it was also recognized that millis() may pose challenges in precise timing applications, which could limit its suitability for certain projects. Despite these drawbacks, the experiment proceeded with the implementation of the digital timer system, incorporating components such as resistors, LEDs, a tilt switch, and an Arduino UNO board.

The developed system was based on the identified logic and methodology, with code tailored to utilize the millis() function for controlling LED illumination based on specified time intervals. The results of the system's operation were observed and noted, with simulations conducted using software such as TinkerCAD and Proteus for verification. The outcomes obtained from both hardware and software implementations were found to align with expectations, achieving the objectives set forth in the experiment.

Conclusion:

In conclusion, the experiment successfully demonstrated the development of a digital timer system using the millis() function on an Arduino platform. By leveraging the capabilities of millis() and integrating components effectively, the system was able to accurately control LED illumination based on predefined time intervals. Although challenges and drawbacks associated with millis() were acknowledged, the experiment's outcomes indicated that the implemented solution met the desired objectives. Through careful observation and documentation, the experiment provided valuable insights into the practical application of Arduino programming techniques and the utilization of timing functions for real-world projects. Overall, the experiment's success highlights the versatility and effectiveness of Arduino-based systems in implementing various functionalities, paving the way for further exploration and experimentation in this domain.

Reference(s):

1) https://www.arduino.cc/.
2) ATMega328 manual
3) https://www.avrfreaks.net/forum/tut-c-newbies-guide-avr-timers
4) http://maxembedded.com/2011/06/avr-timers-timer0/