NIELIT, Gorakhpur

Course Name: A-level (1st Sem.) **Subject:IoT**

Topic: Watchdog Timer In Arduino **Date: 22.04.2020**

Introduction

The ATmega328P has a Watchdog Timer which is a useful feature to help the system recover from scenarios where the system hangs or freezes due to errors in the code written or due to conditions that may arise due to hardware issues.

How does the Watchdog timer work?

The watchdog timer needs to be configured according to the need of the application.

The watchdog timer uses an internal 128kHz clock source.

When enabled, it starts counting from 0 to a value selected by the user. If the watchdog timer is not reset by the time it reaches the user selected value, the watchdog resets the microcontroller.

The ATmega328P watchdog timer can be configured for 10 different time settings (the time after which the watchdog timer overflows, thus causing a reset).

The various times are : 16ms, 32ms, 64ms, 0.125s, 0.25s, 0.5s, 1s, 2s, 4s and 8s.

Example

Let’s see how to configure the watchdog timer for an Arduino UNO board.

Here, we will use a simple example of LED blinking.

The LEDs are blinked for a certain time before entering a while(1) loop. The while(1) loop is used as a substitute for a system in the hanged state.

Since the watchdog timer is not reset when in the while(1) loop, the watchdog causes a system reset and the LEDs start blinking again before the system hangs and restarts again. This continues in a loop.

Here, we will be using the on-board LED connected to the pin 13 of the Arduino UNO board. For this example sketch, the only thing required is the Arduino UNO board.

Word of caution :

The Watchdog timer is disabled at the start of the code. A delay of 3 seconds is used before enabling the Watchdog.

This delay is important in order to let the bootloader in Arduino to check if a new code is being uploaded and to give it time to burn the code into the flash.

This is important as a precaution. A situation may occur, wherein due to faulty coding, or improper considerations; the code written resets the microcontroller at very short durations infinitely.

This will damage the Arduino board and lead to sketches not being uploaded to the board.

This may not be the case for the new Optiboot loader that comes with the newer version of the Arduino, but it will definitely happen to the older ones.

In case if you break the Arduino in this manner, you will have to burn the bootloader using a different Arduino as an ISP into the bricked Arduino.

Sketch

```
#include<avr/wdt.h> /* Header for watchdog timers in AVR */
void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
  Serial.println("Watchdog Demo Starting");
  pinMode(13, OUTPUT);
  wdt_disable();  /* Disable the watchdog and wait for more than 2 seconds */
  delay(3000);  /* Done so that the Arduino doesn't keep resetting infinitely in case of wrong configuration */
  wdt_enable(WDTO_2S);  /* Enable the watchdog with a timeout of 2 seconds */
}

void loop() {
  for(int i = 0; i<20; i++) /* Blink LED for some time */
  {
    digitalWrite(13, HIGH);
    delay(100);
    digitalWrite(13, LOW);
    delay(100);
    wdt_reset();  /* Reset the watchdog */
  }
  while(1); /* Infinite loop. Will cause watchdog timeout and system reset. */
}
```