PERCOBAAN V
Sistem Fault Tolerant menggunakan Watchdog

5.1 Tujuan Percobaan

Dalam bab praktikum ini diharapkan praktikan dapat menerapkan *watchdog* sebagai salah satu cara untuk membangun sistem yang *fault tolerant*

5.2 PERANGKAT YANG DIGUNAKAN

Perangkat:

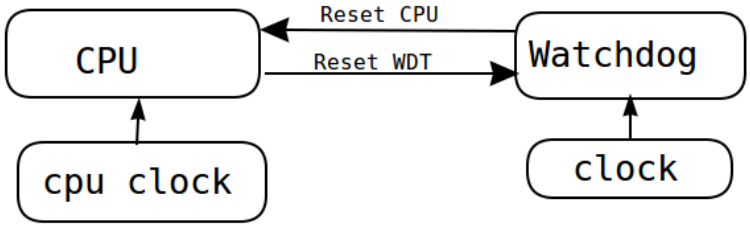
- Perangkat lengkap Arduino UNO
- PC/Laptop yang terinstall software Arduino IDE

5.3 Dasar Teori

Dalam sebuah situasi tertentu, terkadang mikrokontroller dapat *stuck* di sebuah kondisi, misalnya *hang*, atau berhenti ketika mengeksekusi *task*. Biasanya, *troubleshooting* yang dilakukan adalah dengan menekan tombol *reset*, atau melepas dan memasang kembali catu daya. Namun tentu saja hal ini bukan hal yang praktis, terutama untuk sistem yang harusnya bekerja secara otomatis dan jauh dari jangkauan. Akan lebih baik kalau *device* itu sendiri yang mengetahui bahwa dia mengalami *stuck* dan kemudian mereset dirinya sendiri.

Watchdog Timer

Watchdog timer (WDT) adalah *timer* dari *hardware* yang secara otomatis men-*generate reset* ketika program utama gagal menjalankan "servis *reset*" kepada *watchdog* secara periodik. Fitur ini sering digunakan untuk otomatis mereset sistem *embedded* yang mengalami *hang* karena terjadi *fault* software atau *hardware*. *Watchdog timer* memiliki clock dari Osilator On-chip terpisah yang berjalan pada 1 MHz. Ini adalah nilai tipikal pada VCC = 5V. Beberapa sistem mungkin juga menyebutnya sebagai *computer operating properly* (COP) timer. Semua Arduino board memiliki *watchdog timer hardware*.



Program utama pada arduino biasanya memiliki *loop* yang terus-menerus melakukan berbagai fungsi atau *task*. *Watchdog timer* dimuat dengan nilai awal yang lebih besar dari *worst case time delay* melalui *looping* program utama. Setiap kali melewati *loop* utama, kode harus malakukan reset *watchdog timer* (terkadang

disebut “trigger”, "*kick*" atau "*feed the dog*") secara periodik dengan interval tertentu. Analoginya, kita harus memberi makan “*the dog*” secara periodik sesuai interval. Jika “pemberian makan” tidak dilakukan atau terlambat, maka “*dog*” tersebut akan menggigit (dalam hal ini mereset prosesor). Jika terjadi fault dan program utama tidak dapat melakukan reset *watchdog* sebelum penghitungan mundur ke nol, maka akan terjadi interupsi untuk mereset prosesor. Dengan cara ini, watchdog timer dapat mendeteksi *fault* pada program Arduino dan menerapkan tindakan korektif dengan berupa reset. Setelah reset, sebuah register dapat dibaca untuk menentukan apakah *watchdog timer* menghasilkan reset ataukah terjadi reset normal. Di Arduino register ini disebut *Watchdog Reset Flag Register* (WDRF).

Untuk beberapa perangkat yang lebih baru (ATmega88 dan juga AVR yang memiliki fitur menghasilkan *interrupt*), *watchdog timer* tetap aktif bahkan setelah sistem direset (kecuali ketika kondisi *initial power-on*), dengan nilai *prescaler* tercepat (sekitar 15 ms). Oleh karena itu diperlukan untuk mematikan *watchdog* lebih awal saat program dimulai.

Langkah menerapkan *watchdog timer*

Step 1: Library yang dibutuhkan

```
#include <avr/wdt.h>
```

Step 2: Mengaktifkan/menon-aktifkan *Watchdog timer* dengan periode reset tertentu

Mengaktifkan Watchdog timer

```
wdt_enable(WDT Reset Timer);
```

contoh:

```
wdt_enable(WDTO_4S);
```

watchdog timer dapat diaktifkan dengan pengaturan waktu yang berbeda. Pengaturan waktu adalah waktu antara *watchdog* melakukan reset dan “pemberian makan/*feed*”. Waktu interval harus lebih besar dari waktu yang dibutuhkan untuk loop program yang dibutuhkan untuk kembali lagi. Pengaturan waktu maksimum 8 Detik dan minimal 15mSec.

	Constant name	*Supported on*
15 ms	WDTO_15MS	ATMega 8, 168, 328, 1280, 2560
30 ms	WDTO_30MS	ATMega 8, 168, 328, 1280, 2560
60 ms	WDTO_60MS	ATMega 8, 168, 328, 1280, 2560
120 ms	WDTO_120MS	ATMega 8, 168, 328, 1280, 2560
250 ms	WDTO_250MS	ATMega 8, 168, 328, 1280, 2560

500 ms	WDTO_500MS	ATMega 8, 168, 328, 1280, 2560
1 s	WDTO_1S	ATMega 8, 168, 328, 1280, 2560
2 s	WDTO_2S	ATMega 8, 168, 328, 1280, 2560
4 s	WDTO_4S	ATMega 168, 328, 1280, 2560
8 s	WDTO_8S	ATMega 168, 328, 1280, 2560

Menon-aktifkan *Watchdog timer*

```
wdt_disable();
```

Step 3: Mereset/"*Feed the dog*" *Watchdog timer*

Fungsi ini harus dipanggil di awal fungsi loop(). Ini dilakukan untuk mereset hitungan *watchdog timer*. Jika loop program macet/*hang* dan tidak dapat melakukan "*reset/feed the dog*" maka watchdog timer akan mereset Arduino dan mencegah terjadinya *hang* karena *noise* atau *fault*.

```
wdt_reset();
```

5.4 Prosedur Percobaan

A. Prosedur

1. Siapkan perangkat Arduino UNO dan hubungkan dengan PC/laptop yang telah terinstall Arduino IDE
2. Ketik source code berikut pada Arduino IDE

```
1  #include <avr/wdt.h>
2  void setup()
3  {
4  MCUSR &= ~(1<<WDRF); // Clear WDRF in MCUSR
5  wdt_disable();
6  Serial.begin(9600);
7  Serial.println();
8  Serial.println("==BOOTING/REBOOTING==");
9
10 // make a delay before enable WDT
11 // this delay help to complete all initial tasks
12 delay(2000);
13
14 wdt_enable(WDTO_4S);//enabling using 4 sec threshold timer
15 Serial.println("success enabling wdt");
16 }
17
```

```
18  void loop()
19  {
20  Serial.println("LOOP started ! ");
21  for(int i=0; i<=5; i++)
22      {
23         Serial.print("Loop : ");
24         Serial.print(i);
25         delay(1000);//delay 1 sec
26         wdt_reset();//then feed the dog
27         Serial.print(" The dog has been fed");
28         Serial.println();
29      }
30
31  //infinity loop to simulate hang then reset MCU
32  while(1){
33   Serial.println("== simulation of freeze/not feeding the dog ==");
34        }
35  }
```

3. Compile kode di atas, dan upload ke board Arduino IDE
4. Pada Arduino IDE, buka Serial Monitor dan sesuaikan dengan *baudrate* yang telah dikonfigurasi sesuai koding
5. Amati jalannya eksekusi program melalui Serial Monitor

B. Hasil dan Analisis

1. Amati output pada Serial monitor mulai sejak koding dijalankan
2. Perhatikan jalannya eksekusi program ketika *watchdog* ditrigger/direset tiap 1 detik (di dalam *for loop*). Jelaskan apa yang terjadi!
3. Perhatikan jalannya eksekusi program ketika program memasuki void *loop* dimana *watchdog* tidak ditrigger/direset melebihi *threshold watchdognya* (4 detik). Jelaskan apa yang terjadi!
4. Ganti *threshold watchdog* menjadi 1 detik, jalankan simulasi, dan jelaskan apa yang terjadi!
5. Ganti *threshold watchdog* menjadi 8 detik, jalankan simulasi, dan jelaskan apa yang terjadi!
6. Jelaskan bagaimana tiap baris *source code* bisa menghasilkan output sesuai poin nomor 2 dan 3!

5.5 Tugas

Buat koding yang menerapkan *watchdog* yang berfungsi untuk melakukan reset jika *watchdog* tidak ditrigger/direset melebihi durasi tertentu (tentukan periode *watchdognya*) dan eksekusi jalannya koding pada perangkat Arduino UNO. Kriteria: menggunakan minimal 1 input digital, 1 input analog dan memiliki output non serial monitor.

5.6 Kesimpulan

Berdasarkan percobaan yang telah dilakukan, maka tuliskan kesimpulan percobaan di tempat yang telah disediakan di bawah ini: