BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Peserta magang di PT Swarna Prima Manggala memiliki kedudukan sebagai asisten teknisi dalam divisi *project*. Dalam posisi ini, peserta magang bertugas membantu teknisi senior di tim dalam merangkai panel relay. Bantuan dapat berupa menyiapkan kabel untuk panel, hingga pemasangan dan *wiring* komponen panel seperti MCB dan *terminal*.

Peserta magang juga mendapat posisi sebagai teknisi untuk *project andon*. Dalam posisi ini, peserta magang bertugas dalam perangkaian hingga *source code* sistem. Supervisor magang memberikan spesifikasi, serta komponen-komponen yang akan dibutuhkan sesuai dengan perancangan yang ditetapkan oleh peserta magang.

3.2 Tugas dan Uraian Kerja

Selama periode magang di PT Swarna Prima Manggala, peserta magang terlibat dalam pengerjaan berbagai *project* panel *relay* sebagai asisten, serta sebuah *project* mandiri untuk tugas akhir mahasiswa. Uraian pekerjaan selama periode magang dapat dilihat pada tabel 3.1 berikut

Tabel 3.1 Tabel Kegiatan Peserta Magang

No	Bulan	Kegiatan
1	Febuari	• *Maintenance* panel di PT Art Indonesia • Merakit panel *relay* sistem kendali pH untuk PT Sumiden • Mengambil program PLC di PT KDS Indonesia • Merakit panel *waste water treatment* untuk PT NCE Indonesia • Membuat skematik, dan proposal komponen untuk *demo kit andon* • Membuat *source code* untuk Arduino
2	Maret	• Merakit panel *waste water treatment* untuk PT NCE Indonesia • Kunjungan ke PT Arwana Ceramics untuk *maintenance* PLC

		• Uji coba fungsi-fungsi *demo kit* • Membuat program GUI untuk *demo kit* • Solder rangkaian *demo kit*
3	April	• Membuat program GUI untuk *demo kit* • Merancang, dan membuat *exterior* untuk tombol dan Arduino • Menyiapkan komponen, dan merakit panel untuk PT Rohto
4	May	• Merakit panel untuk PT Rohto • Perbaikan rangkaian kabel pada *demo kit andon* • Persiapan untuk perakitan panel di PT Rohto • Mempelajari program PLC yang biasa digunakan PT Swarna Prima Manggala • Perakitan *frame* untuk *demo kit*
5	Juni	• Penambahan fitur ke *demo kit* • Finalisasi *demo kit*

3.3 Uraian Pelaksanaan Kerja

3.3.1 Proses Pelaksanaan

a) Asisten Teknisi Proyek Panel *Relay*

Dalam perangkaian panel *relay*, peserta magang berperan sebagai asisten teknisi. Jalur pengerjaan panel *relay* dimulai dari memasang *cable dump*, dan plat omega ke *baseplate* panel. *Cable dump* berfungsi sebagai penampung kabel dari satu komponen ke komponen lainnya agar kabel tidak berantakan, dan plat omega berfungsi sebagai tempat untuk memasang komponen-komponen panel seperti MCB, *relay*, *coil*, dan *terminal*.

Ketika seluruh komponen terpasang, komponen-komponen kemudian disambungkan menggunakan kabel yang sudah dipasang dengan skun. Sambungan kabel dibuat dengan mengikuti *ladder diagram* yang ada.

Setelah perangkaian komponen di *baseplate* sudah selesai, *baseplate* dipasangkan kembali ke panel *relay*, dan pekerjaan dilanjutkan ke *sideplate*. Prosedur kerja sama dengan perakitan *baseplate*, namun isi komponen lebih sedikit, dan majoritas *sideplate* diisi dengan *cable dump* untuk kabel dari *coil* ke tombol. Selain

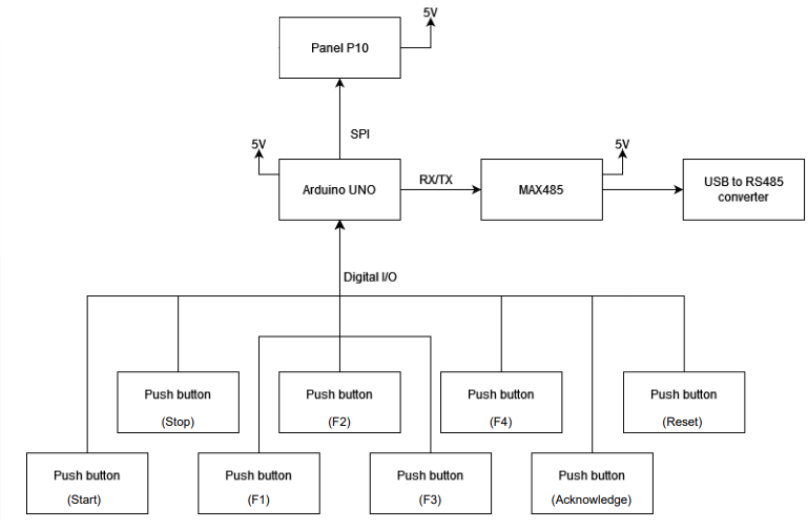
sideplate, tombol mulai dipasang ke panel beserta dengan *nameplate*. Ketika keduanya selesai, *sideplate* dipasang kembali ke panel *relay*, lalu tombol disambungkan ke komponen yang ada di *baseplate* melalui *cable dump* yang ada di *sideplate*. Proyek diakhiri dengan uji coba rangkaian, dan jika sudah bekerja sesuai dengan *ladder diagram*, panel relay siap dikirimkan.

b) Demo kit Andon System

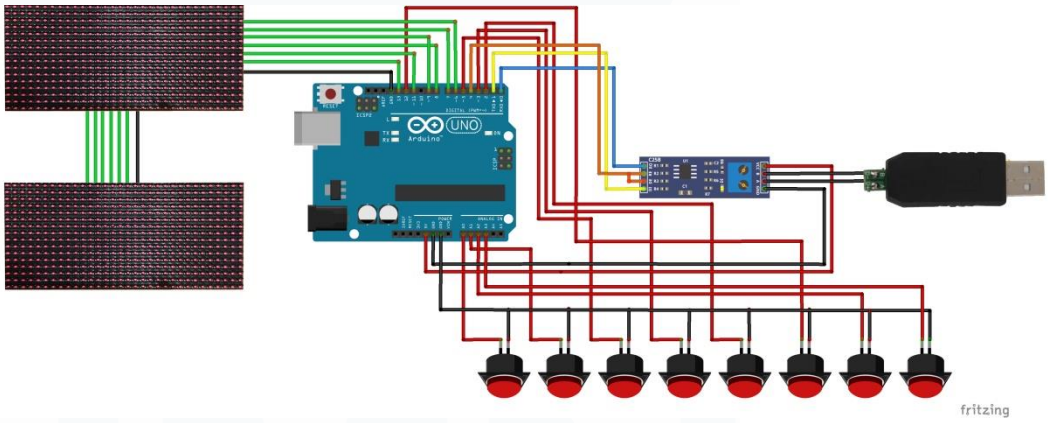
Demo kit Andon System adalah *project* yang ditanggung jawabkan ke peserta magang. Spesifikasi sistem diberikan oleh supervisor, dan peserta magang harus merancang sistem berdasarkan spesifikasi tersebut. *Project andon* harus memenuhi kriteria berikut:

1. Memiliki delapan input. START, STOP, RESET, tiga DOWNTIME, dan ACKNOWLEDGE.
2. Output ditampilkan ke dua Panel P10. Satu panel dapat menampilkan dua informasi.
3. Menampilkan empat informasi dengan nama "TARGET", "ACTUAL", "BALANCE", dan "DOWNTIME".
4. Menggunakan protokol komunikasi Modbus RTU, dengan PC sebagai *master,* dan mikrokontroler sebagai *slave.*

Dengan kriteria yang diberikan, dirancang sistem dengan *design* sebagai berikut:



Gambar 3.1 Diagram blok sistem



Gambar 3.2 *Wiring diagram* sistem

Untuk pilihan komponen yang akan digunakan, disertakan alasan sebagai berikut:

a) Arduino UNO

Gambar 3.3 Arduino Uno

Arduino UNO akan digunakan sebagai sistem kendali sistem, dan perangkat *slave* Modbus. Arduino UNO memiliki pin komunikasi yang dibutuhkan sistem (SPI, dan TTL), serta jumlah pin I/O yang mencukupi. Arduino UNO juga dipilih karena kompatibilitas *board* dengan *library* yang perlu digunakan untuk mengendalikan panel p10.

b) MAX485

Gambar 3.4 Modul MAX485

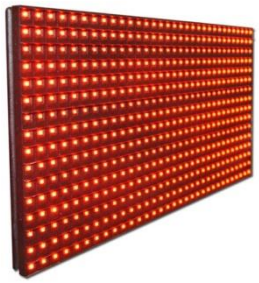
MAX485 terhubung ke pin RX/TX, 5V, dan GND Arduino UNO. MAX485 berfungsi sebagai *transceiver* untuk komunikasi RS485. MAX485 menkonversi pin TTL pada Arduino UNO menjadi RS485.

c) *USB to RS485 Converter*

Gambar 3.5 *USB to RS485 Converter*

Konverter berfungsi untuk menyambungkan MAX485 ke komputer melalui USB *port*. Komputer yang terhubung akan berperan sebagai *master* Modbus.

d) Panel P10



Gambar 3.6 Panel P10

Panel P10 berfungsi untuk menampilkan informasi dari Arduino UNO. Panel terhubung ke Arduino melalui pin SPI, dan panel memiliki *output* yang dapat disambungkan ke panel P10 lainnya, sehingga pin SPI hanya perlu dihubungkan ke satu panel. Satu panel dapat menampilkan hingga dua baris tulisan.

e) Push button

Gambar 3.7 *Push button*

Push button dihubungkan ke pin I/O, dan GND di Arduino UNO. Akan digunakan tujuh *push button*, dan setiap *button* akan berfungsi sebagai *input* untuk Arduino UNO.

Keempat informasi yang ditampilkan berhubungan dengan kondisi jalur produksi. TARGET menampilkan berapa banyak produk yang harusnya sudah dibuat pada saat itu. ACTUAL menampilkan berapa banyak produk yang sudah dibuat aslinya. BALANCE menampilkan selisih antara TARGET dan ACTUAL.

DOWNTIME menampilkan berapa lama jalur produksi terhenti akibat sebuah kendala.

Perhitungan TARGET didapatkan dari *input* melalui *master* Modbus. Perangkat *slave* menyediakan tiga *holding register*(40001, 40002, 40003) yang dapat menampung angka 16-bit *integer*. Menggunakan fungsi Modbus kode 16, *master* Modbus dapat menulis ke beberapa *holding register* secara bersamaan. Ketiga *holding register* berfungsi sebagai berikut:

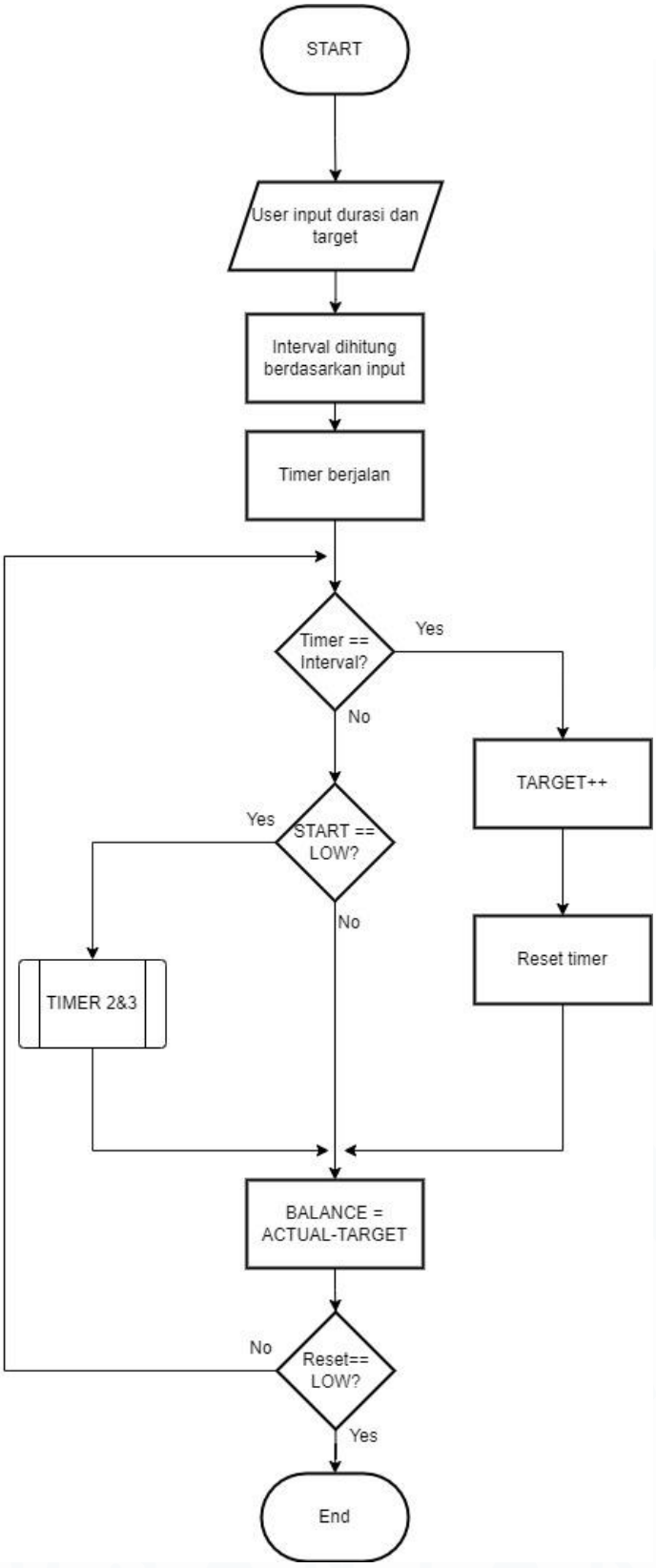
a) 40001: Durasi produksi. *Input* dalam satuan jam.
b) 40002: Target produksi.
c) 40003: Status untuk menyalakan *demokit*.

Target dan durasi akan dimasukan ke persamaan 3.1 untuk mendapatkan sebuah interval dalam *millisecond*. Interval berfungsi sebagai waktu yang dibutuhkan untuk meningkatkan angka pada TARGET.

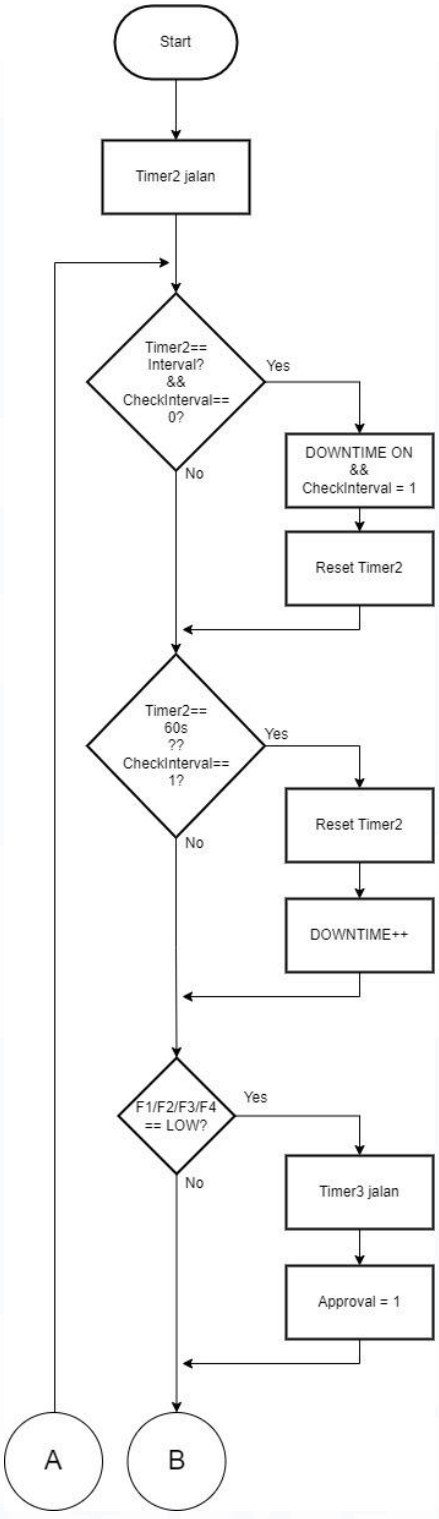
$$Interval = \frac{T \times 3600000}{x} \qquad (3.1)$$

Dimana T merupakan durasi demokit dalam satuan jam, sesuai dengan isi *holding register* 40002, dan x merupakan target yang diinginkan, sesuai dengan isi *holding register* 40001.

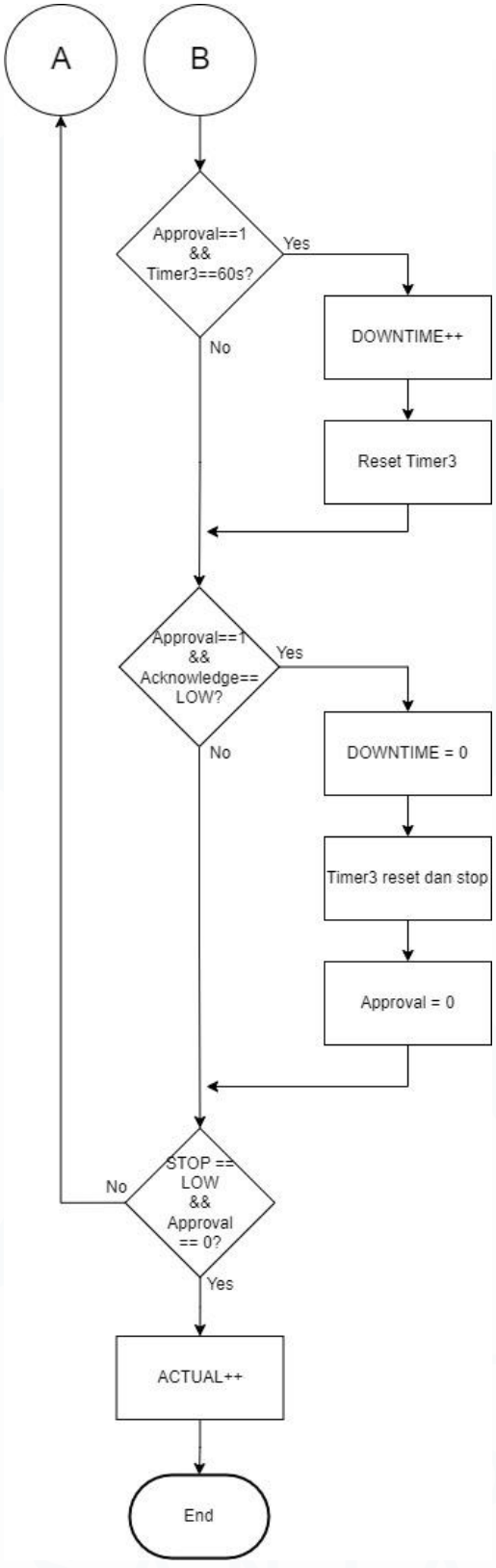
Dengan tambahan ketujuh tombol, cara kerja sistem dapat digambarkan dengan *flowchart* pada gambar 3.3



Gambar 3.8 *Flowchart* kerja sistem



Gambar 3.9 *Sub-flowchart* kerja sistem



Gambar 3.10 *Sub-flowchart* kerja sistem

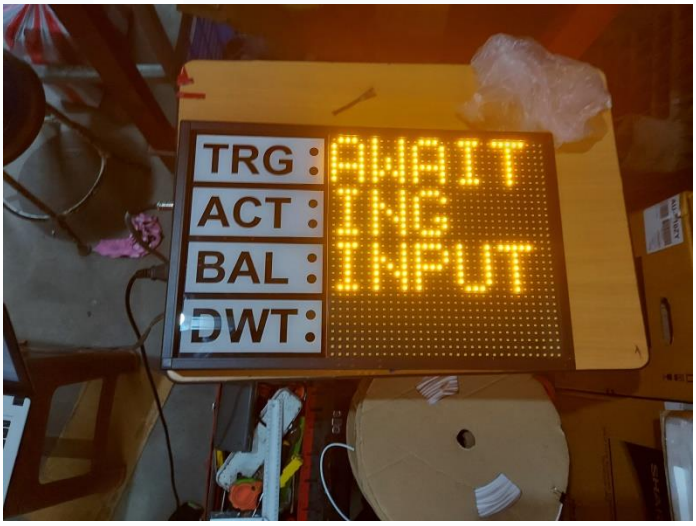
Gambar 3.11 *Layout* tombol

Tombol START dan STOP berfungsi untuk menambahkan angka pada ACTUAL. Untuk angka dapat bertambah, tombol harus ditekan dengan urutan START, lalu STOP. Ketika tombol START ditekan, sebuah *timer* baru akan berjalan. Jika *timer* sudah melebihi interval, dan tombol STOP belum ditekan, DOWNTIME akan mulai berjalan. DOWNTIME akan mati jika tombol STOP ditekan. Tujuan digunakan dua tombol adalah untuk mensimulasikan suatu barang masuk ke pos dalam sebuah jalur produksi, diproses, lalu keluar ke pos berikutnya.

Tombol F1-F4 berfungsi untuk menjalankan DOWNTIME. Fungsi hanya akan berjalan jika tombol START sebelumnya ditekan, dan tombol STOP belum ditekan. Menekan salah satu tombol DOWNTIME akan menjalankan sebuah *timer* baru dalam sistem. Setiap satu menit *timer* berjalan, angka pada DOWNTIME akan bertambah. *Timer* akan berjalan terus sampai tombol ACKNOWLEDGE ditekan. Tombol ACKNOWLEDGE juga akan mengulang, dan mematikan angka pada DOWNTIME. Fungsi ini bermaksud untuk mensimulasikan keadaan pada jalur produksi jika

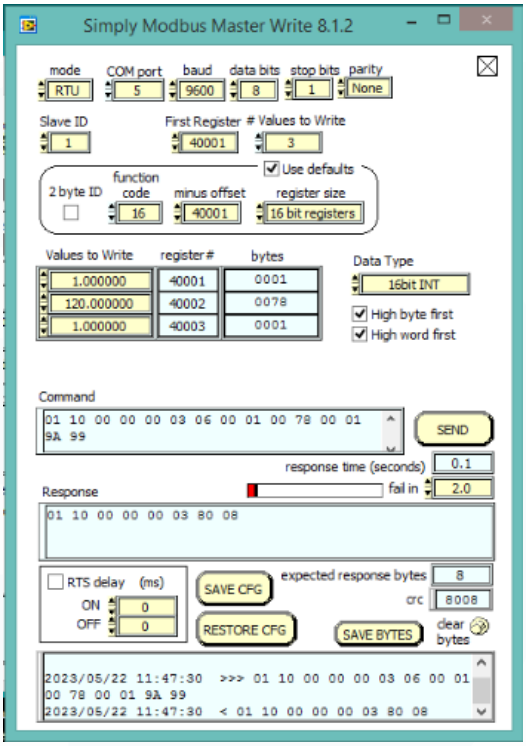
ada kendala. F1-F4 berfungsi sebagai cara untuk menginformasikan empat jenis kendala yang berbeda, dan ACKNOWLEDGE berfungsi sebagai tombol yang ditekan oleh supervisor untuk menandakan bahwa kendala telah ditangani.

Tombol terakhir adalah RESET. RESET akan mematikan panel, dan mengulang sistem. Untuk menjalankan sistem, *user* harus memasukan *input* melalui *master* Modbus lagi.



Gambar 3.12 Pesan AWAITING INPUT

Ketika sistem dinyalakan, panel akan menampilkan pesan "*AWAITING INPUT*". Pesan ini akan ditampilkan jika belum ada input dari *master* kedalam *holding register* milik *slave*. *Master* Modbus akan dijalankan melalui sebuah komputer, dan untuk menjalankannya, digunakan aplikasi *Simply Modbus Master*.

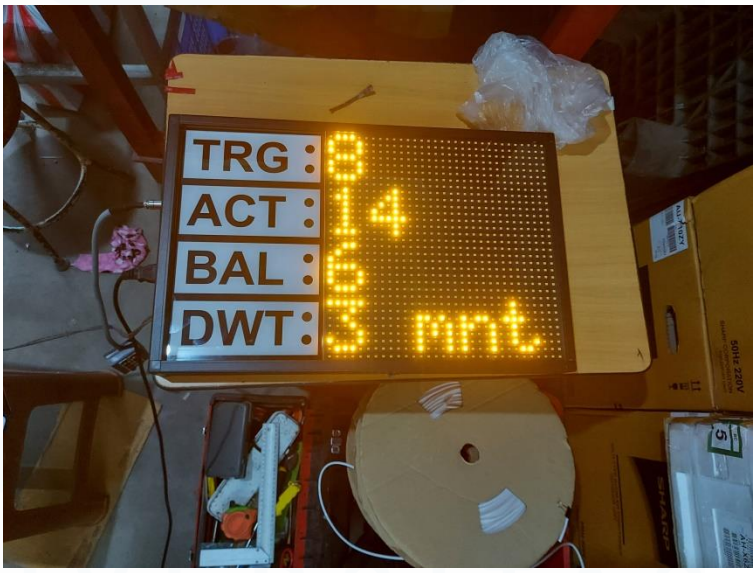


Gambar 3.13 UI Aplikasi *Simply Modbus Master*

Versi *demo* aplikasi ini dapat digunakan untuk menulis ke *slave register* sebanyak lima kali sebelum aplikasi harus dibuka ulang. Pengaturan pada menu aplikasi harus sesuai dengan pengaturan yang ditetapkan pada *source code*. Hal ini mencakup *baud rate*, *slave* ID, *data bits*, *stop bits*, dan *parity*. *COM port* juga harus dipastikan sesuai dengan *port* yang digunakan oleh *slave*. Setelah pengaturan sudah sesuai, ubah opsi "*# Values to Write*" menjadi tiga. Dengan ini, program akan otomasi menjalankan fungsi Modbus kode 16, dan menulis ke tiga *holding register* yang akan dibaca oleh *slave*. Setelah itu, bisa ditulis nilai integer ke *register* 40001, 40002, dan 40003, kemudian ditekan tombol *SEND* untuk dibaca oleh *slave*.

Gambar 3.14 Pesan ANDON BEGIN

Setelah tombol *SEND* ditekan, panel akan menampilkan pesan "*ANDON BEGIN*" untuk lima detik sebelum panel kembali mati. Setelah itu, tombol pada sistem bisa mulai ditekan untuk menjalankan fungsinya, dan panel akan menampilkan info yang sesuai dengan input dari *master*, dan tombol.



Gambar 3.15 Informasi yang ditampilkan panel

Informasi yang ditampilkan dari atas ke bawah adalah "TARGET", "ACTUAL", "BALANCE", dan "DOWNTIME". Pada gambar 3.7, dapat didapatkan bahwa jalur produksi sudah harus menciptakan delapan produk(TARGET), dengan produksi aslinya

sudah menciptakan empat belas produk (ACTUAL), dan selisih antara keduanya adalah enam (BALANCE). Dari gambar juga didapatkan bahwa ada kendala yang sudah berjalan selama tiga menit(DOWNTIME).

Secara keseluruhan, sistem menggunakan tiga *timer* yang berbeda. Arduino UNO aslinya tidak memiliki fungsi *timer*, namun Arduino UNO menghitung durasi program sudah berjalan. Durasi program berjalan dapat diperoleh menggunakan *syntax* millis(). Artinya, dengan menyimpan durasi program ketika suatu tombol ditekan, lalu dibandingkan terus dengan durasi program seterusnya, dapat direplikasikan fungsi yang serupa dengan sebuah *timer*.

```
currentMillis = millis();
if(currentMillis - compensationtime - previousMillis >= interval){//jika waktu melebihi/sama dengan interval, target increment
  targetCounter++;
  previousMillis = currentMillis;
  compensationtime = 0;

  writeToDmd(targetCounter,1);
}
```

Gambar 3.16 Bagian kode yang menjalankan fungsi *timer* untuk TARGET

Pada gambar 3.16 adalah salah satu bagian dari *source code* yang menjalankan fungsi *timer* untuk TARGET. Perbandingan untuk *timer* menggunakan empat variabel, yaitu “currentMillis”, “compensationtime”, “previousmillis”, dan “interval”. Ketika *user* memberi *input* ke sistem, akan disimpan durasi program ketika *input* diterima ke “compensationtime”. Variabel “currentMillis” akan diperbarui terus dengan durasi program, sehingga selisihnya antara “compensationtime” akan selalu bertambah. Ketika selisih antara “currentMillis” dan “compensationtime” telah melebihi “interval”, angka pada panel TARGET akan bertambah. Durasi program yang disimpan pada “compensationtime” akan dijadikan nol, dan durasi program ketika TARGET bertambah terakhir kalinya, akan dimasukan ke “previousmillis”. Untuk seterusnya, “currentMillis” akan dibandingkan dengan “previousmillis”.

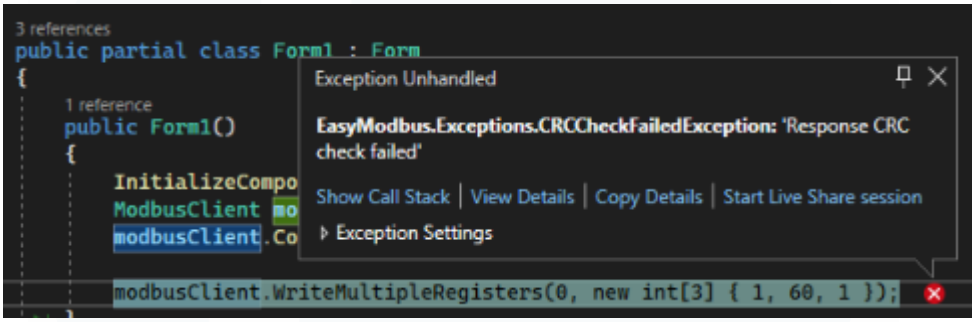
5	Data set ke-1				
6	KONFIGURASI AWAL				
7	Durasi:	1 jam			
8	Target produksi:	120			
9	Interval:	30 s			
10	LOG BEGIN				
11	TARGET:	4			
12	ACTUAL:	4			
13	BALANCE:	0			
14	Downtime ACTUAL:	3 s			
15	Downtime F1:	14 s	2		
16	Downtime F2:	19 s	2		
17	Downtime F3:	68 s	1		
18	Downtime F4:	0 s	0		
19	┼└?				
20	Data set ke-2				
21	KONFIGURASI AWAL				
22	Durasi:	4 jam			
23	Target produksi:	430			
24	Interval:	33 s			
25	LOG BEGIN				
26	TARGET:	2			
27	ACTUAL:	5			
28	BALANCE:	3			
29	Downtime ACTUAL:	0 s			
30	Downtime F1:	3 s	1		
31	Downtime F2:	0 s	0		
32	Downtime F3:	2 s	1		
33	Downtime F4:	29 s	1		

Gambar 3.17 Hasil *Data Logging* pada Microsoft Excel

Data logging juga dapat dilakukan ketika tombol RESET ditekan, jika komputer terhubung ke USB *port* Arduino. Dengan menggunakan fungsi *Record Data* pada *add-ins Data Streamer* untuk Microsoft Excel, informasi yang ditampilkan selama *demo kit* berjalan, beserta konfigurasi awal yang diberikan *user* dapat disimpan ke dalam sebuah *spreadsheet*. Informasi mencakup nilai "TARGET", "ACTUAL", dan "BALANCE" sebelum tombol RESET ditekan, total detik "DOWNTIME" terjadi untuk setiap kendala yang ada, beserta berapa kali tombol F1, F2, F3, dan F4 ditekan. Jika *Record Data* tidak dihentikan, operasi *demo kit* berikutnya juga dapat disimpan.

3.4 Kendala yang Ditemukan

Komunikasi antara Arduino dengan PC melalui komunikasi Modbus memiliki kendala dimana sistem gagal melakukan *cyclic redundancy check* (CRC). Akibatnya, sistem tidak dapat menggunakan program GUI yang dibuat menggunakan Visual Studio.



Gambar 3.17 Pesan *exception* ketika menjalankan kode GUI di Visual Studio

3.5 Solusi atas Kendala yang Ditemukan

Karena sistem tidak bisa menggunakan program GUI yang dibuat, komunikasi *master* ke *slave* dilakukan menggunakan program simulasi Modbus bernama *Simply Modbus Master*. Program ini dapat terus berjalan meskipun ada *error* ketika melakukan CRC. Meskipun komunikasi dapat dilakukan, ada beberapa kekurangan dengan menggunakan aplikasi tersebut. Pertama, aplikasi yang digunakan berupa versi demo. Pada versi demo, program hanya dapat melakukan fungsi WRITE sebanyak lima kali sebelum program harus direset. Kedua, karena aplikasi digunakan untuk simulasi Modbus, penulisan ke *holding register* dilakukan secara langsung ke alamat *register*. Artinya pengguna aplikasi harus mengetahui berapa banyak *holding register* yang digunakan sistem, dan fungsi tiap alamat *register* untuk apa.