**LAPORAN UJIAN TENGAH SEMESTER (UTS) SISTEM EMBEDED**

***LAP COUNTER***

****

**Dosen Pengampu:**

Akhmad Hendriawan, S.T., M.T

**Mahasiswa:**

Zikri Nasution

1110151033

**3 D4 T.ELEKTRONIKA B**

**PROGRAM STUDI**

**D4TEKNIK ELEKTRONIKA**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2017**

**BAB I**

**PENDAHULUAN**

1. **Tujuan**
   1. Mahasiswa mampu mengakses PIN Input dari sensor dan Output berupa Lampu LED dengan module STM32F103.
   2. Mahasiswa mampu mengakses data serial dari STM32F103 sebagai komunikasi mengirimkan data.
   3. Mahasiswa mampu menampilkan data waktu dan jumlah lap dari input sensor.
2. **Rumusan Masalah**

Sebuah embedded multi stopwatch digunakan untuk merekam perlombaan modil dengan jumlah lap sebanyak 4 lap dan diikuti oleh 3 kendaraan. Pembangkitan timer dilakukan oleh embedded kontroller sementara tampilan jarak yang ditempuh untuk setiap lap ditampilkan di LCD. Agar tidak terjadi bouncing pada saat penekanan tombol, maka ditambahkan mekanisme anti vouncing menggunakan *software*. Sistem mempunyai blok diagram berikut

**Spesifikasi**

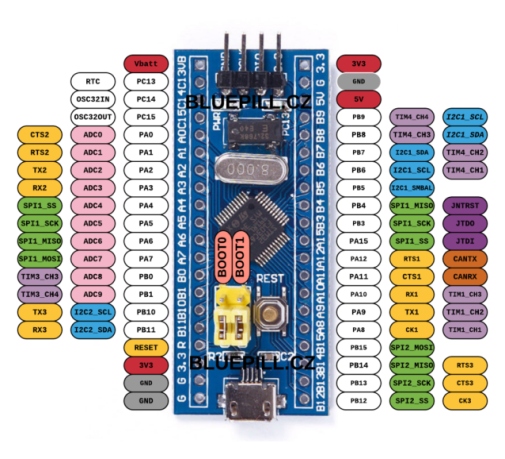
* Tombol start/stop digunakan untuk memulai atau menghentikan timer
* Tombol reset digunakan untuk mereset timer secara manual
* Sensor 1 sampai sensor 3 adalah sensor led dan photodiode yang digunakan untuk mencatat waktu yang ditempuh pada setiap lap (3 sensor untuk 3 kendaraan) dengan tambahan mekanisme anti bouncing menggunakan software.
* Apabila sensor mengalami kerusakan maka sistem akan memberitahu user tentang nomer sensor yang mengalami kegagalab (kode pesan setiap sensor berbeda) dengan menggunakan mekanisme timeout. Sistem juga mengirimkan kode pesan error tersebut ke PC melalui serialntuk keperluan debugging.

**BAB II**

**LANDASAN PUSTAKA**

1. **STM32F103**

STM32F103xx medium-density menggabungkan kinerja high performance ARM Cortex-M3 32-bit RISC yang dapat beroperasi pada frekuensi 72 MHz, tertanam memory highspeed (Flash memori hingga 128 Kbytes dan SRAM hingga 20 Kbytes), dan berbagai pilihan ditingkatkan I / Os dan periferal yang terhubung ke dua bus APB. Semua perangkat menawarkan dua ADC 12-bit, tiga tujuan umum 16-bit timer plus satu PWM timer, serta standar dan maju antarmuka komunikasi: hingga dua I2Cs dan SPI, tiga USARTs, USB dan CAN.



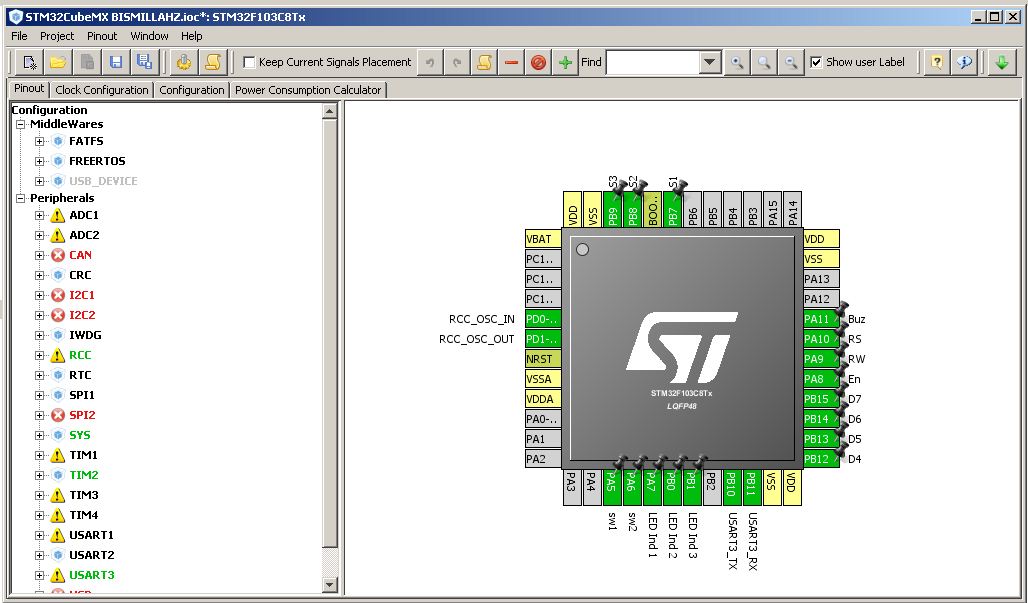
Gambar 1. STM32F103

Berikut adalah fitur yang terdapat pada STM32F103 :

* ARM 32-bit Cortex ™ M3 CPU Inti
  + 72 MHz frekuensi maksimum, 1,25 DMIPS / MHz (Dhrystone 2.1) kinerja pada 0 menunggu akses memori negara
  + Single-siklus perkalian dan pembagian hardware
* Memory
  + 64 atau 128 Kbytes memori Flash
  + 20 Kbytes SRAM
* Clock, reset and supply management
  + 2,0-3,6 V pasokan aplikasi dan I / Os
  + POR, PDR, dan detektor tegangan diprogram (PVD)
  + 4-ke-16 MHz osilator kristal
  + Internal 8 MHz pabrik-dipangkas RC
  + Internal 40 kHz RC
  + PLL untuk jam CPU
  + 32 kHz osilator untuk RTC dengan kalibrasi
* Daya rendah
  + Mode standby, sleep dan stop
  + VBAT supply untuk RTS dan *backup register*
* 7 timers
  + 3 buah 16-bits timer
  + 16 bit, motor control dengan PWM timer
  + 2 *watchdog timer*
  + Systic timer 24-bit *downcounter*

1. **STM32 Cube MX**

STM32CubeTM dari STMicroelectronics adalah alat desain bebas dan perangkat lunak untuk keluarga mikrokontroler STM32. Platform pengembangan STM32Cube baru terdiri dari STM32CubeMX configurator grafis dan inisialisasi C-kode generator yang memberikan panduan langkah-demi-langkah untuk pengguna dan satu set kaya komponen tertanam-software yang menyimpan mengintegrasikan perangkat lunak dari berbagai sumber. Perangkat lunak ini termasuk Hardware Abstraction Layer baru (HAL) yang menyederhanakan port dari satu perangkat STM32 yang lain. Dengan mengumpulkan semua komponen perangkat lunak generik yang dibutuhkan untuk mengembangkan aplikasi pada mikrokontroler STM32 dalam satu paket tunggal, platform ini menghilangkan tugas yang kompleks menilai ketergantungan antara komponen perangkat lunak individu. STM32Cube memberikan ratusan contoh penggunaan dan mekanisme update untuk akses cepat dan efisien untuk versi terbaru dari perangkat lunak.

****

Gambar 2. STM32Cube MX

The STM32CubeMX alat konfigurasi grafis menyederhanakan dan mengotomatisasi konfigurasi dan generasi STM32 inisialisasi C kode siap untuk digunakan dengan beberapa lingkungan pengembangan. STM32CubeMX juga tersedia sebagai Eclipse plug-in, yang memungkinkan digunakan dalam lingkungan pengembangan berbasis Eclipse. Alat ini menyediakan filter yang mudah digunakan untuk memilih perangkat STM32 yang diinginkan dan grafis untuk menyederhanakan konfigurasi: pin-out wizard assists pin assignment untuk menghindari konflik, menanamkan solver untuk kendala; wizard clock tree memberikan clock dan melakukan validasi dinamis; peripheral-dan-middleware konfigurasi Wizard bantu untuk menghindari pengaturan tidak dapat digunakan; dan wizard konsumsi daya memeriksa bahwa aplikasi memenuhi daya yang tersedia.

1. **SW4STM32**

SW4STM32 adalah singkatan dari System Workbench for STM32, seperti yang diperlihatkan pada Gambar 1. Pada dasarnya SW4STM32 adalah sebuah IDE (Integrated Development Environment) yang dibangun di atas IDE Eclipse. Ini hampir mirip dengan salah satu implementasi PlatormIO yang dibangun dengan menggunakan text editor Atom. Dari menu Help dapat terlihat versi Eclipse yang dipergunakan, seperti pada Gambar 2. Dengan catatan tentu setiap pembaruan versi SW2STM32 akan ada kemungkinan bahwa versi Eclipse IDE juga akan berubah.

Dari situs STMicroelectronics, dapat ditemukan keterangan singkat yang baik mengenai SW4STM32. Intinya program SW4STM32 tidak dibuat oleh STMicroelectronics, melainkan oleh pihak ketiga yaitu AC6. Sedangkan software dan layanan yang berikaitan dengannya di www.openstm32.org. Untuk mendapatkan akses ke layanan download maupun panduan instalasi, pengguna perlu terlebih dahulu melakukan registrasi di situs openstm32.



Gambar 3. STM32Cube MX

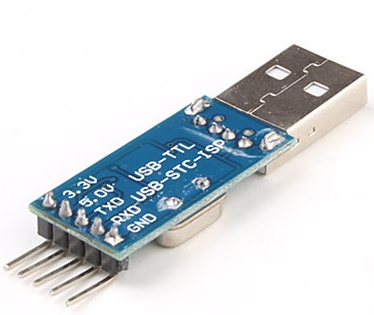
1. **USB to TTL**

USB-TTL merupakan USB *to* TTL *level serial converter board,* yang menyediakan jalur yang terhubung dari USB port pada komputer untuk tersambung pada 5Vatau 3V TTL Level *serial interface devices*. USB to TTL didesain dengan pin I/O yang dapat dikonfigurasi dengan mudah untuk beroperasi pada level tegangan 5V atau 3,3V disesuaikan dengan mikrokontroller yang digunakan.

Selain dari sinyal RxD, TxD, RTS#, CTS#, USB -TTL juga menyediakan tegangan ouput +5VDC atau +3.3VDC dari pengaturan 5V atau 3.3V TTL level serial interface yang sesuai. USB-TTL sangat mudah digunakan. Ketika ber*interface* dengan komputer dan sistem embedded, USB-TTL *converter board* layaknya virtual COM Port.

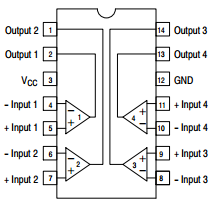
Fitur dari USB to TTL antara lain

1. *High speed* TTL level serial interface
2. Menyediakan *level serial interface* 3.3V atau 5V untuk hampir seluruh mikrokontroller
3. Menyediakan tegangan output +5VDC atau +3.3 VDC untuk perangkat eksternal seperti mikrokontroller, Wi-Fi, *bluetooth*, Xbee, dan sebagainya.
4. Sinyal RxD, TxD, RTS#, CTS#
5. Led sebagai indikator power



Gambar 4. USB to TTL

1. **LM339**

LM339 merupakan IC *single supply quad comparator* yang didesain untuk digunakan pada aplikasi pendeteksi level, *low-level sensing,* dan aplikasi konsumsi memori, automotif, dan aplikasi elektronika industri.

1. Operasi 1 supply 3.0 V to 36V
2. Tegangan rata-rata ±1.5 V to ±18 V
3. Arus input bias yang rendah 25nA
4. Arus input offset rendah ±5.0 nA
5. Tegangan offset input rendah
6. Tegangan output saturasi yang rendah : 130mV @4.0 mA
7. Dapat digunakan untuk TTL dan CMOS

Gambar 5. LM339

1. **USART**

USART (*Universal Synchronous Asynchronous Receiver Transmitter*) merupakan sebuah device komunikasi serial yang sangat *flexibel.* Fitur utama dari USART adalah :

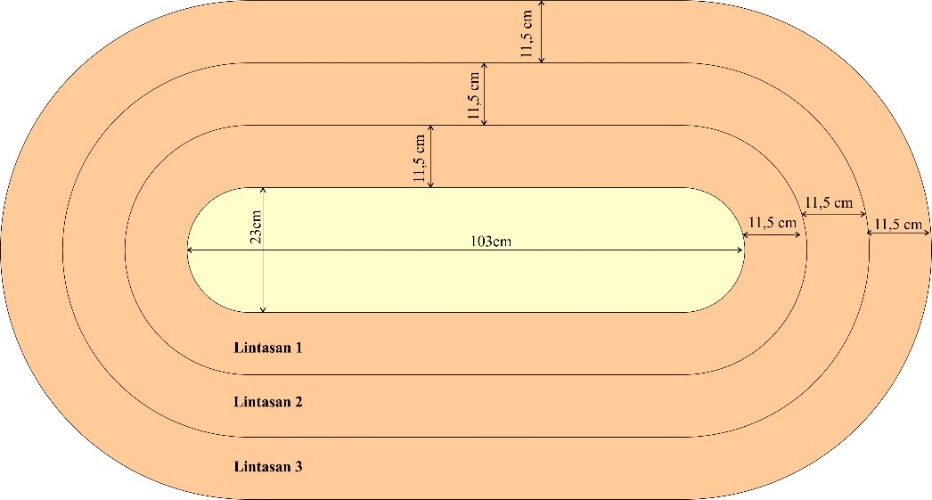
* Operasi full duplex (*Independent Serial Receive* dan *transmit register*)
* Dapat bekerja secara asinkronus atau sinkronus
* Operasi master dan slave *clock synchronous*
* Generator baudrate resolusi yang tinggi
* Serial frame dengan 5,6,7,8, atau 9 data bits dan 1 atau 2 stop bits
* Generator dengan parity odd dan even
* Deteksi data overrun
* Deteksi kesalahan frame
* *Noise filtering* termasuk false start bit detection dan digital *low pass filter*
* *Multi processor communition mode*
* Komunikasi mode asinkronus kecepatan penuh

**BAB III**

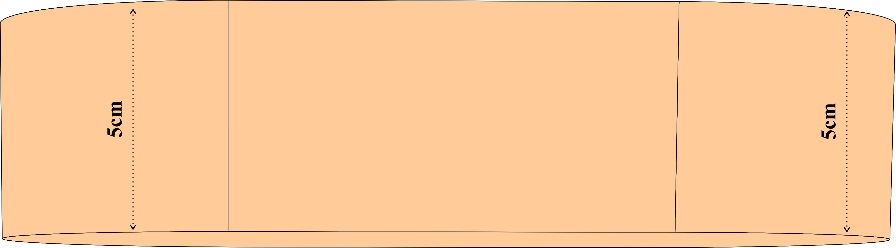
**PERENCANAAN SISTEM**

1. **Perencanaan Track**

*Track* merupakan lintasan yang akan dilalui objek. Pada perencanaan *track* terdiri atas 3 buah lintasan berbentuk elips yang akan dilewati oleh masing masing objek. Lintasan elips terdalam merupakan lintasan yang paling pendek dan lintasan elips terluas merupakan lintasan terpanjang dengan lebar lintasan 11,5cm untuk tiap lintasannya dan tinggi 5 cm. Berikut merupakan dimensi lintasan tampak atas dan samping.



Gambar 6. Track tampak atas



Gambar 7. Track tampak samping

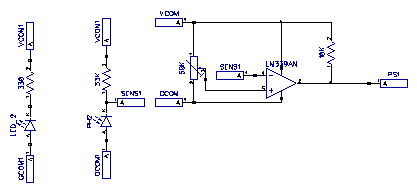
1. **Perencanaan *hardware***

Untuk dapat menghitung frekuensi putaran objek maka perlu diperlukan *device* yang akan mendeteksi objek yang melintasinya. Pada proyek ini digunakan led dan *photodiode* untuk dapat mendeteksi objek (mobil).

Perencanaan *hardware* ini meliputi

1. rangkaian sensor pembaca objek yaitu led, photodiode dan komparator
2. penempatan sensor pembaca dan komparator pada lintasan
3. rangkaian mikrokontroller yang berkolerasi dengan lcd, usart, dan push button
4. Rangkaian sensor

Pada rangkaian sensor ini terdiri atas led yang menggunakan 3 buah *super bright blue,* 3 buah *photodiode* dan 3 buah rangkaian komparator.

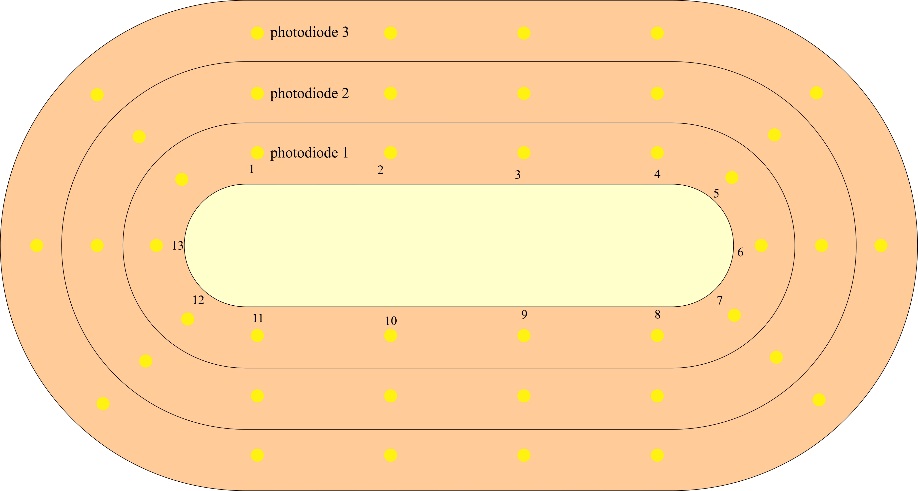


Sebuah led *super bright* biru dihubungkan dengan resistor 330Ω disupply dengan Common Voltage dan Common Ground. Cahaya ini kemudian ditangkap oleh sebuah *photodiode* yang terhubung dengan resistor 33KΩ. dengan Common Voltage dan Common Ground. Output dari *photodiode* kemudian dihubungkan dengan komparator dengan IC LM339 dengan VR bernilai 50KΩ dan *pull up* 10KΩ. output dari komparator akan terhubung ke mikrokontroller.

Tiga buah sensor dan komparator yang terpasang di *track* mempunyai konfigurasi rangkaian yang sama.

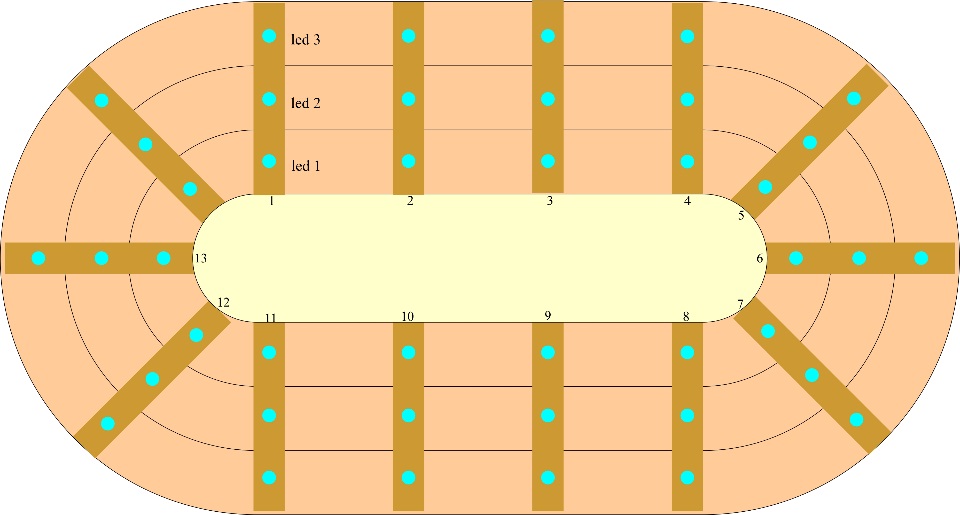
1. Penempatan led-*photodiode* dan komparator pada lintasan

Pada lintasan dipasang sebanyak 42 led-*photodiode* dan 13 buah komparator. *Photodiode* dipasang pada bagian bawah *track* yang terbagi menjadi 14 jalur. Masing-masing jalur terdiri atas 3 *photodiode* untuk masing masing lintasan.



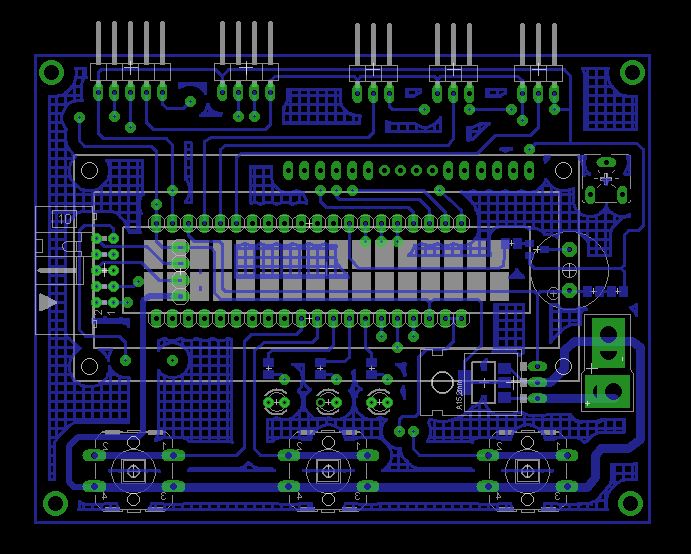
Gambar 8. Pemetaan photodiode

Sedangkan untuk led menggunakan *super bright blue* 3mm dipasang di bagian atas *track.* Led akan menghadap langsung ke *photodiode*

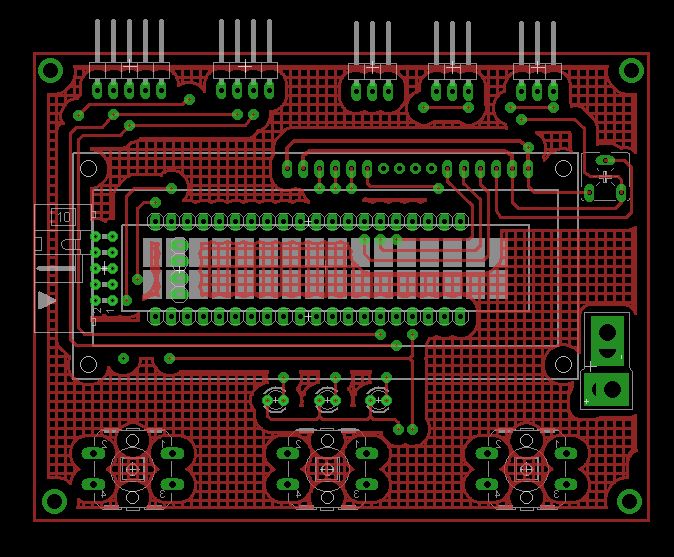


Gambar 9. Pemetaan led pada lintasanUntuk penempatan komparator

1. Rangkaian mikrokontroller yang berkolerasi dengan lcd, usart, dan push button

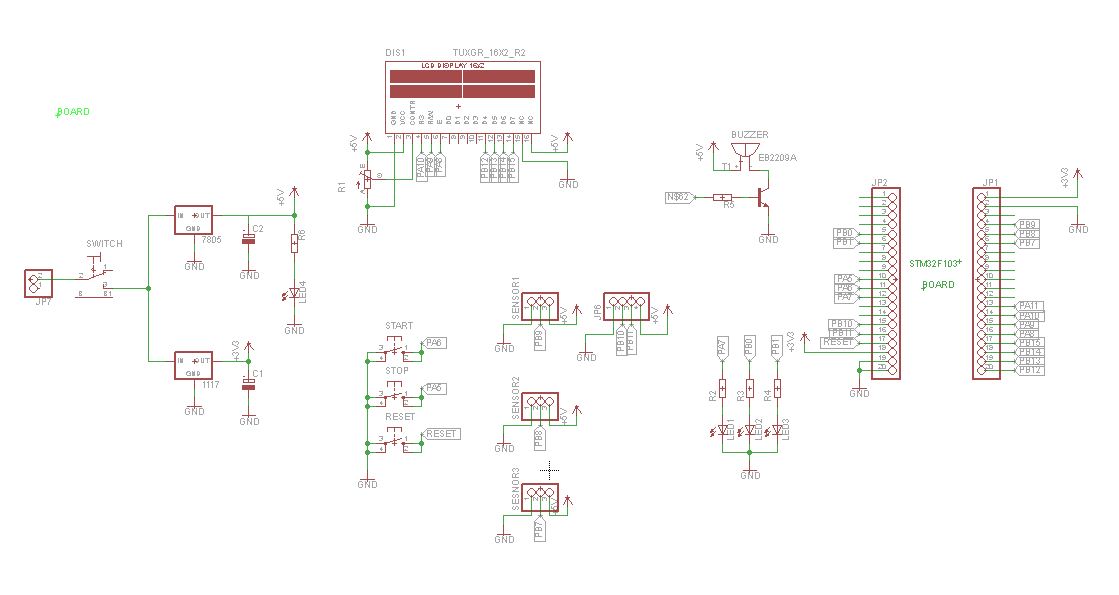


(a)



(b)

Gambar 10. Layout pada board (a) bottom layer, (b) top layer



Gambar 11. Rangkaian pada board



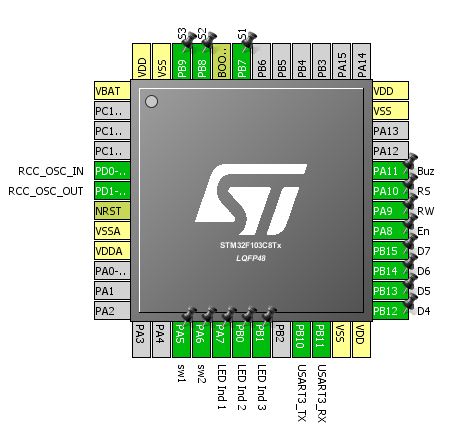
Gambar 12. Board STM32F103, buzzer, push button, dan LCD

Pada *board* yang digunakan terdiri atas

1. *Board* STM32F103 1 unit
2. Push Button 3 buah
3. LCD 1 buah
4. Led 3 buah
5. Buzzer 1 buah
6. **Perencanaan sistem**
7. **Setting IO**

Pada project dilakukan konfigurasi IO untuk ARM32F103C8Tx meliputi IO untuk LCD, komunikasi USART, Push Button, dan led indikator sebagai berikut:

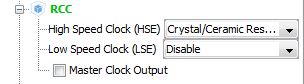
|  |  |  |
| --- | --- | --- |
| ***Peripheral*** | **PIN STM32F103** | **Label Pin** |
| **LCD** | **PA8** | **E** |
| **PA9** | **RW** |
| **PA10** | **RS** |
| **PB12** | **D4** |
| **PB13** | **D5** |
| **PB14** | **D6** |
| **PB15** | **D7** |
| **Buzzer** | **PA11** | **Buz** |
|  |  |  |
| **Led Indikator** | **PA7** | **LED Ind1** |
| **PB0** | **LED Ind 2** |
| **PB1** | **LED Ind 3** |
|  |  |
|  |  |
|  |  |  |
| **Push Button** | **PA5** | **START** |
| **PA6** | **STOP** |
|  |  |
|  |  |
|  |  |
|  |  |  |
| **USART** | **PA2** | **USART3\_TX** |
| **PA3** | **USART3\_RX** |



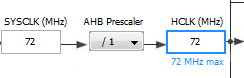
Gambar 13. Konfigurasi IO

1. **Setting Clock**

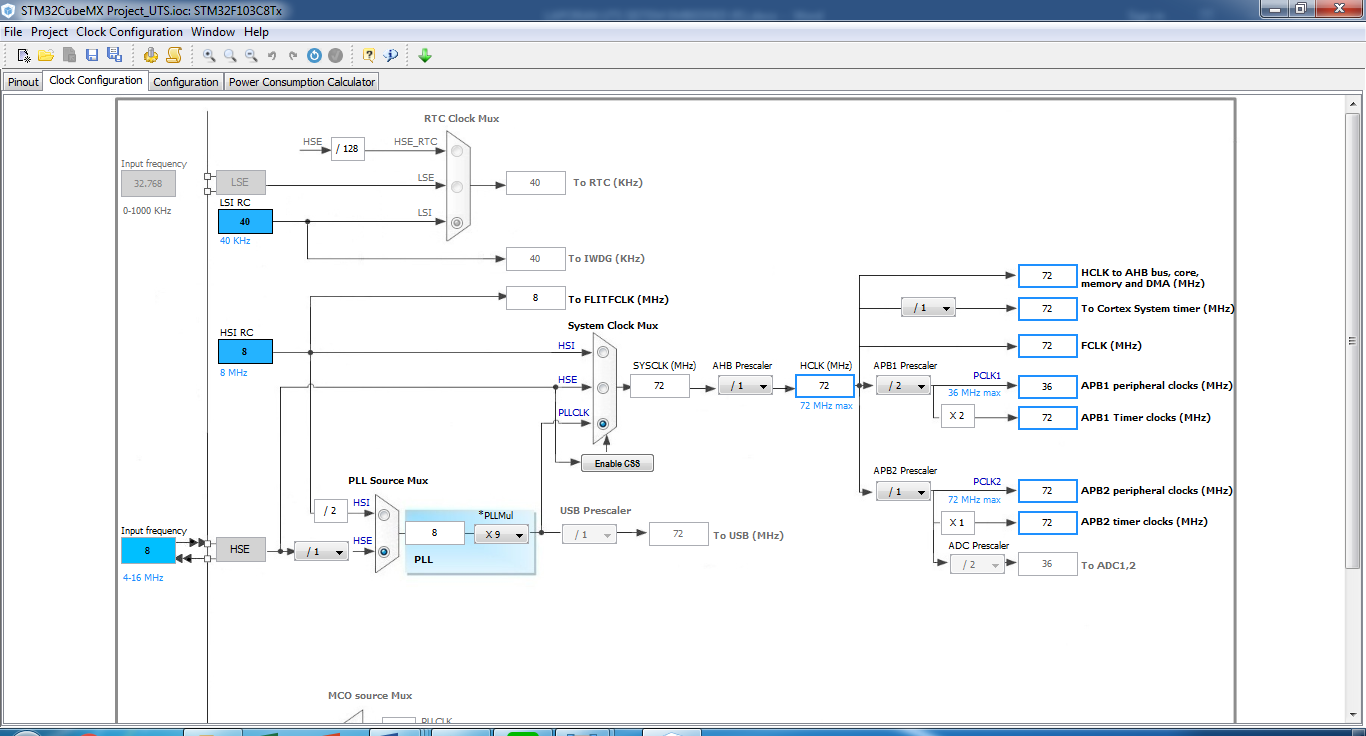
Karena clock dibangkitkan dengan menggunakan Xtal atau external crystal maka pada setting **RCC** pada tab **Pinout** di atur clock HSE dengan “**Crystal/Ceramic Resonator”**



Pada tab **Clock Configuration** mengubah nilai HCLK(MHz) menjadi nilai maksimal yaitu 72 MHz

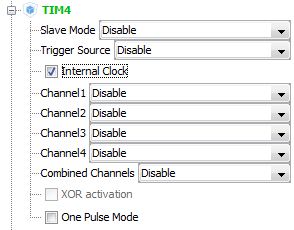


Kemudian menekan enter sehingga pada block **System Clock Mux** terpilih pada jalur PLLCLK.

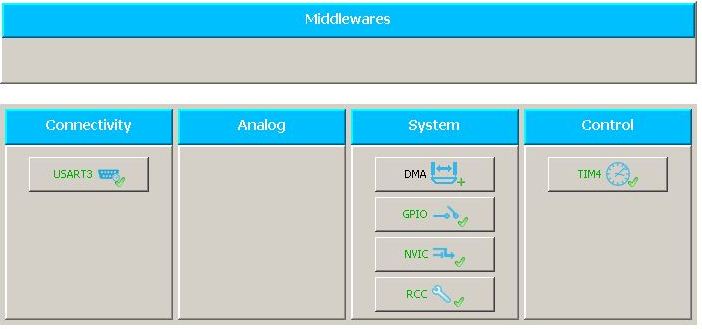


1. **Setting Timer**

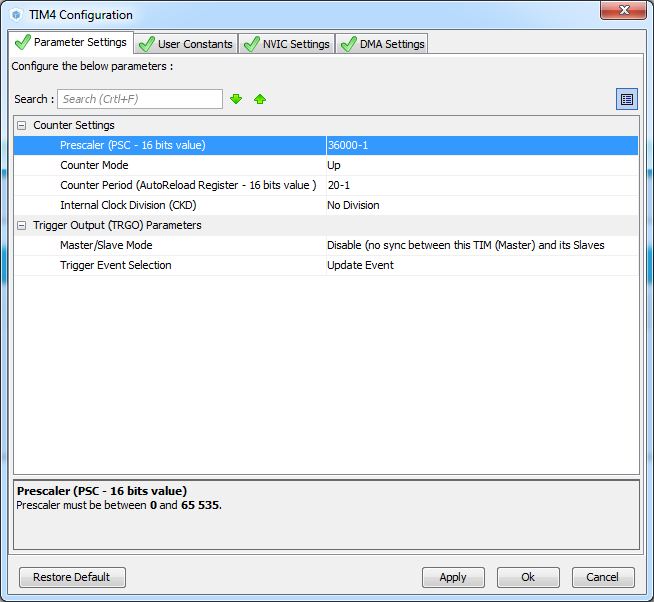
Pada project ini, untuk membangkitkan sebuah sistem pewaktuan atau timer, menggunakan timer 4. Oleh karenanya pada setting **TIM4** pada tab **Pinout** diaktifkan **Internal Clock**

******

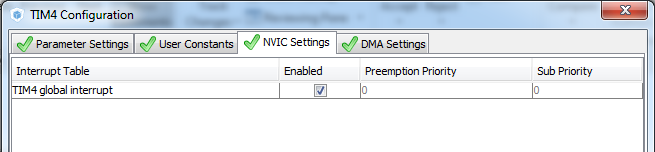
Pada tab **Configuration** mengklik **TIM4**

****

Maka akan muncul dialog TIM4 Configuration, kemudian pada tab **Parameter Settings** nilai dari **Prescaler (PSC-16 bits value)** menjadi 3600-1 dan mengubah nilai **Counter Period (AutoReload Register – 16 bits value)** menjadi 20-1.



Kemudian pada tab **NVIC Settings** mengaktifkan **TIM4 global interrupt**



Dan kemudian mengklik OK sebagai konfirmasi akhir perubahan pengaturan pada TIM4

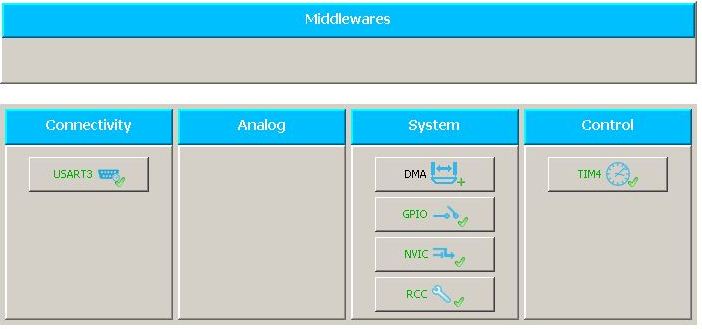
1. **Setting USART3**

Komunikasi yang digunakan untuk mengirim dan menerima data antara mikrokontroler dengan PC menggunakan komunikasi USART. Oleh karenanya pada project ini mengaktifkan USART3 untuk protokol komunikasi yang akan digunakan.

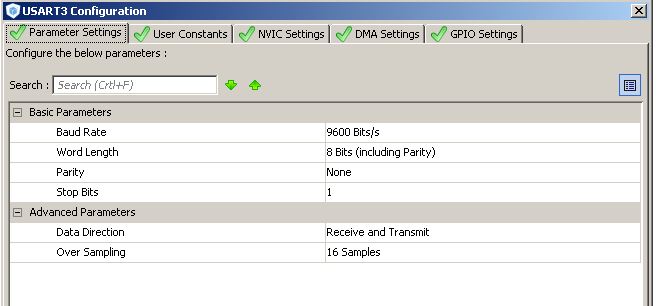
Pada tab **Pinout** setting **USART3**diatur dengan mode **Asynchronous**



Kemudian pada tab **Configuration** mengklik **USART3**



Maka kemudian akan muncul dialog **USART3 Configuration**. Pada tab **Parameter Settings** merubah nilai dari **Baudrate** menjadi 9600 Bits/s



Kemudian klik OK sebagai akhir dari konfigurasi USART3

Setelah dilakukan setting baik dari sisi *software* kemudian men*generate* parameter parameter yang telah diatur.

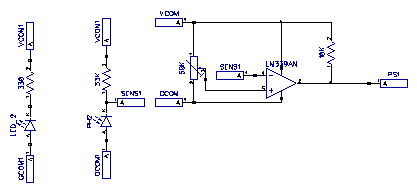
**BAB IV**

**ANALISA**

Pada project ini berujuan untuk Mahasiswa mampu mengakses pin Input dari sensor dan Output berupa Lampu LED dengan module STM32F103, mampu mengakses data serial dari STM32F103 sebagai komunikasi mengirimkan data serta mampu menampilkan data waktu dan jumlah lap dari input sensor. Oleh karenanya dilakukan beberapa pengujian yang terdiri atas pengujian komparator, pengujian program, dan pengujian komunikasi serial.

1. **Pengujian Komparator**

Komparator merupakan rangkaian elektronik yang akan membandingkan suatu input dengan referensi tertentu untuk menghasilkan output berupa dua nilai (high dan low). Suatu komparator mempunyai dua masukan yang terdiri dari tegangan acuan (Vreference) dan tegangan masukan (Vinput) serta satu tegangan ouput (Voutput).



Led akan memancarkan daya yang akan diterima oleh *photodiode*, nilai yang tegangan yang diterima oleh *photodiode* kemudian akan dibandingkan dengan tegangan referensi dari komparator.

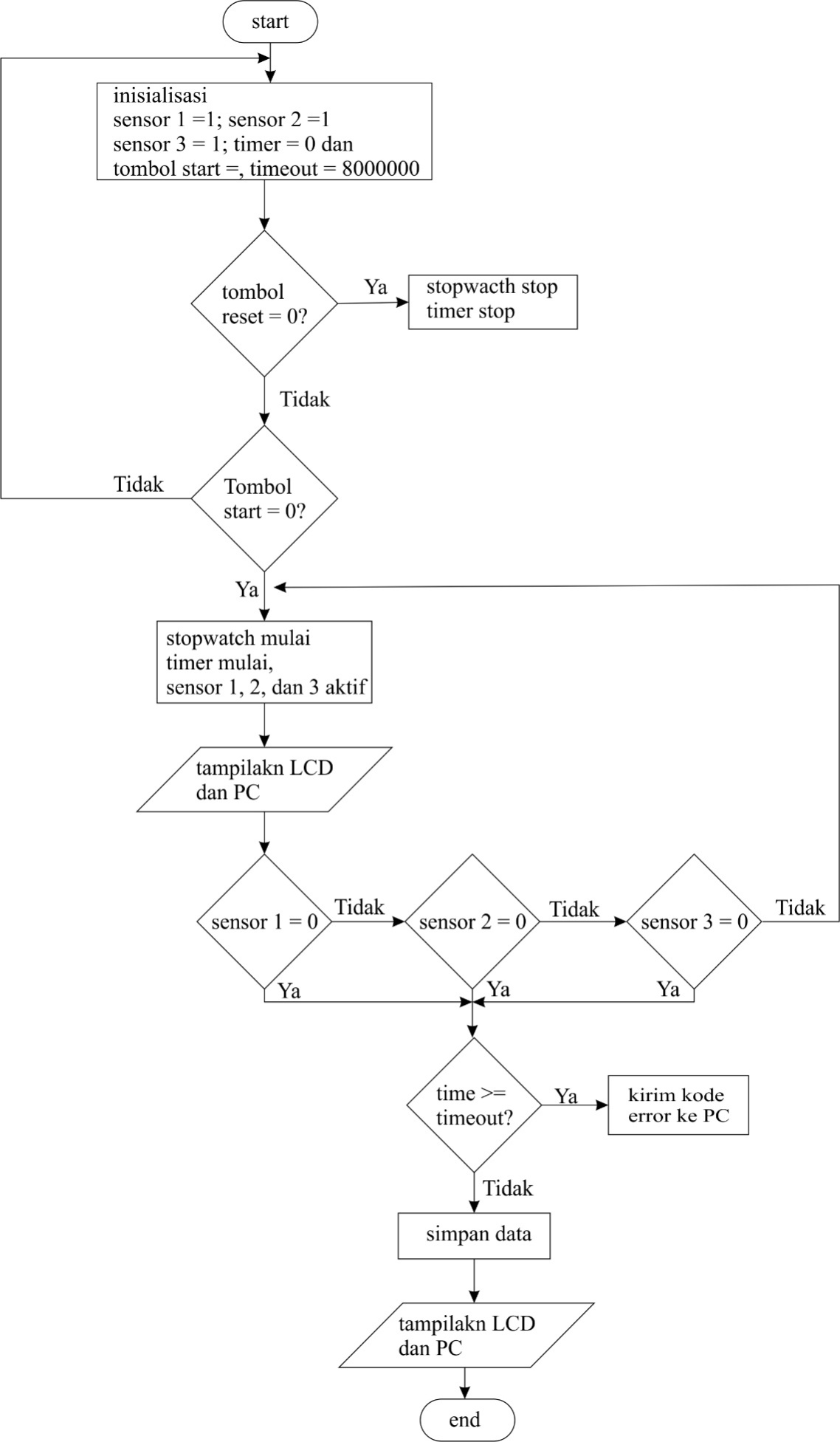
Dengan pengujian Vref sebesar 5V didapatkan hasil pengujian sebagai berikut

|  |  |  |
| --- | --- | --- |
| **Kondisi led - *photodiode*** | **Tegangan Output (V)** | **Keterangan** |
| Tidak terhalang Objek | 0 | Low |
| Terhalang Objek | 5 | High |

Berdasarkan hasil pengujian pada saat cahaya led diterima langsung oleh *photodiode*, tegangan output komparator 0V sedangkan pada saat *photodiode* tidak menerima cahaya dari led karena terhalang suatu objek tegangan output komparator 5V. misalkan kondisi 0V adalah kondisi *low* sedangkan kondisi 5V adalah kondisi *high,* maka ketika tegangan yang masuk ke komparator lebih dari tegangan referensinya maka komparator akan menghasilkan nilai *low* begitu pula sebaliknya ketika tegangan yang masuk ke komparator kurang dari tegangan referensinya maka menghasilkan nilai *high*.

1. **Pengujian Program**

Program yang akan diimplementasikan pada project merupakan program dimana terdapat 3 buah sensor yang akan melakukan perhitungan objek yang melewati sensor tersebut dengan mekanisme *anti* bouncing dan *timeout* dan serta melakukan perhitungan stopwatch. Oleh karena itu dibuat sebuah alur kerja program sebagai berikut



Diawali dengan melakukan inisialisasi sensor1, sensor2 dan sensor 3 untuk yang bernilai awal 1 untuk masing masing sensor. Timer bernilai awal 0 dan timeout 8000000. Push button reset dan start masing-masing bernilai awal 1.

Misalkan push button adalah normally open. Apabila push button reset ditekan maka nilai push button reset = 0 maka stopwatch akan reset dan timer juga akan reset. Jika push button reset tidak bernilai 0 atau tidak ditekan, maka diuji apakan push button start bernilai 0 atau tidak. Apabila push button start benar bernilai 0 maka perhitungan *stopwatch* dimulai, timer dimulai dan sensor 1, 2 dan 3 siap menerima data. Counting dari *stopwatch* akan terus ditampilkan di LCD dan PC.

Selama proses counting *stopwatch*, maka diuji apakah sensor 1 bernilai 0 atau mendeteksi sebuah objek, jika benar maka diuji apakah sensor mendeteksi objek selama waktu yang kurang dari *timeout* jika sensor mendeteksi objek selama waktu kurang dari *timeout*maka hitungan objek akan ditambah dan di tampilkan di LCD tetapi jika sensor mendeteksi objek selama waktu yang melebihi *timeout* maka akan mengirimkan pesan error ke PC dan LCD.

Namun jika sensor 1 tidak bernilai 0 maka dilanjutkan pengujian ke sensor 2, perlakuan kerja yang sama seperti halnya sensor1 juga berlaku di sensor 2. Jika ternyata sensor 2 tidak bernilai 0 atau tidak mendeteksi objek maka pengujian dilanjutkan ke sensor 3. Jika sensor 3 juga tidak mendeteksi objek maka pengujian kembali ke sensor 1.

Untuk dapat mengimplementasikan perencanaan tersebut dalam program, diawali dengan setting yang dilakukan pada bab III. Setelah parameter parameter setting di *generate* dari STMCube MX maka akan menghasilkan kode program berisi settings parameter dalam satu buah file main.c. apabila program dengan sistem kompleks terdeklarasi dalam satu buah file *source* saja maka akan sulit mendeteksi kesalahan apabila program berjalan tidak sesuai yang diinginkan. Oleh karenanya program yang berada dalam 1 program main.c dibagi menjadi beberapa file *source* dan  *header*.

File *header*  yang ditambahkan antara lain

1. *device.h*

*device.h* berisi inisialisasi *port* IO push button dan sensor yang digunakan berdasarkan tabel

|  |
| --- |
| **#ifndef** DEVICE\_H\_  **#define** DEVICE\_H\_  **#define** START\_Pin GPIO\_PIN\_6  **#define** START\_GPIO\_Port GPIOA  **#define** STOP\_Pin GPIO\_PIN\_5  **#define** STOP\_GPIO\_Port GPIOA  **#define** S1\_Pin GPIO\_PIN\_1  **#define** S1\_GPIO\_Port GPIOB  **#define** S2\_Pin GPIO\_PIN\_2  **#define** S2\_GPIO\_Port GPIOB  **#define** S3\_Pin GPIO\_PIN\_3  **#define** S3\_GPIO\_Port GPIOB  **#define** Buz\_Pin GPIO\_PIN\_11  **#define** Buz\_GPIO\_Port GPIOA  **#define** Ind\_S1\_Pin GPIO\_PIN\_7  **#define** Ind\_S1\_GPIO\_Port GPIOA  **#define** Ind\_S2\_Pin GPIO\_PIN\_0  **#define** Ind\_S2\_GPIO\_Port GPIOB  **#define** Ind\_S3\_Pin GPIO\_PIN\_1  **#define** Ind\_S3\_GPIO\_Port GPIOB  **#endif**/\* DEVICE\_H\_ \*/ |

Untuk dapat mengakses peripheral IO yang terhubung dengan board STM32F1 maka perlu mengkakses alamat GPIO

1. *hardwareinit.h*

hardwareinit.h berupa inisialiasi header fungsi SystemClock\_tuConfigdan GPIO\_Inittermasuk mengandung header library stm32f4xx.h, stm32f4\_discovery.hdandefine.h

|  |
| --- |
| **#ifndef** HARDWARE\_INIT\_H\_  **#define** HARDWARE\_INIT\_H\_  **voidSystemClock\_Config**(**void**);  **voidMX\_GPIO\_Init**(**void**);  **voidMX\_USART3\_UART\_Init**(**void**);  **voidError\_Handler**(**void**);  **#endif** |

1. *LCD.h*

*LCD.h* merupakan file *header* yang berisi definisi pin IO yang digunakan untuk akses ke LCD. Selain itu LCD.h juga memanggil file *header* dari*library* include "stm32f4xx.h

|  |
| --- |
| **#ifndef** LCD\_H\_  **#define** LCD\_H\_  **#include**"stm32f1xx\_hal.h"  **#define** EN\_Pin GPIO\_PIN\_8  **#define** EN\_GPIO\_Port GPIOA  **#define** RW\_Pin GPIO\_PIN\_9  **#define** RW\_GPIO\_Port GPIOA  **#define** RS\_Pin GPIO\_PIN\_10  **#define** RS\_GPIO\_Port GPIOA  **#define** D4\_Pin GPIO\_PIN\_12  **#define** D4\_GPIO\_Port GPIOB  **#define** D5\_Pin GPIO\_PIN\_13  **#define** D5\_GPIO\_Port GPIOB  **#define** D6\_Pin GPIO\_PIN\_14  **#define** D6\_GPIO\_Port GPIOB  **#define** D7\_Pin GPIO\_PIN\_15  **#define** D7\_GPIO\_Port GPIOB  /\* Deklarasi Pin Control \*/  **#define** Set\_En HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin, GPIO\_PIN\_SET);  **#define** Clr\_En HAL\_GPIO\_WritePin(EN\_GPIO\_Port, EN\_Pin, GPIO\_PIN\_RESET);  **#define** Set\_RW HAL\_GPIO\_WritePin(RW\_GPIO\_Port, RW\_Pin, GPIO\_PIN\_SET);  **#define** Clr\_RW HAL\_GPIO\_WritePin(RW\_GPIO\_Port, RW\_Pin, GPIO\_PIN\_RESET);  **#define** Set\_RS HAL\_GPIO\_WritePin(RS\_GPIO\_Port, RS\_Pin, GPIO\_PIN\_SET);  **#define** Clr\_RS HAL\_GPIO\_WritePin(RS\_GPIO\_Port, RS\_Pin, GPIO\_PIN\_RESET);  /\* Global Function Here \*/  **voidLCD\_Init**(**void**);  **voidlcd\_putchar**(**unsignedchar** c);  **voidlcd\_putstr**(**char** \*s);  **voidlcd\_clear**(**void**);  **voidlcd\_gotoxy**(**unsignedchar** col, **unsignedchar** row);  **#endif**/\* LCD\_H\_ \*/ |

1. *timer4.h*

*timer4.h* berisi header dari fungsi TIM4\_IRQHandlerdanTIM4\_Initsertamemanggil header dari librarystm32f4xx.h dan stm32f4\_discovery.h. Pada*tim4.h* juga mendeklarasikan variabel external count**,** second**,** minute**,** hourdengan tipe data integer dan TIM\_HandleTypeDef TIM\_Handle

|  |
| --- |
| **#ifndef** TIMER4\_H\_  **#define** TIMER4\_H\_  **#include**"device.h"  **#include**"stm32f1xx\_hal.h"  **#include**"stm32f1xx\_hal\_tim.h"  **#include**"counter.h"  **voidTIM4\_IRQHandler**(**void**);  **voidMX\_TIM4\_Init**();  **voidstart\_timer**(**void**);  **voidstoptimer**(**void**);  **extern**TIM\_HandleTypeDef htim4;  **#endif**/\* TIMER4\_H\_ \*/ |

1. *Sensor.h*

*Sensor.h* merupakan header yang menginisialisasi *header* dari fungsi fungsi yang akan menjadi respon dari push button. Oleh karena itu untuk dapat mengakses push button ini harus mengakses alamat pin yang terhubung dengan push button. Pin IO sudah terdefinisi pada header define.h sehingga agar dapat mengakses push button maka header define.h juga di deklarasikan pada sensor.h. sedangkan bagaimana respon apabila push button di tekan atau dilepaskan, dideklarasikan header masing masing fungsi atau respon dari masing – masing push button.

|  |
| --- |
| **#ifndef** SENSOR\_H\_  **#define** SENSOR\_H\_  **voidsaklarOn**(**void**);  **voidreset\_Timer**(**void**);  **voidcekReset**(**void**);  **voidgetLap**();  **int** flagOn;  **#endif**/\* SENSOR\_H\_ \*/ |

1. *Usart3.h*

*Usart3.h* berisi deklarasi yang berhubungan dengan pengiriman data serial. Data yang akan dikirimkan berupa *countingtimer* dari *stopwatch* yang dibangkitkan dari *tim4*, jumlah lap dan waktu saat melintasi lintasan. Sehingga pada header *usart3* ini dipanggil header dari *tim4.h* serta dideklarasikan variabel yang berhubungan data yang akan dikirim oleh mikrokontroller secara serial.

|  |
| --- |
| #include "stm32f4xx.h"  **#ifndef** USART3\_H\_  **#define** USART3\_H\_  **#include**<string.h>  **#include**"stm32f1xx\_hal.h"  **#include**"timer4.h"  GPIO\_InitTypeDef GPIO\_InitStruct;  **extern**UART\_HandleTypeDef huart3;  **externvolatilechar** received\_data;  **externunsignedchar** pbStartCount;  **externunsignedchar** Car0Count, Car1Count, Car2Count;  **externunsignedint** LapHour[3][4], LapMinute[3][4], LapSecond[3][4], LapCount[3][4];  **voidUSART3\_IRQHandler**(**void**);  **voidMX\_USART3\_UART\_Init**(**void**);  **externvoidUSART3\_PutChar**(**char** c);  **externvoidUSART3\_PutStr**(**char** \*data);  **#endif**/\* USART3\_H\_ \*/ |

Setalah file *header* siap maka selanjutnya adalah mengisi *header* fungsi yang telah dideklarasikan sebelumnya pada file *source*. file *source* ini merupakan file.c yang ditambahkan.

**BAB V**

**KESIMPULAN**

Setelah dilakukan pengerjaan projek UTS ini, dapat disimpulkan bahwa:

1. Untuk dapat menghindari buoncing sebuah sistem *object counting* yang menggunakan pembacaan sensor dan push button, maka diperlukan waktu delay selama 15 – 40 μs. Waktu delay ini untuk memastikan bahwa nilai atau sinyal yang terdeteksi oleh sensor merupakan sinyal pendeteksian objek, bukan merupakan sinyal spike.
2. Sebuah mekanisme timeout dibuat apabila pada sistem merupakan *human-machine interraction* atau *hardware-hardware interaction* untuk menghindari kondisi yang tidak diinginkan. Selain itu mekanisme timeout dengan mengirimkan kode error yang digunakan untuk mempermudah menemukan kesalahan pada system karena setiap erro memiliki kode tersendiri.
3. *Timer* yang dibangkitkan dapat digenerate dengan persamaan
4. *Timer* yang digunakan menggunakan *Timer Hardware ARM F1*