RANCANG BANGUN SISTEM VISUALISASI PEMBACAAN SENSOR GARIS PADA ROBOT *LINE FOLLOWER* BERBASIS KOMUNIKASI *WIRELESS*

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK ELEKTRONIKA

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

AHMAD SIROJUDDIN
NIM.125060300111013

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2016

LEMBAR PERSETUJUAN

RANCANG BANGUN SISTEM VISUALISASI PEMBACAAN SENSOR GARIS PADA ROBOT *LINE FOLLOWER* BERBASIS KOMUNIKASI *WIRELESS*

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

AHMAD SIROJUDDIN
NIM.125060300111013

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
pada 20 Juli 2016

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Ponco Siwindarto, M.Eng, Sc	Akhmad Zainuri, S.T., M.T.
NIP. 19590304 198903 1 001	NIP. 19840120 201212 1 003

Mengetahui,
Ketua Jurusan

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 19741203 200012 1 001

JUDUL SKRIPSI:

RANCANG BANGUN SISTEM VISUALISASI PEMBACAAN SENSOR GARIS PADA ROBOT *LINE FOLLOWER* BERBASIS KOMUNIKASI *WIRELESS*

Nama Mahasiswa : AHMAD SIROJUDDIN

NIM : 125060300111013

Program Studi : TEKNIK ELEKTRO

Konsentrasi : TEKNIK ELEKTRONIKA

Komisi Pembimbing :

Ketua : Dr. Ir. Ponco Siwindarto, M.Eng, Sc ……

Anggota : Akhmad Zainuri, S.T., M.T. ……

Tim Dosen Penguji :

Dosen Penguji 1 : Ir. Nanang Sulistyanto, M.T ……

Dosen Penguji 2 : Dr. Eng. Panca Mudjirahardjo, S.T., M.T. ……

Dosen Penguji 3 : Dr. Ing. Onny Setyawati, S.T., M.T., M.Sc ……

Tanggal Ujian : 23 Juni 2016

SK Peguji : 781/UN10.6/SK/2016

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 20 Juli 2016

Mahasiswa,

AHMAD SIROJUDDIN
NIM. 125060300111013

Kala ucapan selamat bergemuruh berkumandang
Piluh keringat perjuangan sayup-sayup menghilang
Saat kebersamaan memulai kisah album kenangan
Hangatnya persahabatan memudar menjadi bayang-bayang

Bukan ku bermaksud melupakanmu, kawan
Hanya lika-liku kehidupan yang kian bercabang
Bermesraan dengan waktu, ia semakin membentang
Akankah kebahagiaan itu menjelma menjadi memori ingatan

Jangan biarkan toga memadamkan kehangatan
Serta canda tawa yang telah kita ciptakan
Kebahagiaan bersamamu kan selalu terkenang
Terlukis indah tak akan pernah lekang

Siapkah engkau, memulai babak baru kehidupan
Yang mungkin tak seindah gemerlap kampus tersayang
Namun ku yakin piluh keringat yang tlah kita teteskan
Kan menuntun kita hadapi kerasnya peradaban

RINGKASAN

Ahmad Sirojuddin, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juni 2016, Rancang Bangun Sistem Visualisasi Pembacaan Sensor Garis pada Robot *Line Follower* Berbasis Komunikasi *Wireless*, Dosen Pembimbing: Ponco Siwindarto dan Akhmad Zainuri.

Line follower merupakan robot yang memiliki tugas untuk berjalan mengikuti garis yang terdapat pada lintasan. *Line follower* memiliki sensor garis yang dapat mengindra posisi robot relatif terhadap garis. Namun, sering kali terjadi kesalahan pembacaan oleh sensor garis robot *line follower* yang diakibatkan oleh berbagai faktor baik dari dalam maupun dari luar. Penyebab kesalahan pembacaan yang berasal dari sensor itu sendiri antara lain sudut antara pancaran cahaya *transmiter* (LED atau IR) dengan *receiver* (*photodioda* atau *phototransistor*) yang tidak tepat, usia dan kualitas komponen yang digunakan, masalah jalur listrik pada PCB, dan hubungan listrik antara satu komponen dengan komponen lain yang kurang baik. Adapun faktor luar di antaranya terdapat debu pada lintasan, garis pada lintasan yang warnanya mulai memudar, tidak ratanya lintasan, dan interferensi cahaya luar. Sayangnya, proses *troubleshooting* pada sensor garis ini sulit karena dinamika pembacaannya yang tidak kasa mata. Oleh karena itu, dibuatlah suatu sistem visualisasi pembacaan sensor yang mampu merekam serta memvisualisasikan dinamika pembacaan sensor garis pada PC.

Sistem terdiri atas beberapa bagian sebagai berikut : Sensor garis bertugas untuk membaca pantulan cahaya dari lintasan. Hasil keluaran sensor garis adalah 12 buah sinyal tegangan yang merepresentasikan kondisi masing-masing unit sensor cahaya. Mikrokontroler STM32F100RB berfungsi untuk membaca kedua belas sinyal tegangan dari sensor garis menggunakan periferal DAC dan mengubahnya menjadi paket data. Dua buah modul *bluetooth* HC-05 berfungi sebagai media pengirim data dari mikrokontroler dan sebagai media penerima data ke komputer. Program visual komputer berfungsi untuk menampilkan data hasil pembacaan sensor ke layar monitor komputer serta menyimpan data tersebut ke dalam suatu *data logger file*.

Terdapat beberapa pengujian yang dilakukan pada sistem. Hasil pengujian rangkaian sensor cahaya menunjukkan bahwa keluaran tegangan rangkaian sensor cahaya adalah 2,67 V ketika diletakkan di atas permukaan hitam dan 160,9 mV ketika diletakkan di atas permukaan putih. Hasil pengujian rangkaian *logic level shifter* menunjukkan bahwa rangkaian ini dapat menaikkan level tegangan logika HIGH mikrokontroler

STM32F100RB sebesar 2,98 V menjadi 4,92 V. Hasil pengujian rangkaian sensor garis secara keseluruhan menunjukkan bahwa tegangan keluaran yang terhubung dengan pin ADC mikrokontroler sama dengan nilai tegangan pada sensor cahaya sesuai dengan kombinasi logika pin selektor pada multiplekser. Selain itu, resistor variabel juga telah berhasil menurunkan tegangan keluaran sensor cahaya menjadi 0,67 kali. Hasil pengujian jarak modul *bluetooth* HC-05 pada ruang menunjukkan bahwa jarak maksimum komunikasi untuk posisi *bluetooth* horizontal adalah 8 meter dan untuk posisi vertikal adalah 35 meter. Hasil pengujian kecepatan transmisi data menunjukkan bahwa kecepatan transmisi data sistem visualisasi sensor garis adalah 95,98 paket data per detik. Hasil pengujian akurasi menunjukkan bahwa tingkat akurasi sistem visualisasi sensor garis adalah 95% untuk garis hitam dan 90% untuk garis putih. Hasil pengujian presisi sensor garis menunjukkan bahwa nilai *coefficient of variation* pada lintasan berwarna putih adalah sebesar 1,67% dan pada lintasan berwarna hitam adalah 5,19%. Hasil pengujian jarak sensor garis ke lintasan terhadap tegangan keluaran sensor menunjukkan bahwa sensor masih dapat bekerja secara normal untuk jarak maksimum sebesar 7 mm. Adapun pengujian keseluruhan menghasilkan empat buah mode visualisasi sensor garis (*stream data*, grafik, diagram batang, animasi) dan keempat mode visual tersebut dapat menampilkan informasi yang sesuai dengan kondisi sensor garis sebenarnya di lintasan.

Kata Kunci : Sensor Garis, Visualisasi, Komunikasi Serial, Program GUI Komputer

SUMMARY

***Ahmad Sirojuddin**, Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, June 2016. Design and Build of Line Sensor Reading Visualization System on Line Follower Robot Based on Wireless Communication, Academic Supervisor: Ponco Siwindarto and Akhmad Zainuri.*

Line follower is a robot that has task to tracking a line contained in a surface. Line Follower has a line sensor which can sense and read the robot position relatively to the line. However, often times there is an error reading caused by various factors both from within and from outside of sensor. The factors that come from sensor it self is the angle between IR transmitter to the receiver, component problem, electrical wire problem on PCB surface, and connections between component that are not good. The factors that come from outside is dust on the track surface, the color line that begin to fade, the track surface that is not even, and the light interference from other electromagnetic sources. Unfortunately, a troubleshooting process in the line sensor reading is difficult because the line sensor dynamical readings is not stank eye. Therefore, sensor readout visualization system that can record and visualize the dynamics of the line sensor reading on the PC is designed.

The system consist of the following section : line sensor has duty to read the reflected light from the track. The output of the line sensor is 12 pieces of voltage signal that represents the condition of each light sensor. STM32F100RB Microcontroller is used to read signals from the line sensor voltage by using DAC peripheral and turn it into data packets. Two HC-05 Bluetooth module has function as a sender data media from microcontroller and as a receiver data media to a computer. Graphically User Interface program on PC is used to display data of sensor readings to a computer screen and store the data into a data logger files.

There are several tests performed on the system. The tests result shows that the light sensor circuit output voltage of the light sensor circuit is 2.67 V when is placed on a black surface and 160.9 mV when is placed on a white surface. The level shifter circuit testing result indicates that circuit can pull up the voltage level of STM32F100RB microcontroller HIGH logic from 2.98 V to 4.92 V. The test result overall line sensor circuit that the output voltage connected to the microcontroller ADC pin is equal to the voltage value of light sensor according to the combination of the multiplexer selector pin. Moreover, the variable resistor has also succeeded in reducing the light sensor output

voltage to 0.67 times. The test result within the HC-05 Bluetooth module in the room shows that the maximum distance communication for the Bluetooth module in case of horizontal position is 8 m and in case of vertical position is 35 m. The test results of data transmission speed indicate that the data transmission speed of sensor visualization system is the 95.98 line data packets per second. The accuracy test results show that the visualization system's accuracy is 95% line sensor for black lines and 90% for white line. Results of testing precision line sensor indicates that the value of coefficient of variation on a white track is 1.67% and the black track is 5.19%. The test results of distance between sensor and the track surface indicates that the sensor can still work normally for a maximum distance of 7 mm. The entire testing produces four visualization modes (data streams, graphs, bar charts, animations) and the four visual mode can display information in accordance with the actual conditions on the track sensor line.

Keywords : Line Sensor, Visualization, Serial Communication, Graphically User Interface Program.

PENGANTAR

Puji syukur Penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat limpahan rahmat dan hidayat-Nya Penulis dapat menyelesaikan Laporan Skripsi yang berjudul ”Rancang Bangun Sistem Visualisasi Pembacaan Sensor Garis pada Robot *Line Follower* Berbasis Komunikasi *Wireless*”. Laporan ini dibuat dengan tujuan untuk untuk memenuhi persyaratan memperoleh gelar Sarjana Teknik pada Jurusan Teknik Elektro Konsentrasi Teknik Elektronika Fakultas Teknik Universitas Brawijaya Malang.

Dalam penyusunan Laporan ini tidak sedikit hambatan yang penulis hadapi, namun penulis menyadari bahwa kelancaran dam penyusunan Laporan ini berkat bantuan, dorongan, dan bimbingan dari berbagai pihak baik secara langsung maupun tidak langsung, untuk itu penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yang Maha Esa, atas limpahan rahmat dan hidayahNya sehingga penyusunan Skripsi ini dapat diselesaikan
2. Bapak, Ibu dan seluruh anggota keluarga, atas dukungan dan doa yang telah diberikan.
3. Yang terhormat Bapak M. Aziz Muslim, ST., MT., Ph.D. selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya
4. Yang terhormat Ibu Nurrusa’adah selaku Ketua Kelompok Dosen Konsentrasi Elektronika Jurusan Teknik Elektro Universitas Brawijaya.
5. Yang terhormat Bapak Ponco Siwindarto selaku dosen pembimbing 1 yang selalu memberikan bimbingan, arahan, dan motivasi dalam penyusunan Skripsi ini.
6. Yang terhormat Bapak Akhmad Zainuri selaku dosen pembimbing 2 yang selalu memberikan dukungan, semangat, dan solusi dalam penyusunan Skripsi ini.
7. Teman-teman Tim Robot LTC angkatan 2012 Cahyo, Hari, Alfian, Fafa, dan Arif atas dukungan dan bantuannya.
8. Teman-teman Lab. Desain dan Prototipe 2012 Ronny, Guntoro, Wildan, Bagus, Dianata, Bidin dan Rifqa atas dukungan, bantuan dan semangat yang telah diberikan.
9. Teman-teman TEUB tercinta terutama teman-teman Paket B Konsentrasi Teknik Elektronika yang selalu memberikan semangat, dorongan dan bantuan pikiran.

Penulis berharap semoga laporan ini dapat memberikan manfaat dan dapat dijadikan referensi di masa yang akan datang. Penulis sadar bahwa laporan ini masih banyak kekurangan dan jauh dari sempurna, oleh karena itu kritik dan saran yang membangun penulis harapkan demi kesempurnaan laporan ini.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1.Latar Belakang

Kemajuan teknologi sangat penting untuk memenuhi kebutuhan hidup manusia, karena teknologi merupakan salah satu penunjang taraf kehidupan manusia. Pada awalnya teknologi dibuat untuk mempermudah setiap kegiatan manusia. Teknologi bersumber dari pemikiran manusia yang senantiasa ingin mempermudah manusia itu sendiri dalam melakukan berbagai kegiatan di kehidupan.

Universitas sebagai salah satu lembaga pendidikan yang turut memberikan andil dalam regenerasi bangsa harus memberikan pelajaran dan pengetahuan mengenai teknologi dan perkembangannya kepada para mahasiswa yang sedang menuntut ilmu. Berbagai metode telah dilakukan oleh universitas untuk membuat mahasiswa sebagai generasi penerus bangsa ini selalu mengikuti perkembangan teknologi, mulai dari seminar, menjadikan teknologi sebagai salah satu silabus mata kuliah, sistem perkuliahan yang berbasis teknologi, sertifikasi IT, dan pengadaan lomba yang berkaitan dengan teknologi.

Salah satu lomba yang bertemakan teknologi yang sering diadakan di tingkat mahasiswa adalah lomba *line follower robot*. *Line follower* merupakan sebuah robot yang memiliki tugas untuk berjalan mengikuti garis yang terdapat pada lintasan. *Line follower* memiliki sensor garis yang dapat mengindra posisi robot relatif terhadap garis. Dalam perlombaan, robot yang paling cepat menempuh perjalanan mulai dari posisi *start* sampai *finish* dinyatakan sebagai pemenang. Dalam perjalanannya, robot sering kali menemui *intersection* lintasan baik itu berupa pertigaan ataupun perempatan dan robot harus menentukan arah mana yang harus ia ambil agar sampai pada posisi *finish*. Oleh karena itu, peranan dari sensor garis pada *line follower robot* sangatlah penting.

Akan tetapi, sering kali terjadi kesalahan pembacaan oleh sensor garis *line follower robot*. Pangkal dari terjadinya kesalahan ini adalah ketidaksesuaian nilai ADC yang terbaca oleh sensor dibandingkan dengan nilai yang seharusnya. Akibatnya, nilai logika unit sensor menjadi tidak sesuai dengan semestinya. Ketika seluruh unit sensor telah dibaca, akan terbentuk pola pembacaan garis yang tidak sesuai dengan kondisi sesungguhnya di lintasan. Pola baru ini sering kali merupakan pola lain yang tidak sesuai dengan yang diperkirakan oleh perancang robot. Efek dari terjadinya hal ini berbagai

macam, seperti robot yang mengambil arah yang salah ketika bertemu *intersection* lintasan, tiba-tiba berhenti, atau bahkan keluar dari lintasan.

Kesalahan pembacaan yang terjadi bisa disebabkan oleh masalah yang terdapat pada sensor itu sendiri maupun pengaruh *noise* yang berasal dari luar. Penyebab kesalahan pembacaan yang berasal dari sensor itu sendiri antara lain sudut antara pancaran cahaya *transmiter* (LED atau IR) dengan *receiver* (*photodioda* atau *phototransistor*) yang tidak tepat, usia dan kualitas komponen yang digunakan, masalah jalur listrik pada PCB, dan hubungan listrik antara satu komponen dengan komponen lain yang kurang baik. Adapun faktor luar di antaranya terdapat debu pada lintasan, garis pada lintasan yang warnanya mulai memudar, tidak ratanya lintasan, dan interferensi cahaya luar. Akibat dari hal-hal tersebut adalah pola garis yang terbaca oleh robot tidak sama dengan pola garis yang sebenarnya pada lintasan.

Sayangnya, masalah-masalah ini sulit sekali dideteksi dan ditangani oleh perancang robot. Kesulitan ini disebabkan oleh banyaknya faktor yang menyebabkan kesalahan pembacaan pada sensor. Kendala lain adalah tidak dimungkinkannya pengimplementasian alat ukur atau instrumen listrik ketika robot sedang berada di lintasan, terutama ketika robot sedang berjalan.

Oleh karena itu, peneliti bermaksud untuk membuat sistem visualisasi pembacaan sensor garis *line follower robot* pada skripsi ini. Sistem nantinya akan dapat menampilkan nilai pembacaan dari tiap-tiap unit sensor. Bentuk visualisasinya dapat berupa grafik terhadap waktu, *bar chart*, atau gambar sensor garis itu sendiri yang disertai kondisi tiap unit *receiver*. Sistem yang dirancang menggunakan modul *wireless* agar mudah dalam pengaplikasiannya. Dengan dirancangnya sistem visualisasi ini, diharapkan dapat mempermudah tim LTC dalam perancangan dan *troubleshooting* sensor garis *line follower robot*.

1.2.Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya, maka rumusan masalah pada penelitian ini adalah sebagai berikut :

a. Bagaimana karakteristik sensor garis yang akan dirancang, yang meliputi *output* tegangan, presisi, akurasi, dan jarak vertikal dengan lintasan?
b. Bagaimana karakteristik transmisi sistem visualisasi yang akan dirancang, yang meliputi jarak maksimum komunikasi, laju pengiriman data, dan pengaruh jarak terhadap daya disipasi pada modul *wireless*?

1.3.Pembatasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan akan diberi batasan sebagai berikut :

a. Menggunakan desain sensor garis berbentuk setengah lingkaran karena dapat terjadi osilasi sehingga tampilan visualisasi menjadi lebih dinamis (Prakoso, 2014:51).
b. Pengolahan data hasil pembacaan sensor oleh sistem kontroler *line follower robot* tidak dibahas.
c. Analisis mengenai performansi komputer dalam pengolahan data dan instruksi dalam program visual tidak dibahas.
d. Jarak antara modul *wireless* pada robot dan pada komputer tidak pernah melebihi jarak 10 meter.
e. Tidak terdapat penghalang antara kedua modul *wireless* pada mikrokontroler maupun pada komputer.
f. Sistem diimplementasikan dalam ruang tertutup seperti gedung.

1.4.Tujuan

Penelitian ini bertujuan untuk merancang dan membuat sistem yang dapat memvisualisasi serta menampilkan ulang hasil pembacaan sensor garis pada *line follower robot* dan berbasis komunikasi *wireless.*

1.5.Manfaat/Kegunaan

Penelitian mengenai ” Rancang Bangun Sistem Visualisasi Pembacaan Sensor Garis Robot pada *Line Follower* Berbasis Komunikasi *Wireless*” ini memiliki beberapa manfaat dan kegunaan, antara lain :

a. Bagi tim *line follower robot* TEUB khususnya dan bagi akademisi yang melakukan riset di bidang robot ini, skripsi ini berguna untuk membantu perancang dalam melakukan *troubleshooting* sensor garis dan merekam dinamika pembacaan sensor selama robot beroperasi.
b. Bagi civitas akademisi di bidang Teknik Elektro, penelitian ini dapat digunakan untuk merekam hasil pembacaan sensor oleh suatu sistem selain sensor garis sehingga mempermudah dalam kegiatan analisis.

BAB II
TINJAUAN PUSTAKA

Dalam perancangan sistem visualisasi pembacaan sensor garis pada *line follower robot* ini dibutuhkan beberapa teori dan referensi yang harus dipelajari terlebih dahulu sebagai dasar dalam melakukan perancangan. Hal-hal yang harus dipelajari terlebih dahulu antara lain mengenai apa itu *line follower robot*, sensor garis pada *line follower robot*, sensor cahaya fototransistor, rangkaian *logic translator* sebagai antar muka antar *device* yang berbeda operasi tegangannya, mikrokontroler ARM STM32F100RB, *development board* STM32 *value line discovery*, modul *bluetooth* HC-05 sebagai media transmisi data nirkabel, dan bahasa pemrograman java.

2.1.Line Follower Robot

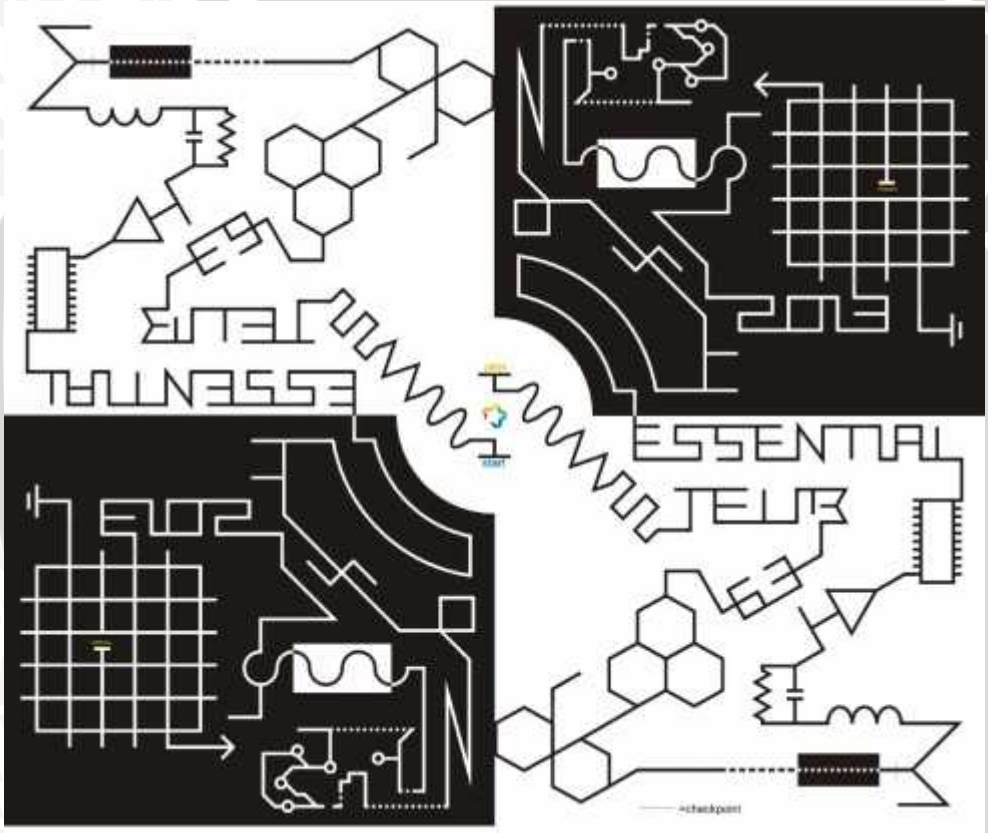
Line follower adalah sebuah mesin yang dapat berjalan mengikuti suatu jalur. Jalur ini dapat berupa sesuatu yang terlihat oleh mata manusia, seperti garis hitam pada permukaan putih (atau sebaliknya) atau dapat juga berupa sesuatu yang tak terlihat oleh mata manusia seperti medan magnet. Gambar 2.1 merupakan contoh dari *line follower robot. Line follower robot* bekerja dengan cara mensensor garis dan melakukan manuver agar robot tetap berada pada jalur. Robot secara terus-menerus membaca posisinya relatif terhadap garis menggunakan suatu mekanisme umpan balik dan melakukan aksi kontrol pada aktuator (pada umumnya berupa motor) agar robot tetap berada pada jalur. Perancang robot harus dapat ”mengajari” robot agar ia dapat menjalankan tugas ini melalui suatu kode program (Patil, 2005).

Gambar 2.1. *Line follower robot*

Sumber : Jatmika (2011 : 133)

Line follower robot memiliki berbagai fungsi dalam berbagai bidang, baik dalam bidang industri, pendidikan, dan lain-lain. Dalam bidang industri, robot ini digunakan sebagai alat transportasi yang mengantarkan barang-barang dari satu tempat ke tempat

yang lain. Dalam bidang pendidikan, robot ini digunakan sebagai dasar pelatihan dan pengenalan robot bagi pemula. Robot ini dapat pula diikutkan pada ajang kompetisi *line follower robot*. Dalam kompetisi *line follower robot*, robot yang berhasil mencapai garis *finish* yang paling cepat lah yang akan menjadi pemenang. Dalam perlombaan, lintasan yang digunakan sering kali terdapat *intersection* (percabangan) baik berupa pertigaan maupun perempatan. Robot harus dapat mengambil jalan yang benar agar berhasil mencapai garis *finish*. Gambar 2.2 merupakan contoh lintasan lomba *line follower robot* pada *event* ESSENTIALS yang diselenggarakan oleh Jurusan Teknik Elektro Universitas Brawijaya pada tahun 2013.



Gambar 2.2 Contoh Lintasan yang Digunakan dalam Lomba *Line follower robot*

Sumber : HME FT UB (2013 : 11)

2.2. Sensor Garis *Line Follower Robot*

Sensor garis pada *line follower robot* berfungsi untuk membaca posisi badan robot relatif terhadap garis pada lintasan. Sensor garis ini pada umumnya terdiri atas susunan sensor cahaya fotodiode atau fototransistor yang membentuk pola tertentu. Sensor cahaya ini memiliki nilai resistansi yang berubah-ubah tergantung dari nilai intensitas cahaya yang diterimanya. Sensor cahaya ini dirangkai sedemikian dengan komponen lain sehingga membentuk rangkaian seperti rangkaian pembagi tegangan.

Pada umumnya, metode yang digunakan sensor garis dalam *line follower robot* untuk membaca intensitas cahaya pantulan lintasan jika menggunakan fototransistor adalah menggunakan rangkaian *common emitter*, seperti yang ditunjukkan pada Gambar 2.3. nilai tegangan V_{out} dapat dihitung dengan menggunakan rumus sebagai berikut (Boylestad, 1996).

$$V_{out} = V_{cc} - R_{seri} \times I_C \quad (6.1)$$

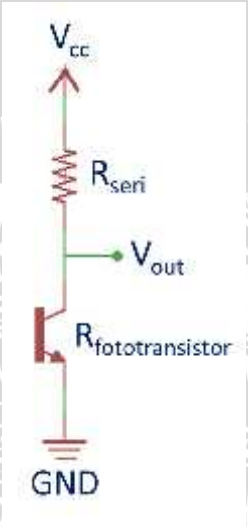
dengan :

V_{out} = nilai tegangan keluaran dari rangkaian *common emitter*

I_C = arus yang mengalir melalui *colector* fototransistor

R_{seri} = nilai resistansi dari resistor yang dirangkai seri dengan fototransistor

V_{cc} = nilai tegangan catu rangkaian.



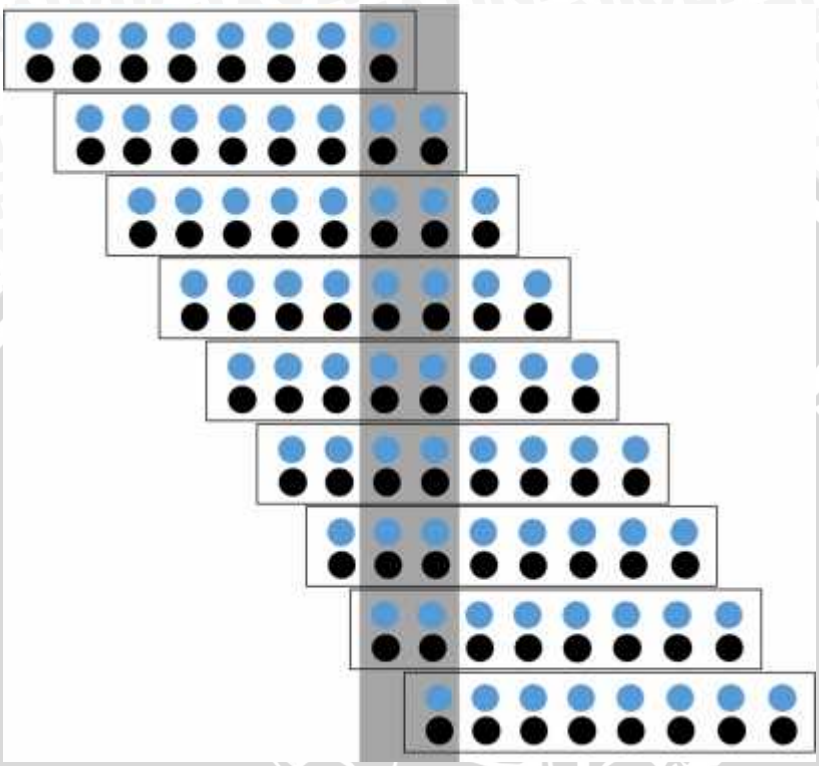
Gambar 2.3. Rangkaian *Common Emitter* untuk *Receiver*

Sumber : Agrawal (2008 : 4)

Ketika fototransistor bersifat isolatif (menerima sedikit cahaya), maka nilai V_{out} akan mendekati nilai V_{cc}. Sedangkan ketika fototransistor bersifat konduktif (menerima banyak cahaya), maka nilai V_{out} akan mendekati GND. Perbedaan nilai inilah yang selanjutnya diolah oleh prosesor untuk menentukan posisi sensor, apakah sedang berada di permukaan hitam ataukah sedang berada di permukaan putih. Semakin besar rentang nilai perbedaan ini, maka dapat dikatakan bahwa nilai sensitivitas sensor garis semakin baik.

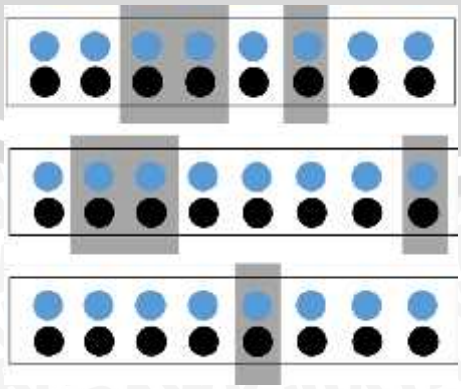
Sensor garis pada *line follower robot* dalam perancangan ini terdiri atas beberapa fototransistor yang disusun dengan pola tertentu untuk memetakan pola garis atau membaca posisi robot relatif terhadap garis. Pada umumnya, susunan fototransistor pada

papan rangkaian PCB berupa garis lurus ataupun berupa lengkungan. Gambar 2.4 merupakan ilustrasi macam-macam posisi *line follower robot* relatif terhadap garis. Dalam keadaan normal, lebar garis pada lintasan lomba bertepatan dengan dua buah fototransistor (khusus untuk sensor paling samping terdapat kemungkinan satu sensor saja).



Gambar 2.4. Macam-Macam Kemungkinan Posisi Sensor Relatif terhadap Garis

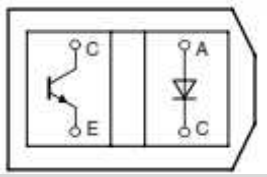
Akan tetapi, sering kali terjadi kesalahan pembacaan garis oleh sensor garis, baik itu disebabkan oleh faktor dari internal sensor maupun faktor eksternal luar sensor. Akibatnya, pola sensor yang terbaca menjadi berada di luar estimasi dan menyebabkan robot bergerak tidak sebagai mana mestinya. Contoh kondisi pembacaan sensor ketika ada gangguan terdapat dalam Gambar 2.5.



Gambar 2.5. Contoh Kondisi Pembacaan Sensor ketika ada Gangguan

2.3. Sensor Cahaya TCRT5000

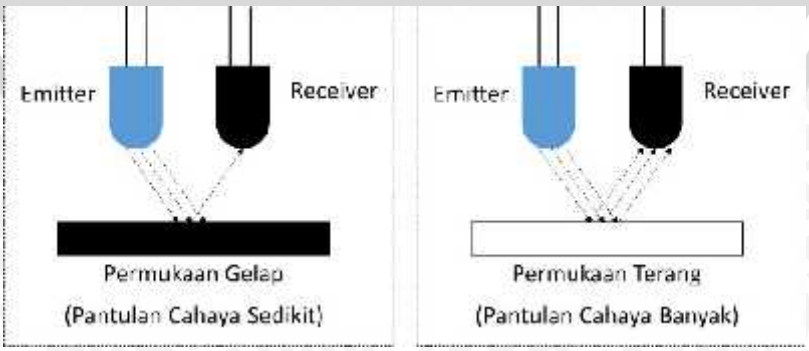
TCRT5000 adalah sebuah sensor yang menggunakan prinsip pemantulan cahaya. Sensor ini terdiri atas sebuah *transmitter* yang memancarkan gelombang dengan frekuensi inframerah dan *tranceiver* yang berupa sebuah fototransistor, seperti yang terdapat dalam Gambar 2.6. *Transceiver* pada TCRT5000 hanya peka terhadap gelombang yang dipancarkan oleh *transmitter* sehingga lebih bersifat resistan terhadap pengaruh cahaya tampak. Kedua komponen ini biasanya sudah terpaket dalam satu kemasan tertentu.



Gambar 2.6. Skema Sensor Cahaya TCRT5000

Sumber : Vishay Semiconductor (2009:1)

Ketika *emitter* diberi nilai tegangan tertentu, maka akan timbul arus yang melalui *emitter* ini sehingga memancarkan cahaya dengan frekuensi inframerah. Cahaya ini tidak langsung dipancarkan menuju fototransistor, akan tetapi dipancarkan menuju ke sebuah permukaan suatu benda dan pantulannya akan diterima oleh fototransistor. Oleh karena itu, intensitas cahaya yang diterima oleh fototransistor ini dipengaruhi oleh warna dari permukaan benda yang memantulkan cahaya. Jika permukaan benda ini berwarna gelap, maka cahaya yang diterima oleh fototransistor menjadi sedikit. Sebaliknya, jika permukaan benda berwarna terang, maka cahaya yang diterima oleh fototransistor menjadi banyak. *Receiver* ini sendiri memiliki sensitivitas terhadap cahaya. Jika cahaya yang diterima fototransistor sedikit, maka resistansinya akan semakin besar. Sebaliknya jika cahaya yang diterima banyak, maka resistansinya akan semakin kecil. Prinsip kerja sensor cahaya TCRT5000 diilustrasikan dalam Gambar 2.7.

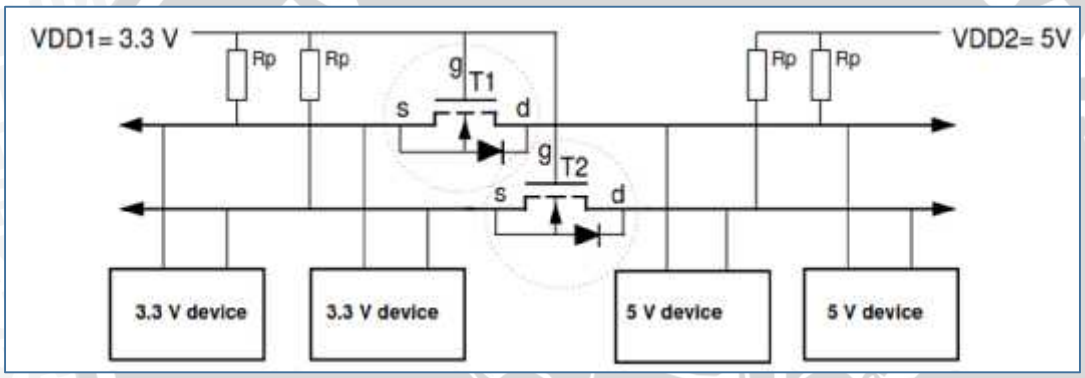


Gambar 2.7. Prinsip Kerja Sensor Cahaya TCRT5000

Sumber : Soebhakti (2008:2)

2.4. *Bi-Directional Logic Level Shifter*

Bi-Directional Logic level shifter digunakan untuk menghubungkan dua buah pin I/O yang memiliki level nilai tegangan yang berbeda ketika berlogika HIGH. Hal ini sering terjadi ketika sebuah desain rangkaian melibatkan antar muka antara *device* berteknologi TTL dan *device* berteknologi CMOS. Selain itu, mikrokontroler juga memiliki nilai tegangan logika HIGH yang berbeda satu dengan lainnya, tergantung jenisnya. Sebagai contoh, mikrokontroler AVR memiliki nilai output tegangan sekitar 5 V untuk logika HIGH, sedangkan mikrokontroler seri STM32 memiliki output tegangan sekitar 3,3 V untuk logika HIGH. Untuk itu, dibutuhkan suatu rangkaian pengondisi agar komponen elektronika yang memiliki nilai level tegangan berbeda tetap dapat berkomunikasi. Rangkaian *bi-directional logic level shifter* ditunjukkan dalam Gambar 2.8.



Gambar 2.8. Rangkaian *Bi-Directional Logic Level Shifter*

Sumber : Schutte (1997:10)

Tiap rangkaian *logic translator* terdiri atas sebuah MOS-FET *enhancement* kanal N. Pin *gate* terhubung dengan catu tegangan yang lebih rendah (V_{DD1}), pin *source* terhubung dengan *device* bertegangan lebih rendah, dan pin *drain* terhubung dengan *device* bertegangan lebih tinggi. Sebagian besar MOSFET memiliki substrat yang telah terhubung dengan *source* secara internal. Diode antara *drain* dan substrat berada di dalam MOSFET itu sendiri dan bertindak sebagai sambungan n-p antara *drain* dengan substrat. Terdapat tiga buah *state* (kondisi) yang menjadi cara kerja rangkaian *logic translator* ini:

- Kondisi 1 : tidak ada satupun *device* yang berlogika LOW. Pada rangkaian, *device* yang bertegangan lebih rendah dihubungkan ke tegangan 3,3 volt melalui resistor *pull-up*. Kaki *gate* dan *source* MOSFET keduanya bertegangan 3,3 volt, sehingga nilai tegangan V_{GS} lebih rendah dari pada nilai tegangan *treshold* sehingga MOSFET tidak terkonduksi. Hal ini menyebabkan *device* bertegangan lebih tinggi memiliki

tegangan 5 volt melalui resistor *pull up*. Alhasil, kedua *device* berlogika HIGH namun pada level tegangan yang berbeda.

- Kondisi 2 : *device* bertegangan 3,3 volt berlogika LOW. Kaki *source* MOSFET juga menjadi LOW sedangkan kaki *gate* tetap bertegangan 3,3 volt. Nilai tegangan V_{GS} menjadi naik di atas nilai tegangan *treshold* sehingga MOSFET menjadi terkonduksi. Akibatnya, *node* yang terhubung dengan *device* bertegangan lebih tinggi juga menjadi LOW melalui MOSFET yang terkonduksi. Alhasil, kedua *device* menjadi berlogika LOW dengan nilai level tegangan yang sama.
- Kondisi 3 : *device* bertegangan 5 volt berlogika LOW. Oleh karena *device* bertegangan 3,3 volt awalnya berlogika HIGH, maka diode yang menghubungkan antara *drain* dengan substrat dan *source* menjadi terkonduksi. Arus mengalir dari *source* ke *drain* melalui diode dan tegangan pada *source* berangsur-angsur turun hingga nilai V_{GS} menjadi di atas nilai tegangan *treshold*. Akibatnya, *node* yang terhubung dengan *device* bertegangan lebih rendah juga menjadi LOW melalui MOSFET yang terkonduksi. Alhasil, kedua *device* menjadi berlogika LOW dengan nilai level tegangan yang sama (Schutte, 1997:10)

2.5. Mikrokontroler STM32F100RB

STM32F100RB merupakan merupakan mikrokontroler 32 bit yang memiliki arsitektur RISC (*Reduced Instruction Set Computing*). Mikrokontroler ini termasuk salah satu seri yang diproduksi oleh STMicroelectronics. STM32F100RB berbasis teknologi ARM cortex-M3 dan memiliki sistem *clock* dengan kecepatan 32 MHz. Memori yang terdapat dalam mikrokontroler ini antara lain memori flash yang berukuran 128 Kbytes dan memori SRAM yang berukuran 8 Kbytes. Tegangan kerja dari STM32F100RB memiliki rentang nilai antara 2.0 sampai 3,6 volt. Namun, yang sering diaplikasikan adalah 3,3 volt. Adapun untuk sistem *clock*, mikrokontroler ini memiliki dapat disetting menggunakan *clock* eksternal dengan nilai 4 – 24 MHz, *clock* internal 8 MHz untuk *factory-trimmed RC*, internal 40 kHz RC, PLL untuk *clock* CPU, dan 32 kHz osilator yang terkalibrasi untuk RTC.

STM32F100RB memiliki periferal yang menunjang dalam pengoperasiannya. Beberapa di antaranya adalah IO, ADC, timer, dan USART.

2.5.1. Input-Output

Setiap pin GPIO pada STM32F100RB dapat dikonfigurasikan sebagai *input floating, input pull-up, input-pull-down, analog, output open-drain, output push-pull, alternate function push-pull,* dan *alternate function open-drain*. Masing-masing dari pin

GPIO tersebut dapat dikonfigurasikan melalui program dan tidak terpengaruh antara satu pin dengan pin lainnya.

Selama dan sesaat setelah reset, mode *alternate function* tidak aktif dan PORT I/O dikonfigurasikan sebagai *input floating* mode. Ketika dikonfigurasikan sebagai *output*, nilai yang tertulis pada *output* data register (GPIOx_ODR) adalah nilai *output* pada pin I/O. Masing-masing dari pin dapat dikonfigurasikan sebagai *output* mode *push-pull* atau pun *output* mode *open-drain*.

Input data register (GPIOx_IDR) membaca data yang terdapat pada pin I/O pada setiap detakan *clock* dari APB2. Semua pin memiliki resistor *pull-up* internal dan *pull-down* internal yang dapat diaktifkan ketika pin yang bersangkutan dikonfigurasikan sebagai *input*.

Kita perlu memprogram *Port Bit Configuration Register* sebelum menggunakan *default alternate function*. Untuk *alternate function input, port* yang bersangkutan harus dikonfigurasikan sebagai mode *input*, baik *floating*, *pull-up*, maupun *pull-down*. Untuk *alternate function output*, pin yang bersangkutan harus dikonfigurasikan sebagai *alternate function output*, baik *push-pull* maupun *open-drain*. Sedangkan untuk *alternate function bidirectional*, pin yang bersangkutan harus dikonfigurasikan sebagai *output* (*push-pull* atau *open-drain*). Pada kasus ini, *driver input* dikonfigurasikan sebagai mode *input floating*.

2.5.2. ADC

STM32F100RB memiliki periferal ADC dengan resolusi sebesar 12 bit. ADC ini memiliki 18 kanal yang terdiri atas 16 kanal untuk mengukur sinyal analog dari eksternal mikrokontroler dan 2 kanal untuk mengukur sinyal dari internal mikrokontroler. Proses konversi sinyal analog ke digital dari berbagai kanal tersebut dapat menggunakan mode *single, continuous, scan,* atau *discountinuous*. Hasil pembacaan nilai ADC disimpan di data register 16 bit.

Fitur-fitur utama yang terdapat pada ADC STM32F100RB antara lain :

- Resolusi 12 bit.
- Pembangkitan interupsi pada *end of conversion, end of injected conversion,* dan *watchdog event*.
- Mode konversi dapat berupa *single* atau *continuous*.
- Mode *scan* untuk konversi sinyal otomatis dari kanal 0 sampai kanal ‘n’.
- Kalibrasi sendiri.

- Waktu sampling yang bisa diatur pada tiap kanal.
- Opsi picu eksternal baik untuk konversi *regular* maupun *injected*.
- Mode *discontinous*.
- Waktu konversi ADC : 1,17 µs pada 24 MHz
- Persyaratan *supply* ADC : 2,4 V sampai 3,6 V.
- Pembangkitan DMA *request* selama mode konversi *regular*.

2.5.3. Timer

Timer pada STM32F100RB terdiri atas *counter* 16 bit yang terhubung dengan *prescaler* yang dapat diprogram. Timer dapat digunakan untuk berbagai tujuan, seperti mengukur lebar pulsa sinyal *input* (*input capture*) atau membangkitkan bentuk sinyal *output* (*output compare* dan PWM). Periode dari lebar pulsa *input* dan pulsa *output* dapat diatur dari beberapa mikro sekon hingga beberapa mili sekon dengan menggunakan *timer prescaler* dan *RCC clock controller prescaler*.

Fitur utama timer yang terdapat pada STM32F100RB antara lain :

- *Counter* 16 bit dengan mode perhitungan *up, down, up/down auto-reload*.
- *Prescaler* 16 bit yang dapat diprogram nilainya dari 1 hingga 65535.
- Setiap *timer* terdapat 4 kanal yang masing-masing dapat dikonfigurasi sebagai *input capture, output compare,* pembangkitan sinyal PWM, dan *One-pulse mode output*.
- *Complementary output* dengan *dead-time* yang dapat diprogram.
- Sinkronisasi sirkuit untuk mengontrol *timer* dengan sinyal eksternal dan untuk interkoneksi beberapa *timer* secara bersama-sama.
- *Repetition counter* untuk *update* register *timer* hanya setelah beberapa siklus *counter*.
- Pembangkitan interupsi / DMA dengan *event* sebagai berikut : *update, trigger event, input capture, output compare, break input*.

2.5.4. USART

USART (*universal synchronous asynchronous receiver transmitter*) digunakan untuk pertukaran data dengan *device* digital lain yang membutuhkan format data serial NRZ *asynchronous* yang standar industri. USART pada STM32F100RB memiliki pengaturan nilai *baud rate* yang dapat diatur bernilai berapa saja dengan menggunakan *fractional baud rate generator*. USART pada STM32F100RB juga dapat digunakan untuk mode *synchronous one-way communications* dan *half-duplex single wire communications*.

Fitur-fitur utama USART pada STM32F100RB antara lain :

- *Full duplex*, komunikasi sinkron
- Format standar NRZ
- Nilai *baud rate* dapat mencapai hingga 4,5 MBits/s
- Lebar data yang dapat diprogram (8 atau 9 data bit)
- Stop bit yang dapat dikonfigurasi (1 atau 2 stop bit)
- *Clock* transmiter *output* untuk komunikasi sinkron.
- Komunikasi *single wire half duplex*
- Bit *enable* yang terpisah antara *transmitter* dengan *receiver*
- Kontrol *parity*, yaitu *transmiter parit bit* dan *cheks parit of received data byte*
- Empat buah mekanisme deteksi eror komunikasi : *overrun error*, *noise error*, *frame error*, dan *parity error*.
- Sepuluh buah sumber pemicu interupsi USART, yaitu : CTS *changes, LIN break detection, transmit data register empty, transmission complete, Recife data register full, idle line received, overrun error, framing error, noise error,* dan *parity error*.

2.6.Board STM32 *Value Line Discovery*

STM32 *value line Discovery* merupakan sebuah *development board* yang menggunakan mikrokontroler STM32F100RB. *Develpment board* ini diproduksi oleh ST Microelectronics yang juga merupakan produsen dari seri mikrokontroler STM32. *Development board* STM32 *value line discovery* ditunjukkan dalam Gambar 2.9. Fitur-fitur yang dimiliki oleh *development board* ini antara lain :

- ST-Link yang terintegrasi pada *board* sebagai *device* yang bertugas untuk memasukkan kode program ke dalam mikrokontroler. ST-Link ini juga dapat digunakan untuk mikrokontroler lain di luar *development board*.
- Mengandung dua buah LED. LD_1 untuk komunikasi USB dan LD_2 untuk tanda catu daya 3,3 volt.
- Didesain untuk diberi catu tegangan dari USB, tegangan eksternal 5 volt, atau tegangan eksternal 3,3 volt.
- Dapat memberikan catu tegangan pada piranti elektronika lain dengan nilai 5 volt atau 3 volt.
- Terdapat dua buah LED untuk pengguna, LD_3 yang berwarna hijau dan LD_4 yang berwarna biru.
- Terdapat dua buah tombol, untuk pengguna (GPIO) dan untuk reset.

- Semua pin pada mikrokontroler dihubungkan dengan *header pin* sehingga mudah untuk digunakan.

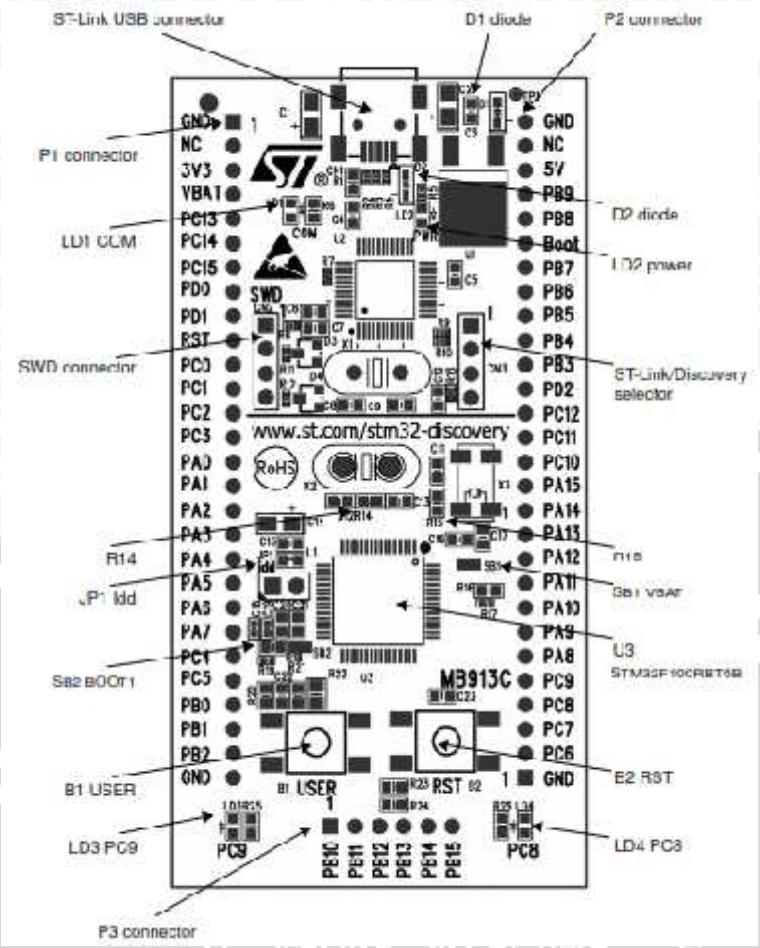
Gambar 2.9. Board STM32 value line Discovery

Sumber : STMicroelectronics (2011 : 1)

Catu daya pada *board* ini dapat diperoleh melalui konektor USB, catu tegangan 5 volt, atau 3,3 volt. Pin 5 volt dan pin 3,3 volt dapat digunakan secara terpisah sebagai *input* atau *output* catu daya karena terdapat dioda D_1 dan D_2 yang memproteksi rangkaian.

LED merah LD_1 yang berlabel COM menunjukkan komunikasi antara PC dengan ST-Link. LED merah LD_2 yang berlabel PWR menunjukkan bahwa *board* ini sedang diberi catu daya. LED hijau LD_3 terhubung dengan pin I/O PC9 pada mikrokontroler. sedangkan LED biru LD_4 terhubung dengan pin I/O PC8 pada mikrokontroler.

Tombol B_1 yang berwarna biru berlabel USER terhubung dengan pin I/O PA0 pada mikrokontroler. sedangkan tombol B2 yang berwarna hitam berlabel RST digunakan untuk mereset mikrokontroler. *Jumper* JP_1 yang berlabel I_{dd} digunakan sebagai fasilitas untuk mengukur nilai arus listrik yang dikonsumsi oleh mikrokontroler. Caranya adalah dengan mengambil *jumper* tersebut dan menggantinya dengan ammeter. Gambar 2.10 menunjukkan *layout* dari atas dari Board STM32 *value line Discovery.*



Gambar 2.10. *Layout* atas Board STM32 value line Discovery

(Sumber : STMicroelectronics, 2011 : 5)

2.7.*Bluetooth Module* HC-05

Modul HC-05 modul *bluetooth* SPP (*Serial Port Protocol*) yang didesain untuk komunikasi serial secara nirkabel. Modul Serial Port *bluetooth* ini menggunakan versi V2.0+EDR (*Enhanced Data Rate*) 3 Mbps modulasi dengan frekuensi kerja 2,4 GHz. Modul ini menggunakan CSR *Bluecore* 04-External *single chip bluetooth system* yang menggunakan teknologi CMOS dan AFH (*Adaptive Frequency Hopping Feature*). Bentuk fisik dari modul *bluetooth* HC-05 ditunjukkan dalam Gambar 2.11.

Gambar 2.11. Board Modul HC-05

(Sumber : ITead Studio, 2010 : 1)

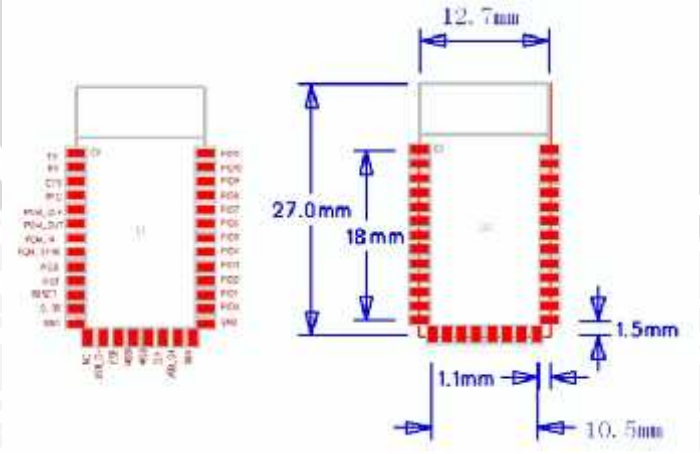
Spesifikasi *hardware* dari modul ini antara lain :

- Sensitivitas pada umumnya -80dBm
- Energi transmisi RF mencapai +4dBm
- Tegangan operasi yang rendah, dari 1,8 volt – 3,6 volt
- Kontrol PIO
- Antarmuka menggunakan UART dan *baudrate* yang dapat diganti nilainya
- Terdapat antena yang terintegrasi
- Terdapat *edge connector*

Spesifikasi *sofware* dari modul ini adalah sebagai berikut :

- *Default baud rate* : 38400, data bit: 8, *Stop* bit: 1, *Parity*: *No parity*, kontrol data: ada. *Baud rate* yang tersedia: 9600, 19200, 38400, 57600, 115200, 230400, 460800.
- Jika pulsa tepi naik diberikan pada PIO0, maka modul akan *disconnect*.
- Instruksi status *port* PIO1: logika LOW-*disconnected*, logika HIGH-*connected.*
- PIO10 dan PIO11 dapat dihubungkan dengan LED merah dan biru secara terpisah. Ketika *master* dan *slave* sedang *pair*, LED merah dan biru akan berkedip dengan periode 2 detik. Sedangkan ketika *disconnect*, hanya LED biru yang berkedip 2 kali per detik.
- *Auto-connect* pada *device* yang terkoneksi terakhir kali secara *default*
- Secara *default*, *auto-pairing* PINCODE:"0000".
- *Auto-reconnect* dalam waktu 30 detik ketika *disconnect* yang disebabkan oleh jarak yang melebihi jangkauan.

Adapun konfigurasi pin dan dimensi modul *bluetooth* HC-05 ditunjukkan dalam Gambar 2.12.



Gambar 2.12. Konfigurasi dan Dimensi Modul HC-05

(Sumber : ITead Studio, 2010 : 2)

2.8.Pemrograman Java

Java merupakan salah satu bahasa pemrograman komputer yang banyak digunakan akhir-akhir ini. Tidak seperti bahasa pemrograman yang lain yang memiliki pengaruh yang menurun tiap tahunnya, Java justru semakin berkembang seiring berjalannya waktu. Java melangkah ke depan dengan sebagai bahasa pemrograman internet pada rilis pertamanya. Setiap versi perbaikan dari Java selalu semakin memperkuat posisinya di mata ahli program (Schildt, 2007).

Java memiliki *library* yang sangat banyak, baik yang terintegrasi dengan JDK (*Java Development Kit*) maupun yang dikembangkan oleh ahli program. *Library* java yang banyak digunakan untuk pemrograman GUI (*Graphically User Interface*) adalah javax.swing. Library java yang banyak sering digunakan antara lain *Scanner* untuk input output, *String* untuk mengolah kalimat atau kata, *File* untuk mengolah *file*, dan lain sebagainya. Sedangkan *library* yang dikembangkan oleh *developer* dan sering digunakan antara lain *JFreeChart* untuk mengolah grafik dan *JSerialComm* untuk mengolah komunikasi serial melalui port USB.

BAB III
METODE PENELITIAN

Metode kajian yang digunakan dalam penelitian bersifat aplikatif mengenai bagaimana rancang bangun sistem yang dapat memvisualisasikan pembacaan sensor garis pada *line follower robot* pada komputer serta menyimpannya ke dalam suatu sistem data logger sehingga dapat divisualkan kembali di lain waktu. Agar penelitian dapat dilakukan seperti yang direncanakan, maka dibutuhkan beberapa langkah dalam pelaksanaannya. Langkah-langkah yang dibutuhkan yaitu penentuan spesifikasi alat, studi literatur, perancangan dan pembuatan alat (baik rangkaian elektrik maupun program), pengujian alat, serta pengambilan kesimpulan dan saran.

3.1 Penentuan Spesifikasi Alat

Sistem visualisasi pembacaan sensor garis pada *line follower robot* ini dirancang agar memiliki spesifikasi tertentu sehingga dapat diaplikasikan dengan mudah pada penerapan sesungguhnya di lapangan. Spesifikasi ini juga menjadi acuan dalam perancangan selanjutnya. Adapun spesifikasi sistem visualisasi yang direncanakan adalah sebagai berikut :

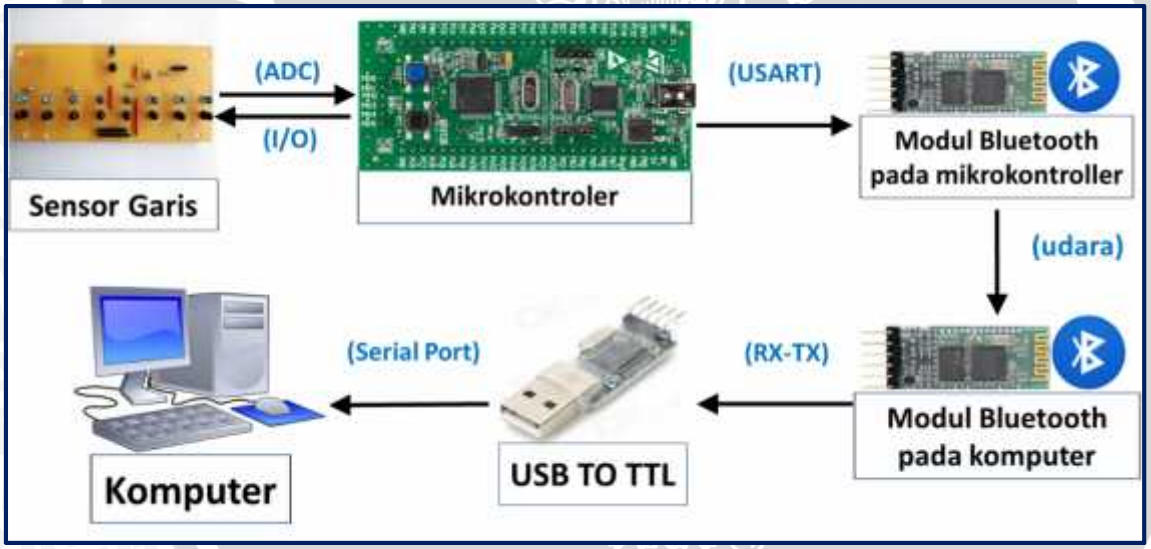
1. Terdiri atas 12 unit sensor cahaya.
2. Jarak maksimum sensor garis dengan lintasan sebesar 10 cm.
3. Menghasilkan beda tegangan sebesar 2 volt ketika diletakkan di atas permukaan hitam dan putih.
4. Memiliki akurasi sebesar 90%.
5. Memiliki rangkaian antarmuka yang menjembatani *device* yang beroperasi pada tegangan 5 V dan 3,3 V.
6. Jarak transmisi data mencapai 5 meter.
7. Menggunakan mikrokontroler dengan arsitektur 32 bit.
8. Memiliki mode kalibrasi untuk penentuan nilai *treshold* hitam dan putih tiap unit sensor.
9. Memiliki 4 buah macam tampilan visual, yaitu *virtual port*, grafik, diagram batang, dan animasi sensor.
10. Memiliki sistem penyimpanan data (*data logger*) yang dapat divisualisasikan kembali.

3.2 Perancangan Alat

Dalam perealisasian alat, dibutuhkan langkah-langkah teknis agar tujuan dapat tercapai. Langkah-langkah tersebut antara lain pembuatan perangkat keras, pembuatan perangkat lunak pada mikrokontroler, dan pembuatan perangkat lunak pada komputer.

3.2.1 Blok Diagram

Tujuan penelitian sistem visualisasi pembacaan sensor garis pada *line follower robot* ini adalah untuk menghasilkan sebuah sistem yang dapat membaca nilai ADC tiap unit rangkaian sensor cahaya TCRT5000, mengolahnya menjadi sebuah paket data, mengirimkannya dari mikrokontroler ke komputer, memecah kembali paket data menjadi data tiap unit sensor cahaya TCRT5000, dan memvisualisasikannya pada layar monitor komputer, serta menjalakan sistem data logger sehingga hasil pembacaan dapat disimpan pada memori komputer dan ditampilkan kembali di lain waktu. Blok diagram sistem visualisasi pembacaan sensor garis pada *line follower robot* ditunjukkan dalam Gambar 3.1.



Gambar 3.1. Blok Diagram Sistem Visualisasi Pembacaan Sensor Garis

Posisi *line follower robot* relatif terhadap garis dibaca oleh sensor garis. Dalam perancangan ini, sensor garis terdiri atas 12 unit rangkaian sensor cahaya TCRT5000. Tiap unit rangkaian sensor cahaya TCRT5000 menghasilkan suatu nilai tegangan tertentu yang merepresentasikan banyaknya pantulan cahaya yang diterima oleh bagian *receiver* sensor cahaya TCRT5000. Oleh karena itu, terdapat 12 buah data analog yang berasal dari masing-masing unit rangkaian sensor cahaya TCRT5000.

Kedua belas data analog yang berasal dari sensor garis ini dibaca oleh pin ADC mikrokontroler. Sebelum membaca nilai ADC, mikrokontroler terlebih dahulu mengatur pin selektor pada IC multiplekser pada sensor garis untuk memilih sensor mana yang akan

dibaca nilai ADCnya. Pin selektor pada IC multiplekser dikendalikan oleh pin I/O mikrokontroler.

Setelah kedua belas data analog dari masing-masing unit rangkaian sensor cahaya TCRT5000 dibaca oleh mikrokontroler, data ini diolah dan dibentuk menjadi sebuah paket data dalam bentuk string. String ini kemudian dikirimkan oleh mikrokontroler ke modul *bluetooth* HC-05 yang terhubung dengannya melalui komunikasi serial USART melalui pin TX. Paket data ini kemudian dipancarkan oleh modul *bluetooth* ke udara dan diterima oleh modul *bluetooth* HC-05 lain yang terpasang pada komputer. Kedua *bluetooth* – yang terhubung dengan mikrokontroler dan yang terhubung dengan komputer – telah diset *pairing* sebelumnya, sehingga transmisi data dapat terjadi pada kedua modul tersebut.

Paket data yang diterima oleh modul *bluetooth* yang lain kemudian dikirimkan ke komputer melalui modul USB to TTL. Modul ini dibutuhkan karena level data keluaran dari modul *bluetooth* HC-05 adalah TTL sedangkan pada komputer adalah USB. Setelah sampai pada komputer, paket data dipecah lagi menjadi unit-unit data tiap sensor cahaya TCRT5000. Data-data ini kemudian divisualisasikan pada layar monitor komputer dengan beberapa mode visualisasi menggunakan *software* yang telah dibuat sebelumnya. Selain ditampilkan, data-data ini dapat disimpan menjadi sebuah *file datalogger* selama proses visualisasi. Di lain waktu, *file data logger* ini dapat divisualisasikan kembali pada monitor komputer untuk mempermudah jika dibutuhkan untuk analisis lebih lanjut.

3.2.2 Perancangan Perangkat Keras

Perancangan perangkat keras sistem visualisasi pembacaan sensor garis pada *line follower robot* ini terbagi menjadi beberapa bagian, antara lain:

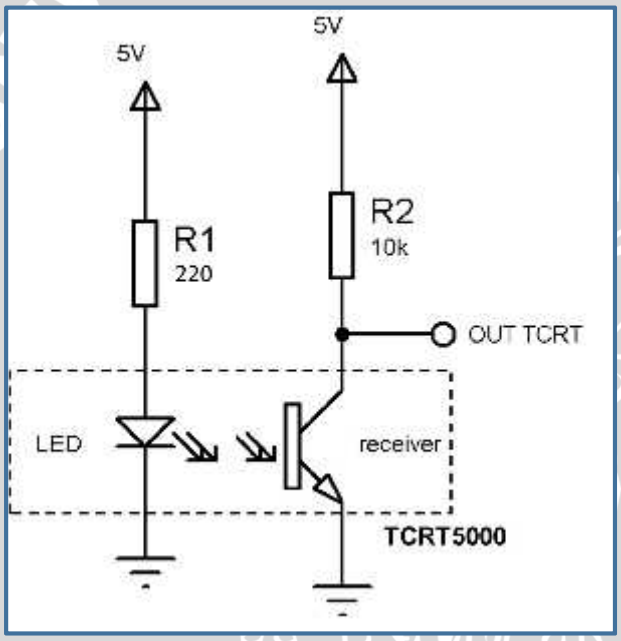
1. Perancangan Rangkaian Sensor cahaya TCRT5000 dan Multiplekser
2. Perancangan Rangkaian *Bi-directional Logic Level Shifter*
3. Perancangan *Input* Tombol dan LED Indikator untuk Kalibrasi
4. Perancangan Kontrol Sistem Mikrokontroler STM32F100RB
5. Perancangan Antarmuka Komputer dengan USB to TTL dan *Bluetooth Module* HC-05

- **Perancangan Rangkaian Sensor cahaya TCRT5000 dan Multiplekser**

Komponen utama dari sensor garis *line follower robot* adalah sensor yang memiliki kepekaan terhadap perubahan intensitas cahaya di lingkungan sekitarnya dan memiliki respons yang cepat. Sensor yang umum digunakan untuk keperluan ini adalah fotodiode dan fototransistor. Pada penelitian ini, komponen yang digunakan adalah fototransistor dengan jenis TCRT5000. Pertimbangannya adalah sensor ini terdiri atas sepasang

pemancar dan penerima. Pemancar berupa LED yang memancarkan gelombang inframerah dengan panjang gelombang 950 nm dan penerima berupa fototransistor yang hanya peka terhadap gelombang yang dipancarkan oleh pemancarnya. Pertimbangan lain adalah penerima pada TCRT5000 memblokir cahaya tampak sehingga relatif lebih aman terdapat gangguan dari pancaran cahaya lain (Vishay Semiconductor, 2009).

Fototransistor pada TCRT5000 bersifat konduktif ketika banyak cahaya yang mengenainya dan bersifat isolatif ketika sedikit cahaya yang mengenainya. Untuk dapat mendeteksi hal ini, dibutuhkan suatu rangkaian tambahan berupa rangkaian pembagi tegangan sehingga keluaran dari rangkaian ini berupa suatu nilai tegangan analog yang bervariasi terhadap banyaknya cahaya yang diterima oleh fototransistor. Rangkaian yang digunakan pada penelitian ini adalah rangkaian pembagi tegangan. Rangkaian ini ditunjukkan dalam Gambar 3.2.



Gambar 3.2. Rangkaian Sensor cahaya TCRT5000

Nilai maksimum arus yang lewat pada *transmitter* (LED) TCRT5000 adalah 60 mA, sedangkan tegangan jatuhnya adalah 1,25 volt. Pada penelitian ini, sebuah resistor dirangkai seri dengan IR LED (R_1) sehingga arus yang mengalir pada pada komponen ini adalah sebesar sepertiga kali arus maksimalnya, yaitu 20 mA. Hal ini dimaksudkan untuk keamanan dan konsumsi daya. Oleh karena itu, perhitungan nilai resistor yang dirangkai seri dengan *transmitter* (LED) adalah sebagai berikut:

$$\frac{V_{cc} - V_F}{R_1} = I_F \quad \ldots(4.1)$$

$$\frac{5\ volt - 1{,}25\ volt}{R_1} = 20\ mA$$

$$R_1 \geq \frac{3{,}75\ volt}{20\ mA}$$

$$R_1 \geq 187{,}5\ \Omega$$

Nilai resistor yang dirangkai seri dengan IR LED (R_1) adalah nilai resistor yang nilainya mendekati hasil perhitungan, lebih besar, dan tersedia di pasaran. Oleh karena itu, nilai resistor ini adalah 220 Ω.

Penentuan resistor yang dirangkai seri dengan kaki kolektor fototransistor dihitung ketika dalam keadaan saturasi, yaitu ketika sensor sedang berada di atas permukaan putih. Berdasarkan *datasheet* TCRT5000, bagian *receiver* memiliki tegangan saturasi V_{CE} sebesar 0,4 V. Sedangkan arus yang I_C pada saat saturasi adalah 0,5 mA. Oleh karena itu, nilai resistor yang dibutuhkan agar bisa dirangkai seri dengan fototransistor TCRT5000 dihitung sebagai berikut :

$$V_{CC} = V_{CE(saturasi)} + I_C \times R_2$$

$$R_2 = \frac{V_{CC} - V_{CE(saturasi)}}{I_C} = \frac{5\ V - 0{,}4\ V}{0{,}5\ mA} = 9{,}2\ k\Omega$$

Nilai resistor yang digunakan adalah yang tersedia di pasar, nilainya mendekati hasil perhitungan, dan nilainya lebih besar dari hasil perhitungan. Oleh karena itu, nilai R_2 yang digunakan adalah 10 kΩ.

Berdasarkan perhitungan tersebut, tegangan keluaran sensor pada node OUT TCRT dalam Gambar 3.2 pada saat sensor berada di atas permukaan putih sama dengan tegangan V_{CE} fototransistor saat saturasi, yaitu 0,4 V. Kuat arus yang masuk pada kolektor (I_C) pada saat sensor berada di atas permukaan hitam adalah 0,24 mA. Nilai ini didapat dari pengukuran. Oleh karena itu, tegangan saat berada di atas permukaan hitam adalah sebagai berikut :

$$V_{OUT\,TCRT} = V_{CC} - I_{C(permukaan\ hitam)} \times R_2$$

$$V_{OUT\,TCRT} = 5\ V - 0{,}24\ mA \times 10\ k\Omega$$

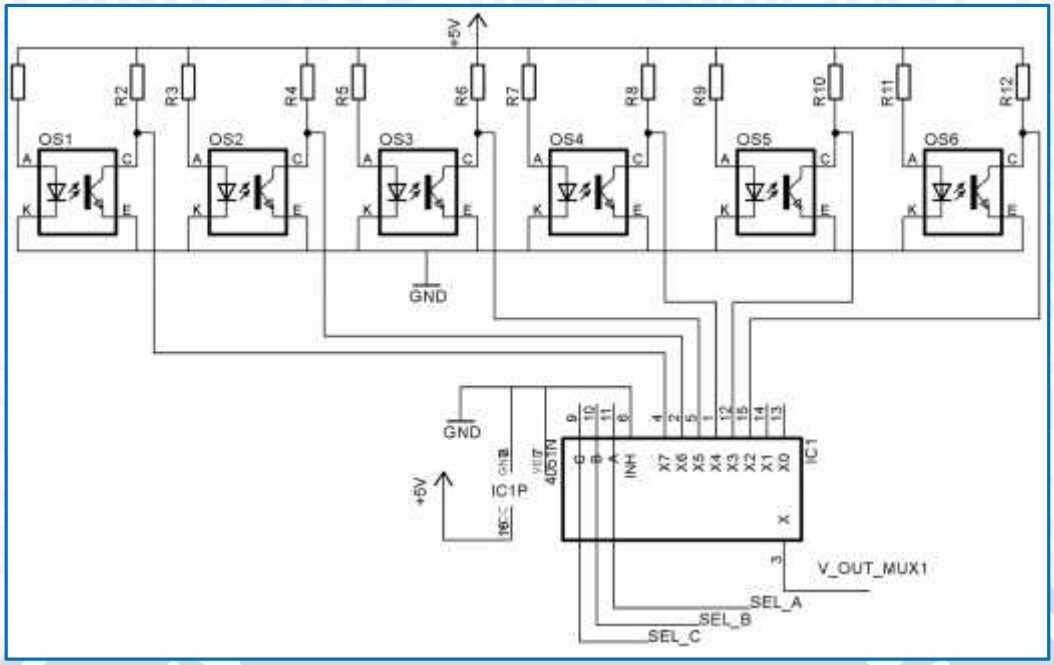
$$V_{OUT\,TCRT} = 5\ V - 2{,}4\ V$$

$$V_{OUT\,TCRT} = 2{,}6\ V$$

Pada penelitian di skripsi ini, sensor cahaya TCRT5000 yang digunakan berjumlah 12 unit. Hal ini berarti terdapat 12 buah data tegangan analog yang keluar dari rangkaian sensor garis dan kesemuanya harus dibaca oleh mikrokontroler. Untuk menghemat penggunaan pin, maka digunakanlah IC multiplekser analog untuk men-switch sinyal tegangan keluaran rangkaian sensor garis. Dalam perancangan ini, IC multiplekser yang digunakan adalah 4051. IC ini memiliki 8 kanal yang dapat dihubungkan dengan sebuah pin *common* secara bergantian. Kanal yang terhubung dengan pin *common* ini dapat diatur dengan memberikan logika tertentu pada tiga buah pin selektor A, B, dan C. Dalam satu waktu, hanya ada satu kanal dari kedelapan kanal yang dapat terhubung dengan pin *common*. Kedelapan pin kanal dan pin *common* dapat bertindak sebagai *input* maupun *output*. Pada IC 4051 terdapat pin *enable* yang bersifat *active LOW*. Pada perancangan ini, pin *enable* ini selalu dihubungkan dengan *ground* agar IC multiplekser selalu aktif.

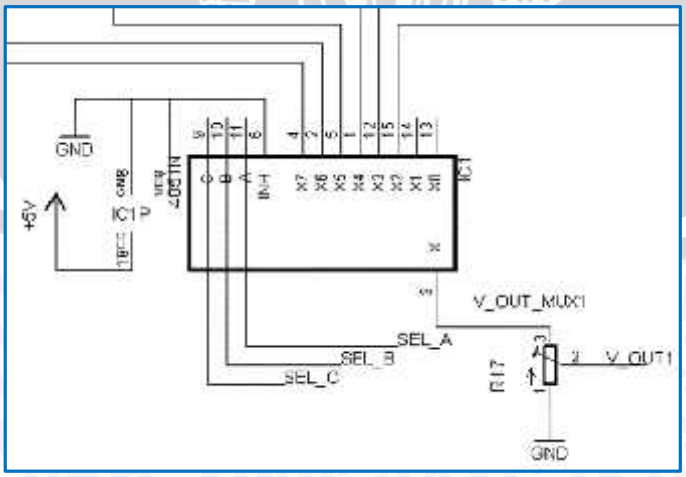
IC multiplekser yang digunakan pada perancangan ini adalah yang menggunakan teknologi CMOS. Beberapa pertimbangan yang menjadi dasar pemilihan ini antara lain : 1) pada proses pembacaan sensor, mekanisme *switching* enam kanal terhadap sebuah pin *common* dilakukan dengan cepat – IC CMOS unggul dari segi kecepatan dibandingkan IC TTL. 2) IC CMOS memiliki daya disipasi yang lebih kecil dari pada IC. TTL 3) di pasaran, hampir semua IC multiplekser yang dijual menggunakan teknologi CMOS.

Oleh karena jumlah tegangan keluaran berjumlah 12 buah, maka dalam penelitian ini digunakan dua buah IC 4051 yang mana masing-masing terhubung dengan enam unit sensor cahaya TCRT5000 melalui pin kanal. Dari kedelapan kanal ($X_0 – X_7$), yang digunakan adalah kanal X_2 sampai kanal X_7. Pada IC Multiplekser sebelah kiri, rangkaian sensor dengan indeks 0 (paling kiri) terkoneksi dengan kanal X_2 pada IC multiplekser. rangkaian sensor dengan indeks 1 terkoneksi dengan kanal X_3 pada IC multiplekser. Begitu seterusnya hingga sensor dengan indeks 5 terkoneksi dengan kanal X_8. Adapun pada IC multiplekser sebelah kanan, susunan koneksinya dibalik dibandingkan dengan yang sebelah kiri. Pin *common* dari kedua IC multiplekser ini dihubungkan dengan pin ADC pada mikrokontroler melalui rangkaian pengondisi sinyal untuk menurunkan tegangannya. Adapun pin selektor kedua IC ini dihubungkan dengan pin *input output* mikrokontroler. Rangkaian sensor garis dengan multiplekser ditunjukkan dalam Gambar 3.3.



Gambar 3.3. Tegangan Analog Keluaran Rangkaian Sensor cahaya TCRT5000 Di-switch Menggunakan IC Multiplekser Analog

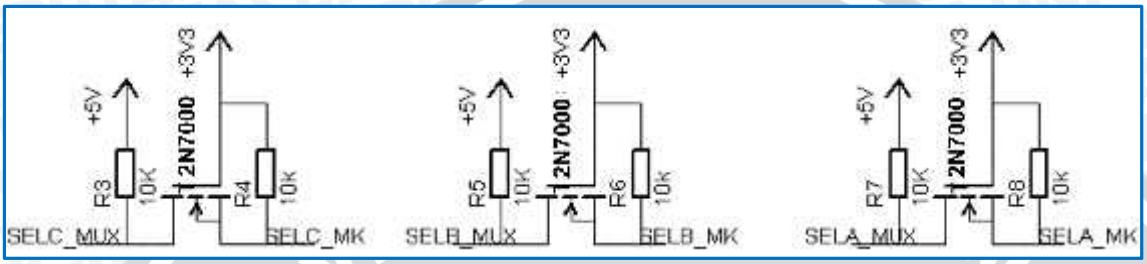
Keluaran rangkaian sensor cahaya TCRT5000 memiliki nilai keluaran maksimum hingga mencapai 5 volt jika bagian fototransistor benar-benar bersifat isolatif. Mikrokontroler ARM STM32F100RB bekerja pada tegangan 3,3 volt. Oleh karena itu, rentang tegangan yang berada di atas nilai 3,3 volt yang masuk ke pin ADC tidak akan terbaca oleh mikrokontroler. Dibutuhkan suatu rangkaian yang dapat menurunkan rentang tegangan 0 – 5 volt menjadi 0 – 3,3 volt secara linier. Sebuah resistor variabel yang difungsikan sebagai pembagi tegangan dapat digunakan untuk menjalankan fungsi ini. Resistor variabel ini diatur sehingga menghasilkan tegangan keluaran sebesar 3,3/5 = 0,66 kali tegangan masukannya. Rangkaian dengan tambahan rangkaian resistor variabel ini ditunjukkan dalam Gambar 3.4.



Gambar 3.4. Tegangan *Output* Sensor cahaya TCRT5000 Diturunkan Menggunakan Resistor Variabel

- **Perancangan Rangkaian *Bi-directional Logic Level Shifter***

Telah diketahui bahwa pin selektor pada IC multiplekser dihubungkan dengan pin I/O mikrokontroler. IC multiplekser yang digunakan pada penelitian ini berbasis teknologi MOS, yang mana nilai V_{IH} sebesar 3,5 volt. Sedangkan mikrokontroler ARM STM32 beroperasi pada tegangan 3,3 volt. Oleh karena itu, dibutuhkan suatu rangkaian *logic level shifter* yang dapat mengubah level tegangan 3,3 volt menjadi 5 volt. Rangkaian *logic level shifter* ditunjukkan dalam Gambar 3.5.



Gambar 3.5. Rangkaian *Logic Level Shifter* 3,3 volt menjadi 5 volt

Prinsip kerja dari rangkaian *logic level shifter* berdasarkan dapat dijelaskan sebagai berikut : ketika pin IO mikrokontroler berlogika HIGH, maka nilai V_{GS} akan rendah dan nilainya lebih kecil dibandingkan tegangan *treshold* sehingga FET akan bersifat isolatif. Sedangkan ketika FET berlogika LOW, maka nilai V_{GS} akan tinggi dan nilainya lebih besar dibandingkan dengan tegangan *treshold* sehingga FET akan bersifat bersifat konduktif. Oleh karena itu, dibutuhkan jenis FET memiliki nilai *treshold* di antara kedua kasus tersebut.

Berdasarkan *data sheet* STM32F100RB, mikrokontroler ini memiliki nilai V_{OL} sebesar 0,4 V dan memiliki nilai V_{OH} sebesar V_{DD}-0,4 V. Dalam rangkaian modul STM32 Value Line Discovery, V_{DD} yang digunakan adalah 3,3 V sehingga STM32F100RB memiliki nilai V_{OH} sebesar 2,9 V. Saat rangkaian *logic level shifter* berlogika LOW, nilai V_{GS} MOSFET adalah :

$$V_{GS}\ (LOW) = 3{,}3\ V - V_{OL} = 3{,}3\ V - 0{,}4\ V = 2{,}9\ V$$

Sedangkan saat berlogika HIGH, nilai V_{GS} MOSFET adalah :

$$V_{GS}\ (HIGH) = 3{,}3\ V - V_{OH} = 3{,}3\ V - 2{,}9\ V = 0{,}4\ V$$

Oleh karena itu, dibutuhkan suatu jenis MOSFET yang memiliki nilai tegangan *treshold* di antara 0,4 V dan 2,9 V.

Frekuensi ADC maksimum yang terdapat dalam STM32F100RB adalah sebesar 12 MHz. Dalam perancangan ini, yang digunakan prescaler ADC sebesar 4. Oleh karena itu, nilai frekuensi ADC dalam perancangan adalah sebesar :

$$f_{ADC} = \frac{f_{ADC_{MAX}}}{Prescaler} = \frac{12MHz}{4} = 3\ MHz$$

Cycle yang digunakan dalam perancangan ini adalah sebesar 13,5 *cycle*. Oleh karena itu, lama konversi ADC adalah sebagai berikut :

$$T_{ADC\_cycle} = \frac{13{,}5\ cycle}{3\ MHz} = 4{,}5\ \mu s$$

Oleh karena itu, dibutuhkan jenis MOSFET yang memiliki nilai *turn-on time* dan *turn-off time* lebih kecil dari pada 4,5 µs. Jenis MOFTET yang sesuai dengan untuk dua persyaratan tersebut adalah 2N7000 yang memiliki tegangan *treshold* sebesar 2,1 V dan *turn-on time* dan *turn-off time* sebesar 20 ns.

Resistor *pull-up* R_4 yang digunakan adalah 10 kΩ. Arus listrik terbesar yang dapat mengalir pada resistor ini adalah ketika pin I/O mikrokontroler memiliki tegangan sebesar 0 volt. pada kondisi seperti ini, arus yang mengalir pada resistor dan kemudian masuk pada pin I/O mikrokontroler dihitung sebagai berikut :

$$I_{R4} = \frac{3{,}3\ V - teganganPinIO}{R4} = \frac{3{,}3\ V - 0\ V}{10\ k\Omega} = 0{,}33\ mA$$

Berdasarkan *data sheet*, arus maksimum pin I/O dalam menerima *sink current* adalah 8 mA. Oleh karena itu, resistor 10 kΩ dapat dijadikan sebagai resistor *pull-up* pin *output* mikrokontroler STM32F100RB.

Resistor *pull-up* R_3 yang digunakan adalah 10 kΩ. Berdasarkan *datasheet*, arus maksimum pada pin selektor multiplekser CD4051 adalah 0,1 µA. Maka, tegangan pada pin selektor multiplekser ketika resistor *pull-up* yang digunakan bernilai 10 kΩ adalah :

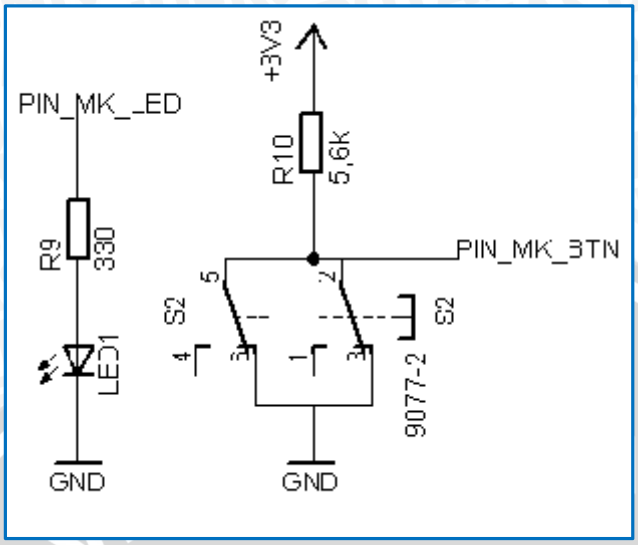
$$V(PIN\_SELEKTOR) = 5\ V - I_{in}(maks) \times R_3$$
$$V(PIN\_MK\_BTN) = 5\ V - 0{,}1\ \mu A \times 10\ k\Omega$$
$$V(PIN\_MK\_BTN) = 5\ V - 1\ mV$$
$$V(PIN\_MK\_BTN) = 4{,}999\ volt$$

Pin selektor IC CD4051 memiliki V_{IH} sebesar 3,5 volt. oleh karena tegangan pada pin selektor IC CD4051 memiliki nilai yang lebih besar dari pada nilai V_{IH}, maka resistor 10 kΩ bisa digunakan sebagai resistor *pull-up*.

- **Perancangan *Input* Tombol dan LED Indikator untuk Kalibrasi**

Rangkaian tombol pada penelitian di skripsi ini digunakan untuk mengatur mikrokontroler agar berpindah dari mode kalibrasi ke mode pembacaan dan pengiriman data. Lebih detail mengenai mode-mode ini akan dipaparkan dalam perancangan perangkat lunak pada mikrokontroler. Tombol dihubungkan dengan pin I/O pada mikrokontroler sebagai *input* data digital. Selain tombol, terdapat juga LED indikator sebagai penunjuk

apakah mikrokontroler sedang dalam mode kalibrasi atau mode pembacaan dan pengiriman data. Rangkaian skematik *input* tombol dan LED indikator ditunjukkan dalam Gambar 3.6.



Gambar 3.6. Rangkaian *Input* Tombol dan LED Indikator untuk Kalibrasi

LED yang digunakan dalam perancangan ini adalah LED 3mm warna hijau. Berdasarkan *datasheet*, LED ini memiliki arus *typical* sebesar 20 mA dan drop tegangan sebesar 2,2 volt. Oleh karena itu, nilai resistansi minimal yang dibutuhkan untuk diseri dengan LED ini dapat dihitung sebagai berikut :

$$\frac{V_{OH} - V_F}{R_1} \leq I_{F_{MAKS}} \quad \ldots (4.2)$$

$$\frac{3{,}3\ volt - 2{,}2\ volt}{R_9} \leq 20\ mA$$

$$R_9 \geq \frac{1{,}1\ volt}{20\ mA}$$

$$R_9 \geq 55\ \Omega$$

Dengan pertimbangan keamanan dan konsumsi daya, maka pada perancangan ini digunakan resistor dengan nilai 330 Ω sehingga arus yang mengalir pada LED sebesar 3,3 mA.

Pin I/O mikrokontroler STM32F100RB memiliki *input leakage current* sebesar 1 µA. Resistor *pull up* yang digunakan harus memiliki nilai resistansi sedemikian rupa sehingga drop tegangan ketika tombol tidak ditekan, tidak menyebabkan tegangan masukan pada PIN_MK_BTN di bawah nilai tegangan V_{IH} mikrokontroler. pada perancangan ini, yang digunakan adalah 5,6 kΩ. Dengan demikian, nilai tegangan pada PIN_MK_BTN adalah :

$$V(PIN\_MK\_BTN) = 3{,}3\ volt - I_{leakage} \times R_{10} \quad \ldots (4.3)$$

$$V(PIN\_MK\_BTN) = 3{,}3\ volt - 1\ \mu A \times 5{,}6\ k\Omega$$

$$V(PIN\_MK\_BTN) = 3{,}3\ volt - 1\ \mu A \times 5{,}6\ k\Omega$$

$$V(PIN\_MK\_BTN) = 3{,}2944\ volt$$

Mikrokontroler STM32F100RB memiliki nilai V_{IH} sebagai berikut :

$$V_{IH} = 0{,}41 \times (V_{DD} - 2) + 1{,}3 \quad \ldots (4.4)$$

$$V_{IH} = 0{,}41 \times (3{,}3 - 2) + 1{,}3 = 1{,}833\ volt$$

Terlihat bahwa nilai tegangan pada PIN_MK_BTN lebih besar dari pada nilai V_{IH}, sehingga nilai resistor sebesar 5,6 kΩ bisa digunakan untuk *pull up*.

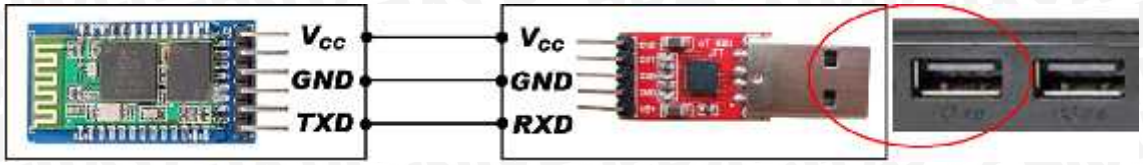
- **Perancangan Kontrol Sistem Mikrokontroler STM32F100RB**

Pada perancangan kontrol sistem perangkat keras digunakan *development board* STM32 VALUELINE DISCOVERY yang menggunakan mikrokontroler STM32F100RB. Mikrokontroler ini memiliki beberapa *peripheral* yang dibutuhkan untuk perancangan sistem visualisasi pembacaan sensor garis ini. Periferal yang dipakai antara lain I/O, ADC, dan USART. Alokasi pin yang digunakan adalah sebagai berikut :

- Pin A3 (GPIOA3) dihubungkan dengan pin selektor C pada IC multiplekser 4051
- Pin A4 (GPIOA4) dihubungkan dengan pin selektor B pada IC multiplekser 4051
- Pin A5 (GPIOA5) dihubungkan dengan pin selektor A pada IC multiplekser 4051
- Pin C1 (ADC1 IN11) dihubungkan dengan pin “common” IC multiplekser yang men*switch* separuh sensor sebelah kanan
- Pin C2 (ADC1 IN12) dihubungkan dengan pin “common” IC multiplekser yang men*switch* separuh sensor sebelah kiri
- Pin A9 (USART1 TX) dihubungkan dengan pin RX pada modul *bluetooth* HC-05
- Pin B14 (GPIOB14) dihubungkan dengan input tombol untuk sistem kalibrasi
- Pin C5 (GPIOC5) dihubungkan dengan LED indikator untuk sistem kalibrasi

- **Perancangan Antarmuka Komputer dengan USB to TTL dan *Bluetooth Module* HC-05**

Data yang ditransmisikan oleh modul *bluetooth* HC-05 pada mikrokontroler diterima oleh modul *bluetooth* yang lain yang akan mengirimkan data tersebut pada komputer. Modul *bluetooth* ini memiliki antar muka TTL dalam komunikasi serial, sedangkan komputer menggunakan USB dalam komunikasi serial. Oleh karena itu, dibutuhkan modul USB to TTL untuk menjembatani antara modul *bluetooth* HC-05 dengan USB komputer. Perancangan antar muka ini ditunjukkan dalam Gambar 3.7.

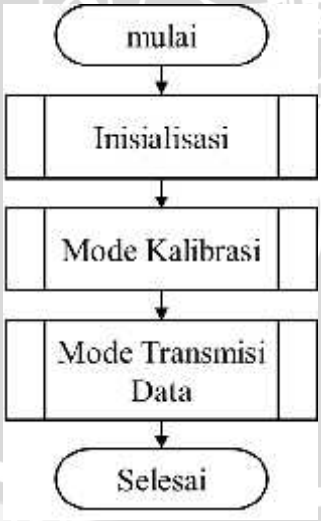


Gambar 3.7. Perancangan Antarmuka Modul *Bluetooth* HC-05 dengan Modul USB to TTL dan USB Komputer

Pin V_{CC} pada modul *bluetooth* HC-05 dihubungkan dengan pin V_{CC} pada modul USB to TTL. Keduanya mendapat catu daya dari USB komputer. Begitu pula dengan *ground*. Pin GND pada modul *bluetooth* HC-05 dihubungkan dengan pin V_{CC} pada modul USB to TTL. Kedua terhubung dengan GND USB komputer. Pin TXD (*transmitter*) pada modul *bluetooth* HC-05 dihubungkan dengan pin RXD (*receiver*) pada modul USB to TTL. Adapun pin-pin lain dapat dibiarkan dalam keadaan tidak terhubung karena tidak difungsikan dalam penelitian ini.

3.2.3 Perancangan Perangkat Lunak pada Mikrokontroler

Agar mikrokontroler dapat menjalankan tugasnya dalam sistem visualisasi sensor ini, maka sebuah algoritma pemrograman harus diterapkan pada mikrokontroler menggunakan sebuah bahasa pemrograman. Pada penelitian ini, digunakan bahasa C sebagai bahasa pemrograman dan *software* CooCox CoIDE sebagai IDEnya. Gambar 3.8 merupakan *flowchart* program pada mikrokontroler dalam menjalankan tugasnya.



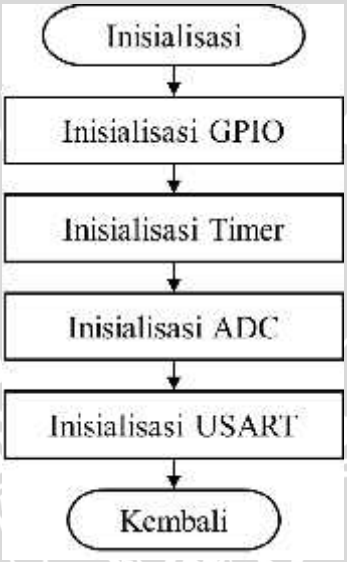
Gambar 3.8 *Flowchart* Program Mikrokontroler Keseluruhan

Saat pertama kali sistem diaktifkan, yang pertama dilakukan mikrokontroler adalah melakukan inisialisasi periferal-periferal yang dibutuhkan. Kemudian, tahap mode kalibrasi dijalankan untuk mengupdate nilai *treshold* tiap rangkaian sensor cahaya TCRT5000. Tahap terakhir adalah tahap mode transmisi data. Dalam tahap ini,

mikrokontroler secara kontinu mengirim hasil pembacaan sensor garis pada komputer melalui komunikasi nirkabel.

- **Tahap Inisialisasi**

Mikrokontroler STM32F100RB memiliki beberapa periferal yang digunakan dalam penelitian ini. Periferal yang dibutuhkan antara lain GPIO (*General Purpose Input Output*), *Timer*, ADC (*Analog to Digital Converter*), dan USART (*Universal Synchronous Asynchronous Receiver Transmitter*). Gambar 3.9 merupakan *flowchart* tahapan inisialisasi periferal pada program mikrokontroler.



Gambar 3.9 *Flowchart* Inisialisasi Periferal pada Mikrokontroler

GPIO digunakan untuk mengatur pin selektor pada IC Multiplekser 4051 dan untuk *input* tombol dan *output* LED indikator pada rangkaian kalibrasi. Terdapat rangkaian *logic translator* yang menghubungkan antara pin selektor IC multiplekser 4051 dengan pin I/O mikrokontroler sehingga *output* menggunakan mode *push-pull*. Begitu pula untuk LED indikator, mode *output* yang digunakan adalah *push-pull*. Adapun mode *input* yang digunakan adalah *input floating* karena sudah terdapat resistor *pull-up* pada rangkaian elektrik.

Software CooCox CoIDE tidak memiliki fungsi delay untuk menghentikan aliran program mikrokontroler untuk sementara waktu. Padahal pada penelitian ini, fungsi delay dibutuhkan untuk mengantisipasi transisi perpindahan *state* tombol agar tidak terjadi kesalahan pembacaan logika pada tombol. Oleh karena itu, sebuah timer digunakan untuk membuat fungsi delay ini. Timer yang digunakan pada penelitian ini adalah TIM2 dan diatur agar memiliki periode sebesar 1 us.

ADC dalam penelitian ini digunakan untuk membaca nilai tegangan keluaran dari rangkaian sensor cahaya TCRT5000. Mikrokontroler STM32F100RB hanya memiliki satu buah modul ADC di dalam *chip*, yaitu ADC1. Pengaturan ADC dalam program mikrokontroler di perancangan ini antara lain sebagai berikut : mode *scan* tidak diaktifkan, menggunakan mode *single conversion*, konversi ADC oleh *trigger* eksternal tidak diaktifkan, *data align* kanan, dan resolusi sebesar 12 bit.

USART digunakan untuk mengirim *string* yang berisi data kedua belas nilai ADC pada modul *bluetooth* HC-05. Pengaturan USART pada perancangan ini adalah sebagai berikut : data bit sebesar 8 bit, 1 stop bit, tanpa *parity*, dan USART yang digunakan adalah USART1.

Pemilihan BaudRate

Pada sistem visualisasi sensor garis ini, terdapat tiga komponen sistem yang berhubungan dengan baud rate : mikrokontroler, modul *bluetooth* HC-05, dan komputer. Komputer memiliki frekuensi kerja prosesor yang relatif tinggi dibanding mikrokontroler dan modul *bluetooth* HC-05. Pada modul *bluetooth* hanya dapat menggunakan baudrate tertentu yang tertera pada *datasheet*. *Baudrate* ini adalah 9600, 19200, 38400, 57600, 115200, 230400, dan 460800. Mikrokontroler STM32F100RB dengan frekuensi *clock* 24 MHz dapat diatur agar dapat memiliki nilai *baudrate* berapa saja dengan maksimal nilai 1,5 MHz. Akan tetapi tidak semua nilai baudrate dapat memiliki error sebesar nol persen. Eror ini disebabkan karena nilai USARTDIV yang seharusnya tidak bisa dipenuhi oleh register USARTDIV pada mikrokontroler. pada STM32F100RB, nilai yang dapat diberikan pada register USARTDIV terbatas pada kelipatan 0,0625 dan harus bernilai lebih dari 1. Pada perancangan ini, dicari nilai baudrate terbesar yang dapat diterapkan pada modul *bluetooth* HC-05 dan memiliki nilai eror sebesar nol persen pada mikrokontroler.

Eror *baudrate* pada mikrontroler dapat dihitung dengan menggunakan persamaan sebagai berikut :

$$Error[\%] = \left(\frac{BaudRate_{pada\ USART}}{BaudRate_{seharusnya}} - 1\right) \times 100\% \quad \ldots(4.5)$$

Sedangkan nilai register USARTDIV pada STM32F100RB dihitung dengan menggunakan persamaan sebagai berikut :

$$USARTDIV = \left(\frac{f_{ck}}{16\times\ baudrate}\right) \quad \ldots(4.6)$$

Berikut ini adalah perhitungan eror baudrate pada mikrokontroler menggunakan baudrate yang tersedia pada modul *bluetooth* HC-05 dan f_{ck} sebesar 24 MHz.

- Baudrate 460800

$$USARTDIV_{seharusnya} = \left(\frac{24000000}{16\times\ 460800}\right) = 3{,}255208333$$

$$USARTDIV_{pada\ resiter} = 3{,}25$$

$$BaudRate_{pada\ USART} = \left(\frac{24000000}{16\times\ 3{,}25}\right) = 461538{,}461538$$

$$Error[\%] = \left(\frac{461538{,}461538}{460800} - 1\right) \times 100\% = 0{,}16\%$$

- Baudrate 230400

$$USARTDIV_{seharusnya} = \left(\frac{24000000}{16\times\ 230400}\right) = 6{,}5104167$$

$$USARTDIV_{pada\ resiter} = 6{,}5$$

$$BaudRate_{pada\ USART} = \left(\frac{24000000}{16\times\ 6{,}5}\right) = 230769{,}230769$$

$$Error[\%] = \left(\frac{230769{,}230769}{230400} - 1\right) \times 100\% = 0{,}16\%$$

- Baudrate 115200

$$USARTDIV_{seharusnya} = \left(\frac{24000000}{16\times\ 115200}\right) = 13{,}020833$$

$$USARTDIV_{pada\ resiter} = 13$$

$$BaudRate_{pada\ USART} = \left(\frac{24000000}{16\times\ 13}\right) = 115384{,}615384$$

$$Error[\%] = \left(\frac{115384{,}615384}{115200} - 1\right) \times 100\% = 0{,}16\%$$

- Baudrate 57600

$$USARTDIV_{seharusnya} = \left(\frac{24000000}{16\times\ 57600}\right) = 26{,}04167$$

$$USARTDIV_{pada\ resiter} = 26{,}0625$$

$$BaudRate_{pada\ USART} = \left(\frac{24000000}{16\times\ 26{,}0625}\right) = 57553{,}9568$$

$$Error[\%] = \left(\frac{57553{,}9568}{57600} - 1\right) \times 100\% = -0{,}08\%$$

- Baudrate 38400

$$USARTDIV_{seharusnya} = \left(\frac{24000000}{16\times\ 38400}\right) = 39{,}0625$$

$$USARTDIV_{pada\ resiter} = 39{,}0625$$

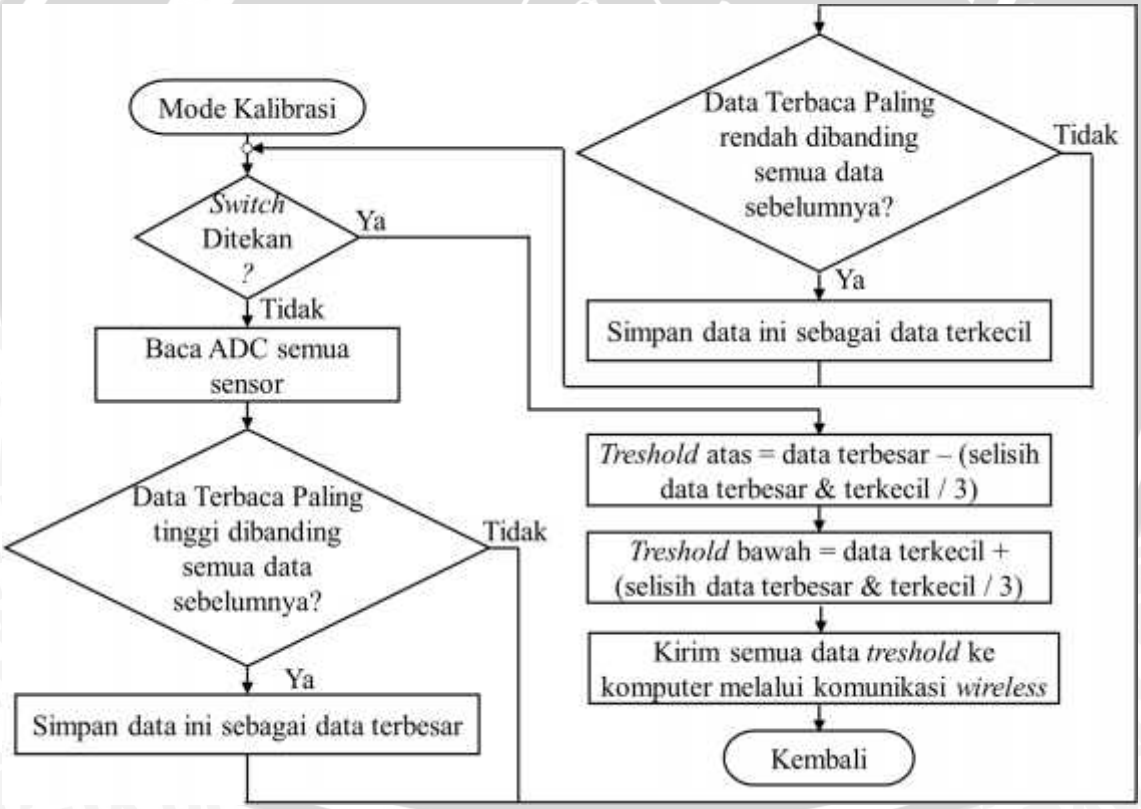
$$BaudRate_{pada\ USART} = \left(\frac{24000000}{16\times\ 39{,}0625}\right) = 38400$$

$$Error[\%] = \left(\frac{38400}{38400} - 1\right) \times 100\% = 0\%$$

Terlihat bahwa *baudrate* sebesar 38400 merupakah nilai *baudrate* terbesar yang dapat diterapkan pada modul *bluetooth* HC-05 dan menyebabkan eror sebesar nol persen pada mikrokontroler STM32F100RB. Oleh karena itu, *baudrate* yang digunakan dalam perancangan ini adalah 38400.

- **Mode Kalibrasi**

Setelah melakukan inisialisasi, mikrokontroler masuk dalam tahap kalibrasi. Kalibrasi ini bertujuan untuk menentukan nilai *treshold* ADC pada masing-masing sensor cahaya TCRT500 yang mana tiap-tiap nilai ini digunakan sebagai acuan untuk menentukan apakah suatu unit sensor cahaya TCRT5000 sedang berada di atas permukaan hitam atau putih. Hal ini perlu dilakukan karena tiap rangkaian sensor cahaya TCRT5000 dalam satu rangkaian sensor garis tidak identik satu sama lain. *Flowchart* mode kalibrasi pada mikrokontroler ditunjukkan dalam Gambar 3.10.



Gambar 3.10 *Flowchart* Mode Kalibrasi pada Mikrokontroler

Ketika masuk dalam tahap kalibrasi, mikrokontroler melakukan *scanning* pembacaan sensor garis secara terus-menerus sampai suatu tombol tertentu ditekan. Dalam satu siklus *scanning*, mikrokontroler membaca nilai ADC tiap unit rangkaian sensor cahaya TCRT5000. Jika data yang terbaca lebih tinggi dibandingkan dengan data-data sebelumnya, maka nilai ini disimpan sebagai nilai pembacaan terbesar. Jika tidak,

maka data dibiarkan dan tidak disimpan. Begitu pula jika data yang terbaca lebih rendah dibandingkan dengan data-data sebelumnya, maka nilai ini disimpan sebagai nilai pembacaan terkecil. Jika tidak, maka data dibiarkan dan tidak disimpan. Siklus ini dijalankan pada tiap-tiap unit keluaran TCRT5000. Jika unit sensor TCRT5000 terakhir telah dibaca, maka proses kembali dilakukan pada sensor TCRT5000 pertama. Proses ini terus berputar sampai *user* menekan sebuah tombol yang mengakibatkan *scanning* pembacaan nilai ADC semua unit sensor berhenti. Ketika proses *scanning* selesai, maka akan didapatkan dua buah data pembacaan tertinggi dan pembacaan terendah untuk masing-masing sensor.

Setelah *scanning* selesai, maka mikrokontroler melakukan perhitungan untuk menentukan nilai *treshold* untuk masing-masing unit keluaran TCRT5000. Nilai *treshold* ini digunakan untuk proses binerisasi agar tiap unit sensor cahaya TCRT5000 dapat ditentukan nilai *logic*nya, apakah hitam (HIGH) ataukah putih (LOW). Nilai *treshold* untuk masing-masing sensor dihitung menggunakan rumus sebagai berikut :

$$nilai\ treshold\ hitam = data\ terbesar - \left(\frac{data\ terbesar - data\ terkecil}{3}\right) \quad \ldots (4.7)$$

$$nilai\ treshold\ putih = data\ terkecil + \left(\frac{data\ terbesar - data\ terkecil}{3}\right) \quad \ldots (4.8)$$

Data terbesar, data terkecil, nilai *treshold* hitam, dan nilai *treshold* putih diilustrasikan dalam Gambar 3.11.

Gambar 3.11 Diagram Data-Data yang Didapatkan dari Proses Kalibrasi

Saat robot sedang digunakan, jika nilai sensor yang terbaca lebih besar dari pada nilai *treshold* hitam, maka sensor memiliki nilai *logic* HIGH. Jika lebih kecil dari pada nilai *treshold* putih, maka sensor memiliki nilai *logic* LOW. Jika di antara keduanya, maka hasil pembacaan dipengaruhi oleh *noise*. Kondisi *logic* sensor ini nantinya akan divisualisasikan dalam program komputer.

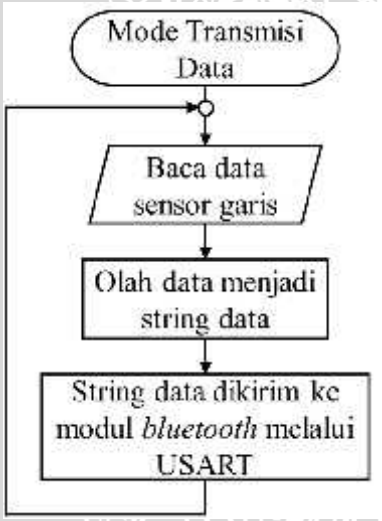
Data nilai *treshold* ini kemudian dikirimkan oleh mikrokontroler pada komputer melalui komunikasi nirkabel. Format paket data yang dikirimkan ke komputer adalah sebagai berikut :

Start string	*Treshold* putih data ke-0	delimiter	*Treshold* hitam data ke-0	delimiter	----	*Treshold* putih data ke-n	delimiter	*Treshold* hitam data ke-n	Stop character

Start string yang digunakan adalah tiga buah karakter "#". Delimiter yang digunakan adalah karakter garis miring ("/"). Sedangkan stop character yang digunakan adalah enter.

- **Mode Transmisi Data**

Setelah tahap kalibrasi selesai, maka tahap selanjutnya adalah mode transmisi data. Pada mode ini, mikrokontroler secara kontinu membaca data ADC kedua belas keluaran rangkaian TCRT5000, menghimpun data-data tersebut menjadi sebuah paket data, dan mengirimkan paket data ini ke komputer melalui komunikasi *wireless*. *Flowchart* proses kerja pada mode transmisi ini ditunjukkan dalam Gambar 3.12.



Gambar 3.12 *Flowchart* Program Mikrokontroler dalam Mode Transmisi Data

Pada mulanya, mikrokontroler membaca nilai ADC kedua belas unit TCRT5000. Dalam perancangan ini, tiap keluaran sensor TCRT5000 dibaca sebanyak tiga kali, kemudian hasil pembacaan ini dihitung nilai rata-ratanya. Sampling sebanyak tiga kali ini bertujuan agar hasil pembacaan menjadi lebih akurat. Setiap membaca keluaran satu buah unit sensor cahaya TCRT5000, pin GPIO mikrokontroler yang terhubung dengan pin selektor IC multiplekser menghasilkan *output* dengan kombinasi tertentu sehingga kanal dalam multiplekser terhubung dengan keluaran unit sensor yang dimaksud. Oleh karena itu, dalam satu waktu *input* ADC mikrokontroler terhubung dengan satu buah tegangan keluaran sensor cahaya TCRT5000.

Format paket data yang digunakan dalam transmisi adalah sebagai berikut :

Data ke-0	delimiter	Data ke-1	delimiter	- - - -	delimiter	Data ke-n	Stop character

n menunjukkan jumlah sensor cahaya yang digunakan dalam perancangan. Oleh karena jumlah sensor yang digunakan adalah 12 buah, maka nilai n=12. Data ke-n menunjukkan nilai ADC pada tiap-tiap sensor. Delimiter yang digunakan adalah karakter garis miring ("/"), sedangkan *stop character* yang digunakan adalah enter. Berdasarkan format data tersebut, terlihat bahwa nilai ADC yang dikirimkan dipisahkan oleh delimiter (tanda garis miring) satu dengan lainnya. Hal ini bertujuan untuk mempermudah pembentukan *string* data pada mikrokontroler dan pemecahan data pada program komputer. Pada komunikasi USART, tiap *frame* data terdiri atas 10 bit (1 *start bit*, 8 data bit, dan 1 stop bit). Nilai ADC maksimal yang mungkin terjadi adalah 255 (3 digit angka). Oleh karena itu, jumlah bit maksimal yang mungkin dalam satu paket data dapat dihitung sebagai berikut :

$$jumlahBit = (jumlahData * 3 + jumlah\ tanda\ "/" + jumlahEnter) * 10bit$$
$$= (12 * 3 + 11 + 1) * 10\ bit = 480\ bits$$

Dengan *baudrate* sebesar 38400, maka waktu yang diperlukan untuk pengiriman satu buah paket data dihitung sebagai berikut :

$$t_{paketData} = \frac{jumlahBit}{BaudRate} = \frac{480\ bit}{38400\ \frac{bit}{sekon}} = 12{,}5\ ms$$

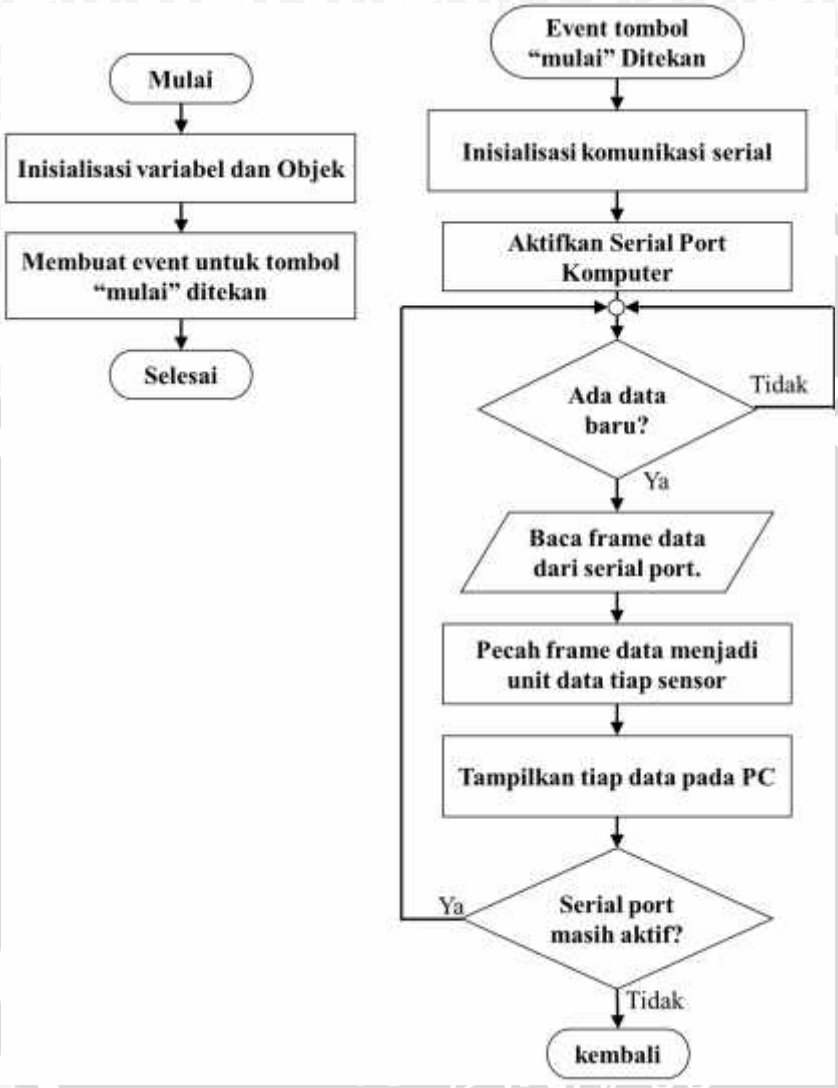
Jadi, waktu maksimal pengiriman satu buah paket data adalah 12,5 ms. Dalam pengaplikasiannya, nilai ini bisa lebih kecil jika banyak nilai ADC sensor yang hanya mencapai nilai puluhan (dua digit).

3.2.4 Perancangan Perangkat Lunak pada Komputer

Pada perancangan program visual untuk menampilkan hasil pembacaan sensor garis ini, secara garis besar program dibagi menjadi dua bagian : bagian untuk visualisasi sensor garis dengan sumber *stream data* dan bagian untuk visualisasi sensor garis dengan sumber *file datalogger*. Pada visualisasi sensor garis dengan *stream data*, data secara terus menerus diterima oleh komputer sehingga tampilan visualisasi menjadi dinamis. Sedangkan pada visualisasi sensor garis dengan *datalogger*, tampilan visual berubah-ubah (dinamis) jika *cursor* indeks data diubah-ubah oleh *user*.

Flowchart program visual untuk mode visualisasi *stream data* ditunjukkan dalam Gambar 3.13. Ketika program pertama kali dijalankan, hal yang dilakukan adalah inisialisasi variabel dan objek. Objek yang dibuat berasal dari berbagai macam kelas, seperti JTabbedPane untuk membuat tabulasi, JLabel untuk membuat teks, JTextField

untuk membuat tempat penampung teks satu baris, JCheckBox untuk fitur cek lis, JComboBox untuk menampilkan menu secara *drop down*, JButton untuk tombol, JPanel untuk menghimpun beberapa komponen menjadi satu tempat, dan JFrame sebagai window yang menampung semua tampilan program.

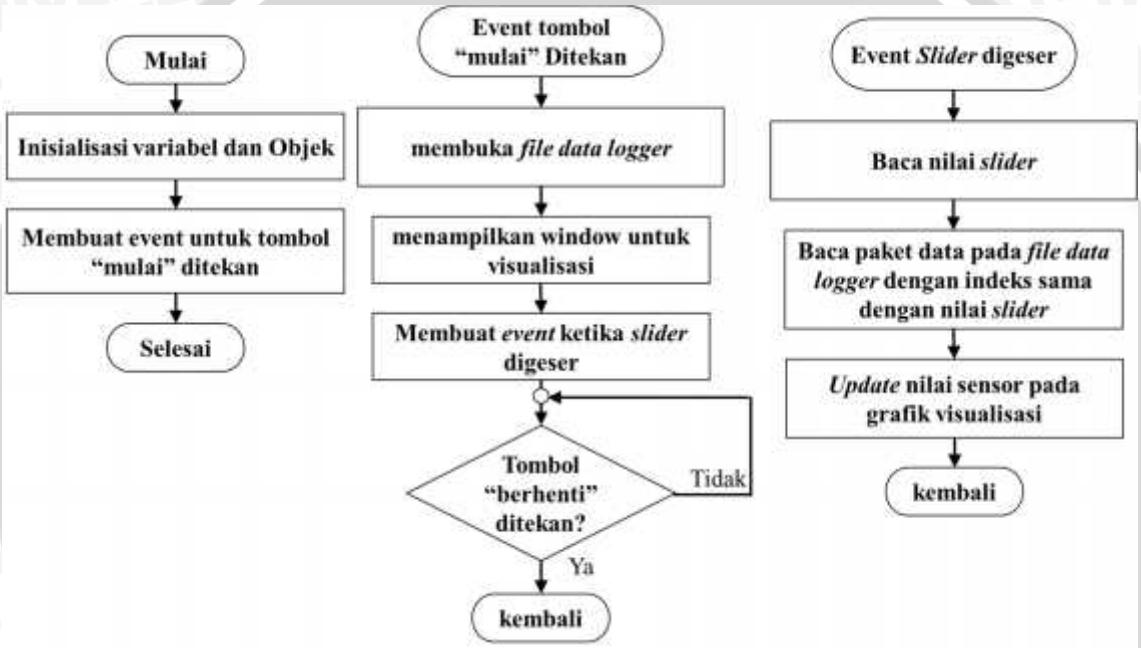


Gambar 3.13 *Flowchart* Program Visual dalam Mode Visualisasi *Stream Data*

Setelah melakukan inisialisasi, program selanjutnya membuat *event* yang dilakukan ketika tombol "mulai" ditekan. Ketika tombol ini ditekan, komputer melakukan inisialisasi komunikasi serial. Tujuannya adalah agar komunikasi serial dengan mikrokontroler dapat berjalan selaras. Hal-hal yang diatur antara lain COM PORT yang digunakan, *baudrate*, data bit, stop bit, dan *parity* bit. Setelah itu, komputer mengaktifkan serial port komputer sehingga data dapat masuk ke sistem USB komputer. Selama terdapat kiriman data dari mikrokontroler, maka program visual akan terus-menerus membaca paket data yang datang ini. Kemudian, program memecah paket data ini menjadi data-data tiap unit sensor. Setelah itu, tampilan visual yang berupa virtual port,

grafik garis, grafik batang, dan animasi sensor diperbarui dengan data yang baru ini. Setelah proses-proses ini dieksekusi, jika tombol "berhenti" ditekan, maka program akan menonaktifkan USB serial *port* sehingga aliran data tertutup dan program pun berhenti. Jika tombol "berhenti" tidak ditekan, maka program akan kembali menjalankan proses pembacaan paket data, pemecahan data, dan visualisasi sensor yang ditampilkan.

Flowchart program visual untuk mode visualisasi *data logger* ditunjukkan dalam Gambar 3.14. Ketika program pertama kali dijalankan, hal yang dilakukan adalah inisialisasi variabel dan objek. Sama seperti pada mode *stream data*, pada mode ini, inisialiasi dilakukan untuk menampilkan tombol, label, *input* teks, dan lain sebagainya.



Gambar 3.14 *Flowchart* Program Visual dalam Mode Visualisasi *Data Logger*

Setelah melakukan inisialisasi, program komputer membuka *file* yang berisi data hasil pembacaan sensor. Setelah itu, window yang berisi tampilan visual pembacaan sensor (panel *stream* data, panel *line chart*, panel *bar chart*, dan panel animasi sensor). Setelah tampilan visual ini selesai dieksekusi, program membuat *event* yang dijalankan ketika *slider* yang terdapat pada panel kontrol diubah posisinya. Ketika *slider* ini diubah posisinya, maka indeks pada slider ini akan berubah. Program kemudian membaca satu paket data hasil pembacaan sensor yang memiliki indeks sama dengan indeks pada *slider*. Kemudian, paket data ini dipecah menjadi unit data tiap sensor cahaya TCRT5000 dan grafik visual diperbarui (di*update*) nilainya dengan nilai sensor yang baru ini.

3.3 Pengujian Alat

Untuk menganalisis kinerja alat apakah sesuai dengan yang direncanakan, maka dilakukan pengujian sistem. Pengujian dilakukan pada masing-masing blok pada

perancangan perangkat keras, perancangan perangkat lunak, serta pengujian keseluruhan untuk mengetahui performa dari keseluruhan sistem yang telah dirancang dan dibuat.

3.3.1 Pengujian Rangkaian Sensor Cahaya TCRT5000

Pengujian dilakukan untuk mengetahui apakah nilai tegangan keluaran rangkaian TCRT5000 telah sesuai dengan kondisi yang seharusnya. Nilai keluaran rangkaian sensor tinggi jika diletakkan pada bagian lintasan yang berwarna hitam dan rendah jika diletakkan pada bagian lintasan yang berwarna putih.

3.3.2 Pengujian Rangkaian *Bi-Directional Logic Level Shifter*

Pengujian dilakukan untuk mengetahui apakah rangkaian *Bi-directional logic level shifter* yang telah dibuat dapat menghasilkan level tegangan *output* HIGH dan LOW yang sesuai walaupun berbeda level tegangannya dengan bagian *input*.

3.3.3 Pengujian Sensor Garis Secara Keseluruhan

Pengujian dilakukan untuk mengetahui apakah keseluruhan rangkaian sensor garis telah dapat merepresentasikan hasil pembacaan tiap unit sensor cahaya TCRT5000 dan mengeluarkan sinyal tegangan sesuai dengan yang diperintahkan oleh mikrokontroler.

3.3.4 Pengujian Pengujian Jarak dan Disipasi Energi Modul *Bluetooth* HC-05

Pengujian dilakukan untuk mengetahui jarak maksimal transmisi data antara dua buah modul *bluetooth* HC-05 dan disipasi energi listrik ketika saling mengirim data. Pengujian dilakukan dua kali, yang pertama ketika kedua modul *bluetooth* berada dalam posisi horizontal dan yang kedua dalam posisi vertikal.

3.3.5 Pengujian Kecepatan Transmisi Data

Pengujian dilakukan untuk mengetahui hubungan antara lama waktu transmisi data terhadap banyaknya paket data yang diterima oleh komputer. Sebagai tambahan, hubungan antara lama waktu transmisi data dengan jumlah paket data yang terkirim juga diuji.

3.3.6 Pengujian Akurasi Pembacaan Sistem Visualisasi Sensor Garis

Pengujian dilakukan untuk mengetahui tingkat akurasi pembacaan sistem visualisasi sensor garis. Tingkat akurasi dikatakan 100% jika pola warna lintasan tempat robot berada sama selalu sama dengan pola yang ditunjukkan pada komputer.

3.3.7 Pengujian Presisi Pembacaan Sistem Visualisasi Sensor Garis

Pengujian dilakukan untuk mengetahui tingkat presisi pembacaan sistem visualisasi sensor garis. Pengujian dilakukan sebanyak beberapa kali dan diharapkan pembacaan memberikan hasil yang sama.

3.3.8 Pengujian Jarak Sensor Garis ke Lintasan terhadap Tegangan Keluaran Sensor

Pengujian dilakukan untuk mengetahui pengaruh jarak sensor garis ke lintasan terhadap hasil pembacaan sensor garis. Pengujian dilakukan dengan cara mengangkat robot sedikit demi sedikit sampai sistem visualisasi tidak lagi mampu menghasilkan pembacaan dengan benar.

3.3.9 Pengujian Keseluruhan Sistem

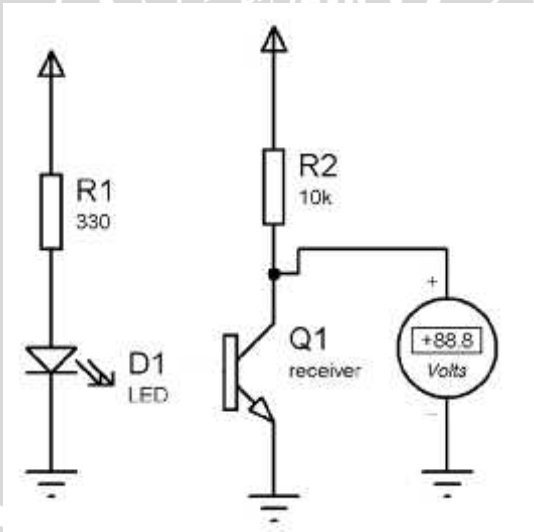
Pengujian keseluruhan sistem ini dilakukan dengan menyambungkan semua *hardware* yang telah dibuat berdasarkan blok diagram dan memasukkan program berupa *software* agar mikrokontroler dan komputer dapat bekerja. Sistem dikatakan bekerja dengan baik jika dapat berjalan sesuai dengan *flowchart* yang telah direncanakan.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Pengujian Rangkaian Sensor Cahaya TCRT5000

Pengujian ini bertujuan untuk mengetahui nilai *output* tegangan rangkaian sensor cahaya TCRT5000 ketika diletakkan di atas permukaan putih dan ketika diletakkan di atas permukaan hitam. Hasil pengujian dikatakan berhasil jika nilai *output* tegangan ketika sensor berada di atas permukaan hitam lebih tinggi dari pada ketika diletakkan di atas permukaan putih.

Pengujian dilakukan di atas lintasan lomba *line follower robot* yang dicetak di atas banner. Lintasan memiliki garis yang berwarna hitam dan *background* yang berwarna putih. Setelah rangkaian sensor garis diberi catu tegangan, sensor pertama (paling kiri) diletakkan di atas garis hitam. Nilai tegangan keluaran dari rangkaian sensor cahaya TCRT5000 kemudian dicatat. Setelah itu, sensor pertama tadi diletakkan di atas garis putih. Nilai tegangan keluarannya kembali dicatat. Kegiatan ini diulang lagi untuk sebelas sensor yang lain. Tiap unit data diambil sebanyak tiga kali dan kemudian dihitung rata-ratanya agar hasilnya lebih valid. Rangkaian pengujian rangkaian sensor cahaya TCRT5000 ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Rangkaian Pengujian Rangkaian TCRT5000

Data hasil pengujian keluaran rangkaian sensor cahaya TCRT5000 baik ketika kondisi ruang terdapat cahaya lampu maupun tidak terdapat cahaya lampu disajikan dalam Tabel 4.1. Adapun diagram batang dari hasil pengujian ini ditunjukkan dalam Gambar 4.2 dan Gambar 4.3.

Tabel 4.1 Hasil Pengujian Keluaran Rangkaian Sensor Cahaya TCRT5000

Rangkaian Sensor yang Diuji	Tanpa Cahaya Lampu					Dengan Cahaya Lampu				
	Tegangan Keluaran (permukaan hitam)	Tegangan Keluaran (permukaan putih)	Selisih tegangan	*Treshold* Putih	Treshold Hitam	Tegangan Keluaran (permukaan hitam)	Tegangan Keluaran (permukaan putih)	Selisih Tegangan	*Treshold* Putih	*Treshold* Hitam
Sensor 0	2,87 V	156 mV	2,71 V	1,060 V	1,963 V	2,86 V	159 mV	2,70 V	1,058 V	1,957 V
Sensor 1	2,73 V	156 mV	2,57 V	1,013 V	1,870 V	2,63 V	159 mV	2,47 V	0,983 V	1,806 V
Sensor 2	2,64 V	163 mV	2,47 V	0,987 V	1,812 V	2,51 V	161 mV	2,35 V	0,944 V	1,727 V
Sensor 3	2,33 V	138 mV	2,19 V	0,869 V	1,599 V	2,27 V	139 mV	2,13 V	0,849 V	1,560 V
Sensor 4	2,75 V	155 mV	2,60 V	1,020 V	1,885 V	2,65 V	152 mV	2,50 V	0,985 V	1,817 V
Sensor 5	2,56 V	155 mV	2,40 V	0,956 V	1,756 V	2,57 V	165 mV	2,40 V	0,966 V	1,766 V
Sensor 6	2,88 V	183 mV	2,69 V	1,081 V	1,979 V	2,86 V	176 mV	2,68 V	1,071 V	1,965 V
Sensor 7	2,48 V	165 mV	2,32 V	0,938 V	1,711 V	2,24 V	164 mV	2,07 V	0,855 V	1,546 V
Sensor 8	3,08 V	177 mV	2,90 V	1,144 V	2,112 V	3,04 V	172 mV	2,86 V	1,127 V	2,082 V
Sensor 9	2,38 V	161 mV	2,22 V	0,902 V	1,643 V	2,29 V	156 mV	2,13 V	0,866 V	1,577 V
Sensor 10	2,90 V	164 mV	2,74 V	1,076 V	1,988 V	2,90 V	167 mV	2,73 V	1,077 V	1,987 V
Sensor 11	2,42 V	159 mV	2,26 V	0,912 V	1,664 V	2,37 V	118 mV	2,25 V	0,869 V	1,619 V
Rata-rata	2,67 V	161 mV	2,51 V	0,996 V	1,832 V	2,60 V	157 mV	2,44 V	0,971 V	1,784 V

Gambar 4.2 Diagram Batang Hasil Pengujian Rangkaian Sensor Cahaya TCRT5000 untuk Kondisi Lingkungan tanpa Cahaya Lampu

Gambar 4.3 Diagram Batang Hasil Pengujian Rangkaian Sensor Cahaya TCRT5000 untuk Kondisi Lingkungan dengan Cahaya Lampu

Berdasarkan Tabel 4.1, terlihat bahwa untuk kondisi intensitas cahaya lingkungan yang sama, tegangan keluaran rangkaian sensor cahaya TCRT5000 yang diletakkan di atas permukaan hitam memiliki nilai yang lebih besar dibandingkan dengan yang diletakkan di atas permukaan putih. Hal ini disebabkan karena fototransistor menjadi bersifat isolatif jika menerima sedikit cahaya (ketika sensor diletakkan di atas permukaan hitam) sehingga tegangan keluaran rangkaian sensor cahaya TCRT5000 mendekati tegangan V_{CC}. Sebaliknya, ketika fototransistor menjadi bersifat konduktif jika menerima banyak

cahaya (ketika sensor diletakkan di atas permukaan putih) sehingga tegangan keluaran rangkaian sensor cahaya TCRT5000 mendekati *ground*. Hasil dari dua macam pengujian untuk kondisi pencahayaan lingkungan yang berbeda (dengan dan tanpa cahaya lampu) menunjukkan *output* tegangan yang hampir sama. Untuk permukaan hitam, rata-rata hasil pengukuran tegangan dengan kondisi lingkungan tanpa cahaya lampu adalah 2,67 V, nilai minimal sebesar 2,33 V dan nilai terbesar sebesar 3,08 V. Sedangkan rata-rata untuk kondisi lingkungan dengan cahaya lampu adalah 2,60 V, nilai terkecil sebesar 2,24 V, dan terbesar 3,04 V. Untuk permukaan putih, rata-rata hasil pengukuran tegangan dengan kondisi lingkungan tanpa cahaya lampu adalah 160,9 mV, terkecil 138 mV, dan terbesar 183 mV. Sedangkan untuk kondisi lingkungan dengan cahaya lampu, rata-ratanya adalah 157 mV, terkecil 118 mV, dan terbesar 176 mV.

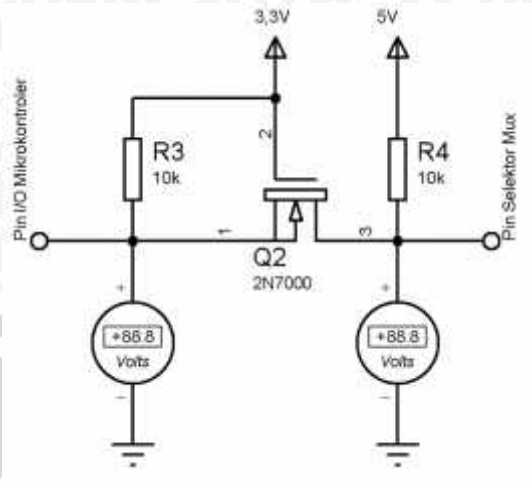
Dari hal ini, dapat ditarik kesimpulan sebagai berikut. Pertama, sensor telah dapat membedakan antara permukaan hitam dan putih. Hal ini ditunjukkan dengan adanya perbedaan tegangan keluaran yang cukup besar untuk kedua kondisi tersebut dengan nilai sebesar 2,47 V. kedua, terlihat bahwa intensitas cahaya lingkungan berpengaruh sedikit sekali terhadap performansi dari sensor cahaya TCRT5000.

4.2. Pengujian Rangkaian *Logic Level Shifter*

Pengujian ini bertujuan untuk mengetahui performasi dari rangkaian *logic translator*. Hasil yang diharapkan dari pengujian ini adalah sisi "*higher voltage*" dapat menghasilkan tegangan 5 volt ketika sisi "*lower voltage*" memiliki tegangan 3,3 volt. Selain itu, sisi "*higher voltage*" dapat menghasilkan tegangan 0 volt ketika sisi "*lower voltage*" memiliki tegangan 0 volt.

Dalam rangkaian sensor garis *line follower robot*, terdapat tiga buah rangkaian *logic level shifter* yang menghubungkan pin mikrokontroler dengan pin selektor IC multiplekser 4051. Pada rangkaian *logic* translator, terdapat dua sisi yang dapat dihubungkan dengan *device* lain : sisi "*lower* voltage" dan sisi "*higher voltage*". Sisi "*lower voltage*" dihubungkan dengan pin mikrokontroler, sedangkan sisi "*higher voltage*" dihubungkan dengan pin selektor IC multiplekser. Pin mikrokontroler yang digunakan adalah Pin A3, Pin A4, dan Pin A5. Pertama-tama, Pin A3 yang digunakan untuk pengujian. Ketika rangkaian *logic level shifter* dihubungkan ke Pin A3 (secara bergantian), Pin A3 diset agar berlogika rendah, kemudian tegangan pada kedua sisi rangkaian *logic level shifter* dicatat. Setelah itu, Pin A3 diset agar berlogika tinggi, kemudian tegangan pada kedua sisi rangkaian *logic level shifter* kembali dicatat.

Kegiatan ini kembali dilakukan untuk pin A4 dan pin A5. Skematik Pengujian untuk rangkaian *logic level shifter* ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Skematik Pengujian untuk Rangkaian *Logic Level Shifter*

Tabel 4.4, 4.5, dan 4.6 berturut-turut menunjukkan nilai tegangan pada tiga buah rangkaian *logic level shifter* ketika dihubungkan dengan Pin A3, Pin A4, dan Pin A5 mikrokontroler.

Tabel 4.2 Nilai Tegangan pada Rangkaian *Logic level shifter* dengan Pin A3 sebagai *Input*

	Rangkaian 1		**Rangkaian 2**		**Rangkaian 3**	
	LOW	HIGH	LOW	HIGH	LOW	HIGH
Sisi *"lower level"*	19,8 mV	2,98 V	19,6 mV	2,98 V	19,8 mV	2,98 V
Sisi *"higher level"*	14,1 mV	4,93 V	15,1 mV	4,92 V	14,4 mV	4,92 V

Tabel 4.3 Nilai Tegangan pada Rangkaian *Logic level shifter* dengan Pin A4 sebagai *Input*

	Rangkaian 1		**Rangkaian 2**		**Rangkaian 3**	
	LOW	HIGH	LOW	HIGH	LOW	HIGH
Sisi *"lower level"*	19,8 mV	2,98 V	19,8 mV	2,98 V	19,9 mV	2,98 V
Sisi *"higher level"*	14,1 mV	4,93 V	14,0 mV	4,92 V	14,1 mV	4,92 V

Tabel 4.4 Nilai Tegangan pada Rangkaian *Logic level shifter* dengan Pin A5 sebagai *Input*

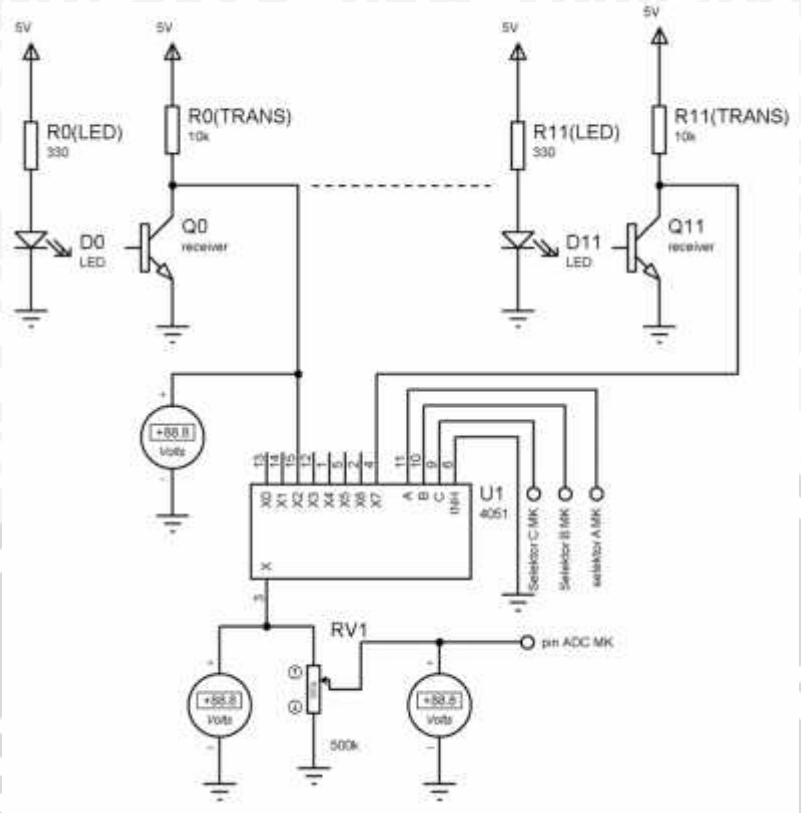
	Rangkaian 1		**Rangkaian 2**		**Rangkaian 3**	
	LOW	HIGH	LOW	HIGH	LOW	HIGH
Sisi *"lower level"*	19,9 mV	2,98 V	19,8 mV	2,98 V	20,0 mV	2,98 V
Sisi *"higher level"*	14,0 mV	4,93 V	14,7 mV	4,92 V	13,9 mV	4,92 V

Berdasarkan data pada Tabel 4.4, 4.5, dan 4.5, terlihat bahwa ketiga rangkaian *logic translator* dapat mengonversikan tegangan dari 3,3 volt menjadi 5 volt. Sisi *lower voltage* pada rangkaian ini dihubungkan dengan pin I/O mikrokontroler yang menghasilkan rata-rata tegangan untuk *logic* LOW sebesar 19,8 mV dan untuk *logic* HIGH sebesar 2,98 volt. Sedangkan pada sisi *higher voltage*, dihasilkan rata-rata tegangan untuk *logic* LOW sebesar 14,3 mV dan untuk *logic* HIGH sebesar 4,92 volt. Telah diketahui dari *data sheet* bahwa nilai V_{IL} IC multiplekser 4051 adalah 1,5 volt dan nilai V_{IH} adalah 3,5 volt. Oleh karena itu, rangkaian *logic translator* ini telah dapat bekerja seperti yang direncanakan.

4.3. Pengujian Sensor Garis Keseluruhan

Tujuan dari pengujian ini adalah untuk mengetahui apakah sensor garis yang dirancang telah bekerja sesuai konsep yang dibuat atau tidak. Sensor dikatakan dapat bekerja dengan baik jika sensor garis mengeluarkan *output* tegangan sesuai dengan *output* rangkaian sensor TCRT500 yang diminta oleh mikrokontroler.

Pengujian dilakukan di atas Pengujian dilakukan secara satu persatu untuk semua unit rangkaian sensor cahaya TCRT5000. Misalkan yang pertama kali digunakan adalah sensor ke-0. Langkah pertama, sensor ini diletakkan di atas permukaan putih, dan tegangan keluaran rangkaian sensor cahaya TCRT5000 diukur dan dicatat. Kemudian, mikrokontroler mengatur selektor pada IC multiplekser agar kanal analog *common* dalam IC ini terhubung dengan keluaran sensor ke-0. Kemudian, keluaran kanal analog *common* ini diukur. Dalam kondisi ideal, nilai tegangan pada kanal *common* ini sama dengan nilai tegangan pada keluaran rangkaian sensor ke-0 TCRT5000. Hal ini disebabkan karena nilai resistansi kanal analog pada IC multiplekser CD4051 memiliki nilai resistansi yang relatif kecil dibandingkan dengan resistansi resistor variabel dan resistansi masukan ADC mikrokontroler. Setelah itu, *output* tegangan pada kaki *wiper* resistor variabel juga diukur. Hasil yang diharapkan adalah nilai tegangan pada kaki *wiper* resistor memiliki 0,66 kali nilai tegangan pada kaki kanal *common* IC multiplekser. Prosedur tersebut diulang, akan tetapi sekarang sensor ke-0 tersebut diletakkan di atas permukaan hitam. Tegangan *output* rangkaian sensor TCRT5000 dan kanal *common* kembali dicatat. Semua langkah-langkah ini kemudian diulang untuk sensor ke-1 sampai sensor ke-11. Rangkaian pengujian sensor garis secara keseluruhan ditunjukkan dalam Gambar 4.5. Sedangkan tabel 4.5 berisi data-data hasil pengujian rangkaian sensor garis *line follower robot* secara keseluruhan. Tiap unit data diambil sebanyak tiga kali dan kemudian dihitung rata-ratanya agar hasilnya lebih valid.



Gambar 4.5 Rangkaian Pengujian Sensor Garis secara Keseluruhan

Berdasarkan pengujian, terlihat bahwa bahwa tegangan keluaran pada kanal "common" IC multiplekser (sebagai *output*) memiliki nilai yang sama dengan nilai tegangan pada keluaran rangkaian TCRT5000, tergantung pengaturan pada pin selektor. Ketika mikrokontroler memberikan kombinasi logika pada pin selekor sehingga kanal *common* terhubung dengan sensor 0, tegangan pada pin "*common*" (sebagai *output*) sama dengan tegangan keluaran pada sensor 0. Ketika kombinasi logika pada pin selektor menyebabkan kanal "*common*" terhubung dengan sensor 1, tegangan pada pin "*commmon*" ini sama dengan tegangan keluarga pada sensor 1. Dapat ditarik kesimpulan bahwa mekanisme *switch* pada IC multiplekser untuk membaca keluaran rangkaian TCRT5000 secara bergantian telah berjalan sebagaimana mestinya.

Tegangan keluaran pada kaki *wiper* resistor variabel memiliki nilai sebesar 0,66 kali dibandingkan tegangan pada kaki *common* IC multiplekser. Hal ini berlaku untuk semua sensor dari sensor 0 hingga sensor 11. Hal ini juga berlaku baik untuk kondisi pembacaan permukaan hitam maupun permukaan putih. Dapat ditarik kesimpulan bahwa resistor variabel telah dapat menurunkan tegangan sesuai dengan perencanaan sehingga *output* tegangan aman untuk dihubungkan dengan pin ADC mikrokontroler ARM yang beroperasi pada tegangan 3,3 volt.

Tabel 4.5 Data Hasil Pengujian Rangkaian Sensor Garis *Line Follower Robot*.

Sensor ke-	Warna Permukaan	Tegangan Keluaran TCRT5000	Pin Selektor*)			Tegangan Keluaran kanal "common"	Tegangan Keluaran resistor variable
			C	B	A		
0	Putih	163,5 mV	L	H	L	162,2 mV	101 mV
	Hitam	2,99 V				2,97 V	1,93 V
1	Putih	159,1 mV	L	H	H	158,8 mV	98,5 mV
	Hitam	2,42 V				2,39 V	1,62 V
2	Putih	167,5 mV	H	L	L	166,7 mV	102,5 mV
	Hitam	2,48 V				2,45 V	1,62 V
3	Putih	139,6 mV	H	L	H	139,2 mV	86,2 mV
	Hitam	2,39 V				2,32 V	1,57 V
4	Putih	159,9 mV	H	H	L	159,8 mV	98,5 mV
	Hitam	2,58 V				2,56 V	1,77 V
5	Putih	175,3 mV	H	H	H	175,3 mV	110,0 mV
	Hitam	2,72 V				2,72 V	1,77 V
6	Putih	187,2 mV	H	H	H	186,6 mV	117,1 mV
	Hitam	2,77 V				2,74 V	1,80 V
7	Putih	166,7 mV	H	H	L	166,7 mV	104,4 mV
	Hitam	2,42 V				2,35 V	1,54 V
8	Putih	176,6 mV	H	L	H	176,3 mV	110,5 mV
	Hitam	3,07 V				3,06 V	2,03 V
9	Putih	161,3 mV	H	L	L	160,7 mV	99,7 mV
	Hitam	2,44 V				2,43 V	1,55 V
10	Putih	168,7 mV	L	H	H	168,3 mV	103,5 mV
	Hitam	2,77 V				2,75 V	1,84 V
11	Putih	162,5 mV	L	H	H	162,1 mV	99,2 mV
	Hitam	2,53 V				2,52 V	1,67 V

*)L = logika LOW, H = logika HIGH

4.4. Pengujian Jarak dan Disipasi Energi Modul *Bluetooth* HC-05

Tujuan dari pengujian ini adalah untuk mengetahui jarak maksimal pengiriman data antara dua buah modul *bluetooth* HC-05. Pengujian juga bertujuan untuk mengetahui pengaruh jarak terhadap konsumsi daya listrik modul *bluetooth* HC-05.

Sebelum pengujian dimulai, dua buah modul *bluetooth* yang akan digunakan untuk pengujian telah diatur sebelumnya agar ketika keduanya dinyalakan, kedua modul *bluetooth* ini langsung *pairing* secara otomatis. Pengaturan ini dilakukan dengan menggunakan AT *command*. Modul *bluetooth* yang pertama dikoneksikan dengan mikrokontroler dan berperan sebagai pengirim data. Sedangkan modul *bluetooth* yang kedua dikoneksikan dengan komputer melalui USB to TTL dan bertindak sebagai penerima data. Percobaan dilakukan sebanyak dua kali dengan kondisi pengujian yang berbeda. Pengujian dilakukan dengan mengirimkan data sebanyak 50 data kemudian setiap 10 data terdapat *delay* selama 1 detik.

Pada percobaan pertama kedua modul *bluetooth* memiliki posisi horizontal. Setelah sistem dinyalakan, kedua modul *bluetooth* diletakkan pada jarak 1 meter. *Bluetooth* pengirim mengirimkan data dan hasilnya dilihat melalui monitor komputer. Pada setiap percobaan, data arus listrik yang mengalir pada pin V_{CC} modul *bluetooth* HC-05 dicatat. Kemudian diletakkan pada jarak 2 meter dan pengiriman data kembali dilakukan. Hal ini terus dilakukan tiap jarak 1 meter dan pengujian berhenti ketika kedua modul *bluetooth* tidak terkoneksi lagi. Pada percobaan kedua, langkah serupa juga dilakukan. Hanya saja, data diambil setiap jarak 5 meter dan kedua modul *bluetooth* dalam posisi vertikal.

Data hasil pengujian modul *bluetooth* HC-05 ditunjukkan dalam Tabel 4.8 dan Tabel 4.9. Adapun Grafiknya ditunjukkan dalam Gambar 4.6 dan Gambar 4.7. Berdasarkan data hasil pengujian, dapat disimpulkan bahwa dalam posisi horizontal, dua buah modul *bluetooth* HC-05 dapat saling berkomunikasi dengan jarak maksimal 8 meter. Sedangkan dalam posisi vertikal, dua buah modul *bluetooth* HC-05 dapat saling berkomunikasi dengan jarak maksimal 35 meter. Dapat disimpulkan juga bahwa dengan posisi vertikal, dua buah modul *bluetooth* HC-05 dapat memiliki jarak maksimum *pairing* lebih jauh dari pada dengan posisi horizontal.

Tabel 4.6 Hasil Pengujian Modul *Bluetooth* dengan Posisi Horizontal

Jarak (m)	Data diterima	delay	Arus Listrik (mA)	Daya (mW) (V_{CC} = 5V)
1	Terkoneksi	Terkoneksi	3,67	18,35
2	Terkoneksi	Terkoneksi	3,70	18,5
3	Terkoneksi	Terkoneksi	3,81	19,05
4	Terkoneksi	Terkoneksi	3,95	19,75
5	Terkoneksi	Terkoneksi	4,12	20,6
6	Terkoneksi	Terkoneksi	4,23	21,15
7	Terkoneksi	Terkoneksi	4,31	21,55
8	Terkoneksi	Terkoneksi	4,52	22,6
9	koneksi terputus	koneksi terputus	9,73	48,65
10	koneksi terputus	koneksi terputus	9,73	48,65

Tabel 4.7 Hasil Pengujian Modul *Bluetooth* dengan Posisi Vertikal

Jarak (m)	Data diterima	delay	Arus Listrik (mA)	Daya (W) (V_{CC} = 5V)
5	Terkoneksi	Terkoneksi	4,12	20,7
10	Terkoneksi	Terkoneksi	4,63	23,15
15	Terkoneksi	Terkoneksi	5,03	25,15
20	Terkoneksi	Terkoneksi	5,85	29,25
25	Terkoneksi	Terkoneksi	6,92	34,6
30	Terkoneksi	Terkoneksi	8,27	41,35
35	Terkoneksi	Terkoneksi	9,65	48,25
40	Koneksi terputus	Koneksi terputus	9,72	48,6

Berdasarkan Grafik yang ditunjukkan dalam Gambar 4.6 dan Gambar 4.7, terlihat bahwa daya disipasi pada modul *bluetooth* HC-05 meningkat seiring dengan meningkatnya jarak transmisi data. Dapat disimpulkan bahwa besarnya daya disipasi pada pemancar maupun penerima modul *bluetooth* HC-05 sebanding dengan jarak antar kedua modul dalam transmisi data.

Gambar 4.6 Grafik Hasil Pengujian Daya HC-05 pada Posisi Horizontal

Gambar 4.7 Grafik Hasil Pengujian Daya HC-05 pada Posisi Vertikal

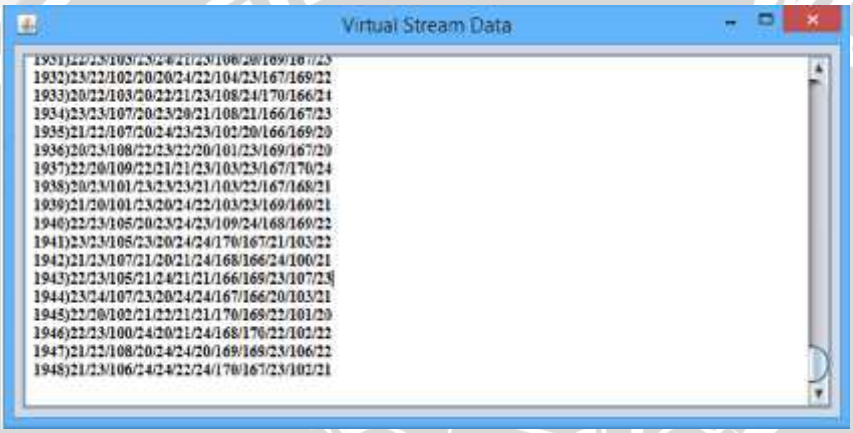
4.5. Pengujian Kecepatan Transmisi Data

Tujuan dari pengujian ini adalah untuk mengetahui hubungan antara banyaknya data yang ditransmisikan dan ukuran *file data logger* yang dihasilkan terhadap lama waktu transmisi data. Hasil dari pengujian ini adalah suatu grafik yang menyatakan hubungan antara banyaknya data yang ditransmisikan dan ukuran *file data logger* dengan lama waktu transmisi.

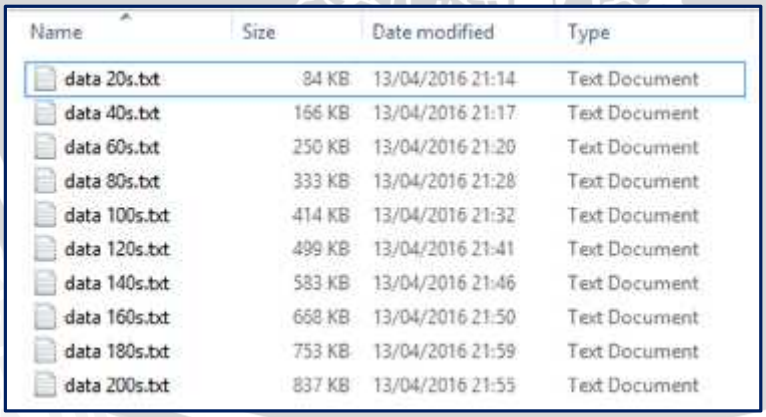
Pada pengujian ini, mikrokontroler berada dalam mode transmisi data. Program visual pun juga berada dalam mode visualisasi *stream data*. Mekanisme penyimpanan *stream data* pada *data logger* diaktifkan. Pada pengujian ini, *baudrate* diatur memiliki nilai 38400, sesuai dengan perancangan. Ketika tombol "*start*" ditekan dan data mulai diterima oleh komputer, *stopwatch* dinyalakan. Ketika waktu telah menunjukkan waktu

20 detik, program visual dihentikan sehingga komputer berhenti menerima *stream data* dari mikrokontroler. Setelah itu, banyaknya paket data yang diterima oleh komputer beserta ukuran *data logger* dicatat. Kegiatan ini diulang untuk waktu-waktu kelipatan 20 detik. Lama waktu transmisi terbesar yang dilakukan dalam pengujian ini adalah 3 menit atau 180 detik.

Jumlah paket data yang diterima oleh komputer dapat dilihat melalui tampilan visual *stream data*. Indeks paket data terbesar menunjukkan jumlah paket data yang telah diterima dan telah tersimpan dalam *data logger file*. Contoh tampilan visual untuk tampilan ini ditunjukkan dalam Gambar 4.8. Adapun ukuran *data logger file* dilihat melalui windows explorer komputer. *Screen shoot* dari ukuran *data logger file* ditunjukkan dalam Gambar 4.9.



Gambar 4.8 Banyaknya *Data Logger File* Dilihat dari Indeks Terbesar

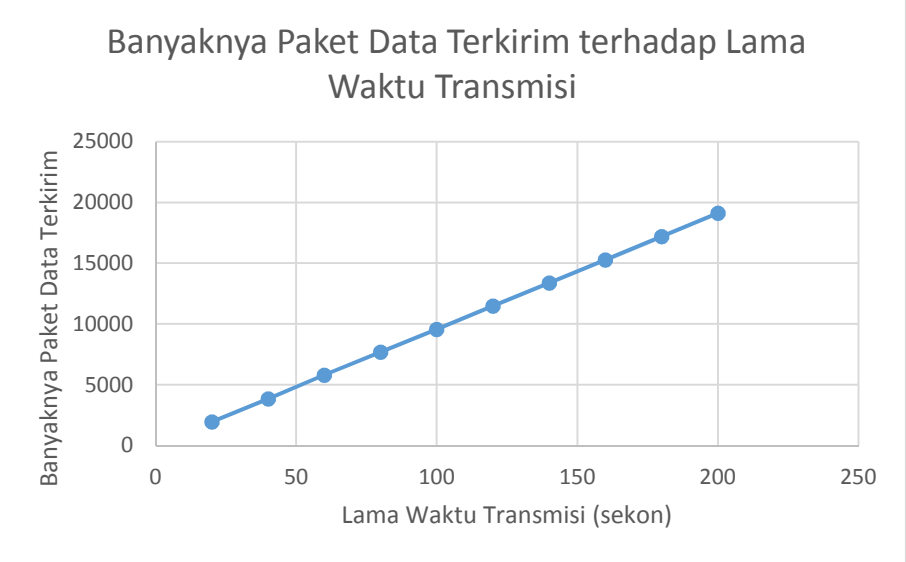


Gambar 4.9 Ukuran *Data Logger File* Dapat Dilihat pada Windows Explorer

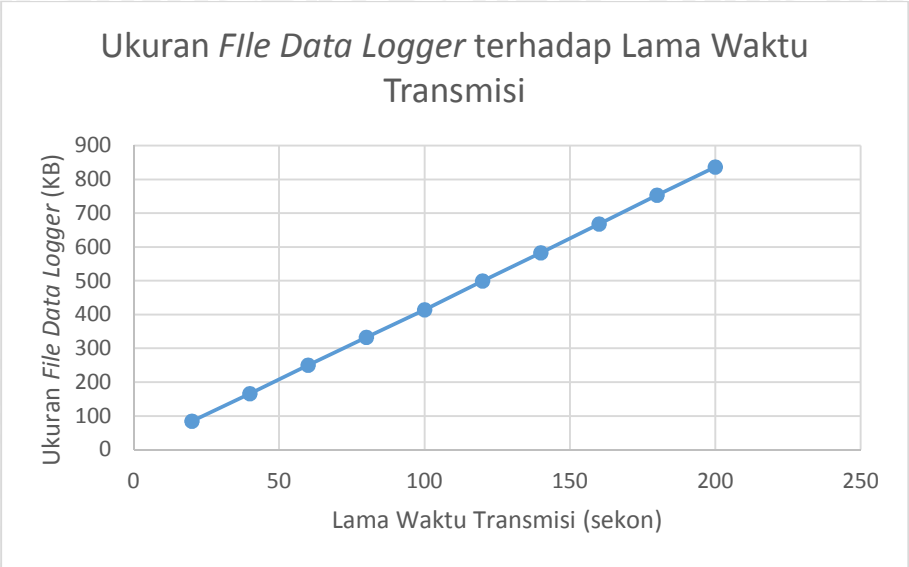
Data hasil pengujian ukuran *file data logger* ditunjukkan dalam Tabel 4.8, sedangkan grafiknya ditunjukkan dalam Gambar 4.10 dan Gambar 4.11.

Tabel 4.8 Data Hasil Pengujian Ukuran *File Data Logger* terhadap Lama Waktu Transmisi.

Lama Waktu Transmisi (sekon)	Banyaknya Paket Data	Ukuran *File Data Logger* (KB)	Kecepatan pengiriman data (paket data / sekon)	Kenaikan ukuran *File Data Logger* (KB / sekon)
20	1948	84	97,4	4,24
40	3846	166	96,15	4,15
60	5786	250	96,43	4,12
80	7688	333	96,1	4,16
100	9565	414	95,65	4,14
120	11488	499	95,73	4,16
140	13384	583	95,6	4,16
160	15288	668	95,55	4,18
180	17207	753	95,59	4,18
200	19110	837	95,55	4,19
Rata-Rata			95,98	4,17



Gambar 4.10 Hubungan antara Jumlah Paket Data Terkirim terdapat Lama Waktu Transmisi



Gambar 4.11 Hubungan antara Ukuran *File Data Logger* terhadap Lama Waktu Transmisi

Berdasarkan Tabel 4.8, terlihat bahwa baik banyaknya paket data terkirim maupun ukuran *file data logger* memiliki hubungan yang linier dengan lamanya waktu transmisi. Rata-rata laju kenaikan ukuran *file data logger* adalah 4,17 KB per detik. Pada umumnya, lomba *line follower robot* maksimal berlangsung selama 3 menit atau 180 detik. Oleh karena itu, jika sistem visualisasi ini diterapkan untuk kegiatan perlombaan, maka akan dihasilkan *data logger file* sebesar 750,6 KB.

Laju pengiriman paket data sebesar 95,98 paket data per detik berarti waktu yang dibutuhkan untuk mengirim satu paket data adalah 10,42 ms per paket data. Artinya, respons sistem visualisasi sensor garis dalam membaca satu buah data adalah 10,42 ms. Dalam perancangan, telah dihitung bahwa waktu maksimum pengiriman data adalah 12,5 ms per paket data. Terlihat bahwa nilai pada pengaplikasian lebih pendek dari pada perhitungan. Hal ini disebabkan karena pada perhitungan, nilai ADC diasumsikan selalu maksimal (mencapai ratusan dan terdiri atas 3 digit angka). Sedangkan pada pengaplikasiannya, sensor lebih banyak berada pada permukaan putih sehingga banyak nilai ADCnya yang hanya mencapai nilai puluhan (2 digit angka).

4.6. Pengujian Akurasi Pembacaan Sistem Visualisasi Sensor Garis

Tujuan dari pengujian ini adalah untuk mengetahui tingkat akurasi dari sensor yang telah dirancang. Data yang digunakan dalam pengujian ini ada dua buah. Data pertama adalah sebuah lintasan yang memiliki garis hitam dan *background* yang berwarna putih. Data kedua adalah lintasan dengan garis putih dan *background* berwarna hitam. Untuk

masing-masing data, sensor diuji cobakan pada 20 tempat yang berbeda di lintasan. Hasil tiap pengujian kemudian dicatat dan dibandingkan dengan data yang seharusnya.

Tabel 4.9 dan Tabel 4.10 menunjukkan hasil pengujian akurasi pembacaan sistem visualisasi sensor garis.

Tabel 4.9 Hasil Pengujian untuk Lintasan dengan Garis Hitam dan *Background* Putih[*)]

	pola acuan	P	P	P	P	P	H	H	P	P	P	P	P	status
Percobaan ke -	1	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	2	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	3	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	4	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	5	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	6	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	7	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	8	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	9	P	P	P	P	P	M	H	P	P	P	P	P	SALAH
	10	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	11	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	12	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	13	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	14	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	15	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	16	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	17	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	18	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	19	P	P	P	P	P	H	H	P	P	P	P	P	BENAR
	20	P	P	P	P	P	H	H	P	P	P	P	P	BENAR

Berdasarkan Tabel 4.9, dapat diketahui bahwa dari 20 kali pengambilan data, satu buah data mengalami ketidaksesuaian dengan pola acuan yang seharusnya. Oleh karena itu, tingkat akurasi sistem visualisasi sensor dengan garis hitam dapat dihitung sebagai berikut:

$$akurasi\ (\%) = \frac{jumlahDataBenar}{totalData} \times 100\% = \frac{19}{20} \times 100\% = 95\%$$

Sedangkan berdasarkan Tabel 4.10, dapat diketahui bahwa dari 20 kali pengambilan data, dua buah data mengalami ketidaksesuaian dengan pola acuan yang seharusnya. Oleh karena itu, tingkat akurasi sistem visualisasi sensor dengan garis putih dapat dihitung sebagai berikut:

$$akurasi\ (\%) = \frac{jumlahDataBenar}{totalData} \times 100\% = \frac{18}{20} \times 100\% = 90\%$$

Tabel 4.10 Hasil Pengujian untuk Lintasan dengan Garis Putih dan *Background* Hitam*)

pola acuan		H	H	H	H	H	P	P	H	H	H	H	H	status
Percobaan ke -	1	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	2	H	H	H	M	H	P	P	M	H	H	H	H	SALAH
	3	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	4	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	5	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	6	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	7	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	8	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	9	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	10	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	11	H	H	H	H	M	P	P	H	H	H	H	H	SALAH
	12	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	13	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	14	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	15	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	16	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	17	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	18	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	19	H	H	H	H	H	P	P	H	H	H	H	H	BENAR
	20	H	H	H	H	H	P	P	H	H	H	H	H	BENAR

*)H=hitam, P=putih, M=merah (daerah terlarang)

Pada garis hitam, sistem visualisasi sensor memiliki tingkat akurasi sebesar 95%. Sedangkan pada garis putih, sistem visualisasi memiliki tingkat akurasi sebesar 90%. Hal ini menunjukkan bahwa sistem telah dapat menjalankan tugas visualisasi sensor garis dengan baik.

4.7. Pengujian Presisi Pembacaan Sistem Visualisasi Sensor Garis

Tujuan dari pengujian ini adalah untuk mengetahui tingkat presisi dari masing-masing unit rangkaian sensor cahaya TCRT5000 yang telah dirancang. Besaran yang digunakan untuk mengukur tingkat presisi dalam pengujian ini adalah *coefficient of variation*.

Gambar 4.12 menunjukkan lintasan yang digunakan untuk pengujian. Percobaan dilakukan sebanyak dua kali, pertama percobaan untuk *background* putih dan yang kedua untuk *background* hitam. Pada masing-masing percobaan, data diambil pada 5 tempat yang berbeda di lintasan, seperti yang ditunjukkan pada Gambar 4.12. Setiap sensor diletakkan pada satu tempat di lintasan, dua belas data ADC yang berasal dari masing-

masing rangkaian sensor cahaya dicatat. Kemudian, nilai koefisien variasi dari masing-masing sensor dihitung. Langkah-langkah ini kemudian diulangi untuk lintasan dengan *background* hitam.

Gambar 4.12 Lintasan untuk Pengujian Presisi

Tabel 4.11 dan Tabel 4.12 berturut-turut menunjukkan hasil pengujian presisi sistem visualisasi pembacaan sensor garis.

Tabel 4.11 Hasil Pengujian Presisi untuk Dasar Lintasan Berwarna Putih

		nilai ADC pada percobaan ke -					Standar Deviasi	*Coefficient of variation (%)*
		1	2	3	4	5		
sensor ke -	0	41	40	40	40	41	0,54772	1,35575
	1	41	40	41	40	40	0,54772	1,35575
	2	21	21	22	21	21	0,44721	2,10949
	3	20	20	20	19	19	0,54772	2,79450
	4	40	39	40	39	39	0,54772	1,39016
	5	41	41	41	41	41	0	0
	6	41	41	41	40	41	0,44721	1,09611
	7	21	21	22	21	22	0,54772	2,55945
	8	40	39	41	39	39	0,89443	2,25866
	9	22	21	22	21	21	0,54772	2,55945
	10	39	39	39	39	39	0	0
	11	22	21	21	21	22	0,54772	2,55945

Berdasarkan Tabel 4.11 dan Tabel 4.12, terlihat bahwa tiap unit sensor memiliki tingkat presisi yang berbeda-beda satu dengan yang lain. Selain itu, tiap unit sensor memiliki tingkat presisi yang berbeda ketika diletakkan pada permukaan hitam dibandingkan dengan ketika diletakkan pada permukaan putih.

Tabel 4.12 Hasil Pengujian Presisi untuk Dasar Lintasan Berwarna Hitam

		nilai ADC pada percobaan ke -					Standar Deviasi	*Coefficient of variation (%)*
		1	2	3	4	5		
sensor ke -	0	185	164	171	170	183	9,01665	5,16418
	1	165	154	148	154	171	9,34345	5,89864
	2	152	137	137	135	145	7,15542	5,06758
	3	157	125	115	143	143	16,57709	12,1355
	4	182	168	150	177	173	12,30853	7,24031
	5	162	160	167	165	164	2,70185	1,65150
	6	187	173	179	176	191	7,56307	4,17388
	7	135	138	142	145	141	3,83406	2,73471
	8	174	191	175	189	191	8,71780	4,73793
	9	145	132	134	134	131	5,63028	4,16441
	10	178	163	168	176	177	6,58027	3,81686
	11	145	129	137	130	144	7,51665	5,48661

Berdasarkan Tabel 4.13 dan Tabel 4.14, terlihat bahwa tiap unit sensor memiliki tingkat presisi yang berbeda-beda satu dengan yang lain. Selain itu, tiap unit sensor memiliki tingkat presisi yang berbeda ketika diletakkan pada permukaan hitam dibandingkan dengan ketika diletakkan pada permukaan putih. Rata-rata nilai *coefficient of variation* untuk permukaan putih adalah 1,67% dan untuk permukaan hitam adalah sebesar 5,19%. Nilai *coefficient of variation* sebesar ini menunjukkan bahwa sensor memberikan hasil pembacaan yang mendekat satu dengan yang lain jika dilakukan pembacaan secara berulang-ulang. Nilai sebesar ini juga sudah cukup baik untuk diterapkan pada sensor garis *line follower robot*.

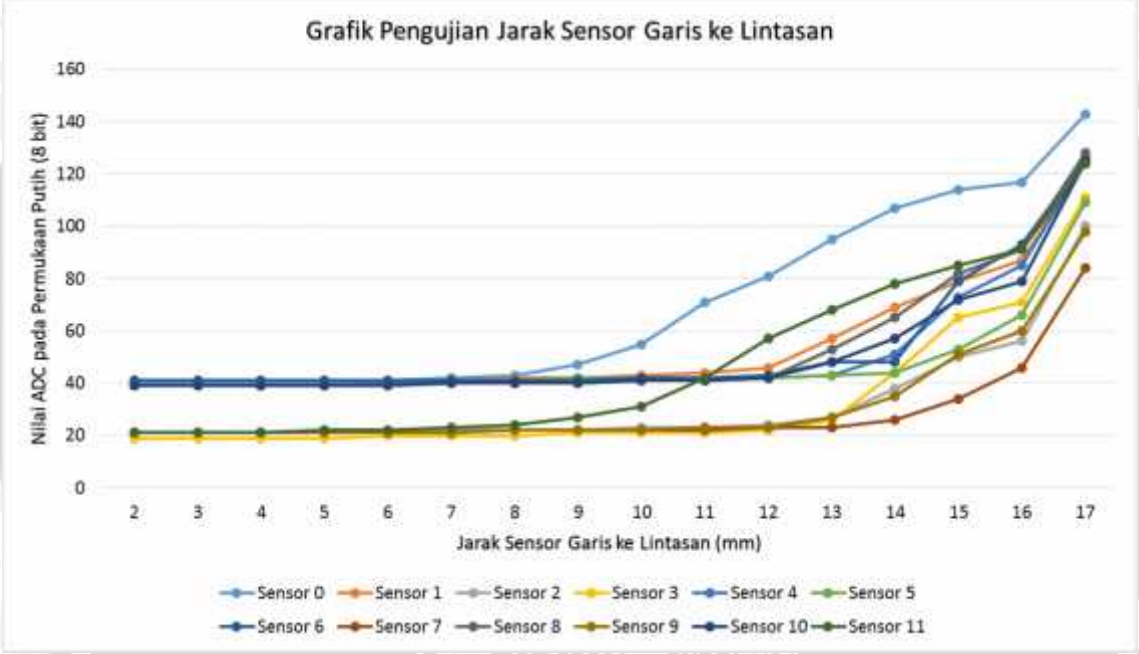
4.8. Pengujian Jarak Sensor Garis ke Lintasan terhadap Tegangan Keluaran Sensor

Pengujian ini bertujuan untuk mengetahui pengaruh jarak vertikal sensor dengan lintasan terhadap *output* tegangan yang dihasilkan.

Pengujian dilakukan di atas lintasan putih untuk semua unit sensor cahaya TCRT5000, tanpa ada pengaruh pengaruh garis hitam. hal ini dapat dijelaskan sebagai berikut. Semakian jauh jarak sensor garis dari lintasan, maka pantulan cahaya yang diterima oleh *receiver* yang berasal dari *transmitter* akan semakin sedikit. Pada suatu jarak tertentu, pancaran gelombang IR dari *transmitter* tidak lagi dapat diterima oleh *receiver*. Pada kondisi ini, *receiver* akan menjadi bersifat isolatif dan menjadi berlogika hitam. Oleh karena itu, efek dari perubahan jarak sensor terhadap lintasan ini dapat diamati hanya pada lintasan dengan permukaan putih.

Pada mulanya, kalibrasi dilakukan untuk mendapat data nilai *treshold* putih untuk masing-masing unit sensor cahaya. Pengujian dilakukan dengan jarak awal sebesar 2 mm, dan kemudian nilai ADC masing-masing keluaran tiap unit sensor cahaya dicatat. Setelah itu, secara berangsur-angsur jarak dinaikkan setiap 1 mm, dan hasil pembacaan ADC kembali dicatat. Pengujian dilakukan sampai suatu jarak tertentu di mana nilai ADC semua unit sensor melebihi nilai *treshold* putihnya. Tiap unit data diambil sebanyak tiga kali dan kemudian dihitung rata-ratanya agar hasilnya lebih valid.

Hasil pengujian jarak sensor garis terhadap lintasan disajikan dalam Tabel 4.13 dan grafiknya ditunjukkan dalam Gambar 4.13.



Gambar 4.13 Grafik Hasil Pengujian Jarak Sensor Garis ke Lintasan terhadap Pembacaan ADC

Dari hasil pengujian, terlihat bahwa bahwa semakin jauh jarak vertikal antara sensor garis dengan lintasan, maka tegangan keluaran unit sensor cahaya TCRT5000 akan semakin tinggi. Hal ini direpresentasikan dengan nilai ADC yang semakin tinggi. Tingkat kenaikan nilai ADC untuk tiap-tiap unit sensor cahaya berbeda-beda. Hal ini dapat dipahami bersama bahwa memang rangkaian sensor cahaya TCRT5000 tidak identik satu dengan yang lain.

Tabel 4.13 Hasil Pengujian pengaruh jarak sensor garis ke lintasan terhadap pembacaan ADC[1)]

Jarak	Sensor 0	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8	Sensor 9	Sensor 10	Sensor 11
2 mm	40/P	41/P	21/P	19/P	39/P	41/P	41/P	21/P	39/P	21/P	39/P	21/P
3 mm	40/P	41/P	21/P	19/P	39/P	41/P	41/P	21/P	39/P	21/P	39/P	21/P
4 mm	40/P	41/P	21/P	19/P	39/P	41/P	41/P	21/P	39/P	21/P	39/P	21/P
5 mm	40/P	41/P	21/P	19/P	39/P	41/P	41/P	21/P	39/P	21/P	39/P	22/P
6 mm	41/P	41/P	21/P	20/P	40/P	41/P	41/P	21/P	39/P	21/P	39/P	22/P
7 mm	42/P	41/P	22/P	20/P	40/P	41/P	41/P	21/P	40/P	21/P	40/P	23/P
8 mm	43/P	42/P	22/P	20/P	40/P	41/P	41/P	22/P	40/P	22/P	40/P	24/P
9 mm	47/P	42/P	22/P	21/P	40/P	42/P	41/P	22/P	40/P	22/P	40/P	27/P
10 mm	55/P	43/P	23/P	21/P	41/P	42/P	42/P	22/P	41/P	22/P	41/P	31/P
11 mm	71/P	44/P	23/P	21/P	41/P	42/P	42/P	22/P	41/P	23/P	41/P	42/P
12 mm	81/M	46/P	24/P	22/P	42/P	42/P	43/P	23/P	42/P	23/P	42/P	57/P
13 mm	95/M	57/P	27/P	26/P	43/P	43/P	48/P	23/P	53/P	27/P	48/P	68/M
14 mm	107/M	69/P	38/P	44/P	51/P	44/P	48/P	26/P	65/P	35/P	57/P	78/M
15 mm	114/M	79/P	50/P	65/M	73/P	53/P	79/P	34/P	82/P	51/P	72/P	85/M
16 mm	117/M	87/M	56/P	71/M	85/P	66/P	93/M	46/P	91/M	60/M	79/P	91/M
17 mm	143/H	125/H	100/M	111/H	125/M	109/M	126/M	84/M	128/M	98/M	125/M	124/H

[1)]H=hitam, P=putih, M=merah

Berdasarkan grafik dalam Gambar 4.13, dapat dilihat bahwa tegangan keluaran sensor (yang direpresentasikan dengan nilai ADC) stabil untuk jarak sensor di bawah 7 mm. Artinya, jarak sensor garis ke lintasan mulai dari 2 mm sampai dengan 7 mm memberikan hasil yang sama dan tidak ada perubahan yang signifikan pada tegangan keluaran sensor. Untuk jarak sebesar 8 mm, mulai terdapat perubahan tegangan keluaran pada sensor 0 dan sensor 11. Sedangkan pada jarak 13 mm, semua *output* tegangan sensor mulai mengalami perubahan.

Berdasarkan Tabel 4.13, terlihat bahwa semua sensor masih mendeteksi permukaan putih hingga jarak sensor garis ke lintasan sebar 11 mm. Ketika jarak terus diperbesar menjadi 12 mm, sensor 0 menjadi berwarna merah. Artinya, nilai ADC sensor 0 menjadi di atas batas nilai *treshold* putih dan masuk pada daerah terlarang. Pada jarak 13 mm, sensor 11 menjadi berwarna merah dan nilai ADCnya berada di atas batas *trehold* nilai putih. Semua unit sensor menjadi semua sensor menjadi tak bekerja ketika jarak sensor ke lintasan sebesar 17 mm.

4.9. Pengujian Keseluruhan Sistem

Pengujian ini dilakukan untuk membuktikan bahwa sub-sub sistem yang telah teruji sebelumnya dapat dirangkai menjadi satu sistem yang utuh dan dapat beroperasi sesuai dengan perencanaan. Parameter dalam pengujian keseluruhan ini adalah sesuainya tampilan visualisasi pembacaan sensor pada monitor komputer berdasarkan posisi sensor garis di arena lomba.

Sistem visualisasi pembacaan sensor ini secara keseluruhan terdiri atas dua buah mode pengopersian, yaitu mode visualisasi *stream data* dan mode visualisasi *datalogger*. Saat berada dalam mode visualisasi *stream data*, semua bagian dari sistem ini dilibatkan. Mulai dari sensor garis, mikrokontroler, tombol dan LED indikator, modul *bluetooth* HC-05, dan program visual, semuanya bekerja dalam mode ini. Langkah-langkah pengujian keseluruhan sistem adalah sebagai berikut : bagian sistem yang diaktifkan terlebih dahulu adalah program visual pada komputer. Pada program visual, semua mode visualisasi (visualisasi *stream data*, visualisasi diagram garis, visualisasi diagram batang, dan visualisasi animasi sensor) ditampilkan serta fitur penyimpanan pada *data logger* diaktifkan. Ketika tombol *start* ditekan, maka komputer akan mempersiapkan sistem komunikasi serial, *file* penyimpanan *data logger*, dan tampilan visual. Setelah ini semua diproses oleh komputer, maka komputer akan menunggu kiriman data dari mikrokontroler.

Langkah selanjutnya adalah mempersiapkan bagian sensor dan mikrokontroler untuk sistem kalibrasi. Seperti pada pengujian sebelumnya mengenai kalibrasi, modul sensor garis digerakkan sedemikian rupa pada lintasan dan kemudian sensor digeser sehingga tiap unit rangkaian TCRT5000 pada *board* sensor garis pernah berada di atas permukaan hitam dan pernah berada di atas pemukaan putih. Setelah itu, sebuah tombol pada rangkaian ditekan untuk mengubah mode sistem dari kalibrasi menjadi mode *streaming* data. Mikrokontroler akan mengirim data hasil pembacaan sensor garis ke komputer dan program di PC akan menampilkan hasil pembacaan ini pada tampilan visual.

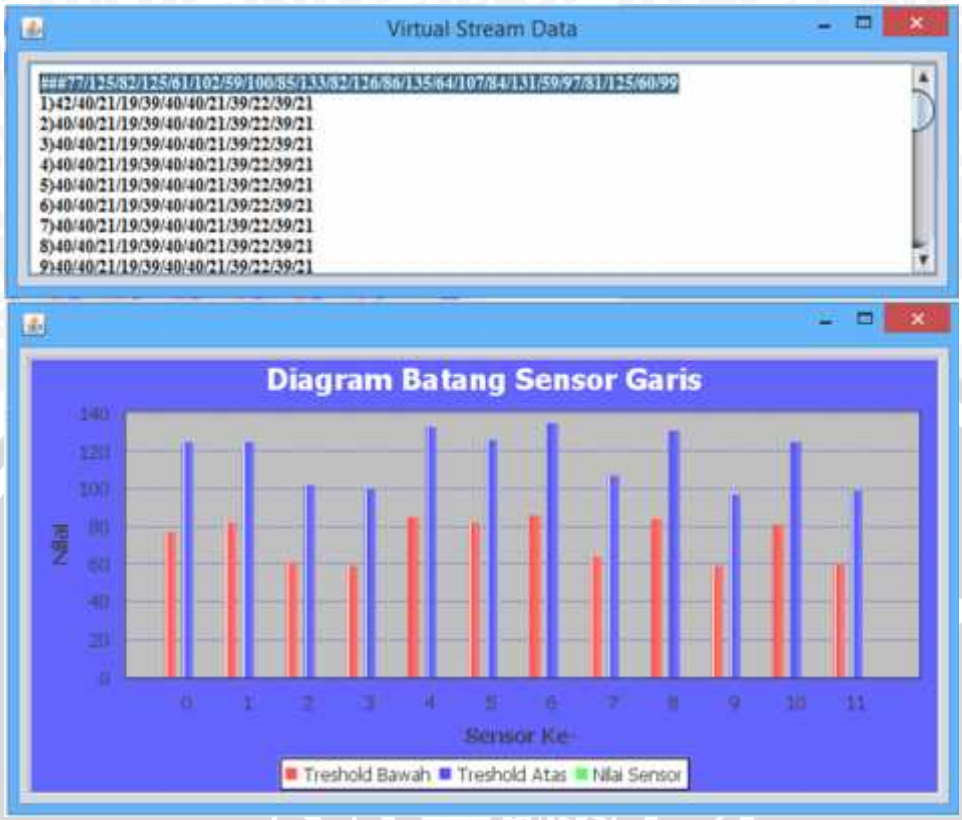
Sensor paling kiri (sensor 0 dan sensor 1) diletakkan di atas garis hitam dan sensor lainnya diletakkan di atas permukaan putih. Tampilan pada program visual komputer kemudian diperhatikan dan di*screenshoot*. Kemudian sensor 1 dan sensor 2 diletakkan di atas garis hitam dan sensor lain diletakkan di atas permukaan putih. Tampilan pada program visual komputer kemudian diperhatikan di*screeshoot*. Langkah ini terus diulang sampai sensor 10 dan sensor 11 yang diletakkan di atas garis hitam dan sensor lain diletakkan di atas permukaan putih.

Saat berada dalam mode visualisasi *data logger*, bagian sistem yang bekerja hanyalah program komputer. Pada pengujian ini, *file data logger* yang digunakan adalah *file* hasil pengujian sebelumnya, yaitu ketika sistem dalam mode visualisasi *stream data*. Parameter keberhasilan dari pengujian ini ada dua. Pertama adalah jika *slider* dijalankan dari kiri ke kanan, maka akan nampak animasi seperti ketika sensor garis digeser ke kanan. Parameter kedua adalah jika keempat model visualisasi secara kompak dapat menunjukkan hasil pembacaan yang sama untuk paket data dengan indeks tertentu.

Terdapat tiga buah data dalam satu unit rangkaian TCRT5000. Data tersebut adalah nilai *treshold* putih, nilai *treshold* hitam, dan nilai hasil pembacaan. Nilai *treshold* didapatkan dari proses kalibrasi. Nilai *treshold* ini juga turut divisualiasikan dalam program komputer, khususnya terdapat dalam visualisasi *stream data* dan diagram batang. Hasil dari visualisasi nilai *treshold* ditunjukkan dalam Gambar 4.14.

Pada tampilan visual dalam panel *stream data*, paket data pertama (yang diawali dengan tiga buah simbol ‘#’) berisi kumpulan nilai-nilai *treshold* sensor dari paling kiri (indeks 0) sampai sensor paling kanan (indeks 11). Pada tampilan visual dalam panel diagram batang, nilai *treshold* putih suatu unit sensor direpresentasikan dalam batang berwarna merah, sedangkan nilai *treshold* hitam suatu unit sensor direpresentasikan dalam batang berwarna biru. Tampak bahwa nilai-nilai *treshold* yang terdapat dalam

diagram batang sama dengan nilai-nilai *treshold* yang terdapat dalam panel *stream data*. Oleh karena itu, visualisasi nilai-nilai *treshold* telah sesuai dengan perencanaan.



Gambar 4.14 Tampilan Visualisasi nilai *Treshold* pada Komputer

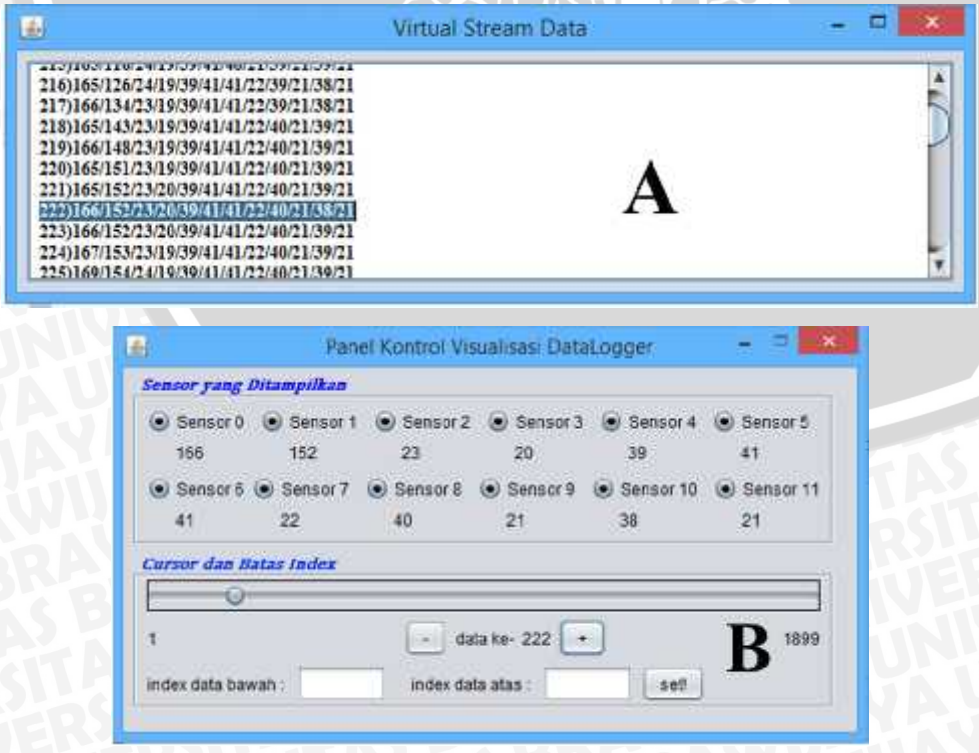
Gambar 4.15 merupakan tampilan program visual yang dihasilkan menggunakan bahasa java. Penjelasan tampilan visual pada Gambar 4.15 dijelaskan sebagai berikut. Gambar 4.15 (a) menampilkan *stream* data yang dikirimkan oleh mikrokontroler ke komputer selama proses transmisi data. Gambar ini mirip dengan *software* virtual terminal seperti parallax atau teraterm. Hanya saja, program visualisasi *stream* data ini didesain khusus untuk visualisasi sensor garis *line follower robot* dengan fasilitas penambahan nomor indeks paket data secara otomatis.

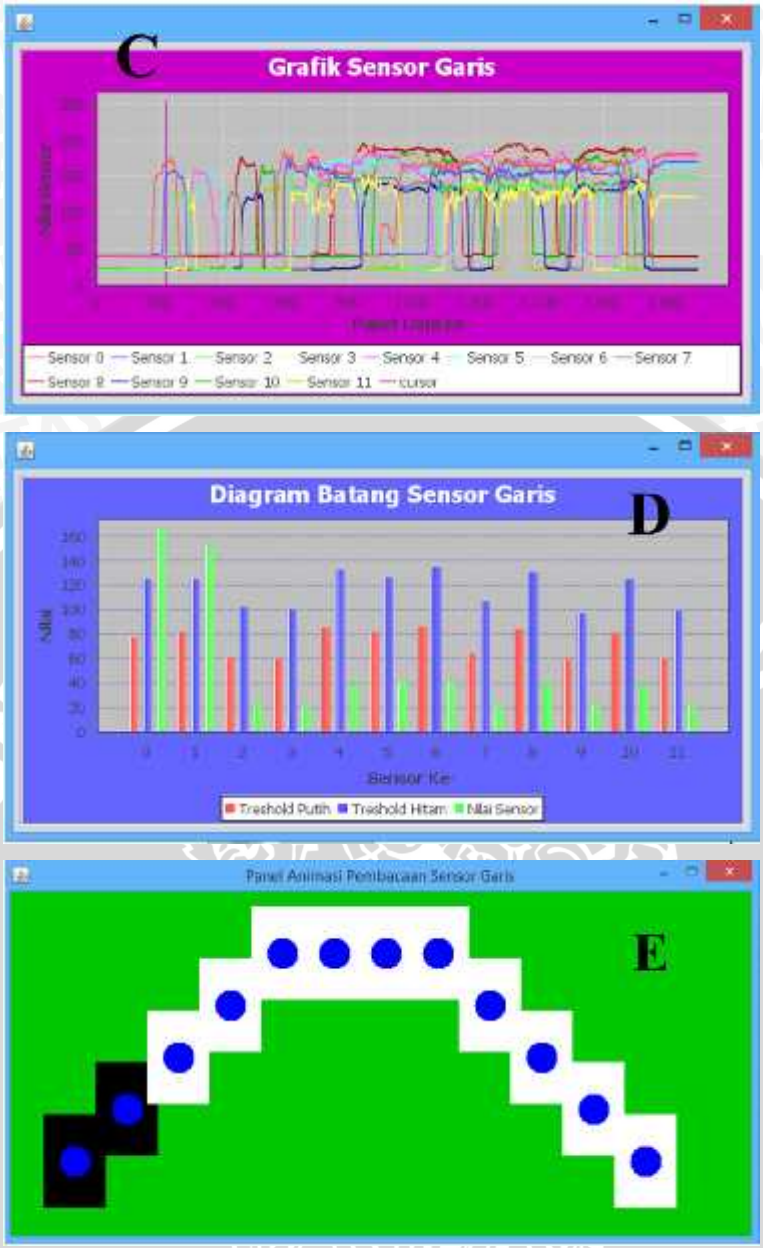
Gambar 4.15 (b) merupakan tampilan panel kontrol sistem visualisasi. Panel kontrol memiliki tiga buah fungsi selama proses visualisasi. Pertama, melalui panel ini *user* dapat memilih sensor berapa saja yang ingin ditampilkan dalam sistem visualisasi. Hal ini dapat dilakukan dengan melakukan klik pada *radio button* pada tiap-tiap sensor. Kedua, terdapat *slider* yang berfungsi untuk mengatur posisi kursor pada tampilan grafik sensor garis. Ketiga, *user* dapat mengatur batas indeks paket data dalam *range* tertentu yang ingin divisualisasikan.

Gambar 4.15 (c) merupakan tampilan grafik visualisasi sensor garis. Dalam grafik, sumbu x merupakan indeks paket data sedangkan sumbu y merupakan nilai ADC. Dalam grafik ini, terdapat 12 kurva sensor garis yang menunjukkan hasil pembacaan mulai awal transmisi sampai akhir transmisi. Terdapat pula kursor yang membantu *user* dalam melakukan pembacaan grafik.

Gambar 4.15 (d) merupakan tampilan diagram batang visualisasi sensor garis. Telah dijelaskan sebelumnya bahwa tiap unit sensor cahaya memiliki tiga buah data, yaitu *treshold* putih, *treshold* hitam, dan nilai ADC yang sedang terbaca. Nilai *treshold* putih direpresentasikan oleh batang vertikal berwarna merah. Nilai *treshold* hitam direpresentasikan oleh batang vertikal berwarna biru. Sedangkan nilai ADC yang sedang terbaca direpresentasikan oleh batang vertikal berwarna hijau.

Gambar 4.15 (e) merupakan tampilan animasi pembacaan sensor garis. Terdapat 12 buah bulatan berwarna biru yang merepresentasikan posisi tiap sensor cahaya. Tiap bulatan biru berada di dalam kotak yang bisa berwarna putih, hitam, maupun merah. Jika suatu sensor dinyatakan berada di atas permukaan putih oleh sistem visualisasi, maka kotak yang melingkupi bulatan biru akan berwarna putih. Jika suatu sensor dinyatakan berada di atas permukaan hitam oleh sistem visualisasi, maka kotak yang melingkupi bulatan biru akan berwarna hitam. Jika nilai suatu sensor dinyatakan berada di antara nilai *treshold* putih dan nilai *treshold* hitam, maka sensor yang bersangkutan dinyatakan berada dalam *forbidden area* dan kotak akan menjadi berwarna merah.





Gambar 4.15 Tampilan Visual Saat Sensor 0 dan Sensor 1 Berada di Atas Permukaan Hitam

Gambar 4.15 juga merupakan hasil tampilan visual saat sensor garis diletakkan sedemikian rupa di atas lintasan sehingga sensor 0 dan sensor 1 berada di atas garis hitam dan sensor yang lain berada di atas permukaan putih. Sebagai contoh, data memenuhi hal ini adalah data ke 222. Urutan data ini dapat ditampilkan dengan menggeser *slider* pada panel kontrol sehingga indeks paket data yang ditampilkan adalah data ke 222 dan kursor grafik berada pada paket data ke 222, seperti yang ditunjukkan dalam Gambar 4.15 (b) dan (c). Berdasarkan Gambar 4.15 (a) dan (b), sensor 0 memiliki nilai ADC sebesar 166 dan sensor 1 memiliki nilai ADC sebesar 152. Berdasarkan Gambar 4.14, sensor 0 memiliki nilai *treshold* hitam sebesar 125 dan sensor 1 memiliki nilai *treshold* hitam

sebesar 125. Oleh karena nilai ADC yang terbaca lebih besar dari pada nilai *treshold* hitamnya, maka batang "nilai sensor" yang berwarna hijau lebih tinggi dari pada batang "*treshold* hitam" yang berwarna biru. Hal ini ditunjukkan dalam Gambar 4.15 (d). Nilai ADC yang terbaca pada sensor 0 dan sensor 1 lebih tinggi dari pada nilai *treshold* hitamnya, maka kedua sensor ini dinyatakan berada di atas permukaan hitam. Hal ini ditampilkan dalam panel animasi pembacaan sensor garis, seperti yang ditunjukkan dalam Gambar 4.15 (e).

Adapun sensor ke 2 sampai sensor ke 11, semuanya berada di atas permukaan putih. Nilai-nilai ADC yang berbaca relatif rendah, dengan nilai tidak lebih dari 50. Hal ini ditunjukkan dalam Gambar 4.15 (a) dan (b). Nilai-nilai ADC sensor ini lebih kecil dari pada nilai *treshold* putihnya, sehingga batang "nilai sensor" yang berwarna hijau lebih rendah dari pada batang "*treshold* putih" yang berwarna merah, seperti yang ditunjukkan dalam Gambar 4.15 (d). Oleh karena nilai ADCnya lebih rendah dari pada nilai *treshold* putihnya, maka sensor ke 2 sampai sensor ke 11 ini dinyatakan berada di atas permukaan putih, seperti yang ditampilkan dalam panel animasi pembacaan sensor garis dalam Gambar 4.15 (e).

Hasil pengujian ketika sensor lain yang berada di atas garis hitam Kami sajikan pada bagian lampiran di laporan skripsi ini. Berdasarkan gambar-gambar hasil pengujian sistem secara keseluruhan ini, dapat disimpulkan bahwa hasil penelitian skripsi ini telah dapat berjalan sebagaimana yang direncanakan.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil percobaan, dapat ditarik kesimpulan sebagai berikut :

1. Sensor garis yang telah dirancang memiliki karakteristik sebagai berikut : menghasilkan rata-rata tegangan sebesar 2,67 V ketika mendeteksi permukaan hitam dan 160,9 mV ketika mendeteksi permukaan putih; memiliki tingkat akurasi sebesar 95% untuk pendeteksian garis hitam dan 90% untuk pendeteksian garis putih; memiliki tingkat presisi yang cukup baik dengan nilai *coefficient of variation* sebesar 1,67% untuk permukaan putih dan 5,19% untuk pemukaan hitam; dapat bekerja dengan baik untuk jarak antara sensor dengan lintasan sebesar 7 mm atau lebih kecil; memiliki rangkaian antar muka *logic level shifter* yang dapat mengubah level logika HIGH dari 2,98 volt menjadi 4,92 volt.
2. Dari sisi transmisi data, jarak maksimum komunikasi adalah 8 meter untuk posisi pemancar horizontal dan 35 meter untuk posisi vertikal. Laju pengiriman data memiliki hubungan yang linier dengan lama waktu transmisi dengan laju pengiriman sebesar 95,98 paket data/detik. Adapun daya disipasi pada modul *bluetooth* HC-05 memiliki berbanding lurus dengan jarak transmisi antara kedua modul tersebut.

5.2. Saran

Untuk mendukung penelitian ke depan yang lebih baik, maka beberapa saran berikut dapat diterapkan, antara lain :

1. Menggunakan modul komunikasi nirkabel dengan kecepatan pengiriman data yang lebih tinggi.
2. Penelitian mengenai rangkaian filter pada sensor garis sehingga *output* sensor lebih resistan terhadap *noise*.
3. Menampilkan parameter-parameter lain pada *line follower robot* untuk divisualkan, seperti kecepatan motor, sistem kontrol, dan lain-lain.

DAFTAR PUSTAKA

Agrawal, A., Agrawal, P., Gupta, K. N., Meghani, H. 2008. *Line Follower Robot*. Tamil Nadu : EEE Department of NIT Trichy.

Boylestad, Robert dan Nashelsky, Louis. 1987. *Electronics Devices and Circuit Theory*. Ohio : Prentice Hall.

Chapra, Steven C dan Canale, Raymod P. 2010. *Numerical Methods for Engineers Sixth Edition*. New York : Mc-GrawHill Companies.

Fairchild Semiconductor. 1995. *2N7000/2N7002.NDS7002A N-Channel Enhancement Mode Field Effect Transistor*. Sunnyvale : Fairchild Semiconductor.

HME FT-UB. 2013. *Panduan Lomba Line Follower Essentials.* Malang : HME FT-UB.

ITead Studio. 2010. *HC-05 Bluetooth to Serial Port Module*. Shenzhen : ITead Studio.

Jatmika, Yusep Nur. 2011. *Cara Mudah Merakit Robot.* Yogyakarta : FlashBooks.

Patil, Priyank. 2005. *Line Following Robot*. Mumbai : K.J. Somaiya College of Engineering.

Prakoso, Bambang Dwi. 2004. *Perancangan dan Analisis Perbandingan Posisi Sensor Garis pada Robot Management Sampah*. Malang : Universitas Brawijaya.

Schildt, Herbert. 2007. *The Complete Reference Java Seventh Edition*. New York: McGraw-Hill Companies.

Schutte, Herman. 1997. *Bi-Directional Level Shifter for I^2C-Bus and Other System – Application Note*. Eindhoven : Philips Semiconductor.

Soebhakti, Hendawan. 2008. *Membuat Robot Tidak Susah.* Batam: Politeknik Negeri Batam.

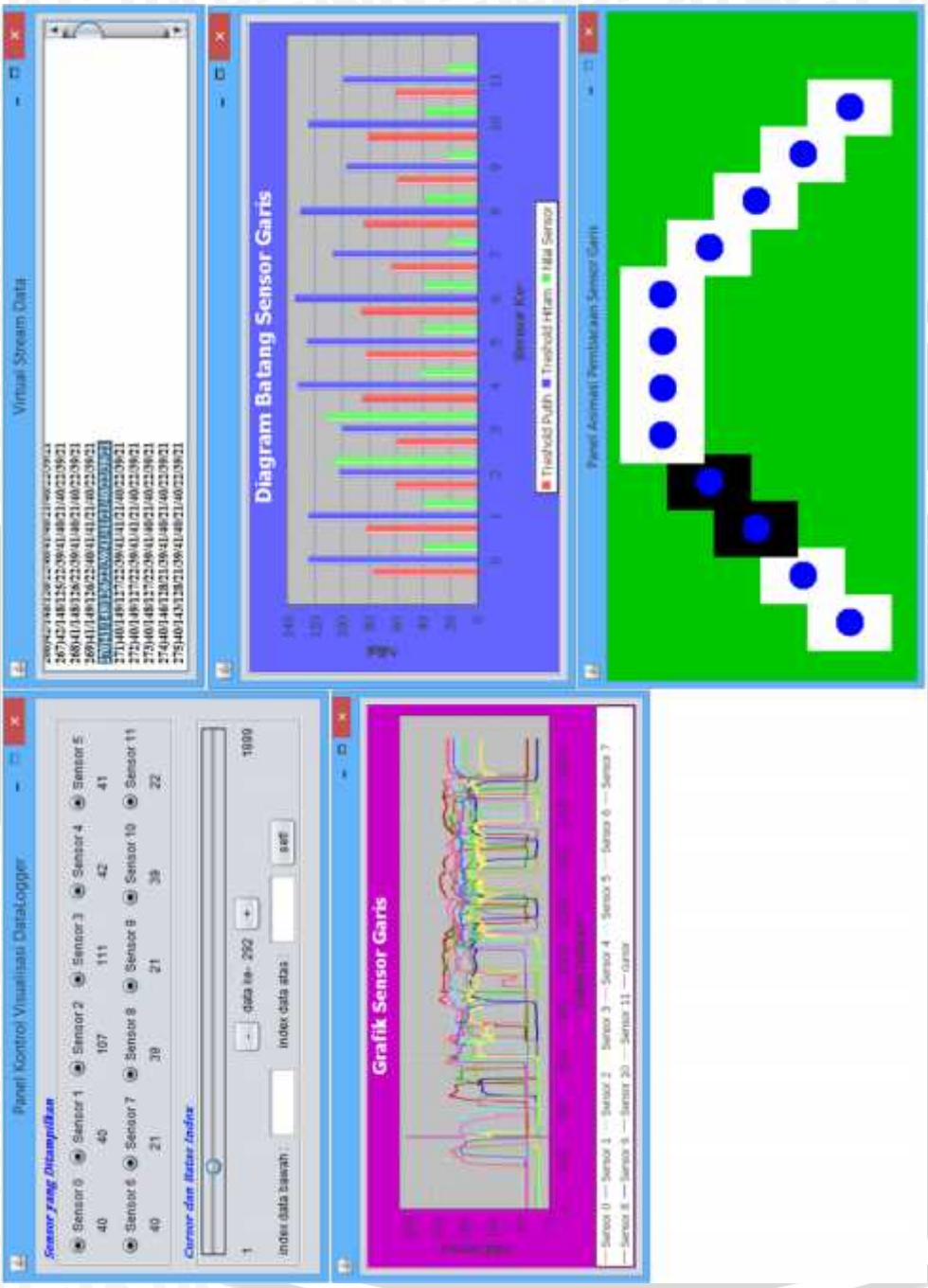
STMicroelectronics. 2011. *RM0041 Reference Manual. STM32F100xx advanced ARM-based 32-bit MCUs*. Geneva : STMicroelectronics.

STMicroelectronics. 2011. *UM0919 User Manual. STM32VLDISCOVERY STM32 value line Discovery*. Geneva : STMicroelectronics.

Texas Instruments. 2003. *CD4051B, CD4052B, CD4053B*. Texas : Texas Instruments.

Vishay. 2009. *Reflective Optical Sensor with Transistor Output*. Malvern:Vishay Semiconductors.

Lampiran 1 Gambar-Gambar Hasil Pengujian Keseluruhan



Gambar 1. Tampilan Visual Saat Sensor ke-2 dan ke-3 Berada di Atas Garis

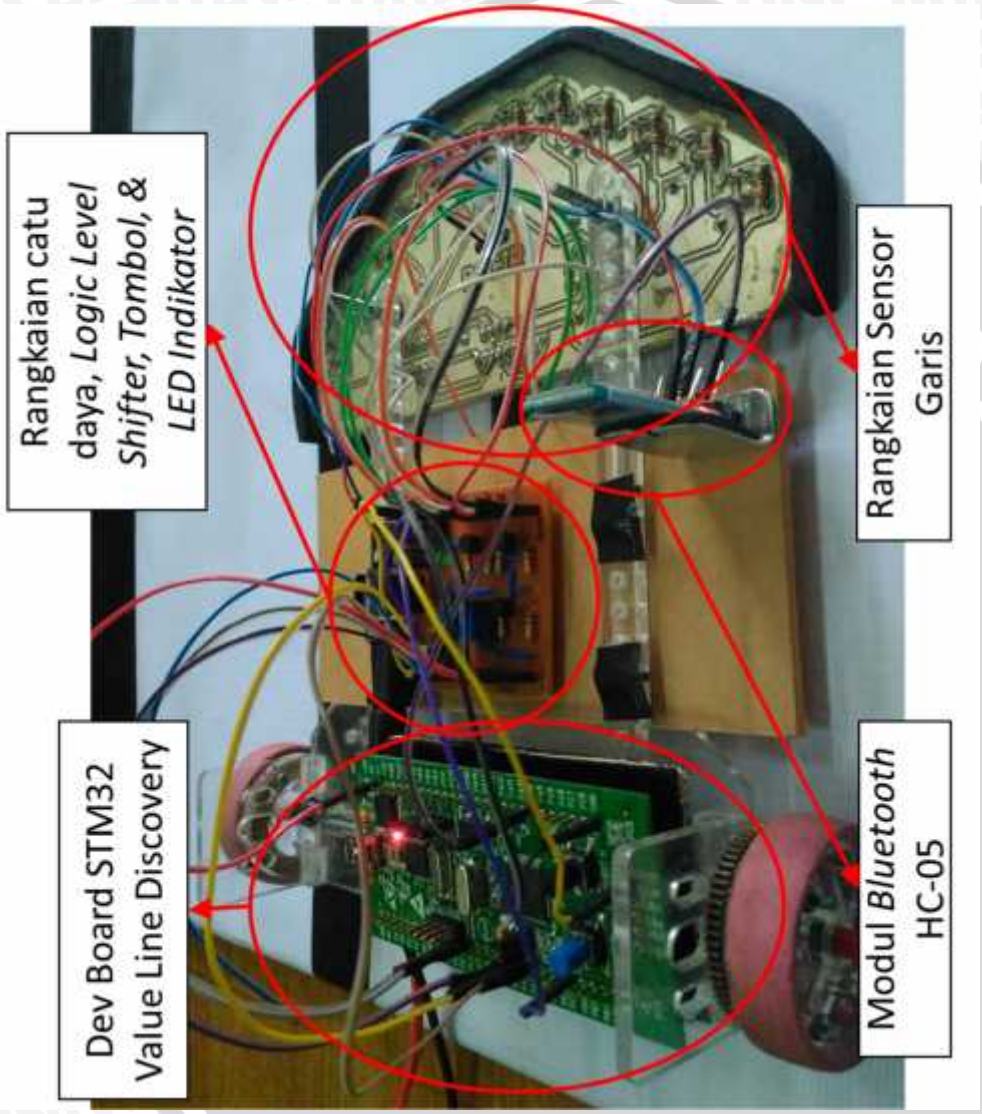
Gambar 2. Tampilan Visual Saat Sensor ke-4 dan ke-5 Berada di Atas Garis

Gambar 3. Tampilan Visual Saat Sensor ke-6 dan ke-7 Berada di Atas Garis

Gambar 4. Tampilan Visual Saat Sensor ke-8 dan ke-9 Berada di Atas Garis

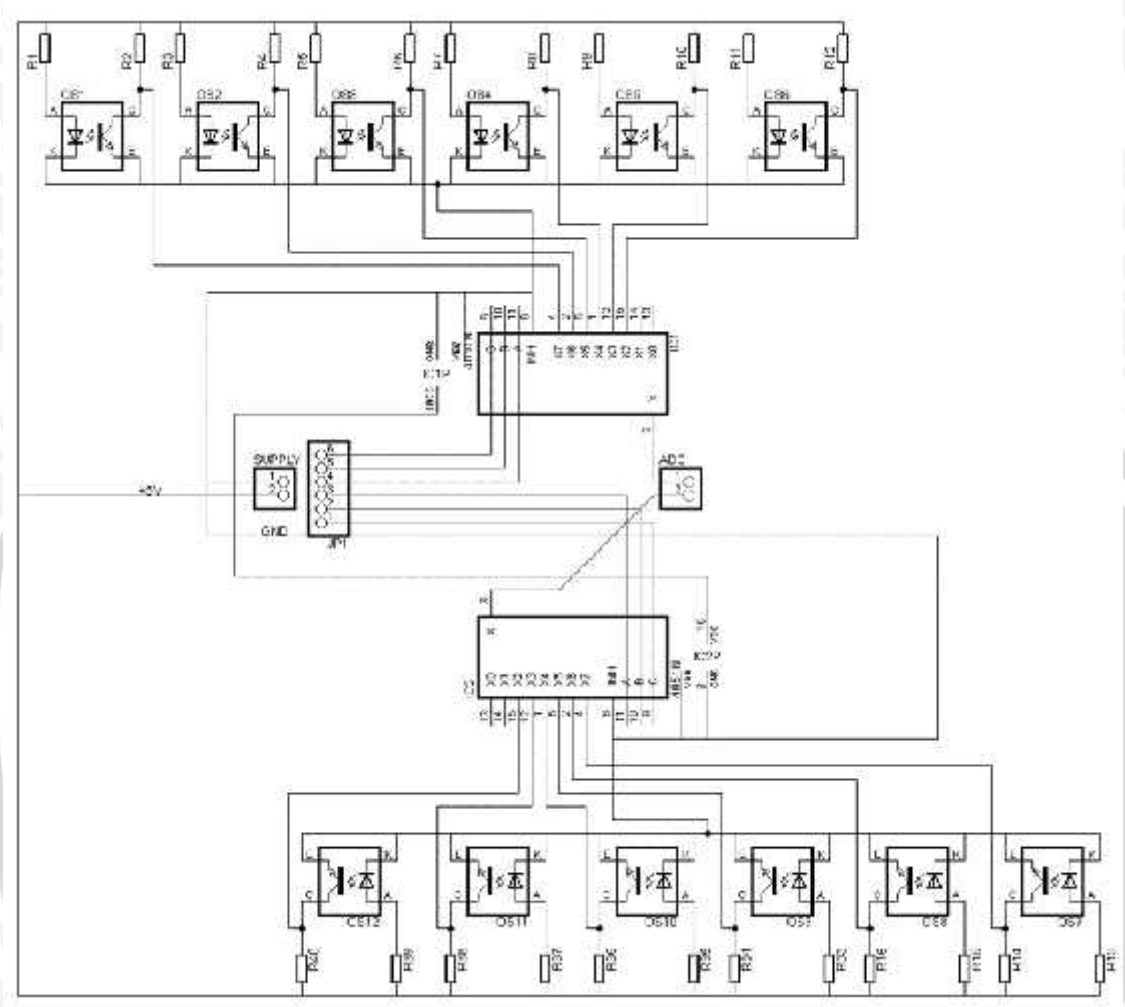
Gambar 5. Tampilan Visual Saat Sensor ke-10 dan ke-11 Berada di Atas **Garis**

Lampiran 2 Dokumentasi Alat

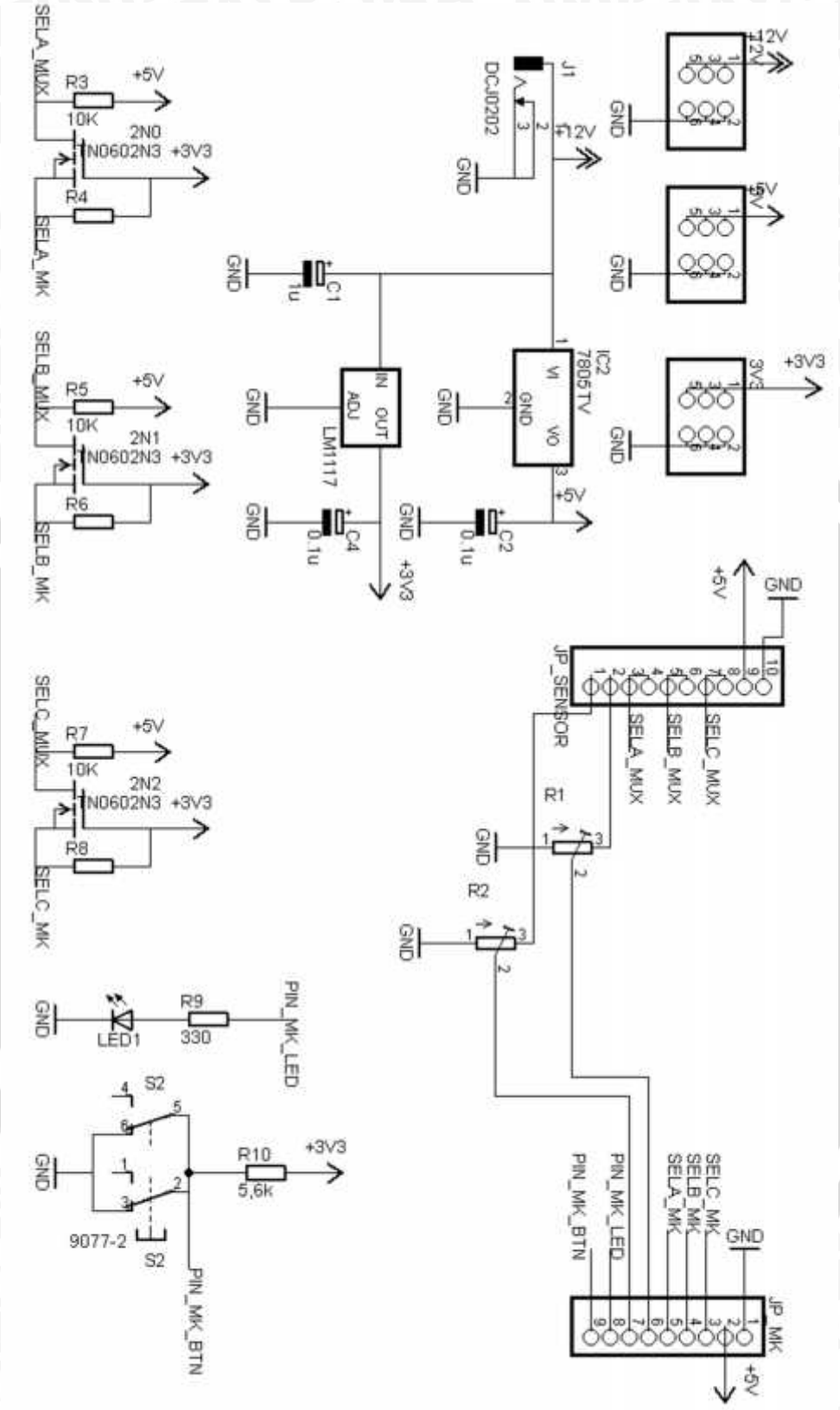


Gambar 6. Sistem Alat secara Keseluruhan

Lampiran 3 Rangkaian Keseluruhan



Gambar 7. Rangkaian Skematik Sensor



Gambar 8. Rangkaian Skematik Catu Daya, *Logic Level Shifter*, Tombol, dan LED Indikator

Lampiran 4 Program Mikrokontroler

File : main.c

```
#include "stm32f10x_myusart.h"
#include "stm32f10x_mydelay.h"
#include "stm32f10x_my_adc.h"
#include "ltc_config.h"
#include <stdio.h>

uint8_t adcSensor[jmlSensor];
char buffer[30];
char bffr[30];

bool sensLogic[jmlSensor];
bool
sensLogicPatterned[jmlSensor];
uint16_t counter;
uint16_t
highTresholdValue[jmlSensor];
uint16_t
lowTresholdValue[jmlSensor];
uint8_t jmlSampling=3;
unsigned char sensKe;
uint32_t sensIndex=1;
uint8_t noiseMarginScaling=3;

void buttonInit(){
	RCC_APB2PeriphClockCmd(RCC
_allButton, ENABLE);

	GPIO_InitTypeDef
GPIO_InitBtn;
	GPIO_InitBtn.GPIO_Mode =
GPIO_Mode_IN_FLOATING;
	GPIO_InitBtn.GPIO_Speed =
GPIO_Speed_10MHz;

	GPIO_InitBtn.GPIO_Pin =
Pin_btnHome;
	GPIO_Init(GPIO_btnHome,
&GPIO_InitBtn);

	GPIO_InitBtn.GPIO_Pin =
Pin_btnOk;
	GPIO_Init(GPIO_btnOk,
&GPIO_InitBtn);

	GPIO_InitBtn.GPIO_Pin =
Pin_btnCncl;
	GPIO_Init(GPIO_btnCncl,
&GPIO_InitBtn);

	GPIO_InitBtn.GPIO_Pin =
Pin_btnUp;
	GPIO_Init(GPIO_btnUp,
&GPIO_InitBtn);

	GPIO_InitBtn.GPIO_Pin =
Pin_btnDown;
	GPIO_Init(GPIO_btnDown,
&GPIO_InitBtn);
}
void ledIndikatorInit(){
	RCC_APB2PeriphClockCmd(RCC
_LED, ENABLE);
	GPIO_InitTypeDef
GPIO_InitStruct;
	GPIO_InitStruct.GPIO_Mode
= GPIO_Mode_Out_PP;
	GPIO_InitStruct.GPIO_Pin =
Pin_LED;
	GPIO_InitStruct.GPIO_Speed
= GPIO_Speed_10MHz;
	GPIO_Init(GPIO_LED,
&GPIO_InitStruct);
}
void selektor_init(){
	RCC_APB2PeriphClockCmd(RCC
_APB2_AllSelector, ENABLE);
	GPIO_InitTypeDef
GPIO_InitStruct;

	GPIO_InitStruct.GPIO_Speed
= GPIO_Speed_10MHz;
	GPIO_InitStruct.GPIO_Mode
= GPIO_Mode_Out_PP;	//semua
pakai push-pull

	GPIO_InitStruct.GPIO_Pin =
GPIO_Pin_Sel0;
	GPIO_Init(GPIO_Sel0,
&GPIO_InitStruct);

	GPIO_InitStruct.GPIO_Pin =
GPIO_Pin_Sel1;
	GPIO_Init(GPIO_Sel1,
&GPIO_InitStruct);

	GPIO_InitStruct.GPIO_Pin =
GPIO_Pin_Sel2;
	GPIO_Init(GPIO_Sel2,
&GPIO_InitStruct);
}
void setSelector(BitAction
Sel2_logic, BitAction
Sel1_logic, BitAction
Sel0_logic){
	GPIO_WriteBit(GPIO_Sel2,
GPIO_Pin_Sel2, Sel2_logic);
	GPIO_WriteBit(GPIO_Sel1,
GPIO_Pin_Sel1, Sel1_logic);
	GPIO_WriteBit(GPIO_Sel0,
GPIO_Pin_Sel0, Sel0_logic);
}
uint8_t readAdcOneSensor(uint8_t
bacaSensorKe){
	uint8_t channelADC;
	switch (bacaSensorKe){
		case 0 : {
```

```
    setSelector(RESET, SET,
RESET); channelADC =
ChAdcLeftSens; break;
          }
          case 1 : {

    setSelector(RESET, SET,
SET); channelADC =
ChAdcLeftSens; break;
          }
          case 2 : {

    setSelector(SET, RESET,
RESET); channelADC =
ChAdcLeftSens; break;
          }
          case 3 : {

    setSelector(SET, RESET,
SET); channelADC =
ChAdcLeftSens; break;
          }
          case 4 : {

    setSelector(SET, SET,
RESET); channelADC =
ChAdcLeftSens; break;
          }
          case 5 : {

    setSelector(SET, SET, SET);
channelADC = ChAdcLeftSens;
break;
          }
          case 6 : {

    setSelector(SET, SET, SET);
channelADC = ChAdcRightSens;
break;
          }
          case 7 : {

    setSelector(SET, SET,
RESET); channelADC =
ChAdcRightSens; break;
          }
          case 8 : {

    setSelector(SET, RESET,
SET);  channelADC =
ChAdcRightSens; break;
          }
          case 9 : {

    setSelector(SET, RESET,
RESET); channelADC =
ChAdcRightSens; break;
          }
          case 10 : {

  setSelector(RESET, SET,
SET); channelADC =
ChAdcRightSens; break;
          }
          case 11 : {

  setSelector(RESET, SET,
RESET); channelADC =
ChAdcRightSens; break;
          }
          case 12 : {
                channelADC =
ChAdcLeftSwingSens; break;
          }
          case 13 : {
                channelADC =
ChAdcRightSwingSens; break;
          }
  }
  TimerDelay(1, us);
//propagation delay
multiplekser
  if (bacaSensorKe >=
jmlSensor) return 0;
  else return
(My_ADC_Read(ADC_SENSOR,
channelADC) / 16);
}
void SetVal_toAllSensor(){
  uint8_t
adcVal_temp[jmlSampling];
  uint16_t kumulatifSensor;
  uint8_t temp_jmlSampling;

  for (sensKe=0;
sensKe<jmlSensor; sensKe++){
          ulangBaca:

          temp_jmlSampling =
jmlSampling-1;
          do{

  adcVal_temp[temp_jmlSamplin
g]=readAdcOneSensor(sensKe);

  }while(temp_jmlSampling--);

//perhitungan rata2//
          temp_jmlSampling =
jmlSampling-1;
          kumulatifSensor=0;
          do{

  kumulatifSensor+=adcVal_tem
p[temp_jmlSampling];

  }while(temp_jmlSampling--);

  adcSensor[sensKe]=kumulatif
Sensor/jmlSampling;
```

```
            if
(adcSensor[sensKe] <
lowTresholdValue[sensKe]){

      sensLogic[sensKe] = 0;
            }
            else if
(adcSensor[sensKe] >
highTresholdValue[sensKe]){

      sensLogic[sensKe] = 1;
            }
            else {

            }
      }
}
void my_utoa(int dataIn, char*
bffr, int radix){
      int temp_dataIn;
      temp_dataIn = dataIn;
      int stringLen;

      stringLen = strlen(bffr);
      do{
            *(bffr+stringLen-
1)='\0';
      }while(stringLen--);

      stringLen=1;
      while
((int)temp_dataIn/radix != 0){
            temp_dataIn =
(int)temp_dataIn/radix;
            stringLen++;
      }

      temp_dataIn = dataIn;
      do{
            *(bffr+stringLen-1)
= (temp_dataIn%radix)+'0';
            if (radix>10){
                  if
((temp_dataIn%radix) > 9)
*(bffr+stringLen-1)+=7;
            }
            temp_dataIn = (int)
temp_dataIn / radix;
      }while(stringLen--);
}

void sensorCalibration(){
      bool isFirstSampling=true;
      uint16_t tempReading;
      uint16_t tempLowValue;
      uint16_t tempHighValue;

      while(btn_enter==isNotPush
){
            if
(isFirstSampling==true){
                  for (sensKe=0;
sensKe<jmlSensor; sensKe++){

      lowTresholdValue[sensKe] =
readAdcOneSensor(sensKe);

      highTresholdValue[sensKe]=
lowTresholdValue[sensKe];
                  }

      isFirstSampling=false;
            }
            else{
                  for(sensKe=0;
sensKe<jmlSensor; sensKe++){

      tempReading=readAdcOneSens
or(sensKe);
                  if
(tempReading<lowTresholdValue[se
nsKe]){

      lowTresholdValue[sensKe]=t
empReading;
                  }

      if(tempReading>highTreshol
dValue[sensKe]){

      highTresholdValue[sensKe]=
tempReading;
                  }
            }
      }
      for (sensKe=0;
sensKe<jmlSensor; sensKe++){
            tempLowValue =
lowTresholdValue[sensKe];
            tempHighValue =
highTresholdValue[sensKe];

      lowTresholdValue[sensKe]=t
empLowValue+(tempHighValue-
tempLowValue)/noiseMarginScaling
;

      highTresholdValue[sensKe]=
tempHighValue-(tempHighValue-
tempLowValue)/noiseMarginScaling
;
      }

      while(btn_enter==isPush);
}


int main(void)
{
```

```
    usart_init(38400);
    selektor_init();
    buttonInit();
    ledIndikatorInit();

    My_ADC_Init(ADC_SENSOR);
    My_ADC_InitChannels(ADC_SENS
OR, ChAdcLeftSwingSens);
    My_ADC_InitChannels(ADC_SENS
OR, ChAdcLeftSens);
    My_ADC_InitChannels(ADC_SENS
OR, ChAdcRightSwingSens);
    My_ADC_InitChannels(ADC_SENS
OR, ChAdcRightSens);

    TimerDelayInit();
    TimerDelay(100, ms);
//bouncing time or capacitor
filling
    GPIO_SetBits(GPIO_LED,
Pin_LED);
    sensorCalibration();
    GPIO_ResetBits(GPIO_LED,
Pin_LED);
    TimerDelay(100, ms);
//bouncing time or capacitor
filling

    USARTSendString("  \r\n");
    USARTSendString("###");
    for (counter = 0;
counter<jmlSensor; counter++){
          if(counter==jmlSensor-
1){
                sprintf(buffer,
  "%d/%d",
  lowTresholdValue[counter],
  highTresholdValue[counter]);
          }
          else{
                sprintf(buffer,
"%d/%d/",
lowTresholdValue[counter],
highTresholdValue[counter]);
          }

  USARTSendString(buffer);
  }
  USARTSendString("\r\n");


  USARTSendString("+++\r\n");
  while(1)
    {
        SetVal_toAllSensor();
        for (counter=0;
counter<jmlSensor;
counter++){

  my_utoa(adcSensor[counter],
bffr, 10);

  USARTSendString(bffr);

  if(counter<jmlSensor-1){

  USARTSendString("/");
          }
        }

  USARTSendString("\r\n");
        sensIndex++;
    }
}
```

Lampiran 5 Program GUI Komputer

```
import com.fazecast.jSerialComm.SerialPort;
import java.awt.Color;
import java.io.File;
import java.util.Scanner;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/*
 * To change this license header, choose
License Headers in Project Properties.
 * To change this template file, choose Tools |
Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Sirojuddin
 */
public class menujuFixVisualSensor1 extends
javax.swing.JFrame {
    private final int defaultBaudRate = 38400;
    public final int nbrOfSensor = 12;

    //-----------Serial Communication----------//
    SerialPort chosenPort;
    int baudRateVal;
    int dataBits;
    int parityBits;

    File absoluteFile = null;
    String strFileName;

    int stopBits;
    SerialPort[] portNames = null;
    ControlPanelFrame
controlPanelFrameStreamData = null;
    ControlPanelFrame
controlPanelFrameReadFile = null;

    /**
     * Creates new form
menujuFixVisualSensor1
     */
    public menujuFixVisualSensor1() {
        initComponents();
        this.setResizable(true);
        this.setLocation(10, 10);

        //tab "terima data"
        cb_notSave.setSelected(true);
        txt_FilePath.setEnabled(false);
        btn_createFile.setEnabled(false);
        txt_fileName.setEnabled(false);

        cb_showBarChartNo.setSelected(true);
      cb_showXYChartNo.setSelected(true);
        cb_showStreamDataNo.setSelected(true);

cb_showAnimatedSensorNo.setSelected(true);
        cb_baudRate.setSelectedItem("38400");
        txt_baudRateCustom.setEnabled(false);
        updateCommPortList();

        //tab "Baca file"
        cb_showFileContentNo.setSelected(true);
        cb2_showXYChartNo.setSelected(true);
        cb2_showBarChartNo.setSelected(true);

cb2_showAnimatedSensorNo.setSelected(true)
;
    }

    private void updateCommPortList(){
        cmbo_portList.removeAllItems();
        portNames = SerialPort.getCommPorts();
        for (SerialPort portName : portNames) {

cmbo_portList.addItem(portName.getSystemP
ortName());
        }
//      for (int i = 0; i<portNames.length; i++){
//
cmbo_portList.addItem(portNames[i].getSyste
mPortName());
//      }
        if (cmbo_portList.getItemCount() == 0)
          cmbo_portList.addItem("<no
COMM>");
    }
    /**
     * This method is called from within the
constructor to initialize the form.
     * WARNING: Do NOT modify this code.
The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">
    private void initComponents() {

        groupSaveFileOrNot = new
javax.swing.ButtonGroup();
        bg_showBarCharOrNot = new
javax.swing.ButtonGroup();
```

```
        bg_showXYChartOrNot = new
javax.swing.ButtonGroup();
        bg_showAnimatedSensorOrNot = new
javax.swing.ButtonGroup();
        jLabel7 = new javax.swing.JLabel();
        bg_showStreamDataOrNot = new
javax.swing.ButtonGroup();
        bg_showFileContentOrNot = new
javax.swing.ButtonGroup();
        bg2_showXYChartOrNot = new
javax.swing.ButtonGroup();
        bg2_showBarChartOrNot = new
javax.swing.ButtonGroup();
        bg2_showAnimatedSensorOrNot = new
javax.swing.ButtonGroup();
        jProgressBar1 = new
javax.swing.JProgressBar();
        jPanel1 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new
javax.swing.JTabbedPane();
        panel_terimaData = new
javax.swing.JPanel();
        panel_barChart = new
javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        cb_showBarChartNo = new
javax.swing.JCheckBox();
        cb_showBarChartYes = new
javax.swing.JCheckBox();
        panel_fileStorage = new
javax.swing.JPanel();
        cb_notSave = new
javax.swing.JCheckBox();
        txt_FilePath = new
javax.swing.JTextField();
        cb_saveCmd = new
javax.swing.JCheckBox();
        jLabel3 = new javax.swing.JLabel();
        btn_createFile = new
javax.swing.JButton();
        jLabel8 = new javax.swing.JLabel();
        txt_fileName = new
javax.swing.JTextField();
        panel_xyChart = new
javax.swing.JPanel();
        cb_showXYChartNo = new
javax.swing.JCheckBox();
        cb_showXYChartYes = new
javax.swing.JCheckBox();
        jLabel5 = new javax.swing.JLabel();
        panel_commSetting = new
javax.swing.JPanel();
        cmbo_portList = new
javax.swing.JComboBox();
        btn_refresh = new javax.swing.JButton();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        cb_baudRate = new
javax.swing.JComboBox();
        jLabel12 = new javax.swing.JLabel();
        txt_baudRateCustom = new
javax.swing.JTextField();
        jLabel13 = new javax.swing.JLabel();
        cb_dataBits = new
javax.swing.JComboBox();
        jLabel14 = new javax.swing.JLabel();
        cb_parity = new
javax.swing.JComboBox();
        jLabel15 = new javax.swing.JLabel();
        cb_stopBits = new
javax.swing.JComboBox();
        btn_start = new javax.swing.JButton();
        panel_streamData = new
javax.swing.JPanel();
        jLabel16 = new javax.swing.JLabel();
        cb_showStreamDataYes = new
javax.swing.JCheckBox();
        cb_showStreamDataNo = new
javax.swing.JCheckBox();
        panel_sensorAnimation = new
javax.swing.JPanel();
        jLabel17 = new javax.swing.JLabel();
        cb_showAnimatedSensorYes = new
javax.swing.JCheckBox();
        cb_showAnimatedSensorNo = new
javax.swing.JCheckBox();
        panel_bacaFile = new
javax.swing.JPanel();
        panel_chooseFile = new
javax.swing.JPanel();
        jLabel6 = new javax.swing.JLabel();
        txt_absoluteFileName = new
javax.swing.JTextField();
        btn_fileLocation = new
javax.swing.JButton();
        panel_fileContent = new
javax.swing.JPanel();
        jLabel18 = new javax.swing.JLabel();
        cb_showFileContentYes = new
javax.swing.JCheckBox();
        cb_showFileContentNo = new
javax.swing.JCheckBox();
        panel2_xyChart = new
javax.swing.JPanel();
        jLabel19 = new javax.swing.JLabel();
        cb2_showXYChartYes = new
javax.swing.JCheckBox();
        cb2_showXYChartNo = new
javax.swing.JCheckBox();
```

```
        panel2_barChart = new
javax.swing.JPanel();
        jLabel20 = new javax.swing.JLabel();
        cb2_showBarChartYes = new
javax.swing.JCheckBox();
        cb2_showBarChartNo = new
javax.swing.JCheckBox();
        panel2_sensorAnimation = new
javax.swing.JPanel();
        jLabel21 = new javax.swing.JLabel();
        cb2_showAnimatedSensorYes = new
javax.swing.JCheckBox();
        cb2_showAnimatedSensorNo = new
javax.swing.JCheckBox();
        btn2_show = new javax.swing.JButton();

        jLabel7.setText("jLabel7");


setDefaultCloseOperation(javax.swing.Windo
wConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(255,
255, 255));


jPanel1.setBorder(javax.swing.BorderFactory.c
reateBevelBorder(javax.swing.border.BevelBor
der.RAISED, new java.awt.Color(0, 204, 51),
new java.awt.Color(0, 0, 204), null, null));

        jLabel2.setFont(new
java.awt.Font("Times New Roman", 1, 14)); //
NOI18N

jLabel2.setHorizontalAlignment(javax.swing.S
wingConstants.CENTER);
        jLabel2.setText("Robot Line Follower");

        jLabel1.setFont(new
java.awt.Font("Times New Roman", 1, 14)); //
NOI18N
        jLabel1.setText("Visualisasi Pembacaan
Sensor Garis");

        javax.swing.GroupLayout jPanel1Layout
= new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGro
up()

.addGroup(jPanel1Layout.createParallelGroup(
```



```
javax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(jPanel1Layout.createSequentialGro
up()
                .addContainerGap()
                .addComponent(jLabel1))

.addGroup(jPanel1Layout.createSequentialGro
up()
                .addGap(54, 54, 54)
                .addComponent(jLabel2)))
            .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGro
up()
            .addContainerGap()
            .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(jLabel2)
            .addContainerGap())
        );


panel_terimaData.setBorder(javax.swing.Borde
rFactory.createEtchedBorder());


panel_barChart.setBorder(javax.swing.BorderF
actory.createEtchedBorder());

        jLabel4.setText("4. Tampilkan BarChart
:");


bg_showBarCharOrNot.add(cb_showBarChart
No);
        cb_showBarChartNo.setText("Tidak");


bg_showBarCharOrNot.add(cb_showBarChart
Yes);
        cb_showBarChartYes.setText("Ya");

        javax.swing.GroupLayout
panel_barChartLayout = new
javax.swing.GroupLayout(panel_barChart);
```

```
panel_barChart.setLayout(panel_barChartLayo
ut);

panel_barChartLayout.setHorizontalGroup(

panel_barChartLayout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_barChartLayout.createSeque
ntialGroup()
            .addContainerGap()
            .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.UNRELATED)

.addComponent(cb_showBarChartYes)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb_showBarChartNo)
            .addContainerGap())
      );
      panel_barChartLayout.setVerticalGroup(

panel_barChartLayout.createParallelGroup(jav
ax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_barChartLayout.createSeque
ntialGroup()
            .addContainerGap()

.addGroup(panel_barChartLayout.createParalle
lGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                  .addComponent(jLabel4)

.addComponent(cb_showBarChartYes)

.addComponent(cb_showBarChartNo))
            .addContainerGap())
      );


panel_fileStorage.setBorder(javax.swing.Borde
rFactory.createEtchedBorder());

      groupSaveFileOrNot.add(cb_notSave);
      cb_notSave.setText("Tidak disimpan");
      cb_notSave.addActionListener(new
java.awt.event.ActionListener() {
          public void
actionPerformed(java.awt.event.ActionEvent
evt) {
              cb_notSaveActionPerformed(evt);
          }
      });

      groupSaveFileOrNot.add(cb_saveCmd);
      cb_saveCmd.setText("Simpan");
      cb_saveCmd.addActionListener(new
java.awt.event.ActionListener() {
          public void
actionPerformed(java.awt.event.ActionEvent
evt) {
              cb_saveCmdActionPerformed(evt);
          }
      });

      jLabel3.setText("1. File Penyimpanan :");

      btn_createFile.setText("lokasi
penyimpanan");
      btn_createFile.addActionListener(new
java.awt.event.ActionListener() {
          public void
actionPerformed(java.awt.event.ActionEvent
evt) {
              btn_createFileActionPerformed(evt);
          }
      });

      jLabel8.setText("Nama File :");

      txt_fileName.addKeyListener(new
java.awt.event.KeyAdapter() {
          public void
keyReleased(java.awt.event.KeyEvent evt) {
              txt_fileNameKeyReleased(evt);
          }
      });

      javax.swing.GroupLayout
panel_fileStorageLayout = new
javax.swing.GroupLayout(panel_fileStorage);

panel_fileStorage.setLayout(panel_fileStorage
Layout);

panel_fileStorageLayout.setHorizontalGroup(

panel_fileStorageLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(panel_fileStorageLayout.createSequ
entialGroup()
            .addContainerGap()

.addGroup(panel_fileStorageLayout.createPara
```

```
llelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                .addComponent(txt_FilePath)

.addGroup(panel_fileStorageLayout.createSequ
entialGroup()

.addGroup(panel_fileStorageLayout.createPara
llelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

.addGroup(panel_fileStorageLayout.createSequ
entialGroup()
                    .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.UNRELATED)

.addComponent(cb_saveCmd)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb_notSave)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(btn_createFile))

.addGroup(panel_fileStorageLayout.createSequ
entialGroup()
                    .addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(txt_fileName,
javax.swing.GroupLayout.PREFERRED_SIZE
, 152,
javax.swing.GroupLayout.PREFERRED_SIZE
)))
                .addGap(0, 0,
Short.MAX_VALUE)))
            .addContainerGap())
    );

panel_fileStorageLayout.setVerticalGroup(

panel_fileStorageLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(panel_fileStorageLayout.createSequ
entialGroup()
            .addContainerGap()

.addGroup(panel_fileStorageLayout.createPara
llelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                .addComponent(jLabel3)
                .addComponent(cb_saveCmd)
                .addComponent(cb_notSave)
                .addComponent(btn_createFile))

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(txt_FilePath,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addGroup(panel_fileStorageLayout.createPara
llelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)
                .addComponent(jLabel8)
                .addComponent(txt_fileName,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
))

.addContainerGap(javax.swing.GroupLayout.D
EFAULT_SIZE, Short.MAX_VALUE))
        );


panel_xyChart.setBorder(javax.swing.BorderF
actory.createEtchedBorder());


bg_showXYChartOrNot.add(cb_showXYChart
No);
        cb_showXYChartNo.setText("Tidak");


bg_showXYChartOrNot.add(cb_showXYChart
Yes);
        cb_showXYChartYes.setText("Ya");

        jLabel5.setText("3. Tampilkan Grafik :");

        javax.swing.GroupLayout
panel_xyChartLayout = new
javax.swing.GroupLayout(panel_xyChart);

panel_xyChart.setLayout(panel_xyChartLayout
);
```

```
panel_xyChartLayout.setHorizontalGroup(

panel_xyChartLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_xyChartLayout.createSequen
tialGroup()
            .addContainerGap()
            .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb_showXYChartYes)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb_showXYChartNo)
            .addContainerGap())
        );
        panel_xyChartLayout.setVerticalGroup(

panel_xyChartLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_xyChartLayout.createSequen
tialGroup()
            .addContainerGap()

.addGroup(panel_xyChartLayout.createParallel
Group(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(jLabel5)

.addComponent(cb_showXYChartYes)

.addComponent(cb_showXYChartNo))
            .addContainerGap())
        );


panel_commSetting.setBorder(javax.swing.Bor
derFactory.createEtchedBorder());

        btn_refresh.setText("Refresh");
        btn_refresh.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent
evt) {
                btn_refreshActionPerformed(evt);
            }
        });

        jLabel9.setText("6. Mode Komunikasi :");

        jLabel10.setText("- COMM PORT :");

        jLabel11.setText("- BaudRate :");

        cb_baudRate.setModel(new
javax.swing.DefaultComboBoxModel(new
String[] { "600", "1200", "2400", "4800",
"9600", "14400", "19200", "28800", "38400",
"56000", "57600", "115200", "128000",
"256000", "custom..." }));
        cb_baudRate.addPopupMenuListener(new
javax.swing.event.PopupMenuListener() {
            public void
popupMenuCanceled(javax.swing.event.Popup
MenuEvent evt) {
            }
            public void
popupMenuWillBecomeInvisible(javax.swing.
event.PopupMenuEvent evt) {

cb_baudRatePopupMenuWillBecomeInvisible(
evt);
            }
            public void
popupMenuWillBecomeVisible(javax.swing.ev
ent.PopupMenuEvent evt) {
            }
        });

        jLabel12.setText("(custom)");

        jLabel13.setText("- Data bits :");

        cb_dataBits.setModel(new
javax.swing.DefaultComboBoxModel(new
String[] { "8", "7", "6", "5" }));

        jLabel14.setText("- Parity :");

        cb_parity.setModel(new
javax.swing.DefaultComboBoxModel(new
String[] { "none", "even", "odd", "mark",
"space" }));

        jLabel15.setText("- Stop bits :");

        cb_stopBits.setModel(new
javax.swing.DefaultComboBoxModel(new
String[] { "1", "1.5", "2" }));

        javax.swing.GroupLayout
panel_commSettingLayout = new
javax.swing.GroupLayout(panel_commSetting)
;
```

```
panel_commSetting.setLayout(panel_commSettingLayout);

panel_commSettingLayout.setHorizontalGroup(

panel_commSettingLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_commSettingLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(panel_commSettingLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel9)

.addGroup(panel_commSettingLayout.createSequentialGroup()
                .addComponent(jLabel10)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(cmbo_portList, javax.swing.GroupLayout.PREFERRED_SIZE, 117, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(btn_refresh))

.addGroup(panel_commSettingLayout.createSequentialGroup()
                .addComponent(jLabel13)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(cb_dataBits, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel14)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(cb_parity, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel15)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(cb_stopBits, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(panel_commSettingLayout.createSequentialGroup()
                .addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(cb_baudRate, javax.swing.GroupLayout.PREFERRED_SIZE, 90, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(txt_baudRateCustom, javax.swing.GroupLayout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel12)))
            .addContainerGap(46, Short.MAX_VALUE))
        );

panel_commSettingLayout.setVerticalGroup(

panel_commSettingLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_commSettingLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel9)
```

```
.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addGroup(panel_commSettingLayout.createPa
rallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(jLabel10)
                .addComponent(cmbo_portList,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                .addComponent(btn_refresh))

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addGroup(panel_commSettingLayout.createPa
rallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(jLabel11)
                .addComponent(cb_baudRate,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(txt_baudRateCustom,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                .addComponent(jLabel12))

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addGroup(panel_commSettingLayout.createPa
rallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(jLabel13)
                .addComponent(cb_dataBits,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                .addComponent(jLabel14)
                .addComponent(cb_parity,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                .addComponent(jLabel15)
                .addComponent(cb_stopBits,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
))
                .addContainerGap(12,
Short.MAX_VALUE))
        );

        btn_start.setText("Mulai");
        btn_start.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent
evt) {
                btn_startActionPerformed(evt);
            }
        });


panel_streamData.setBorder(javax.swing.Bord
erFactory.createEtchedBorder());

        jLabel16.setText("2. Tampilkan Stream
Data : ");

bg_showStreamDataOrNot.add(cb_showStrea
mDataYes);
        cb_showStreamDataYes.setText("Ya");

bg_showStreamDataOrNot.add(cb_showStrea
mDataNo);
        cb_showStreamDataNo.setText("Tidak");

        javax.swing.GroupLayout
panel_streamDataLayout = new
javax.swing.GroupLayout(panel_streamData);

panel_streamData.setLayout(panel_streamData
Layout);

panel_streamDataLayout.setHorizontalGroup(

panel_streamDataLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(panel_streamDataLayout.createSeq
uentialGroup()
                .addContainerGap()
                .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb_showStreamDataYes)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb_showStreamDataNo)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

panel_streamDataLayout.setVerticalGroup(

panel_streamDataLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_streamDataLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panel_streamDataLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel16)

.addComponent(cb_showStreamDataYes)

.addComponent(cb_showStreamDataNo))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );


panel_sensorAnimation.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jLabel17.setText("5. Tampilkan Animasi Sensor : ");


bg_showAnimatedSensorOrNot.add(cb_showAnimatedSensorYes);

cb_showAnimatedSensorYes.setText("Ya");


bg_showAnimatedSensorOrNot.add(cb_showAnimatedSensorNo);

cb_showAnimatedSensorNo.setText("Tidak");

        javax.swing.GroupLayout panel_sensorAnimationLayout = new javax.swing.GroupLayout(panel_sensorAnimation);

panel_sensorAnimation.setLayout(panel_sensorAnimationLayout);

panel_sensorAnimationLayout.setHorizontalGroup(

panel_sensorAnimationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_sensorAnimationLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel17)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb_showAnimatedSensorYes)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb_showAnimatedSensorNo)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

panel_sensorAnimationLayout.setVerticalGroup(

panel_sensorAnimationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_sensorAnimationLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panel_sensorAnimationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel17)

.addComponent(cb_showAnimatedSensorYes)

.addComponent(cb_showAnimatedSensorNo))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
```

```
        javax.swing.GroupLayout
panel_terimaDataLayout = new
javax.swing.GroupLayout(panel_terimaData);

panel_terimaData.setLayout(panel_terimaData
Layout);

panel_terimaDataLayout.setHorizontalGroup(

panel_terimaDataLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(panel_terimaDataLayout.createSeq
uentialGroup()
            .addContainerGap()

.addGroup(panel_terimaDataLayout.createPara
llelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

.addGroup(panel_terimaDataLayout.createSeq
uentialGroup()

.addComponent(panel_fileStorage,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addContainerGap())

.addGroup(panel_terimaDataLayout.createSeq
uentialGroup()

.addGroup(panel_terimaDataLayout.createPara
llelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

.addComponent(panel_commSetting,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                .addComponent(btn_start,
javax.swing.GroupLayout.PREFERRED_SIZE
, 86,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel_streamData,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel_sensorAnimation,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel_xyChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel_barChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
))
                .addGap(0, 0,
Short.MAX_VALUE))))
        );

panel_terimaDataLayout.setVerticalGroup(

panel_terimaDataLayout.createParallelGroup(j
avax.swing.GroupLayout.Alignment.LEADIN
G)

.addGroup(panel_terimaDataLayout.createSeq
uentialGroup()
            .addContainerGap()
            .addComponent(panel_fileStorage,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel_streamData,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel_xyChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel_barChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(panel_sensorAnimation,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel_commSetting,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(btn_start)

.addContainerGap(javax.swing.GroupLayout.D
EFAULT_SIZE, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Terima Stream
Data", panel_terimaData);


panel_chooseFile.setBorder(javax.swing.Borde
rFactory.createEtchedBorder());

        jLabel6.setText("1. Pilih File :");

        btn_fileLocation.setText("Lokasi File");
        btn_fileLocation.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent
evt) {

btn_fileLocationActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout
panel_chooseFileLayout = new
javax.swing.GroupLayout(panel_chooseFile);

panel_chooseFile.setLayout(panel_chooseFileL
ayout);

panel_chooseFileLayout.setHorizontalGroup(

panel_chooseFileLayout.createParallelGroup(ja
vax.swing.GroupLayout.Alignment.LEADING
)

.addGroup(panel_chooseFileLayout.createSequ
entialGroup()
            .addContainerGap()

.addGroup(panel_chooseFileLayout.createParal
lelGroup(javax.swing.GroupLayout.Alignment.
LEADING)

.addGroup(panel_chooseFileLayout.createSequ
entialGroup()
                .addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(btn_fileLocation)
                .addGap(0, 0,
Short.MAX_VALUE))

.addComponent(txt_absoluteFileName))
            .addContainerGap())
        );

panel_chooseFileLayout.setVerticalGroup(

panel_chooseFileLayout.createParallelGroup(ja
vax.swing.GroupLayout.Alignment.LEADING
)

.addGroup(panel_chooseFileLayout.createSequ
entialGroup()
            .addContainerGap()

.addGroup(panel_chooseFileLayout.createParal
lelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(jLabel6)
                .addComponent(btn_fileLocation))

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(txt_absoluteFileName,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addContainerGap(javax.swing.GroupLayout.D
EFAULT_SIZE, Short.MAX_VALUE))
        );
```

```
panel_fileContent.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jLabel18.setText("2. Tampilkan isi file : ");

bg_showFileContentOrNot.add(cb_showFileContentYes);
        cb_showFileContentYes.setText("Ya");

bg_showFileContentOrNot.add(cb_showFileContentNo);
        cb_showFileContentNo.setText("Tidak");

        javax.swing.GroupLayout panel_fileContentLayout = new javax.swing.GroupLayout(panel_fileContent);

panel_fileContent.setLayout(panel_fileContentLayout);

panel_fileContentLayout.setHorizontalGroup(

panel_fileContentLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_fileContentLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel18)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb_showFileContentYes)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb_showFileContentNo)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

panel_fileContentLayout.setVerticalGroup(

panel_fileContentLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_fileContentLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panel_fileContentLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel18)

.addComponent(cb_showFileContentYes)

.addComponent(cb_showFileContentNo))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

panel2_xyChart.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jLabel19.setText("3. Tampilkan grafik :");

bg2_showXYChartOrNot.add(cb2_showXYChartYes);
        cb2_showXYChartYes.setText("Ya");

bg2_showXYChartOrNot.add(cb2_showXYChartNo);
        cb2_showXYChartNo.setText("Tidak");

        javax.swing.GroupLayout panel2_xyChartLayout = new javax.swing.GroupLayout(panel2_xyChart);

panel2_xyChart.setLayout(panel2_xyChartLayout);

panel2_xyChartLayout.setHorizontalGroup(

panel2_xyChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel2_xyChartLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel19)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb2_showXYChartYes)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb2_showXYChartNo)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        panel2_xyChartLayout.setVerticalGroup(

panel2_xyChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel2_xyChartLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panel2_xyChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel19)

.addComponent(cb2_showXYChartYes)

.addComponent(cb2_showXYChartNo))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );


panel2_barChart.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jLabel20.setText("4. Tampilkan Bar Chart :");
        jLabel20.setToolTipText("Untuk menampilkan BarChart, tampilan grafik harus diaktifkan");


bg2_showBarChartOrNot.add(cb2_showBarChartYes);
        cb2_showBarChartYes.setText("Ya");


bg2_showBarChartOrNot.add(cb2_showBarChartNo);
        cb2_showBarChartNo.setText("Tidak");

        javax.swing.GroupLayout panel2_barChartLayout = new javax.swing.GroupLayout(panel2_barChart);
        panel2_barChart.setLayout(panel2_barChartLayout);

panel2_barChartLayout.setHorizontalGroup(

panel2_barChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel2_barChartLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel20)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(cb2_showBarChartYes)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(cb2_showBarChartNo))
            );
            panel2_barChartLayout.setVerticalGroup(

panel2_barChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel2_barChartLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panel2_barChartLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel20)

.addComponent(cb2_showBarChartYes)

.addComponent(cb2_showBarChartNo))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            );


panel2_sensorAnimation.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jLabel21.setText("5. Tampilkan Animasi Sensor :");
```

```
        jLabel21.setToolTipText("Untuk
menampilkan animasi sensor, tampilan grafik
harus diaktifkan");

bg2_showAnimatedSensorOrNot.add(cb2_sho
wAnimatedSensorYes);

cb2_showAnimatedSensorYes.setText("Ya");

bg2_showAnimatedSensorOrNot.add(cb2_sho
wAnimatedSensorNo);

cb2_showAnimatedSensorNo.setText("Tidak")
;

        javax.swing.GroupLayout
panel2_sensorAnimationLayout = new
javax.swing.GroupLayout(panel2_sensorAnim
ation);

panel2_sensorAnimation.setLayout(panel2_sen
sorAnimationLayout);

panel2_sensorAnimationLayout.setHorizontal
Group(

panel2_sensorAnimationLayout.createParallel
Group(javax.swing.GroupLayout.Alignment.L
EADING)

.addGroup(panel2_sensorAnimationLayout.cre
ateSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.D
EFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel21)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb2_showAnimatedSensorYes
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(cb2_showAnimatedSensorNo)
)
        );

panel2_sensorAnimationLayout.setVerticalGro
up(

panel2_sensorAnimationLayout.createParallel
Group(javax.swing.GroupLayout.Alignment.L
EADING)

.addGroup(panel2_sensorAnimationLayout.cre
ateSequentialGroup()
                .addContainerGap()

.addGroup(panel2_sensorAnimationLayout.cre
ateParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                .addComponent(jLabel21)

.addComponent(cb2_showAnimatedSensorYes
)

.addComponent(cb2_showAnimatedSensorNo)
)

.addContainerGap(javax.swing.GroupLayout.D
EFAULT_SIZE, Short.MAX_VALUE))
        );

        btn2_show.setText("Tampilkan");
        btn2_show.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent
evt) {
                btn2_showActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout
panel_bacaFileLayout = new
javax.swing.GroupLayout(panel_bacaFile);

panel_bacaFile.setLayout(panel_bacaFileLayo
ut);

panel_bacaFileLayout.setHorizontalGroup(

panel_bacaFileLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_bacaFileLayout.createSequen
tialGroup()
                .addContainerGap()

.addGroup(panel_bacaFileLayout.createParalle
lGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(panel_chooseFile,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
.addGroup(panel_bacaFileLayout.createSequen
tialGroup()

.addGroup(panel_bacaFileLayout.createParalle
lGroup(javax.swing.GroupLayout.Alignment.L
EADING)

.addComponent(panel_fileContent,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel2_xyChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel2_barChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addComponent(panel2_sensorAnimation,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
                    .addComponent(btn2_show))
                .addGap(0, 202,
Short.MAX_VALUE)))
            .addContainerGap())
    );
    panel_bacaFileLayout.setVerticalGroup(

panel_bacaFileLayout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel_bacaFileLayout.createSequen
tialGroup()
            .addContainerGap()
            .addComponent(panel_chooseFile,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel_fileContent,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel2_xyChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(panel2_barChart,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)

.addComponent(panel2_sensorAnimation,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(btn2_show)
            .addContainerGap(169,
Short.MAX_VALUE))
    );

    jTabbedPane1.addTab("Baca File Data
Logger", panel_bacaFile);

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
            .addGap(112, 112, 112)
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)
            .addContainerGap(133,
Short.MAX_VALUE))
```

```
.addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jTabbedPane1)
            .addContainerGap())
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE
)

.addPreferredGap(javax.swing.LayoutStyle.Co
mponentPlacement.RELATED)
            .addComponent(jTabbedPane1))
        );

        pack();
    }// </editor-fold>

    private void
btn_startActionPerformed(java.awt.event.Actio
nEvent evt) {
        //----------Menampilkan Frame Panel
Kontrol---------//
        controlPanelFrameStreamData = new
ControlPanelFrame(ControlPanelFrame.STRE
AMING_DATA_MODE);

controlPanelFrameStreamData.setTitle("Panel
Kontrol Visualisasi Stream Data");

        //Jika Tombol "start" telah memiliki
tulisan "MULAI".
        if (btn_start.getText().equals("Mulai")){
            try{
                //----------Mengecek Persyaratan
Penyimpanan File----------//

txt_FilePath.setBackground(Color.white);

txt_fileName.setBackground(Color.white);
                if (cb_saveCmd.isSelected()){

controlPanelFrameStreamData.isDataLoggerPe
rformed = true;
                    if
(txt_FilePath.getText().isEmpty()){

JOptionPane.showMessageDialog(null,
"Tentukan lokasi penyimpanan "
                            + "data terlebih dahulu !");

txt_FilePath.setBackground(Color.yellow);
                    txt_FilePath.requestFocus();
                    return;
                }
                else if
(txt_fileName.getText().isEmpty()){

JOptionPane.showMessageDialog(null,
"Berilah nama file terlebih dahulu !");

txt_fileName.setBackground(Color.yellow);
                    txt_fileName.requestFocus();
                    return;
                }
                else{

controlPanelFrameStreamData.isDataLoggerPe
rformed = false;
                }

                //----------Mempersiapkan COMM
PORT untuk Komunikasi----------//
                String strPortName =
cmbo_portList.getSelectedItem().toString();
                if (strPortName.equals("<no
COMM>")){

JOptionPane.showMessageDialog(null, "Tidak
ada koneksi ke serial port");
                    return;
                }
                else{
                    chosenPort =
SerialPort.getCommPort(strPortName);

chosenPort.setComPortTimeouts(SerialPort.TI
MEOUT_SCANNER, 0, 0);
                    if (chosenPort.openPort()){

cmbo_portList.setEnabled(false);
                    }
                    else{

JOptionPane.showMessageDialog(null, "Tidak
bisa menyambungkan ke port");
                    return;
                }
            }
```

```
            //----------Mengeset nilai BaudRate
untuk Komunikasi Serial----------//
            if
(cb_baudRate.getSelectedItem().toString().equ
als("custom...")){
                if
(txt_baudRateCustom.getText().isEmpty()){

JOptionPane.showMessageDialog(null, "Nilai
BaudRate akan disetting "
                        + "ke nilai default : 9600");
                    baudRateVal =
defaultBaudRate;
                }
                else
                baudRateVal =
Integer.parseInt(txt_baudRateCustom.getText()
);
            }
            else{
                baudRateVal =
Integer.parseInt(cb_baudRate.getSelectedItem(
).toString());
            }

chosenPort.setBaudRate(baudRateVal);

            //----------Mengeset Nilai DataBit
untuk Komunikasi Serial----------//
            dataBits =
Integer.parseInt(cb_dataBits.getSelectedItem().
toString());

chosenPort.setNumDataBits(dataBits);

            //----------Mengatur Parity
Komunikasi Serial----------//
            switch
(cb_parity.getSelectedItem().toString()) {
                case "none":
                    parityBits =
SerialPort.NO_PARITY;
                    break;
                case "even":
                    parityBits =
SerialPort.EVEN_PARITY;
                    break;
                case "odd":
                    parityBits =
SerialPort.ODD_PARITY;
                    break;
                case "mark":
                    parityBits =
SerialPort.MARK_PARITY;
                    break;
                case "space":
                    parityBits =
SerialPort.SPACE_PARITY;
                    break;
                default:

JOptionPane.showMessageDialog(null,
"terdapat kesalahan dalam pengaturan Parity");
                    return;
            }
            chosenPort.setParity(parityBits);

            //----------Mengatur Nilai Stop Bits
Komunikasi Serial----------//
            switch
(cb_stopBits.getSelectedItem().toString()) {
                case "1":
                    stopBits =
SerialPort.ONE_STOP_BIT;
                    break;
                case "1.5":
                    stopBits =
SerialPort.ONE_POINT_FIVE_STOP_BITS;
                    break;
                case "2":
                    stopBits =
SerialPort.TWO_STOP_BITS;
                    break;
                default:

JOptionPane.showMessageDialog(null,
"Terdapat kesalahan dalam pengaturan stop
bits");
                    return;
            }

chosenPort.setNumStopBits(stopBits);

            //----------Beri Peringatan tidak ada
satupun visualisasi data yang Ditampilkan------
----//
            if
(!(cb_showXYChartYes.isSelected() ||
cb_showBarChartYes.isSelected() ||
cb_showStreamDataYes.isSelected() ||
cb_showAnimatedSensorYes.isSelected())){
                chosenPort.closePort();

JOptionPane.showMessageDialog(null, "Anda
tidak Menampilkan Grafik Apapun");
                cmbo_portList.setEnabled(true);
                return;
            }

            //----------Mempersiapkan
Penyimpanan File----------//
            if (cb_saveCmd.isSelected()){
```

```
                strFileName =
txt_FilePath.getText()+"\\"+txt_fileName.getT
ext()+".txt";

controlPanelFrameStreamData.setDataLoggerF
ileLocation(strFileName,
ControlPanelFrame.TO_WRITE_FILE);
            }

            //----------Mempersiapkan Stream
Data Frame----------//

controlPanelFrameStreamData.setParticularFra
meEnable(ControlPanelFrame.STREAM_DAT
A_FRAME,
cb_showStreamDataYes.isSelected());

            //----------Mempersiapkan Grafik-----
-----//

if(cb_showXYChartYes.isSelected()){

controlPanelFrameStreamData.setParticularFra
meEnable(ControlPanelFrame.XYCHART_FR
AME, cb_showXYChartYes.isSelected());

controlPanelFrameStreamData.xyChartFrame.d
ataset.removeSeries(controlPanelFrameStream
Data.xyChartFrame.cursorSeries);
            }

            //----------Mempersiapkan BarChart--
--------//

controlPanelFrameStreamData.setParticularFra
meEnable(ControlPanelFrame.BARCHART_F
RAME, cb_showBarChartYes.isSelected());

            //----------Mempersiapkan Animated
Sensor Frame----------//

controlPanelFrameStreamData.setParticularFra
meEnable(ControlPanelFrame.ANIMATED_S
ENSOR_FRAME,
cb_showAnimatedSensorYes.isSelected());


controlPanelFrameStreamData.setVisible(true);

controlPanelFrameStreamData.addWindowList
ener(new java.awt.event.WindowAdapter(){
                @Override public void
windowClosed(java.awt.event.WindowEvent
evt){
                    terminateStreamingData();
                }
            });

            Thread listenAndShowDataThread =
new Thread(){
                @Override public void run(){
                    receiveAndShowData();
                }
            };
            listenAndShowDataThread.start();
            btn_start.setText("Berhenti");
        }
        catch(Exception e){

JOptionPane.showMessageDialog(null, e);
        }
    }
    else{ //Jika Tombol "start" telah memiliki
tulisan "BERHENTI".
        terminateStreamingData();
    }
    //----------AKHIR PROGRAM----------//
}

  private void
cb_baudRatePopupMenuWillBecomeInvisible(
javax.swing.event.PopupMenuEvent evt) {
    if
(cb_baudRate.getSelectedItem().toString().equ
als("custom...")){
      txt_baudRateCustom.setEnabled(true);
    }
    else{
      txt_baudRateCustom.setEnabled(false);
    }
  }

  private void
btn_refreshActionPerformed(java.awt.event.Ac
tionEvent evt) {
    updateCommPortList();
  }

  private void
txt_fileNameKeyReleased(java.awt.event.Key
Event evt) {
    if (!txt_FilePath.getText().isEmpty()){
      String strAbsoluteFile =
txt_FilePath.getText() + "\\" +
txt_fileName.getText() + ".txt";
      absoluteFile = new
File(strAbsoluteFile);
      if (absoluteFile.exists()){
        String strConfirmOverWrite = "File
dengan nama "'+txt_fileName.getText()+
```

```
            "' telah ada"+"\n\nFile yang lama
akan dihapus jika nama file baru sama";

JOptionPane.showMessageDialog(null,
strConfirmOverWrite);
        }
    }
  }

   private void
btn_createFileActionPerformed(java.awt.event.
ActionEvent evt) {
      JFileChooser chooser = new
JFileChooser("E:\\Lecture\\Skripsi\\Latihan
Program java netBeans\\belajar buat coba2");

chooser.setFileSelectionMode(JFileChooser.DI
RECTORIES_ONLY);
     while(true){
        if (chooser.showOpenDialog(null) ==
JFileChooser.APPROVE_OPTION){
           File tempDirectoryFile =
chooser.getSelectedFile();
           String absoluteFilePath;
           File tempAbsoluteFile;

           if (txt_fileName.getText().isEmpty()
== false){
              absoluteFilePath =
tempDirectoryFile.getPath() + "\\" +
txt_fileName.getText() + ".txt";
              tempAbsoluteFile = new
File(absoluteFilePath);
              if (tempAbsoluteFile.exists()){
                 String strConfirmOverWrite =
"File dengan nama "'+txt_fileName.getText()+
                 "' telah ada"+"\nApakah Anda
akan mengganti file lama dengan yang baru?";
                 int decision =
JOptionPane.showConfirmDialog(null,
strConfirmOverWrite,
                    "Overwrite Confirmation",
JOptionPane.YES_NO_CANCEL_OPTION);
                 if (decision ==
JOptionPane.YES_OPTION){

txt_FilePath.setText(tempDirectoryFile.getPath
());
                    System.out.println("File telah
ada, dan dioverwrite");
                    break;
                 }
                 else if (decision ==
JOptionPane.CANCEL_OPTION){
                    System.out.println("file telah
ada, dan cancel, PERHATIKAN bahwa
txt_pathFile harus tidak berubah");
                    break;
                 }
                 else if (decision ==
JOptionPane.NO_OPTION){
                    System.out.println("file telah
ada, dan chooserFile kembali dibuka");
                    continue;
                 }
              }
              else{

txt_FilePath.setText(tempDirectoryFile.getPath
());
                 System.out.println("Ada nama
file, tapi belum ada sebelumnya");
                 break;
              }
           }
           else{

txt_FilePath.setText(tempDirectoryFile.getPath
());
              System.out.println("nama file
empty");
              break;
           }
        }
        else break;
     }
  }

  private void
cb_saveCmdActionPerformed(java.awt.event.
ActionEvent evt) {
    txt_FilePath.setEnabled(true);
    btn_createFile.setEnabled(true);
    txt_fileName.setEnabled(true);
  }

  private void
cb_notSaveActionPerformed(java.awt.event.Ac
tionEvent evt) {
    txt_FilePath.setEnabled(false);
    btn_createFile.setEnabled(false);
    txt_fileName.setEnabled(false);
  }

  private void
btn_fileLocationActionPerformed(java.awt.eve
nt.ActionEvent evt) {
     JFileChooser fileChooser = new
JFileChooser("E:\\Lecture\\Skripsi\\Latihan
Program java netBeans\\belajar buat coba2");
```

```
fileChooser.setFileSelectionMode(JFileChoose
r.FILES_ONLY);
      if (fileChooser.showOpenDialog(null) ==
JFileChooser.APPROVE_OPTION){

txt_absoluteFileName.setText(fileChooser.getS
electedFile().getAbsolutePath());
      }
   }

   private void
btn2_showActionPerformed(java.awt.event.Act
ionEvent evt) {
   try{
      String dataSensorPath =
txt_absoluteFileName.getText();
      if (dataSensorPath.isEmpty()){
         JOptionPane.showMessageDialog(null,
"Pilih file yang menyimpan data sensor terlebih
dahulu");

txt_absoluteFileName.setBackground(Color.ye
llow);
         txt_absoluteFileName.requestFocus();
         return;
      }
      if
(dataSensorPath.endsWith(".txt")==false){
         JOptionPane.showMessageDialog(null,
"Pilihlah file dengan benar");
         txt_absoluteFileName.setText(null);

txt_absoluteFileName.setBackground(Color.ye
llow);
         txt_absoluteFileName.requestFocus();
         return;
      }

txt_absoluteFileName.setBackground(Color.w
hite);

      if (cb_showFileContentNo.isSelected()
&& cb2_showXYChartNo.isSelected() &&
cb2_showBarChartNo.isSelected() &&
cb2_showAnimatedSensorNo.isSelected()){
         JOptionPane.showMessageDialog(null,
"Anda tidak menampilkan apapun");
         return;
      }

      controlPanelFrameReadFile = new
ControlPanelFrame(ControlPanelFrame.READ
_DATAFILE_MODE);

controlPanelFrameReadFile.setTitle("Panel
Kontrol Visualisasi DataLogger");

controlPanelFrameReadFile.setDataLoggerFile
Location(dataSensorPath,
ControlPanelFrame.TO_READ_FILE);

controlPanelFrameReadFile.isDataLoggerPerfo
rmed = false;

      //----------Mempersiapkan Stream Data
Frame----------//

controlPanelFrameReadFile.setParticularFrame
Enable(ControlPanelFrame.STREAM_DATA_
FRAME,
cb_showFileContentYes.isSelected());

      //----------Mempersiapkan Grafik----------//

controlPanelFrameReadFile.setParticularFrame
Enable(ControlPanelFrame.XYCHART_FRA
ME, cb2_showXYChartYes.isSelected());

      //----------Mempersiapkan BarChart--------
--//

controlPanelFrameReadFile.setParticularFrame
Enable(ControlPanelFrame.BARCHART_FRA
ME, cb2_showBarChartYes.isSelected());

      //----------Mempersiapkan Animated
Sensor Frame----------//

controlPanelFrameReadFile.setParticularFrame
Enable(ControlPanelFrame.ANIMATED_SEN
SOR_FRAME,
cb2_showAnimatedSensorYes.isSelected());


controlPanelFrameReadFile.setVisible(true);

controlPanelFrameReadFile.addWindowListen
er(new java.awt.event.WindowAdapter() {
      @Override public void
windowClosed(java.awt.event.WindowEvent
evt){
      try{

controlPanelFrameReadFile.br.close();
      }
      catch(Exception e){
         System.out.println("error when
closing buffered reader");
         System.out.println(e);
```

```
        }
        }
    });

controlPanelFrameReadFile.readAndDisplayAl
lFileContent();
  }
  catch(Exception e){
    System.out.println("error pada akhir
method btn2_showActionPerformed:");
    System.out.println(e);
  }
  }

  private void terminateStreamingData(){
    try{
      chosenPort.closePort();
      cmbo_portList.setEnabled(true);
      btn_start.setText("Mulai");
    }
    catch(Exception e){
      JOptionPane.showMessageDialog(null,
e);
    }
  }
  private void receiveAndShowData(){
    int caretPosition=0;
    Scanner streamDataIn = new
Scanner(chosenPort.getInputStream());
    int[] marginDataSet = new
int[nbrOfSensor*2];

    //----------Bagian penerimaan data
treshold----------//
    while(streamDataIn.hasNextLine()){
      try{
      String dataFrame =
streamDataIn.nextLine();
      if(dataFrame.startsWith("###")){
        String marginDataFrame =
dataFrame.substring(3);
        String[] marginDataSetStr =
marginDataFrame.split("/");
        for (int i=0;
i<marginDataSetStr.length; i++){
          marginDataSet[i] =
Integer.parseInt(marginDataSetStr[i]);
        }

        //----------Inputkan String Data
Margin pada Window Stream Data----------//
        if
(controlPanelFrameStreamData.isShowStream
Data==true){

controlPanelFrameStreamData.streamDataFra
me.ta_streamData.append(dataFrame+"\n");

caretPosition=controlPanelFrameStreamData.st
reamDataFrame.ta_streamData.getDocument().
getLength();

controlPanelFrameStreamData.streamDataFra
me.ta_streamData.setCaretPosition(caretPositio
n);
        }

        //----------Inputkan data margin pada
barChart----------//
        if
(controlPanelFrameStreamData.isShowBarCha
rtFrame==true){

controlPanelFrameStreamData.barchartFrame.s
etDataMargin(marginDataSet);
        }

        //----------Inputkan Data Margin pada
File .txt----------//
        if
(controlPanelFrameStreamData.isDataLoggerP
erformed==true){

controlPanelFrameStreamData.bw.write(dataFr
ame+"\n");

controlPanelFrameStreamData.bw.flush();
        }

        break;
      }
      }
      catch(Exception e){
        System.out.println("eror di stream
data margin : "+e);
      }
    }

    while(streamDataIn.hasNextLine()){
      String dataFrame =
streamDataIn.nextLine();
      if (dataFrame.startsWith("+++")){
        break;
      }
    }

    long dataTh = 0;
    //----------Bagian penerimaan current data
sensor----------//
    while(streamDataIn.hasNextLine()){
```

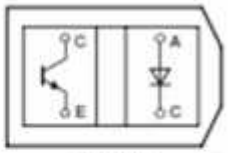
```
        try{
            dataTh++;
            String dataFrame =
streamDataIn.nextLine();
            //----------Menampilkan Stream Data----
------//
            if
(controlPanelFrameStreamData.isShowStream
Data==true){

controlPanelFrameStreamData.streamDataFra
me.ta_streamData.append(dataTh+")"+dataFra
me+"\n");

caretPosition=controlPanelFrameStreamData.st
reamDataFrame.ta_streamData.getDocument().
getLength();

controlPanelFrameStreamData.streamDataFra
me.ta_streamData.setCaretPosition(caretPositio
n);
            }

            String[] sensValStr =
dataFrame.split("/");
            int nbrOfDataReceived =
sensValStr.length;

            //----------Menampilkan Nilai Sensor
pada Control Panel----------//

controlPanelFrameStreamData.setSensorsValu
e(sensValStr);

            int[] sensVal = new int
[nbrOfDataReceived];
            for (int i=0; i<nbrOfDataReceived;
i++){
              sensVal[i] =
Integer.parseInt(sensValStr[i]);
            }

            //----------Menampilkan XYChart--------
--//
            if
(controlPanelFrameStreamData.isShowXYCha
rtFrame==true){

controlPanelFrameStreamData.xyChartFrame.a
ddGraphData(dataTh, sensVal);

//controlPanelFrameStreamData.xyChartFrame
.deleteMinimumIndex();
            }

            //----------Menampilkan BarChart--------
--//
            if
(controlPanelFrameStreamData.isShowBarCha
rtFrame==true){
                for (byte cnt=0;
cnt<nbrOfDataReceived; cnt++){

controlPanelFrameStreamData.barchartFrame.
updateBarChart(cnt, sensVal[cnt],
controlPanelFrameStreamData.sensorState[cnt]
);
                }
            }

            //----------Menuliskan Stream Data pada
File .txt----------//
            if
(controlPanelFrameStreamData.isDataLoggerP
erformed==true){

controlPanelFrameStreamData.bw.write(dataT
h+")"+dataFrame+"\n");

controlPanelFrameStreamData.bw.flush();
            }

            //----------Menampilkan Animasi
Sensor----------//

if(controlPanelFrameStreamData.isShowAnim
atedSensorFrame){
                byte[] senStatus = new
byte[nbrOfDataReceived];
                for (int i=0; i<nbrOfDataReceived;
i++){
                    if
(sensVal[i]<=marginDataSet[2*i])
senStatus[i]=AnimatedSensorFrame.LOGIC_L
OW;
                    else if
(sensVal[i]>=marginDataSet[2*i+1])
senStatus[i]=AnimatedSensorFrame.LOGIC_H
IGH;
                    else senStatus[i] =
AnimatedSensorFrame.LOGIC_UNDETERMI
NED;
                }

controlPanelFrameStreamData.animatedSensor
Frame.sensLogicState = senStatus;

controlPanelFrameStreamData.animatedSensor
Frame.repaint();
            }
        }
```

```
        catch(Exception e){
            System.out.println("error pada
penerimaan stream data:"+e);
        }
        }
        streamDataIn.close();
        System.out.println("stream
ditutup");//stream ditutup jika usb to ttl dicabut
sehingga Thread juga selesai dan berhenti
                                //jika usb to ttl
TIDAK dicabut tetapi MK mati, maka stream
data tetep ada
                                //sehingga
streamDataIn.hasNextLine() tetep polling
    }

    /**
     * @param args the command line
arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed"
desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is
not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uisw
ing/lookandfeel/plaf.html
         */
        try {
            for
(javax.swing.UIManager.LookAndFeelInfo
info :
javax.swing.UIManager.getInstalledLookAndF
eels()) {
                if ("Nimbus".equals(info.getName()))
{

javax.swing.UIManager.setLookAndFeel(info.
getClassName());
                    break;
                }

//UIManager.setLookAndFeel("com.jtattoo.pla
f.smart.SmartLookAndFeel");

//UIManager.setLookAndFeel("com.jtattoo.pla
f.acryl.AcrylLookAndFeel");

//UIManager.setLookAndFeel("com.jtattoo.pla
f.aero.AeroLookAndFeel");

//UIManager.setLookAndFeel("com.jtattoo.pla
f.aluminium.AluminiumLookAndFeel");

//UIManager.setLookAndFeel("com.jtattoo.pla
f.graphite.GraphiteLookAndFeel");
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(menujuFix
VisualSensor1.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(menujuFix
VisualSensor1.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(menujuFix
VisualSensor1.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        } catch
(javax.swing.UnsupportedLookAndFeelExcept
ion ex) {

java.util.logging.Logger.getLogger(menujuFix
VisualSensor1.class.getName()).log(java.util.lo
gging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new
Runnable() {
            public void run() {
                new
menujuFixVisualSensor1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup
bg2_showAnimatedSensorOrNot;
    private javax.swing.ButtonGroup
bg2_showBarChartOrNot;
    private javax.swing.ButtonGroup
bg2_showXYChartOrNot;
    private javax.swing.ButtonGroup
bg_showAnimatedSensorOrNot;
    private javax.swing.ButtonGroup
bg_showBarCharOrNot;
    private javax.swing.ButtonGroup
bg_showFileContentOrNot;
    private javax.swing.ButtonGroup
bg_showStreamDataOrNot;
    private javax.swing.ButtonGroup
bg_showXYChartOrNot;
```

```
    private javax.swing.JButton btn2_show;
    private javax.swing.JButton btn_createFile;
    private javax.swing.JButton
btn_fileLocation;
    private javax.swing.JButton btn_refresh;
    private javax.swing.JButton btn_start;
    private javax.swing.JCheckBox
cb2_showAnimatedSensorNo;
    private javax.swing.JCheckBox
cb2_showAnimatedSensorYes;
    private javax.swing.JCheckBox
cb2_showBarChartNo;
    private javax.swing.JCheckBox
cb2_showBarChartYes;
    private javax.swing.JCheckBox
cb2_showXYChartNo;
    private javax.swing.JCheckBox
cb2_showXYChartYes;
    private javax.swing.JComboBox
cb_baudRate;
    private javax.swing.JComboBox
cb_dataBits;
    private javax.swing.JCheckBox cb_notSave;
    private javax.swing.JComboBox cb_parity;
    private javax.swing.JCheckBox
cb_saveCmd;
    private javax.swing.JCheckBox
cb_showAnimatedSensorNo;
    private javax.swing.JCheckBox
cb_showAnimatedSensorYes;
    private javax.swing.JCheckBox
cb_showBarChartNo;
    private javax.swing.JCheckBox
cb_showBarChartYes;
    private javax.swing.JCheckBox
cb_showFileContentNo;
    private javax.swing.JCheckBox
cb_showFileContentYes;
    private javax.swing.JCheckBox
cb_showStreamDataNo;
    private javax.swing.JCheckBox
cb_showStreamDataYes;
    private javax.swing.JCheckBox
cb_showXYChartNo;
    private javax.swing.JCheckBox
cb_showXYChartYes;
    private javax.swing.JComboBox
cb_stopBits;
    private javax.swing.JComboBox
cmbo_portList;
    private javax.swing.ButtonGroup
groupSaveFileOrNot;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JProgressBar
jProgressBar1;
    private javax.swing.JTabbedPane
jTabbedPane1;
    private javax.swing.JPanel panel2_barChart;
    private javax.swing.JPanel
panel2_sensorAnimation;
    private javax.swing.JPanel panel2_xyChart;
    private javax.swing.JPanel panel_bacaFile;
    private javax.swing.JPanel panel_barChart;
    private javax.swing.JPanel
panel_chooseFile;
    private javax.swing.JPanel
panel_commSetting;
    private javax.swing.JPanel
panel_fileContent;
    private javax.swing.JPanel
panel_fileStorage;
    private javax.swing.JPanel
panel_sensorAnimation;
    private javax.swing.JPanel
panel_streamData;
    private javax.swing.JPanel
panel_terimaData;
    private javax.swing.JPanel panel_xyChart;
    private javax.swing.JTextField txt_FilePath;
    private javax.swing.JTextField
txt_absoluteFileName;
    private javax.swing.JTextField
txt_baudRateCustom;
    private javax.swing.JTextField
txt_fileName;
    // End of variables declaration
}
```

Lampiran 6 *Datasheet*

Reflective Optical Sensor with Transistor Output



Top view

FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 10.2 x 5.8 x 7
- Peak operating distance: 2.5 mm
- Operating range within > 20 % relative collector current: 0.2 mm to 15 mm
- Typical output current under test: I_C = 1 mA
- Daylight blocking filter
- Emitter wavelength: 950 nm
- Lead (Pb)-free soldering released
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

RoHS COMPLIANT

DESCRIPTION

The TCRT5000 and TCRT5000L are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light. The package includes two mounting clips. TCRT5000L is the long lead version.

APPLICATIONS

- Position sensor for shaft encoder
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR
- General purpose - wherever the space is limited

PRODUCT SUMMARY

PART NUMBER	DISTANCE FOR MAXIMUM CTR_{rel} [1] (mm)	DISTANCE RANGE FOR RELATIVE I_{out} > 20 % (mm)	TYPICAL OUTPUT CURRENT UNDER TEST [2] (mA)	DAYLIGHT BLOCKING FILTER INTEGRATED
TCRT5000	2.5	0.2 to 15	1	Yes
TCRT5000L	2.5	0.2 to 15	1	Yes

Notes

[1] CTR: current transfere ratio, I_{out}/I_{in}

[2] Conditions like in table basic charactristics/sensors

ORDERING INFORMATION

ORDERING CODE	PACKAGING	VOLUME [1]	REMARKS
TCRT5000	Tube	MOQ: 4500 pcs, 50 pcs/tube	3.5 mm lead length
TCRT5000L	Tube	MOQ: 2400 pcs, 48 pcs/tube	15 mm lead length

Note

[1] MOQ: minimum order quantity

ABSOLUTE MAXIMUM RATINGS [1]

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
INPUT (EMITTER)				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	3	A
Power dissipation	$T_{amb} \leq 25$ °C	P_V	100	mW
Junction temperature		T_j	100	°C

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
OUTPUT (DETECTOR)				
Collector emitter voltage		V_{CEO}	70	V
Emitter collector voltage		V_{ECO}	5	V
Collector current		I_C	100	mA
Power dissipation	$T_{amb} \leq 55$ °C	P_V	100	mW
Junction temperature		T_j	100	°C
SENSOR				
Total power dissipation	$T_{amb} \leq 25$ °C	P_{tot}	200	mW
Ambient temperature range		T_{amb}	- 25 to + 85	°C
Storage temperature range		T_{stg}	- 25 to + 100	°C
Soldering temperature	2 mm from case, t ≤ 10 s	T_{sd}	260	°C

Note
(1) T_{amb} = 25 °C, unless otherwise specified

ABSOLUTE MAXIMUM RATINGS

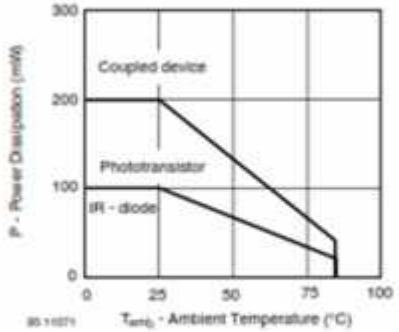


Fig. 1 - Power Dissipation Limit vs. Ambient Temperature

BASIC CHARACTERISTICS (1)						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
INPUT (EMITTER)						
Forward voltage	I_F = 60 mA	V_F		1.25	1.5	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j		17		pF
Radiant intensity	I_F = 60 mA, t_p = 20 ms	I_e			21	mW/sr
Peak wavelength	I_F = 100 mA	λ_p	940			nm
Virtual source diameter	Method: 63 % encircled energy	d		2.1		mm
OUTPUT (DETECTOR)						
Collector emitter voltage	I_C = 1 mA	V_{CEO}	70			V
Emitter collector voltage	I_E = 100 µA	V_{ECO}	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0 A, E = 0 lx	I_{CEO}		10	200	nA
SENSOR						
Collector current	V_{CE} = 5 V, I_F = 10 mA, D = 12 mm	I_C (2) (3)	0.5	1	2.1	mA
Collector emitter saturation voltage	I_F = 10 mA, I_C = 0.1 mA, D = 12 mm	V_{CEsat} (2) (3)			0.4	V

Note
(1) T_{amb} = 25 °C, unless otherwise specified
(2) See figure 3
(3) Test surface: mirror (Mfr. Spindler a. Hoyer, Part No. 340005)

TEXAS INSTRUMENTS

CD4051B, CD4052B, CD4053B

Data sheet acquired from Harris Semiconductor
SCHS047G

August 1998 - Revised October 2003

Features

- Wide Range of Digital and Analog Signal Levels
  - Digital ..... 3V to 20V
  - Analog ..... $\leq 20V_{P-P}$
- Low ON Resistance, 125Ω (Typ) Over $15V_{P-P}$ Signal Input Range for V_{DD}-V_{EE} = 18V
- High OFF Resistance, Channel Leakage of ±100pA (Typ) at V_{DD}-V_{EE} = 18V
- Logic-Level Conversion for Digital Addressing Signals of 3V to 20V (V_{DD}-V_{SS} = 3V to 20V) to Switch Analog Signals to $20V_{P-P}$ (V_{DD}-V_{EE} = 20V)
- Matched Switch Characteristics, r_{ON} = 5Ω (Typ) for V_{DD}-V_{EE} = 15V
- Very Low Quiescent Power Dissipation Under All Digital-Control Input and Supply Conditions, 0.2μW (Typ) at V_{DD}-V_{SS} = V_{DD}-V_{EE} = 10V
- Binary Address Decoding on Chip
- 5V, 10V, and 15V Parametric Ratings
- 100% Tested for Quiescent Current at 20V
- Maximum Input Current of 1μA at 18V Over Full Package Temperature Range, 100nA at 18V and 25°C
- Break-Before-Make Switching Eliminates Channel Overlap

Applications

- Analog and Digital Multiplexing and Demultiplexing
- A/D and D/A Conversion
- Signal Gating

CMOS Analog Multiplexers/Demultiplexers with Logic Level Conversion

The CD4051B, CD4052B, and CD4053B analog multiplexers are digitally-controlled analog switches having low ON impedance and very low OFF leakage current. Control of analog signals up to $20V_{P-P}$ can be achieved by digital signal amplitudes of 4.5V to 20V (if V_{DD}-V_{SS} = 3V, a V_{DD}-V_{EE} of up to 13V can be controlled; for V_{DD}-V_{EE} level differences above 13V, a V_{DD}-V_{SS} of at least 4.5V is required). For example, if V_{DD} = +4.5V, V_{SS} = 0V, and V_{EE} = -13.5V, analog signals from -13.5V to +4.5V can be controlled by digital inputs of 0V to 5V. These multiplexer circuits dissipate extremely low quiescent power over the full V_{DD}-V_{SS} and V_{DD}-V_{EE} supply-voltage ranges, independent of the logic state of the control signals. When a logic "1" is present at the inhibit input terminal, all channels are off.

The CD4051B is a single 8-Channel multiplexer having three binary control inputs, A, B, and C, and an inhibit input. The three binary signals select 1 of 8 channels to be turned on, and connect one of the 8 inputs to the output.

The CD4052B is a differential 4-Channel multiplexer having two binary control inputs, A and B, and an inhibit input. The two binary input signals select 1 of 4 pairs of channels to be turned on and connect the analog inputs to the outputs.

The CD4053B is a triple 2-Channel multiplexer having three separate digital control inputs, A, B, and C, and an inhibit input. Each control input selects one of a pair of channels which are connected in a single-pole, double-throw configuration.

When these devices are used as demultiplexers, the "CHANNEL IN/OUT" terminals are the outputs and the "COMMON OUT/IN" terminals are the inputs.

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE
CD4051BF3A, CD4052BF3A, CD4053BF3A	-55 to 125	16 Ld CERAMIC DIP
CD4051BE, CD4052BE, CD4053BE	-55 to 125	16 Ld PDIP
CD4051BM, CD4051BMT, CD4051BM96 CD4052BM, CD4052BMT, CD4052BM96 CD4053BM, CD4053BMT, CD4053BM96	-55 to 125	16 Ld SOIC
CD4051BNSR, CD4052BNSR, CD4053BNSR	-55 to 125	16 Ld SOP
CD4051BPW, CD4051BPWR, CD4052BPW, CD4052BPWR CD4053BPW, CD4053BPWR	-55 to 125	16 Ld TSSOP

NOTE: When ordering, use the entire part number. The suffixes 96 and R denote tape and reel. The suffix T denotes a small-quantity reel of 250.

Electrical Specifications

Common Conditions Here: If Whole Table is For the Full Temp. Range, V_{SUPPLY} = ±5V, A_V = +1, R_L = 100Ω, Unless Otherwise Specified **(Continued)** (Note 3)

PARAMETER	CONDITIONS				LIMITS AT INDICATED TEMPERATURES (°C)							UNITS
									25			
	V_{IS} (V)	V_{EE} (V)	V_{SS} (V)	V_{DD} (V)	-55	-40	85	125	MIN	TYP	MAX	
CONTROL (ADDRESS OR INHIBIT), V_C												
Input Low Voltage, V_{IL}, Max	V_{IL} = V_{DD} through 1kΩ; V_{IH} = V_{DD} through 1kΩ	V_{EE} = V_{SS}, R_L = 1kΩ to V_{SS}, I_{IS} < 2µA on All OFF Channels		5	1.5	1.5	1.5	1.5	-	-	1.5	V
				10	3	3	3	3	-	-	3	V
				15	4	4	4	4	-	-	4	V
Input High Voltage, V_{IH}, Min				5	3.5	3.5	3.5	3.5	3.5	-	-	V
				10	7	7	7	7	7	-	-	V
				15	11	11	11	11	11	-	-	V
Input Current, I_{IN} (Max)	V_{IN} = 0, 18			18	±0.1	±0.1	±1	±1	-	$±10^{-5}$	±0.1	µA
Propagation Delay Time: Address-to-Signal OUT (Channels ON or OFF) See Figures 10, 11, 14	t_r, t_f = 20ns, C_L = 50pF, R_L = 10kΩ	0	0	5	-	-	-	-	-	450	720	ns
		0	0	10	-	-	-	-	-	160	320	ns
		0	0	15	-	-	-	-	-	120	240	ns
		-5	0	5	-	-	-	-	-	225	450	ns
Propagation Delay Time: Inhibit-to-Signal OUT (Channel Turning ON) See Figure 11	t_r, t_f = 20ns, C_L = 50pF, R_L = 1kΩ	0	0	5	-	-	-	-	-	400	720	ns
		0	0	10	-	-	-	-	-	160	320	ns
		0	0	15	-	-	-	-	-	120	240	ns
		-10	0	5	-	-	-	-	-	200	400	ns
Propagation Delay Time: Inhibit-to-Signal OUT (Channel Turning OFF) See Figure 15	t_r, t_f = 20ns, C_L = 50pF, R_L = 10kΩ	0	0	5	-	-	-	-	-	200	450	ns
		0	0	10	-	-	-	-	-	90	210	ns
		0	0	15	-	-	-	-	-	70	160	ns
		-10	0	5	-	-	-	-	-	130	300	ns
Input Capacitance, C_{IN} (Any Address or Inhibit Input)					-	-	-	-	-	5	7.5	pF

NOTE:

2. Determined by minimum feasible leakage measurement for automatic testing.

Electrical Specifications

PARAMETER	TEST CONDITIONS					LIMITS	UNITS
	V_{IS} (V)	V_{DD} (V)	R_L (kΩ)			TYP	
Cutoff (-3dB) Frequency Channel ON (Sine Wave Input)	5 (Note 3)	10	1	V_{OS} at Common OUT/IN	CD4053	30	MHz
	V_{EE} = V_{SS}, $20\text{Log}\frac{V_{OS}}{V_{IS}}$ = -3dB				CD4052	25	MHz
					CD4051	20	MHz
				V_{OS} at Any Channel		60	MHz

Absolute Maximum Ratings

Supply Voltage (V_+ to V_-)
Voltages Referenced to V_{SS} Terminal ..... -0.5V to 20V
DC Input Voltage Range ..... -0.5V to V_{DD} +0.5V
DC Input Current, Any One Input ..... ±10mA

Operating Conditions

Temperature Range ..... -55°C to 125°C

Thermal Information

Package Thermal Impedance, θ_{JA} (see Note 1):
E (PDIP) package ..... 67°C/W
M (SOIC) package ..... 73°C/W
NS (SOP) package ..... 64°C/W
PW (TSSOP) package ..... 108°C/W
Maximum Junction Temperature (Ceramic Package) ..... 175°C
Maximum Junction Temperature (Plastic Package) ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 265°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. The package thermal impedance is calculated in accordance with JESD 51-7.

Electrical Specifications

Common Conditions Here: If Whole Table is For the Full Temp. Range, V_{SUPPLY} = ±5V, A_V = +1, R_L = 100Ω, Unless Otherwise Specified (Note 3)

PARAMETER	CONDITIONS V_{IS} (V)	V_{EE} (V)	V_{SS} (V)	V_{DD} (V)	LIMITS AT INDICATED TEMPERATURES (°C) -55	-40	85	125	25 MIN	25 TYP	25 MAX	UNITS
SIGNAL INPUTS (V_{IS}) AND OUTPUTS (V_{OS})												
Quiescent Device Current, I_{DD} Max	-	-	-	5	5	5	150	150	-	0.04	5	µA
	-	-	-	10	10	10	300	300	-	0.04	10	µA
	-	-	-	15	20	20	600	600	-	0.04	20	µA
	-	-	-	20	100	100	3000	3000	-	0.08	100	µA
Drain to Source ON Resistance r_{ON} Max $0 \le V_{IS} \le V_{DD}$	-	0	0	5	800	850	1200	1300	-	470	1050	Ω
	-	0	0	10	310	330	520	550	-	180	400	Ω
	-	0	0	15	200	210	300	320	-	125	240	Ω
Change in ON Resistance (Between Any Two Channels), Δr_{ON}	-	0	0	5	-	-	-	-	-	15	-	Ω
	-	0	0	10	-	-	-	-	-	10	-	Ω
	-	0	0	15	-	-	-	-	-	5	-	Ω
OFF Channel Leakage Current: Any Channel OFF (Max) or ALL Channels OFF (Common OUT/IN) (Max)	-	0	0	18	±100 (Note 2)		±1000 (Note 2)		-	±0.01	±100 (Note 2)	nA
Capacitance: Input, C_{IS}	-	-5	5	5	-	-	-	-	-	5	-	pF
Output, C_{OS} CD4051					-	-	-	-	-	30	-	pF
CD4052					-	-	-	-	-	18	-	pF
CD4053					-	-	-	-	-	9	-	pF
Feedthrough C_{IOS}					-	-	-	-	-	0.2	-	pF
Propagation Delay Time (Signal Input to Output	V_{DD} 	R_L = 200kΩ, C_L = 50pF, t_r, t_f = 20ns		5	-	-	-	-	-	30	60	ns
				10	-	-	-	-	-	15	30	ns
				15	-	-	-	-	-	10	20	ns

TRUTH TABLES

INPUT STATES				
INHIBIT	C	B	A	"ON" CHANNEL(S)
CD4051B				
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	None
CD4052B				
INHIBIT	B		A	
0	0		0	0x, 0y
0	0		1	1x, 1y
0	1		0	2x, 2y
0	1		1	3x, 3y
1	X		X	None
CD4053B				
INHIBIT	A OR B OR C			
0	0			ax or bx or cx
0	1			ay or by or cy
1	X			None

X = Don't Care

Electrical Specifications

PARAMETER	TEST CONDITIONS					LIMITS	UNITS
	V_{IS} (V)	V_{DD} (V)	R_L (kΩ)			TYP	
Total Harmonic Distortion, THD	2 (Note 3)	5	10			0.3	%
	3 (Note 3)	10				0.2	%
	5 (Note 3)	15				0.12	%
	$V_{EE} = V_{SS}$, f_{IS} = 1kHz Sine Wave						%
-40dB Feedthrough Frequency (All Channels OFF)	5 (Note 3)	10	1	V_{OS} at Common OUT/IN	CD4053	8	MHz
	$V_{EE} = V_{SS}$, $20\text{Log}\frac{V_{OS}}{V_{IS}} = -40\text{dB}$				CD4052	10	MHz
					CD4051	12	MHz
				V_{OS} at Any Channel		8	MHz
-40dB Signal Crosstalk Frequency	5 (Note 3)	10	1	Between Any 2 Channels		3	MHz
	$V_{EE} = V_{SS}$, $20\text{Log}\frac{V_{OS}}{V_{IS}} = -40\text{dB}$			Between Sections, CD4052 Only	Measured on Common	6	MHz
					Measured on Any Channel	10	MHz
				Between Any Two Sections, CD4053 Only	In Pin 2, Out Pin 14	2.5	MHz
					In Pin 15, Out Pin 14	6	MHz
Address-or-Inhibit-to-Signal Crosstalk	-	10	10 (Note 4)			65	mV_{PEAK}
	$V_{EE} = 0$, $V_{SS} = 0$, t_r, t_f = 20ns, $V_{CC} = V_{DD} - V_{SS}$ (Square Wave)					65	mV_{PEAK}

NOTES:

3. Peak-to-Peak voltage symmetrical about $\frac{V_{DD} - V_{EE}}{2}$

4. Both ends of channel.

Typical Performance Curves

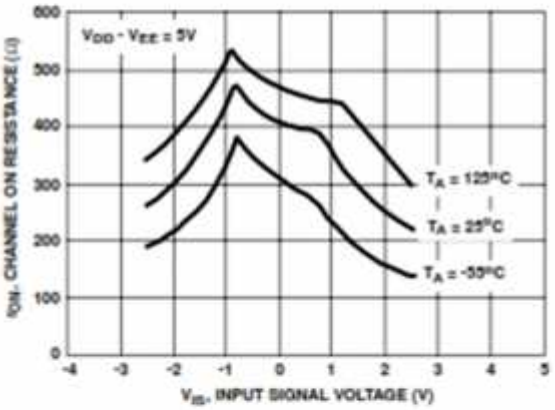


FIGURE 1. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

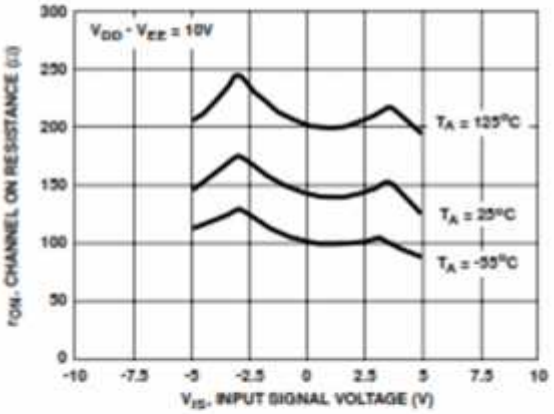


FIGURE 2. CHANNEL ON RESISTANCE vs INPUT SIGNAL VOLTAGE (ALL TYPES)

HC-05

-Bluetooth to Serial Port Module

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH(Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

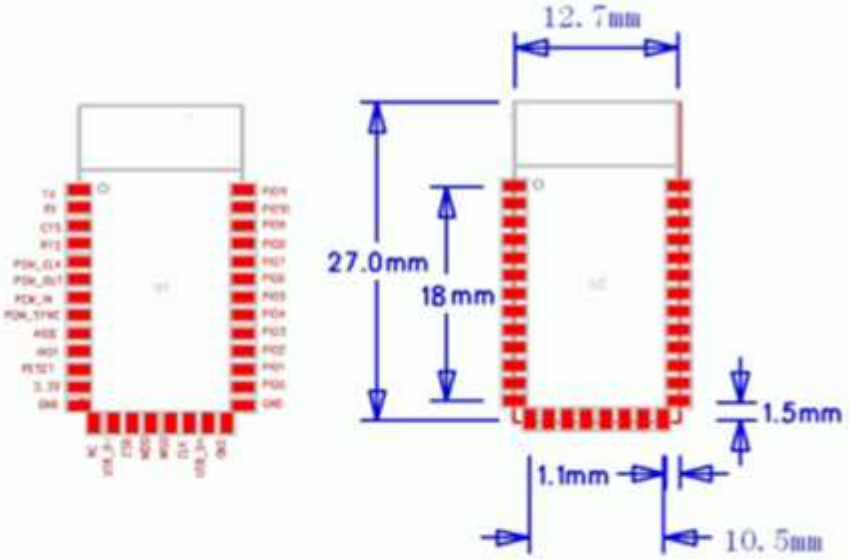
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



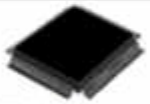
STM32F100x4 STM32F100x6 STM32F100x8 STM32F100xB

Low & medium-density value line, advanced ARM-based 32-bit MCU with 16 to 128 KB Flash, 12 timers, ADC, DAC & 8 comm interfaces

Datasheet – production data

Features

- Core: ARM 32-bit Cortex™-M3 CPU
  - 24 MHz maximum frequency, 1.25 DMIPS/MHz (Dhrystone 2.1) performance
  - Single-cycle multiplication and hardware division
- Memories
  - 16 to 128 Kbytes of Flash memory
  - 4 to 8 Kbytes of SRAM
- Clock, reset and supply management
  - 2.0 to 3.6 V application supply and I/Os
  - POR, PDR and programmable voltage detector (PVD)
  - 4-to-24 MHz crystal oscillator
  - Internal 8 MHz factory-trimmed RC
  - Internal 40 kHz RC
  - PLL for CPU clock
  - 32 kHz oscillator for RTC with calibration
- Low power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC and backup registers
- Debug mode
  - Serial wire debug (SWD) and JTAG interfaces
- DMA
  - 7-channel DMA controller
  - Peripherals supported: timers, ADC, SPIs, I^2Cs, USARTs and DACs
- 1 × 12-bit, 1.2 µs A/D converter (up to 16 channels)
  - Conversion range: 0 to 3.6 V
  - Temperature sensor
- 2 × 12-bit D/A converters
- Up to 80 fast I/O ports
  - 37/51/80 I/Os, all mappable on 16 external interrupt vectors and almost all 5 V-tolerant



LQFP100 14 × 14 mm
LQFP64 10 × 10 mm
LQFP48 7 × 7 mm

TFBGA64 (5 × 5 mm)

- Up to 12 timers
  - Up to three 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter
  - 16-bit, 6-channel advanced-control timer: up to 6 channels for PWM output, dead time generation and emergency stop
  - One 16-bit timer, with 2 IC/OC, 1 OCN/PWM, dead-time generation and emergency stop
  - Two 16-bit timers, each with IC/OC/OCN/PWM, dead-time generation and emergency stop
  - 2 watchdog timers (Independent and Window)
  - SysTick timer: 24-bit downcounter
  - Two 16-bit basic timers to drive the DAC
- Up to 8 communications interfaces
  - Up to two I^2C interfaces (SMBus/PMBus)
  - Up to 3 USARTs (ISO 7816 interface, LIN, IrDA capability, modem control)
  - Up to 2 SPIs (12 Mbit/s)
  - Consumer electronics control (CEC) interface
- CRC calculation unit, 96-bit unique ID
- ECOPACK® packages

Table 1. Device summary

Reference	Part number
STM32F100x4	STM32F100C4, STM32F100R4
STM32F100x6	STM32F100C6, STM32F100R6
STM32F100x8	STM32F100C8, STM32F100R8, STM32F100V8
STM32F100xB	STM32F100CB, STM32F100RB, STM32F100VB

5.3.13 I/O port characteristics

General input/output characteristics

Unless otherwise specified, the parameters given in *Table 34* are derived from tests performed under the conditions summarized in *Table 8*. All I/Os are CMOS and TTL compliant.

Table 34. I/O static characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	Standard I/O input low level voltage		–0.3		0.28*(V_{DD}–2 V)+0.8 V	V
	I/O FT(1) input low level voltage		–0.3		0.32*(V_{DD}–2 V)+0.75 V	
V_{IH}	Standard I/O input high level voltage		0.41*(V_{DD}–2 V) +1.3 V		V_{DD}+0.3	
	I/O FT(1) input high level voltage	V_{DD} > 2 V	0.42*(V_{DD}–2)+1 V		5.5	
		V_{DD} ≤ 2 V			5.2	
V_{hys}	Standard I/O Schmitt trigger voltage hysteresis(2)		200			mV
	I/O FT Schmitt trigger voltage hysteresis(2)		5% V_{DD}(3)			mV
I_{lkg}	Input leakage current(4)	V_{SS} ≤ V_{IN} ≤ V_{DD} Standard I/Os			±1	µA
		V_{IN} = 5 V I/O FT			3	
R_{PU}	Weak pull-up equivalent resistor(5)	V_{IN} – V_{SS}	30	40	50	kΩ
R_{PD}	Weak pull-down equivalent resistor(5)	V_{IN} – V_{DD}	30	40	50	kΩ
C_{IO}	I/O pin capacitance			5		pF

1. FT = 5V tolerant. To sustain a voltage higher than V_{DD}+0.3 the internal pull-up/pull-down resistors must be disabled.
2. Hysteresis voltage between Schmitt trigger switching levels. Guaranteed by design, not tested in production.
3. With a minimum of 100 mV.
4. Leakage could be higher than max. if negative current is injected on adjacent pins.
5. Pull-up and pull-down resistors are designed with a true resistance in series with a switchable PMOS/NMOS. This PMOS/NMOS contribution to the series resistance is minimum (~10% order).

All I/Os are CMOS and TTL compliant (no software configuration required). Their characteristics cover more than the strict CMOS-technology or TTL parameters. The coverage of these requirements is shown in *Figure 22* and *Figure 23* for standard I/Os, and in *Figure 24* and *Figure 25* for 5 V tolerant I/Os.

Table 42. ADC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDA}	Power supply		2.4		3.6	V
V_{REF+}	Positive reference voltage		2.4		V_{DDA}	V
I_{VREF}	Current on the V_{REF} input pin			160(1)	220(1)	µA
f_{ADC}	ADC clock frequency		0.6		12	MHz
f_S(2)	Sampling rate		0.05		1	MHz
f_{TRIG}(2)	External trigger frequency	f_{ADC} = 12 MHz			823	kHz
					17	$1/f_{ADC}$
V_{AIN}(3)	Conversion voltage range		0 (V_{SSA} tied to ground)		V_{REF+}	V
R_{AIN}(2)	External input impedance	See *Equation 1* and *Table 43* for details			50	kΩ
R_{ADC}(2)	Sampling switch resistance				1	kΩ
C_{ADC}(2)	Internal sample and hold capacitor				8	pF
t_{CAL}(2)	Calibration time	f_{ADC} = 12 MHz	5.9			µs
			83			$1/f_{ADC}$
t_{lat}(2)	Injection trigger conversion latency	f_{ADC} = 12 MHz			0.214	µs
					3(4)	$1/f_{ADC}$
t_{latr}(2)	Regular trigger conversion latency	f_{ADC} = 12 MHz			0.143	µs
					2(4)	$1/f_{ADC}$
t_S(2)	Sampling time	f_{ADC} = 12 MHz	0.125		17.1	µs
			1.5		239.5	$1/f_{ADC}$
t_{STAB}(2)	Power-up time		0	0	1	µs
t_{CONV}(2)	Total conversion time (including sampling time)	f_{ADC} = 12 MHz	1.17		21	µs
			14 to 252 (t_S for sampling +12.5 for successive approximation)			$1/f_{ADC}$

1. Based on characterization results, not tested in production.
2. Guaranteed by design, not tested in production.
3. V_{REF+} can be internally connected to V_{DDA} and V_{REF-} can be internally connected to V_{SSA}, depending on the package. Refer to *Table 4: Low & medium-density STM32F100xx pin definitions* and *Figure 6* for further details.
4. For external triggers, a delay of $1/f_{PCLK2}$ must be added to the latency specified in *Table 42*.

Equation 1: R_{AIN} max formula:

$$R_{AIN} < \frac{T_S}{f_{ADC} \times C_{ADC} \times \ln(2^{N+2})} - R_{ADC}$$

The above formula (*Equation 1*) is used to determine the maximum external impedance allowed for an error below 1/4 of LSB. Here N = 12 (from 12-bit resolution).

Table 43. R_{AIN} max for f_{ADC} = 12 MHz(1)

T_s (cycles)	t_S (µs)	R_{AIN} max (kΩ)
1.5	0.125	0.4
7.5	0.625	5.9
13.5	1.125	11.4
28.5	2.375	25.2
41.5	3.45	37.2
55.5	4.625	50
71.5	5.96	NA
239.5	20	NA

1. Guaranteed by design, not tested in production.

Table 44. ADC accuracy - limited test conditions(1)(2)

Symbol	Parameter	Test conditions	Typ	Max	Unit
ET	Total unadjusted error	f_{PCLK2} = 24 MHz, f_{ADC} = 12 MHz, R_{AIN} < 10 kΩ, V_{DDA} = 3 V to 3.6 V V_{REF+} = V_{DDA} T_A = 25 °C Measurements made after ADC calibration	±1.3	±2.2	LSB
EO	Offset error		±1	±1.5	
EG	Gain error		±0.5	±1.5	
ED	Differential linearity error		±0.7	±1	
EL	Integral linearity error		±0.8	±1.5	

1. ADC DC accuracy values are measured after internal calibration.
2. Based on characterization, not tested in production.

Table 45. ADC accuracy(1) (2) (3)

Symbol	Parameter	Test conditions	Typ	Max	Unit
ET	Total unadjusted error	f_{PCLK2} = 24 MHz, f_{ADC} = 12 MHz, R_{AIN} < 10 kΩ, V_{DDA} = 2.4 V to 3.6 V T_A = Full operating range Measurements made after ADC calibration	±2	±5	LSB
EO	Offset error		±1.5	±2.5	
EG	Gain error		±1.5	±3	
ED	Differential linearity error		±1	±2	
EL	Integral linearity error		±1.5	±3	

1. ADC DC accuracy values are measured after internal calibration.
2. Better performance could be achieved in restricted V_{DD}, frequency, V_{REF} and temperature ranges.
3. Based on characterization, not tested in production.

ADC accuracy vs. negative injection current: Injecting a negative current on any analog input pins should be avoided as this significantly reduces the accuracy of the conversion being performed on another analog input. It is recommended to add a Schottky diode (pin to ground) to analog pins which may potentially inject negative currents.
Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in *Section 5.3.12* does not affect the ADC accuracy.

7 General-purpose and alternate-function I/Os (GPIOs and AFIOs)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

7.1 GPIO functional description

Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), two 32-bit data registers (GPIOx_IDR, GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR) and a 32-bit locking register (GPIOx_LCKR).

Subject to the specific hardware characteristics of each I/O port listed in the *datasheet*, each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed). The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. This way, there is no risk that an IRQ occurs between the read and the modify access.

Figure 11 shows the basic structure of an I/O Port bit.

10 Analog-to-digital converter (ADC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This Section applies to the whole STM32F100xx family, unless otherwise specified.

10.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 multiplexed channels allowing it measure signals from 16 external and two internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.

The ADC input clock is generated from the PCLK2 clock divided by a prescaler , refer to *Figure 8: STM32F100xx clock tree (low and medium-density devices)* and *Figure 9: STM32F100xx clock tree (high-density devices).*

10.2 ADC main features

- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel 'n'
- Self-calibration
- Data alignment with in-built data coherency
- Channel by channel programmable sampling time
- External trigger option for both regular and injected conversion
- Discontinuous mode
- ADC conversion time:
  - STM32F100xx value line devices: 1.17 µs at 24 MHz
- ADC supply requirement: 2.4 V to 3.6 V
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion

The block diagram of the ADC is shown in *Figure 24.*

Note: V_{REF-} if available (depending on package), must be tied to V_{SSA}.

10.6 Channel-by-channel programmable sample time

ADC samples the input voltage for a number of ADC_CLK cycles which can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sample time.

The total conversion time is calculated as follows:

Tconv = Sampling time + 12.5 cycles

Example:

With an ADCCLK = 12 MHz and a sampling time of 1.5 cycles:

Tconv = 1.5 + 12.5 = 14 cycles = 1.17 µs

10.7 Conversion on external trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTTRIG control bit is set then external events are able to trigger a conversion. The EXTSEL[2:0] and JEXTSEL[2:0] control bits allow the application to select decide which out of 8 possible events can trigger conversion for the regular and injected groups.

Note: When an external trigger is selected for ADC regular or injected conversion, only the rising edge of the signal can start the conversion.

Table 59. External trigger for regular channels for ADC1

Source	Type	EXTSEL[2:0]
TIM1_CC1 event	Internal signal from on-chip timers	000
TIM1_CC2 event		001
TIM1_CC3 event		010
TIM2_CC2 event		011
TIM3_TRGO event		100
TIM4_CC4 event		101
EXTI line 11	External pin	110
SWSTART	Software control bit	111

Table 60. External trigger for injected channels for ADC1

Source	Connection type	JEXTSEL[2:0]
TIM1_TRGO event	Internal signal from on-chip timers	000
TIM1_CC4 event		001
TIM2_TRGO event		010
TIM2_CC1 event		011
TIM3_CC4 event		100
TIM4_TRGO event		101
EXTI line 15	External pin	110
JSWSTART	Software control bit	111

13.2 TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

23 Universal synchronous asynchronous receiver transmitter (USART)

23.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.

High speed data communication is possible by using the DMA for multibuffer configuration.

23.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Fractional baud rate generator systems
  - A common programmable transmit and receive baud rates up to 4.5 MBits/s
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- LIN Master Synchronous Break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- Transmitter clock output for synchronous transmission
- IrDA SIR Encoder Decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard Emulation Capability
  - The Smartcard interface supports the asynchronous protocol Smartcards as defined in ISO 7816-3 standards
  - 0.5, 1.5 Stop Bits for Smartcard operation
- Single wire half duplex communication
- Configurable multibuffer communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for Transmitter and Receiver

- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- Ten interrupt sources with flags:
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error
  - Framing error
  - Noise error
  - Parity error
- Multiprocessor communication - enter into mute mode if address match does not occur
- Wake up from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9th bit), Idle line

23.3 USART functional description

The interface is externally connected to another device by three pins (see *Figure 243*). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its IO port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this IO is used to transmit and receive the data (at USART level, data are then received on SW_RX).