

TUGAS AKHIR

MODUL PEMBELAJARAN RASPBERRY PI

Diajukan untuk memenuhi syarat
Memperoleh gelar Sarjana Teknik pada
Program Studi Teknik Elektro



Disusun oleh:

FRENDY CHRISTIAN

NIM : 135114037

JURUSAN TEKNIK ELEKTRO
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2017

FINAL PROJECT

RASPBERRY PI LEARNING MODULE

In a partial fulfillment of the requirements
for the degree of Sarjana Teknik
Electrical Engineering Study Program

Created by:

FRENDY CHRISTIAN

Student's Number: 135114037

**ELECTRICAL ENGINEERING STUDY PROGRAM
DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF SCIENCE AND TECHNOLOGY
SANATA DHARMA UNIVERSITY
YOGYAKARTA
2017**

HALAMAN PERSETUJUAN

TUGAS AKHIR

MODUL PEMBELAJARAN RASPBERRY PI
(RASPBERRY PI LEARNING MODULE)



HALAMAN PENGESAHAN

TUGAS AKHIR

**MODUL PEMBELAJARAN RASPBERRY PI
(RASPBERRY PI LEARNING MODULE)**

Disusun oleh:

FRENDY CHRISTIAN

NIM: 135114037

Telah dipertahankan di depan tim penguji

pada tanggal 20 Juli 2017

dan dinyatakan memenuhi syarat

Susunan Tim Penguji:

Nama Lengkap

Ketua : Ir. Tjendro, M.Kom

Sekertaris : Martanto, S.T., M.T.

Anggota : Djoko Untoro Suwarno, S.Si.,M.T.

Tanda-Tangan

Yogyakarta, 31 Juli 2017

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

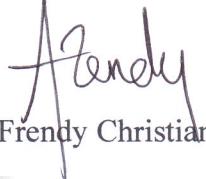
Dekan,

Sudi Mungkasi, S.si, M.Math.Sc., Ph.D.

PERNYATAAN KEASLIAN KARYA

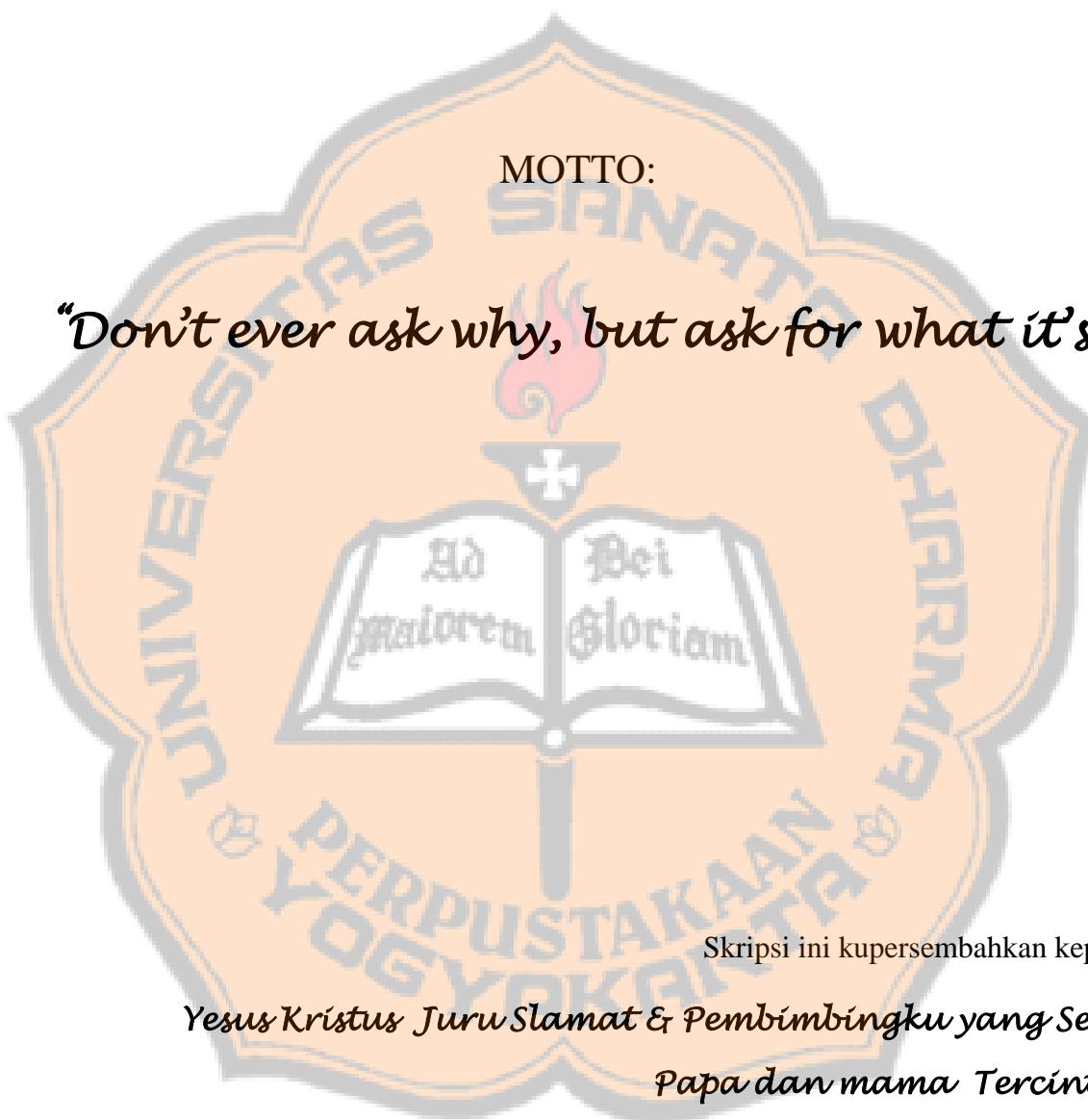
Saya menyatakan dengan sesungguhnya bahwa Tugas Akhir ini tidak memuat karya atau bagian karya orang lain, kecuali yang tidak disebutkan dalam kutipan dan daftar pustaka sebagaimana karya ilmiah.

Yogyakarta, 30 Mei 2017


Frendy
Christian



HALAMAN PERSEMBAHAN DAN MOTO HIDUP



MOTTO:

"Don't ever ask why, but ask for what it's"

Skripsi ini kupersembahkan kepada

*Jesus Kristus Juru Slamat & Pembimbingku yang Setia
Papa dan mama Tercinta,*

Ryan, levina dan suci.

*Teman -teman seluruh elektro,
Sahabat dan Teman-teman Seperjuangan
Almamater Tercinta, Universitas Sanata Dharma.*

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA
ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Frendy Christian

Nomor Mahasiswa : 135114037

Demi pengembangan ilmu pengetahuan, saya memberikan kepada perpustakaan Universitas Sanata Dharma yang berjudul:

MODUL PEMBELAJARAN RASPBERRY PI

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelolanya dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini yang saya buat dengan sebenarnya.

Yogyakarta, 30 Mei 2017

Frendy Christian

INTISARI

Perkembangan teknologi yang sangat pesat di era globalisasi saat ini telah memberikan banyak manfaat dalam kemajuan di dalam kehidupan manusia. Perkembangan teknologi yang pesat harus diikuti dengan perkembangan pada sumber daya manusia juga, sehingga dunia pendidikan yang menjadi tempat menuntut ilmu harus juga mengikuti perkembangan teknologi. *Raspberry pi* merupakan salah satu teknologi baru saat ini yang akan digunakan sebagai modul pembelajaran *raspberry pi*.

Modul pembelajaran *raspberry pi* membutuhkan sebuah monitor dan *keyboard* untuk membantu pengoperasian modul pembelajaran. Fitur-fitur yang terdapat pada modul pembelajaran yaitu project board, LED (*Light Emitting Dioda*), LCD (*Liquid Crystal Display*), *Push button*, saklar *toggle*, *keypad 4x4*, *dot matrix LED 7x5*, *seven segment*, motor *stepper*, motor *servo*, *power supply*, modul RTC (*real time clock*) ds 3231, dan *raspberry pin out*. Modul pembelajaran *raspberry pi* menggunakan program *python* sebagai pengoperasian fitur-fitur yang digunakan.

Hasil dari penelitian ini adalah modul pembelajaran dan beserta fitur-fitur yang digunakan dapat bekerja dengan baik menggunakan program *python*.

Kata kunci : modul pembelajaran *raspberry pi*, *raspberry pi*.

ABSTRACT

In the current globalization era, the development of technology has provided many benefits in the advancement of human life. The technology which grows rapidly in this world must be followed by the development of the human resources as well. Therefore, to make it balance, the education field which becomes the place to harvest knowledge must be supported by some technologies also. Raspberry pi is one of the new technologies which is currently being used as a learning module.

The raspberry pi learning module requires a monitor and keyboard to aid the implementation of the learning module itself. The features which are included in the *Raspberry pi* learning module are project board, LED (Light Emitting Diode), LCD (Liquid Crystal Display), Push button, toggle switch, 4x4 keypad, 7x5 LED dot matrix, seven segment, stepper motor, servo motor, power Supply, RTC module (real time clock) ds 3231, and raspberry pin out. The raspberry pi learning module uses the python program in operating the features used in the learning module.

The results of this research show that the *Raspberry pi* learning module and the features used in the module can work well using the python program.

Keywords: *Raspberry pi* learning module, *Raspberry pi*.

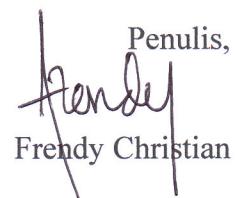
KATA PENGANTAR

Puji syukur dan terima kasih penulis ucapkan kepada Allah Tri Tunggal Maha kudus atas segala berkat dan penyertaan-Nya sehingga penulis dapat melewati proses penulisan dan menyelesaikan laporan Tugas Akhir. Tugas Akhir ini bertujuan untuk memenuhi syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Sanata Dharma.

Dalam menyelesaikan laporan Tugas Akhir ini penulis mendapatkan bantuan, arahan, bimbingan, dan motivasi dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Drs. Johannes Eka Priyatma, M.Sc., Ph.D. selaku Rektor Universitas Sanata Dharma yang telah memberikan kesempatan penulis untuk menimba ilmu dan berkembang.
2. Sudi Mungkasi, S.si, M.Math.Sc., Ph.D., selaku Dekan Fakultas Sains dan Teknologi Universitas Sanata Dharma.
3. Petrus Setyo Prabowo, S.T., M.T., selaku Ketua Program Studi Teknik Elektro Fakultas Sains dan Teknologi Universitas Sanata Dharma.
4. Martanto, S.T., M.T., selaku dosen pembimbing yang telah memberikan masukan, bimbingan, saran untuk penyusunan laporan Tugas Akhir ini.
5. Segenap dosen Fakultas Sains dan Teknologi yang telah membagikan ilmu selama penulis menempuh Pendidikan di Program Studi Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Sanata Dharma.
6. Segenap staf Sekretariat Fakultas Sains dan Teknologi yang telah membantu penulis mengurus segala sesuatu hal yang berkaitan dengan Tugas Akhir ini dan membantu serta melayani mahasiswa.
7. Semua pihak yang secara langsung dan tidak langsung mendukung, membantu, dan memberikan doa untuk kelancaran penulisan Tugas Akhir ini.

Pada akhirnya, penulis sangat menyadari bahwa laporan Tugas Akhir ini masih terdapat banyak kekurangan. Oleh karena itu, penulis sangat mengharapkan, kritik dan saran yang sifatnya membangun agar laporan Tugas Akhir ini nantinya bisa menjadi lebih baik dan dapat bermanfaat sebagaimana mestinya.

Penulis,

Frendy Christian

DAFTAR ISI

HALAMAN JUDUL (Bahasa Indonesia)	i
HALAMAN JUDUL (Bahasa Inggris)	ii
HALAMAN PERSETUJUAN	iii
HALAMAN PEENGESAHAN	iv
PERNYATAAN KEASLIAN KARYA	v
HALAMAN PERSEMBAHAN DAN MOTTO HIDUP	vi
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS	vii
INTISARI	viii
ABSTRACT	ix
KATA PENGANTAR	x
DAFTAR ISI	xi
DAFTAR GAMBAR	xiv
DAFTAR TABEL	xvii
 BAB I PENDAHULUAN	
1.1. Latar Belakang	1
1.2. Tujuan dan Manfaat	2
1.3. Batasan Masalah	3
1.4. Metodologi Penelitian	3
 BAB II DASAR TEORI	
2.1. Raspberry Pi	5
2.2. LED (Light Emitting Dioda)	6
2.3. LCD (Liquid Crystal Display)	8
2.4. Push button	9
2.5. Saklar Toggle	10
2.6. Keypad 4x4	10
2.7. Dot matrix LED 7x5	11
2.8. Seven segment	11

2.9. Motor Servo	12
2.10. Motor stepper.....	13
2.11. Driver Motor stepper (ULN2003).....	15
2.12. Eight Channel Logic Level Converter	15
2.13. Modul RTC (Real Time Clock) DS 3231	16
2.14. Pull Up Resistors	17
2.15. Komunikasi Serial.....	17
2.15.1. I2C.....	17
2.15.2. UART	18

BAB III PERANCANGAN

3.1. Proses Kerja Sistem	20
3.2. Perancangan Perangkat Keras	20
3.2.1.Desain boks Modul Pembelajaran Raspberry pi 3	20
3.2.2. Rancangan Pinout Raspberry pi 3	21
3.2.3. Rangkaian LED (Light Emitting Dioda)	23
3.2.4. Rangkaian LCD (Liquid Crystal Display).....	23
3.2.5. Rangkaian Motor Servo.....	24
3.2.6. Rangkaian Driver dan Motor stepper	24
3.2.7. Rangkaian Keypad 4x4.....	25
3.2.8. Rangkaian Dot matrix LED 7x5.....	25
3.2.9. Rangkaian 4 Digit Seven Segment	26
3.2.10. Rangkaian Saklar Toggle	27
3.2.11. Rangkaian Push button	27
3.2.12. Rangkaian Komunikasi I2C	28
3.2.13. Rangkaian Komunikasi UART	28
3.2.14. Perancangan Power supply.....	29
3.3. Perancangan Perangkat Lunak.....	30
3.3.1. Diagram Alir Push button dan LED	30
3.3.2. Diagram Alir Saklar Toggle dan LED.....	31
3.3.3. Diagram Alir Keypad dan LCD	32
3.3.4. Diagram Alir Keypad dan Dot matrix LED	33
3.3.5. Diagram Alir Motor Servo	34

3.3.6. Diagram Alir Motor stepper	35
3.3.7. Diagram Alir Komunikasi I2C Modul RTC Ds 3231 dan Seven segment .	36
3.3.8. Diagram Alir Komunikasi UART	37

BAB IV HASIL DAN PEMBAHASAN

4.1. Bentuk Fisik Modul Pembelajaran Raspberry Pi	39
4.2. Cara Penggunaan Modul Pembelajaran Raspberry Pi	43
4.3. Pengujian Perangkat Modul Pembelajaran Raspberry Pi	44
4.3.1. Raspberry pi pinout	44
4.3.2. Push button dan LED	45
4.3.3. Pengujian Saklar Toggle dan LED	48
4.3.4. Pengujian Keypad dan LCD	50
4.3.5. Pengujian Keypad dan Dot matrix LED.....	52
4.3.6. Pengujian Motor Servo.....	55
4.3.7. Pengujian Motor Stepper.....	58
4.3.8. Pengujian Komunikasi <i>I2c</i> Modul RTC Ds 3231 dan Seven Segment.....	62
4.3.9. Penggunaan dan Pengujian komunikasi UART dan Arduino Uno	65
4.3.9. Pembahasan Fungsi Program Modul Pembelajaran Raspberry Pi	70

BAB V KESIMPULAN DAN SARAN

5.1. Kesimpulan.....	71
5.2. Saran	71

DAFTAR PUSTAKA	72
-----------------------------	----

LAMPIRAN

DAFTAR GAMBAR

	Halaman
Gambar 1.1 Diagram blok model perancangan	4
Gambar 2.1. Raspberry pi 3 model B	6
Gambar 2.2. Raspberry pi 3 pinout.	6
Gambar 2.3. LED (Light Emitting Dioda)	7
Gambar 2.4. Datasheet LED (Light Emitting Dioda).....	7
Gambar 2.5. LCD (Liquid Crystal Display).....	8
Gambar 2.6. LCD description	9
Gambar 2.7. Push button	9
Gambar 2.8. Saklar toggle.	10
Gambar 2.9. Susunan Matrix Keypad 4x4	10
Gambar 2.10. Dot matrix LED 7x5	11
Gambar 2.11. Dot matrix characteristics	11
Gambar 2.12. Seven segment	12
Gambar 2.13. Seven segment characteristics	12
Gambar 2.14. Motor servo 180°	13
Gambar 2.15. Motor stepper berdasarkan struktur rotor dan stator.....	14
Gambar 2.16. Motor stepper berdasarkan perancangan rangkaian pengendali	14
Gambar 2.17. Driver motor stepper ULN2003	15
Gambar 2.18. Eight channel bi-directional logic level converter.....	15
Gambar 2.19. Modul RTC DS 3231.....	16
Gambar 2.20. Datasheet RTC DS 3231.....	16
Gambar 2.21. Pull up resistors	17
Gambar 2.22. Prinsip komunikasi serial I2C.....	18
Gambar 2.23. Prinsip komunikasi serial UART	19
Gambar 3.1. Desain boks modul pembelajaran raspberry pi 3.....	21
Gambar 3.2. Rangkaian LED	23
Gambar 3.3. Rangkaian LCD (Liquid Crystal Display).....	24
Gambar 3.4. Rangkaian motor servo	24
Gambar 3.5. Rangkaian driver dan motor stepper.....	25
Gambar 3.6. Rangkaian keypad 4x4.....	25

Gambar 3.7. Rangkaian dot matrix LED 7x5	26
Gambar 3.8. Rangkaian 4 digit seven segment	26
Gambar 3.9. Rangkaian saklar toggle.....	27
Gambar 3.10. Rangkaian push button	27
Gambar 3.11. Rangkaian komunikasi I2C.....	28
Gambar 3.12. Rangkaian komunikasi UART.....	28
Gambar 3.13. Perancangan power supply	29
Gambar 3.14. Diagram alir push button dan LED.....	30
Gambar 3.15. Diagram alir saklar toggle dan LED	31
Gambar 3.16. Diagram alir keypad dan LCD.....	32
Gambar 3.17. Diagram alir keypad dan dot matrix LED	33
Gambar 3.18. Diagram alir motor servo.....	34
Gambar 3.19. Diagram alir motor stepper.....	35
Gambar 3.20. Diagram alir komunikasi modul RTC dan seven segment	36
Gambar 3.21. Diagram alir komunikasi UART raspberry pi	37
Gambar 3.22. Diagram alir komunikasi UART arduino	38
Gambar 4.1. Tampak atas boks modul raspberry pi	39
Gambar 4.2. Tampak belakang boks modul <i>raspberry pi</i>	40
Gambar 4.3. Tampak rangkaian dalam boks modul <i>raspberry pi</i> bagian 1	41
Gambar 4.4. Tampak rangkaian dalam boks modul <i>raspberry pi</i> bagian 2	41
Gambar 4.5. Pengujian <i>raspberry pi</i> pinout	44
Gambar 4.6. Rangkaian <i>push button</i> dan LED	45
Gambar 4.7. Pengkondisian <i>input</i> dan <i>output push button</i> dan LED	46
Gambar 4.8. Pengkondisian penyalaan <i>push button</i> dan LED	46
Gambar 4.9. Hasil rangkaian pengujian <i>push button</i> dan LED	47
Gambar 4.10. Rangkaian saklar <i>toggle</i> dan LED	48
Gambar 4.11. Hasil rangkaian pengujian saklar <i>toggle</i> dan LED	49
Gambar 4.12. Rangkaian <i>keypad</i> dan LCD 16x2	50
Gambar 4.13. Pengkondisian <i>input</i> dan <i>output keypad</i> dan LCD 16x2	51
Gambar 4.14. Pengkondisian <i>keypad</i> dan LCD16x2.....	51
Gambar 4.15. Hasil rangkaian pengujian <i>keypad</i> dan LCD 16x2	52
Gambar 4.16. Gambar 4.16. Rangkaian <i>keypad</i> dan <i>dot matrix LED</i>	53
Gambar 4.17. Pengkondisian <i>input</i> dan <i>output keypad</i> dan <i>dot matrix LED</i>	54

Gambar 4.18. Hasil pengujian <i>keypad</i> dan <i>dot matrix LED</i>	55
Gambar 4.19. Rangkaian pengujian motor servo	57
Gambar 4.20. Pengkondisian <i>output</i> dan keputusan posisi motor <i>servo</i>	57
Gambar 4.21. Hasil rangkaian pengujian motor <i>servo</i>	58
Gambar 4.22. Hasil pengujian motor servo.....	58
Gambar 4.23. Rangkaian pengujian motor <i>stepper</i>	59
Gambar 4.24. Pengkondisian <i>output</i> motor <i>stepper</i>	60
Gambar 4.25. Hasil rangkaian pengujian motor <i>stepper</i>	61
Gambar 4.26. Hasil pengujian motor <i>stepper</i>	61
Gambar 4.27. Hasil pengujian alamat I2c	62
Gambar 4.28. Rangkaian komunikasi I2c	63
Gambar 4.29. Pengkondisian komunikasi I2c	63
Gambar 4.30. Rangkaian hasil pengujian komunikasi <i>I2c</i>	64
Gambar 4.31. Menyalakan dan mengoperasikan program arduino.....	66
Gambar 4.32. Pengaturan baudrate komunikasi <i>raspberry pi</i>	66
Gambar 4.33. Data karakter yang akan dikirim ke arduino.....	67
Gambar 4.34. Data karakter yang diterima dari <i>raspberry pi</i>	67
Gambar 4.35. Data karakter yang dikirim ke <i>raspberry pi</i>	68
Gambar 4.36. Data karakter yang diterima dari arduino	69
Gambar 4.37. Hasil pengujian komunikasi <i>UART</i>	69

DAFTAR TABEL

	Halaman
Tabel 3.1 Rancangan <i>pinout raspberry pi 3</i>	22
Tabel 4.1 Rangkaian <i>eight channel LLC</i> dengan <i>raspberry pi</i>	42
Tabel 4.2 Perangkat modul pembelajaran <i>raspberry pi</i>	43
Tabel 4.3 <i>Library</i> yang digunakan pada modul pembelajaran <i>raspberry pi</i>	43
Tabel 4.4 Rangkaian pengujian <i>push button</i> dan LED	45
Tabel 4.5 Pengujian <i>push button</i> dan LED	47
Tabel 4.6 Rangkaian pengujian saklar <i>toggle</i> dan LED	48
Tabel 4.7 Pengujian saklar <i>toggle</i> dan LED	49
Tabel 4.8 Rangkaian pengujian <i>keypad</i> dan LCD 16x2	50
Tabel 4.9 Rangkaian pengujian <i>keypad</i> dan <i>dot matrix LED</i>	53
Tabel 4.10 Rangkaian pengujian motor <i>servo</i>	56
Tabel 4.11 Rangkaian pengujian motor <i>stepper</i>	60
Tabel 4.12 Rangkaian pengujian <i>I2c</i> dan <i>seven segment</i>	62
Tabel 4.12 (lanjutan)Rangkaian pengujian <i>I2c</i> dan <i>seven segment</i>	63
Tabel 4.13 Rangkaian komunikasi UART <i>raspberry pi</i> ke arduino.....	66
Tabel 4.14 Rangkaian komunikasi UART arduino ke <i>raspberry pi</i>	68
Tabel 4.15 Keterangan fungsi-fungsi <i>raspberry pi</i>	70

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi yang sangat pesat di era globalisasi saat ini telah memberikan banyak manfaat dalam kemajuan di dalam kehidupan manusia. Penggunaan teknologi oleh manusia dalam membantu meyelesaikan pekerjaan merupakan hal yang menjadi keharusan dalam kehidupan. Perkembangan teknologi yang pesat harus diikuti dengan perkembangan pada sumber daya manusia juga.

Manusia sebagai pengguna teknologi harus mampu memanfaatkan teknologi yang ada saat ini, maupun perkembangan teknologi yang akan datang selanjutnya. Adaptasi manusia dengan teknologi baru yang berkembang harus dilakukan melalui pendidikan. Hal ini dilakukan agar generasi penerus tidak tertinggal dalam hal perkembangan teknologi baru.

Salah satu teknologi yang saat ini sedang sedang berkembang yaitu *raspberry pi*. *Raspberry pi* merupakan *computer single-board* yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi. Untuk membantu mempelajari *raspberry pi* maka dibutuhkan sebuah media pembelajaran.

Media pembelajaran yang digunakan berupa modul pembelajaran *raspberry pi*. Modul pembelajaran *raspberry pi* merupakan merupakan media pembelajaran yang belum ada digunakan di kampus Sanata Dharma, yang mungkin dikarenakan teknologi ini masih baru dalam materi perkuliahan.

Pada proses pembelajaran diperlukan perkembangan dalam materi pembelajaran maupun media penunjang pembelajaran agar lebih kompeten. Pada proses pembelajaran mikro lanjut sebelumnya mahasiswa lebih banyak diajarkan dalam proses program dikarenakan implementasi yang masih terbatas. Mahasiswa dibuat berkelompok untuk menyelesaikan tugas *project* yang berdeda-beda dan pada batas ketentuan tugas *project* dikumpulkan. Pada proses tersebut mahasiswa hanya mengerjakan tugas *project* kelompoknya sendiri, sehingga mahasiswa kehilangan kesempatan untuk mencoba tugas *project* lainnya. Hasil pengamatan seperti modul pembelajaran PLC (*Programmable Logic Controller*) pada mata kuliah kendali terprogram Teknik Elektro di Sanata Dharma dalam

proses belajar mahasiswa sangat terbantu dengan adanya media pembelajaran yang menjadi implementasi dari hasil program yang dibuat dan melihat secara langsung hasil dari program yang dirancang. Pada proses materi pembelajaran harus dilakukan bertahap mulai dari pengenalan terhadap *raspberry pi*, memahami fungsi-fungsi program *python* dan pengaplikasian *raspberry pi*. Pengenalan terhadap *raspberry pi* seperti memahami *port digital input / output (GPIO)*, memahami fungsi-fungsi program *python* yang fungsi sebagai penghubung terhadap aplikasi *raspberry pi*. Pengaplikasian *raspberry pi* sebaiknya dimulai dari sederhana seperti mengatur *input / output* digitar menyalakan lampu, mengatur *input / output* yang ditampilkan LCD dan *dot matrix LED*, mengatur *input / output* menjalankan *motor servo* dan *stepper*, dan *input / output* untuk komunikasi *I2C* dan *UART*.

Berdasarkan permasalahan tersebut, penulis berusaha mengembangkan sebuah media pembelajaran agar dapat membantu proses pembelajaran *raspberry pi*. Media pembelajaran yang dibuat oleh penulis adalah berupa modul pembelajaran *raspberry pi* 3 model B.

Penelitian ini akan menggunakan *raspberry pi* sebagai pengendali utama. Pada modul pembelajaran tersebut dilengkapi peranti *input* dan *output*. Peranti *input* berupa *push button*, saklar, *keypad 4x4*. Peranti *output* berupa *motor servo*, *motor stepper*, *LED*, *LCD 16x2*, *seven segment*, *dot matrix*, komunikasi *USART* dan *I2C*. Diharapkan dengan fitur-fitur tersebut dapat membantu dalam mempelajari *raspberry pi*.

1.2. Tujuan dan Manfaat Penelitian

Tujuan penelitian ini ialah untuk menjadikan media pembelajaran *raspberry pi* yang diintegrasikan pada rangkaian masukan dan rangkaian keluaran. Rangkaian masukan terdiri dari saklar *toggle*, *push button*, *keypad* sedangkan rangkaian keluaran terdiri dari *LED*, *dot matrix LED*, *seven segment*, *motor servo*, *motor stepper* dan komunikasi Serial.

Manfaat penelitian ini adalah sebagai berikut :

1. Memperdalam pengetahuan penulis tentang *raspberry pi*.
2. Membantu proses pembelajaran tentang *raspberry pi* untuk mahasiswa Teknik Elektro Sanata Dharma.
3. Membantu masyarakat luas dalam mempelajari *raspberry pi*.

1.3. Pembatasan Masalah

Agar Tugas Akhir ini bisa mengarah pada tujuan dan untuk menghindari terlalu kompleksnya permasalahan yang muncul, maka perlu adanya batas-batasan masalah yang sesuai dengan judul dari tugas akhir ini. Adapun batasan masalah adalah seperti :

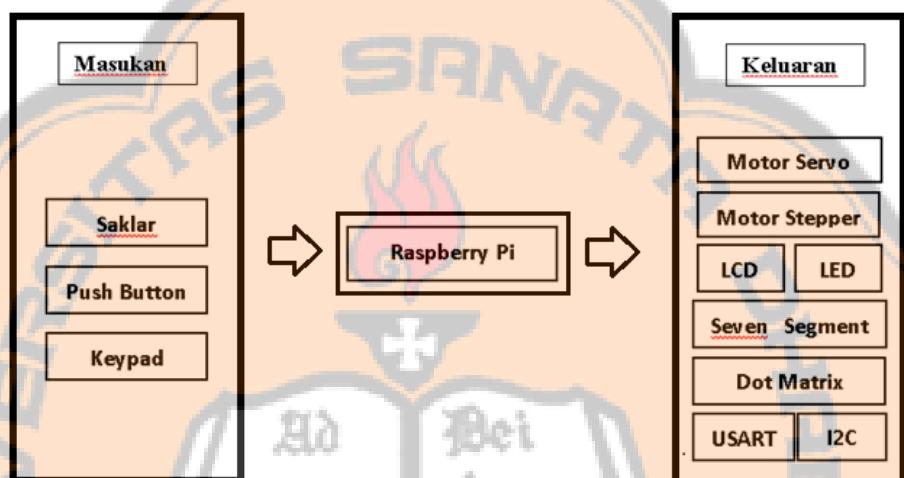
1. Menggunakan *Raspberry pi 3 Model B V 1.2* sebagai pusat pengendalian rangkaian dengan OS *raspbian GNU/LINUX* versi 8 (*jessie*), *Python 2.7.9*.
2. Delapan buah LED sebagai indikator.
3. LCD 16x2 sebagai penampil.
4. Delapan buah *push button* sebagai *input*.
5. Delapan buah saklar sebagai *input*.
6. *Keypad 4x4* sebagai *input*.
7. *Dot matrix LED 7x5* sebagai *output*.
8. Empat digit *seven segment* sebagai *output*.
9. Modul RTC (*Real Time Clock*) DS 3231
10. *Motor Servo 5 volt* sebagai *output*.
11. *Motor stepper 12 volt* sebagai *output*.
12. *Driver Motor stepper 12 volt* sebagai pengendali *Motor stepper*.
13. Tiga buah 8 Channel Bi-Directional Logic Level Converter sebagai pengubah logika tegangan.
14. Komunikasi Serial : USART dan I2C sebagai sarana komunikasi.

1.4. Metodologi Penelitian

Berdasarkan pada tujuan yang ingin dicapai metode-metode yang digunakan dalam penyusunan tugas akhir ini adalah :

1. Studi literatur, yaitu dengan cara mendapatkan data dengan membaca buku-buku dan jurnal-jurnal yang berkaitan dengan permasalahan yang dibahas dalam tugas akhir ini.
2. Perancangan subsistem *hardware*. Tahap ini bertujuan untuk mencari bentuk model yang optimal dari sistem yang akan dibuat dengan mempertimbangkan dari berbagai faktor-faktor permasalahan dan kebutuhan yang telah ditentukan. Gambar 1.1 memperlihatkan blok model yang dirancang.
3. Pembuatan subsistem *software*. Tahap ini bertujuan untuk membuat model yang sesuai dengan yang telah dirancang pada tahap sebelumnya.

4. Proses pengambilan data. Dilakukan dengan melihat hasil keluaran dari aplikasi pengujian alat menggunakan *raspberry pi* yang dilakukan seperti pengujian saklar *toggle* dan *pushbutton* dengan LED, *keypad 4x4* dengan LCD dan *dot matrix LED*, *raspberry pi* dengan *motor servo* dan *stepper*, dan komunikasi *I2C* dan *UART*.
5. Analisa dan penyimpulan hasil percobaan. Analisis data dilakukan dengan menganalisa *performance* dari fitur-fitur yang ada pada *hardware* ini.



Gambar 1.1 Diagram blok model perancangan

BAB II

DASAR TEORI

2.1. *Raspberry pi*

Raspberry pi merupakan komputer dalam satu singleboard. Operating System (OS) pada *raspberry pi* yaitu *Linux*. *Raspberry pi* memiliki beberapa seri seperti *raspberry pi* 1, 2, 3, model A, model A+, model B, model B+. Seri yang akan digunakan dalam penelitian kali ini adalah *Raspberry pi* 3 model B yang merupakan seri terbaru. Berikut ini adalah spesifikasi dari *Raspberry pi* 3 model B [1]:

a) Prosesor

System on a Chip (SoC) berupa jenis chip jenis *Broadcom BCM2837R*. CPU *4x ARM Cortex-A53*, kecepatan prosesor *1.2 GHz*, GPU berupa *Broadcom VideoCore IV*, dengan RAM *1 GB LPDDR2 (900 MHz)*.

b) Slot Secure Digital Card (SD Card)

Raspberry pi 3 model B dilengkapi dengan slot SD card sebagai *hard drive* untuk menyimpan seluruh data.

c) Port USB

Raspberry pi 3 model B mempunyai 4 port USB tipe 2.0.

d) Bluetooth

Raspberry pi 3 model B mempunyai jenis *Bluetooth 4.1 Classic* yang berfungsi sebagai media penghubung komunikasi dengan perangkat komunikasi lainnya.

e) Konektor HDMI

Raspberry pi 3 model B mempunyai port HDMI sebagai perantara *audio/video* yang akan ditampilkan pada sebuah monitor.

f) Output Audio Analog

Port *audio analog* berfungsi sebagai penyedia keluaran *audio analog* untuk disambungkan pada perangkat *speaker* dengan *jack* sebesar 3,5 mm.

Raspberry pi 3 model B terdapat 40 buah pin. Pin pada *raspberry pi* 3 terdiri dari beberapa bagian yaitu bagian VCC, GND, dan GPIO (*General Purpose Input/Output*). Terdapat 3 pin VCC dan 8 pin GND. Pin GPIO mulai dari GPIO2 hingga GPIO27, pada pin GPIO terdapat fungsi lain yang dapat dilihat pada gambar 2.1.

Gambar 2.1. *Raspberry pi 3 model B [1]*

Raspberry Pi 3 GPIO Header	
Pin#	NAME
01	3.3v DC Power
03	GPIO02 (SDA1 , I ² C)
05	GPIO03 (SCL1 , I ² C)
07	GPIO04 (GPIO_GCLK)
09	Ground
11	GPIO17 (GPIO_GEN0)
13	GPIO27 (GPIO_GEN2)
15	GPIO22 (GPIO_GEN3)
17	3.3v DC Power
19	GPIO10 (SPI_MOSI)
21	GPIO09 (SPI_MISO)
23	GPIO11 (SPI_CLK)
25	Ground
27	ID_SD (I ² C ID EEPROM)
29	GPIO05
31	GPIO06
33	GPIO13
35	GPIO19
37	GPIO26
39	Ground
29/02/2016	
NAME	Pin#
DC Power 5v	02
DC Power 5v	04
Ground	06
(TXD0) GPIO14	08
(RXD0) GPIO15	10
(GPIO_GEN1) GPIO18	12
Ground	14
(GPIO_GEN4) GPIO23	16
(GPIO_GEN5) GPIO24	18
Ground	20
(GPIO_GEN6) GPIO25	22
(SPI_CE0_N) GPIO08	24
(SPI_CE1_N) GPIO07	26
(I ² C ID EEPROM) ID_SC	28
Ground	30
GPIO12	32
Ground	34
GPIO16	36
GPIO20	38
GPIO21	40

Gambar 2.2. *Raspberry pi 3 pinout [29]*

2.2. LED (*Light Emitting Dioda*)

LED (*Light Emitting Dioda*) adalah *dioda* yang memancarkan cahaya pada saat mendapatkan arus bias maju. LED (*Light Emitting Dioda*) dapat memancarkan cahaya

karena menggunakan *dropping gallium*, *arsenic* dan *phosphorus*. Jenis *dropping* yang berbeda dapat menghasilkan cahaya dengan warna yang berbeda. LED merupakan salah satu jenis *dioda*, sehingga hanya akan mengalirkan arus listrik satu arah saja. LED yang digunakan memiliki kemampuan mengalirkan arus maksimal 20 mA, jika arus yang mengalir lebih dari 20 mA maka LED akan rusak, sehingga pada LED ditambahkan resistor untuk mengendalikan arus listrik yang mengalir [2].



Gambar 2.3. LED (*Light Emitting Dioda*) [2]

Electrical Characteristics: ($T_A = +25^\circ\text{C}$ unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Luminous Intensity	I_V	$I_F = 20 \text{ mA}$	0.9	3.0	-	med
Peak Wavelength	λ_p	$I_F = 20 \text{ mA}$	-	-	660	nm
Spectral Line Half Width	$\Delta\lambda$	$I_F = 20 \text{ mA}$	-	20	-	nm
Forward Voltage	V_F	$I_F = 20 \text{ mA}$	-	1.65	2.0	V
Reverse Current	I_h	$V_R = 5.0\text{V}$	-	-	100	λA
Reverse Voltage	λA	$I_R = 100 \lambda\text{A}$	-	5.0	-	V
Capacitance	C	$V = 0$	-	35	-	pF
Viewing Angle	2θ _{1/2}	Between 50% Points	-	60	-	degree
Rise Time	t_r	10% - 90% 50Ω	-	50	-	ns
Fall Time	t_f	90% - 10% 50Ω	-	50	-	ns

Gambar 2.4. Datasheet LED (*Light Emitting Dioda*) [30]

Besar arus maksimum pada LED adalah 20 mA, sehingga nilai resistor harus ditentukan. Dimana besar nilai resistor berbanding lurus dengan besar tegangan sumber yang digunakan[2]. Mencari besarnya nilai pada resistor dapat menggunakan persamaan berikut :

$$R = \frac{V_s - V_{led}}{I} \quad (2.1)$$

Keterangan:

R = nilai resistor (*ohm*)

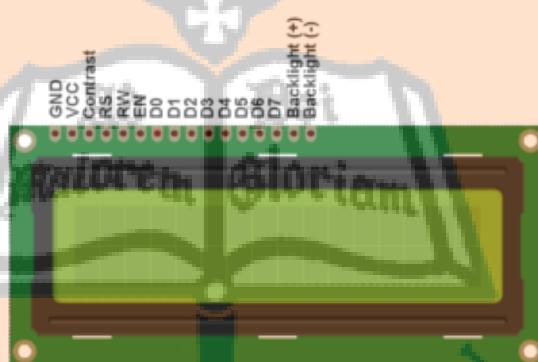
V_s = teganganan sumber yang digunakan (*volt*)

V_{led} = tegangan pada LED (*volt*)

I = arus maksimal pada LED (20 mA)

2.3. LCD (*Liquid Crystal Display*)

LCD (*Liquid Crystal Display*) adalah suatu alat yang berfungsi sebagai *display* elektronik yang dapat menampilkan data, karakter, angka, dan huruf. LCD (*Liquid Crystal Display*) memiliki 2 baris dan 16 kolom dan juga satu jenis *display* elektronik yang dibuat dengan teknologi CMOS *logic* yang bekerja dengan tidak menghasilkan cahaya tetapi memantulkan cahaya yang ada disekelilingnya terhadap *front-lit* atau mentranmisikan cahaya dari *back-lit*.



Gambar 2.5. LCD (*Liquid Crystal Display*) [4]

Pengendalian modul LCD (*Liquid Crystal Display*) terdapat mikrokontroller yang berfungsi sebagai pengendali tampilan karakter LCD. Mikrokontroller pada suatu LCD dilengkapi dengan memori dan register. Memori yang digunakan mikrokontroller LCD yaitu:

- a) DDRAM (*Display Data Random Access Memory*) merupakan memori tempat karakter yang akan ditampilkan berada.
- b) CGRAM (*Character Generator Random Access Memory*) merupakan memori untuk menggambarkan pola sebuah karakter dimana bentuk dari karakter dapat diubah-ubah sesuai dengan keinginan.

- c) CGROM (*Character Generator Read Only Memory*) merupakan memori untuk menggambarkan pola sebuah karakter dimana pola tersebut merupakan karakter dasar yang sudah ditentukan secara permanen oleh pabrikan pembuat LCD (*Liquid Crystal Display*) tersebut sehingga pengguna tinggal mangambilnya sesuai alamat memorinya dan tidak dapat merubah karakter dasar yang ada dalam CGROM [3].

Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V _{DD}		Power supply for logic for LCM
3	V ₀		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL

Gambar 2.6. LCD description [5]

2.4. Push button

Push button adalah saklar tekan yang berfungsi untuk menghubungkan arus listrik dan juga sebagai pemutus arus listrik DC (*Direct Current*). *Push button* akan terhubung atau bekerja selama tombol ditekan, ketika tombol sudah tidak ditekan maka *push button* akan kembali pada posisi awal atau tidak terhubung [6].



Gambar 2.7. Push button [7]

2.5. Saklar Toggle

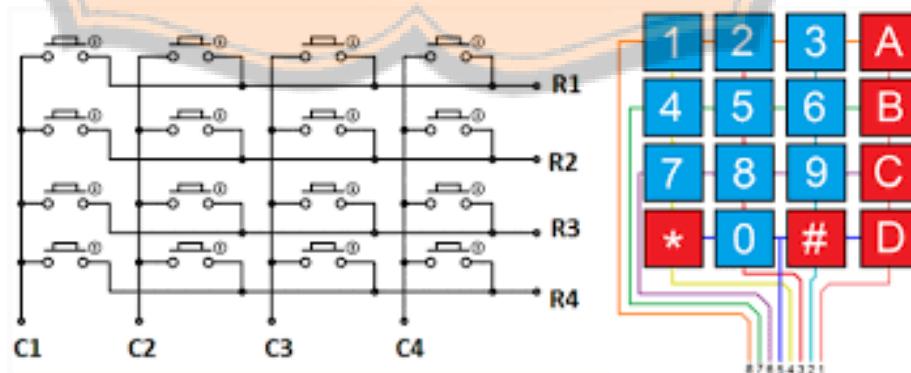
Saklar yang digunakan adalah saklar *on/off*, dimana saklar hanya memiliki 2 kondisi yaitu kondisi terhubung ON dan kondisi tidak terhubung OFF. Saklar ini dihubungkan dengan menggerakkan *toggle* atau tuas. Selama tuas dalam kondisi terhubung ON maka saklar akan menghubungkan arus listrik DC (*Direct Current*), sebaliknya jika tuas dalam kondisi tidak terhubung OFF maka tidak ada arus listrik DC (*Direct Current*) yang disalurkan [8].



Gambar 2.8. Saklar *toggle* [9]

2.6. Keypad 4x4

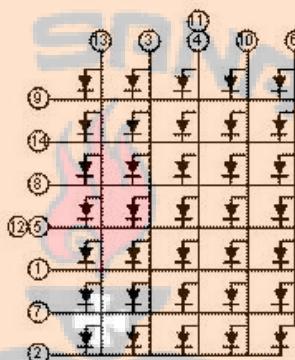
Keypad berfungsi sebagai *interface* antara perangkat elektronik dengan manusia atau dikenal dengan istilah HMI (*Human Machine Interface*). Matriks *keypad* 4x4 memiliki konstruksi atau susunan yang terdiri dari 4 baris dan 4 kolom dengan *keypad* berupa saklar *push button*. Rangkaian matriks *keypad* terdiri dari 16 saklar *push button* dengan pengaturan dari 4 baris dan 4 kolom. Proses untuk membaca penekanan tombol pada matriks *keypad* 4x4 dilakukan secara bertahap kolom demi kolom dari kolom pertama sampai kolom keempat dan dari baris pertama sampai baris keempat [10].



Gambar 2.9. Susunan Matrix Keypad 4x4 [11]

2.7. Dot matrix LED 7x5

Dot matrix LED yang digunakan pada penelitian ini dapat menghasilkan angka, huruf, dan karakter. Pada *dot matrix* ini memiliki 5 pin kolom dan 7 pin baris. Pada dasarnya *dot matrix* adalah *display* LED yang disusun. Dibutuhkan kombinasi tegangan antara pin baris dan kolom. *Dot matrix* LED terbagi atas 2 jenis yaitu *common anode* dan *common cathode*. Dimana *common anode* aktif saat diberi “0” dan *common cathode* aktif saat diberi “1” [12].



Gambar 2.10. *Dot matrix* LED 7x5 [12]

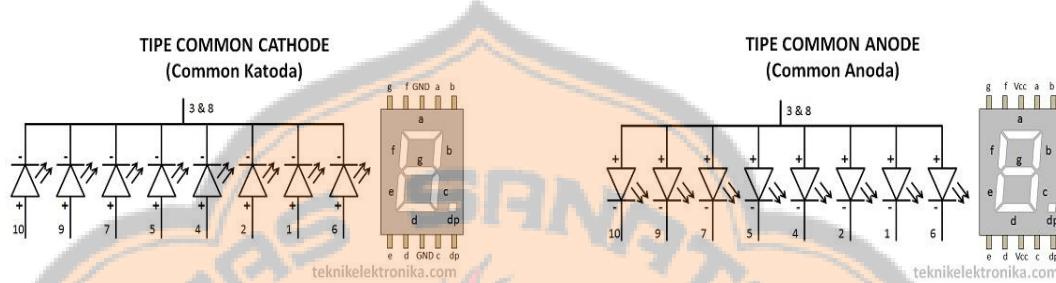
Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λ_{peak}	Peak Wavelength	High Efficiency Red	627		nm	$I_F=20mA$
λ_D [1]	Dominant Wavelength	High Efficiency Red	625		nm	$I_F=20mA$
$\Delta\lambda_{1/2}$	Spectral Line Half-width	High Efficiency Red	45		nm	$I_F=20mA$
C	Capacitance	High Efficiency Red	15		pF	$V_F=0V; f=1MHz$
V_F [2]	Forward Voltage	High Efficiency Red	2.0	2.5	V	$I_F=20mA$
I_R	Reverse Current	High Efficiency Red		10	uA	$V_R=5V$

Gambar 2.11. *Dot matrix characteristics* [13]

2.8. Seven segment

Seven segment display memiliki 7 segmen yang dikendalikan secara *on* dan *off*. setiap segmen tersebut biasa dikodekan dari huruf A hingga G. *Seven segment* terbagi menjadi 2 jenis yaitu *seven segment* tipe *common cathode* (Katoda) dan *seven segment* tipe *common anode* (Anoda).

Pada *common catnode* (Katoda), kaki katoda pada semua segmen LED terhubung menjadi 1 pin, sedangkan kaki anoda akan masukan untuk masing-masing segmen LED. Kaki katoda terhubung menjadi 1 pin yang terhubung terminal *negative* (-) sedangkan signal kendali (*control signal*) akan diberikan kepada masing-masing kaki anoda segmen LED. Sedangkan *seven segment* tipe *common anode* (Anoda) kebalikannya [14].



Gambar 2.12. Seven segment [14]

Seven segment yang digunakan adalah 4 digit yang sudah menjadi satu kesatuan, jumlah pin pada 4 digit *seven segment* berjumlah 12 pin. *Seven segment* pada setiap digit digabung menjadi satu rangkaian, selebihnya adalah pin yang menyalakan titik (dot) pada setiap digit. Pada *seven segment* ini terdiri dari 4 buah COM yang berfungsi untuk mengaktifkan digit yang akan digunakan.

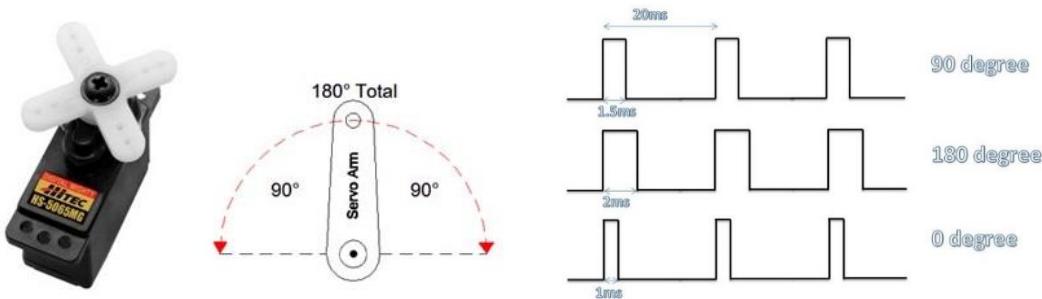
PARAMETER	RED	AMBER	GREEN	BLUE	WHITE	UNITS
DC Forward Current Per Segment	30	30	25	30	20	mA
Peak Current Per Segment ⁽¹⁾	70	50	50	25	25	mA
Avg. Forward Current (Pulse Operation) Per Segment	30	30	25	25	25	mA
Derating Linear From 25°C Per Segment			0.3			mA/°C
Reverse Voltage ⁽²⁾			3			V
Operating Temperature			-25 to +85			°C
Storage Temperature			-30 to +85			°C

(1) Pulse conditions of 1/10 duty and 0.1msec width, for long operating life, max. of 20mA recommended

Gambar 2.13. Seven segment characteristics [15]

2.9. Motor Servo

Motor servo terdiri dari beberapa bagian yaitu motor DC (*Direct Current*), serangkaian *gear*, potensiometer, dan rangkaian kontrol. Batas sudut dari putaran *servo* ditentukan dari potensiometer, sedangkan untuk mengatur sudut sumbu *motor servo* berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel *motor servo*. *Motor servo* yang digunakan memiliki gerakan hingga 180° [16].



Gambar 2.14. Motor servo 180° [16]

2.10. Motor stepper

Motor stepper dikendalikan dengan mengubah pulsa elektronis dan bergerak berdasarkan urutan pulsa yang diberikan pada *motor stepper*. *Motor stepper* merupakan salah satu jenis motor DC (*Direct Current*). Berdasarkan struktur rotor dan stator *motor stepper* terbagi menjadi 3 jenis yaitu :

a) *Motor stepper Variable Reluctance (VR)*

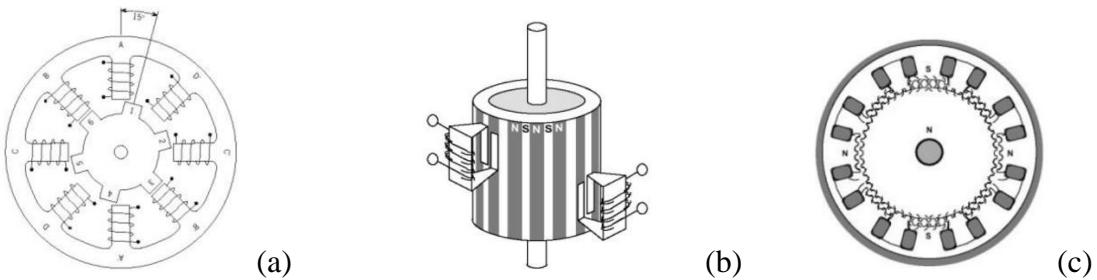
motor jenis ini terdiri dari sebuah rotor besi lunak dengan gerigi dan sebuah lilitan stator, cara kerja pada motor VR adalah ketika lilitan stator diberikan energi dengan arus DC maka kutub-kutub akan menjadi termagnetasi. Gigi-gigi rotor tertarik oleh kutub-kutub stator ketika terjadi perputaran [17].

b) *Motor stepper Permanent Magnet (PM)*

Pada jenis *motor stepper* ini memiliki rotor yang berbentuk seperti kaleng bulat yang dimana didalamnya terdiri atas magnet permanen yang disusun selang-seling dengan kutub yang berlawanan. Biasanya motor jenis ini memiliki resolusi langkah (*step*) yang rendah antara 7,5° hingga 15° per langkah atau 48 hingga 24 langkah setiap putarannya [17].

c) *Motor stepper Hybrid (HB)*

Motor stepper tipe *hybrid* memiliki struktur yang merupakan kombinasi dari kedua tipe *motor stepper* sebelumnya. *Motor stepper* juga memiliki gigi-gigi seperti pada *motor stepper* tipe VR dan juga memiliki magnet permanen yang tersusun menyerupai seperti *motor stepper* tipe PM. *Motor stepper* tipe *hybrid* dapat menghasilkan resolusi langkah yang lebih tinggi antara 0,9° hingga 3,6° per langkah atau 100 hingga 400 setiap putarannya [17].

Gambar 2.15. *Motor stepper* berdasarkan struktur rotor dan stator [17]

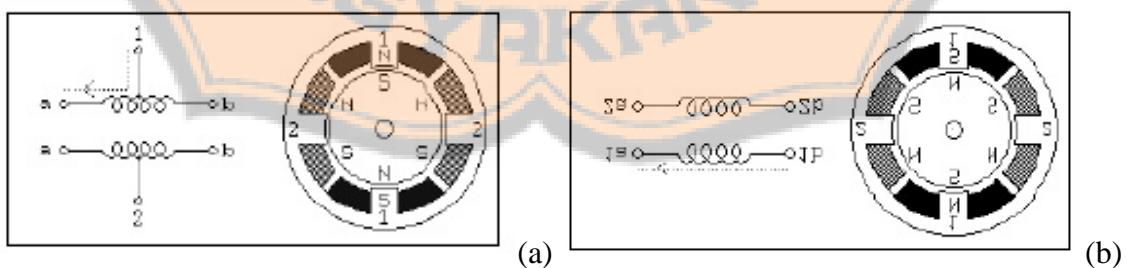
Berdasarkan metode perancangan rangkaian pengendalian *motor stepper* dibagi menjadi 2 jenis yaitu :

a) *Motor stepper Unipolar*

Motor stepper unipolar terdiri dari dua lilitan yang memiliki *center tap*. *Center tap* dari masing-masing lilitan ada yang berupa kabel terpisah, ada juga yang sudah terhubung di dalamnya sehingga *center tap* yang keluar hanya satu kabel. Untuk *motor stepper* yang *center tap*-nya ada pada masing-masing lilitan, kabel masukannya ada 6 kabel. Namun jika *center tap*-nya sudah terhubung di dalam, kabel *input*annya hanya 5 kabel. *Center tap* dari *motor stepper* dapat dihubungkan ke GND (*ground*) atau ada juga yang dihubungkan ke +Vcc, hal ini sangat dipengaruhi oleh *driver* yang digunakan [18].

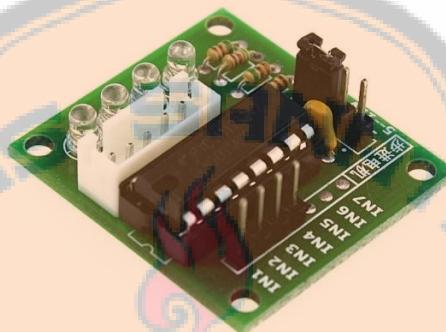
b) *Motor stepper Bipolar*

Motor stepper bipolar memiliki dua lilitan. Perbedaannya dengan tipe *unipolar* adalah bahwa pada tipe *bipolar* lilitannya tidak memiliki *center tap*. Keunggulan tipe *bipolar* yaitu memiliki torsi yang lebih besar jika dibandingkan dengan tipe *unipolar* untuk ukuran yang sama. *Motor stepper* tipe ini hanya memiliki empat kabel masukan [18].

Gambar 2.16. *Motor stepper* berdasarkan perancangan rangkaian pengendali [18]

2.11. Driver Motor stepper (ULN2003)

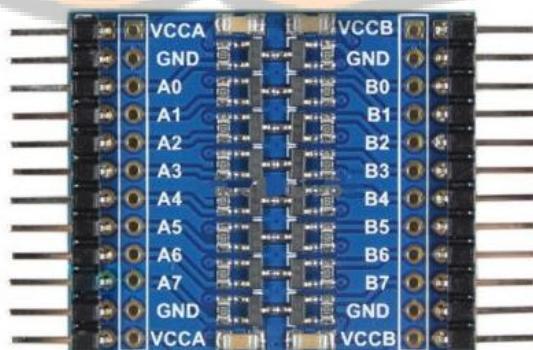
Pada modul ini beroperasi dengan catu daya TTL (5V), dan pada pin masukan kendali (*input control pins*) sudah dipasangkan resistor sebesar $2,7\text{ K}\Omega$ pada masing-masing pin, dimana untuk kendali hanya dibutuhkan arus sebesar 1,85 mA. Tegangan beban yang dapat dikendalikan pada modul ULN2003 mulai dari 5V hingga 50V DC (*Direct Current*) [19].



Gambar 2.17. Driver motor stepper ULN2003 [32]

2.12. Eight Channel Logic Level Converter

Eight channel logic level converter merupakan 8 saluran dua arah penjembatan dari ringkat logika yang berbeda. *Eight channel logic level converter* menggunakan kapasitor tantalum sebagai filter daya yang memberikan kestabilan lebih. Penyambungan untuk perubahan tegangan dari 3,3 volt menjadi 5 volt, maka VCCA/VA dihubungkan dengan 3,3 volt dan VCCB/VB dihubungkan dengan tegangan 5 volt sehingga ketika Ax mempunyai masukan TTL 3,3 volt, Bx akan mendapatkan keluaran TTL 5 volt dan sebaliknya ketika Bx mempunyai TTL 5 volt masukan, Ax akan mendapatkan keluaran TTL 3,3 volt. *Ground* pada masing-masing disambungkan menjadi satu bagian [20].



Gambar 2.18. Eight channel bi-directional logic level converter [31]

2.13. Modul RTC (*Real Time Clock*) DS 3231

Ds 3231 merupakan *I₂C real time clock* yang sangat akurat yang terintegrasi dengan Temperature Compensated Crystal Oscillator (TCXO) dan kristal. Pada perangkat Ds 3231 dilengkapi dengan baterai agar menjaga ketepatan waktu ketika tidak ada *power supply* pada perangkat ini. RTC pada modul dilengkapi penunjuk waktu detik, menit, jam, hari, tanggal, bulan, tahun dan juga *alarm* [21].



Gambar 2.19. Modul RTC DS 3231 [22]

Figure 1. Timekeeping Registers

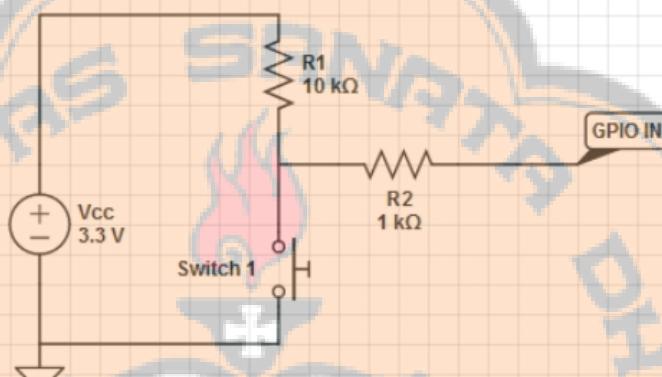
ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0		10 Seconds					Seconds	00-59	
01h	0		10 Minutes					Minutes	00-59	
02h	0	12/24	AM/PM	10 Hour				Hour	1-12 + AM/PM 00-23	
03h	0	0	0	0	0			Day	1-7	
04h	0	0	10 Date					Date	01-31	
05h	Century	0	0	10 Month				Month	01-12 + Century	
06h			10 Year					Year	00-99	
07h	A1M1		10 Seconds					Alarm 1 Seconds	00-59	
08h	A1M2		10 Minutes					Alarm 1 Minutes	00-59	
09h	A1M3	12/24	AM/PM	10 Hour				Alarm 1 Hours	1-12 + AM/PM 00-23	
0Ah	A1M4	DY/DT	10 Date					Alarm 1 Day	1-7	
0Bh	A2M2		10 Minutes					Alarm 1 Date	1-31	
0Ch	A2M3	12/24	AM/PM	10 Hour				Alarm 2 Hours	1-12 + AM/PM 00-23	
0Dh	A2M4	DY/DT	10 Date					Alarm 2 Day	1-7	
								Alarm 2 Date	1-31	
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.

Gambar 2.20. *Datasheet* RTC DS 3231 [23]

2.14. Pull Up Resistors

Pull Up resistors merupakan rangkaian yang berguna untuk mencegah terjadinya nilai yang mengambang (*float state*) antara “high” dan “low” [24]. Dalam rangkaian digital memiliki sumber tegangan (VCC) sebesar 5 volt atau 3,3 volt. Pada rangkaian digital 5 volt sinyal yang dibaca adalah “high” dan 0 volt dibaca sebagai “low” begitu juga dengan 3,3 volt. Pada penggunaan *push button* sebagai data masukan pada *raspberry pi* terkadang terjadi masalah nilai tidak terbaca sehingga digunakanlah rangkaian *pull up resistors* [25].

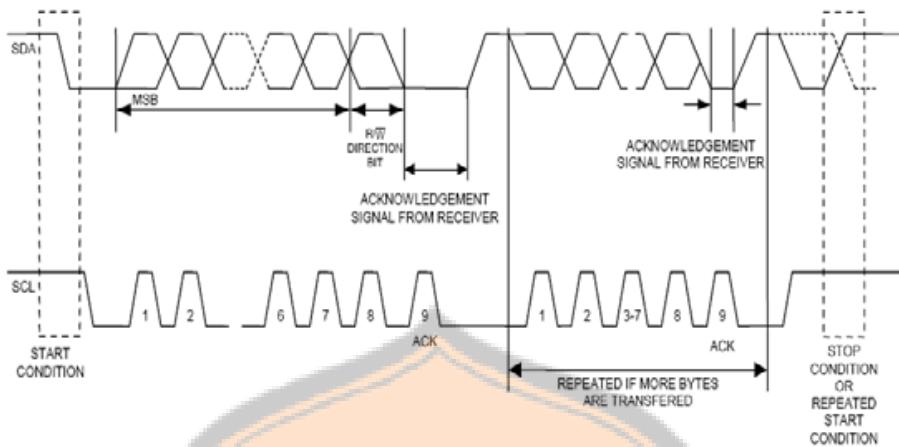


Gambar 2.21. *Pull up resistors* [25]

2.15. Komunikasi Serial

2.15.1. I2C

I2C singkatan dari *Inter Integrated Circuit*, adalah sebuah protokol untuk komunikasi serial antar IC, dan sering disebut juga *Two Wire Interface* (TWI). Komunikasi dilakukan melalui dua jalur: SDA (*serial data*) dan SCL (*serial clock*). Setiap *device* *I2C* memiliki 7-bit alamat. Sebagai contoh, 1010 biner ditujukan untuk serial EEPROM. Tiga bit berikutnya memungkinkan 8 kombinasi alamat *I2C*. Selama pengiriman data, saluran data (SDA) harus dalam keadaan stabil ketika saluran *clock* (SCL) dalam keadaan *high*. Perubahan kondisi SDA pada saat SCL *high* akan dianggap sebagai sinyal-sinyal kendali, seperti: sinyal *START* (*HIGH* ke *LOW*) atau sinyal *STOP* (*LOW* ke *HIGH*) [26].



Gambar 2.22. Prinsip komunikasi serial I2C [26]

2.15.2. UART

UART merupakan singkatan dari *Universal Asynchronous Receiver*, merupakan komunikasi serial tanpa menggunakan *clock* dari eksternal. Pada komunikasi *asynchronous* memiliki aturan dalam menjalankan komunikasi untuk memastikan pengiriman data tidak terjadi kesalahan. Urutan data komunikasi *asynchronous* yaitu, *start bit*, *data bit*, *parity bit*, *baud rate*, *stop bit* [27].

Start bit pada UART dimulai dari menerima tegangan tinggi untuk transisi tegangan rendah, saat keadaan itu memulai membaca *data bit* pada frekuensi *baud rate*. Nilai data bit berkisar dari 5 hingga 9 bit. *Parity bit* merupakan cara pada penerima UART untuk memberitahu jika terjadi perubahan data selama transmisi. Bit dapat berubah disebabkan oleh radiasi elektromagnetik, ketidak cocokan *baud rate*, atau jarak pengiriman data. Penentuan *parity bit* ditentukan dengan data yang dikirim jika 1 bit berjumlah genap, maka *parity bit* yang dicantumkan adalah 0 sebaliknya jika 1 bit berjumlah ganjil maka *parity bit* yang dicantumkan adalah 1 [28].

Baud rate menentukan seberapa cepat data dikirim melalui komunikasi serial. Syarat dalam komunikasi adalah kedua perangkat pengirim dan penerima *baud rate* diatur sama. Nilai kecepatan *baud rate* terdiri dari 9600, 1200, 2400, 4800, 19200, 38400, 57600, dan 115200 bps. Untuk hal-hal sederhana *baud rate* yang digunakan 9600 bps dan selebihnya merupakan standart *baud rate*. Semakin tinggi *baud rate* maka semakin cepat data terkirim / diterima, tetapi ada batas untuk cepat data yang dikirim tidak akan lebih dari 115200 bps dikarenakan terlalu cepat *baud rate* tersebut maka akan terlihat *error* saat data diterima [29].

Untuk mengakhiri pengiriman data pada komunikasi UART adalah pada bagian *stop bit*. Menandakan akhir dari pengiriman data dilakukan transmisi data dari tegangan rendah ke tegangan tinggi [29].



Gambar 2.23. Prinsip komunikasi serial UART [27]



BAB III

RANCANGAN PENELITIAN

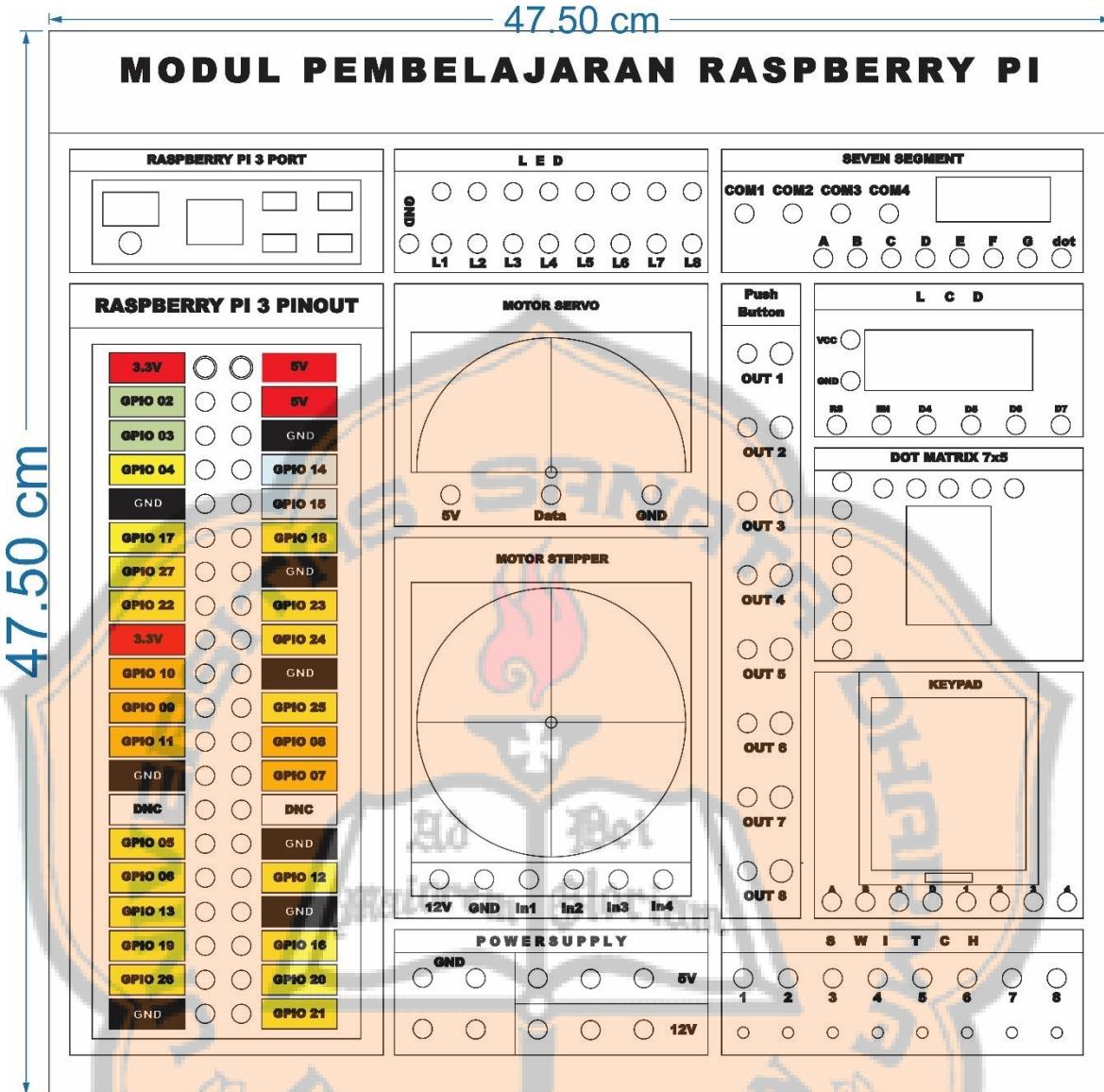
3.1. Proses Kerja Sistem

Perancangan alat ini terdiri dari beberapa bagian utama, yaitu *Raspberry pi*, saklar, *push button*, *keypad*, *motor servo*, *motor stepper*, *LED*, *LCD*, *seven segment*, *dot matrix* LED, komunikasi serial *I2C*, *UART*, *driver motor stepper* dan penyearah gelombang. *Raspberry pi* berfungsi sebagai pusat pemograman untuk mengontrol keluaran yang akan dioperasikan. Masukan pada operasi pemograman berupa saklar, *push button*, dan *keypad*. Sedangkan keluaran pada operasi pemograman berupa *motor servo*, *motor stepper*, *LED*, *LCD*, *seven segment*, dan *dot matrix* LED. Komunikasi serial *I2C* dan *UART*. Implementasi komunikasi *I2C* yaitu membaca data waktu yang telah diatur kemudian ditampilkan pada *4 digit seven segment* sedangkan *UART* diaplikasikan ke *arduino* dengan komunikasi dua arah sebagai pengirim dan sebagai penerima. Penyearah gelombang digunakan dapat menghasilkan tegangan DC 5 volt dan 12 volt. Tegangan DC 5 volt merupakan *power supply* untuk menyalakan *raspberry pi 3* dan tegangan Dc 12 volt merupakan *power supply* untuk menyalakan *driver motor stepper*.

3.2. Perancangan Perangkat Keras

3.2.1. Desain boks Modul Pembelajaran *Raspberry pi 3*

Desain boks modul pembelajaran *raspberry pi 3* menggunakan kata akrilik dengan ketebalan 5 mm. Desain boks pada modul pembelajaran memiliki panjang 47,5 cm dan lebar 47,5 cm. Modul pembelajaran *raspberry pi 3* terdiri dari 2 bagian yaitu, bagian *input* dan bagian *output*. Bagian *input* terdiri dari saklar *toggle*, *push button* dan *keypad 4x4* sedangkan bagian *output* terdiri dari *seven segment*, *LCD (Liquid Crystal Display)*, *Dot matrix 7x5*, *motor servo* dan *motor stepper*. Peletakan *raspberry pi 3* dirangkai didalam boks dan rangkaian pada modul pembelajaran *raspberry pi 3* sudah terpasang didalam boks. *Raspberry pi 3* dilengkapi dengan *port-port* yang membantu pengoperasian, *port-port* tersebut akan disambungkan ke boks modul.



Gambar 3.1. Desain boks modul pembelajaran *raspberry pi* 3

3.2.2. Rancangan Pinout Raspberry pi 3

Pada gambar desain modul pembelajaran *raspberry pi* 3 terlihat *pinout* atau keluaran dari kaki-kaki *raspberry pi* 3 dari dalam boks. *Pinout* yang keluar berasal dari 8 *channel bi-directional logic level converter* yang telah dihubungkan dengan *raspberry pi* 3. *Eight channel bi-directional logic level converter* berfungsi sebagai pengubah *level tegangan*, yang awalnya tegangan pada pin *raspberry pi* 3 adalah 3,3 volt akan diubah menjadi tegangan 5 volt. Dapat dilihat pada table 3.1 rancangan *pinout raspberry pi* 3.

Tabel 3.1. Rancangan pinout raspberry pi 3

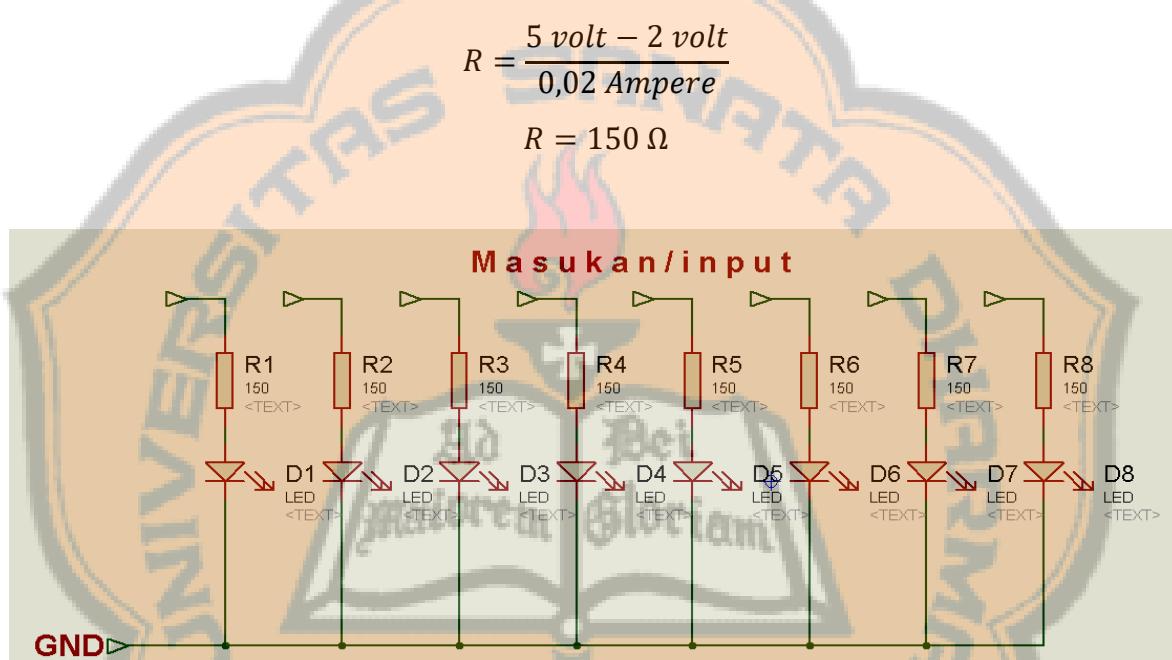
<i>Bx connect to Boks Hardware</i>	<i>Connect to Ax 8 Channel Bi-Directional LLC</i>	<i>Raspberry pi 3 Pinout</i>				<i>Connect to Ax 8 Channel Bi-Directional LLC</i>	<i>Bx connect to Boks Hardware</i>
3,3 V	—	3,3 V	1	2	5 V	—	5 V
GPIO2	A0 (1)	GPIO2	3	4	5 V	—	5 V
GPIO3	A1 (1)	GPIO3	5	6	GND	—	GND
GPIO4	A2 (1)	GPIO4	7	8	GPIO14	A4 (2)	GPIO14
GND	—	GND	9	10	GPIO15	A5 (2)	GPIO15
GPIO17	A7 (2)	GPIO17	11	12	GPIO18	A0 (3)	GPIO18
GPIO27	A1 (4)	GPIO27	13	14	GND	—	GND
GPIO22	A4 (3)	GPIO22	15	16	GPIO23	A5 (3)	GPIO23
3,3 V	—	3,3 V	17	18	GPIO24	A6 (3)	GPIO24
GPIO10	A0 (2)	GPIO10	19	20	GND	—	GND
GPIO9	A7 (1)	GPIO9	21	22	GPIO25	A7 (3)	GPIO25
GPIO11	A1 (2)	GPIO11	23	24	GPIO8	A6 (1)	GPIO8
GND	—	GND	25	26	GPIO7	A5 (1)	GPIO7
—	—	DNC	27	28	DNC	—	—
GPIO5	A3 (1)	GPIO5	29	30	GND	—	GND
GPIO6	A4 (1)	GPIO6	31	32	GPIO12	A2 (2)	GPIO12
GPIO13	A3 (2)	GPIO13	33	34	GND	—	GND
GPIO19	A1 (3)	GPIO19	35	36	GPIO16	A6 (2)	GPIO16
GPIO26	A0 (4)	GPIO26	37	38	GPIO20	A2 (3)	GPIO20
GND	—	GND	39	40	GPIO21	A3 (3)	GPIO21

Komunikasi Pin

GPIO 2	(SDA)	GPIO 14	Tx
GPIO 3	(SCL)	GPIO 15	Rx

3.2.3. Rangkaian LED (*Light Emitting Dioda*)

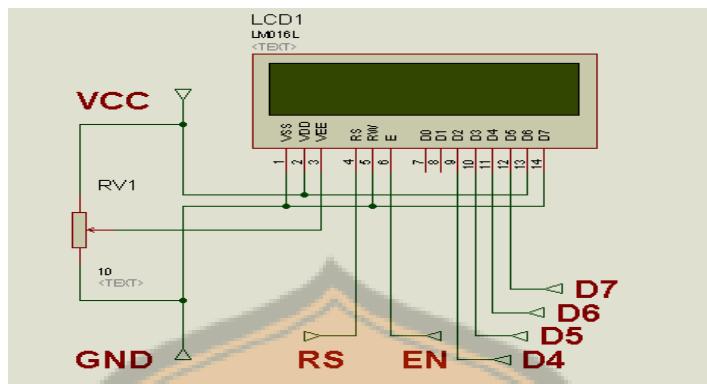
Pada rancangan LED (*Light Emitting Dioda*) terdapat kaki anoda dan katoda, dimana kaki anoda merupakan kaki positif dan kaki katoda merupakan kaki negatif. Rangkaian pada kaki katoda LED digabung menjadi satu untuk menghemat kabel jamperan pada modul pembelajaran *raspberry pi 3*. Nilai hambatan (R) didapat berdasarkan persamaan (2.1). Keterangan LED yang digunakan memiliki tegangan 2 volt (V_{led}) dan arus 20 mA (I). Tegangan sumber yang digunakan adalah 5 volt (V_s).



Gambar 3.2. Rangkaian LED

3.2.4. Rangkaian LCD (*Liquid Crystal Display*)

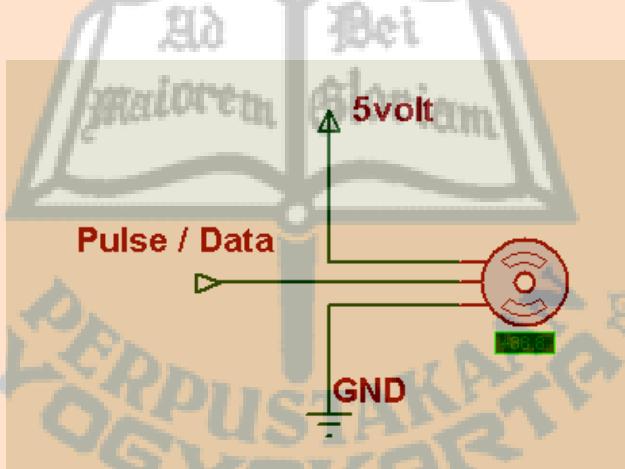
LCD (*Liquid Crystal Display*) memiliki 16 kaki keluaran, tetapi dalam penelitian ini menggunakan 12 kaki LCD. Pada perancangan boks modul pembelajaran *raspberry pi 3* kaki keluaran pada LCD hanya 8 buah, dikarenakan 4 kabel buah lainnya telah disambung didalam boks untuk menghemat jamperan kabel pada boks. Rangkaian LCD dilengkapi dengan *potensio* dengan nilai 10 KΩ yang berfungsi sebagai pengatur kontras pada tampilan LCD. Kaki RS merupakan *Register Selector* yang berfungsi untuk memilih register data seperti yang digunakan untuk menulis data karakter ke memori *display* LCD. Kaki EN merupakan *Enable* digunakan untuk mengaktifkan LCD pada proses penulisan data ke register data LCD. Kaki D4, D5, D6, dan D7 merupakan jalur data pada LCD.



Gambar 3.3. Rangkaian LCD (*Liquid Crystal Display*)

3.2.5. Rangkaian Motor Servo

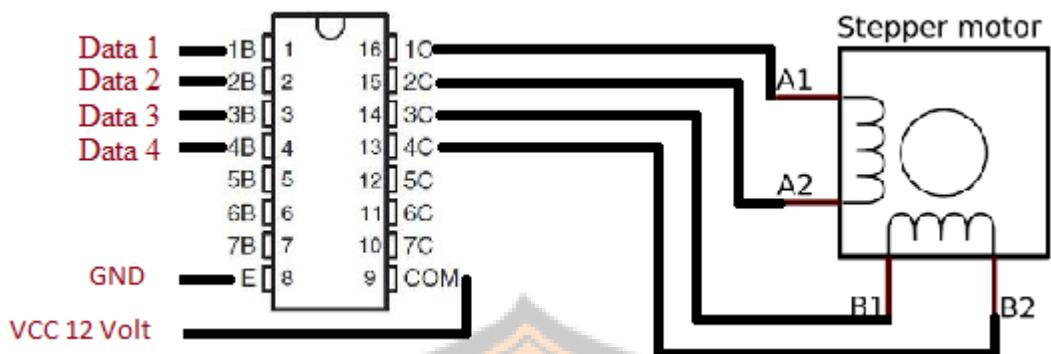
Rangkaian *motor servo* yang menggunakan tegangan 5 volt. *Motor servo* memiliki 3 kabel diantaranya kabel masukan tegangan (*input*) 5 volt, kabel *ground*, dan kabel pulsa data. Rangkaian *motor servo* pada kabel *ground* sudah langsung terangkai dalam boks modul pembelajaran *raspberry pi* 3, sedangkan kabel lainnya terhubung pada *pinout raspberry pi*.



Gambar 3.4. Rangkaian *motor servo*

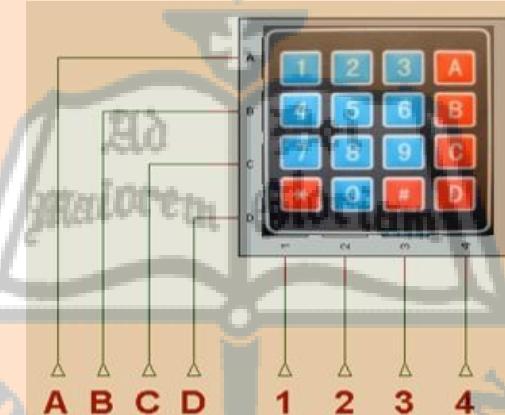
3.2.6. Rangkaian Driver dan Motor stepper

Perancangan modul *driver motor stepper* ULN2003 dirangkai menjadi satu dengan *motor stepper bipolar 35PM48L01*. Berdasarkan struktur rotor dan stator *motor stepper* yang digunakan adalah tipe *Permanent Magnet* (PM). Modul driver ini menggunakan IC ULN 2003 yang berisi rangkaian *transistor darlington* yang berfungsi sebagai penguat arus pada *motor stepper*.

Gambar 3.5. Rangkaian *driver* dan *motor stepper*

3.2.7. Rangkaian Keypad 4x4

Pada rangkaian *keypad* 4x4 dihubungkan langsung ke *pinout* atau keluaran pada boks modul pembelajaran *raspberry pi* 3.

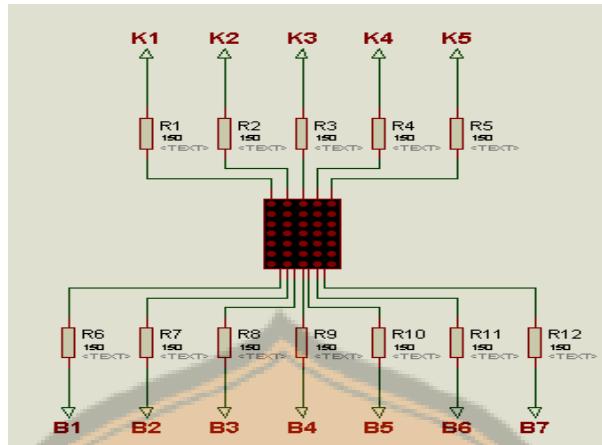
Gambar 3.6. Rangkaian *keypad* 4x4

3.2.8. Rangkaian Dot matrix LED 7x5

Pada rangkaian *dot matrix* LED 7x5 dihubungkan langsung ke *pinout* atau keluaran pada boks modul pembelajaran *raspberry pi* 3. Terdiri dari 7 baris dan 5 kolom dengan menggunakan hambatan sebesar 150Ω . Tegangan sumber adalah 5 volt dan *typical* tegangan pada *dot matrix* yaitu 2 volt dengan arus 20 mA. Nilai hambatan didapat berdasarkan persamaan (2.1).

$$R = \frac{5 \text{ volt} - 2 \text{ volt}}{0,02 \text{ Ampere}}$$

$$R = 150\Omega$$

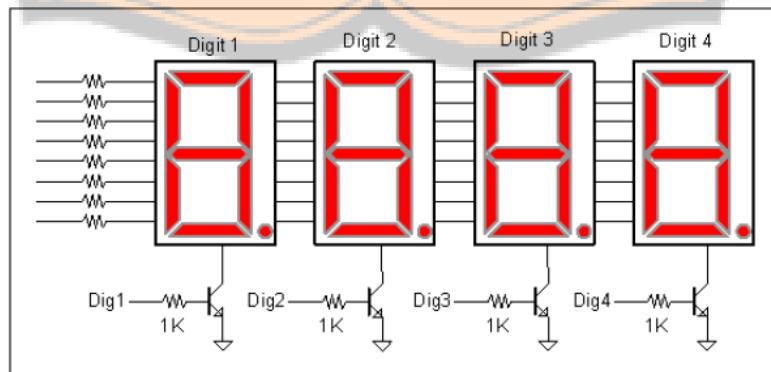
Gambar 3.7. Rangkaian *dot matrix* LED 7x5

3.2.9. Rangkaian 4 Digit Seven segment

Pada rangkaian 4 digit *seven segment* terdapat 12 pin, 7 diantaranya merupakan kaki pada *segment* yang memiliki keterangan A hingga G, 4 diantaranya merupakan *common* (COM) yang terdiri dari COM1 (Digit 1), COM2 (Digit 2), COM3 (Digit 3), dan COM4 (Digit 4) untuk mengaktifkan *seven segment*. Pin terakhir merupakan pengaktifan titik atau *dot* pada *seven segment* yang terdiri dari D1, D2, D3, D4, dan DP. Pada rangkaian 4 digit *seven segment* menggunakan resistor 200Ω dan penggunaan transistor 2N222. Nilai hambatan didapat berdasarkan persamaan (2.1). Keterangan tegangan sumber yang digunakan adalah 5 volt, tegangan pada *seven segment* adalah 3 volt dan arus senilai 10 mA. Hambatan bernilai 1 K Ω sesuai dengan *datasheet*.

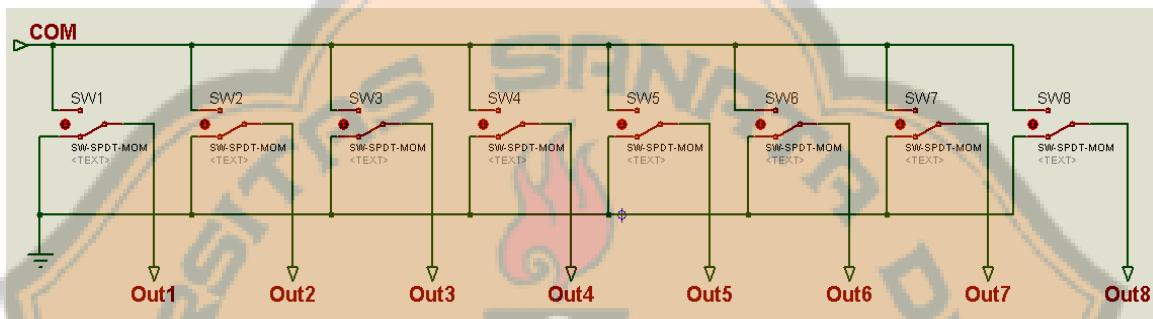
$$R = \frac{5 \text{ volt} - 3 \text{ volt}}{0,01 \text{ Ampere}}$$

$$R = 200 \Omega$$

Gambar 3.8. Rangkaian 4 digit *seven segment*

3.2.10. Rangkaian Saklar Toggle

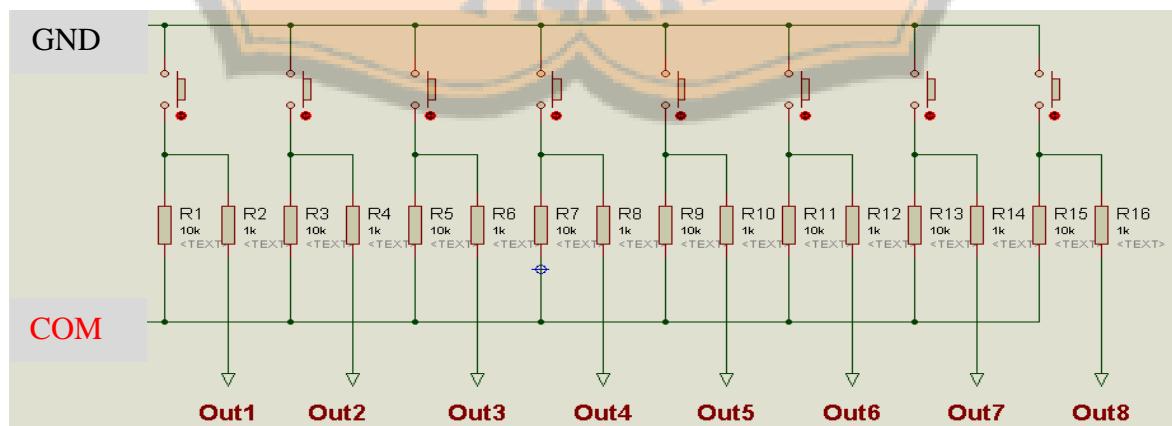
Pada rangkaian saklar *toggle* tegangan (COM) dan *ground* telah dihubungkan langsung didalam boks modul *raspberry pi* 3. Rangkaian COM merupakan masukan tegangan 5 volt. Keluaran saklar *toggle* akan dihubungkan pada pin boks. Pilihan pada saklar *toggle* yaitu antara terhubung dengan COM dan *ground*, yang bertujuan penegasan saat pembacaan nilai logika “high” dan “low”. Kondisi “high” yaitu saat terhubung dengan 5 volt dan kondisi “low” saat terhubung dengan *ground*.



Gambar 3.9. Rangkaian saklar *toggle*

3.2.11. Rangkaian Push button

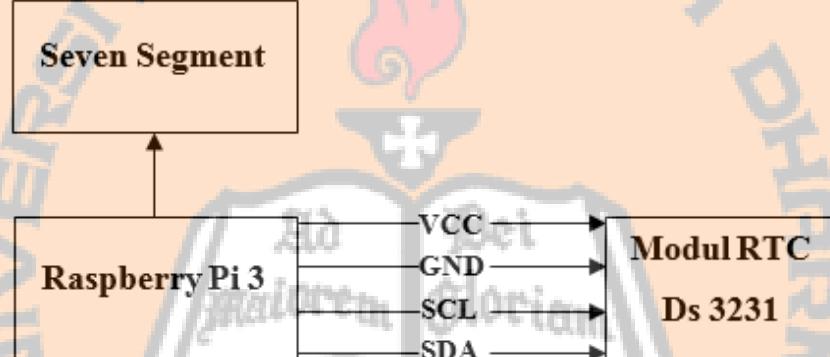
Pada rangkaian *push button* tegangan (COM) dan *ground* telah dihubungkan langsung didalam boks modul *raspberry pi* 3. Rangkaian COM merupakan masukan tegangan 5 volt. Keluaran *push button* akan dihubungkan pada pin boks. Didalam rangkaian terdapat rangkaian resistor *pull up* yang bertujuan mencegah terjadinya masalah pada nilai yang tidak terbaca seperti nilai yang mengambang antara “high” dan “low”. Nilai hambatan yang digunakan mengikuti dari acuan *datasheet* yang digunakan.



Gambar 3.10. Rangkaian *push button*

3.2.12. Rangkaian Komunikasi I2C

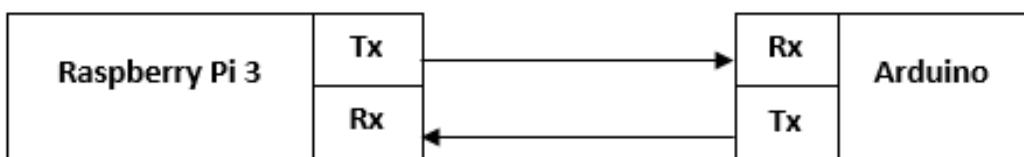
Rangkaian komunikasi *I2C* diaplikasikan pada modul RTC (*Real Time Clock*) Ds 3231. Modul RTC Ds 3231 merupakan modul penghitung waktu yang didalamnya dapat diatur hari, tanggal, bulan, tahun, dan waktu. Modul RTC Ds 3231 menggunakan komunikasi serial yaitu *I2C* dan juga terdapat baterai sebagai sumber daya tersendiri pada modul. Modul RTC Ds 3231 akan tetap menghitung hari dan waktu yang telah diatur walaupun tidak mendapat sumber daya dari luar. Pin yang akan digunakan ada 4 yaitu, pin SCL, SDA, VCC, dan GND. Pin SCL, SDA, VCC, dan GND akan terhubung dengan pin *raspberry pi 3* sesuai dengan kode pada pin modul RTC Ds 3231. Waktu yang telah diatur akan ditampilkan pada *seven segment*.



Gambar 3.11. Rangkaian komunikasi *I2C*

3.2.13. Rangkaian Komunikasi UART

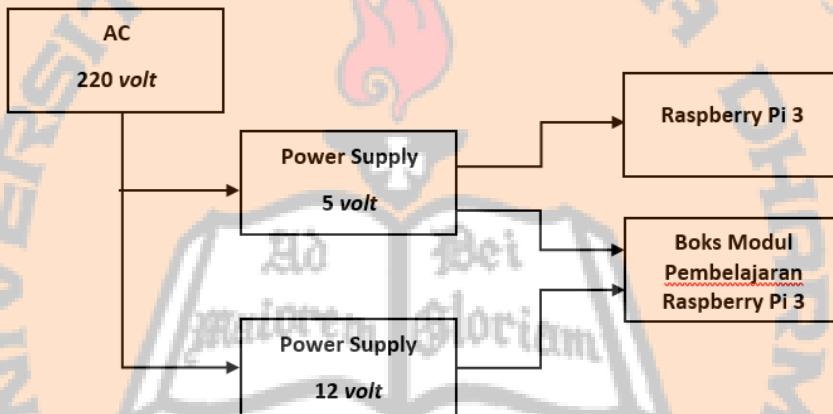
Rangkaian komunikasi UART diaplikasikan dari *raspberry pi 3* ke *arduino*. Komunikasi yang diaplikasikan yaitu komunikasi dua arah (*full duplex*). Rx biasa disebut *received*, yang berguna menangkap data yang dikirim oleh transmitter (Tx). Tx disebut *transmitter* yang berfungsi untuk mengirim data. Data akan dikirim melalui Tx (*Transmitter*) dan di ujung lainnya data akan diterima melalui Rx (*Received*).



Gambar 3.12. Rangkaian komunikasi UART

3.2.14. Perancangan Power supply

Dari komponen yang digunakan maka dibutuhkan sebuah *power supply*. Perancangan *power supply* yang digunakan adalah *power supply switching* yang keluarannya 5 volt dengan arus 3 Ampere dan 12 volt dengan arus 1 Ampere. Berdasarkan keterangan komponen-komponen yang digunakan sehingga diperoleh nilai tegangan dan arus yang digunakan. Perancangan *power supply* 5 volt dan 12 volt ditempatkan didalam boks modul pembelajaran *raspberry pi 3*. *Power supply* 5 volt akan terhubung dengan *raspberry pi 3* model b dan boks modul pembelajaran sedangkan 12 volt hanya terhubung dengan boks modul pembelajaran *raspberry pi 3*. Pada bagian *ground* akan disambungkan menjadi satu bagian.

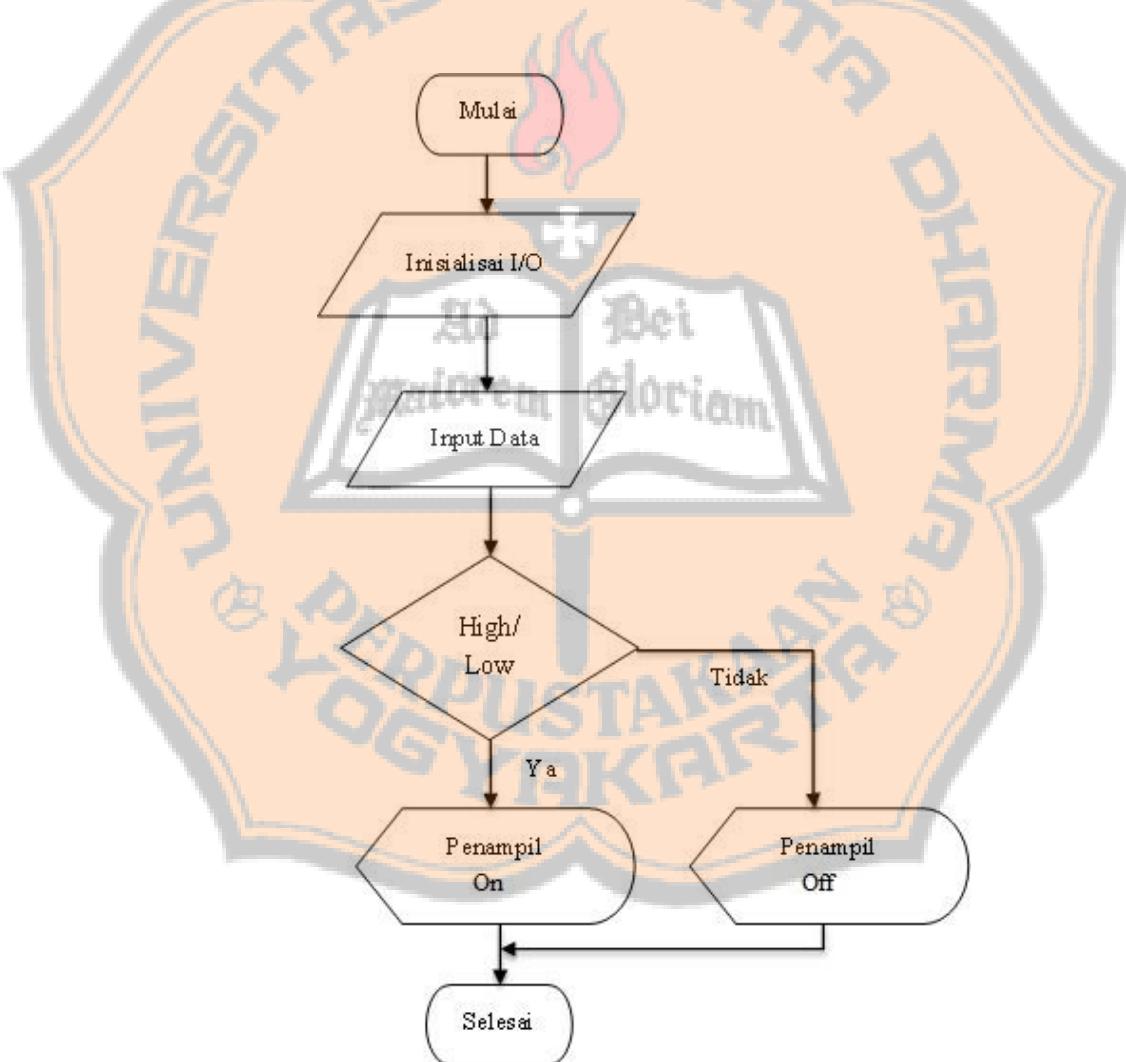


Gambar 3.13. Perancangan *power supply*

3.3. Perancangan Perangkat Lunak

3.3.1. Diagram Alir *Push button* dan LED

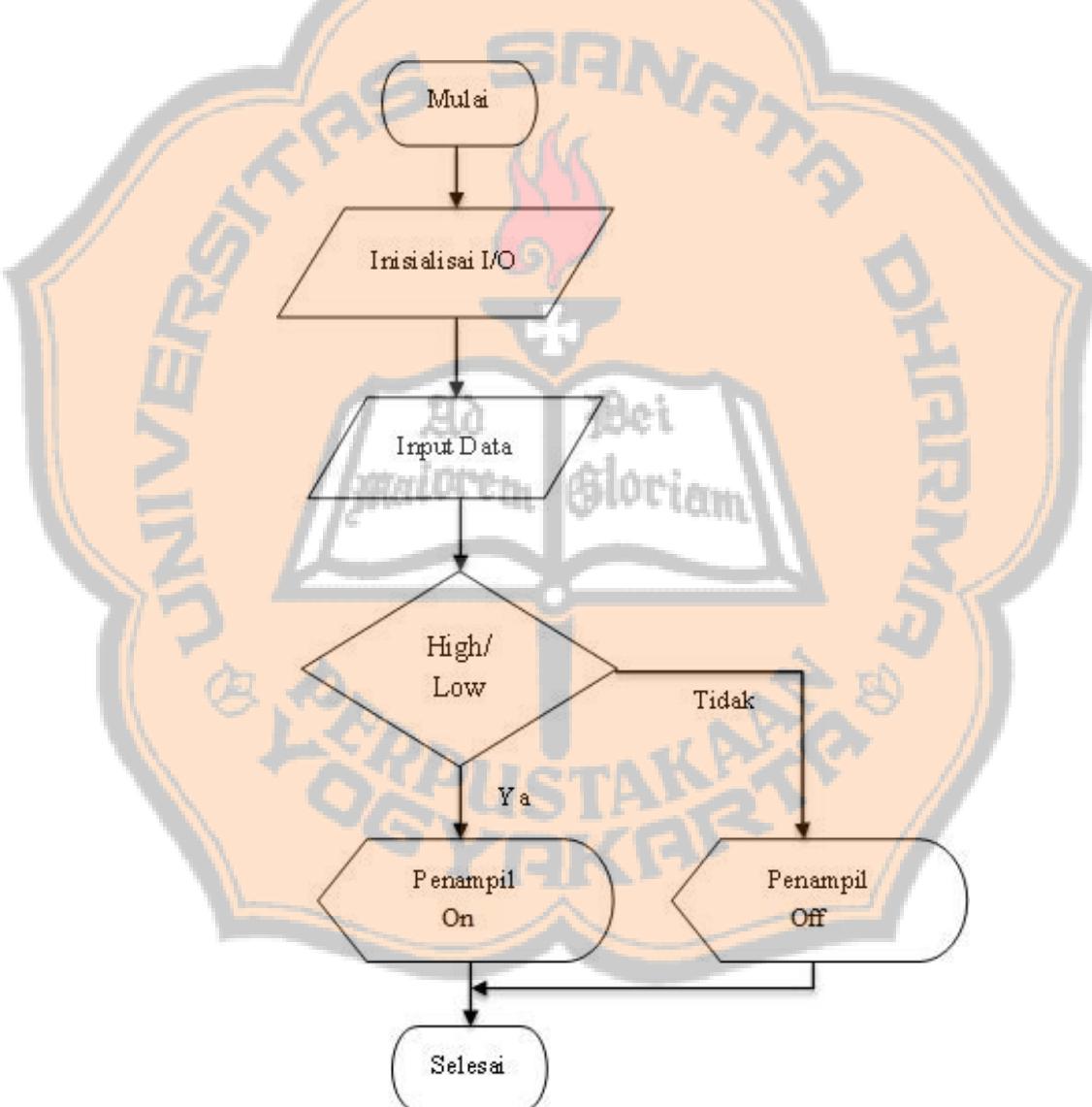
Diagram alir *push button* dan LED ditunjukkan pada gambar 3.14. Proses diagram menunjukkan proses *raspberry pi 3* menjalankan *system* mulai dari mulai, program *python* melakukan inisialisasi terhadap *input* dan *output*. Data berupa “high” dan “low”, jika data diketahui adalah *high* maka LED sebagai penampil akan menyala. Jika sebaliknya data diketahui adalah *low* maka LED sebagai penampil akan padam atau tidak menyala. LED sebagai penampil akan tetap menyala selama *push button* ditekan, jika tidak ditekan maka *push button* akan kembali pada posisi awal yaitu *off* atau tidak terhubung.



Gambar 3.14. Diagram alir *push button* dan LED

3.3.2. Diagram Alir Saklar *Toggle* dan LED

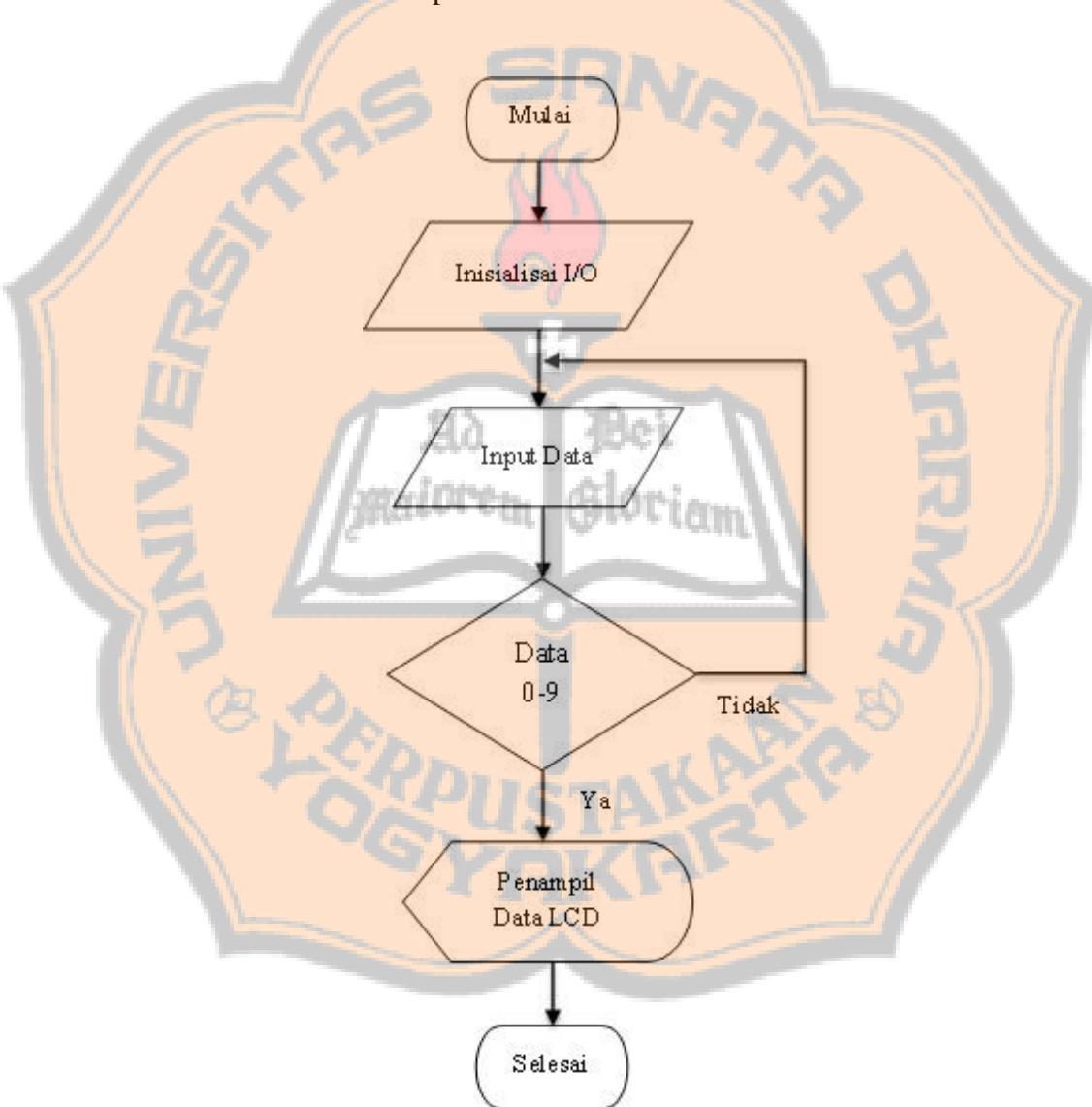
Diagram alir saklar *toggle* dan LED ditunjukkan pada gambar 3.15. Proses diagram menunjukkan proses *raspberry pi 3* menjalankan *system* mulai dari mulai, program *python* melakukan inisialisasi terhadap *input* dan *output*. Data berupa *high* dan *low*, jika data diketahui adalah *high* maka LED sebagai penampil akan menyala. Jika sebaliknya data diketahui adalah *low* maka LED sebagai penampil akan padam atau tidak menyala. Pada saklar *toggle* sekali dalam menghubungkan maka data akan selamanya *high*, dan untuk kembali pada posisi awal yaitu *low* maka harus mengatur kembali tuas pada posisi awal.



Gambar 3.15. Diagram alir saklar *toggle* dan LED

3.3.3. Diagram Alir Keypad dan LCD

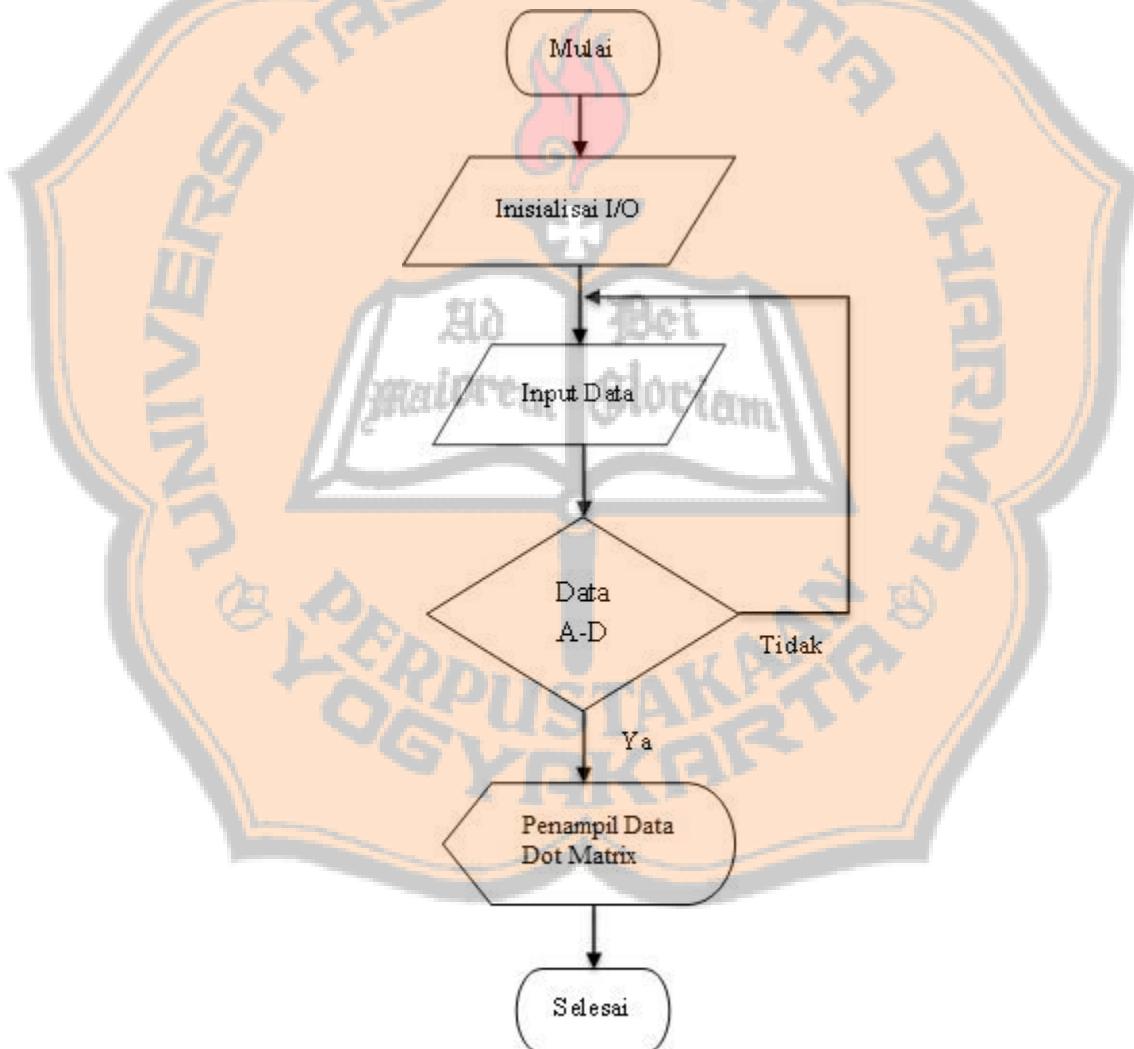
Diagram alir *keypad* dan LCD ditunjukkan pada gambar 3.16. Proses diagram menunjukkan proses *raspberry pi 3* menjalankan *system* mulai dari mulai, inisialisasi terhadap *input* dan *output*, dan menampilkan data. Pada proses inisialisasi terdapat pada *raspberry pi* yang telah diprogram menggunakan *python* dengan data berupa angka mulai dari 0 hingga 9. Proses *input data* menggunakan *keypad* dan penampil data menggunakan LCD 16x2. Jika data tidak sesuai dengan angka 0 hingga 9 maka akan kembali pada proses *input data*. Jika data sesuai maka data akan tertampil.



Gambar 3.16. Diagram alir *keypad* dan LCD

3.3.4. Diagram Alir *Keypad* dan *Dot matrix LED*

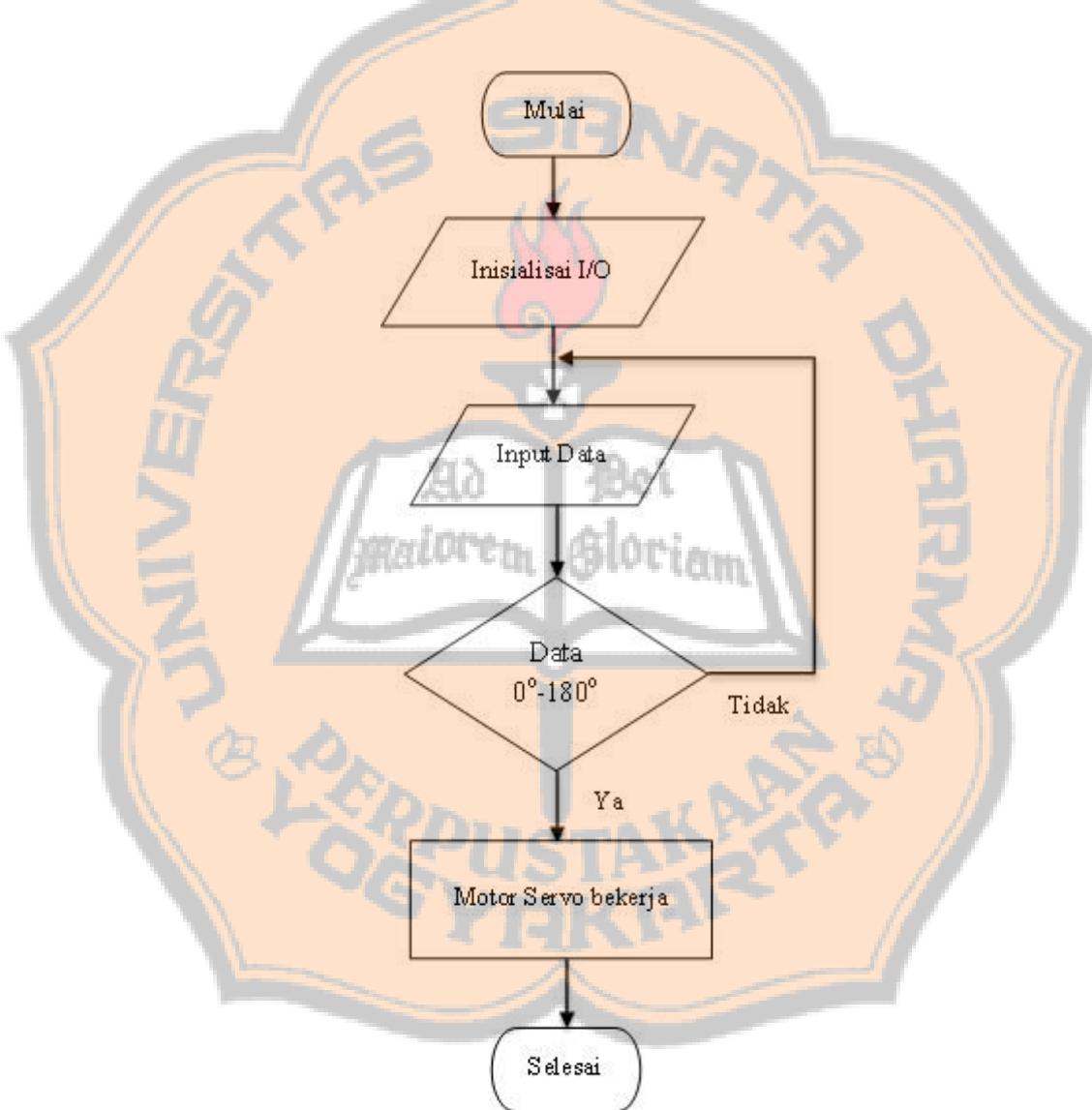
Diagram alir *keypad* dan *dot matrix LED* ditunjukkan pada gambar 3.17. Proses diagram menunjukkan proses *raspberry pi* 3 menjalankan mulai, inisialisasi I/O, *input data*, dan menampilkan data. Pada proses inisialisasi terdapat pada *raspberry pi* yang telah deprogram oleh *python* dengan data berupa huruf yaitu A, B, C, dan D. Proses *input data* menggunakan *keypad* dan penampil data menggunakan *dot matrix LED* 5x7. Jika *input data* sesuai maka akan tertampil pada *dot matrix LED*, dan sebaliknya jika data tidak sesuai maka data akan kembali pada proses *input data* untuk memastikan kembali data yang dimasukan sesuai dengan program *python*.



Gambar 3.17. Diagram alir *keypad* dan *dot matrix LED*

3.3.5. Diagram Alir Motor Servo

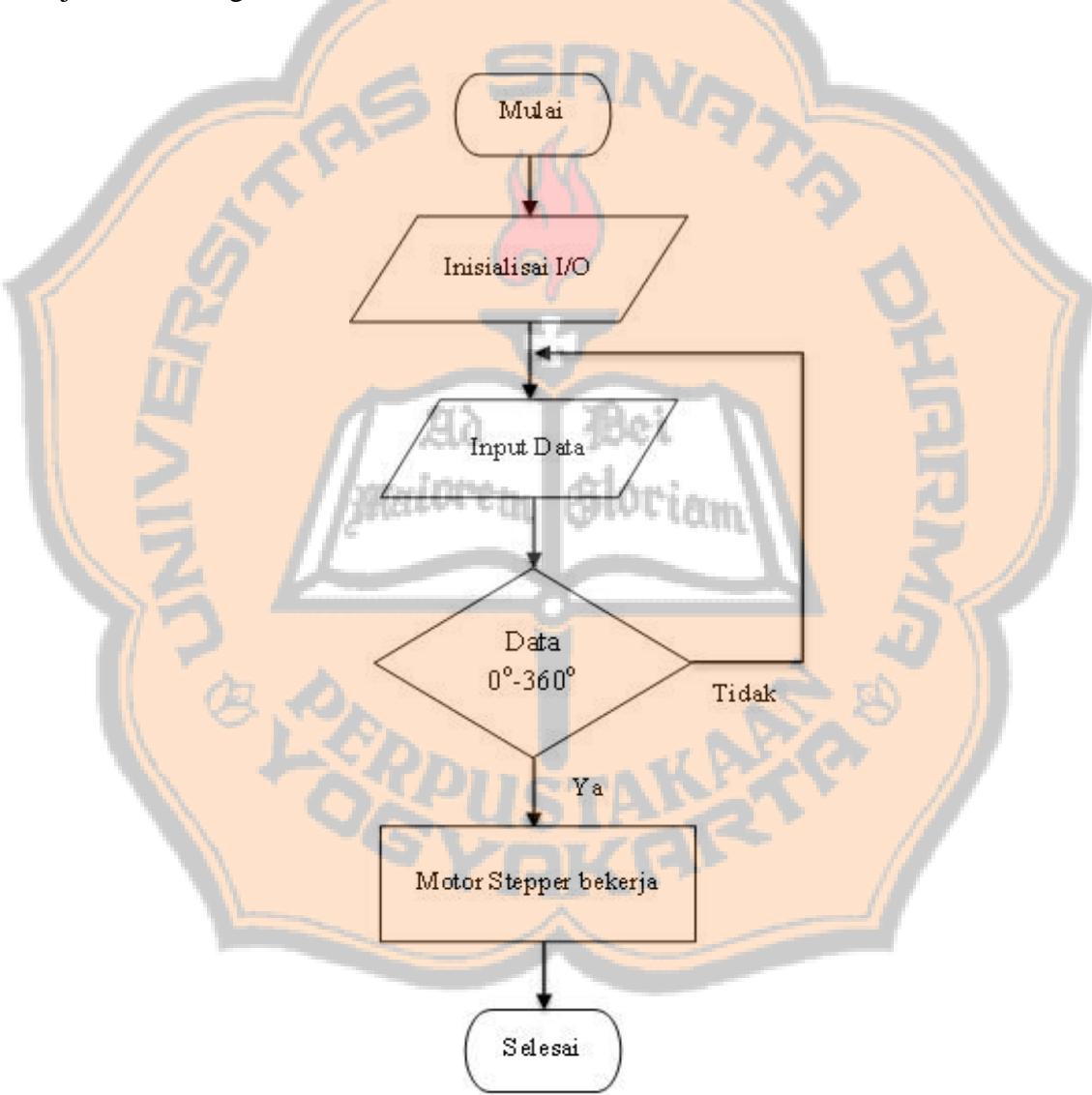
Diagram alir *motor servo* ditunjukkan pada gambar 3.18. Proses diagram menunjukkan proses *raspberry pi 3* menjalankan sistem mulai dari mulai, program melakukan inisialisasi terhadap *input* dan *output*. Data berupa nilai 0 hingga 180 derajat. Pada saat menjalankan program *python* pengguna (*user*) akan diminta untuk memasukan data derajat yang ingin ditampilkan. Kemudian data akan ditampilkan melalui gerakan *motor servo* yang akan menunjukkan derajat sesuai dengan masukan oleh *user*.



Gambar 3.18. Diagram alir *motor servo*

3.3.6. Diagram Alir Motor stepper

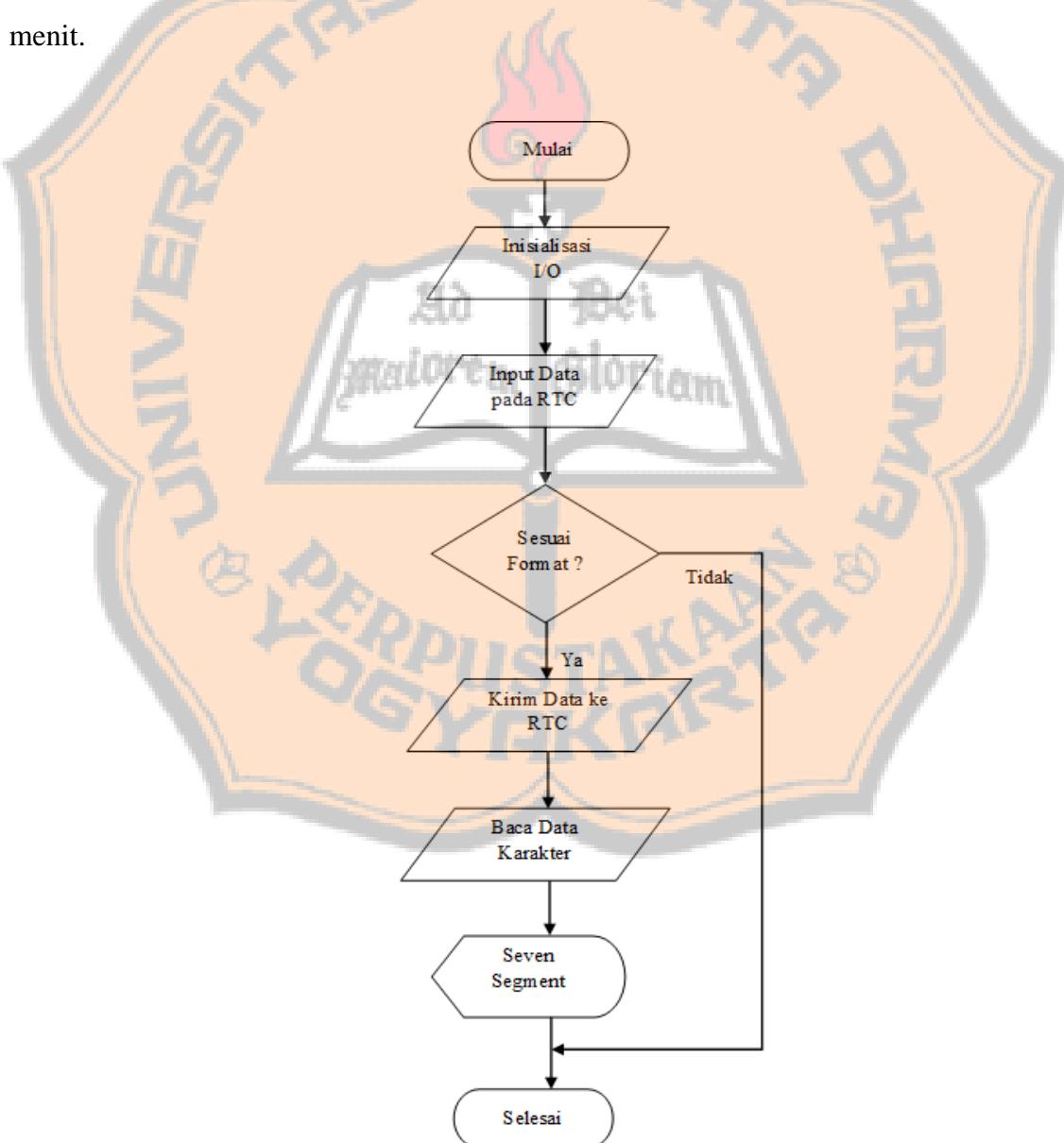
Diagram alir saklar *toggle* dan LED ditunjukkan pada gambar 3.19. Proses diagram menunjukkan proses *raspberry pi 3* menjalankan *system* mulai dari mulai, program *python* melakukan inisialisasi terhadap *input* dan *output*. Data berupa nilai 0 hingga 360 derajat. *Motor stepper* bergerak sebesar $7,5^\circ$ per *step*. Pada saat menjalankan program *python* pengguna (*user*) akan diminta untuk memasukan data derajat yang ingin ditampilkan. Kemudian data akan ditampilkan melalui gerakan *motor stepper* yang akan menunjukkan derajat sesuai dengan masukan oleh *user*.



Gambar 3.19. Diagram alir *motor stepper*

3.3.7. Diagram Alir Komunikasi I2C Modul RTC Ds 3231 dan *Seven segment*

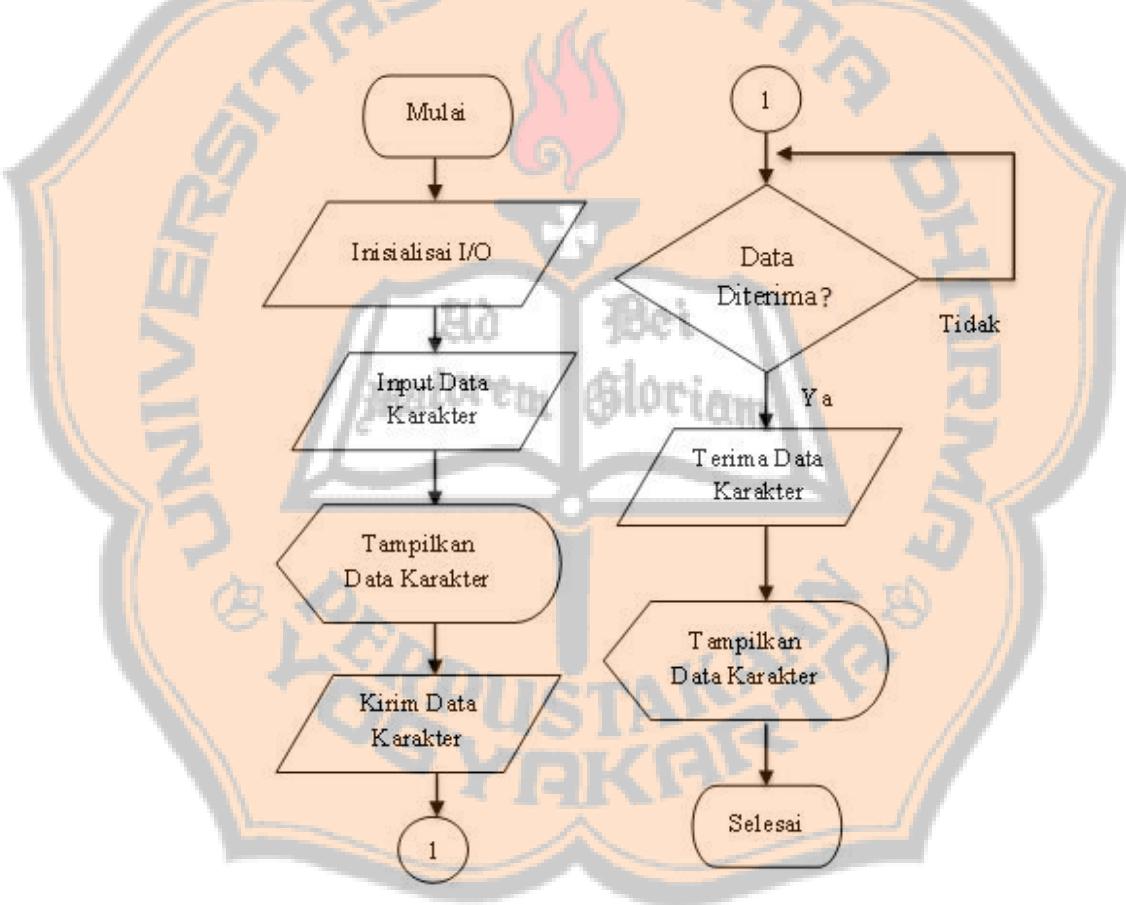
Diagram alir komunikasi *I2C* modul RTC (*Real Time Clock*) Ds 3231 dan *seven segment* pada gambar 3.20. Proses diagram menunjukkan proses komunikasi serial *I2C* antara *raspberry pi 3* dengan modul RTC Ds 3231. Masukan data berupa waktu yang menunjukkan jam, menit, dan detik. *Input* data dimasukan oleh *user*. Format pada masukan data harus sesuai dengan ketentuan waktu seperti jam tidak lebih dari nilai 24 dan menit tidak lebih dari nilai 60. Jika data tidak sesuai maka program *python* akan selesai atau keluar, sebaliknya jika bener maka *raspberry pi* akan membaca data pada modul RTC kemudian akan ditampilkan pada *seven segment*. Data yang tertampil yaitu menunjukkan waktu jam dan menit.



Gambar 3.20. Diagram alir komunikasi modul RTC dan *seven segment*

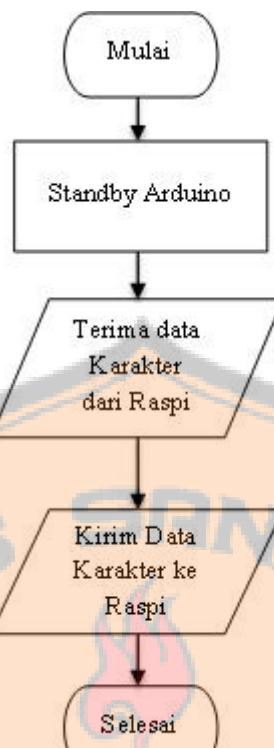
3.3.8. Diagram Alir Komunikasi UART

Diagram alir komunikasi UART *raspberry pi* 3 ke *arduino* dapat dilihat pada gambar 3.21 dan 3.22. Proses diagram menunjukkan proses komunikasi serial UART antara *raspberry pi* 3 dengan modul RTC Ds 3231. Komunikasi yang dilakukan adalah komunikasi dua arah (*full duplex*). Masukan data berupa karakter yang akan dikirim ke *arduino*. Aduino akan menerima data yang telah dikirim dari *raspberry pi* 3, kemudian akan mengirim kembali data yang telah diterima ke *raspberry pi* 3. Data yang dikirim dari *arduino* akan diterima oleh *raspberry pi* 3, jika data yang diterima sama dengan data yang telah dikirim maka komunikasi berhasil sebaliknya jika data tidak sesuai maka terdapat kesalahan pada komunikasi. Penampilan data yang dikirim dan diterima dapat dilihat pada *Phython*.



Gambar 3.21. Diagram alir komunikasi UART *raspberry pi*

Pada gambar 3.21 merupakan proses alir diagram *raspberry pi* 3 sebagai pengirim data karakter, data karakter dikirimkan ke *arduino*. Data karakter yang diterima pada *arduino* akan dikirimkan kembali ke *raspberry pi* 3 dapat dilihat pada alir diagram gambar 3.22.



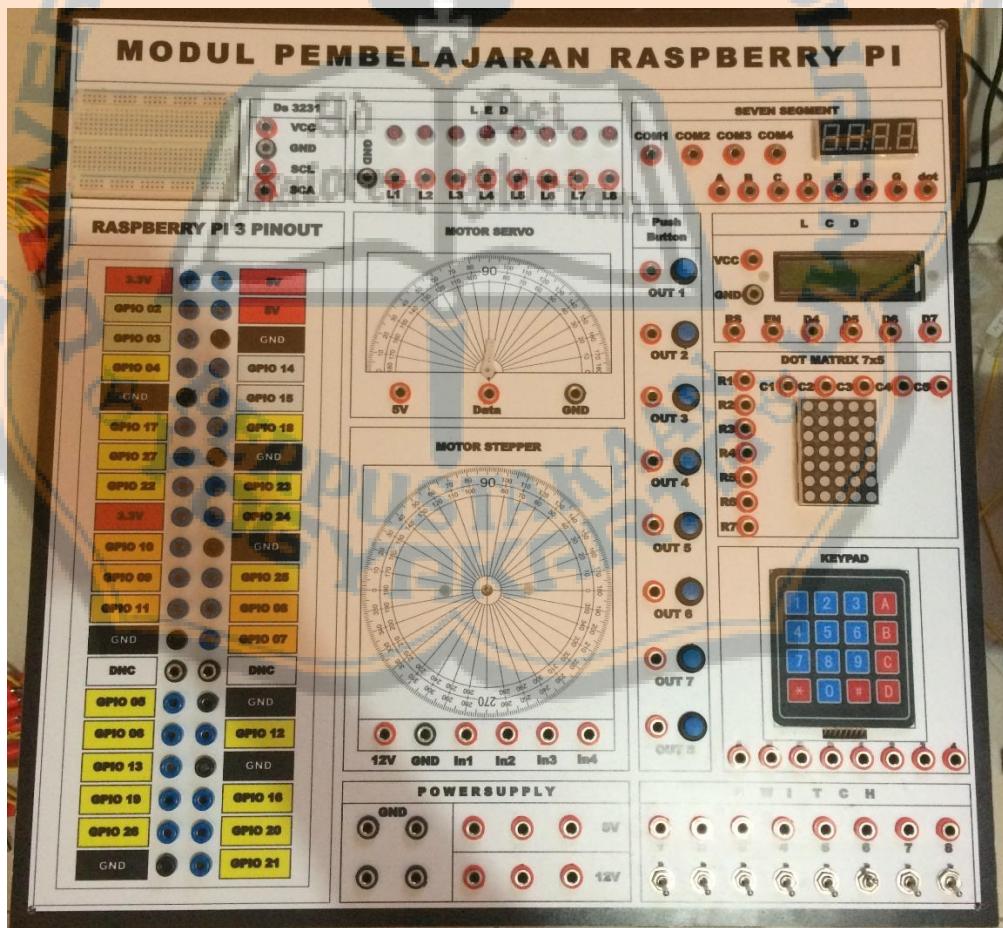
Gambar 3.22. Diagram alir komunikasi UART *arduino*

BAB IV

HASIL DAN PEMBAHASAN

Bab ini berisi pembahasan alat yang telah dikerjakan, yang meliputi hasil pengamatan dari percobaan. Hasil pengamatan yang akan dibahas terdiri dari perangkat-perangkat yang telah terpasang pada Modul Pembelajaran *Raspberry pi*. Hasil pengujian berupa data-data yang diperoleh untuk memperlihatkan bahwa *hardware* ataupun *software* yang dirancang telah berjalan dengan baik atau tidak. Berdasarkan data-data tersebut maka dapat dilakukan analisis terhadap fungsi kerja dari perangkat-perangkat pada Modul Pembelajaran *Raspberry pi* yang kemudian dapat digunakan untuk menarik kesimpulan akhir.

4.1. Bentuk Fisik Modul Pembelajaran *Raspberry pi*



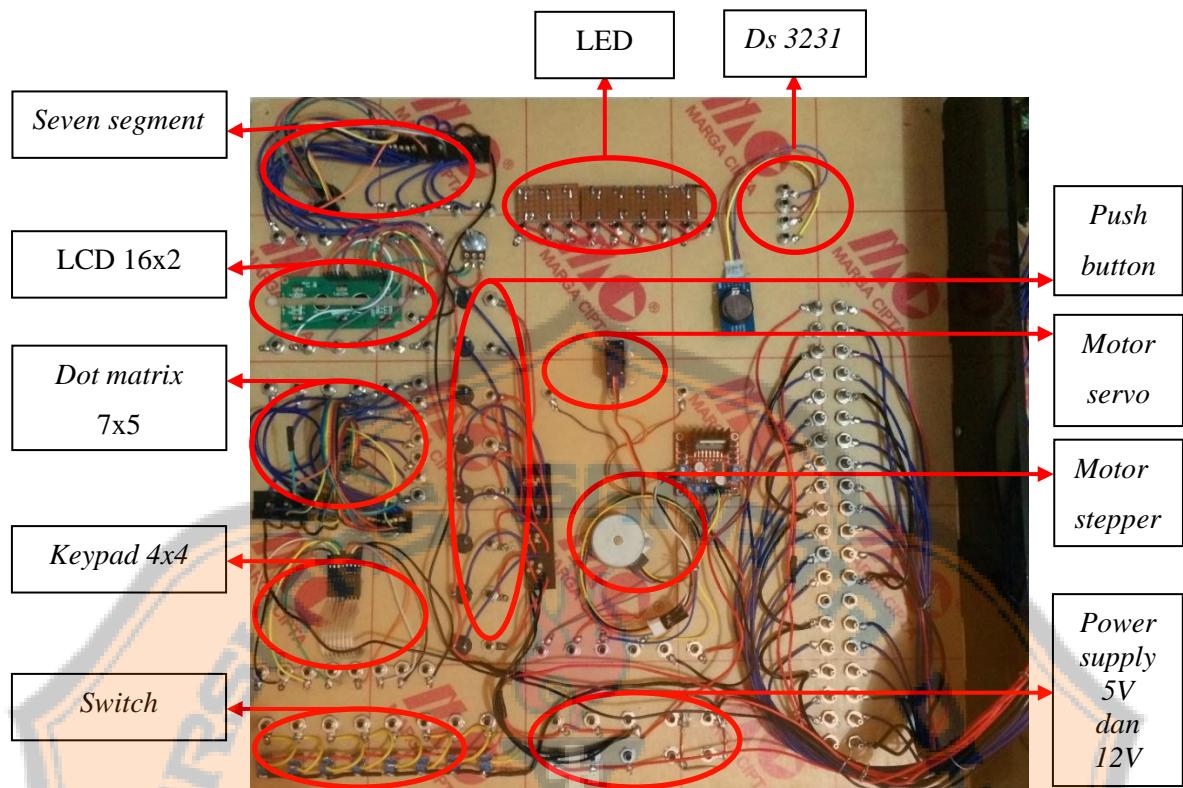
Gambar 4.1. Tampak atas boks modul *raspberry pi*



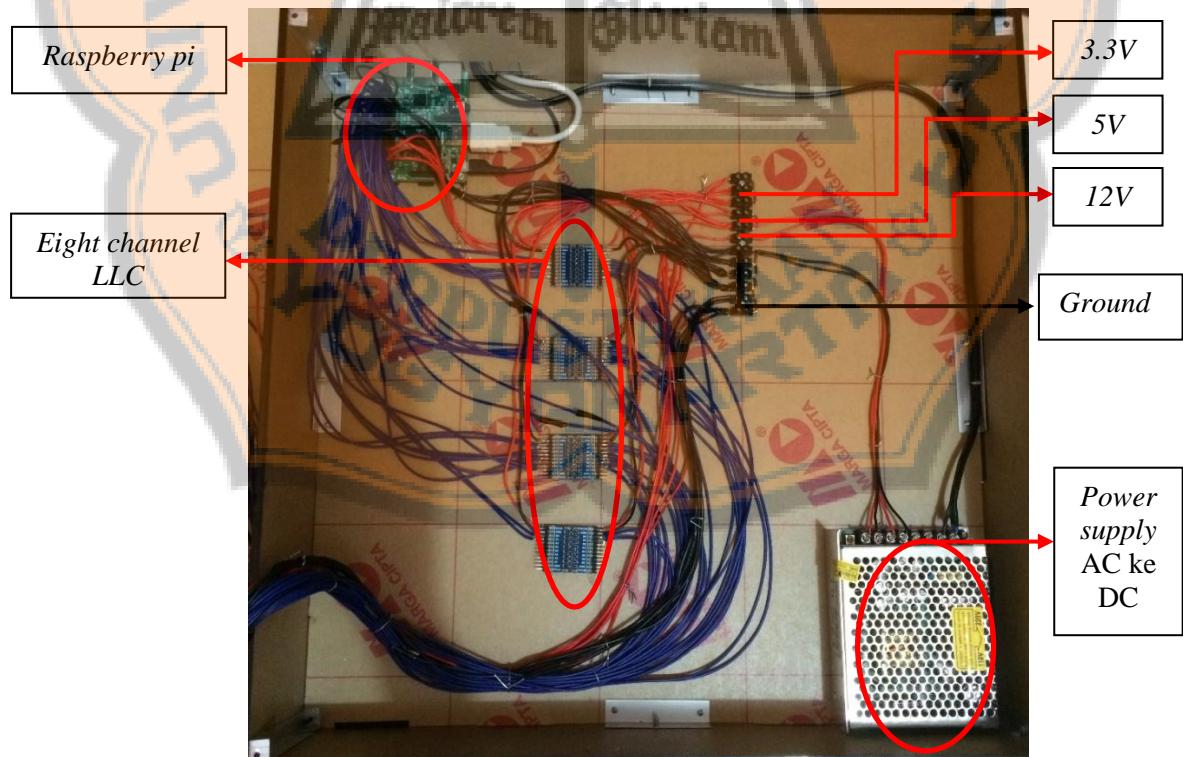
Gambar 4.2. Tampak belakang boks modul *raspberry pi*

Pada bagian ini adalah tampak fisik luar yang meliputi bagian atas boks gambar 4.1. Perangkat-perangkat pada Modul Pembelajaran *Raspberry pi* telah dilegkapi dengan lubang-lubang yang akan membantu dalam pengoperasian perangkat yang akan diujikan. Boks pada modul memiliki panjang 47,5 cm dan lebar 47,5 cm dengan material yang digunakan berupa kaca mika atau akrilik dengan ketebalan 5 mm, diharapkan dengan ketebalnya kaca mika atau akrilik mampu menahan tekanan saat menyambungkan perangkat yang akan diuji. Perangkat-perangkat pada Modul Pembelajaran *Raspberry pi* terdiri dari *raspberry pi 3 pinout*, *project board*, *ds 3231*, *LED*, *motor servo*, *motor stepper*, *seven segment*, *push button*, *LCD 16x2*, *dot matrix 7x5 LED*, *keypad 4x4*, *saklar toggle*, *power supply* dan *project board*. Bagian tampak depan pada gambar 4.2 merupakan bagian keluaran dari kabel *power supply*, kabel *HDMI ke VGA*, dan *port-port raspberry pi* (USB dan LAN).

Bagian dalam boks bagian 1 dapat dilihat pada gambar 4.3 dan bagian 2 pada gambar 4.4. Dalam boks dirangkai menggunakan *eight channel logic level control* (LLC) untuk mengubah tegangan logika pada *raspberry pi* yang awalnya 3.3 volt menjadi tegangan 5 volt yang kemudian disambungkan pada boks modul. Ada beberapa pin yang disambungkan langsung tanpa melalui *eight channel LLC*, yaitu pin komunikasi *I2c* (SDL dan SCA) dan *UART* (Tx dan Rx) karena jika melalui *eight channel LLC* pin komunikasi tidak dapat bekerja dengan baik. *Eight channel LLC* membutuhkan tegangan 3.3 volt dan juga tegangan 5 volt untuk mengubah tegangan keluar menjadi 5 volt, tegangan 3.3 volt yang masuk ke setiap *eight channel LLC* diperoleh dari 3.3 volt tegangan dari *raspberry pi* sedangkan tegangan 5 volt didapat dari tegangan *power supply* 5 volt. *Power supply* 5 volt terhubung dengan boks dan juga pada *eight channel LLC* sedangkan 12 volt hanya pada boks.



Gambar 4.3. Tampak rangkaian dalam boks modul *raspberry pi* bagian 1



Gambar 4.4. Tampak rangkaian dalam boks modul *raspberry pi* bagian 2

Tabel 4.1. Rangkaian *eight channel LLC* dengan *raspberry pi*

Raspberri Pi	3.3V/GND	Eight Channel LLC		5V/GND	Boks Modul <i>Raspberry pi</i>
--	3.3V	3.3V	5V	5V	--
--	GND	GND	GND	GND	--
--	--	A0	B0	--	--
--	--	A1	B1	--	--
GPIO18	--	A2	B2	--	GPIO18
GPIO23	--	A3	B3	--	GPIO23
GPIO24	--	A4	B4	--	GPIO24
GPIO25	--	A5	B5	--	GPIO25
GPIO08	--	A6	B6	--	GPIO08
GPIO07	--	A7	B7	--	GPIO07
--	3.3V	3.3V	5V	5V	--
--	GND	GND	GND	GND	--
GPIO12	--	A0	B0	--	GPIO12
GPIO16	--	A1	B1	--	GPIO16
GPIO20	--	A2	B2	--	GPIO20
GPIO21	--	A3	B3	--	GPIO21
GPIO04	--	A4	B4	--	GPIO04
GPIO17	--	A5	B5	--	GPIO17
GPIO27	--	A6	B6	--	GPIO27
GPIO22	--	A7	B7	--	GPIO22
--	3.3V	3.3V	5V	5V	--
--	GND	GND	GND	GND	--
GPIO10	--	A0	B0	--	GPIO10
GPIO09	--	A1	B1	--	GPIO09
GPIO11	--	A2	B2	--	GPIO11
GPIO05	--	A3	B3	--	GPIO05
GPIO06	--	A4	B4	--	GPIO06
GPIO13	--	A5	B5	--	GPIO13
GPIO19	--	A6	B6	--	GPIO19
GPIO26	--	A7	B7	--	GPIO26

4.2. Cara Penggunaan Modul Pembelajaran *Raspberry pi*

Modul pembelajaran *raspberry pi* membutuhkan perangkat untuk saat pengoperasian maka, terlebih dahulu pastikan perangkat modul pembelajaran *raspberry pi* sudah terpasang dengan baik, seperti :

Tabel 4.2. Perangkat modul pembelajaran *raspberry pi*

No	Perangkat
1	<i>Keyboard</i>
2	<i>Mouse</i>
3	<i>Power supply</i>
4	Kabel HDMI (<i>Definition Multimedia Interface</i>)

Raspberry pi pada modul pembelajaran menggunakan OS (*Operating System*) *raspbian GNU / LINUX* versi 8 (*jessie*). Pada *raspberry pi* terdapat *library-library* yang digunakan untuk membantu menjalankan modul pembelajaran *raspberry pi* yang dapat dilihat pada tabel 4.3. Untuk masuk pada proses pemrograman terdapat 2 cara, pertama melalui pilihan Menu >> Programming >> *Python 2* (IDLE) dan kedua dengan klik icon tx_terminal kemudian ketik “*sudo idle*”. *Python 2 Interactive DeveLopment Environment* (IDLE) merupakan salah satu bahasa pemrograman yang digunakan pada modul pembelajaran *raspberry pi*, untuk versi yang digunakan yaitu *python 2.7.9*.

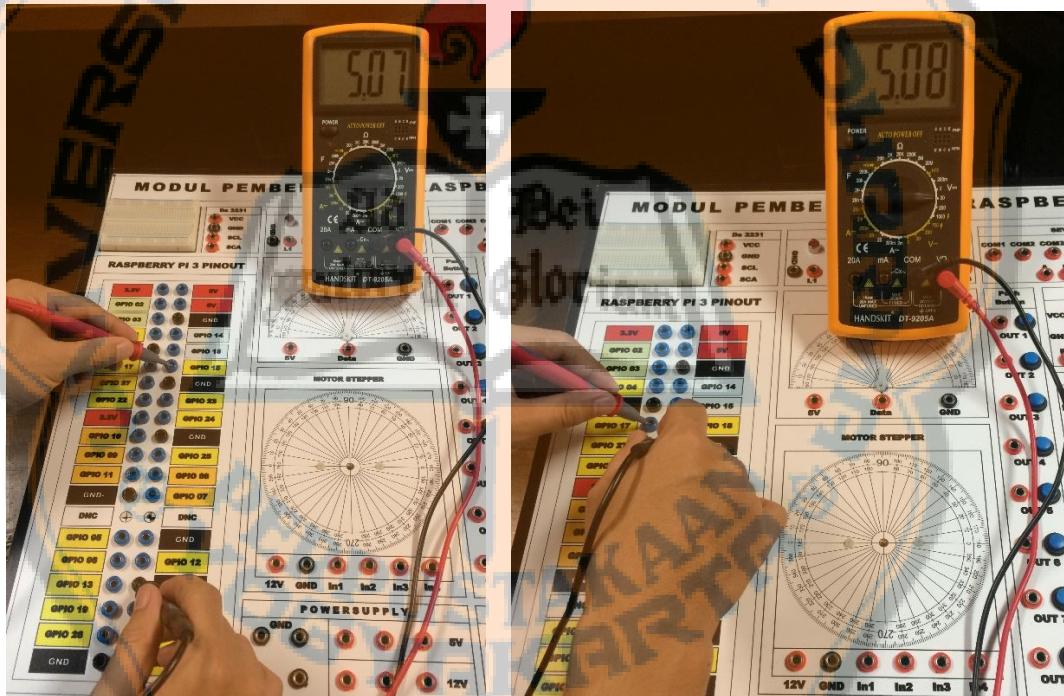
Tabel 4.3. *Library* yang digunakan pada modul pembelajaran *raspberry pi*

No	Library
1	Rpi.GPIO
2	<i>Time</i>
3	Adafruit_charLCD
4	Sys
5	SDL_DS3231
6	Serial

4.3. Pengujian Perangkat Modul Pembelajaran *Raspberry pi*

4.3.1. *Raspberry pi pinout*

Raspberry pi pinout pada modul pembelajaran *raspberry pi* berasal dari keluaran *eight channel LLC* yang keluarannya menjadi tegangan logika 5 volt. *Eight channel LLC* merubah logika tegangan awal *raspberry pi* yaitu 3,3 volt. Pada modul pembelajaran *raspberry pi* menggunakan 3 buah *eight channel LLC* untuk mengubah gpio pin *raspberry pi* menjadi 5 volt. Untuk mengetahui keluaran pada *pinout raspberry pi* diukur menggunakan multimeter tetapi sebelumnya pin *raspberry pi* diprogram menggunakan *python* untuk mengatur keluaran pin semuanya “high”. Gambar 4.5 merupakan pengujian keluaran *raspberry pi* *pinout*.

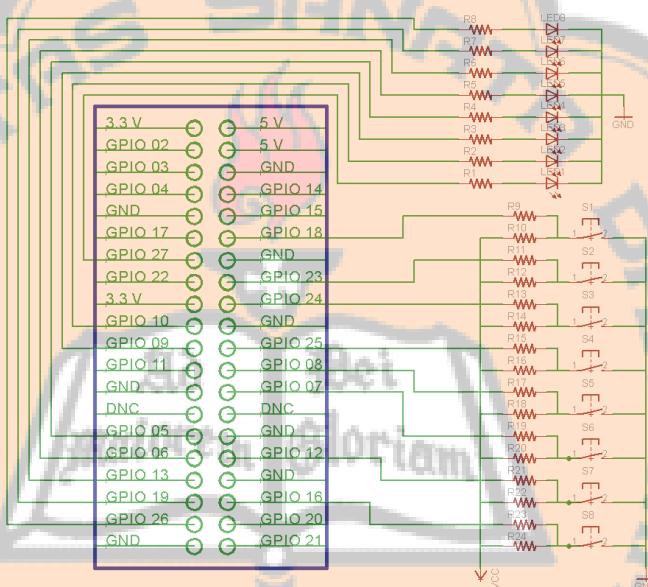


Gambar 4.5. Pengujian *raspberry pi pinout*

Dari gambar 4.5 pengujian *raspberry pi pinout* dapat disimpulkan bahwa *eight channel LLC* berfungsi dengan baik, diketahui dari pengukuran menggunakan multimeter dengan tegangan yang tertampil sebesar 5 volt.

4.3.2. Push button dan LED

Rangkaian *push button* dengan menggunakan *pull up resistors* yang dioperasikan sebagai *input* untuk menyalakan LED. *Output* yang berupa LED akan menyala selama *push button* ditekan. Rangkaian *push button* dan LED yang dihubungkan seperti pada gambar 4.6, untuk menghubungkan rangkaian *push button* dan LED dengan *raspberry pi pinout* seperti tabel 4.4, kemudian pengujian pada program *python* untuk mengkondisikan *input* dan *output* yang akan digunakan sesuai dengan *setmode* yang telah ditetapkan yaitu BOARD seperti pada gambar gambar 4.7.



Gambar 4.6. Rangkaian *push button* dan LED

Tabel 4.4. Rangkaian pengujian *push button* dan LED

<i>Push button</i>	<i>GPIO (BOARD)</i>		<i>LED</i>
Out 1	12 IN	13 OUT	L1
Out 2	16 IN	19 OUT	L2
Out 3	18 IN	21 OUT	L3
Out 4	22 IN	29 OUT	L4
Out 5	24 IN	31 OUT	L5
Out 6	26 IN	33 OUT	L6
Out 7	32 IN	35 OUT	L7
Out 8	36 IN	37 OUT	L8

```

import RPi.GPIO as gpio
import time
gpio.setwarnings(False)
gpio.setmode (gpio.BORD)
#Keluaran
gpio.setup(13,gpio.OUT)
gpio.setup(19,gpio.OUT)
gpio.setup(21,gpio.OUT)
gpio.setup(29,gpio.OUT)
gpio.setup(31,gpio.OUT)
gpio.setup(33,gpio.OUT)
gpio.setup(35,gpio.OUT)
gpio.setup(37,gpio.OUT)
#Masukan
gpio.setup(12,gpio.IN)
gpio.setup(16,gpio.IN)
gpio.setup(18,gpio.IN)
gpio.setup(22,gpio.IN)
gpio.setup(24,gpio.IN)
gpio.setup(26,gpio.IN)
gpio.setup(32,gpio.IN)
gpio.setup(36,gpio.IN)

```

Gambar 4.7. Pengkondisian *input* dan *output push button* dan LED

Pada gambar 4.7 merupakan *listing* program pengkondisian *input* dan *output push button* dan LED. Pengkondisian gpio sebagai *output* seperti “`gpio.setup(13,gpio.OUT)`”, perintah tersebut merupakan pengkondisian gpio pin 13 sebagai *output* sedangkan untuk pengkondisian *input* seperti “`gpio.setup(12,gpio.IN)`”, perintah pengkondisian gpio pin 12 sebagai *input*.

```

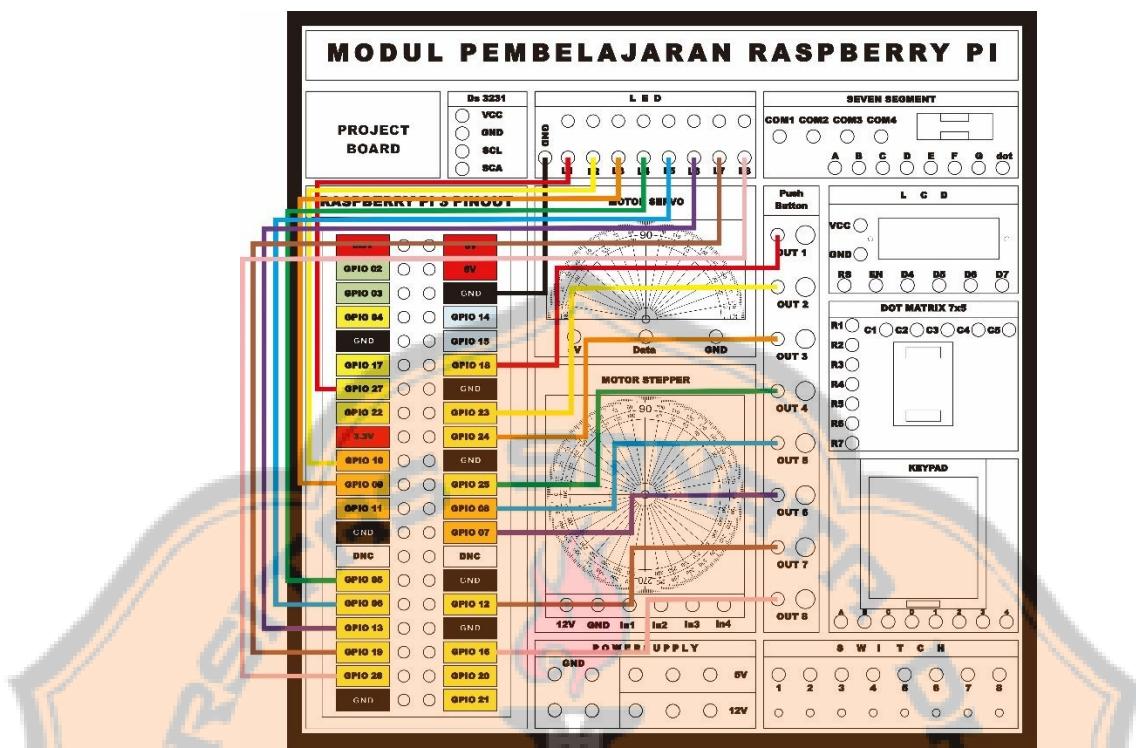
if masukan7==False:
    print "LED7_ON"
    gpio.output (35,gpio.HIGH)
if masukan8==False:
    print "LED8_ON"
    gpio.output (37,gpio.HIGH)

if masukan1==True:
    gpio.output (13,gpio.LOW)
    print "LED1_OFF"

```

Gambar 4.8. Pengkondisian penyalaan *push button* dan LED

Pada gambar 4.8 merupakan *listing* program pengkondisian penyalaan *push button* dan LED. Sebelumnya LED yang telah dikondisikan sebagai *output* kemudian dikondisikan lagi untuk penyalaannya dengan menggunakan fungsi jika (*if*) seperti “`if masukan7==False:`”, “`print “LED7_ON”`”, “`gpio.output(35,gpio.HIGH)`”, perintah tersebut merupakan kondisi penyalaan (*on*) atau diberi logika *high* pada gpio pin 35 dengan kondisi jika masukan==False dan sebaliknya jika ingin mengkondisikan LED mati (*off*) seperti perintah “`if masukan1==True:`”, “`gpio.output(13,gpio.LOW)`”, perintah tersebut merupakan kondisi mematikan lampu atau diberi logika *low* pada gpio pin 13 dengan kondisi jika masukan==True.



Gambar 4.9. Hasil rangkaian pengujian *push button* dan LED

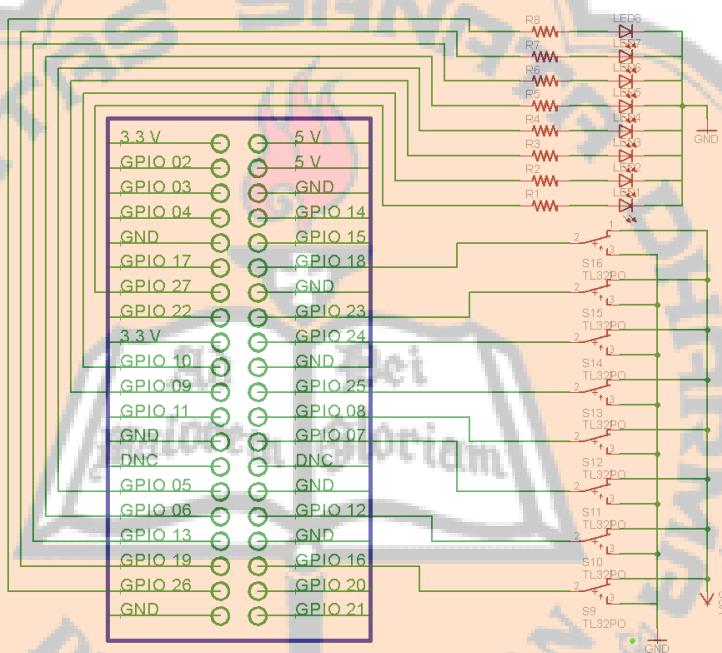
Tabel 4.5. Pengujian *push button* dan LED

No	<i>Push button</i>	LED	No	<i>Push button</i>	LED
1	<i>Out 1 OFF</i>	L1 OFF	9	<i>Out 1 ON</i>	L1 ON
2	<i>Out 2 OFF</i>	L2 OFF	10	<i>Out 1 ON</i>	L2 ON
3	<i>Out 3 OFF</i>	L3 OFF	11	<i>Out 1 ON</i>	L3 ON
4	<i>Out 4 OFF</i>	L4 OFF	12	<i>Out 1 ON</i>	L4 ON
5	<i>Out 5 OFF</i>	L5 OFF	13	<i>Out 1 ON</i>	L5 ON
6	<i>Out 6 OFF</i>	L6 OFF	14	<i>Out 1 ON</i>	L6 ON
7	<i>Out 7 OFF</i>	L7 OFF	15	<i>Out 1 ON</i>	L7 ON
8	<i>Out 8 OFF</i>	L8 OFF	16	<i>Out 1 ON</i>	L8 ON

Dari pengujian *push button* dan LED pada tabel 4.5 maka dapat disimpulkan *push button* dan LED berhasil bekerja dengan baik, gambar 4.9 merupakan hasil rangkaian pengujian *push button* dan LED.

4.3.3. Pengujian Saklar *Toggle* dan LED

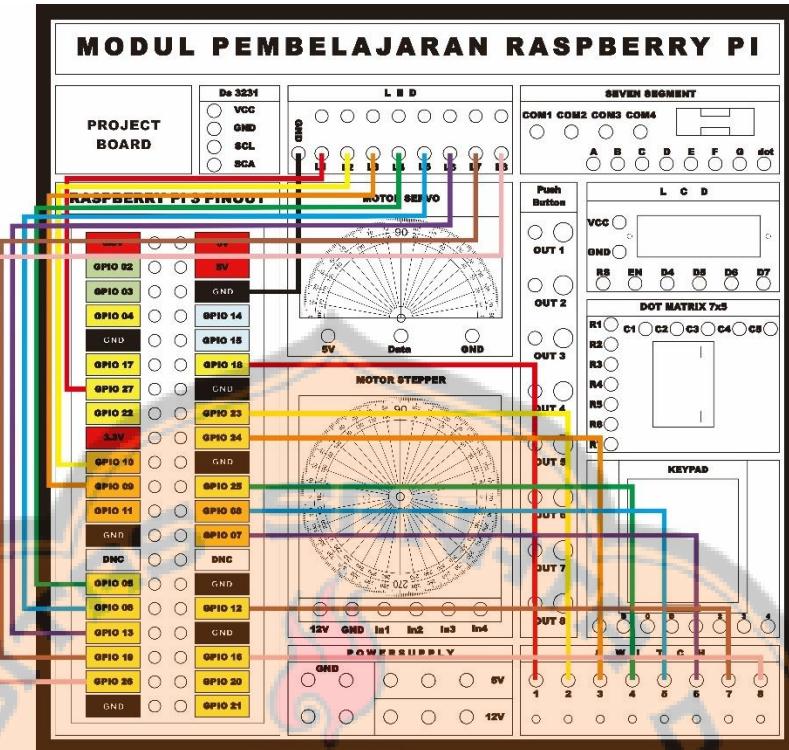
Rangkaian saklar *toggle* yang terhubung dengan keluaran pada boks merupakan kaki tengah pada saklar, kemudian 2 kaki lainnya terhubung dengan tegangan 5 volt dan *ground*. Saklar *toggle* dikondisikan sebagai *input* sedangkan *output* berupa LED menggunakan *setmode BOARD*. Rangkaian saklar *toggle* dan LED yang dihubungkan seperti pada gambar 4.10, untuk pengkondisian *input* dan *output* program sama dengan pengujian *push button* dan LED, *listing* program keseluruhan dapat dilihat pada lampiran *listing* program saklar *toggle* dan LED Rangkaian pengujian saklar *toggle* dan LED dapat dilihat pada tabel 4.6.



Gambar 4.10. Rangkaian saklar *toggle* dan LED

Tabel 4.6. Rangkaian pengujian saklar *toggle* dan LED

Saklar <i>Toggle</i> (<i>Switch</i>)	GPIO (BOARD)		LED
Saklar <i>Toggle</i> 1	13 IN	12 OUT 16 OUT 18 OUT 22 OUT 24 OUT 26 OUT 32 OUT 36 OUT	L1
Saklar <i>Toggle</i> 2	19 IN		L2
Saklar <i>Toggle</i> 3	21 IN		L3
Saklar <i>Toggle</i> 4	29 IN		L4
Saklar <i>Toggle</i> 5	31 IN		L5
Saklar <i>Toggle</i> 6	33 IN		L6
Saklar <i>Toggle</i> 7	35 IN		L7
Saklar <i>Toggle</i> 8	37 IN		L8

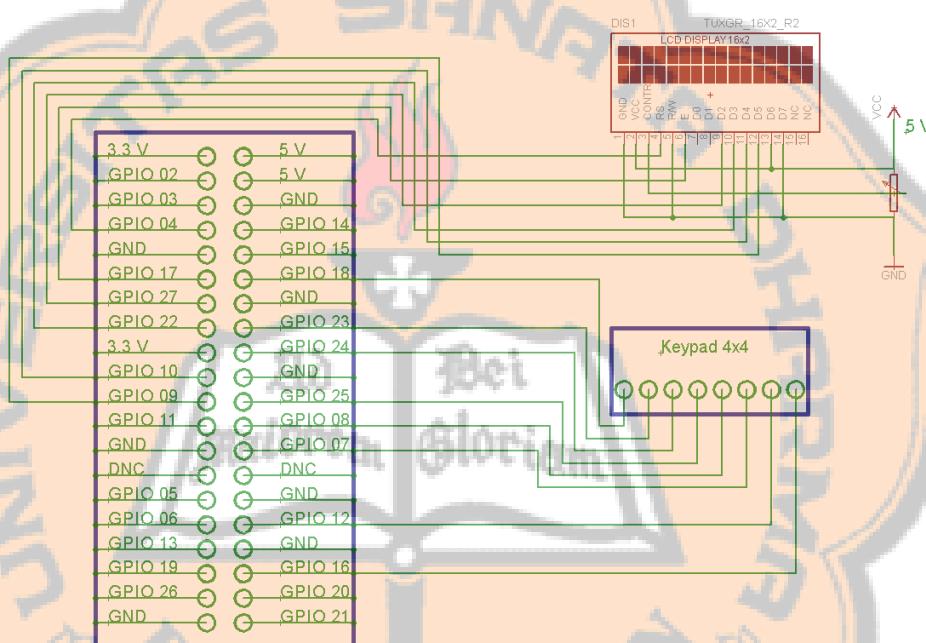
Gambar 4.11. Hasil rangkaian pengujian saklar *toggle* dan LEDTabel 4.7. Pengujian saklar *toggle* dan LED

No	Saklar Toggle	LED	No	Saklar Toggle	LED
1	Saklar <i>Toggle</i> 1 OFF	L1 OFF	9	Saklar <i>Toggle</i> 1 ON	L1 ON
2	Saklar <i>Toggle</i> 2 OFF	L2 OFF	10	Saklar <i>Toggle</i> 1 ON	L2 ON
3	Saklar <i>Toggle</i> 3 OFF	L3 OFF	11	Saklar <i>Toggle</i> 1 ON	L3 ON
4	Saklar <i>Toggle</i> 4 OFF	L4 OFF	12	Saklar <i>Toggle</i> 1 ON	L4 ON
5	Saklar <i>Toggle</i> 5 OFF	L5 OFF	13	Saklar <i>Toggle</i> 1 ON	L5 ON
6	Saklar <i>Toggle</i> 6 OFF	L6 OFF	14	Saklar <i>Toggle</i> 1 ON	L6 ON
7	Saklar <i>Toggle</i> 7 OFF	L7 OFF	15	Saklar <i>Toggle</i> 1 ON	L7 ON
8	Saklar <i>Toggle</i> 8 OFF	L8 OFF	16	Saklar <i>Toggle</i> 1 ON	L8 ON

Dari pengujian saklar *toggle* dan LED pada tabel 4.7 maka dapat disimpulkan pengujian *push button* dan LED berhasil bekerja dengan baik, gambar 4.11 merupakan hasil rangkaian pengujian *push button* dan LED.

4.3.4. Pengujian Keypad dan LCD

Rangkaian *keypad* dan LCD diuji dengan memasukan data berupa angka 0 hingga 9 yang akan tertampil pada LCD 16x2. Terdapat potensio didalam boks modul *raspberry pi* untuk mengatur kontras pada tampilan LCD, *keypad* 4x4 sebagai *input* data angka 0 hingga 9 dan kemudian akan ditampilkan LCD 16x2. Rangkaian *keypad* dan LCD yang dihubungkan seperti pada gambar 4.12, untuk rangkaian pengujian menggunakan *setmode BCM* yang dapat dilihat pada tabel 4.8, *listing* program keseluruhan bisa dilihat pada lampiran *listing* program *keypad* dan LCD.



Gambar 4.12. Rangkaian *keypad* dan LCD 16x2

Tabel 4.8. Rangkaian pengujian *keypad* dan LCD 16x2

<i>Keypad</i>	<i>GPIO (BCM)</i>		LCD
A	GPIO 18 IN	5 V GND GPIO 4 OUT GPIO 17 OUT GPIO 27 OUT GPIO 22 OUT GPIO 10 OUT GPIO 9 OUT	5 V
B	GPIO 23 IN		GND
C	GPIO 24 IN		RS
D	GPIO 25 IN		EN
1	GPIO 08 IN		D4
2	GPIO 07 IN		D5
3	GPIO 12 IN		D6
4	GPIO 16 IN		D7

```

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)

row =2
col =16
rs =4
e =17
d4 =27
d5 =22
d6 =10
d7 =9

lcd = LCD.Adafruit_CharLCD(rs,e,d4,d5,d6,d7,col,row)

MATRIX = [[1,2,3,'A'],
           [4,5,6,'B'],
           [7,8,9,'C'],
           ['*',0,'#','D']]

ROW = [18,23,24,25]
COL = [8,7,12,16]

```

Gambar 4.13. Pengkondisian *input* dan *output keypad* dan LCD 16x2

Pada gambar 4.13 merupakan *listing* program pengkondisian *input* dan *output keypad* dan LCD yang digunakan untuk menguji *keypad* dan LCD, untuk merangkai pengujian *keypad* dan LCD dapat dilihat pada tabel 4.8.

```

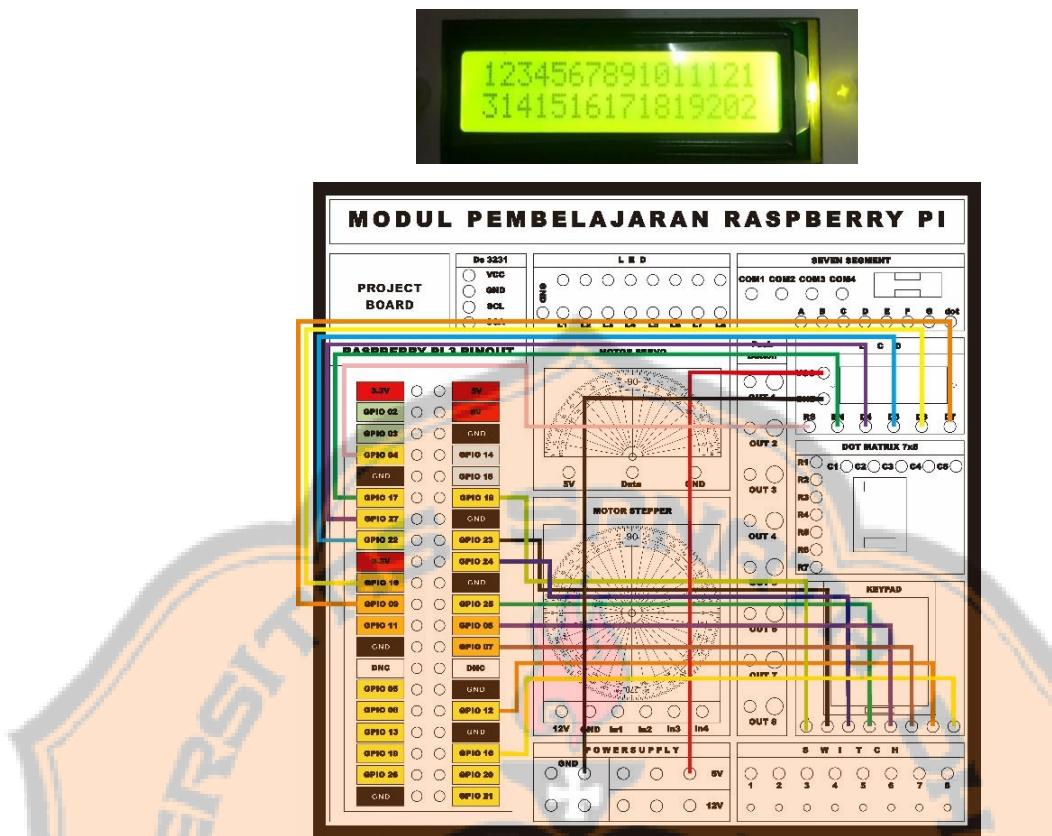
try:
    while(True):
        if c>15 and b==1:
            lcd.clear()
            c=-1
            b=0
        if c>14 and b==0:
            c=-1
            b=1

        if bar==0 and kol==0:
            c=c+1
            lcd.set_cursor(c,b)
            lcd.message('1')

```

Gambar 4.14. Pengkondisian *keypad* dan LCD16x2

Pada gambar 4.14 merupakan *listing* program pengkondisian *keypad* dan LCD. *Keypad* dikondisikan sebagai *input* data kemudian data tersebut akan ditampilkan pada LCD, gambar 4.14 merupakan pengkondisian *keypad* dengan menggunakan fungsi jika (*if*) seperti “*if bar==0 and kol==0:*”, “*c=c+1*”, *lcd.set_cursor(c,b)*”, *lcd.message('1')*” merupakan perintah pada *keypad* yang alamat baris 0 dan kolom 0 maka LCD dikirim data 1, kemudian *set cursor* akan berpindah kekolom selanjutnya dari perintah *c=c+1*, maka *cursor* akan bergeser terus ketika data sudah dikirim.



Gambar 4.15. Hasil rangkaian pengujian *keypad* dan LCD 16x2

Dari gambar 4.15 merupakan hasil pengujian *keypad* dan LCD 16x2 yang masukan data berupa angka 0-9 yang tertampil pada LCD, maka dapat disimpulkan bahwa pengujian *keypad* dan LCD berhasil bekerja dengan baik.

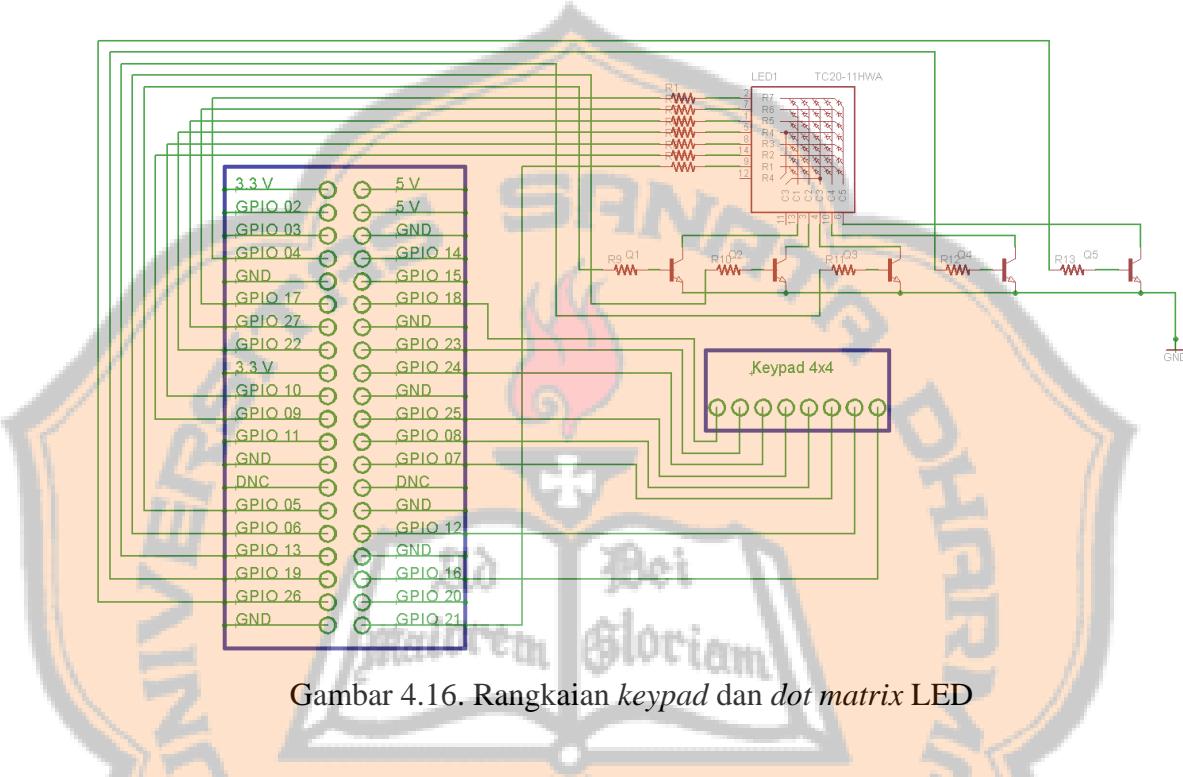
4.3.5. Pengujian *Keypad* dan *Dot matrix LED*

Pada rangkaian *dot matrix* LED terdapat tambahan komponen yaitu transistor 2n2222A dikarenakan saat *dot matrix* LED menyala tidak terang sehingga dibutuhkan penguatan arus. Pada transistor kaki emitor terhubung dengan ground, kaki kolektor terhubung dengan *dot matrix* LED, sedangkan kaki basis terhubung dengan boks modul pembelajaran *raspberry pi*.

Pengkondisian *output* pada *dot matrix* LED adalah per baris dan kolom, sehingga untuk pemanggilan satu huruf terdapat 7 baris dan kolumnya yang dikondisikan. Rangkaian *keypad* dan *dot matrix* LED yang dihubungkan seperti pada gambar 4.16, untuk gambar 4.17 merupakan pengkondisian *input* dan *output* *keypad* dan *dot matrix* menggunakan setmode

BOARD, untuk *listing* program keseluruhan bisa dilihat pada lampiran *listing* program *keypad* dan *dotmatrix*.

Rangkaian pengujian *keypad* dan *dot matrix* dapat dilihat pada tabel 4.9. Keberhasilan pengujian berupa masuknya data angka A-D dan tertampil pada *dot matrix* LED yang dapat dilihat pada gambar 4.18.



Gambar 4.16. Rangkaian *keypad* dan *dot matrix* LED

Tabel 4.9. Rangkaian pengujian *keypad* dan *dot matrix* LED

Keypad	GPIO (BOARD)		Dot Matrix
A	12 IN	07 OUT 11 OUT 13 OUT 15 OUT 19 OUT 21 OUT 40 OUT 29 OUT 31 OUT 33 OUT 35 OUT 38 OUT	R1
B	16 IN		R2
C	18 IN		R3
D	22 IN		R4
1	24 IN		R5
2	26 IN		R6
3	32 IN		R7
4	36 IN		C1
--	--		C2
--	--		C3
--	--		C4
--	--		C5

```

def A():
#ROWS1A
    gpio.output (7, gpio.HIGH)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom1A
    gpio.output (29,  gpio.LOW)
    gpio.output (31,  gpio.LOW)
    gpio.output (33,  gpio.HIGH)
    gpio.output (35,  gpio.LOW)
    gpio.output (38,  gpio.LOW)
    time.sleep(t)

#ROWS2A
    gpio.output (7,  gpio.LOW)
    gpio.output (11,  gpio.HIGH)
    gpio.output (13,  gpio.LOW)
    gpio.output (15,  gpio.LOW)
    gpio.output (19,  gpio.LOW)
    gpio.output (21,  gpio.LOW)
    gpio.output (40,  gpio.LOW)
#Colom2A
    gpio.output (29,  gpio.LOW)
    gpio.output (31,  gpio.HIGH)
    gpio.output (33,  gpio.LOW)
    gpio.output (35,  gpio.HIGH)
    gpio.output (38,  gpio.LOW)
    time.sleep(t)

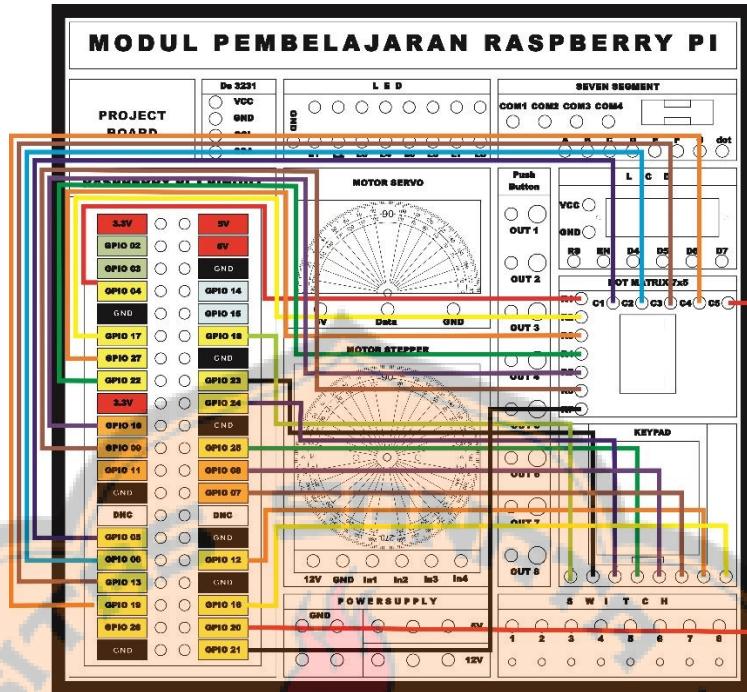
try:
    while(True):
        if bar==0 and kol==3:
            A()
        elif bar==1 and kol==3:
            B()
        elif bar==2 and kol==3:
            C()
        elif bar==3 and kol==3:
            D()
        for j in range(4):
            gpio.output(COL[j],0)
            for i in range(4):
                if gpio.input(ROW[i]) == 0:
                    print MATRIX[i][j]
                    bar = i
                    kol = j
                    time.sleep(0.2)
                while(gpio.input(ROW[i]) == 0):
                    pass
                    gpio.output(COL[j],1)
except KeyboardInterrupt:
    gpio.cleanup()

```

Gambar 4.17. Pengkondisian *input* dan *output keypad* dan *dot matrix LED*

Pada gambar 4.17 merupakan *listing* program *input* dan *output keypad* dan *dot matrix LED* yang digunakan juga sebagai pengujian *keypad* dan *dot matrix LED*. Untuk menyalakan *dot matrix LED* harus diatur satu per satu seperti baris dan kolom contoh pada *listing* program “*def A():*”, perintah tersebut merupakan pemanggilan karakter huruf “A” yang sebelumnya telah dikondisikan *high/low* untuk membentuk huruf “A”.

Pada program seperti “*try*” yang didalam program tersebut sama dengan pengujian *keypad* dan LCD, dengan mengatur baris dan kolom tetapi perbedaannya adalah pengujian ini memanggil fungsi seperti “*A()*”, “*B()*”, “*C()*”, dan “*D()*” kemudian akan tertampil pada *dot matrix LED*.



Gambar 4.18. Hasil pengujian *keypad* dan *dot matrix* LED

Dari gambar 4.18 merupakan hasil pengujian *keypad* dan *dot matrix* LED yang masukan data berupa angka A-D yang tertampil pada *dot matrix* LED, maka dapat disimpulkan bahwa pengujian *keypad* dan *dot matrix* LED berhasil bekerja dengan baik.

4.3.6. Pengujian Motor Servo

Rangkaian motor *servo* dapat dilihat pada tabel 4.10. motor *servo* dihubungkan langsung pada boks modul *raspberry pi*. Motor *servo* pada pengujian ini memiliki nilai *dutycycle* 12% untuk gerakan kekiri penuh dan 2% untuk gerakan kekanan penuh sedangkan untuk gerakan tengah yaitu 7.5%. Posisi *servo* dikendalikan oleh lebar pulsa sebesar 50 Hz *PWM (Pulse Width Modulation) signal*. Berdasarkan perhitungan seperti berikut:

$$\text{DutyCycle} = \text{PulseWidth}/\text{Period} \quad (4.1)$$

$$\text{Period} = 1/\text{Frequency} \quad (4.2)$$

$$\begin{aligned} \text{DutyCycle} &= \text{PulseWidth} : \left(\frac{1}{\text{frequency}} \right) \\ &= \text{PulseWidth} * \text{frequency} \end{aligned} \quad (4.3)$$

Jadi untuk gerakan penuh kekiri, ketengah, dan kekanan didapat dari persamaan diatas. Dengan persamaan diatas gerakan untuk ketengah kurang tepat sehingga *dutycycle* dicoba melalui program *python* hingga ditemukan gerakan ketengah dibutuhkan *dutycycle* sebesar 7%. Sehingga didapat untuk gerakan penuh menuju 0 derajat dibutuhkan *dutycycle* sebesar 2% dan untuk menuju 180 derajat dibutuhkan *dutycycle* sebesar 12%.

Kemudian dari *dutycycle* yang sudah didapat masing-masing gerakan penuh kekiri dan kekanan dapat dihitung persamaan garis untuk membantu pengoperasian motor *servo* saat bekerja. Untuk posisi 0 derajat dengan *dutycycle* 2 sedangkan 180 derajat dengan *dutycycle* 12 sehingga :

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{12 - 2}{180 - 0} = 1/8 \quad (4.4)$$

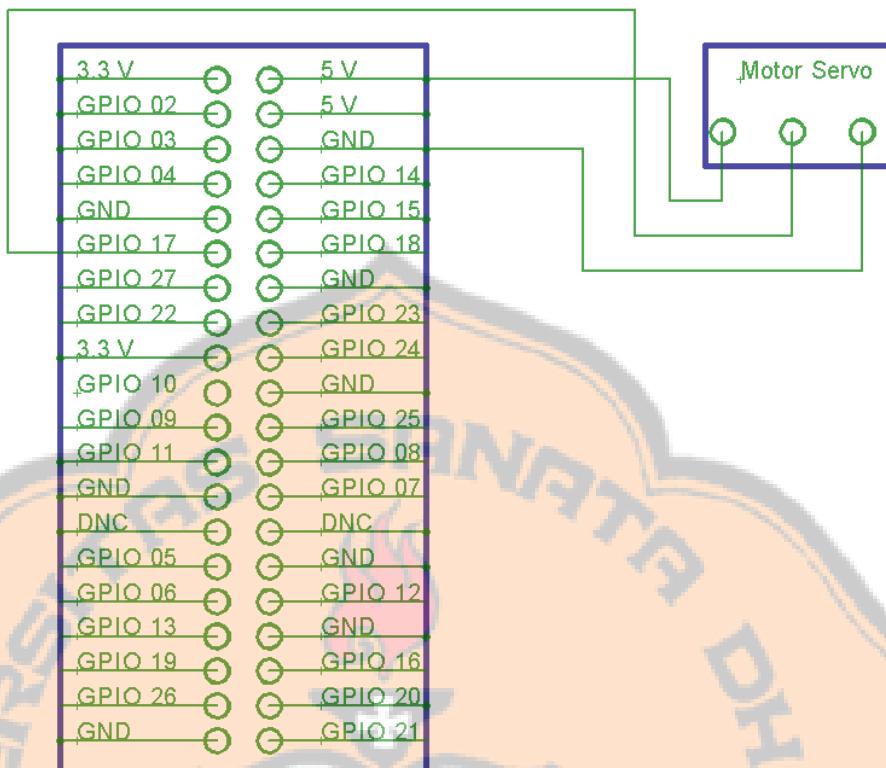
Sehingga dapat diasumsikan posisi awal yaitu *dutycycle* 2 dengan posisi awal 0 derajat menjadi:

$$\begin{aligned} y - y_1 &= m * (x - x_1) \\ y - 2 &= \frac{1}{8} * (x - 0) \\ y &= \frac{1}{8} * x + 2 \end{aligned} \quad (4.5)$$

Sehingga x sebagai masukan pada program *python* sebagai *input* untuk keputusan menuju derajat yang diinginkan. Pada gambar 4.20 merupakan hasil dari perhitungan diatas yang digunakan sebagai pengoperasian motor *servo* untuk program keseluruhan dapat dilihat pada lampiran *listing* program motor *servo*, rangkaian motor *servo* yang dihubungkan seperti pada gambar 4.19, untuk hasil rangkaian motor *servo* yang telah diuji dapat dilihat pada gambar 4.22.

Tabel 4.10. Rangkaian pengujian motor *servo*

GPIO (BOARD)	Motor Servo
5 V	5 V
GND	GND
11 IN	Data



Gambar 4.19. Rangkaian pengujian motor servo

```

import RPi.GPIO as gpio
gpio.setmode (gpio.BOARD)
gpio.setwarnings(False)

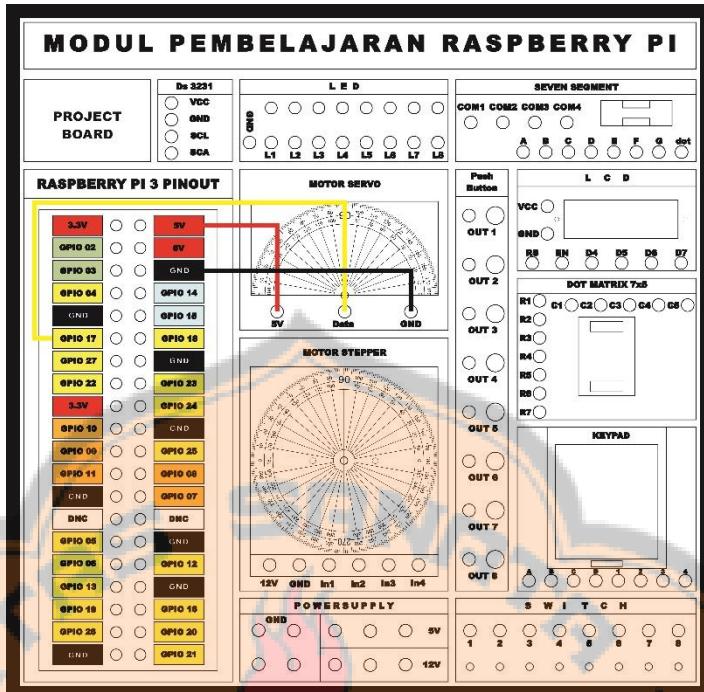
gpio.setup(11,gpio.OUT)
pwm= gpio.PWM(11,50)

pwm.start(2)
##pwm.ChangeDutyCycle(1)
for i in range (0,100):
    desiredPosition=input("kemana servo? 0-180 derajat : ")
    DC=1./18.* (desiredPosition)+2
    pwm.ChangeDutyCycle(DC)

```

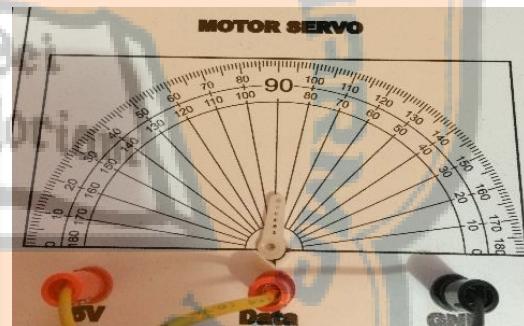
Gambar 4.20. Pengkondisian output dan keputusan posisi motor servo

Pada gambar 4.20 merupakan *listing* program pengkondisian *output* dan keputusan posisi motor *servo* sekaligus pengujian pada motor *servo*, gpio pin 11 diatur sebagai *output* kemudian pada program seperti “`pwm= gpio.PWM(11,50)`” merupakan perintah *output* pada gpio pin 11 sama dengan `pwm` dan gpio pin 11 diberikan frekuensi 50hz, nilai tersebut merupakan spesifikasi motor *servo* yang digunakan.



Gambar 4.21. Hasil rangkaian pengujian motor servo

```
Python 2.7.9 (default, Sep 17 2011, [GCC 4.9.2] on linux2
Type "copyright", "credits" or "l
>>> =====
>>>
kemana servo? 0-180 derajat : 90
kemana servo? 0-180 derajat :
```



Gambar 4.22. Hasil pengujian motor servo

Dari gambar 4.22 merupakan hasil pengujian motor *servo* yang masukan data berupa sudut derajat yang dapat dilihat pada boks modul pembelajaran *raspberry pi*, maka dapat disimpulkan bahwa pengujian motor *servo* berhasil bekerja dengan baik.

4.3.7. Pengujian Motor Stepper

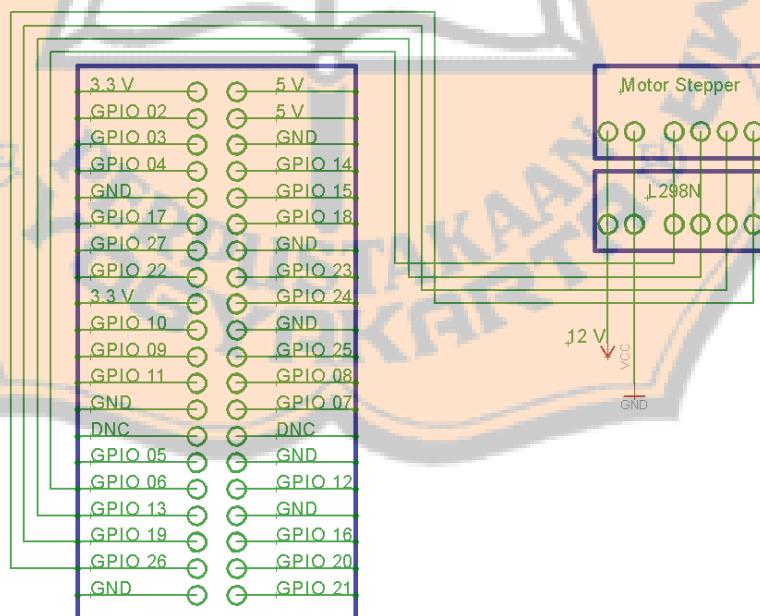
Pada pengujian motor *stepper* tidak menggunakan *driver* motor seperti perancangan sebelumnya dikarenakan kesalahan dalam menentukan *driver* yang sesuai dengan motor yang digunakan dan juga saat pengujian *driver* motor (ULN2003) hanya bisa aktif satu kutub

pada motor *stepper* yang digunakan, sedangkan motor *stepper* yang digunakan akan aktif bergerak satu step ketika dua kutub pada motor *stepper* diaktifkan.

Driver motor *stepper* yang gunakan adalah L298N dengan fungsi yang sama dan dapat digunakan pada motor yang bertegangan 12 volt. L298N bisa digunakan pada motor *stepper* dari tegangan 5 volt hingga 35 volt, sesuai dengan karakter *driver* ini mampu mengendalikan dua motor dengan menggunakan satu *driver*. Sehingga digunakanlah module L298N sebagai *driver* motor *stepper* pada pengujian ini. Tabel 4.11 menunjukkan rangkaian pengujian motor *stepper*.

Motor *stepper* dikondisikan sebagai *output* yang akan aktif ketika diberi logika “high” sebaliknya tidak aktif ketika diberi logika “low” seperti gambar 4.24 program keseluruhan dapat dilihat pada lampiran *listing* program motor *stepper*. Untuk mengerakkan satu step motor *stepper* dibutuhkan 2 kutub yang aktif, pergerakan setiap satu step pada motor *stepper* yaitu 7.5 derajat. Rangkaian motor *stepper* yang dihubungkan seperti pada gambar 4.23.

Motor *stepper* pada pengujian ini tidak dapat mengetahui posisi nol karena motor *stepper* tidak memiliki sensor gerak, motor *stepper* akan bekerja ketika nilai yang dimasukan lebih besar dan lebih kecil dari nilai sebelumnya dengan catatan sesuai kemampuan spesifikasi motor *stepper*.



Gambar 4.23. Rangkaian pengujian motor *stepper*

Tabel 4.11. Rangkaian pengujian motor *stepper*

<i>Power supply</i>	GPIO (BOARD)	Motor <i>Stepper</i>
12 V	--	12 V
GND	--	GND
--	31 IN	Ln 1
--	33 IN	Ln 2
--	35 IN	Ln 3
--	37 IN	Ln 4

```

import RPi.GPIO as gpio
from time import sleep

a=31
b=33
c=35
d=37

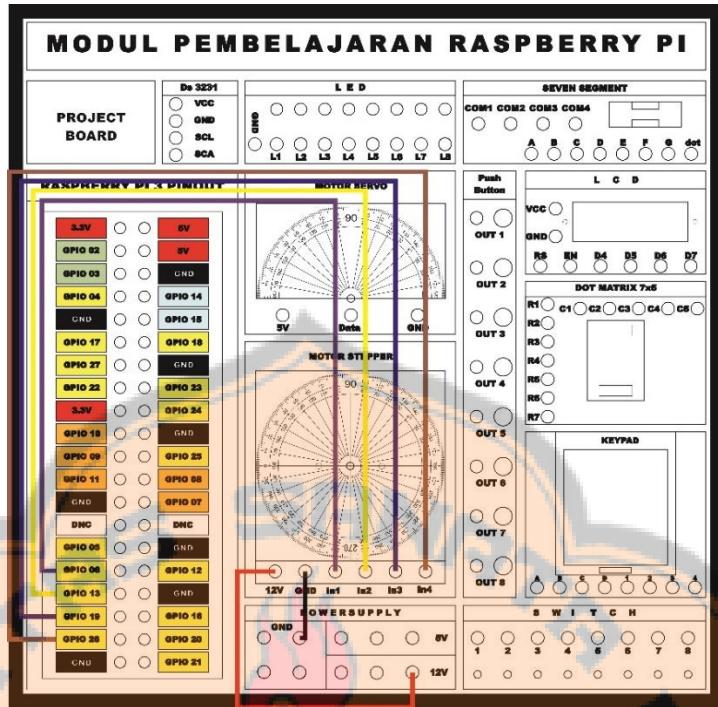
gpio.setmode(gpio.BOARD)
gpio.setwarnings(False)

gpio.setup(a,gpio.OUT)
gpio.setup(b,gpio.OUT)
gpio.setup(c,gpio.OUT)
gpio.setup(d,gpio.OUT)
t=0.5

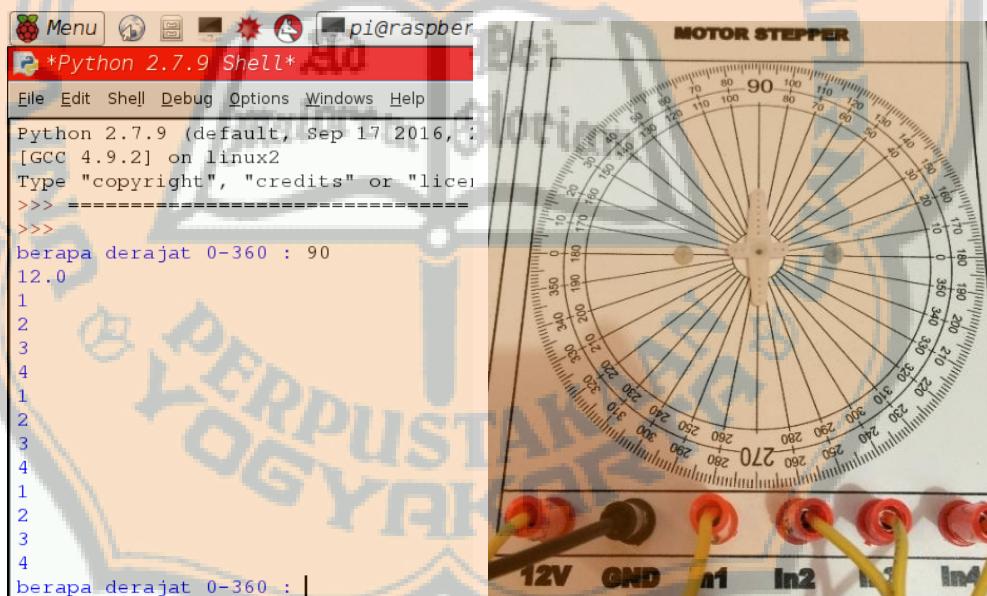
```

Gambar 4.24. Pengkondisian *output* motor *stepper*

Pada gambar 4.24 merupakan *listing* program pengkondisian *output* motor *stepper*, motor *stepper* yang digunakan memiliki 4 *input* sebagai pengatur putaran *step*. Pada pengujian motor *stepper* menggunakan gpi pin 31, 33, 35, dan 37 yang diatur sebagai *output*, motor *stepper* menggunakan tegangan sebesar 12V.



Gambar 4.25. Hasil rangkaian pengujian motor stepper

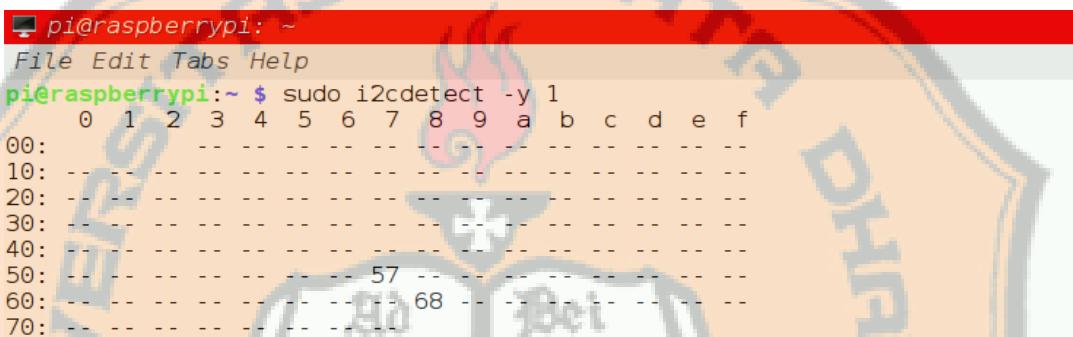


Gambar 4.26. Hasil pengujian motor stepper

Dari gambar 4.26 merupakan hasil pengujian motor *stepper* yang masukan data berupa sudut derajat yang dapat dilihat pada boks modul pembelajaran *raspberry pi*, maka dapat disimpulkan bahwa pengujian motor *stepper* berhasil bekerja dengan baik.

4.3.8. Pengujian Komunikasi *I2c* Modul RTC Ds 3231 dan *Seven segment*

Pengujian komunikasi *I2c* pada modul pembelajaran *raspberry pi* dengan *seven segment* menampilkan penunjuk waktu berupa jam dan menit. Sebelum masuk dalam program terlebih dahulu menghubungkan *port I2c* berupa SDA (*Serial Data*) dan SCL (*Serial Clock*) dengan modul RTC ds 3231 dan tegangan 3.3 volt kemudian *ground*. Langkah selanjutnya buka Tx_Terminal pada *raspberry pi* >> ketik “ sudo raspi-config” >> Edvanced Options >> *I2c* >> Enable *I2c* Interface >> Finish. Untuk melihat apakah *I2c* sudah tersambung kemudian ketik “ sudo *I2cdetect -y 1*”, ketika muncul gambar 4.27 maka komunikasi *I2c* telah terhubung dengan modul pembelajaran *raspberry pi*.



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi: ~ $ sudo i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: -----
20: -----
30: -----
40: -----
50: -----
60: -----
70: -----
          57
          68
```

Gambar 4.27. Hasil pengujian alamat *I2c*

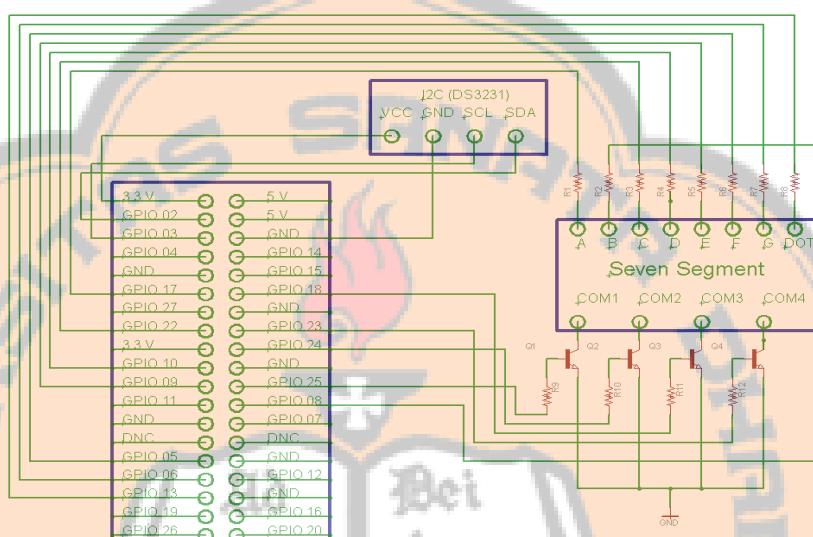
Sambungkan rangkaian *seven segment* sesuai dengan *input* dan *output* pada program *python* seperti tabel pengujian komunikasi *I2c* modul RTC ds3231 dan *seven segment*. Sebelumnya telah dipastikan bahwa *seven segment* bekerja dengan baik sehingga *file seven segment* dapat dipanggil pada program *I2c* pada *python*. Program pada *I2c* dapat dilihat pada gambar 4.29 keseluruhan progam pada lampiran *listing* program *I2c* dan *seven segment*. Rangkaian pengujian komunikasi *i2c* seperti gambar 4.28. Hasil pengujian *I2c* dapat dilihat pada gambar 4.30.

Tabel 4.12. Rangkaian pengujian *I2c* dan *seven segment*

<i>I2c</i>	<i>GPIO (BOARD)</i>			<i>Seven segment</i>
3.3 V	3.3 V		22 OUT	COM 1
GND	GND		18 OUT	COM 2
SDA	3 IN		12 OUT	COM 3
SCL	5 IN		16 OUT	COM 4
--	--		11 OUT	A
--	--		24 OUT	B

Tabel 4.12. (lanjutan) Rangkaian pengujian I₂C dan seven segment

I ₂ C	GPIO (BOARD)			Seven segment
--	--	15 OUT 19 OUT 21 OUT 29 OUT 31 OUT 33 OUT	15 OUT	C
--	--		19 OUT	D
--	--		21 OUT	E
--	--		29 OUT	F
--	--		31 OUT	G
--	--		33 OUT	DOT

Gambar 4.28. Rangkaian komunikasi I₂C

```

import sys
sys.path.insert(0,'/home/pi')
import segmen
import time
import datetime
import random
import SDL_DS3231

ds3231 = SDL_DS3231.SDL_DS3231(1, 0x68)
ds3231.write_now()
t=0.001

while True:

    bil=ds3231._read_minutes()
    satuan = bil % 10
    puluhan = bil / 10

    bil2=ds3231._read_hours()
    satuan2 = bil2 % 10
    puluhan2 = bil2 / 10

    segmen.t_jam(satuan2,puluhan2,t)
    segmen.t_menit(satuan,puluhan,t)

import RPi.GPIO as gpio
import time

gpio.setmode(gpio.BOARD)
gpio.setwarnings(False)
#COM1-4
gpio.setup(22,gpio.OUT)
gpio.setup(18,gpio.OUT)
gpio.setup(12,gpio.OUT)
gpio.setup(16,gpio.OUT)
#A-G
gpio.setup(11,gpio.OUT)
gpio.setup(24,gpio.OUT)
gpio.setup(15,gpio.OUT)
gpio.setup(19,gpio.OUT)
gpio.setup(21,gpio.OUT)
gpio.setup(29,gpio.OUT)
gpio.setup(31,gpio.OUT)
#::: with com2/3
gpio.setup(33,gpio.OUT)

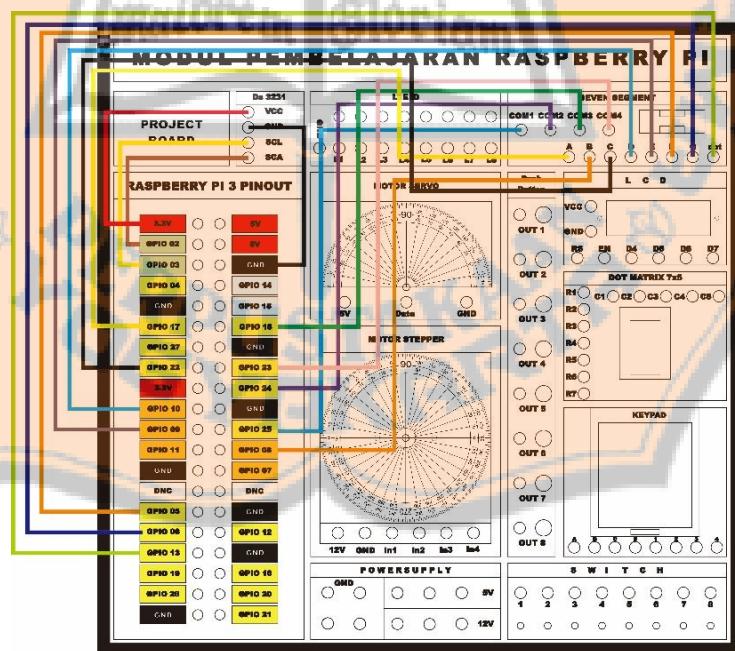
def com(n):
    if n == 1:
        gpio.output(22,gpio.HIGH)
        gpio.output(18,gpio.LOW)
        gpio.output(12,gpio.LOW)
        gpio.output(16,gpio.LOW)
    if n == 2:

```

Gambar 4.29. Pengkondisian komunikasi I₂C

Pada gambar 4.29 merupakan *listing* program pengkondisian komunikasi *I2c* sekaligus merupakan program pengujian komunikasi *I2c* dengan *seven segment*. Didalam program “while True:” terdapat fungsi pemanggilan yaitu pada program “bil=ds3231._read_minutes()” merupakan perintah bil sama dengan membaca modul rtc ds3231 penunjuk waktu menit bagitu juga dengan pembacaan bil2 untuk membaca penunjuk waktu jam. Untuk menampilkan satuan digunakan perhitungan seperti “bil % 10” yang artinya sisa bagi dari nilai waktu baca dibagi 10 kemudian untuk perhitungan puluhan seperti “bil / 10” yang artinya nilai waktu yang dibaca dibagi 10, kemudian satuan dan puluhan akan ditampilkan pada *seven segment*.

Bilangan yang telah dibaca melalui hasil perhitungan akan ditampilkan tetapi sebelumnya harus mengatur posisi segmen mana yang akan menyala untuk waktu menit dan jam dapat dilihat bagian program pada “def com(n):”. angka terdiri dari 0-9 yang sebelumnya telah diatur dalam sebuah program segmen untuk ditampilkan seperti sama halnya dengan *dot matrix* yang digunakan sebelumnya, angka tersebut telah diatur kemudian tinggal dengan pemanggilan angka berapa yang akan ditampilkan pada *seven segment*.



Gambar 4.30. Rangkaian hasil pengujian komunikasi *I2c*

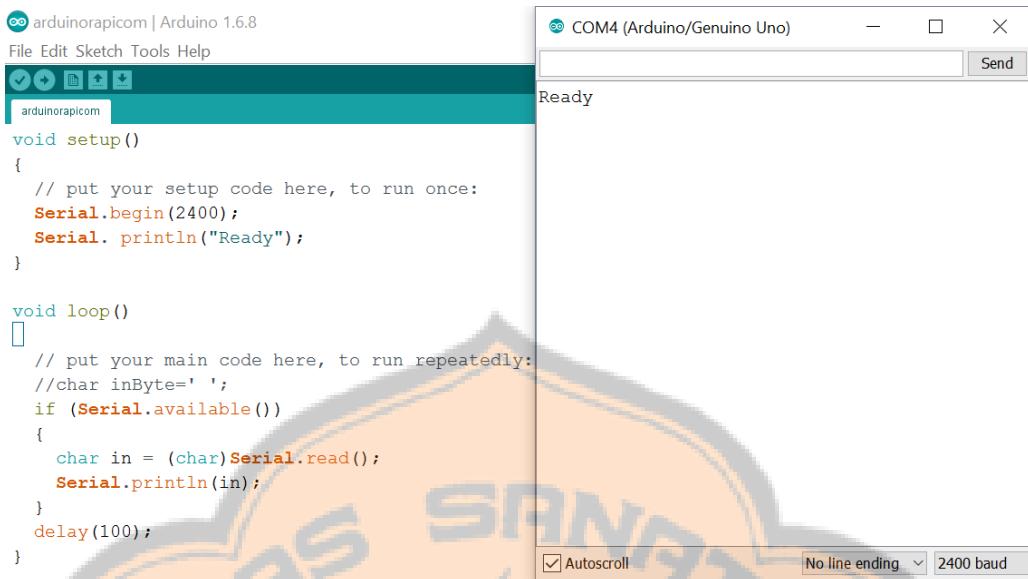
Dari gambar 4.30 merupakan hasil pengujian komunikasi *I2c* yang masukan data berupa penunjuk waktu yaitu jam dan menit yang tertampil pada *seven segment* pada boks

modul pembelajaran *raspberry pi*, maka dapat disimpulkan bahwa pengujian komunikasi *I2c* dengan *seven segment* berhasil bekerja dengan baik.

4.3.9. Penggunaan dan Pengujian komunikasi UART dan Arduino Uno

Pengujian komunikasi UART dibagi menjadi 2 tahap, tahap pertama komunikasi UART dari raspberry ke arduino dan tahap kedua arduino ke *raspberry pi*. Pengujian komunikasi UART antara modul pembelajaran *raspberry pi* dengan menggunakan arduino uno. Tahap pertama data dikirim dari modul pembelajaran *raspberry pi* yang akan diterima oleh arduino, penampilan data yang diterima oleh arduino dapat dilihat pada serial monitor. Langkah penggunaan dan pengujian diurutkan sebagai berikut:

1. Menyalakan arduino dengan laptop dan megoperasikan program arduino sebagai penerima data tanpa terhubung dengan modul pembelajaran *raspberry pi* gambar 4.31.
2. Upload program penerima data arduino dan buka serial monitor, kemudian sambungkan pin Tx *raspberry pi* ke Rx arduino dan hubungkan ground *raspberry pi* dengan arduino tabel 4.13.
3. Kemudian buka terminal untuk mengatur baudrate komunikasi yang akan digunakan dengan sintak gambar 4.32, baudrate komunikasi pada arduino dan *raspberry pi* juga disamakan.
4. Jalankan program pada modul pembelajaran *raspberry pi* dengan menekan tombol keyboard “F5” atau dengan klik “Run >> Run Module” kemudian masukan karakter yang akan dikirim gambar 4.33.
5. Data akan terkirim pada serial monitor arduino gambar 4.34.



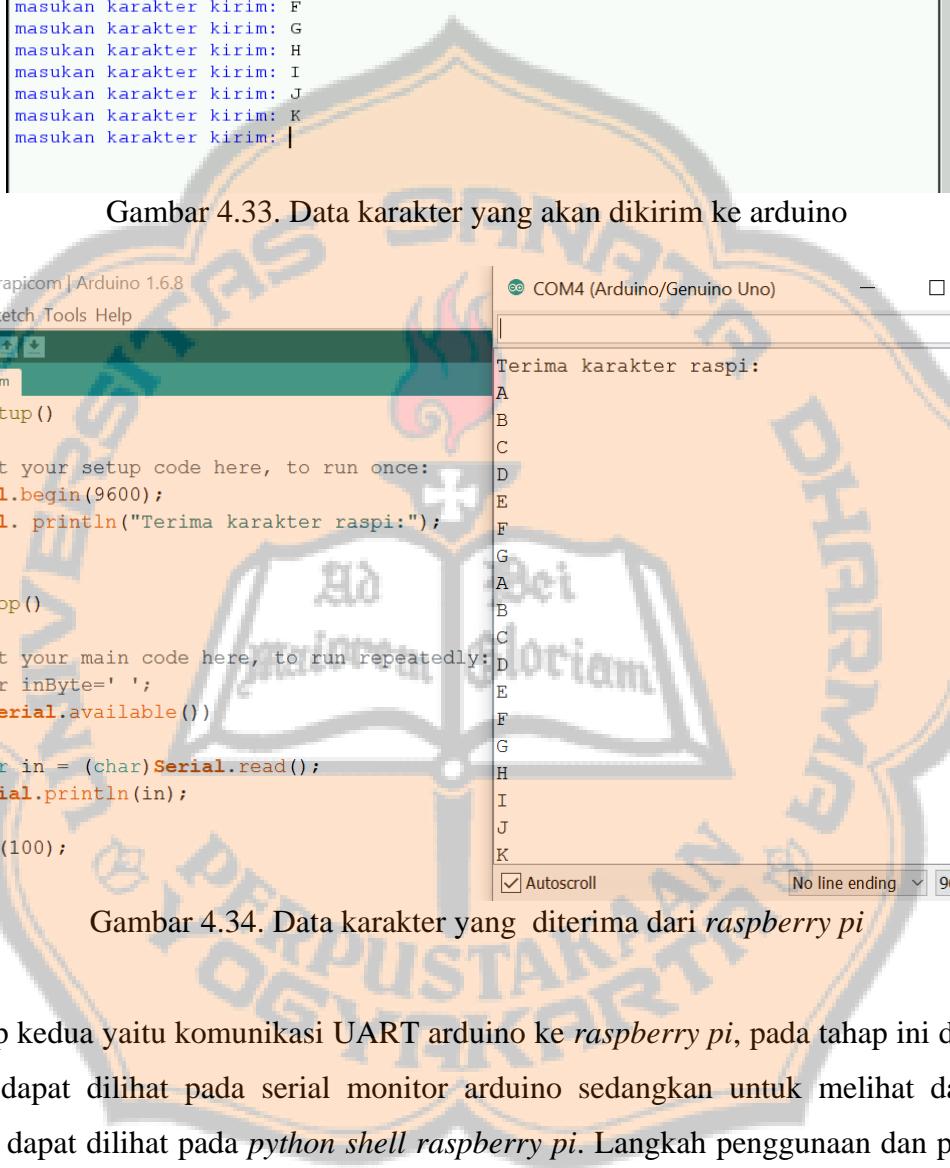
Gambar 4.31. Menyalakan dan mengoperasikan program arduino

Pada gambar 4.31 merupakan *listing* program menyalakan dan mengoperasikan program arduino sekaligus untuk pengujian, arduino sebagai penerima data yang akan dikirim dari *raspberry pi*. Fungsi untuk membaca data yang akan diterima dapat dilihat pada program “char in = (char)Serial.read();” merupakan perintah untuk membaca komunikasi serial dan juga membaca data karakter yang dikirim dari *raspberry pi*.

Tabel 4.13. Rangkaian komunikasi UART *raspberry pi* ke arduino

Modul Pembelajaran <i>Raspberry pi</i>	Arduino
GPIO 14 (Tx)	Rx
<i>Ground</i>	<i>Ground</i>

Gambar 4.32. Pengaturan baudrate komunikasi *raspberry pi*



```
*Python 2.7.9 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
masukan karakter kirim: A
masukan karakter kirim: B
masukan karakter kirim: C
masukan karakter kirim: D
masukan karakter kirim: E
masukan karakter kirim: F
masukan karakter kirim: G
masukan karakter kirim: H
masukan karakter kirim: I
masukan karakter kirim: J
masukan karakter kirim: K
masukan karakter kirim: |
```

Gambar 4.33. Data karakter yang akan dikirim ke arduino



arduinorapicom | Arduino 1.6.8

```
File Edit Sketch Tools Help
arduinorapicom
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println("Terima karakter raspi:");
}

void loop()
{
    // put your main code here, to run repeatedly:
    //char inByte=' ';
    if (Serial.available())
    {
        char in = (char)Serial.read();
        Serial.println(in);
    }
    delay(100);
}
```

COM4 (Arduino/Genuino Uno)

Terima karakter raspi:
A
B
C
D
E
F
G
A
B
C
D
E
F
G
H
I
J
K

Autoscroll No line ending 9600 baud

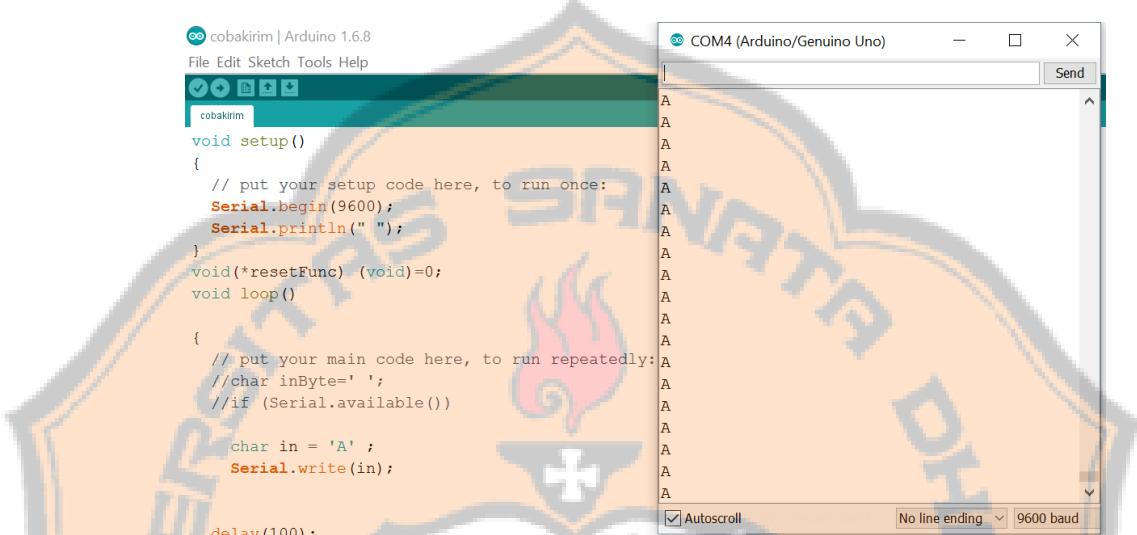
Gambar 4.34. Data karakter yang diterima dari raspberry pi

Tahap kedua yaitu komunikasi UART arduino ke *raspberry pi*, pada tahap ini data yang dikirim dapat dilihat pada serial monitor arduino sedangkan untuk melihat data yang diterima dapat dilihat pada *python shell raspberry pi*. Langkah penggunaan dan pengujian diurutkan sebagai berikut:

1. Menyalakan arduino dengan laptop dan megoperasikan program arduino sebagai pengirim data tanpa terhubung dengan modul pembelajaran *raspberry pi* gambar 4.35.
2. Masukkan data yang sudah diterima sebelumnya dari *raspberry pi*, upload program penerima data arduino dan buka serial monitor, kemudian sambungkan pin Tx

arduino pi ke Rx *raspberry pi* dan hubungkan ground *raspberry pi* dengan arduino tabel 4.14.

3. Jalankan program pada modul pembelajaran *raspberry pi* dengan menekan tombol keyboard “F5” atau dengan klik “Run >> Run Module” kemudian akan muncul data yang diterima gambar 4.36.

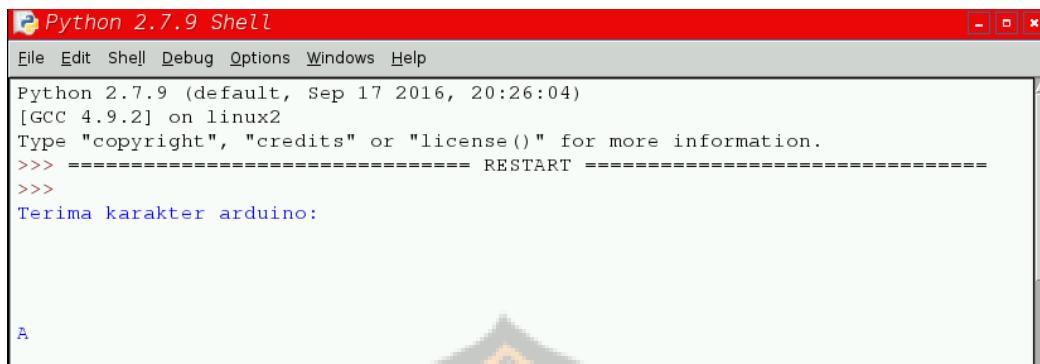


Gambar 4.35. Data karakter yang dikirim ke *raspberry pi*

Pada gambar 4.35 merupakan *listing* program menyalakan dan mengoperasikan program arduino sekaligus untuk pengujian, arduino sebagai pengirim data yang akan diterima oleh *raspberry pi*. Fungsi untuk mengirim data dapat dilihat pada program “char in =’A’ ;” dan “Serial.write(in);” merupakan perintah untuk mengirim data karakter “A” dalam fungsi in kemudian serial akan menulis data karakter “A” yang akan dikirim ke *raspberry pi*.

Tabel 4.14. Rangkaian komunikasi UART arduino ke *raspberry pi*

Modul Pembelajaran <i>Raspberry pi</i>	Arduino
GPIO 15 (Rx)	Tx
Ground	Ground

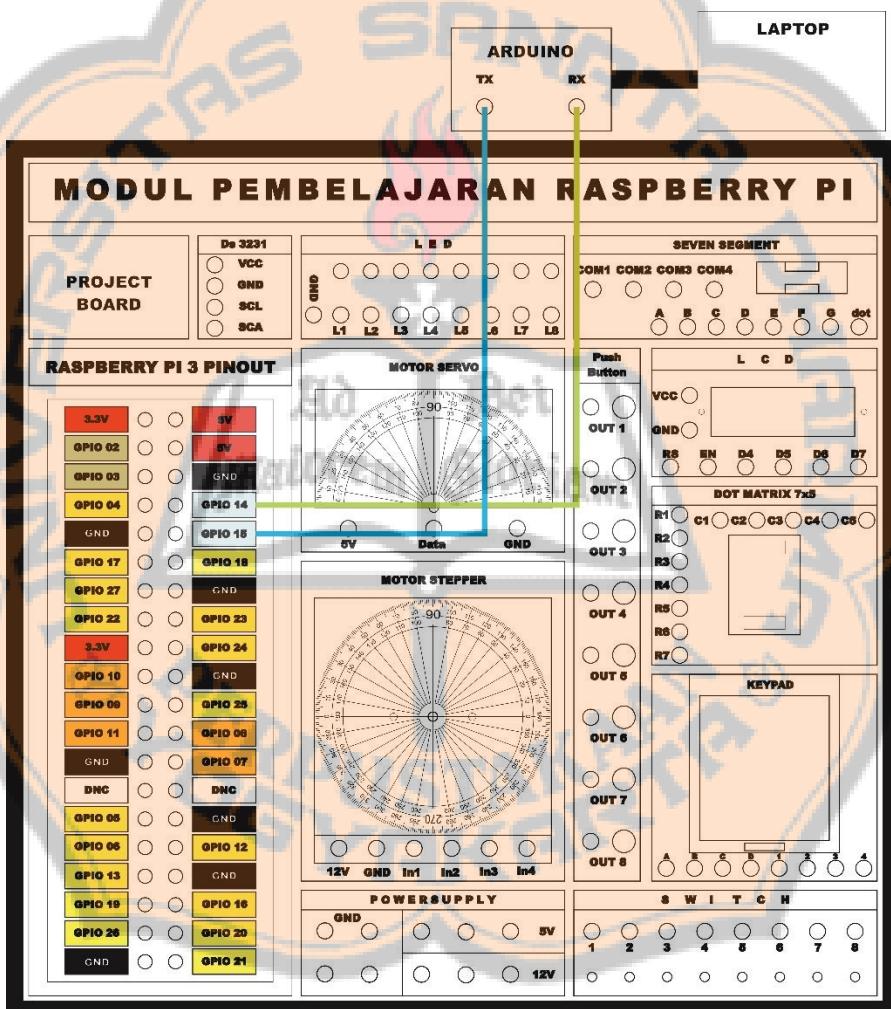


```

Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Terima karakter arduino:
A

```

Gambar 4.36. Data karakter yang diterima dari arduino



Gambar 4.37. Hasil pengujian komunikasi UART

Dari langkah-langkah penggunaan komunikasi UART sekaligus merupakan hasil pengujian komunikasi antara *raspberry pi* dengan arduino seperti pada gambar 4.37. Data awal yang dikirim sama dengan data yang diterima, maka dari pengujian ini dapat disimpulkan bahwa komunikasi berhasil bekerja dengan baik.

4.4. Pembahasan Fungsi Program Modul Pembelajaran *Raspberry pi*

Modul pembelajaran *raspberry pi* memiliki fungsi-fungsi yang digunakan pada program *python*. Pada pembahasan ini bertujuan supaya pengguna mengerti dan paham terhadap fungsi yang digunakan pada modul pembelajaran *raspberry pi* ini.

Tabel 4.15. Keterangan fungsi-fungsi *raspberry pi*

Program <i>python</i>	Fungsi
Import Rpi.GPIO as Gpio	Memanggil fungsi mengaktifkan pin Rpi.GPIO sebagai gpio.
Gpio.setwarnings (False)	Mengatur menipulasi kesalahan pada program <i>raspberry pi</i> .
Gpio.setmode (Gpio.BOARD/BCM)	Mengatur pin yang digunakan pada <i>raspberry pi</i> . BCM sebagai GPIO atau BOARD yang mengacu pada nomer pin.
Import time	Menanggil fungsi waktu sebagai delay yang akan digunakan para program.
Gpio.setup(13,Gpio.OUT)	Mengkondisikan pin 13 sebagai <i>output</i> (keluaran).
Gpio.setup(13,Gpio.IN)	Mengkondisikan pin 13 sebagai <i>input</i> (masukan).
While True:	Fungsi program untuk pengulangan.
If	Fungsi program untuk mengkondisikan
(13,Gpio.HIGH/LOW)	Mengkondisikan pin 13 diberi logika tegangan. Jika <i>high</i> tegangan 5 volt atau <i>low</i> 0 volt.
import sys	Pemanggilan fungsi sistem pada <i>raspberry pi</i>
Import segmen	Pemanggilan fungsi pada segmen.
Import serial	Pemanggilan fungsi pada serial.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah melakukan perancangan, pembuatan, dan pengujian alat modul pembelajaran *raspberry pi* dapat diambil kesimpulan fitur-fitur pada modul pembelajaran berfungsi dengan baik. Fitur-fitur pada modul pembelajaran sebagai berikut :

1. Pengujian *pinout raspberry pi* melalui *eight channel LLC* berfungsi dengan baik.
2. Pengujian *push button* dan LED dapat berfungsi dengan baik.
3. Pengujian saklar *toggle* dan LED dapat berfungsi dengan baik.
4. Pengujian *keypad* dan LCD dapat berfungsi dengan baik.
5. Pengujian *keypad* dan dotmatrix dapat berfungsi dengan baik.
6. Pengujian motor *servo* dapat berfungsi dengan baik.
7. Pengujian motor *stepper* dapat berfungsi dengan baik.
8. Pengujian komunikasi *I2c* dan *seven segment* dapat berfungsi dengan baik.
9. Pengujian komunikasi *UART* dan arduino dapat berfungsi dengan baik.
10. *Eight channel LLC* bekerja dengan baik sebagai pengubah tegangan logika *raspberry pi* dari *3,3 volt* menjadi *5 volt*.
11. Pin komunikasi seperti *I2c* dan *UART* tidak dapat bekerja dengan baik jika melalui *eight channel LLC*.

5.2. Saran

Berdasarkan hasil yang diperoleh, untuk pengembangan modul pembelajaran *raspberry pi* pada penggunaan motor *servo* bisa ditambahkan dengan *driver motor servo* agar perpindahan motor *servo* menunjukkan sudut derajat lebih akurat. Komunikasi dapat dikembangkan dengan menggunakan komunikasi *SPI* modul pembelajaran *raspberry pi* untuk memaksimalkan penggunaan modul pembelajaran. Keterangan pin pada modul pembelajaran *raspberry pi* hanya menggunakan keterangan pin *BCM*, bisa ditambahkan dengan keterangan pin *BOARD* untuk mengetahui penomeran pin *BOARD* saat digunakan.

DAFTAR PUSTAKA

- [1] ----, 2016, *Raspberry pi 3 Specs, Benchmarks, and More*, <https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>, diakses 23 September 2016.
- [2] Herry Lestari, 2015, *LED (Light Emitting Dioda)*, <http://duniafree.besaba.com/elektronika/led-light-emitting-dioda/>, diakses 23 September 2016.
- [3] ----, 2012, *LCD (Liquid Crystal Display)*, <http://elektronika-dasar.web.id/lcd-liquid-crystal-display/>, diakses 23 September 2016.
- [4] ----, 2016, *16x2 LCD Display Interface with Arduino*, <http://circuits4you.com/2016/05/15/arduino-lcd-display/>, diakses 6 Oktober 2016.
- [5] RD ENGINEER, 2008, *Datasheet Specification of LCD Module*, <https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf>, diakses diakses 6 Oktober 2016.
- [6] Nono Haryono, 2015, *Saklar Tekan / Push button*, <http://www.otowater.com/2009/12/saklar-tekan-push-button.html>, diakses diakses 6 Oktober 2016.
- [7] *SPST Normally Open Push button Switch – Short*, <https://www.amazon.com/SPST-NORMALLY-OPEN-BUTTON-SWITCH/dp/B006WRZ6GI>, diakses diakses 6 Oktober 2016.
- [8] Dickson Kho, 2015, *Pengertian Saklar Listrik dan Cara Kerjanya*, <http://teknikelektronika.com/pengertian-saklar-listrik-cara-kerjanya/>, diakses 6 Oktober 2016.
- [9] Nur Shiyaam, 2014, *Industrial Control Device*, <https://nurshiyaam.wordpress.com/2014/04/19/industrial-control-device/>, diakses 6 Oktober 2016.
- [10] ----, 2012, *Matrix Keypad 4x4 Untuk Mikrokontroler*, <http://elektronika-dasar.web.id/matrix-keypad-4x4-untuk-mikrokontroler/>, diakses 23 September 2016.

- [11] Jayant, 2016, *4x4 Matrix Keypad Interfacing with 8051 Microcontroller*, <http://circuitdigest.com/microcontroller-projects/keypad-interfacing-with-8051-microcontroller>, diakses 23 September 2016.
- [12] Nyoman Yudi, 2011, *Dot matrix*, <http://www.aisi555.com/2011/08/dot-matrix.html>, diakses 16 November 2016.
- [13] ----, 2011, *5x7 Dot matrix Display*, <https://www.kingbrightusa.com/images/catalog/spec/TA12-11EWA.pdf>, diakses 16 November 2016.
- [14] Dickson Kho, 2015, *Pengertian Seven segment Display*, <http://teknikelektronika.com/pengertian-seven-segment-display-layar-tujuh-semen/>, diakses 19 November 2016.
- [15] ShenZhen, 2009, *Datasheet Segment LED Display*, Wayjun Technology.
- [16] ----, 2014, *Motor Servo*, <http://zonaelektro.net/motor-servo/>, diakses 19 November 2016.
- [17] ----, 2015, *Teori Motor stepper : Jenis dan Prinsip Motor stepper*, <http://zonaelektro.net/motor-stepper/>, diakses 19 November 2016.
- [18] Listiono G, 2016, *BAB II*, eprints.uny.ac.id/30119/3/4.%20BAB%20II.pdf, diakses 9 desember 2016.
- [19] ----, 2014, *Stepper Motor Driver 3- Axis ULN2003 Board Module*, <http://www.vcc2gnd.com/sku/MDULN2003>, diakses 25 November 2016.
- [20] 8 Channel Bi-Directional Logic Level Converter, <http://lapantech.com/8-Channel-Bi-Directional-Logic-Level-Converter-arduino-raspberry-TTL-mikrokontroler-surabaya>, diakses 21 Januari 2017.
- [21] ----, *Datasheet Extream Accurate I2C-Integrated RTC/TCXO/Crystal*, Maxim Integrated.
- [22] RTS & EEPROM Module DS3231 AT24c32, http://www.jogjarobotika.com/rtc-module/902-rtc-eeprom-module-ds3231-at24c32.html?search_query=ds+3231&results=5, diakses 25 November 2016.
- [23] James McDuffie, 2014, *DS3231 RTS Control and Theory with the Bus Pirate*, <http://notes.pitfall.org/ds3231-rtc-control-and-theory-with-the-bus-pirate.html>, diakses 20 November 2016.

- [24] Bayati, Chairinisa Napitupulu, Khairina Ulfa Nst, 2014, *Resistor Pull Up dan Pull Down*, <http://ilmukomputer.org/wp-content/uploads/2016/02/Irin-ResistorPullUpdanPullDown.pdf>, diakses 30 November 2016.
- [25] ----, *Pull Up and Pull Down Resistors*, https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/, diakses tanggal 30 November 2016
- [26] Hasibuan AS, 2010, *Bab II Landasan Teori*, <http://repository.usu.ac.id/bitstream/123456789/17018/4/Chapter%20II.pdf>, diakses 21 Desember 2016.
- [27] Jimb0, 2009, *Serial Communication*, <https://learn.sparkfun.com/tutorials/serial-communication>, diakses 20 Desember 2016.
- [28] *Basics Of UART Communication*, <http://www.circuitbasics.com/basics-uart-communication/>, diakses 20 Desember 2016.
- [29] Satya Sankar Sahoo, 2016, *Raspberry pi 3 Pinout Model B, RPI2, B+ 40 Pin GPIO Pinout*, <https://www.myelectronicslab.com/tutorial/raspberry-pi-3-gpio-model-b-block-pinout/>, diakses 21 Desember 2016.
- [30] ----, *Datasheet Light Emitting Dioda (LED) Red Diffused 5mm*, NTE ELECTRONICS.
- [31] *8-Channel IIC UART TTL Logic Level Converter 5V/3.3V Bi-Directional Module*, https://www.aliexpress.com/store/product/8-Channel-IIC-UART-SPI-TTL-Logic-Level-Converter-5V-3-3V-Bi-Directional-Module/1602024_32576211190.html, diakses 21 Januari 2017.
- [32] *High-power ULN2003 Stepper Motor Driver Board Test Module For arduino AVR SM*, <https://www.aliexpress.com/item/Free-shipping-Four-phase-five-wire-driver-board-stepper-motor-driver-board-driver-board-ULN2003-drive/32566749004.html?spm=2114.40010508.4.7.mvbG2G>, diakses 21 Januari 2017.

LAMPIRAN



LAMPIRAN I

Listing Program

1. Push Button/Saklar toggle dan LED

```
import RPi.GPIO as gpio
import time
gpio.setwarnings(False)
gpio.setmode (gpio.BOARD)

#Keluaran
gpio.setup(13,gpio.OUT)
gpio.setup(19,gpio.OUT)
gpio.setup(21,gpio.OUT)
gpio.setup(29,gpio.OUT)
gpio.setup(31,gpio.OUT)
gpio.setup(33,gpio.OUT)
gpio.setup(35,gpio.OUT)
gpio.setup(37,gpio.OUT)
#Masukan
gpio.setup(12,gpio.IN)
gpio.setup(16,gpio.IN)
gpio.setup(18,gpio.IN)
gpio.setup(22,gpio.IN)
gpio.setup(24,gpio.IN)
gpio.setup(26,gpio.IN)
gpio.setup(32,gpio.IN)
gpio.setup(36,gpio.IN)

while True:
    masukan1=gpio.input(12)
    masukan2=gpio.input(16)
    masukan3=gpio.input(18)
    masukan4=gpio.input(22)
    masukan5=gpio.input(24)
    masukan6=gpio.input(26)
    masukan7=gpio.input(32)
    masukan8=gpio.input(36)
    if masukan1==False:
        print "LED1_ON"
        gpio.output(13,gpio.HIGH)
    if masukan2==False:
        print "LED2_ON"
        gpio.output(19,gpio.HIGH)
    if masukan3==False:
        print "LED3_ON"
        gpio.output(21,gpio.HIGH)
    if masukan4==False:
        print "LED4_ON"
        gpio.output(29,gpio.HIGH)
    if masukan5==False:
        print "LED5_ON"
```

```
    gpio.output(31,gpio.HIGH)
if masukan6==False:
    print "LED6_ON"
    gpio.output(33,gpio.HIGH)
if masukan7==False:
    print "LED7_ON"
    gpio.output(35,gpio.HIGH)
if masukan8==False:
    print "LED8_ON"
    gpio.output(37,gpio.HIGH)

if masukan1==True:
    gpio.output(13,gpio.LOW)
if masukan2==True:
    gpio.output(19,gpio.LOW)
if masukan3==True:
    gpio.output(21,gpio.LOW)
if masukan4==True:
    gpio.output(29,gpio.LOW)
if masukan5==True:
    gpio.output(31,gpio.LOW)
if masukan6==True:
    gpio.output(33,gpio.LOW)
if masukan7==True:
    gpio.output(35,gpio.LOW)
if masukan8==True:
    gpio.output(37,gpio.LOW)

gpio.cleanup()
```

2. Keypad dan LCD

```
import RPi.GPIO as gpio
import Adafruit_CharLCD as LCD
import time

gpio.setmode(gpio.BCM)
gpio.setwarnings(False)

row =2
col =16
rs =4
e =17
d4 =27
d5 =22
d6 =10
d7 =9

lcd = LCD.Adafruit_CharLCD(rs,e,d4,d5,d6,d7,col,row)
lcd.home()

MATRIX = [[1,2,3,'A'],
           [4,5,6,'B'],
```

```
[7,8,9,'C'],
['*',0,'#','D']]
```

```
ROW = [18,23,24,25]
COL = [8,7,12,16]
```

```
for j in range(4):
    gpio.setup(COL[j], gpio.OUT)
    gpio.output(COL[j], 1)
```

```
for i in range(4):
    gpio.setup(ROW[i], gpio.IN )
```

```
bar=4
kol=4
c=-1
b=0
```

```
try:
    while(True):
        if c>15 and b==1:
            lcd.clear()
            c=-1
            b=0
        if c>14 and b==0:
            c=-1
            b=1

        if bar==0 and kol==0:
            c=c+1
            lcd.set_cursor(c,b)
            lcd.message('1')
            time.sleep(0.250)

        elif bar==0 and kol==1:
            c=c+1
            lcd.set_cursor(c,b)
            lcd.message('2')
            time.sleep(0.250)

        elif bar==0 and kol==2:
            c=c+1
            lcd.set_cursor(c,b)
            lcd.message('3')
            time.sleep(0.250)

        elif bar==1 and kol==0:
            c=c+1
            lcd.set_cursor(c,b)
            lcd.message('4')
            time.sleep(0.250)

        elif bar==1 and kol==1:
            c=c+1
            lcd.set_cursor(c,b)
```

```
lcd.message('5')
time.sleep(0.250)

elif bar==1 and kol==2:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('6')
    time.sleep(0.250)

elif bar==2 and kol==0:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('7')
    time.sleep(0.250)

elif bar==2 and kol==1:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('8')
    time.sleep(0.250)

elif bar==2 and kol==2:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('9')
    time.sleep(0.250)

elif bar==3 and kol==0:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('*')
    lcd.clear()
    time.sleep(0.250)

elif bar==3 and kol==1:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('0')
    time.sleep(0.250)

elif bar==3 and kol==2:
    c=c+1
    lcd.set_cursor(c,b)
    lcd.message('#')
    time.sleep(0.250)

bar =4
kol =4

for j in range(4):
    gpio.output(COL[j],0)
    for i in range(4):
        if gpio.input(ROW[i]) == 0:
            print MATRIX[i][j]
            bar = i
            kol = j
```

```

        time.sleep(0.2)
    while(gpio.input(ROW[i]) == 0):
        pass
    gpio.output(COL[j],1)

except KeyboardInterrupt:
    gpio.cleanup()

```

3. Keypad dan dot matrix LED

```

import RPi.GPIO as gpio
import time
t=0.001
gpio.setmode(gpio.BOARD)
gpio.setwarnings(False)

#ROW
gpio.setup(7,gpio.OUT)
gpio.setup(11,gpio.OUT)
gpio.setup(13,gpio.OUT)
gpio.setup(15,gpio.OUT)
gpio.setup(19,gpio.OUT)
gpio.setup(21,gpio.OUT)
gpio.setup(40,gpio.OUT)
#colom
gpio.setup(29,gpio.OUT)
gpio.setup(31,gpio.OUT)
gpio.setup(33,gpio.OUT)
gpio.setup(35,gpio.OUT)
gpio.setup(38,gpio.OUT)

def A():
#ROWS1A
    gpio.output (7, gpio.HIGH)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom1A
    gpio.output (29, gpio.LOW)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS2A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.HIGH)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)

```

```
gpio.output (19, gpio.LOW)
gpio.output (21, gpio.LOW)
gpio.output (40, gpio.LOW)

#Colom2A
    gpio.output (29, gpio.LOW)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS3A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.HIGH)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)

#Colom3A
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS4A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.HIGH)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)

#Colom4A
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS5A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.HIGH)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)

#Colom5A
    gpio.output (29, gpio.HIGH)
```

```
gpio.output (31, gpio.LOW)
gpio.output (33, gpio.LOW)
gpio.output (35, gpio.LOW)
gpio.output (38, gpio.HIGH)
time.sleep(t)

#ROWS6A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.HIGH)
    gpio.output (40, gpio.LOW)

#Colom6A
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS7A
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.HIGH)
#Colom7A
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

def B():
#ROWS1B
    gpio.output (7, gpio.HIGH)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom1B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)
```

```
#ROWS2B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.HIGH)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom2B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS3B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.HIGH)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom3B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS4B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.HIGH)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom4B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS5B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.HIGH)
```

```
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom5B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS6B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.HIGH)
    gpio.output (40, gpio.LOW)
#Colom6B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS7B
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.HIGH)
#Colom7B
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

def C():
    #ROWS1C
    gpio.output (7, gpio.HIGH)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom1C
    gpio.output (29, gpio.LOW)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
```

```
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)
#ROWS2C
    gpio.output (7,  gpio.LOW)
    gpio.output (11, gpio.HIGH)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom2C
    gpio.output (29,  gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS3C
    gpio.output (7,  gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.HIGH)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom3C
    gpio.output (29,  gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS4C
    gpio.output (7,  gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.HIGH)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom4C
    gpio.output (29,  gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS5C
    gpio.output (7,  gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
```

```
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.HIGH)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom5C
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

#ROWS6C
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.HIGH)
    gpio.output (40, gpio.LOW)

#Colom6C
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS7C
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.HIGH)
#Colom7C
    gpio.output (29, gpio.LOW)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

def D () :
    #ROWS1D
    gpio.output (7, gpio.HIGH)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom1D
```

```
gpio.output (29, gpio.HIGH)
gpio.output (31, gpio.HIGH)
gpio.output (33, gpio.HIGH)
gpio.output (35, gpio.HIGH)
gpio.output (38, gpio.LOW)
time.sleep(t)

#ROWS2D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.HIGH)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom2D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS3D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.HIGH)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom3D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS4D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.HIGH)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)
#Colom4D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)
```

```

#ROWS5D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.HIGH)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.LOW)

#Colom5D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS6D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.HIGH)
    gpio.output (40, gpio.LOW)

#Colom6D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.LOW)
    gpio.output (33, gpio.LOW)
    gpio.output (35, gpio.LOW)
    gpio.output (38, gpio.HIGH)
    time.sleep(t)

#ROWS7D
    gpio.output (7, gpio.LOW)
    gpio.output (11, gpio.LOW)
    gpio.output (13, gpio.LOW)
    gpio.output (15, gpio.LOW)
    gpio.output (19, gpio.LOW)
    gpio.output (21, gpio.LOW)
    gpio.output (40, gpio.HIGH)

#Colom7D
    gpio.output (29, gpio.HIGH)
    gpio.output (31, gpio.HIGH)
    gpio.output (33, gpio.HIGH)
    gpio.output (35, gpio.HIGH)
    gpio.output (38, gpio.LOW)
    time.sleep(t)

MATRIX = [[1,2,3,'A'],
          [4,5,6,'B'],
          [7,8,9,'C'],
          ['*',0,'#','D']]
```

```

ROW = [12,16,18,22]
COL = [24,26,32,36]

for j in range(4):
    gpio.setup(COL[j], gpio.OUT)
    gpio.output(COL[j], 1)

for i in range(4):
    gpio.setup(ROW[i], gpio.IN)

bar =0
kol =0

try:
    while(True):
        if bar==0 and kol==3:
            A()
        elif bar==1 and kol==3:
            B()
        elif bar==2 and kol==3:
            C()
        elif bar==3 and kol==3:
            D()
        for j in range(4):
            gpio.output(COL[j],0)
            for i in range(4):
                if gpio.input(ROW[i]) == 0:
                    print MATRIX[i][j]
                    bar = i
                    kol = j
                    time.sleep(0.2)
                while(gpio.input(ROW[i]) == 0):
                    pass
                gpio.output(COL[j],1)

except KeyboardInterrupt:
    gpio.cleanup()

```

4. Motor servo

```

import RPi.GPIO as gpio
gpio.setmode (gpio.BORD)
gpio.setwarnings(False)

gpio.setup(11,gpio.OUT)
pwm=gpio.PWM(11,50)

pwm.start(0)
for i in range (0,100):
    desiredPosition=input("kemana servo? 0-180 derajat : ")
    DC=1./18.* (desiredPosition)+2
    pwm.ChangeDutyCycle(DC)

```

```
pwm.stop()  
gpio.cleanup()
```

5. Motor stepper

```
import RPi.GPIO as gpio  
from time import sleep
```

```
a=31  
b=33  
c=35  
d=37  
x=8  
y=23  
  
gpio.setmode(gpio.BOARD)  
gpio.setwarnings(False)  
  
gpio.setup(x,gpio.IN)  
gpio.setup(y,gpio.IN)  
gpio.setup(a,gpio.OUT)  
gpio.setup(b,gpio.OUT)  
gpio.setup(c,gpio.OUT)  
gpio.setup(d,gpio.OUT)  
t=0.5
```

```
def putarl():  
    print "1"  
    gpio.output(a,True)  
    gpio.output(b,True)  
    gpio.output(c,False)  
    gpio.output(d,False)  
    sleep(t)  
    print "2"  
    gpio.output(a,False)  
    gpio.output(b,True)  
    gpio.output(c,False)  
    gpio.output(d,True)  
    sleep(t)  
    print "3"  
    gpio.output(a,False)  
    gpio.output(b,False)  
    gpio.output(c,True)  
    gpio.output(d,True)  
    sleep(t)  
    print "4"  
    gpio.output(a,True)  
    gpio.output(b,False)  
    gpio.output(c,True)  
    gpio.output(d,False)  
    sleep(t)
```

```
def putarr():  
    print "4"
```

```
gpio.output(a,True)
gpio.output(b,False)
gpio.output(c,True)
gpio.output(d,False)
sleep(t)
print "3"
gpio.output(a,False)
gpio.output(b,False)
gpio.output(c,True)
gpio.output(d,True)
sleep(t)
print "2"
gpio.output(a,False)
gpio.output(b,True)
gpio.output(c,False)
gpio.output(d,True)
sleep(t)
print "1"
gpio.output(a,True)
gpio.output(b,True)
gpio.output(c,False)
gpio.output(d,False)
sleep(t)

posisi=0
for i in range (0,10):
    masukan=input("berapa derajat 0-360 : ")
    hitung=(masukan/7.5)

    print hitung
    k=int(hitung/4)

    for j in range(0,k):
        if masukan>posisi:
            putarl()
        if masukan<posisi:
            putarr()
    posisi=masukan
```

6. Komunikasi i2c modul RTC ds3231 dan seven segment

```
import sys
sys.path.insert(0,'/home/pi')
import segmen
import time
import SDL_DS3231

ds3231 = SDL_DS3231.SDL_DS3231(1, 0x68)
#comment out the next line after the clock has been initialized
ds3231.write_now()
t=0.001

while True:
```

```

bil=ds3231._read_minutes()
satuan = bil % 10
puluhan = bil / 10

bil2=ds3231._read_hours()
satuan2 = bil2 % 10
puluhan2 = bil2 / 10

segmen.t_jam(satuan2,puluhan2,t)
segmen.t_menit(satuan,puluhan,t)

```

7. Komunikasi UART raspberry pi ke arduino

----KIRIM dari raspberry pi ke arduino----

```

import serial
import time
port=
serial.Serial("/dev/ttyAMA0",baudrate=9600,bytesize=8,parity='N',stopbits=1,timeout=3.0)
while True:
#for i in range (100000):
    SEND= raw_input("masukan karakter kirim: ")
    port.write(SEND)
    time.sleep(1)

```

----TERIMA arduino dari raspberry pi----

```

void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial. println("Terima karakter raspi:");
}

void loop()
{
    if (Serial.available())
    {
        char in = (char)Serial.read();
        Serial.println(in);
    }
    delay(100);
}

```

----KIRIM dari arduino ke raspberry pi----

```

void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial. println("Kirim data karakter");
}

```

```
void(*resetFunc) (void)=0;
void loop()

{
    char in = 'A' ;
    Serial.write(in);

    delay(100);
    resetFunc();
}

-----TERIMA arduino dari raspberry pi-----

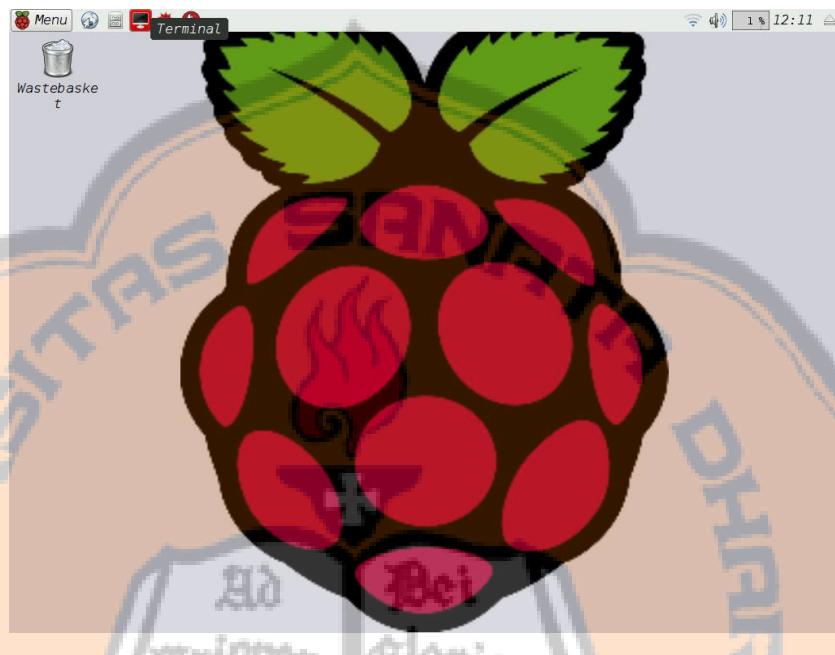
import serial
import time
port=
serial.Serial("/dev/ttyAMA0",baudrate=9600,bytesize=8,parity='N',s
topbits=1,timeout=10.0)
port.close()
time.sleep(1)
port.open()
print "Terima karakter arduino:"

for i in range (100):
    x=port.readline(1)
    print x
```

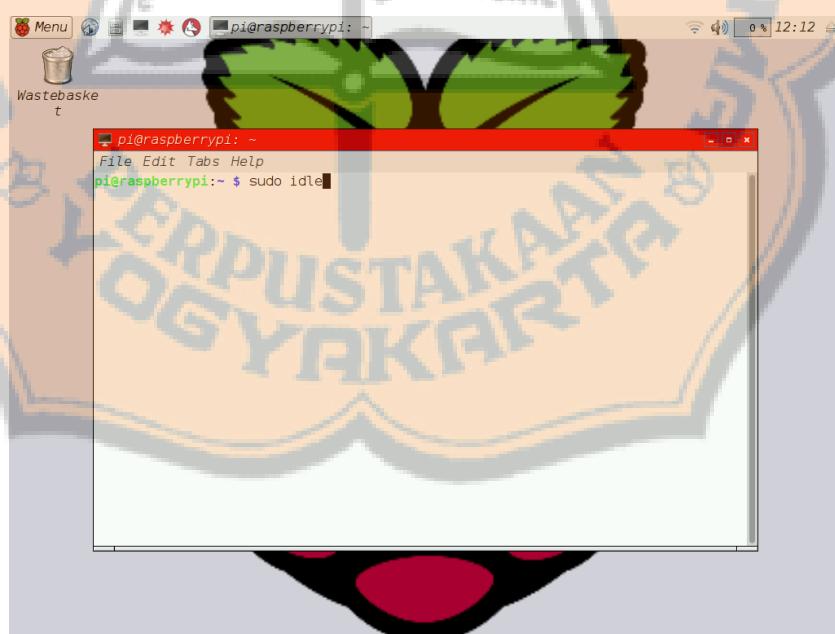
LAMPIRAN II

Pengunaan Modul Pembelajaran Raspberry Pi

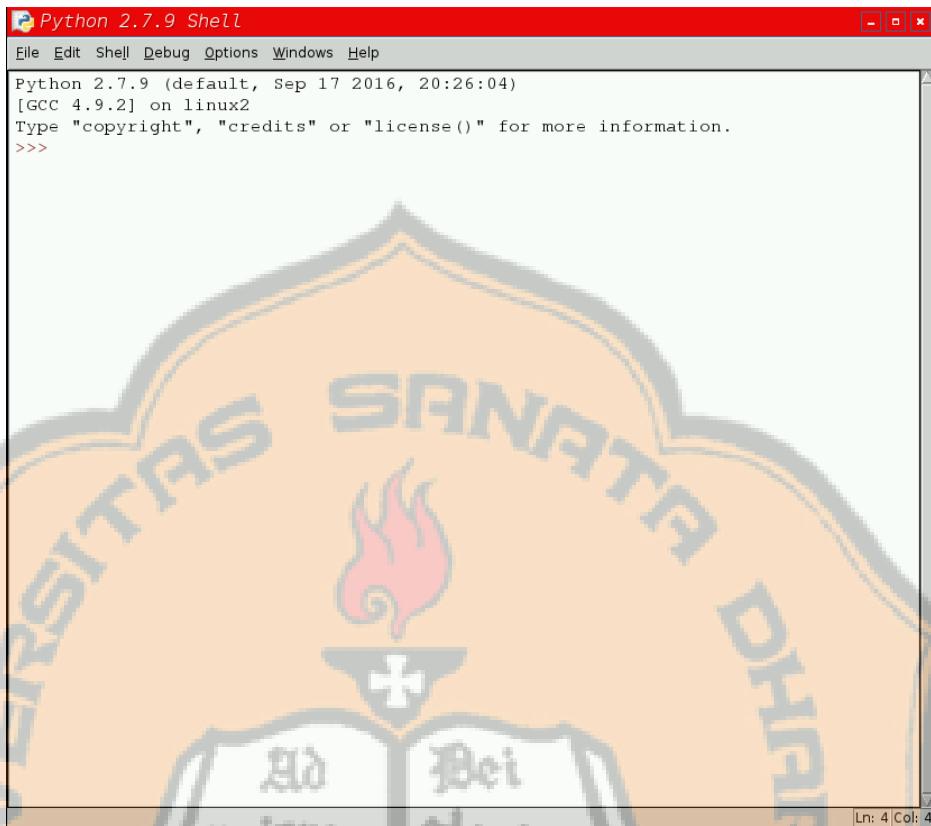
1. Menyalakan raspberry pi dengan menyambungkan pada sumber listrik 220V.
2. Kemudian klik icon terminal.



3. Ketik "sudo idle" kemudian enter.



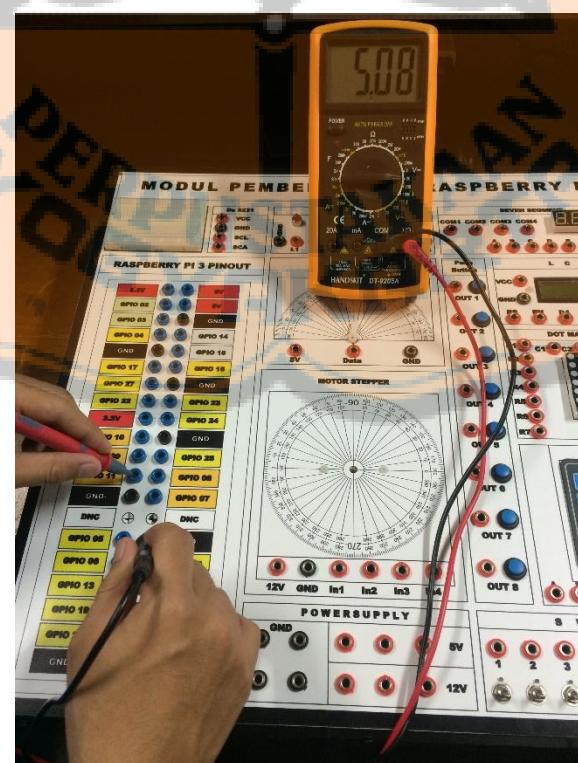
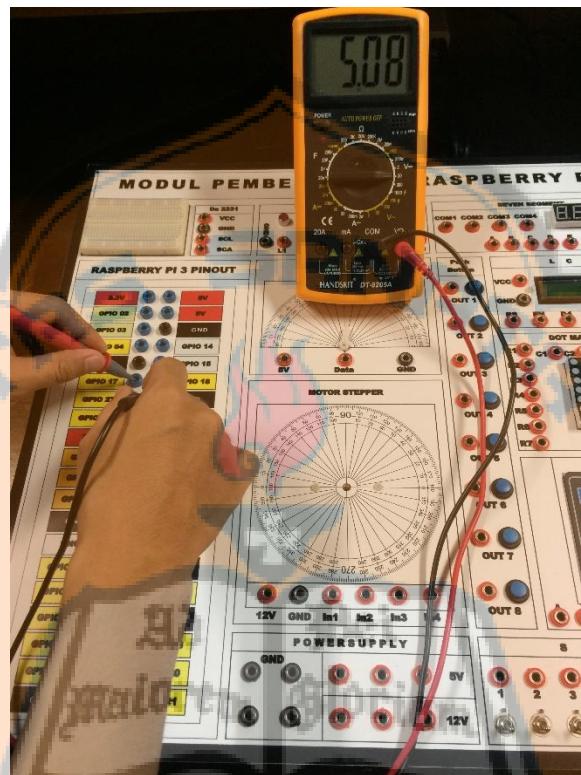
4. Akan muncul jendele seperti berikut, kemudian pilih File >> Open, untuk membuka file program yang akan digunakan.

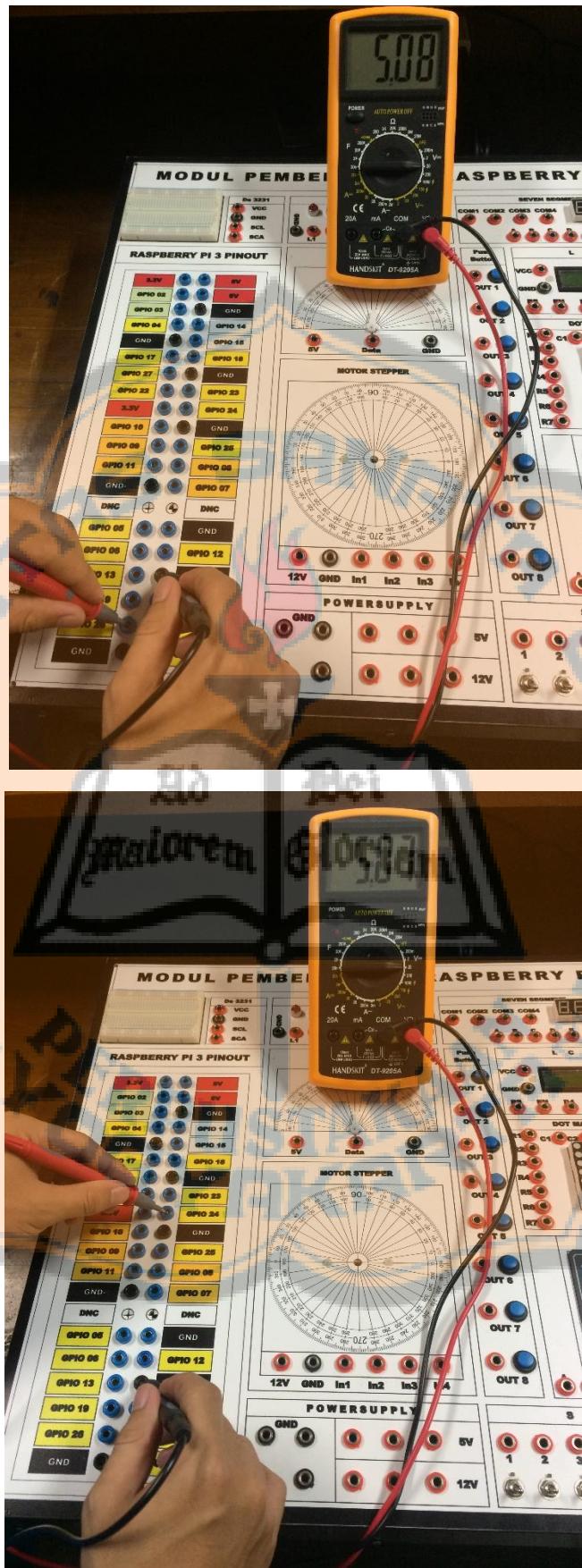


LAMPIRAN III

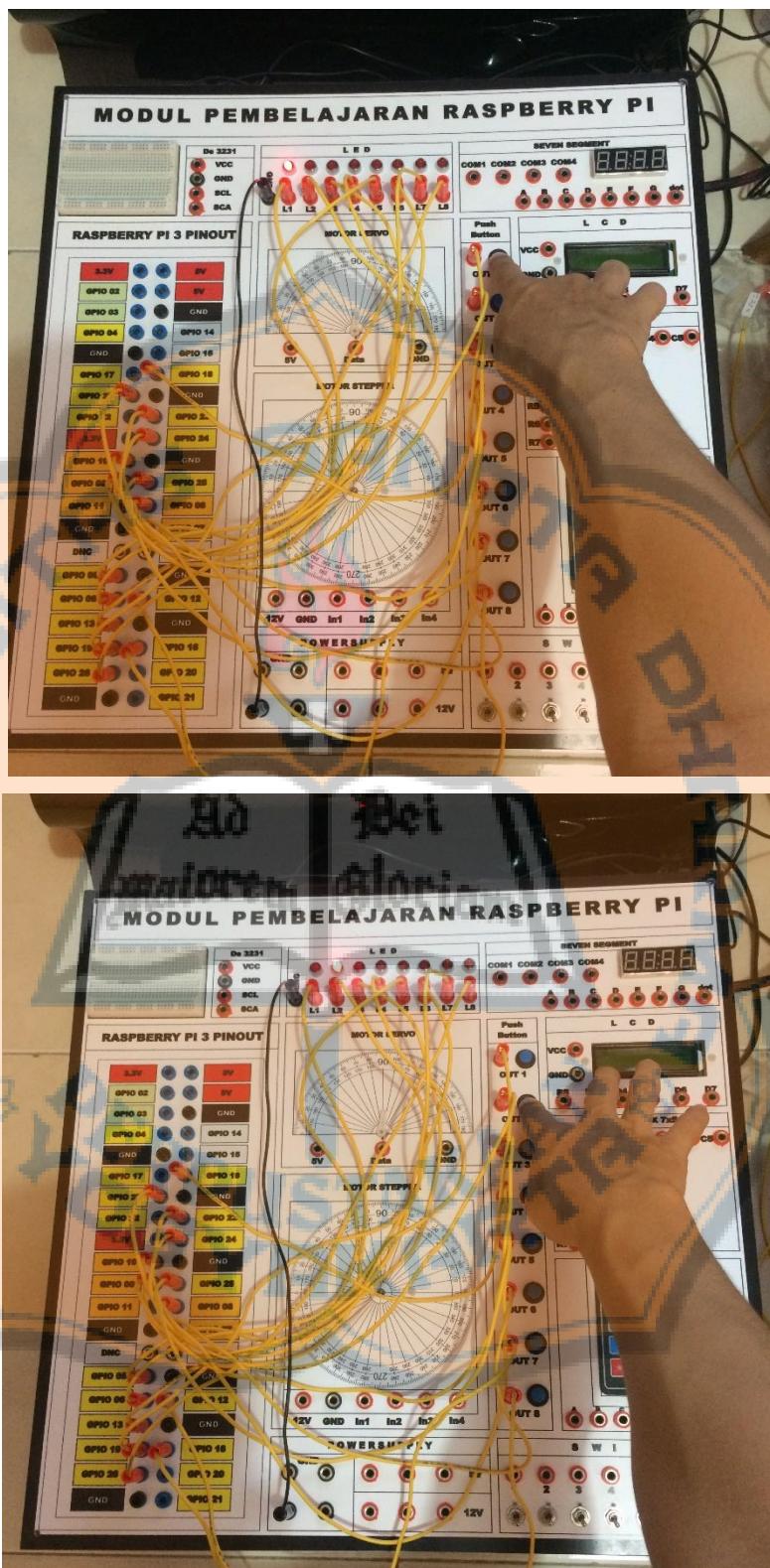
Gambar Pengujian Alat

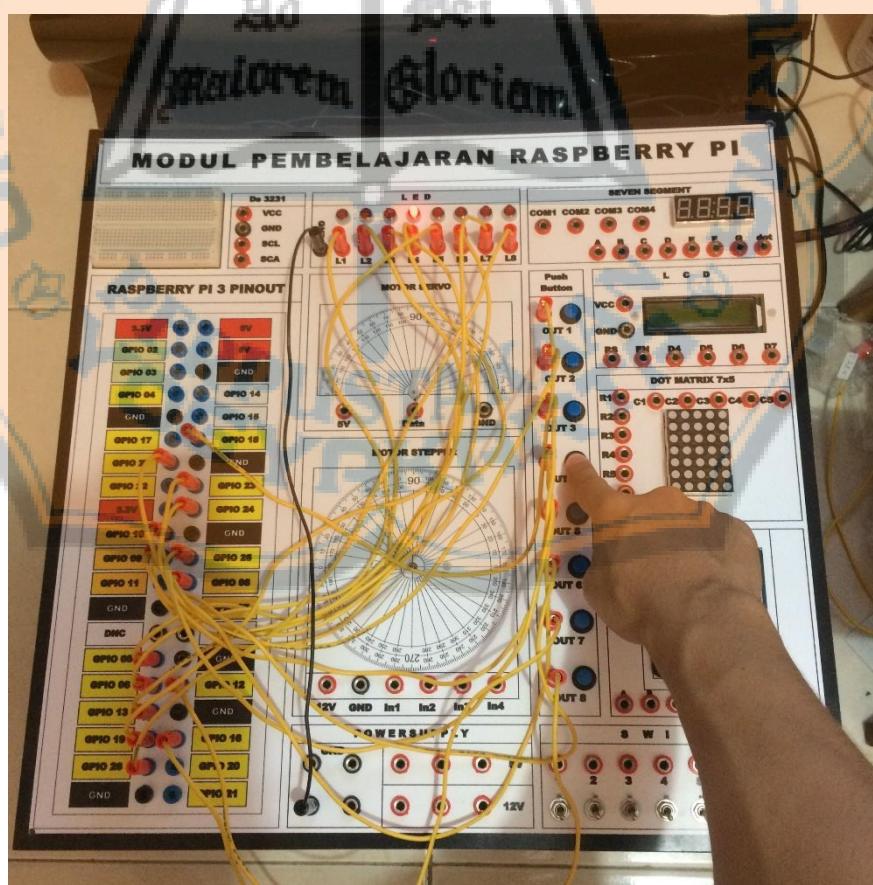
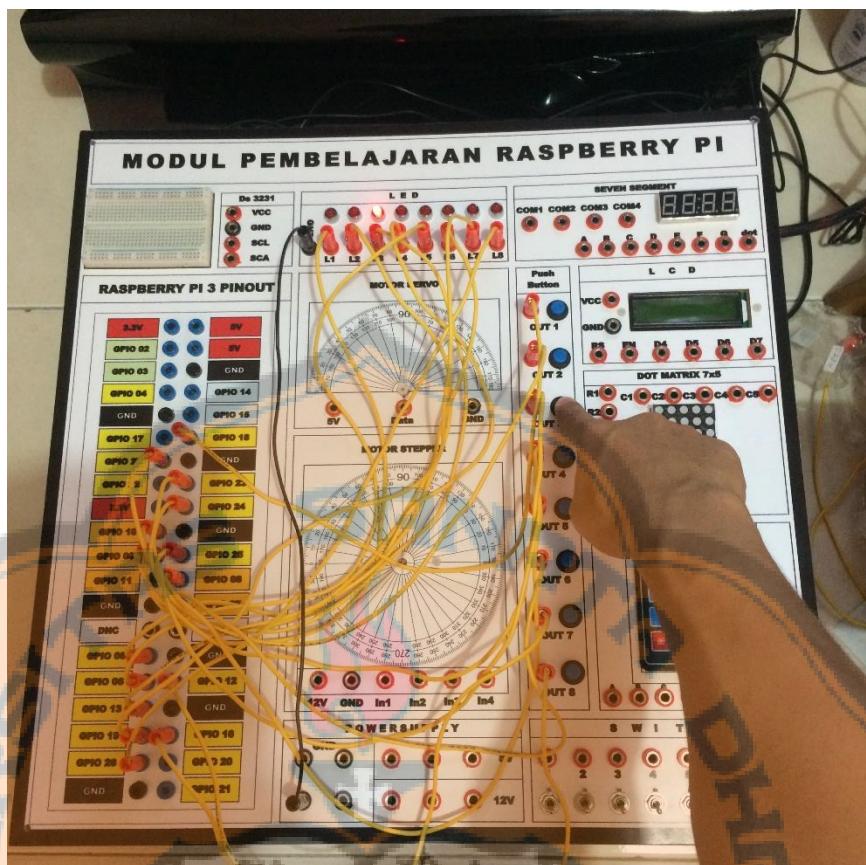
1. Pengujian raspberry pi pinout

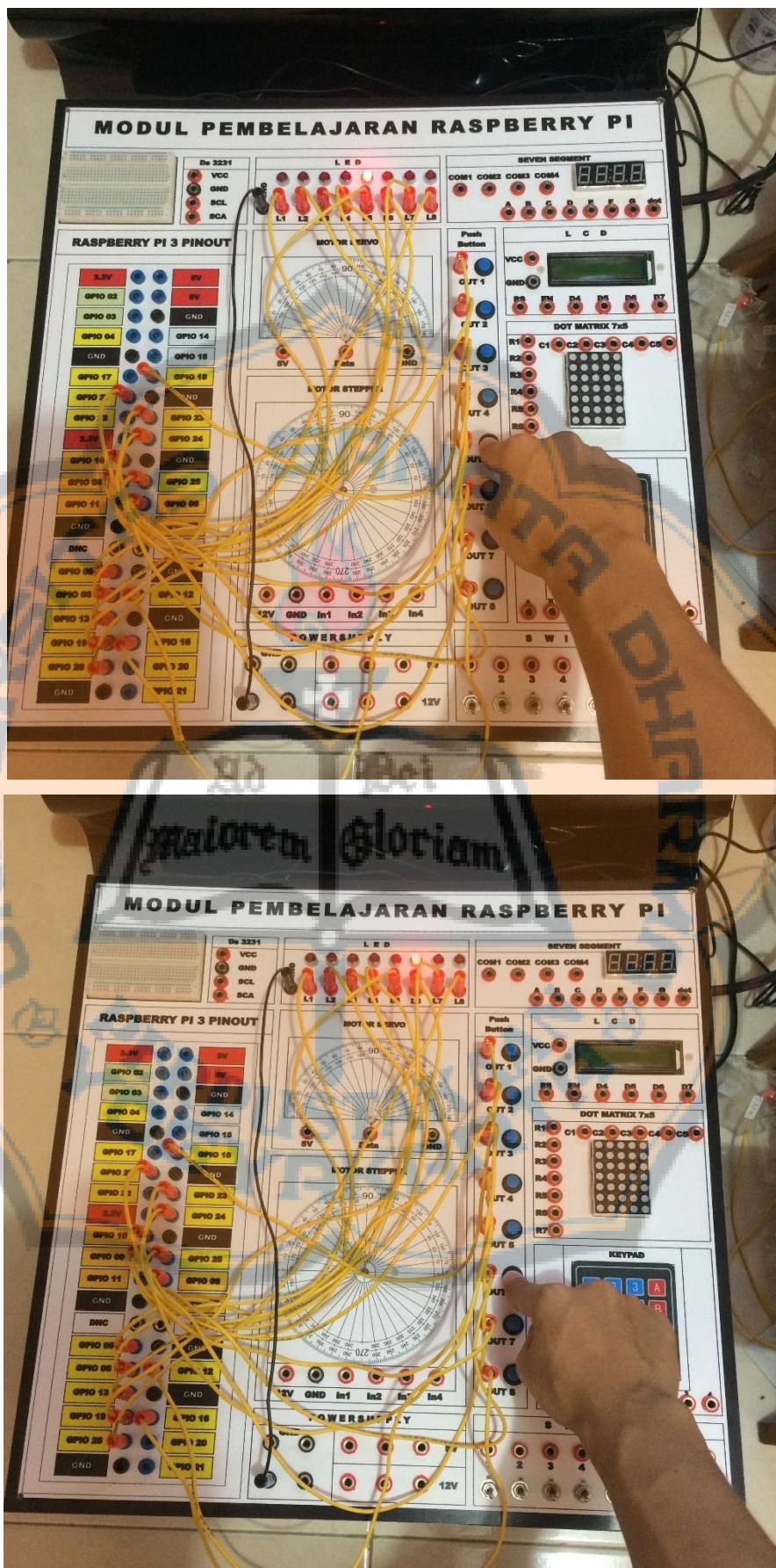


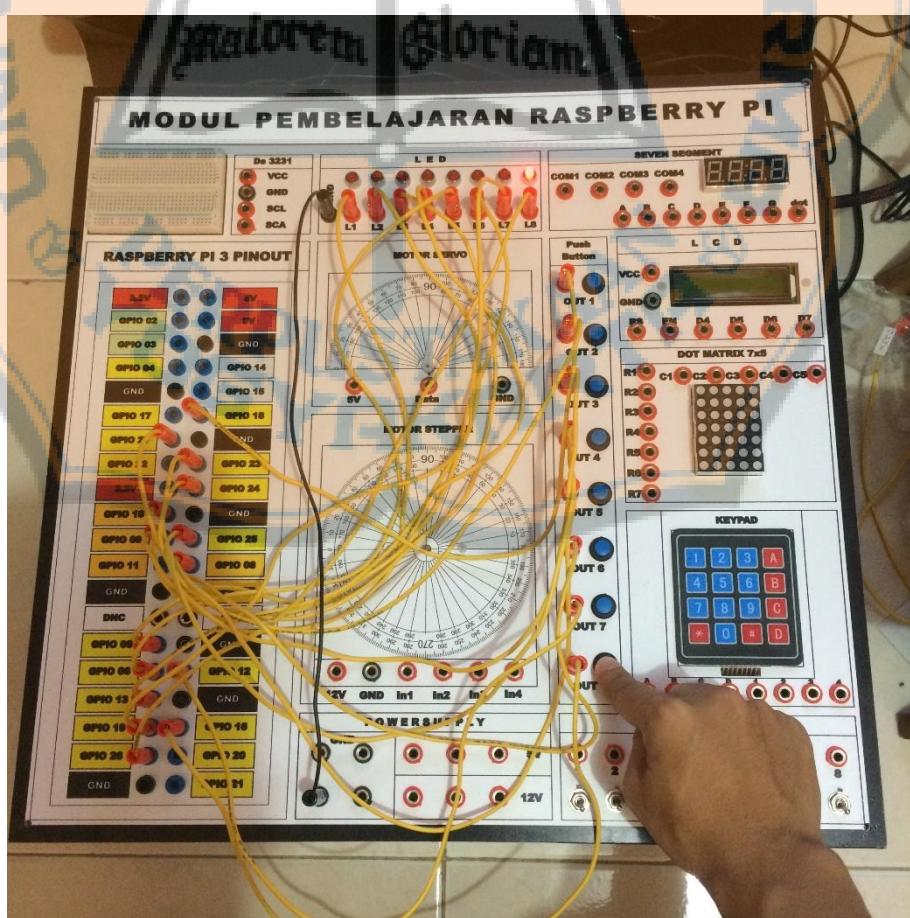
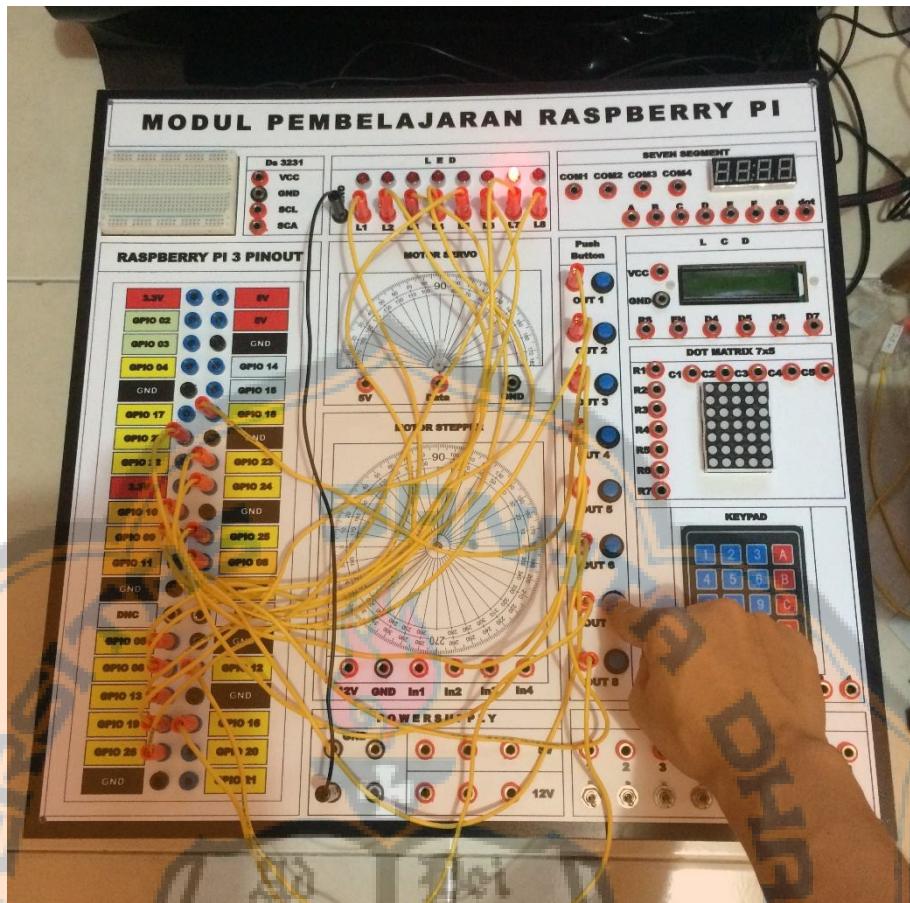


2. Pengujian push button dan LED

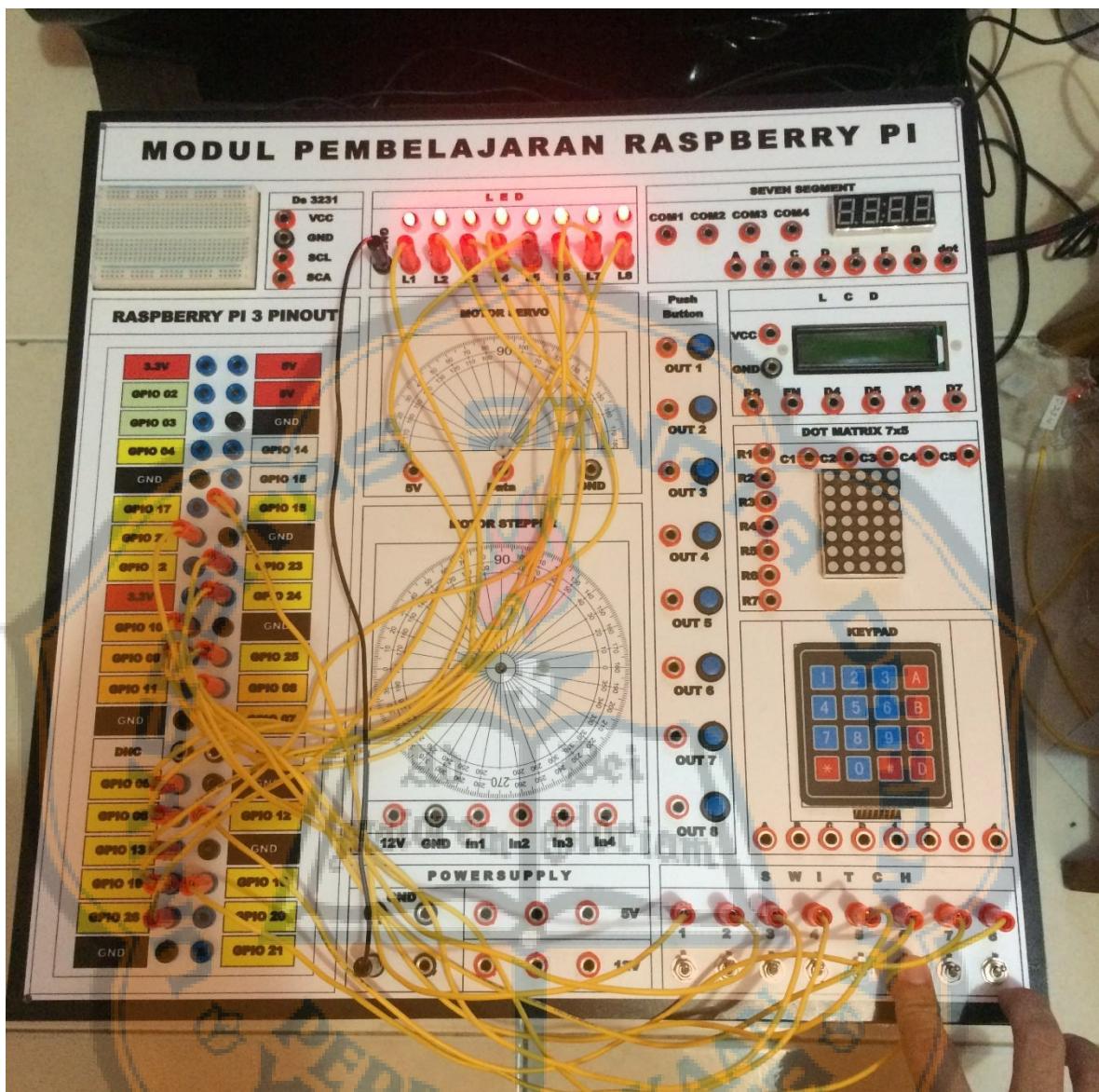




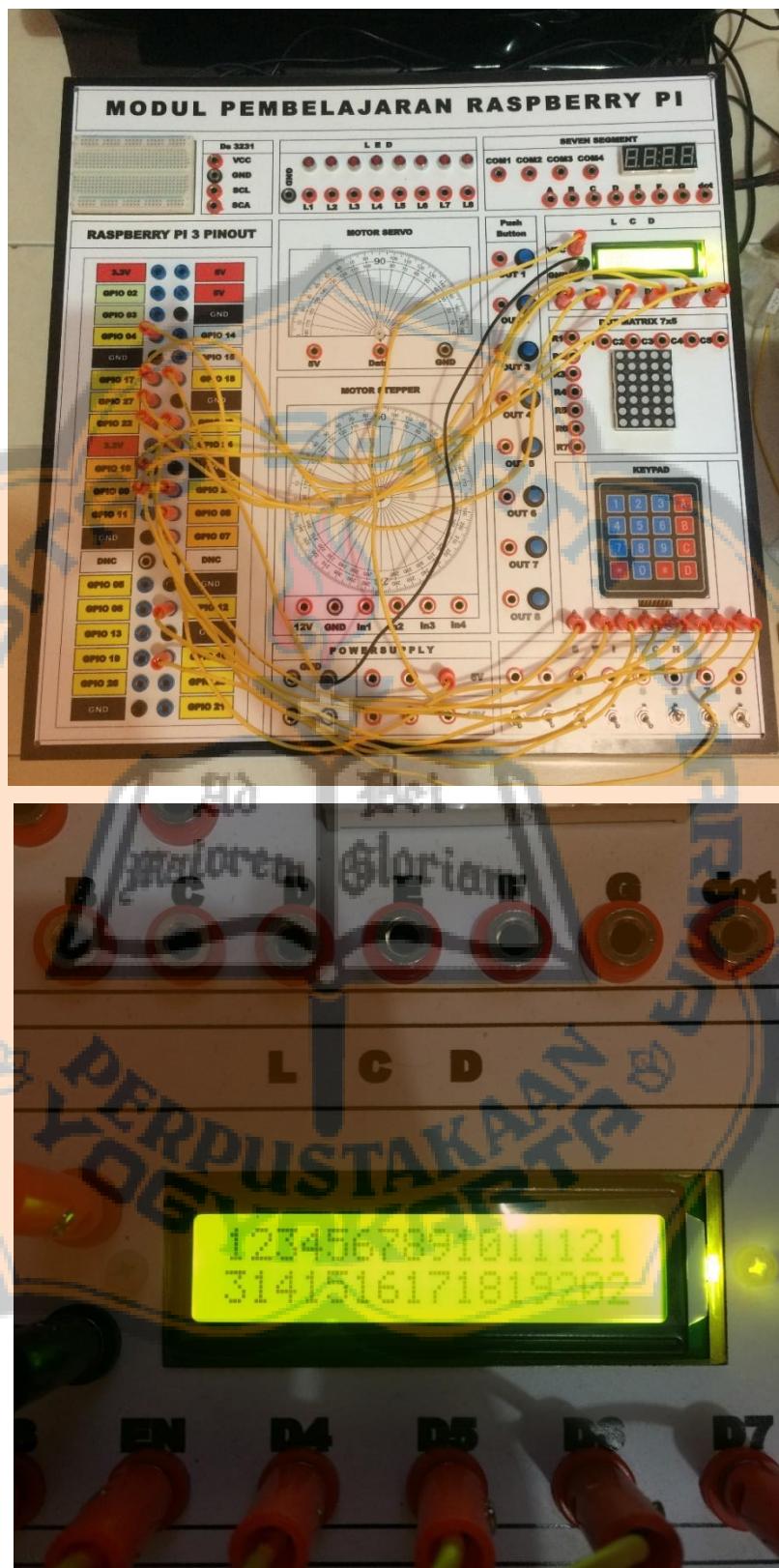




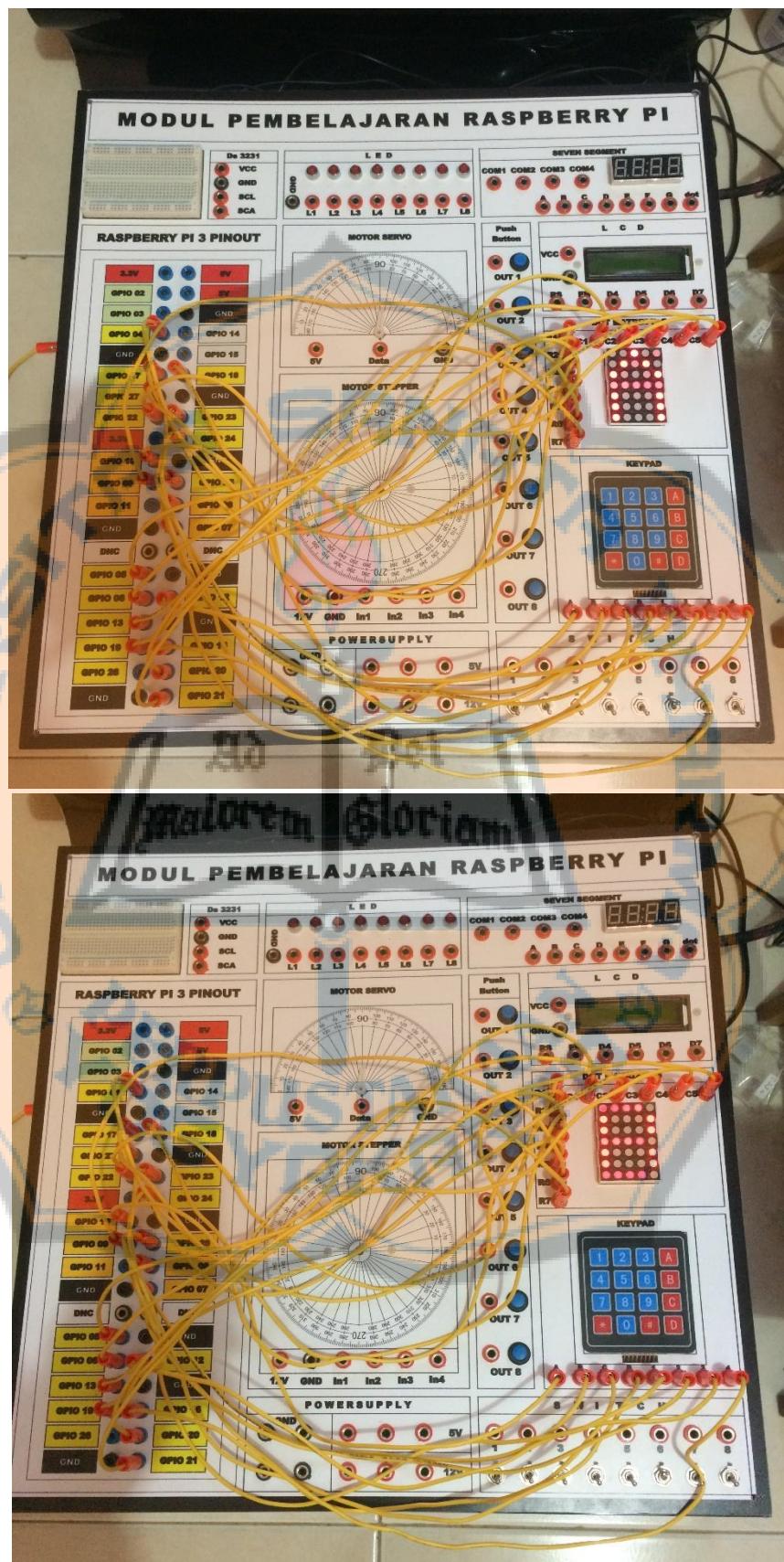
3. Pengujian saklar toggle dan LED

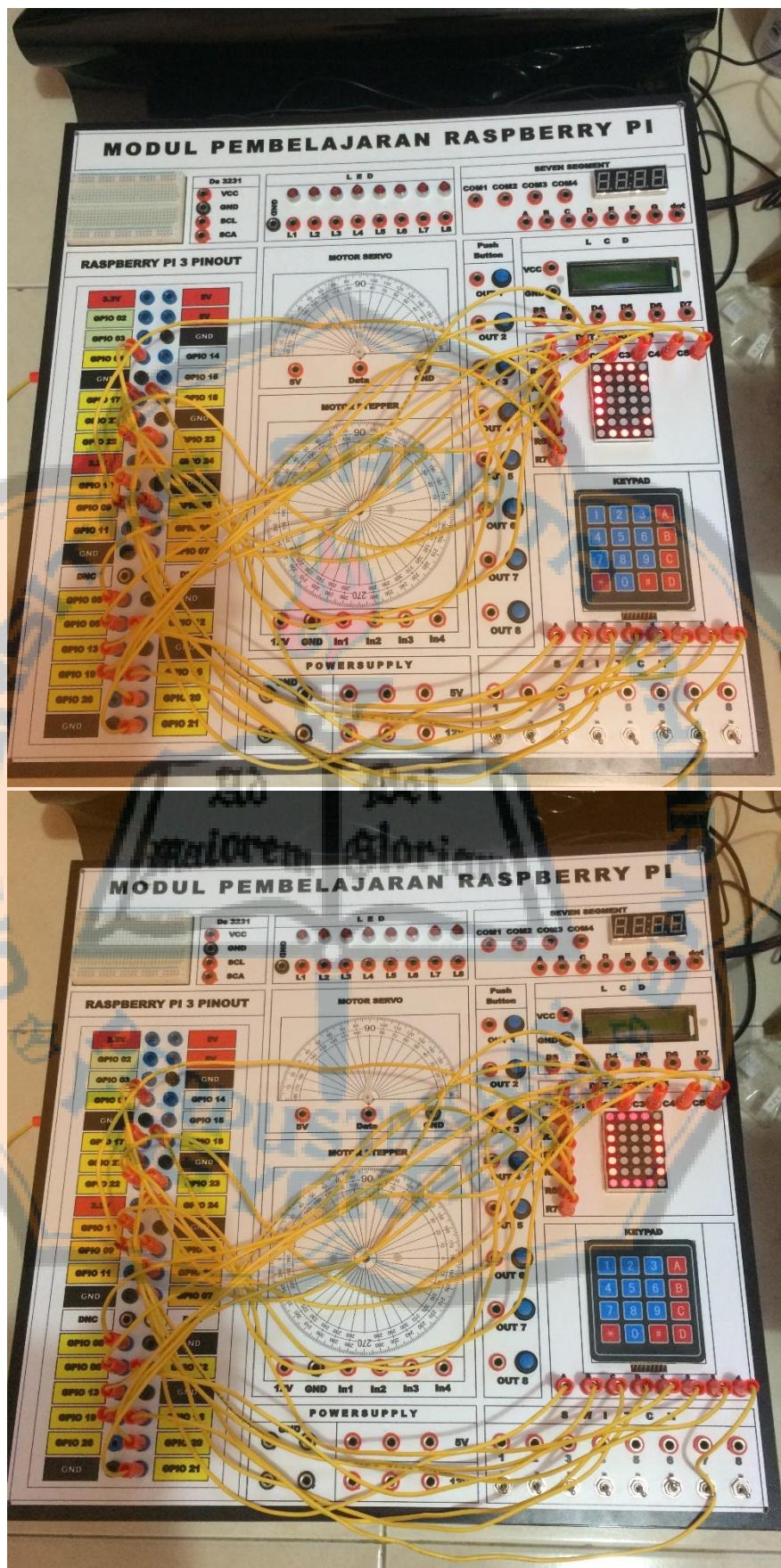


4. Pengujian keypad dan LCD

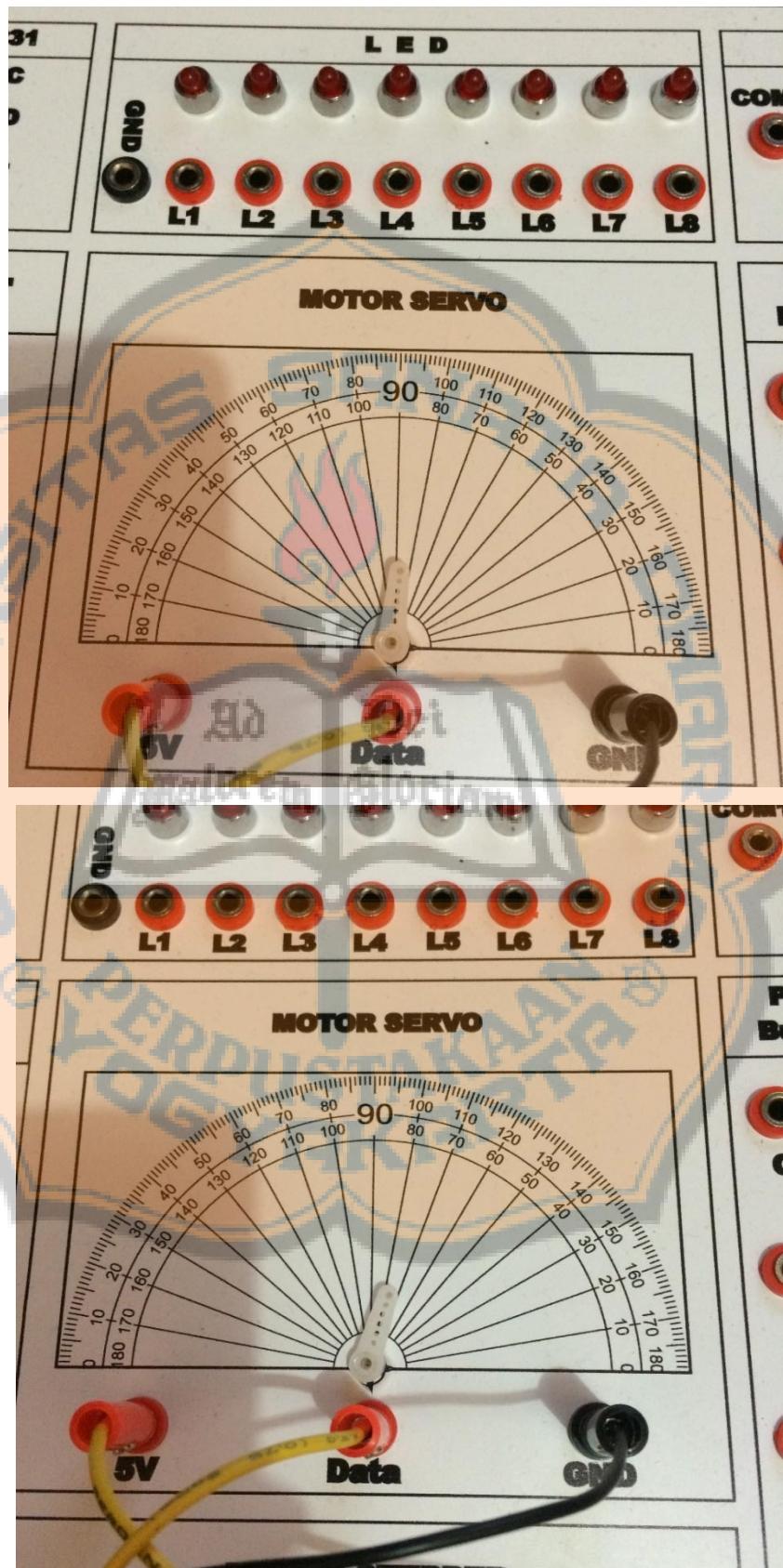


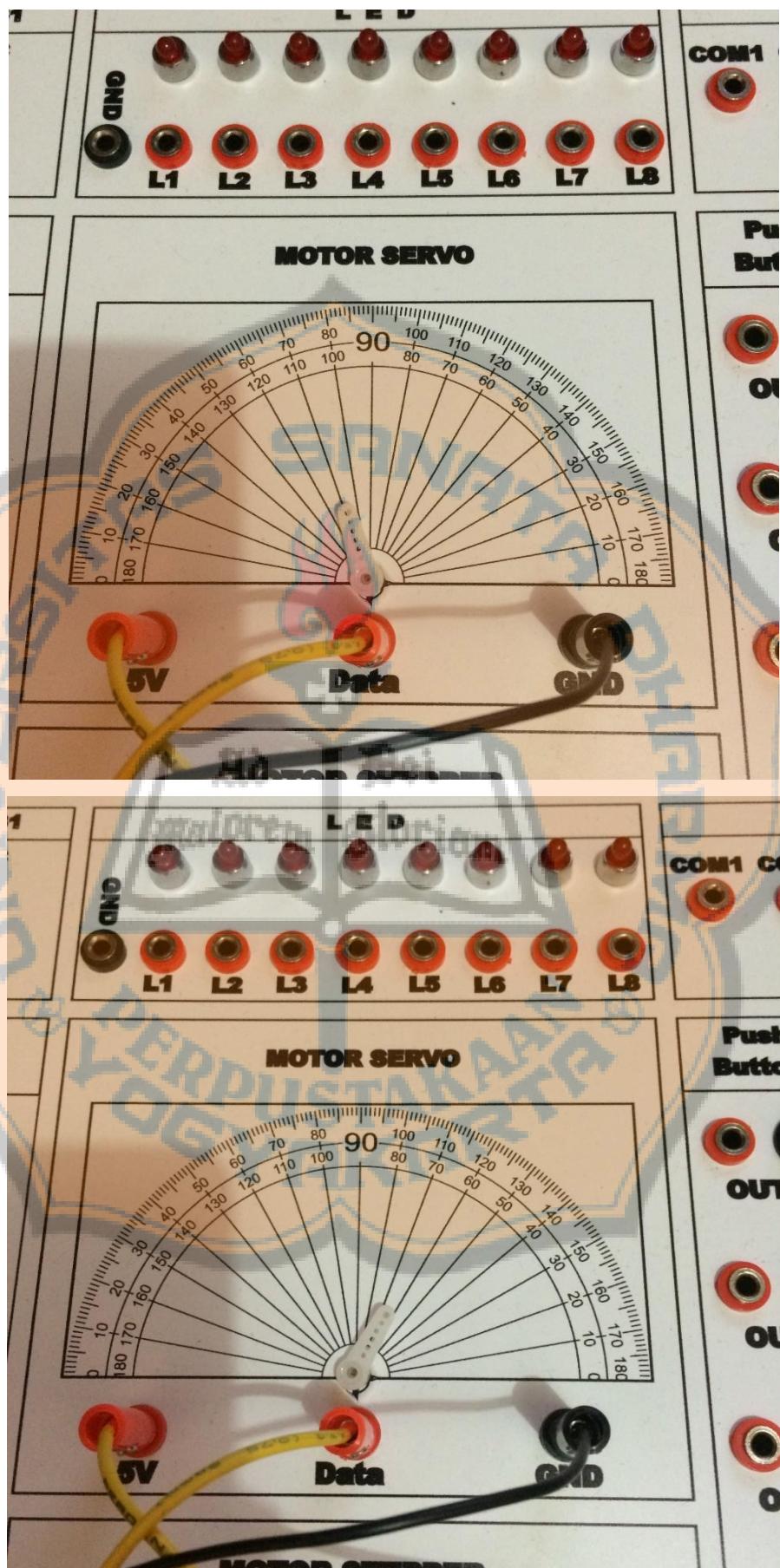
5. Pengujian keypad dan dot matrix

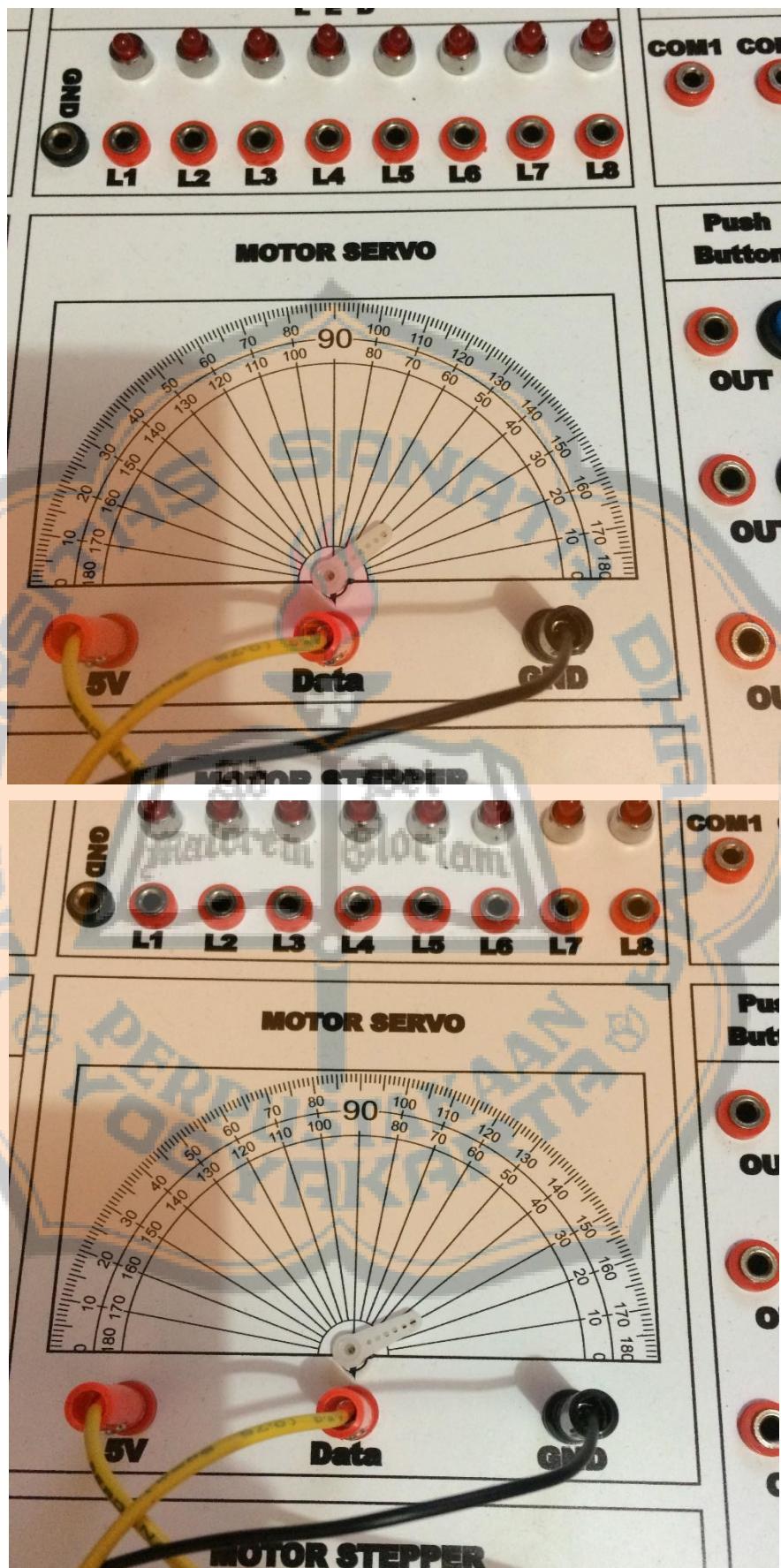


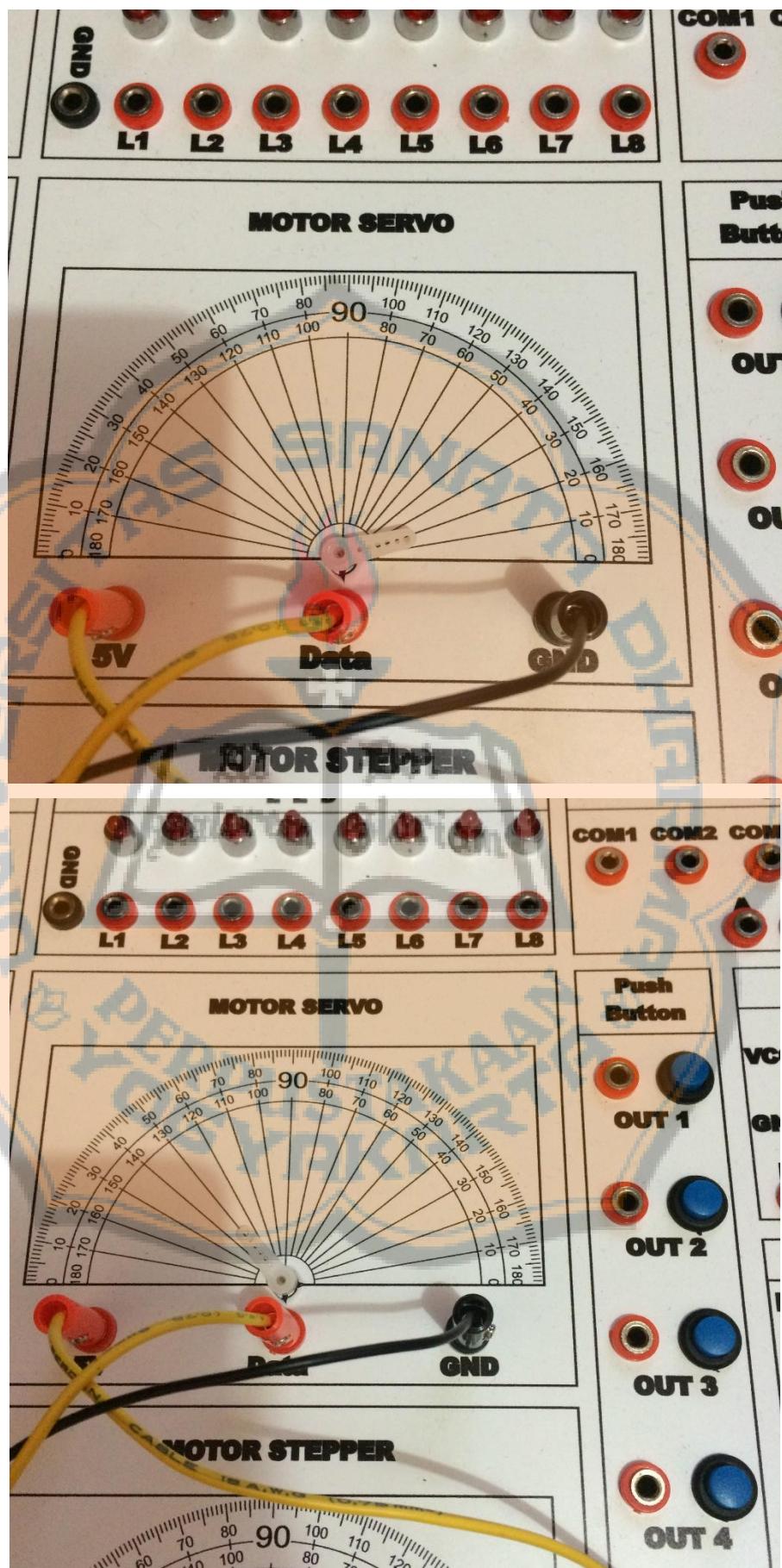


6. Pengujian motor servo

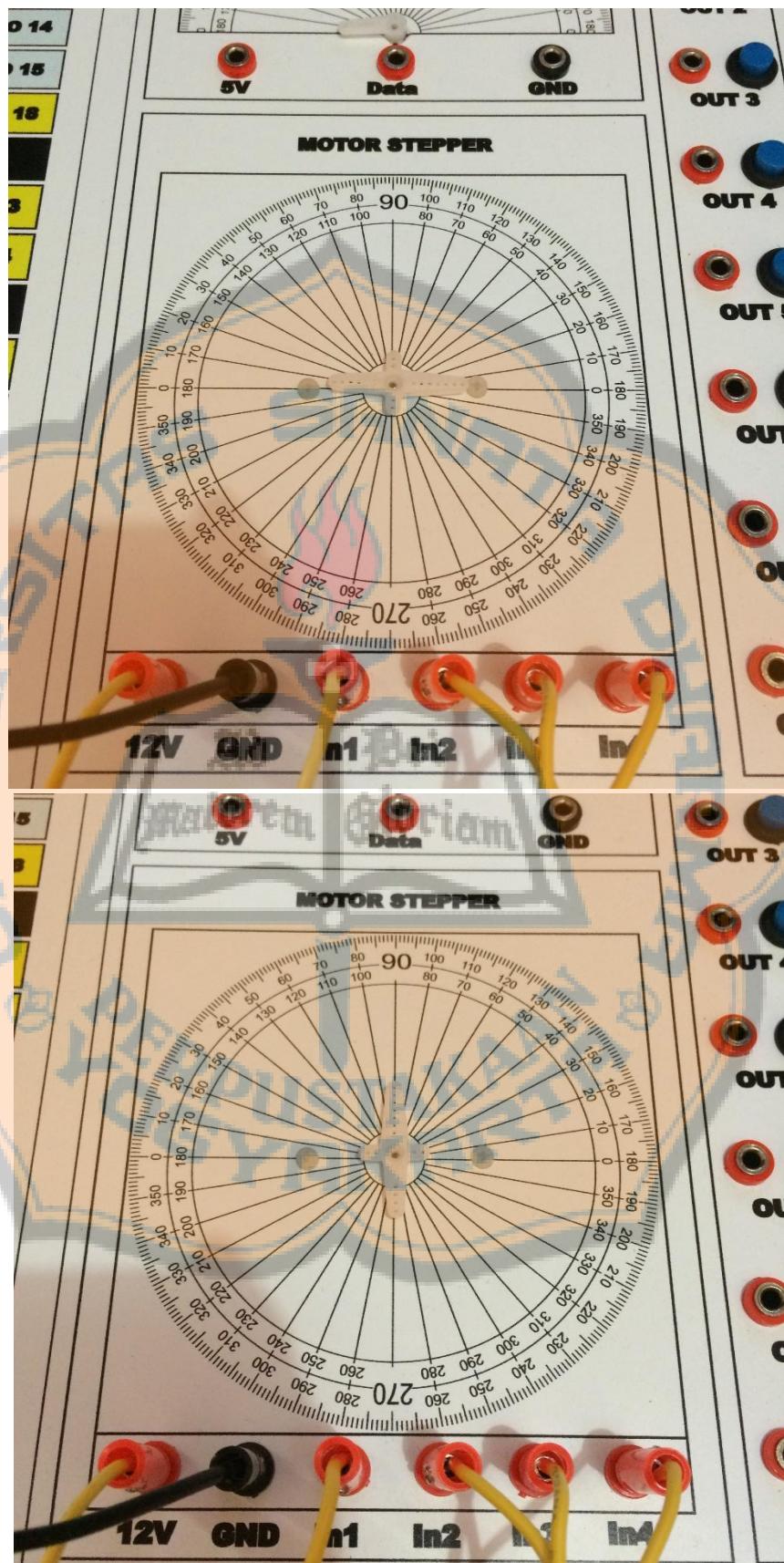




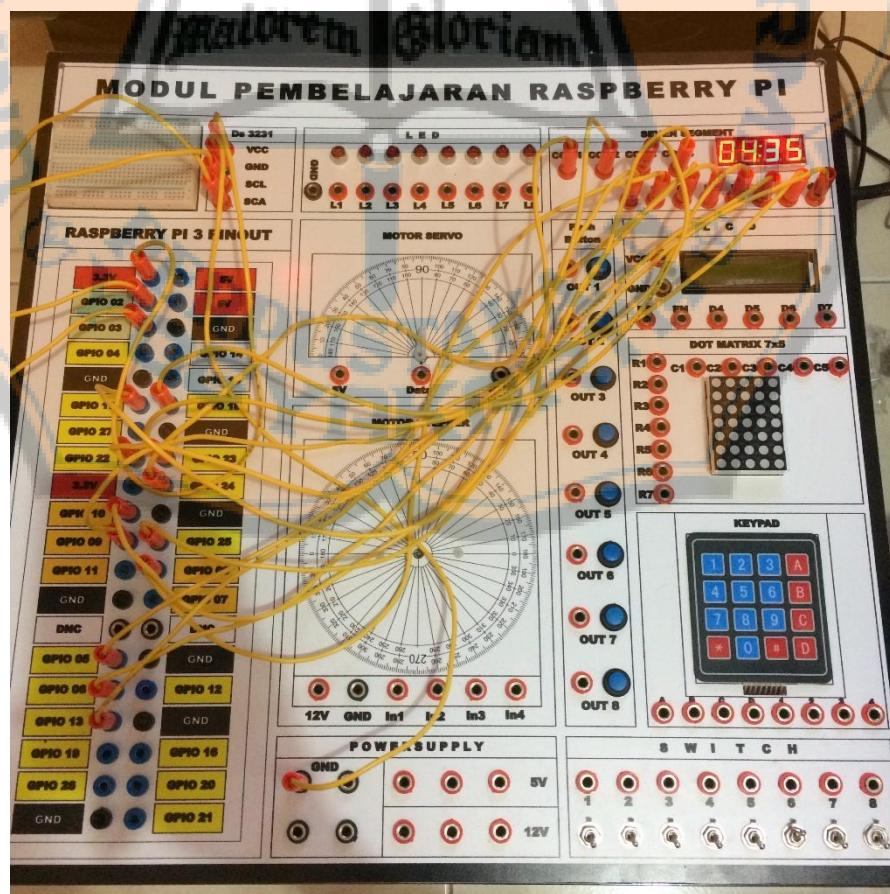
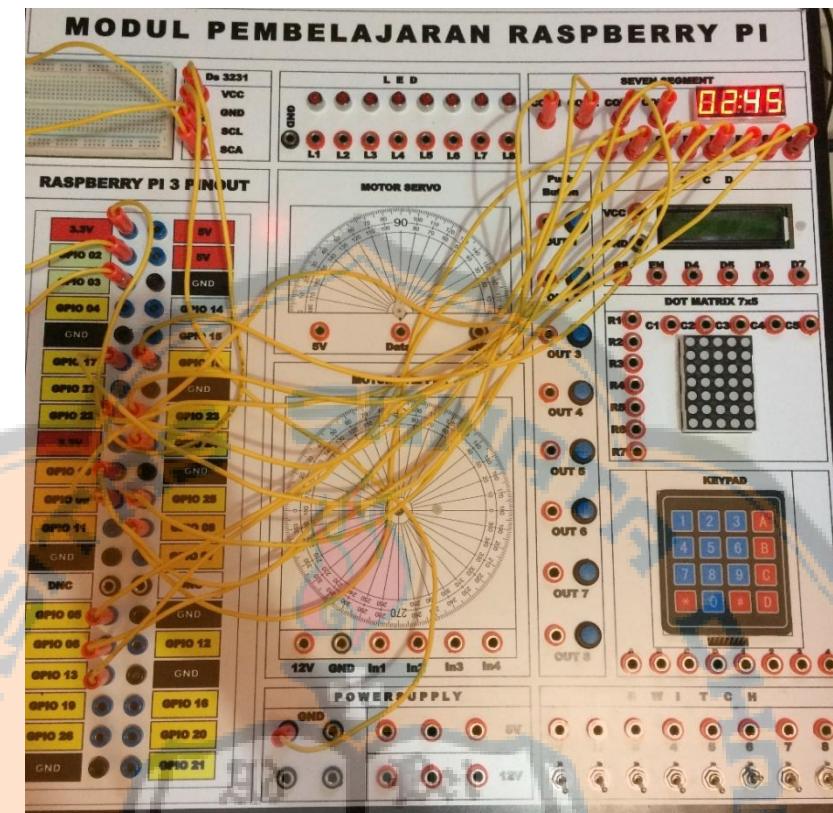


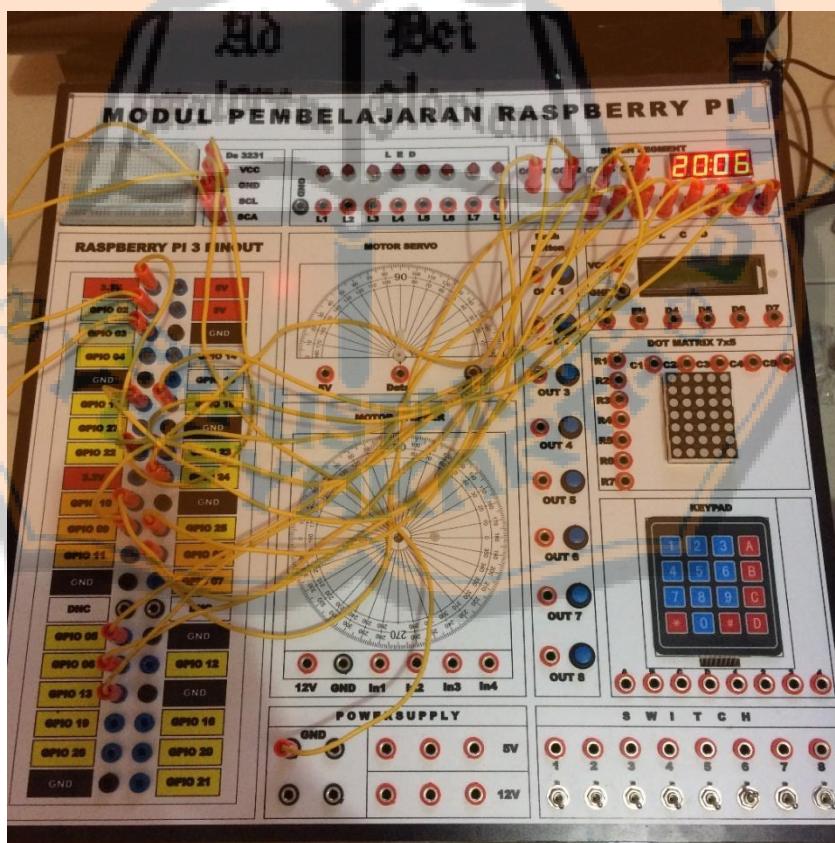
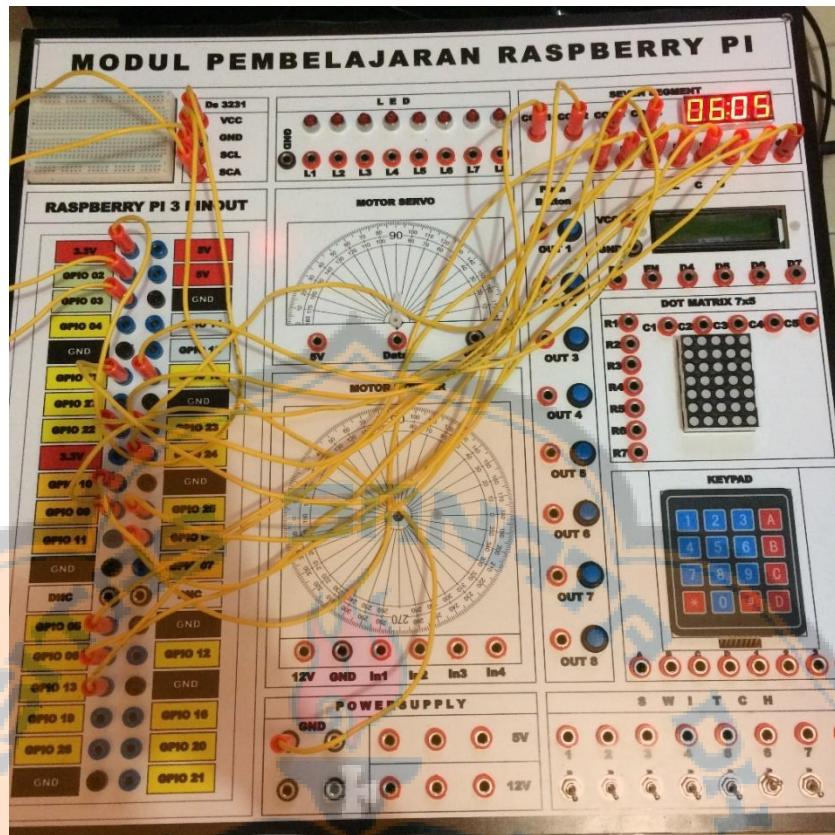


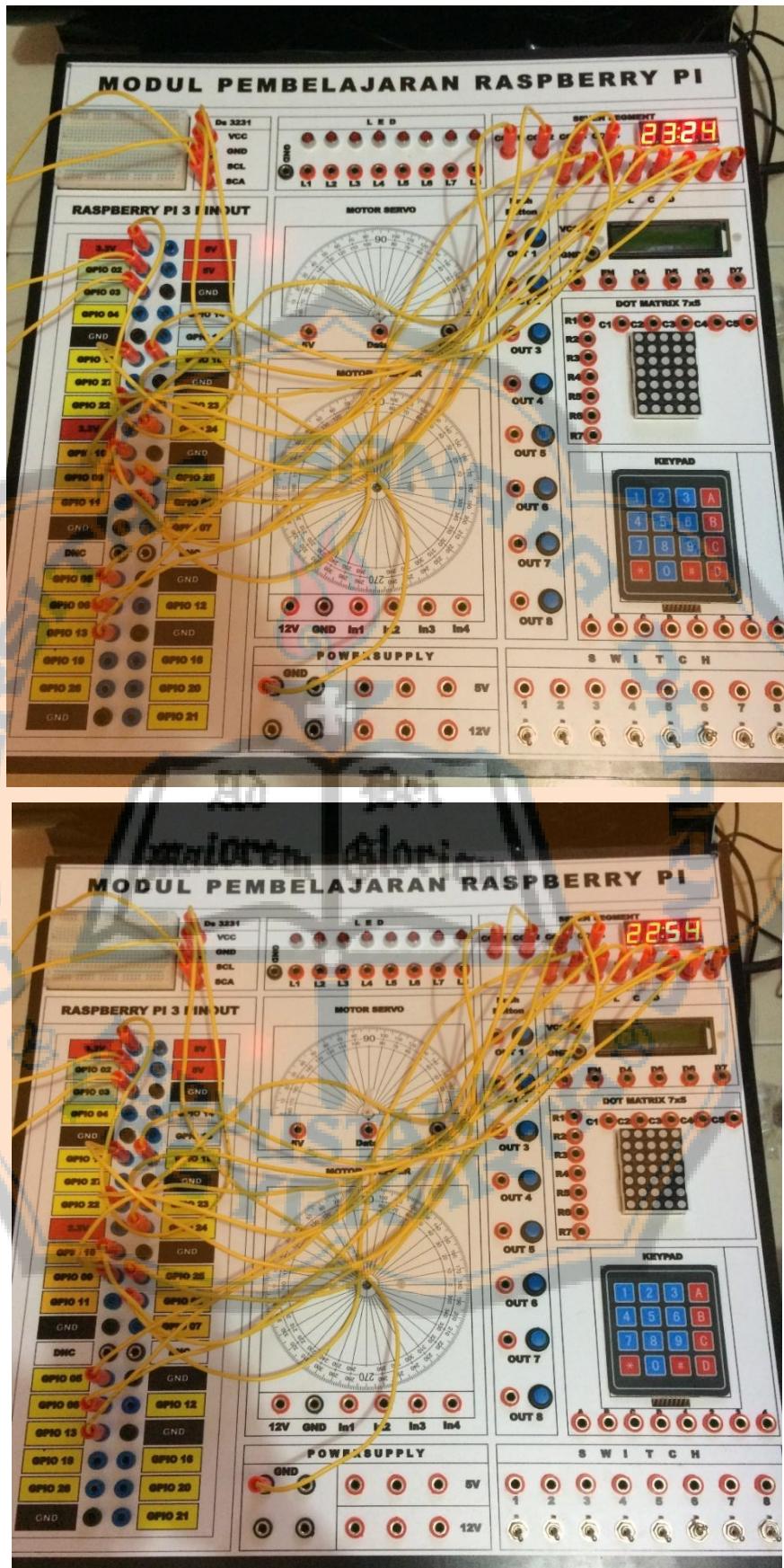
7. Pengujian motor stepper



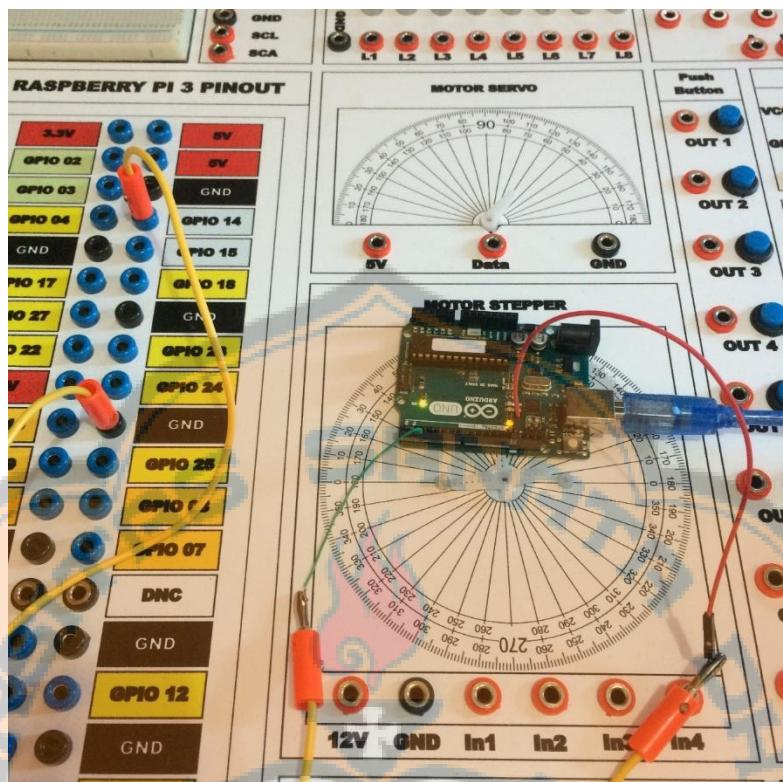
8. Pengujian komunikasi i2c dan seven segment







9. Pengujian komunikasi UART



-----Data Pengujian Komunikasi UART Raspberry Pi ke Arduino-----

```
*Python 2.7.9 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
masukan karakter kirim: A
masukan karakter kirim: B
masukan karakter kirim: C
masukan karakter kirim: D
masukan karakter kirim: E
masukan karakter kirim: F
masukan karakter kirim: G
masukan karakter kirim: H
masukan karakter kirim: I
masukan karakter kirim: J
masukan karakter kirim: K
masukan karakter kirim: L
masukan karakter kirim: M
masukan karakter kirim: N
masukan karakter kirim: O
masukan karakter kirim: P
masukan karakter kirim: Q
masukan karakter kirim: R
masukan karakter kirim: S
masukan karakter kirim: T
masukan karakter kirim: U
masukan karakter kirim: V
masukan karakter kirim: W
masukan karakter kirim: X
masukan karakter kirim: Y
masukan karakter kirim: Z
masukan karakter kirim: |
```

arduinorapicom | Arduino 1.6.8
File Edit Sketch Tools Help
arduinorapicom
void setup()
{
 // put your setup code here, to run once:
 Serial.begin(9600);
 Serial.println("Terima karakter raspi:");
}

void loop()
{
 // put your main code here, to run repeatedly:
 //char inByte=' ';
 if (Serial.available())
 {
 char in = (char)Serial.read();
 Serial.println(in);
 }
 delay(100);
}

Done uploading.

COM4 (Arduino/Genuino Uno)
Terima karakter raspi:
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
 Autoscroll No line ending 9600 baud

-----Data Pengujian Arduino ke Raspberry Pi -----

cobakirim | Arduino 1.6.8
File Edit Sketch Tools Help
cobakirim
void setup()
{
 // put your setup code here, to run once:
 Serial.begin(9600);
 Serial.println(" ");
}
void(*resetFunc) (void)=0;
void loop()
{
 // put your main code here, to run repeatedly:
 //char inByte=' ';
 //if (Serial.available())

 char in = 'A' ;
 Serial.write(in);

 delay(100);
 resetFunc();
}

COM4 (Arduino/Genuino Uno)
A
A
A
A
A
A
A
A
A
A
A
A
A
A
A
A
 Autoscroll No line ending 9600 baud

