



**PENGEMBANGAN SISTEM ROBOT BANTU TERAPI PASIEN
PASCA STROKE BERBASIS PERMAINAN KOMPUTER**

SKRIPSI

Oleh

Kukuh Priambodo

NIM. 151910201102

PROGRAM STUDI TEKNIK ELEKTRO STRATA 1

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS JEMBER

2019



**PENGEMBANGAN SISTEM ROBOT BANTU TERAPI PASIEN
PASCA STROKE BERBASIS PERMAINAN KOMPUTER**

SKIRPSI

Diajukan guna melengkapai tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Teknik Elektro (S1)
dan mencapai gelar Sarjana Teknik

Oleh

Kukuh Priambodo

NIM. 151910201102

PROGRAM STUDI TEKNIK ELEKTRO STRATA 1

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS JEMBER

2019

PERSEMBAHAN

Dengan mengucapkan puji dan syukur kehadirat Allah ‘Azza Wa Jalla limpahan kasih dan karunia -Mu telah memberikan kekuatan dan kemudahan sehingga skripsi ini bisa terselesaikan. Sholawat dan salam selalu dipanjatkan kepada Rasulullah Muhammad Shallallahu ‘Alaihi Wasallam. Dengan tulus ikhlas dan penuh kerendahan hati skripsi ini saya persembahkan kepada :

1. Kedua orang tua tercinta, Bapak Sudarman Rahimahullah dan Ibu Sriwati yang senantiasa mendidik saya dengan sabar, disiplin, keras dan penuh kasih sayang terutama mbak saya Triyas Wulandari Iswati S.Psi yang sudah menjadi donatur terbesar dalam proses perkuliahan saya.
2. Keluarga besar Komunitas Elektronika (ELCO) dan Tim Riset Mobil Listrik Titen Universitas Jember yang sudah memberikan pengalaman dan ilmu yang sangat berharga.
3. Keluarga besar Asisten Laboratorium Elektronika dan Terapan Universitas Jember Mas Wawan, Sigit Wibisono, Turasno, Wardatul dan Novita Murti.
4. Keluarga besar D15TORSI yang selalu menyemangati, mendampingi, dan menghibur saya selama menjadi bagian dari keluarga besar warga teknik elektro 2015.
5. Keluarga besar konsentrasi Elektronika dan Sistem Kendali teknik elektro 2015 Universitas Jember yang selalu ada, selalu kompak dan selalu bergotong royong dalam menyelesaikan permasalahan. Kalian Luarbiasa!!
6. Teman spesial, Aprilya Pravika Sari, S.T., yang sudah rela membantu saya dalam menyelesaikan skripsi selama ini. Terima kasih banyak.
7. Almamater Teknik Elektro Universitas Jember.
8. Pihak-pihak yang tidak dapat disebutkan satu per satu, saya berterima kasih telah membantu saya sampai selesainya pengerjaan skripsi ini.

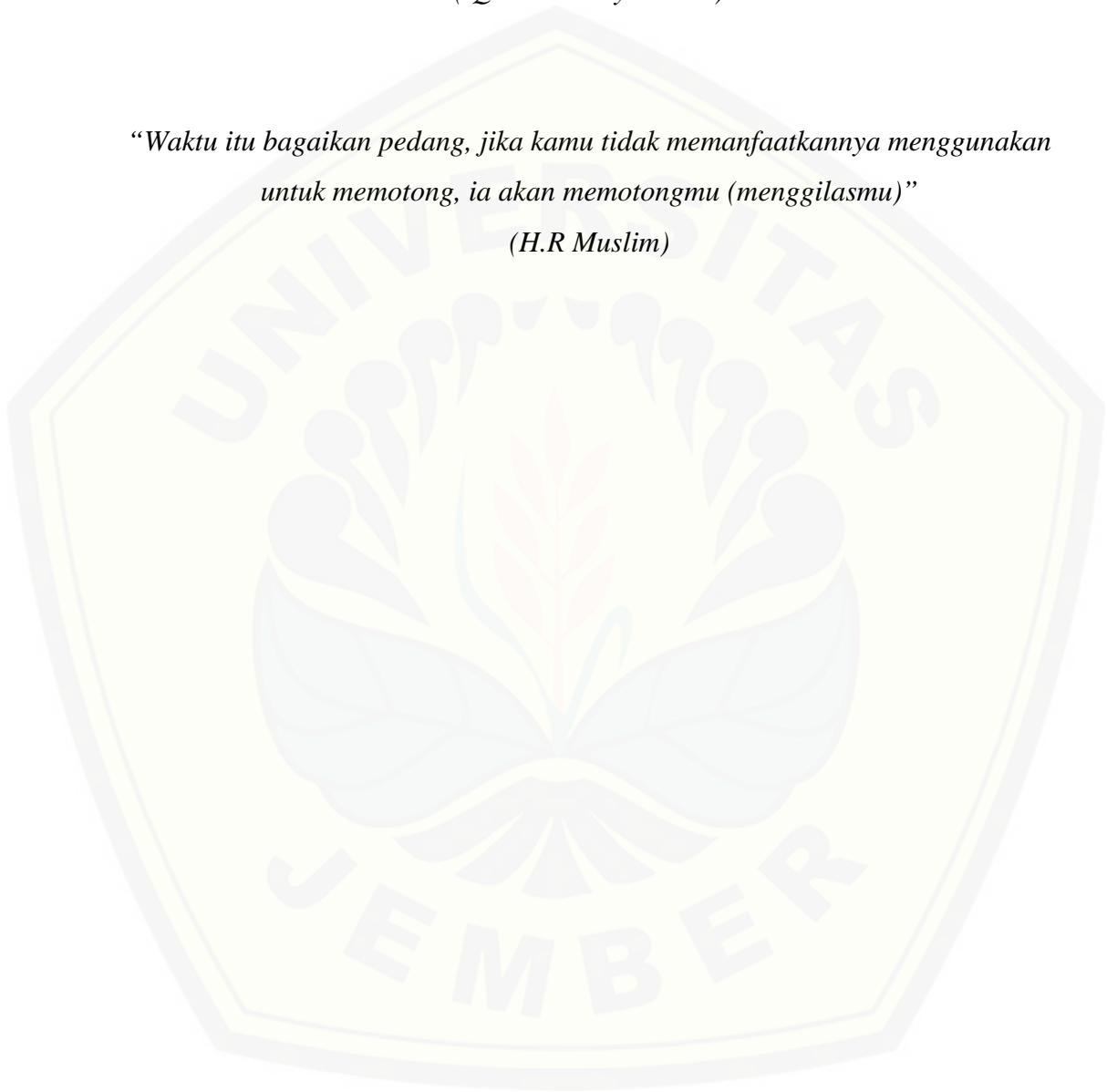
MOTTO

“Maka sesungguhnya bersama kesulitan itu ada kemudahan”

(Q.S. Al - Insiroh : 5)

*“Waktu itu bagaikan pedang, jika kamu tidak memanfaatkannya menggunakan
untuk memotong, ia akan memotongmu (menggilasmu)”*

(H.R Muslim)



PERNYATAAN

Saya yang bertanda tangan dibawah ini:

nama : Kukuh Priambodo

NIM : 151910201102

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab penuh atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 27 Mei 2019

Yang menyatakan,

Kukuh Priambodo

NIM 151910201102

SKRIPSI

**PENGEMBANGAN SISTEM ROBOT BANTU TERAPI PASIEN
PASCA STROKE BERBASIS PERMAINAN KOMPUTER**

Oleh

Kukuh Priambodo

NIM 151910201102

Pembimbing

Dosen Pembimbing Utama : Khairul Anam, S.T., M.T., Ph.D

Dosen Pembimbing Anggota : Ali Rizal Chaidir, S.T., M.T

PENGESAHAN

Skripsi berjudul “Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer” karya Kukuh Priambodo telah diuji dan disahkan pada:

hari, tanggal : Rabu, 29 Mei 2019

tempat : Fakultas Teknik Universitas Jember.

Tim Penguji :

Ketua,

Anggota I,

Khairul Anam, S.T., M.T., Ph.D

Ali Rizal Chaidir, S.T., M.T.

NIP 198002072015042001

NIP 760015754

Anggota II,

Anggota III,

Ike Fibriani, S.T., M.T.

Wahyu Muldayani, S.T., M.T.

NIP 198002072015042001

NIP 760016799

Mengesahkan

Dekan,

Dr. Ir. Entin Hidayah, M.UM.

NIP 196612151995032001

RINGKASAN

Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer : Kukuh Priambodo, 151910201102 : 2019 :65halaman; Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Stroke atau Cerebrovascular Accident (CVA) adalah suatu kondisi dimana pasokan darah ke otak terganggu. Gangguan tersebut diakibatkan adanya sumbatan pada aliran darah yang dikarenakan pecahnya pembuluh darah pada otak. Menurut Stroke Association (2013) Stroke dapat menyerang orang yang berusia diatas 50 tahun akan tetapi usia muda dibawah 45 tahun juga dapat terserang stroke. Pasien yang menderita stroke secara mendadak akan mengalami kelemahan ekstremitas yang mengganggu kegiatan hidup sehari-hari. Terdapat beberapa proses terapi pasien stroke secara medis maupun secara teknologi. Penggunaan terapi secara medis ini masih memiliki kelemahan yaitu membutuhkan waktu yang lama dan membutuhkan pengawasan secara intensif oleh tenaga medis sehingga hal tersebut menyebabkan semakin besarnya biaya terapi.

Pemanfaatan teknologi robot di era sekarang telah mencakup berbagai bidang, salah satunya yaitu di bidang kesehatan. Dalam dunia medis, teknologi robot mampu membantu kinerja proses pemulihan atau rehabilitasi. Robot bantu terapi ini bekerja untuk melatih tangan pasien yang terkena stroke agar dapat pulih kembali. Dalam perancangan robot bantu terapi yang dikendalikan dengan *push button* dan *flex sensor*, kemudian gerak robot bantu terapi yang difungsikan untuk melatih tangan pasien yang terkena stroke. Terdapat metode untuk mengendalikan robot yaitu menggunakan metode *Finite State Machine* atau FSM. FSM adalah sebuah metode yang terdiri dari beberapa *State* berupa perilaku yang dapat dengan mudah terhubung pada setiap *State* secara bersamaan membentuk sistem kontrol. Metode ini menggunakan sistem kontrol *loop* tertutup karena ketika tidak ada gerakan dari sensor yang terpasang pada tangan yang normal maka akan terjadi peralihan *state* dengan perilaku yang berbeda (Satriyo M.B, 2017). Sedangkan

dalam perancangan permainan untuk membantu proses terapi ini menggunakan bahasa pemrograman Python. Permainan yang dikembangkan yaitu berupa permainan Tetris. Permainan Tetris sebuah permainan yang sangat sederhana, permainan ini dapat dimainkan diberbagai konsol video *game* dan untuk algoritmanya bersifat *open source* sehingga sangat cocok untuk dilakukan pengembangan.

Pada pengujian *hardware*, setiap sensor yang digunakan diambil nilai resistansi dan tegangannya. Pada pengujian *flex sensor*, nilai resistansi terhadap sudut tekuk sensor. Dengan sudut 0°, 15°, 30°, 45°, 60°, 75°, dan 90° didapatkan nilai resistansi sebesar 10,1 k ; 10,9 k ; 11,9 k ; 12,7 k ; 14,1 k , 15,3 k , 17,4 k . Dari pengujian tersebut dapat diketahui semakin besar nilai sudut tekuk sensor maka semakin besar juga nilai resistansi yang dihasilkan. Sedangkan pengujian nilai tegangan terhadap sudut tekuk sensor dengan sudut yang sama didapatkan nilai tegangan sebesar 2,5V; 2,4V; 2,29V; 2,21V; 2,08V; 1,98V; 1,83V. Dari pengujian tersebut dapat diketahui bahwa semakin besar nilai sudut tekuk sensor maka semakin kecil nilai tegangan yang dihasilkan.

Pengujian *joystick* dengan melakukan variasi arah dari setiap axis nya dan keluaran dari *joystick* berupa nilai tegangan dimana nilai tegangan pada axis X saat tidak diarahkan keatas dan ke bawah (diam) sebesar 2,52V, saat diarahkan ke atas sebesar 5,17V dan saat diarahkan kebawah sebesar 0V. Sedangkan pada axis Y saat diam tidak diarahkan ke kanan dan kiri sebesar 2,52V, saat diarahkan ke kanan sebesar 5,17V dan saat diarahkan ke kiri sebesar 0V. Pada pengujian motor untuk mencari waktu *rise* dan *fall* pada keluaran *feedback* sistem kontrol yang ada pada motor Linier L12 dengan melakukan variasi waktu kontrol yaitu pada saat waktu 1 mS, 2 mS, dan 3 mS. Dari hasil pengujian ini waktu *rise* dan *fall* tidak dipengaruhi oleh waktu kontrol namun dipengaruhi oleh mekanik robot dan tangan pasien yang memakai robot tangan.

Pada pengujian *software*, pengujian permainan Tetris dilakukan dengan cara mengamati saat responden memainkan permainan Tetris selama 10 menit dengan mode yang berbeda seperti mode normal, mode terapi pemula, dan mode terapi lanjutan. Dan selama 10 menit permainan Tetris tidak mengalami *bug*

sehingga dapat dimainkan dengan waktu yang lama selama proses terapi. Untuk pengujian integrasi robot dan permainan menggunakan komunikasi antar *pin* Gpio pada Raspberry pi.

Pengujian metode FSM dibagi menjadi beberapa *state* dengan waktu aktif setiap *state* sebesar 3 detik sedangkan untuk waktu memindai gerakan yaitu selama 20 detik dan untuk waktu pergantian gerakan robot tangan dalam *state* selama 1,5 detik. Pada pengujian *state free motion*, ketika tidak ada gerakan dari tangan yang terdeteksi oleh sensor maka *state* ini aktif setiap 20 detik. Pada pengujian *state* gerakan terdeteksi 1x, ketika sensor mendeteksi gerakan tangan sebanyak 1x maka *state* ini aktif dan robot akan bergerak dengan gerakan membuka dan menggenggam. Pada pengujian *state* gerakan terdeteksi 2x, ketika sensor mendeteksi gerakan tangan sebanyak 2x maka *state* ini aktif dan robot akan bergerak dengan gerakan yoga jari tangan Apana Mudra dan Prana Mudra. Dan Pada pengujian *state* gerakan terdeteksi 3x, ketika sensor mendeteksi gerakan tangan sebanyak 3x maka *state* ini aktif dan robot akan bergerak dengan gerakan yoga jari tangan Shunya Mudra dan Surya Mudra.

Dalam pengujian keseluruhan robot bantu terapi stroke, robot yang dikendalikan dengan menggunakan metode FSM sangat efektif dalam penggunaan robot sebagai alat bantu terapi. Hal ini dikarenakan setiap gerakan yang dideteksi oleh sensor memiliki fungsi gerakan yang berbeda dalam robot karena setelah gerakan robot masuk dalam *state* yang akan aktif apabila sensor mendeteksi banyaknya gerakan pada tangan yang normal. Dan penggunaan permainan dalam membantu proses terapi juga sangat efektif karena, pasien sebagai pengguna robot akan lebih fokus dan asyik pada permainan sehingga selama proses terapi tidak merasakan jenuh dan pemilihan permainan Tetris sangat tepat karena merupakan permainan yang paling mudah dan familiar saat dimainkan.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala limpahan rahmat, taufik, serta hidayah-Nya sehingga penulis dapat menyelesaikan penelitian sekaligus penyusunan skripsi yang berjudul ” Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer”. Skripsi ini disusun untuk memenuhi salah satu syarat dalam penyelesaian pendidikan strata satu (S1) pada Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Dalam penulisan skripsi ini tentunya banyak pihak yang telah memberikan bantuan baik moril ataupun materil. Oleh karena itu penulis menyampaikan terimakasih kepada :

1. Ibu Dr. Ir. Entin Hidayah, M.UM. selaku Dekan Fakultas Teknik Universitas Jember.
2. Bapak Bambang Srikaloko S.T., M.T selaku Ketua Jurusan Teknik Elektro Universitas Jember.
3. Bapak Khairul Anam, S.T., M.T., P.Hd. selaku Dosen Pembimbing Utama dan Bapak Ali Rizal Chaidir, S.T., M.T. selaku Dosen Pembimbing Anggota yang telah meluangkan waktu dan pikiran serta perhatiannya guna memberikan bimbingan dan arahan demi terselesainya skripsi ini.
4. Ibu Ike Fibriani S.T., M.T. selaku dosen penguji utama dan Bapak Wahyu Mudayani, S.T., M.T. selaku dosen penguji anggota yang telah memberikan kritik dan saran yang membangun sehingga sangat membantu terhadap penyempurnaan skripsi ini.
5. Bapak Widya Cahyadi S.T., M.T. selaku dosen pembimbing akademik yang telah membimbing dan menanamkan rasa disiplin dan tanggung jawab dengan apa yang dilakukan selama penulis menjadi mahasiswa.
6. Kedua orang tua tercinta, Bapak Sudarman Rahimahullah dan Ibu Sriwati yang senantiasa mendidik saya dengan sabar, disiplin, keras dan penuh kasih sayang.

7. Teman seperjuangan se-DPU, se-DPA dan sepenelitian yang saling mendukung, saling membantu, saling menyemangati dan memotivasi dalam penyusunan skripsi.
8. Rekan asisten Laboratorium Elektronika dan Terapan, Mas Wawan, Sigit Wibisono, Turasno, Wardatul dan Novita Murti yang telah menjadi keluarga dari semester 3 hingga sekarang.
9. Rekan Tim HEPINAR, Alumni PIMNAS 3, serta pembimbing PKM bapak Sumardi S.T., M.T. yang sudah memberikan pengalaman dan ilmu yang sangat luarbiasa berharga bagi saya.
10. Rekan tim Mobil Listrik Titen Unej dan teman-teman Komunitas Elektronika (ELCO) yang sudah mau berbagi ilmu, berbagi pengalaman serta berbagi kenangan selama menjadi mahasiswa.
11. Serta seluruh pihak yang telah membantu penulisan dalam menyelesaikan skripsi yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih jauh dari kesempurnaan, karena sempurna hanya milik Allah SWT. Harapan penulis adalah supaya informasi dari skripsi ini dapat memberikan manfaat bagi penulis dan pembaca.

Jember, 27 Mei 2019

Penulis

DAFTAR ISI

HALAMAN SAMPUL.....	i
HALAMAN JUDUL	ii
HALAMAN PERSEMBAHAN.	iii
HALAMAN MOTTO.	iv
HALAMAN PERNYATAAN.....	v
HALAMAN PEMBIMBING.	vi
HALAMAN PENGESAHAN.....	vii
RINGKASAN.	viii
SUMMARY.	ix
PRAKATA.	xvi
DAFTAR ISI	xvii
DAFTAR TABEL.	xx
DAFTAR GAMBAR.....	xxi
BAB 1. PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	4
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	5
BAB 2. TINJAUAN PUSTAKA	6
2.1 Permainan (<i>Game</i>)	6
2.2 Terapi Stroke Berbasis Permainan	8
2.3 Permainan Tetris.	9
2.4 <i>Finite State Machine</i> (FSM)	9
2.5 Sistem Kontrol	12
2.5.1 Sistem Kontrol <i>Loop</i> Terbuka	12
2.5.2 Sistem Kontrol <i>Loop</i> Tertutup	13
2.5.3 Pendeteksi <i>Error</i>	14
2.6 Raspberry Pi 3 B+	15

2.7	<i>Flex Sensor</i>	16
2.8	Modul <i>Joystick</i>	18
2.9	Motor Linier L12	19
BAB 3. METODOLOGI PENELITIAN		21
3.1	Tempat Penelitian	21
3.2	Waktu Penelitian	21
3.3	Tahapan Penelitian	22
3.4	Alat dan Bahan Penelitian	24
3.5	Rancangan Sistem	25
3.5.1	Diagram Blok Sistem	25
3.5.2	Desain Elektronika Konsol Permainan	26
3.5.3	Akuisisi Deteksi Kondisi Tangan	26
3.5.4	Rangkaian Pembagi Tegangan	27
3.5.5	Akuisisi Deteksi Posisi Tangan	28
3.5.6	Desain Mekanik Robot Tangan	29
3.5.7	<i>Flowchart</i> Konsol	33
3.5.8	<i>Flowchart</i> Raspberry Pi	36
3.5.9	Desain Sistem Aplikasi Permainan Terapi	37
3.6	Metode Penelitian	39
3.6.1	<i>Finite State Machine</i> (FSM)	39
a.	<i>Free Motion</i>	40
b.	Memindai Gerakan tangan	41
c.	Terdeteksi Gerakan Tangan 1x	41
d.	Terdeteksi Gerakan Tangan 2x	41
e.	Terdeteksi Gerakan Tangan 3x	42
3.6.2	<i>Flowchart</i> FSM	44
3.6.3	Sistem Kontrol PID	45
3.7	Pengujian Robot Tangan	46
3.7.1	Pengujian <i>Flex Sensor</i>	46
3.7.2	Pengujian Modul <i>Joystick</i>	47
3.7.3	Pengujian Permainan Tetris	47

3.7.4	Pengujian Integrasi Konsol Permainan	48
3.7.5	Pengujian Sistem Kontrol PID Motor Linier L12.	49
3.7.6	Pengujian Metode <i>Finite State Machine</i> (FSM).....	50
3.7.7	Pengujian Kelayakan Robot.	50
BAB.4	HASIL DAN PEMBAHASAN.....	51
4.1.	Analisa Data Pengujian Sensor <i>Flex</i>	51
4.2.	Analisa Data Pengujian Modul <i>Joystick</i>	56
4.3.	Pengujian Permainan Tetris.	57
4.4.	Pengujian Robot Tangan sebagai Konsol Permainan.	58
4.5.	Pengujian Integrasi Robot dan Konsol Permainan.	59
4.6.	Pengujian Sistem Kontrol PID Motor Linier L12.	60
4.7.	Pengujian Metode <i>Finite State Machine</i> (FSM).....	65
BAB.5	PENUTUP.....	75
5.1.	Kesimpulan.	75
5.2.	Saran.	76
DAFTAR PUSTAKA	77
LAMPIRAN	79

DAFTAR TABEL

Tabel 2.1 Spesifikasi dan fitur Raspberry Pi 3 B+	15
Tabel 2.2 Spesifikasi <i>Flex Sensor</i>	17
Tabel 2.3 Spesifikasi Modul <i>Joystick</i> Bi-axial	18
Tabel 2.4 Spesifikasi Motor Linier L12	19
Tabel 3.1 Jadwal Kegiatan Penelitian	21
Tabel 4.1 Pengujian Resistansi <i>Flex sensor</i> terhadap Sudut Tekuk.....	54
Tabel 4.2 Pengujian Tegangan <i>Flex sensor</i> terhadap Sudut Tekuk.	55
Tabel 4.3 Pengujian Modul <i>Joystick</i> sebagai Konsol Permainan.	57
Tabel 4.4 Pengujian Robot Tangan Sebagai Konsol Permainan.....	58
Tabel 4.5 Pengujian Sistem Kontrol PID Motor Linier L12 Kontrol 1mS.....	60
Tabel 4.6 Pengujian Sistem Kontrol PID Motor Linier L12 Kontrol 2mS.....	61
Tabel 4.7 Pengujian Sistem Kontrol PID Motor Linier L12 Kontrol 3mS.....	62
Tabel 4.8 Hasil Pengujian Pengujian State Free Motion.	66

DAFTAR GAMBAR

Gambar 2.1 Contoh game berplatform android untuk terapi stroke	9
Gambar 2.2 Diagram Finite State Machine untuk mesin counter	11
Gambar 2.3 Diagram Blok Sistem Kontrol <i>Loop</i> Terbuka	12
Gambar 2.4 Sistem Kontrol <i>Loop</i> Tertutup	13
Gambar 2.5 Lokasi Pendeteksi <i>Error</i> dalam Sistem Kontrol <i>Loop</i> Tertutup	14
Gambar 2.6 Raspberry Pi 3 B+	16
Gambar 2.7 <i>Flex Sensor</i>	18
Gambar 2.8 Modul <i>Joystick</i>	19
Gambar 2.9 Bentuk Fisik Motor Linier L12	20
Gambar 3.1 <i>Flowchart</i> Tahapan Pelaksanaan Penelitian	22
Gambar 3.2 Diagram Blok Sistem	25
Gambar 3.3 Desain Elektronika Konsol Permainan	26
Gambar 3.4 <i>Flex Sensor</i>	27
Gambar 3.5 Rangkaian Pembagi Tegangan	28
Gambar 3.6 Rangkaian pada Modul <i>Joystick</i>	29
Gambar 3.7 Desain rancangan Robot tangan Tampak Kiri Atas	30
Gambar 3.8 Desain rancangan Robot tangan Tampak Kanan Atas	30
Gambar 3.9 Desain rancangan Robot tangan Tampak Depan	31
Gambar 3.10 Kondisi Robot Tangan saat Menggenggam	32
Gambar 3.11 Kondisi Robot Tangan saat Membuka	32
Gambar 3.12 <i>Flowchart</i> Konsol.....	33
Gambar 3.13 <i>Flowchart</i> Raspberry Pi	36
Gambar 3.14 Diagram FSM.....	40
Gambar 3.15 Contoh Gerakan Senam Jari Tangan Apana Mudra.....	41
Gambar 3.16 Contoh Gerakan Senam Jari Tangan Prana Mudra	42
Gambar 3.17 Contoh Gerakan Senam Jari Tangan Shunya Mudra	43
Gambar 3.18 Contoh Gerakan Senam Jari Tangan Surya Mudra.....	43
Gambar 3.19 <i>Flowchart</i> FSM	44

Gambar 3.20 Diagram Blok Sistem Kontrol PID	45
Gambar 3.21 Proses Pengujian <i>Flex Sensor</i> Terhadap Sudut.....	46
Gambar 3.22 Proses Pengujian <i>Joystick</i> dengan Kombinasi Arah.....	47
Gambar 3.23 Proses Pengujian Permainan Tetris pada Raspberry Pi 3 B+.....	48
Gambar 3.24 Hasil Pengujian Integrasi Robot Sebagai Konsol Dengan Sebuah Permainan Tetris.....	48
Gambar 3.25 Proses Pengujian Sistem Kontrol PID Motor Linier Actuonix L12 Tanpa Beban Tangan.....	49
Gambar 4.1 Robot Bantu Terapi Stroke.....	51
Gambar 4.2 Grafik data Resistansi terhadap Sudut Tekuk.....	53
Gambar 4.3 Grafik Data Tegangan terhadap Sudut Tekuk.....	55
Gambar 4.4 Tampilan Menu Permainan Tetris.....	57
Gambar 4.5 Tampilan Utama Permainan Tetris.....	58
Gambar 4.6 Sampel Grafik Motor Linier L12 Ketika Kedaaan <i>Minimal Force</i> Ke <i>Maximal Force</i> Tanpa Beban Tangan.....	64
Gambar 4.7 Sampel Grafik Motor Linier L12 Ketika Kedaaan <i>Maximal Force</i> Ke <i>Minimal Force</i> Tanpa Beban Tangan.....	64
Gambar 4.8 Proses Pengujian <i>State Free Motion</i>	65
Gambar 4.9 Gerakan <i>Free Motion</i>	67
Gambar 4.10 hasil pengujian memindai pergerakan.....	68
Gambar 4.11 Pengujian State Terdeteksi Gerakan Tangan 1x.....	69
Gambar 4.12 Pengujian State Terdeteksi Gerakan Tangan 2x.....	70
Gambar 4.13 Gerakan yoga jari Apana Mudra.....	70
Gambar 4.14 Pengujian State Terdeteksi Gerakan Tangan 3x.....	71
Gambar 4.15 Gerakan yoga jari Surya Mudra.....	72
Gambar 4.16 Robot Bantu Terapi Dengan Push Button Yang Terhubung Dengan Permainan Tetris.....	73
Gambar 4.16 Robot Bantu Terapi Dengan Push Button Yang Terhubung Dengan Permainan Tetris.....	73
Gambar 4.17 Poin Akhir Dari Permainan Tetris.....	74

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Stroke atau *Cerebrovascular Accident* (CVA) adalah suatu kondisi dimana pasokan darah ke otak terganggu. Gangguan tersebut diakibatkan adanya sumbatan pada aliran darah yang dikarenakan pecahnya pembuluh darah pada otak. Ketika otak manusia mengalami kekurangan pasokan aliran darah maka terjadi reaksi biokimia yang dapat merusak sel-sel saraf otak sehingga akan mengakibatkan hilangnya fungsi yang dikendalikan oleh jaringan itu dan otak tidak lagi berfungsi sebagaimana mestinya. Menurut *Stroke Association* (2013) Stroke dapat menyerang orang yang berusia diatas 50 tahun akan tetapi usia muda dibawah 45 tahun juga dapat terserang stroke. Pasien yang menderita stroke secara mendadak akan mengalami kelemahan ekstremitas yang mengganggu kegiatan hidup sehari-hari.

Terdapat beberapa proses terapi pasien stroke secara medis maupun secara teknologi. Pada proses terapi secara medis setelah pasien tersebut benar-benar dinyatakan positif terkena stroke, sehingga mampu meningkatkan harapan hidup dan kembali normal bagi pasien. Penggunaan terapi secara medis ini masih memiliki kelemahan yaitu membutuhkan waktu yang lama dan membutuhkan pengawasan secara intensif oleh tenaga medis sehingga hal tersebut menyebabkan semakin besarnya biaya terapi.

Teknologi robot telah digunakan untuk terapi stroke yang dikembangkan di Shanghai, China yaitu Robot “Hand of Hope” (Rehab,-robotics, 2016). Robot tangan ini bekerja dengan menggunakan sensor otot dengan mendeteksi sinyal otot atau *electromyography* (EMG). *Electromyography* (EMG) merupakan teknik yang digunakan untuk mengevaluasi kinerja otot berdasarkan aktivitas elektrik yang terjadi pada otot (K.Y. Tong, et al., 2011) . Aktivitas elektrik pada otot terjadi sebelum otot berkontraksi dan aktivitas tersebut dapat dideteksi secara sederhana menggunakan elektroda. Robot “Hand of Hope” yang dikembangkan di Hongkong telah dibekali fitur-fitur yang kompleks, seperti berbagai macam permainan sehingga proses terapi pada pasien dapat berjalan efisien dan tidak

membosankan. Penggunaan terapi secara teknologi ini masih memiliki kelemahan yaitu biaya robot yang mahal dan proses pembelian robot harus *import* sehingga hal ini yang mengakibatkan sulit terjangkaunya robot oleh masyarakat Indonesia.

Pada beberapa penelitian sebelumnya, robot tangan ini sudah menggunakan beberapa pengendali seperti pada penelitian dengan judul “Rancang Bangun Robot Tangan Berbasis EMG Menggunakan *Extreme Learning Machine*” (Iqbal Gilang Wildana, 2017). Pada penelitian tersebut telah menghasilkan robot bantu terapi stroke yang dapat dikendalikan dengan sensor EMG yang diletakkan dilengan pasien yang tidak mengalami stroke. Hasil dari metode pada penelitian ini hanya mampu menggerakkan robot tangan membuka dan menutup semua jari tangan. Pada penelitian yang kedua dengan judul “Rancang Bangun Robot Tangan Untuk Terapi Stroke Menggunakan *Flex sensor* Berbasis Arduino” (Herlambang Pulung Samudra, 2018). Pada penelitian tersebut penulis membuat sebuah robot bantu terapi stroke yang dapat dikendalikan dengan *Flex sensor* secara wireless dengan menggunakan *bluetooth* dan robot tangan ini juga dapat menggerakkan jari satu per satu sehingga untuk proses terapi bisa lebih efektif. Namun, pada penelitian yang pertama dan kedua proses terapinya tidak menggunakan permainan sebagai alat bantu untuk menanggulangi timbulnya rasa bosan pasien. Kemudian pada penelitian yang ketiga dengan judul “Rancang Bangun Sistem Elektronik Robot Tangan Untuk Mendukung Pemulihan Penyandang Disabilitas Tangan Berbasis Genuino 101” (Rifqi Afkar, 2018). Pada penelitian ini robot tangan dapat dikendalikan dengan *push button* yang ditekan oleh tangan pasien yang tidak mengalami stroke. Hasil dari penelitian ini pasien dapat melakukan proses terapi dengan bantuan permainan sebagai stimulus agar pasien tidak merasa bosan saat proses terapi berlangsung. Namun, permainan tersebut tidak terintegrasi dengan robot hanya sekedar stimulus bagi pasien terapi sehingga pasien tidak dapat dimonitoring perkembangannya secara valid oleh tenaga medis.

Berdasarkan permasalahan dan fakta diatas, maka salah satu cara untuk proses penyembuhan penyakit stroke dengan melakukan pendekatan terapi, salah satunya yaitu dengan mengkombinasikan alat bantu terapi dengan sebuah permainan. Dalam penelitian ini robot bantu terapi akan diintegrasikan dengan

sebuah permainan. Permainan ini akan membantu pasien yang melakukan proses terapi agar tidak bosan selama proses terapi. Penelitian ini menggunakan metode analisis sistem kontrol PID yang terdapat pada motor linier dan metode *Finite State Machine* (FSM). Metode sistem kontrol PID ini penulis tidak mendesain sistem kontrol karena dalam motor Linier L12 sudah terdapat fitur sistem kontrol PID didalamnya, namun menganalisa waktu *rise time* dan *fall time* yang paling sesuai untuk robot tangan ini. Sedangkan untuk metode FSM ini pada robot tangan yaitu agar pasien dapat memperoleh banyak variasi pergerakan robot selama proses terapi, sehingga tangan pasien tidak hanya diterapi membuka dan menggenggam saja. Dan fungsi skor pada permainan untuk mengetahui perkembangan pasien selama proses terapi, jika nilainya besar maka pasien tersebut dapat dikatakan mengalami peningkatan selama proses terapi. Sedangkan jika skor pada permainan nilainya kecil maka pasien masih belum mengalami peningkatan selama proses terapi.

Dalam penelitian ini maka penulis mengusulkan sebuah robot yang dapat terintegrasi dengan sebuah permainan komputer untuk membantu proses terapi pasien pasca stroke dengan nyaman tanpa ada rasa bosan selama proses terapi. Mengingat jumlah penderita stroke di Indonesia yang tinggi dan teknologi robot yang sudah dikembangkan masih belum dapat dijangkau oleh mayoritas penduduk Indonesia.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, dapat dirumuskan beberapa rumusan masalah sebagai berikut:

1. Bagaimana merancang sebuah robot yang dapat melatih pergerakan tangan pada pasien pasca stroke ?
2. Bagaimana kinerja dari robot terapi yang dilengkapi dengan permainan ?
3. Bagaimana sistem kontrol robot terapi dengan menggunakan metode *Finite State Machine* yang dapat memindai pergerakan tangan pasien yang tidak mengalami stroke ?

1.3 Batasan Masalah

Untuk memfokuskan tujuan penelitian maka penulis memberi batasan masalah rencana penelitian ini. Adapun yang menjadi batasan penelitian ini adalah sebagai berikut, yaitu :

1. Robot tangan dikendalikan oleh *Embedded System* yang sudah digunakan sebelumnya seperti *Flex sensor* dan *push button*.
2. Mode *Flex sensor* yang terhubung dengan permainan hanya satu sensor yang digunakan.
3. Tidak mengakses nilai ADC *Flex sensor* dan *Joystick* pada Raspberry pi.
4. Permainan komputer hanya permainan Tetris yang sudah di modifikasi.
5. Sistem Kontrol PID tidak mendesain sendiri melainkan menggunakan bawaan spesifikasi dari motor Linier L12
6. Perancangan desain permainan dan kontrol robot menggunakan bahasa Python pada Raspberry pi.
7. Target pencapaian dari penelitian ini adalah sebuah robot yang dapat terintegrasi dengan sebuah permainan yang dengan gerakan robot tangan membuka dan menggenggam.

1.4 Tujuan Penelitian

Dengan meninjau latarbelakang pada permasalahan yang telah diuraikan, maka dapat dirumuskan tujuan utama dari penelitian ini diantaranya :

1. Merancang sebuah robot yang dapat melatih pergerakan tangan pada pasien pasca stroke.
2. Mengetahui kinerja dari robot terapi yang dilengkapi dengan permainan.
3. Mengetahui sistem kontrol robot terapi dengan menggunakan metode *Finite State Machine* yang dapat memindai pergerakan tangan pasien yang tidak mengalami stroke.

1.5 Manfaat Penelitian

Dengan dilakukannya penelitian ini, diharapkan dapat memberikan manfaat yaitu merancang robot tangan yang terintegrasi oleh sebuah permainan yang menyenangkan sebagai alat bantu terapi bagi penyandang disabilitas pasca stroke dengan biaya yang murah sehingga dapat terjangkau seluruh masyarakat Indonesia dan diharapkan dengan adanya penelitian ini dapat memberikan warna baru terhadap penelitian dibidang elektronika biomedis di Indonesia.



BAB 2. TINJAUAN PUSTAKA

Pada tinjauan pustaka ini akan dijelaskan beberapa bagian yang menjadi dasar dalam pembuatan sebuah aplikasi permainan yang digunakan untuk membantu terapi pasien pasca stroke. Pada bab ini juga dijelaskan bahasan utama yaitu penjelasan tentang pembuatan robot tangan yang terintegrasi dengan sebuah permainan untuk membantu proses terapi pasien pasca stroke.

2.1 Permainan (*Game*)

Permainan (*game*) merupakan sebuah media yang dapat digunakan untuk menyampaikan sebuah tujuan. Tujuan yang terdapat dalam *game* mempunyai macam-macam jenis yaitu pendidikan, hiburan dan simulasi. Dalam kehidupan manusia, *game* selalu ada dan terus diminati oleh berbagai kalangan dan usia baik sebagai penghilang penat setelah seharian belajar atau bekerja (Martono K.T, 2015). Awal mula munculnya *game* hanya sebagai hiburan semata, namun seiring berkembangnya teknologi *game* sudah digunakan untuk keperluan di bidang yang lain misalkan di bidang pendidikan, bisnis, politik, kedokteran dan militer. Perkembangan yang begitu pesat dengan jenis yang beragam, mulai *game* yang hanya dapat dimainkan oleh satu orang saja hingga *game* yang dapat dimainkan oleh beberapa orang sekaligus (Putra & Muslim, 2013).

Klasifikasi *game* dimaksudkan untuk memudahkan pengelompokan jenis *game*. Beberapa klasifikasi *game* adalah seperti berikut :

1. *Game as game* yaitu *game* yang dimaksud adalah *game* untuk kesenangan. Ketika para pemain memainkan jenis *game* ini maka pemain akan merasakan kesenangan untuk menghilangkan kepenatan dalam dirinya.
2. *Game as media* yaitu *game* yang bertujuan utama untuk menyampaikan pesan tertentu dari pembuat *game*, dari sebuah sejarah dan dari sebuah kisah yang dapat membuat para pemain mengikuti apa pesan moral apa yang terkandung dalam *game* tersebut.

3. *Game beyond game* yaitu bisa disebut juga dengan istilah *gamification*. *Gamification* adalah penerapan konsep atau cara berfikir *game design* ke dalam lingkup *non-game* (Martono K.T,2015).

Selain klasifikasi *game*, terdapat jenis platform yang digunakan dalam pengembangan sebagai berikut :

1. *Arcade game*, yaitu sebuah permainan yang semua misinya mudah dimengerti oleh para permainan, menyenangkan dan grafiknya bagus meskipun terlihat sederhana. Yang dimaksud dengan misinya mudah dimengerti dan menyenangkan karena permainan ini hanya sesuatu yang disenangi secara umum seperti pukul memukul, tembak menembak, tusuk menusuk, kejar kejaran dan semua yang mudah. Dan biasanya juga permainan ini ada sebuah tata cara memainkan dan instruksi pada setiap *chapter*.
2. *PC game* adalah permainan yang dapat dimainkan di komputer pribadi, bukan pada video permainan konsol atau mesin dingdong. Permainan yang dibuat oleh sebuah perusahaan pengembang dibidang aplikasi permainan, biasanya akan mempublikasikan baik secara mandiri atau dengan bantuan penerbit pihak ketiga. Kemudian akan didistribusikan pada media fisik yaitu berupa CD maupun DVD dan pada media elektronik seperti internet agar para pemain dapat mendownload dan menginstall secara langsung tanpa media fisik. permainan komputer membutuhkan sebuah media khusus untuk menggunakan seperti generasi spesifik unit pemrosesan grafik, koneksi internet untuk bermain secara online dan spesifikasi komputer. Fitur yang menarik pada permainan komputer dapat membuat para pemainnr tidak merasa bosan karena permainan saat ini sudah dapat berinteraksi permainan antar pemainnr secara langsung dengan media internet secara online.
3. *Konsol game* adalah sebuah perangkat elektronik khusus untuk menjalankan sebuah permainan video. Perngkat keluarannya dapat berupa

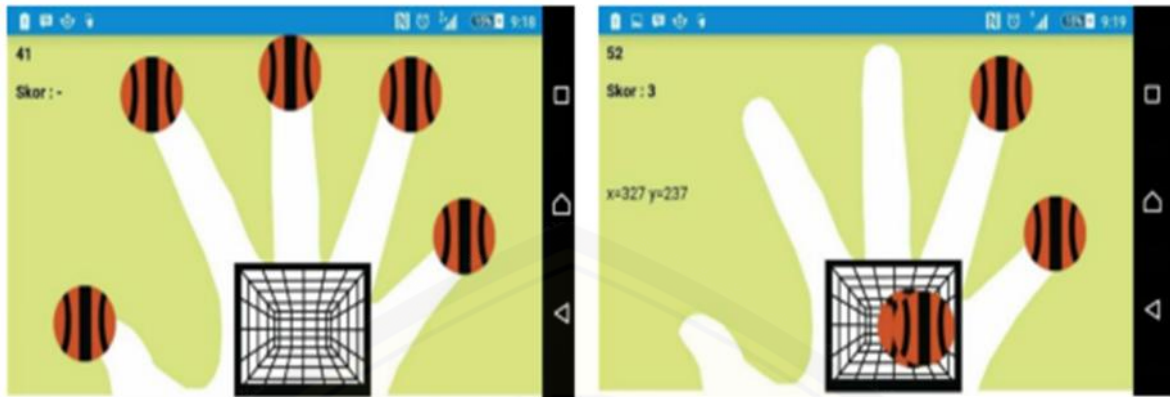
layar monitor dan dimainkan dengan konsol *game* tertentu seperti Playstation, XBOX dan Nitendo Wii.

4. *Mobile game* adalah sebuah *game* yang dapat dimainkan atau khusus untuk mobile phone seperti android.

2.2 Terapi Stroke berbasis Permainan

Pasien stroke yang lolos dari kematian, bukan berarti masalahnya sudah selesai. Ini adalah sebuah awal dari perjuangan untuk kembali normal. Terapi adalah salah satu cara yang terbaik dengan melakukannya secara berulang-ulang untuk melatih anggota tubuh yang belum bisa berfungsi secara normal setelah terkena stroke. Terapi bertujuan untuk mendorong perubahan neuroplastis otak (Kristanto A.N & Herianto, 2016). Terapi yang dilakukan secara berulang-ulang bukanlah aktivitas pengisi waktu luang yang menyenangkan karena selama proses terapi pasien tidak mempelajari keterampilan baru dan menarik. Semua orang akan termotivasi ketika melakukan aktivitas yang mereka sukai dan menyenangkan, sebab ada kecenderungan alamiah untuk memusatkan perhatian, melatih, dan melaksanakan aktivitas yang disukai. Sehingga selama proses terapi akan terasa seperti bermain (Levine, 2011).

Freitas et al. (2012) mengembangkan sebuah game untuk membantu proses rehabilitasi pasien stroke dan menguji usabilitasnya. Pengujian ini dilakukan dengan tiga grup yang berbeda yaitu fisioterapis, ahli komputer dan orang awam. Pasien stroke belum dilibatkan dalam penelitian tersebut. Terapi bermain adalah salah satu cara menyenangkan yang digunakan untuk terapi sebagai usaha pemulihan dari seorang pasien. Telah banyak inovasi game terapi yang telah diciptakan saat diketahui bahwa dengan bermain ternyata kemampuan pasien dapat lebih baik daripada melakukan terapi tanpa adanya permainan (Zakiyah L & Mahtarami A, 2015). Berikut contoh game yang digunakan untuk membantu proses terapi.



Gambar 2.1 Contoh game berplatform android untuk terapi stroke

(sumber : Zakiyah L & Mahtarami A, 2015)

2.3 Permainan Tetris

Tetris adalah sebuah tipe permainan yang dibuat oleh sepasang *programmer* asal Rusia pada tahun 1985. Dalam permainan tetris, balok-balok yang berjatuh dari atas ke bawah dalam waktu yang konstan namun waktu akan bertambah cepat seiring dengan bertambahnya level dalam permainan tetris. Balok tetris terdiri dari 4 balok kecil yang berbentuk 7 macam rupa yang unik (Ridwan, Adipradana B, & Syahdana W.A, 2005).

Tetris dapat dimainkan diberbagai setiap konsol permainan video dan komputer bahkan dapat dimainkan di kalkulator dan ponsel. Tujuan dari permainan ini adalah untuk memanipulasi tetromino yang jatuh dengan menggerakannya ke samping atau memutarnya. Sehingga akan berbentuk garis mendatar tanpa celah. Ketika sudah berbentuk tetromino, maka akan menghilang dan akan muncul lagi objek tetromino dari atas dengan rupa yang berbeda.

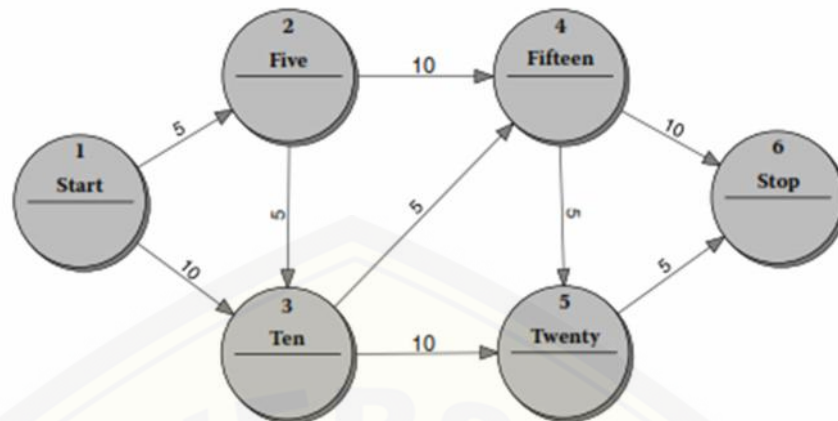
2.4 *Finite State Machine* (FSM)

Sebuah metode sistem kontrol yang dapat digunakan untuk kontrol sebuah alat, salah satunya yaitu robot tangan. *Finite state machine* merupakan sebuah mesin abstrak yang berfungsi untuk mendefinisikan berbagai kondisi yang menentukan kapan suatu *state* harus berubah. Setiap *state* yang sedang dijalankan tersebut dapat menentukan perilaku yang terjadi pada objek yang bersangkutan

(Satriyo M.B, 2017). Dalam sebuah jurnal, *finite state machine* termasuk kedalam jenis *High level control*. FSM sendiri diartikan suatu mekanisme untuk menentukan solusi berdasarkan perubahan-perubahan keadaan (*state*) waktu demi waktu. Dalam teknik FSM, perubahan suatu *state* terjadi berdasarkan informasi data umpan balik dari sensor (Purnama W, Puspitasari R, & Haritman E, 2014).

Selain itu dalam jurnal yang lain, FSM adalah metode perancangan sistem kontrol terdiri dari beberapa *state* berupa perilaku yang dapat dengan mudah dihubungkan setiap *state* tersebut secara bersama-sama hingga membentuk sistem *loop* terbuka. Metode ini juga dapat menggunakan sistem kontrol *loop* tertutup jika dalam suatu *state* diinginkan untuk melakukan sebuah perulangan perilaku. Jika dalam *loop* tertutup tidak terjadi perulangan perilaku, maka akan terjadi peralihan *state* untuk perilaku yang berbeda (Satriyo M.B, 2017).

Diagram transisi *state* adalah representasi grafis yang setara dengan matriks transisi. Grafik transisi *state* menggunakan dua elemen: lingkaran untuk menunjukkan *state* dan busur untuk transisi. (Secara teknis, untuk matematikawan, lingkaran mewakili simpul dan garis mewakili tepi dari grafik berarah.) Dalam diagram yang sangat sederhana, busur dapat berbentuk garis lurus dan kondisi transisi dituliskan di atas busur. Diagram transisi *state* untuk penghitung mesin penghitung otomatis ditunjukkan pada Gambar 2.2. Matriks transisi adalah tabel dan dalamnya terdapat beberapa situasi yang lebih mudah jika digambar daripada berbentuk grafik. Di sisi lain, diagram transisi *state* lebih mudah dipahami daripada matriks transisi. Dalam hal Parser kedua presentasi menunjukkan informasi yang sama dan setara 100%. Tugas utama mesin *state* adalah menghasilkan aksi.



Gambar 2.2 Diagram *Finite State Machine* untuk mesin *counter*
(sumber : Wagner F, Schmuki R dkk, 2006)

Hasil yang bisa didapat oleh model Parser tidak banyak. Pertama, gagasan *state* awal Mulai dan berhenti *state* sangat membatasi mesin *state* harus bekerja terus menerus, sepanjang waktu, dalam kasus umum. Aspek penting lainnya: sebagian besar hasil juga memerlukan beberapa aksi yang harus dilakukan (*output*); aksi yang diperlukan oleh sistem yang dikendalikan. Dalam desain perangkat keras istilah "*output*" umumnya digunakan, sedangkan untuk desain perangkat lunak istilah "aksi" lebih populer. Beberapa jenis aksi dapat ditentukan tergantung pada kondisi dan saat mereka dilakukan seperti :

- Aksi Entri
- Aksi keluar
- Aksi *input*
- Aksi transisi

Aksi Entri adalah aksi yang dilakukan ketika mesin *state* memasuki kondisi. Aksi Keluar adalah aksi yang dilakukan ketika mesin *state* meninggalkan *state*. Aksi *Input* adalah aksi yang dilakukan ketika *input (state)* benar. Setiap *state* bagian memiliki Aksi *Input* yang masing-masing. *Input* Aksi yang dilakukan di *state* bagian mana pun juga digunakan. Aksi Transisi adalah aksi yang dilakukan selama perubahan *state*. Perhatikan bahwa meskipun mirip dengan

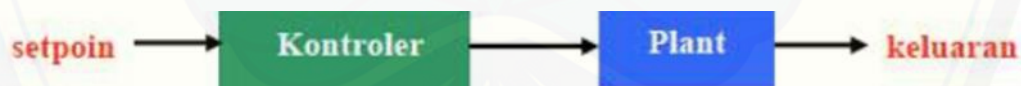
mereka itu bukan Aksi *input* atau Aksi Keluar, yang keduanya tergantung pada *state*; Aksi Transisi tergantung pada transisi (Wagner F, Schmuki R & dkk, 2006).

2.5 Sistem Kontrol

Kontrol adalah upaya yang dilakukan untuk mencapai kondisi yang diinginkan pada sistem dengan merubah variabel tertentu yang dipilih. Kontrol juga dapat diartikan mengukur nilai dari variabel suatu sistem yang dikontrol dan menerapkan variabel yang dimanipulasi ke dalam sistem untuk mengetahui nilai yang menyimpang dari nilai yang diinginkan. Sedangkan kontroler adalah komponen dalam sistem kontrol yang menghasilkan sinyal kontrol. Sistem kontrol terbagi menjadi dua yaitu sistem kontrol *loop* tertutup dan sistem kontrol *loop* terbuka.

2.5.1 Sistem Kontrol *loop* Terbuka

Merupakan sistem kontrol yang keluarannya tidak mempunyai pengaruh terhadap aksi kontrol karena tidak memiliki sinyal umpan balik. Sistem kontrol *loop* terbuka ini hanya dapat digunakan jika hubungan antara masukan dengan keluaran sistem diketahui dan tidak terdapat *noise* dari luar dan dari dalam. Berikut diagram blok sistem kontrol *loop* terbuka.



Gambar 2.3 Diagram Blok Sistem Kontrol *Loop* Terbuka
(Sumber: Buku Ajar Sistem Kontrol, 2016)

Kelebihan dari sistem kontrol *loop* terbuka :

1. Kontruksi sederhana
2. Tidak memerlukan banyak komponen
3. Tidak ada persoalan stabilitas
4. Cocok untuk sebuah sistem yang memiliki keluaran yang tidak bisa diukur

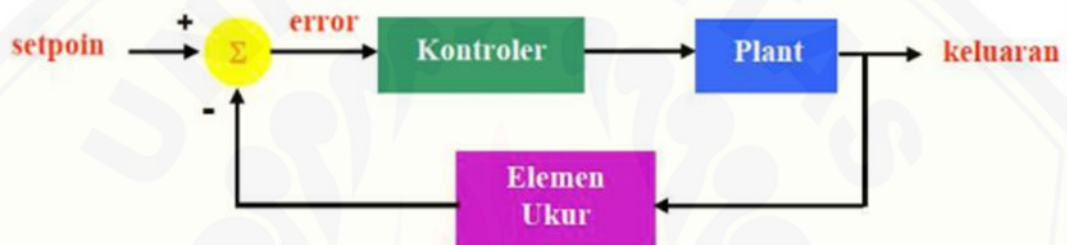
Kekurangan dari sistem kontrol *loop* terbuka :

1. Keluaran sistem akan berbeda dengan yang diinginkan

2. Rekalibrasi membutuhkan waktu lama
3. Dapat digunakan jika hubungan antara masukan dan keluaran sudah diketahui
4. Dapat digunakan jika antara masukan dan keluaran tidak ada *noise*

2.5.2 Sistem Kontrol *Loop* Tertutup

Sistem kontrol *loop* tertutup merupakan sistem kontrol dimana sinyal keluaran mempunyai pengaruh terhadap sinyal kontrol karena adanya sinyal umpan balik. Berikut adalah diagram blok dari sistem kontrol *loop* tertutup.



Gambar 2.4 Sistem Kontrol *Loop* Tertutup
(Sumber: Buku Ajar Sistem Kontrol, 2016)

Pada sistem kontrol *loop* tertutup, sinyal keluaran plant dari elemen ukur seperti sensor atau transduser akan diumpan balikkan untuk dibandingkan dengan *set point*. Perbedaan antara sinyal keluaran dengan *set point* disebut dengan *error*. Kemudian *error* akan menjadi masukan pada kontroler untuk mengurangi kesalahan dan membawa keluaran sistem ke nilai yang diinginkan.

Kelebihan dari sistem kontrol *loop* tertutup :

1. dapat mengatasi ketidakpastian pengetahuan akan plant dan perubahan karakteristik plant.
2. Nonlinearitas komponen tidak terlalu mengganggu
3. Ketelitian terjaga

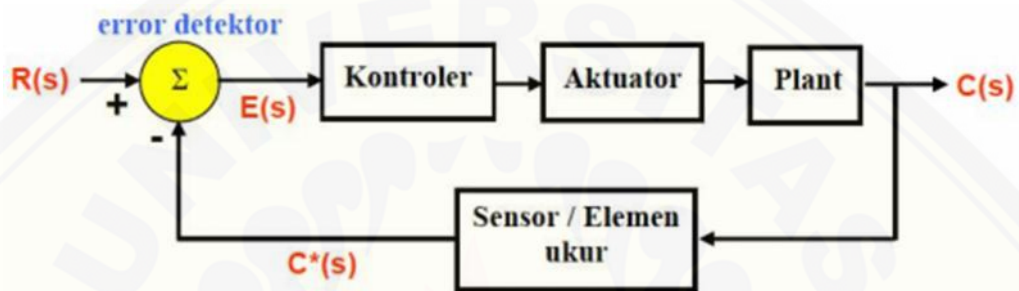
Kekurangan dari sistem kontrol *loop* tertutup :

1. perlengkapannya lebih kompleks dan lebih mahal

- 2. instalasi dan perawatannya lebih sulit
- 3. kecenderungan ke arah osilasi

2.5.3 Pendeteksi *Error*

Pendeteksi *error* merupakan salah satu komponen sistem kontrol *loop* tertutup yang biasanya digunakan untuk membandingkan sinyal keluaran yang terukur dengan sinyal *set point*. Kedudukan pendeteksi *error* dalam sistem kontrol *loop* tertutup dapat dilihat dalam gambar blok diagram berikut.



Gambar 2.5 Lokasi Pendeteksi *Error* dalam Sistem Kontrol *Loop* Tertutup
(Sumber: Buku Ajar Sistem Kontrol, 2016)

Persamaan simbol sebuah *error* sebagai berikut :

$$\begin{array}{c}
 R(s) \rightarrow \begin{array}{c} + \\ \Sigma \\ - \\ \uparrow \\ C^*(s) \end{array} \rightarrow E(s) = R(s) - C^*(s) \dots\dots\dots (2.1)
 \end{array}$$

Atau

$$\begin{array}{c}
 R(s) \rightarrow \begin{array}{c} \otimes \\ + \quad - \\ \uparrow \\ C^*(s) \end{array} \rightarrow E(s) = R(s) - C^*(s) \dots\dots\dots (2.2)
 \end{array}$$

Dimana,

E = Sinyal *Error*

R = Sinyal *Set Point*

C^* = Sinyal keluaran yang terukur

2.6 Raspberry Pi 3 B+

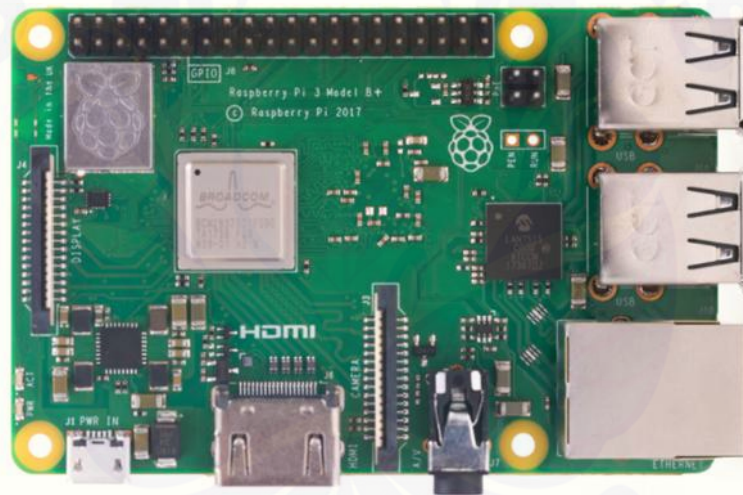
Raspberry Pi 3 Model B+ adalah versi terbaru dari seri Raspberry Pi 3, Pi 3B+ memiliki bentuk dan ukuran yang identik dengan Pi 3B, Namun apabila dibandingkan dengan Raspberry Pi 3 Model B, Pi 3B+ mengalami peningkatan di beberapa bagian *hardware*, mulai dari prosesor 64-bit yang kini memiliki clockspeed maksimum 1.4 GHz (sebelumnya 1.2 GHz pada Pi 3B), memiliki Gigabit Ethernet (*support* PoE) yang tentu jauh lebih kencang dari versi sebelumnya, memiliki heatsink pada prosesor untuk distribusi panas yang lebih baik, serta mendukung dual band WLAN 5 GHz dan 2.4 GHz. Raspberry pi 3 B+ juga memiliki beberapa fitur yang dapat mendukung untuk membuat sebuah proyek berbasis komputer secara portabel. Berikut spesifikasi dari Raspberry Pi 3 B+ pada tabel 2.1 dibawah ini.

Tabel 2.1 Spesifikasi dan fitur Raspberry Pi 3 B+

Processor	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memory	1GB LPDDR2 SDRAM
Konektivitas	<ul style="list-style-type: none"> • 2.4 GHz dan 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE • Gigabit Ethernet over USB 2.0 (Max. 300Mbps) • 4x USB 2.0 Ports
Akses	Extended 40-pin GPIO
Video & Sound	<ul style="list-style-type: none"> • 1x full size HDMI • MIPI DSI display port • MIPI CSI camera Port
Multimedia	H.264, MPEG-4 decode (1080p30);

	H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
Tegangan Masukan	<ul style="list-style-type: none"> • 5V/2.5A DC via mikro USB • 5V DC via GPIO • Power over Ethernet(PoE)
Penyimpanan Data	Mikro SD Card
Temperatur Operasi	0° - 50° C

(static.raspberrypi.org, 2018)



Gambar 2.6 Raspberry Pi 3 B+
(Sumber :raspberrypi.org, 2018)

2.7 Flex Sensor

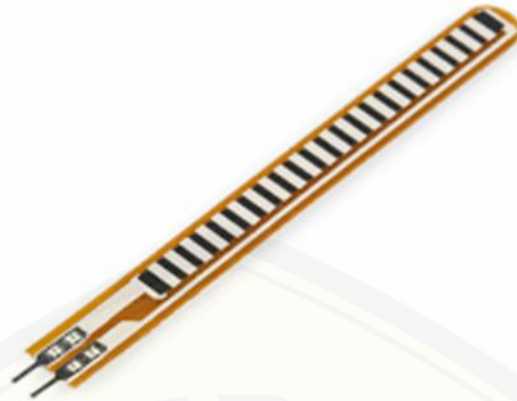
Flex sensor merupakan resistor analog sehingga sensor ini bekerja sebagai pembagi tegangan analog. *Flex sensor* tersusun dari elemen resistif karbon dengan substrat fleksibel tipis (Jsea, 2012). Bila substratnya membungkuk sensor menghasilkan resistansi resistansi terhadap kelengkungan jari-jari *Flex sensor*. Bila substratnya bengkak, sensor menghasilkan *keluaran* resistansi yang

berkorelasi dengan radius kelengkungan. Sirkuit pengkondisi sinyal digunakan untuk membaca resistansi perubahan resistansi yang selanjutnya akan diproses dalam ADC. ADC mengubah nilai ini menjadi nilai digital (Ijecet, 2017). Perangkat *Flex sensor* terbuat dari bahan tipis (kurang dari 0,005in) yaitu bahan plastik fleksibel yang dilapisi dengan film resistif, tinta karbon / polimer, yang dapat diterapkan pada hampir semua bentuk dan ukuran. Lapisan resistif dicetak pada film plastik seperti polimida yang membentuk ikatan yang sangat kuat (Ijarece, 2015).

Tabel 2.2 Spesifikasi *Flex Sensor*

Spesifikasi Mekanik	
<i>Life Cycle</i>	>1 million
Tinggi	0,43 mm
Range Temperatur	-35°C sampai +80 °C
Spesifikasi Elektrik	
Resistansi <i>Flat</i>	25 K
Toleransi	±30%
Resistansi Tekukan	45 K sampai 125 K
<i>Power Rating</i>	0.50 W – 1 W (peak)

(Sprakfun, 2015)



Gambar 2.7 *Flex Sensor*
(Sumber: Sprakfun, 2015)

2.8 Joystick

Modul *joystick* adalah komponen yang berbentuk seperti tuas yang dapat digerakan ke berbagai arah untuk mendapatkan posisi yang diinginkan. Modul ini memiliki 2 sumbu axis yaitu X axis dan Y axis. Sumbu axis ini bernilai tegangan dimana tegangan tersebut didapat dari potensiometer yang dipasang pada modul *joystick* sehingga ketika *joystick* diputar maka potensiometer dari axis X dan Y akan keluar nilai tegangannya. Modul ini yang banyak dipakai yaitu tipe bi-axial karena tipe ini merupakan tipe yang sama dengan *joystick* yang digunakan pada ganggang kendali analog pada konsol Sony Playstation dan X-box. Berikut adalah spesifikasi *joystick* modul bi-axial.

Tabel 2.3 Spesifikasi Modul *Joystick* Bi-axial

Axis	X axis dan Y axis
Dimensi	4cm x 2.6cm x 3.2cm
<i>Switch</i>	1 <i>push button</i>
Pin	5 (Vcc, Gnd, VRx, Vry, SW)

(Depoinovasi, 2018)



Gambar 2.8 Modul *Joystick*
(Sumber: Depoinovasi, 2018)

2.9 Motor Linier

Motor linier adalah salah satu jenis motor bersumber tegangan DC (*Direct Current*) yang dapat menghasilkan gerakan linier. Motor linier L12 ini dapat juga dikendalikan dengan mode RC servo, mode kendali PWM (*pulse Width Modulation*), mode kendali arus (4-20 mA) dan terdapat sinyal *feedback* posisi motor ketika aktif. Spesifikasi dari Motor Linear L12 dapat dilihat pada tabel 6.5 dibawah ini.

Tabel 2.4 Spesifikasi Motor Linier L12

<i>Gearing Option</i>	50:1	100:1	210:1
<i>Peak Power Point</i>	17N @ 14mm/s	31N @ 7mm/s	62N@3,2mm/s
<i>Peak Efficiency Point</i>	10N @ 19mm/s	17N @ 10mm/s	36N@4,5mm/s
<i>Max Speed (no load)</i>	25mm/s	13mm/s	6,5mm/s
<i>Max Force (lifted)</i>	22N	42N	80N

<i>Voltage Option</i>	6 VDC	12 VDC
<i>Max Masukan Voltage</i>	7,5V	13,5V
<i>Stall Current</i>	460mA	185mA
<i>Standby Current</i>	7,2mA	3,3mA
<i>Operating Temperature</i>	-10 ⁰ C to +50 ⁰ C	
<i>Potentiometer Linearity</i>	Less than 2,00%	
<i>Max Duty Cycle</i>	20%	
<i>Audible Noise</i>	5,5dB @ 45cm	

(Actuonix, 2018)



Gambar 2.9 Bentuk Fisik Motor Linier L12

(Sumber : Actuonix, 2018)

BAB 3. METODE PENELITIAN

Pada bab ini dijelaskan tentang beberapa hal yaitu tentang objek penelitian, tahap penelitian, tempat penelitian, waktu penelitian, alat dan bahan yang digunakan dalam penelitian serta perancangan sistem elektronika dan sistem kendali pada objek penelitian. Pada bab ini juga dijelaskan tahapan-tahapan proses akuisisi dari *Flex sensor*, modul *joystick* dan metode *Finite State Machine (FSM)* yang digunakan untuk menggerakkan robot tangan diintegrasikan dengan sebuah aplikasi permainan dengan menggunakan aplikasi Geany yang dapat membantu proses terapi penderita pasca stroke.

3.1 Tempat Penelitian

Pelaksanaan pembuatan alat dan pengambilan data ini dilakukan di Laboratorium CDAST (*Center for Development of Advance Science and Technology*) Universitas Jember

3.2 Waktu Penelitian

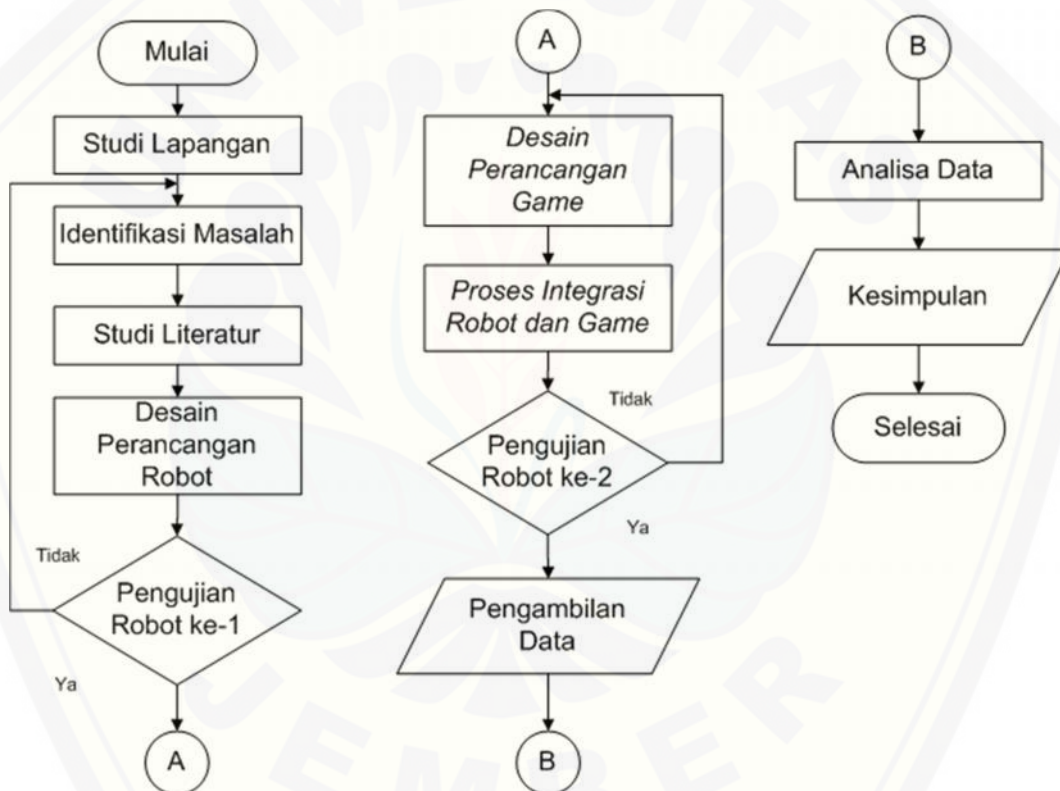
Waktu penelitian dilaksanakan dilaksanakan selama kurang lebih 6 bulan, berikut adalah tabel jadwal penelitian.

Tabel 3.1 Jadwal Kegiatan Penelitian

No	Kegiatan	Bulan											
		Maret			April			Mei			Juni		
1	Studi Literatur	■	■	■									
2	Perancangan Alat		■	■	■	■	■	■	■				
3	Pengambilan Data					■	■	■	■	■			
4	Analisa Data							■	■	■	■		
5	Pengambilan Kesimpulan									■	■		

3.3 Tahap penelitian

Tahap penelitian “Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer” mulai dari Studi lapangan, identifikasi masalah, studi literatur, desain perancangan robot, pengujian robot pertama, desain perancangan permainan, proses integrasi robot dan permainan, pengujian robot kedua, pengambilan data dan analisa data kemudian ditarik kesimpulan. Secara sederhana tahapan penelitian yang akan dilakukan dapat digambarkan dalam *flowchart* sebagai berikut :



Gambar 3.1 *Flowchart* Tahapan Pelaksanaan Penelitian
(sumber: Penulis)

Langkah-langkah yang akan dilakukan dalam penelitian ini adalah sebagai berikut :

1. Studi Lapangan

Tahap awal pelaksanaan penelitian ini adalah mengamati secara langsung robot yang telah dibuat sebelumnya dengan mencari masalah yang

didapatkan oleh peneliti untuk dijadikan sebuah data identifikasi masalah yang akan dilakukan penelitian pengembangan selanjutnya.

2. Studi Literatur

Tahap kedua pelaksanaan penelitian ini adalah dengan mencari literatur dari hasil penelitian sebelumnya melalui buku atau internet untuk mengetahui karakteristik komponen sistem, prinsip kerja serta teori yang menunjang lainnya. Diharapkan dengan literatur yang telah didapat dapat memberikan arahan untuk mengurangi *error* dalam penelitian.

3. Perancangan Robot

Tahapan ini merupakan tahap merancang konstruksi secara sistematis dari robot yang akan dilakukan penelitian. Diharapkan dari proses perancangan konstruksi yang sistematis ini, alat yang nantinya akan diteliti dapat terbentuk. Hal – hal lain yang dilakukan yakni seperti proses 3D printing, proses perakitan, perancangan elektronika dan perancangan sistem kendali.

4. Implementasi Robot

Setelah *hardware* terbentuk, maka dilakukan pengujian pada masing–masing blok dan kemudian pengujian dilakukan pada keseluruhan sistem. Dalam implementasi robot ini juga dilakukan proses kalibrasi dimana pada proses kalibrasi ini bertujuan agar pembacaan sensor akurat agar dapat memberikan perintah pada robot untuk membuka tangan dan menggenggam tangan.

5. Perancangan Permainan

Tahapan ini merupakan tahap merancang sebuah permainan yang nantinya akan dihubungkan dengan robot yang telah dibuat sebelumnya. Diharapkan dari proses perancangan *software* yang sistematis ini, alat yang nantinya akan diteliti dapat terhubung dengan sebuah permainan. Hal – hal lain yang dilakukan yakni seperti proses algoritma permainan dengan menggunakan aplikasi python, proses eksekusi permainan pada laptop atau pada Raspberry Pi 3 B+.

6. Integrasi Robot dan Permainan

Tahapan ini merupakan tahap integrasi antara robot dengan sebuah permainan yang nantinya permainan akan dikendalikan oleh sensor yang terpisah dari sensor yang ada pada robot. Proses ini menggunakan komunikasi serial antara komputer dengan arduino. Hal – hal lain yang dilakukan yaitu seperti proses komunikasi serial dengan menggunakan Arduino IDE dan python atau Microsoft Visual Studio 2015.

7. Analisa dan Pengambilan Data

Setelah melakukan pengujian pada keseluruhan sistem dan memastikan bekerja dengan baik dan hasilnya memenuhi target, maka yang dilakukan selanjutnya adalah pengambilan data yang diperlukan untuk kemudian dianalisa dari data yang telah didapatkan. Analisa yang dilakukan adalah respon sensor, kecepatan robot, *delay* komunikasi serial dan respon permainan.

8. Penyusunan Laporan

Pada tahap akhir ini, hasil pengambilan data dan analisa dimasukkan ke pembahasan. Kemudian, dari apa yang telah dianalisa dapat ditarik beberapa kesimpulan yang menyangkut kinerja dari alat yang dibuat dan memberikan saran untuk memperbaiki kekurangan yang ada, kemungkinan pengembangan, serta penyempurnaan alat di masa mendatang.

3.4 Alat dan Bahan

3.4.1 *Hardware*

1. Arduino Nano328
2. Laptop atau Raspberry Pi 3 B+
3. Motor Linear L12
4. *Flex sensor*
5. Modul *Joystick*
6. Printer 3D

3.4.2 *Software*

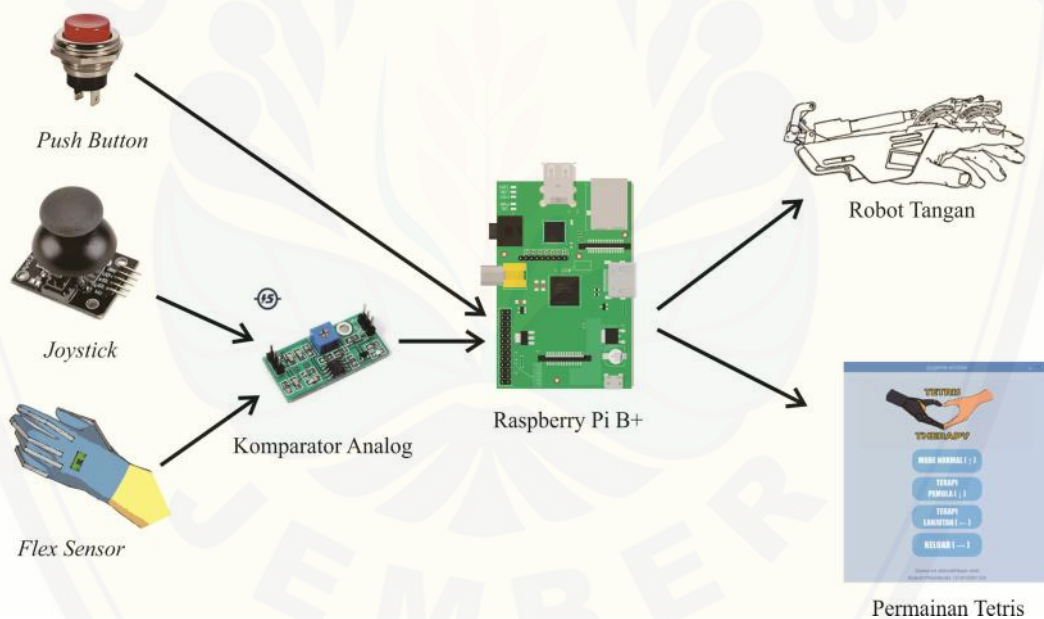
1. Geany

2. Eagle
3. Arduino IDE
4. Photoshop
5. Corel Draw
6. Linux

3.5 Rancangan Sistem

Rancangan dari penelitian dengan judul “Pengembangan Sistem Robot Bantu Terapi Pasien Pasca Stroke Berbasis Permainan Komputer” tersusun atas diagram blok sistem, desain sistem elektronika konsol permainan, desain sistem aplikasi permainan dan *flowchart system*.

3.5.1 Diagram Blok Sistem



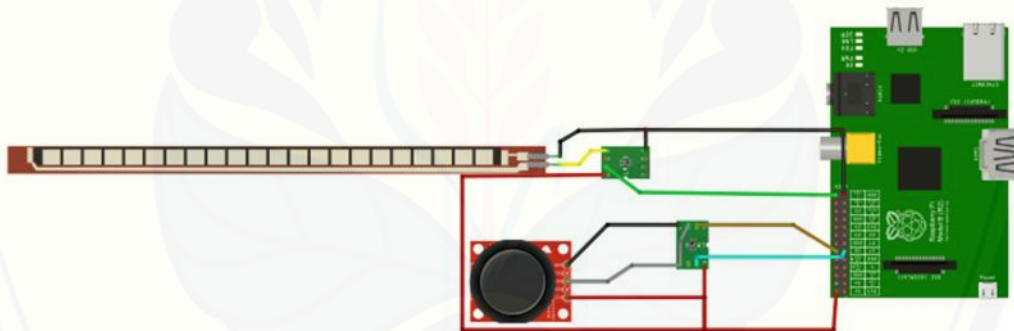
Gambar 3.2 Diagram Blok Sistem

(Sumber: penulis, 2018)

Rancangan sistem dari robot tangan ini memiliki tiga versi *embedded control* yang sudah dikembangkan sebelumnya. Pada penelitian ini robot dikendalikan dengan menggunakan *Flex sensor* yang diletakkan pada tangan pasien yang tidak mengalami stroke. *Flex sensor*, modul *joystick*, dan *push button*

juga digunakan sebagai konsol sebuah permainan pada komputer. Fungsi dari *Flex sensor* yaitu memberikan perintah pada robot untuk membuka tangan dan menggenggam tangan serta untuk memberikan perintah pada permainan agar objek permainan dapat bergerak melompat, menembak dan lain-lainnya seperti fungsi *klick* pada *mouse* komputer. Sedangkan fungsi dari modul *joystick* dan *push button* yaitu untuk memberikan perintah pada objek permainan untuk bergerak berpindah, menentukan sasaran dan lain-lainnya seperti fungsi *cursor* pada mouse komputer. Kedua sensor tersebut akan dikomparasi nilai tegangannya untuk menjadi nilai digital *high* dan *low* kemudian dihubungkan ke permainan yang ada di Raspberry Pi B+ dengan menggunakan pin GPIO.

3.5.2 Desain Elektronika Konsol Permainan



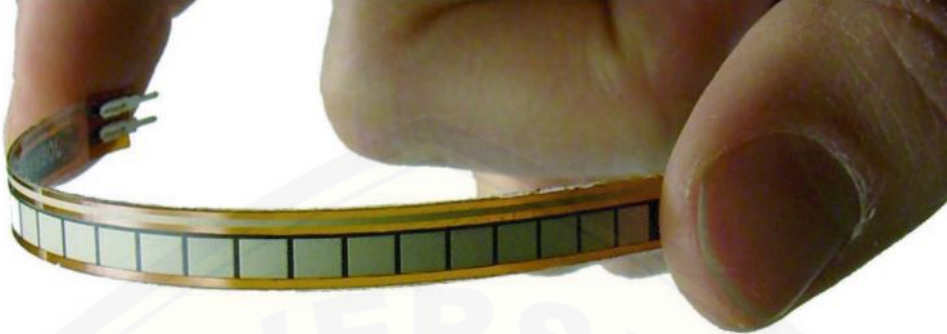
Gambar 3.3 Desain Elektronika Konsol Permainan

(Sumber: penulis, 2018)

3.5.3 Akuisisi Deteksi Kondisi Tangan

Data akuisisi untuk mendeteksi kondisi tangan dalam keadaan membuka atau menggenggam menggunakan *Flex sensor*, sensor ini mampu mengukur aktivitas perpindahan sudut. Sebagaimana ditunjukkan oleh Gambar 3.4 jumlah sensor yang digunakan adalah satu sensor. Sensor ini akan dikalibrasi dengan menggunakan rangkaian komparator dimana kesensitifan sensor akan disesuaikan

dengan gerakan robot tangan ketika membuka dan menggenggam agar permainan dapat dikendalikan oleh pasien terapi.

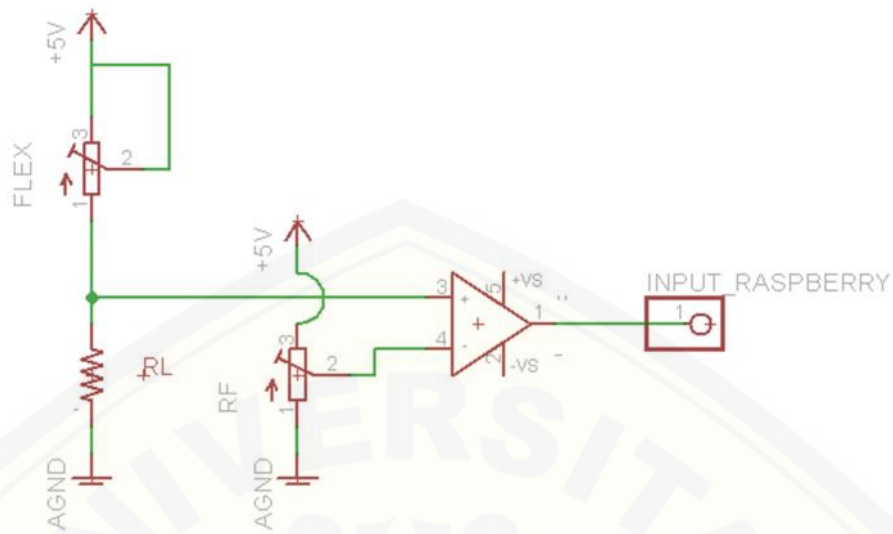


Gambar 3.4 *Flex Sensor*

(Sumber : Spectrasymbol.com, 2018)

3.5.4 Rangkaian pembagi tegangan

Flex sensor tidak dapat dihubungkan langsung dengan arduino, karena sensor ini memiliki nilai *keluaran* berupa nilai resistansi sehingga diperlukan sebuah rangkaian tambahan berupa pembagi tegangan agar dapat diproses oleh arduino dalam bentuk nilai tegangan yang biasa disebut ADC (*Analog to Digital Converter*) seperti pada gambar 3.5 rangkaian pembagi tegangan. Kemudian dalam penelitian ini tidak dibutuhkan sebuah nilai ADC untuk mengendalikan sebuah objek pada permainan, sehingga setelah nilai ADC diterima oleh komparator maka langkah selanjutnya adalah mencari nilai *threshold* yang berfungsi untuk merubah nilai ADC dari sensor menjadi nilai digital 0 atau 1. Namun proses awal sebelum menentukan nilai *threshold* adalah mencari nilai tegangan pada setiap sudut.

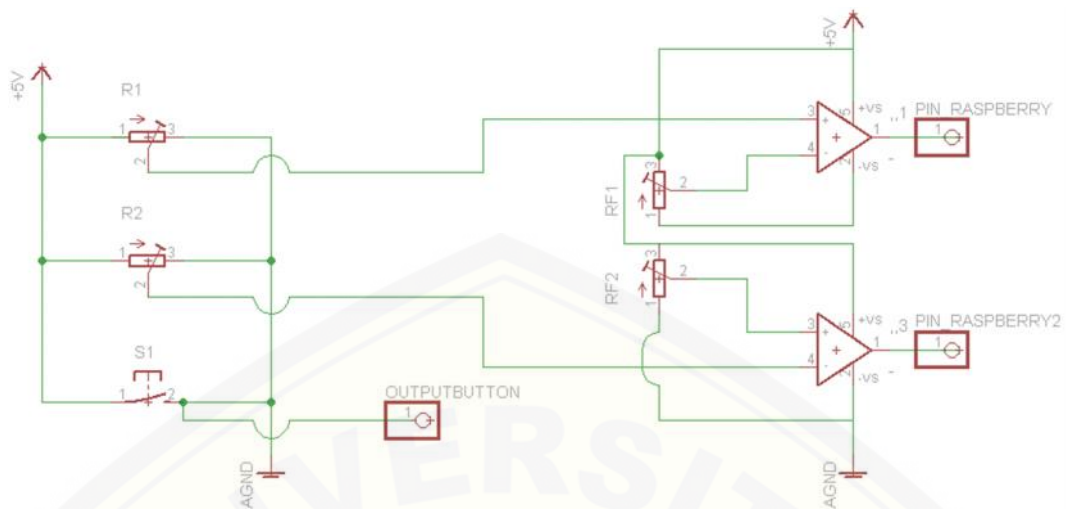


Gambar 3.5 Rangkaian Pembagi Tegangan

(Sumber : Penulis, 2018)

3.5.5 Akuisisi Deteksi Posisi Tangan

Data akuisisi untuk mendeteksi posisi tangan menggunakan modul *joystick*, modul ini memiliki dua sumbu axis yaitu axis-X dan axis-Y. Axis tersebut dapat membantu pasien saat melakukan terapi dengan menggunakan permainan karena dapat menggerakkan permainan untuk bergeser ke kanan-kiri atau ke atas-bawah selama proses terapi menggunakan permainan. *Keluaran* dari modul ini berupa nilai tegangan karena dari setiap axis-nya berupa potensiometer sehingga membutuhkan rangkaian komparator agar dapat diakses langsung pada Raspberry pi yang tidak bisa membaca nilai ADC.



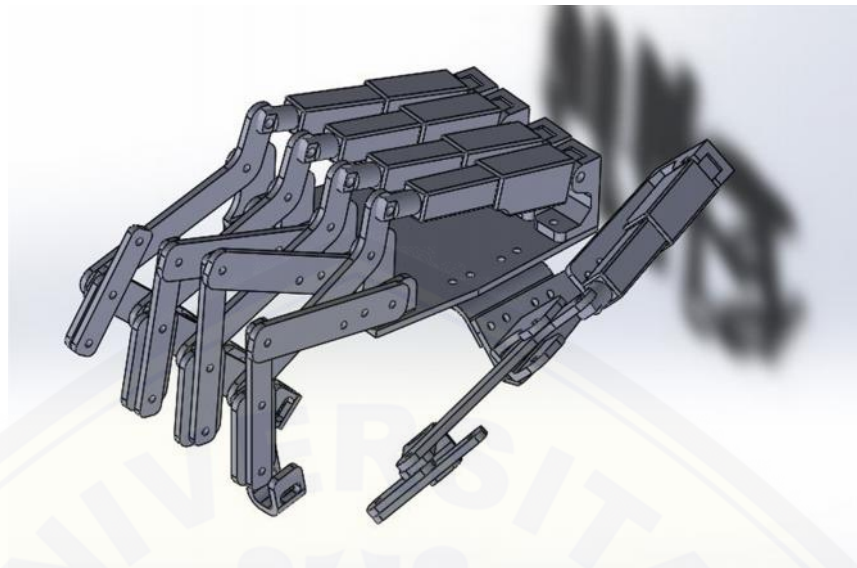
Gambar 3.6 Rangkaian pada Modul *Joystick*

(Sumber : Penulis, 2018)

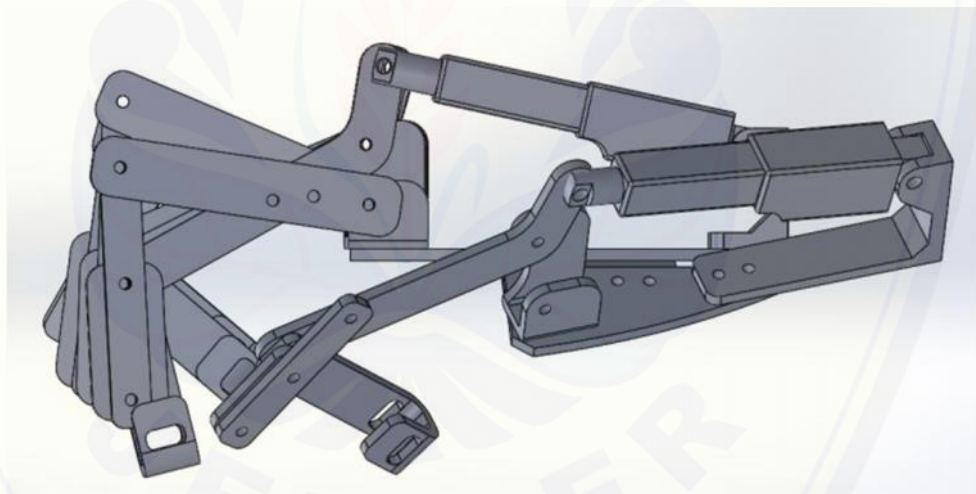
Ketika *joystick* diputar ke arah axis-X maka nilai ADC pada axis-X akan besar dan nilai axis-Y tetap begitu juga sebaliknya. Sedangkan jika diputar ke arah antara axis-X dan axis-Y maka nilai tegangan axis-X dan axis-Y akan mengalami perubahan

3.5.6 Desain Mekanik Robot Tangan

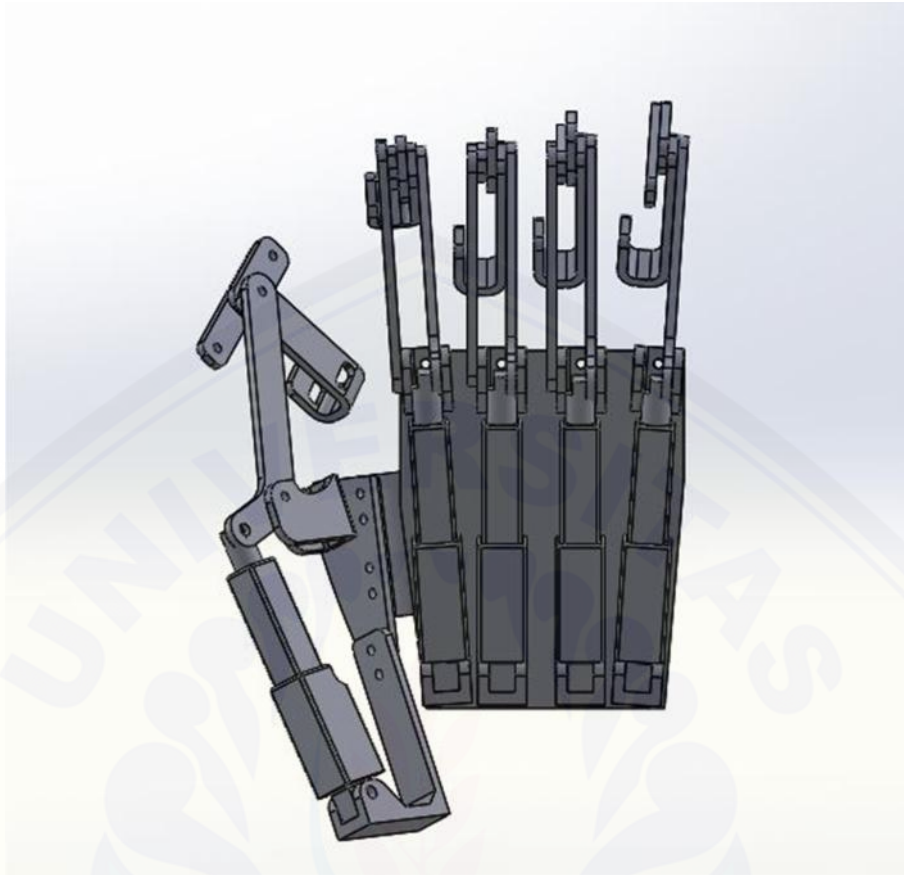
Desain rancangan robot tangan yang digunakan dalam penelitian ini sama seperti desain rancangan robot yang pernah digunakan dalam penelitian sebelumnya. Dimana robot ini digerakkan oleh lima motor linier actuonix L12 yang dikendalikan dengan arduino.



Gambar 3.7 Desain Rancangan Robot Tangan Tampak Kiri Atas
(Sumber : Penulis, 2019)

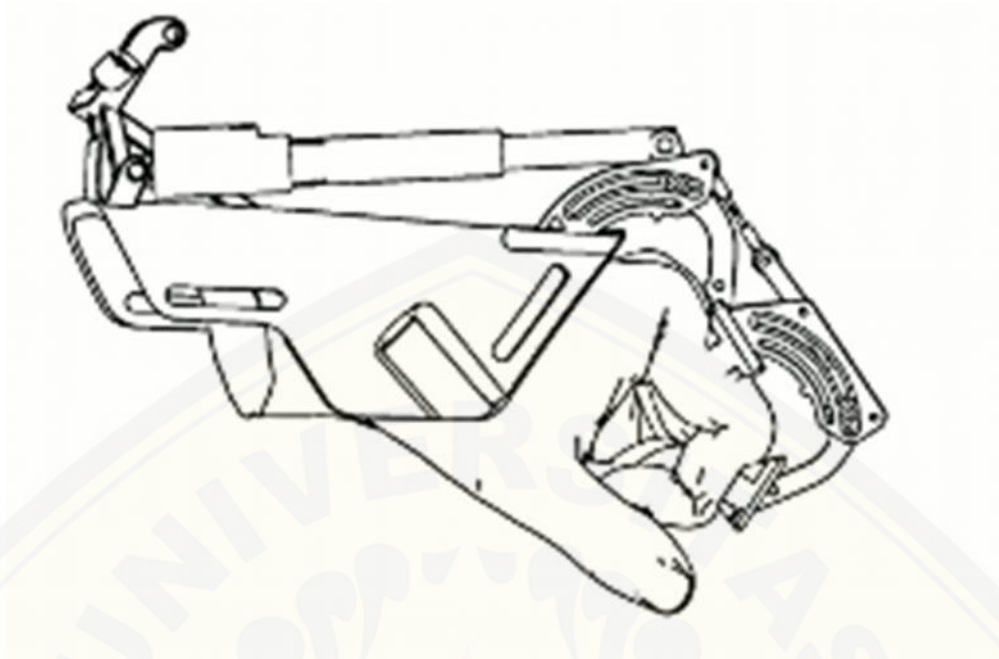


Gambar 3.8 Desain Rancangan Robot Tangan Tampak Kiri
(Sumber : Penulis, 2019)

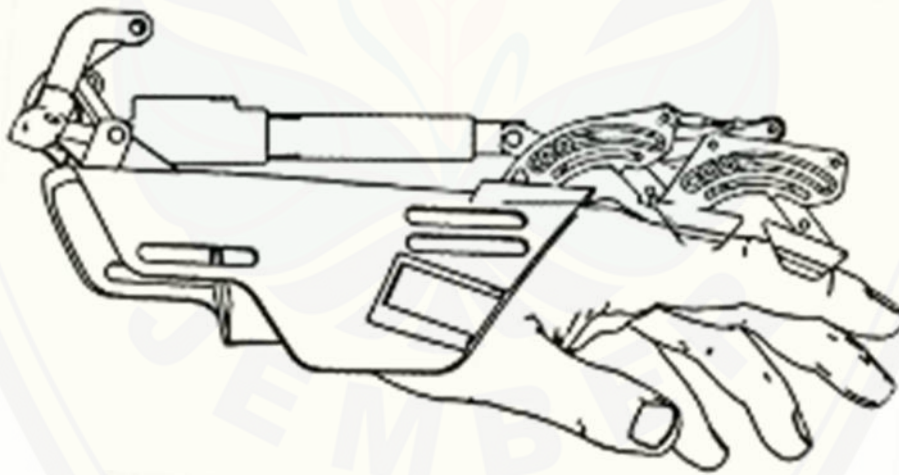


Gambar 3.9 Desain Rancangan Robot Tangan Tampak Atas
(Sumber : Penulis, 2019)

Part komponen robot ini menggunakan bahan dasar filamen yang dicetak atau dibentuk dengan menggunakan printer 3D sehingga robot ini berbeda dengan robot sebelumnya. Robot ini lebih ringan, lebih rapih, dan lebih nyaman dipakai seperti yang terlihat pada gambar 3.7 desain robot tangan. Ketika motor diberi PWM (*Pulse Width Modulation*) 255, maka motor linier akan bergerak maju sehingga robot tangan akan menggenggam seperti pada gambar 3.8 Kondisi robot tangan saat menggenggam. Sedangkan Ketika motor diberi PWM (*Pulse Width Modulation*) 0, maka motor linier akan bergerak mundur sehingga robot tangan akan membuka seperti pada Gambar 3.9 Kondisi robot tangan saat membuka.

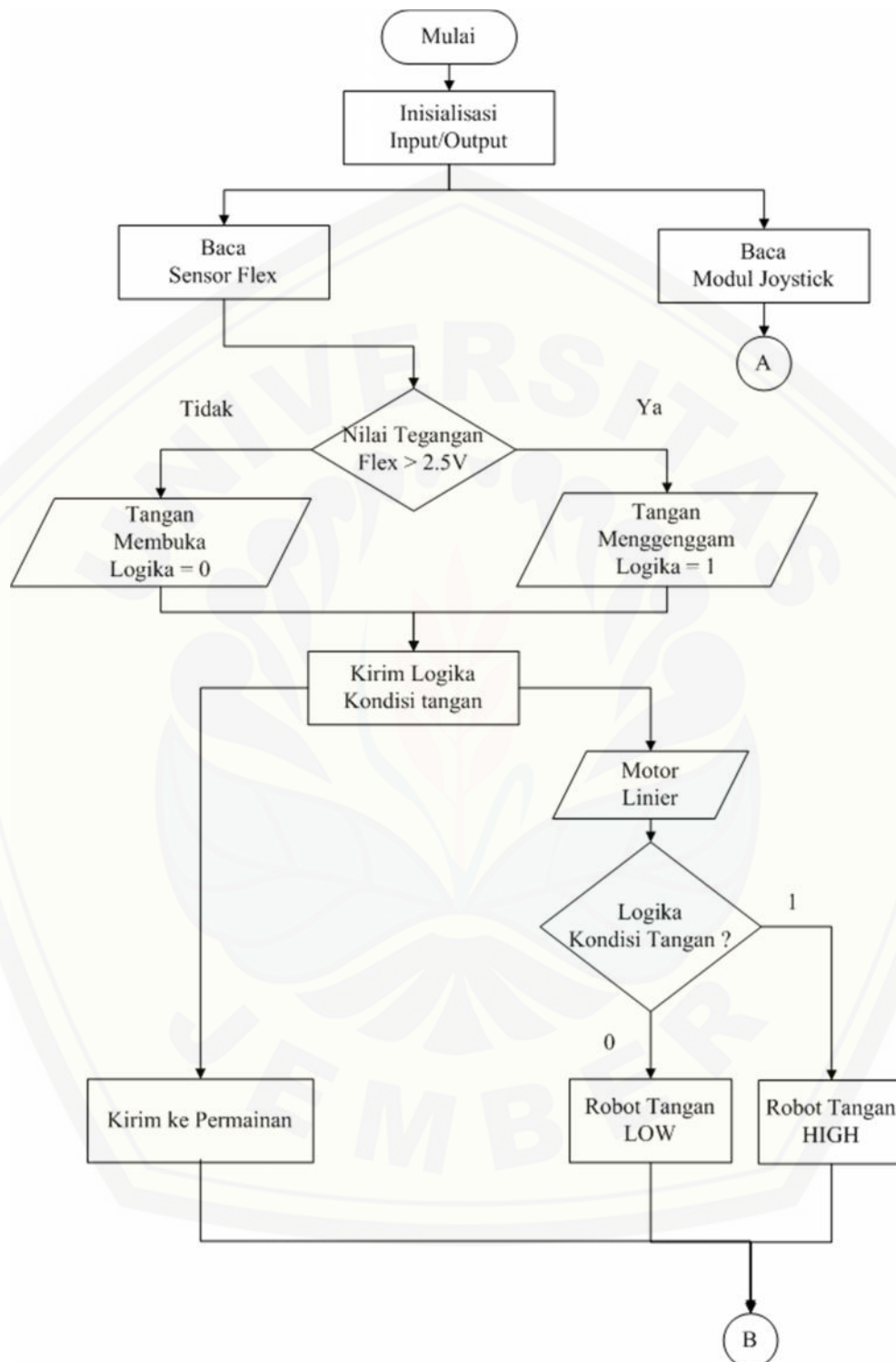


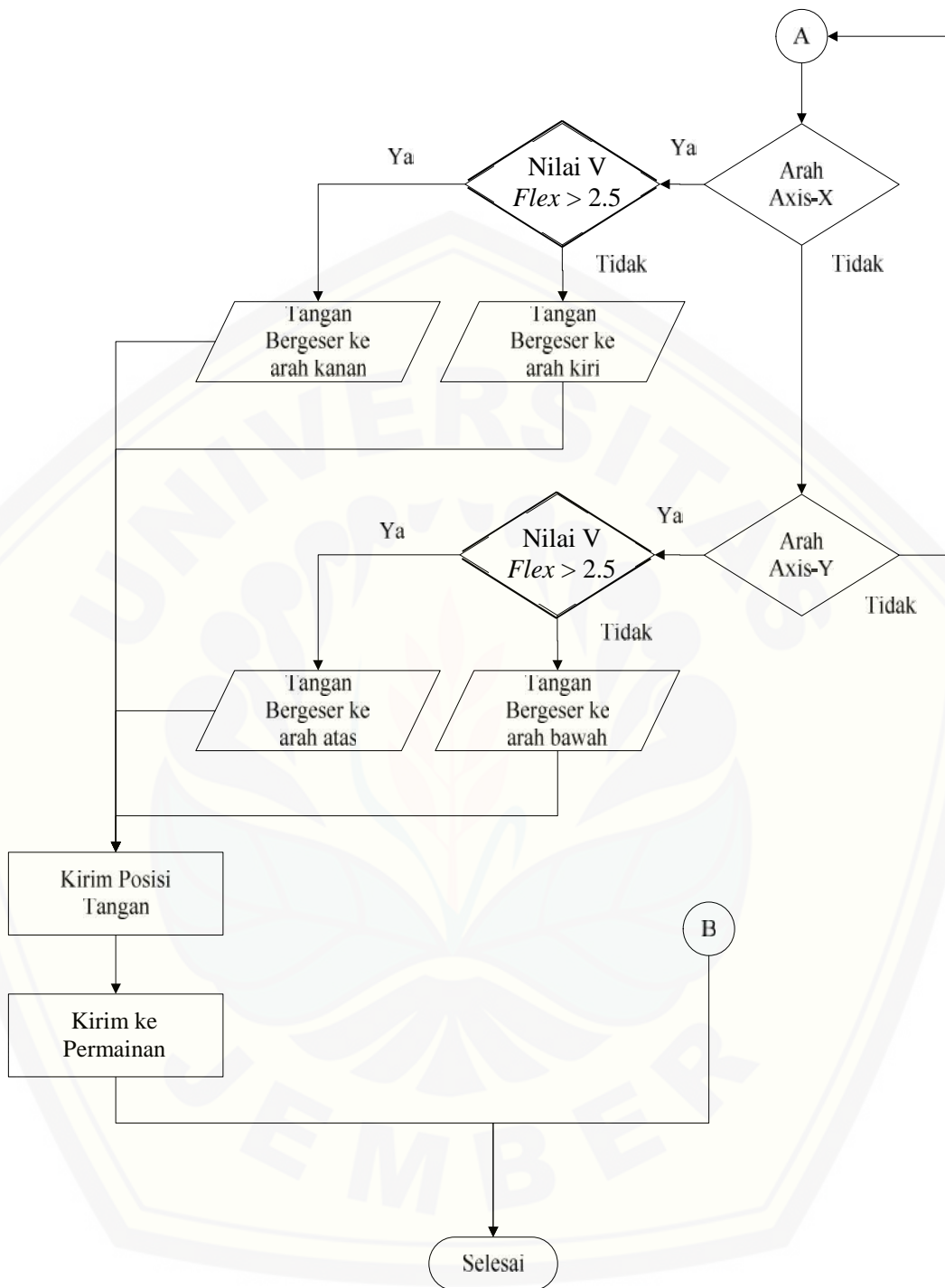
Gambar 3.10 Kondisi robot tangan saat menggenggam
(Sumber: Robot Hand of Hope, 2018)



Gambar 3.11 Kondisi robot tangan saat membuka
(Sumber: Robot Hand of Hope, 2018)

3.5.7 Flowchart Konsol



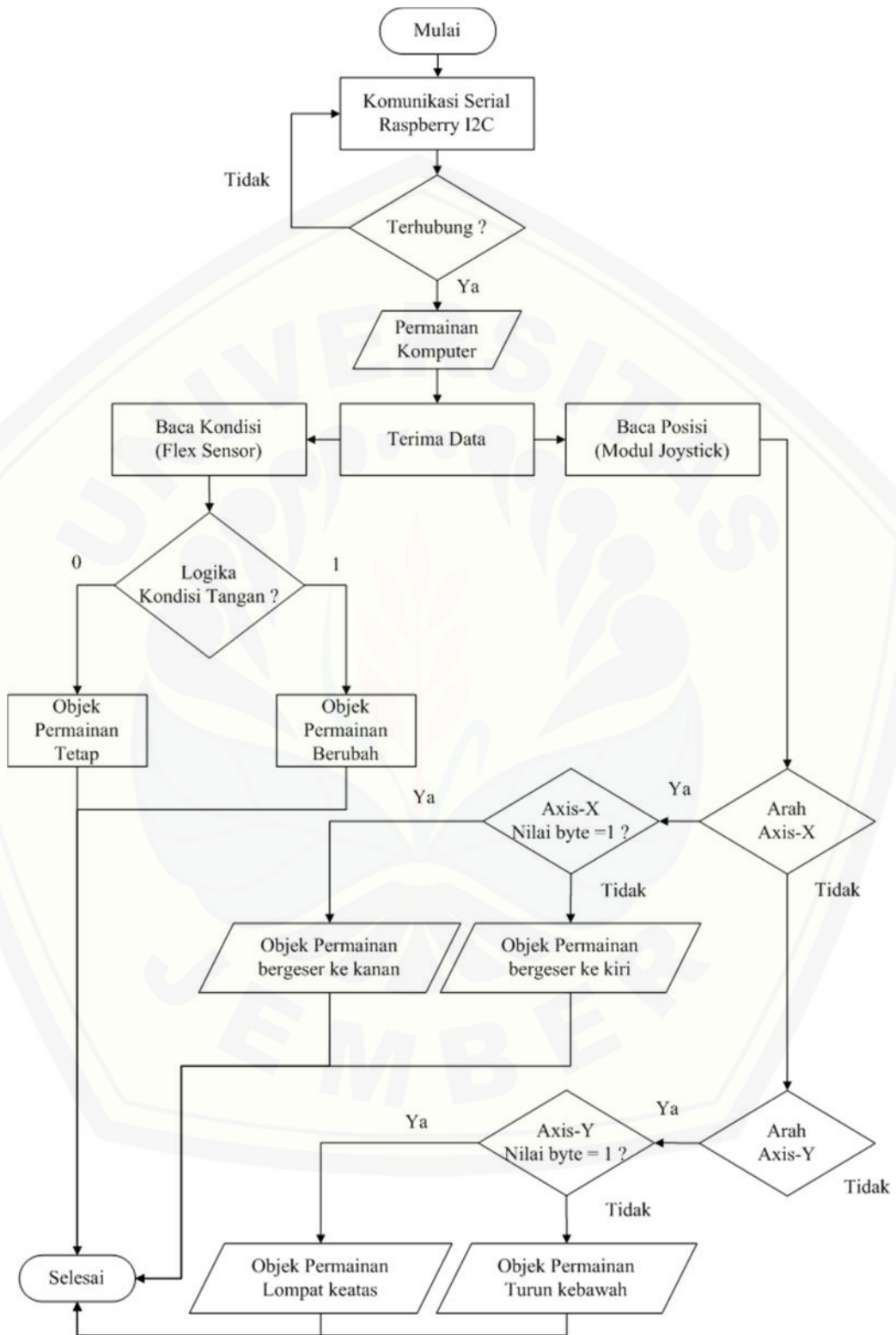


Gambar 3.12 Flowchart Arduino
(Sumber : Penulis, 2018)

Pada gambar 3.12 *flowchart* konsol, dimulai dengan proses inialisasi masukan dan keluaran yang digunakan, kemudian proses pembacaan sinyal oleh *Flex sensor* yang dipasang pada tangan pasien yang tidak mengalami stroke dan modul *joystick* yang terpasang pada robot tangan pada tangan stroke, kemudian akan dilakukan proses kalibrasi *Flex sensor* untuk mendapatkan nilai *treshold*. Jika nilai sensor sudah dapat bekerja sesuai dengan kondisi untuk menggerakkan objek pada permainan maka proses kalibrasi tidak perlu dilakukan karena proses kalibrasi ini bisa dilakukan dengan cara yaitu melalui pemrograman. Proses selanjutnya yaitu proses mengkomunikasikan antara arduino sebagai pengolah data dari sensor dengan Raspberry Pi sebagai sistem operasi dari permainan.

Sedangkan pada gambar 3.13 *flowchart* Raspberry Pi, dimulai dengan proses integrasi antara raspberry pi dengan arduino. Setelah raspberry pi menerima data berupa data kondisi tangan dengan logika 0 atau 1 dan data posisi tangan yang digunakan untuk mengganti *cursor* pada sebuah komputer. Ketika menerima kondisi tangan dengan logika 1, maka objek pada permainan akan berubah dari bentuk awal menjadi bentuk yang lain sedangkan ketika logika 0 maka objek permainan tidak berubah sama sekali. Dan ketika menerima dari data posisi tangan pada gerakan kanan dan kiri maka objek pada permainan juga bergerak ke arah kanan dan kiri sedangkan pada gerakan atas dan bawah, objek permainan juga akan bergerak ke arah atas dan bawah sesuai dengan posisi tangan yang digerakkan.

3.5.8 Flowchart Raspberry Pi



Gambar 3.13 Flowchart Raspberry Pi

(Sumber : Penulis, 2018)

3.5.9 Desain Sistem Aplikasi Permainan Terapi

Pada desain sistem aplikasi permainan terapi ini menggunakan aplikasi python. Aplikasi python ini dapat dijalankan pada operasi Windows, Mac OS X maupun Linux. Dan aplikasi python ini juga dapat didownload secara gratis dan bersifatnya *open sources* hampir sama seperti arduino IDE. Dalam penelitian ini penulis menggunakan standart operasi yang ada pada Raspberry Pi yaitu sistem operasi Rasbian (Raspberry Debian) berbasis Linux. Ada beberapa tahap dalam mendesain sebuah permainan di python yaitu instalasi *library* pygame, proses pemrograman dan menjalankan pada Raspberry Pi.

1. Instalasi *Library* Pygame

Pygame adalah sebuah modul *cross-platform* dari aplikasi python dimana untuk membuat sebuah permainan. Modul ini dirancang untuk menjadi sederhana, mudah digunakan, dan menyenangkan. Python biasa disebut sebagai bahasa pemrograman terbaik untuk pertama kali belajar, dan banyak dipuji karena sintaks yang mudah dipelajari dan kurva belajar yang gradual. Pygame menargetkan agar mudah digunakan oleh permainan *developer* baru dengan pengalaman yang minim dalam membuat dan mengembangkan sebuah permainan pada komputer. Pygame juga memiliki komunitas yang cukup besar sehingga beberapa permainan hasil pengembangan bersifat *open source* dan pygame juga sangat portable sehingga dapat dijalankan pada sistem operasi apapun.

Pada proses instalasi *library* pygame pada python dapat dilakukan pada jendela terminal Linux di Raspberry Pi yaitu dengan cara pilih dan klik pada dekstop>Application>Accessories>Terminal. Kemudian masukan sebuah perintah pada linux `sudo apt-get install python-pygame` setelah selesai proses instalasi buka aplikasi Python kemudian masukan perintah pada python `import pygame` . Jika di *RUN* instruksi tersebut tidak *error* maka instalasi pygame pada python sudah berhasil.

2. Proses Pemrograman

Proses pemrograman permainan pada python hampir sama seperti koding pada aplikasi berbasis *Graphical User Interface* lainnya namun yang

membedakan hanya bahasa pemrogramannya saja. Struktur pemrograman yang paling awal dalam permainan pada python yaitu Permainan *class initialiser* atau inisialisasi seperti `import pygame*, from pygame.local import*, pygame.init()`, dan lain-lainnya. Kemudian struktur *Game Loop* dan permainan *state*. *Game Loop* (disebut juga *main loop*) adalah *loop* di mana kode melakukan tiga hal:

- *Handles Events*
- *Update Game state*
- *Print game state* pada tampilan

Sedangkan *game State* cara mengacu pada serangkaian nilai untuk semua variabel dalam program permainan. Di banyak permainan, *Game state* mencakup nilai-nilai dalam variabel yang melacak kesehatan dan posisi pemain, kesehatan dan posisi musuh, yang menandai telah dibuat di papan, skor, atau giliran siapa. *Game state* biasanya diperbarui sebagai *respons* terhadap peristiwa (seperti klik *mouse* atau penekanan *keyboard*) atau berlalunya waktu, putaran permainan terus-menerus memeriksa dan memeriksa ulang berkali-kali setiap detik untuk setiap peristiwa baru yang telah terjadi. Di dalam *main loop* adalah kode yang terlihat di mana peristiwa telah dibuat (dengan Pygame, ini dilakukan dengan memanggil fungsi `pygame.event.get()`). *Main loop* juga memiliki kode yang memperbarui status permainan berdasarkan peristiwa mana yang telah dibuat. Ini biasanya disebut penanganan *event*.

3. Proses *Run*

Pada proses *Run* ada dua cara yaitu yang pertama secara manual dengan membuka koding permainan pada Python IDLE kemudian pilih *Run* pada Python Shell dan klik *Run Module* (F5). Kemudian cara yang kedua yaitu dengan cara setting Raspberry Pi *autorun* ketika dinyalakan. Cara ini tidak dilakukan pada Python melainkan melalui jendela terminal Linux Raspberry Pi. Dalam metode ini ada beberapa cara yang pertama dengan perintah `rc.local`. `Rc.local` adalah skrip yang akan berjalan di ujung setiap *run* level multiuser dengan perintah pada Linux `$ sudo nano /etc/rc.local`. Metode selanjutnya yaitu `cron`. `Cron`

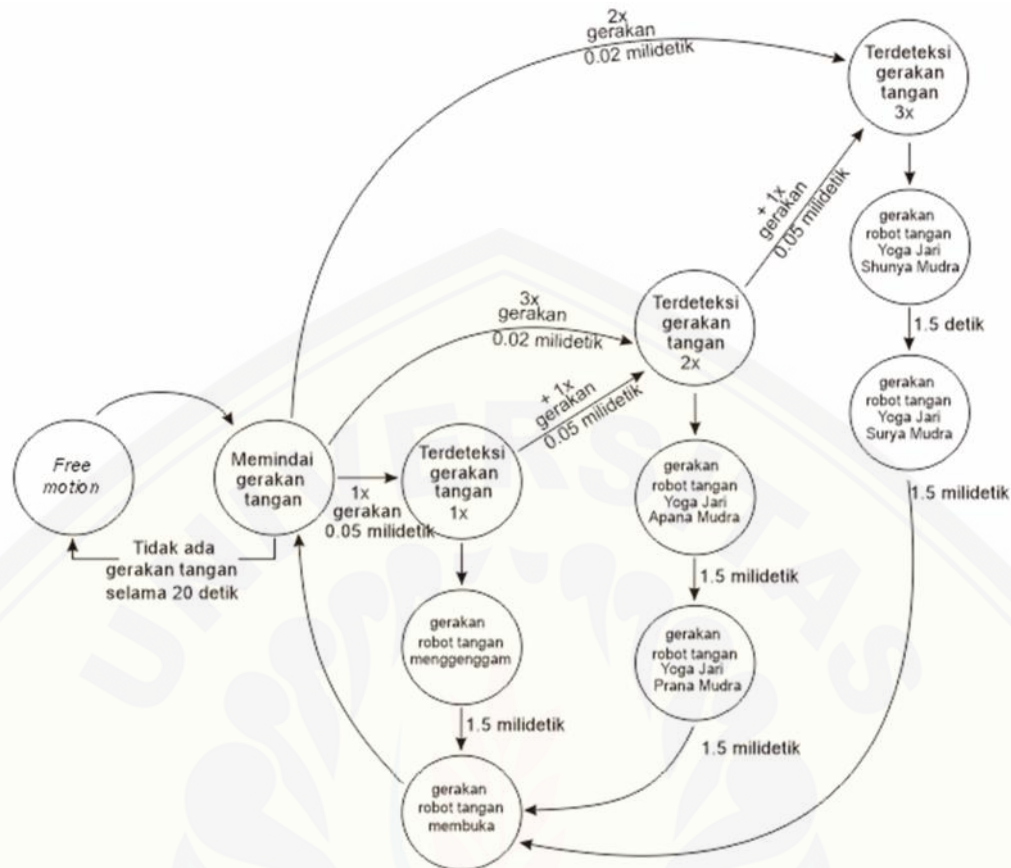
adalah sebuah program yang memungkinkan Anda menjalankan program pada waktu yang ditetapkan. Misalnya. saat boot. Untuk membuka file setup Cron yaitu `$ sudo crontab -e` Di bagian bawah jenis file (jangan gunakan .bin di akhir file program) seperti `@reboot /home/pi/myProgram` dan untuk menjalankan beberapa perintah ketik '&&' di antara program seperti `@reboot sleep 60 && /home/pi/myProgram` maksud dari 'Sleep 60' yaitu menunggu selama 60 detik sebelum program berjalan.

Metode yang terakhir yaitu Daemontools. Daemontools adalah kumpulan alat untuk mengelola layanan UNIX. Layanan untuk mengawasi monitor ini memulai layanan dan me-*restart* layanan jika mati dengan menyiapkan layanan baru. Daemontools mengawasi memastikan program Anda selalu berjalan. Misalnya. jika Anda secara tidak sengaja keluar dari program *openFrameworks* dengan tombol 'esc', itu akan secara otomatis memulai ulang program Anda. Langkah awal untuk menggunakan metode ini dengan menginstall aplikasinya melalui jendela terminal Linux pada Raspberry Pi dengan `$ sudo apt-get install daemontools` kemudian setelah proses instalasi selesai masukan perintah pada jendela terminal Linux `$ sudo mkdir -p /etc/service` untuk memulai setting.

3.6 Metode Penelitian

3.6.1 *Finite State Machine* (FSM)

Finite State Machine (FSM) diaplikasikan untuk mengendalikan gerakan robot tangan. Berikut diagram FSM beserta penjelasannya dan *flowchart* dalam proses pemrograman robot tangan ini :



Gambar 3.14 diagram FSM

(sumber : Penulis, 2019)

a. *Free motion*

Free motion adalah gerakan bebas robot tangan saat awal dinyalakan. Gerakan ini akan berlangsung selama kurang dari 10 detik. Gerakan bebas ini difungsikan untuk pasien agar dapat menyesuaikan tangan dengan robot tangan yang dikenakan. Gerakan *free motion* ini dibagi menjadi beberapa gerakan dimana gerakan *free motion* sendiri merupakan gabungan dari semua gerakan yang terdapat dalam beberapa *state*.

Free motion akan kembali dijalankan apabila pada proses pemindaian gerakan tangan pasien yang tidak mengalami stroke tidak terdeteksi adanya gerakan dalam waktu 10 detik. Namun, ketika dalam waktu 10 detik tersebut terdapat gerakan dari tangan pasien yang tidak mengalami stroke robot tidak akan menjalankan *Free motion* namun akan masuk dalam *state* yang sudah terdesain.

b. Memindai gerakan tangan

Memindai gerakan tangan adalah suatu tindakan robot tangan untuk mengetahui gerakan tangan pasien yang tidak mengalami stroke yang difungsikan untuk menggerakkan robot tangan pasien yang mengalami stroke.

c. Terdeteksi gerakan tangan 1x

Merupakan *state* yang didalamnya terdapat sebuah perintah gerakan robot. Dalam *state* ini terdapat dua gerakan yaitu gerakan membuka jari tangan secara keseluruhan dan gerakan menggenggam jari tangan secara keseluruhan. *State* ini tidak ada gerakan modifikasi seperti gerakan yoga jari tangan.

d. Terdeteksi gerakan tangan 2x

Merupakan *state* yang didalamnya terdapat sebuah perintah gerakan robot. Dalam *state* ini terdapat dua gerakan modifikasi senam yoga jari tangan yaitu :

- Gerakan senam yoga jari tengah dan manis dipegang oleh ibu jari (Apana Mudra), gerakan ini membantu mengatur ginjal dan usus sehingga membantu dalam membersihkan tubuh anda dari racun dan mencegah sembelit. Manfaat lain adalah mencegah mual dan muntah serta sensasi terbakar saat buang air kecil. Posisi ini juga sangat berguna bagi penderita diabetes karena mengatur kadar glukosa darah.



Gambar 3.15 contoh gerakan senam jari tangan Apana Mudra

(Sumber : h2healthandhappiness.com, 2019)

- Gerakan senam yoga jari tangan (Prana Mudra), ujung jari manis dan kelingking menyentuh ibu jari anda sementara sisa jari tetap lurus.

Gerakan ini membantu memperkuat pikiran anda dan dapat meningkatkan motivasi. Hal ini juga meningkatkan penglihatan anda, mengurangi kelelahan dan depresi, dan meningkatkan sistem kekebalan tubuh.



Gambar 3.16 contoh gerakan senam jari tangan Prana Mudra
(Sumber : h2healthandhappiness.com, 2019)

Gerakan tersebut akan dijalankan secara berurutan kemudian robot akan kembali ke gerakan membuka jari tangan. Setelah itu, robot tangan akan kembali melakukan proses memindai gerakan tangan selama 10 detik. Jika tidak terdeteksi maka robot akan melakukan *free motion*.

e. Terdeteksi gerakan tangan 3x

Merupakan *state* yang didalamnya terdapat sebuah perintah gerakan robot. Dalam *state* ini terdapat tiga gerakan modifikasi senam yoga jari tangan yaitu :

- Gerakan senam yoga jari tangan Shunya Mudra, gerakan ibu jari memegang jari tengah ke bawah sementara sisa jari tetap lurus. Gerakan ini efektif dalam kasus sakit telinga dan berfungsi sebagai penguat keyakinan. Posisi gerakan ini juga dapat mencegah perasaan kosong, tinnitus dan bahkan vertigo.



Gambar 3.17 contoh gerakan senam jari tangan Shunya Mudra

(Sumber : h2healthandhappiness.com, 2019)

- Gerakan senam yoga jari tangan Surya Mudra, gerakan ibu jari memegang jari manis turun sementara sisa jari tetap lurus. Gerakan ini membantu mengaktifkan tiroid, menurunkan berat badan dan merevitalisasi sistem pencernaan.

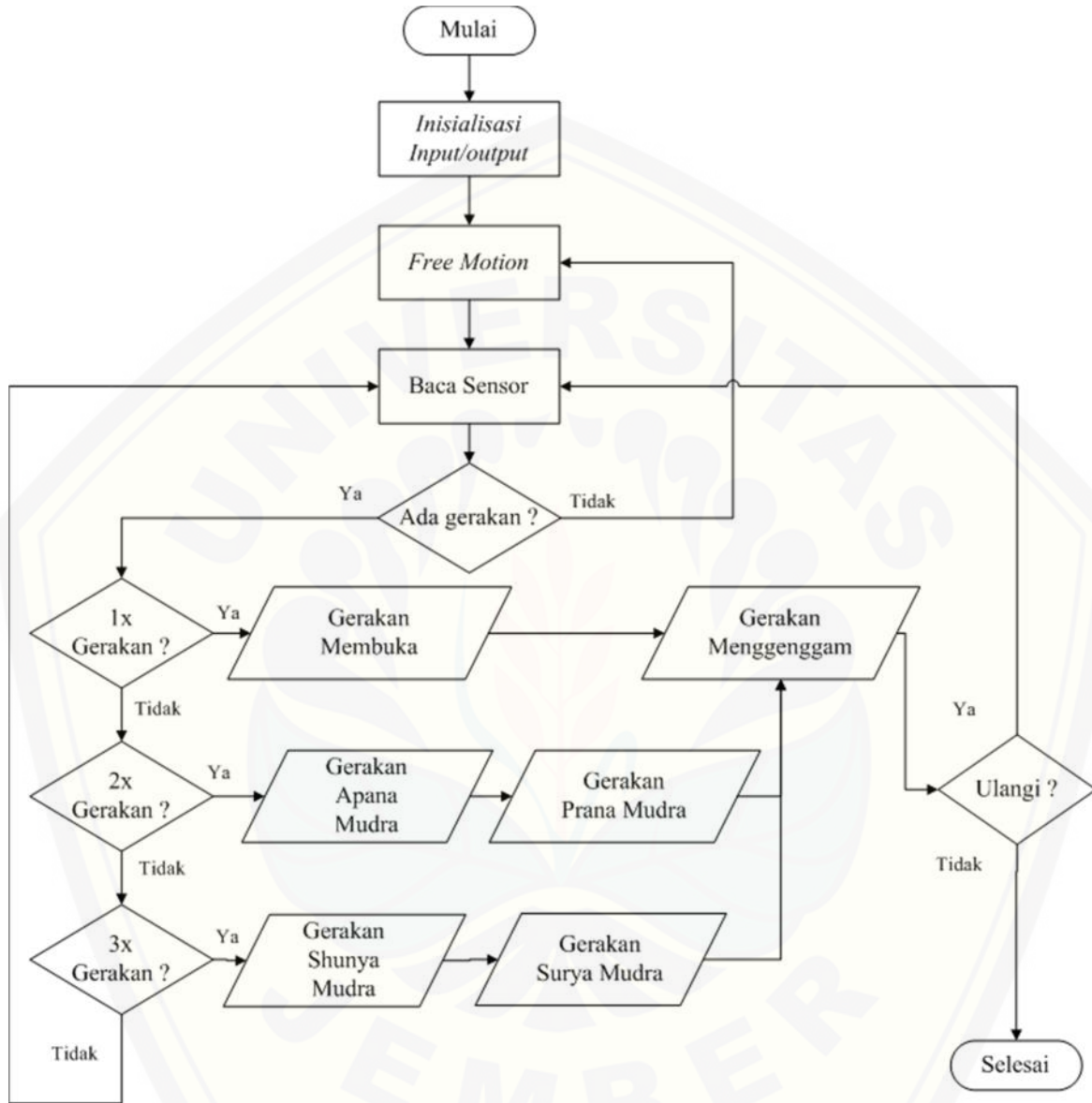


Gambar 3.18 contoh gerakan senam jari tangan Surya Mudra

(Sumber : h2healthandhappiness.com, 2019)

Gerakan tersebut akan dijalankan secara berurutan kemudian robot akan kembali ke gerakan membuka jari tangan. Setelah itu, robot tangan akan kembali melakukan proses memindai gerakan tangan selama 10 detik. Jika tidak terdeteksi maka robot akan melakukan *free motion*.

3.6.2 Flowchart FSM

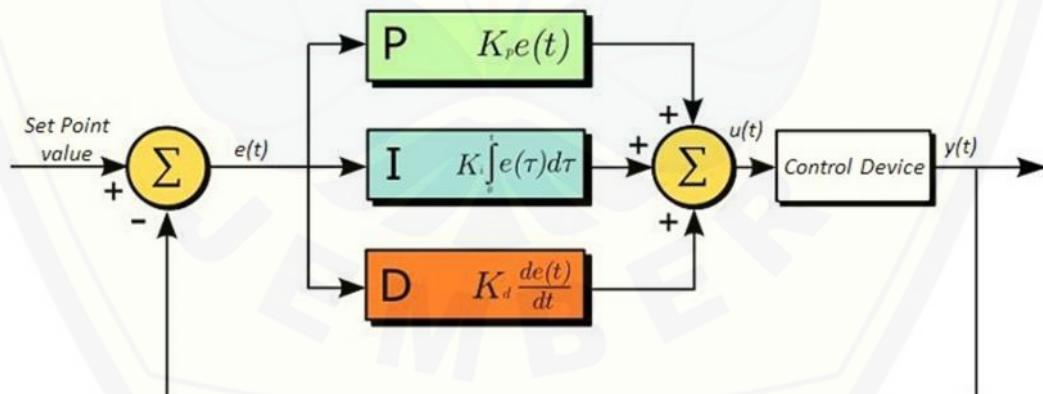


Gambar 3.19 Flowchart FSM

(Sumber : Penulis, 2019)

3.6.3 Sistem kontrol PID

Sistem kontrol PID merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik dari sinyal umpan balik yang diterima. Sistem kontrol PID tersusun atas tiga komponen yaitu Proporsional (P), Integral (I), dan Derivatif (D). Kontrol PID secara aktif mengontrol sistem sehingga menahannya pada *set point* dengan menghasilkan sinyal *error* yang pada dasarnya adalah perbedaan antara *set point* dan nilai saat ini. Ketiga kontrol terkait dengan sinyal *error* tergantung waktu; paling sederhananya, ini dapat dianggap sebagai berikut: Proporsional bergantung pada *error* saat ini, Integral bergantung pada akumulasi *error* masa lalu, dan Derivatif adalah prediksi *error* di masa depan. Hasil masing-masing kontrol kemudian dimasukkan ke dalam penjumlahan tertimbang, yang kemudian menyesuaikan keluaran dari rangkaian $u(t)$. Keluaran ini diumpankan ke perangkat kontrol, nilainya dimasukkan kembali ke dalam rangkaian, dan prosesnya dimungkinkan untuk secara aktif menstabilkan keluaran sirkuit untuk mencapai dan menahan pada nilai *set point*. Diagram blok di bawah ini menggambarkan sistem kontrol PID.



Gambar 3.20 Diagram Blok Sistem Kontrol PID

(Sumber: Thorlabs, 2018)

Keluaran dari kontrol PID $u(t)$ didapat dari :

$$u(t) = K_p e(t) + K_i \int_0^t e(t) d\tau + K_d \frac{d}{dt} e(t) \dots\dots\dots(3.1)$$

Dimana,

K_p = Penguat Proposional

K_i = Penguat Integral

K_d = Penguat Derivatif

$e(t)$ = *Set point* – nilai saat ini

3.7 Pengujian Robot Tangan

Dalam penelitian ini, pengujian robot tangan terbagi menjadi beberapa bagian antara lain mencakupi pengujian *Flex Sensor*, pengujian modul *joystick*, pengujian permainan tetris, pengujian integrasi konsol permainan, pengujian sistem kontrol PID, pengujian metode FSM, dan pengujian kelayakan robot.

3.7.1 Pengujian *Flex Sensor*

Pengujian *Flex sensor* ini menggunakan variasi sudut tekuk sensor yang terlihat pada gambar 3.21 untuk mengetahui nilai resistansi dan nilai tegangan yang muncul dari keluaran rangkaian pembagi tegangan sehingga dihasilkan persamaan berikut :

$$V_{output} = \left(\frac{R_{trimpot}}{R_{sensor} + R_{trimpot}} \right) \cdot 5 \text{ volt} \dots\dots\dots(3.2)$$



Gambar 3.21 Proses Pengujian *Flex Sensor* Terhadap Sudut

Sedangkan untuk mencari nilai ADC dapat menggunakan serial monitor pada Arduino IDE dan bisa juga menggunakan perhitungan ADC seperti pada persamaan berikut :

$$\text{Nilai ADC} = \left(\frac{V_{\text{output}}}{5\text{volt}} \right) \cdot 1024 \dots\dots\dots(3.3)$$

3.7.2 Pengujian Modul Joystick

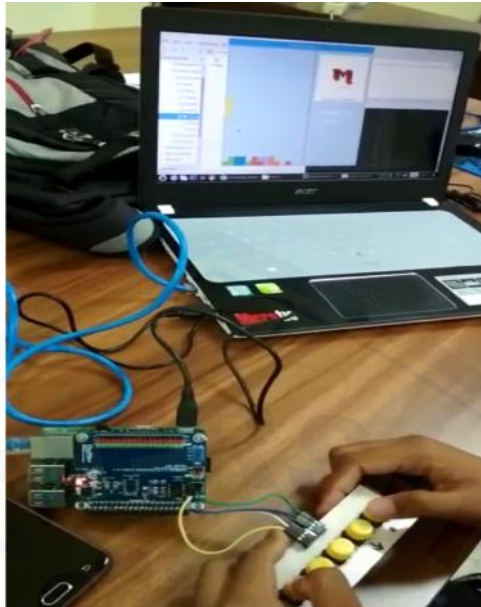
Pengujian modul *joystick* ini hampir sama dengan pengujian *Flex sensor* yaitu dengan membaca nilai tegangan pada pin keluaran axis-X dan axis-Y. Pengujian axis-X dan axis-Y ini tidak hanya ketika *joystick* digerakkan ke arah Xatas, Xbawah, Ykanan dan Ykiri tetapi juga digerakkan ke arah antara Xatas dengan Ykanan, Xatas dengan Ykiri, Xbawah dengan Ykanan, Xbawah, dan Ykiri seperti pada gambar 3.22 dibawah, sehingga mendapatkan data ADC yang nanti dikirim secara serial untuk menggerakkan objek pada permainan.



Gambar 3.22 Proses Pengujian joystick dengan Kombinasi Arah

3.7.3 Pengujian Permainan Tetris

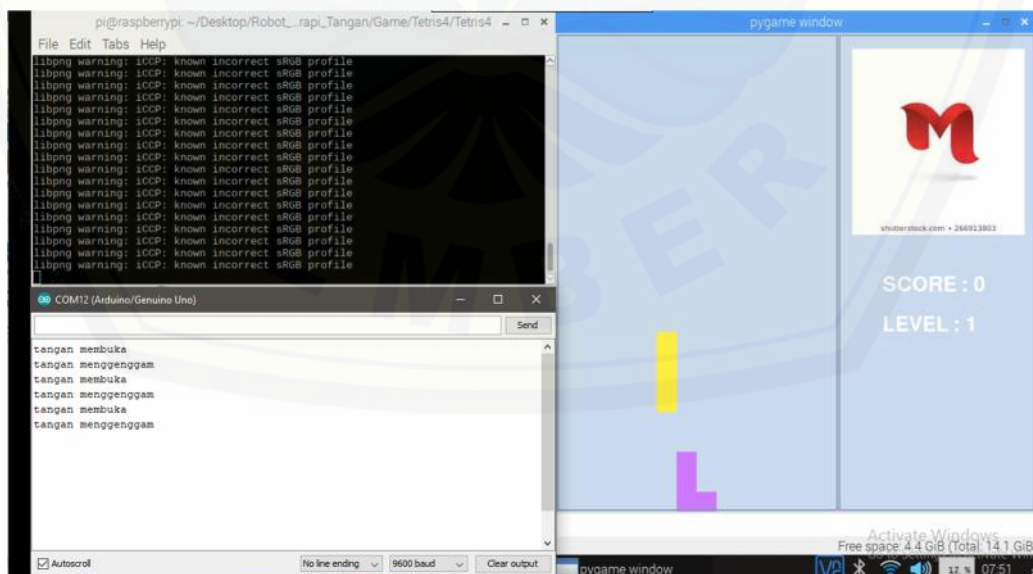
Pada pengujian ini, penulis ingin mengetahui seberapa lama permainan tetris dapat dimainkan. Penulis mendapatkan beberapa responden untuk memainkan permainan ini selama selang waktu 10 menit dengan mencoba 3 mode yang berbeda yaitu mode normal, mod terapi pemula dan mode terapi lanjutan.



Gambar 3.23 Proses Pengujian Permainan Tetris pada Raspberry Pi 3 B+

3.7.4 Pengujian Integrasi Konsol Permainan

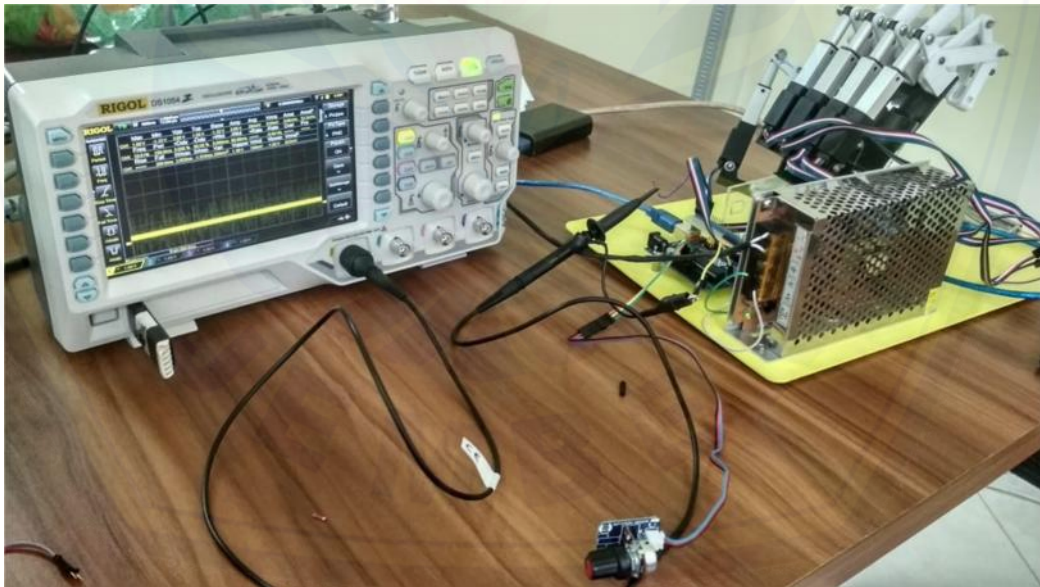
Pada pengujian ini penulis ingin mengetahui seberapa besar pengaruh antara nilai *delay* dengan respon *game* pada Python. Pengujian ini bertujuan untuk mencari nilai *delay* yang memberi respon yang tepat pada objek permainan sehingga permainan dapat dimainkan dengan nyaman oleh pasien selama proses terapi.



Gambar 3.24 Hasil Pengujian Integrasi Robot Sebagai Konsol Dengan Sebuah Permainan Tetris

3.7.5 Pengujian Sistem Kontrol PID Motor Linier L12

Pada pengujian ini terdapat dua bagian yaitu pengujian sistem kontrol pid motor linier actuonix L12 tanpa beban tangan dan pengujian sistem kontrol pid motor linier actuonix L12 dengan beban tangan. Sistem kontrol PID untuk mengatasi apabila jari pada pasien yang mengalami stroke susah digerakkan (kaku) sehingga motor linier akan berhenti bergerak sebelum waktu yang sudah ditentukan. Dengan metode ini diharapkan agar motor linier terus bergerak hingga dorongan yang diinginkan $u(t)$ namun dalam batasan beban maksimal motor linier yaitu sebesar 40 Newton atau sebesar 4.07 Kilogram. Dalam metode ini metode sistem kontrol PID ini menggunakan fitur yang ada pada spesifikasi motor linier L12 sehingga penulis tidak mendesain sendiri sistem kontrol PID. Penulis hanya mencari waktu terbaik saat keadaan motor linier *rise time* dan *fall time* yang dimaksudkan agar robot dapat bergerak sesuai dengan waktu yang sudah ditentukan dan agar robot dapat bergerak seirama dengan permainan tetris ketika objek tetris berganti posisi.



Gambar 3.25 Proses Pengujian Sistem Kontrol PID Motor Linier Actuonix L12
Tanpa Beban Tangan

3.7.6 Pengujian Metode *Finite State Machine* (FSM)

metode FSM diaplikasikan unuk mengontrol pergerakan robot tangan. Ketika robot tidak mendeteksi posisi tangan normal pasien stroke selama 10 detik, maka robot tangan akan melakukan *wander*. Ketika robot mendeteksi posisi tangan sehat pasien, maka robot akan bergerak sesuai pergerakan posisi tangan sehat pasien yang terpasang sensor *flex*. Dan ketika robot mendeteksi pergerakan posisi tangan sehat pasien lebih dari satu kali, maka robot akan melakukan pergerakan yang berbeda saat robot mendeteksi pergerakan posisi tangan sehat sebanyak satu kali. Tujuan dari penerapan metode FSM ini pada robot tangan yaitu agar pasien dapat memperoleh banyak variasi pergerakan robot selama proses terapi, sehingga tangan pasien tidak hanya diterapi membuka dan menggenggam saja.

3.7.7 Pengujian Kelayakan Robot

Tahapan terakhir yaitu tahapan pengujian robot tangan. Robot akan dipasang pada tangan orang normal dan tangan orang penderita stroke. Pengujian pada orang normal atau orang yang tidak mengalami stroke dimaksudkan sebagai referensi data skor maksimal pada permainan. Sedangkan pengujian pada penderita stroke merupakan pengujian yang akan dilakukan penelitian secara bertahap terhadap pasien penderita stroke. Setiap responden akan diberi kuesioner untuk diisi sesuai dengan apa yang merasakan ketika menggunakan robot tangan.

BAB 5. PENUTUP

Berdasarkan data hasil pengujian yang telah dilakukan mengenai Pengembangan sistem robot bantu terapi pasien pasca stroke berbasis permainan komputer dapat ditarik kesimpulan dan saran yang berguna untuk penelitian lebih lanjut.

5.1 Kesimpulan

Berdasarkan data hasil pengujian didapatkan kesimpulan bahwa:

1. Metode *Finite State Machine* (FSM) merupakan metode kontrol yang menggambarkan tingkah laku yang terdiri dari *State*, *Event*, dan *Action*. Dimana *State* yang terhubung oleh sensor, *Event* merupakan kondisi awal gerakan robot tangan dan *Action* merupakan gerakan setelah salah satu *state* aktif dan dapat diketahui bahwa proses perpindahan setiap *state* sangat cepat, proses implementasinya juga mudah dan cepat serta dalam proses *debugging* juga lebih mudah karena telah dipecah menjadi bagian-bagian yang lebih kecil dan lebih meminimalkan proses pemrogramannya. Kemudian dilihat bahwa *state* dapat aktif dalam yang beraturan maupun dalam waktu yang tidak beraturan tergantung dari proses pemindaian sensor.
2. Hasil perbandingan pengujian sistem kontrol PID pada motor linier L12 terhadap waktu kontrol. Waktu kontrol tidak mempengaruhi proses *force* motor linier, hal ini dikarenakan adanya pengaruh dari mekanik setiap jari tangan robot dan adanya pengaruh dari tangan yang menggunakan robot tangan tersebut. Sehingga waktu kontrol yang dapat digunakan untuk mengantisipasi *delay* yang terlalu lama saat keadaan *force* motor minimal ke maksimal yaitu waktu kontrol yang semakin pendek.
3. Proses integrasi robot tangan dengan permainan Tetris menggunakan metode penggabungan program dalam Raspberry Pi dengan bahasa pemrograman Python, hal ini sangat efektif karena tidak membutuhkan lagi komponen dari luar untuk membantu proses kontrol robot atau proses

kontrol permainan. Dengan menggunakan *pin* Gpio pada Raspberry Pi, robot dan permainan dapat dijalankan dalam bersamaan karena Raspberry Pi dapat menjalankan beberapa program dalam waktu yang bersamaan.

5.2 Saran

Pada penelitian tentang Pengembangan sistem robot bantu terapi pasien pasca stroke berbasis permainan komputer terdapat beberapa saran untuk penelitian selanjutnya yaitu :

1. Desain mekanik robot yang perlu diperbaiki karena masih belum dapat menggenggam secara maksimal.
2. Permainan dapat dikembangkan lagi dan dapat menambahkan beberapa permainan klasik yang menyenangkan lainnya seperti Mole shooter, Super Mario, dan lainnya.

DAFTAR PUSTAKA

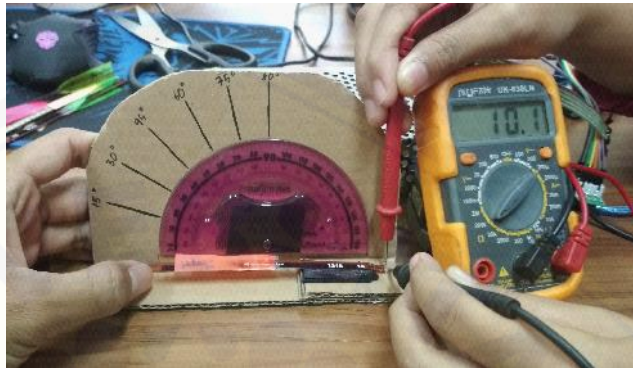
- Afkar, Rifqi. 2018. *Rancang Bangun Sistem Elektronik Robot Tangan Untuk Mendukung Pemulihan Penyandang Disabilitas Tangan Berbasis Genuino 101* [skripsi]. Jember (ID): Univeritas Jember.
- Bagaimana mulai belajar Pygame, Diakses pada tanggal 25 Oktober 2018 <https://gamedevelopment.tutsplus.com/id/tutorials/how-to-learn-pygame--cms-24184>
- Katsuhiko Ogata (2009). *Modern Control Engineering* (5th Edition). Prentice Hall
- Wagner F, Schmuki R, dkk. (2006). *Modeling Software with Finite State Machines - A Practical Approach*. Boca Raton: Taylor & Francis Group
- Phillips, Charles (1998). *Sistem Kontrol : dasar-dasar*. Prentice Hall
- Kelly, Sloan. (2016). *Python, Pygame and Raspberry Pi Game Development*. Ontario. Apress
- PC Mouse Made with Arduino Uno adn *Joystick*, Diakses pada tanggal 22 Oktober 2018 <https://www.instructables.com/id/PC-Mouse-Made-With-Arduino-Uno-and-Joystick/>
<https://circuitdigest.com/microcontroller-projects/arduino-angry-bird-game-controller-with-Flex-sensor>
- Samudra, H. P. 2017. *Rancang Bangun Robot Tangan Untuk Terapi Stroje Menggunakan Flex Sensor Berbasis Arduino* [skripsi]. Jember (ID): Univeritas Jember.
- Wildana, I. G. 2017. *Rancang Bangun Robot Tangan Berbasis EMG Menggunakan Extreme Learning Machine* [skripsi]. Jember (ID): Univeritas Jember.
- Rehab-robotics. (2016). *Hand of Hope*, Retrieved from <https://www.rehab-robotics.com/>
- K.Y. Tong, N.S.K. Ho, X.L. Hu, K.L. Fung, X.J. Wei, W. Rong, et al. (2011). An EMG-driven Exoskeleton Hand Robotic Training. IEEE International Conference on Rehabilitation Robotics, 1-5

- Satriyo, M.B. (2017). *Penggunaan Metode Finite State Machine (FSM) untuk Mengontrol Robot Sepak Bola Beroda* [skripsi]. Jember (ID) :Universitas Jember.
- Levine, P.G. (2011). *Stronger After Stroke: Panduan Lengkap dan Efektif Terapi Pemulihan Stroke*. Penerbit Etera, Jakarta
- Negara, (2016). *Buku Ajar Sistem Kontrol Jilid 1*. Universitas Jember, Jember
- Putra, Y.S., & Muslim, M.A. (2013). *Game Chicken Roll dengan menggunakan metode Forward Chaining*. Jurnal *eccis*, 7(1), 41-46
- Martono, K.T. (2015). *Pengembangan Game dengan Menggunakan Game Engine Game Maker*. Jurnal *sistem Komputer*, 5(1), 23- 24.
- Nugroho, A.K, & Herianto. (2016). *Pengembangan Alat bantu Rehabilitasi Pasien Pascastroke Berbasis Virtual Reality*. Jurnal *Teknik Industri*, 11(1), 45-46.
- Zakiyah L & Mahtarami A. (2015). *Games Berplatform Android untuk Terapi Pasce Stroke*. Jurnal *Seminar Nasional Informatika Medis*, 6(1), 13-15.
- Susilo, Wilhelmus Hary dan Limakrisna Namdan. (2012). *Cermat Menyusun Kuisoner Penelitian Ilmu Keperawatan*. Penerbit Trans Info Media. Jakarta.
- wikipedia, Diakses pada tanggal 22 April 2019 <https://id.wikipedia.org/wiki/Tetris>
- Rdwan, Adipradana B, & Adji Syahdana W. 2005. *Optimasi Permainan "Tetris" dengan pendekatan Algoritma Greedy da Algoritma Brute Force*. Jurnal *Teknik Informatika*. 1(3).
- Actuonix Motion Device. (2019). *L12 Micro Linier Actuators* Retrived from <https://www.actuonix.com/L12-Micro-Linear-Actuators-s/1821.htm>
- SparkFun Electronics. (2019). *Flex Sensor Special Edition Length* Retrived from <https://www.sparkfun.com/datasheets/Sensors/Flex/flex22.pdf>
- Components 101. (2019). *Joystick Modul* Retrived from <https://components101.com/modules/joystick-module>
- Raspberrypi.org. (2019). *Raspberry Pi Compute Module 3+* Retrived from https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf

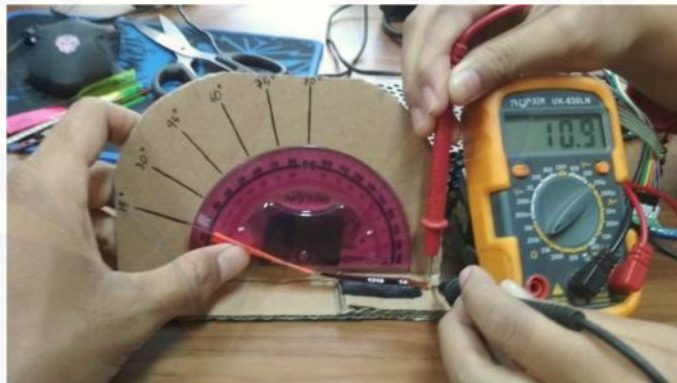
LAMPIRAN

A. Dokumentasi

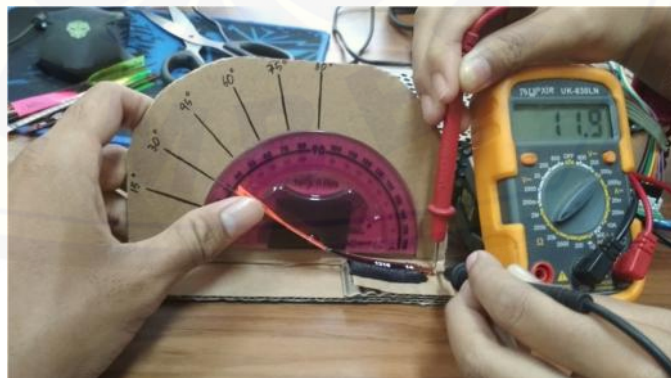
1. Pengujian *flex sensor*



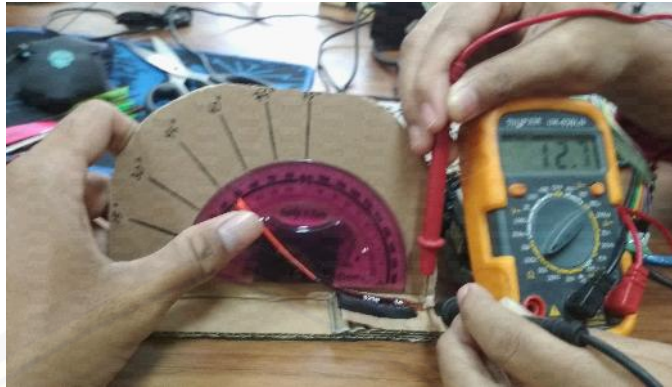
Gambar 1. Flex Sensor Saat Sudut 0°



Gambar 2. Flex Sensor Saat Sudut 15°



Gambar 3. Flex Sensor Saat Sudut 30°



Gambar 4. Flex Sensor Saat Sudut 45°



Gambar 5. Flex Sensor Saat Sudut 60°

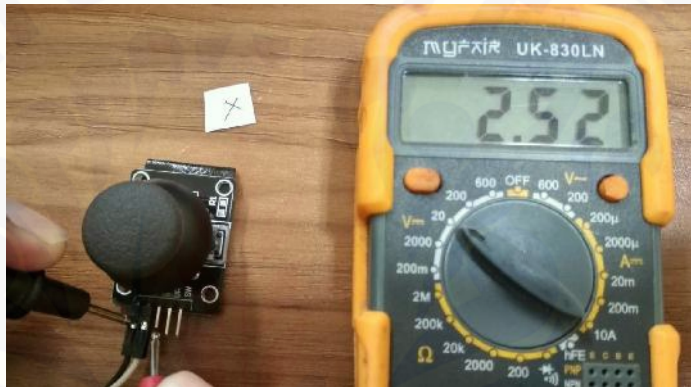


Gambar 6. Flex Sensor Saat Sudut 75°

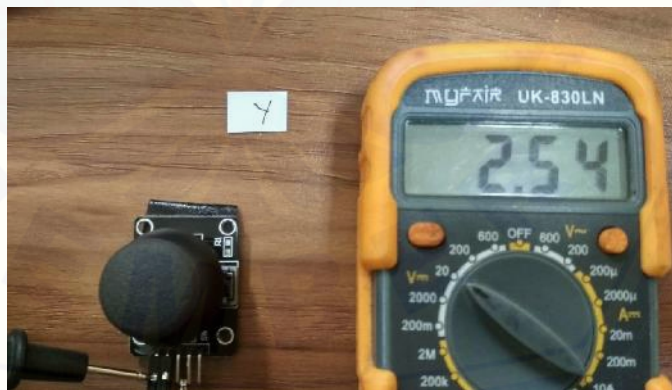


Gambar 7. Flex Sensor Saat Sudut 90°

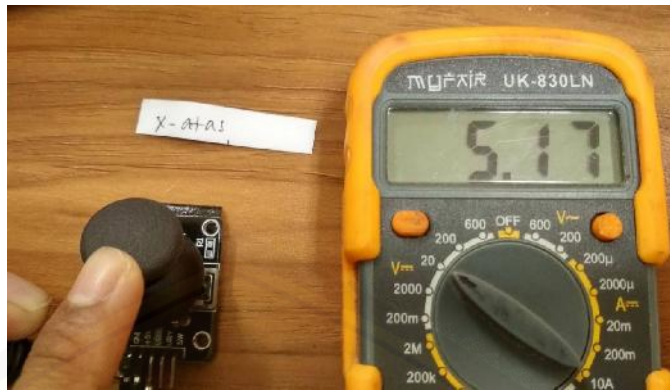
2. Pengujian *Joystick*



Gambar 8. Modul *Joystick* Posisi Awal Nilai Arah X



Gambar 9. Modul *Joystick* Posisi Awal Nilai Arah Y



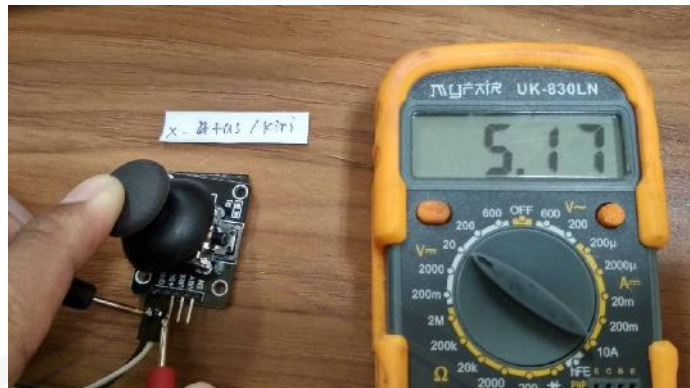
Gambar 10. Modul *Joystick* Posisi Atas Nilai Arah X



Gambar 11. Modul *Joystick* Posisi Bawah Nilai Arah X



Gambar 12. Modul *Joystick* Posisi Atas-Kanan Nilai Arah X



Gambar 13. Modul *Joystick* Posisi Atas-Kiri Nilai Arah X



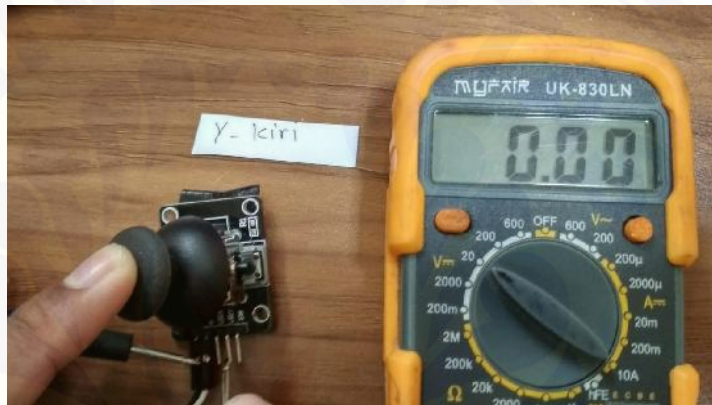
Gambar 14. Modul *Joystick* Posisi Bawah-Kanan Nilai Arah X



Gambar 15. Modul *Joystick* Posisi Bawah-Kiri Nilai Arah X



Gambar 16. Modul *Joystick* Posisi Kanan Nilai Arah Y



Gambar 17. Modul *Joystick* Posisi Kiri Nilai Arah Y



Gambar 18. Modul *Joystick* Posisi Kanan-Atas Nilai Arah Y



Gambar 19. Modul *Joystick* Posisi Kanan-Bawah Nilai Arah Y



Gambar 20. Modul *Joystick* Posisi Kiri-Atas Nilai Arah Y



Gambar 21. Modul *Joystick* Posisi Kiri-Bawah Nilai Arah Y

3. Pengujian Permainan Tetris



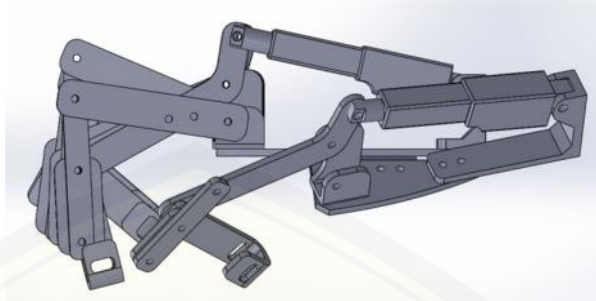
Gambar 22. Pengujian Permainan Tetris Menggunakan Konsol Buatan

4. Desain Logo Permainan Tetris Terapi

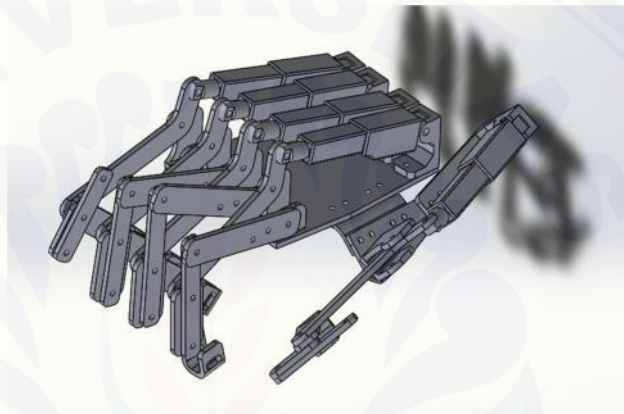


Gambar 23. Logo Tetris Terapi Robot Bantu Terapi Stroke pada Permainan

5. Desain Robot Tangan

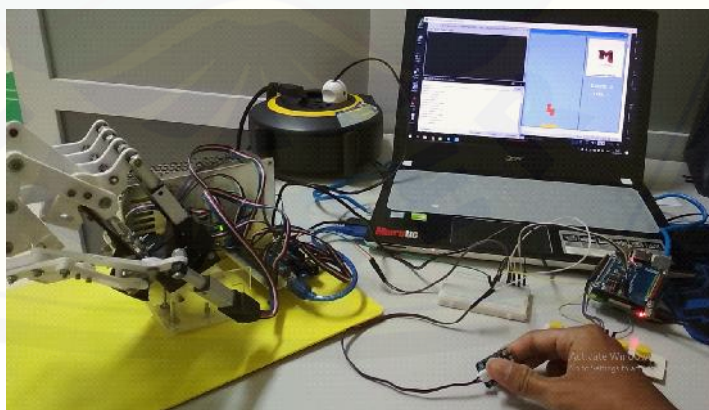


Gambar 24. Desain Robot Tangan dari Kanan

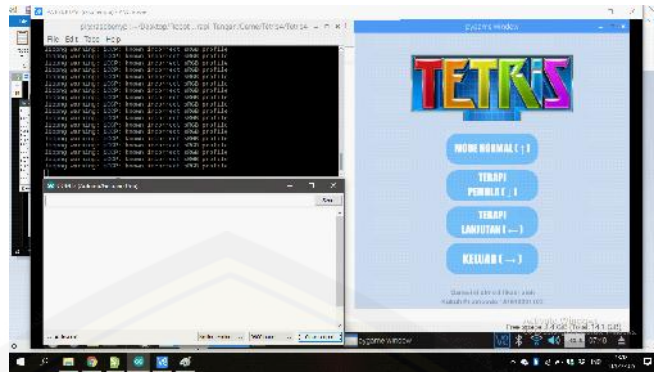


Gambar 25. Desain Robot Tangan dari Kanan Atas

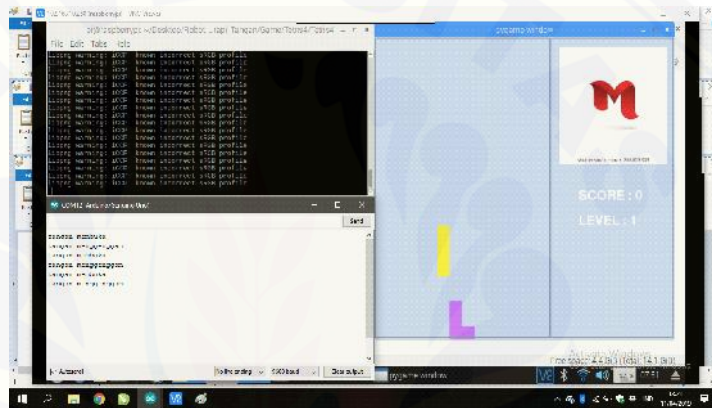
6. Pengujian Permainan dan Robot



Gambar 26. Pengujian Integrasi Permainan dan Robot Tangan

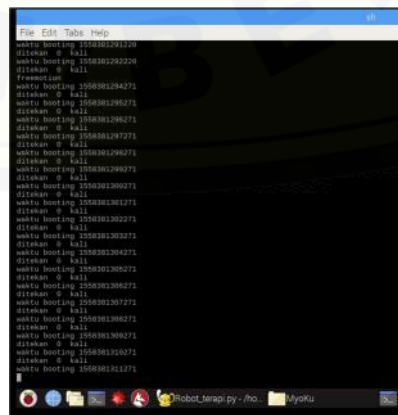


Gambar 27. Pengujian Integrasi Permainan dan Robot Tangan saat Menu awal Permainan



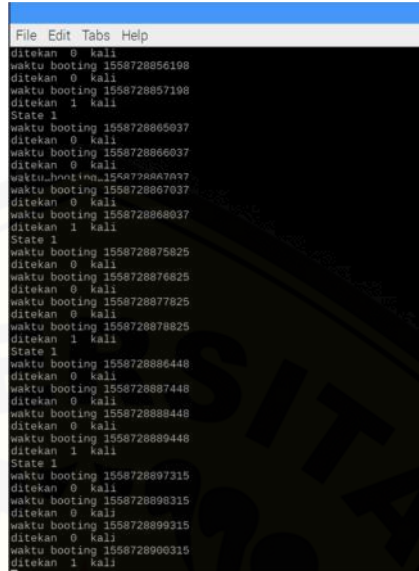
Gambar 28. Pengujian Integrasi Permainan dan Robot Tangan saat permainan Tetris dimainkan

7. Pengujian *Finite State Machine* - *Free Motion*



Gambar 29. Pengujian *State Free Motion*

- *State* Terdeteksi gerakan 1x



```
File Edit Tabs Help
ditekan 0 kali
waktu booting 1558728856198
ditekan 0 kali
waktu booting 1558728857198
ditekan 1 kali
State 1
waktu booting 155872886037
ditekan 0 kali
waktu booting 155872886037
ditekan 0 kali
waktu booting 1558728867037
ditekan 0 kali
waktu booting 1558728868037
ditekan 1 kali
State 2
waktu booting 1558728875825
ditekan 0 kali
waktu booting 1558728876825
ditekan 0 kali
waktu booting 1558728877825
ditekan 0 kali
waktu booting 1558728878825
ditekan 1 kali
State 1
waktu booting 1558728886448
ditekan 0 kali
waktu booting 1558728887448
ditekan 0 kali
waktu booting 1558728888448
ditekan 0 kali
waktu booting 1558728889448
ditekan 1 kali
State 1
waktu booting 1558728897315
ditekan 0 kali
waktu booting 1558728898315
ditekan 0 kali
waktu booting 1558728899315
ditekan 0 kali
waktu booting 1558728900315
ditekan 1 kali
```

Gambar 30. Pengujian *State* Terdeteksi gerakan 1x

- *State* Terdeteksi gerakan 2x

```
File Edit Tabs Help
ditekan 0 kali
waktu booting 1558730060797
ditekan 2 kali
State 2
waktu booting 1558730066049
ditekan 1 kali
State 1
waktu booting 1558730069513
ditekan 0 kali
waktu booting 1558730070513
ditekan 0 kali
waktu booting 1558730071513
ditekan 2 kali
State 2
waktu booting 1558730076547
ditekan 0 kali
waktu booting 1558730077547
ditekan 0 kali
waktu booting 1558730078547
ditekan 0 kali
waktu booting 1558730079547
ditekan 2 kali
State 2
waktu booting 1558730084696
ditekan 0 kali
waktu booting 1558730085696
ditekan 2 kali
State 2
waktu booting 1558730091117
ditekan 0 kali
waktu booting 1558730092117
ditekan 2 kali
State 2
waktu booting 1558730097539
ditekan 0 kali
waktu booting 1558730098539
ditekan 0 kali
waktu booting 1558730099539
ditekan 2 kali
State 2
waktu booting 1558730104542
ditekan 0 kali
waktu booting 1558730105542
```

Gambar 31. Pengujian *State* Terdeteksi gerakan 2x

- *State* Terdeteksi gerakan 3x

```
File Edit Tabs Help
State 3
waktu booting 1558733969706
ditekan 0 kali
waktu booting 1558733970706
ditekan 0 kali
waktu booting 1558733971706
ditekan 3 kali
State 3
waktu booting 1558733977217
ditekan 0 kali
waktu booting 1558733978217
ditekan 0 kali
waktu booting 1558733979217
ditekan 0 kali
waktu booting 1558733980217
ditekan 3 kali
State 3
waktu booting 1558733985344
ditekan 0 kali
waktu booting 1558733986344
ditekan 3 kali
State 3
waktu booting 1558733992249
ditekan 0 kali
waktu booting 1558733993249
ditekan 0 kali
waktu booting 1558733994249
ditekan 3 kali
State 3
waktu booting 1558734000098
ditekan 0 kali
waktu booting 1558734001098
ditekan 0 kali
waktu booting 1558734002098
ditekan 0 kali
waktu booting 1558734003098
ditekan 3 kali
State 3
waktu booting 1558734008776
ditekan 0 kali
waktu booting 1558734009776
ditekan 0 kali
waktu booting 1558734010776
```

Gambar 32. Pengujian *State* Terdeteksi gerakan 2x

B. Listing Program

-Permainan Tetris

```
#!/usr/bin/env python

from random import randrange as rand
import pygame, sys
import time
import RPi.GPIO as io

io.setmode(io.BCM)

pin_up = 17
pin_right = 18
pin_down = 20
pin_left = 19

# set pin sebagai input, comment jika dirun tanpa raspberry
io.setup(pin_up,io.IN) # make pin into an input
io.setup(pin_right,io.IN) # make pin into an input
io.setup(pin_down,io.IN) # make pin into an input
io.setup(pin_left,io.IN) # make pin into an input

# The configuration
config = {
    'cell_size': 25, #20
    'cols': 14, #8
    'rows': 24, #16
    'delay': 750, #750
    'maxfps': 30
}

colors = [
(0, 0, 0 ), # 0 0 0
(247, 75, 37 ), # 255 0 0
(250, 131, 43 ), # 0 150 0
(103, 229, 52 ), # 0 0 255
(117, 216, 252 ), # 255 120 0
(205, 120, 251), # 255 255 0
(255, 239, 66), # 180 0 255
(0, 102, 248) # 0 220 220
]

# Define the shapes of the single parts
```

```
tetris_shapes = [
    [[1, 1, 1],
     [0, 1, 0]],

    [[0, 2, 2],
     [2, 2, 0]],

    [[3, 3, 0],
     [0, 3, 3]],

    [[4, 0, 0],
     [4, 4, 4]],

    [[0, 0, 5],
     [5, 5, 5]],

    [[6, 6, 6, 6]],

    [[7, 7],
     [7, 7]]
]

mode = 0; #0 = normal, 1 = pemula, 2 = lanjutan

def rotate_clockwise(shape):
    return [ [ shape[y][x]
              for y in xrange(len(shape)) ]
            for x in xrange(len(shape[0]) - 1, -1,
-1) ]

def check_collision(board, shape, offset):
    off_x, off_y = offset
    for cy, row in enumerate(shape):
        for cx, cell in enumerate(row):
            try:
                if cell and board[ cy
+ off_y ][ cx + off_x ]:
                    return True
            except IndexError:
                return True
    return False

def remove_row(board, row):
    del board[row]
```



```
        return [[0 for i in xrange(config['cols'])]] +
board

def join_matrixes(mat1, mat2, mat2_off):
    off_x, off_y = mat2_off
    for cy, row in enumerate(mat2):
        for cx, val in enumerate(row):
            mat1[cy+off_y-1 ][cx+off_x] +=
val
    return mat1

def new_board():
    board = [ [ 0 for x in xrange(config['cols'])
]
            for y in
xrange(config['rows']) ]
    board += [[ 1 for x in
xrange(config['cols'])]]
    return board

class halamanAwal(object):
    def __init__(self):
        pygame.init()
        pygame.key.set_repeat(250,25)
        self.width =
config['cell_size']*config['cols'] + 250
        self.height =
config['cell_size']*config['rows']
        self.screen =
pygame.display.set_mode((self.width, self.height))

pygame.event.set_blocked(pygame.MOUSEMOTION) # We do
not need

# mouse movement

# events, so we

# block them.
        self.init_awal()

    def init_awal(self):
        dont_burn_my_cpu = pygame.time.Clock()
        while 1:
```

```
self.screen.fill((201, 218, 237))
#warna dasar awal

#logo
logo = pygame.image.load
('image/Logo2.JPG')
logo_center_x, logo_center_y =
logo.get_size()
logo_center_x //= 2
logo_center_y //= 2
self.screen.blit(logo,(self.width
// 2-logo_center_x,20))

#normal
normal_btn = pygame.image.load
('image/normal.png')
normal_btn_center_x,
normal_btn_center_y = normal_btn.get_size()
normal_btn_center_x //= 2
normal_btn_center_y //= 2

self.screen.blit(normal_btn,(self.width // 2-
normal_btn_center_x,220))

#pemula
pemula_btn = pygame.image.load
('image/pemula.png')
pemula_btn_center_x,
pemula_btn_center_y = pemula_btn.get_size()
pemula_btn_center_x //= 2
pemula_btn_center_y //= 2

self.screen.blit(pemula_btn,(self.width // 2-
pemula_btn_center_x,300))

#lanjutan
lanjutan_btn = pygame.image.load
('image/lanjutan.png')
lanjutan_btn_center_x,
lanjutan_btn_center_y = lanjutan_btn.get_size()
lanjutan_btn_center_x //= 2
lanjutan_btn_center_y //= 2

self.screen.blit(lanjutan_btn,(self.width // 2-
lanjutan_btn_center_x,380))
```

```
        #keluar
        keluar_btn = pygame.image.load
('image/keluar.png')
        keluar_btn_center_x,
keluar_btn_center_y = keluar_btn.get_size()
        keluar_btn_center_x //= 2
        keluar_btn_center_y //= 2

self.screen.blit(keluar_btn,(self.width // 2-
keluar_btn_center_x,460))

        #nama
        nama =
pygame.font.Font(pygame.font.get_default_font(),
14).render("Game ini dimodifikasi oleh",True,(139,
156, 185))
        nama_center_x, nama_center_y =
nama.get_size()
        nama_center_x //=2
        nama_center_y //=2
        self.screen.blit(nama,(self.width
// 2-nama_center_x,560))

        #nama_game
        nama_game=
pygame.font.Font(pygame.font.get_default_font(),
14).render("Kukuh Priambodo 151910201102",True,(139,
156, 185))
        nama_game_center_x,
nama_game_center_y = nama_game.get_size()
        nama_game_center_x //=2
        nama_game_center_y //=2

self.screen.blit(nama_game,(self.width // 2-
nama_game_center_x,580))

        pygame.display.update()
if io.input(pin_left) == 1:
    App = TetrisApp()
    App.change_mode(2)
    App.run()
if io.input(pin_right) == 1:
    self.quit()
```

```
if io.input(pin_up) == 1:
    mode = 0
    App = TetrisApp()
    App.run()
if io.input(pin_down) == 1:
    mode = 1
    App = TetrisApp()
    App.run()
for event in pygame.event.get():
    if event.type ==
pygame.QUIT:
    self.quit()
pygame.KEYDOWN:
    elif event.type ==
key_actions:
    # for key in
    #NORMAL
    if event.key
== eval("pygame.K_+"UP"):
    mode = 0
    App =
TetrisApp()
    App.run()
    #PEMULA
    elif event.key
== eval("pygame.K_+"DOWN"):
    mode = 1
    App =
TetrisApp()
    App.run()
    #LANJUTAN
    if event.key
== eval("pygame.K_+"LEFT"):
    App =
TetrisApp()
App.change_mode(2)
    # print
"ok ", mode
    App.run()
== eval("pygame.K_+"RIGHT"):
    elif event.key
self.quit()
```

```
dont_burn_my_cpu.tick(config['maxfps'])

    def quit(self):
        #self.center_msg("Exiting...")
        pygame.display.update()
        sys.exit()
class TetrisApp(object):
    def __init__(self):
        pygame.init()
        pygame.key.set_repeat(250,25)
        self.width =
config['cell_size']*config['cols'] + 250
        self.height =
config['cell_size']*config['rows']
        self.nilai_score = 0
        self.nilai_level = 1

pygame.time.set_timer(pygame.USEREVENT+1, 750)

pygame.time.set_timer(pygame.USEREVENT+2, 500)

pygame.time.set_timer(pygame.USEREVENT+3, 300)

pygame.time.set_timer(pygame.USEREVENT+4, 160)

pygame.time.set_timer(pygame.USEREVENT+5, 100)

        self.screen =
pygame.display.set_mode((self.width, self.height))

pygame.event.set_blocked(pygame.MOUSEMOTION) # We do
not need

# mouse movement

# events, so we

# block them.

        self.init_game()
```



```
def change_mode(self, mode):
    if mode == 2:
        self.nilai_level = 3
    else :
        self.nilai_level = 1
    print mode
def new_stone(self):
    self.stone =
tetris_shapes[rand(len(tetris_shapes))]
    self.stone_x = int(config['cols'] / 2
- len(self.stone[0])/2)
    self.stone_y = 0

    if check_collision(self.board,
                        self.stone,
                        (self.stone_x,
self.stone_y)):
        self.gameover = True

def init_game(self):
    self.board = new_board()
    self.new_stone()

def center_msg(self, msg):
    for i, line in
enumerate(msg.splitlines()): #Tulisan game over
        msg_image = pygame.font.Font(
pygame.font.get_default_font(), 16).render(
                                line, False,
(255,255,255))

        msgim_center_x, msgim_center_y
= msg_image.get_size()
        msgim_center_x //= 2
        msgim_center_y //= 2

        self.screen.blit(msg_image, (
            self.width // 2-
msgim_center_x,
            self.height // 2-
msgim_center_y+i*22))
def msg_notif(self, msg):
    for i, line in
enumerate(msg.splitlines()): #Tulisan Naik Level
```

```
msg_image = pygame.font.Font(
pygame.font.get_default_font(), 18).render(
    line, False,
    (255,255,255))

msgim_center_x, msgim_center_y
= msg_image.get_size()
msgim_center_x //= 2
msgim_center_y //= 2

self.screen.blit(msg_image, (
config['cell_size']*config['cols'] // 2-
msgim_center_x,
    self.height // 2-
msgim_center_y+i*22))

def draw_matrix(self, matrix, offset):
    off_x, off_y = offset
    for y, row in enumerate(matrix):
        for x, val in enumerate(row):
            if val:

pygame.draw.rect(
self.screen,
colors[val],
pygame.Rect(
(off_x+x) *
config['cell_size'],
(off_y+y) *
config['cell_size'],
config['cell_size'],
config['cell_size']),0)

def move(self, delta_x):
```

```
        if not self.gameover and not
self.paused:
            new_x = self.stone_x + delta_x
            if new_x < 0:
                new_x = 0
            if new_x > config['cols'] -
len(self.stone[0]):
                new_x = config['cols']
- len(self.stone[0])
                if not
check_collision(self.board,
self.stone,
                (new_x,
self.stone_y)):
                    self.sound_move.play()
                    self.stone_x = new_x
            def quit(self):
                self.center_msg("Exiting...")
                pygame.display.update()
                sys.exit()
            def down(self):
                self.sound_move.play()
                self.drop()
            def drop(self):
                if not self.gameover and not
self.paused:
                    self.stone_y += 1
                    if check_collision(self.board,
                    self.stone,
                    (self.stone_x, self.stone_y)):
                        self.board =
join_matrixes(
                            self.board,
                            self.stone,
                            (self.stone_x,
self.stone_y))
                        self.new_stone()
                    while True:
```

```

                                                for i, row in
enumerate(self.board[:-1]):
                                                if 0
not in row:
self.nilai_score+=10
self.sound_remove_row.play()
self.board = remove_row(
self.board, i)
break
                                                else:
                                                break

    def rotate_stone(self):
        if not self.gameover and not
self.paused:
            new_stone =
rotate_clockwise(self.stone)
            if not
check_collision(self.board,
new_stone,
(self.stone_x, self.stone_y)):
self.sound_rotate.play()
            self.stone = new_stone

    def toggle_pause(self):
        self.paused = not self.paused

    def start_game(self):
        if self.gameover:
            self.init_game()
            self.gameover = False

    def run(self):
        key_actions = {
            'ESCAPE': self.quit,
```

```

                                'LEFT':
lambda:self.move(-1),
                                'RIGHT':
lambda:self.move(+1),
                                'DOWN':          self.down,
                                'UP':
self.rotate_stone,
                                'p':
self.toggle_pause,
                                'SPACE':
self.start_game
                                }

                                self.gameover = False
                                self.paused = False
                                self.popup = 1;

                                # print config['delay']
                                dont_burn_my_cpu = pygame.time.Clock()

                                # music dan sound
                                pygame.mixer.init()

                                pygame.mixer.music.load("sound/Tetris.ogg")
                                pygame.mixer.music.play(-1, 0.0)
                                pygame.mixer.music.set_volume(0.25)
                                self.sound_rotate =
                                pygame.mixer.Sound("sound/block-rotate.ogg")
                                self.sound_rotate.set_volume(0.35)
                                self.sound_move =
                                pygame.mixer.Sound("sound/block-rotate.ogg")
                                self.sound_move.set_volume(0.35)
                                self.sound_remove_row =
                                pygame.mixer.Sound("sound/line-remove.ogg")
                                self.sound_remove_row.set_volume(0.35)

                                detik = 0

                                while 1:
                                    self.screen.fill((166, 212,
174)) #warna dasar game
                                    if self.gameover:
```



```
self.center_msg("""Game Over!\nPress space to
continue\n SCORE :"""+str(self.nilai_score)+"\nLvl
:"+str(self.nilai_level))
                else:
                    if self.paused:

self.center_msg("Paused")
                    else:

self.draw_matrix(self.board, (0,0))

self.draw_matrix(self.stone,
                (self.stone_x,
                self.stone_y))

pygame.draw.lines(self.screen, (139, 156, 185), True,
                [(0, 2), (350, 2), (350, 597), (0, 597)], 2)

pygame.draw.lines(self.screen, (139, 156, 185), True,
                [(353, 2), (597, 2), (597, 597), (353, 597)], 2)
                    score =
                pygame.font.Font(pygame.font.get_default_font(),
                26).render("SCORE : "+str(self.nilai_score),True,(0,
                50, 98))

self.screen.blit(score,(408,300))
                    level =
                pygame.font.Font(pygame.font.get_default_font(),
                26).render("LEVEL : "+str(self.nilai_level),True,(0,
                50, 98))

self.screen.blit(level,(408,350))
                    #image =
                pygame.image.load ('image/download.jpeg')
                    image = pygame.image.load
                ('image/Logo.jpg')

self.screen.blit(image,(370,20))

                    # merubah
                batasan level
```

```

if mode == 0 :
    if
self.nilai_score >= 200 :
self.nilai_level = 5
self.popup<self.nilai_level and detik !=30:
self.msg_notif("Selamat naik level 5")
detik+=1
else :
self.popup=5
detik = 0
elif
self.nilai_score >= 150 :
self.nilai_level = 4
self.popup<self.nilai_level and detik !=30:
self.msg_notif("Selamat naik level 4")
detik+=1
else :
self.popup=4
detik = 0
elif
self.nilai_score >= 40 :
self.nilai_level = 3
print
"detik : ",detik
print
"popup : ",self.popup
if
self.popup<self.nilai_level and detik !=30:
self.msg_notif("Selamat naik level 3")
detik+=1
```

```
else :  
  
self.popup=3  
  
detik = 0  
  
self.nilai_score >= 20 :  
  
self.nilai_level = 2  
  
self.popup<self.nilai_level and detik !=30:  
self.msg_notif("Selamat naik level 2")  
detik+=1  
  
self.popup=2  
detik = 0  
  
self.nilai_level # print  
  
self.nilai_level == 1 :  
self.kecepatan = pygame.USEREVENT+1  
  
self.nilai_level == 2 :  
self.kecepatan = pygame.USEREVENT+2  
  
self.nilai_level == 3 :  
self.kecepatan = pygame.USEREVENT+3  
  
self.nilai_level == 4 :  
self.kecepatan = pygame.USEREVENT+4  
  
self.nilai_level == 5 :  
self.kecepatan = pygame.USEREVENT+5
```

```

# print
config['delay']

pygame.display.update()

for event in
pygame.event.get():
    if event.type ==
self.kecepatan:
        self.drop()
    elif event.type ==
pygame.QUIT:
        self.quit()
    elif event.type ==
pygame.KEYDOWN:
        for key in
key_actions:
            if
event.key == eval("pygame.K_"
                    +key):
key_actions[key]()
            if io.input(pin_left)
== 1:
                time.sleep(0.05)
key_actions['LEFT']()
            if io.input(pin_right)
== 1:
                time.sleep(0.05)
key_actions['RIGHT']()
            if io.input(pin_up) ==
1:
                time.sleep(0.05)
key_actions['UP']()
            if io.input(pin_down)
== 1:
                time.sleep(0.05)
key_actions['DOWN']()
            if self.gameover and
io.input(pin_down) == 1 :
                App = halamanAwal()
```

```
dont_burn_my_cpu.tick(config['maxfps'])

if __name__ == '__main__':
    #App = TetrisApp()
    #App.run()
    App = halamanAwal()
```



-Robot

```
import wiringpi as wpi
from time import sleep, time

''' pin tombol '''
DIGITAL_PUSH_BUTTON = 21 #2 ## 5

''' Game Object'''
OBJEK_GAME = 5 #18 ## 24

''' pin Motor Linier '''
MOTOR_LINIER_1= 2 # 16 ## 27
MOTOR_LINIER_2= 25 # 15 ## 26
MOTOR_LINIER_3= 6 # 14 ## 25
MOTOR_LINIER_4= 3 # 13 ## 22
MOTOR_LINIER_5= 29 # 12 ## 21

''' pin flex sensor '''
SENSOR_1= 11 # 3 ## 7
SENSOR_2= 10 # 4 ## 8
SENSOR_3= 26 # 5 ## 12
SENSOR_4= 23 # 6 ## 13
SENSOR_5= 27 # 7 ## 16

wpi.wiringPiSetup()
wpi.pinMode(DIGITAL_PUSH_BUTTON, wpi.INPUT)
wpi.pullUpDnControl(DIGITAL_PUSH_BUTTON, wpi.PUD_DOWN)
wpi.pinMode(OBJEK_GAME, wpi.OUTPUT)
wpi.pinMode(SENSOR_1, wpi.INPUT)
wpi.pinMode(SENSOR_2, wpi.INPUT)
wpi.pinMode(SENSOR_3, wpi.INPUT)
wpi.pinMode(SENSOR_4, wpi.INPUT)
wpi.pinMode(SENSOR_5, wpi.INPUT)
wpi.pinMode(MOTOR_LINIER_1, wpi.OUTPUT)
wpi.pinMode(MOTOR_LINIER_2, wpi.OUTPUT)
wpi.pinMode(MOTOR_LINIER_3, wpi.OUTPUT)
wpi.pinMode(MOTOR_LINIER_4, wpi.OUTPUT)
wpi.pinMode(MOTOR_LINIER_5, wpi.OUTPUT)

count = 0
```

```
def Free_Motion(times):
    for i in range(0, times, 1):
        wpi.digitalWrite(MOTOR_LINIAR_1, wpi.HIGH)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_2, wpi.HIGH)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_3, wpi.HIGH)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_4, wpi.HIGH)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_5, wpi.HIGH)
        sleep(3)
        #jeda
        wpi.digitalWrite(MOTOR_LINIAR_1, wpi.LOW)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_2, wpi.LOW)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_3, wpi.LOW)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_4, wpi.LOW)
        sleep(1.5)
        wpi.digitalWrite(MOTOR_LINIAR_5, wpi.LOW)
        sleep(3)
        #jeda

def Buka_Genggam(times):
    for i in range(0, times, 1):
        wpi.digitalWrite(MOTOR_LINIAR_1, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIAR_2, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIAR_3, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIAR_4, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIAR_5, wpi.HIGH)
        sleep(1)
        wpi.digitalWrite(MOTOR_LINIAR_1, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIAR_2, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIAR_3, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIAR_4, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIAR_5, wpi.LOW)
        sleep(1)

'''def Buka_Genggamu(times):
    for i in range(0, times, 1):
        wpi.digitalWrite(MOTOR_LINIAR_1, wpi.LOW)
```

```
wpi.digitalWrite(MOTOR_LINIER_2, wpi.LOW)
wpi.digitalWrite(MOTOR_LINIER_3, wpi.LOW)
wpi.digitalWrite(MOTOR_LINIER_4, wpi.LOW)
wpi.digitalWrite(MOTOR_LINIER_5, wpi.LOW)
sleep(3)'''

def gerakan_yoga_mudra(times):
    for i in range(0, times, 1):
        #print("mudra")
        wpi.digitalWrite(MOTOR_LINIER_1, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_2, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_3, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIER_4, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIER_5, wpi.LOW)
        sleep(1.5)
        #print("relax")
        wpi.digitalWrite(MOTOR_LINIER_1, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_2, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_3, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_4, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_5, wpi.LOW)
        sleep(1.5)

def gerakan_yoga_vayu(times):
    for i in range(0, times, 1):
        #print("vayu")
        wpi.digitalWrite(MOTOR_LINIER_1, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIER_2, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_3, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_4, wpi.HIGH)
        wpi.digitalWrite(MOTOR_LINIER_5, wpi.HIGH)
        sleep(1.5)
        #print("relax")
        wpi.digitalWrite(MOTOR_LINIER_1, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_2, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_3, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_4, wpi.LOW)
        wpi.digitalWrite(MOTOR_LINIER_5, wpi.LOW)
        sleep(1.5)

def Games(times):
    for i in range(0, times, 1):
        wpi.digitalWrite(OBJEK_GAME, wpi.HIGH)
        sleep(0.05)
        wpi.digitalWrite(OBJEK_GAME, wpi.LOW)
```

```
        sleep(0.05)

def millis():
    return int(round(time() * 1000))

try:
    cur_time = millis()
    cur_time2 = millis()
    while True:
        cur_time = millis() + 1000
        print("waktu booting",cur_time)
        while cur_time - millis():
            if
wpi.digitalRead(DIGITAL_PUSH_BUTTON) == wpi.HIGH:
                sleep(0.2)
                count = count + 1
                if count>=4:
                    count=0
                cur_time = millis() + 1000
                cur_time2 = millis()
            print("ditekan ", count, " kali")

        if count == 1:
            Buka_Genggam(1)
            print("State 1")
        elif count == 2:
            gerakan_yoga_mudra(1)
            print("State 2")
        elif count == 3:
            gerakan_yoga_vayu(1)
            print("State 3")
        elif count == 0:
            if(millis()-cur_time2>20000):
                print("freemotion")
                Free_Motion(1)
                cur_time2 = millis()
            #print("-----")
            count=0

except KeyboardInterrupt:
    print("exit")
```