

Dr. Basuki Rahmat, S.Si, MT
Budi Nugroho, S.Kom, M.Kom

PEMROGRAMAN ROBOT CERDAS DENGAN ARDUINO



Dilengkapi Kamera Kinect Xbox 360,
Pemrograman Deep Learning dengan
Python dan Koneksi Internet of Things (IoT)

 **Indomedia
Pustaka**

Dr. Basuki Rahmat, S.Si, MT
Budi Nugroho, S.Kom, M.Kom

PEMROGRAMAN ROBOT CERDAS DENGAN ARDUINO

**Dilengkapi Kamera Kinect Xbox 360,
Pemrograman Deep Learning dengan Python
dan Koneksi Internet of Things (IoT)**



PEMROGRAMAN ROBOT CERDAS DENGAN ARDUINO

Dr. Basuki Rahmat, S.Si, MT
Budi Nugroho, S.Kom, M.Kom



Edisi Asli
Hak Cipta © 2020 pada penulis
Griya Kebonagung 2, Blok I2, No.14
Kebonagung, Sukodono, Sidoarjo
Telp.: 0812-3250-3457
Website: www.indomediapustaka.com
E-mail: indomediapustaka.sby@gmail.com

Hak cipta dilindungi undang-undang. Dilarang memperbanyak sebagian atau seluruh isi buku ini dalam bentuk apa pun, baik secara elektronik maupun mekanik, termasuk memfotokopi, merekam, atau dengan menggunakan sistem penyimpanan lainnya, tanpa izin tertulis dari Penerbit.

UNDANG-UNDANG NOMOR 19 TAHUN 2002 TENTANG HAK CIPTA

1. Barang siapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling lama **7 (tujuh) tahun** dan/atau denda paling banyak **Rp 5.000.000.000,00 (lima miliar rupiah)**.
2. Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud pada ayat (1), dipidana dengan pidana penjara paling lama **5 (lima) tahun** dan/atau denda paling banyak **Rp 500.000.000,00 (lima ratus juta rupiah)**.

Rahmat, Basuki
Nugroho, Budi

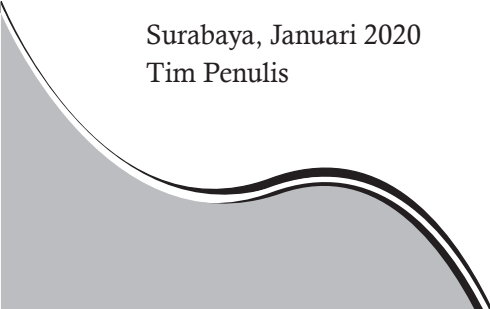
Pemrograman Robot Cerdas dengan Arduino/Basuki Rahmat, Budi Nugroho
Edisi Pertama
—Sidoarjo: Indomedia Pustaka, 2020
Anggota IKAPI No. 195/JTI/2018
1 jil., 17 × 24 cm, 128 hal.

ISBN: 978-623-7137-88-7

1. Pemrograman	2. Pemrograman Robot Cerdas dengan Arduino
I. Judul	II. Basuki Rahmat, Budi Nugroho



Prakata



Segala puji bagi Allah S.W.T yang telah melimpahkan rahmat, hidayah dan pertolongan-Nya sehingga kami dapat menyelesaikan buku yang berjudul “Pemrograman Robot Cerdas dengan Arduino”. Semoga buku ini bisa menjadi salah satu rujukan Buku Teks untuk para mahasiswa Informatika, Ilmu Komputer, Sistem Komputer, Teknik Elektro, Mekatronika, dan para mahasiswa teknik lainnya yang mencari literatur Mikrokontroler, Robotika, dan Pemrograman Sistem Cerdas.

Dengan selesainya buku ini, kami mengucapkan banyak terimakasih kepada:

1. Teman-teman di Jurusan Informatika dan Jurusan Sistem Informasi UPN “Veteran” Jawa Timur.
2. LPPM dan UPN “Veteran” Jawa Timur yang telah mengusahakan dan membiayai penerbitan buku ini, dalam Program Penelitian Mandiri Skim Peningkatan Mutu Pembelajaran (PMP) UPN "Veteran" Jawa Timur Tahun Anggaran 2019.
3. Dan semua pihak yang telah membantu hingga terselesaikannya buku ini.

Kami menyadari bahwa buku ini masih banyak kekurangan. Kritik, saran dan diskusi lebih lanjut, serta peluang kerjasama riset, dan lain-lain, bisa disampaikan melalui alamat email: basukirahmat.if@upnjatim.ac.id. Terimakasih.

Surabaya, Januari 2020
Tim Penulis

Daftar Isi

Prakata	iii
Daftar Isi.....	v
Bab 1 Pendahuluan	1
1.1. Perkembangan Robot Cerdas.....	1
1.2. Tantangan Robot Pengawasan.....	3
1.3. Gambaran Isi Buku.....	5
Bab 2 Perancangan dan Pembuatan Robot Cerdas	7
2.1. Desain Robot.....	7
2.2. Diagram Blok dan Rangkaian Elektronik Sistem.....	8
2.3. Cara Kerja Sistem	9
2.4. Gambaran Robot Cerdas.....	11
Bab 3 Internet of Things (IoT)	13
3.1. Apa itu IoT.....	13
3.2. Aplikasi IoT.....	15
3.3. Sistem IoT	16

3.4.	IoT dengan Cloud MQTT	16
3.5.	IoT Mobile dengan IoT MQTT Panel.....	21
Bab 4	Pemrograman Mikrokontroller	29
4.1.	Mikrokontroller NodeMCU V3	29
4.2.	Instalasi Driver NodeMCU V3	31
4.3.	Instalasi Arduino IDE	33
4.4.	Pemrograman NodeMCU V3.....	39
Bab 5	Kamera Kinect Xbox 360.....	49
5.1.	Tentang Kamera Kinect Xbox 360	49
5.2.	Instalasi Kamera Kinect Xbox 360.....	52
5.3.	Pemrograman Kamera Kinect Xbox 360.....	55
5.3.1	Pemrograman Python Melalui Lingkungan Anaconda	55
5.3.2	Pemrograman Kamera Kinect Xbox 360 dengan Python	65
Bab 6	Machine Learning dan Deep Learning.....	71
6.1.	Machine Learning.....	71
6.2.	Deep Learning dan Deep Network	73
6.3.	Pemrograman Deep Learning	78
6.3.1.	Framework Deep Learning	78
6.3.2.	Pemrograman Deep Neural Network (DNN) dengan OpenCV	86
6.3.3.	Pengenalan Objek Waktu Nyata dengan DNN	88
Bab 7	Pemrograman Robot Cerdas.....	95
7.1.	Kendali Robot via Web dan Mobile	95
7.2.	Pengenalan Objek dengan Deep Learning Menggunakan Kamera Kinect Xbox 360	108
Bab 8	Penutup	119
8.1.	Kesimpulan	119
8.2.	Ucapan Terimakasih	119
	Daftar Pustaka	121



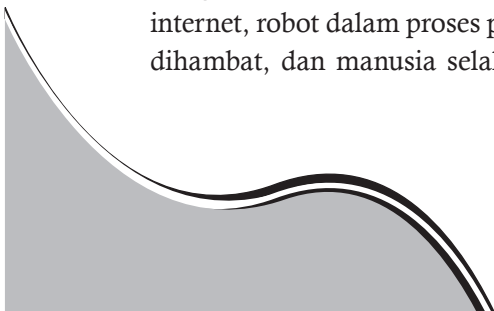
BAB 1

Pendahuluan

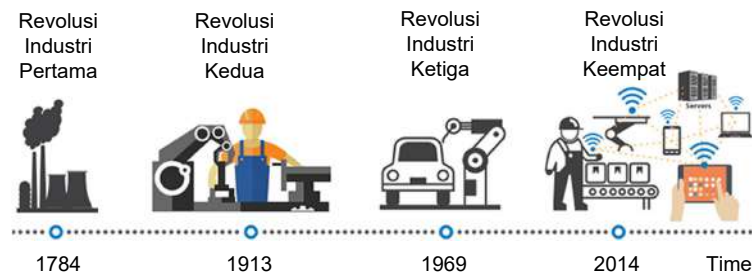
1.1. Perkembangan Robot Cerdas

Robot Cerdas atau *Intelligent Robot* adalah robot yang dirancang dan dibuat untuk menyelesaikan permasalahan tertentu secara cerdas menggunakan algoritma kecerdasan buatan (*Artificial Intelligence*). Pemanfaatan robot cerdas untuk berbagai keperluan terus berkembang dan semakin canggih, seperti robot industri dan robot pelayanan. Robot industri sering digunakan untuk membantu menyelesaikan proses produksi di pabrik-pabrik atau industri. Sedangkan robot pelayanan banyak dimanfaatkan untuk membantu layanan medis, pekerjaan rumah tangga, restoran, pendidikan, permainan, hiburan, dan lain sebagainya.

Di sisi lain, revolusi industri saat ini telah memasuki tahap Revolusi Industri Keempat atau industri 4.0. Revolusi industri pada dasarnya merupakan perubahan besar dan radikal terhadap cara manusia memproduksi barang atau produk. Revolusi Industri Pertama (1.0) di tahun 1784-an ditandai dengan penggunaan teknologi mesin uap dan mekanismenya. Revolusi Industri Kedua (2.0) di tahun 1913-an ditandai dengan pemanfaatan tenaga listrik dan produksi massal. Kemudian dunia mengalami Industri Ketiga (3.0) mulai tahun 1969-an ditandai dengan penggunaan Teknologi Informasi, internet, robot dalam proses produksinya. Perkembangan ilmu dan teknologi tidak bisa dihambat, dan manusia selalu mencari inovasi dalam berbagai kehidupan, sehingga

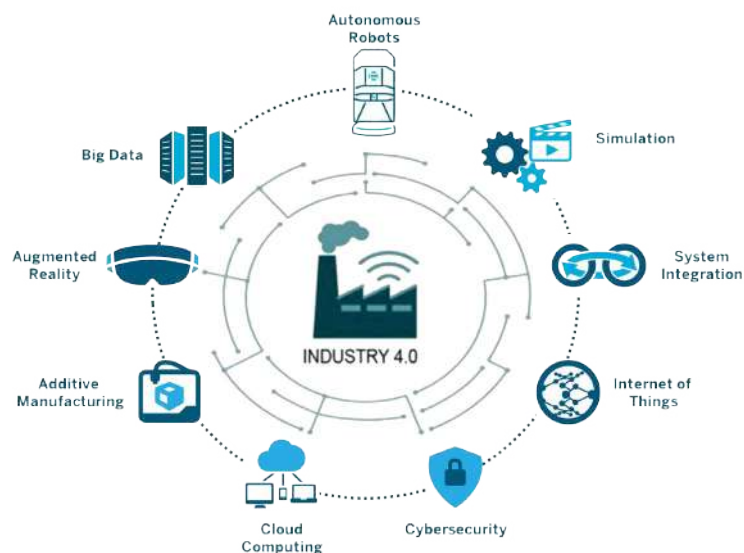


lahirlah Industri Keempat (4.0) saat ini. Secara umum, tahap perkembangan revolusi industri diperlihatkan pada Gambar 1.1 (Day, 2018).



Gambar 1.1. Tahap perkembangan revolusi industri (Day, 2018)

Seperti terlihat pada Gambar 1.1, saat ini kita telah memasuki era Revolusi Industri Keempat atau industri 4.0, yang salah satunya ditandai dengan banyaknya penggunaan teknologi robotik serta bermunculannya jenis pekerjaan baru. Di era revolusi industri 4.0 peranan robot cerdas semakin penting. *Autonomous robots* merupakan salah satu dari sembilan pilar teknologi industri 4.0, seperti diperlihatkan pada Gambar 1.2 (Nayyar and Kumar, 2019). Selain kedelapan pilar teknologi yang lain yaitu: *Simulation*, *System Integration*, *Internet of Things (IoT)*, *Cyber Security*, *Cloud Computing*, *Additive Manufacturing*, *Augmented Reality*, dan *Big Data*.

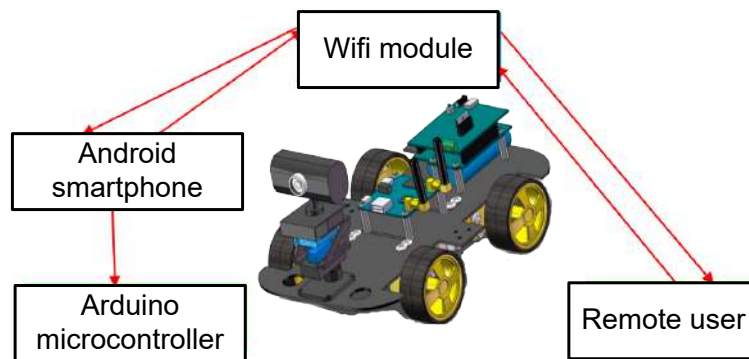


Gambar 1.2. Sembilan pilar teknologi industri 4.0 (Nayyar and Kumar, 2019)

Cepatnya perkembangan teknologi di era industri 4.0 saat ini, maka perlu diimbangi dengan peningkatan kualitas Sumber Daya Manusia yang kelak akan terus mengembangkan dan mungkin bersaing dengan teknologi robotik yang sudah mulai diperkenalkan ke masyarakat. Karena itu sangat penting bagi masyarakat untuk mempelajari setiap hal-hal baru yang berkaitan dengan perkembangan itu sendiri. Oleh karena itu, barangkali buku ini menjadi salah satu jawabannya.

1.2. Tantangan Robot Pengawasan

Dari sekian banyak robot cerdas, salah satu yang langsung terlihat kecerdasannya, adalah robot pengawasan (*surveillance robot*). Robot yang memanfaatkan kamera sebagai sensor andalannya dan tentunya dilengkapi dengan algoritma pengenalan objek. Robot pengawasan ini selanjutnya digunakan sebagai contoh pemrograman robot cerdas dari buku ini. Beberapa contoh penerapan robot pengawasan yang sudah dilakukan oleh para peneliti sebelumnya dan sekilas teknologinya diuraikan di sini.

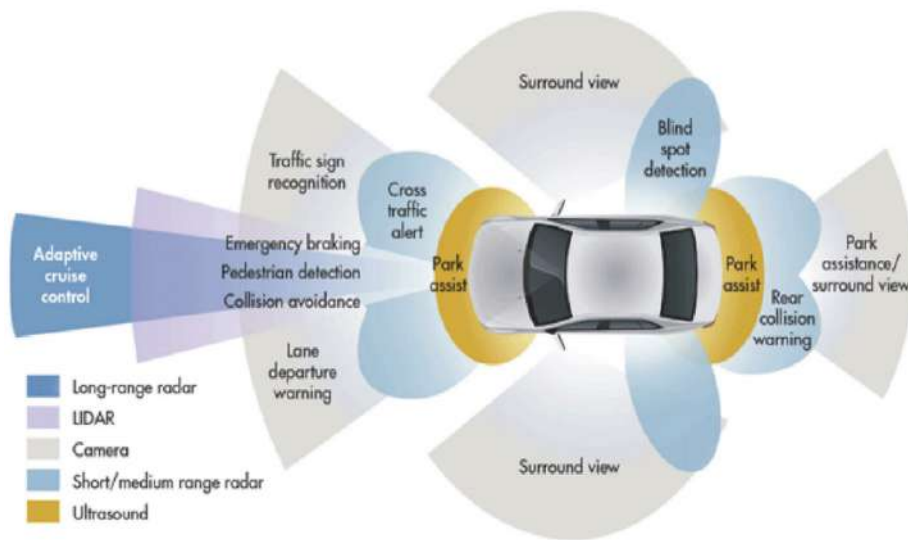


Gambar 1.3. Contoh robot pengawasan (Azeta *et al.*, 2019)

Salah satu contoh robot *mobile* berbasis android untuk pemantauan dan pengawasan seperti diperlihatkan pada Gambar 1.3 (Azeta *et al.*, 2019). Robot pengawasan ini dirancang dan dibuat secara hemat biaya dengan menggunakan mikrokontroler Arduino bersama dengan pelindung motor (*motor shield*) dan *smartphone* android yang menjalankan Sistem Operasi. Robot dilengkapi dengan kamera video dan tautan robot *Wifi*. Sistem memanfaatkan API (*Application Programming Interfaces*) yang disediakan untuk sistem operasi. Robot dapat dikendalikan dari jarak jauh menggunakan modul *Wifi* dan mikrokontroler, dan antarmuka ponsel pintar yang tertanam pada robot. Kamera pada robot digunakan untuk menangkap dan merekam video secara *real-time* dari robot. Robot dapat dikontrol berdasarkan umpan balik visual

dari *smartphone* yang sama. Motor DC roda empat membantu menavigasi robot dan sensor ultrasonik untuk menghindari rintangan. Kamera terpasang ke tautan robot *Wifi* yang memungkinkannya menangkap lingkungan atau objek yang menjadi perhatian. Hasil percobaan dengan berbagai posisi rintangan menunjukkan fleksibilitas robot untuk menghindarinya dan telah menunjukkan kinerja yang layak dan mendapatkan jangkauan komunikasi hampir 50 meter, yang cukup baik untuk banyak aplikasi pengawasan.

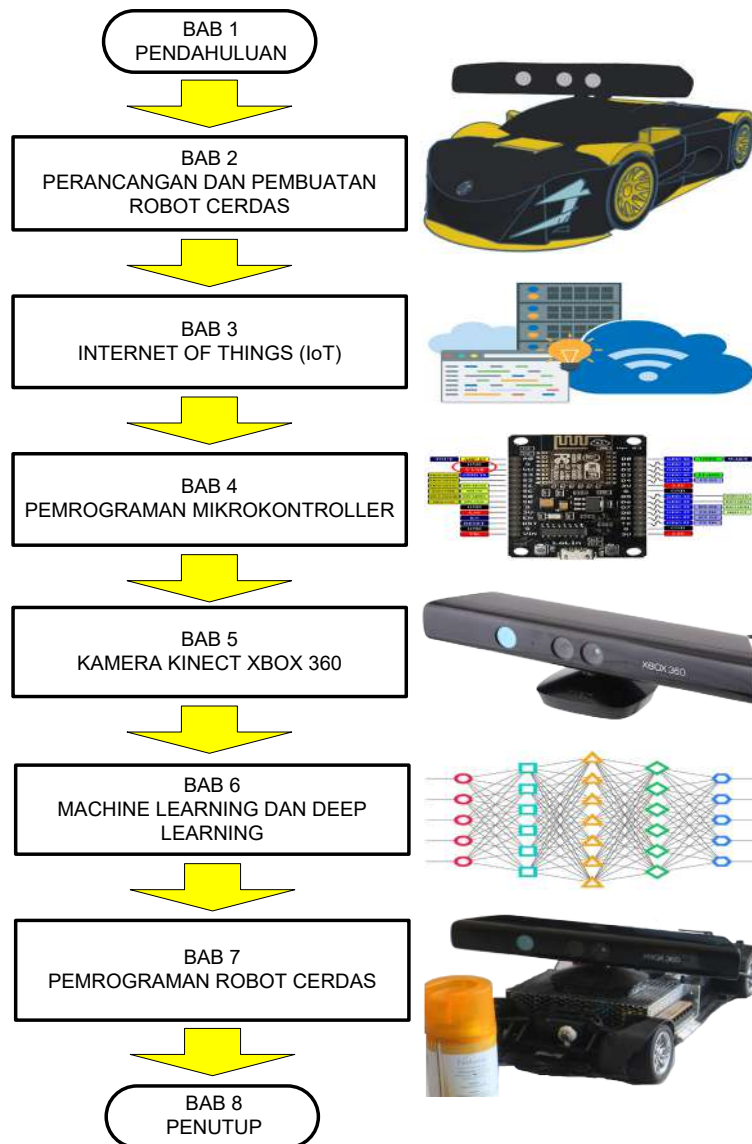
Contoh lain robot pengawasan yang sangat menantang saat ini berupa kendaraan tanpa awak atau tanpa pengemudi otomatis (*autonomously driving car*) seperti diperlihatkan pada Gambar 1.4 (Józwiak, 2017). Kendaraan ini dapat diwujudkan seiring kemajuan teknologi mikroelektronika, informasi, komunikasi, material, sensor, *cyber-physical systems* (CPS) dan *Internet of Things* (IoT). CPS dan IoT mengalami pertumbuhan eksplosif terkait dengan sistem seluler yang canggih seperti sistem otomotif dan avionik cerdas, robot seluler, dan perangkat yang dapat dipakai (*wearable devices*). Secara khusus, kendaraan atau mobil pintar melibatkan data instan besar dari berbagai sensor kompleks atau sistem lain, serta diharuskan menyediakan layanan otonom berkelanjutan dalam waktu yang lama. Untuk memenuhi tuntutan ini secara memadai, maka dibutuhkan komputasi tertanam (*embedded computing*) yang canggih dan teknologi desain tertanam (*embedded design technologies*) yang handal.



Gambar 1.4. Contoh sistem kendaraan tanpa awak (Józwiak, 2017)

1.3. Gambaran Isi Buku

Buku Pemrograman Robot Cerdas dengan Arduino ini, selanjutnya mengambil studi kasus pemrograman robot pengawasan. Gambaran isi buku secara keseluruhan, dibahas secara sistematis dengan urutan pembahasan seperti terlihat pada Gambar 1.5.



Gambar 1.5. Gambaran isi buku

Bab 1 Pendahuluan, berisi tentang perkembangan robot cerdas, tantangan robot pengawasan, dan gambaran isi buku. Bab 2 Perancangan dan Pembuatan Robot Cerdas, dibahas tentang desain robot, diagram blok dan rangkaian elektronik sistem, cara kerja sistem, dan gambaran robot cerdas. Bab 3 *Internet of Things* (IoT), dibahas tentang apa itu IoT, aplikasi IoT, sistem IoT, IoT dengan cloud MQTT, dan IoT *mobile* dengan IoT MQTT Panel. Bab 4 Pemrograman Mikrokontroller, dibahas tentang mikrokontroller NodeMCU V3, instalasi driver NodeMCU V3, instalasi Arduino IDE, dan pemrograman NodeMCU V3. Bab 5 Kamera Kinect Xbox 360, dibahas tentang kamera Kinect Xbox 360, instalasi kamera Kinect Xbox 360, dan pemrograman kamera Kinect Xbox 360. Bab 6 *Machine Learning* dan *Deep Learning*, dibahas tentang *Machine Learning*, *Deep Learning* dan *Deep Network*, serta pemrograman *Deep Learning*. Dimana pada pemrograman *Deep Learning*, dibahas tentang *framework Deep Learning*, pemrograman *Deep Neural Network* (DNN) dengan OpenCV, dan pengenalan objek waktu nyata dengan DNN. Bab 7 Pemrograman Robot Cerdas, dibahas tentang kendali robot via web dan *mobile*, dan pengenalan objek dengan *Deep Learning*. Dan terakhir, Bab 8 Penutup, berisi tentang kesimpulan, dan ucapan terimakasih.

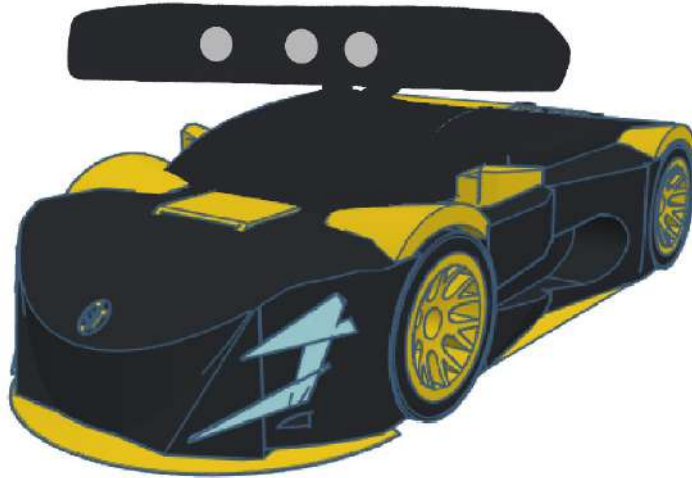


BAB 2

Perancangan dan Pembuatan Robot Cerdas

2.1. Desain Robot

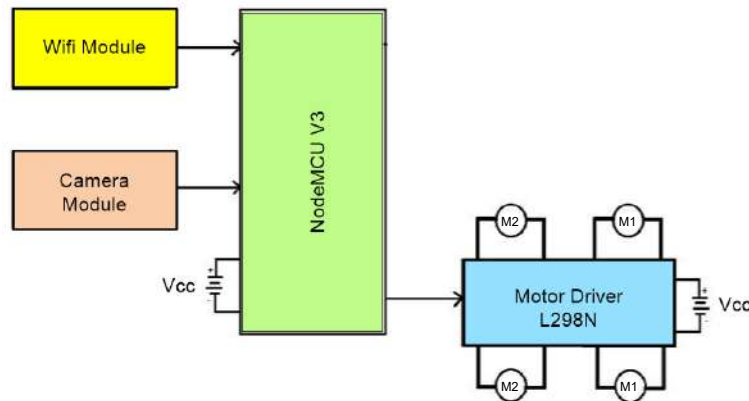
Robot Cerdas ini selanjutnya diberi nama BNU 4.0. BNU kependekan dari *Bela Negara University*, dedikasi untuk Kampus Bela Negara. Badan robot adalah *Remote Control car* (RC car) yang dimodifikasi. Otak kendali robot ini berupa mikrokontroller NodeMCU V3. *Board* elektronik yang berbasis *chip* ESP8266. Dengan *chip* ini, NodeMCU V3 memiliki kemampuan koneksi ke *cloud Internet of Things* (IoT). Agar robot dapat bergerak maju, mundur, belok kanan, dan belok kiri, digunakan driver motor L298N. Robot ini juga dilengkapi dengan kamera Kinect Xbox 360. Rancangan robot ini seperti diperlihatkan pada Gambar 2.1.



Gambar 2.1. Rancangan robot cerdas

2.2. Diagram Blok dan Rangkaian Elektronik Sistem

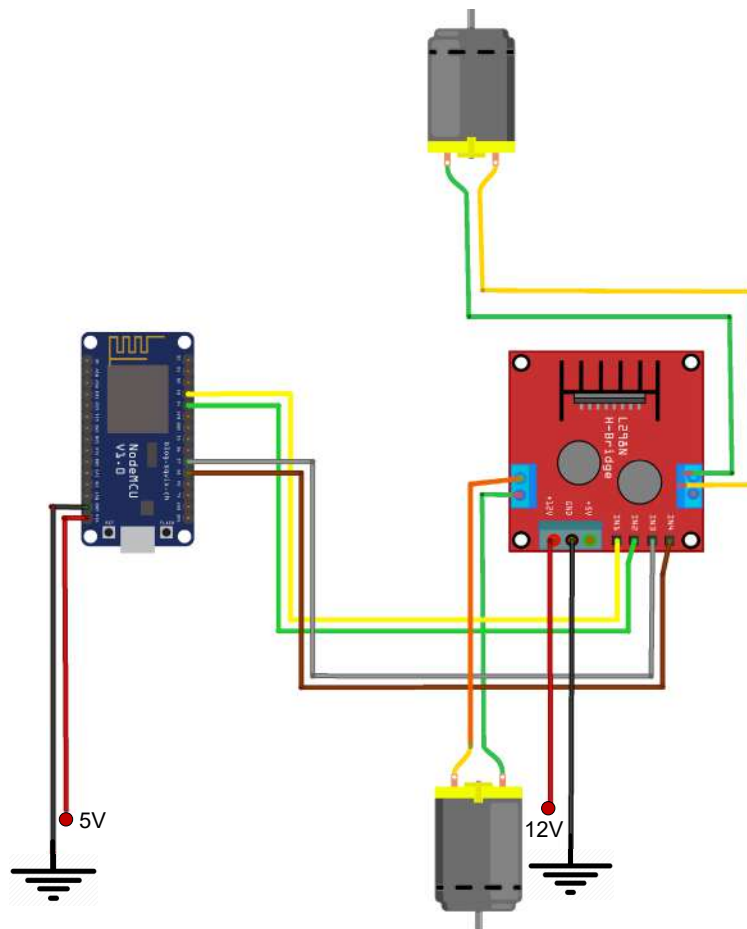
Diagram blok sistem Robot Cerdas BNU 4.0 seperti diperlihatkan pada Gambar 2.2.



Gambar 2.2. Diagram blok sistem robot cerdas

Sedangkan rangkaian elektronik sistem Robot Cerdas BNU 4.0 seperti diperlihatkan pada Gambar 2.3. Dimana IN 1 dan IN 2 dari driver motor L298N, dihubungkan ke pin

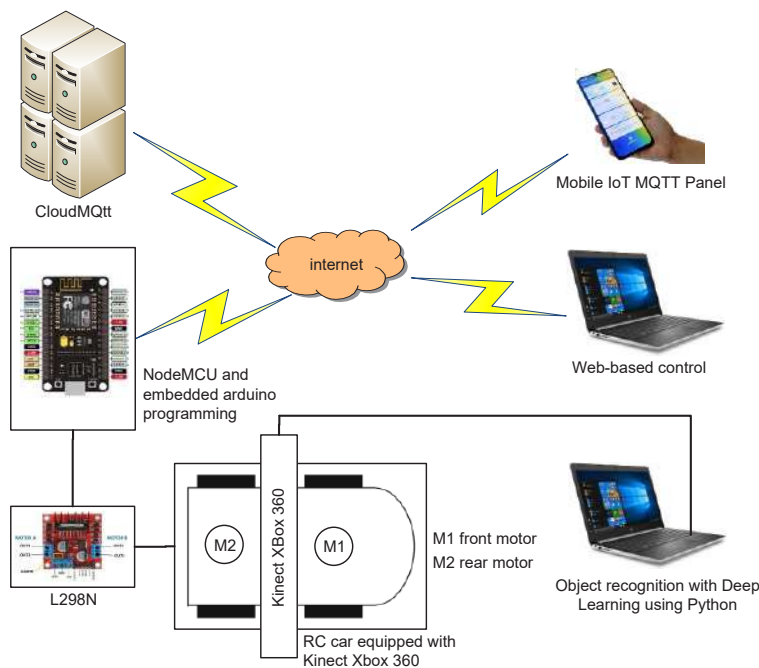
D3 dan D4 dari mikrokontroler NodeMCU V3. Hal ini digunakan untuk mengendalikan motor belakang (M2), untuk gerak maju dan mundur. Sedangkan IN 3 dan IN 4 dari driver motor L298N, dihubungkan ke pin D7 dan D8 dari mikrokontroler NodeMCU V3. Hal ini digunakan untuk mengendalikan motor depan (M1), untuk belok kanan dan belok kiri.



Gambar 2.3. Rangkaian elektronik sistem robot cerdas

2.3. Cara Kerja Sistem

Cara kerja sistem atau bagaimana Robot Cerdas BNU 4.0 dijalankan dan digunakan untuk mengenali objek di depannya, diperlihatkan pada Gambar 2.4.



Gambar 2.4. Cara kerja robot cerdas

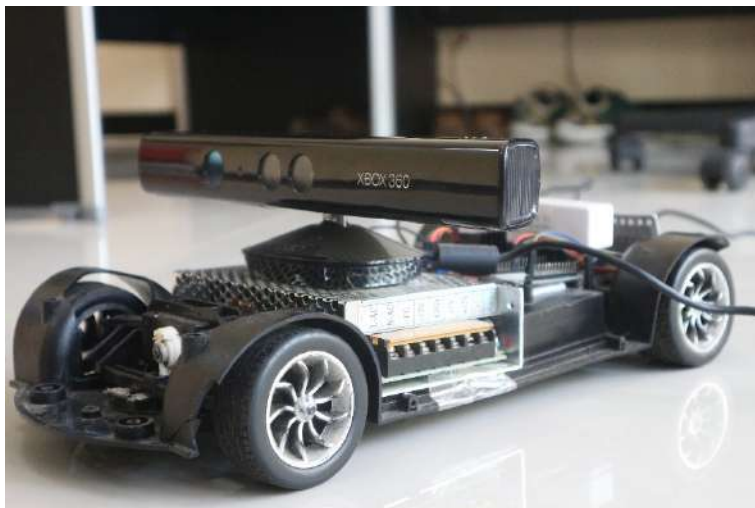
Dari Gambar 2.4 dapat dijelaskan sebagai berikut. Badan robot berupa mobil RC yang telah dimodifikasi. Mobil RC biasanya memiliki dua motor, yaitu motor depan (M1) dan motor belakang (M2). Motor depan digunakan untuk belok kanan dan belok kiri, sedangkan motor belakang digunakan untuk gerak maju dan mundur. Untuk menjalankan kedua motor ini, digunakan driver motor L298N. Driver motor terhubung ke mikrokontroller NodeMCU V3 dengan hubungan rangkaian seperti pada Gambar 2.3. NodeMCU V3 berbasis *chip* ESP8266. Dengan *chip* ini, NodeMCU V3 memiliki kemampuan koneksi ke *cloud Internet of Things* (IoT). *Cloud* IoT yang digunakan di buku ini yaitu *cloud MQTT* (<https://www.cloudmqtt.com>). Dengan program Arduino yang tertanam di mikrokontroller NodeMCU V3, selanjutnya dapat dijalankan program kendali robot berbasis web. Dengan program Arduino yang dirancang dan tertanam di mikrokontroller NodeMCU V3 ini juga, selanjutnya dapat dijalankan program kendali robot melalui *Android Apps* (IoT MQTT Panel). Robot ini juga dilengkapi dengan kamera Kinect Xbox 360. Kamera ini terhubung ke komputer (laptop). Dengan program *Deep Learning* yang ada di komputer selanjutnya sistem akan mengenali objek yang ada di depannya.

2.4. Gambaran Robot Cerdas

Dengan demikian gambaran Robot Cerdas BNU 4.0 yang dijalankan dengan misi pengenalan objek menggunakan algoritma *Deep Learning*, diperlihatkan pada Gambar 2.5 sampai dengan Gambar 2.8.



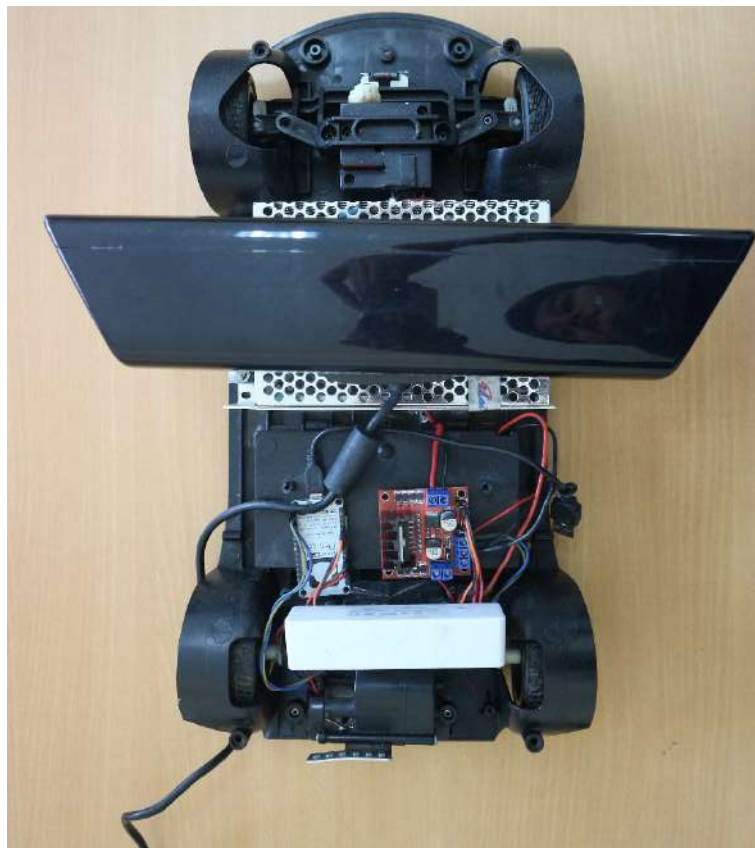
Gambar 2.5. Robot tampak depan



Gambar 2.6. Robot tampak samping



Gambar 2.7. Robot tampak samping dengan objek di depannya



Gambar 2.8. Robot tampak atas




BAB 3

Internet of Things (IoT)

3.1. Apa itu IoT

Internet of Things (IoT) adalah area yang muncul di mana milyaran objek pintar saling berhubungan satu sama lain menggunakan internet untuk berbagi data dan sumber daya (Chahal, Kumar and Batra, 2020). Teknologi IoT memungkinkan benda-benda di sekitar kita saling terhubung dengan jaringan internet. Dimana setiap benda yang terhubung dengan internet bisa diakses kapan saja dan dimana saja. Contohnya, dari jarak jauh kita bisa menghidup-matikan peralatan di rumah (lampu, televisi, kompor, pemanas, dan lain-lain) selama peralatan terhubung ke *cloud* IoT dan tersedia koneksi internet. Secara umum arsitektur IoT terdiri dari *Application Layer*, *Middleware Layer*, *Network Layer*, dan *Physical Layer*, seperti diperlihatkan pada Gambar 3.1 (Ravidas *et al.*, 2019).

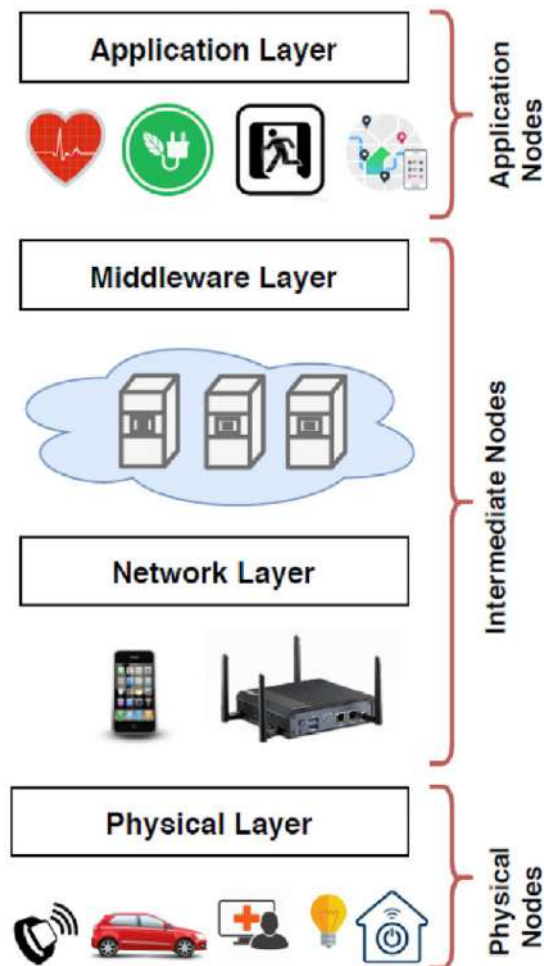
Lapisan Aplikasi (*Application Layer*): bertujuan untuk menyediakan layanan kepada pengguna akhir. Lapisan ini terdiri dari simpul aplikasi yang menangani logika aplikasi serta semantik data dan presentasi. Simpul ini menerima data dari *middleware* dan memprosesnya tergantung pada persyaratan pengguna akhir dan jenis layanan yang disediakan. Selain itu, lapisan aplikasi mencakup *Application Programming Interface* (API) untuk memfasilitasi komunikasi dengan *middleware* dan antarmuka pengguna yang digunakan pengguna akhir untuk mengakses layanan.



Lapisan Middleware (Middleware Layer): untuk memastikan konektivitas dan interoperabilitas dalam ekosistem IoT. Ini terdiri dari simpul menengah yang memproses data yang diterima dari lapisan bawah dan meneruskannya ke lapisan aplikasi.

Lapisan Jaringan (Network Layer): untuk mendukung jaringan dan transfer data antar simpul. Lapisan jaringan mengimplementasikan protokol komunikasi yang diperlukan untuk pertukaran data dalam ekosistem IoT.

Lapisan Fisik (Physical Layer): untuk mengkarakterisasi kemampuan penginderaan dan kontrol dari sistem IoT. Lapisan ini terdiri dari simpul fisik seperti sensor dan aktuator yang merasakan lingkungan dan berinteraksi dengannya dalam menanggapi perubahan atau permintaan pengguna. Node ini menghasilkan sumber daya (merasakan data) yang dilewatkan ke simpul aplikasi melalui jaringan dan lapisan *middleware*.



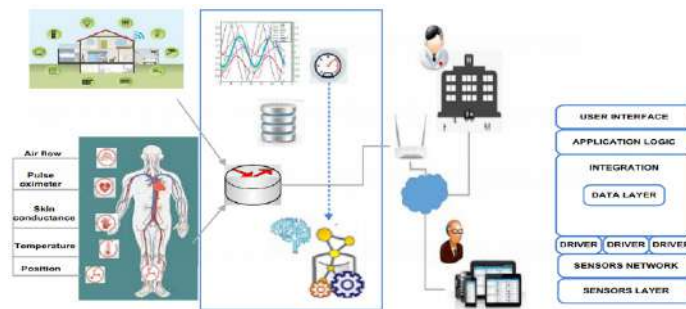
Gambar 3.1. Arsitektur IoT (Ravidas *et al.*, 2019)

3.2. Aplikasi IoT

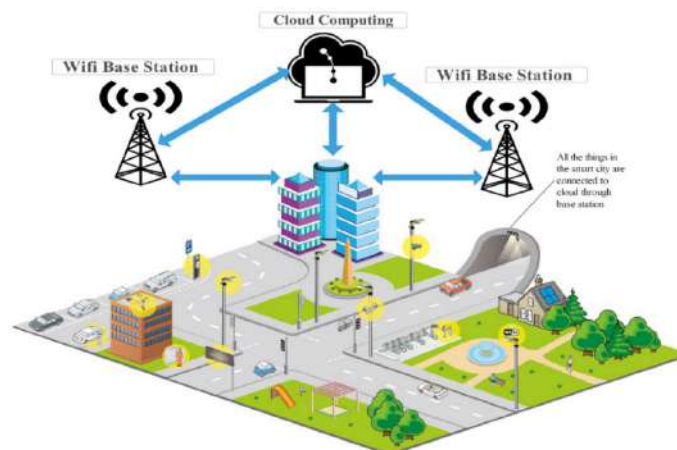
Banyak sekali contoh penerapan teknologi IoT, beberapa contohnya:

- *Smart Home* (sistem keamanan rumah berbasis internet, dapat mengetahui keadaan rumah serta mengontrol peralatan rumah tangga melalui jaringan internet).
- *Smart Farming* (sistem pertanian cerdas berbasis internet, untuk pemantauan dan pengendalian kualitas air dan tanah pertanian serta pertumbuhan tanaman melalui jaringan internet).
- *Internet industry* (pemantauan dan pengendalian peralatan serta proses di industri).
- Kesehatan (pemantauan kondisi kesehatan seseorang).
- Transportasi (majemen dan informasi lalu lintas).

Gambaran contoh penerapan IoT untuk bidang kesehatan dan *Smart City* diperlihatkan pada Gambar 3.2 dan Gambar 3.3.



Gambar 3.2. Contoh arsitektur berbasis IoT untuk bidang kesehatan (Zeadally and Bello, 2019)



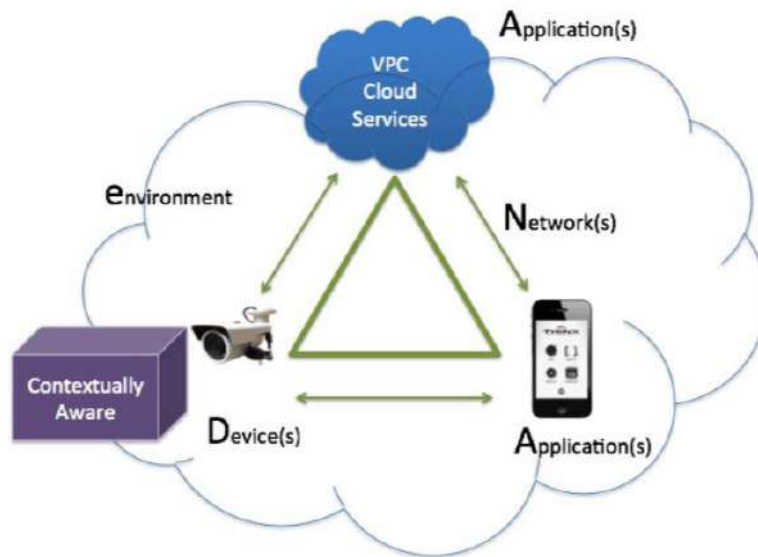
Gambar 3.3. Contoh penerapan IoT pada Smart City (Gheisari, Wang and Chen, 2020)

3.3. Sistem IoT

Sistem dasar dari IoT terdiri dari 3 hal, yaitu:

- Hardware/fisik (*Things*).
- Koneksi internet.
- *Cloud data center* sebagai tempat untuk menyimpan atau menjalankan aplikasinya.

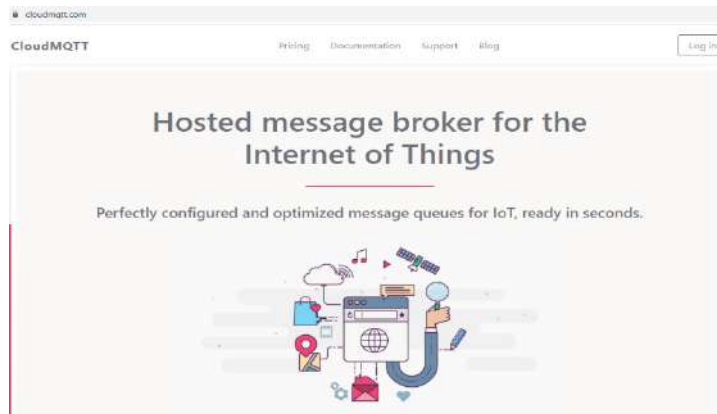
Masing-masing seperti diperlihatkan pada Gambar 3.4 (anonymous, 2019).



Gambar 3.4. Sistem IoT (anonymous, 2019)

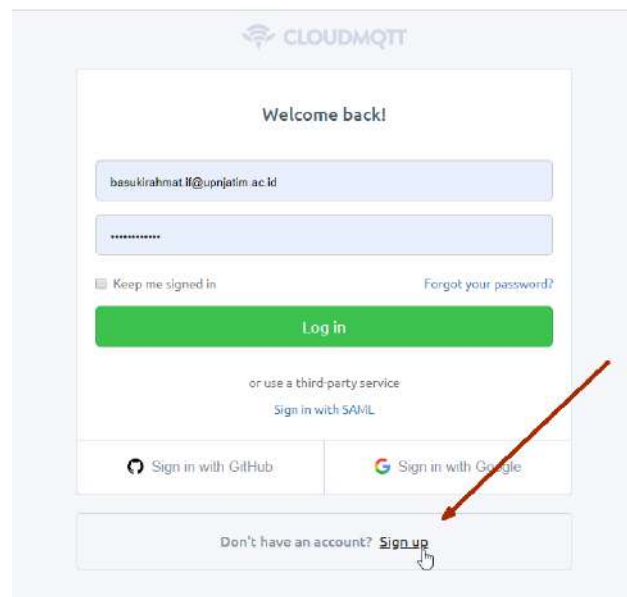
3.4. IoT dengan Cloud MQTT

Seperti telah disebutkan pada sistem IoT, untuk membuat proyek berbasis IoT kita membutuhkan *cloud data center* atau *cloud IoT*. Salah satu *cloud IoT* yang bisa dimanfaatkan adalah *cloud MQTT* (<https://www.cloudmqtt.com>). Tampilan halaman website *cloud MQTT* seperti diperlihatkan pada Gambar 3.5.



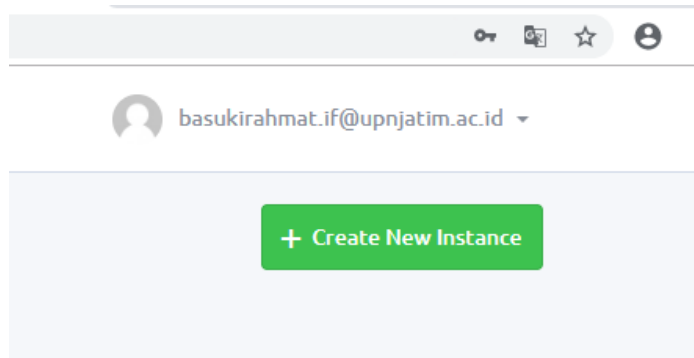
Gambar 3.5. Cloud MQTT

Untuk memanfaatkan *cloud* MQTT harus memiliki akun di *cloud* ini. Jika belum punya, silahkan daftar terlebih dahulu. Seperti diperlihatkan pada Gambar 3.6.



Gambar 3.6. Daftar akun cloud MQTT

Setelah memiliki akun silahkan login ke sistem. Setelah berhasil masuk ke halaman login, silahkan buat proyek baru dengan menekan tombol *Create New Instance*.



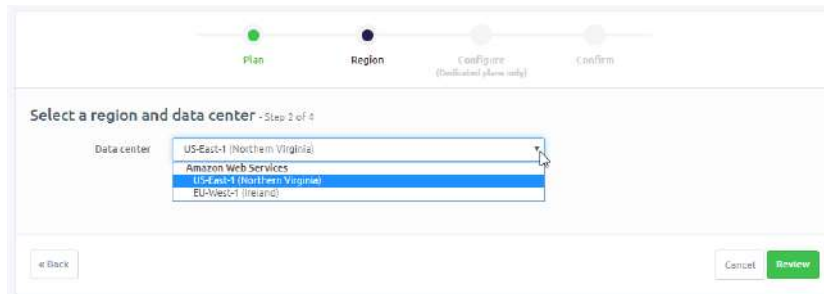
Gambar 3.7. Create New Instance untuk membuat proyek baru

Silahkan isikan *Name* dan *Tags* sembarang, selanjutnya tekan tombol *Select Region*. Contoh misalkan nama proyeknya bisa diketikkan: Robot BNU 4.0.

A screenshot of a 'Create new instance' form. At the top, there are two yellow warning banners: 'No credit card Please add a credit card if you want to subscribe to a paid plan' and 'Missing billing information Please fill in all required information if you want to subscribe to a paid plan'. Below these is a progress bar with four steps: 'Plan' (selected), 'Region', 'Configure (dedicated plans only)', and 'Confirm'. The main section is titled 'Select a plan and name - Step 1 of 4'. It contains three input fields: 'Name' with the value 'Robot BNU 4.0', 'Plan' with a dropdown menu showing 'Cute Cat (Free)', and 'Tags' with the value 'Robot Bela Negara'. To the right of these fields is a 'Plan' section featuring a 'Cute Cat' character icon and the text 'Cute Cat'. Below this, there is a link: 'See the plan page to learn about the different plans.' At the bottom right, there are two buttons: 'Cancel' and 'Select Region'.

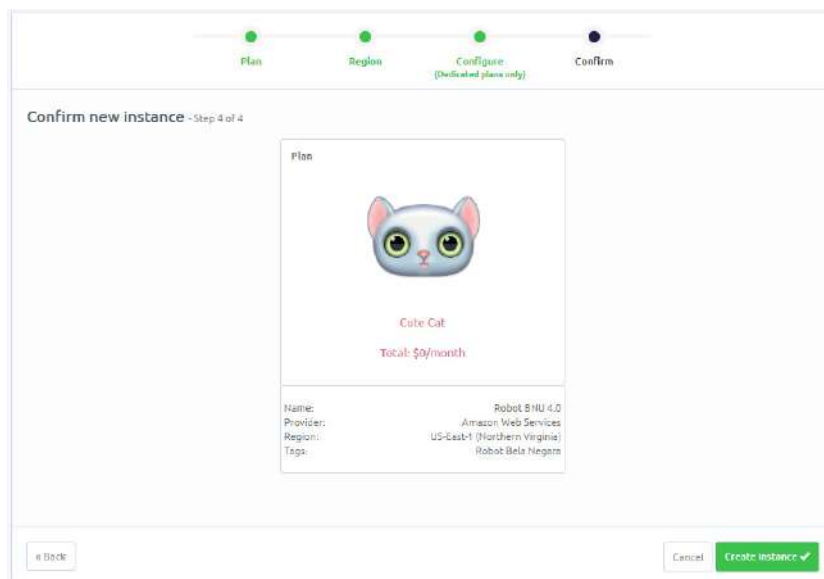
Gambar 3.8. Nama proyek

Setelah menekan tombol *Select Region*, silahkan dipilih salah satu server yang ada di *region* tersebut. Setelah itu silahkan lanjut dengan menekan tombol *Review*.



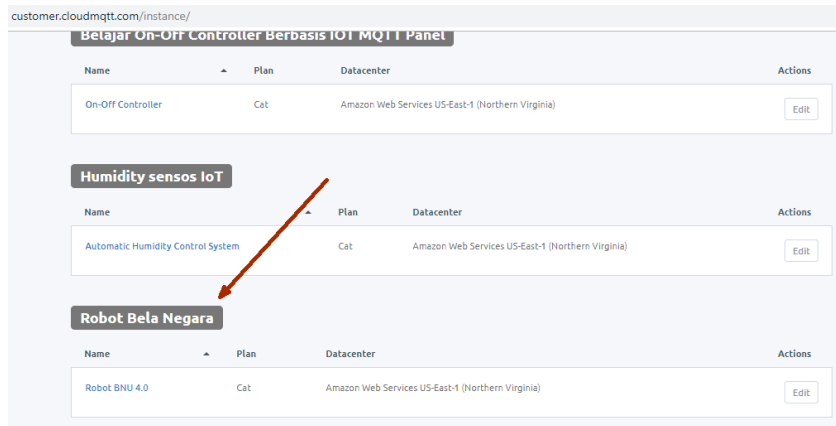
Gambar 3.9. Pilihan lokasi data center

Setelah menekan tombol *Review*, muncul konfirmasi proyek baru (*confirm new instance*). Untuk paket gratis, tampil gambar kucing lucu (*Cute Cat*). Setelah itu silahkan tekan tombol *Create instance*.



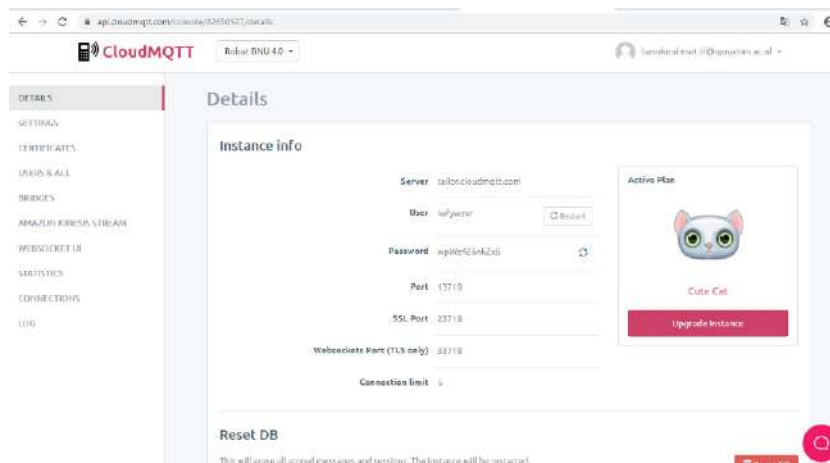
Gambar 3.10. Konfirmasi proyek baru

Setelah menekan tombol *Create instance*, silahkan diperiksa, seharusnya proyek baru berhasil ditambahkan. Server *cloud IoT* untuk proyek kita, siap digunakan.

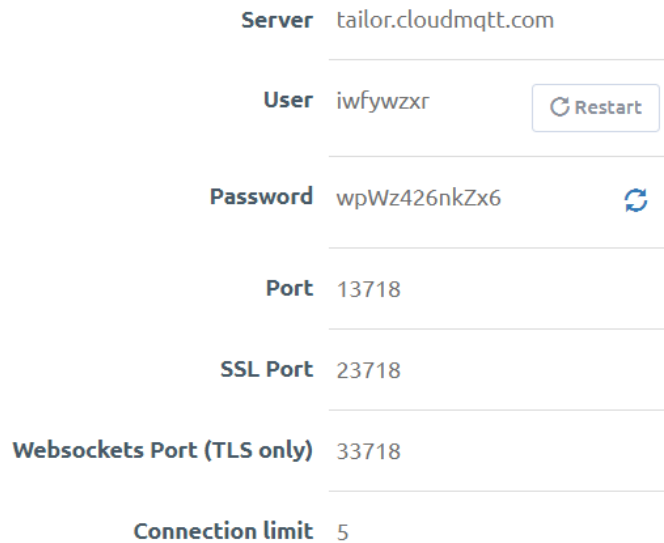


Gambar 3.11. Proyek baru berhasil ditambahkan

Untuk melihat detail informasi server dan lain-lain, silahkan tekan proyek kita (Robot BNU 4.0). Selanjutnya silahkan disimpan, nama *server*, *user*, *password*, *port*, dan lain-lain.



Gambar 3.12. Informasi detail proyek IoT di cloud MQTT



Server	tailor.cloudmqtt.com	
User	iwfywzxr	Restart
Password	wpWz426nkZx6	Refresh
Port	13718	
SSL Port	23718	
Websockets Port (TLS only)	33718	
Connection limit	5	

Gambar 3.13. Informasi detail server dan lain-lain

Nama *server*, *user*, *password* dan *port* yang kita peroleh, nantinya kita gunakan untuk pengaturan program IoT pada IoT MQTT Panel dan di mikrokontroller NodeMCU V3.

3.5. IoT Mobile dengan IoT MQTT Panel

MQTT kependekan dari *Message Queuing Telemetry Transport*, yaitu protokol pesan berbasis penerbitan standar berlangganan *International Organization for Standardization* (ISO). MQTT adalah protokol yang digunakan dalam IoT untuk transmisi data. Lapisan Aplikasi dan protokol MQTT merupakan lapisan paling atas dari TCP/IP. MQTT merupakan protokol berbasis penerbit dan pelanggan yang memungkinkan beberapa perangkat berkomunikasi satu sama lain melalui jaringan nirkabel (Kashyap, Sharma and Gupta, 2018).

Beberapa istilah penting terkait MQTT, antara lain:

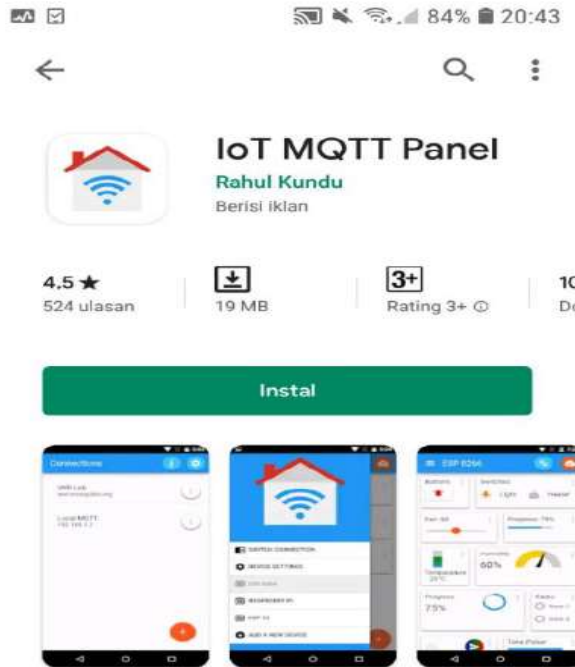
- **Broker** - Broker adalah server yang mendistribusikan informasi kepada klien yang terhubung ke server.
- **Client** (Klien) - Perangkat yang terhubung ke broker untuk mengirim atau menerima informasi.
- **Topic** (Topik) - Nama tentang pesan itu. Klien mempublikasikan, berlangganan, atau melakukan keduanya untuk suatu topik.

- **Publish** (Terbitkan) - Klien yang mengirim informasi ke broker untuk dibagikan kepada klien yang tertarik berdasarkan nama topik.
- **Subscribe** (Berlangganan) - Klien memberi tahu broker topik apa yang mereka minati. Ketika klien berlangganan suatu topik, pesan apa pun yang diterbitkan ke broker didistribusikan ke pelanggan topik itu. Klien juga dapat berhenti berlangganan untuk berhenti menerima pesan dari broker tentang topik itu.
- **QoS** (*Quality of Service*) - Kualitas Layanan. Setiap koneksi dapat menentukan kualitas layanan ke broker dengan nilai integer mulai dari 0-2. QoS tidak mempengaruhi penanganan transmisi data TCP, hanya antara klien MQTT.
 - 0 menentukan paling banyak sekali, atau sekali dan hanya sekali tanpa memerlukan pemberitahuan pengiriman. Ini sering disebut sebagai *fire* (api) dan *forget* (lupa).
 - 1 menentukan setidaknya satu kali. Pesan itu dikirim beberapa kali sampai sebuah pengakuan diterima, dikenal sebaliknya sebagai pengiriman yang diakui.
 - 2 menentukan tepat sekali. Klien pengirim dan penerima menggunakan jabatan tangan dua tingkat untuk memastikan hanya satu salinan pesan yang diterima, yang dikenal sebagai pengiriman yang terjamin.

Di MQTT, penerbit dan pelanggan (atau klien) tidak perlu saling mengenal identitas satu sama lain. MQTT mengirimkan informasi dari sumber ke tujuan dan diimplementasikan pada lapisan TCP. MQTT lebih cocok untuk simpul IoT yang memiliki kemampuan dan aset terbatas. Setiap koneksi MQTT mempertimbangkan dua jenis agen: yang pertama adalah klien MQTT dan yang lainnya adalah *server* broker MQTT. Informasi yang dikirimkan oleh protokol dikenal sebagai pesan aplikasi. Klien MQTT mengacu pada perangkat atau objek yang terhubung ke jaringan yang mengambil bagian dalam komunikasi atau pertukaran pesan melalui MQTT. Klien MQTT disebut sebagai penerbit dan pelanggan. Penerbit dapat mengirim pesan aplikasi dan pelanggan dapat meminta pesan aplikasi itu untuk mendapatkan informasi yang terkait dengan pesan itu. Pialang memungkinkan klien yang berbeda untuk terhubung satu sama lain. Ini mengakui dan mentransmisikan pesan aplikasi di antara berbagai klien yang terkait dengannya. Klien MQTT dapat berupa sensor, perangkat seluler, dll.

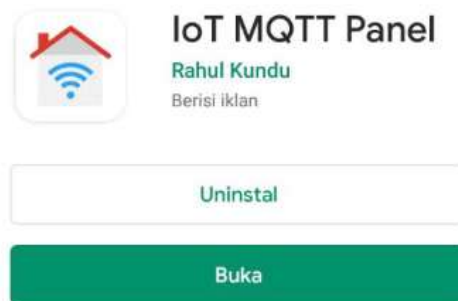
Sederhananya adalah MQTT adalah protokol untuk menyampaikan pesan dari server ke klien maupun sebaliknya. Kelebihan dari MQTT antara lain: mengirim pesan secepat mungkin, meminimalisir *encoding* dan *decoding* data, dan memanfaatkan *storage* (penyimpanan) sekecil mungkin.

Selanjutnya untuk kebutuhan pengendalian robot cerdas secara *mobile* menggunakan ponsel, dibutuhkan *Android Apps* IoT MQTT Panel. Silahkan dicari IoT MQTT Panel di *Play Store* dengan tampilan seperti diperlihatkan pada Gambar 3.14.



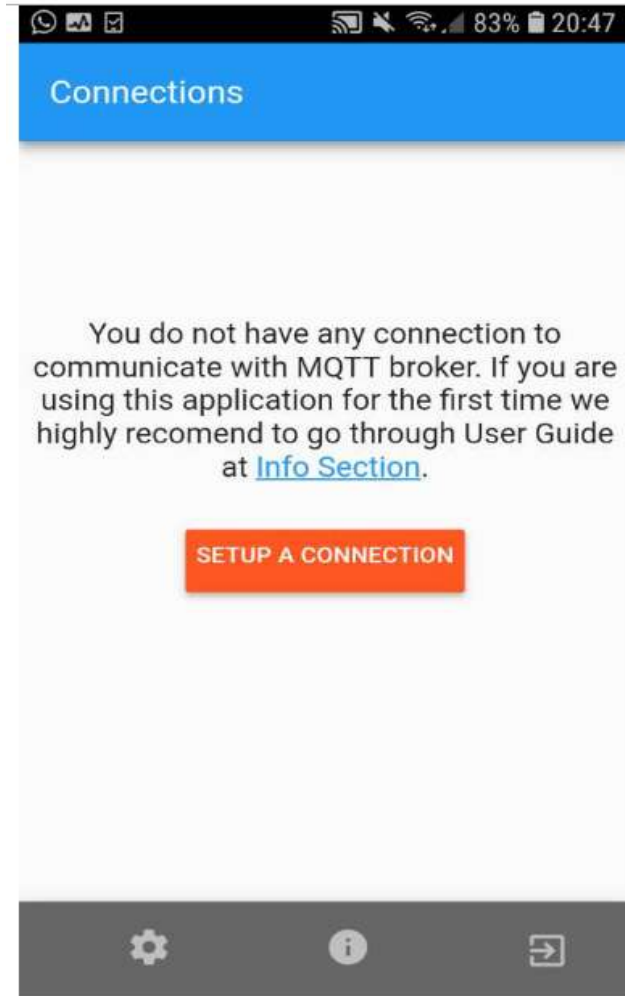
Gambar 3.14. IoT MQTT Panel

Silahkan diinstal di ponsel, aplikasi IoT MQTT Panel, sampai aplikasi siap digunakan.



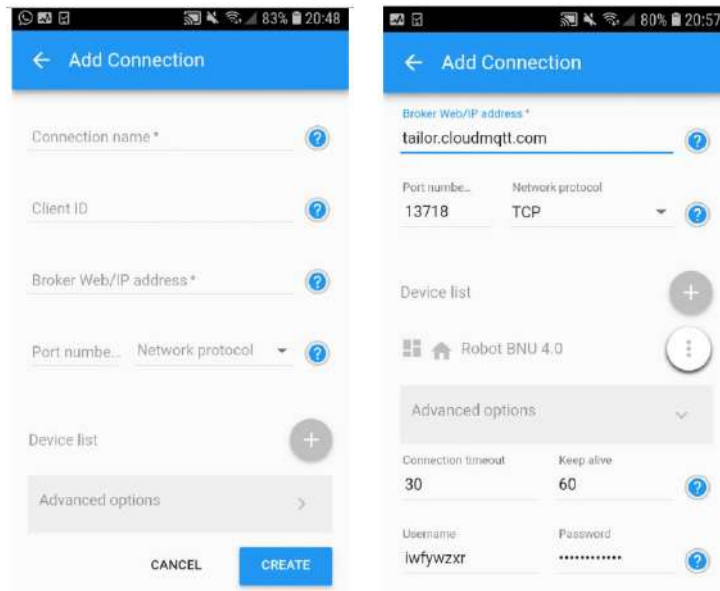
Gambar 3.15. IoT MQTT Panel siap digunakan

Jika baru pertamakali dijalankan, akan muncul seperti pada Gambar 3.16, silahkan tekan tombol *Setup A Connection*.



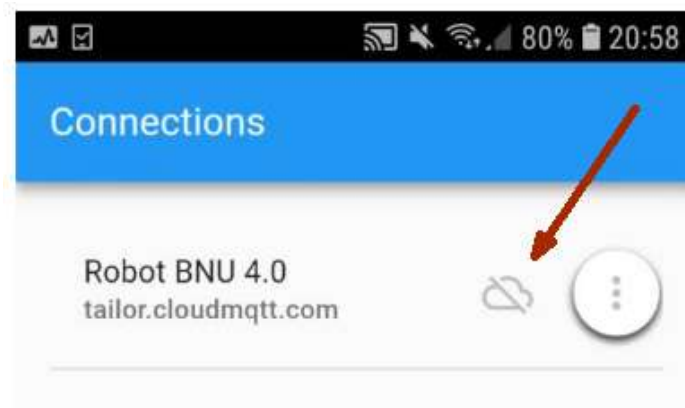
Gambar 3.16. Tampilan IoT MQTT Panel pertama kali

Setelah ditekan tombol *Setup A Connection*, silahkan tuliskan *Connection name* (nama koneksi) dan *Client ID* sembarang. Dalam contoh ini misalkan diisi dengan nama koneksi Robot BNU 4.0. Selanjutnya, silahkan diisi secara lengkap *web broker*, *port*, protokol TCP, *user*, *password*, dan seterusnya, sesuai dengan informasi detail *server cloud* IoT seperti pada Gambar 3.13. Kemudian silahkan tekan tombol *Create*.



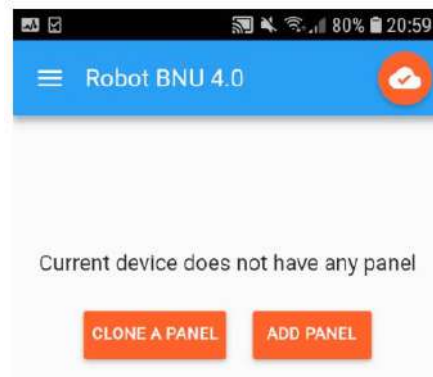
Gambar 3.17. Seting koneksi ke server cloud IoT

Setelah menekan tombol *Create*, seharusnya koneksi ke *cloud* IoT siap digunakan. Silahkan dicoba melakukan koneksi dengan menekan tanda gambar *cloud* seperti pada Gambar 3.18.



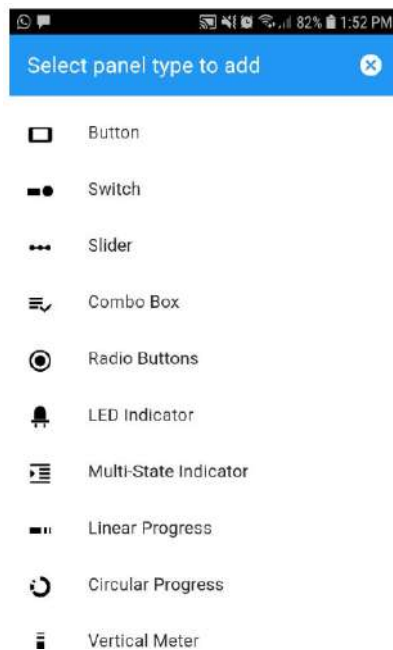
Gambar 3.18. Koneksi ke cloud IoT siap digunakan

Jika isian nama *server* (*web broker*), *user*, *password*, *port*, dan lain-lain sudah benar, seharusnya koneksi ke *server cloud* IoT akan sukses. Jika koneksi sukses dan belum ada panel yang pernah dibuat sebelumnya, maka akan muncul seperti pada Gambar 3.19.



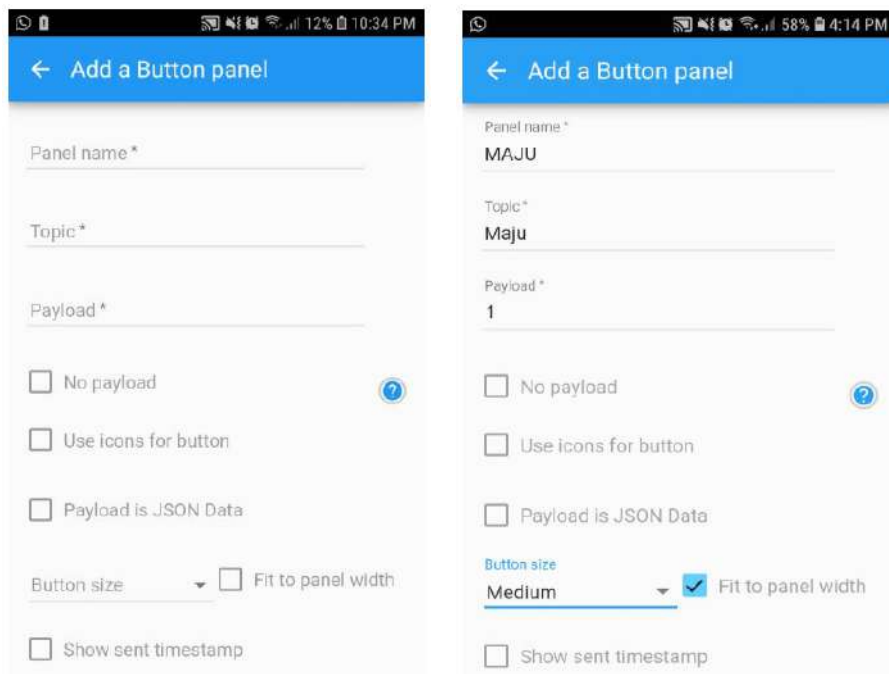
Gambar 3.19. Belum ada panel

Silahkan tekan tombol *Add Panel*, maka akan muncul banyak pilihan panel sesuai kebutuhan. Lihat Gambar 3.20. Untuk pengendalian robot BNU 4.0 ini cukup gunakan panel *Button* sebanyak empat (yaitu tombol maju, mundur, belok kiri, dan belok kanan).



Gambar 3.20. Pilihan panel

Silahkan tekan Panel *Button*. Isikan *Panel name* (nama panel) misalkan MAJU, nantinya untuk tombol MAJU. Digunakan untuk gerak maju robot cerdas. Kemudian, *topic* (topik) adalah nama tentang pesan. Klien (mikrokontroller NodeMCU V3) mempublikasikan, berlangganan, atau melakukan keduanya untuk suatu topik. Misalkan isikan topik dengan Maju dan isikan *payload* dengan 1. Lihat Gambar 3.21.



The image shows two screenshots of a mobile application interface for adding a button panel. The left screenshot shows the form with empty fields for Panel name, Topic, and Payload. The right screenshot shows the form filled with 'MAJU' for Panel name, 'Maju' for Topic, and '1' for Payload. The 'Fit to panel width' checkbox is checked in the right screenshot.

Gambar 3.21. Contoh isian panel

Setelah diisi nama panel, topik, *payload*, ukuran panel *Medium*, dan dicentang lebar panel (*Fit to panel width*), selanjutnya tekan tombol *Create*. Setelah jadi satu tombol, silahkan dilakukan cara yang sama, untuk ketiga tombol yang lain (Kiri, Kanan, dan Mundur). Masing-masing dengan *payload* 2,3, dan 4. Kemudian silahkan dilakukan penyesuaian letak dan ukuran panel. Sehingga setelah keempat tombol lengkap, bentuk panel pengendalian robot cerdas, menjadi seperti pada Gambar 3.22. Selanjutnya untuk pemrograman mikrokontroller NodeMCU V3 sebagai klien yang akan berlangganan dan mengirimkan pesan atau data ke *cloud* IoT dibahas pada Bab 7.



Gambar 3.22. Panel kendali robot cerdas via ponsel

BAB 4

Pemrograman Mikrokontroller

4.1. Mikrokontroller NodeMCU V3

NodeMCU V3 adalah *board* pengembangan yang mengintegrasikan *system on chip* (SoC) ESP8266, dirancang dan diproduksi oleh Espressif Systems (Anzola, Jiménez and Tarazona, 2019). *Board* ini termasuk mikrokontroler 32-bit, daya rendah (Tensilica L106), modul Wifi 2,4 GHz, memori RAM 50 kB, konverter analog-ke-digital (ADC) dan 17 pin tujuan umum input dan output atau *general-purpose input/output* (GPIO). NodeMCU V3 berukuran panjang 4.83cm, lebar 2.54cm, dan berat 7 gram. Karena biayanya yang rendah, NodeMCU V3 adalah salah satu platform terbaik bersifat *opensource* untuk pengembangan aplikasi IoT.

NodeMCU V3 telah memaket ESP8266 ke dalam sebuah *board* yang kompak dengan berbagai fitur layaknya mikrokontroler + kapabilitas akses terhadap Wifi juga *chip* komunikasi *USB to serial*. Sehingga untuk memprogramnya hanya diperlukan ekstensi kabel data USB persis yang digunakan sebagai kabel data dan kabel *charging smartphone* Android.

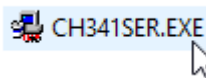
Spesifikasi yang dimiliki oleh NodeMCU V3 sebagai berikut:

- 1) *Board* ini berbasis ESP8266 serial Wifi SoC (*Single on Chip*) dengan *onboard USB to TTL*. *Wireless* yang digunakan adalah IEEE 802.11b/g/n.
- 2) 2 tantalum kapasitor 100 *micro farad* dan 10 *micro farad*.

- 3) EN: *Chip Enable, Active High*
- 4) IO16: GPIO16, dapat digunakan untuk membangunkan *chipset* dari mode *deep sleep*
- 5) IO14: GPIO14; HSPI_CLK
- 6) IO12: GPIO12; HSPI_MISO
- 7) IO13: GPIO13; HSPI_MOSI; UART0_CTS
- 8) VCC: Catu daya 3.3V (VDD)
- 9) CS0: *Chip selection*
- 10) MISO: *Slave output, Main input*
- 11) IO9: GPIO9
- 12) IO10 GBIO10
- 13) MOSI: *Main output slave input*
- 14) SCLK: *Clock*
- 15) GND: *Ground*
- 16) IO15: GPIO15; MTDO; HSPICS; UART0_RTS
- 17) IO2: GPIO2; UART1_TXD
- 18) IO0: GPIO0
- 19) IO4: GPIO4
- 20) IO5: GPIO5
- 21) RXD: UART0_RXD; GPIO3
- 22) TXD: UART0_TXD; GPIO1

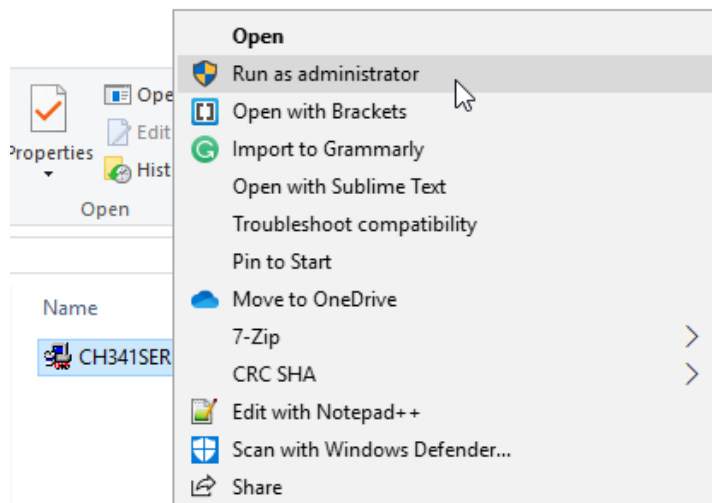
4.2. Instalasi Driver NodeMCU V3

Selanjutnya agar *board* NodeMCU V3 dikenali oleh komputer atau laptop kita, maka terlebih dahulu harus diinstal *driver*-nya. *Driver* NodeMCU V3 dapat diperoleh melalui alamat berikut: <https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers>. Untuk yang versi Windows, silahkan pilih CH341SER_WINDOWS.zip. Setelah didownload, silahkan diekstrak. Setelah diekstrak, tampil seperti pada Gambar 4.2.



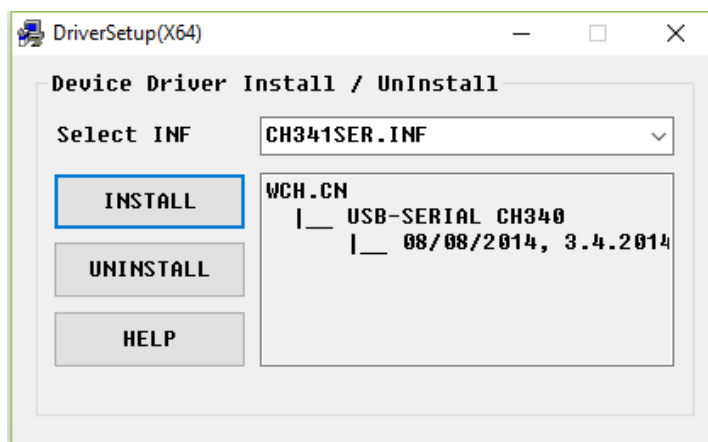
Gambar 4.2. Driver NodeMCU V3

Setelah itu silahkan diinstal driver tersebut menggunakan level Administrator.



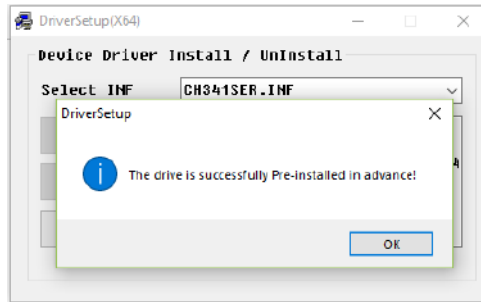
Gambar 4.3. Instalasi driver NodeMCU V3 sebagai Administrator

Jika muncul pertanyaan *Do you want to allow this app to make changes to your device?* Silahkan jawab (tekan) *Yes*. Selanjutnya akan muncul seperti pada Gambar 4.4. Silahkan tekan *Install*, dan tunggu sampai proses instalasi driver selesai.



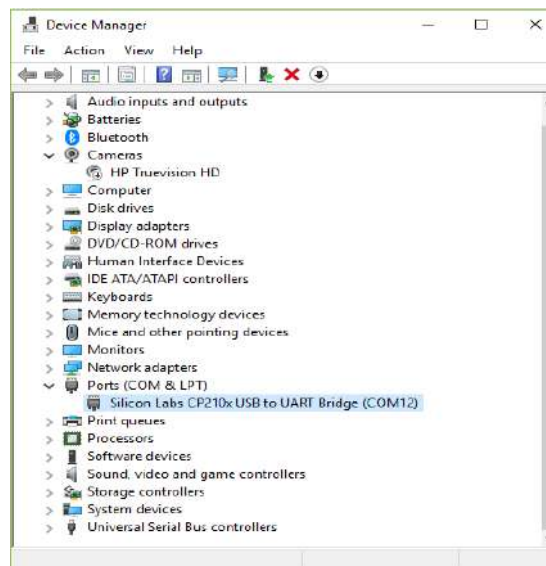
Gambar 4.4. Instalasi driver NodeMCU

Jika proses instalasi driver sukses, maka akan muncul seperti pada Gambar 4.5. Silahkan tekan OK.



Gambar 4.5. Instalasi driver NodeMCU V3 sukses

Selanjutnya periksa koneksi NodeMCU V3 ke komputer atau laptop. Posisi mikrokontroller NodeMCU V3 terhubung ke komputer melalui *port* USB. Perhatikan pada *Device Manager*, lihat pada bagian *Ports (COM & LPT)*. Pada contoh buku ini, NodeMCU V3 terhubung ke COM12. Lihat Gambar 4.6.



Gambar 4.6. Posisi NodeMCU V3 di port komputer

4.3. Instalasi Arduino IDE

Selanjutnya, setelah instalasi *driver*, untuk keperluan pemrograman mikrokontroller NodeMCU V3 butuh aplikasi Arduino IDE (*Integrated Development Environment*). Aplikasi

Arduino IDE berfungsi untuk membuat, membuka, dan mengedit program yang akan ditanam (di-embed) ke dalam *board* Arduino. Aplikasi Arduino IDE dirancang agar memudahkan penggunaanya dalam membuat berbagai aplikasi. Arduino IDE memiliki struktur bahasa pemrograman yang sederhana dan fungsi yang lengkap, sehingga mudah untuk dipelajari oleh pemula sekalipun. Untuk mendapatkan aplikasi Arduino IDE dapat diunduh melalui website resmi Arduino yaitu di alamat: <https://www.Arduino.cc/en/Main/Software>.

Seperti terlihat pada Gambar 4.7, tersedia aplikasi Arduino IDE untuk berbagai sistem operasi komputer diantaranya Windows *Installer/Non Installer*, Mac OS, Linux 32 bit, Linux 54 bit, dan Linux ARM. Cukup dengan menekan link unduhnya (dalam hal ini kita pilih Windows *Installer*) maka akan muncul pilihan *download and donate* dan *just download*. Pilih *just download* maka secara otomatis file akan terunduh.



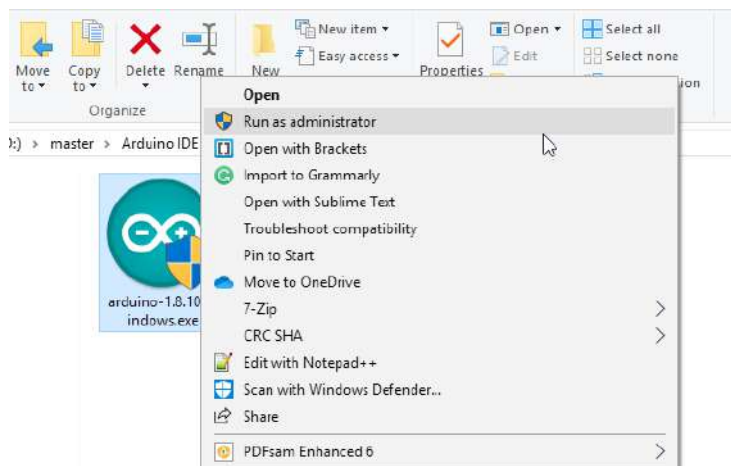
Gambar 4.7. Website resmi Arduino.cc

Selanjutnya akan dijelaskan cara instalasi aplikasi Arduino IDE pada sistem operasi Windows 10. Perlu diketahui, ada aplikasi Arduino IDE yang memerlukan instalasi dan ada yang tidak perlu instalasi.

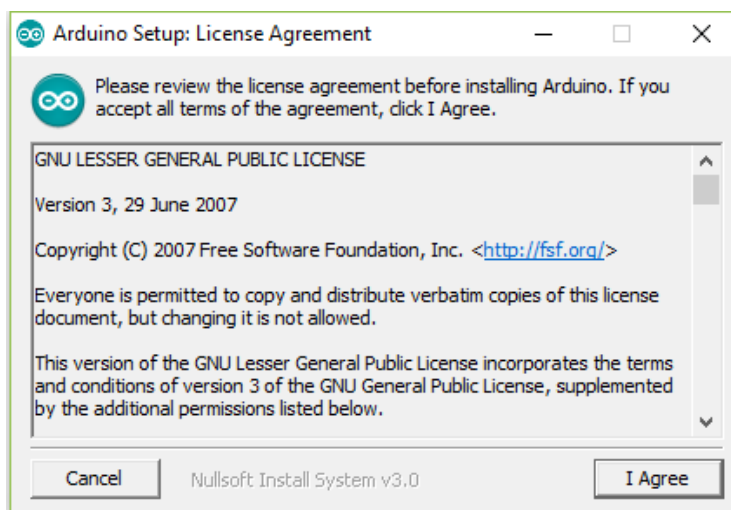
Untuk proses instalasi, silahkan buka file Arduino-1.8.10-windows.exe (1.8.10 adalah versi dari Arduino IDE), seperti pada Gambar 4.8. Klik kanan dan silahkan *install* sebagai Administrator, seperti pada Gambar 4.9. Jika muncul pertanyaan *Do you want to allow this app to make changes to your device?* Silahkan jawab (tekan) *Yes*. Maka selanjutnya akan muncul seperti pada Gambar 4.10.



Gambar 4.8. File Arduino-1.8.10-windows.exe

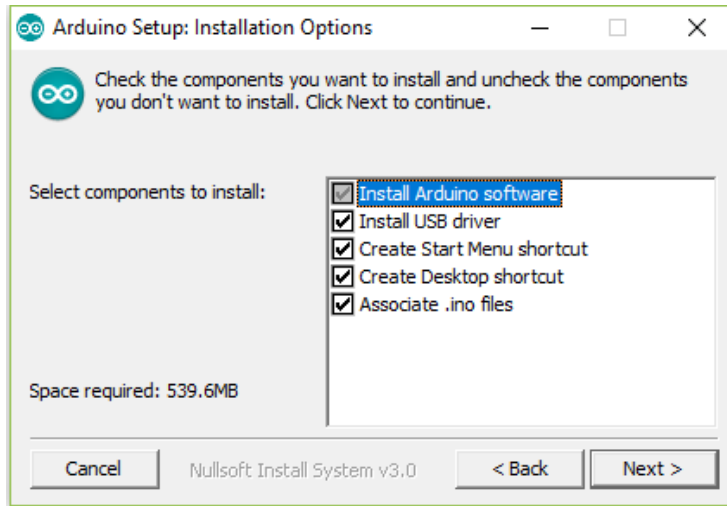


Gambar 4.9. Instalasi Arduino IDE sebagai Administrator



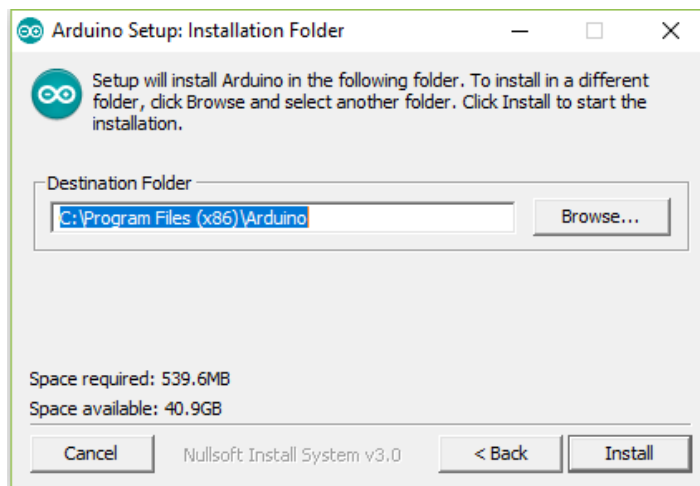
Gambar 4.10. Persetujuan instalasi aplikasi IDE Arduino

Dilanjutkan dengan menekan tombol *I Agree*, maka akan muncul jendela *Installation Option* seperti Gambar 4.11. Pastikan semua komponen terpilih/tercentang.



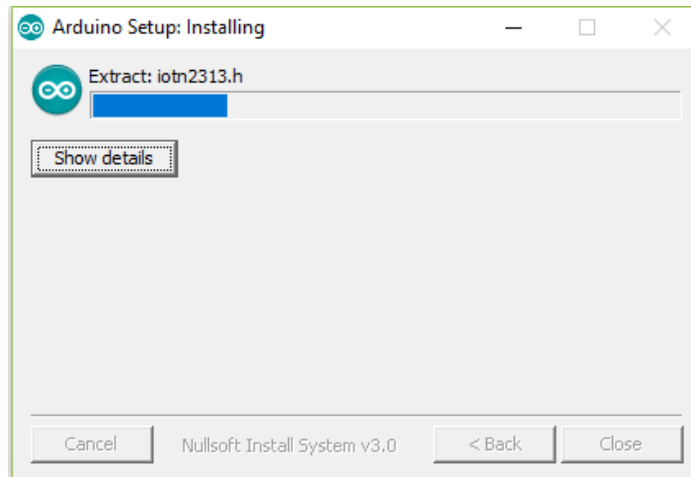
Gambar 4.11. Pilihan komponen instalasi

Tekan tombol *Next*, kemudian pilih Folder untuk menyimpan Aplikasi Arduino IDE.



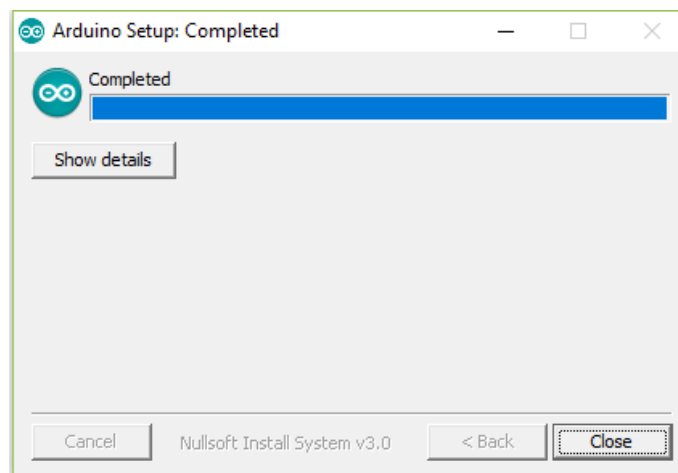
Gambar 4.12. Menentukan Folder instalasi

Tekan *Install* untuk melanjutkan ke proses instalasi.



Gambar 4.13. Proses extract dan instalasi

Silahkan tunggu sampai proses instalasi selesai.



Gambar 4.14. Instalasi selesai

Jika sudah selesai proses instalasi, silahkan tekan *Close*. Aplikasi Arduino IDE siap untuk digunakan.

Untuk membuka aplikasi Arduino IDE, carilah file hasil instalasi *Arduino.exe*, atau silahkan cari *shortcut* program Arduino IDE di desktop. Kemudian silahkan *double click* file tersebut.



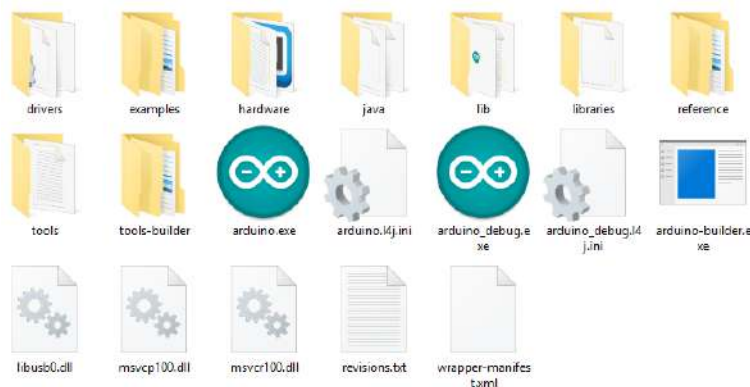
Gambar 4.15. Aplikasi Arduino IDE

Untuk aplikasi Arduino IDE yang tidak memerlukan instalasi, dapat diunduh juga dari website resmi Arduino yaitu <https://www.Arduino.cc/en/Main/Software> dan silahkan pilih *Windows ZIP file for non admin install*. Ketika muncul pilihan *download and donate* dan *just download*, silahkan pilih *just download* maka secara otomatis file akan terunduh.



Gambar 4.16. File Arduino-1.8.10-windows.zip

Setelah file Arduino IDE terunduh, silahkan diekstrak file tersebut dengan cara klik kanan → Ekstrak disini/ *Extract here*, maka hasilnya seperti pada Gambar 4.17.



Gambar 4.17. File Arduino.zip setelah diekstrak

Untuk membuka aplikasi Arduino IDE, silahkan jalankan file Arduino.exe seperti pada Gambar 4.18.



Gambar 4.18. Aplikasi Arduino IDE

Setelah dijalankan tampilan utama dari aplikasi Arduino IDE seperti terlihat pada Gambar 4.19.



Gambar 4.19. Tampilan utama aplikasi Arduino IDE v1.8.10

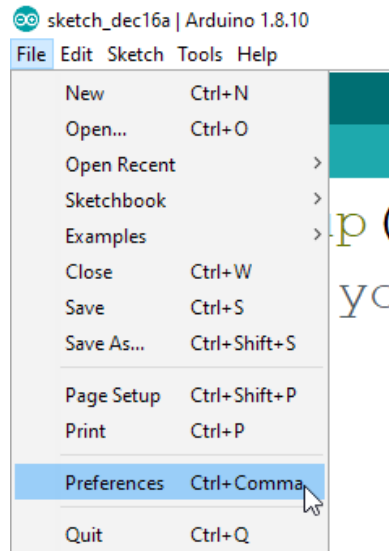
4.4. Pemrograman NodeMCU V3

Setelah aplikasi Arduino IDE terinstal, selanjutnya perlu dilakukan pengaturan sesuai dengan jenis mikrokontroler NodeMCU V3 yang digunakan. Silahkan jalankan

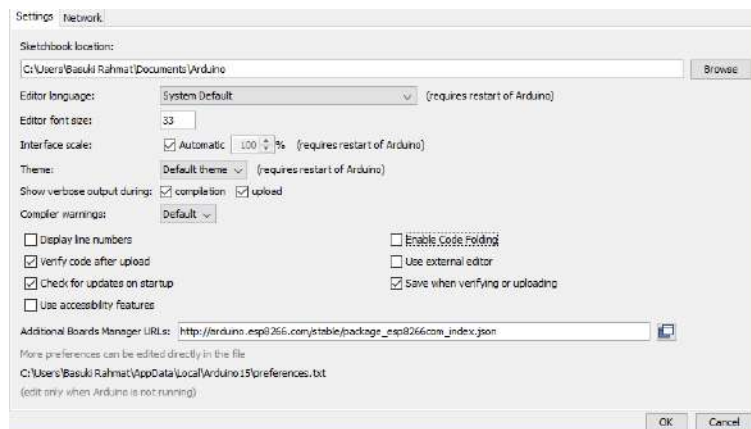
aplikasi Arduino IDE, sehingga muncul tampilan seperti pada Gambar 4.19. *Board* mikrokontroller NodeMCU V3 dalam keadaan terhubung ke komputer atau laptop.

Beberapa pengaturan Arduino IDE untuk NodeMCU V3, diuraikan sebagai berikut:

- 1). Jalankan aplikasi Arduino IDE, pada menu *File* silahkan klik pada bagian *Preference*:

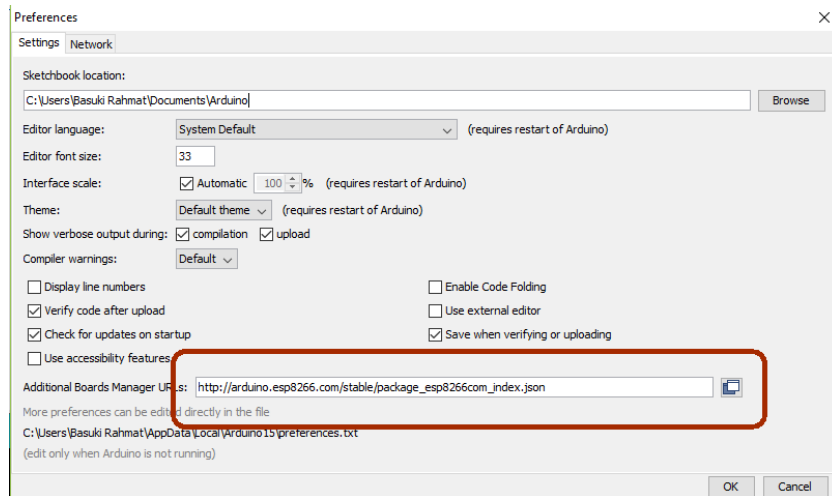


Gambar 4.20. File – Preference



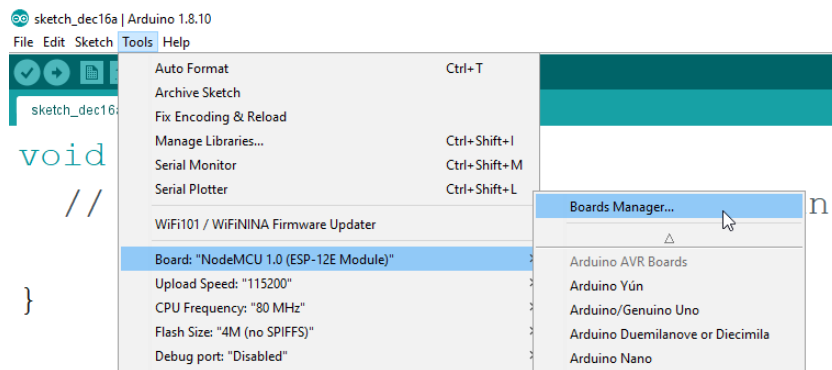
Gambar 4.21. Preferences

- 2). Pada *Additional Boards Manager URLs* tambahkan http://Arduino.esp8266.com/stable/package_esp8266com_index.json.



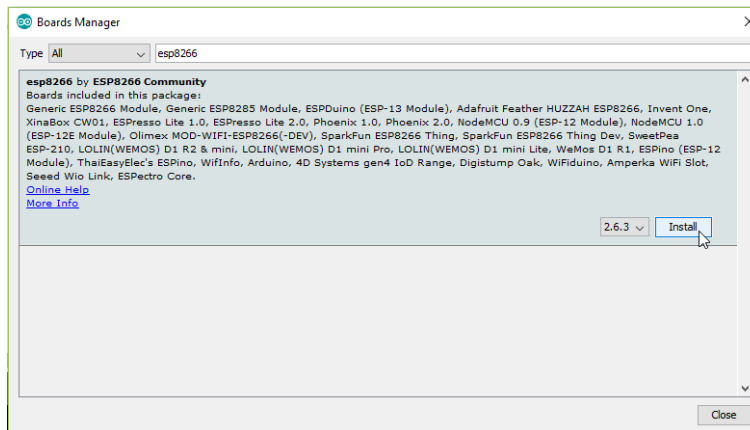
Gambar 4.22. Seting additional boards

- 3). Selanjutnya silahkan *update board*-nya. Dari menu *Tool Board Board Manager*.



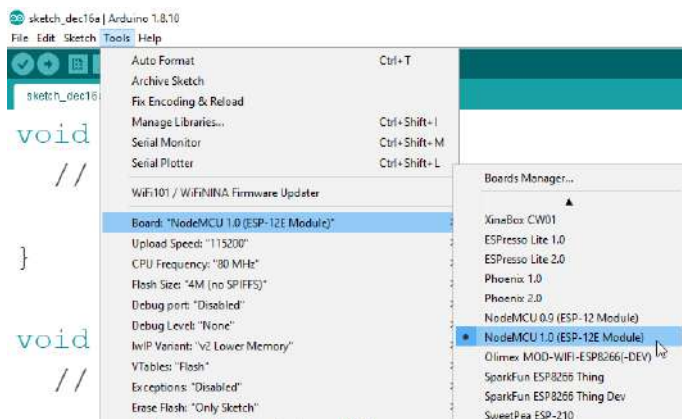
Gambar 4.23. Seting Board Manager

Akan muncul jendela *Boards Manager*. Pada bagian bawah cari esp8266, kemudian silahkan klik install.



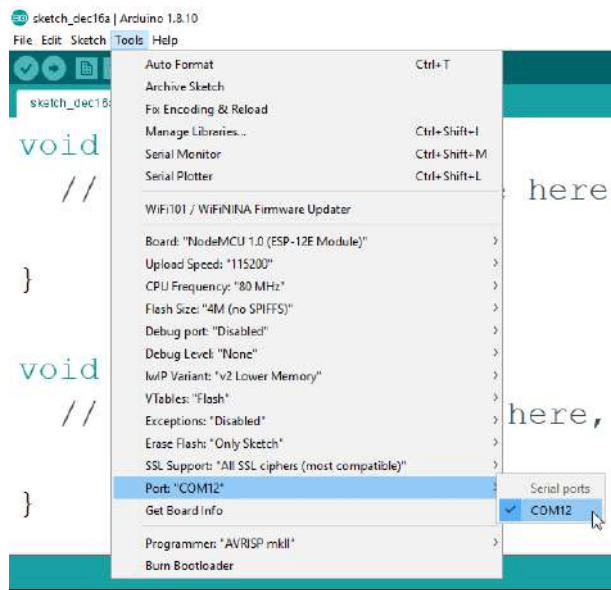
Gambar 4.24. Instal modul esp8266

- 4). Sekarang silahkan dicek, apakah NodeMCU sudah terinstal di Arduino IDE, atautkah belum. Lihat pada bagian menu *Tools - Board - NodeMCU* seperti pada Gambar 4.25.



Gambar 4.25. Board NodeMCU siap digunakan

- 5). Selanjutnya, sesuaikan *port* yang digunakan. Silahkan klik pada menu *Tools – Port*, pilih *port* yang sesuai. Pada contoh buku ini, NodeMCU V3 terhubung ke COM12 (sesuai yang muncul di *Device Manager*, Gambar 4.6).



Gambar 4.26. Pengaturan port NodeMCU V3

Selanjutnya sudah bisa dicoba, pemrograman menggunakan NodeMCU V3. Berikut ini contoh program untuk tes koneksi Wifi. Silahkan tuliskan kode program berikut ini, dan sesuaikan nama dan kata sandi Wifi yang digunakan.

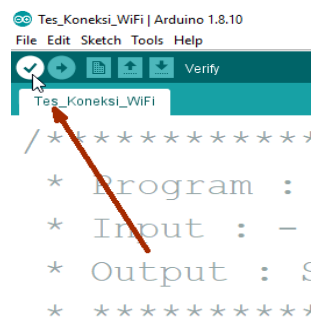
	Program Tes Koneksi WiFi NodeMCU V3
1	/******
2	* Program: Tes Koneksi WiFi NodeMCU V3
3	* Input: -
4	* Output: Serial Monitor
5	* *****/
6	
7	#include <ESP8266WiFi.h>
8	
9	const char* ssid = "alamkubur";//Ganti dengan nama Wifi anda
10	const char* password = "password_wifi";//Ganti dengan password wifi
11	anda
12	
13	WiFiServer server(80);
14	

```

15 void setup() {
16   Serial.begin(115200);
17   delay(1000);
18
19   //Connect to WiFi network
20   Serial.println();
21   Serial.println();
22   Serial.print("Connecting to ");
23   Serial.println(ssid);
24   WiFi.begin(ssid, password);
25
26   while (WiFi.status() != WL_CONNECTED) {
27     delay(500);
28     Serial.print(".");
29   }
30   Serial.println("");
31   Serial.println("WiFi connected");
32
33   server.begin();
34   Serial.println("Server started");
35   Serial.print("Use this URL to connect: ");
36   Serial.print("http://");
37   Serial.print(WiFi.localIP());
38   Serial.println("/");
39 }
40
41 void loop() {
42 }
43

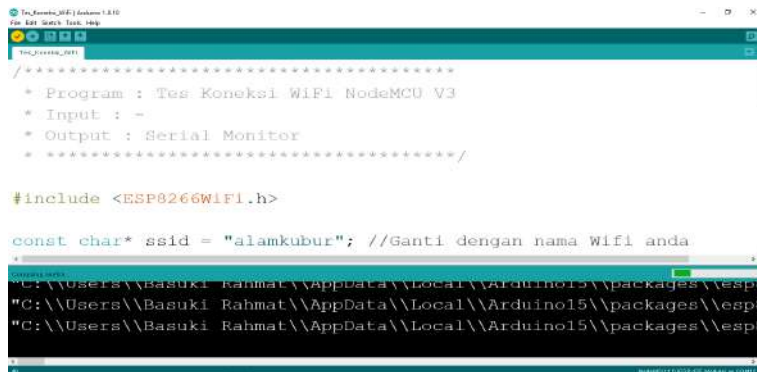
```

Setelah itu, lakukan proses verifikasi atau kompilasi dan *upload* programnya jika sudah tidak ada kesalahan.



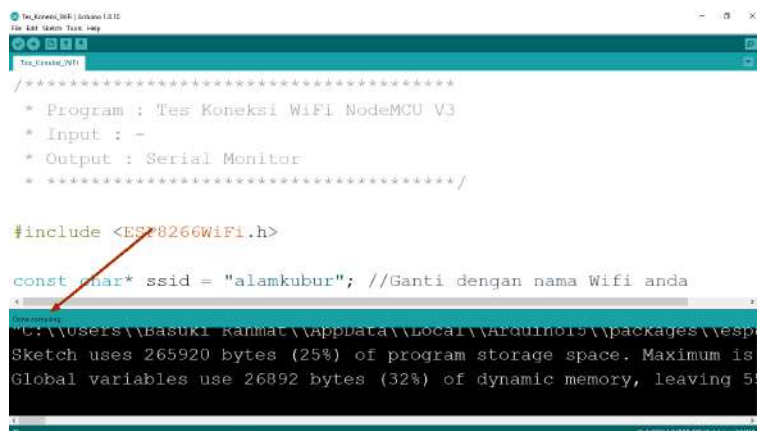
Gambar 4.27. Tombol verifikasi program

Tekan tombol proses verifikasi (*Verify*), tunggu proses verifikasi atau kompilasi program selesai dan sukses.



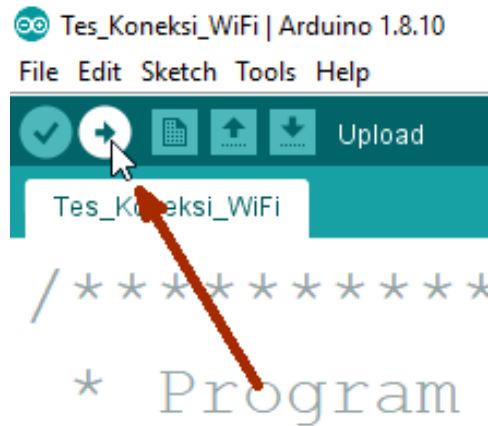
Gambar 4.28. Proses verifikasi (kompilasi)

Jika proses verifikasi atau kompilasi sudah selesai dan sukses (tidak ada kesalahan), maka terdapat tulisan *Done compiling*.



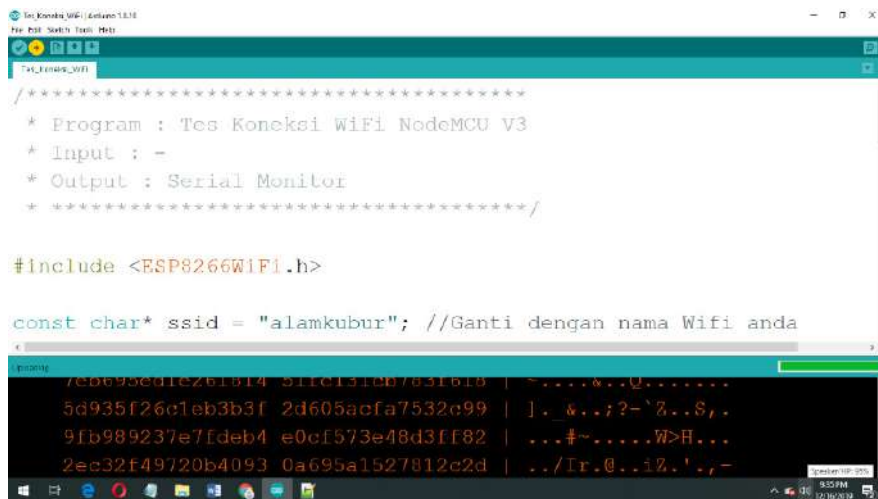
Gambar 4.29. Proses verifikasi atau kompilasi sukses

Selanjutnya untuk menanam (*embed*) program kedalam mikrokontroller NodeMCU V3, silahkan tekan tombol *Upload*.



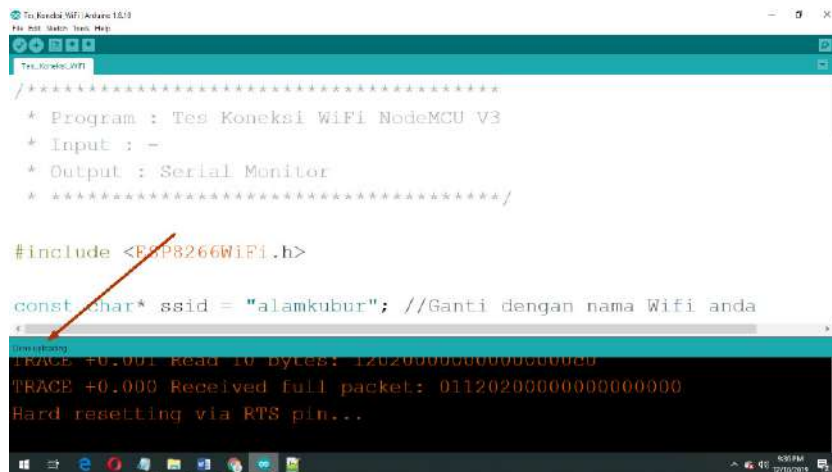
Gambar 4.30. Tombol upload program

Silahkan tekan tombol *Upload*, tunggu sampai proses *upload* program selesai.



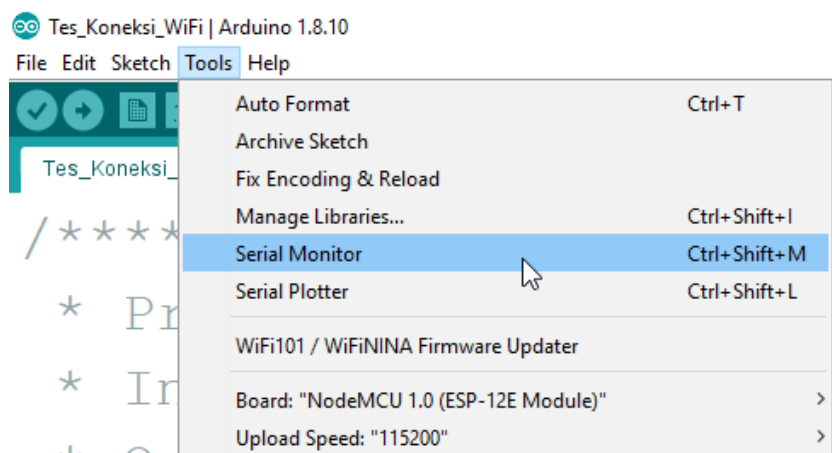
Gambar 4.31. Proses upload program

Jika proses upload program sudah selesai dan sukses (tidak ada kesalahan), maka terdapat tulisan *Done uploading*.



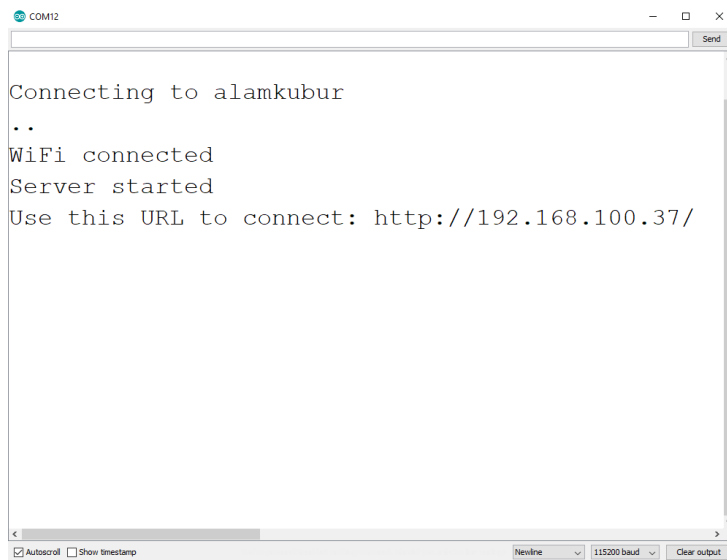
Gambar 4.32. Proses upload program sukses

Untuk melihat hasil program, silahkan arahkan ke menu *Tools – Serial Monitor*.



Gambar 4.33. Tools - serial monitor

Maka akan tampil, hasilnya di *serial monitor*, seperti pada Gambar 4.34. Jika tidak muncul, silahkan cek posisi *baud rate* 11520. Jika masih tidak muncul juga, silahkan tekan tombol RST NodeMCU V3.



Gambar 4.34. Koneksi ke Wifi sukses



BAB 5

Kamera Kinect Xbox 360

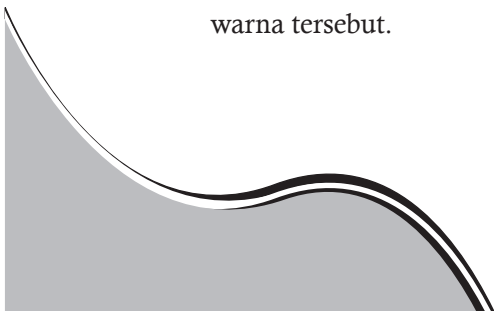
5.1. Tentang Kamera Kinect Xbox 360

Kinect adalah perangkat yang diperkenalkan pada November 2010 sebagai aksesori Xbox 360. Perangkat ini dikembangkan oleh perusahaan PrimeSense bekerja sama dengan Microsoft (Cruz, Lucio and Velho, 2012). Kinect Xbox 360 adalah *Controller-Free Gaming* dan *Gaming Experience* yang ditetapkan sebagai kontrol untuk Video game platform Xbox 360. Data yang diperoleh memiliki perbedaan dan sifat komplementer, menggabungkan geometri dengan visual atribut. Dengan kemampuan ini, Kinect secara fleksibel bisa digunakan dalam aplikasi di berbagai bidang seperti: komputer grafik, pemrosesan gambar, visi komputer, interaksi manusia-mesin, industri game, robotika, pemain teater, dan banyak penelitian lainnya.

Ada tiga inovasi perangkat keras yang bekerja bersama-sama di dalam sensor Kinect yaitu:

1. *Color VGA video camera* (Kamera video VGA berwarna)

Kamera video ini membantu dalam pengenalan wajah dan deteksi fitur lainnya dengan mendeteksi tiga komponen warna yaitu *Red*, *Green*, dan *Blue* (RGB). Microsoft menamakannya Kamera RGB dengan mengacu pada tiga komponen warna tersebut.



2. *Depth sensor* (sensor kedalaman)

Depth sensor atau sensor kedalaman merupakan sebuah proyektor *infrared* dan sebuah sensor *monochrome* CMOS yang bekerja secara bersama-sama untuk “melihat” ruangan atau area dalam bentuk tiga dimensi (3D) dengan tanpa memperdulikan kondisi cahaya.

3. *Multi-array michrophones* (Mikrofon multi-array)

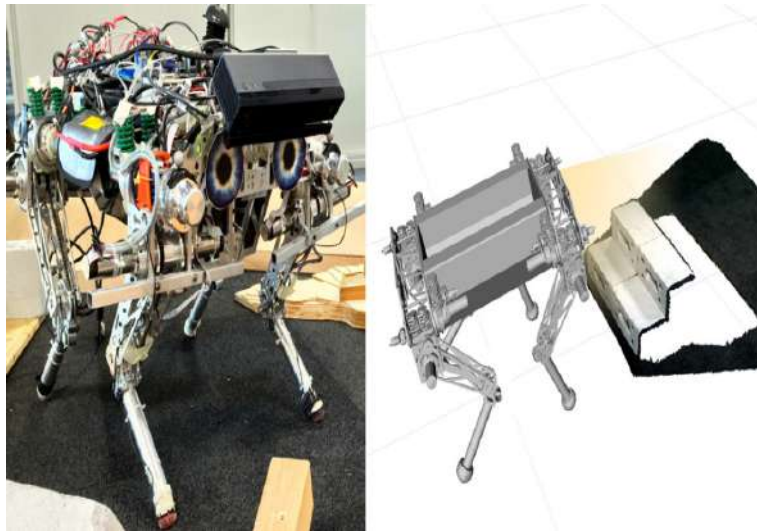
Merupakan sebuah susunan yang terdiri dari empat kamera yang dapat mengisolasi suara dari pemain dengan suara-suara lain (*noise*) yang ada di ruangan. Hal ini memungkinkan pemain game untuk berada agak jauh dari mikrofon dan masih dapat menggunakan kontrol suara atau *voice control*.

Lebih jauh tentang spesifikasi Kinect diketahui bahwa baik video ataupun sensor kedalaman memiliki resolusi sebesar 640 x 480 pixel dan berjalan pada 30 *Frame Per Second*. Dari spesifikasi Kinect juga diketahui bahwa bisa berada pada jarak sekitar 1.8 meter antara objek dan kamera kinect. Gambaran kamera ini diperlihatkan pada Gambar 5.1.



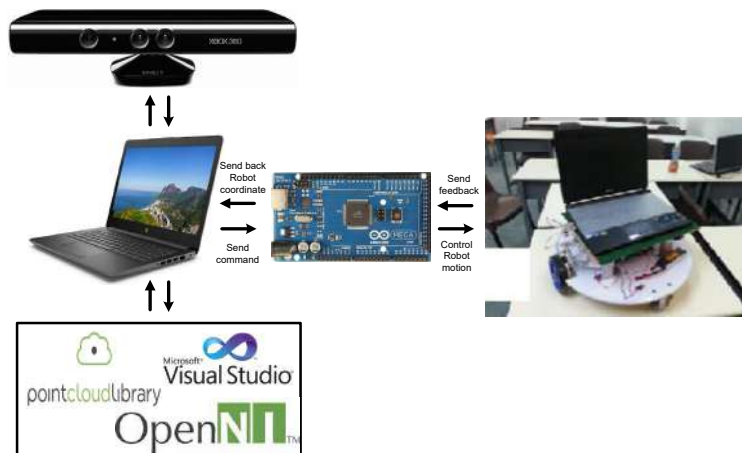
Gambar 5.1. Kamera Kinect Xbox 360

Penelitian tentang robot cerdas sebelumnya yang memanfaatkan Kinect sebagai sensor andalannya, antara lain robot berkaki empat StarLETH menggunakan sensor Kinect v2 untuk pemetaan medan kasar dan penggerak (Fankhauser *et al.*, 2015).



Gambar 5.2. Robot Kinect StarLETH (Fankhauser *et al.*, 2015)

Penelitian lain tentang robot cerdas yang memanfaatkan Kinect sebagai sensor utamanya, yaitu penelitian navigasi *Autonomous Mobile Robot* (Zainuddin *et al.*, 2015). Penelitian ini menyajikan navigasi otonom dari robot seluler dengan menggunakan sensor Kinect. Dengan menggunakan Microsoft Kinect Xbox 360 sebagai sensor utama, robot dapat menavigasi dan menghindari rintangan dengan aman. Dengan menggunakan data kedalaman, awan titik 3D, proses penyaringan dan pengelompokan, sensor Kinect dapat membedakan hambatan dan jalur untuk menavigasi dengan aman.



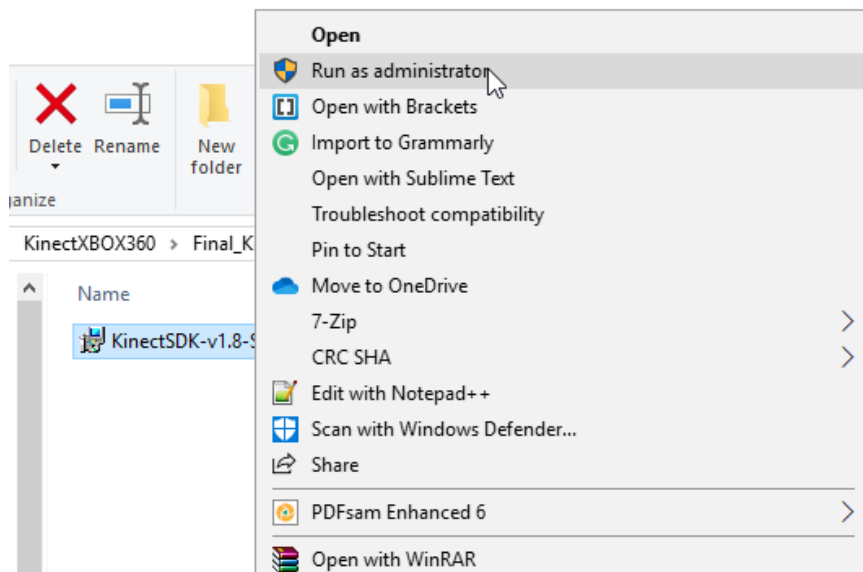
Gambar 5.3. Autonomous Mobile Robot (Zainuddin *et al.*, 2015)

5.2. Instalasi Kamera Kinect Xbox 360

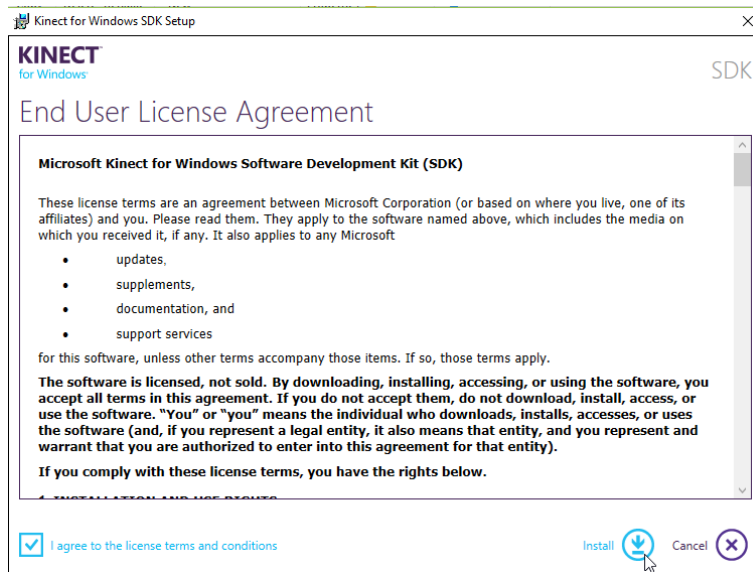
Instalasi kamera Kinect Xbox 360 membutuhkan *software development kit* (SDK). SDK yang dibutuhkan (yang perlu diinstal), agar kamera Kinect Xbox 360 dapat digunakan di komputer atau laptop dengan sistem operasi Windows 10, yaitu: *Kinect for Windows SDK v1.8*. Cara instalasinya, dijelaskan dalam uraian berikut ini.

Cara instal *Kinect for Windows SDK v1.8*

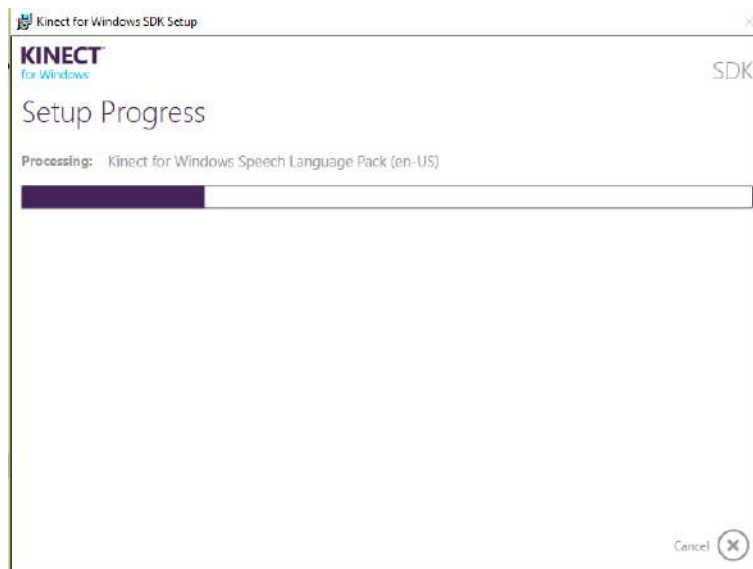
Untuk mendapatkan SDK ini, silahkan unduh di alamat: <https://www.microsoft.com/en-us/download/details.aspx?id=40278>. Setelah SDK telah berhasil diunduh, silahkan klik kanan dan instal sebagai Administrator. Lihat Gambar 5.4. Jika muncul pertanyaan *Do you want to allow this app to make changes to your device?* Silahkan jawab (tekan) *Yes*. Selanjutnya akan muncul seperti pada Gambar 5.5. Persetujuan perjanjian lisensi SDK Kinect dan silahkan klik *Install*. Tunggu sampai proses instalasi selesai. Jika proses instalasi selesai, silahkan tekan tombol *Close*. Lihat Gambar 5.7



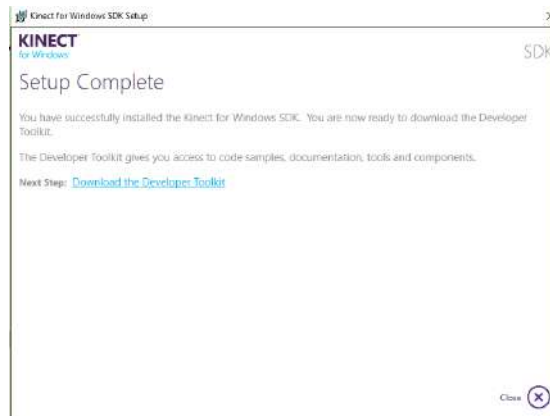
Gambar 5.4. Instal SDK Kinect sebagai Administrator



Gambar 5.5. Lisensi SDK Kinect

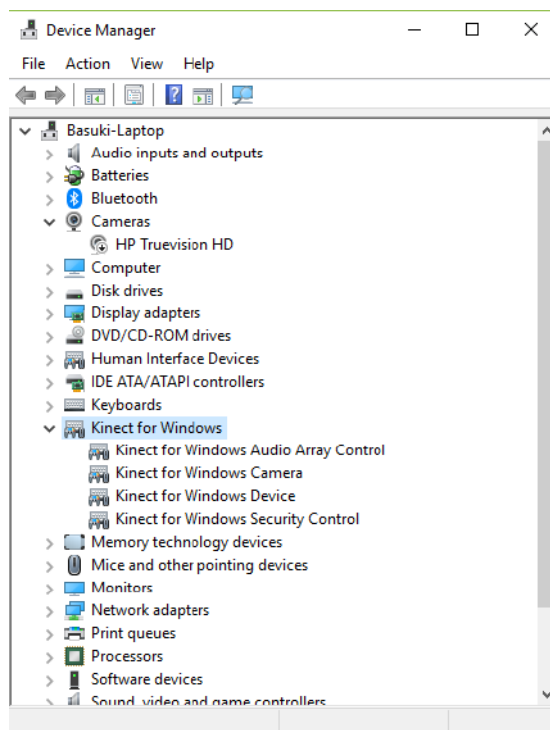


Gambar 5.6. Proses instalasi SDK Kinect



Gambar 5.7. Proses instalasi SDK Kinect selesai

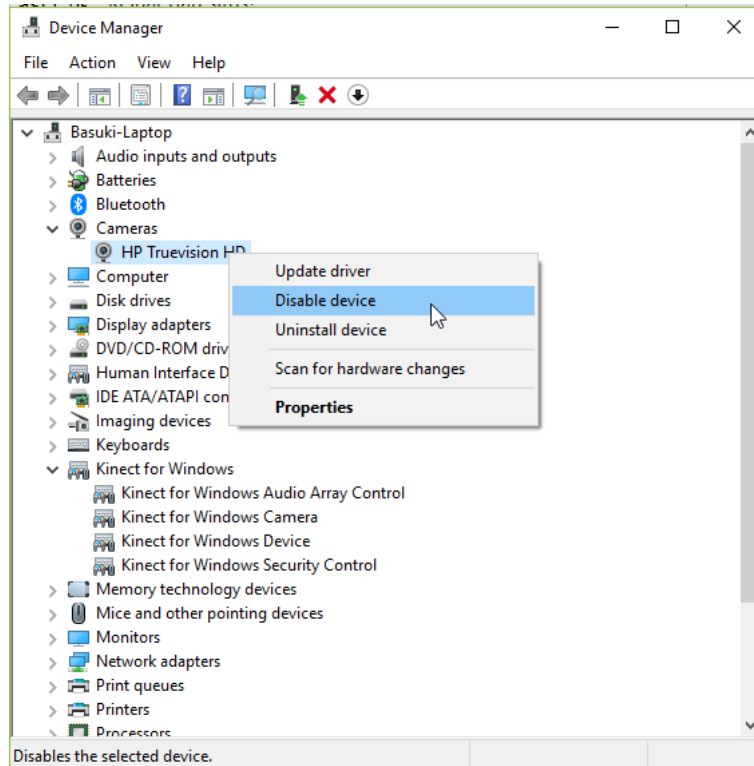
Setelah instal SDK, selanjutnya kamera Kinect Xbox 360 dalam posisi terhubung ke komputer atau laptop, dan kabel *power* terhubung ke listrik, silahkan jalankan *Device Manager*. Periksalah apakah hardware Kinect telah berhasil dikenali. Jika proses instalasi sukses, seharusnya akan muncul seperti terlihat pada Gambar 5.8.



Gambar 5.8. Kinect sudah muncul di Device Manager

5.3. Pemrograman Kamera Kinect Xbox 360

Selanjutnya, kamera yang digunakan oleh Robot Cerdas BNU 4.0 adalah kamera Kinect. Oleh karena itu kamera *webcam* komputer atau laptop harus di-*disable*. Caranya, masuk ke *Device manager*, cari kamera *default*-nya, kemudian silahkan di-*disable*. Lihat Gambar 5.9.



Gambar 5.9. Kamera webcam komputer di-disable

Dengan demikian, untuk selanjutnya kamera Kinect Xbox 360 difungsikan seperti kamera *webcam*.

5.3.1 Pemrograman Python Melalui Lingkungan Anaconda

Lihat kembali gambaran rancangan kerja sistem Robot Cerdas BNU 4.0 pada Gambar 2.4 di Bab 2. Dari gambaran tersebut dapat dilihat bahwa untuk keperluan pengenalan objek digunakan Bahasa Pemrograman Python. Termasuk bagaimana memfungsikan kamera Kinect Xbox 360 sebagai *webcam* menggunakan Python.

Jika belum familier dengan pemrograman Python, berikut secara ringkas diuraikan bagaimana memprogram Python secara mudah melalui lingkungan Anaconda. Melalui

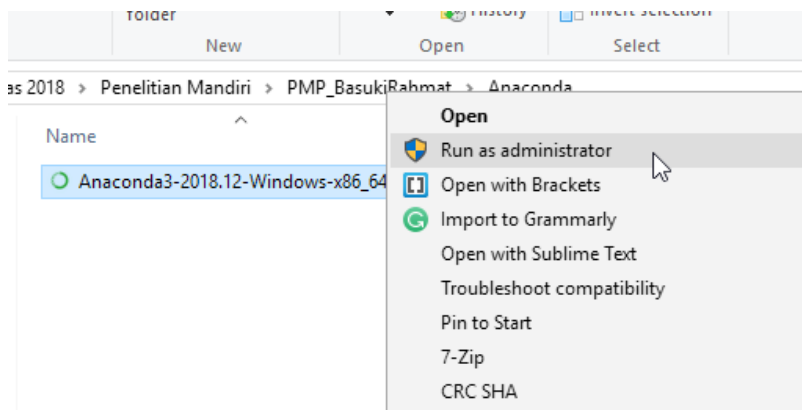
Anaconda, Python dapat dijalankan dengan dua cara, yaitu berbasis web menggunakan Jupyter Notebook dan berbasis desktop menggunakan Spyder. Untuk mendapatkan *software* Anaconda, silahkan diunduh secara gratis di alamat <https://www.anaconda.com/download>, seperti terlihat pada Gambar 5.10.



Gambar 5.10. Unduh Anaconda

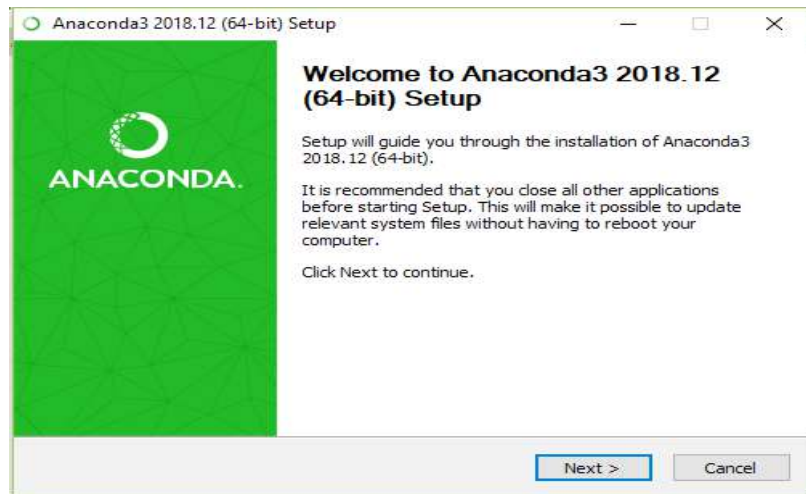
Selanjutnya, silahkan dilakukan instalasi Anaconda. Proses instalasi Anaconda di Windows 10, sesuai langkah-langkah berikut ini.

1. Setelah unduh Anaconda, silahkan dilakukan proses instalasi dengan *run as administrator*, seperti terlihat pada Gambar 5.11.



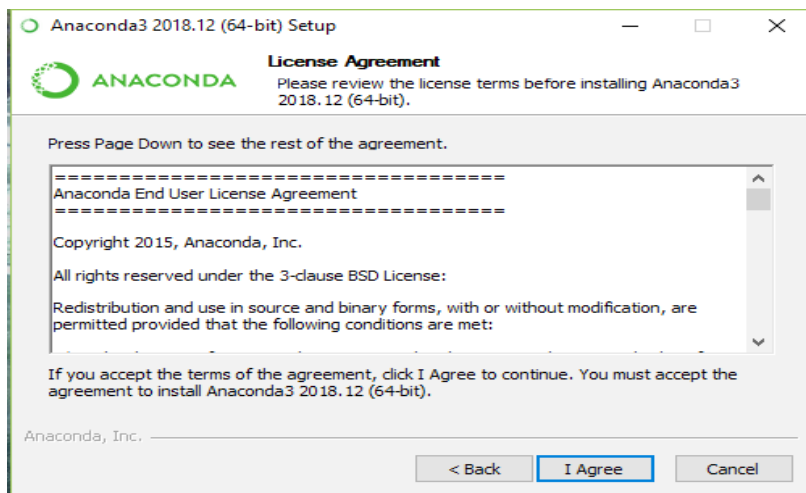
Gambar 5.11. Instal Anaconda

2. Silahkan klik Next.



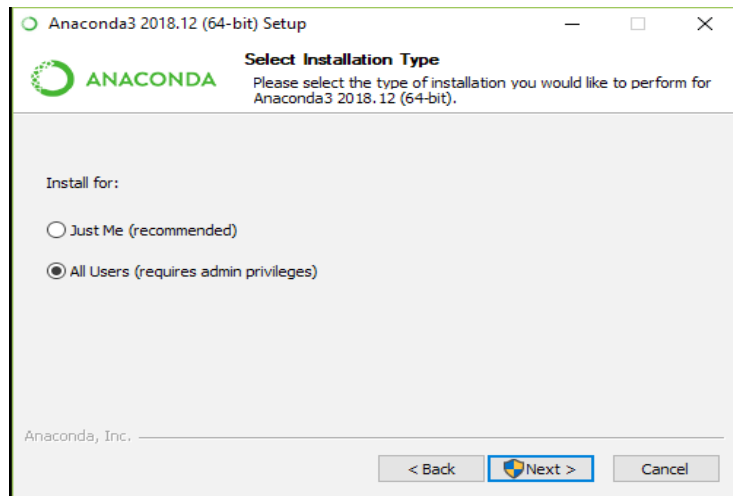
Gambar 5.12. Tekan Next

3. Kemudian baca lisensi dan klik *I Agree*.



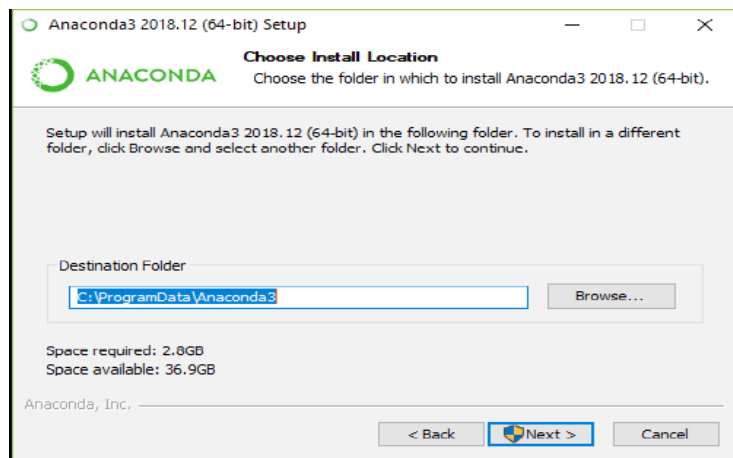
Gambar 5.13. Lisensi anaconda

4. Pilih *Intall for: All Users*, lalu tekan Next.



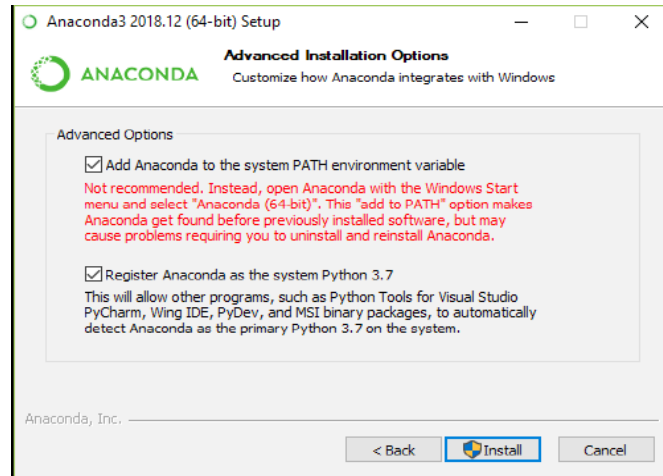
Gambar 5.14. Pilihan jenis instalasi

5. Silahkan pilih lokasi instalasi, lalu tekan Next.



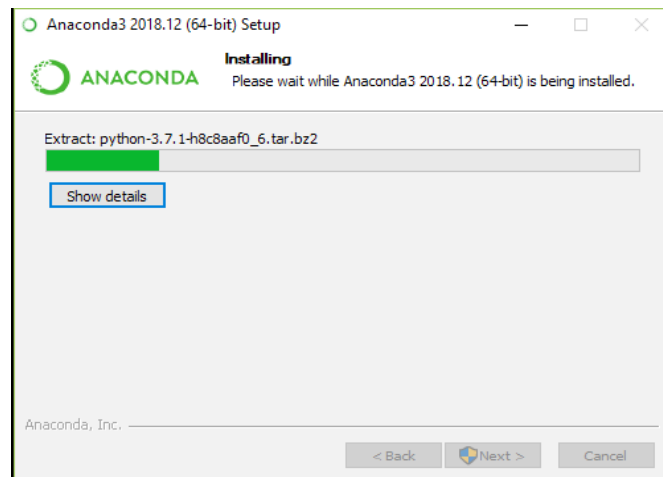
Gambar 5.15. Pilihan lokasi instalasi

6. Selanjutnya pilihan integrasi anaconda dengan lingkungan windows, pilih *Add Anaconda to the system PATH environment variable*, lalu tekan *Install*.



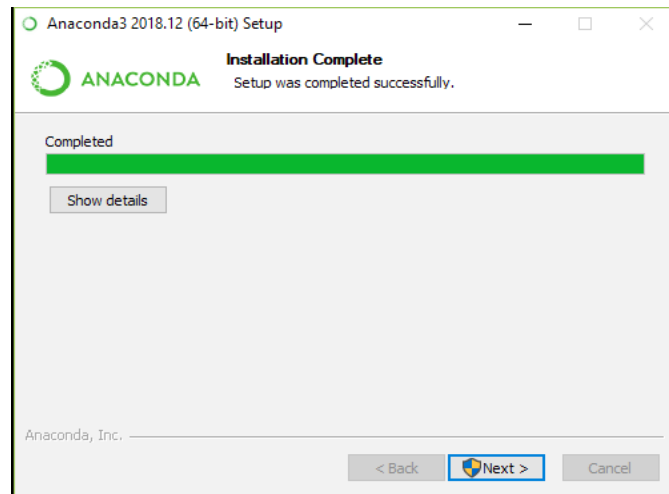
Gambar 5.16. Pilihan integrasi dengan windows

7. Setelah tekan Install, tinggal tunggu sampai proses instalasi selesai.



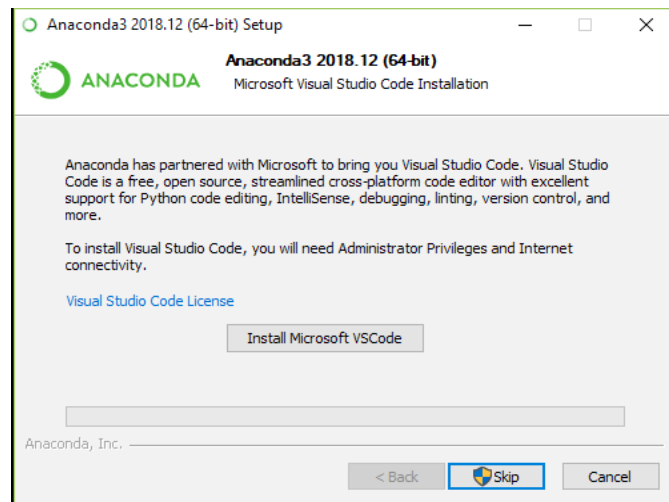
Gambar 5.17. Proses instalasi

8. Proses instalasi selesai, tekan Next.



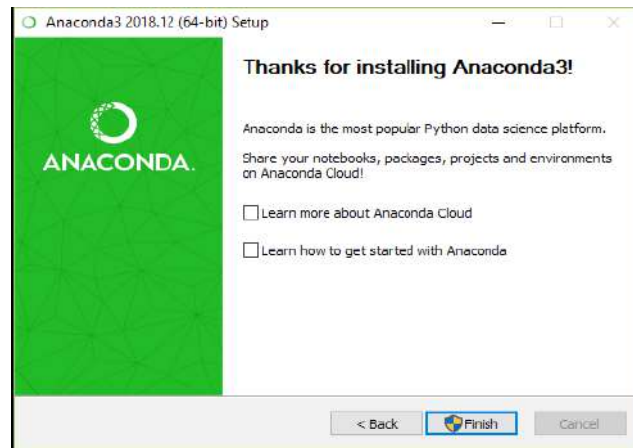
Gambar 5.18. Instalasi selesai

9. Selanjutnya, ada pilihan instal Microsoft Visual Studio Code (VSCode), dapat dilewati dengan menekan tombol Skip.



Gambar 5.19. Pilihan instal VSCode

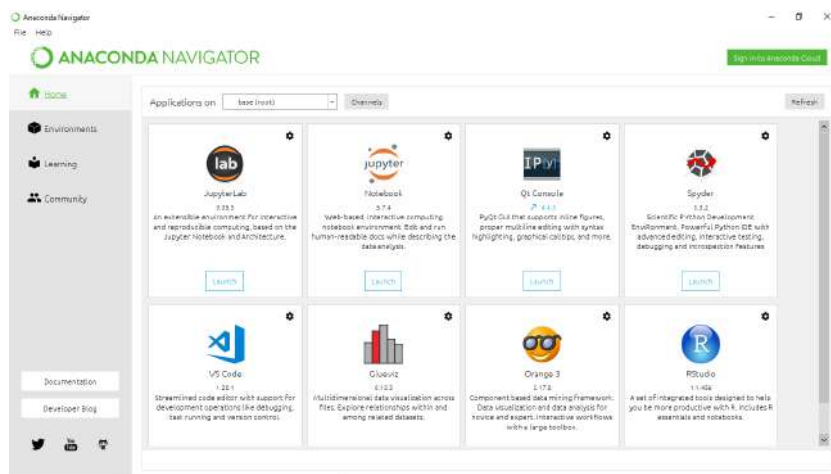
10. Instalasi anaconda selesai, tinggal tekan Finish.



Gambar 5.20. Instalasi anaconda selesai

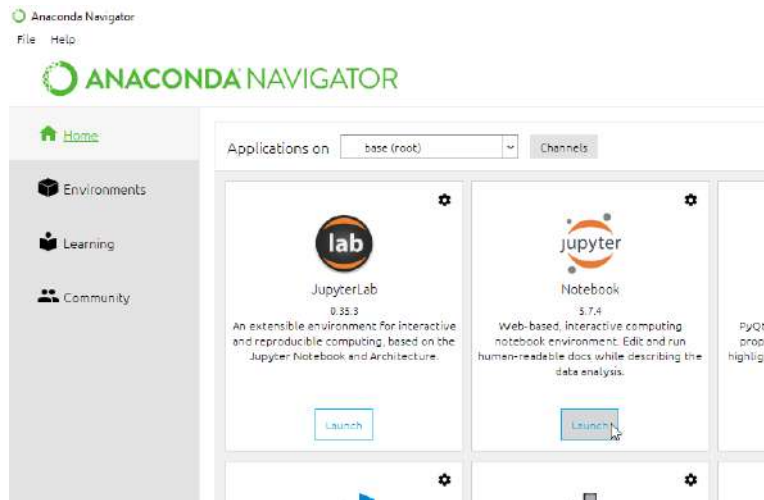
Python Menggunakan Jupyter Notebook

Selanjutnya, untuk memulai pemrograman Python melalui lingkungan Anaconda, silahkan dijalankan Anaconda di windows. Jalankan *Anaconda Navigator* sebagai administrator, *run as administrator*.



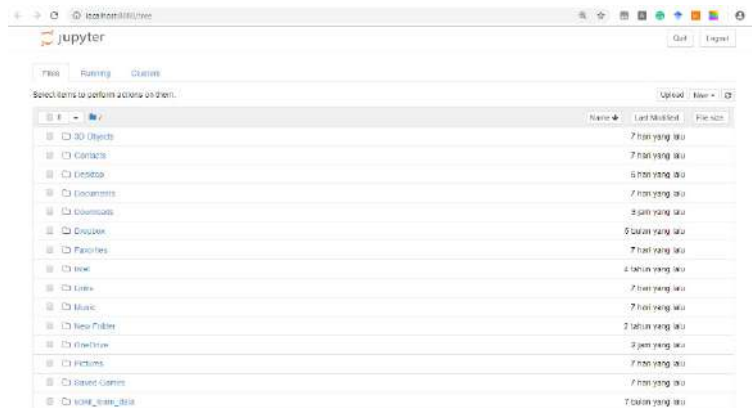
Gambar 5.21. Tampilan awal Anaconda

Untuk memulai Python menggunakan Jupyter Notebook, silahkan tekan Jupyter Notebook.



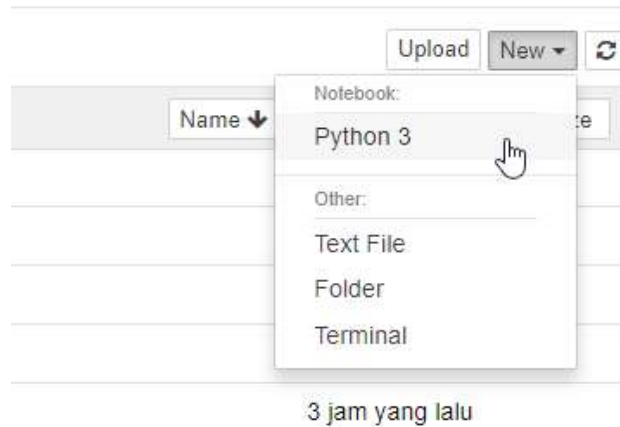
Gambar 5.22. Menjalankan Jupyter Notebook

Setelah dijalankan, maka akan menuju ke *default browser* yang digunakan.



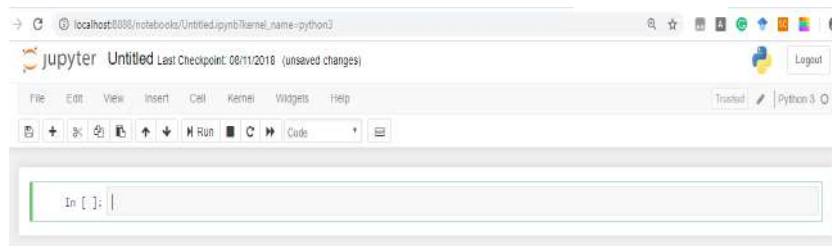
Gambar 5.23. Tampilan Jupyter Notebook di browser

Untuk mencoba menjalankan skrip Python, silahkan klik tombol *New* dan arahkan ke Python, seperti terlihat pada Gambar 5.24.



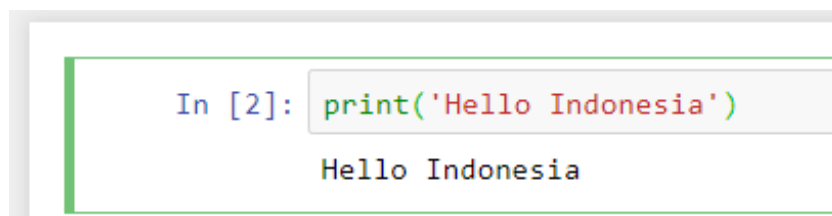
Gambar 5.24. Menjalankan Python dari Jupyter Notebook

Setelah siap, bisa langsung diketikkan skrip Python.



Gambar 5.25. Python Jupyter Notebook siap digunakan

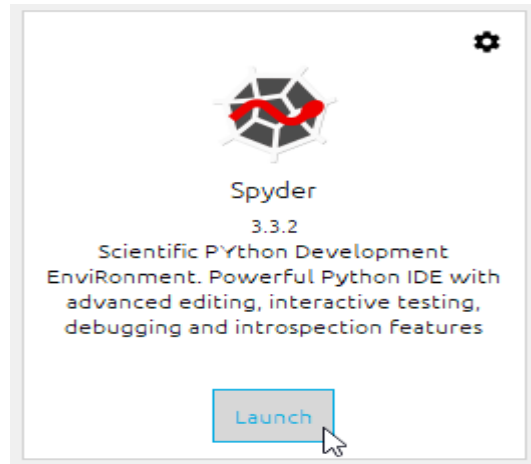
Silahkan tuliskan skrip Python, `print('Hello Indonesia')`.



Gambar 5.26. Contoh hasil keluaran skrip program Python

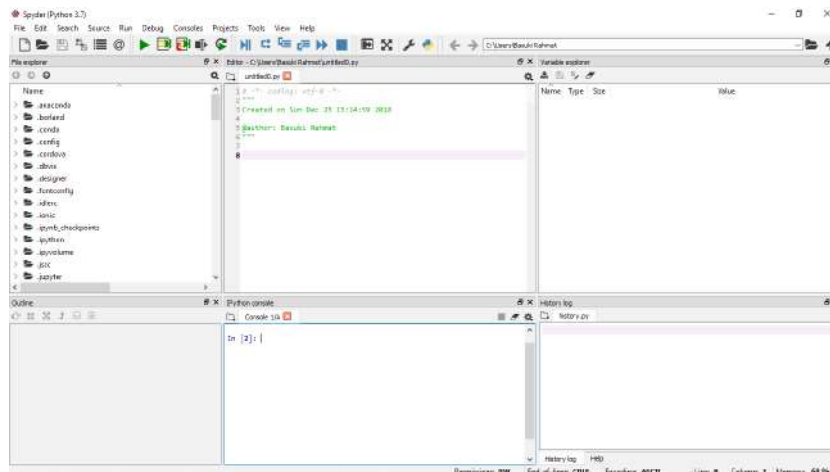
Python Menggunakan Spyder

Cara lain menjalankan Python melalui lingkungan Anaconda, dapat digunakan Spyder. Silahkan tekan Spyder.



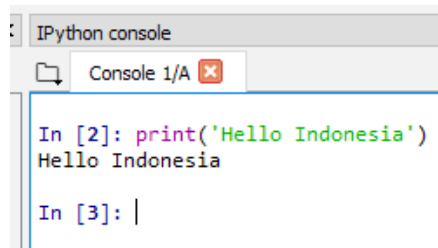
Gambar 5.27. Menjalankan Spyder melalui anaconda

Tunggu sampai keluar tampilan Spyder, seperti terlihat pada Gambar 5.28.



Gambar 5.28. Tampilan Spyder

Untuk mencoba, silahkan tuliskan skrip Python, `print('Hello Indonesia')`.



```
IPython console
Console 1/A x
In [2]: print('Hello Indonesia')
Hello Indonesia
In [3]: |
```

Gambar 5.29. Tampilan hasil menjalankan skrip Python

5.3.2 Pemrograman Kamera Kinect Xbox 360 dengan Python

Pemrograman kamera Kinect Xbox 360 dengan Python untuk mendapatkan data visual objek dari Robot Cerdas BNU 4.0, setidaknya membutuhkan *library-library* berikut:

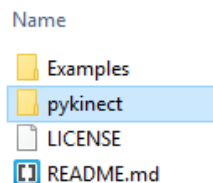
- Pykinect
- Numpy, dan
- OpenCV.

Pykinect adalah antarmuka sederhana untuk memungkinkan programmer mengembangkan aplikasi berbasis kamera Kinect yang diaktifkan menggunakan Python.

NumPy singkatan dari “*Numeric Python*” atau “*Numerical Python*”, adalah *library* untuk mendukung operasi array multi-dimensi dan matriks, memiliki banyak koleksi fungsi matematika tingkat tinggi yang beroperasi pada array ini.

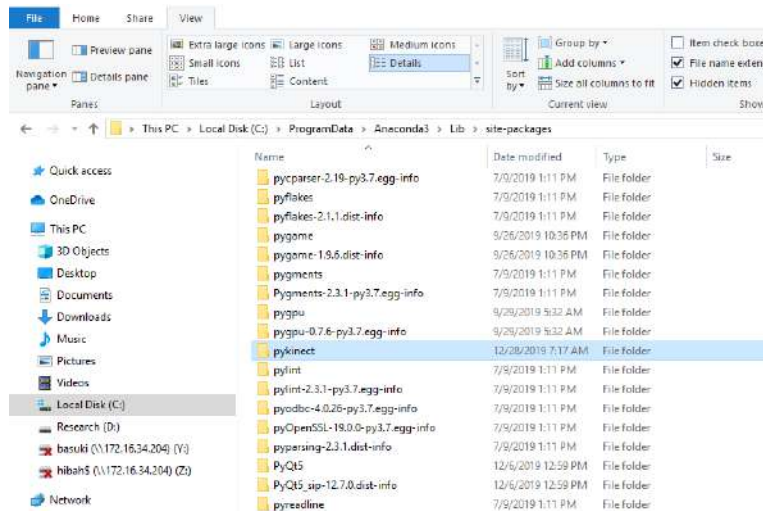
OpenCV (*Open Source Computer Vision Library*) adalah *software library open source* untuk visi komputer dan pembelajaran mesin. OpenCV dibangun untuk menyediakan infrastruktur umum aplikasi berbasis visi komputer dan untuk mempercepat penggunaan persepsi mesin pada produk komersial.

Untuk mendapatkan *library* pykinect, silahkan unduh dari alamat: <https://github.com/ShrirangaKadam/pykinect-python3.6>. Setelah diunduh, silahkan di-ekstrak. Diambil satu folder pykinect, untuk ditempatkan di *library* Anaconda.



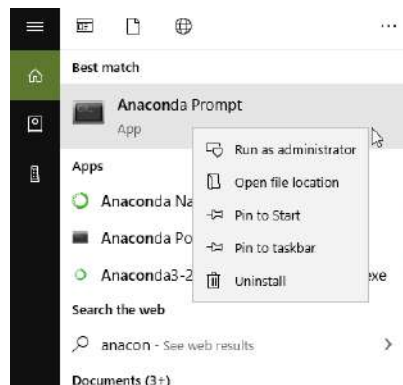
Gambar 5.30. Library pykinect

Satu folder pykinect di-copy untuk ditempatkan (di-paste) di library Anaconda. Biasanya berada di folder C:\ProgramData\ Anaconda3\Lib\site-packages. Lihat Gambar 5.31.



Gambar 5.31. Penempatan library pykinect di Anaconda

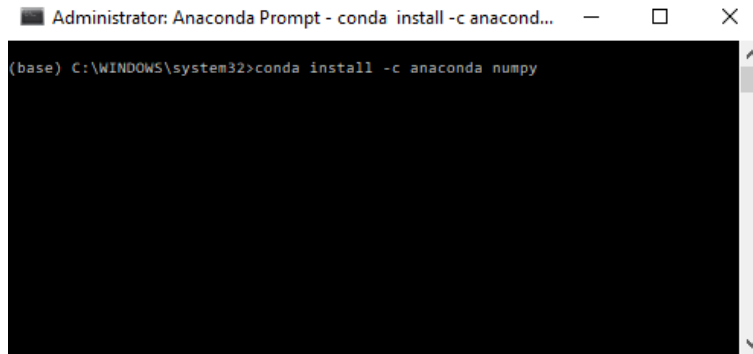
Selanjutnya, untuk instal *library* numpy, silahkan jalankan Anaconda Prompt sebagai Administrator.



Gambar 5.32. Menjalankan Anaconda Prompt sebagai Administrator

Kemudian, silahkan ketikkan:

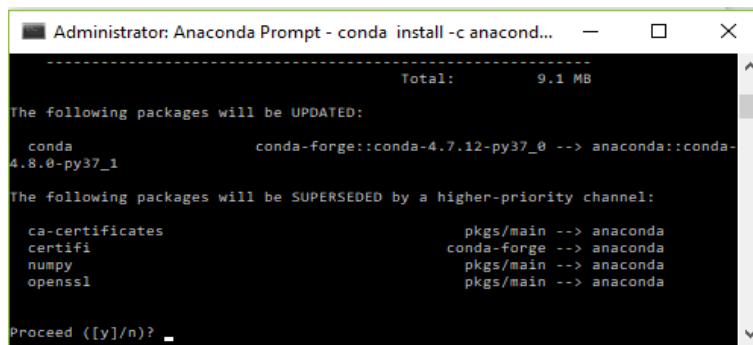
```
>> conda install -c anaconda numpy
```



```
Administrator: Anaconda Prompt - conda install -c anaconda numpy
(base) C:\WINDOWS\system32>conda install -c anaconda numpy
```

Gambar 5.33. Instal library numpy

Jika ada pertanyaan Proceed ([y]/n)? Silahkan jawab (tekan) y. Tunggu sampai proses instalasi selesai.



```
Administrator: Anaconda Prompt - conda install -c anaconda numpy
-----
Total: 9.1 MB

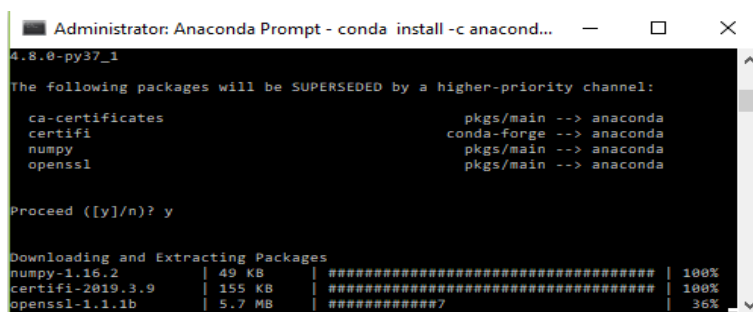
The following packages will be UPDATED:
conda                  conda-forge::conda-4.7.12-py37_0 --> anaconda::conda-4.8.0-py37_1

The following packages will be SUPERSEDED by a higher-priority channel:
ca-certificates       pkgs/main --> anaconda
certifi               conda-forge --> anaconda
numpy                 pkgs/main --> anaconda
openssl               pkgs/main --> anaconda

Proceed ([y]/n)?
```

Gambar 5.34. Pertanyaan instal library numpy

Jika instalasi *library* numpy sukses, maka akan tampil seperti pada Gambar 5.36.



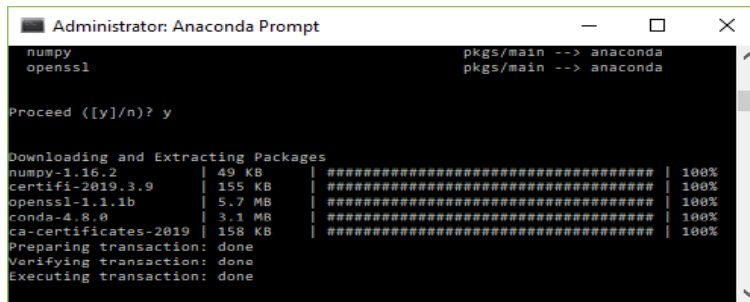
```
Administrator: Anaconda Prompt - conda install -c anaconda numpy
4.8.0-py37_1

The following packages will be SUPERSEDED by a higher-priority channel:
ca-certificates       pkgs/main --> anaconda
certifi               conda-forge --> anaconda
numpy                 pkgs/main --> anaconda
openssl               pkgs/main --> anaconda

Proceed ([y]/n)? y

Downloading and Extracting Packages
numpy-1.16.2          | 49 KB      | ##### 100%
certifi-2019.3.9      | 155 KB     | ##### 100%
openssl-1.1.1b        | 5.7 MB     | ##### 36%
```

Gambar 5.35. Proses instalasi library numpy



```
Administrator: Anaconda Prompt
numpy                                pkgs/main --> anaconda
openssl                              pkgs/main --> anaconda

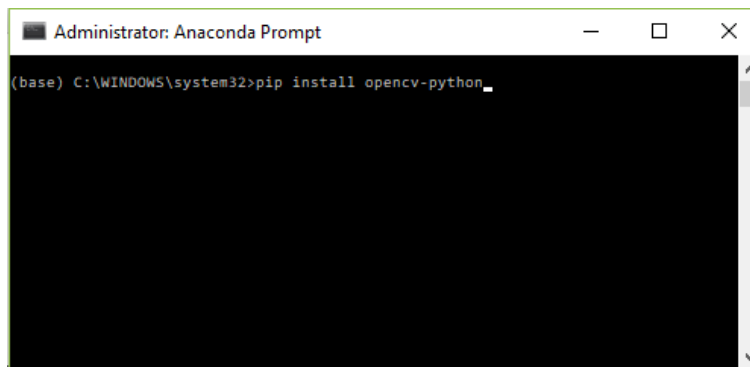
Proceed ([y]/n)? y

Downloading and Extracting Packages
numpy-1.16.2             | 49 KB | ##### | 100%
certifi-2019.3.9        | 155 KB | ##### | 100%
openssl-1.1.1b          | 5.7 MB | ##### | 100%
conda-4.8.0             | 3.1 MB | ##### | 100%
ca-certificates-2019    | 158 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Gambar 5.36. Instalasi library numpy sukses

Selanjutnya, untuk instal *library* OpenCV, silahkan ketikkan di Anaconda Prompt:

>> pip install opencv-python

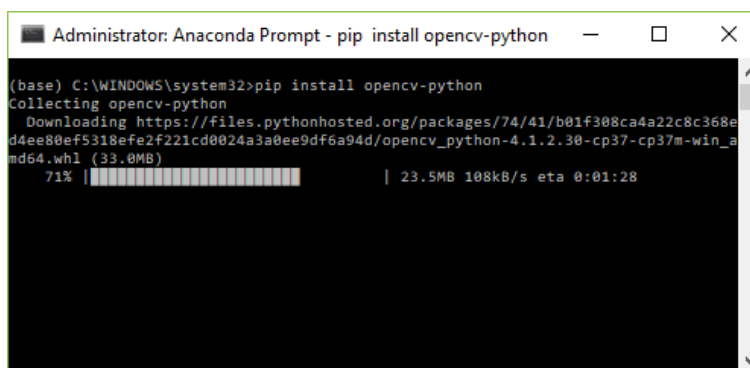


```
Administrator: Anaconda Prompt

(base) C:\WINDOWS\system32>pip install opencv-python_
```

Gambar 5.37. Instal library OpenCV

Silahkan tunggu, sampai proses instalasi *library* OpenCV selesai.



```
Administrator: Anaconda Prompt - pip install opencv-python

(base) C:\WINDOWS\system32>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/74/41/b01f308ca4a22c8c368e
d4ee80ef5318efe2f221cd0024a3a0ee9df6a94d/opencv_python-4.1.2.30-cp37-cp37m-win_a
md64.whl (33.0MB)
  71% |#####| 23.5MB 108kB/s eta 0:01:28
```

Gambar 5.38. Proses instalasi library OpenCV

Jika proses instalasi *library* OpenCV sukses, maka akan tampil seperti pada Gambar 5.39.

```

Administrator: Anaconda Prompt

(base) C:\WINDOWS\system32>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/74/41/b01f308ca4a22c8c368e
d4ee80ef5318efe2f221cd0024a3a0ee9df6a94d/opencv_python-4.1.2.30-cp37-cp37m-win_a
md64.whl (33.0MB)
    100% |#####| 33.0MB 154kB/s
Requirement already satisfied: numpy>=1.14.5 in c:\programdata\anaconda3\lib\sit
e-packages (from opencv-python) (1.16.2)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.1.2.30

(base) C:\WINDOWS\system32>

```

Gambar 5.39. Instal library OpenCV sukses

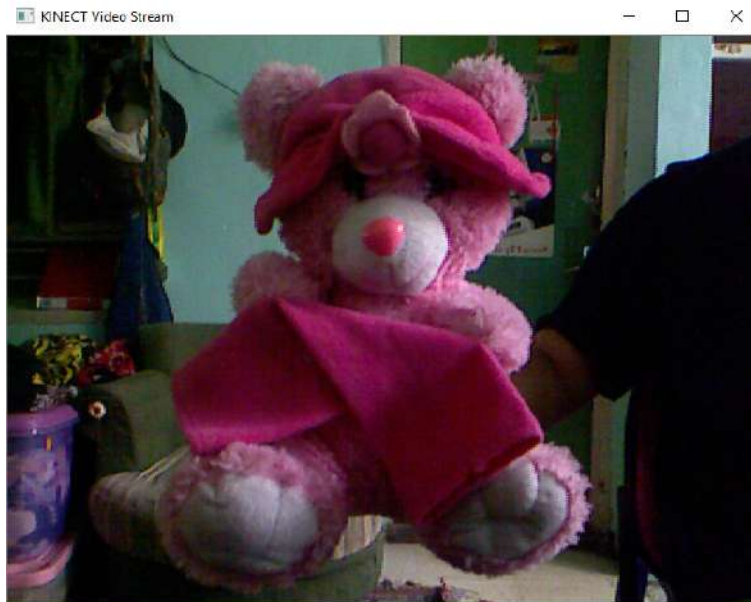
Setelah instalasi ketiga *library*: Pykinect, Numpy, dan OpenCV berhasil, selanjutnya tinggal dicoba buat program sederhana untuk mengambil data visual objek menggunakan kamera Kinect Xbox 360. Disini kamera Kinect Xbox 360 berfungsi seperti kamera *webcam*.

Berikut ini contoh program sederhana untuk menguji kamera Kinect Xbox 360. Silahkan ketikkan kode program berikut:

	Program Pengujian kamera Kinect Xbox 360
1	from pykinect import nui
2	import numpy
3	import cv2
4	
5	def video_handler_function(frame):
6	video = numpy.empty((480,640,4),numpy.uint8)
7	frame.image.copy_bits(video.ctypes.data)
8	cv2.imshow('KINECT Video Stream', video)
9	
10	kinect = nui.Runtime()
11	kinect.video_frame_ready += video_handler_function
12	kinect.video_stream.open(nui.ImageStreamType.Video, 2,nui.
13	ImageResolution.Resolution640x480,nui.ImageType.Color)
14	
15	cv2.namedWindow('KINECT Video Stream', cv2.WINDOW_AUTOSIZE)
16	
17	while True:
18	
19	key = cv2.waitKey(1)
20	if key == 27: break


```
21 kinect.close()
22 cv2.destroyAllWindows()
23
```

Jika dijalankan dan sukses tidak ada kesalahan, maka kamera Kinect Xbox 360 akan berfungsi seperti *webcam*. Berikut ini contoh hasil tangkapan kamera ini.



Gambar 5.40. Contoh hasil tangkapan kamera Kinect Xbox 360



BAB 6

Machine Learning dan Deep Learning

6.1. Machine Learning

Machine Learning merupakan bagian dari Kecerdasan Buatan atau *Artificial Intelligence* (AI). AI didefinisikan sebagai situasi di mana mesin dapat mensimulasikan pikiran manusia dalam pembelajaran dan analisis, yang dengan demikian dapat bekerja dalam pemecahan masalah (Rong *et al.*, 2020). Dengan kata lain situasi dimana mesin seolah bisa menirukan bagaimana manusia berpikir, belajar, menganalisis dan mengambil keputusan untuk memecahkan masalah. AI juga didefinisikan sebagai studi tentang “agen cerdas” yaitu, agen atau perangkat apa pun yang dapat merasakan dan memahami lingkungannya sehingga dapat mengambil tindakan yang tepat agar bisa memaksimalkan peluangnya dalam mencapai tujuannya (Rong *et al.*, 2020).

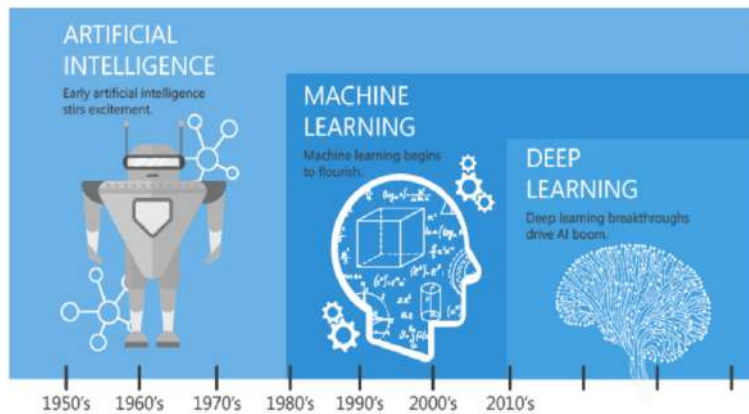
AI mulai dikutip pada tahun 1940-an dalam studi tentang apakah mesin dapat membuat sebuah keputusan (de Sousa *et al.*, 2019). Pada tahun 1970-an, pengembangan solusi AI terapan dimulai, dengan studi di berbagai bidang pengetahuan. Pada tahun 2010-an, studi dan aplikasi AI telah mencakup beberapa fungsi sektor publik, kesehatan masyarakat, transportasi, pendidikan, keamanan, komunikasi, dan bahkan angkatan bersenjata (de Sousa *et al.*, 2019).

Perkembangan lebih lanjut dari AI melahirkan *Machine Learning*. Perkembangan lebih lanjut dari *Machine Learning* melahirkan *Deep Learning*. Secara umum perkembangan

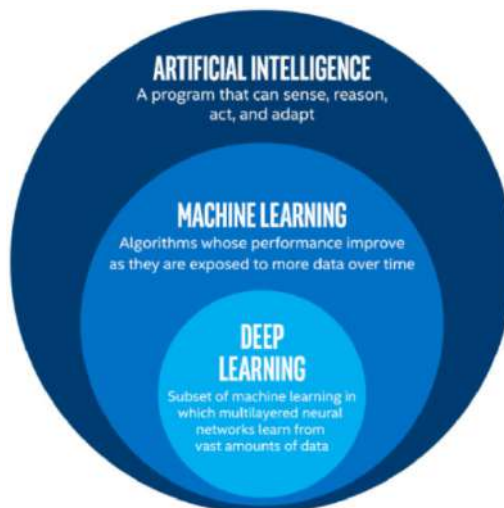


AI, *Machine Learning* dan *Deep Learning* dari tahun ke tahun diperlihatkan pada Gambar 6.1 (Oppermann, 2019). Dikaitkan dengan AI dan *Machine Learning*, *Deep Learning* merupakan bagian dari *Machine Learning* dan AI. Hubungan antara AI, *Machine Learning* dan *Deep Learning* diperlihatkan pada Gambar 6.2 (Oppermann, 2019).

Machine Learning sendiri merupakan teknik yang memungkinkan komputer “belajar” dari data-data yang disediakan tanpa pemrograman yang menyeluruh dan eksplisit dari setiap masalah (Meng *et al.*, 2020). Ini bertujuan memodelkan hubungan yang mendalam terhadap input data dan merekonstruksi skema pengetahuan. Hasil dari proses pembelajaran dapat digunakan untuk estimasi, prediksi, dan klasifikasi (Meng *et al.*, 2020).



Gambar 6.1. Perkembangan Artificial Intelligence, Machine Learning dan Deep Learning (Oppermann, 2019)



Gambar 6.2. Deep Learning bagian dari Machine Learning, keduanya bagian dari Artificial Intelligence (Oppermann, 2019)

6.2. Deep Learning dan Deep Network

Deep Learning telah menjadi tantangan untuk didefinisikan oleh banyak orang dan para peneliti karena telah berubah bentuk secara perlahan selama dekade terakhir. Satu definisi yang berguna menetapkan bahwa *Deep Learning* berkaitan dengan “Jaringan Saraf dengan lebih dari dua lapisan.” Namun jika dari definisi ini, maka seolah-olah *Deep Learning* sudah ada sejak tahun 1980-an (lihat Gambar 6.1). Padahal *Deep Learning* dianggap baru muncul pada tahun 2006, setelah Geoffrey Hinton memperkenalkan salah satu varian Jaringan Saraf Tiruan yang disebut *Deep Belief Nets* (Hinton, Osindero and Teh, 2006). Ide untuk melatih model Jaringan Saraf ini adalah dengan melatih dua lapisan kemudian ditambahkan satu lapisan di atasnya, kemudian dilatih hanya lapisan teratas dan begitu seterusnya. Dengan strategi ini didapat pelatihan model Jaringan Saraf yang memiliki lapisan lebih banyak dari model-model sebelumnya. Tulisan Geoffrey Hinton ini merupakan awal populernya istilah *Deep Learning* untuk membedakan arsitektur Jaringan Saraf Tiruan dengan banyak lapisan.

Setelah istilah *Deep Learning* populer, *Deep Learning* belum menjadi daya tarik yang besar bagi para peneliti karena Jaringan Saraf Tiruan dengan banyak lapisan memiliki kompleksitas algoritma yang besar, sehingga membutuhkan komputer dengan spesifikasi tinggi, dan tidak efisien secara komputasi saat itu. Hingga pada tahun 2009 Andrew Y. Ng dkk memperkenalkan penggunaan GPU untuk *Deep Learning* (Raina, Madhavan and Ng, 2009). Dengan penggunaan GPU Jaringan Saraf Tiruan dapat berjalan lebih cepat dibanding dengan menggunakan CPU. Dengan tersedianya *hardware* yang memadai perkembangan *Deep Learning* mulai pesat, dan menghasilkan produk-produk yang dapat kita nikmati saat ini seperti pengenalan wajah, *self-driving car*, pengenalan suara, dan lain lain.

Perkembangan Jaringan Saraf sudah jauh melampaui arsitektur gaya jaringan sebelumnya (diiringi dengan lebih banyak kekuatan pemrosesan) sebelum menunjukkan hasil spektakuler yang terlihat dalam beberapa tahun terakhir. Berikut ini adalah beberapa aspek dalam evolusi Jaringan Saraf ini, yang menjadi ciri khas dari *Deep Learning*:

- Lebih banyak neuron daripada jaringan sebelumnya.
- Cara yang lebih kompleks untuk menghubungkan lapisan/neuron di Jaringan Saraf.
- Ledakan dalam jumlah daya komputasi yang tersedia untuk pelatihan.
- Ekstraksi fitur otomatis.

Deep Learning mencoba memodelkan abstraksi data skala besar dengan menggunakan *Deep Neural Networks* (DNNs) multi lapisan, sehingga mereka dapat memahami gambar, suara, dan teks (Cao et al., 2018). *Deep Learning* umumnya memiliki dua sifat (Cao et al., 2018):

- (1) terdiri dari beberapa lapisan unit pemrosesan nonlinear, dan
- (2) presentasi fitur pada setiap lapisan menggunakan proses pembelajaran yang diawasi atau tidak diawasi.

Pembahasan tentang *Deep Learning*, tidak dapat dipisahkan dari *Deep Network* yang membahas tentang arsitektur jaringannya. Arsitektur utama *Deep Network* yang terkenal dan banyak digunakan di berbagai bidang penerapan, yaitu (Adam Gibson, 2017):

- a. *Unsupervised Pretrained Networks* (UPNs)
- b. *Convolutional Neural Networks* (CNNs)
- c. *Recurrent Neural Networks*, dan
- d. *Recursive Neural Networks*.

Masing-masing secara ringkas diuraikan sebagai berikut.

a. *Unsupervised Pretrained Networks* (UPNs)

Dalam kelompok ini, setidaknya terdapat tiga arsitektur spesifik, yaitu (Adam Gibson, 2017):

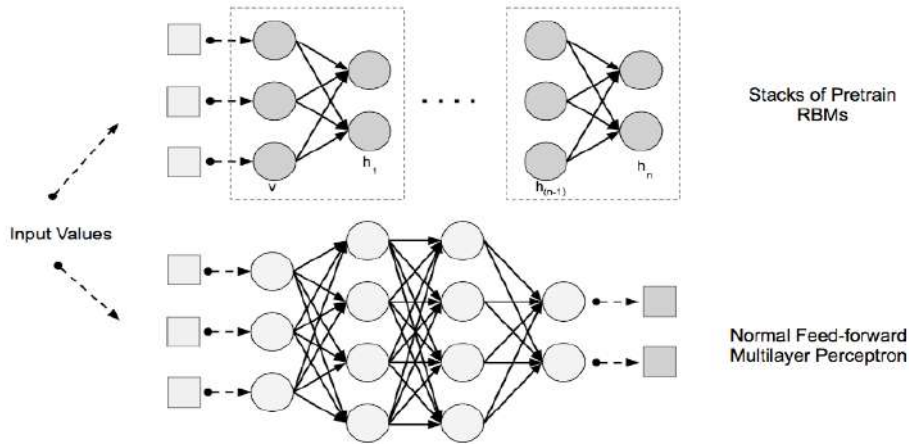
- *Autoencoders*
- *Deep Belief Networks* (DBNs), dan
- *Generative Adversarial Networks* (GANs).

Autoencoders

Autoencoder adalah varian dari Jaringan Saraf umpan-maju yang memiliki bias ekstra untuk menghitung kesalahan merekonstruksi input asli. Setelah pelatihan, *autoencoder* kemudian digunakan sebagai Jaringan Saraf umpan-maju normal untuk aktivasi. Ini adalah bentuk ekstraksi fitur yang tidak diawasi karena Jaringan Saraf hanya menggunakan input asli untuk pembelajaran bobot daripada *backpropagation*, yang memiliki label. *Deep Network* dapat menggunakan *Restricted Boltzmann Machines* (RBMs) atau *autoencoder* sebagai blok penyusun untuk jaringan yang lebih besar (namun jarang ada satu jaringan yang menggunakan keduanya).

Deep Belief Networks (DBNs)

DBN terdiri dari lapisan *Restricted Boltzmann Machines* (RBMs) untuk fase pra-terlatih dan kemudian jaringan umpan-maju untuk fase *fine-tune*. Gambar 6.3 menunjukkan arsitektur jaringan DBN (Adam Gibson, 2017).



Gambar 6.3. Arsitektur jaringan DBN (Adam Gibson, 2017)

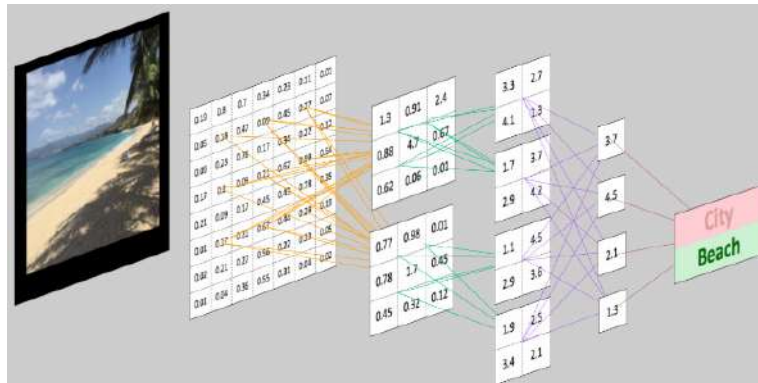
Generative Adversarial Networks (GANs)

GAN adalah contoh jaringan yang menggunakan pembelajaran tanpa pengawasan untuk melatih dua model secara paralel. Aspek kunci dari GAN (dan model generatif pada umumnya) adalah bagaimana mereka menggunakan jumlah parameter yang secara signifikan lebih kecil dari normal sehubungan dengan jumlah data yang dilatih dalam jaringan. Jaringan dipaksa untuk secara efisien mewakili data pelatihan, membuatnya lebih efektif dalam menghasilkan data yang mirip dengan data pelatihan.

b. Convolutional Neural Networks (CNNs)

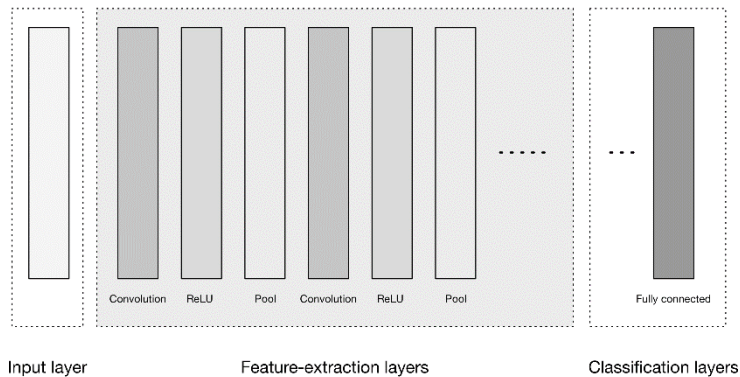
Tujuan dari CNN adalah mempelajari fitur tingkat tinggi dalam data melalui konvolusi. CNN sangat cocok untuk pengenalan objek melalui gambar dan memegang kompetisi klasifikasi gambar teratas secara konsisten. CNN dapat mengidentifikasi wajah, individu, rambu jalan, platipus, dan banyak aspek lain dari data visual. Untuk analisis teks, CNN tumpang tindih dengan *Optical Character Recognition* (OCR), karena CNN juga mampu menganalisis kata-kata sebagai unit tekstual diskrit. CNN juga pandai menganalisis suara.

Kemampuan CNN dalam pengenalan gambar adalah salah satu alasan utama mengapa dunia mengakui kehebatan *Deep Learning*. Seperti yang diilustrasikan pada Gambar 6.4, CNN sangat bagus dalam mendapatkan fitur-fitur dalam posisi tegak dan agak bagus dalam posisi invarian rotasi dari data gambar mentah (Adam Gibson, 2017).



Gambar 6.4. Contoh CNN dalam visi komputer (Adam Gibson, 2017)

CNN mengubah data input dari lapisan input melalui semua lapisan yang terhubung ke satu set skor kelas yang diberikan oleh lapisan output. Ada banyak variasi arsitektur CNN, tetapi umumnya mereka didasarkan pada pola lapisan, seperti yang ditunjukkan pada Gambar 6.5 (Adam Gibson, 2017).



Gambar 6.5. Arsitektur CNN (Adam Gibson, 2017)

Gambar 6.5 menggambarkan tiga kelompok utama:

- Lapisan input
- Lapisan ekstraksi fitur, dan
- Lapisan klasifikasi

Lapisan input menerima input tiga dimensi umumnya dalam bentuk spasial dari ukuran gambar (lebar \times tinggi) dan memiliki kedalaman yang mewakili saluran warna (umumnya tiga, untuk saluran warna *Red Green Blue* (RGB)).

Lapisan ekstraksi fitur memiliki pola pengulangan urutan yang umum:

- Lapisan konvolusi
- Lapisan (fungsi aktivasi) *Rectified Linear Unit* (ReLU), dan
- Lapisan pooling.

Lapisan-lapisan ini menemukan sejumlah fitur dalam gambar dan secara progresif membangun fitur tingkat tinggi.

Lapisan klasifikasi memiliki satu atau lebih lapisan yang terhubung sepenuhnya untuk mengambil fitur tingkat tinggi dan menghasilkan probabilitas atau skor kelas. Lapisan-lapisan ini sepenuhnya terhubung ke semua neuron di lapisan sebelumnya. Keluaran dari lapisan-lapisan ini biasanya menghasilkan keluaran dua dimensi dari dimensi $[b \times N]$, di mana b adalah jumlah contoh-contoh dalam *mini-batch* dan N adalah jumlah kelas dalam penyusunan skor.

c. *Recurrent Neural Networks*

Recurrent Neural Networks berada dalam keluarga Jaringan Saraf umpan-maju. Jaringan yang memungkinkan komputasi paralel dan berurutan. Jaringan ini serupa dengan otak manusia, yang memiliki jaringan umpan balik besar dari neuron yang terhubung. Otak adalah panutan yang luar biasa karena dapat memecahkan banyak masalah yang belum dapat diselesaikan oleh mesin saat ini. Jaringan ini mengambil setiap vektor dari urutan vektor input dan memodelkannya satu per satu. Ini memungkinkan jaringan untuk mempertahankan status, sementara ia memodelkan setiap vektor input melintasi jendela vektor input. Memodelkan dimensi waktu adalah ciri khas dari jaringan ini.

d. *Recursive Neural Networks*

Recursive Neural Networks, seperti halnya *Recurrent Neural Networks*, dapat menangani input variabel panjang. Bedanya *Recurrent Neural Networks* memiliki kemampuan untuk memodelkan struktur hierarki dalam dataset pelatihan. Lalu apa yang menarik dari jaringan ini? Secara umum, gambar memiliki pemandangan yang terdiri dari banyak objek. Adegan-adegan yang mendekonstruksi sering kali merupakan domain masalah yang menarik. Sifat rekursif dari dekonstruksi inilah yang menantang, karena bukan hanya permasalahan identifikasi objek dalam adegan saja, tetapi juga bagaimana agar objek-objek terkait dapat membentuk adegan.

Arsitektur jaringan ini terdiri dari matriks berbobot bersama dan struktur pohon biner yang memungkinkan jaringan rekursif mempelajari berbagai urutan kata atau bagian gambar. Ini berguna sebagai parser kalimat dan adegan. Jaringan ini menggunakan variasi *backpropagation* yang disebut *backpropagation through structure* (BPTS). Umpan-

umpan maju terjadi dari bawah ke atas, dan *backpropagation* adalah dari atas ke bawah. Bisa dianggap, bahwa tujuan adalah seperti bagian atas pohon, sedangkan inputnya adalah bagian bawahnya.

6.3. Pemrograman Deep Learning

6.3.1. Framework Deep Learning

Lihat kembali gambaran rancangan kerja sistem Robot Cerdas BNU 4.0 pada Gambar 2.4 di Bab 2. Dari gambaran tersebut dapat dilihat bahwa untuk keperluan pengenalan objek digunakan Bahasa Pemrograman Python. Dimana inti pengenalan objeknya menggunakan metode *Deep Learning*. Oleh karena itu, bagaimana mewujudkan pemrograman *Deep Learning* menggunakan Bahasa Pemrograman Python, inilah yang menjadi topik bahasan saat ini.

Pemrograman *Deep Learning* menggunakan Bahasa Pemrograman Python, sesuai dengan arsitektur-arsitektur *Deep Network* yang telah dibahas sebelumnya, saat ini tersedia banyak pilihan *framework*, antara lain:

- a. TensorFlow
- b. PyTorch
- c. Caffe
- d. Sonnet
- e. Keras
- f. MXNet
- g. Gluon
- h. Swift
- i. Chainer
- j. DL4J
- k. ONNX
- l. Darknet
- m. Darkflow
- n. YOLO

Masing-masing dijelaskan secara ringkas sebagai berikut. Sebagian besar diambil dari referensi (Kharkovyna, 2019).

a. TensorFlow

TensorFlow adalah *framework* yang dibuat dan dikembangkan oleh Google yang digunakan untuk merancang, membangun, dan melatih model *Deep Learning*. *Framework* ini bisa dibilang paling populer saat ini. Gmail, Uber, Airbnb, Nvidia, dan banyak merek terkenal lainnya menggunakannya.



Gambar 6.6. Logo TensorFlow

Python adalah bahasa klien yang paling nyaman untuk bekerja dengan TensorFlow. Namun, ada juga antarmuka eksperimental yang tersedia di JavaScript, C ++, Java dan Go, C # dan Julia. TensorFlow tidak hanya memperhitungkan kluster komputasi yang kuat tetapi juga kemampuan untuk menjalankan model pada platform seluler seperti iOS dan Android.

TensorFlow membutuhkan banyak pengkodean. Itu tidak akan menghasilkan AI yang kuat dalam semalam, tetapi bisa jadi alat bantu penelitian *Deep Learning* agar sedikit kurang rumit. Perlu berpikir hati-hati tentang arsitektur Jaringan Saraf, penilaian dengan benar terhadap dimensi dan volume data input dan outputnya, agar dihasilkan karya AI yang hebat.

TensorFlow beroperasi dengan grafik perhitungan statis. Yaitu, pertama-tama didefinisikan grafiknya, lalu dijalankan perhitungannya. Jika diperlukan, dibuat perubahan pada arsitekturnya, dan dilatih kembali modelnya. Pendekatan seperti itu dipilih demi efisiensi. Tentunya perhitungan perbaikan dalam proses pembelajaran tanpa harus kehilangan kecepatan belajarnya. Dalam hal ini, pesaing utama TensorFlow adalah PyTorch.

TensorFlow berguna untuk membuat dan bereksperimen dengan arsitektur *Deep Learning*. Formulasinya nyaman untuk integrasi data seperti memasukkan grafik, tabel SQL, dan gambar secara bersamaan. Hal itu didukung oleh Google yang menjamin akan tetap ada untuk sementara waktu. Oleh karena itu masuk akal untuk menginvestasikan waktu dan sumber daya untuk mempelajarinya.

b. PyTorch

Alat perangkat lunak utama untuk *Deep Learning* setelah Tensorflow adalah PyTorch. *Framework* PyTorch dikembangkan untuk layanan Facebook tetapi sudah digunakan untuk menyelesaikan tugas-tugasnya sendiri oleh perusahaan seperti Twitter dan Salesforce.



Gambar 6.7. Logo PyTorch

Tidak seperti TensorFlow, *library* PyTorch beroperasi dengan grafik yang diperbarui secara dinamis. Ini berarti dimungkinkan dibuat perubahan pada arsitektur ketika proses sedang berlangsung. Di PyTorch, bisa digunakan *debugger* standar, seperti pdb atau PyCharm.

PyTorch memiliki proses pelatihan Jaringan Saraf yang sederhana dan jelas. Pada saat yang sama, PyTorch mendukung paralelisme data dan model pembelajaran terdistribusi, dan juga berisi banyak model pra-terlatih. PyTorch jauh lebih cocok untuk proyek kecil dan pembuatan prototipe. Namun ketika dibutuhkan solusi lintas platform, maka TensorFlow sepertinya pilihan yang lebih cocok.

c. Caffe

Caffe (*Convolutional Architecture for Fast Feature Embedding*) adalah *framework Deep Learning* yang mendukung banyak antarmuka seperti C, C++, Python, MATLAB serta Antarmuka Baris Perintah.



Gambar 6.8. Logo Caffe

Caffe mendukung berbagai jenis arsitektur *Deep Learning* yang diarahkan pada segmentasi gambar dan klasifikasi gambar. Ini dikembangkan oleh *Berkeley AI Research* (BAIR) dan oleh kontributor komunitas.

Namun, Caffe tidak mendukung lapisan jaringan granularitas yang baik seperti pada TensorFlow. Mengingat arsitekturnya, dukungan keseluruhan terhadap jaringan berulang dan pemodelan bahasanya cukup buruk, dan pembangunan tipe lapisan kompleks harus dilakukan dalam bahasa tingkat rendah.

Caffe adalah jaringan *Deep Learning* yang populer untuk pengenalan gambar. Ini digunakan oleh para akademisi dan perusahaan pemula (*startup*) tetapi juga beberapa perusahaan besar seperti Yahoo. Caffe juga dapat melakukan komputasi pada *Central Processing Unit* (CPU) dan *Graphics Processing Unit* (GPU).

d. Sonnet

Sonnet adalah *framework Deep Learning* yang dibangun di atas TensorFlow. Ini dirancang untuk membuat Jaringan Saraf dengan arsitektur yang kompleks oleh perusahaan terkenal dunia DeepMind. *Library* Sonnet berorientasi objek tingkat tinggi yang menghasilkan abstraksi ketika mengembangkan Jaringan Saraf atau algoritma *Machine Learning* lainnya.



Gambar 6.9. Logo Sonnet

Ide Sonnet adalah untuk membangun objek Python primer yang sesuai dengan bagian tertentu dari Jaringan Saraf. Selanjutnya, objek-objek ini terhubung secara independen ke komputasi grafik TensorFlow. Pemisahan proses pembuatan objek dan menghubungkannya dengan grafik, dapat menyederhanakan desain arsitektur tingkat tinggi.

Keuntungan utama dari Sonnet, adalah dapat digunakan untuk mereproduksi penelitian-penelitian seperti yang telah ditunjukkan oleh makalah-makalah DeepMind. Hal ini tentunya DeepMind akan lebih memilih menggunakan Sonnet sendiri, daripada menggunakan framework-framework Deep Learning lainnya.

e. Keras

Keras adalah *framework Machine Learning* yang mungkin bisa menjadi teman baru ketika bekerja dengan banyak data, terutama jika mengikuti *state-of-the-art* dari AI: *Deep Learning*. Keras dapat digunakan sebagai API tingkat tinggi di atas *library* tingkat bawah populer lainnya seperti Theano dan CNTK selain Tensorflow. Penciptaan model *Deep Learning* yang sangat besar di Keras direduksi menjadi fungsi garis tunggal. Tetapi strategi ini membuat Keras menjadi lingkungan yang kurang dapat dikonfigurasi dibandingkan dengan framework tingkat rendah.



Gambar 6.10. Logo Keras

Keras adalah *framework Deep Learning* yang terbaik bagi mereka yang baru memulai. Ini ideal untuk belajar dan membuat prototipe konsep-konsep sederhana, untuk memahami inti dari berbagai model dan proses pembelajaran mereka.

Keras adalah API yang ditulis dengan indah. Sifat fungsional API membantu sepenuhnya dan sebagai jalan keluar menghasilkan aplikasi yang lebih eksotis. Keras tidak memblokir akses ke *framework* tingkat bawah. Keras menghasilkan kode yang jauh lebih mudah dibaca dan ringkas. Keras dengan model API serialisasi/deserialisasi,

panggilan balik, dan streaming data menggunakan generator Python, sangatlah matang. Namun, Keras tidak dapat dibandingkan dengan Tensorflow karena mereka berada pada level abstraksi yang berbeda.

Tensorflow ada di Level Bawah. Di sinilah *framework* seperti MXNet, Theano, dan PyTorch. Ini adalah tingkat di mana operasi matematika seperti perkalian matriks-matriks umum, dan Jaringan Saraf primitif seperti operasi konvolusi diimplementasikan.

Keras ada di level yang lebih tinggi. Pada level ini, level primitif bawah digunakan untuk mengimplementasikan abstraksi Jaringan Saraf seperti lapisan-lapisan dan model. Secara umum, pada level ini, API memiliki manfaat lainnya seperti penghematan model dan pelatihan model.

f. MXNet

MXNet adalah *framework Deep Learning* yang sangat skalabel yang dapat digunakan pada berbagai perangkat. Meskipun tampaknya tidak banyak digunakan dibandingkan dengan TensorFlow. Pertumbuhan MXNet kemungkinan banyak didorong oleh perkembangan proyek-proyek Apache.



Gambar 6.11. Logo MXNet

Framework ini awalnya mendukung sejumlah besar bahasa (C ++, Python, R, Julia, JavaScript, Scala, Go, dan bahkan Perl). Penekanan utamanya pada kenyataan bahwa *framework* ini paralel sangat efektif pada banyak GPU dan banyak mesin. Ini, khususnya, telah ditunjukkan oleh karyanya di Amazon *Web Services*.

MXNet mendukung banyak GPU (dengan perhitungan yang dioptimalkan dan pengalihan konteks cepat). Kode bersih dan mudah dirawat (Python, R, Scala, dan API lainnya). Kemampuan pemecahan masalah yang cepat (penting, bagi pemula yang belajar Deep Learning).

Meskipun tidak sepopuler TensorFlow, MXNet memiliki dokumentasi terperinci dan mudah digunakan. Dengan kemampuan untuk memilih antara gaya pemrograman imperatif dan simbolis, menjadikannya kandidat *framework* yang baik bagi para pemula dan programmer yang berpengalaman.

g. Gluon

Gluon adalah *framework Deep Learning* yang luar biasa yang dapat digunakan untuk membuat model sederhana maupun canggih. Kekhususan proyek Gluon adalah

antarmuka yang fleksibel yang menyederhanakan prototipe, membangun dan melatih model Deep Learning tanpa mengorbankan kecepatan belajar.



Gambar 6.12. Logo Gluon

Gluon didasarkan pada MXNet dan menawarkan API sederhana yang menyederhanakan pembuatan model Deep Learning. Mirip dengan PyTorch, *framework* Gluon mendukung pekerjaan dengan grafik dinamis. Penggabungan Gluon dengan MXNet akan menghasilkan kinerja yang tinggi. Dari perspektif ini, Gluon terlihat seperti alternatif yang sangat menarik dari Keras untuk komputasi terdistribusi.

Gluon dapat mendefinisikan Jaringan Saraf menggunakan kode sederhana, jelas, dan ringkas. Ini menyatukan algoritma pelatihan dan model Jaringan Saraf, sehingga memberikan fleksibilitas dalam proses pengembangan tanpa mengorbankan kinerja. Gluon memungkinkan untuk mendefinisikan model Jaringan Saraf yang dinamis, artinya mereka dapat dibangun dengan cepat, dengan struktur apa pun, dan menggunakan aliran kontrol asli Python.

h. Swift

Bagi sebagian programmer, ketika mendengar Swift, mungkin yang dipikirkan adalah pengembangan aplikasi berbasis iOS atau MacOS. Bagi yang tertarik dengan *Deep Learning*, maka seharusnya tidak asing dengan istilah Swift for Tensorflow (disingkat S4TF). Dengan integrasi langsung dengan bahasa pemrograman umum, S4TF memungkinkan menjadi algoritma yang lebih kuat untuk diekspresikan dalam penyelesaian masalah. Dengan menggunakan TensorFlow, API Swift memberi akses transparan ke semua operator TensorFlow tingkat rendah.



Gambar 6.13. Logo Swift

i. Chainer

Sampai munculnya DyNet di CMU, dan PyTorch di Facebook, Chainer adalah *framework* Jaringan Saraf terkemuka untuk grafik komputasi dinamis atau jaringan yang memungkinkan input dengan panjang berbeda-beda. Fitur yang populer untuk tugas-tugas *Neuro-linguistic programming* (NLP).



Gambar 6.14. Logo Chainer

Kode ini ditulis dalam Python murni di atas *library* Numpy dan CuPy. Chainer adalah *framework* pertama yang menggunakan model arsitektur dinamis (seperti pada PyTorch).

Chainer beberapa kali memegang rekor mengalahkan *framework* yang lain (TensorFlow, MxNet dan CNTK) dalam hal efektivitas penskalaan ketika masalah pemodelan diselesaikan oleh Jaringan Saraf. Dengan tolok ukur kecepatan proses, Chainer lebih cepat daripada *framework* yang berorientasi Python lainnya. Terutama dengan kinerja pusat data berbasis GPU, Chainer memiliki kecepatan proses yang lebih baik daripada TensorFlow. TensorFlow sendiri dioptimalkan untuk arsitektur TPU.

j. DL4J

Bagi penggemar pemrograman Java, seharusnya atidak asing dengan DL4J (kependekan dari *Deep Learning for Java*). Pelatihan Jaringan Saraf dalam DL4J dilakukan secara paralel melalui iterasi menggunakan kluster. Proses ini didukung oleh arsitektur Hadoop dan Spark.



Gambar 6.15. Logo DL4J

k. ONNX

Proyek ONNX lahir dari kolaborasi Microsoft dan Facebook dalam hal pencarian format terbuka untuk presentasi model *Deep Learning*. ONNX menyederhanakan proses transfer model antar algoritma yang bekerja dengan kecerdasan buatan. ONNX memungkinkan model untuk dilatih dalam satu *framework* dan ditransfer ke *framework* yang lain untuk menghasilkan kesimpulan. Model ONNX saat ini didukung oleh Caffe2, Microsoft *Cognitive Toolkit*, MXNet, PyTorch, konektor ke banyak *framework* dan library umum lainnya.



Gambar 6.16. Logo ONNX

l. Darknet

Darknet adalah kerangka kerja Jaringan Saraf *open source* yang ditulis dalam bahasa pemrograman C dan CUDA. Cepat, mudah dipasang, dan mendukung komputasi CPU dan GPU.



Gambar 6.17. Logo Darknet

m. Darkflow

Darkflow adalah *library Deep Learning* yang menerjemahkan *library* Darknet ke TensorFlow. Darkflow dapat digunakan dalam aplikasi lain yang mendukung bahasa pemrograman Python sebagai alternatif jika suatu perangkat tidak mendukung CUDA dan cuDNN. Melalui konfigurasi yang sederhana, adanya file *cfg* merupakan kode untuk model sedangkan *weights* merupakan bobot hasil pelatihan yang dapat digunakan untuk melakukan *transfer learning*.

n. YOLO

You only look once (YOLO) adalah sistem deteksi objek *real-time* yang canggih. Pada Pascal Titan X dapat memproses gambar pada 30 FPS dan memiliki pemetaan sebesar 57,9% pada COCO *test-dev*. YOLOv3 menggunakan beberapa trik untuk meningkatkan pelatihan dan meningkatkan kinerja, termasuk: prediksi multi-skala, pengklasifikasi *backbone* yang lebih baik, dan banyak lagi.



Gambar 6.18. Logo YOLO

Jika mayoritas sistem deteksi menggunakan pengklasifikasian atau *localizer* untuk melakukan deteksi dengan menerapkan model ke gambar di beberapa lokasi dan skala dan memberi nilai pada gambar sebagai bahan untuk pendeteksian. YOLO menggunakan pendekatan yang berbeda, yakni menerapkan Jaringan Saraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas. Untuk setiap kotak wilayah pembatas ditimbang probabilitasnya untuk diklasifikasi sebagai objek atau bukan. YOLO memiliki arsitektur CNN.

6.3.2. Pemrograman Deep Neural Network (DNN) dengan OpenCV

OpenCV menyediakan modul *Deep Neural Network* (DNN) untuk pemrograman *Deep Learning*. Sejak OpenCV 3.1 ada modul DNN di *library*-nya, yang mengimplementasikan *forward pass* (tahap maju) menggunakan *Deep Network*, dengan pra-terlatih menggunakan beberapa *framework Deep Learning* yang populer, seperti Caffe, dan lain-lain. Cara instalasi OpenCV sudah dijelaskan di Bab 5, saat dijelaskan pemrograman kamera Kinect Xbox 360 dengan Python.

Dalam OpenCV 3.3 modul ini telah dipromosikan dari `opencv_contrib` ke repositori utama (<https://github.com/opencv/opencv/tree/master/modules/dnn>). Modul ini tidak memiliki dependensi tambahan, kecuali `libprotobuf`. `Libprotobuf` sekarang sudah dimasukkan ke dalam OpenCV.

Framework yang didukung:

- Caffe
- TensorFlow
- Torch
- Darknet, dan
- Model-model dalam format ONNX.

Lapisan yang didukung:

- AbsVal
- AveragePooling
- BatchNormalization
- Concatenation
- Convolution (termasuk *dilated convolution*)
- Crop
- Deconvolution, juga dikenal sebagai *transposed convolution* atau *full convolution*
- DetectionOutput (SSD-specific layer)
- Dropout
- Eltwise (+, *, max)
- Flatten
- FullyConnected
- LRN
- LSTM
- MaxPooling
- MaxUnpooling
- MVN
- NormalizeBBBox (SSD-specific layer)
- Padding
- Permute
- Power
- PReLU (termasuk *ChannelPReLU with channel-specific slopes*)
- PriorBox (SSD-specific layer)
- ReLU
- RNN
- Scale
- Shift
- Sigmoid

- Slice
- Softmax
- Split, dan
- TanH.

Beberapa fungsi dari modul dnn untuk keperluan pemrograman *Deep Learning* antara lain dijabarkan berikut ini.

Untuk memuat gambar dari disk menggunakan fungsi-fungsi berikut:

- cv2.dnn.blobFromImage
- cv2.dnn.blobFromImages

Untuk dapat langsung mengimpor model dari berbagai *framework* melalui metode “create”:

- cv2.dnn.createCaffeImporter
- cv2.dnn.createTensorFlowImporter
- cv2.dnn.createTorchImporter

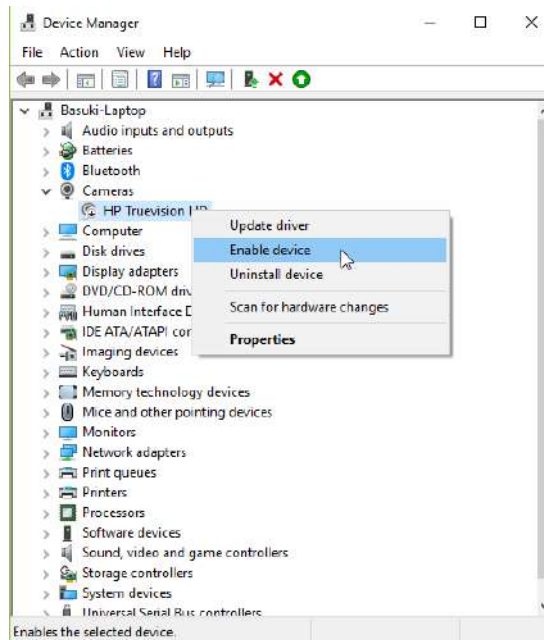
Jika hanya ingin baca model serial dari disk secara langsung menggunakan metode “read”:

- cv2.dnn.readNetFromCaffe
- cv2.dnn.readNetFromTensorFlow
- cv2.dnn.readNetFromTorch
- cv2.dnn.readhTorchBlob

Setelah diambil model dari disk, metode .forward digunakan untuk meneruskan-propagasi gambar hingga diperoleh klasifikasi aktual.

6.3.3. Pengenalan Objek Waktu Nyata dengan DNN

Selanjutnya akan dicoba pemrograman *Deep Learning* untuk pengenalan objek secara waktu nyata menggunakan modul DNN dari OpenCV. Untuk keperluan ini, sementara di-*enable* terlebih dahulu webcam dari komputer atau laptop yang digunakan. Melalui *Device Manager*. Kemudian kamera Kinect Xbox 360 juga dilepas dulu untuk sementara waktu dari port USB.



Gambar 6.19. Kamera webcam di-enable

Berikut ini langkah demi langkah, pemrograman *Deep Learning* untuk pengenalan objek secara waktu nyata menggunakan modul DNN (dnn) dari OpenCV.

1. Buat program baru, simpan dengan nama `deteksi_objek.py` dan masukkan kode berikut:

	deteksi_objek.py
1	# import the necessary packages
2	from imutils.video import VideoStream
3	from imutils.video import FPS
4	import numpy as np
5	import argparse
6	import imutils
7	import time
8	import cv2

Dari kode di atas, dibutuhkan *library* *imutils*. *Library* yang menyediakan serangkaian fungsi kenyamanan untuk membuat fungsi pemrosesan gambar dasar seperti translasi, rotasi, mengubah ukuran, skeletonisasi, menyortir kontur, mendeteksi tepi, dan menampilkan gambar Matplotlib.

Cara instal *library* *imutils*, seperti biasa ketikkan di Anaconda prompt dengan level Administrator:

```
>> pip install imutils
```

Selanjutnya, ketikkan lanjutan kode programnya.

deteksi_objek.py	
9	# construct the argument parse and parse the
10	# arguments
11	ap = argparse.ArgumentParser()
12	ap.add_argument("-p", "--prototxt", required=True,
13	help="path to Caffe 'deploy' prototxt file")
14	ap.add_argument("-m", "--model", required=True,
15	help="path to Caffe pre-trained model")
16	ap.add_argument("-c", "--confidence", type=float, default=0.2,
17	help="minimum probability to filter weak detections")
18	args = vars(ap.parse_args())

Keterangan:

- prototxt : *path* ke file prototxt Caffe.
- model : *path* ke model pra-terlatih.
- confidence : Batas probabilitas minimum untuk memfilter deteksi lemah. Standarnya adalah 20%.

Selanjutnya, dilakukan inisialisasi daftar kelas dan set warna:

deteksi_objek.py	
19	# initialize the list of class labels MobileNet SSD was
20	# trained to detect, then generate a set of bounding
21	# box colors for each class
22	CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
	"bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
	"dog", "horse", "motorbike", "person", "pottedplant", "sheep",
	"sofa", "train", "tvmonitor"]
23	COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

Selanjutnya, dimuat model yang digunakan dan diatur aliran videonya.

deteksi_objek.py	
24	# load our serialized model from disk
25	print("[INFO] loading model...")
26	net = cv2.dnn.readNetFromCaffe(args["prototxt"],
27	args["model"])
28	
29	# initialize the video stream, allow the camera
30	# sensor to warmup, and initialize the FPS counter
31	print("[INFO] starting video stream...")
32	vs = VideoStream(src=0).start()
33	time.sleep(2.0)
34	fps = FPS().start()

Selanjutnya, bikin program *lup* untuk membaca *frame-frame* dari aliran video (webcam).

deteksi_objek.py	
35	# loop over the frames from the video stream
36	while True:
37	# grab the frame from the threaded video stream
38	# and resize it to have a maximum width of 400
39	# pixels
40	frame = vs.read()
41	frame = imutils.resize(frame, width=400)
42	
43	# grab the frame dimensions and convert it to a
44	# blob
45	(h, w) = frame.shape[:2]
46	blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
	0.007843, (300, 300), 127.5)
47	
48	# pass the blob through the network and obtain
49	# the detections and predictions
50	net.setInput(blob)
51	detections = net.forward()

Pada posisi ini, telah dideteksi objek dalam bingkai input. Sekarang saatnya untuk dilihat nilai-nilai keyakinan dan ditentukan apakah harus digambar kotak+label yang mengelilingi objek.

	deteksi_objek.py
52	# loop over the detections
53	for i in np.arange(0, detections.shape[2]):
54	# extract the confidence (i.e., probability)
55	# associated with the prediction
56	confidence = detections[0, 0, i, 2]
57	
58	# filter out weak detections by ensuring the
59	# `confidence` is greater than the minimum
60	# confidence
61	if confidence > args["confidence"]:
62	# extract the index of the class
63	# label from the `detections`, then
64	# compute the (x, y)-coordinates of
65	# the bounding box for the object
66	idx = int(detections[0, 0, i, 1])
67	box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
68	(startX, startY, endX, endY) = box.astype("int")
69	
70	# draw the prediction on the frame
71	label = "{: {:.2f}%".format(CLASSES[idx], confidence * 100)
72	cv2.rectangle(frame, (startX, startY), (endX, endY), COLORS[idx], 2)
73	y = startY - 15 if startY - 15 > 15 else startY + 15
74	cv2.putText(frame, label, (startX,y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

Selanjutnya, lup tangkapan *frame* melibatkan (1) tampilkan *frame*, (2) periksa penekanan tombol keluar, dan (3) perbarui *frame* perdetik.

	deteksi_objek.py
75	# show the output frame
76	cv2.imshow("Frame", frame)
77	key = cv2.waitKey(1) & 0xFF
78	
79	# if the `q` key was pressed, break from the
80	# loop
81	if key == ord("q"):
82	break
83	
84	# update the FPS counter
85	fps.update()

Akhirnya hentikan timer dan tampilkan informasi *frame* perdetik.

	deteksi_objek.py
86	# stop the timer and display FPS information
87	fps.stop()
88	print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
89	print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
90	
91	# do a bit of cleanup
92	cv2.destroyAllWindows()
93	vs.stop()
94	

2. Langkah berikutnya, silahkan unduh file MobileNetSSD_deploy.prototxt.txt, dan MobileNetSSD_deploy.caffemodel, kemudian taruh di folder yang sama dengan program deteksi_objek.py.

Alamatnya:

https://github.com/djmv/MobilNet_SSD_opencv

Keterangan:

Jika diinginkan menggunakan model hasil pelatihan dataset sendiri (*own dataset*), antara lain dapat dipelajari dari alamat:

<https://github.com/chuanqi305/MobileNet-SSD>

3. Akhirnya, tinggal dijalankan program deteksi_objek.py. Kalo dijalankan melalui Jupyter Notebook, perintahnya sebagai berikut:

```
>> run -i deteksi_objek.py --prototxt MobileNetSSD_deploy.prototxt.txt  
--model MobileNetSSD_deploy.caffemodel
```

Contoh hasil deteksi objek setelah dijalankan, seperti terlihat pada Gambar 6.20.



Gambar 6.20. Contoh hasil deteksi objek




BAB 7

Pemrograman Robot Cerdas

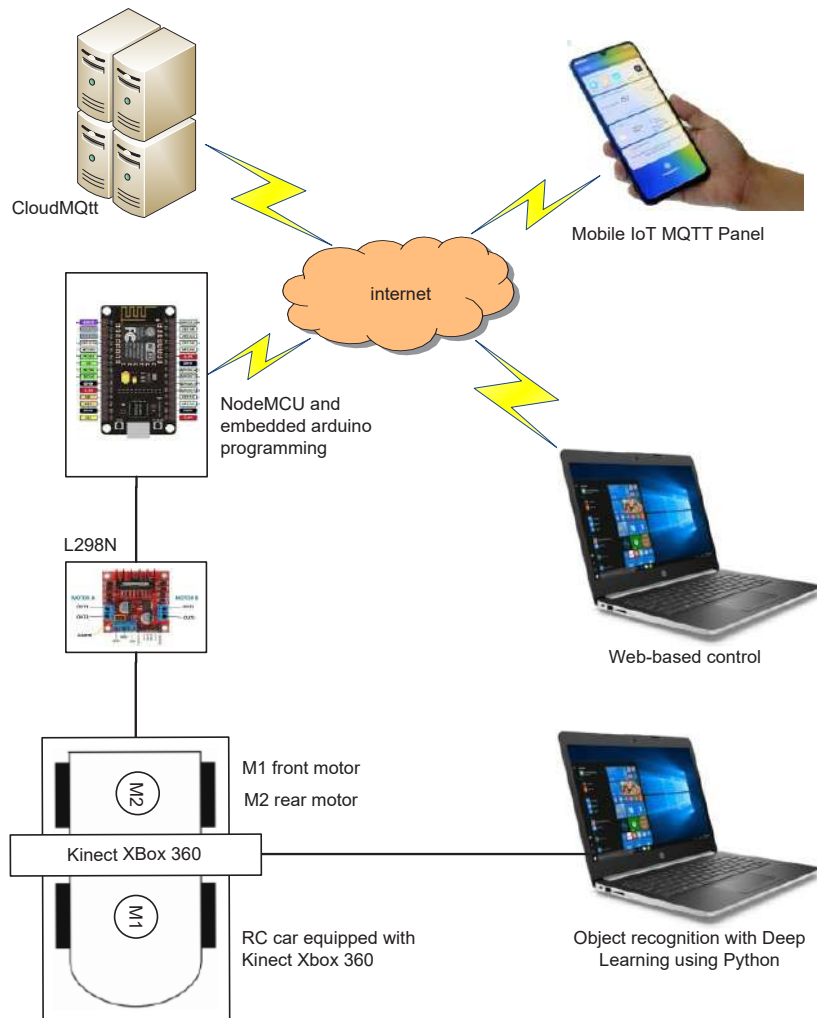
7.1. Kendali Robot via Web dan Mobile

Perhatikan kembali gambaran rancangan kerja sistem Robot Cerdas BNU 4.0 pada Gambar 2.4 di Bab 2. Di Bab ini, akan dijelaskan inti pemrograman Arduino untuk menggerakkan robot. Pengendaliannya via web dan *mobile*. Serta pemrograman Python untuk pengenalan objek menggunakan kamera Kinect Xbox 360, dengan metode *Deep Learning*. Guna mempermudah penjelasan, Gambar 2.4 ditampilkan kembali dengan sedikit tampilan yang berbeda, seperti terlihat pada Gambar 7.1.

Seperti yang sudah dijelaskan di Bab 2, badan Robot Cerdas BNU 4.0 ini berupa mobil RC yang telah dimodifikasi. Mobil RC ini memiliki dua motor, yaitu motor depan (M1) dan motor belakang (M2). Motor depan digunakan untuk belok kanan dan belok kiri, sedangkan motor belakang digunakan untuk gerak maju dan mundur. Untuk menjalankan kedua motor ini, digunakan driver motor L298N. Driver motor terhubung ke mikrokontroler NodeMCU V3 yang berbasis *chip* ESP8266. Dengan *chip* ini, NodeMCU V3 memiliki kemampuan koneksi ke *cloud Internet of Things* (IoT). *Cloud* IoT yang digunakan di buku ini juga sudah dijelaskan sebelumnya di Bab 3, yaitu *cloud* MQTT (<https://www.cloudmqtt.com>). Dengan demikian, program Arduino yang dibuat



akan digunakan untuk mengendalikan robot via web dan *mobile* melalui koneksi internet. Untuk pengendalian secara *mobile*, seperti yang telah dijelaskan sebelumnya di Bab 3, menggunakan *Android Apps* (IoT MQTT Panel).



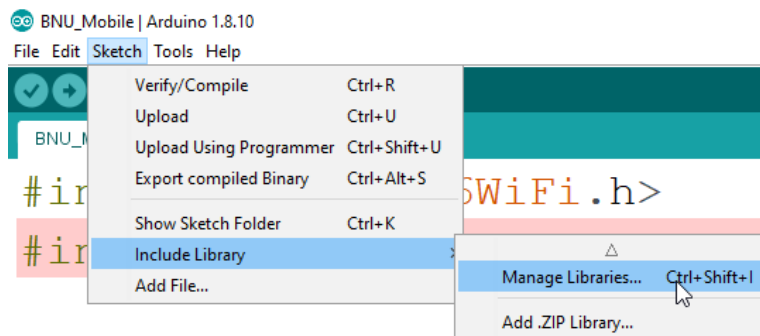
Gambar 7.1. Rancangan sistem Robot BNU 4.0

Program Arduino untuk pengendalian robot, disesuaikan dengan rangkaian elektronik robot, seperti yang telah dijelaskan di Bab 2. Dimana IN 1 dan IN 2 dari

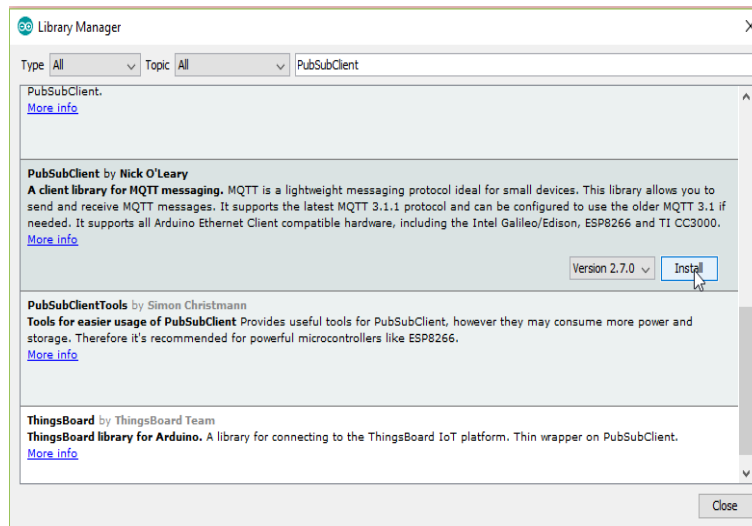
driver motor L298N, dihubungkan ke pin D3 dan D4 dari mikrokontroler NodeMCU V3. Hal ini digunakan untuk mengendalikan motor belakang (M2), untuk gerak maju dan mundur. Sedangkan IN 3 dan IN 4 dari driver motor L298N, dihubungkan ke pin D7 dan D8 dari mikrokontroler NodeMCU V3. Hal ini digunakan untuk mengendalikan motor depan (M1), untuk belok kanan dan belok kiri.

Program Arduino ini, juga disesuaikan dengan pengendalian via IoT secara *mobile*, seperti yang telah dijelaskan di Bab 3. Penggunaan Server, Port, dan lain-lain disesuaikan dengan informasi dari server Cloud MQTT seperti yang telah dijelaskan di Bab 3. Lihat Gambar 3.13. Demikian juga mengenai panel-panel yang digunakan pada IoT MQTT Panel, yang terdiri dari panel MAJU, KIRI, KANAN, dan MUNDUR. Diisi dengan topik Maju, Kiri, Kanan, dan Mundur. Masing-masing dengan *payload* 1, 2, 3, dan 4. Seperti yang telah dijelaskan di Bab 3. Topik adalah nama tentang pesan, dimana klien dalam hal ini mikrokontroler NodeMCU V3 mempublikasikan, berlangganan, atau melakukan keduanya untuk suatu topik. Di Robot Cerdas BNU 4.0 ini, pengaturannya ada di program Arduino. Sehingga saat panel yang ada di ponsel melalui aplikasi IoT MQTT Panel, topik atau panel yang bersesuaian ditekan, maka klien yang sudah berlangganan topik tersebut dapat merespon. Dalam hal ini klien (Robot BNU 4.0 dengan mikrokontroler NodeMCU V3) akan bergerak maju, belok kiri, belok kanan atau mundur.

Kemudian, untuk kebutuhan *library*, selain esp8266 seperti yang telah dijelaskan di Bab 4, perlu diinstal satu *library* lagi, yaitu PubSubClient. Lihat pada bagian menu Sketch - Include Library - Manage Libraries, cari PubSubClient. Lihat Gambar 7.2 dan Gambar 7.3. Silahkan tekan *Install*, tunggu sampai proses selesai.



Gambar 7.2. Manage Libraries



Gambar 7.3. Instal library PubSubClient

Setelah semua *library* yang dibutuhkan terinstal, silahkan buat program baru, dengan nama file sembarang. Misalkan BNU_Web_Mobile.ino. Kemudian tuliskan kode program berikut. Program Arduino lengkap untuk pengendalian Robot Cerdas BNU 4.0, via web dan *mobile*.

	BNU_Web_Mobile.ino
1	/*****
2	* Program: ROBOT BNU 4.0
3	*****/
4	#include <ESP8266WiFi.h>
5	#include <PubSubClient.h>
6	
7	String Topic;
8	String Payload;
9	
10	const char* ssid = "wifi@yah";
11	const char* password = "password_wifi";
12	
13	#define IN_1 D3//Maju
14	#define IN_2 D4//Mundur
15	#define IN_3 D7//Belok Kanan
16	#define IN_4 D8//Belok Kiri
17	
18	#define mqttServer "tailor.cloudmqtt.com"
19	#define mqttPort 13718
20	#define mqttUser "iwfywzxr"
21	#define mqttPassword "wpWz426nkZx6"

```

22
23 WiFiServer server(80);
24 WiFiClient espClient;
25 PubSubClient client(espClient);
26
27 void receivedCallback(char* topic, byte* payload, unsigned int length)
28 {
29     Serial.print("Message received: ");
30     Serial.println(topic);
31     Serial.print("payload: ");
32     for (int i = 0; i < length; i++) {
33         Serial.print((char)payload[i]);
34     }
35     Serial.println();
36
37     /* we got '1' -> Maju */
38     if ((char)payload[0] == '1') {
39         digitalWrite(IN_1, HIGH);
40         delay(100);
41         digitalWrite(IN_1, LOW);
42     }
43
44     /* we got '2' -> Kiri */
45     if ((char)payload[0] == '2') {
46         digitalWrite(IN_4, HIGH);
47         delay(300);
48         digitalWrite(IN_1, HIGH);
49         delay(100);
50         digitalWrite(IN_4, LOW);
51         digitalWrite(IN_1, LOW);
52     }
53
54     /* we got '3' -> Kanan */
55     if ((char)payload[0] == '3') {
56         digitalWrite(IN_3, HIGH);
57         delay(300);
58         digitalWrite(IN_1, HIGH);
59         delay(100);
60         digitalWrite(IN_3, LOW);
61         digitalWrite(IN_1, LOW);
62     }
63
64     /* we got '4' -> Mundur */
65     if ((char)payload[0] == '4') {
66         digitalWrite(IN_2, HIGH);
67         delay(100);
68         digitalWrite(IN_2, LOW);
69     }
70 }
71

```

```

72
73 void setup() {
74   Serial.begin(115200);
75   delay(10);
76   pinMode(IN_1, OUTPUT);
77   pinMode(IN_2, OUTPUT);
78   pinMode(IN_3, OUTPUT);
79   pinMode(IN_4, OUTPUT);
80
81   digitalWrite(IN_1, LOW);
82   digitalWrite(IN_2, LOW);
83   digitalWrite(IN_3, LOW);
84   digitalWrite(IN_4, LOW);
85
86   //Connect to WiFi network
87   Serial.println();
88   Serial.println();
89   Serial.print("Connecting to ");
90   Serial.println(ssid);
91   WiFi.begin(ssid, password);
92
93   while (WiFi.status() != WL_CONNECTED) {
94     delay(500);
95     Serial.print(".");
96   }
97   Serial.println("");
98   Serial.println("WiFi connected");
99
100  server.begin();
101  Serial.println("Server started");
102
103  Serial.print("Use this URL to connect: ");
104  Serial.print("http://");
105  Serial.print(WiFi.localIP());
106  Serial.println("/");
107
108  //Connect to Server IoT (CloudMQTT)
109
110  client.setServer(mqttServer, mqttPort);
111  client.setCallback(receivedCallback);
112
113  while (!client.connected()) {
114    Serial.println("Connecting to CCloudMQTT...");
115
116    if (client.connect("ESP32Client", mqttUser, mqttPassword)) {
117      Serial.println("connected");
118
119    } else {

```

```

120 Serial.print("failed with state ");
121 Serial.print(client.state());
122 delay(2000);
123 }
124 }
125 client.subscribe("Maju");
126 client.subscribe("Kiri");
127 client.subscribe("Kanan");
128 client.subscribe("Mundur");
129 }
130
131 void loop() {
132
133     client.loop();
134
135     WiFiClient client = server.available();
136     if (!client) {
137         return;
138     }
139
140     Serial.println("new client");
141     while(!client.available()){
142         delay(1);
143     }
144
145     String request = client.readStringUntil('\r');
146     Serial.println(request);
147     client.flush();
148
149     if (request.indexOf("/IN_1on") > 0) {
150         digitalWrite(IN_1, HIGH);
151         delay(100);
152         digitalWrite(IN_1, LOW);
153     }
154     if (request.indexOf("/IN_1off") > 0) {
155         digitalWrite(IN_1, LOW);
156     }
157     if (request.indexOf("/IN_2on") > 0) {
158         digitalWrite(IN_2, HIGH);
159         delay(100);
160         digitalWrite(IN_2, LOW);
161     }
162     if (request.indexOf("/IN_2off") > 0) {
163         digitalWrite(IN_2, LOW);
164     }
165     if (request.indexOf("/IN_3on") > 0) {
166         digitalWrite(IN_3, HIGH);
167         delay(300);
168         digitalWrite(IN_1, HIGH);

```



```

169 delay(100);
170 digitalWrite(IN_3, LOW);
171 digitalWrite(IN_1, LOW);
172 }
173 if (request.indexOf("/IN_3off") > 0) {
174 digitalWrite(IN_3, LOW);
175 }
176 if (request.indexOf("/IN_4on") > 0) {
177 digitalWrite(IN_4, HIGH);
178 delay(300);
179 digitalWrite(IN_1, HIGH);
180 delay(100);
181 digitalWrite(IN_4, LOW);
182 digitalWrite(IN_1, LOW);
183 }
184 if (request.indexOf("/IN_4off") > 0) {
185 digitalWrite(IN_4, LOW);
186 }
187
188 //Return the response
189 client.println("HTTP/1.1 200 OK");
190 client.println("Content-Type: text/html");
191 client.println("");
192 client.println("<!DOCTYPE HTML>");
193 client.println("<html>");
194 client.println("<head>");
195 client.println("<meta name='apple-mobile-web-app-capable'
content='yes' />");
196 client.println("<meta name='apple-mobile-web-app-status-bar-style'
content='black-translucent' />");
197 client.println("</head>");
198 client.println("<body bgcolor = \"#f7e6ec\">");
199 client.println("<hr/><hr>");
200 client.println("<h4><center> Robot BNU 4.0 </center></h4>");
201 client.println("<hr/><hr>");
202 client.println("<br><br>");
203 client.println("<br><br>");
204 client.println("<center>");
205 client.println("ROBOT");
206 client.println("<a href='\"/IN_1on\"'><button>Maju </button></a>");
207 client.println("<a href='\"/IN_2on\"'><button>Mundur </button></
a><br/>");
208 client.println("</center>");
209 client.println("<br><br>");
210 client.println("<center>");
211 client.println("ROBOT");
212 client.println("<a href='\"/IN_3on\"'><button>Belok Kanan </button></
a>");

```

```

213 client.println("<a href=\""/IN_4on\""/><button>Belok Kiri </button></
a><br/>");
214 client.println("</center>");
215 client.println("<br><br>");
216 client.println("<center>");
217 client.println("<table border=\"5\">");
218 client.println("<tr>");
219 //=====
220 if (digitalRead(IN_1))
221 {
222 client.print("<td>Maju = ON</td>");
223 }
224 else
225 {
226 client.print("<td>Maju = OFF</td>");
227 }
228 client.println("<br/>");
229 //=====
230 if (digitalRead(IN_2))
231 {
232 client.print("<td>Mundur = ON</td>");
233 }
234 else
235 {
236 client.print("<td>Mundur = OFF</td>");
237 }
238 client.println("</tr>");
239 //=====
240 if (digitalRead(IN_3))
241 {
242 client.print("<td>Belok Kanan = ON</td>");
243 }
244 else
245 {
246 client.print("<td>Belok Kanan = OFF</td>");
247 }
248 client.println("<br/>");
249 //=====
250 if (digitalRead(IN_4))
251 {
252 client.print("<td>Belok Kiri = ON</td>");
253 }
254 else
255 {
256 client.print("<td>Belok Kiri = OFF</td>");
257 }
258 client.println("</tr>");
259 //=====
260

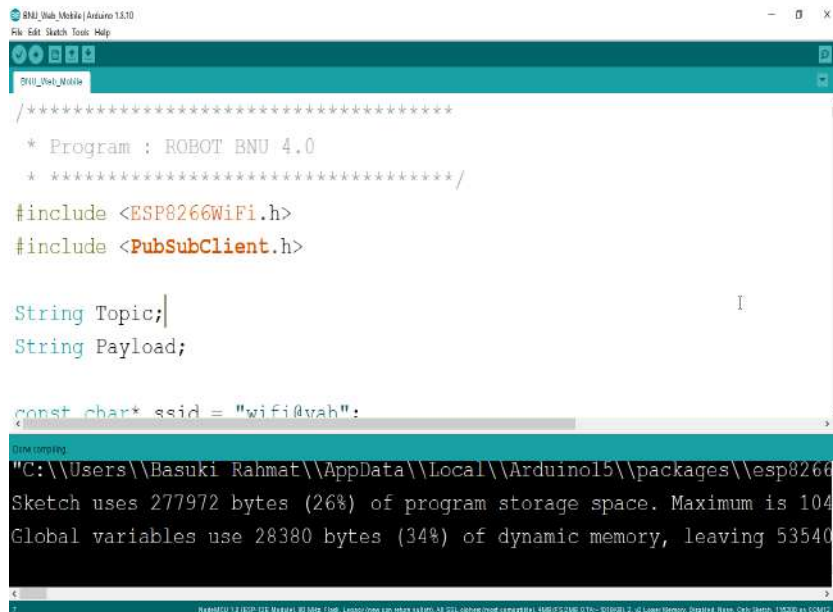
```

```

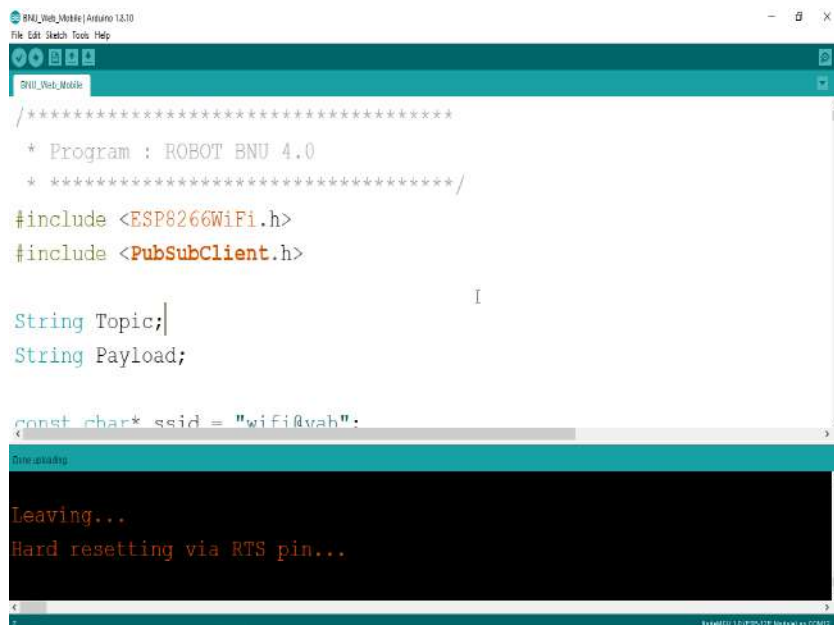
261 client.println("<tr>");
262 client.println("</table>");
263 client.println("</center>");
264 client.println("</html>");
265 delay(1);
266 Serial.println("Client disconnected");
267 Serial.println("");
268 }

```

Setelah itu, lakukan proses verifikasi atau kompilasi dan silahkan *upload* programnya jika sudah tidak ada kesalahan. Lihat Gambar 7.4 dan Gambar 7.5. Kemudian, untuk melihat hasil program, silahkan arahkan ke menu Tools – Serial Monitor. Maka akan tampil, hasilnya di serial monitor, seperti pada Gambar 7.6. Jika tidak muncul, silahkan cek posisi *baud rate* 11520. Jika masih tidak muncul juga, silahkan tekan tombol RST NodeMCU V3. Cara lain, tinggal cabut kabel USB dan tancap kembali.

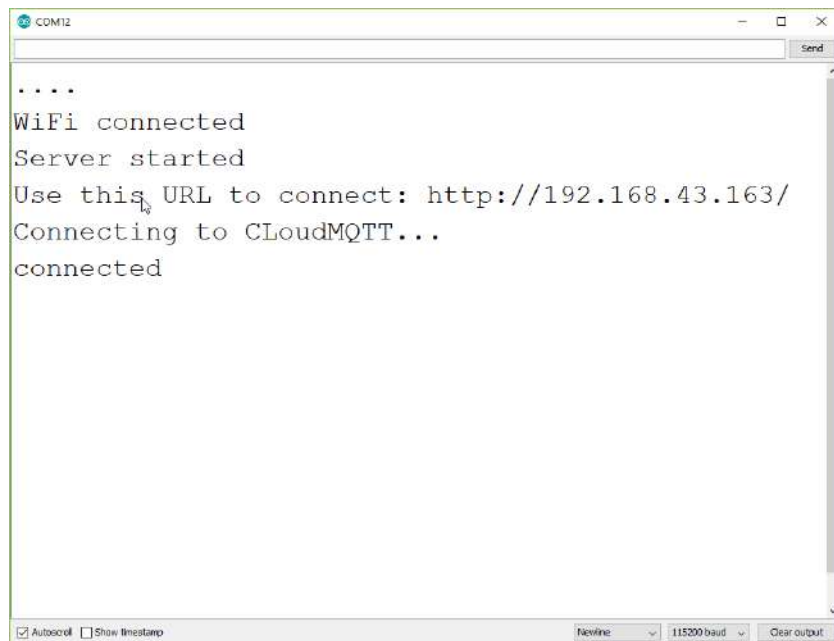


Gambar 7.4. Kompilasi sukses



```
*****  
* Program : ROBOT BNU 4.0  
* *****/  
#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
  
String Topic;  
String Payload;  
  
const char* ssid = "wifi@yah".  
  
leaving...  
Hard resetting via RTS pin...
```

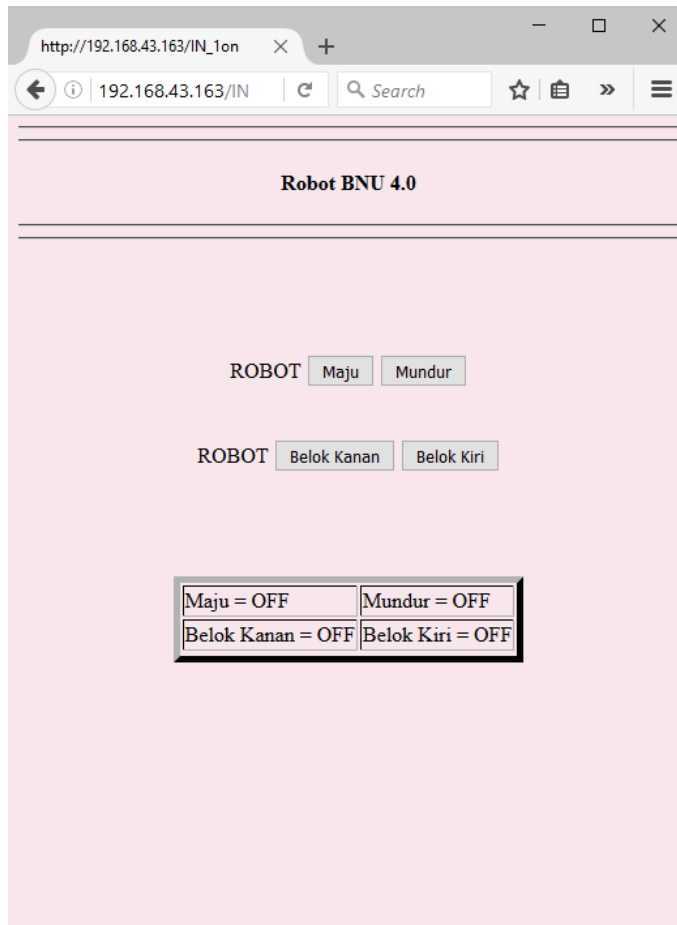
Gambar 7.5. Upload sukses



```
....  
WiFi connected  
Server started  
Use this URL to connect: http://192.168.43.163/  
Connecting to CCloudMQTT...  
connected
```

Gambar 7.6. Tampilan di serial monitor

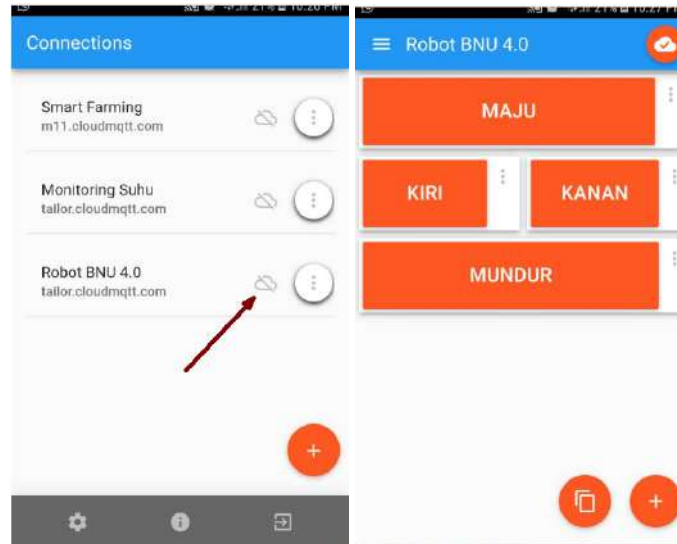
Selanjutnya, untuk pengendalian Robot BNU 4.0 melalui web, tinggal jalankan browser, dan ketikkan alamat IP seperti yang terlihat pada serial monitor. Dalam contoh ini diperoleh alamat IP `http://192.168.43.163`. Setelah dijalankan di browser maka akan tampil tombol-tombol pengendalian untuk menggerakkan Robot BNU 4.0 berjalan maju, mundur, belok kanan dan belok kiri. Lihat Gambar 7.7. Silahkan dicoba, dan dilihat respon bagaimana robot bergerak. Jangan lupa baterai driver motor L298N dinyalakan.



Gambar 7.7. Pengendalian Robot BNU 4.0 via web

Kemudian, untuk pengendalian Robot BNU 4.0 secara *mobile*, silahkan jalankan IoT MQTT Panel di ponsel. Dan seperti yang telah dijelaskan di Bab 3, harus terhubung ke Cloud MQTT. Jika sudah berhasil terhubung ke Cloud MQTT, maka akan tampil tombol-tombol pengendalian untuk menggerakkan Robot BNU 4.0 berjalan maju, belok kiri, belok kanan dan mundur, melalui ponsel. Lihat Gambar 7.8. Silahkan dicoba,

dan dilihat respon bagaimana robot bergerak. Jangan lupa baterai driver motor L298N dinyalakan.



Gambar 7.8. Koneksi ke Cloud MQTT dan panel pengendalian Robot BNU 4.0 via ponsel

Untuk melihat respon hasil penekanan tombol di ponsel, juga bisa dipantau melalui serial monitor. Gambar 7.9, menunjukkan contoh tampilan di serial monitor jika keempat tombol sudah ditekan. Dan Robot BNU 4.0 bergerak sesuai yang diprogram.



Gambar 7.9. Contoh pantauan penekanan tombol ponsel melalui serial monitor

7.2. Pengenalan Objek dengan Deep Learning Menggunakan Kamera Kinect Xbox 360

Akhirnya, sampailah pada tahap akhir, yaitu pemrograman pengenalan objek dari Robot Cerdas BNU 4.0. Pemrograman *Deep Learning* untuk pengenalan objek secara waktu nyata memanfaatkan modul DNN dari OpenCV. Kamera yang digunakan, yaitu Kinect Xbox 360. Untuk keperluan ini, kamera webcam dari komputer atau laptop di-*disable* kembali. Selanjutnya digunakan kamera Kinect Xbox 360, yang menjadi andalan dari Robot Cerdas BNU 4.0.

Program pengenalan objek secara waktu nyata berbasis *Deep Learning* di Bab 7 ini, adalah gabungan dari program yang telah dibuat sebelumnya di Bab 6 dan Bab 5. Program yang ada di Bab 6, silahkan di-*copy* dengan nama lain, misalnya `deteksi_objek_kinect.py`. Kemudian silahkan disesuaikan dengan beberapa penambahan pemrograman kamera Kinect Xbox 360 yang sudah dijelaskan di Bab 5. Sehingga secara lengkap, programnya menjadi berikut ini.

	deteksi_objek_kinect.py
1	# import the necessary packages
2	from imutils.video import VideoStream
3	from imutils.video import FPS
4	import numpy as np
5	import argparse
6	import imutils
7	import time
8	
9	from pykinect import nui
10	import cv2
11	from pykinect.nui import JointId
12	from numpy import *
13	import datetime
14	import os
15	import os.path
16	from os.path import join as pjoin
17	from pykinect.nui import SkeletonTrackingState
18	from PIL import Image
19	import shutil
20	import copy
21	
22	save_image = True
23	
24	current_directory = os.getcwd()
25	kinect = nui.Runtime()
26	
27	def video_handler_function(frame):
28	video = np.empty((480,640,4),np.uint8)
29	frame.image.copy_bits(video.ctypes.data)

```

30 cv2.imshow('KINECT Video Stream', video)
31 # Update Gambar hasil tracking
32 image_name = 'belgedes.jpg'
33 if save_image:
34     cv2.imwrite(image_name, video)
35
36 # construct the argument parser and parse the
37 # arguments
38 ap = argparse.ArgumentParser()
39 ap.add_argument("-p", "--prototxt", required = True,
40 help = "path to Caffe 'deploy' prototxt file")
41 ap.add_argument("-m", "--model", required = True,
42 help = "path to Caffe pre-trained model")
43 ap.add_argument("-c", "--probability", type = float,
44 default = 0.2, help = "minimum probability to filter weak detections")
45 args = vars(ap.parse_args())
46
47 # initialize the list of class labels MobileNet SSD was
48 # trained to detect
49 # and generate a set of bounding box colors for each
50 # class
51 CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
52 "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
53 "dog", "horse", "motorbike", "person", "pottedplant", "sheep", "sofa",
54 "train", "tvmonitor"]
55
56 COLORS = np.random.uniform(0, 255, size = (len(CLASSES), 3))
57
58 # load our serialized model from disk
59 print("[INFO] loading model...")
60 net = cv2.dnn.readNetFromCaffe(args["prototxt"],
61 args["model"])
62
63 # initialize the video stream, allow the camera sensor
64 # to warmup, and initialize the FPS counter
65 print("[INFO] starting video stream...")
66 #vs = VideoStream(src=0).start()
67
68 kinect = nui.Runtime()
69 kinect.video_frame_ready += video_handler_function
70 kinect.video_stream.open(nui.ImageStreamType.Video, 2, nui.
71 ImageResolution.Resolution640x480, nui.ImageType.Color)
72 cv2.namedWindow('KINECT Video Stream', cv2.WINDOW_AUTOSIZE)
73
74 time.sleep(2.0)
75 fps = FPS().start()
76
77 # loop over the frames from the video stream
78 while True:
79     # resize the video stream window at a maximum
80     # width of 500 pixels

```



```

74
75     shutil.copy("belgedes.jpg","belgedes2.jpg")
76
77     frame_gambar = cv2.imread("belgedes2.jpg")
78
79     frame_gambar = imutils.resize(frame_gambar, width=500)
80
81     # grab the frame dimensions and convert it to a
82     # blob Binary Large Object = BLOB
83
84     (h, w) = frame_gambar.shape[:2]
85     blob = cv2.dnn.blobFromImage(cv2.resize(frame_gambar,
86     (500, 500)), 0.007843, (500, 500), 127.5)
87
88     # pass the blob through the network and get the
89     # detections
90     net.setInput(blob)
91     detections = net.forward()
92
93     # loop over the detections
94     for i in np.arange(0, detections.shape[2]):
95         # extract the probability of the prediction
96         probability = detections[0, 0, i, 2]
97
98         # filter out weak detections by ensuring that
99         # probability is greater than the min
100         # probability
101         if probability > args["probability"]:
102             # extract the index of the class label
103             # from the 'detections', then compute
104             # the (x, y)-coordinates of
105             # the bounding box for the object
106             idx = int(detections[0, 0, i, 1])
107             box = detections[0, 0, i, 3:7] *
108             np.array([w, h, w, h])
109             (startX, startY, endX, endY) = box.astype("int")
110
111             # draw the prediction on the frame
112             label = "{}:
113             {:.2f}%".format(CLASSES[idx], probability * 100)
114             cv2.rectangle(frame_gambar, (startX, startY), (endX,
115             endY), COLORS[idx], 2)
116             y = startY - 15 if startY - 15 > 15 else startY + 15
117             cv2.putText(frame_gambar, label, (startX, y), cv2.FONT_
118             HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
119
120     # show the output frame
121     cv2.imshow("KINECT Video Stream", frame_gambar)
122     key = cv2.waitKey(1) & 0xFF

```

```

120     # if the 'q' key was pressed, break from the loop
121     if key == ord("q"):
122         break
123
124     # update the FPS counter
125     fps.update()
126
127     # stop the timer and display FPS information
128     fps.stop()
129     print("[INFO] elapsed time:
130     {:.2f}".format(fps.elapsed()))
131     print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
132
133     # cleanup
134     kinect.close()
135     cv2.destroyAllWindows()
136     #vs.stop()

```

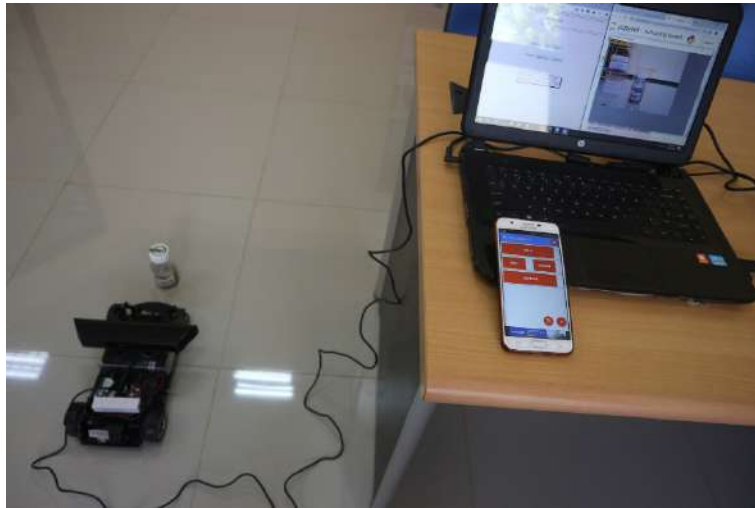
Keterangan:

Dari skrip program deteksi_objek_kinect.py ini, untuk menangkap hasil gambar *frame* sementara (*temporary*), digunakan file gambar belgedes.jpg dan belgedes2.jpg. Bisa disiapkan dari file JPG sembarang dengan nama dua file tersebut.

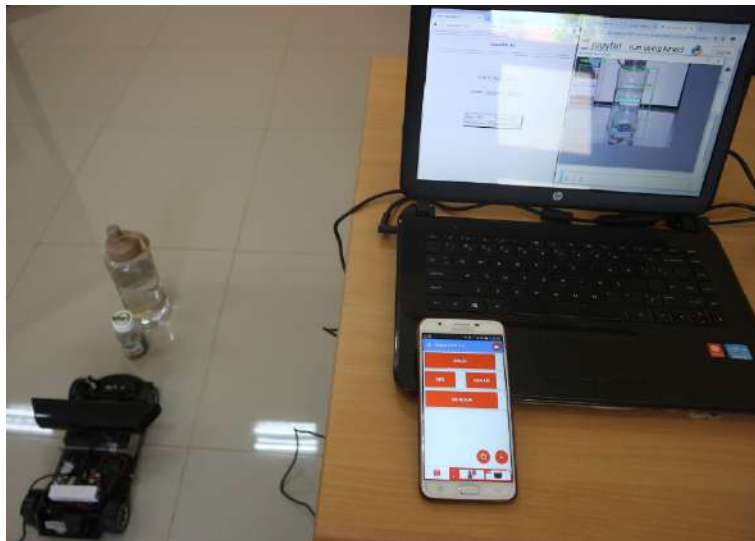
Gambar 7.10 sampai dengan Gambar 7.24 berikut ini adalah gambaran saat Robot Cerdas BNU 4.0 diuji-coba maupun saat digunakan untuk pengambilan data. Termasuk hasil pengenalan objek yang telah dilakukan. Robot ini bisa dibilang Robot Cerdas BNU 4.0 Generasi Pertama, terbuka peluang untuk terus dikembangkan menjadi Generasi Kedua dan seterusnya dengan peningkatan kemampuan fisik (*mechatronics system*) dan algoritma kecerdasannya.



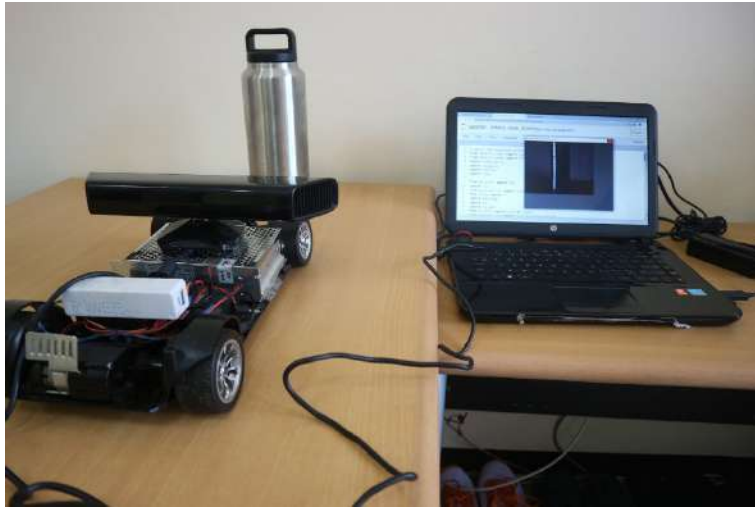
Gambar 7.10. Pemrograman Arduino Robot Cerdas BNU 4.0 melalui koneksi USB



Gambar 7.11. Posisi pengujian Robot Cerdas BNU 4.0 dan pengambilan data



Gambar 7.12. Posisi pengujian Robot Cerdas BNU 4.0 dan pengambilan data (lanjutan)



Gambar 7.13. Posisi pengujian Robot Cerdas BNU 4.0 dan pengambilan data (lanjutan)



Gambar 7.14. Pengujian Robot Cerdas BNU 4.0 dan pengambilan data



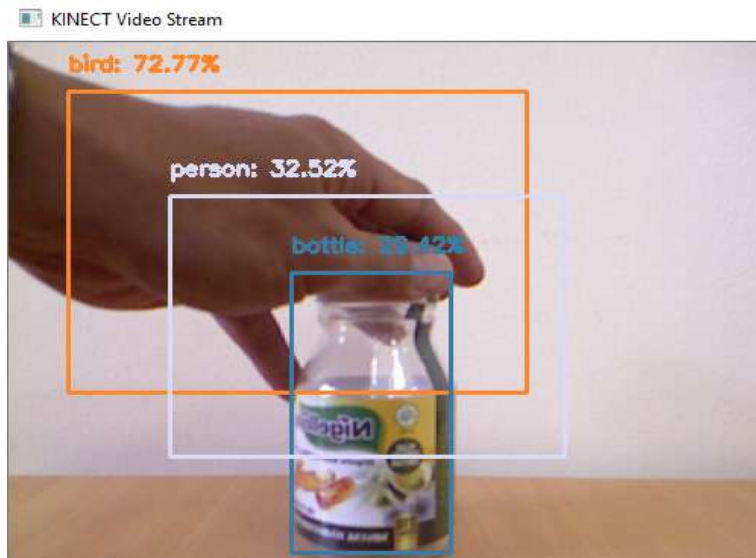
Gambar 7.15. Pengujian Robot Cerdas BNU 4.0 dan pengambilan data (lanjutan)



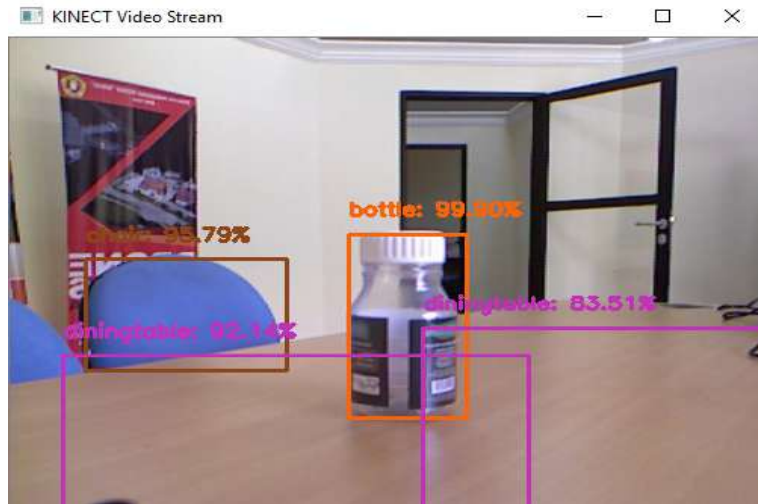
Gambar 7.16. Pengujian Robot Cerdas BNU 4.0 dan pengambilan data (lanjutan)



Gambar 7.17. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning



Gambar 7.18. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



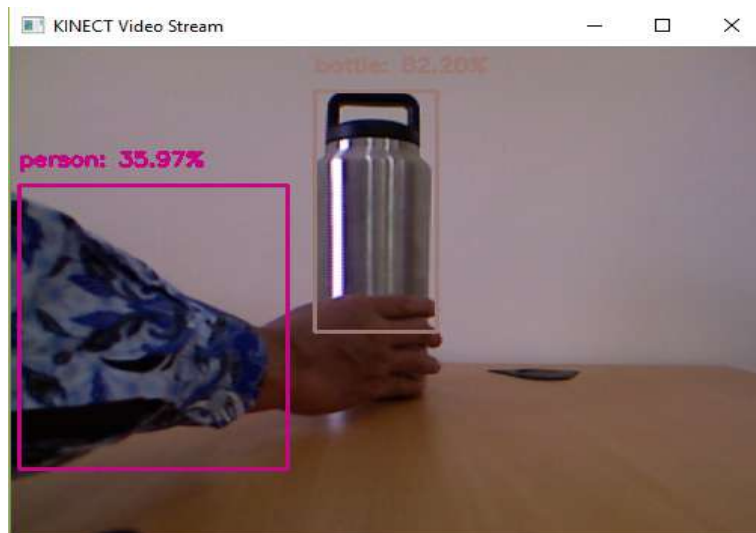
Gambar 7.19. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



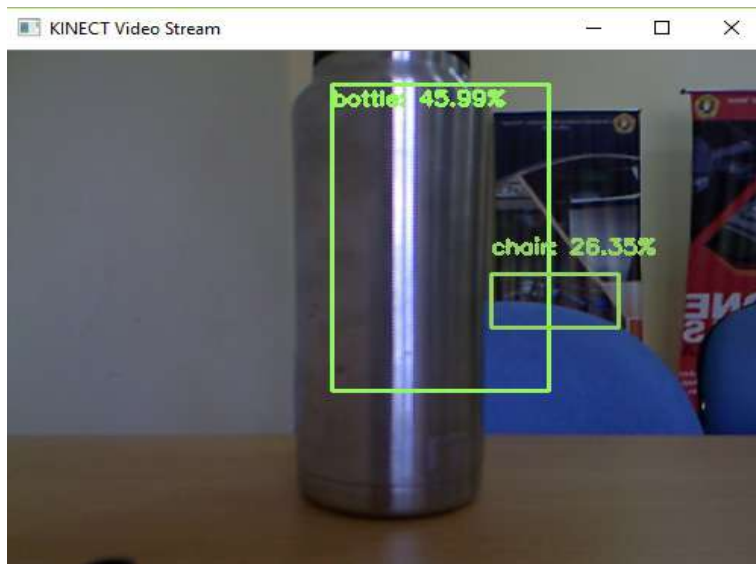
Gambar 7.20. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



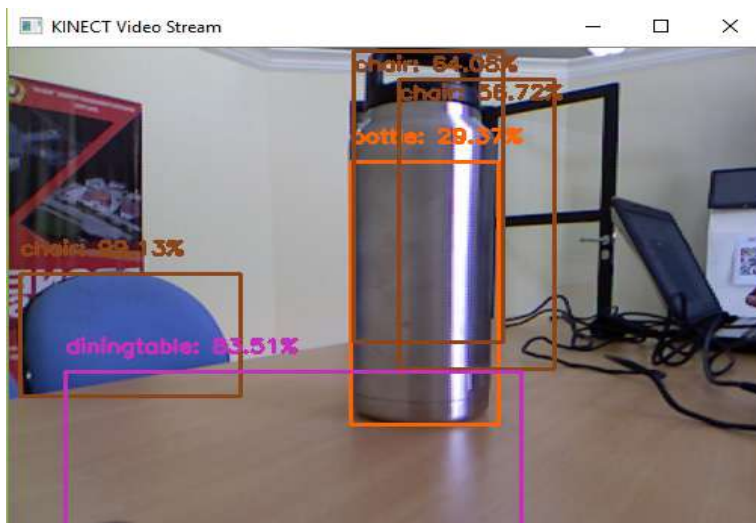
Gambar 7.21. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



Gambar 7.22. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



Gambar 7.23. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



Gambar 7.24. Contoh hasil deteksi objek Robot BNU 4.0 menggunakan Deep Learning (lanjutan)



BAB 8

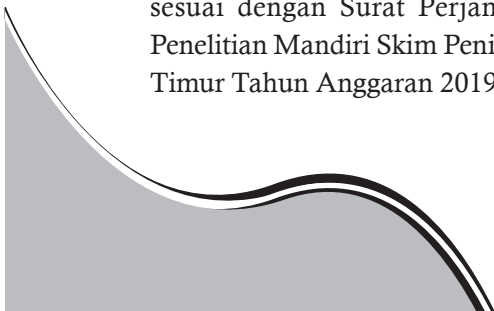
Penutup

8.1. Kesimpulan

Telah diuraikan dalam buku ini bagaimana merancang dan membuat Robot Cerdas BNU 4.0 berupa robot pengawasan. Otak kendali robot ini berupa mikrokontroler NodeMCU V3. Board elektronik yang berbasis chip ESP8266. Dengan chip ini, NodeMCU V3 memiliki kemampuan koneksi ke *cloud Internet of Things* (IoT). Cloud IoT yang digunakan di buku ini yaitu cloudmqtt (<https://www.cloudmqtt.com>). Dengan program Arduino yang tertanam di mikrokontroler NodeMCU V3, selanjutnya dapat dijalankan program kendali robot berbasis web dan *mobile*. Program kendali robot secara *mobile* memanfaatkan *Android Apps* IoT MQTT Panel. Robot ini juga dilengkapi dengan kamera Kinect Xbox 360. Kamera ini terhubung ke komputer. Dengan program *Deep Learning* menggunakan bahasa pemrograman Python yang ada di komputer, selanjutnya sistem akan mengenali objek yang ada di depannya.

8.2. Ucapan Terimakasih

Terimakasih kepada Kementerian Riset, Teknologi dan Pendidikan Tinggi Universitas Pembangunan Nasional “Veteran” Jawa Timur yang telah memberikan dana penelitian sesuai dengan Surat Perjanjian Penugasan dalam Rangka Pelaksanaan Program Penelitian Mandiri Skim Peningkatan Mutu Pembelajaran (PMP) UPN “Veteran” Jawa Timur Tahun Anggaran 2019, Nomor: SPP/1/UN63.8/LT/IV/2019.



Daftar Pustaka

- Adam Gibson, J. P. (2017) 'Deep Learning', *Deep Learning*. O'Reilly Media, Inc. Available at: <https://learning.oreilly.com/library/view/deep-learning/9781491924570/>.
- anonymous (2019) 'mCLOUD IoT Platform Services', *mthinx.com*.
- Anzola, J., Jiménez, A. and Tarazona, G. (2019) 'Self-sustainable power-collecting node in IoT', *Internet of Things*, 7, p. 100082. doi: <https://doi.org/10.1016/j.iot.2019.100082>.
- Azeta, J. *et al.* (2019) 'An Android Based Mobile Robot for Monitoring and Surveillance', *Procedia Manufacturing*, 35, pp. 1129–1134. doi: <https://doi.org/10.1016/j.promfg.2019.06.066>.
- Chahal, R. K., Kumar, N. and Batra, S. (2020) 'Trust management in social Internet of Things: A taxonomy, open issues, and challenges', *Computer Communications*, 150, pp. 13–46. doi: <https://doi.org/10.1016/j.comcom.2019.10.034>.
- Cruz, L., Lucio, D. and Velho, L. (2012) 'Kinect and RGBD Images: Challenges and Applications', in. doi: 10.1109/SIBGRAPI-T.2012.13.
- Day, C.-P. (2018) 'Robotics in Industry—Their Role in Intelligent Manufacturing', *Engineering*, 4(4), pp. 440–445. doi: <https://doi.org/10.1016/j.eng.2018.07.012>.
- Fankhauser, P. *et al.* (2015) 'Kinect v2 for Mobile Robot Navigation: Evaluation and Modeling', in. doi: 10.1109/ICAR.2015.7251485.
- Gheisari, M., Wang, G. and Chen, S. (2020) 'An Edge Computing-enhanced Internet of Things Framework for Privacy-preserving in Smart City', *Computers*

- & *Electrical Engineering*, 81, p. 106504. doi: <https://doi.org/10.1016/j.compeleceng.2019.106504>.
- Hinton, G. E., Osindero, S. and Teh, Y.-W. (2006) 'A Fast Learning Algorithm for Deep Belief Nets', *Neural Comput.* Cambridge, MA, USA: MIT Press, 18(7), pp. 1527–1554. doi: [10.1162/neco.2006.18.7.1527](https://doi.org/10.1162/neco.2006.18.7.1527).
- Józwiak, L. (2017) 'Advanced mobile and wearable systems', *Microprocessors and Microsystems*, 50, pp. 202–221. doi: <https://doi.org/10.1016/j.micpro.2017.03.008>.
- Kashyap, M., Sharma, V. and Gupta, N. (2018) 'Taking MQTT and NodeMcu to IOT: Communication in Internet of Things', *Procedia Computer Science*, 132, pp. 1611–1618. doi: <https://doi.org/10.1016/j.procs.2018.05.126>.
- Kharkovyna, O. (2019) 'Top 10 Best Deep Learning Frameworks in 2019'. Towards Data Science. Available at: <https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de>.
- Meng, T. *et al.* (2020) 'A survey on machine learning for data fusion', *Information Fusion*, 57, pp. 115–129. doi: <https://doi.org/10.1016/j.inffus.2019.12.001>.
- Nayyar, A. and Kumar, A. (2019) *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*. Springer International Publishing (Advances in Science, Technology & Innovation). Available at: https://books.google.co.id/books?id=fA_gwQEACAAJ.
- Oppermann, A. (2019) 'Artificial Intelligence vs. Machine Learning vs. Deep Learning', in. DeepLearning Academy. Available at: <https://www.deeplearning-academy.com/p/ai-wiki-machine-learning-vs-deep-learning>.
- Raina, R., Madhavan, A. and Ng, A. Y. (2009) 'Large-Scale Deep Unsupervised Learning Using Graphics Processors', in *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery (ICML '09), pp. 873–880. doi: [10.1145/1553374.1553486](https://doi.org/10.1145/1553374.1553486).
- Ravidas, S. *et al.* (2019) 'Access control in Internet-of-Things: A survey', *Journal of Network and Computer Applications*, 144, pp. 79–101. doi: <https://doi.org/10.1016/j.jnca.2019.06.017>.
- Rong, G. *et al.* (2020) 'Artificial Intelligence in Healthcare: Review and Prediction Case Studies', *Engineering*. doi: <https://doi.org/10.1016/j.eng.2019.08.015>.
- de Sousa, W. G. *et al.* (2019) 'How and where is artificial intelligence in the public sector going? A literature review and research agenda', *Government Information Quarterly*, 36(4), p. 101392. doi: <https://doi.org/10.1016/j.giq.2019.07.004>.
- Zainuddin, N. *et al.* (2015) 'Autonomous Navigation of Mobile Robot Using Kinect Sensor', pp. 28–31. doi: [10.1109/ICCCE.2014.21](https://doi.org/10.1109/ICCCE.2014.21).
- Zeadally, S. and Bello, O. (2019) 'Harnessing the power of Internet of Things based connectivity to improve healthcare', *Internet of Things*, p. 100074. doi: <https://doi.org/10.1016/j.iot.2019.100074>.