# ROBOTIKA


Disusun Oleh :

**Dr. Raden Supriyanto**

**Hustinawati, SKom., MMSI.**

**Rigathi Widya Nugraini, SKom.**

**Ary Bima Kurniawan, ST., MT.**

**Yogi Permadi, SKom.**

**Abdurachman Sa'ad, SKom.**

Jurusan Sistem Komputer Fakultas Ilmu Komputer

**UNIVERSITAS GUNADARMA**



**2010**

# KATA PENGANTAR

Dengan memuji dan mengucap syukur kepada Allah Tuhan Yang Maha Agung, yang telah memberikan karunia kekuatan dan kesabaran kepada Penulis. Berkat karunia ini, Penulis sampai pada langkah akhirnya, pembuatan aplikasi dan penulisan Buku Ajar ini dapat diselesaikan.

Penulis mengharapkan pembaca dapat menyerap informasi secara keseluruhan dari buku ajar ini.

Penulis menyadari bahwa dokumen ini masih banyak kekurangan, baik penyajian ataupun kekurangtepatan dalam penjelasan. Penulis dengan senang hati akan menerima saran dan perbaikan dari pembaca.

Semoga Buku Ajar ini mampu memberikan pengetahuan dan manfaat yang sebesar-besarnya bagi pembaca dan merangsang perbaikan lebih lanjut, Amien.

Jakarta, November 2010

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

Halaman

# DAFTAR TABEL

Halaman

# BAB I
# DASAR-DASAR ROBOTIKA

## 1.1 Pendahuluan

Istilah robot berasal dari bahasa Cekoslowakia. Kata robot berasal dari kosakata "Robota" yang berarti "kerja cepat". Istilah ini muncul pada tahun 1920 oleh seorang pengarang sandiwara bernama Karel Capec. Karyanya pada saat itu berjudul "Rossum's Universal Robot" yang artinya Robot Dunia milik Rossum. Rossum merancang dan membangun suatu bala tentara yang terdiri dari robot industri yang akhirnya menjadi terlalu cerdik dan akhirnya menguasai manusia.

Kata Robotika juga berasal dari novel fiksi sains "runaround" yang ditulis oleh Isaac Asimov pada tahun 1942. Sedangkan pengertian robot secara tepat adalah sistem atau alat yang dapat berperilaku atau meniru perilaku manusia dengan tujuan untuk menggantikan dan mempermudah kerja/aktifitas manusia.

Untuk dapat diklasifikasikan sebagai robot, maka robot harus memiliki dua macam kemampuan yaitu:

1) Bisa mendapatkan informasi dari sekelilingnya.
2) Bisa melakukan sesuatu secara fisik seperti bergerak atau memanipulasi objek.

Untuk dapat dikatakan sebagai robot sebuah sistem tidak perlu untuk meniru semua tingkah laku manusia, namun suatu sistem tersebut dapat mengadopsi satu atau dua dari sistem yang ada pada diri manusia saja sudah dapat dikatakan sebagai robot. Sistem yang diadopsi dapat berupa sistem penglihatan (mata), sistem pendengaran (telinga) ataupun sistem gerak.

Sebuah robot dapat saja dibuat untuk berbagai macam aktifitas, namun sebuah robot harus dibuat dengan tujuan untuk kebaikan manusia.

Ada beberapa fungsi robot, sehingga manusia memerlukan kehadirannya yaitu:
1. Meningkatkan produksi, akurasi dan daya tahan. Robot ini banyak digunakan di industri.
2. Untuk tugas-tugas yang berbahaya, kotor dan beresiko. Robot ini digunakan ketika manusia tidak mampu masuk ke daerah yang beresiko. Seperti Robot Untuk menjelajah planet, robot untuk mendeteksi limbah nuklir, robot militer dll.
3. Untuk pendidikan. Banyak robot yang digunakan untuk menarik pelajar belajar teknologi seperti robot lego, dll.
4. Untuk menolong manusia. Seperti di rumah untuk membersihkan rumah pakai penghisap debu otomatis, di rumah sakit untuk menghantar makanan, membantu operasi, dll.

## 1.2 Sejarah dan Perkembangan Teknologi Robot

Keunggulan dalam teknologi robotika tak dapat dipungkiri telah lama dijadikan ikon kebanggan negara-negara maju di dunia. Kecanggihan teknologi yang dimiliki, gedung-gedung tinggi yang mencakar langit, tingkat kesejahteraan rakyatnya yang tinggi, kota-kotanya yang modern, belumlah terasa lengkap tanpa popularitas kepiawaian dalam dunia robotik.

Menurut fu, et al. (1987) penelitian dan pengembangan pertama yang berbuah produk robotika dapat dilacak mulai dari tahun 1940-an ketika Argone National Laboratories di Oak Ridge, Amerika, memperkenalkan sebuah mekanisme robotika yang dinamai master-slave manipulator. Robot ini digunakan untuk menangani material radioaktif. Kemudian produk pertama robot komersial diperkenalkan oleh Unimation Incorporated, Amerika pada tahun 1950-an. Hingga belasan tahun kemudian langkah komersial ini telah diikuti oleh perusahaan-perusahaan lain. Namun demikian, seperti ditulis dalam beberapa sumber, penelitian intensif dibidang teknologi robotika dan keinginan menjadikan robotika sebagai sebuah disiplin ilmu kala itu belum terpikirkan.

Baru setelah dunia mulai menapak ke jaman industri pada pertengahan tahun 60-an kebutuhan akan otomasi makin menjadi-jadi. Pada saat itulah robotika diterima sebagai disiplin ilmu baru yang mendampingi ilmu-ilmu dasar dan teknik yang telah mapan sebelumnya. Di negara-negara yang telah

mapan kala itu, seperti Amerika, Inggris, Jerman dan Perancis mulai bermunculan grup-grup riset yang menjadikan robotika sebagai temanya, kemudian diikuti oleh Jepang, yang dipelopori oleh ilmuwan-ilmuwan yang baru pulang dari menimba ilmu di Amerika. Bahkan, di kemudian hari Jepang-lah yang tercatat sebagai negara yang paling produktif dalam mengembangkan teknologi robot. Hal ini tidak lain karena jepang gigih dalam melakukan penelitian teknologi infrastruktur seperti komponen dan piranti mikro (microdevices) yang akhirnya bidang ini terbukti sebagai inti dari pengembangan robot modern.

Dewasa ini mungkin definisi robot industri itu sudah tidak sesuai lagi karena teknologi mobile robot sudah dipakai secara meluas sejak tahun 80-an. Seiring itu pula kemudian muncul istilah robot humanoid, animaloid, dan sebagainya. Bahkan kini dalam industri spesifik seperti industri perfilman, industri angkasa luar dan industri pertahanan atau mesin perang, robot arm atau manipulator bisa jadi hanya menjadi bagian saja dari sistem robot secara keseluruhan.

Robotika memiliki unsur yang sedikit berbeda dengan ilmu-ilmu dasar atau terapan yang lain dalam berkembang. Ilmu dasar biasanya berkembang dari suatu asa atau hipotesis yang kemudian diteliti secara metodis. Ilmu terapan dikembangkan setelah ilmu-ilmu yang mendasarinya berkembang dengan baik. Sedangkan ilmu robotika lebih sering berkembang melalui pendekatan secara praktis pada awalnya. Kemudian melalui suatu pendekatan atau perumpamaan dari hasil pengamatan perilaku mahluk hidup atau benda/mesin/peralatan bergerak lainnya dikembangkanlah penelitian secara teoritis. Dari teori kembali kepada praktis, dan dari robot berkembang menjadi lebih canggih.

Mekatronik adalah istilah umum yang menjadi popular seiring dengan perkembangan padu mekanik dan elektronik. Mekatronik terdiri dari 4 disiplin ilmu, yaitu mekanik (mechanics), elektronik, teknik kontrol berbasis prosesor serta pemrograman seperti halya dalam bidang robotik. Sebuah produk mekatronik belum tentu robot, namun robot pasti mekatronik. Banyak produk mekatronik disekeliling kita, misalnya mesin cuci, CD/DVD/ video/cassette player, walkman hingga vacuum cleaner. Dalam bidang otomotif produk mekatronik yang diterapkan pada mobil yaitu ABS (anti lock breaking sistem),

active suspension sistem, dsb. Dalam dunia industri, perdagangan dan gedung-gedung perkantoran dikenal berbagai peralatan otomatis seperti pintu otomatis, lift, escalator, mesin fotocopy, dan masih banyak lagi.

Penelitian bidang robotika dalam kehidupan organik (bio science) juga semakin mendalam dan bahkan cenderung tak teduga arahnya. Jika dalam dunia kedokteran telah dikenal teknik kloning mahluk hidup yang kontroversial itu, maka dalam dunia robotika juga dikenal suatu proyek penelitian yang disebut sebagai implant sensor/aktuator atau implant interface. Interface berupa chip IC berukuran mikro ditanamkan ke dalam mahluk hidup dengan tujuan agar komputer dapat di luar dapat mengendalikan dan atau memonitor kegiatan saraf organik manusia secara langsung di dalam pembuluh darah atau saraf tubuh.

## 1.3 Penelitian di Bidang Robotika

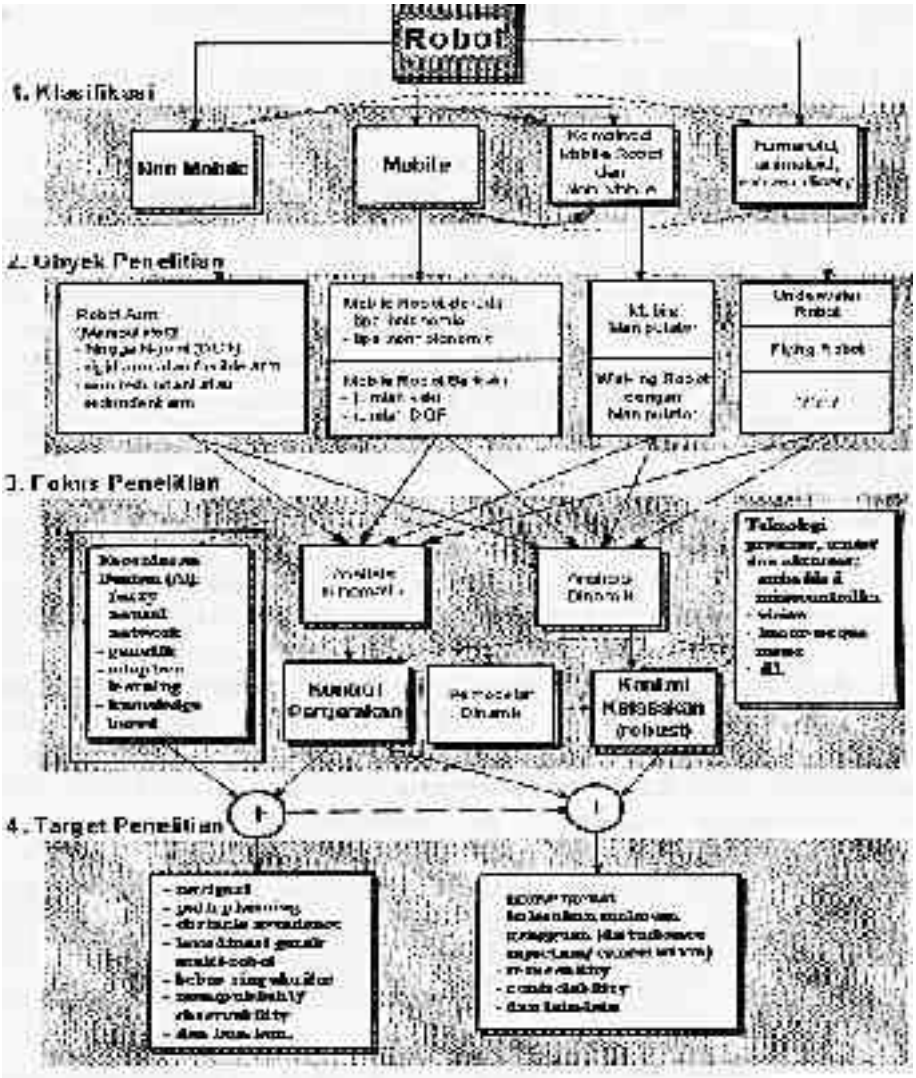
Perkembangan suatu ilmu tak lepas dari peran para peneliti kalau tak dapat dikatakan bahwa justru penelitilah yang menyebabkan suatu ilmu itu berkembang. Robotika memiliki unsur yang sedikit berbeda dengan ilmu-ilmu dasar atau terapan yang lain dalam berkembang. Ilmu dasar biasanya berkembang dari suatu asas atau hipotesis yang kemudian diteliti secara metodis. Ilmu terapan dikembangkan setelah ilmu-ilmu yang mendasarinya berkembang dengan baik, sedangkan ilmu robotika lebih sering berkembang melalui pendekatan praktis pada awalnya. Kemudian melalui suatu pendekatan atau perumpamaan (asumsi) dari hasil pengamatan perilaku mahluk hidup atau benda/mesin/peralatan bergerak lainnya dikembangkanlah penelitian secara teoritis. Dari teori kembali kepada praktis, dan dari sini robot berkembang menjadi canggih.

Perkembangan penelitian di bidang robotika lazimnya dapat segera diketahui dengan mencermati aplikasinya di dunia industri atau produk kegiatan penelitian skala laboratorium di group-group penelitian yang tersebar di berbagai institusi pendidikan dan penelitian di negara-negara maju. Dengan mudahnya mengakses internet sekarang ini dan banyaknya sumber-sumber informasi masa kini yang tersebar secara terbuka di situs-situs penelitian, maka dalam mencari tahu suatu perkembangan terbaru dalam dunia robotika menjadi sangat mudah.

Untuk mengetahui dalam tema apa saja robotika dapat diteliti, maka **Gambar 1.1** dapat mengilustrasikannya. Di dalam gambar dijelaskan tentang keterkaitan seluruh komponen atau sub-domain dalam ruang lingkup penelitian di bidang robotika. Secara garis besar penelitian di bidang robotika dapat dilakukan dengan memilih tema berdasarkan alur dalam 4 tahapan, yaitu klasifikasi, obyek penelitian, fokus penelitian dan target penelitian. Dari blok klarifikasi, struktur robot dapat diketahui berada dalam kelompok mana. Dari sini, obyek penelitian dapat ditentukan dan dijabarkan secara detil parameter-parameternya.

Pada dasarnya dilihat dari struktur dan fungsi fisiknya (pendekatan visual) robot terdiri dari dua bagian, yaitu non-mobile robot dan mobile robot. Kombinasi keduanya dapat menghasilkan kelompok kombinasi konvensional (mobile & non-mobile) dan kelompok non-konvensional. Kelompok pertama sengaja diberi nama konvensional karena nama yang dipakai dalam konteks penelitian adalah nama-nama yang dianggap sudah umum, seperti mobile manipulator, climbing robot (robot pemanjat), walking robot (misalkan : bi-ped robot) dan nama-nama lain yang sudah populer. Sedangkan kelompok non-konvensional dapat berupa robot humanoid, animaloid, extra-ordinary, atau segala bentuk inovasi penyerupaan yang bisa dilakukan. Kelompok kedua ini banyak di manfaatkan sebagai ikon keunggulan dalam penelitian robotika, seperti robot ASIMO buatan jepang. Sementara robot dalam air dan robot terbang lebih banyak dikembangkan sebagai peralatan untuk membantu penelitian yang berkaitan dan untuk proyek pertahanan atau mesin perang.

Dari kelompok non-mobile yang sering disebut sebagai“keluarga robot” adalah robot arm atau robot manipulator saja. Sementara yang lebih mudah dikenali sebagai mesin cerdas (intelligent machine) yang tidak selalu tampak memiliki bagian tangan, kaki atau roda untuk bergerak lebih lazim disebut dengan nama khusus sesuai fungsinya. Mereka biasanya memiliki nama-nama tersendiri. Misalnya mesin-mesin otomatis Lathe, milling, drilling machine, CNC (Computer Numerical Control) Machine, EDM (Electric Discharge Machine), dan berbagai peralatan ototmatis yang biasa dijumpai di pabrik-pabrik modern.




Gambar 1.1. Ilustrasi penelitian dalam domain robot

Mobile Robot adalah tipe robot yang paling popular dalam dunia penelitian robot. Sebutan ini biasa digunakan sebagai kata kunci utama untuk mencari rujukan atau referensi yang berkaitan dengan robotika di internet. Publikasi dengan judul yang berkaitan dengan mobile robot sering menjadi daya tarik, tidak hanya bagi kalangan peneliti, tetapi juga bagi kalangan awam. Dari segi manfaat, penelitian tentang berbagai tipe mobile robot diharapkan

dapat membantu manusia dalam melakukan otomasi dalam transportasi, platform bergerak untuk robot industri, eksplorasi tanpa awak, dan masih banyak lagi.

Fokus penelitian dapat diambil dengan titik berat perhatian lebih kepada kinematik atau dinamik atau kedua-duanya. Dari analisa kinematik saja, bila obyek penelitian yang diambil adalah konfigurasi robot yang benar-benar baru (belum ada penelitian sebelumnya yang mengkaji) kontribusi keilmuan dapat diperoleh hanya dengan mengkaji persamaan kinematik dan kontrol dasarnya. Dalam hal ini seringkali pembahasan yang mendalam secara matematik diperlukan. Beberapa hasil penelitian yang difokuskan pada pembahasan kinematik dapat dijumpai pada paper-paper Bayle, et al. (2002), D'Souza, et al. (2001), dan Tchon (2002).

Pembahasan khusus dalam hal dinamika robot juga sangat menjanjikan dalam perolehan kontribusi keilmuannya. Tujuan utama kajian dinamik ini adalah untuk mendapatkan disain kontrol yang lasak (robust) yang mampu meredam gangguan dengan baik. Masih banyak struktur-struktur robot yang kompleks belum dikaji secara mendalam model dinamiknya oleh karena rumitnya persoalan pemodelan matematik sistem robot, sifat-sifat alami (friksi pada poros aktuator, backlash pada gearbox, noise pada sensor, nonlinieritas dari pada aktuator, dsb.) dan lingkungan (gangguan luar berupa efek pembebanan, jalan yang tidak rata, getaran,dll.). Dari persamaan dinamik ini kontrol dasarnya dapat dirancang secara sistematis. Bahasan kontrol robot yang dimulai dari pemodelan robot secara penuh ini (kinematik dan dinamik) biasa disebut sebagai model-based control. Beberapa kajian yang sangat mendalam tentang dinamik robot dan kontrolnya dapat dujumpai pada paper-paper Hewit dan Burdess (1981), Arimoto (1984), Yamamoto dan Yun (1996), dan Godler, et al. (2002). Pada kasus dinamik robot yang rumit seringkali dibutuhkan bantuan kecerdasan buatan untuk mengidentifikasi model matematiknya. Lin dan Goldenberg(2001) menggunakan jaringan saraf tiruan (artificial neural network) untuk mengidentifikasi model dan kontrol yang sesuai untuk sebuah mobile manipulator. Sedangkan Sakka dan Chochron (2001) menggunakan algorithma genetic. Metoda sistem berbasis pengetahuan (knowledge-based system) juga dapat digunakan sebagai pilihan untuk menyelesaikan masalah ketidakpastian dalam pemodelan dinamik, seperti pada paper Pitowarno, et al. (2001).

Gabungan kontrol kinematik dan kontrol dinamik yang baik akan menghasilkan kontrol gerak robot (robot motion control) yang lasak. Hal ini adalah merupakan tujuan utama dalam rancang bangun robot ideal. Namun demikian, dewasa ini penelitian tentang aplikasi kecerdasan buatan dalam kontrol robot lebih banyak ditujukan untuk memperoleh kontrol kinematik yang canggih. Lebih-lebih kebutuhan akan metoda navigasi, pemetaan medan jelajah (path planning), kemampuan untuk menghindari halangan (obstacle avoidance), dan kemampuan untuk menghindari tabrakan sesama robot (collision) masih dianggap lebih utama daripada mengkaji kesempurnaan dan kepresisian gerak robot, kalau tidak dapat dikatakan bahwa kajian dinamik memang lebih rumit dibandingkan dengan kajian kinematik.

Kelompok no.4 dalam **Gambar 1.1** mengisyaratkan bahwa tujuan penelitian dengan titik berat pada analisis kinematik memang berbeda dengan tujuan penelitian dengan titik berat pada kajian dinamik. Kalau kedua goal ini dapat dikolaborasikan dengan baik maka tidak mustahil dalam waktu dekat para peneliti mampu menciptakan robot-robot mirip manusia yang mampu bekerja sama seperti mengangkat dan memindah barang, bermain bola dalam suatu kesebelasan, bahkan menjadi tentara.

### 1.3.1 Mekatronik vs Robotik

Mekatronik adalah istilah umum yang menjadi popular seiring dengan perkembangan padu mekanik dan elektronik. Mekatronik terdiri dari 4 disiplin ilmu, yaitu mekanik (mechanics), elektronik, teknik kontrol berbasis prosesor dan pemrograman seperti halnya pada bidang robotika. Sebuah produk mekatronik belum tentu robotika, namun robot adalah bagian dari mekatronik. Banyak produk mekatronik di sekeliling kita, misalkan mesin cuci, CD/DVD/video/cassette player, walkman, hingga vacuum cleaner. Di dalam dunia otomotif ada mobil yang dilengkapi dengan sistem parkir otomatis tanpa sopir, ABS (anti lock braking system), active suspension system, dsb. Dalam dunia industri, perdagangan dan gedung-gedung perkantoran dikenal berbagai peralatan otomatis seperti pintu otomatis, lift, eskalator, mesin fotokopi, dan masih banyak lagi.

Dengan berkembangnya ilmu di bidang control cerdas (intelligent control) maka dikenal pula istilah intelligent mechatronics yang dimaksudkan untuk mendeskripsikan produk mekatronik yang telah dimuati suatu kecerdasan buatan. Sebagai contoh, mesin cuci berbasis control fuzzy (fuzzy control washing machine), mesin penjual minuman otomatis yang dilengkapi sistem validasi uang menggunakan metoda jaringan saraf tiruan (artificial neural network), dll. Sistem printer, scanner dan fotokopi dalam satu alat juga termasuk dalam kategori ini.

Penelitian mutakhir dalam bidang mekatronik hampir tak dapat dipisahkan dengan penelitian di bidang robotika itu sendiri. Sebagai contoh, ultrasonic motor dan teknologi MEMS (micro electro mechanical system) yang dikembangkan untuk pembuatan sistem aktuator berukuran mikro atau lebih kecil dari 1 mm (Simokohbe, 2005). Meski penelitian ini masih sangat baru dan belum menunjukkan kemajuan berarti, namun jika berhasil, dipercayai manfaatnya sangat besar dalam aplikasi di dunia kedokteran dan rekayasa genetika.

### 1.3.2 Robotik vs Bio-Science

Dalam dekade terakhir ini penelitian bidang robotika dalam dunia kehidupan organik (bio science) juga semakin mendalam dan bahkan cenderung tak terduga arahnya. Jika dalam dunia kedokteran telah dikenal teknik kloning mahluk hidup yang kontroversial itu, maka dalam dunia robotika juga dikenal suatu proyek penelitian yang disebut sebagai implant sensor/aktuator atau impant interface. Interface berupa chip IC berukuran mikro ditanamkan ke dalam tubuh mahluk hidup dengan tujuan agar komputer di luar dapat mengendalikan dan atau memonitor kegiatan saraf organic manusia secara langsung di dalam pembuluh darah atau saraf tubuh.

Hasil-hasil awal penelitian ini sudah mulai dipublikasikan (Warwick 2005). Di dalam paper ini diterangkan tentang sebuah eksperimen pengendalian tikus agar berjalan sesuai dengan perintah dari computer. Sebuah chip telah ditanamkan di kepalanya.

## 1.4 Jenis Robot

Secara umum, jenis robot dapat dibedakan dalam 4 kategori, yaitu :

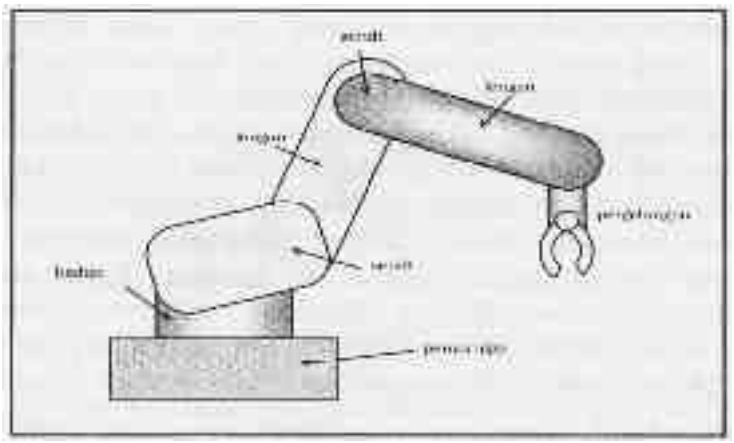
### 1.4.1 Non-mobile Robot

Robot ini tidak dapat berpindah posisi dari satu tempat ke tempat lainnya, sehingga robot tersebut hanya dapat menggerakkan beberapa bagian dari tubuhnya dengan fungsi tertentu yang telah dirancang.

Contoh :

***Robot Industri***

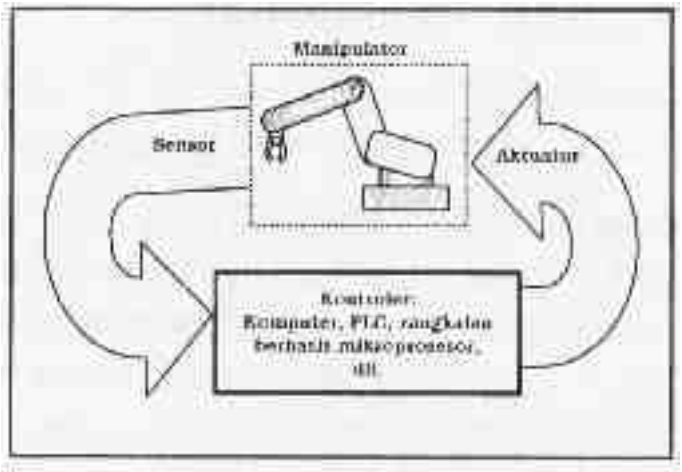
Anatomi robot industri secara umum dapat diilustrasikan seperti pada **gambar 1.2**. Robot industri yang diilustrasikan ini adalah robot tangan yang memiliki dua lengan (dilihat dari persendian), dan pergelangan. Di ujung pergelangan dapat diinstal berbagai tool sesuai dengan fungsi yang diharapkan. Jika dipandang dari sudut pergerakan maka terdiri dari tiga pergerakan utama, yaitu badan robot yang dapat berputar ke kiri dan kanan, lengan yang masing-masing dapat bergerak rotasi ke arah atas dan bawah, dan gerak pergelangan sesuai dengan sifat tool.



Gambar 1.2. Anatomi robot industri

Perangkat pendukung robot industri secara umum dapat diilustrasikan dalam **gambar 1.3** berikut ini. Komponen utamanya terdiri dari 4 bagian, yaitu:

- Manipulator
- Sensor
- Aktuator, dan
- Kontroler




Gambar 1.3. Sistem robot industri

Manipulator adalah bagian mekanik yang dapat difungsikan untuk memindah, mengangkat, dan memanipulasi benda kerja. Sensor adalah komponen berbasis instrumentasi (pengukuran) yang berfungsi sebagai pemberi informasi tentang berbagai keadaan atau kedudukan dari bagian-bagian manipulator. Output sensor dapat berupa nilai logika ataupun nilai analog. Dalam berbagai kasus dewasa ini penggunaan kamera sebagai sensor sudah menjadi lazim. Output perangkat kamera berupa citra (image) harus diubah dahulu ke besaran digital ataupun analog sesuai dengan kebutuhan. Kajian teknologi tranformasi image ke bentuk biner (nilai acuan dalam proses perhitungan komputer) ini banyak di kaji dalam konteks terpisah, yaitu pengolahan citra (image processing).

Aktuator adalah komponen bergerak yang jika dilihat dari prinsip penghasil geraknya dapat di bagi menjadi 3 bagian, yaitu penggerak berbasis motor listrik (motor DC servo, stepper moto, motor AC, dsb.), penggerak

pneumatik (berbasis kompresi gas: udara, nitrogen, dsb.), dan penggerak hidrolik (berbasis kompresi benda cair:minak pelumas, dsb.).

Kontroler adalah rangkaian elektronik berbasis mikroprosesor yang berfungsi sebagai pengatur seluruh komponen dalam membentuk fungsi kerja. Tipe pengaturan yang bisa diprogramkan mulai dari prinsip pengurut (sequencer) yang bekerja sebagai open loop hingga prinsip umpan balik yang melibatkan kecerdasan buatan.

Gambar 1.4. Gambar Robot Manipulator

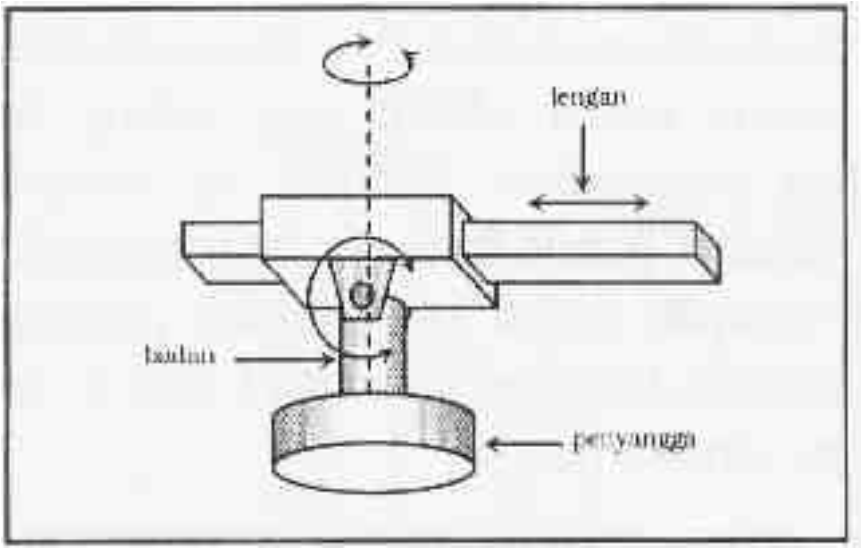
- **Konfigurasi Manipulator**

  Secara klasik konfigurasi robot manipulator dapat dibagi dalam 4 kelompok, yaitu polar, silindris, cartesian dan sendi-lengan (joint-arm).

  1. Polar

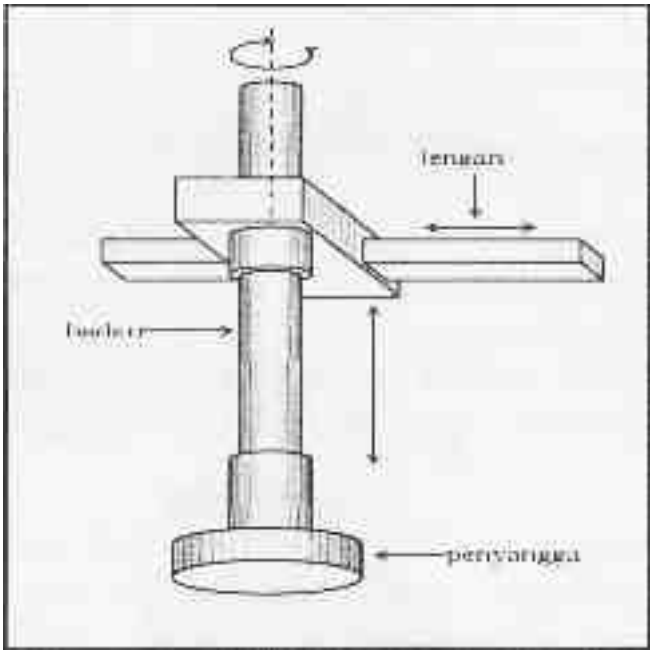
     Manipulator yang memiliki konfigurasi polar padat di ilustrasikan seperti pada gambar 1.5., badan dapat berputar ke kiri atau kanan. Sendi pada badan dapat mengangkat atau menurunkan pangkal lengan secara polar. Lengan ujung dapat digerakkan maju-mundur secara translasi. Konfigurasi ini dikenal cukup kokoh karena sambungan lengan dan gerakan maju-mundur memiliki cara yang secara mekanik sangat kokoh. Kemampuan jangkauan ke atas dan bawah kurang bagus karena badan tidak mengangkat lengan secara

vertikal, namun memiliki gerakan yang khas yaitu mampu memanipulasi ruang kerja yang berbentuk bola dengan algoritma gerak yang paling sederhana dibanding tipe konfigurasi yang lain.



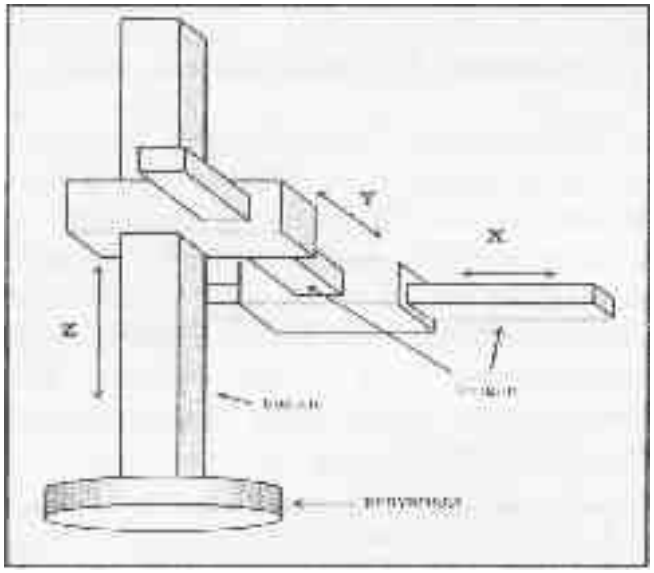

Gambar 1.5. Konfigurasi polar

2. Silinder
   Konfigurasi silinder mempunyai jangkauan berbentuk ruang silinder yang lebih baik, meskipus sudut lengan terhadap garis penyangga tetap. Konfigurasi ini banyak diadopsi untuk sistem gantry atau crane karena strukturnya yang kokoh untuk tugas mengangkat beban. Pemasangan lengan ujung yang segaris dengan badan dapat lebih menguntungkan kinematiknya menjadi lebih sederhana. Selain itu struktur secara keseluruhan bisa lebih kokoh. Contoh yang mudah dijumpai adalah sistem crane yang biasa digunakan dalam pembangunan gedung-gedung bertingkat tinggi.




Gambar 1.6. Konfigurasi silinder

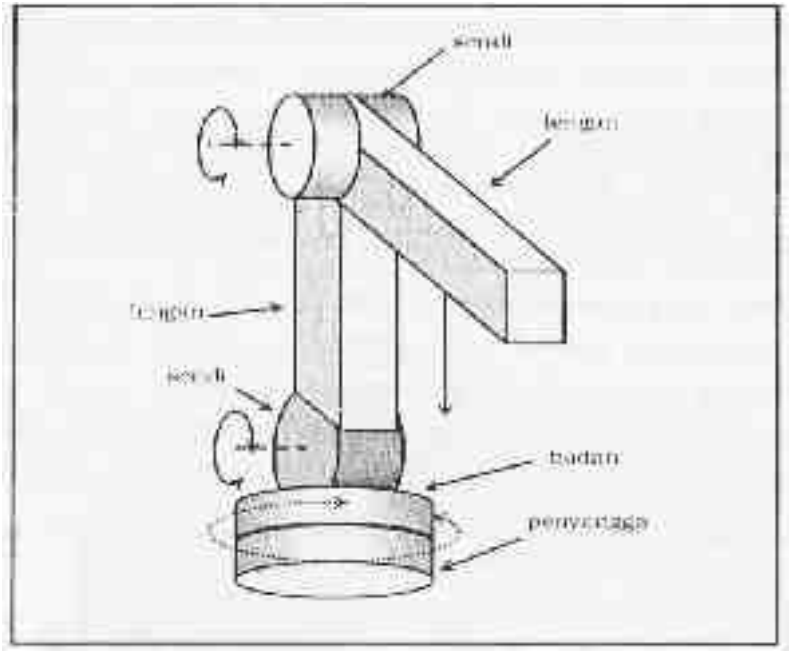
3. Cartesian
   Manipulator berkonfigurasi Cartesian ditunjukkan dalam gambar 1.7. Konfigurasi ini secara relatif adalah yang paling kokoh untuk tugas mengangkat beban yang berat. Struktur ini banyak dipakai secara permanen pada instalasi pabrik, baik untuk mengangkat dan memindah barang produksi maupun untuk mengangkat peralatan-peralatan berat pabrik ketika melakukan kegiatan instalasi. Crane di galangan kapal juga banyak mengadopsi struktur ini.




Gambar 1.7. Konfigurasi Cartesian

Pada aplikasi yang sesungguhnya, biasanya struktur penyangga, badan dan lengan dibuat sedemikian rupa hingga tumpuan beban merata pada struktur. Misalnya, penyanggah dipasang dari ujung ke ujung. Mekanik pengangkat di badan menggunakan sistem rantai dan sprocket atau sistem belt. Pergerakan lengan dapat menggunakan sistem seperti rel di kiri-kanan lengan.

4. Sendi-lengan
   Konstruksi ini yang paling popular untuk tugas-tugas regular di dalam pabrik, terutama untuk dapat melaksanakan fungsi layaknya pekerja pabrik, seperti mengangkat barang dari konveyor, mengelas, memasang komponen mur, baut pada produk, dan sebagainya. Dengan tool pergelangan yang khusus struktur lengan-sendi ini cocok digunakan untuk menjangkau daerah kerja yang sempit dengan sudut jangkauan yang beragam.




Gambar 1.8. Konfigurasi sendi-lengan

### 1.4.1 Mobile Robot

Mobile dapat diartikan bergerak, sehingga robot ini dapat memindahkan dirinya dari satu tempat ke tempat lain. Robot ini merupakan robot yang paling populer dalam dunia penelitian robotika. Dari segi manfaat, robot ini diharapkan dapat membantu manusia dalam melakukan otomasi dalam transportasi, platform bergerak untuk robot industri , eksplorasi tanpa awak, dan masih banyak lagi.

Contoh :

- ***Robot Line Tracker***

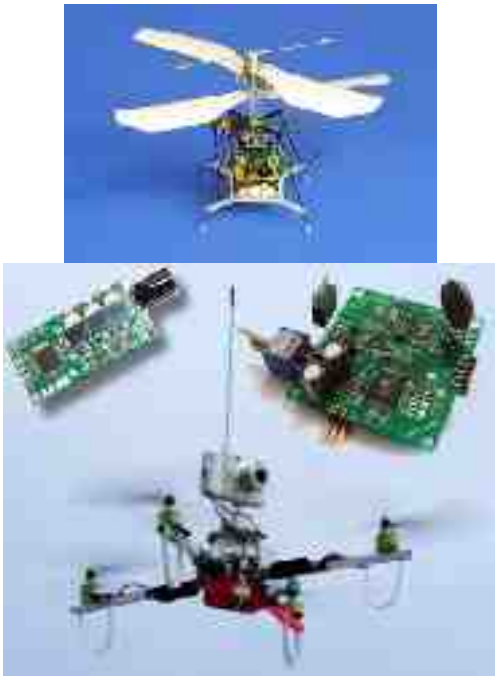
Robot line tracker merupakan robot yang dapat bergerak mengikuti track berupa garis hitam setebal ±3 cm. Untuk membaca garis, robot dilengkapi dengan sensor proximity yang dapat membedakan antara garis hitam dengan lantai putih. Sensor proximity ini dapat dikalibrasi untuk menyesuaikan

pembacaan sensor terhadap kondisi pencahayaan ruangan. Sehingga pembacaan sensor selalu akurat.

Agar pergerakan robot menjadi lebih halus, maka kecepatan robot diatur sesuai dengan kondisi pembacaan sensor proximity. Jika posisi robot menyimpang dari garis, maka robot akan melambat. Namun jika robot tepat berada diatas garis, maka robot akan bergerak cepat. Robot juga dapat kembali ke garis pada saat robot terlepas sama sekali dari garis. Hal ini bisa dilakukan karena robot selalu mengingat kondisi terakhir pembacaan sensor. Jika terakhir kondisinya adalah disebelah kiri garis, maka robot akan bergerak ke kanan, demikian pula sebaliknya.

- **Flying Robot (Robot Terbang)**

Robot yang mampu terbang, robot ini menyerupai pesawat model yang diprogram khusus untuk memonitor keadaan di tanah dari atas, dan juga untuk meneruskan komunikasi.



Gambar 1.9 Contoh Flying Robot (Robot Terbang)

- **Under Water Robot (Robot dalam air)**

Robot ini digunakan di bawah laut untuk memonitor kondisi bawah laut dan juga untuk mengambil sesuatu di bawah laut.

Gambar 1.10 Contoh Under Water Robot (Robot Dalam Air)

## 1.4.2 Kombinasi Mobile dan Non-Mobile Robot

Robot ini merupakan penggabungan dari fungsi-fungsi pada robot mobile dan non-mobile. Sehingga keduanya saling melengkapi dimana robot nonmobile dapat terbantu fungsinya dengan bergerak dari satu tempat ke tempat yang lain.

Gambar 1.11 Contoh robot kombinasi Mobile dan Non-mobile

### 1.4.3 Humanoid

Sebuah robot humanoid adalah robot otonom yang dapat beradaptasi dengan perubahan lingkungan atau dirinya sendiri. Ini merupakan perbedaan utama antara jenis humanoid dan jenis robot.

Dalam konteks, robot humanoid dapat mencakup, antara lain:

- Dapat merawat dirinya sendiri (seperti pengisian sumber tenaga sendiri)
- Dapat belajar otonom (belajar atau memiliki kemampuan baru tanpa bantuan dari luar (manusia), menyesuaikan diri berdasarkan lingkungan dan beradaptasi dengan lingkunganyang baru)
- Dapat menghindari hal-hal yang berbahaya bagi manusia, properti, dan dirinya sendiri
- Dapat berinteraksi dengan manusia dan lingkungan

Seperti robot mekanis lainnya, humanoid mengacu pada komponen dasar sebagai berikut : Sensing (Penginderaan), Actuating, Planning (Perencanaan) dan Controling (Pengendalian). Karena untuk mensimulasikan struktur, perilaku manusia dan sistem otonomi, sebagian besar robot humanoid lebih kompleks dibandingkan jenis robot lainnya.

Kompleksitas ini mempengaruhi semua skala robot (mekanik, ruang, waktu, sistem dan kompleksitas komputasi), tetapi lebih terlihat pada densitas daya dan skala kompleksitas sistem. Hal pertama, robot humanoid tidak cukup kuat bahkan untuk melompat dan ini terjadi karena kekuatan atau perbandingan berat tidak sebaik seperti tubuh manusia. Ada algoritma yang sangat baik untuk beberapa bidang konstruksi robot humanoid, tapi sangat sulit untuk menggabungkan semuanya menjadi satu sistem yang efisien (sistem kompleksitas sangat tinggi).


Gambar 1.12 TOSY TOPIO , robot humanoid yang dapat main ping pong

Robot humanoid diciptakan untuk meniru beberapa tugas fisik dan mental yang sama seperti manusia menjalani kehidupan setiap harinya. Para ilmuwan dan spesialis dari berbagai bidang termasuk teknik , ilmu kognitif , dan linguistik menggabungkan upaya mereka untuk menciptakan robot yang mirip dengan manusia. Tujuan ilmuwan dan spesialis menciptakan robot humanoid adalah agar robot humanoid dapat memahami kecerdasan akal manusia dan bertindak layaknya seperti manusia. Jika robot humanoid mampu melakukannya, mereka akhirnya bisa bekerja dalam kohesi dengan manusia untuk menciptakan masa depan yang lebih produktif dan berkualitas tinggi. Manfaat lain yang penting untuk mengembangkan robot humanoid adalah untuk memahami tubuh manusia biologis dan proses mental, dari yang sederhana hingga yang berjalan dengan konsep kesadaran dan spiritualitas.

Dalam perencanaan dan pengendalian antara robot humanoid dengan robot jenis lain (seperti robot industri) memiliki perbedaan yaitu bahwa gerakan robot harus menyerupai manusia, dengan menggunakan penggerak berkaki, terutama biped kiprah. Perencanaan ideal untuk gerakan robot humanoid saat berjalan normal harus menghasilkan konsumsi energi minimum, seperti seperti halnya tubuh manusia. Untuk alasan ini, studi tentang dinamika dan kontrol dari jenis struktur menjadi lebih penting.

Untuk menjaga keseimbangan dinamis selama berjalan, robot membutuhkan informasi tentang gaya kontak saat ini dan gerakannya yang diinginkan. Solusi untuk masalah ini bergantung pada konsep utama, Zero Moment Point (ZMP).

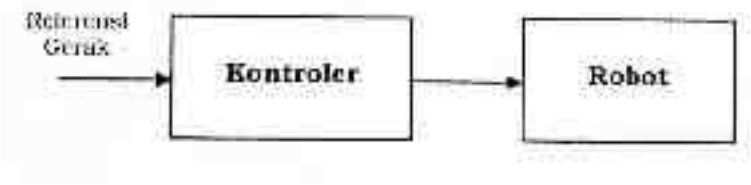
Karakteristik lain tentang robot humanoid adalah bahwa mereka bergerak, mengumpulkan informasi (menggunakan sensor) pada "dunia nyata" dan berinteraksi dengan itu, mereka tidak tinggal tetap seperti manipulator pabrik dan robot lain yang bekerja di lingkungan yang sangat terstruktur. Perencanaan dan Pengendalian harus fokus tentang deteksi self-collision, perencanaan jalur dan penghindaran rintangan untuk memungkinkan humanoids untuk bergerak dalam lingkungan yang kompleks.

Ada fitur dalam tubuh manusia yang belum dapat ditemukan di robot humanoid. Mereka mencakup struktur dengan fleksibilitas variabel, yang memberikan keselamatan (untuk robot itu sendiri dan kepada orang-orang), dan redundansi gerakan, yaitu lebih derajat kebebasan dan karena itu ketersediaan tugas lebar. Meskipun karakteristik ini diinginkan untuk robot humanoid, mereka akan membawa kerumitan yang lebih dan masalah baru untuk perencanaan dan kontrol.

## 1.5 Sistem Kontrol Robotik

Sistem kontrol robotik pada dasarnya terbagi dua kleompok, yaitu sistem kontrol loop terbuka(open loop) dan loop tertutup (close loop).

Diagram kontrol loop terbuka pada sistem robot dapat dinyatakan dalam gambar 1.13 berikut ini.

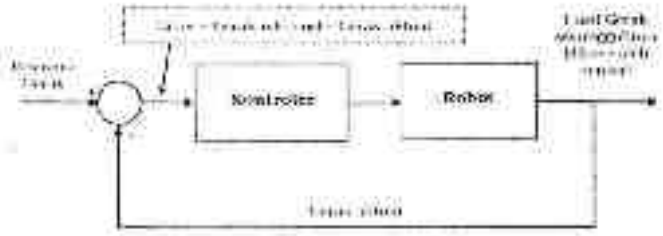



*Gambar 1.13 Kontrol robot loop terbuka*

Kontrol loop terbuka atau umpan maju ( feedforward control) dapat dinyatakan sebgai sistem kontrol yang outputnya tidak diperhitungkan ulang oleh kontroler.Keadaan apakah robot benar-benar telah mencapai target seperti yang dikehendaki sesuai referensi, adalah tidak dapat mempengaruhi kinerja kontroler. Kontrol ini sesuai untuk sistem operasi robot yang memiliki akuator yang beroperasi berdasarkan umpan logika berbasis konfigurasi langkah sesuai urutan, misalnya stepper motor. Stepper motor tidak perlu dipasangi sensor pada porosnya untukmengetahui posisi akhir. Jika dalam keadaan berfungsi dengan baik dan tidak ada masalah beban lebih maka stepper motor akan berputar sesuai dengan perintah kontroler dan mencapai posisi target dengan tepat.

Perlu di garis bawahi disini bahwa kontrol sekuensi (urutan) dalam gerak robot dalam suatu tugas yang lengkap, misalnya memiliki urutan sebagai berikut: menuju ke posisi obyek, mengankat obyekmemindah obyek ke posisi tertentu, dan meletakkan obyek adalah tidak selalu semua langkah operasi ini termasuk dalam kontrol loop terbuka. Dapat saja langkah menuju posis obyek dan memindah obyek menuju posisi akhir adalah gerakgerak berdasarkan loop tertutup. Sedangkan yang lainnya adalah loop terbuka berdasarkan perintah langkah berbasis delay.

Kontrol robot loop tertutup dapat dinyatakan seperti dalam Gambar 1.14.




*Gambar 1.14 Kontrol robot loop tertutup*

Pada gambar di atas, jika hasil gerak aktual telah sama dengan referensi maka input kontroler akan nol. Artinya kontroler tidak lagi memberikan sinyal akurasi kepada robot karena target akhir perintah gerak telah diperoleh. Makin kecil error terhitung maka makin kecil pula sinyal pengemudian kontroler terhadap robot. Sampai akhirnya mencapai kondisi tenang(steady state).

Referensi gerak dan gerak aktual dapat berupa posisi(biasanya didefinisikan melalui kedudukan ujung lengan terakhir/end of effector), kecepatan, akselerasi, atau gabungan di antaranya. Kontrol bersifat konvergen jika dalam rentang waktu pengontrollan nilai error menuju nol, dan keadaan dikatakan stabil jika setelah konvergen kotroler mampu menjaga agar error selalu nol. Dua pengertian dasar; konvergen dan stabil, adalah sangat penting dalam kontrol loop tertutup. Stabil dan konvergen diukur dari sifat referensinya. Posisi akhir dianggap konvergen bila makin lama gerakan makin perlahan dan akhirnya diam pada posisi seperti yang dikehendaki referensi, dan dikatakan stabil jika posisi akhir yang diam ini dapat dipertahankan dalam masa-masa berikutnya. Jika referensinya adalah kecepatan maka disebut stabil jika pada keadaan tenang kecepatan akhirnya adalah sama dengan referensi (atau mendekati) dan kontroler mampu menjaga ‘ kesamaan’ ini pada masa-masa berikutnya. Dalam hal kecepatan, keadaan tenang yang dimaksud adalah bukan berarti output kontroler bernilai nol(tegangan nol Volt) seperti keadaan sesungguhnya pada kontrol posisi, namun kontroler tidak lagi memberikan penguatan (amplify) atau pelemahan (attenuate) pada akuator. Demikian juga bila referensinya adalah percepatan (akselerasi). Pembahasan yang lebih detil

tentang hal ini diterangkan melalui contoh-contoh praktis di Bab-bab berikutnya.

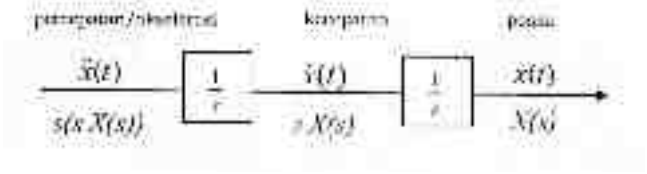
### 1.5.1 Sekilas tentang penggunaan Transformasi Laplace

Mendalami kontrol robotik secara teoritis memerlukan dasar-dasar pemahaman tentang sistem sinyal. Semua gerakan yang diasumsikan sebagai visualisasi operasi robot berbasis waktu yang berjalan dapat dinyatakan sebagai fungsi sinyal. Artinya , karakteristik input(referensi), sistem kontroler . sistem dikontrol, dan output dapat dinyatakan dalam persamaan matematik yang merepresentasikan sifat atau respon terhadap perubahan waktu.

Transformasi Laplace adalah salah satu metoda untuk menyatakan persamaan sinyal dalam fungsi waktu . Metoda ini sangat berguna dalam analisa sinyal untuk kontrol robotik selain metodea baku yang lain (representasi Fourier, dan transformasi *Z*). Bentuk dasar ekspresi matematikanya adalah sebagai berikut.

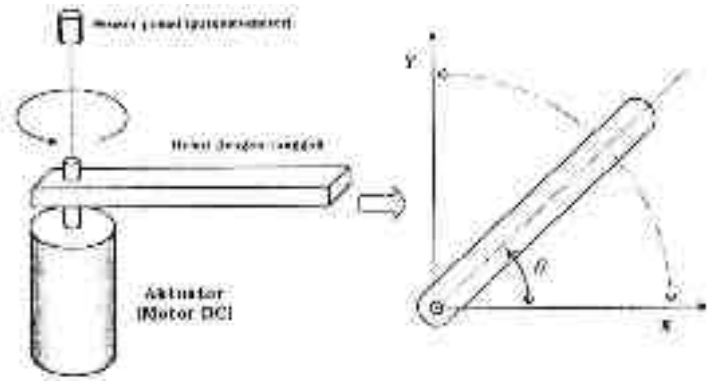
$$L\{f(t)\} = \int_0^\infty f(t)e^{-st}dt \tag{1.1}$$

Dengan pendekatan ini jika $L\{x(t)\} = X(s)$ dengan asumsi nila pada kondisi awal adalah nol, maka $L\{x(t)=sX(s)$. Demikian juga maka $L\{\dot{x}(t)\} = s(sX(s))$. Jika $x(t)$ adalah fungsi dari posisi , $\dot{x}(t)$ adalah kecepatan , dan $\ddot{x}(t)$ adalah percepatan maka peggunaan transformais Laplace untuk menyatakan hubungan ini dapat diilustrasikan seperti dalam Gambar 1.16 berikut ini.



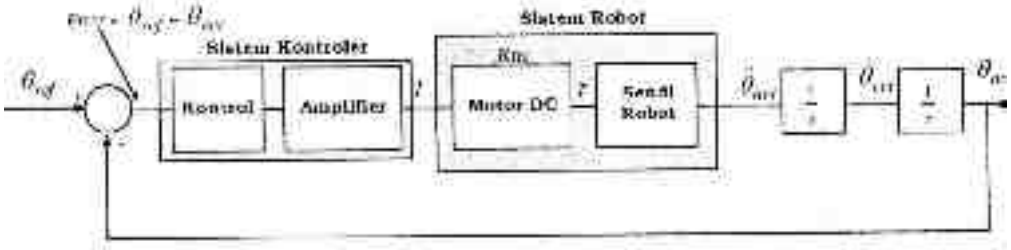
*Gambar 1.15 Penggunaan transformasi Laplace*

Sebagai contoh kita bahas sebuah robot lengan yang memiliki lengan tunggal atau satu sendi. Akuatornya adalah sebuah motor DC sedang sensornya adalah potensiometer. Illustrasinya diberikan dalam Gambar 1.17 berikut ini.




*Gambar 1.16 Robot Tangan Satu Sendi*

Robot diatas dapat dikontrol berdasarkan posisi, kecepatan dan percepatan. Di sebelah kanan gerakan lengan diilustrasikan dapat membentuk gerakan searah dan berlawanan dengan jarum jam. Posisi gerakan dinyatakan oleh $\theta$. Diagram kontrolnya dapat dinyatakan seperti gambar 1.17 berikut ini.




*Gambar 1.17 Diagram Kontrol Robot Tangan Satu Sendi*

Dalam gambar, $\theta_{ref}$ adalah posisi referensi (dalam radian), $\theta_{ref}$ adalah posisi aktul, *I* adalah arus motor . *Ktn* adalah konstanta motor, τ adalah torsi yang dihasilkan poros motor , $\ddot{\theta}_{act}$ adalah percepatan sudut aktual , $\dot{\theta}_{act}$ adalah kecepatan sudut aktual, dan $\theta_{act}$ adalah posisi aktual.

Gambar 1.17 diatas sebenarnya tidak berbeda dengan ilustrasi pada gambar 1.14. Blok kontroler pada gambar 1.14 dinyatakan sebgai sitem kontroler pada Gambar 1.17 dapat memberikan penjelasan yang lebih rinci tentang fenomena kontrol yang sesungguhnyaterjadi. Pemberian torsi oleh motor pada lengan robot memberikan dampak dinamik pada percepatan sudut, kecepatan sudut dan posisi ujung lengan robot sekaligus.

Dalam aplikasi yang sebenarnya pernyataan pada Gambar 1.17 memberika pengertian bahwa output dari sistem robot (putaran sudut pada poros atau sendi lengan dapat dibaca dalam tiga parameter , yaitu percepatan, kecepatan dan posisi. Namun tiga parameter ini tidak selalu diakomodasikan dengan pemasngan sensor yang bersesuaian. Jika sensor yang tersedia adlah sensor posisi saja maka kecepatan dapat diperoleh dengan mengintegrasi bacaan posisi terhadap waktu sebagai berikut.

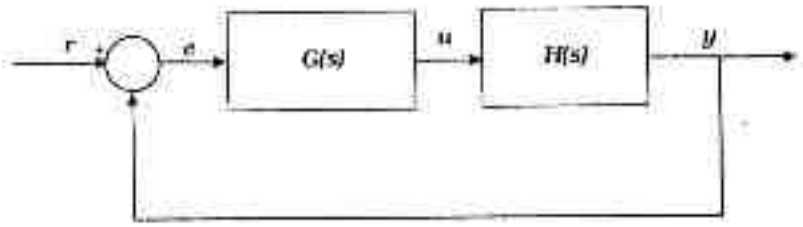
$$\dot{\theta}_{rata} = \frac{\Delta\theta}{\Delta t} \tag{1.2}$$

Demikian juga, percepatan dapat diperoleh denagn mengintegrasi bacaan atau perhitungan kecepatan terhadap waktu,

$$\ddot{\theta}_{rata} = \frac{\Delta\dot{\theta}}{\Delta t} \tag{1.3}$$

### 1.5.2 Kontrol Proporsional, Integral dan Derivatif

Kembali pada gambar 1.14. Sekarang akan kita gambar ulang dalam bentuk pernyataan standar dalam sistem kontrol seperti dalam gambar 1.18. dalam gambar, *r* adalah input, *e* adalah error , *u* adalah sinyal output kontroler, *G(s)* adalah kontroler, *H(s)* adalah dinamik robot, dan *y* adalah output. Sekarang permasalahannya adalah bagaimana *G(s)* didesain.




*Gambar 1.18 Kontrol robot loop tertutup*

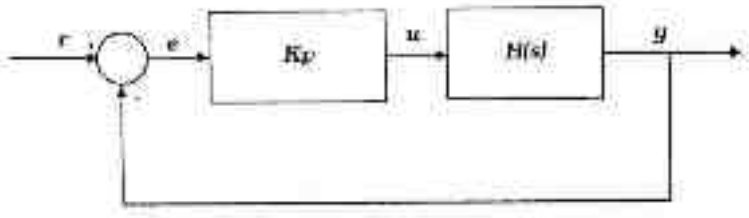
**Kontrol Proporsional (Proportional control, P)**

Kontroler adalah kontrol P jika $G(s) = k$ , denga $k$ adalah konstanta .

Jika, $u=G(s).e$ maka

$u= K_p.e$

dengan $K_p$ adalah kontanta proporsional. $K_p$ berlaku sebagai Gain (penguat) saja tanpa memberikan efek dinamik kepada kinerja kontroler. Dengan demikian gambar 1.18 dapat dinyatakan ulang sebagai berikut,




*Gambar 1.19 Kontrol Proporsional, P*

Penggunaan kontrol p memiliki berbagai keterbatasan karena sifat kontrol yang tidak dinamik ini. Walaupun demikian dalam aplikasi-aplikasi dasar yang sederhana kontrol P ini cukup mampu untuk mencapai konvergensi meskipun error keadaan tenangnya (steady-state error) relatif besar. Sebagai materi pembelajaran, kontrol P dianggap sangat baik untuk permulaaan.

**Kontrol Integral ( Integral Control, I)**

Jika *G(s)* adalah kontrol *I* maka *u* dapat dinyatakan sebagai

$u(t) = \left[ \int_0^t e(T)dT \right] Ki$ , *Ki* adalah konstanta integral

(1.5)

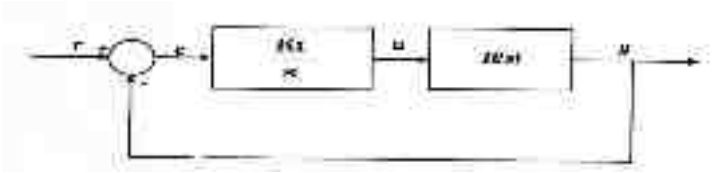
Dari persamaan (1.5) , *G(s)* dapat dinyatakan sebagai,

$$G(s) = \frac{Ki}{s}$$

Jika *e(T)* mendekati konstan (bukan nol maka *u(t)* kan menjadi sangat besar sehingga diharapkan dapat memperbaiki error . Jika *e(T)* mendekati nol maka efek kontrol I ini semakin kecil.

Kontrol I dapat memperbaiki respon steady-state. Namun pemilihan *Ki* yang tidak tepat dapat menyebabkan respon transien (transient response) yang tinggi sehingga dapat menyebabkan ketidakstabilan sistem. Pemilihan *Ki* yang sangat tinggi justru dapat menyebabkan output berosilasi.

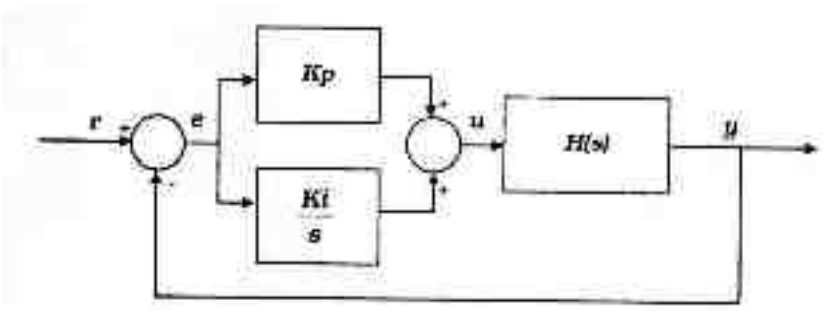
Diagram kontrol I dapat diilustrasikan sebagai berikut,



*Gambar 1.20 Kontrol Integral,I*

**Kombinasi kontrol P dan I**

Dengan sifat dasar kontrol P yang cenderung konvergen dan I yang dapat memperbaiki respon steady-state maka kombinasi P-Idapat memberikan hasil yang lebih baik. Dalam diagram blok dapt dinyatakan sebagai berikut.




*Gambar 1.21 Kontrol Proporsional-Integral, P-I*

Dari gambar 1.21 di atas , persamaan kontroler G(s) dapat dinyatakan sebgai berikut,

$$G(s) = Kp + \frac{Ki}{s} \tag{1.7}$$

atau

$$G(s) = \frac{sKp + Ki}{s} \tag{1.8}$$

**Kontrol Derivatif (Derivative Control , D)**

Sinyal kontrol u yang dihasilkan oleh kontrol D dapat dinyatakan sebagai,

$$u = Kd \cdot \dot{e} \tag{1.9}$$

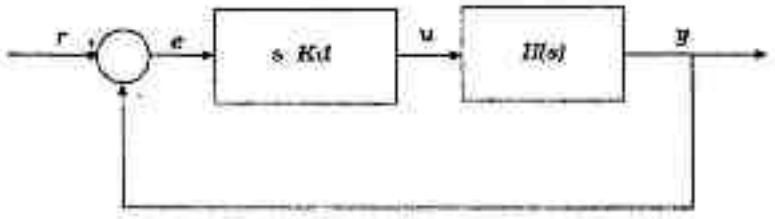
atau

$$u = Kd \cdot \frac{\Delta e}{\Delta t} \tag{1.10}$$

Sehingga *G(s)* dapat dinyatakan,

$$G(s) = s \cdot Kd \tag{1.11}$$

Dari persamaan (1.9) , nampak bahwa sifat dan kontrol D ini bermain dalam konteks"kecepatan" atau rate dari error. Dengan sifat ini ia dapat digunakan untuk memprediksi error yang akan terjadi . umpan balik yang diberikan adalah sebanding dengan kecepatan perubahan *e(t)* sehingga kontroler dapat mengantisipasi error yang akan terjadi. Dalam blok diagram dapat dinyatakan sebagai berikut,



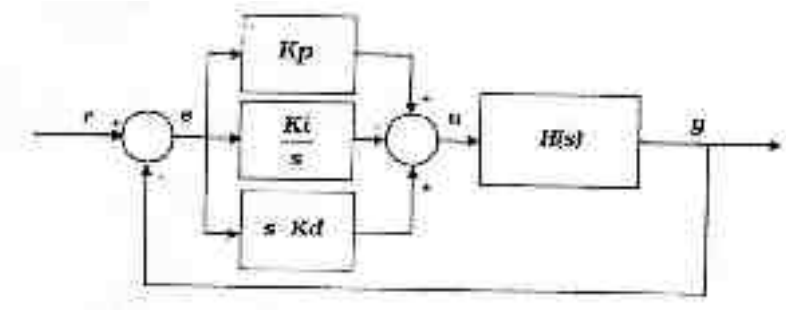

*Gambar 1.22 Kontrol derivatif, D*

**Kombinasi kontrol P,I dan D**

Diagram kombinasi ketiga kontrol klasik yang diterangkan di atas dapat dinyatakan seperti dalam gambar 1.23. Dengan menggabungkan kontrol P,I dan D maka masing-masing kelebihannya dapat disatukan untuk mendapatkan kontrol yang ideal.

Namun demikian, suatu sistem kontrol klasik kombinasi , baik PI ataupun PID, hanya dapat bekerja baik untuk sistem *H(s)* yang cenderung linier dalam

fungsi waktu. Artinya , persamaan dinamik dari model *H(s)* yang cenderung linier dalam fungsi waktu. Artinya , persamaan dinamik dari model *H(s)* relatif tidak berubah selama rentang waktu pengontrolan. Padahal kenyataannya , tidak ada fenomena sistem riil yang benar-benar linier . bahkan hampir semua fenomena kontrol mulai dari skala , misalnya kontrol motor DC, hingga skala sistem besar , misalnya kontrol pesawat terbang tanpa awak , jika dilakukan permodelan secara rinci dan lengkap adalah sangat tidak liner ( non liniear). Setiap sistem riil selalu berhadapan dengan gangguan (disturbance). Motor selalu bermasalah dengan friksi pada poros , gerabox , perubahan karakteristik karena temperatur, dll. Pesawat diudara selalu berhadapan dengan tekanan udara yang berubah-ubah,angin,hujan , dsb




*Gambar 1.23 Kontrol PID*

Untuk kontrol klasik ini , yang dapat dilakukan oleh engineer hanyalah melakukan pendekatan atau asumsi model sistem secara linier dengan mengabaikan faktor-faktor nonlinier yang dianggap terlalu sulit untuk dimodelkan secara matematik. Sehingga Kp, Ki dan Kd yang dipilih (tuned) adalah yang dianggap paling tepat (optimum) untuk kondisi ideal model.

## 1.6 Penggunaan Kontrol Cerdas

Kontrol Cerdas (Intelligent Control) adalah sistem kontrol yang berdasarkan algoritma yang dipandang cerdas. Kata Intelligent control telah menjadi baku dalam dunia kontrol seetelah berbagai teori dan algoritma pemrogramman yang dapat meniru “kecerdasan manusia” berhasil dikaji

dengan baik. Dalam konteks ini kemudian muncul istilah kecerdasan buatan. Kecerdasan buatan dalam robotik adalah suatu algoritma (yang dipandang) cerdas yang diprogramkan ke dalam kontroler robot. Pengertian cerdas di sini sangat relatif, karena tergantung dari sisi mana seseorang memandang.

Para filsuf diketahui telah memulai ribuan tahun yang lalu mencoba untuk memahami dua pertanyaan mendasar: bagaimanakah piikiran manusia itu bekerja, dan, dapatkah yang bukan-manusia itu berpikir? (Negnevitsky, 2004). Hingga sekarang, tak satupun orang mampu menjawab dengan tepat dua pertanyaan ini. Pernyataan cerdas yang pada dasarnya digunakan untuk mengukur kemampuan berpikir manusia selalu menjadi perbincangan menarik karena yang melakukan penilaian cerdas atau tidak adalah juga manusia. Sementara itu, manusia tetap bercita-cita untuk menularkan "kecerdasan manusia" kepada mesin.

Dalam literatur, orang pertama yang dianggap sebagai pionir dalam mengembangkan mesin cerdas (intelligence machine) adalah Alan Turing, seorang matematikawan asal Inggris yang memulai karir saintifiknya di awal tahun 1930-an. Di tahun 1937 ia menulis paper tentang konsep mesin universal (universal machine). Kemudian, selama perang dunia ke-2 ia dikenal sebagai pemain kunci dalam penciptaan Enigma, sebuah mesin encoding milik militer Jerman, setelah perang, Turing membuat "automatic computing engine". Ia dikenal juga sebagai pencipta pertama program komputer untuk bermain catur, yang kemudian program ini dikembangkan dan dimainkan di komputer Manchester University. Karya-karyanya ini, yang kemudian dikenal sebagai Turing Machine, dewasa ini masih dapat ditemukan aplikasi-aplikasinya. Beberapa tulisannya yang berkaitan dengan prediksi perkembangan komputer di masa datang akhirnya juga ada yang terbukti. Misalnya tentang ramalanya bahwa di tahun 2000-an komputer akan mampu melakukan percakapan dengan manusia. Meski tidak ditemukan dalam paper-paper tentang istilah "resmi": Artificial Intelligence, namun para peneliti di bidang ini sepakat untuk menobatkan Turing sebagai orang pertama yang mengembangkan kecerdasan buatan.

Secara saintifik, istilah kecerdasan buatan – untuk selanjutnya disebut AI ( artificial Intelligence) – pertama kali diperkenalkan oleh Warren McCulloch, seorang filsuf dan ahli perobatan dari Columbia University , dan Warren Pitts, seorang matematikawan muda tahun 1943, (Negnevitsky, 2004). Mereka mengajukan suatu teori tentang jaringan saraf tiruan (artificial nueral network, ANN) – untuk selanjutnya disebut sebagia ANN – bahwa setiap nuron dapat dipostulasikan dalam dua keadaan biner , yaitu ON dan OFF. Mereka mencoba menstimulasikan model neuron ini secara teori dan eksperimen di Laboratorium. Dari percobaan, telah didemonstrasikan bahwa model jaringan syaraf yang mereka ajukan mempunyai kemiripan dengan mesin Turing, dan setiap fungsi perhitungan dapat diselesaikan melalui jaringan neuron yang mereka modelkan.

Kendati mereka meraih sukses dalam pembuktian aplikasinya, pada akhirnya melalui eksperimen lanjut diketahui bahwa model ON-OFF pada ANN yang mereka ajukan adalah kurang tepat. Kenyataannya, neuron memiliki karakteristik yang sangat nonlinier yang tidak hanya memiliki keadaan ON-OFF saja dalam aktifitasnya. Walau demikian, McCulloch akhirnya dikenal sebagai orang kedua setelah Turing yang gigih mendalami bidang kecerdasan buatan dan rekayasa mesin cerdas. Perkembangan ANN sempat mengalami masa redup pada tahun 1970-an. Baru kemudian pada pertengahan 1980-an ide ini kembali banyak dikaji oleh para peneliti.

Sementara itu, metoda lain dalam AI yang sama terkenalnya dengan ANN adalah Fuzzy Logic (FL) – untuk selanjutnya ditulis FL. Kalau ANN didisain berdasarkan kajian cara otak biologis manusiabekerja (dari dalam), maka FL justru merupakan representasi cara berfikir manusia yang nampak dari sisi luar. Jika ANN dibuat berdasarkan model biologis teoritis , maka FL dibuat berdasarkan model pragmatis praktis. FL adalah representasi logika berpikir manusia yang tertuang dalam bentuk kata-kata.

Kajian saintifik pertama tentang logika berfikir manusia ini dipublikasikan oleh Lukazewics, seorang filsuf, sekitar tahun 1930-an. Ia

mengajukan beberapa representasi matematika tentang “kekaburan” (fuzziness) logika ketika manusia mengungkapkan atau menyatakan penilaian terhadap tinggi , tua dan panas (tall,old & hot). Jika logika klasik hanya menyatakan 1 atau 0 , ya taua tidak , maka ia mencoba mengembangkan pernyataan ini dengan menambah faktor kepercayaan (truth value) diantara 0 dan 1.

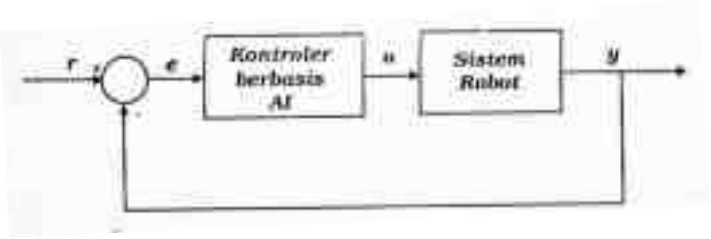
Di tahun 1965 , Lotfi Zadeh , seorang profesor di University of California , Berkeley US, mempublikasikan papernya yang terkenal , “Fuzzy Sets” . Penelitian-penelitian tentang FL dan Fuzzy system dalam AI yang berkembang dewasa ini hampir selalu menyebutkan papaer zadeh itulah sebagai basis pijakannya. Ia mampu menjabarkan FL dengan pernyataan matematik dan visual yang relatif mudah untuk dipahami. Karena basis kajian FL ini kental berkaitan dengan sistem kontrol (Zadeh adalah profesor dibidang teknik elektro ) maka pernyataan matematiknya banyak dikembangkan dalam konteks pemrograman komputer.

Metoda AI lain yang juga berkembang adalah algoritma genetik (genetic algorithm , GA) – untuk selanjtnya disebut GA. Dalam pemrograman komputer, aplikasi GA ini dikenal sebagai pemrograman berbasis teori evolusi (evolutionary computation , EC) – untuk selanjutnya disebut sebagai EC . Konsep EC ini publikasikan pertama kali oleh Holand (1975),. Ia mengajukan konsep pemrograman berbasis GA yang diilhami oleh teori Darwin. Intinya alam(nature), seperti manusia, memiliki kemampuan adaptasi dan pembelajaran alami “tanpa perlu dinyatakan : apa uyang harus dialkukan ”. Dengan kata lain , alam memilih “kromosom yang baik” secara “buta” / alami. Seperti pada ANN , kajian GA juga pernah mengalami masa vakum sebelum akhirnya banyak peneliti memfokuskan kembali perhatiannya pada teori EC.

GA pada dasarnya terdiri dari dua macam mekanisme , yaitu encoding dan evaluation. Davis (1991) mempublikasikan papernya yang berisi tentang beberapa metoda encoding. Dari berbagai literatur diketahui bahwa tidak ada metoda encoding yang mampu menyelesaikan semua permasalahan dengan

sama baiknya. Namun demikian, banyak peneliti yang menggunakan metoda bit string dalam kajian-kajian EC dewasa ini.

Aplikasi AI dalam kontrol robotik dapat iidlustrasikan sebagai berikut,




*Gambar 1.24 Kontrol robot loop tertutup berbasis AI*

Penggunaan AI dalam kontroler dilakukan untuk mendapatkan sifat dinamik kontroler “secara cerdas”. Seperti telah dijelaskan di muka, secara klasik, kontrol P,I,D atau kombinasi, tidak dapat melakukan adaptasi terhadap perubahan dinamik sistem selama operasi karena parameter P, I dan D itu secara teoritis hanya mapu memberikan efek kontrol terbaik pada kondisi sistem yang sama ketika parameter tersebut di-tune. Disinilah kemudian dikatakan bahwa kontrol klasik ini “belum cerdas” karena belum mampu mengakomodasi sifat-sifat nonlinieritas atau perubahan-perubahan dinamik , baik pada sistem robot itu sendiri maupun gangguan lingkungan.

Banyak kajian tentang bagaimana membuat P,I dan D menjadi dinamis , seperti misalnya kontrol adaptif, namun disini hanya akan dibahas tentang rekayasa bagaiman membuat sistem kontrol bersifat “cerdas” melalui pendekatan –pendekatan AI yang populer, seperti ANN, FL dan EC atau GA. Gambar 1.24 mengilustrasikan tentang AI yang diinstal secara langsung sebagai kontroler sistem robot . Dalam aplikasi lain, AI juga dapat digunakan untuk membuat proses identifikasi model dari sistem robot , model lingkungan atau gangguan, model dari tugas robot (task) seperti membuat rencana trajektori, dan sebagainya. Dalam hal ini konsep AI tidak digunakan secara langsung (direct) ke dalam kontroler, namun lebih bersifat tak langsung (indirect).

## 1.7 SENSOR

Sebuah sensor adalah sebuah perangkat yang mengukur beberapa atribut lingkungan. Menjadi salah satu dari tiga hal terpenting dalam robotika (selain perencanaan dan pengendalian), sensor berperan penting dalam paradigma robot .

**Exteroceptive Sensor**

Exteroceptive sensor memberikan informasi tentang lingkungan sekitar. Memungkinkan pada robot untuk berinteraksi dengan dunia. Sensor exteroceptive diklasifikasikan menurut fungsi mereka.

Sensor jarak digunakan untuk mengukur jarak relatif (kisaran) antara sensor dan objek dalam lingkungan. Sensor jarak melakukan tugas yang sama dengan indera taktil yang dilakukan pada manusia. Ada jenis lain pengukuran jarak, seperti laser , penggunaan kamera, atau proyeksi grid, garis berwarna atau pola titik untuk mengamati bagaimana pola terdistorsi oleh lingkungan. Untuk pendekatan pada akal manusia, robot humanoid dapat menggunakan sonars dan sensor infra merah, atau sensor taktil seperti sensor bump, kumis (atau antena), kapasitif dan sensor piezoresistif.

Array tactels dapat digunakan untuk menyediakan data tentang apa yang telah tersentuh. The Hand Shadow menggunakan sebuah array 34 tactels yang diatur di bawah poliuretan kulit pada setiap ujung jari. Sensor taktil juga memberikan informasi tentang kekuatan dan torsi yang ditransfer antara robot dan benda lainnya.

Sensor Vision mengacu pada pengolahan data dari setiap modalitas yang menggunakan spektrum elektromagnetik untuk menghasilkan gambar. Dalam robot humanoid ini digunakan untuk mengenali objek dan menentukan sifat mereka. Sensor vision bekerja paling mirip dengan mata manusia. Sebagian besar robot humanoid menggunakan CCD kamera sebagai sensor penglihatan.

Sensor suara memungkinkan robot humanoid untuk mendengar pidato dan suara lingkungan, dan tampil sebagai telinga manusia. Mikrofon yang biasanya digunakan untuk fungsi ini.

## 1.8 AKTUATOR

Aktuator adalah motor yang bertanggung jawab untuk pergerakan robot. Robot humanoid yang dibangun sedemikian rupa sehingga mereka dapat meniru pergerakan tubuh manusia, sehingga mereka menggunakan aktuator yang melakukan seperti otot-otot dan sendi, meskipun dengan struktur yang berbeda. Untuk mencapai efek yang sama seperti gerakan manusia, robot humanoid menggunakan aktuator beserta rotari. Aktuator dapat berupa listrik, pneumatik , hidrolik, piezoelektrik atau ultrasonik .

Hidrolik dan aktuator listrik memiliki pergerakan yang sangat kaku dan hanya dapat dibuat untuk bergerak dengan cara yang sesuai dengan menggunakan complex feedback control strategies. Sementara listrik aktuator motor tanpa keping lebih cocok untuk kecepatan yang tinggi dan aplikasi beban rendah, sehingga hidrolik beroperasi dengan baik pada kecepatan rendah dan aplikasi beban tinggi.

Aktuator piezoelectric menghasilkan gerakan kecil dengan kemampuan kekuatan tinggi ketika tegangan diberikan. Aktuator piezoelectric dapat digunakan untuk penentuan posisi gerak yang tepat dan untuk menghasilkan maupun penanganan dengan kekuatan yang tinggi atau tekanan dalam situasi statis atau dinamis.

Aktuator ultrasonik dirancang untuk menghasilkan gerakan dalam urutan mikrometer pada frekuensi ultrasonik (lebih dari 20 kHz). Aktuator ultrasonik berguna untuk mengendalikan getaran, aplikasi positioning dan switching cepat.

Pneumatik aktuator beroperasi berdasarkan gas kompresibilitas . Karena dapat meningkat, dapat memperluas sepanjang sumbu, dan dapat mengempis. Jika salah satu ujung tetap, maka yang lain akan bergerak dalam linier lintasan . Aktuator ini digunakan untuk kecepatan rendah dan aplikasi beban rendah / menengah. Aktuator pneumatik terdiri dari : silinder , bellow , mesin pneumatik, motor stepper pneumatik dan otot-otot buatan pneumatik.

## 1.9 Interaksi Manusia dan Robot

Kehadiran robot dalam kehidupan manusia semakin hari disadari semakin banyak manfaatnya. Robotika tidak lagi dipandang sebagai ilmu yang berkembang hanya dalam konteks teknologi (fisik) saja, namun semakin hari semakin banyak masalah yang berkaitan dengan lingkungan hidup manusia yang perlu juga diambil perhatian.

Seperti telah diketahui, robot berkembang dari aplikasi-aplikasi di industri dalam struktur lingkungan yang lebih dikondisikan sebagai kawasan pabrik. Sehingga robot lebih banyak didisain dalam bentuk yang relatif khas sesuai dengan kebutuhan pabrik, seperti manipulator, dan kebanyakan tidak bersifat mobile atau tidak otonomous. Namun kehadiran robot di lingkungan yang bersifat lebih fleksibel, seperti misalnya rumah sakit, rumah tangga, perkantoran, eksplorasi hutan, dan pembangunan kawasan-kawasan berbahaya (plant nuklir, kimia, dsb.) telah membuat manusia harus menata ulang definisi, konstruksi dan fungsi robot. Keadaan ini telah menempatkan robot sebagai kehidupan keseharian sehingga dikenal istilah human-robot interaction.

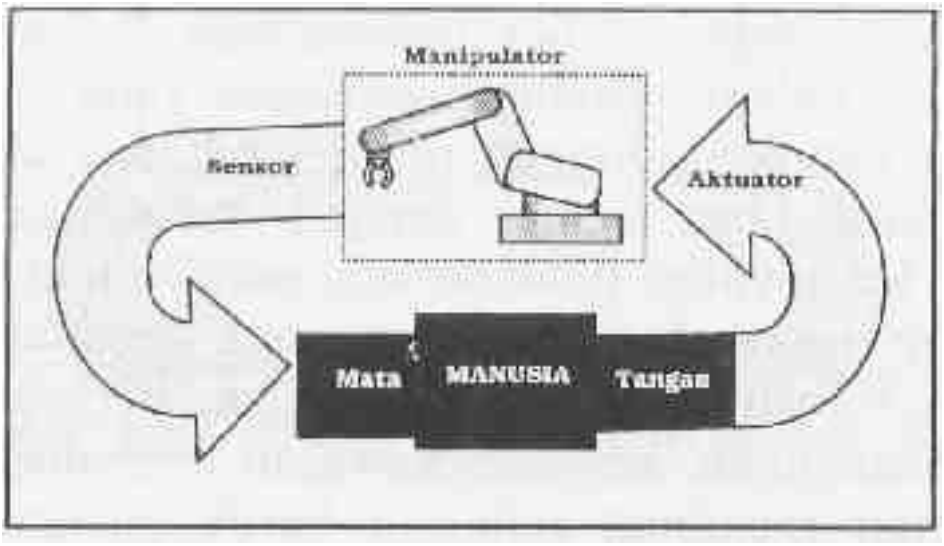
Interaksi antara manusia dengan robot atau mesin (human-machine interactions) dapat dinyatakan dalam 3 tingkatan, yaitu:

- Manusia sebagai kontroler robot sepenuhnya,
- Manusia sebagai manager dari operasi robot, dan
- Manusia dan robot berada dalam kesetaraan.

Dalam dunia industri, faktor interaksi antara manusia dan mesin sangat penting. Makin sedikit ketergantungan mesin terhadap manusia maka secara relatif makin tinggi tingkat otomasinya. Pada gilirannya biaya produksi untuk membayar “keahlian” manusia dapat dikurangi dan digantikan oleh mesin (robot). Perangkat yang digunakan dalam interaksi ini dikenal sebagai human-machine interface. Interface dapat berupa perangkat keras ataupun perangkat lunak.

Interaksi yang paling dasar antara manusia dengan robot adalah interaksi yang menempatkan manusia sebagai pengontrol gerakan robot sepenuhnya. Dalam hal ini biasanya robot tidak memiliki kemampuan untuk melakukan sendiri segala gerakan. Semua titik aktuator hanya dapat digerakan melalui “perintah” operator atau manusia. Robot hampir tidak lagi memerlukan sensor

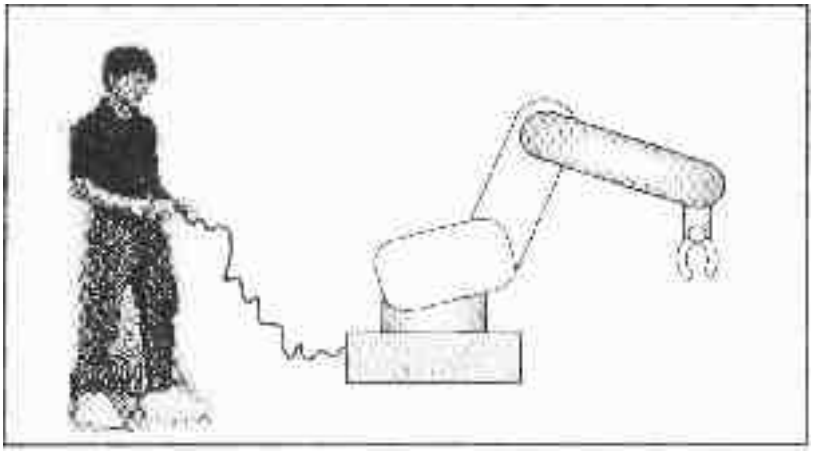
pada sendi-sendi ataupun pergerakan. Dengan campur tangan manusia ini maka pergerakan robot dapat langsung "dideteksi" secara visual melalui penglihatan mata. Sensor secara perangkat keras yang diperlukan mungkin hanya berupa switch pembatas (limit switch) untuk menghindari gerakan yang berbahaya atau di luar kontrol. Cara ini dikenal sebagai pengendalian robot menggunakan remote control, baik secara wireless (tanpa kabel) maupun menggunakan kabel.




Gambar 1.25 Sistem Remote Control

Secara umum sistem remote control dapat diilustrasikan seperti **Gambar 1.25**. Mata menggantikan fungsi sensor, sedangkan tangan menggantikan fungsi pemberi sinyal control kepada aktuator.

**Gambar 1.26** menunjukkan sebuah robot manipulator yang dikontrol sepenuhnya oleh operator melalui kabel (cable remote control).



Gambar 1.26 Sistem Remote Control pada manipulator

Pada panel (boks), kontrol yang dipegang oleh operator terdapat tombol-tombol untuk mengontrol seluruh pergerakan sendi robot. Robot jenis remote control ini banyak digunakan untuk tugas yang sangat rumit yang jika dibuat secara otomatis terlalu banyak kendala yang dihadapi. Salain itu disain dengan sistem remote dapat menekan pembiayaan dalam investasi maupun dalam hal biaya pemrograman (computational cost). Dengan cara ini pula "sifat cerdas" tidak perlu dituangkan dalam sistem perangkat keras kontroler karena tugas "berfikir" diambil alih oleh operator.

Contoh klasik dalam bentuk permainan, misalnya robot manual yang digunakan dalam kompetisi atau kontes robot (Robocon). Spesifikasi manual menunjukkan bahwa robot dikontrol sepenuhnya secara manual oleh operator. Walaupun manual atau tidak otomatis, dengan mempertimbangkan bahwa manusia atau operator adalah termasuk dalam sistem robot secara keseluruhan maka fungsi atau kinerja robot itu dapat bersifat "cerdas" tergantung kecerdasan atau keahlian sang operator.

Contoh aplikasi yang lain dapat dijumpai dalam peralatan militer. Robot penjinak bom (bom disposal robot) justru dianggap lebih aman jika dikendalikan oleh operator. Setidaknya hingga sekarang, masih belum dijumpai robot penjinak bom yang sepenuhnya dapat bergerak secara otomatis. Alasannya adalah bahwa penggunaan robot otomatis dapat lebih mencelakakan jika robot gagal berfungsi dan dapat bertindak liar.

Pada tingkatan berikutnya, manusia bertindak sebagai "manajer" bagi robot. Tugas secara detil dilakukan sendiri oleh robot, sedangkan tugas secara keseluruhan diatur oleh operator. Dalam hal tertentu robot sudah dimuati kemampuan kontrol otomatis (umpan balik), seperti kontrol posisi setiap sendi, kontrol kecepatan rotasi dan kontrol torsi (reaktif terhadap pembebanan). Dalam segi operasional, dalam tugas-tugas tertentu robot dalam kelas ini masih memerlukan arahan dari operator. Misalnya, dalam hal penentuan trajektori dari suatu manipulator. Operator dapat memprogram secara off-line. Gerakan ujung lengan dilatih dengan menggerakkan secara manual menuju sasaran. Pada saat yang sama kontroler merekam trajektori ini. Setelah proses pelatihan selesai, robot dapat di-run secara mandiri.

Konteks interaksi manusia dengan robot dalam kesetaraan (dengan tentunya tetap berpegang pada prinsip bahwa robot adalah pembantu manusia) perlu dijabarkan lebih rinci dengan beberapa alasan yang dapat ditunjukkan seperti dalam tabel berikut ini.

**Tabel 1.1.** Spesifikasi kemampuan bekerja manusia vs robot

| Manusia | Robot |
|---|---|
| **Mudah letih** | Tidak pernah letih |
| **Kurang presisi** | Sangat presisi |
| **Kualitas kerja tidak stabil** | Kualitas kerja stabil |
| **Pengalaman banyak dan dinamis** | Sukar dibuat dinamis dalam mengakomodasikan pengalaman kerja |
| **Pengetahuan bersifat global** | Pengetahuan tergantung program |
| **Mengerti tugas secara alami (mudah beradaptasi)** | Kemampuan beradaptasi sangat terbatas |

Dengan memperhatikan masing-masing kelebihan dan kekurangan maka manfaat akan lebih besar jika manusia dan robot dapat berkolaborasi. Dalam hal ini, karena manusia berada pada posisi sebagai "tuan" maka beberapa keunggulan sebagai "expert" sepertinya dapat dijadikan sebagai bahan ajar dalam membuat robot lebih "pintar" seperti manusia melalui komunikasi. Dengan demikian robot tidak perlu dibuat sangat canggih (karena sangat mahal), namun cukup diberi kemampuan dapat menerima pengajaran terus-

menerus dari manusia. Kemampuan dasar robot otonomous, seperti navigasi (gerak mobile) dan manipulasi (gerak tangan) tetap harus ada.

Sedangkan tugas apa yang harus dikerjakan robot ini dan dengan cara bagaimana, akan diarahkan oleh manusia/operator melalui interaksi dan komunikasi. Jadi, robot harus diprogram agar mampu melakukan interaksi dengan manusia dan dapat memahami pengetahuan yang di-supply oleh manusia selama dalam proses pengajaran atau pemberian perintah. Prosedur dan media interaksi ini dapat berupa sinyal frekuensi radio, suara, gambar (Visual), sentuhan (tangan dan badan), dan sebagainya. Oleh karena itu, robot ideal yang dapat melakukan interaksi dengan manusia setidaknya harus memiliki kemampuan panca indra seperti pada manusia.

# Latihan

1. Apa yang dimaksud dengan Robot dan Robotika, Jelaskan berikut contohnya!
2. Sebutkan dan jelaskan aplikasi atau penerapan robot dalam berbagai bidang yang ada hingga saat ini!
3. Disiplin ilmu apa saja yang terkait dengan robotika dan apa peranan bidang ilm tersebut dalam menunjang kemajuan robotika?
4. Sebutkan dan jelaskan komponen-komponen utama dari sebuah robot!
5. Jelaskan perbedaan antara sistem kontrol loop terbuka dengan sistem kontrol loop tertutup!
6. Apa yang dimaksud dengan sensor dan aktuator berikan contohnya masing-masing?
7. Jelaskan perbedaan antara kontrol : proporsional, integral dan derivative!
8. Sebutkan dan jelaskan tiga tingkatan interkasi antara manusia dan robot!

# REFERENSI

http://ekstrarobotik.tripod.com/id3.html

http://en.wikipedia.org/wiki/Humanoid_robot

http://newstekno.blogspot.com/2009/02/fungsi-robot.html

Agilent.(1999). *Quadrature Decoder/Counter Interface ICs. Technical Data*, Agilent Technologies, Inc., http://www.semiconductor.agilent.com

Analog Devices. (1999). *ADXL105 Datasheet*, Analog Devices, Inc., http://www.analog.com

Applied Measurement. (1998). *Miniature Series LVDT: Displacement Transducer. AML/M Series Data sheet*, Applied Measurement, Ltd., http://www.appmeas.co.uk

Braunl, T. (2003.*Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.*BukuTeks. Berlin: Springer Verlag Berlin Heidelberg, Inc.

Dinsmore. (1999). *Datasheet Dinsmore Analog Sensor No. 1525*, Dinsmore Instrument Co., http://dinsmoregroup.com/dico

Honeywell.(2005). *Digital Compass Solution.* Sensor Product Datasheet, Honewell, Inc., http://www.magneticsensors.com

Precision Navigation. (1998). *Vector Electronic Module. Application Notes*, Precision Navigation, Inc., http://www.precisionnav.com

http://www.societyofrobots.com/actuators.shtml

http://ocw.gunadarma.ac.id/course/diploma-three-program/study-program-of-computer-engineering-d3/robotika/mekanika-robotika

# BAB II

# TEKNIK PEMROGRAMAN ROBOT

## 2.1. Pendahuluan

Pada bab ini akan dibahas struktur pemrograman bahasa C dan Assembly, sekilas tentang code vision AVR. Sistem instalasi code vision, membuat project dan kompilasi pada code vision AVR. Debugging, downloader dan uploader serta contoh program.

## 2.2. Sekilas Struktur Bahasa C

Struktur penulisan bahasa C secara umum terdiri atas empat blok, yaitu :

1. Header,
2. Deklarasi konstanta global atau variabel,
3. Fungsi dan prosedur
4. Program utama

Secara umum, pemrograman C paling sederhana dilakukan dengan hanya menuliskan program utamanya saja, yaitu :

```
void main (void)

{

...
```

## A. HEADER

Header berisi include file (.hex), yaitu library (pustaka) yang akan digunakan dalam pemrograman.

```
Contoh:
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
. . .
```

## B. TIPE DATA

Berikut ini adalah tabel tipe-tipe variabel data yang dapat digunakan di kompiler Code Vision AVR:

Tabel 2.1 Tipe-tipe variable data

| **Type** | **Size(Bits)** | **Range** |
|---|---|---|
| Bit | 1 | 0, 1 |
| Char | 8 | -128 to 127 |
| Unsigned char | 8 | 0 to 255 |
| Signed char | 8 | -128 to 127 |
| Int | 16 | -32768 to 32767 |
| Short int | 16 | -32768 to 32767 |
| Unsigned int | 16 | 0 to 65535 |
| Signed int | 16 | -32768 to 32767 |
| Long int | 32 | -2147483648 to 2147483647 |
| Unsigned long int | 32 | 0 to 4294967295 |
| Signed long int | 32 | -2147483648 to 2147483647 |
| Float | 32 | ±1.175e-38 to ±3.402e38 |
| Double | 32 | ±1.175e-38 to ±3.402e38 |

Khusus untuk tipe data *bit* hanya bisa dideklarasikan untuk variabel global.

## C. KONSTANTA

Penulisan konstanta adalah sebagai berikut:

- Integer atau lng integer dapat ditulis dengan format desimal (contoh 1234), biner dengan awlan **0b** contoh (0b101001), heksadesimal dengan awalan **0x** (contoh 0xff) atau oktal dengan awalan **0** (0777).
- Unsigned integer ditulis dengan diakhiri **U** (contoh 10000U).
- Long integer diulis dengan diakhiri **L** (contoh99L)
- Unsigned long integer ditulis dengan diakhiri **UL** (contoh 99UL)
- Floating point ditulis dengan diakhiri **F** (contoh 1.234F)

Karakter konstanta harus dituliskan dalam tanda kutip (contoh 'a'), sedangkan konstanta string harus dalam tanda kutip dua (contoh “Saya Belajar C).

## D. LABEL, VARIABEL, FUNGSI

Identifikasi label, variabel dan fungsi dapat berupa huruf (A....Z, a...z) dan angka (0...9), juga karakter underscore ( _ ). Meskipun begitu identifikasi hanya bisa dimulai dengan huruf ataun karakter underscore. Yang lebih penting lagi, identifikasi ini *Case is significant*, yaitu huruf besar dan kecil berbeda. Misal, *variabel1* tidak sama *Variabel1*. Identifikasi bisa memuat sebanyak 32 karakter.

## E. KOMENTAR

Contoh:

```
/* ini komentar */
/* ini komentar
   multi baris*/
```

Komentar diawali dengan tanda '/*' dan diakhiri dengan '*/'.

Sedangkan komentar satu baris bisa dengan tanda '//'.

Contoh:
*// Ini juga komentar*

## F. RESERVED KEYWORDS

Berikut ini adalah daftar kata baku yang tidak bisa dipakai (*reserved keywords*) untuk label, identifikasi atau variable

| | | |
|---|---|---|
| Break | Flash | Signed |
| Bit | Float | Sizeof |
| Case | For | Sfrb |
| Char | funcused | Sfrw |
| Const | Goto | Static |
| Continue | If | struct |
| Default | Inline | switch |
| Do | Int | typedef |
| Double | interrupt | union |
| eeprom | Long | unsigned |
| Else | Register | void |
| Enum | Return | volatile |
| Extern | Short | while |

.

## G. OPERATOR

Suatu instruksi pasti mengandung operator dan operand. Operand adalah variabel atau konstanta yang merupakan bagian pernyataan sedangkan operator adalah suatu simbol yang menyatakan operasi mana yang akan dilakukan oleh operand tersebut.

Contoh:

```
c = a + b;
```

Ada tiga operand (a, b dan c) dan dua operator (= dan +).

Operator dalam C dibagi menjadi 3 kelompok, yaitu:

1. Unary

   Operator yang beroperasi pada satu operand, misal: -n.

2. Binary

   Operator yang beroperasi pada dua operand, misal: a-n.

3. Ternary

   Operator yang memerlukan tiga atau lebih operand,

   misal: a=(b*c)+d.

## H. ARITMATIKA

Tabel 2.2 Aritmatika

| Simbol | Contoh | Aritmatika |
|---|---|---|
| + | c=a+b<br>n=n+2 | Penjumlahan |
| - | c=a-b<br>n=n-2 | Pengurangan |
| ++ | ++i | Kenaikan(increment, sama dengan i=i+1 |
| - | --i | Penurunan(decrement, sama dengan i=i-1 |
| * | c=a*b<br>n=n*2 | Perkalian |
| / | c=a/b<br>n=n/2 | Pembagian |

| | | |
|---|---|---|
| % | sisa=a% b | Menghasilkan sisa dari pembagian. A dan b bilangan bulat |
| = | a=b | Pemberian nilai |
| += | a+=2 | Penambahan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a=a+2 |
| -= | a-=2 | Pengurangan suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a=a-2 |
| *= | a*=2 | Pengalian suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a=a*2 |
| /= | a/=2 | Pembagian dari suatu nilai pada suatu variabel yang sudah ada sebelumnya. Sama dengan a=a/2 |
| %= | a/=2 | Sisa dari suatu nilai pada suatu variabel yang sudah ada sebelumnya yang dibagi oleh nilai atau variabel lain. Sama dengan a=a/2 |
| * | *pointer | Menunjukkan isi dari pointer |

## I. SIMBOL

Tabel 2.3 Simbol

| **Symbol** | **Contoh** | **Logika Pembanding** |
|---|---|---|
| "==" | if(a==b) | Logika sama dengan, digunakan untuk pembanding. Menghasilkan nilai *true* jika a = b. |
| != | if(a!=b) | Tidak sama dengan. Menghasilkan nilai *true* jika a ≠ b. |
| < | if(a<b) | Logika lebih kecil dari. Menghasilkan nilai *true* jika a < b. |
| <= | if(a<=b) | Logika lebih kecil sama dengan dari. Menghasilkan nilai *true* jika a ≤ b. |
| > | if(a>b) | Logika lebih besar dari. Menghasilkan nilai *true* jika a > b. |
| >= | if(a>=b) | Logika lebih besar sama dengan dari. Menghasilkan nilai *true* jika a ≥ b. |

| | | |
|---|---|---|
| ! | If(!a) | NOT |
| && | if(a==b && a==c) | AND |
| \|\| | if(a==b \|\| a==c) | OR |

## J. MANIPULASI BIT

Tabel 2.4 Manipulasi bit

| | | |
|---|---|---|
| ~ | a=~b | Complement b=1100;a=0011 |
| & | c= a & b | AND untuk manipulasi bit. a=1100; b=1001; maka c=1000 |
| \| | c=a \| b | OR untuk manipulasi bit. a=1100; b=1001; maka c=1101 |
| ^ | c=a ^ b | XOR untuk manipulasi bit. a=1100; b=1001; makac=0101. |
| << | c=a << n | Shift left, manipulasi bit menggeser ke kiri sejauh n bit. a=1101; n=2; maka c=110100 |
| >> | C= a >> n | Shift Right, manipulasi bit menggeser ke kiri sejauh n bit. a=11010; n=2; maka c=0110. |

## K. PERCABANGAN

- **if-then**

Bentuk umum dari percabangan ini adalah:

```
if (kondisi) {
   // pernyataan
  };
```

Artinya adalah pernyataan akan dijalankan jika kondisi terpenuhi.

Contoh :

```
if (a<0x50) {
   PORTC=0x55; // PORTC
akan dikirim data 0x55
 };
```

- **if-then-else**

Bentuk umum dari percabangan ini adalah:

```
if (kondisi) {
   // pernyataan a
  }
else {
  // pernyataan b
 };
```

Artinya adalah pernyataan a akan dijalankan jika kondisi terpenuhi dan pernyataan b akan dijalankan jika kondisi tidak terpenuhi.

```
if (a<0x50) {
   PORTC=0x55;
  }
else {
  PORTC=0xAA;
 };
```

PORTC akan dikirim data 0x55 jika nilai a lebih kecil dari 0x50 dan
PORTC akan dikirim data 0xAA jika a≥0x55.

- **Switch – case**

Pernyataan switch – case digunakan jika terjadi banyak percabangan. Struktur penulisan pernyataan ini adalah sebagai berikut:

```
. . .
switch (ekspresi) {
        case konstanta1:
        penyataan1
        break;
        case konstanta2:
        penyataan2
        break;
        . . .
        case konstanta N:
        penyataan N
        break;
        }
. . .
```

Contoh :

```
. . .
switch (a) {
        case 1:
        PORTC=0x01;
        break;
        case 2:
        PORTC=0x02;
        break;
        case 3:
        PORTC=0x03;
        break;
        }
. . .
```

PORTC akan dikirim data 0x01 jika nilai a=1, PORTC akan dikirim data 0x02 jika nilai a=2 dan PORTC akan dikirim data 0x03 jika nilai a=3.

- **Switch – case – default**

Pernyataan switch – case – default hampir sama dengan switch – case. Yang membedakan adalah bahwa dengan adanya default maka jika tidak terdapat kondisi case yang sesuai dengan ekspresi switch maka akan menuju pernyataan yang terdapat di bagian default. Struktur penulisan pernyataan ini adalah sebagai berikut:

```
. . .
switch (ekspresi) {
        case konstanta1:
        penyataan1
        break;
        case konstanta2:
        penyataan2
        break;
        . . .
        case konstanta N:
        penyataan N
        break;
        default:
        pernyataan-
pernyataan;
        }
. . .
```

```
Contoh:
. . .
switch (a) {
        case 1:
        PORTC=0x01;
        break;
        case 2:
        PORTC=0x02;
        break;
        case 3:
        PORTC=0x03;
        break;
        default:
        PORTC=0xFF;
        }
. . .
```

PORTC akan dikirim data 0x01 jika nilai a=1, PORTC akan dikirim data 0x02 jika nilai a=2 dan PORTC akan dikirim data 0x03 jika nilai a=3 dan jika kondisi case tidak sesuai dengan ekspresi maka pernyataan di default akan dijalankan.

## L. PERULANGAN

- **For**

Pernyataan for akan melakukan perulangan beberapa kali sesuai yang diinginkan. Struktur penulisan perulangan for adalah sebagai berikut:

```
. . .
for (mulai; kondisi; penambahan atau pengulangan) {
        pernyataan-pernyataan;
);
```

Mulai adalah pemberian nilai awal, kemudian kondisi adalah pengondisi dalam for yaitu jika kondisibernilai *true* maka pernyataan dalam for akan dijalankan. Penambahan atau pengurangan adalah penambahan atau pengurangan terhadap nilai awal.

Contoh :

```
. . .
a=1;
for (i=1; i<50; i++) {
        a=a*2
        PORTC=a;
};
. . .
```

Contoh di atas akan melakukan perulangan 50 kali, yaitu dari 1 hingga 50 dengan penambahan 1 (i++, lihat operator aritmatik). Hasilnya PORTC akan dikirim data 1, kemudian data 2,4,8,...

- ***While***

Bentuk dari perulangan ini adalah sebagai berikut:

```
while (kondisi) {
        pernyataan-pernyataan;
}
```

Jika kondisi memenuhi (bernilai true) maka pernyataan-pernyataan di bawahnya akan dijalankan hingga selesai, kemudian akan menguji kembali kondisi di atas.

```
Contoh:
. . .
i=1;
a=1;
while (i<50) {
        a=a*2;
        PORTC=a;
        i++;
};
. . .
```

*bandingkan dengan perulangan for.*

- ***Do – While***

Bentuk perulangan ini kebalikan dari while – do, yaitu pernyataan dilakukan terlebih dahulu kemudian diuji kondisinya.

```
Do {
pernyataan-pernyataan;
}
while (kondisi);
```

```
Contoh:
. . .
i=1;
a=1;
do {
        a=a*2;
        PORTC=a;
        i++;
}
while (i<50);
. . .
```

Bandingkan dengan perulangan for dan while-do.

## M. KONVERSI POLA (%)

Karakter %_ dipakai sebagai operator konversi pola. Konversi pola akan sangat berguna pada saat kita menampilkan hasil ke LCD.

Contoh :

```
sprintf(buf,"Angka %d", 14);
```

- %o menampilkan bilangan oktal bulat.
- %d menampilkan bilangan bulat positif
- %x menampilkan bilangan heksadesimal bulat.
- %u menampilkan bilangan desimal tanpa tanda
- %f menampilkan bilangan pecahan
- %i menampilkan bilangan integer
- %c menampilkan karakter yang ditujukkan bilangan ASCII

## N. PROSEDUR DAN FUNGSI

Seringkali dalam suatu program kita menemukan kelompok instruksi untuk suatu keperluan tertentu yang sering dijalankan. Kelompok instruksi ini bisa dibuat sebagai prosedur atau fungsi. Langkah ini akan dapat menghemat memori dibandingkan bila instruksi-instruksi tersebut ditulis berulang-ulang. Ingat bahwa disini kita akan memprogram mikrokontroler yang memiliki memori yang terbatas.

- **PROSEDUR**

Prosedur adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu tanpa mengembalikan suatu nilai.

```
. . .
        void nama_prosedur (parameter1, parameter2,...parameterN) {
        pernyataan-pernyataan;
        }
. . .
```

Contoh:

```
. . .
        void delay(unsigned char I) {
        while (i--) {
                /*penulisan untuk bahasa assembly*/
                /*akan dibahas tersendiri*/
        #asm
                nop
                nop
        #endasm
        };
}
. . .
```

- **FUNGSI**

Fungsi adalah suatu kumpulan instruksi untuk mengerjakan suatu keperluan tertentu dengan hasil akhir pengembalian nilai dari keperluan tersebut.

```
. . .
        type data nama_fungsi (parameter1, parameter2,... parameterN)
        {
                pernyataan-pernyataan;
                return variable_hasil;
        }
        . . .
```

Contoh :

```
. . .
int luas(int pj, int lb) {
        luas = pj*lb;
        return luas;
}
. . .
```

Pemanggilan prosedur atau fungsi dilakukan dengan langsung menuliskan prosedur atau fungsinya.

```
Contoh:
. . .
delay(150);                 // cara memanggil prosedure
dt = luas(5,10);  // cara menggunakan fungsi
)
. . .
```

## O. MEMASUKKAN BAHASA ASSEMBLY

Kita sebut sebagai in-line assembly. Dalam pemrograman dengan bahasa C ini kita masih dapat memasukkan bahasa assembly ke dalam program C. Struktur penulisannya pun juga mudah, yaitu:

```
. . .
#asm            // dimulai dengan #asm
        nop     // blok bahasa assembly
        nop
#endasm         // diakhiri dengan #endasm
. . .
```

atau jika hanya beberapa instruksi maka kita bisa melakukannya dengan cara:

```
. . .
#asm("nop\nop\nop")
. . .
```

**P. PERNYATAAN KENDALI LAINNYA**

- **BREAK**

Pernyataan ini akan menghentikan atau menyebabkan keluar dari suatu blok program.

- **CONTINUE**

Pernyataan ini akan menyebabkan kendali melakukan kembali proses perulangan dari awal.

- **GOTO – LABEL**

Pernyataan ini akan melakukan loncatan ke label yang dituju.

## 2.3. Assembler

Struktur bahasa Assembler untuk mikrokontroler AVR secara umum terdiri atas.

- Inisialisasi program
- Program utama

Berikut adalah contoh sebuah program aplikasi untuk mikrokontroler AVR :

```
.include "m8535def.inc"
.org 0x0000
       rjmp main

main:  ldi r16,low(RAMEND)
       out SPL,r16                    Inisialisasi
       ldi r16,high(RAMEND)
       out SPH,r16

       ldi r16,0xff
       out ddra,r16                   Program utama
       out PortA,r16
       cbi PortA,0
       cbi PortA,1
```

1. Menentukan jenis mikrokontroler yang digunakan dengan cara memasukkan file definisi device (m8535def.inc) ke dalam program utama.

   *.include "m8535def.inc" ;*

2. Menuliskan original address program, yaitu 0x0000. Kemudian dilanjutkan dengan instruksi rjmp / relative jump ke label main. Hal ini dimaksudkan agar program memory tidak tumpang tindih dengan data memory.

   *.org 0x0000*
   *rjmp main*

3. Menentukan isi Stack Pointer dengan address terakhir RAM (RAMEND). Untuk ATMega8535 yaitu 0x025F. Ini dimaksudkan agar program utama mulai ditulis setelah address terakhir RAM.

   *main: ldi r16,low(RAMEND) ; low byte address RAM = 5F*
   *out SPL,r16*
   *ldi r16,high(RAMEND) ; high byte address RAM = 02*
   *out SPH,r16*

### 2.3.1. Register I/O

Setiap port ATMega8535 terdiri dari 3 register I/O yaitu DDRx, Portx dan PINx.

- DDRx (Data Direction Register)
  Register DDRx digunakan untuk memilih arah pin. Jika DDRx = 1 maka Pxn sebagai pin output Jika DDRx = 0 maka Pxn sebagai input.
- Portx (Port Data Register)Register Portx digunakan untuk 2 keperluan yaitu untuk jalur output atau untuk mengaktifkan resistor pullup.
  1. Portx berfungsi sebagai output jika DDRx = 1 maka :
     Portxn = 1 maka pin Pxn akan berlogika high.
     Portxn = 0 maka pin Pxn akan berlogika low.
  2. Portx berfungsi untuk mengaktifkan resistor pullup jika DDRx = 0 maka :
     Portxn = 1 maka pin Pxn sebagai pin input dengan resistor pull up.
     Portxn = 0 maka pin Pxn sebagai output tanpa resistor pull up.

Tabel 2.5 Konfigurasi Port

| **DDRxn** | **Portxn** | **I/O** | **Pull up** | **Comment** |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri state (Hi-Z) |
| 0 | 1 | Input | Yes | Pull up aktif |
| 1 | 0 | Output | No | Output Low |
| 1 | 1 | Output | No | Output High |

Catatan :
x menunjukkan nama port (A,B,C,D)
n menunjukkan nomor bit (0,1,2,3,4,5,6,7)
Nilai awal *(initial value)* seluruh register I/O adalah 00h.

- PINx (Port Input Pin Address)
  Digunakan sebagai register input.

### 2.3.2. Instruksi I/O

- in : membaca data I/O port ke dalam register
  contoh : in r16,PinA

- out : menulis data register ke I/O port
  contoh : out PortA,r16

- ldi : (load immediate) : menulis konstanta ke register sebelum konstanta tersebut dikeluarkan ke I/O port
  contoh : ldi r16,0xff

- sbi : (set bit in I/O) : membuat logika high pada sebuah bit I/O port
  contoh : sbi PortB,7

- cbi : (clear bit in I/O) : membuat logika low pada sebuah bit I/O port
  contoh : cbi PortB,5

- sbic : (skip if bit in I/O is clear) : lompati satu instruksi jika bit I/O port dalam kondisi clear/low
  contoh : sbic PortA,3

- sbis : (skip if bit in I/O is set) : lompati satu instruksi jika bit I/O port dalam kondisi set/high
  contoh : sbis PortB,3

Contoh Program 1:

```
.include "m8535def.inc"
.org 0x00
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
 r16,high(RAMEND)
out SPH,r16
```

```
ldi r16,0x00
out ddra,r16          ; PortA as input
ldi r16,0xff
out ddrb,r16          ; PortB as output
out ddrc,r16          ; PortC as output
ulang: in r16,PortA
out PortB,r16
ldi r16,0x0f
out PortC,r16
cbi PortC,0
sbic PortA,5
cbi PortC,1
sbi PortC,6
sbis PortA,5
sbi PortC,7
ldi r16,0x00
out PortB,r16
out PortC,r16
rjmp ulang
```

### 2.3.3 Operasi Aritmatika

Instruksi Aritmatika

- add : Menambahkan isi dua register.
  Contoh : add r15,r14 ; r15=r15+r14

- adc : Menambahkan isi dua register dan isi carry flag
  Contoh : adc r15,r14 ; r15=r15+r14+C

- sub : Mengurangi isi dua register.
  Contoh : sub r19,r14 ; r19=r19-r14

- mul : Mengalikan dua register. Perkalian 8 bit dengan 8 bit menghasilkan bilangan 16 bit yang disimpan di r0untuk byte rendah dan di r1 untuk byte tinggi. Untuk memindahkan bilangan 16 bit antar registerdigunakan instruksi **movw (copy register word)**

Contoh : mul r21,r20 ; r1:r0=r21*r20

- **Contoh Program**

- **Penjumlahan**

```
.include "m8535def.inc"
.org 0x00
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
ldi r16,0x80
ldi r17,0x80
add r16,r17
ldi r18,0x02
adc r16,r18
here: rjmp here
```

- **Pengurangan**

```
.include "m8535def.inc"
.org 0x00
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
ldi r16,0x09
ldi r17,0x06
sub r16,r17
ldi r17,0x03
sub r16,r17
ldi r17,0x06
sub r16,r17
here: rjmp here
```

- **Perkalian**

```
.include "m8535def.inc"
.org 0x00
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
ldi r16,5
ldi r17,100
mul r16,r17
movw r17:r16,r1:r0              ; Copy r1:r0 to r17:r16
here: rjmp here
```

- **Pembagian**

```
.include "m8535def.inc"
.org 0x00
.def drem8u =r15                ;remainder/sisa
.def dres8u =r16                ;result/hasil
.def dd8u =r16                  ;dividend/yang dibagi
.def dv8u =r17                  ;divisor/pembagi
.def dcnt8u =r18                ;loop counter
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
ldi dd8u,4
ldi dv8u,2
rcall div8u
here: rjmp here
;
div8u: sub drem8u,drem8u        ;clear remainder and carry
ldi dcnt8u,9                    ;init loop counter
d8u_1: rol dd8u                 ;shift left dividend
```

```
dec dcnt8u                          ;decrement counter
brne d8u_2                          ;if done
ret ;return
d8u_2: rol drem8u                   ;shift dividend into remainder
sub drem8u,dv8u remainder = remainder - divisor
brcc d8u_3                          ;if result negative
add drem8u,dv8u                     ;restore remainder
clc                                 ;clear carry to be shifted into result
rjmp d8u_1                          ;else
d8u_3: sec                          ;set carry to be shifted into result
rjmp d8u_1
```

### 2.3.4 Operasi Logika

Instruksi Logika

- and : Untuk meng-and-kan dua register
  Contoh : and r23,r27 ; r23=r23 and r27

- andi : Untuk meng-and-kan register dengan konstanta immediate
  Contoh : andi r25,0b11110000

- or : Untuk meng-or-kan dua register
  Contoh : or r18,r17 ; r18=r18 or r17

- ori : Untuk meng-or-kan register dengan konstanta immediate
  Contoh : ori r15,0xfe

- inc : Untuk menaikkan satu isi sebuah register
  Contoh : inc r14

- dec : Untuk menurunkan satu isi sebuah register
  Contoh : dec r15

- clr : Untuk mengosongkan (membuat jadi nol) isi register
  Contoh : clr r15 ; r15=0x00

- ser : Set all bit in register. Membuat jadi satu isi register
  Contoh : ser r16 ; r16=0xff

➢ **Contoh Program**

- **Operasi Logika**

```
.include "m8535def.inc"
.org 0x00
rjmp main
main:
ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
ldi r16,0b01110111
ldi r17,0b00001111
and r16,r17
ori r16,0b00001000
clr r16
inc r16
ser r16
dec r16
here:
rjmp here
```

### 2.3.5 Operasi Percabangan

Instruksi Percabangan

- **sbic** (skip if bit in I/O is cleared) : Skip jika bit I/O yang diuji clear
- **sbis** (skip if bit in I/O is set) : Skip jika bit I/O yang diuji set
- **sbrc** (skip if bit in register is clear) : Skip jika bit dalam register yang diuji clear
- **cp** (compare) : Membandingkan isi dua register
- **mov** (move) : Meng-copy isi dua register
- **cpi** (compare with immediate) : Membandingakan isi register dengan konstanta tertentu.

- **breq** (branch if equal) : Lompat ke label tertentu jika suatu hasil perbandingan adalah sama.
- **brne** (branch if not equal) : Lompat ke label tertentu jika suatu hasil perbandingan adalah tidak sama.
- **rjmp** (relative jump) : Lompat ke label tertentu.
- **rcall** (relative call) : Memanggil subrutin.
- **ret** (return) : Keluar dari sub rutin.

➢ **Contoh Program**

- **Operasi Percabangan**

```
.include "m8535def.inc"
.org 0x00
rjmp main
main: ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
clr r16                 ; r16=0x00
naik: inc r16           ; increment r16
cpi r16,5               ; r16=5 ?
breq lagi               ; branch to lagi if r16 = 5
rjmp naik               ; jump to naik if r16 ≠ 5
lagi: ldi r18,5         ; r18 = 5
dec r16                 ; decrement r16
cp r16,r18              ; compare r16 & r18
brne lompat             ; branch to lompat if r16=r18
rjmp lagi               ; jump to lagi if r16≠r18
lompat: rcall rutin1
rcall rutin2
henti: rjmp henti
rutin1: mov r17,r16
ret
rutin2: mov r19,r18
ret
```

## 2.4. Sekilas tentang CodeVisionAVR

CodeVisionAVR adalah C cross-compiler, Integrated Development Environment dan Automatic Program Generator dirancang untuk keluarga dari mikrokontroler AVR Atmel.

Program ini didesain untuk berjalan di bawah Windows 2000, XP, Vista dan Windows 7 32bit dan 64bit sistem operasi. C cross-compiler melaksanakan semua elemen dari bahasa C ANSI, sebagaimana yang diperbolehkan oleh arsitektur AVR, dengan beberapa fitur yang ditambahkan untuk mengambil keuntungan dari spesifisitas arsitektur AVR dan kebutuhan sistem embedded. File COFF objek dapat dikompilasi C tingkat debugged sumber, menggunakan Atmel AVR Studio debugger. Integrated Development Environment (IDE) memiliki built-in AVR Chip Programmer Sistem In-software yang memungkinkan transfer otomatis dari program ke chip mikrokontroler kompilasi sukses setelah / perakitan. In-System Programmer perangkat lunak ini dirancang untuk bekerja bersama dengan STK500 Atmel, STK600, AVRISP, AVRISP MkII, AVR Dragon, JTAGICE MkII, AVRProg (AVR910 aplikasi catatan), Kanda Sistem STK200 +, STK300, Dontronics DT006, Vogel Elektronik VTEC- Mega2000 papan pengembangan ISP, Futurlec JRAVR dan MicroTronics 'ATCPU,. Untuk debugging embedded system, yang menggunakan komunikasi serial, IDE memiliki built-in Terminal.

Selain standar C library, compiler C CodeVisionAVR telah mendedikasikan perpustakaan untuk [11]:

- alfanumerik modul LCD
- Philips bus I2C
- Sensor Suhu LM75 Semikonduktor Nasional
- Philips PCF8563, PCF8583, Maxim / Dallas DS1302 dan DS1307 Semikonduktor Real Time Clock
- Maxim / Dallas Semiconductor 1 Wire protokol
- Maxim / Dallas Semiconductor DS1820, DS18S20 dan DS18B20 Temperatur Sensor
- Maxim / Dallas DS1621 Semikonduktor Thermometer / Thermostat
- Maxim / Dallas DS2430 dan DS2433 Semikonduktor EEPROMs

- SPI
- Manajemen Sumber Daya
- Penundaan
- Gray kode konversi
- MMC / SD / SD HC FLASH kartu memori akses tingkat rendah
- FAT akses pada MMC / SD / SD HC kartu memori FLASH.

CodeVisionAVR juga berisi CodeWizardAVR Program Otomatis Generator, yang memungkinkan Anda untuk menulis, dalam hitungan menit, semua kode yang dibutuhkan untuk melaksanakan fungsi-fungsi berikut:

- Eksternal memori akses setup
- Chip sumber identifikasi ulang
- Input / Output Port inisialisasi
- Eksternal interupsi inisialisasi
- Timer / Loket inisialisasi
- Watchdog Timer inisialisasi
- UART (USART) inisialisasi dan mengganggu didorong buffer komunikasi serial
- Analog komparator inisialisasi
- inisialisasi ADC
- Antarmuka inisialisasi SPI
- Dua Wire Interface inisialisasi
- Antarmuka inisialisasi BISA
- I2C Bus, LM75 Sensor Suhu, DS1621 Thermometer / Thermostat dan PCF8563, PCF8583, DS1302, DS1307 Real Time Jam inisialisasi
- 1 Wire Bus dan DS1820/DS18S20 inisialisasi Sensor Suhu
- inisialisasi modul LCD.

## 2.5. Sistem Instalasi

Anda dapat memperoleh file instalasi CodeVisionAVR dengan cara mendownload pada situs pembuatnya yaitu HP InfoTech di http://www.hpinfotech.com. File yang dapat didownload adalah tipe evaluation yang artinya mempunyai keterbatasan, salah satunya adalah ukuran program

yang dapat dikompilasi terbatas. Contoh yang digunakan adalah CodeVisionAVR versi 1.25.3

### 2.5.1. Instalasi CodeVision

- Jalankan Setup.exe dengan men-doublé klik pada mouse


Gambar 2.1 Ikon file setup.exe

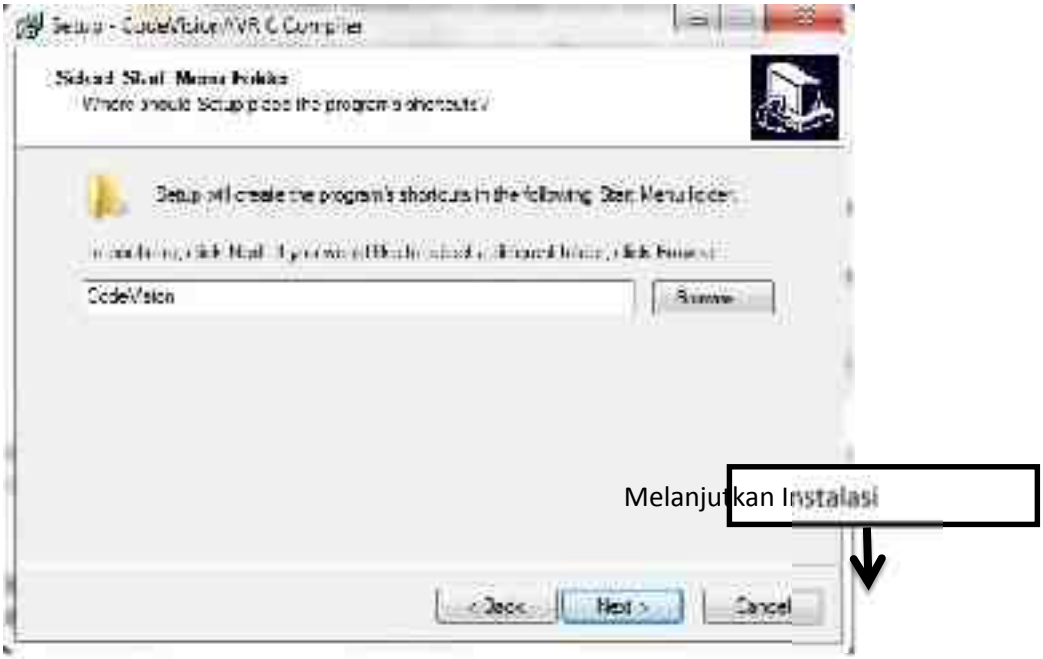
- Tutup Aplikasi yang sedang berjalan, kemudian Pilih *Next*. Untuk melanjutkan instalasi CodeVisionAVR.


Gambar 2.2. Klik tombol next

- Kemudian letakkan instalasi CodeVisionAVR yang diinginkan. Klik *Browse* untuk meletakkan file instalasi CodeVisionAVR. Default instalasi CodeVisionAVR di c:\cvavr , pilih *Next* untuk melanjutkan instalasi CodeVisionAVR.

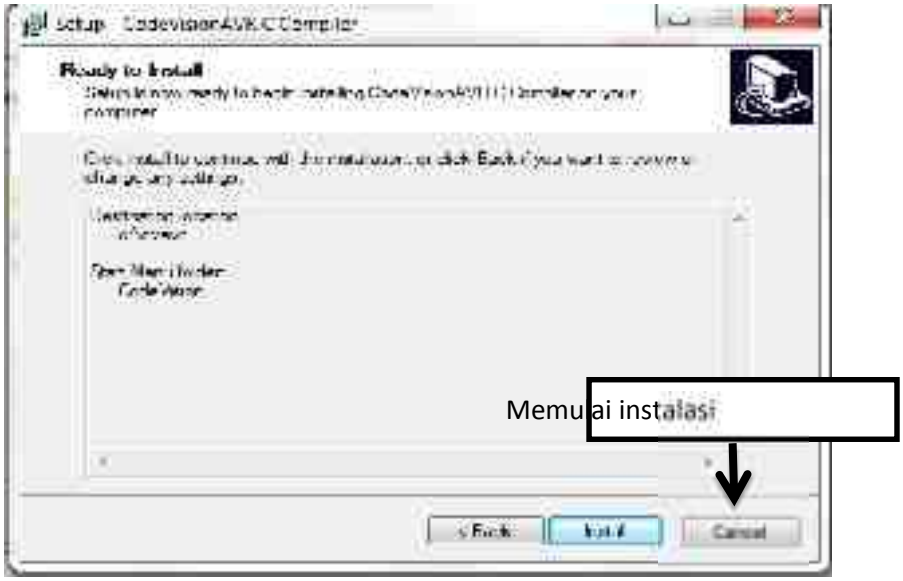

Gambar 2.3. Menentukan lokasi tujuan

- Tahap selanjutnya, Pemberian nama untuk folder start menú. Secara default nama yang tercantum adalah *CodeVision*. Lalu klik *Next* untuk melanjutkan instalasi.



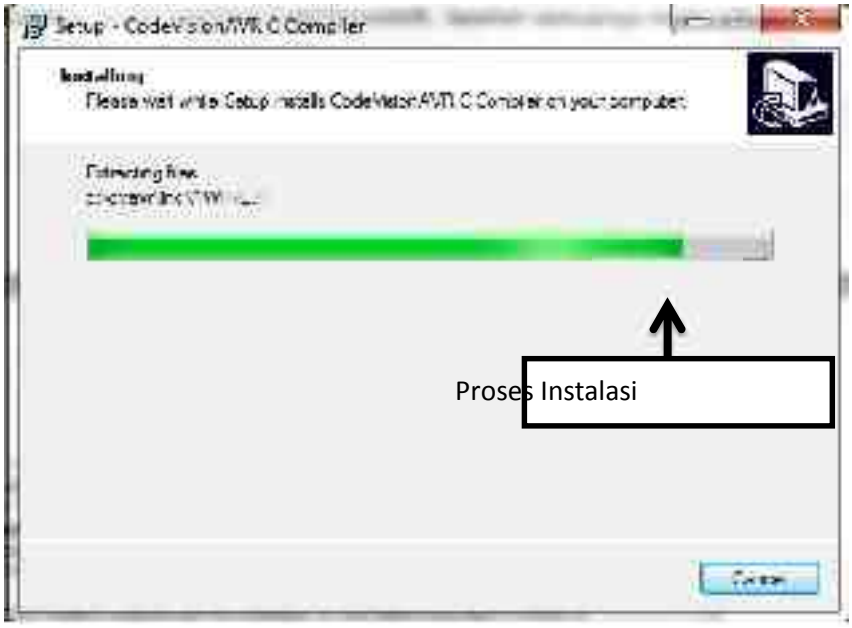

Gambar 2.4 Nama folder pada Start Menu

- Klik *Install*, untuk melakukan proses instalasi CodeVisionAVR. Setelah semuanya telah selesai dilengkapi.




Gambar 2.5 Nama folder pada Start Menu

- Proses Instalasi sedang dilakukan.




Gambar 2.6 Proses instalasi sedang berlangsung

- Setelah selesai proses instalasi telah selesai, lalu klik *Finish*

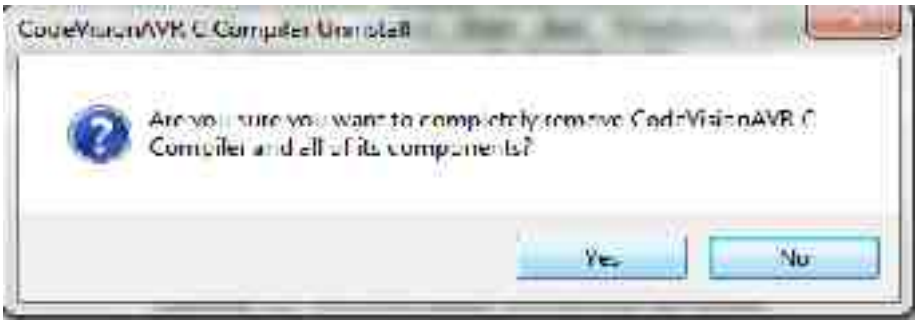

Gambar 2.7 Proses instalasi selesai

## 2.5.2. Un-Install CodeVisionAVR

Bila suatu saat Anda tidak membutuhkan aplikasi CodeVisionAVR, Anda dapat membuang hasil instalasi dari komputer Anda. Pada menu Start dari Windows, klik *shortcut* "Uninstall CodeVisionAVR C Compiler Evaluation" seperti pada **Gambar 2.8**.


Gambar 2.8 *Shortcut* untuk melakukan un-install

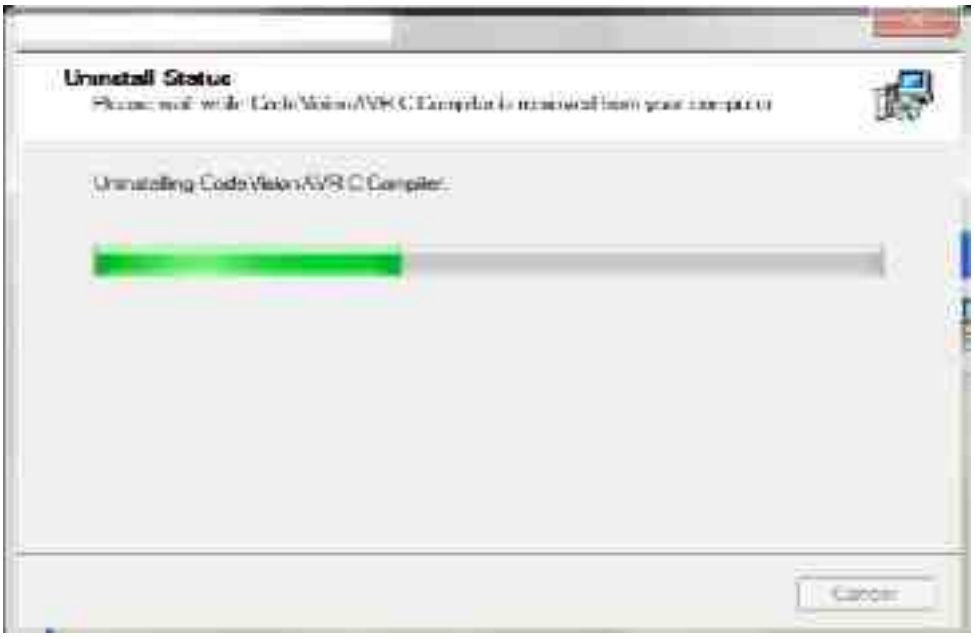
Maka kotak dialog seperti pada **Gambar 2.9** akan muncul untuk menanyakan keseriusan Anda. Klik tombol Yes untuk membuang aplikasi tersebut dari komputer Anda.



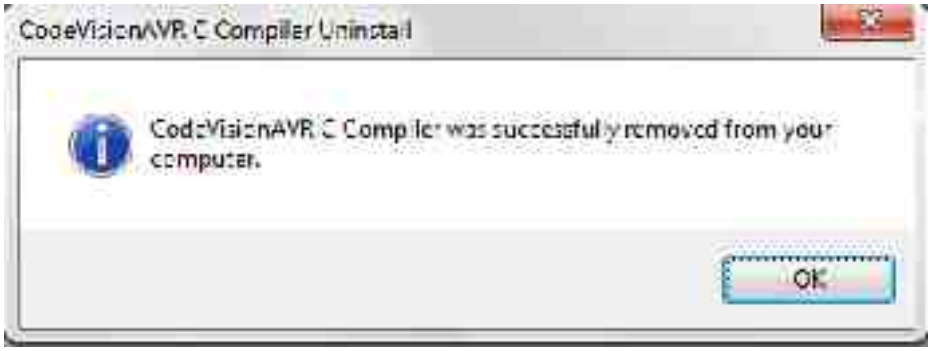

Gambar 2.9 Yakin akan membuang aplikasi dari computer

Berikutnya proses pembuangan aplikasi berlangsung seperti ditunjukkan oleh **Gambar 2.10.**

Lalu kotak dialog seperti **Gambar 2.11** akan muncuk, klik OK untuk menutup prosespembuangan.




Gambar 2.10 Proses un-install sedang berlangsung




Gambar 2.11 Proses selesai

Proses pembuangan tersebut biasanya tidak bersih, artinya masih ada file yang tertinggal. Anda dapat melanjutkan dengan melakukan proses *delete* secara manual menggunakan aplikasi windows explorer.

## 2.6 Membuat Project dengan CodeVisionAVR

Pada penjelasan berikutnya, sebagai contoh digunakan modul AVR yang mempunyai hubungan sebagai berikut:

- PortA terhubung dengan 8 buah LED dengan operasi aktif high
- PortB terhubung dengan 8 buah saklar dengan operasi aktif high
- PortC terhubung dengan LCD alphanumeric 16 kolom x 2 baris

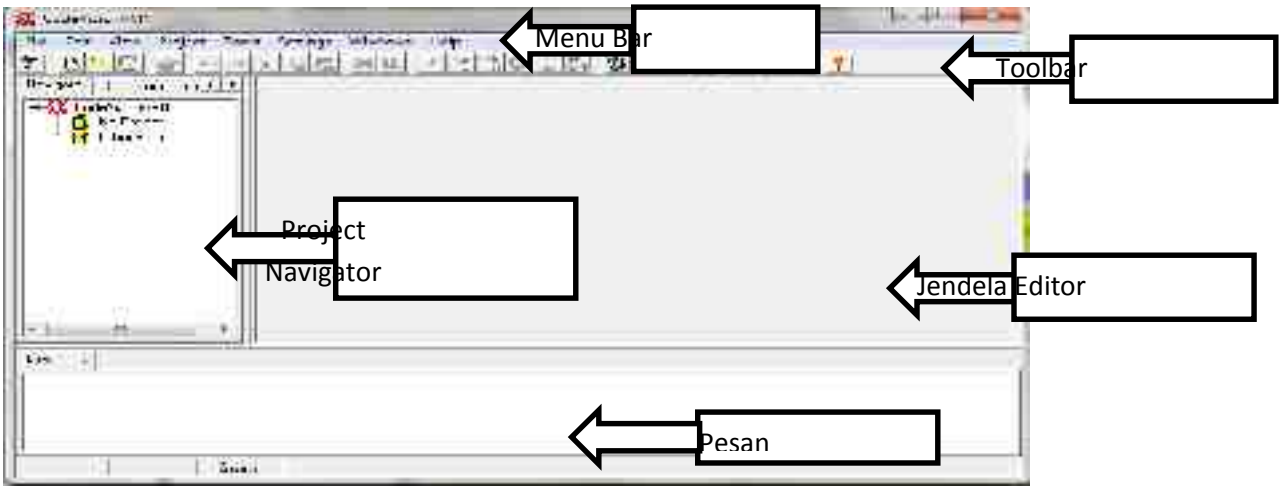
Jalankan aplikasi CodeVisionAVR dengan cara melakukan klik ganda pada *shortcut* ikonCodeVisionAVR yang terbentuk pada Desktop.

Gambar 2.12 Ikon CodeVisionAVR pada Desktop

Sebuah *Splash Screen* akan muncul seperti ditunjukkan oleh **Gambar 2.13**. Informasi tentangversi yang dipakai.

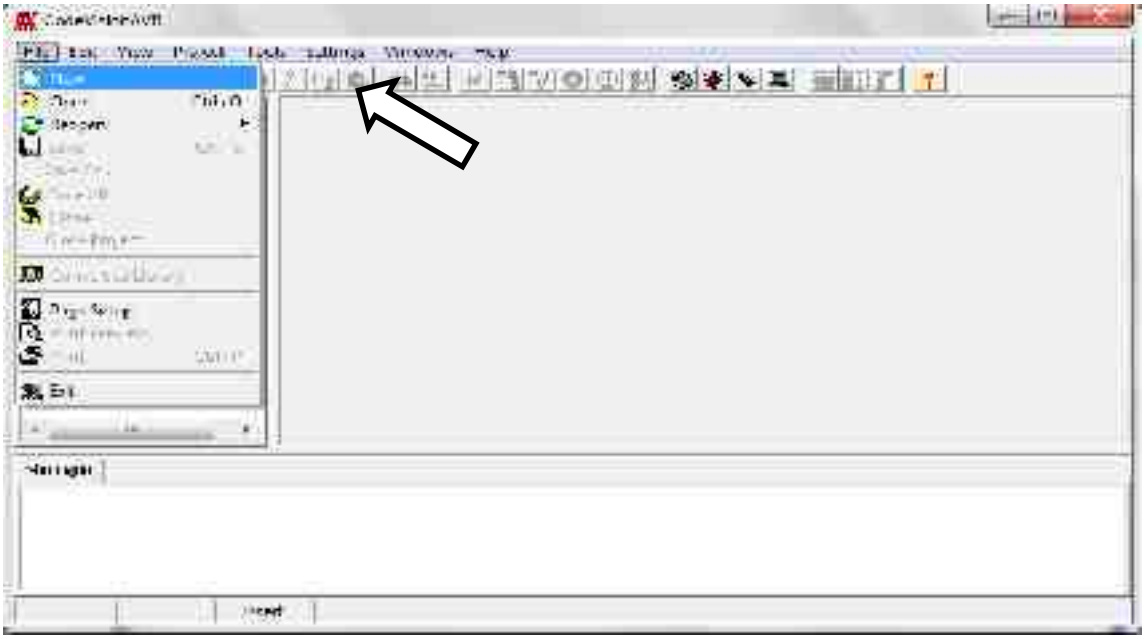

Gambar 2.13 Tampilan *Splash Screen*

Beberapa detik kemudian IDE dari CodeVisionAVR akan muncul seperti yang ditunjukkan oleh **Gambar 2.14**.




Gambar 2.14 IDE CodeVisionAVR

Untuk memulai membuat *project* baru, pada menu bar, pilih File >New, seperti yang ditunjukkan oleh **Gambar 2.15**

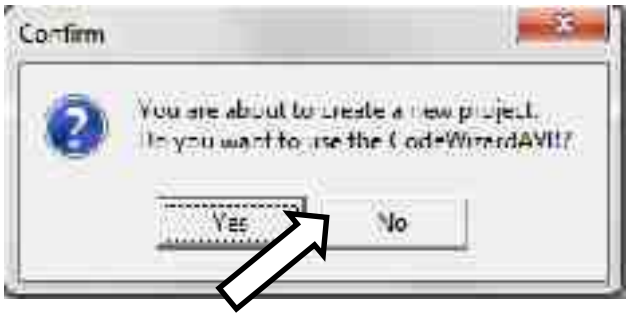


Gambar 2.15 Membuat *file* baru

Anda harus membuat sebuah project sebagai induk desain dengan memilih Project, lalu klik tombol OK seperti pada **Gambar 2.16**.

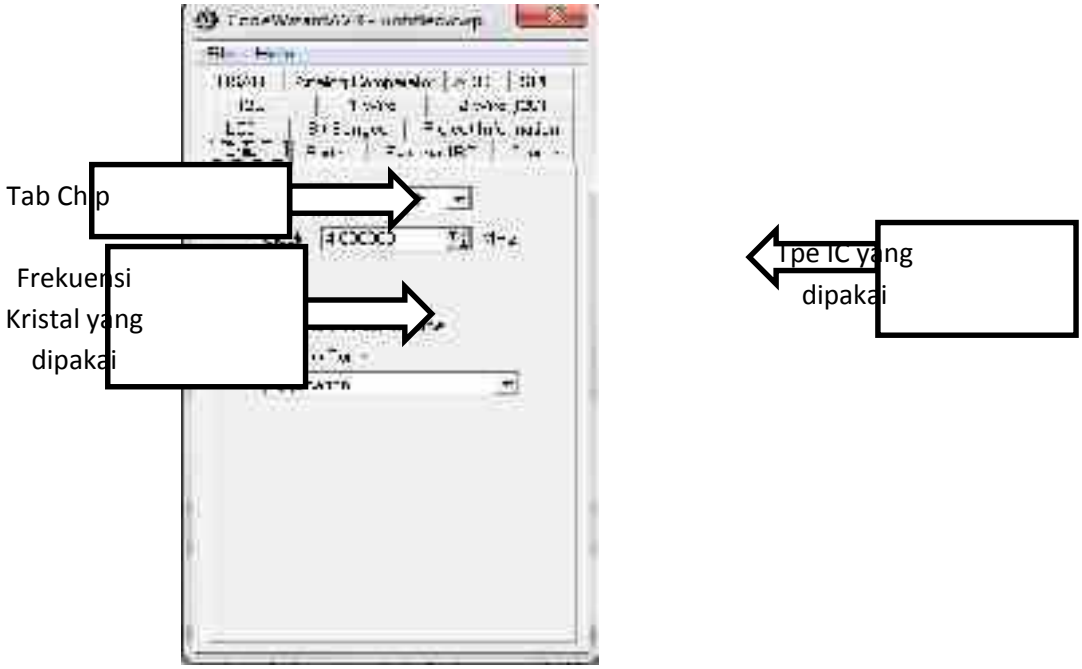

Gambar 2.16 Membuat *project* baru

Berikutnya Anda akan ditanya apakah akan menggunakan CodeWizardAVR. Tentu saja lebih menyenangkan bila Anda memilih jawaban "ya" dengan cara menekan tombol Yes seperti pada **Gambar 2.17**.



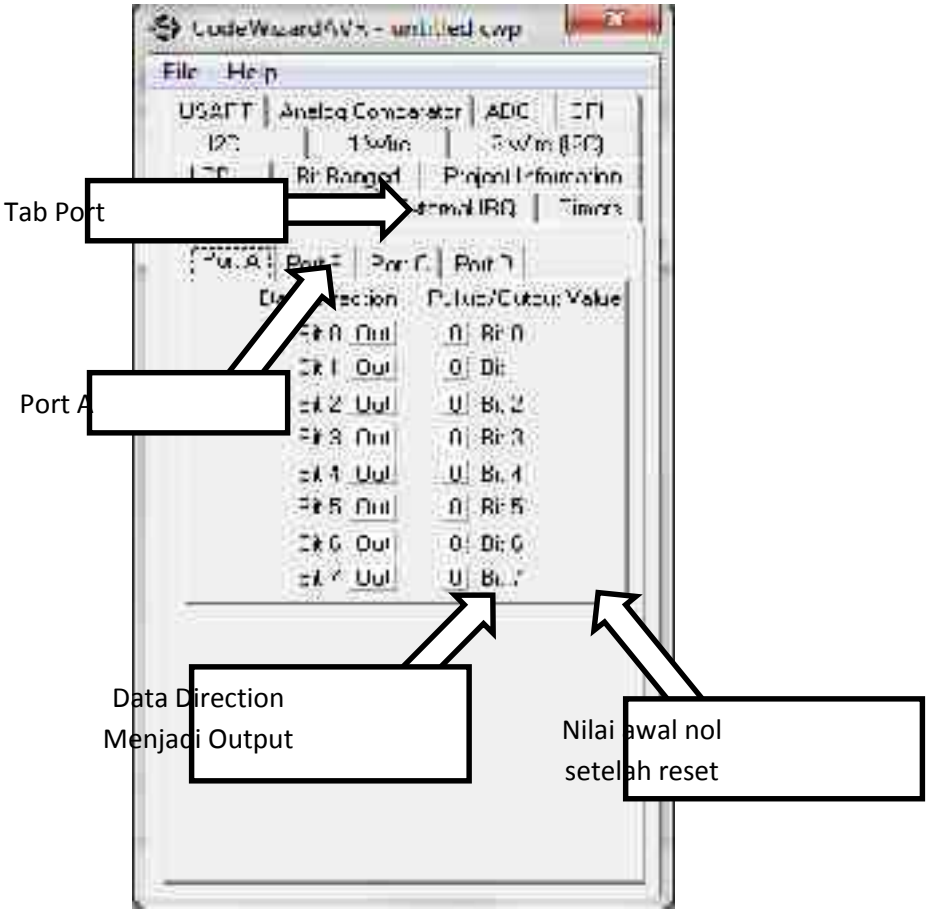

Gambar 2.17 Memilih untuk menggunakan CodeWizardAVR

Tampilan CodeWizardAVR yang sederhana namun lengkap ditunjukkan oleh **Gambar 2.22**. Pilih Chip dengan IC yang Anda gunakan. Sebagai contoh Anda memilih Chip ATmega8535. Tab-tab pada CodeWizardAVR menunjukkan fasilitas yang dimiliki oleh chip yang Anda pilih. Cocokkan pula frekuensi kristal yang Anda gunakan pada bagian Clock. Pengisian frekuensi clock digunakan oleh software untuk menghitung rutin-rutin seperti *delay* agar diperoleh perhitungan yang cukup akurat.

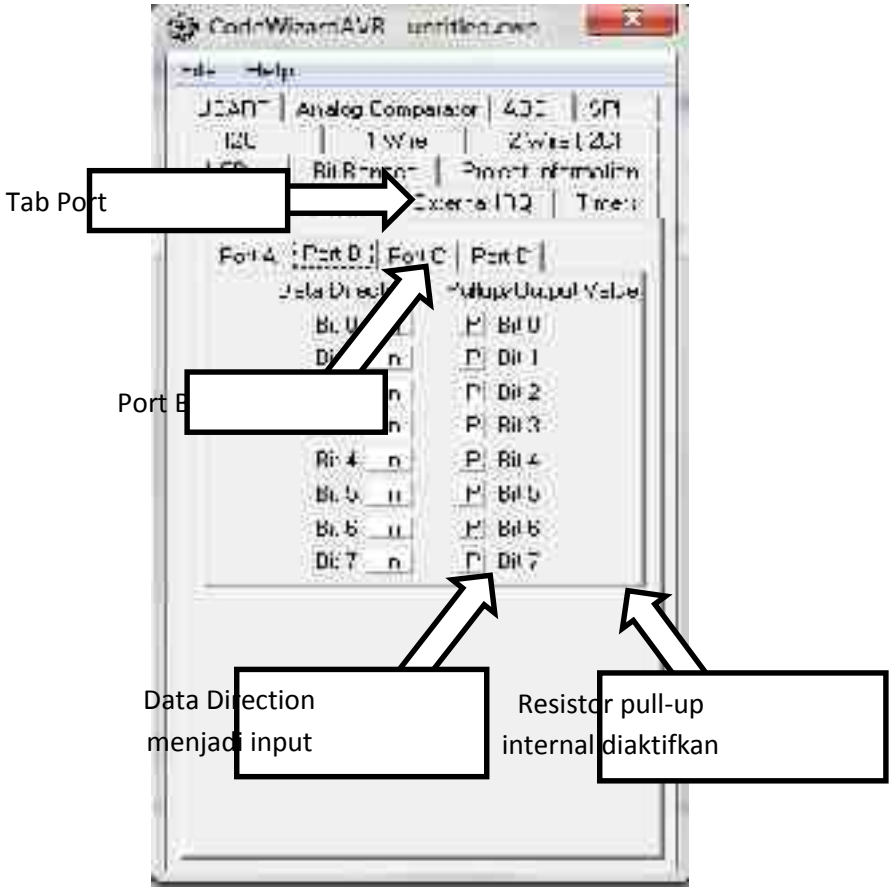



Gambar 2.18 CodeWizardAVR pada tab Chip

Berikutnya Anda akan menginisialisasi Port A yang terhubung dengan LED. LED merupakan modul output. Pada tab Port bagian Port A, ubah bagian Data Direction menjadi OUT dengan nilai output sama dengan 0 seperti pada **Gambar 2.19**. Artinya Port A digunakan sebagai port output dengan nilai awal nol setelah kondisi reset. Kemudian lakukan inisialisasi Port B seperti pada **Gambar 2.20**. Port B tersambung dengan saklar sebagai modul input. Pada sub-tab Port B, yakinkan Data Direction pada posisi IN dengan resistor pullup internal yang disingkat dengan huruf P. Dengan mengaktifkan resistor pull-up internal, Anda tidak perlu menambahkan resistor pull-up pada saklar.

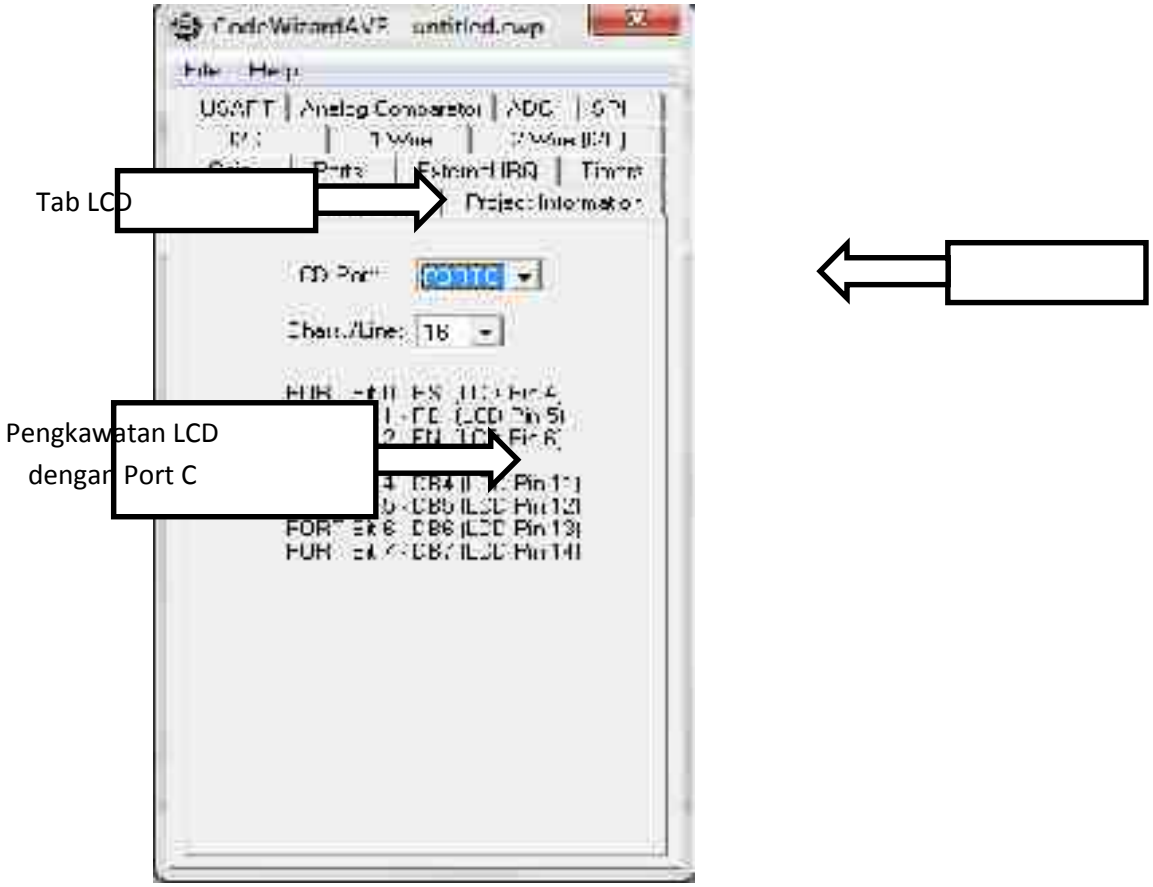



Gambar 2.19 Seting Port A sebagai pin output



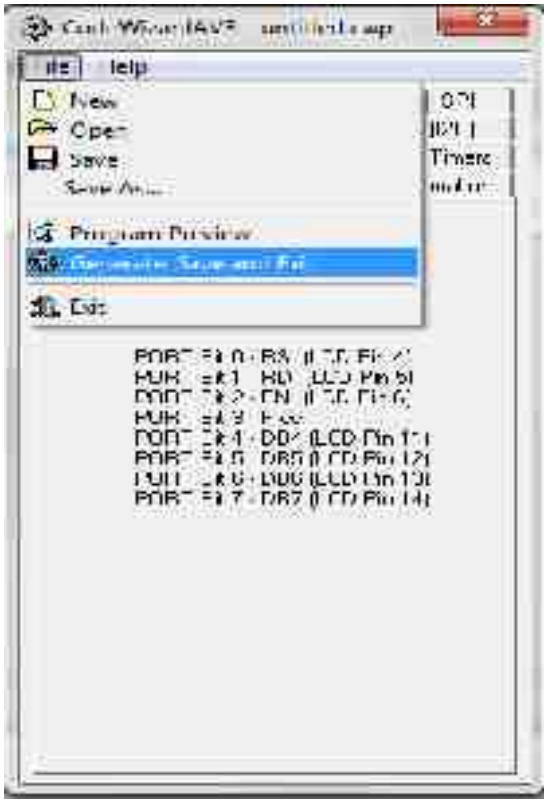

Gambar 2.20 Seting Port B sebagai pin input dengan pull-up resistor

LCD alphanumerik yang dihubungkan dengan Port C haruslah mempunyai pengkawatan seperti yang ditunjukkan oleh **Gambar 2.21**. Pada tab LCD, pilihlah Port C.



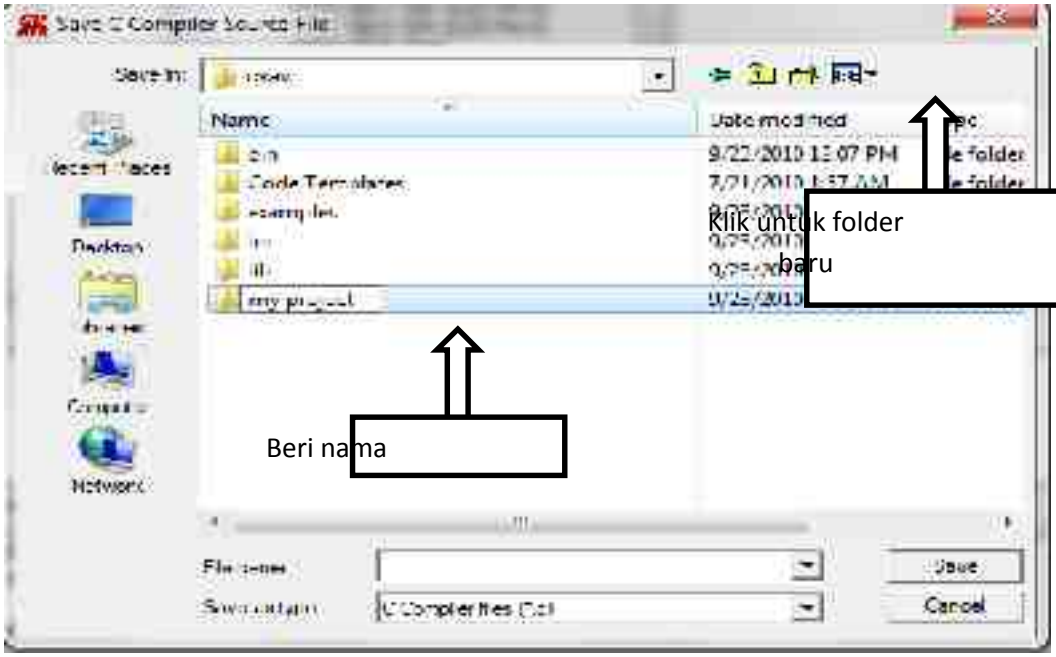

Gambar 2.21 Seting LCD pada Port C

Karena pada contoh ini tidak digunakan fasilitas lain maka seting CodeWizardAVR siap disimpan dalam file. Pada menu CodeWizardAVR, pilih File >Generate, Save and Exit, seperti yang ditunjukkan oleh **Gambar 2.22**.



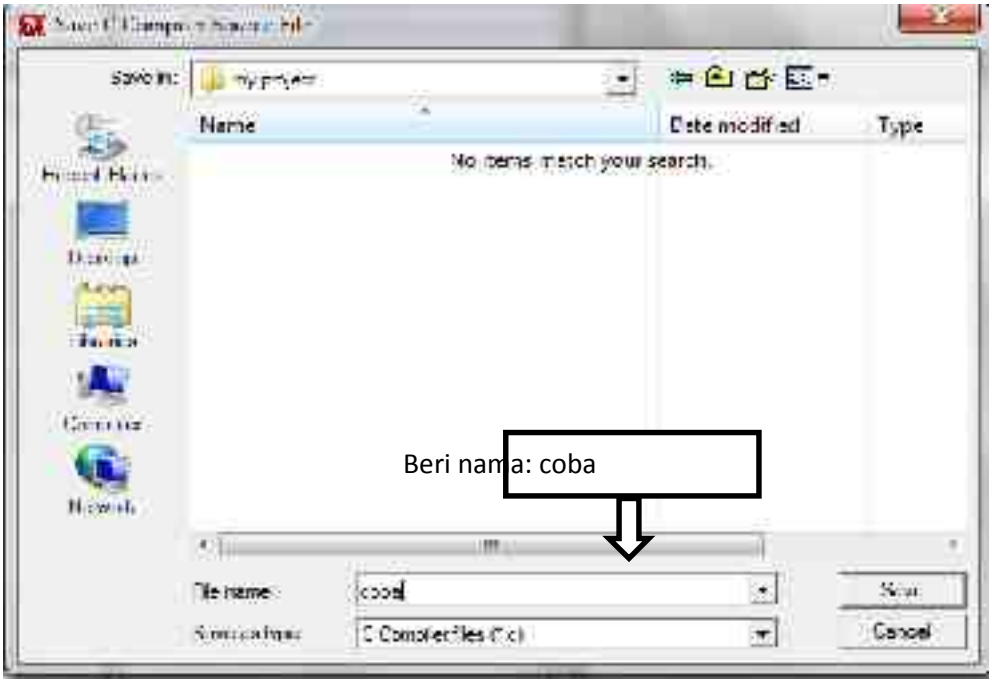

Gambar 2.22 Menyimpan setting

Agar file yang dihasilkan tidak berantakan, buatlah sebuah folder baru, misalnya folder bernama “my project”, seperti yang ditunjukkan oleh **Gambar 2.23**.



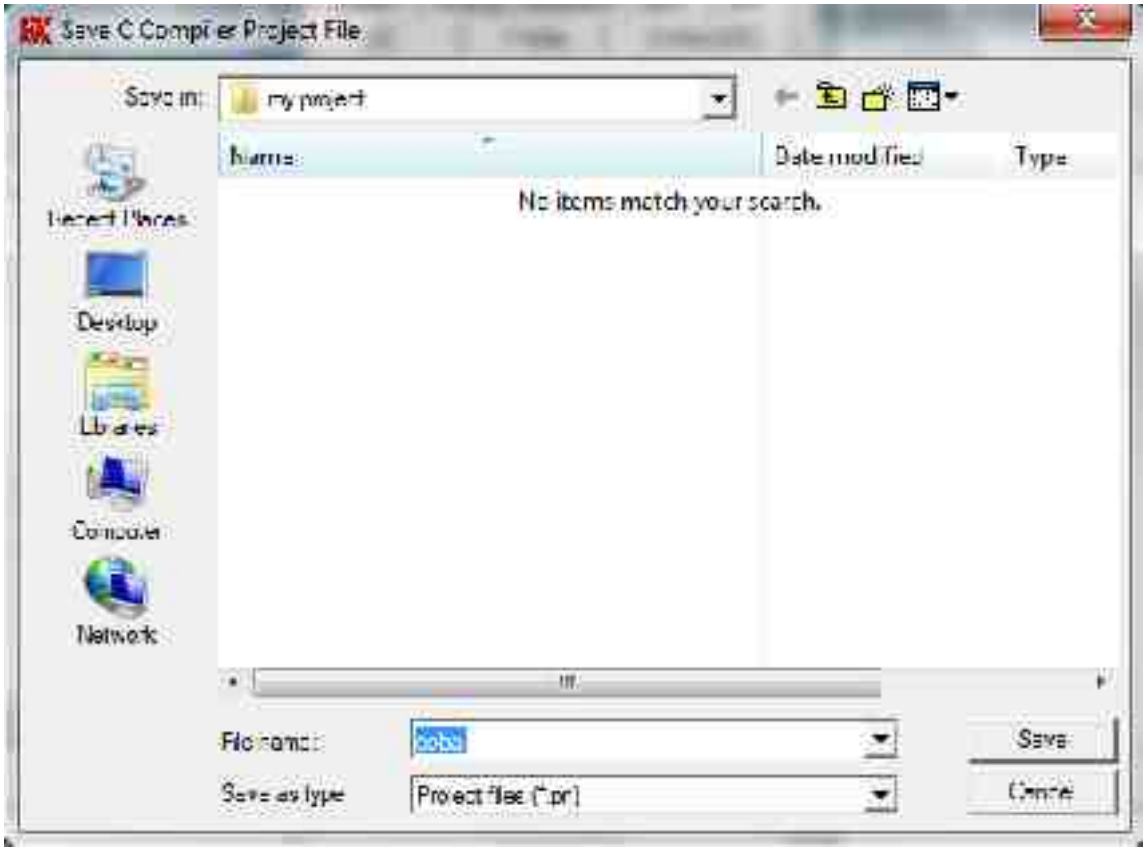

Gambar 2.23 Membuat folder baru

Kemudian masuk kedalam folder tersebut untuk menyimpan file-file yang dihasilkan oleh CodeWizardAVR. Yang pertama Anda diminta untuk memberikan nama file C yang dihasilkan. Misalnya beri nama “coba”, lalu klik tombol Save. Lebih jelas pada **Gambar 2.24**. File tersebut nantinya akan mempunyai akhiran .C.



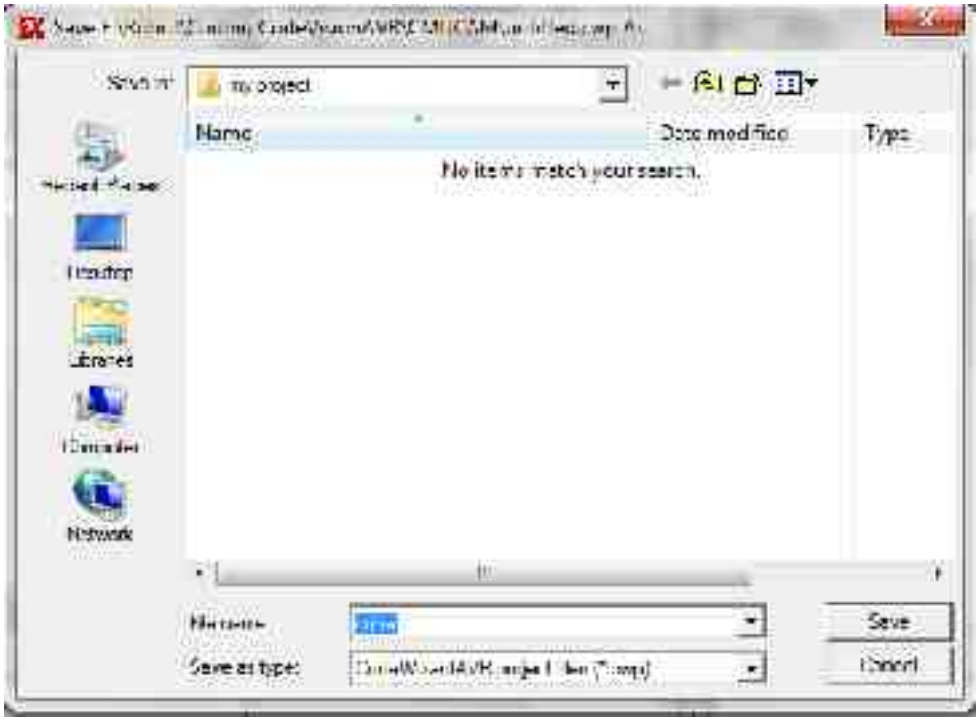

Gambar 2.24 Menyimpan file pertama

Yang kedua Anda diminta untuk memberikan nama file project yang dihasilkan. Misalnya beri nama “coba”, lalu klik tombol Save. Lebih jelas pada **Gambar 2.25**. File tersebut nantinya akan mempunyai akhiran .prj.



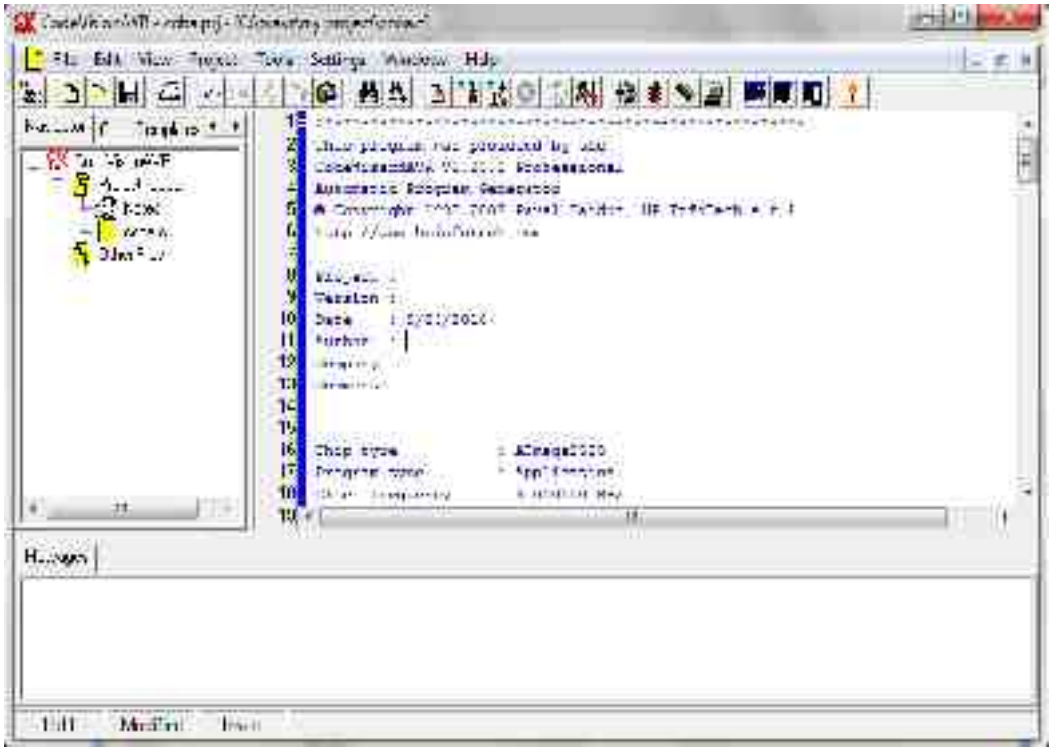

Gambar 2.25 Menyimpan file kedua

Yang terakhir Anda diminta untuk memberikan nama file project CodeWizard yang dihasilkan. Misalnya beri nama “coba”, lalu klik tombol Save. Lebih jelas pada **Gambar 2.26**. File tersebut nantinya akan mempunyai akhiran .cwp.




Gambar 2.26 Menyimpan file ketiga

Setelah ketiga file disimpan maka pada Project Navigator akan muncul nama project beserta file C-nya. Secara bersamaan isi file C akan dibuka pada jendela editor seperti ditunjukkan oleh **Gambar 2.27**.



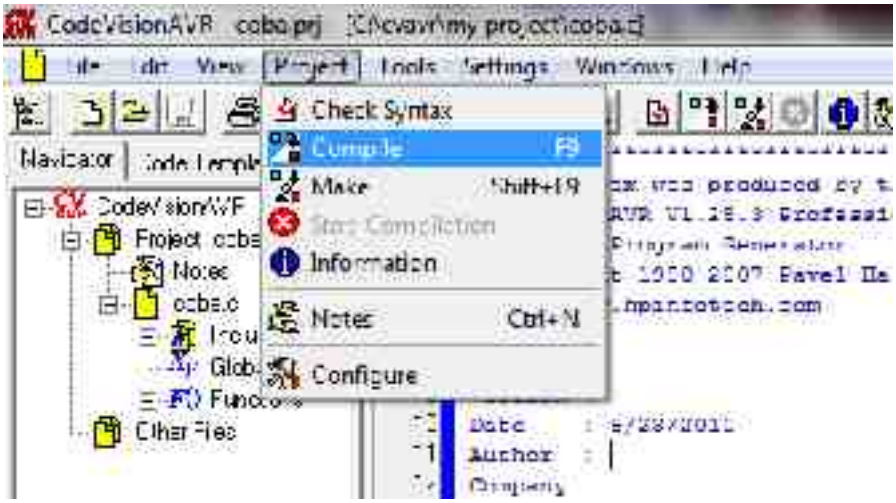
Gambar 2.27 Project baru telah siap dalam hitungan detik

## 2.7 Kompilasi C pada CodeVisionAVR

Kompilasi C pada CodeVisionAVR merupakan proses pengecekan setiap sintax program di tiap-tiap baris yang berisikan sekumpulan sintax program. Hasil kompilasi menentukan error tidaknya program yang telah dibuat. Bila terjadi kesalahan pada program, maka codevisionAVR akan memberikan pesan error pada baris yang terdapat kesalahan.

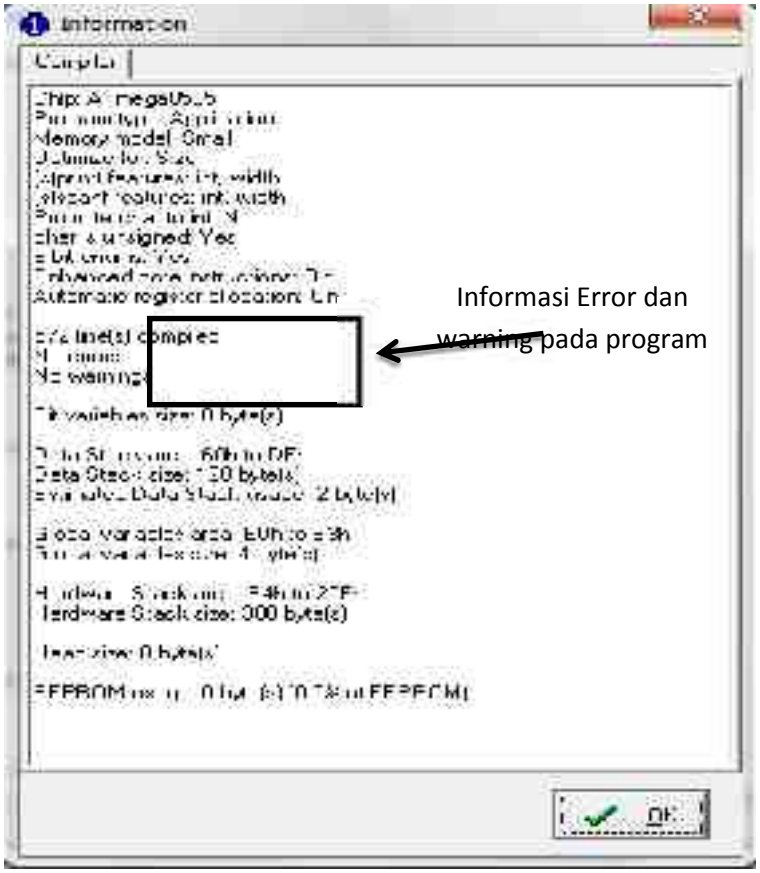
Proses kompilasi pada codevisionAVR berfungsi untuk menghasilkan file assembler dan hexa yang nantinya di-*upload* kedalam mikrokontroler AVR jika program tidak terdapat pesan kesalahan (*error*)dan pesan peringatan (*warnings*).

Untuk dapat meng-kompilasi program pada codevisionAVR dapat menekan pada keyboard *F9*atau klik icon ( ) pada toolbar. Atau klik *Project>>Compile* . Seperti gambar berikut.



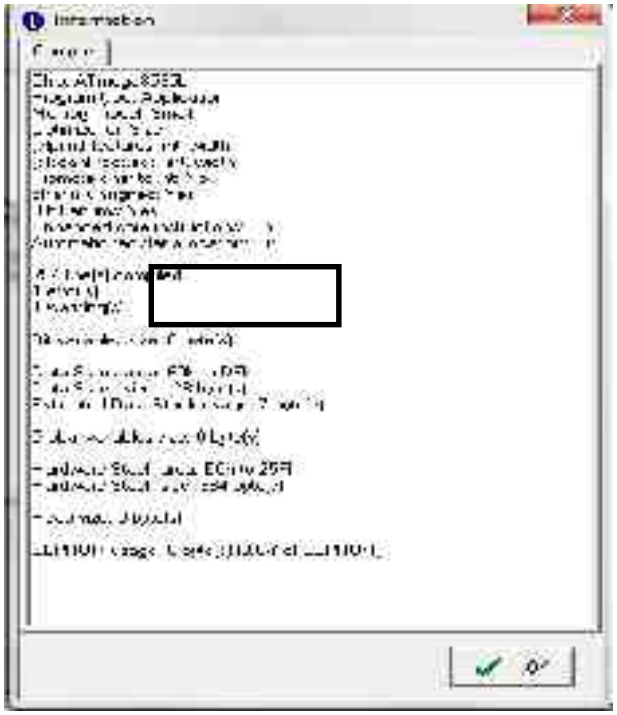

Gambar 2.28 mengkompilasi pada CodeVisionAVR

Setelah program telah di kompilasi maka akan muncul tampilan seperti pada **Gambar 2.29**. Jika tidak terjadi *error dan* warning, maka informasi hasil kompilasi akan menampilkan pesan *No errors* dan *No warnings*.



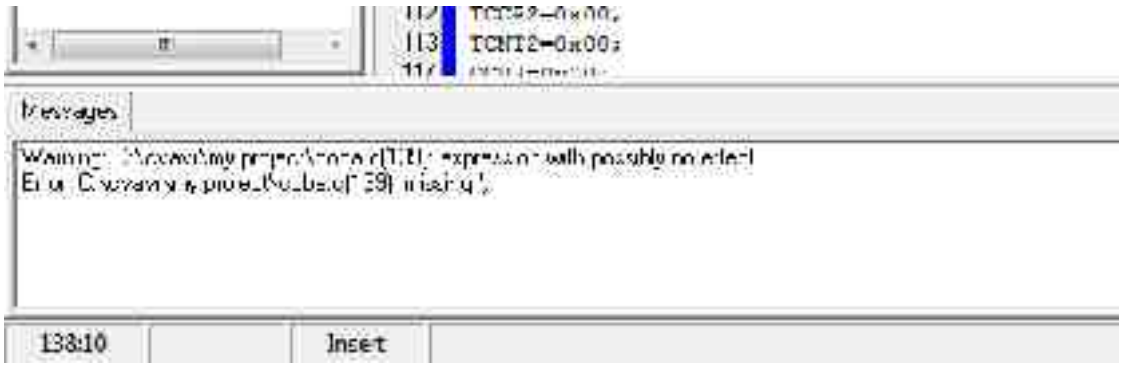

Gambar 2.29 Informasi hasil kompilasi

Kompilasi menghasilkan informasi mikrokontroler yang dipakai, tipe program yang dipakai, model memori yang dipakai, dan kelengkapan yang lainnya. Jika dilihat pada gambar diatas, terdapat informasi kesalahan pada syntax. CodeVisionAVR akan mengecek semua baris-baris program yang telah dibuat, jika ditemukan kesalahan maka CodeVisionAVR secara otomatis memberitahukan berapa banyak kesalahan dan peringatan yang ada. Banyaknya kesalahan pada baris program akan diberitahukan melalui jendela pesan. Berikut contoh jika terjadi *errors* dan *warnings* pada program.



Gambar 2.30 Informasi *Errors* dan *warnings*

Pada contoh gambar diatas, CodeVisionAVR memberikan suatu informasi errors dan warnings pada program. Jika dilihat pada gambar, disitu terdapat 1 *errors* dan 1 *warnings*. Berarti program yang telah dibuat memiliki suatu kesalahan dan CodeVisionAVR memperingati bahwa terdapat baris program yang belum difungsikan. Jika dilihat dari jendela pesan seperti pada gambar dibawah ini.



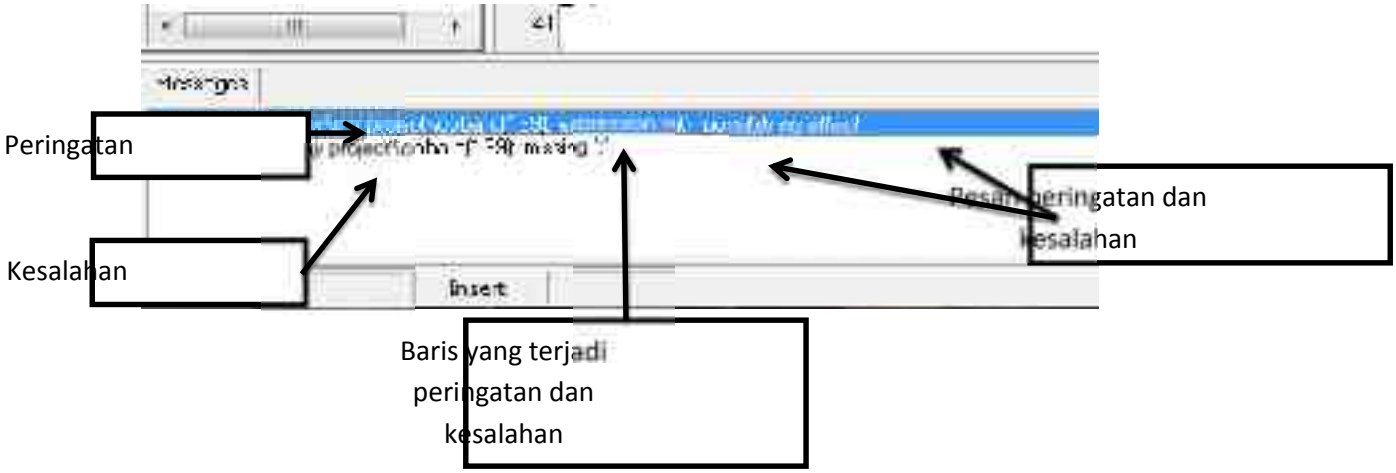

Gambar 2.31 Jendela pesan

Pada gambar diatas, terlihat pesan kesalahan dan peringatan pada jendela pesan. Pada peringatan, terdapat sintax yang apabila dijalankan nantinya dalam Chip mikrokontroler AVR, sintax program tersebut tidak akan berfungsi apa-apa. Pada pesan error, terjadi kesalahan pada baris program karena ada salah satu sintax yang belum terdapat titi-koma (";").

## 2.8 Debug

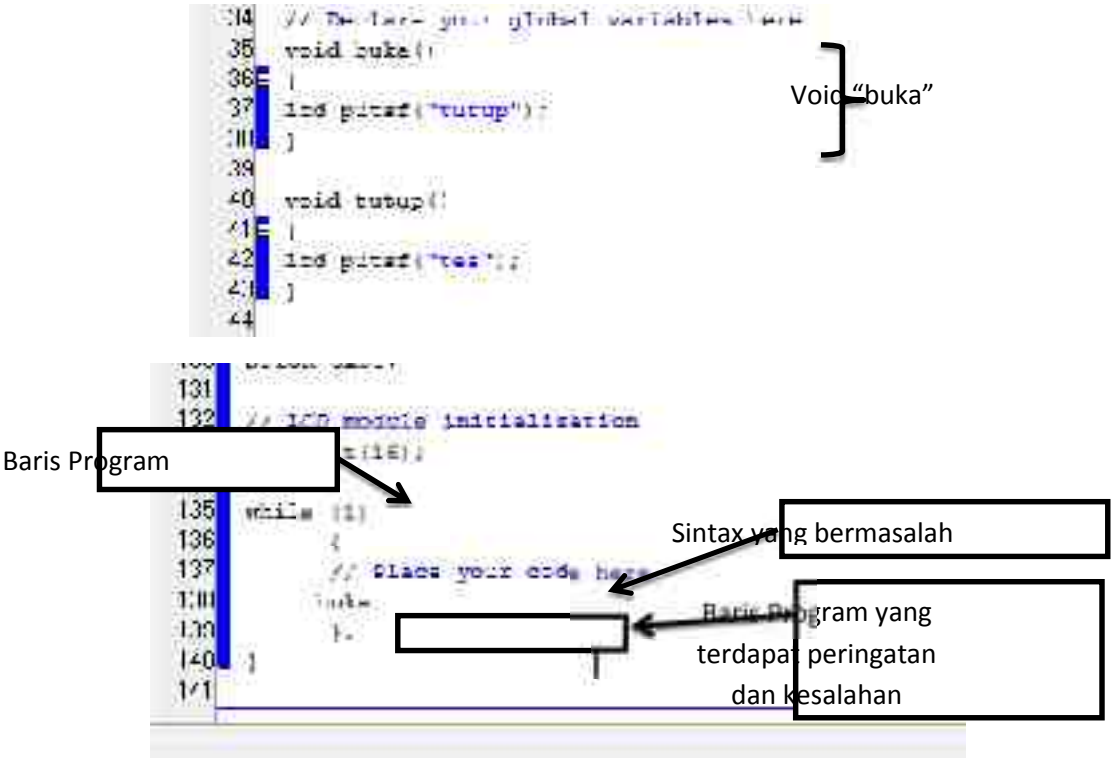
Debugging adalah sebuah metode yang dilakukan oleh para pemrogram dan pengembangan perangkat lunak untuk menganalisa alur kerja program, mencari dan mengurangi bug, atau kerusakan didalam sebuah program komputer atau perangkat keras sehingga perangkat tersebut bekerja sesuai dengan harapan. Debugging cenderung paling rumit ketika beberapa sub sistem lainnya terikat dengan ketat dengannya, mengingat sebuah perubahan disatusisi, mungkin dapat menyebabkan munculnya bug lain didalam subsistem lainnya.

Pada sub-bab ini akan di jelaskan mengenai mengatasi masalah dalam memrogram mikrokontroler AVR dengan CodeVisionAVR. Dalam menggunakan CodeVisionAVR, bila ternyata program yang telah dibangun memiliki suatu error sintax, secara otomatis CodeVisionAVR tidak dapat meng-*upload* program tersebut kedalam mikrokontroler AVR dan menampilkan pesan error pada jendela pesan. berikut cara mengetahui keberadaan error program pada CodeVision.




Gambar 2.32 Pesan peringatan dan kesalahan pada jendela pesan

Gambar diatas dapat diartikan, terjadi peringatan (*warnings*) dengan nama file *coba.c* pada baris 139 pesan peringatan adalah ekspresi dengan kemungkinan sintax tidak akan ada pengaruh apa-apa pada program dan terjadi kesalahan dengan nama file *coba.c* pada baris 139 pesan kesalahan adalah ada satu sintax yang belum disertakan titik-koma " ; ". Untuk mengetahui baris tersebut bisa dapat dilihat pada gambar berikut.




Gambar 2.33 Baris program yang terjadi kesalahan sintax

Pada baris 139 hanya terdapat sebuah tanda " } " dan " ; " itu di artikan CodeVisionAVR mengecek program dan telah terhenti pengecekannya pada baris 139 tersebut. Karena pada baris sebelumnya terdapat sintax yang bermasalah. Sintax tersebut bermaksud memanggil void yang berada di baris 35 namun prosedur pemanggilan void salah yang benar yaitu "buka();". Dalam prosedur pemanggilan void harus di akhiri " () " jika tidak terdapat itu, maka pemanggilan tersebut tidak berpengaruh apa-apa atau program tidak menganggap sintax tersebut suatu pemanggilan void. Itu sebabnya pada jendela pesan menampilkan "*expression with possibly no effect*" dan untuk setiap sintax pada bahasa C harus diakhiri dengan tanda " ; ". Itu yang menyebabkan jendela pesan menampilkan " *missing ';'* "

## 2.9 Downloader Dan Upload

Program yang telah di-buat agar bisa dimasukkan kedalam sebuah chip diperlukan suatu alat dan beberapa proses pada CodeVision AVR untuk dapat program dapat di jalankan pada chip.

### 2.9.1 Downloader

Untuk dapat memasukkan program dari computer ke dalam sebuah mikrokontroler berjenis AVR dibutuhkan sebuah alat tambahan yaitu ISP Programmer kabel (In-Sytem programmer) atau sering disebut *Downloader*. Isp Programmer kabel menggunakan port yang terdapat pada computer. Ada beberapa jenis port yang bisa digunakan. Diantaranya port Paralel LPT(DB25), port serial (DB9), dan *Universal Serial Bus* (*USB*). Berikut Kabel Isp programmer yang kompatibel dengan CodeVision.

**a. Kanda Systems STK200+/300**

Gambar 2.34 DT-HiQ AVR In System Programmer

- Menggunakan Parallel (LPT) Port Interface
- Kompatibel dengan software yang mendukung AVR ISP In-System Programmer Kanda System STK200/300 seperti CodeVisionAVR©
- Konektor standar Atmel
- Beroperasi pada tegangan 3,3 - 5 V
- Port tidak terbebani saat "Running"
- Tersedia Pin Output untuk indikator pemrograman
- Sistem operasi komputer: Windows® 9x/NT™ 4/2000/XP

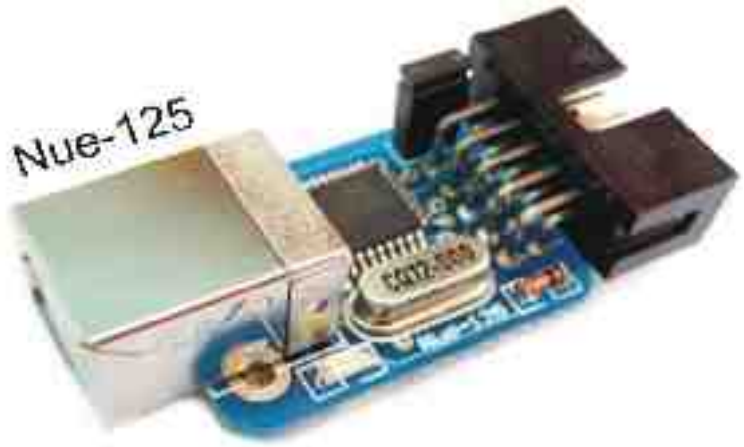
- Kompatibel dengan Atmel® Microcontroller seri AVR ISP, antara lain: AT90S1200, AT90S2313, AT90S2323(L), AT90S2343(L), AT90S8515, AT90S8535(L), ATmega8515(L), ATmega8535(L), ATmega16(L), ATmega162(L/U/V), ATmega169(L/V), ATtiny13, ATtiny22(L), ATtiny26(L)

**b. Atmel STK500/AVRISP**


Gambar 2.35 DT-HiQ AVR USB ISP

- Beroperasi pada tegangan target 2,7V sampai 5,5V.
- Antarmuka USB ke PC.
- Mengambil daya dari t*arget board*(50mA @ 5,5V). Tidak memerlukan catu daya tersendiri dan aman bagi PC jika terjadi hubungan singkat pada *target board.*
- Menggunakan protokol ATMEL STK500/AVRISP dengan *baud rate* 115200 bps.
- Kompatibel dengan perangkat lunak AVR Studio©, CodeVisionAVR©, AVRDUDE (WinAVR), BASCOMAVR ©, dan perangkat lunak lain yang mendukung protokol ATMEL STK500/AVRISP.
- Kompatibel dengan Windows® XP/Vista.

**c. Atmel AVRISP MkII (USB)**




Gambar 2.36 Nue-125

- Format file yang didukung adalah * .hex
- Target ISP untuk semua AVR dan MCS-51
- Kompatibel dengan Windows XP
- Kompatibel Software: AVR studio 4, CodeVision AVR dan software lainnya yang mendukung AVRISP MKII.
- Tidak membutuhkan catu daya tambahan dari luar
- Terdapat selector jumper untuk power board mikrokontroler AVR jika membutuhkan power dari USB
- Dilengkapi dioda pengaman sehingga tidak terjadi arus balik ke laptop/ komputer jika menggunakan power dari luar walau jumper dilepas nue-125 masih mengeluarkan tegangan namun dibawah 5volt dan arus sekitar 80mA.

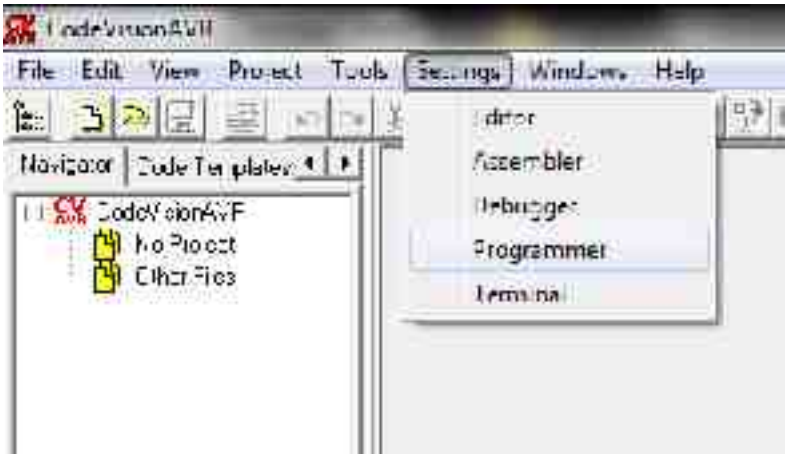
**d. Atmel AVRProg (AVR910)**

Gambar 2.37 ET-AVRProg mini

- Beroperasi dengan AVR Prog (termasuk dalam AVR Studio 4), CodeVisionAVR, AVRDUDE, Avr-OSP II dan Uisp
- Bekerja dalam banyak platform. Linux, Mac OS X dan Windows
- Menghubungkan dengan port USB computer
- Program mikrokontroler AVR melalui konektor ISP
- Plug langsung ke papan sasaran. Kabel ISP tidak diperlukan
- Mendukung Baca, Tulis, fungsi Menghapus dan Perlindungan
- Membutuhkan catu daya +5 V dari sasaran / papan induk
- Memiliki 10 pin ke 6 pin Adapter AVR ISP

**Pengaturan ISP Programmer kabel pada CodeVisionAVR**

Pilih setting kemudian pilih Programmer pada *menu bar* CodeVisionAVR. Untuk lebih jelasnya lihat gambar berikut.




Gambar 2.38 Pemilihan AVR Chip Programmer

Kemudian pilih **AVR Chip Programmer***Kanda System STK200+/300* dan **Printer Port***LPT1:378h*. Berikut pengaturan pada *programmer settings*.

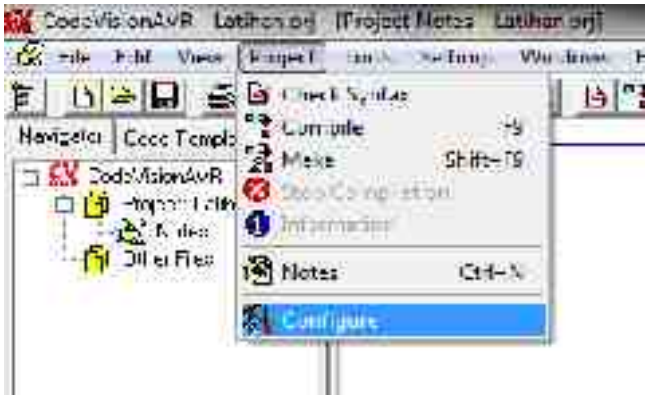

Gambar 2.39 Programmer Settings

### 2.9.2 Uploader

*Upload* merupakan tahap proses memasukkan program ke Chip setelah melewati proses *compile* dan program benar-benar sudah tidak terdapat kesalahan pada sintax program serta tidak terdapat pesan error. Jika terdapat

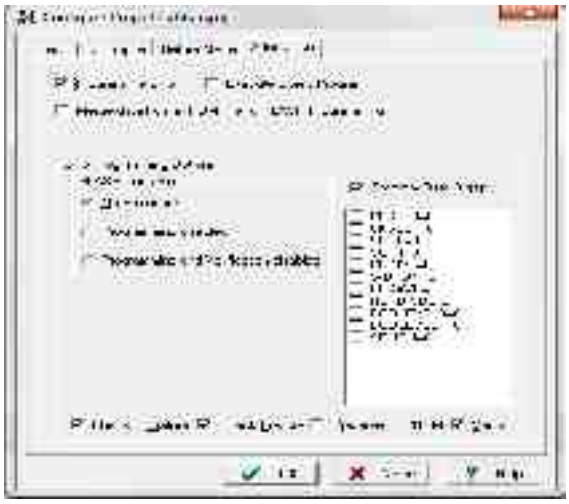
error maka secara otomatis program tidak dapat di-*upload* ke dalam chip. Berikut pengaturan pada CodeVision dalam meng-*upload* program ke dalam chip.

- Pilih Project kemudian Configure pada *Menu Bar* CodeVisionAVR



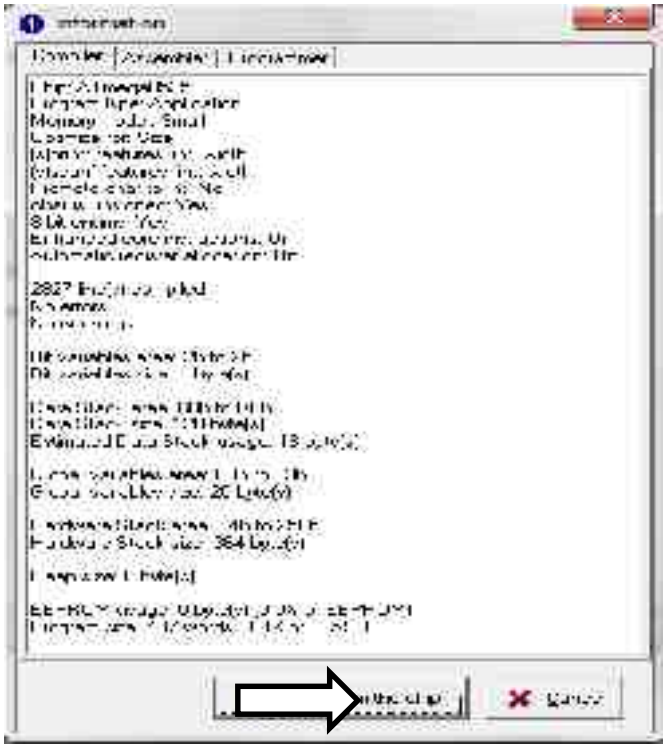

Gambar 2.40 Membuka Configure

- Pada Tab After Make beri centang (✓) pada Program the Chip



Gambar 2.41 Pemberian tanda centang pada form configure

Untuk meng-*upload* program tekan shift+F9 atau tekan ( ) pada toolbar CodeVisionAVR. Lalu pilih *Program the chip* untuk memulai upload program.



Gambar 2.42 Kotak dialog informasi hasil *make*

Apabila muncul kotak dialog seperti pada **Gambar 2.43** menandakan telah terjadi suatu hal yang menyebabkan proses transfer gagal. Penyebabnya adalah: suplai tegangan mikrokontroler dan programmer belum dinyalakan, tipe programmer tidak sama dengan yang digunakan, alamat port paralel tidak cocok, atau mikrokontrolernya rusak.


Gambar 2.43 Gagal melakukan transfer program

Bila kerusakan seperti yang ditampilkan oleh **Gambar 2.43** telah diperbaiki atau bila tidak ada kerusakan maka proses transfer atau yang umum disebut dengan proses download akan berlangsung seperti yang ditunjukkan oleh **Gambar 2.44**.


Gambar 2.44 Proses transfer ke mikrokontroler

## 2.10 Contoh-contoh Program

### Program 1

Program Untuk menyalakan LED pada PORTA.0 (PA0)

```
#include <mega8535.h>            //file definisi untuk
                                 //Mikrokontroler
                                 ATmega8535

// Declare your global variables here

void main(void)

{

// Declare your local variables here

PORTA=0x00;          //konfigurasi pull-up

DDRA=0x01;               //konfigurasi  PORTA.0  Sebagai
output

while (1)

      {

      // Place your code here

      };

}
```

**Program 2.**

Program Untuk menyalakan LED berkedip selama 0,5 detik.

Penambahan prosedur *delay* sebesar 500 ms (0,5 detik).

```
#include <mega8535.h>            //file definisi untuk
```

```
                                    //Mikrokontroler
                                    ATmega8535
#include <delay.h>          //Prosedur delay
// Declare your global variables here


void main(void)
{
// Declare your local variables here


PORTA=0x04;            //konfigurasi nilai awal output
PORTA.2
                    //berlogika ‘1’ sehingga LED tidak
                    //menyala
DDRA=0x04;        //konfigurasi PORTA.2 Sebagai output


//program akan berulang pada perintah dibawah ini
while (1)
      {
        PORTA.2=0;       //LED pada PORTA.2 menyala
        Delay_ms(500);         //delay 0,5 detik
        PORTA.2=1;       //LED pada PORTA.2 mati
        Delay_ms(500);         //delay 0,5 detik
      };
}
```

# LATIHAN

1. Buat Program untuk menyalakan 4 LED berkedip selama 2 detik!
2. Buat program untuk membaca penekanan saklar push-button pada PORTC kemudian dikeluarkan ke PORTA untuk menyalakan LED!
3. Buat Program untuk menampilkan karakter ke LCD!

# REFERENSI

http://id.wikipedia.org/wiki/debugging

Agilent.(1999). *Quadrature Decoder/Counter Interface ICs. Technical Data*, Agilent Technologies, Inc., http://www.semiconductor.agilent.com

Analog Devices. (1999). *ADXL105 Datasheet,* Analog Devices, Inc., http://www.analog.com

Applied Measurement. (1998). *Miniature Series LVDT: Displacement Transducer. AML/M Series Data sheet*, Applied Measurement, Ltd., http://www.appmeas.co.uk

Braunl, T. (2003.*Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.*BukuTeks. Berlin: Springer Verlag Berlin Heidelberg, Inc.

Dinsmore. (1999). *Datasheet Dinsmore Analog Sensor No. 1525,* Dinsmore Instrument Co., http://dinsmoregroup.com/dico

Honeywell.(2005). *Digital Compass Solution.* Sensor Product Datasheet, Honewell, Inc., http://www.magneticsensors.com

Precision Navigation. (1998). *Vector Electronic Module. Application Notes*, Precision Navigation, Inc., http://www.precisionnav.com

http://www.societyofrobots.com/actuators.shtml

http://ocw.gunadarma.ac.id/course/diploma-three-program/study-program-of-computer-engineering-d3/robotika/mekanika-robotika

# BAB III

# TEKNIK PERANCANGAN ROBOT

## 3.1. Pendahuluan

Sebelum membahas teori robotika secara mendalam, bab ini akan mengajak anda langsung menerjuni teknik disain robot dengan pendekatan praktis. Teori yang terlalu bersifat matematik yang senderung melemahkan semangat mahasiswa dan mereka yang belajar robotika untuk pertama kalinya akan dihindari dalam bahasan-bahasan di bab ini. Pertimbangannya, biasanya dalam benak mereka yang ingin menerjuni dunia robotika pertama kali adalah segera bereksperimen dengan bentuk fisik robot, segera mencipta sistem perangkat keras, dan segera memfungsikan robot dengan program-program aplikasi yang efektif yang sesuai dengan tujuan.

Dalam bab ini anda akan menjumpai bahasan-bahasan tentang :

- Prinsip dasar tentang teknik disain robot berorientasi fungsi,
- Teknik disain robot dengan pendekatan mekatronik, yaitu disain struktur mekanik, teknik transmisi tenaga, penerapan berbagai sensor dan aktuator dilengkapi bahasan tentang rangkaian interface, disain perangkat kontrol berupa rangkaian elektronik berbasis mikrokontroler dan pengembangan sistem berbasis komputer,
- Berbagai perangkat dan teknik sensor: sensor biner, analog, rotary/shaft encoder, kamera digital, dsb.,
- Teknik rangkaian signal conditioning: penggunaan amplifier, filter, parallel bus, komunikasi serial, dll.,
- Berbagai perangkat dan teknik aktuator : motor DC magnet permanen, motor DC stepper, motor DC brushless, motor DC servo, teknik Pulse Width Modulation, Direct Drive Motor, dan linear motor.