# Module Materi Robotik

Dibuat oleh ilham nurhikmat

## Perulangan (looping)



Gambar .contoh perulangan yang memiliki hasil mustahil

Sebuah robot mengerjakan tugasnya secara terus menerus dengan instruksi yang diberikan oleh pembuatnya. Kemampuan robot dalam melakukan tugasnya secara terus menerus ini dikarenakan oleh program didalamnya. Dalam program robot tersemat sebuah kode yang bertugas untuk mengulang satu atau beberapa baris perintah, dalam Bahasa C++ kita akan mengenal 3 bentuk perulangan yaitu, for, while dan do-while. Bagaimana cara kerjanya?, tetap baca modul ini ya!.

## Perulangan for

Dalam sebuah sistem, kita mungkin pernah mengalami untuk melakukan tugas secara rutin dengan jumlah tugas yang telah diketahui sebelumnya. Nah, dalam dunia robot, kita dapat melakukan hal yang sama dengan menggunakan keyword atau kata kunci for. Kata kunci for merupakan salah satu bentuk looping yang digunakan untuk mengulang sintaks atau baris printah pada Bahasa C/C++. Baris perintah yang diulang bisa satu atau lebih dari satu. Bentuk dari perulangan for adalah sebagai berikut:

for(inisialiasi; kondisi; inkrementasi/dekrementasi)

    // baris perintah 1 yang diulang

Contoh diatas merupakan bentuk perulangan for dimana baris perintah yang diulang hanya satu. Penulisanya tanpa menggunakan kurung kurawal ({ }} atau curly bracket. Bagaimana dengan baris perintah yang lebih dari satu?, perhatikan contoh berikut:

for(inisialiasi; kondisi; inkrementasi/dekrementasi){

    // baris perintah 1 yang diulang

    // baris peirntah 2 yang diulang

}

Contoh diatas merupakan bentuk perulangan for dimana baris peirntah yang diulang memiliki lebih dari satu kode baris perintah. Perhatikan pada kode diatas, posisi baris perintah 1 dan dua ada didalam kurung kurawal ({ }), ini menandakan bahwa baris perintah 1 dan 2 merupakan bagian dari for. Namun bentuk diatas juga bisa digunakan untuk 1 baris perintah saja. Tergantung kepada programmer untuk menggunakanya disesuaikan dengan kebutuhan dan kemudahaan dalam membaca Kembali kode sumber yang telah dibuat.

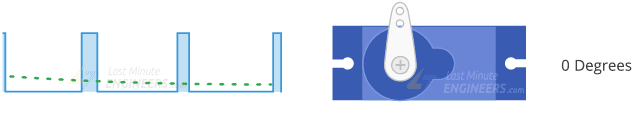
Nah, setelah kita mengenal bentuk perulangan, maka sekarang kita akan mencoba untuk menggunakanya dalam Arduino. Berikut adalah contoh perulangan for yang berfungsi untuk merubah posisi servo motor secara bertahap.

for(int i=0; i < 180; i++){

    servo.write(i); //merubah posisi servo motor berdasarkan nilai variable i

    delay(100); //kode didelay karena servo motor bergerak lebih lambat daripada sistem arduino

}



Gambar 2. ilustrasi servo yang digerakan secara beraturan dari 0 derajat sampai 180

Pertama variable i di inisialiasai didalam perulangan dengan nilai 0. Inisialisasi ini hanya dilakukan satu kali pada awal perulangan. Setelah itu, nilai variable i kemudian dibandingkan dengan nilai 180, yang kita ketahui sebagai batas dari servo (0 derajat sampai dengan 180 derajat), jika nilai variable i lebih kecil dari pada 180 maka akan menghasilkan nilai true dan jika nilai i sama atau lebih besar dari 180, maka akan memberikan nilai false. Selanjutnya adalah perubahaan nilai pada variable i, i kemudian ditambahkan dengan nilai 1 (i++) pada setiap perulangan dilakukan, artinya setiap terjadi perulangan, maka nilai variable i akan berubah. Nah, pada bagian kondisi menentukan keberlanjutan perulangan tersebut, jika pada bagian kondisi masih memberikan nilai true, maka perulangan akan terus dilakukan, yang hasilnya adalah, servo bergerak secara bertahap dari derajat 0 (sebelah kiri) sampai derajat 180 (sebelah kanan).

Sekarang kita coba contoh yang lain, menggunakan LED built in pada development board Arduino yang terpasang dengan pin 13. Berikut adalah contohnya.

for(int i=0; i < 255; i++){

    digitalWrite(13, i); // memberikan nilai sinyal pwm terhadap pin 13 dengan besaran i, secara meningkat

    delay(100); // delay selama 100 micro second

}

Pada contoh diatas, sekarang perulangan for akan mengulang baris perintah digitalWrite(13, i) dan delay(100). Selama nilai variable i masih lebih kecil dari 255 (bagian kondisi), maka perulangan akan terus dilakukan. Hal ini akan memberikan dampak pemberian nilai pwm secara bertahap pada pin 13, dari 0 sampai 254 yang mengakibatkan LED menyala dari redup menjadi terang secara bertahap. Bagaimana kalau kita ingin melakukan kebalikanya, meredupkan LED yang menyala dari terang menjadi redup sampai mati (keadaan off). Perhatikan kode berikut:

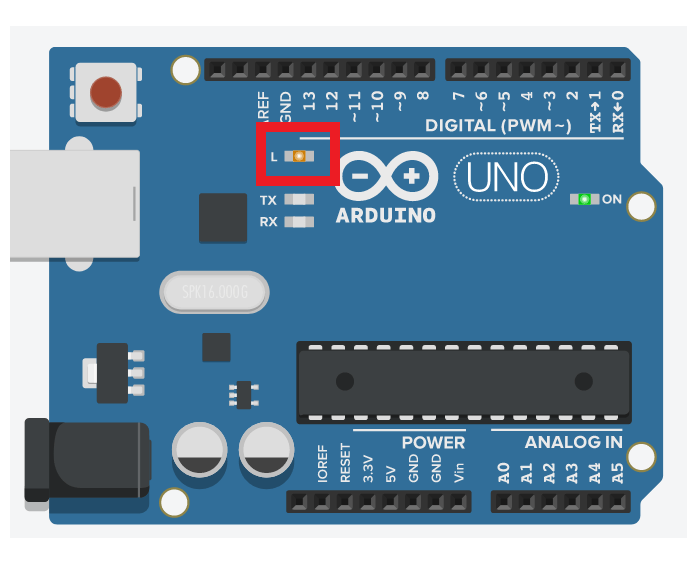
for(int i=255; i > 0; i--){

    digitalWrite(13, i); // memberikan nilai sinyal pwm terhadap pin 13 dengan besaran i, secara menurun

    delay(100) // delay selama 100 micro second

}

Pada kode diatas, variable i diberikan nilai 255 (inisialisasi), kemudian i dicheck terhadap nilai 0, apakah nilainya lebih besar (pada kondisi), jika benar, maka akan menghasilkan nilai true dan perulangan terus berajalan, namun jika nilai menghasilkan false, maka perulangan akan berhenti. Selanjutnya variable i dikurangi dengan nilai 1 (dekrementasi). Hal ini akan Membuat lampu LED pada Arduino akan meredup sampai off.



Gambar . built in LED pada Arduino Uno

Nah sangat mudah kan, hal yang perlu kalian perhatikan yaitu pada perulangan for memiliki tiga bagian utama, yaitu inisialisasi, kondisi, dan inkrementasi atau dekrementasi. Bagian bagian for tersebut yang membedakan perulangan for dengan perulangan lainya. Bagaimana jika beberapa bagian tersebut, atau seluruh bagian tersebut tidak diberikan?, pertanyaan bagus. Untuk itu kita coba perhatikaan contoh berikut.

for(;;)

    Serial.println("hallo world");

Jika kode diatas dijalankan, maka serial output akan menampilkan kalimat “hallo robotik” secara terus menerus atau selamanya (setidaknya sampai batrai habis atau tidak ada energi listrik). Hal ini dikarekanakan pada bagian kondisi pada perulangan for tidak diberikan, hal itu sama saja dengan memberikan nilai true pada kondisi, sehingga jika programmer tidak Membuat cara untuk keluar dari for, maka perulangan akan terus di eksekusi atau dijalankan.

Nah bagaimana jika kondisi tetap ada, tetapi bagian inisialiasi atau decrement atau inkrementasi dihilangkan, pertanyaan menarik. Ayo kita coba lihat juga contoh dibawah ini:

int angka=0;

for(;angka<10;angka++) //contoh for tanpa inisialiasi

    Serial.println("hallo world");

for(int angka=0;angka<10;){ // contoh for tanpa increment/decrement

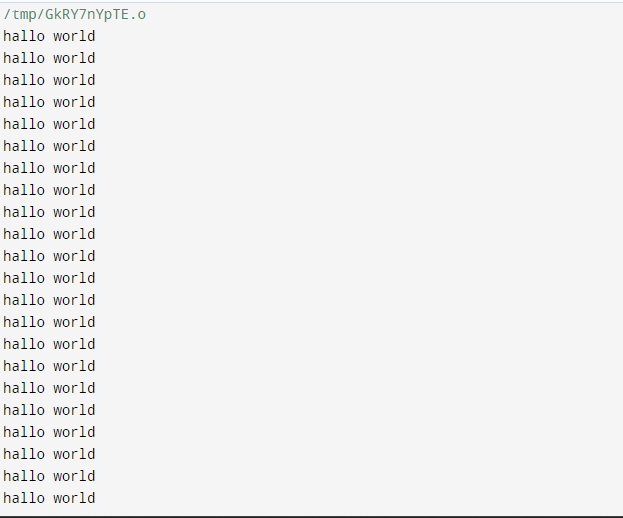
    Serial.println("hallo world");

    angka++;

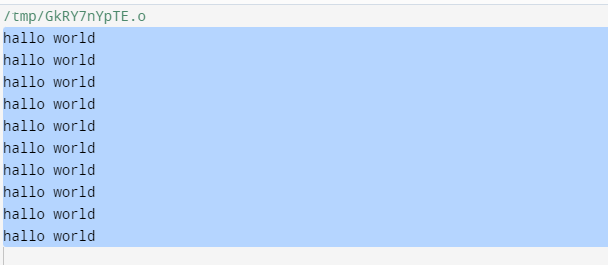
}

Nah pada kode diatas ada dua bentuk perulangan, dimana pada contoh pertama perulangan for, bentuk for tidak menyertakaan inisialisasi, apakah boleh?, tentu saja bisa!. Namun perhatikan, variable angka digunakan dalam kondisi sehingga variable angka harus di inisialisasi di salah satu baris kode sebelum for. Dalam hal ini variable angka di buat tepat diatas perulangan for. Hal ini diperbolehkan, namun tidak dianjurkan. Kenapa?, karena variable angka akan tetap hidup (memiliki nilai) dan dapak diakses selain didalam perulangan for. Hal ini disebut dengan scoping (perwilayahan) dimana variable angka pada contoh pertama memiliki scope lebih luas (dapat diakses oleh fungsi lainya). Jika kalian masih belum faham dengan scoping, tenang saja, kita akan belajar scoping variable pada materi yang akan datang, tunggu saja ya.

Kemudian pada contoh perulangan for yang kedua, dapat dilihat bagian decrement/increment hilang. Apakah diperbolehkan?, dan jawabanya tentu saja bisa!. Namun jika kalian perhatikan, variable angka diinisialiasikan dan digunakan dalam kondisi, maka programmer harus membuat cara agar variable angka dapat berubah nilainya, dan hal ini dilakukan didalam perulangan for. Jika diperhatikan bagian increment atau penambahan tidak hilang, namun berpindah ke dalam nya saja. Hal ini tentu saja valid dan diperbolehkan, namun tidak dianjurkan karena sudah ada tempatnya (bagian increment/decrement). Lalu bagaimana dengan hasil yang diberikan oleh ketiga bentuk for diatas, ayo coba kita lihat gambar dibawah ini.



Gambar . hasil dari perulangan for pertama, kalimat "hallo world" di cetak secara terus menerus



Gambar . hasil pada perulangan for kedua dan ketiga, “hallo world” dicetak sepuluh kali.

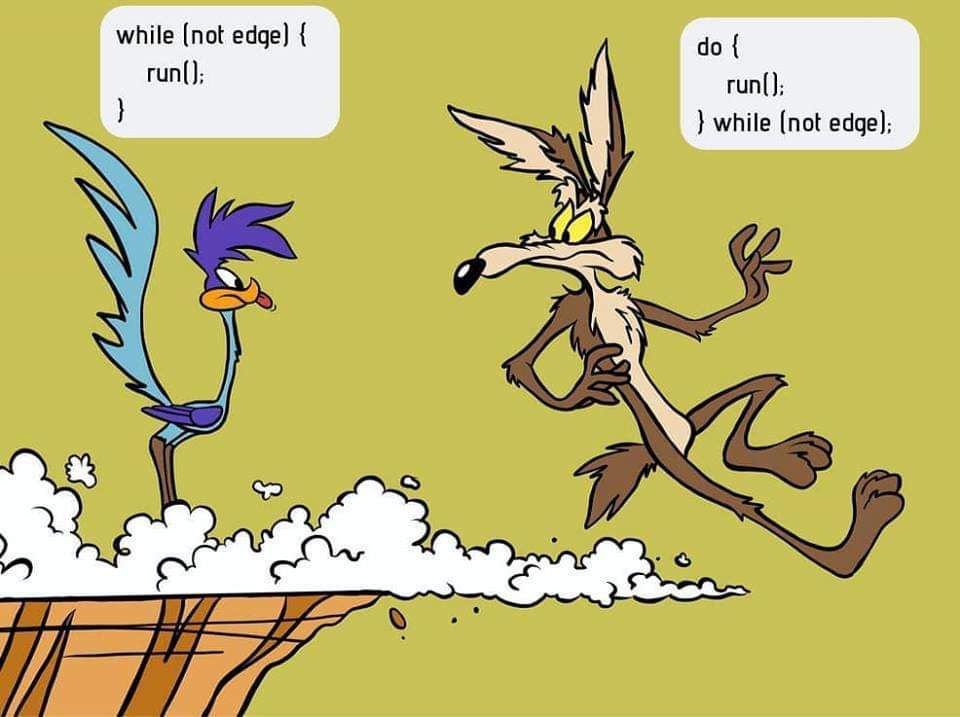
Seperti yang telah dibahas, perulangan for tanpa menggunakan inisialisasi, kondisi, dan decrement/increment akan menjalankan kode dibawahnya secara terus menerus, dibuktikan dengan kalimat “hallo world” yang dicetak secara terus menerus tanpa henti pada gambar 1. Nah untuk yang for selanjutnya memiliki hasil yang sama, yaitu mencetak kalimat “hallo world” sebanyak 10 kali, sesuai dengan kondisi pada perulangan tersebut (angka < 10).

Semoga kalian paham dengan penjelasaan perulangan for yang digunakan dalam Bahasa C/C++. Hal ini penting untuk pengembangan kode dalam robotic. Jika masih bingung, silahkan kalian tanyakan kepada kaka, kaka akan selalu membantu 😊.

Tugas mandiri

Didalam module ini terdapat beberapa contoh penggunaan perulangan for, silahkan kalian coba pada simulator Arduino seperti tinkercad atau langsung menggunakan Arduino development kit!.

## Perulangan While



Gambar 6. Pentingya mengetahui perbedaan while dan do-while dapat menyelamatkan banyak jiwa

Perulangan atau looping merupakan bagian kode yang di jalankan atau dieksekusi secara terus menerus. Kita telah mempelajari perulangan for yang dimana kita sudah ketahui sebelumnya berapa kali kode kita harus berjalan, lantas bagaimana jika pada kondisi berjalan kita tidak tahu harus kapan kode berhenti?, nah disini sudah ada keyword atau kata kunci dalam Bahasa C/C++ yang digunakan untuk itu, yaitu while. Bagaimana cara menggunakanya?, tetap baca modul ini!.

Bentuk pola penulisan perulangan while (while loop)

while(kondisi) // selama kondisi miliki hasil true, jalankan perulangan

penulisan perulangan while sangat sederhana, while hanya memerlukan kondisi untuk dichek. Selama kondisi itu bernilai benar (true) maka perulangan akan terus di jalankan. Lantas bagimana menetukan agar perulangan dapat berhenti sesuai dengan keinginan programmer?, maka kondisi harus memiliki nilai yang berubah ubah selama perulangan berjalan, berarti nilai variable yang dicheck dalam kondisi harus dirubah dan ditempatkan didalam perulangan for. Berikut contoh untuk menjalankan perintah, selama pin 13 bernilai 0 (LOW) dan berhenti jika pin bernilai 1 (HIGH).

int ukuran=digitalRead(13);  // check nilai pin 13 dan masukan nilainya kedalam variable nilai

while(ukuran != 0){ // selama nilai tidak sama dengan 0 (LOW), maka terus jalankan loop

    ukuran=digitalRead(13); // check kembali nilai pin 13 dan masukan kembali kedalam variable nilai

}

Pada contoh diatas, perulangan while akan terus berjalan jika variable ukuran memiliki nilai 1 (HIGH) dan akan berhenti jika ukuran memiliki nilai 0 (LOW). Jika dicermati, pada bagian baris kode didalam while, terdapat baris kode yang berfungsi untuk mengecheck Kembali nilai variable ukuran. Nah hal ini perlu dilakukan agar memastikan perulangan dapat berhenti, jika nilai ukuran berubah menjadi nilai 0 (LOW).

Lalu bagaimana jika sebuah while tidak memiliki kondisi atau tidak ada baris kode yang melakukan check dan merubah nilai variable yang akan di test didalam bagian kondisi?, pertanyaan menarik. Jawabanya adalah sama seperti for maka dia akan berjalan secara terus menerus (infinite) atau selama batrai masih ada. Maka kondisi diatas jika dituliskan dalam kode akan terlihat seperti contoh berikut:

// ketiga contoh kode ini memiliki arti sama, yaitu jalankan perulangan secara terus menerus (infinite)

while(true);

while(1);

while(HIGH);

dalam beberapa kasus, hal ini diperlukan, untuk menjalankan program yang kondisinya dipengaruhi beberapa variable. Maksudnya bagaimana?, jadi jika while biasa hanya mengecheck satu kondisi, maka dengan bentuk ini kita bisa mengecheck beberapa kondisi secara bersamaan Ketika dilakukan perulangan sebagai syarat untuk memberhentikan perulangan. Lalau bagaimana cara untuk keluar dari perulangan while yang berjalan selamanya?, lagi pertanyaan sangat menarik, yaitu dengan menggunakan keyword break. Namun kita tidak akan mempelajarinya dulu dan akan bertemu Kembali pada bagian materi selanjutnya yaitu branching.

Nah sebagai penutup, berikut adalah contoh penggunaan perulangan while yang digunakan untuk beberapa project, yuk kita lihat.

## Project 1: Sensor Parking

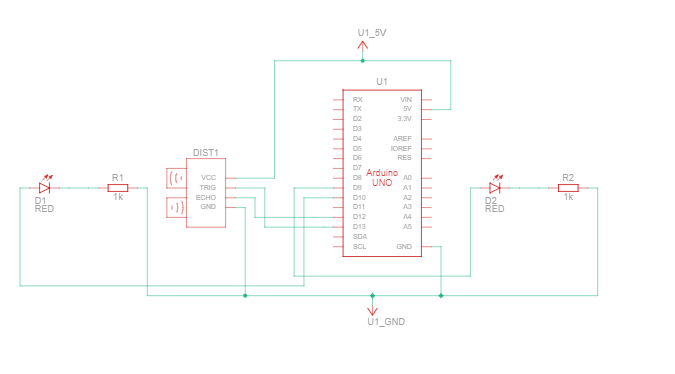
Sensor parking yang akan kita demonstrasikan menggunakan module sensor ultrasonic dimana sensor itu mengeluarkan suara (ultrasonic) yang kemudian dipantulkan oleh benda, lalu dihitung jaraknya dengan cara mengkalikanya dengan kecepatan suara (3820 m/s).

Alat dan bahan:

1. SRF04/05 atau HC-SR04/05
2. Arduino Uno atau Nano
3. Kabel dupont male to female
4. Bread board
5. Lampu LED merah
6. Lampu LED kuning
7. Resistor 330 ohm

Perangkaian

Untuk perangkaian, kita Membuat alatnya mengkuti skema berikut ini:



Kemudian untuk kode sumber adalah sebagai berikut:

const int trigger = 13;

const int echo = 12;

const int amanLed = 10;

const int bahayaLed = 9;

const float jarakAman = 30.0;

void setup()

{

  pinMode(trigger, OUTPUT);

  pinMode(echo, INPUT);

  pinMode(amanLed, OUTPUT);

  pinMode(bahayaLed, OUTPUT);

}

void loop()

{

  float jarak;

  long waktu;

  digitalWrite(trigger, LOW);

  delay(2);

  digitalWrite(trigger, HIGH);

  delay(10);

  digitalWrite(trigger, LOW);

  waktu = pulseIn(echo, HIGH);

  jarak = waktu \* 0.032/2.0; // jarak = waktu \* kecepatan suara / 2

  while(jarak > jarakAman){

    digitalWrite(amanLed, HIGH);

    digitalWrite(bahayaLed, LOW);

    digitalWrite(trigger, LOW);

    delay(2);

    digitalWrite(trigger, HIGH);

    delay(10);

    digitalWrite(trigger, LOW);

    waktu = pulseIn(echo, HIGH);

    jarak = waktu \* 0.032/2.0; // jarak = waktu \* kecepatan suara / 2

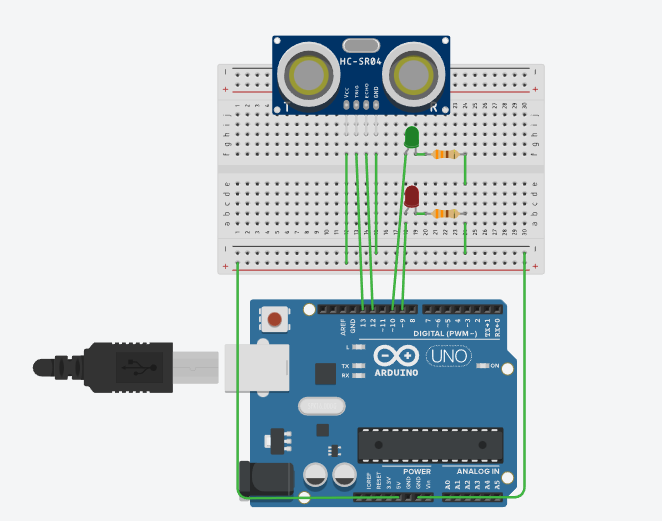
  }

  digitalWrite(bahayaLed, HIGH);

  digitalWrite(amanLed, LOW);

}

Jika sudah dirangkai, maka akan terlihat seperti ini:



Bagaimana cara kerja project sensor parkir diatas?, cara kerjanya adalah dengan menghitung jarak antara benda disekitar dengan modul ultrasonic (HC-SR04/SRF-04). Jika jarak kendaraan dalam rentang aman, yang dimana dalam kode jarak amanya adalah 30 cm, maka lampu led warna hijau akan menyala, dan led warna merah akan mati. Sebaliknya jika jarak kendaraan lebih kecil dari 30 cm, maka lampu led warna hijau akan mati dan led warna merah akan menyala, menandakan jarak kendaraan dengan benda sangat dekat.