FUNGSI DAN PERINTAH YANG DIJALANKAN ARDUINO

A. STRUCTURE

1. setup()

berfungsi ketika sketsa dimulai

2. loop()

menginisialisasi dan menetapkan nilai awal, untuk melakukan perulangan pada program.

Control Structure

1. If

Sebagai operator perbandingan untuk mencapai suatu kondisi tertentu.

2. If.. else

Untuk control yang mempunyai lebih dari satu kondisi.

3. for

Digunakan untuk peningkatan dan mengakhiri loop. Berguna untuk setiap operasi berulang-ulang, dan sering digunakan dalam kombinasi dengan array untuk beroperasi pada koleksi data / pin.

4. Switch case

5. Mengontrol aliran program dengan memungkinkan programmer untuk menentukan kode yang berbeda yang harus dilaksanakan dalam berbagai kondisi. membandingkan nilai variabel dengan nilai-nilai yang ditentukan jika ditemukan kecocokan maka dijalakan.

6. While

Selama statement bernilai TRUE maka akan terus melakukan perulangan perintah.

7. Do..while

Hampir sama dengan pernyataan “while”, bedanya bila pada pernyataan “while” , test kondisi diuji dahulu, dan bila test kondisi bernilai benar maka pernyataan yang ada di dalam blok “while” akan dieksekusi, kemudian test kondisi akan diinkremental. Pada pernyataan “do-while”, kondisi menjadi terbalik, yaitu pernyataan utama akan dieksekusi terlebih dahulu, setelah itu baru test kondisi diuji, jika test kondisi benar maka pernyataan utama akan diulang, dan jika salah program akan keluar dari blok “do-while”.

8. Break

Digunakan untuk keluar dari pernyataan do , atau while loop, melewati kondisi loop normal. Hal ini juga digunakan untuk keluar dari pernyataan switch.

9. Continue

Pernyataan ini melewatkan sisa dari iterasi saat ini loop (lakukan, untuk, atau saat). Dengan memeriksa ekspresi kondisional dari loop, dan melanjutkan dengan iterasi berikutnya.

10. Return

Untuk menghentikan fungsi dan mengembalikan nilai dari fungsi ke fungsi panggilan, jika diinginkan

11. Goto

Instruksi untuk mengarahkan eksekusi program ke pernyataan yang diawali oleh sebuah label atau identifier khusus seperti tanda (:) lalu mengarahkan eksekusi program tampil yang diikuti dengan mengerjakan perintah yang sudah ditaruh pada label. Perintah goto ini juga bisa digunakan untuk keluar dari loop dan nested loop dengan lebih mudah dan ringkas.

Further Syntax

1. ; (semicolon)

Digunakan setelah berakhirnya statement. Merupakan suatu pembatas disetiap statement.

2. {} (curly braces)

Mendefinisikan awal dan akhir dari sebuah blok fungsi

3. // (single line comment)

Sama dengan multi-line coment hanya saja perbaris.

4. /\*\*/ (multi-line comment)

Semua statement yang di tulis dalam block comments tidak akan di eksekusi dan tidak akan di compile sehingga tidak mempengaruhi besar program yang di buat untuk di masukan dalam board arduino

5. #define

Komponen yang digunakan untuk memberi nama pada nilai konstanta sebelum program dikompilasi

6. #include

Memasukkan sketsa program ke dalam libraries.

Arithmetic Operator

Operator yang digunakan untuk memanipulasi angka.

1. = (assignment operator)

Sama dengan (hasil)

2. + (addition)

Penambahan

3. – (subtraction)

Pengurangan

4. \* (multiplication)

Perkalian

5. / (division)

Pembagian

6. % (modulo)

Menghasilkan sisa dari hasil pembagian suatu angka dengan angka yang lain

Comprasion Operator

Operator yang digunakan untuk membandingkan nilai logika.

1. == (equal to)

Sama dengan (misalnya: 12 == 10 adalah FALSE (salah) atau 12 == 12 adalah TRUE (benar))

2. != (not equal to)

Tidak sama dengan (misalnya: 12 != 10 adalah TRUE (benar) atau 12 != 12 adalah FALSE (salah))

3. < (less than)

Lebih kecil dari (misalnya: 12 < 10 adalah FALSE (salah) atau 12 < 12 adalah FALSE (salah) atau 12 < 14 adalah TRUE (benar))

4. > (greater than)

Lebih besar dari (misalnya: 12 > 10 adalah TRUE (benar) atau 12 > 12 adalah FALSE (salah) atau 12 > 14 adalah FALSE (salah))

5. <= (less than or equal to)

Menyatakan operand pertama lebih kecil atau lebih kecil sama dengan operand kedua.

6. >= (greater than or equal to)

Menyatakan operand pertama lebih besar atau lebih besar sama dengan operand kedua.

Boolean Operators

Digunakan pada kondisi if dalam statement.

1. && (and)

Ouput akan bernilai TRUE jika kedua operand bernilai TRUE.

2. || (or)

Output akan bernilai FALSE jika kedua operand bernilai FALSE.

3. ! (not)

Operator NOT menyatakan kebalikan nya. Contoh jika input bernilai FALSE output akan bernilai TRUE.

Pointer Access Operators

Berfungsi sebagai penunjuk baik ke variable, fungsi atau objek.

1. \* dereference operator

Address of atau alamat dari variable/fungsi tersebut di dalam memory cel

2. & reference operator

Value pointed by atau nilai dari variable yang direferensikan.

Bitwise Operators

1. & (bitwise and)

Digunakan diantara dua ekpresi bilangan bulat yang dioperasikan pada perhitungan biner. Jika kedua masukan bit adalah 1, output yang dihasilkan adalah 1, jika output adalah 0.

2. | (bitwise or)

Digunakan diantara dua ekpresi bilangan bulat yang dioperasikan pada perhitungan biner. Output bernilai 1 kecuali jika input kedua bilangan biner adalah 0.

3. ^ (bitwise xor)

Operator ini mirip dengan bitwise operator OR kecuali jika bernilai 1 untuk posisi tertentu kapan tepatnya salah satu bit masukan untuk posisi yang 1. Jika keduanya 0 atau keduanya 1, operator XOR mengevaluasi ke 0

4. ~ (bitwise not)

Tidak seperti & dan |, bitwise NOT operator diterapkan pada operan tunggal. Bitwise NOT berubah setiap bit untuk kebalikannya: 0 menjadi 1, dan 1 menjadi 0.

5. << (bitshift left)

Digunakan untuk operator pergeseran nilai biner ke kiri.

Contoh : int a = 00000111

int b = a << 3

int c = b << 3

artinya b = a digeser kekiri 3 bit atau b = 00111000

6. >> (bitshift right)

Digunakan untuk operator pergeseran nilai biner ke kanan.

Contoh : int a = 00000111

int b = a >> 2

int c = b >> 3

artinya b = a digeser kekanan 2 bit atau b = 11000001

Compound Operators

1. ++ (increment)

Kenaikan variable. Menambahkan suatu nilai variabel

2. -- (decrement)

Penurunan variable. Mengurangi suatu nilai variable.

Melakukan operasi matematika pada variabel dengan konstanta lain atau variabel.

3. += (compound addition)

Penjumlahan gabungan

4. -= (compound subtraction)

Pengurangan gabungan

5. \*= (compound multiplication)

Perkalian gabungan

6. /= (compound division)

Pembagian gabungan

7. %= (compound modulo)

Hasil bagi gabungan

8. &= (compound bitwise and)

sering digunakan dengan variabel dan konstanta untuk memaksa bit tertentu dalam variabel untuk negara LOW (dengan 0). Hal ini sering disebut dalam panduan pemrograman sebagai "clearing" atau "ulang" bit.

9. |= (compound bitwise or)

Digunakan dengan variabel dan konstanta untuk "mengatur" (set ke 1) bit tertentu dalam variabel.

B. FUNCTION

1. Digital I/O

1. pinMode()

Mengkonfigurasi pin yang ditentukan untuk berperilaku baik sebagai input atau output. Lihat deskripsi pin digital untuk rincian tentang fungsi pin.

2. digitalWrite()

Tulis HIGH atau nilai LOW ke pin digital.

Jika pin telah dikonfigurasi sebagai OUTPUT dengan pinMode (), tegangan akan diatur ke nilai yang sesuai: 5V (atau 3.3V di papan 3.3V) untuk HIGH, 0V (ground) untuk LOW.

Jika pin dikonfigurasi sebagai INPUT, digitalWrite () akan mengaktifkan (HIGH) atau menonaktifkan (LOW) yang pullup internal pada pin input. Dianjurkan untuk mengatur pinMode () untuk INPUT\_PULLUP untuk mengaktifkan internal pull-up resistor. Lihat pin digital tutorial untuk informasi lebih lanjut.

CATATAN: Jika Anda tidak mengatur pinMode () ke OUTPUT, dan menghubungkan LED untuk pin, saat memanggil digitalWrite (HIGH), LED mungkin muncul redup. Tanpa secara eksplisit menetapkan pinMode (), digitalWrite () akan telah mengaktifkan pull-up resistor internal, yang bertindak seperti sebuah resistor pembatas arus besar.

3. digitalRead()

Membaca nilai dari pin digital ditentukan, baik HIGH atau LOW.

2. Analog I/O

1. analogReference()

Mengkonfigurasi tegangan referensi yang digunakan untuk input analog (yaitu nilai yang digunakan sebagai bagian atas berbagai masukan). Pilihannya adalah:

DEFAULT: referensi analog default 5 volt (pada papan 5V Arduino) atau 3,3 volt (di papan 3.3V Arduino)

• INTERNAL: built-in referensi, sama dengan 1,1 volt pada ATmega168 atau ATmega328 dan 2,56 volt pada ATmega8 (tidak tersedia pada Mega Arduino)

• INTERNAL1V1: built-in referensi 1.1V (Arduino mega saja)

• INTERNAL2V56: built-in referensi 2.56V (Arduino mega saja)

EKSTERNAL: tegangan diterapkan pada pin AREF (0 5V hanya) digunakan sebagai referensi.

2. analogRead()

Membaca nilai dari pin analog yang ditentukan. Papan Arduino berisi 6 channel (8 saluran pada Mini dan Nano, 16 di Mega), 10-bit analog ke digital. Ini berarti bahwa itu akan memetakan tegangan masukan antara 0 dan 5 volt menjadi nilai integer antara 0 dan 1023. Ini menghasilkan resolusi antara pembacaan dari: 5 volt / 1024 unit atau, 0,0049 volt (4,9 mV) per unit. Berbagai masukan dan resolusi dapat diubah dengan menggunakan analogReference ().

Dibutuhkan sekitar 100 mikrodetik (0,0001 s) untuk membaca input analog, sehingga tingkat membaca maksimum adalah sekitar 10.000 kali per detik.

3. analogWrite()

Menulis nilai analog (gelombang PWM) ke pin. Dapat digunakan untuk menyalakan LED di berbagai brightnesses atau mengendarai motor pada berbagai kecepatan. Setelah panggilan untuk analogWrite (), pin akan menghasilkan gelombang persegi stabil siklus tertentu sampai panggilan berikutnya ke analogWrite () (atau panggilan untuk digitalRead () atau digitalWrite () pada pin yang sama). Frekuensi sinyal PWM pada kebanyakan pin adalah sekitar 490 Hz. Di Uno dan mirip papan, pin 5 dan 6 memiliki frekuensi sekitar 980 Hz. Pin 3 dan 11 pada Leonardo juga berjalan pada 980 Hz.

Fungsi analogWrite tidak ada hubungannya dengan pin analog atau fungsi analogRead.

3. Due & Zero Only

1. analogReadResolution ()

Merupakan perluasan dari API Analog untuk Arduino Due and Zero.

Menetapkan ukuran (dalam bit) dari nilai yang dikembalikan oleh analogRead (). Standarnya 10 bit (mengembalikan nilai antara 0-1023) untuk kompatibilitas dengan papan berbasis AVR.

The due and the Zero memiliki kemampuan ADC 12-bit yang dapat diakses dengan mengubah resolusi ke 12. Ini akan mengembalikan nilai dari analogRead () antara 0 dan 4095.

2. analogWriteResolution ()

Merupakan perluasan dari API Analog untuk Arduino Due, Genuino dan Arduino Zero dan MKR1000.

AnalogWriteResolution () menetapkan resolusi fungsi analogWrite (). Standarnya 8 bit (nilai antara 0-255) untuk kompatibilitas dengan papan berbasis AVR.

4. Advanced I/O

1. tone()

Menghasilkan gelombang persegi frekuensi yang ditentukan (dan 50% siklus) pada pin. Sebuah durasi dapat ditentukan, dinyatakan gelombang terus sampai panggilan untuk noTone (). pin dapat dihubungkan ke buzzer piezo atau pembicara lain untuk bermain nada.

Penggunaan fungsi tone () akan mengganggu output PWM pada pin 3 dan 11 (pada papan selain Mega).

2. noTone()

Menghentikan generasi gelombang persegi dipicu oleh tone(). Tidak berpengaruh jika ada nada yang dihasilkan.

3. shiftOut()

Bergeser keluar byte data satu bit pada satu waktu. Mulai dari salah sebagian (yaitu paling kiri) atau setidaknya (paling kanan) bit signifikan. Setiap bit ditulis pada gilirannya ke pin data, setelah mana pin jam berdenyut (diambil tinggi, maka rendah) untuk menunjukkan bahwa bit yang tersedia.

Catatan: jika Anda berinteraksi dengan perangkat yang clock oleh meningkatnya tepi, Anda harus memastikan bahwa jam pin rendah sebelum panggilan untuk shiftOut (), misalnya dengan panggilan untuk digitalWrite (clockPin, LOW).

4. shiftIn()

Pergeseran dalam byte data satu bit pada satu waktu. Mulai dari salah sebagian (yaitu paling kiri) atau setidaknya (paling kanan) bit signifikan. Untuk setiap bit, jam pin ditarik tinggi, bit berikutnya dibaca dari baris data, dan kemudian jam pin diambil rendah.

Jika Anda berinteraksi dengan perangkat yang clock oleh meningkatnya tepi, Anda harus memastikan bahwa jam pin rendah sebelum panggilan pertama yang shiftIn (), misalnya dengan panggilan untuk digitalWrite (clockPin, LOW).

Catatan: ini adalah implementasi perangkat lunak; Arduino juga menyediakan perpustakaan SPI yang menggunakan implementasi hardware, yang lebih cepat tetapi hanya bekerja pada pin tertentu.

5. pulseIn()

Membaca pulsa (baik high atau low) pada pin. Misalnya, jika nilai adalah HIGH, pulseIn () menunggu untuk pin untuk pergi HIGH, dimulai waktu, kemudian menunggu untuk pin untuk pergi LOW dan berhenti waktu. Mengembalikan panjang dari pulsa di mikrodetik atau 0 jika tidak ada pulsa yang lengkap diterima dalam batas waktu.

Waktu fungsi ini telah ditentukan secara empiris dan mungkin akan menunjukkan kesalahan dalam pulsa pendek. Bekerja pada pulsa dari 10 mikrodetik sampai 3 menit. Perlu diketahui juga bahwa jika pin sudah tinggi saat fungsi dipanggil, ia akan menunggu pin untuk pergi LOW dan kemudian HIGH sebelum mulai menghitung. rutin ini dapat digunakan hanya jika interupsi diaktifkan. Selain resolusi tertinggi diperoleh dengan interval pendek.