**TUGAS KE - 1** **PEMROGRAMAN SISTEM EMBEDDED**

**REFERENSI FUNGSI PROGRAM ARDUINO**

Pengampu : Bapak Bana Handaga, Dr. Ir, M.T.



Oleh :

1. Frendi Wijayanto (L200140017)

Kelas A

Universitas Muhammadiyah Surakarta

2017

**STRUKTUR**

Setiap program Arduino (biasa disebut sketch) mempunyai dua buah fungsi yang harus ada.

setup()

Digunakan untuk inisialisasi variable, pin mode, penggunaan library, dan lain sebagainya.

Hanya dijalankan sekali, pada saat Arduino pertama kali dinyalakan, atau setelah reset.

loop()

Dijalankan setelah function setup selesai, function loop() yang sesuai dengan namanya, untuk

menjalankan program utama dalam Arduino secara berulang terus-menerus, hingga Arduino

dimatikan atau reset.

**STRUKTUR PENGENDALI**

Program yang digunakan untuk menentukan sebuah kondisi, dan jika kondisinya sudah terpenuhi maka akan melaksanakan perintah yang sudah ditentukan. Dan saat tidak memenuhi kondisinya juga ada perintah yang dilaksanakan oleh Arduino.

if(kondisi A)

{

Kode Perintah A

}

else if(kondisi B)

{

Kode Perintah B

}

else

{

Kode Perintah C

}

Pertama Arduino akan lihat Kondisi A. Jika terpenuhi, maka akan melaksanakan Kode Perintah A.

Tapi jika TIDAK, Arduino akan lihat Kondisi B. Jika terpenuhi, maka akan melaksanakan Kode Perintah B. Tapi jika TIDAK juga, maka Arduino akan melaksanakan Kode Perintah C.

for(int i = 0; i < #repeats; i++)

{

Kode Perintah

}

Kode diatas digunakan saat kita ingin mengulangi kode atau nilai dalam beberapa kali.

switch (var) {

case 1:

// melakukan sesuatu ketika var sama dengan 1

break;

case 2:

// melakukan sesuatu ketika var sama dengan 2

break;

default:

// Jika tidak ada lagi cocok, lakukan default

// Default adalah opsional

break;

}

Seperti pernyataan if, switch...case mengontrol aliran program untuk menentukan kode yang berbeda yang harus dilaksanakan dalam berbagai kondisi. Secara khusus, pernyataan switch membandingkan nilai variabel dengan nilai-nilai yang ditentukan dalam laporan case/kasus.

break untuk keluar dari pernyataan switch, dan biasanya digunakan pada akhir setiap kasus. Tanpa pernyataan break, pernyataan switch akan terus mengeksekusi ekspresi berikut ("*falling-through*") sampai break, atau akhir dari pernyataan switch tercapai.

while (ekspresi) {

// Pernyataan (s)

}

while loop akan loop terus menerus, dan tak terhingga, sampai ekspresi dalam kurung, () menjadi salah. Sesuatu harus mengubah variabel diuji, atau while loop tidak akan pernah keluar. Hal ini bisa dalam kode Anda, seperti variabel bertambah, atau kondisi eksternal, seperti menguji sensor.

do

{

// Blok pernyataan

} While (kondisi test);

Melakukan do loop dalam cara yang sama seperti while loop, dengan pengecualian bahwa kondisi ini diuji pada akhir loop, sehingga melakukan loop akan selalu berjalan setidaknya sekali.

for (x = 0; x < 255; x ++)  
{  
    analogWrite(PWMpin, x);  
    sens = analogRead(sensorPin);    
    if (sens > threshold){      // jaminan untuk sensor mendeteksi   
       x = 0;  
       break;  
    }    
    delay(50);  
}

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

for (x = 0; x <255; x ++)   
{   
 if (x> 40 && x <120) {// membuat lonjakan nilai   
 continue;   
 }   
 analogWrite (PWMpin, x);   
 delay (50);

}

Continue untuk melewatkan pernyataan sisa dari iterasi saat ini loop (do, for, atau while). Hal ini terus dengan memeriksa ekspresi kondisional dari loop, dan melanjutkan dengan iterasi berikutnya.

Sebuah fungsi untuk membandingkan input sensor untuk ambang

int checkSensor(){

if (analogRead(0) > 400) {

return 1;

else{

return 0;

}

}

Menghentikan fungsi dan return/mengembalikan nilai dari fungsi ke fungsi panggilan, jika diinginkan.

for(byte r = 0; r < 255; r++){

for(byte g = 255; g > -1; g--){

for(byte b = 0; b < 255; b++){

if (analogRead(0) > 250){ goto bailout;}

// more statements ...

}

}

}

bailout:

goto untuk transfer aliran program ke titik berlabel dalam program

**SELANJUTNYA SINTAKS**

; (titik koma) Digunakan untuk mengakhiri sebuah pernyataan.

{ } (Kurung kurawal) juga disebut hanya sebagai "kawat gigi" atau sebagai "kurung keriting" adalah bagian utama dari bahasa pemrograman C. Mereka digunakan dalam beberapa konstruksi yang berbeda, diuraikan di bawah ini, dan ini kadang-kadang dapat membingungkan bagi pemula.

// (Baris komentar tunggal) Komentar adalah garis dalam program yang digunakan untuk menginformasikan diri sendiri atau orang lain tentang cara program bekerja. Mereka diabaikan oleh compiler, dan tidak diekspor ke prosesor, sehingga mereka tidak mengambil ruang apapun pada chip Atmega.

/ \* \* / (Multi-line comment) sama halnya dengan // untuk membantu Anda memahami (atau ingat) bagaimana program Anda bekerja atau untuk memberitahu orang lain bagaimana program Anda bekerja.

#define adalah komponen C yang berguna yang memungkinkan programmer untuk memberi nama untuk nilai konstan sebelum program dikompilasi. konstanta didefinisikan dalam Arduino tidak mengambil setiap ruang memori program pada chip. Compiler akan menggantikan referensi untuk konstanta ini dengan nilai yang ditentukan pada waktu kompilasi.

#include digunakan untuk memasukkan perpustakaan di luar di sketsa Anda. Hal ini memberikan akses programmer untuk kelompok besar C perpustakaan standar (kelompok fungsi pra-dibuat), dan juga perpustakaan khusus ditulis untuk Arduino.

**OPERATOR MATEMATIKA**

Operator yang digunakan untuk memanipulasi angka (bekerja seperti matematika yang sederhana).

**=** Sama dengan

Membuat sesuatu menjadi sama dengan nilai yang lain (misalnya: x = 10 \* 2, x sekarang sama dengan 20).

**+** Penjumlahan

**–** Pengurangan

**\*** Perkalian

**/** Pembagian

**%** Modulo

Menghasilkan sisa dari hasil pembagian suatu angka dengan angka yang lain (misalnya: 12 % 10, ini akan menghasilkan angka 2).

**OPERATOR PEMBANDING**

Digunakan untuk membandingkan nilai logika.

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

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

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

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

<= Kurang dari atau sama dengan. Memeriksa apakah nilai operan kiri kurang dari atau sama dengan nilai operan kanan, jika ya maka kondisi menjadi benar. (A <= B) benar

>= Lebih dari atau sama dengan. Memeriksa apakah nilai operan kiri lebih besar dari atau sama dengan nilai operan kanan, jika ya maka kondisi menjadi benar.

**OPERATOR BOOLEAN**

&& (And) Benar hanya jika kedua operan adalah benar, misalnya

if (digitalRead (2) == HIGH && digitalRead (3) == HIGH) {// membaca dua switch

// ...

}

Benar hanya jika kedua input yang tinggi.

|| (or) Benar jika operan baik adalah benar, misalnya

if (x> 0 || y> 0) {

// ...

}

benar jika salah x atau y lebih besar dari 0.

! (not) Benar jika operan adalah palsu, misalnya

if (! x) {

// ...

}

adalah benar jika x adalah palsu (yaitu jika x sama dengan 0).

**POINTER ACCESS OPERATOR**

\* dereference operator

Memanipulasi struktur data tertentu, penggunaan pointer dapat menyederhanakan kode, dan pengetahuan memanipulasi pointer berguna untuk memiliki dalam toolkit seseorang.

& reference operator

Sama halnya &reference operator yang tujuannya menyederhanakan kode, dan pengetahuan memanipulasi pointer.

**BITWISE OPERATOR**

& (Bitwise and) Biner DAN Operator salinan sedikit untuk hasil jika ada di kedua operan.

(A & B) akan memberikan 12 yang 0000 1100

| (Bitwise or) Biner OR Operator salinan sedikit jika ada di salah satu operan.

(A | B) akan memberikan 61 yang 0011 1101

^ (Bitwise XOR) Biner salinan XOR Penyelenggara bit jika sudah diatur dalam satu operan tetapi tidak keduanya. (A ^ B) akan memberikan 49 yang 0011 0001

~ (Bitwise tidak) Ones biner Pelengkap Operator adalah unary dan memiliki efek 'flipping' bit.

(~ A) akan memberikan -60 yang 1100 0011

<< (Bitshift kiri) Biner Kiri Pergeseran Operator. Nilai operan kiri pindah ditinggalkan oleh jumlah bit yang ditentukan oleh operan kanan. Sebuah << 2 akan memberikan 240 yang 1111 0000

>> (Bitshift kanan) Biner Pergeseran Operator kanan. Nilai operan kiri pindah kanan dengan jumlah bit yang ditentukan oleh operan kanan. Sebuah >> 2 akan memberikan 15 yang 0000 1111

**OPERATOR SENYAWA**

++ (kenaikan) operator increment, meningkatkan nilai integer oleh salah satu.

x ++; // Increment x per satu dan mengembalikan nilai lama x

++ X; // Increment x per satu dan mengembalikan nilai baru dari x

- (Pengurangan) operator pengurangan, menurunkan nilai integer oleh salah satu.

x--; // Penurunan x per satu dan mengembalikan nilai lama x

--x; // Penurunan x per satu dan mengembalikan nilai baru dari x

+ = (Selain itu senyawa) Tambahkan DAN operator penugasan. Ia menambahkan operan yang tepat untuk operan kiri dan menetapkan hasil untuk operan kiri.

B + = A setara dengan B = B +A

- = (Senyawa pengurangan) Kurangi DAN operator penugasan. Ini mengurangi operan kanan dari operan kiri dan menetapkan hasil untuk operan kiri.

B - = A setara dengan B = B – A

\* = (Senyawa multiplikasi) Kalikan DAN operator penugasan. Mengalikan operan kanan dengan operan kiri dan menetapkan hasil untuk operan kiri.

B \* = A setara dengan B = B \* A

/ = (Divisi compound) Divide DAN operator penugasan. Ini membagi operan kiri dengan operan kanan dan menetapkan hasil untuk operan kiri.

B / = A setara dengan B = B / A

% = (Senyawa modulo) Modulus DAN operator penugasan. Dibutuhkan modulus menggunakan dua operan dan menetapkan hasil untuk operan kiri.

B% = A setara dengan B = B% A

& = (Senyawa bitwise dan) Bitwise DAN operator penugasan.

A & = 2 adalah sama dengan A = A & 2

| = (Senyawa bitwise atau) bitwise inklusif OR dan operator penugasan

A | = 2 adalah sama dengan A = A | 2

**VARIABEL**

**KONSTANTA**

HIGH | LOW membaca atau menulis ke pin digital hanya ada kemungkinan dua nilai pin dapat mengambil diatur-to-: TINGGI dan RENDAH.

INPUT | OUTPUT | INPUT\_PULLUP Pin digital dapat digunakan sebagai INPUT, INPUT\_PULLUP, atau OUTPUT. Mengubah pin dengan pinMode() mengubah perilaku listrik dari pin.

LED\_BUILTIN Kebanyakan papan Arduino memiliki pin terhubung ke on-board LED di seri dengan resistor. Konstanta LED\_BUILTIN adalah jumlah pin yang on-board LED terhubung.

true | false true menyatakan bahwa pernyataan itu benar didefinisikan 1, kebalikannya false salah didefinisikan sebagai 0 (nol).

integer constants Konstanta bilangan bulat adalah nomor yang digunakan langsung dalam sketsa, seperti 123 . Secara default, angka-angka ini diperlakukan sebagai [int](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/Int&usg=ALkJrhh-wZKx8Jlv-KXrbpyTvklic9ok5A) 's tetapi Anda dapat mengubah ini dengan U dan pengubah L.

floating point constants Mirip dengan konstanta integer, konstanta floating point digunakan untuk membuat kode lebih mudah dibaca. konstanta floating point tertukar pada waktu kompilasi untuk nilai yang ekspresi mengevaluasi.

**JENIS DATA**

Void digunakan dalam deklarasi fungsi. Hal ini menunjukkan bahwa fungsi diharapkan untuk kembali ada informasi untuk fungsi dari yang disebut.

Boolean tipe boolean hanya dapat di gunakan untuk menampung data nilai saja true dan false.

Char Tipe char berguna untuk menyimpan sebuah nilai karakter seperti ‘A’,’9’, dan ‘\*’. Perlu di ketahui konstanta karakter di tulis di dalam tanda petik tunggal. Di dalam memori, karakter di simpan dalam bentuk bilangan (nilai ASCII), yang berkisar antara 128 hingga 127.

unsigned char Serupa dengan tipe char, tetapi bilangn akan di kodekan dalam bentuk bilangan positif antara 0 sampai dengan 255.

Byte Tipe data ini berguna untuk menampung bilangan bulat yang berkisar antara 0 ampai 255.

Int Tipe data ini berguna untuk manampung bilangan bulat yang berkisar antara -32768 hingga 32767

unsigned int Tipe data ini berguna untuk menampung bilangan bulat yanf berkisaran antara 0 hingga 65535. Konstanta bertipe unsigned int bisa di tulis dengan akhiran U atau u.

word Tipe word identik dengan unsigned int

long Tipe data ini berguna untuk menampung bilangan bulat yang berkisar antara

-2.147.483.648 hingga 2.147.483.647. konstanta bertipe long di tulis dengan akhiran L atau l

unsigned long Tipe data ini berguna untuk menampung bilangan bulat yang berkisar antara 0 hingga 4.294.967.295 (232-1). Konstanta bertipe Unsigned long bisa di tulis dengan akhiran UL atau ul.

Contoh : 234567989UL.

Short Sama dengan long namun jangkauanya lebih pendek 16-bit data-jenis yang berkisar antara 32.768 hingga 32.767.

Float Tipe data ini berguna untuk menyimpan bilangan real. Angka yang bisa di simpan dari

3,4 x 10^-38 hingga 3,4 x 10^+38 tingkat presisi hingga 4 byte.

Double Sama dengan float namun memiliki jangkauan dua kali dari float 8 byte yang berkisar antara 1,7 x 10^-308 hingga 1,7 x 10^+308

string - char array Anda dapat menggunakan tipe data String, yang merupakan bagian dari inti sebagai versi 0019, atau Anda dapat membuat string dari array tipe char dan null-menghentikannya.

String – object Kelas String, bagian dari inti sebagai versi 0019, memungkinkan Anda untuk menggunakan dan memanipulasi string teks dalam cara yang lebih kompleks daripada [array karakter](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/String&usg=ALkJrhgedAGqVtI6AYju6MrSbNtSDUPgew) lakukan.

Array adalah kumpulan variabel yang diakses dengan nomor indeks.

**CONVERSION**

char() Menyimpan 1 karakter menggunakan kode ASCII (misalnya ‘A’ = 65). Hanya memakai 1 byte (8 bit) dari RAM.

byte() Mengubah suatu nilai ke tipe data byte

int() Digunakan untuk menyimpan angka dalam 2 byte (16 bit). Tidak mempunyai angka desimal dan menyimpan nilai dari -32,768 dan 32,767.

word() Mengkonversi nilai ke tipe data word atau membuat sebuah word dari dua byte atau intinya mengubah suatu variable menjadi word.

long() Biasa digunakan jika nilai datanya lebih besar dari integer. Menggunakan 4 bytes (32 bits).

**VARIABLE SCOPE & QUALIFIERS**

variabel Scope Variabel yang mengatur jangkauan, hak akses sifat scope/ruang lingkup. Sumber akan berlaku lokal jika sumber di dirikan di dalam lingkup lokal (didalam blok) yang berarti hanya bisa dipanggil pada blok ruang lingkup tersebut. untuk sumber diluar blok (global) akan berlaku sebaliknya, sumber diluar blok bisa kita panggil di dalam beberapa ruang lingkup lainya di di dalam ruang lingkup utama. dan untuk bersifat sangat public dan bisa di peroses oleh semua ruang lingkup Biasanya variabel didirikan diluar blok sebelum fungsi dan sesudah file header, tdak didalam suatu blok ruang lngkup.

variabel Statis Variabel Statis dapat berupa variabel local atau variabel eksternal. Jika variabel statis bersifat local, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan. Jika variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama ditempat variabel statis dideklarasikan. ika tidak ada inisialisasi oleh pemrograman secara otomatis akan diberikan nilai awal nol.

variabel volatile volatile kata kunci yang dikenal sebagai kualifikasi variabel, biasanya digunakan sebelum datatype variabel, untuk memodifikasi cara di mana compiler dan program yang selanjutnya memperlakukan variabel.

variabel Const Kata kunci const singkatan konstan. Ini adalah kualifikasi variabel yang mengubah perilaku dari variabel, membuat variabel "read-only"(variable yang tidak dapat diubah nilainya).

**UTILITIES**

Sizeof() Operator sizeof mengembalikan jumlah byte dalam jenis variabel, atau jumlah byte yang ditempati oleh sebuah array. Operator sizeof berguna untuk berurusan dengan array (seperti string) di mana akan lebih mudah untuk dapat mengubah ukuran array tanpa melanggar bagian lain dari program tersebut.

PROGMEM menyimpan data dalam flashdisk (program) memori bukan SRAM. Menggunakan PROGMEM juga merupakan prosedur dua langkah. Setelah mendapatkan data ke dalam memori Flash, membutuhkan metode khusus (fungsi), juga didefinisikan dalam library pgmspace.h, untuk membaca data dari memori program kembali ke SRAM, sehingga kita bisa melakukan sesuatu yang berguna dengan itu.

**FUNCTION**

**DIGITAL I/O**

pinMode() Digunakan untuk menetapkan mode dari suatu pin, pin adalah nomor pin yang akan digunakan dari 0-19 (pin analog 0-5 adalah 14-19). Mode yang bisa digunakan adalah INPUT atau OUTPUT.

digitalWrite() Ketika sebuah pin ditetapkan sebagai OUTPUT, pin tersebut dapat dijadikan HIGH (ditarik menjadi 5 volts) atau LOW (diturunkan menjadi ground).

digitalRead() Ketika sebuah pin ditetapkan sebagai INPUT maka anda dapat menggunakan kode ini untuk mendapatkan nilai pin tersebut apakah HIGH (ditarik menjadi 5 volts) atau LOW (diturunkan menjadi ground).

**ANALOG I/O**

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.

analogRead() Ketika pin analog ditetapkan sebagai INPUT anda dapat membaca keluaran voltase-nya. Keluarannya berupa angka antara 0 (untuk 0 volts) dan 1024 (untuk 5 volts).

analogWrite-PWM Beberapa pin pada Arduino mendukung PWM (pulse width modulation) yaitu pin 3, 5, 6, 9, 10, 11. Ini dapat merubah pin hidup (on)atau mati (off) dengan sangat cepat sehingga membuatnya dapat berfungsi layaknya keluaran analog. Value (nilai) pada format kode tersebut adalah angka antara 0 ( 0% duty cycle ~ 0V) dan 255 (100% duty cycle ~ 5V).

**DUE & ZERO ONLY**

analogReadResolution() analogReadResolution () merupakan perluasan dari API Analog untuk Arduino Due dan 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.

AnalogWriteResolution() merupakan perluasan dari API Analog untuk Arduino Due, Genuino dan Arduino Zero dan MKR1000. menetapkan resolusi fungsi analogWrite (). Standarnya 8 bit (nilai antara 0-255) untuk kompatibilitas dengan papan berbasis AVR. Jika Anda mengatur analogWriteResolution () nilai ke nilai yang lebih tinggi dari kemampuan papan Anda, dewan akan membuang bit ekstra. Sebagai contoh: menggunakan Karena dengan analogWriteResolution (16) pada 12-bit DAC pin, hanya 12 bit pertama dari nilai yang dikirimkan ke analogWrite () akan digunakan dan 4 bit terakhir akan dibuang. Jika Anda mengatur analogWriteResolution () nilai ke nilai yang lebih rendah daripada kemampuan papan Anda, bit yang hilang akan melangkah dengan nol untuk mengisi hardware yang dibutuhkan ukuran. Sebagai contoh: menggunakan Karena dengan analogWriteResolution (8) pada 12-bit DAC pin, Arduino akan menambah 4 bit nol ke nilai 8-bit yang digunakan dalam analogWrite () untuk mendapatkan 12 bit yang diperlukan.

**ADVANCED I/O**

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. Hanya satu nada dapat dihasilkan pada suatu waktu. Jika nada sudah bermain di pin yang berbeda, panggilan untuk tone () tidak akan berpengaruh. Jika nada diputar pada pin yang sama, panggilan akan mengatur frekuensi. Penggunaan fungsi tone () akan mengganggu output PWM pada pin 3 dan 11 (pada papan selain Mega).

noTone() Menghentikan generasi gelombang persegi dipicu oleh tone(). Tidak berpengaruh jika ada nada yang dihasilkan. Jika Anda ingin bermain titinada yang berbeda di beberapa pin, Anda perlu memanggil noTone() pada satu pin sebelum memanggil tone() pada pin berikutnya.

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. 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).

shiftln() 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.

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

TIME

milis() digunakan untuk mengembalikan jumlah milidetik pada saat itu, papan Arduino mulai menjalankan program saat ini.

micros() mengembalikan fungsi jumlah mikrodetik dari waktu, papan Arduino mulai menjalankan program saat ini. Jumlah ini meluap yaitu kembali ke nol setelah sekitar 70 menit.

delay() Cara fungsi delay() bekerja cukup sederhana. Ia menerima satu bilangan bulat (atau nomor) argumen. Jumlah ini mewakili waktu (diukur dalam milidetik).

delayMicroseconds() menerima satu bilangan bulat (atau nomor) argument, ada seribu mikrodetik di milidetik, dan satu juta mikrodetik dalam detik.

**MATCH**

min() Menghitung nilai yang paling minimum.

max() Menentukan nilai yang paling maksimum.

abs() Menghitung nilai absolut dari suatu variabel.

constrain() Menentukan nilai batas untuk suatu variabel.

map() Menentukan pemetaan nilai suatu variable

pow() Memberikan nilai pangkat

sqrt() Memdberikan nilai akar

**TRIGONOMETRY**

sin() Memberikan nilai sinus dan nilai sudut harus dalam radian.

cos() Memberikan nilai cosinus dan nilai sudut harus daam radian.

tan() Memberikan nilai tangen dan nilai sudut harus dalam radian.

**CHARACTERS**

isAlphaNumeric() Pengembalian 1 jika parameter adalah angka atau huruf a dan 0 sebaliknya.

isAlpha() Pengembalian 1 jika parameter adalah huruf bukan angka dan 0 sebaliknya.

isAscii() Pengembalian 1 jika parameter adalah bilangan Ascii dan 0 untuk sebaliknya.

isWhitespace() Pengembalian 1 jika parameter adalah-ruang putih karakter-newline ( '\ n'), space( ''), Bentuk pakan ( '\ f'), carriage return ( '\ r'), tab horizontal ( '\ t'), atau tab vertikal ( '\ v') - dan 0 sebaliknya.

isControl() Pengembalian 1 jika parameter adalah karakter kontrol, seperti baris baru ( '\ n'), bentuk pakan ( '\ f'), carriage return ( '\ r'), tab horizontal ( '\ t'), tab vertikal ( ' \ v '), alert (' \ a '), atau backspace (' \ b ') - dan 0 sebaliknya.

isDigit() Pengembalian 1 jika parameter adalah karakter digit heksadesimal dan 0 sebaliknya.

isGraph() Pengembalian 1 jika parameter adalah karakter cetak selain space ( '') dan 0 sebaliknya.

isLowerCase() Pengembalian 1 jika parameter adalah huruf kecil dan 0 sebaliknya.

isPrintable() Pengembalian 1 jika parameter adalah karakter cetak termasuk ruang ( '') dan 0 sebaliknya.

isPunct() Pengembalian 1 jika parameter adalah karakter cetak selain ruang, angka, atau huruf a dan 0 sebaliknya.

isSpace() Pengembalian 1 jika c adalah-ruang putih karakter-newline ( '\ n'), space( ''), Bentuk pakan ( '\ f'), carriage return ( '\ r'), tab horizontal ( '\ t'), atau tab vertikal ( '\ v') - dan 0 sebaliknya.

isUpperCase() Pengembalian 1 jika parameter adalah huruf besar; 0 sebaliknya.

isHexadecimalDigit() Pengenbalian 1 jika parameter berupa digit Hexadecimal dan 0 jika sebaliknya.

**RANDOM NUMBERS**

randomSpeed() me-reset jumlah pseudorandom generator yang Arduino ini. Meskipun distribusi nomor dikembalikan oleh random () pada dasarnya acak, urutan diprediksi. Anda harus me-reset generator untuk beberapa nilai acak. Jika Anda memiliki pin analog tidak berhubungan, mungkin mengambil suara acak dari lingkungan sekitarnya. Ini mungkin gelombang radio, sinar kosmik, interferensi elektromagnetik dari ponsel, lampu neon dan sebagainya.

random() Fungsi acak menghasilkan angka pseudo-random.

**BITS AND BYTES**

lowByte() Ekstrak order rendah(paling kanan) byte dari variabel (misalnya kata).

highByte() Ekstrak order tinggi (paling kiri) byte dari sebuah kata (atau byte terendah kedua tipe data yang lebih besar).

bitRead() Membaca sebuah bit ke dalam suatu angka angka.parameternya adalah jumlah yang membaca dan yang sedikit untuk membaca, mulai dari 0 untuk setidaknya-signifikan (paling kanan) bit.

bitWrite() Menulis sebuah bit kedalam suatu variabel angka. Parameternya adalah variabel numerik yang menulis, yang sedikit jumlahnya untuk menulis, mulai dari 0 untuk setidaknya-signifikan (paling kanan) bit, dan nilai untuk menulis ke bit (0 atau 1).

bitSet() Sets(1) suatu bit kedalam suatu variabel angka.

bitClear() membersihkan atau menghapus(0) suatu bit kedalam suatu variabel angka.

bit() Menghitung nilai bit tertentu (bit 0 adalah 1, bit 1 adalah 2, bit 2 adalah 4, dll).

**EXTERNAL INTERRUPTS**

attachInterrupt() Parameter pertama yang attachInterrupt adalah nomor interupsi. Biasanya Anda harus menggunakan digitalPinToInterrupt (pin) untuk menerjemahkan pin digital sebenarnya untuk jumlah interrupt tertentu. Misalnya, jika Anda terhubung ke pin 3, menggunakan digitalPinToInterrupt (3) sebagai parameter pertama yang attachInterrupt.

detachInterrupt() Mematikan interupsi yang diberikan.

**INTERRUPTS**

Interrupts() Re-memungkinkan interupsi (setelah mereka telah dinonaktifkan oleh noInterrupts ()). Interupsi memungkinkan tugas-tugas penting tertentu terjadi di latar belakang dan diaktifkan secara default. Beberapa fungsi tidak akan bekerja saat interupsi dinonaktifkan, dan komunikasi yang masuk dapat diabaikan. Interupsi dapat sedikit mengganggu waktu kode, bagaimanapun, dan mungkin dinonaktifkan karena terutama bagian penting dari kode.

noInterrupts() Menonaktifkan interupsi (Anda dapat mengaktifkan kembali mereka dengan interupsi() ). Interupsi memungkinkan tugas-tugas penting tertentu terjadi di latar belakang dan diaktifkan secara default. Beberapa fungsi tidak akan bekerja saat interupsi dinonaktifkan, dan komunikasi yang masuk dapat diabaikan. Interupsi dapat sedikit mengganggu waktu kode, bagaimanapun, dan mungkin dinonaktifkan karena terutama bagian penting dari kode.

**COMMUNICATION**

Serial Serial digunakan untuk komunikasi antara Arduino dan komputer atau perangkat lain. Semua papan Arduino memiliki setidaknya satu port serial (juga dikenal sebagai UART atau USART): Serial. Ini berkomunikasi pada digital pin 0 (RX) dan 1 (TX) serta dengan komputer melalui USB. Jadi, jika Anda menggunakan fungsi-fungsi ini, Anda juga tidak dapat menggunakan pin 0 dan 1 untuk input digital atau output. Anda dapat menggunakan built-in monitor seri lingkungan Arduino untuk berkomunikasi dengan papan Arduino. Klik tombol Monitor seri pada toolbar dan pilih baud rate yang sama digunakan dalam panggilan untuk begin ().

Stream Streaming adalah kelas dasar untuk karakter dan biner aliran berbasis. Hal ini tidak disebut secara langsung, tapi dipanggil setiap kali Anda menggunakan fungsi yang bergantung pada itu. Stream mendefinisikan fungsi membaca di Arduino. Ketika menggunakan fungsionalitas inti yang menggunakan read () atau metode yang sama, Anda dapat dengan aman berasumsi itu panggilan pada kelas Stream. Untuk fungsi seperti print(), Streaming mewarisi dari kelas Print.

**USB(32U4 BASED BOARDS AND DUE/ZERO ONLY)**

Keyboard menambah fungsi dari operator yang mengontrol semua gerakan yang terhubung dengan microcontroller. [Keyboard.begin](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardBegin&usg=ALkJrhgV_ywA2ILzAeqrfr1EF64-Wu_W9w) () ,[Keyboard.end](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardEnd&usg=ALkJrhimxYCVVEoiZ-cUJWGPC5MqlICBUA) (),[Keyboard.press](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardPress&usg=ALkJrhimNTUuWS10zs03zV34j04mWAv0xA) () ,[Keyboard.print](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardPrint&usg=ALkJrhgJzBycj3VTqrfLqSnNRMfBE7jBPw) (),[Keyboard.println](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardPrintln&usg=ALkJrhh6VR8OeBpwfSo6kN77ADsu_YcKpA) () ,[Keyboard.release](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardRelease&usg=ALkJrhiCwKeeRJMuqDV2gLo4GOfwMHn2Ng) () ,[Keyboard.releaseAll](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardReleaseAll&usg=ALkJrhj92dE--0MJybZYhFmubsuh-N5aCw) (),[Keyboard.write](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/KeyboardWrite&usg=ALkJrhhxUJsM1dk70jbFzlODH9s6jqxDHQ) ()

Mouse menambah fungsi dari operator yang mengontrol gerakan kursor yang terhubung dengan microcontroller. [Mouse.begin](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseBegin&usg=ALkJrhj1eSzL6MeaGWDyxX6rfcj6jeGQ_g) (),[Mouse.click](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseClick&usg=ALkJrhiOa1g-8_hXxVzIz8aa4hxpMYdU1g) () ,[Mouse.end](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseEnd&usg=ALkJrhh8LGLTMrkLojLiVFIP4ekJ1cbY5A) () ,[Mouse.move](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseMove&usg=ALkJrhiyxsHU22cDblANsfITZM7D55Cx_A) (),[Mouse.press](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MousePress&usg=ALkJrhhWi3Y3P9DpUdNNf0OdSI3Gk4xgmA) () [Mouse.release](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseRelease&usg=ALkJrhjVwv-QBTCfzX9eQRsqgVePbGVJgQ) (),[Mouse.isPressed](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&rurl=translate.google.com&sl=en&sp=nmt4&tl=id&u=https://www.arduino.cc/en/Reference/MouseIsPressed&usg=ALkJrhhAIvh36nUmHWgUN65Lvpcwka1l4g) ()