Résumé Arduino/C

Vous savez, mais vous n'êtes plus très sûr

Définitions, déclarations utiliser des mots qui parlent	#define Led 13 // ! pas de ; int Duree = 100 ; const int Led=13 ; // constante
variables char codeA = 'A'; 8 bits signé unsigned int nom; 16 bits	byte (non signé 0..255) int (-32768..+32767)
tableau taVal [5]; taVal []={18,5,3,7,5};	
void set-up () { configuration, initialisations }	pinmode (LedTop, OUTPUT); DDRC = 0b00000111; // 0 = in 1 = out

void loop () {
les instructions se terminent par un ;
}
	loop() {
	digitalWrite (LedTop,LOW ;
	} ! LOW ne veut pas dire inactif !

Calcul aa=aa+2 ; ou aa += 2 ; + - * / % j++ ; ajoute 1 j--; soustrait 1 **bitwise** & (and) , | (or), ^ (xor) , ~ (not) >> , << décale	**Comparaison** == (égalité), !=(différent), < , > **boolean** &&, || , ! (not) valeur =0 faux ou différent de 0 vrai

if (condition) { bloc d'instructions; }
si vrai on fait
On teste et on passe plus loin
	if (condition) {
	instructions ;
	}

if (condition) { } else { } si vrai on fait, autrement on fait autre chose	if (condition) uneInstruction ; else uneInstruction ;
while (condition) { } on fait et refait tant que la condition est vraie	while (condition) { // instructions ; }
while (1) { bloc d'instructions; } on fait en boucle les instruction (comme loop),	while (1){ instructions; }
while (1) { } on ne fait plus rien	
do { bloc d'instructions; } **while (condition) ;**	do { a++ ;} // a déclaré avant while (a < 5) ;
for (init ; condition ; modif) { bloc d'instructions; }	for (byte i; i<5; i++) { Cligno (); }

byte etat;
switch (etat) {
case 0: instructions; break;
case 1: instructions; break;
default: instructions; // optionnel
}
	enum { Avance, Recule,..} **etat;**
	switch (etat) {
	case avance: instructions;
	etat = recule; break;
	case recule: . . . .
	}

Commentaires // commentaire jusqu'à la fin de ligne	/* Commentaire sur plusieurs lignes */

Constantes: HIGH (maj) Variables: temp (minuscules) Fonctions: FaireCeci (maj-min)

Fonctions Arduino

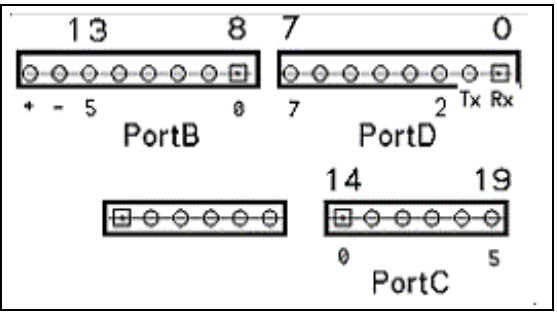
pinMode(pin,b); digitalWrite (pin,b); digitalRead (pin);	boolean boolean boolean	0 = LOW 1 = HIGH
analogWrite (pin,val) analogRead (pin);	byte unsigned int	0..255 pins 5,6, 3,11, 9,10 0..1023 pins 14,15,16,17,18,19 A0 .. A5
delay (ms); delayMicroseconds ();	unsigned long unsigned int	0..~10^{10} millisecondes (~50 jours) 0.. 65535 us (~0.06 sec)
millis (temps); micros(temps);	unsigned long unsigned long	0..~10^{10} millisecondes (~50 jours) depuis reset 0.. ~10^{10} us (~60 sec)
pulseIn (pin,b);	unsigned long	b=HIGH mesure imp à 1 HIGH attend l'impulsion et mesure sa durée
min(a,b) max(a,b)		choisit le min ou max

constrain (x, a, b) ;		garde la vaeur x entre a et b
map (value, fromLow, fromHigh, toLow, toHigh) ;		applique une régle de 3
bitSet(var,n); bitClear(var,n);	boolean boolean	bits numérotés depuis la gauche. s'applique aussi aux PORTs, DDRs
bitRead(var,n); bitWrite(var,n,b);	boolean boolean	if (!bitRead(PORTC,bMousD)) - vrai si MousD pressée PORTC&(1<<bMousD) préférable
lowByte(var); highByte(var);	int, long int, long	prend les 8 bits de poids faible var&0xFF plus rapide prend les 8 bits de poids fort var&0xFF00 var&0xFF000000
Serial.begin(9600); Serial.end();		setup canal série du terminal désactive, rarement utilisé
Serial.print (); Serial.println();		(75) → 75 (75,BIN) → 100101 (75,HEX) → 4B ("Texte 1""2") → Texte 1"2 '1' code Ascii de 1 = 0x31=49
Serial.write();		(65) → A ("abcd") → abcd
Serial.available() Serial.read(); Serial.flush():		if (Serial.available() > 0) { octetRecu = Serial.read(); } vide le tampon
tone (pin, fréqHz); tone (pin, fréqHz,durée); notone ();		démarre un son démarre un son pour la durée spécifiée en ms stoppe le son
shiftOut(D,Ck,dir,val8);		décale 8 bits – très lent

randomSeed(valeur);
random(max);
random(min,max);
	int
	long
	long
		randomseed (analogRead(A0);) A0 flottant
		valeur rendue entre 0 et max-1
		valeur rendue entre min et max-1

attachInterrupt(pin,fct,m); detachInterrupt();		Appelle la fonction s'il y a transition selon mode sur la pin 2 ou 3
interrupt();nointerrupt();		active/désactive toutes les interruptions

Correspondance pins Arduino – bits AVR 168/328
Les pins 0 à 7 vont sur les bits 0 à 7 du portD. Les pins 0 et 1 ne sont pas utilisées pour ne pas interférer avec USB.
Les pins 8 à 13 vont sur les bits 0 à 5 du portB
Les pins 14 à 19 vont sur les bits 0 à 5 du portC; elles acceptent l'ordre analogRead (pin); (valeur 10 bits)
Les pins 3, 5, 6, 9, 10, and 11 acceptent l'ordre analogWrite (pin,pwm8bits);



Interrupt Timer0 8 bits (delay(), millis(), PWM 5 and 6) coupe 8us toutes les ~700 us
Timer1 16 bits (PWM 9 and 10, servos)
Timer2 8 bits (PWM 3 and 11) xBot Interruption 100us

Pour accéder aux pins, on a le choix entre les fonctions Arduino, ou l'accès direct aux bits des registres

`pinMode (2,OUTPUT) ;`
`pinMode (2,INPUT) ;`
durée 3.1 µs
	`bitSet (DDRD,2);`
	`bitClear (DDRD,2);`
	durée 0.13 µs
		`DDRD = 1<< 2;` durée 0.06 µs
		`DDRD |= 1<< 2;`
		`DDRD &= ~(1<< 2);` 0.13 µs

`digitalWrite (2,HIGH);`
`digitalWrite (2,LOW);`
durée 3.8 /4.0 µs
	`bitSet (DDRD,2);`
	`bitClear (DDRD,2);`
	durée 0.13 µs
		`DDRD |= 1<< 2;`
		`DDRD &= !(1<< 2);`
		durée 0.13 µs

`digitalRead (2)` 3.7 µs
`if (digitalRead(2)){}`
durée 3.7 µs
	bitRead (PORTD, 2) 0.13 µs
	`a= bitRead (PORTD,2);`
	durée 0.6 µs
	`if (bitRead (PORTD,2)){}`
	durée 0.13 µs
		`PIND & (1<<2)` durée 0.06 µs
		`a = PIND & (1<<2);` 0.3 µs
		`if (PIND & (1<<2)){}`
		durée 0.13 µs