Arduinokurs

Kurstillfälle 5

Logik

Mycket programmering bygger på logiska samband och det är därför viktigt att kunna grunderna i logik, även kallad boolesk algebra.
Logiska funktioner brukar presenteras i sanningstabeller där ”f” är resultatet på funktionen.
De grundläggande är OCH, ELLER och ICKE. AND, OR och NOT

AND		f
0	0	0
0	1	0
1	0	0
1	1	1

OR		f
0	0	0
0	1	1
1	0	1
1	1	1

NOT	f
0	1
1	0

NAND		f
0	0	1
0	1	1
1	0	1
1	1	0

NOR		f
0	0	1
0	1	0
1	0	0
1	1	0

XOR		f
0	0	0
0	1	1
1	0	1
1	1	0

Som exempel tar vi AND som är sant endast om båda operatorerna är ETT. NAND som är inversen av AND blir NOLL när båda operatorerna är ETT.
Operatorn kan vara t.ex. en digital ingång.

Bitvis och logiska operationer.
Det är stor skillnad på bitvisa och logiska operationer. De bitvisa opererar på bitnivå medan de logiska arbetar på ’hela’ uttrycket.

I C skrivs operanderna för:

Bitvis
AND - &
OR - | (pipe “Alt + >”)
NOT - !
XOR - ^
Logiskt
AND - &&
OR - ||

Ex. BITVIS ELLER
Bitvis. C = A | B. Används t. ex. för att 'sätta' bit 2
Talet A = 0101 (= 5 i dagligt tal)
Talet B = 1000 (=8)
C = 1101 eftersom det räcker om den ena biten är 1 för att svaret på funktionen ska bli 1.

Bit	3	2	1	0	
A	0	1	0	1	(5)
B	1	0	0	0	(8)
C	1	1	0	1	(13)

Ex. Logiskt ELLER
Logiskt blir C = A || B = 1 eftersom alla tal utom 0 tolkas som sanna. A är sann, B är sann och svaret är alltså SANT.

Logiska signalnivåer

Den logiska nivån för '0' är – tro det eller ej – 0V och logisk nivå för '1' är 5V (eller 3,3V). Det är mycket viktigt att dessa nivåer följs och att man inte blandar 3,3V och 5V logik hur som helst.
Logisk nolla vid 3,3V och 5V är ca 0 – 0,8V
Logisk etta vid 3,3V är ca 2V – 3,3V och vid 5V ca 2,5V – 5V

Vid minsta tvekan kontrolleras kretsens datablad där även annat av intresse kan hittas.

Implementering
Vi ska nu prova logiken mot samma koppling som vi hade på kurs 4. Prova koden för alla funktionerna. Koden använder lysdioden vi digital 13 på Arduinon.

AND / OCH
Dioden lyser endast om båda knapparna är intryckta.

```
int buttonPin1 = 2;
int buttonPin2 = 3;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
```

```
  buttonState2 = digitalRead(buttonPin2);
  if (buttonState1 && buttonState2)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

OR / ELLER
Dioden lyser om någon av knapparna är intryckta.

```
int buttonPin1 = 2;
int buttonPin2 = 3;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  if (buttonState1 || buttonState2)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

XOR / EXKLUSIVT ELLER
Dioden lyser om en av knapparna är intryckt.

```
int buttonPin1 = 2;
int buttonPin2 = 3;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;
```

```
void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  if (buttonState1 ^ buttonState2)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

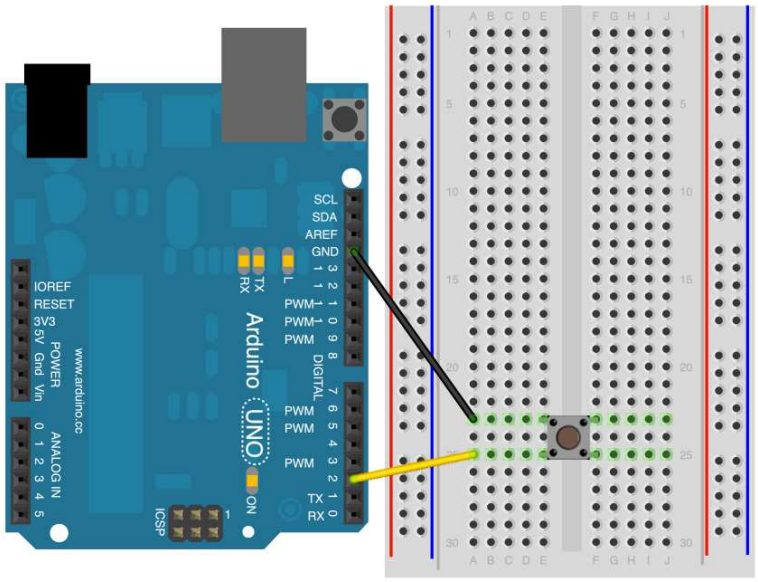
NOT / ICKE

Dioden lyser om knappen inte är intryckt. OBS endast ena knappen används men låt kopplingen vara samma,

```
int buttonPin1 = 2;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
  if (!buttonState1)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Arduinokurs

Två funktioner återstår nu men innan vi kikar på dem förenklar vi uppkopplingen samt lägger till en ny funktion.
Det finns en uppsjö av inbyggda funktioner i en mikrokontroller och det gäller att hitta eller känna till dessa. Allt som kan förenklas är intressant. Dagen till ära tar vi bort pull-up motstånden!
Det finns inbyggda sådana och för att aktivera funktionen använder man pinMode(buttonPin1, INPUT_PULLUP) för att ta buttonPin1 som exempel.
Utan något anslutet till portpinnen är den nu HÖG och vi nollar ingången genom att koppla den till jord. Ta alltså bort motstånden och 5V matningen till tryckknapparna. Koppla ingångarna till var sin knapp och andra sidan på knappen till jord.



Ex. för knappen till digital 2

AND / OCH igen
Tänk på att ingångsvärdena är inverterade. ETT på ingången om ingen knapp är intryckt. Vi måste då invertera ingångsvärdet!

```
int buttonPin1 = 2;
int buttonPin2 = 3;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);              //Pull-up
  pinMode(buttonPin2, INPUT_PULLUP);              //Pull-up
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
```

```
  if (!buttonState1 && !buttonState2)       //OBS knappen jordar ingången. Alltså LÅG vid tryck
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

NAND
NAND är inversen av AND.

```
int buttonPin1 = 2;
int buttonPin2 = 3;
int ledPin =  13;
int buttonState1 = 0, buttonState2 = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);                //Pull-up
  pinMode(buttonPin2, INPUT_PULLUP);                //Pull-up
}

void loop()
{
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  if (!(!buttonState1 && !buttonState2))    //OBS knappen jordar ingången. Alltså LÅG vid tryck
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

NOR kan du prova på egen hand. Jämför med tabellerna på första sidan.

for
Nu tar vi en ny funktion (statement)
for(initiering; jämförelse; nästa)

for-loopen används för upprepning av kommandon ett visst antal gånger. Det som behöver anges är en variabel, en jämförelse och ökning eller minskning av variabeln.
I C finns olika sätt att öka / minska en variabel. Om vi har variabeln A kan den ökas på följande sätt:
A = A +1;
A += 1;
A++;
Alla tre sätten betyder samma sak – öka A med 1. Motsvarande gäller även för minus.

Öka A med 2.
A = A + 2;
A += 2;

for kan användas för att blinka med dioden ett bestämt antal gånger. Vi vill blinka tre ggr.

```
int ledPin =  13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  for(int n = 0; n < 3;n++)
  {
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
  }
  while(1){}
}
```

Vi räknar 0, 1, 2. Alltså 3 ggr i loopen. 3 är inte mindre än 3, mao. utförs inte loopen.

Samma ex igen men med jämförelsen <=. Byt ut i koden ovan. Vad händer? Jo dioden blinkar 4 gånger. Det är mao. MYCKET viktigt att jämföra på ”rätt” sätt!

Läsning av en hel port
Vi avslutar med ett lite mer avancerat exempel där vi utnyttjar det vi nyss har lärt oss. Hittills har vi bara läst digitala ingångar bit för bit. I många sammanhang räcker det gott och väl men ibland räcker det inte till eller så blir programmeringen allt för invecklad.

Antag att dioden ska blinka 0, 1, 2 eller 3 gånger beroende på hur många knappar som är intryckta.

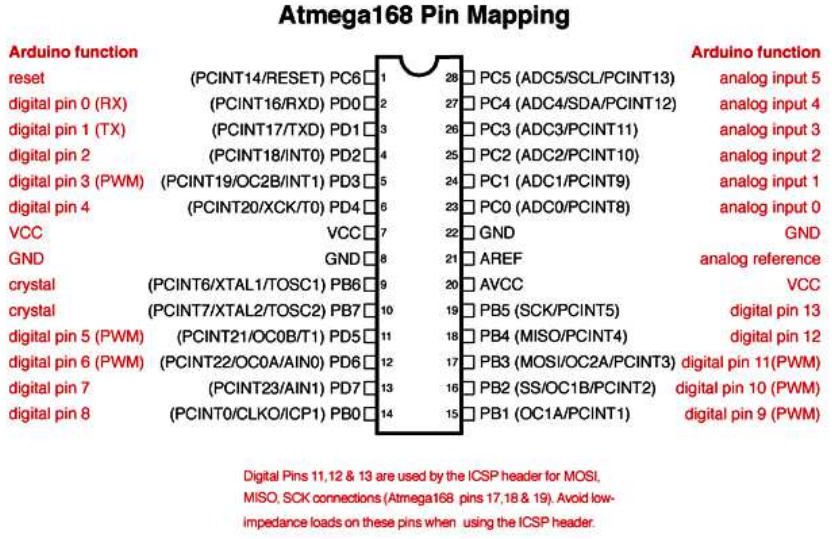
Ny funktion **PIND** läser hela port D med en instruktion. Port D är en 8-bitars port och således läses 8 bitar in vid läsning. Innan vi tittar på koden måste du koppla om lite. Flytta knapparna från digital in 2 & 3 till digital in 0 & 1.

```
int ledPin =  13;
int blink = 0;                  //antal blinkningar
char digitalPORTD = 0;     //spara läsning av port

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(0, INPUT_PULLUP);  //Pull-up
  pinMode(1, INPUT_PULLUP);  //Pull-up
}

void loop()
{
  digitalPORTD = (PIND & 0x03);          //Läs PORTD men endast ingång 0 & 1 sparas
  for(int n = 0; n < digitalPORTD ;n++)    //Räkna avläst värde
  {
    digitalWrite(ledPin, HIGH);
    delay(300);
    digitalWrite(ledPin, LOW);
    delay(300);
  }
  delay(500);
}
```

digitalPORTD = (PIND & 0x03);
Här använder vi en sk. mask för att ta bort de ingångar som vi inte är intresserade av.
Tryckknapparna är kopplade till ingångarna PD0 och PD1. Se bilden uppe tv.

ATmega168/328-Arduino Pin Mapping

Note that this chart is for the DIP-package chip. The Arduino Mini is based upon a smaller physical IC package that includes two extra ADC pins, which are not available in the DIP-package Arduino implementations.



Här MÅSTE bitvis AND användas. Båda operanderna måste vara ett för att funktionen ska bli ett. Eftersom 0x03 (Hexadecimalt 3) = 00000011 binärt kommer bit 7 till bit 2 ej att sparas i variabeln utan endast bit 1 och bit 0.

7	6	5	4	3	2	1	0	DIGITAL PIN
0	0	0	0	0	0	X	X	PORTD
0	0	0	0	0	0	1	1	MASK
0	0	0	0	0	0	X	X	digitalPORTD

Alltså kommer variabeln digitalPORTD att få värdet av digital in 0 och digital in 1 (XX)