Artem Los
artemL@kth.se
(artemlos.net)

Abdallah Hassan
abdhas@kth.se

Skapat: 2015.10.20

Ändrat: 2015.11.16

Table of Contents

1 Komma igång med utvecklingsmiljön 2
1.1 Utvecklingsmiljö 2
1.2 En blinkande lampa 2
1.3 Berätta till Arduino att vi vill få en lampa att blinka 3
2 Lägga till mer komponent 3
2.1 Grundläggande idé 3
2.2 Kopplingsbräda 3
2.3 Lägga till en LED 4
2.4 Övning 5
3 Ta emot information från omgivningen 5
3.1 Koppla in en knapp 5
3.2 Övning: Få lampan att blinka 6
3.3 *Övning: En extern LED 6
4 Binär klocka 7
4.1 For-loopar 7
4.2 Övning 1 7
4.3 Binära tal-systemet 7
4.4 Övning 2 7
5 If satser och aritmetik 8
5.1 Övning 3 8
5.2 Mer om jämförelser 8
5.3 Övning 4a 8
5.4 Övning 4b 9
5.5 If-else 9
5.6 Compound assignment 9
5.7 Övning 5 9
6 Appendix A: An Introduction to Number Bases 11
6.1 Investigating base 20 12
6.2 What is a number 12

Credits

Skrivet av Artem Los (avsnitt 1,2,3,6) och Abdallah Hassan (avsnitt 4,5).

1. Komma igång med utvecklingsmiljön

Man kan fråga sig, vad är en Arduino och vad används den till? Arduino är liten dator som kan göra precis vadsomhelst, t.ex. tända en lampa eller få reda på om någon befinner sig i rummet (se bilden nedan).

Användningsområdet varierar mycket, men huvudsakligen används Arduino till olika sorts hobbyprojekt. Men, det finns inga gränser till vad den kan göra. Från robotar till bilar… .Vem vet?

1. Utvecklingsmiljö

För att kunna prata med den här datorn behöver man installera en app (program) som förenklar jobbet.

- Ladda ned utvecklingsmiljön: https://www.arduino.cc/en/Guide/MacOSX#toc2 och följ instruktionerna på sidan.

2. En blinkande lampa

Nu ska vi få en lampa att blinka. Detta kan vi göra på följande vis:

- Öppna Arduino IDE
- Tryck på File > Examples > 01. Basics > Blink

Nu ska du se följande kod. Notera att "//" är en kommentar och påverkar inte resultatet.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

3. Berätta för Arduino att vi vill få en lampa att blinka

Nu kan vi äntligen få en lampa att blinka. Det enda vi behöver göra är:

- Se till så att Arduino är kopplat till datorn med USB kabeln.
- Tryck på  för att se att programmet inte har några direkta fel.
- Tryck på  för att skicka instruktionerna till Arduino.
- Kolla på lampan bredvid pin 13. Vad händer här?

2. Lägga till fler komponenter

1. Grundläggande idé

I fysiken går man oftast igenom hur det fungerar för att elektricitet ska kunna flöda igenom en krets. Man måste ha en positiv ände och en negativ ände. Här är ett exempel:

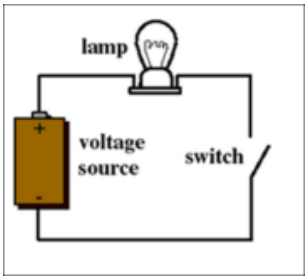


Figure 1: Från här[1].

I Arduino är det precis samma sak. Tänk er att en Arduino är som strömkälla som antingen kan vara på eller av beroende på vad man säger till den. Alla pinnar (numrerade från 1 till 13, t.ex.) är den positiva änden, medan GND (jord) är den negativa änden.

2. Kopplingsbräda

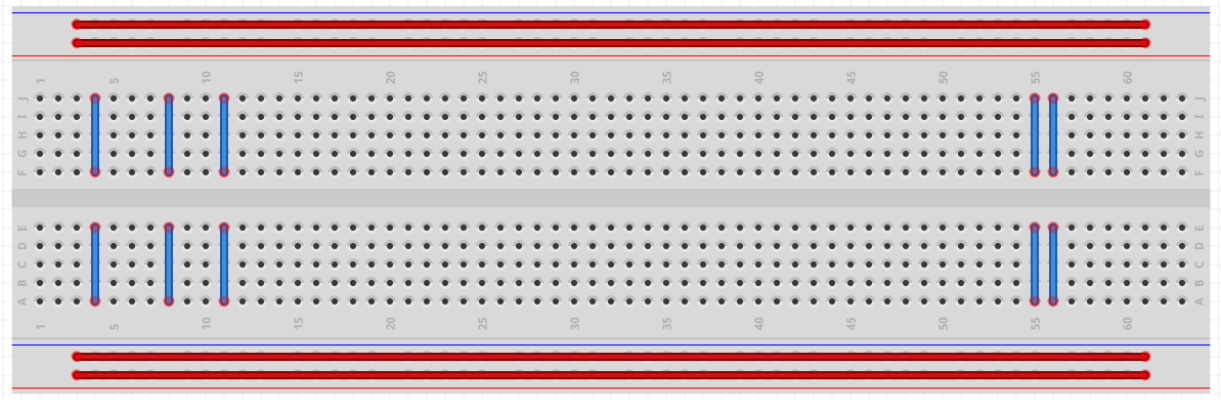


Figure 2: En kopplingsbräda.

[1] https://upload.wikimedia.org/wikipedia/commons/1/19/Simple_electric_circuit.png

Här är en enkel kopplingsbräda som du har i starter kittet. De röda strecken visar hur punkterna är uppkopplade. Om du kopplar någonting i den enda änden kommer det att vara kopplat tillsammans med den andra änden. Om du däremot kopplar någonting i det första röda strecket in i det andra kommer kretsen vara öppen och ingen ström kommer att kunna gå igenom. Samma sak gäller för de blåa strecken. Allt som ligger längs med de blåa strecket kommer att vara ihopkopplat.

3. Lägga till en LED

En LED är en diod som lyser om ström går igenom den. Den långa pinnen på dioden är den positiva änden (som ska kopplas till pinnarna 1-13) och den korta ska kopplas till GND. Notera dock att en diod har ett mycket litet motstånd, alltså är det viktigt att ha en resistor (vi vill ju inte få kortslutning), som kan se ut såhär:

Figure 3: Från här[2].

Här är ett exempel på hur man kan koppla ihop en LED lampa, en resistor och en Arduino tillsammans.

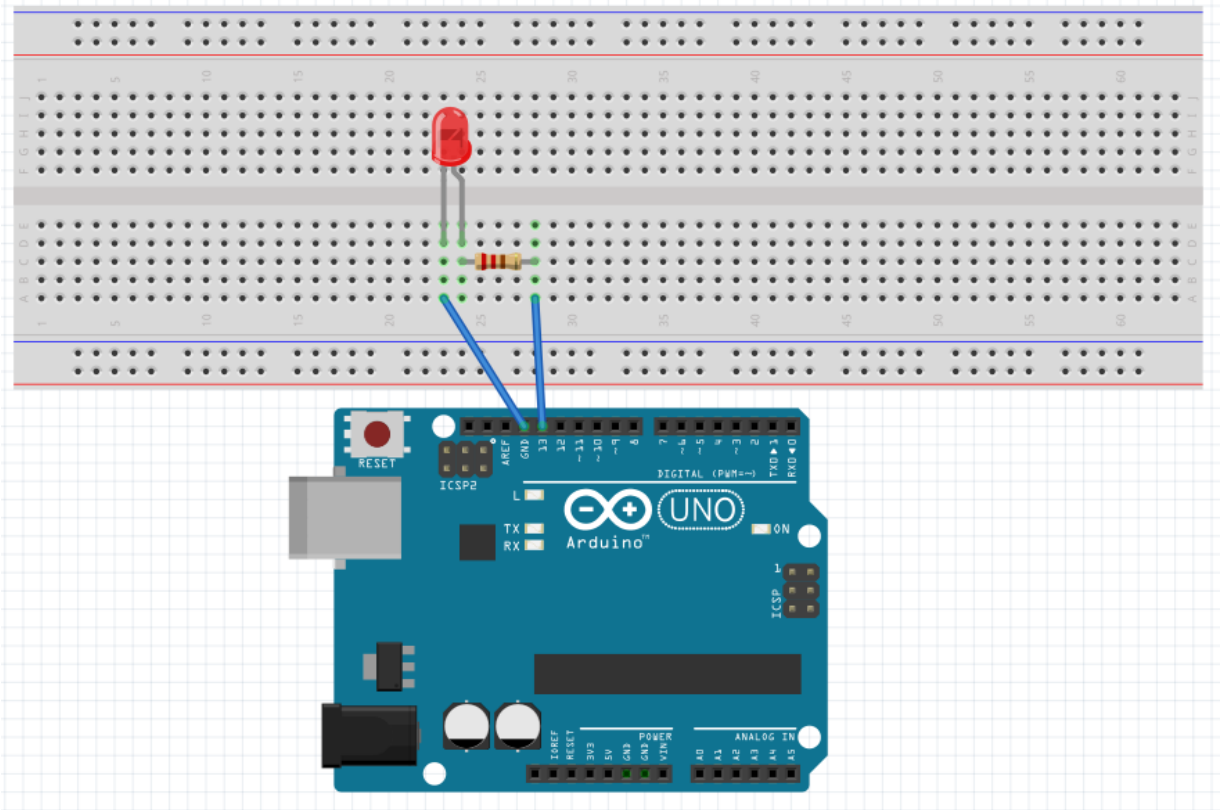


Figure 4: Ett exempel på hur man kopplar ihop en LED med Arduino.

[2] http://cfnewsads.thomasnet.com/images/cmsimage/image/automation-electronics/resistor-sample.JPG

4. Övning

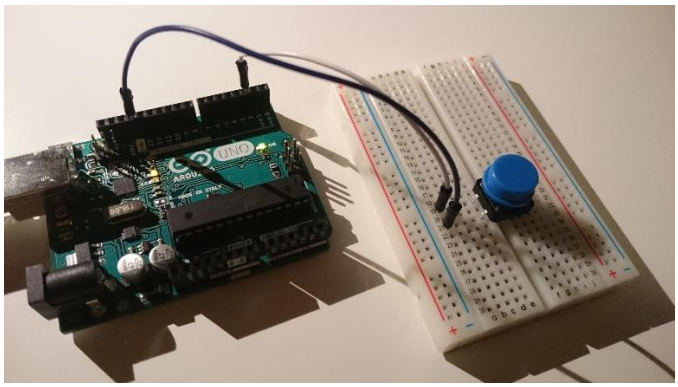
Koppla ihop Arduino med en LED lampa som beskrivet i *Figur 4*. Använd koden från föregående avsnitt (2.2). Vad inträffade?

3. Ta emot information från omgivningen

I många fall vill man ta in information från omgivningen, t.ex. temperatur, ljus, rörelse, osv. I vårt fall vill vi kunna upptäcka när någon trycker på en knapp. Då någon trycker på knappen ska en LED lampa tändas.

1. Koppla in en knapp

Tänk er att en knapp fungerar som en strömbrytare. När vi trycker på knappen blir kretsen sluten och ström kan gå igenom den. Då vi inte trycker på knappen är kretsen öppen och ingen ström kan passera. Så här kan det se ut:



Eftersom vi får en krets med en litet resistans bör vi, som då vi använde en LED, lägga till en resistor. Lyckligtvis finns det en resistans som är inbyggd i Arduino som kallas ”Pullup resistance”. Om vi, som i bilden ovan, väljer pin 2, måste vi initialisera (konfigurera) detta i setup funktionen, dvs.

```
pinMode(2, INPUT_PULLUP);
```

Om vi nu skulle läsa av värdet på pin 2 skulle det vara HIGH då vi inte trycker på knappen, och LOW annars. Nedan är ett kodskelett som kommer göra just det vi vill:

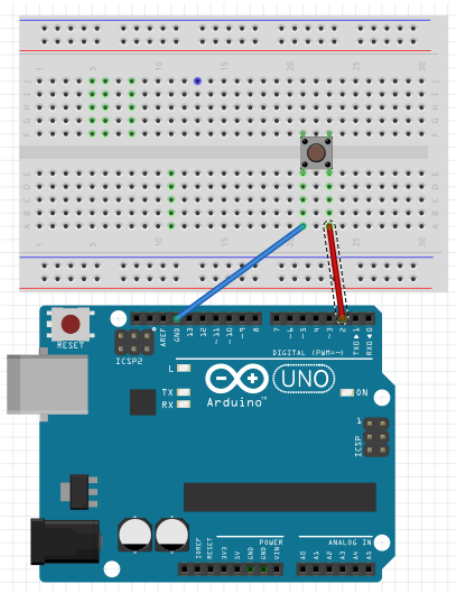
```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  pinMode(13, OUTPUT);

}

void loop() {
  int sensorVal = digitalRead(2);
  Serial.println(sensorVal);

  if (sensorVal == HIGH) {
    // gör någonting
```

```
  } else {
    // gör någonting annat.
  }
}
```

Om du nu öppnar Serial Monitor (finns i Tools menyn), kommer du se massa ettor. Då vi trycker på knappen kommer vi se nollor så länge knappen är intryckt.

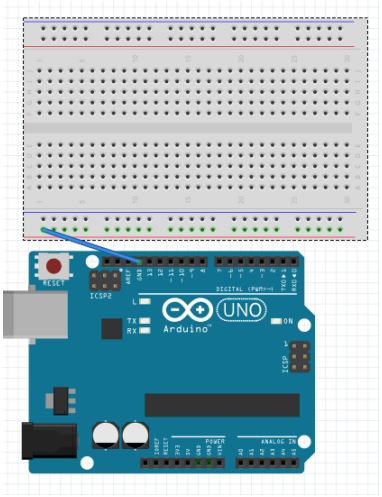
2. Övning: Få lampan att blinka

Vårt uppdrag är nu att få lampan att lysa då knappen är intryckt (pin 13). Använd koden från föregående avsnitt för att uppnå detta. Här är ett exempel på ett kopplingsschema:



3. *Övning: En extern LED

Nu vill vi lägga till en extern LED (som i föregående avsnitt) i uppgift 4.2. Men, vi har bara en GND (jord, minus). Här är ett tips. Alla gröna prickar nedan är GND.



4. Binär klocka

1. For-loopar

En for-loop är ett sätt att köra liknande instruktion(er) flera gånger utan att repetera att skriva onödig kod. Till exempel om vi vill berätta för Arduinot att pinnarna med nummer 3-8 ska ge output och sen få dem att skicka ut elektricitet kan vi göra det på följande sätt

```
void setup() {
  for (int number = 3; number <= 8; number = number + 1) {
        pinMode(number, OUTPUT);
        digitalWrite(number, 1);
    }
}
```

For-loopen kommer att skapa en variabel "number" och ge det värdet 3, sedan kommer den att köra alla instruktioner som finns inom måsvingarna, i det här fallet är det pinMode och digitalWrite. Sedan kommer den att öka variabeln "number" med 1, och köra samma instruktioner igen. Detta håller på så länge som "number" är mindre eller lika med 8. Om vi ville göra samma sak fast med pinnar nummer 4-7 skulle vi ha skrivit

```
for (int number = 4; number <= 7; number++)
```

i början istället.

2. Övning 1

Koppla 6 röda lampor till pinnar nummer 2-7. Gör plats så att 3 fler lampor kan läggas till senare. Skriv ett program som gör så att den första lampan (den som är kopplad mot pinne 2) tänds under en sekund, och efter det tänds lampa 1 och 2 under en sekund osv tills alla lampor är tända under en sekund. Efter det ska programmet börja om igen med bara en lampa och repetera det ovanstående.

3. Binära tal-systemet

Tal kan representeras på flera sätt, ett sätt är det binära som bara använder sig av ettor och nollor. Till exempel, 9 har den binära representationen 1001 eftersom $(\mathbf{1} * 2^3) + (\mathbf{0} * 2^2) + (\mathbf{0} * 2^1) + (\mathbf{1} * 2^0) = 9$. Tanken är att man delar upp talet till en summa av tvåor upphöjt till heltal där heltalet 0 är det som ligger längst till höger och heltalet 1 ligger till vänster om den osv. Sen multiplicerar man varje tvåa upphöjt med ett tal med antingen en 0:a eller en 1:a.

Det faktum att binära tal bara består av två tal kan vi använda för att visa tal på de 6 lamporna från förra övningen. Varje lampa kan representera en nolla om den är släckt, eller en etta om den är tänd. I filen writebin.c[3] finns funktionen `writebin(number, from, to)` som skriver talet "number" till pinnarna "from" till "to". T.ex. kommer `writebin(25, 2, 7)` att skriva den binära formen av talet 25 till lamporna som är kopplade till pinnar 2-7.

4. Övning 2

Kopiera in funktionen writebin till din egen fil och skriv en program som räknar sekunderna och skriver antalet sekunder som har passerat till lamporna från förra övningen.

[3] Filen kan ni hitta här: https://github.com/artemlos/Teaching/blob/master/writebin.c

5. If satser och aritmetik

En if sats är användbar om man vill bara köra en del av programmet under vissa förhållanden. Betrakta följande program

```
int number = 0;
void loop() {
    if (number > 10) {
        digitalWrite(3, 1);
        delay(1000);
    }
    number++;
    digitalWrite(3, 0);
    delay(1000);
}
```

Först definierar vi en variabel "number" och ger den värdet 0. Sedan inuti loop() kommer vi att kolla om "number" är större än 10, och om den är det, så kommer instruktionerna inom måsvingarna efter if (number > 10) att köras. Annars om "number" inte är större än 10, då kommer programmet att strunta i denna del av koden. I just detta exempel så kommer programmet att strunta i denna del av koden de första 11 sekunderna, eftersom vi ökar variabeln "number" med 1 genom att skriva `number++`, så kommer "number" att bli större än 10 efter 11 sekunder, och då kommer alla instruktioner i koden ovan att köras.

1. Övning 3

Utöka lamporna från förra övningen med tre gröna lampor som kan kopplas till t.ex. pinnar 8-10. Skriv ett program där de nya lamporna visar antalet minuter som har passerat. Tänk på att de röda lamporna ska visa 0 och börja om räkningen varje 60 sekunder.

2. Mer om jämförelser

Som tidigare sagt så används if satser för att köra en del av programmet bara under vissa förhållanden. T.ex. `if (number > 5)` kommer att köra efterföljande kod bara om number är större än fem. Men det finns andra typer av jämförelser som man kan använda med if:

`if (number == 12)` är sann om number är lika med 12.

`if (!number)` är sann om number är lika med 0, dvs samma sak som `if (number == 0)`

`if (number)` är sann om number **inte** är lika med 0.

`if (digitalRead(10))` är sann om pinne 10 får en signal.

`if (digitalRead(10) && !number)` är sann om pinne 10 får en signal och number är lika med noll. Symbolen && läses 'och', och den kollar om två saker är sanna samtidigt.

Notera skillnaden mellan '=' och '==='. När vi skriver `if (number == 12)` så frågar vi datorn om number är lika med 12, men om vi skriver `number = 12` så berättar vi för datorn att number ska sättas till värdet 12.

3. Övning 4a

Ta bort de gröna lamporna och ersätt dem med en knapp istället. Knappen ska kunna pausa räkningen när den trycks ner, och starta igång räkningen igen när man trycker på den en annan gång.

4. Övning 4b

Modifiera programmet så att knappen fungerar även om man trycker på den snabbt (mindre än en sekund).

5. If-else

If-else är en variant av if där man lägger till kod som körs bara när if jämförelsen är falsk. T.ex. i följande program:

```
int x = 1;
void loop() {
    if (x) {
        digitalWrite(3,1);
        delay(250);
        x = 0;
    } else {
        digitalWrite(3,0);
        delay(500);
        x = 1;
    }
}
```

Programmet ovan kommer att kolla om x inte är noll (vilket är sant när loop() först körs), och då kommer den att ge signal 1 till pinne 3, vänta 250 millisekunder och sen ge x värdet 0. Den kommer att strunta i allting som finns inom måsvingarna efter else. Men när loop() körs igen så kommer den igen att kolla om x inte är noll, vilket är falskt eftersom vi ändrade värdet på x till 0, så då kommer den att hoppa fram till else och köra det som kommer efter den. I det här fallet kommer vi att ge signal 0 till pinne 3, vänta i 500 millisekunder och ge x värdet 1. Om vi hade en lampa kopplat till pinne 3 så skulle den blinka i en fjärdedelssekund och vara släckt under en halv sekund och sen repetera samma sak.

6. Compound assignment

Compound assignment är ett sätt att ändra på värdet av en variabel. Här än några exempel:

`x += 5;` kommer att öka x med 5. Kan också skrivas som `x = x + 5;`

`x -= 5;` kommer att minska x med 5. Kan också skrivas som `x = x - 5;`

`x *= 3;` kommer att multiplicera x med 3. Kan också skrivas som `x = x * 5;`

`x /= 2;` kommer att dela x med 2 och sen avrunda neråt. Kan också skrivas som `x = x / 5;`

7. Övning 5

Koppla två knappar istället för en till arduinot. Den ena ska öka räkningshastigheten dubbelt så mycket när den trycks ner, och den andra ska minska hastigheten lika mycket när den trycks ner. Vad händer om du trycker på knappen som ökar hastigheten många gånger?

6. Appendix A: An Introduction to Number Bases

[*Ett avsnitt från http://blog.artemlos.net/wp-content/uploads/2014/11/2-to-16-v2.pdf, skrivet 2012 av Artem Los*]

Please take a look at these expressions:

$$2 + 2 = 11$$

$$5 \times 5 = 31$$

$$10 - 1 = 1$$

$$\frac{53}{6} = 8$$

$$9 + 1 = A$$

Probably, they look a bit strange, but, in fact, they are all true. That is because we are dealing with different numerical systems, also called number bases. As you might have figured out, the first expression is written in base 3, the second in base 8, the third in base 2 (also: binary), the fourth in base 9, and the final expression is base 16 (also: hexadecimal).

The concept of numerical systems is as old as our civilisation. The old Greeks had a numerical system, already in the antiquity. Although we are not using it to perform any mathematical calculations, we still use it to, for instance, write kings' name, e.g. Charles XII. Imagine how difficult it would have been, if we continued to use the old numerical system today[4].

However, nowadays, people still have a use of other numerical system, such as the binary (base 2), octal (base 8), hexadecimal (base 16). You have probably heard that these systems are mainly used in the computer's world, and certainly, because of a good reason. Binary numbers are great because it is easy to implement these in an electric circuit.1 stands for on, and 0 stand for off. Morse code uses binary numbers to express each letter in the English alphabet. Unfortunately, because it only contains two symbols, numbers expressed in binary get really big, and in a long run, it makes calculation really difficult, and inefficient.

$$587_{10} = 1001001011_2$$

As you can see the tree digit number in base 10 turns into a ten digit long binary number. It works, but why?

One of the theories of how a numerical system was created is the finger counting. The most common system used in everyday life is base 10, but there are also base 5, base 20 as well. All of these are based on the way we usually count – using fingers.

[4] Please note that the system used by the Greeks differs from base 2, base 5, and our system. Even though the principle is similar, the way it expresses a number is entirely different.

We begin on 1 and continue till 10, and then we restart. However, in some cultural groups, base 20 is being used. That means that it starts from 1and goes up to 20, until it restarts again. You can probably guess – this system is not only using fingers, but also toes. Let us investigate base 20!

1. Investigating base 20

In order to keep it simple, let's decide our digits in our new system using letters from the English alphabet.

Base 20	Base 10
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15
G	16
H	17
I	18
J	19

Base 10 is the system we are used to, but base 20 is our invented system.

Probably, you would ask, why there is no symbol for *20*. The explanation to that is simple; think of it as it is in base 10. The number *10* is actually built up of two digits – namely *1* and *0*. Our system consists of only 10 symbols:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

So, the question is, how would 20 be expressed, if not using one single symbol? The number 20 in base 20, the number 10 in base 10 are basically indicating that we should restart our counting, by adding a zero at the end. Therefore, 20 in base 20 is:

$$10_{20} = 20_{10}$$

So, if we would place base 20 numbers in order, we would get following (from 0 two 21):

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, 10, 11$$

The same in our system (base 10) it would look like:

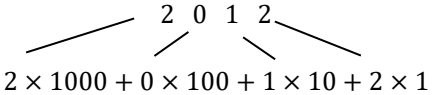
$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21$$

You might already see, using base 20, we save space, i.e. less symbols required to express a number.

2. What is a number

In order to understand a numerical system even better, you need to know how a number in our system might be expressed.

Say we take *2012* as our number. Hopefully, you will agree that it is the same as:



10^0	Digits
10^1	Tens
10^2	Hundreds
10^3	Thousands

As you can see, as we go to the left, i.e. from tens to hundreds, etc, we increase the power of 10 (which is the base) with 1.

By understanding that, let's look at why 10 in base 20, is 20 in our system.

$$10$$
$$1 \times 20^1 + 0 \times 20^0$$

20^0	Digits
20^1	Twenties
20^2	Two hundreds
20^3	Two thousands

Note, the same rule applies to any base. In base 5 for instance, you would have fives, fifth tens, five hundreds, etc.

In the same way, try to predict what values $A2$, JA, 100 have!