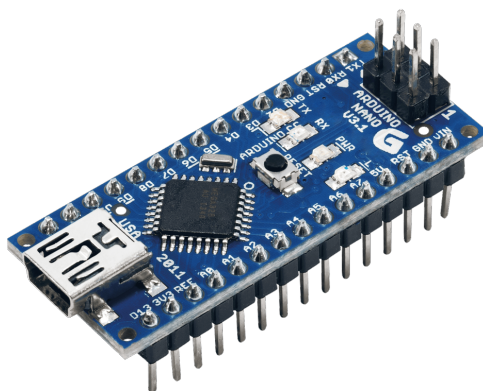


Introduktion till Arduino — MEKMEK01 v10



Innehåll

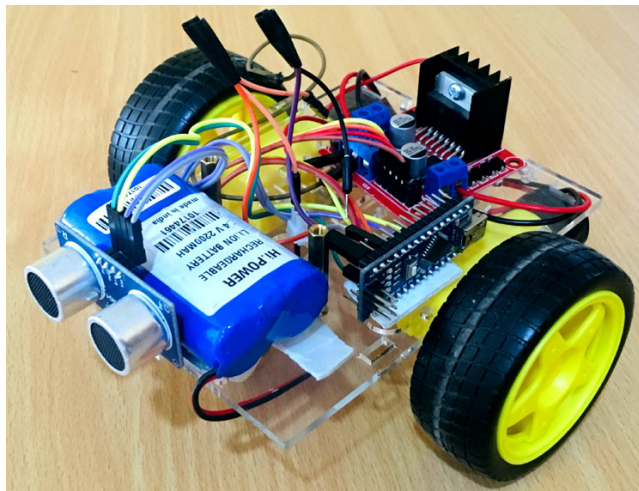
1 Teori	2
1.1 Vad är en mikrokontroller?	2
1.2 Vad är en Arduino?	2
1.3 Hur programmerar man en Arduino?	2
2 Uppgifter	3
2.1 Blink med digitalWrite och delay	3
2.2 Knapp med digitalRead	4
2.3 Seriell kommunikation med Serial	5
2.4 Reaktionstest med millis	6

1 Teori

I uppgifterna kommer du lära känna Arduino genom praktiska övningar. Här kommer lite kort teori om vad en Arduino är och vad den kan användas till.

1.1 Vad är en mikrokontroller?

En mikrokontroller är i princip en programmerbar liten dator som kan styra elektronik. En mikrokontroller kan användas för att styra en mängd olika saker, till exempel en robot, en lampa eller en kaffebryggare.



Figur 1: En robot som använder en mikrokontroller för att undvika hinder.

1.2 Vad är en Arduino?

Arduino är som ett skal runt en mikrokontroller (vanligtvis AVR-processorer). Den har en stor community som delar med sig av sina projekt och kod, så ni kommer kunna använda det som inspiration för ert projekt. Det finns många olika modeller av Arduino, men vi kommer använda oss av en **Arduino Nano**.

1.3 Hur programmerar man en Arduino?

Man skriver kod för Arduino i ett programmeringsspråk som liknar C++. Man kan både skriva- och ladda upp koden till Arduino-enheten från en dator i programmet `Arduino IDE`.

2 Uppgifter

Vi kommer att använda Wokwi för att simulera Arduino, både med kretskopplingar och kod.

2.1 Blink med digitalWrite och delay

Det enklaste programmet: blinka den interna lampan på Arduino-enheten med ett fast intervall.

1. Öppna [mallen för övningarna](#)

2. Skriv in följande kod i void setup():

```
pinMode(13, OUTPUT);
```

3. Skriv in följande kod i void loop():

```
digitalWrite(13, HIGH);  
delay(1000);  
digitalWrite(13, LOW);  
delay(1000);
```

4. Tryck på den gröna "play"-knappen  för att starta simuleringen.

Vad ser du? Vad tror du att koden gör?

Vad händer om du ändrar delay(1000) till delay(500)?

Vad händer om du ändrar alla 13 till någon annan siffra? Vad betyder siffran?


2.2 Knapp med digitalRead

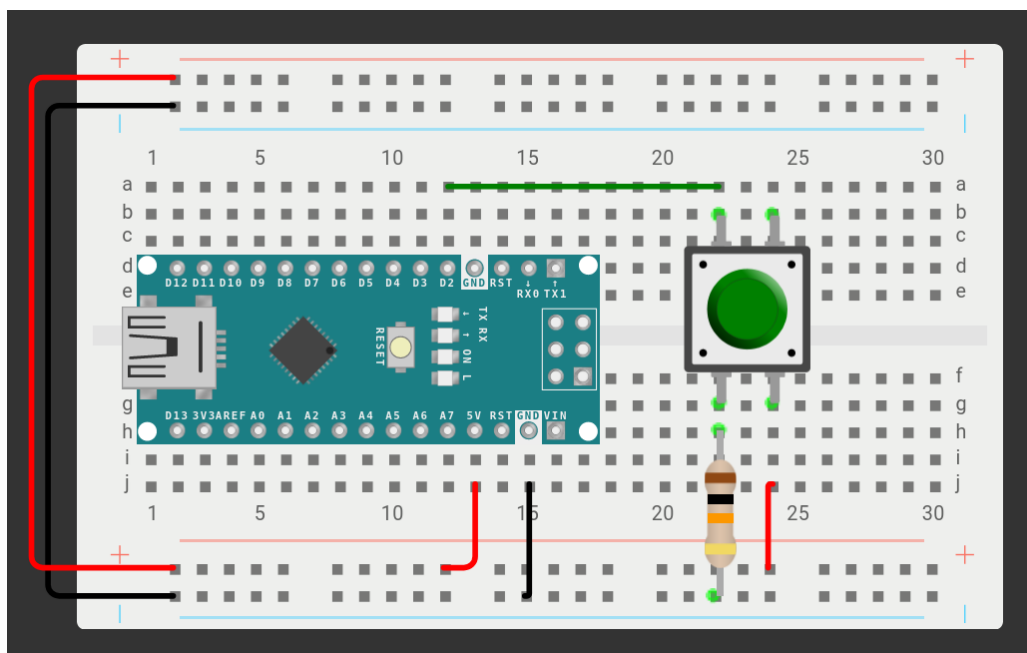
1. Öppna [mallen för övningarna](#)
2. Skriv in följande kod i void setup():

```
pinMode(2, INPUT);  
pinMode(13, OUTPUT);
```

3. Skriv in följande kod i void loop():

```
bool buttonValue = digitalRead(2);  
digitalWrite(13, buttonValue);
```

4. Lägg till en knapp i kretsen genom att klicka plus-knappen  och sedan välja "Pushbutton". Lägg till en resistor på 10 kΩ på samma sätt.
5. Koppla ihop allting enligt figur 2 nedan. Du drar sladdar genom att klicka på en kopplingspunkt och sedan på en annan kopplingspunkt.



Figur 2: Krets med knapp och resistor

6. Starta simuleringen som i förra uppgiften Blink med digitalWrite och delay.
7. Testa att trycka på knappen med musmarkören. Vad händer?
8. Testa att ta bort resistorn. Vad händer? Vad är resistorns syfte?

2.3 Seriell kommunikation med Serial

Arduinon kan "print:a" precis som du är van vid i andra programmeringsspråk. För att göra detta måste den vara ansluten till en dator via USB. Detta kan vara användbart när man felsöker eller vill skicka data till datorn. Vi ska nu skriva ett program som räknar antal gånger en knapp trycks på och skickar detta till datorn.

1. Fortsätt från **Knapp med digitalWrite**

2. Skriv in följande kod högst upp i filen, utanför funktionerna:

```
int count = 0;
```

3. Skriv in följande kod i `void setup()`:

```
pinMode(2, INPUT);  
Serial.begin(9600);
```

4. Skriv in följande kod i `void loop()`:

```
bool buttonValue = digitalRead(2);  
if (buttonValue) {  
    count++;  
    Serial.println(count);  
}
```

5. Starta simuleringen. Nu öppnas en terminal längst ner i skärmen.

6. Tryck på knappen och se vad som händer i terminalen.

Varför tror du att vi skriver `Serial.begin(9600);` i `void setup()`?

Vad betyder `Serial.println(count);`? Vad händer om du ändrar det till `Serial.print(count);`?

2.4 Reaktionstest med millis

Nu ska vi skapa ett lite mer avancerat program- ett reaktionstest. Arduinos inbyggda funktion `millis()` kan användas för att mäta hur långt tid som har passerat sedan den startats.

1. Fortsätt från **Knapp med digitalWrite**

2. Skriv in följande kod i `void setup()`:

```
pinMode(BTN_PIN, INPUT);  
pinMode(13, OUTPUT);
```

3. Skriv in följande kod i `void loop()`:

```
digitalWrite(13, HIGH);  
  
while (digitalRead(BTN_PIN) == LOW) {} // Vänta tills knapp trycks  
  
digitalWrite(13, LOW);  
Serial.println("Tryck så fort du ser lampan lysa!");  
  
delay(random(1000,5000)); // Vänta slumpmässigt mellan 1-5 sekunder  
  
digitalWrite(13, HIGH);  
Serial.println("Nu!!");  
  
int millisBefore = millis();  
while (digitalRead(BTN_PIN) == LOW) {} // Vänta tills knapp trycks  
  
digitalWrite(13, LOW);  
int reactionTime = millis() - millisBefore; // Räkna ut reaktionstid  
Serial.print("Din reaktionstid: ");  
Serial.print(reactionTime);  
Serial.println("millisekunder");  
  
delay(1000); // Vänta tills man kan börja reaktionstestet igen
```

4. Försök att starta simuleringen. Varför funkar det inte?

Tips: Läs vad kompilatorn "klagar" på. Kan du fixa det?

5. När du har fått igång simuleringen, testa programmet genom att trycka på knappen. Ser du någonting i `Serial Monitor` (terminalen i nedre delen av skärmen)? Varför inte?

Tips: Läs igenom koden från **Seriell kommunikation med Serial**.