

# Appendix 1 - Debugging

*“Om debugging är processen att ta bort bugar, då måste programmering vara processen att lägga till dom.”*

- E. W. Dijkstra

För att tycka om upplevelsen att programmera och för att bli en bra programmerare behöver man kunna hantera fel och felmeddelanden. Detta appendix är menat att ge instruktioner om det. Detta dokument är inte labb eftersom det inte passar någonstans i ordningen, utan är menat att användas vid behov under tiden de ordinarie labbarna görs.

Hur hanterar man ett fel eller felmeddelande? Generellt består processen av tre steg,

1. Identifiera problemet
2. Bestäm lösningen på problemet
3. Implementera lösningen

Men det är lättare sagt än gjort. Hur identifierar man ett problem? Ta detta som ett exempel.

```
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000) // wait for a second
  digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

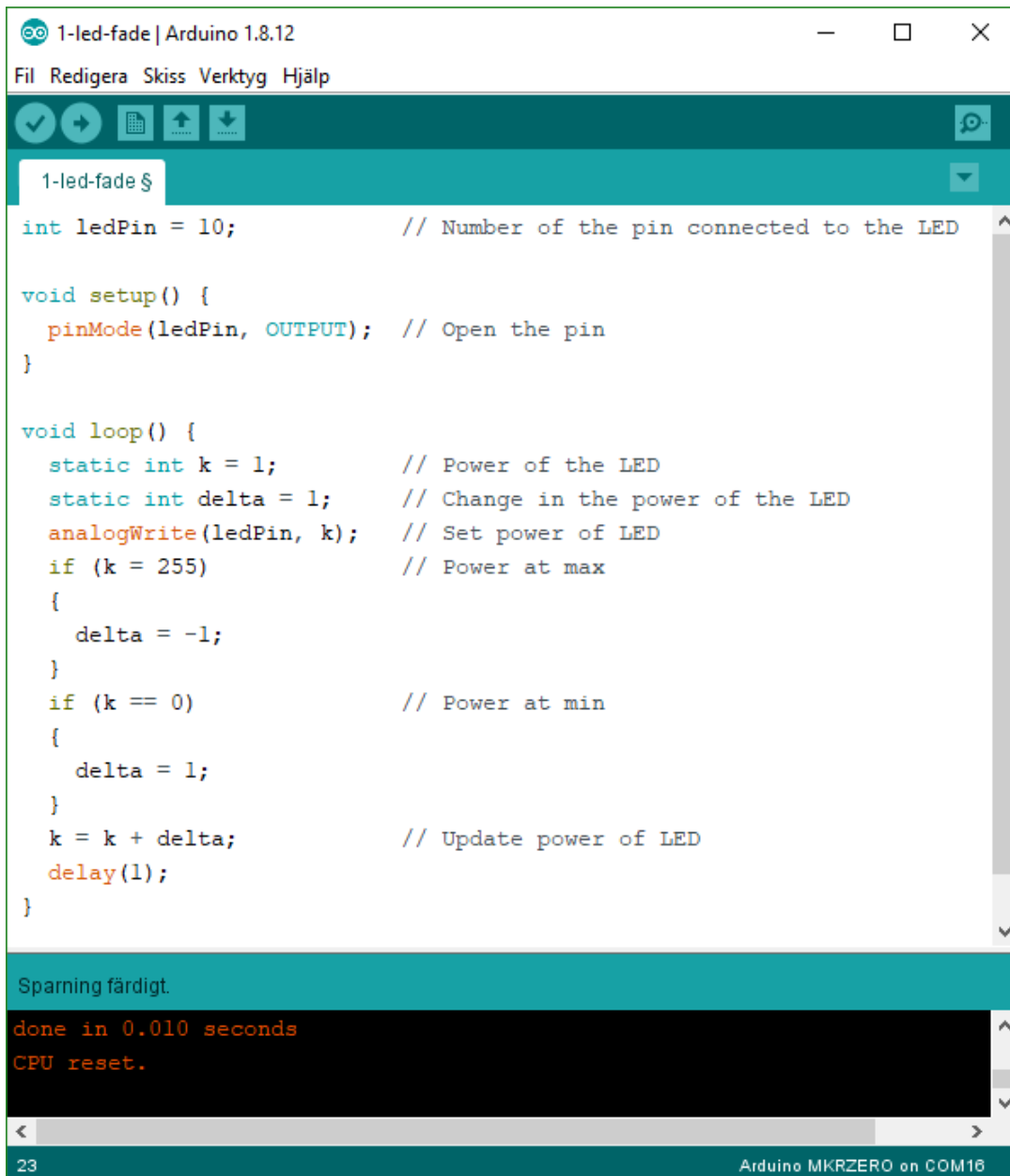
expected ';' before 'digitalWrite' Kopiera felmeddelanden

```
expected ';' before 'digitalWrite'
```

25 Arduino MKRZERO on COM18

Felmeddelandet säger `expected ';' before 'digitalWrite'` och raden med `digitalWrite` är rödmarkerad. Detta exempel är ganska rättfram, programmet påstår att den förväntade sig en markering för att en rad är slut (;) innan funktionen `digitalWrite`, men fann ingen sådan. Då kollar man på raden ovanför den markerade och ser efter om där finns ett

semikolon, och då ser man att det inte gör det. Problemet är att det fattas ett semikolon, lösningen är att skriva dit ett.



```
int ledPin = 10;           // Number of the pin connected to the LED

void setup() {
  pinMode(ledPin, OUTPUT); // Open the pin
}

void loop() {
  static int k = 1;        // Power of the LED
  static int delta = 1;   // Change in the power of the LED
  analogWrite(ledPin, k); // Set power of LED
  if (k = 255)             // Power at max
  {
    delta = -1;
  }
  if (k == 0)              // Power at min
  {
    delta = 1;
  }
  k = k + delta;          // Update power of LED
  delay(1);
}
```

Sparning färdigt.

```
done in 0.010 seconds
CPU reset.
```

23 Arduino MKRZERO on COM16

Denna är betydligt svårare. Koden ger inga felmeddelanden men lysdioden håller en konstant, hög, ljusstyrka istället för pulserna den ska ge. En bra strategi när programmet inte säger att något är fel är att få det att skriva ut mer data. Det görs med Serial monitorn. Genom att modifiera programmet enligt bilden nedan kan man se vad som händer.



```
1-led-fade | Arduino 1.8.12
Fil Redigera Skiss Verktyg Hjälp

1-led-fade $
int ledPin = 10;           // Number of the pin connected to the LED

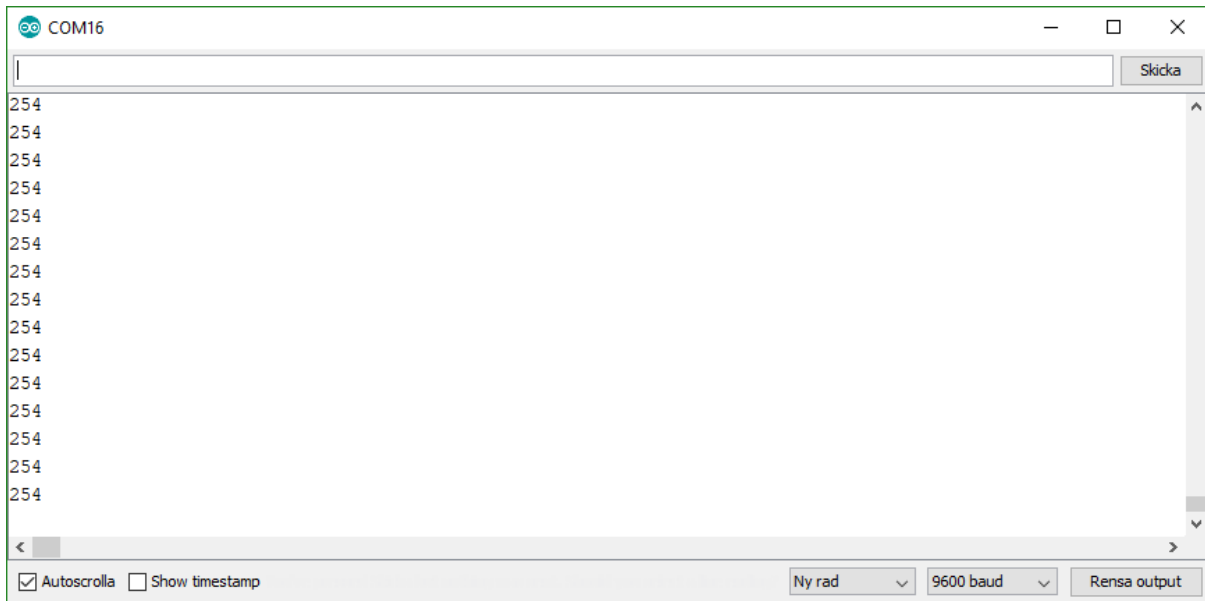
void setup() {
  pinMode(ledPin, OUTPUT); // Open the pin
  Serial.begin(9600);
}

void loop() {
  static int k = 1;       // Power of the LED
  static int delta = 1;   // Change in the power of the LED
  analogWrite(ledPin, k); // Set power of LED
  if (k = 255)           // Power at max
  {
    delta = -1;
  }
  if (k == 0)            // Power at min
  {
    delta = 1;
  }
  k = k + delta;         // Update power of LED
  Serial.println(k);
  delay(1);
}

Uppladdning färdig.
done in 0.010 seconds
CPU reset.
```

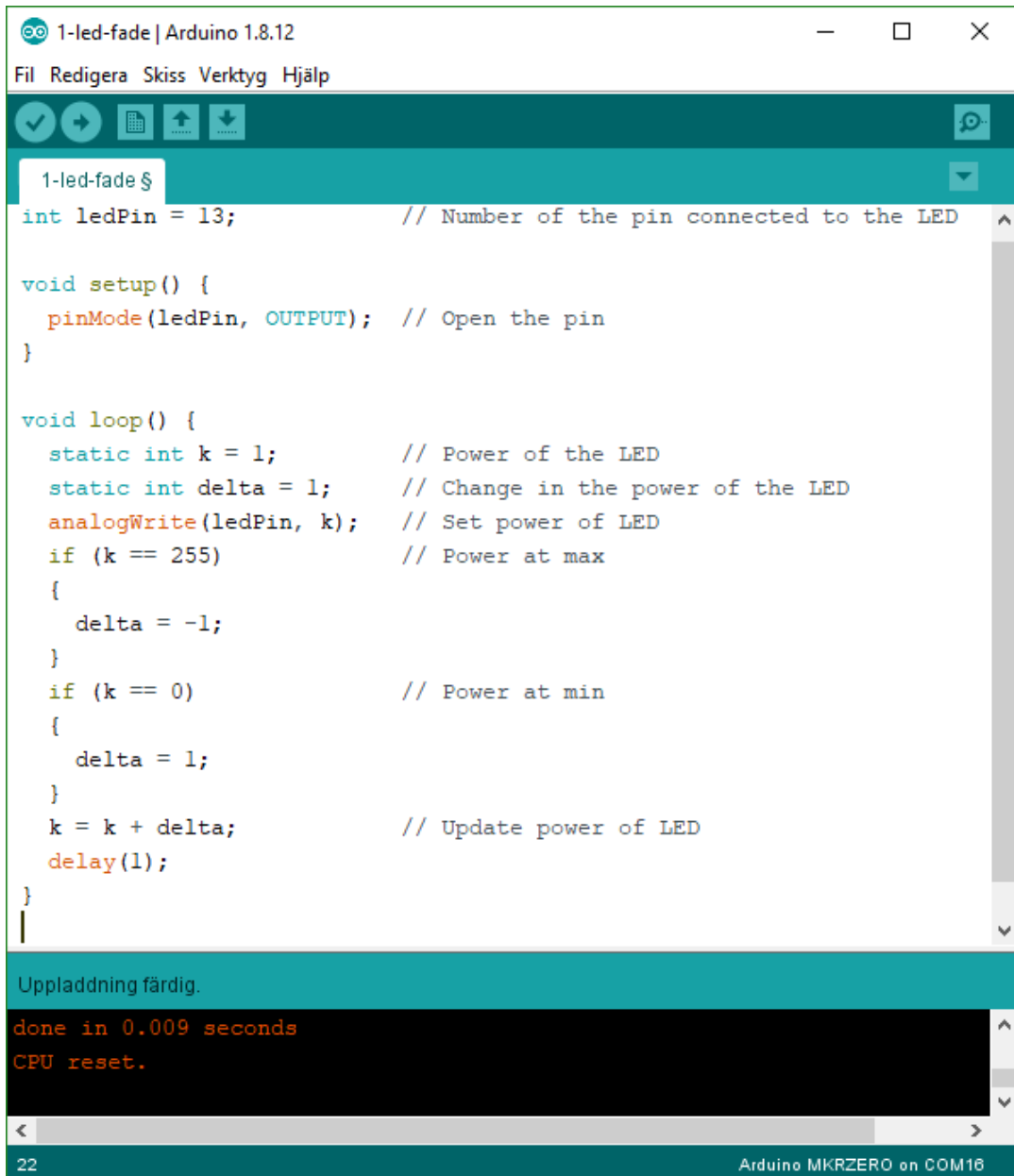
24 Arduino MKRZERO on COM16

Nu kommer programmet tala om vilket värde `k` har varje gång det går igenom loopen. Genom att kolla i Serial monitor ser vi följande.



$k$  har värdet 254 varje gång och det verkar inte ändras. Det sista som händer innan  $k$  skrivs ut är att beräkningen  $k = k + \text{delta}$ ; görs, så  $k$  är 255 och  $\text{delta}$  är -1. Om vi fortsätter att kolla tidigare i programmet ser vi att om  $\text{delta}$  är -1 måste den ha fått det värdet i den första if-satsen. När vi tittar närmare på den if-satsen ser vi att kriteriet är fel. Det är ett enkelt likamedtecken istället för ett dubbelt.

Ett enkelt likamedtecken tillskriver ett värde, vilket (om det lyckas) är sant. Ett dubbelt likamedtecken är en boolesk operator som jämför två värden och blir sant om båda är lika. Lösningen är att skriva till det saknade likamedtecknet.



```
1-led-fade | Arduino 1.8.12
Fil Redigera Skiss Verktyg Hjälp

1-led-fade $
int ledPin = 13;           // Number of the pin connected to the LED

void setup() {
  pinMode(ledPin, OUTPUT); // Open the pin
}

void loop() {
  static int k = 1;       // Power of the LED
  static int delta = 1;   // Change in the power of the LED
  analogWrite(ledPin, k); // Set power of LED
  if (k == 255)           // Power at max
  {
    delta = -1;
  }
  if (k == 0)             // Power at min
  {
    delta = 1;
  }
  k = k + delta;          // Update power of LED
  delay(1);
}

Uppladdning färdig.
done in 0.009 seconds
CPU reset.

22 Arduino MKRZERO on COM16
```

Nästa exempel har en annan typ av problem. Felet denna gång är att lysdioden inte pulserar utan blinkar. Den går direkt från maximal till minimal ljusstyrka utan något mellanting. Vi skriver in koden för Serial monitor igen, för att se vilka värden programmet har.

```
COM16
169
168
167
166
165
164
163
162
161
160
159
158
157
156
155
154
153
152
151
150
149
148
147
146
145
144
```

Resultatet visar att k ändrar värde som den ska. Därför tror vi att problemet inte ligger i koden utan i hårdvaran. En koll i dokumentationen för funktionen `analogWrite`<sup>1</sup> (vilket är den som ska skriva värdena till lysdioden) avslöjar att den måste använda en specifik pin, och vilka pins som fungerar beror på vilken Arduino man använder. Vi använder en MKR ZERO och kan därför välja på pin nr 0-8, 10, A3 eller A4. Koden men problemet använder pin nr 13 vilken inte är på listan. Problemet var att Arduinon inte kunde skriva ut värden med så små skillnader som vi bad den att göra på den pin-en. Lösningen var att byta pin till en som var med på listan.

---

<sup>1</sup> <https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/>



Från dessa tre exempel kan vi se en process för problemlösning. Den ser ut ungefär såhär

1. Läs och tolka eventuella felmeddelanden
2. Få programmet att skriva ut mer information
3. Läs dokumentationen
4. Googla problemet

Den fjärde punkten är inte något vi har pratat om än. Anledningen till att Arduino används i dessa labbar är att det är många andra som har använt Arduino tidigare. Dessa människor har antagligen redan haft alla problem vi kommer stöta på. Genom att klistra in ett felmeddelande i Google kommer ett svar nästan säkert dyka upp.