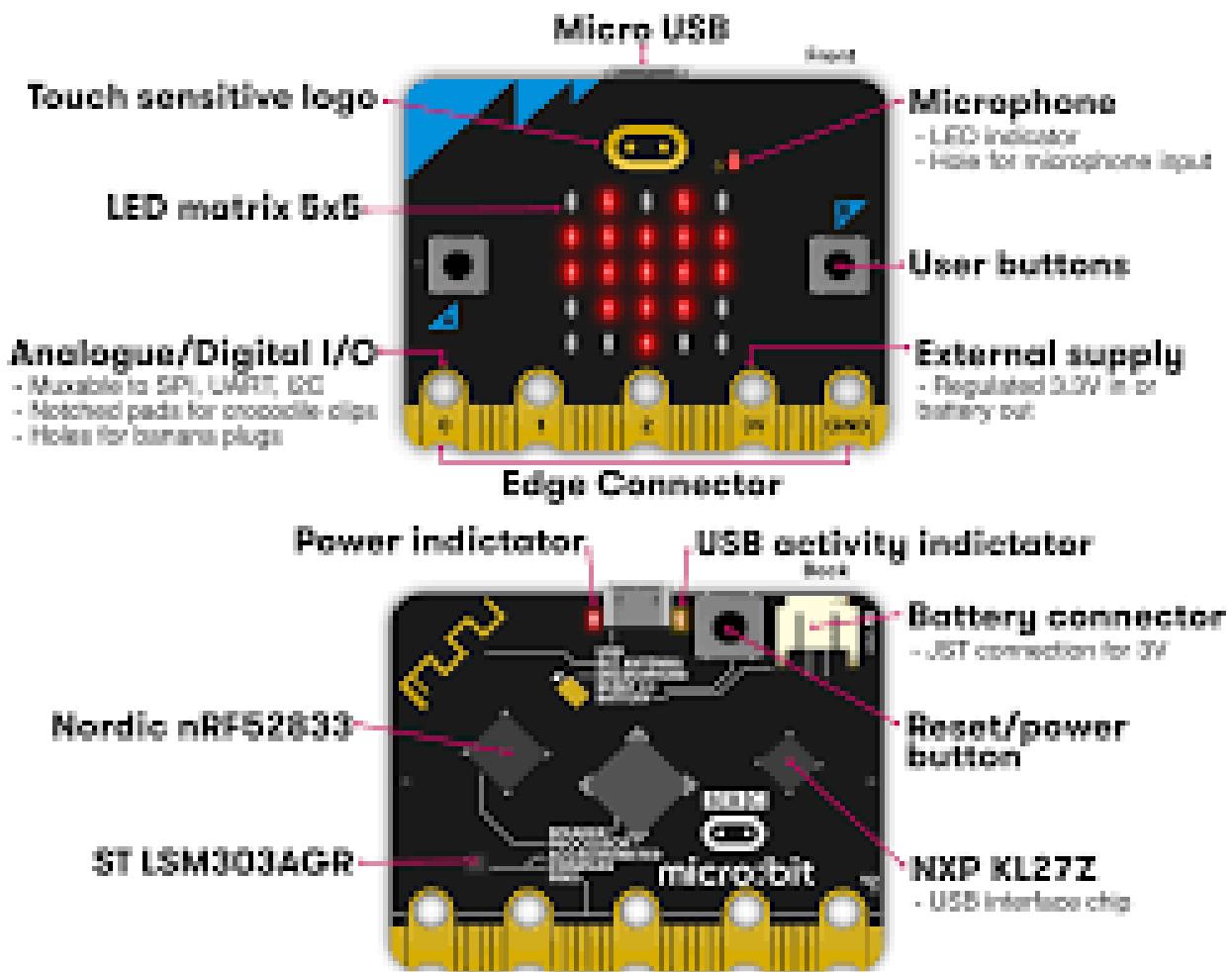


# Technical Skills 2

## Embedded Systems

(practicumhandleiding)



# Contents

<b>1 Inleiding</b>	<b>4</b>
<b>2 De practicum omgeving</b>	<b>6</b>
2.1 Start met de BBC microbit . . . . .	6
2.1.1 Eerste kennismaking met de Microbit . . . . .	7
2.1.2 De Microbit in “The Challenge” . . . . .	7
2.2 Kennismaken met de Arduino omgeving . . . . .	7
2.2.1 Het installeren van de Arduino omgeving . . . . .	8
2.2.2 De arduino omgeving geschikt maken voor de microbit. . . . .	9
<b>3 Introductie in de Arduino omgeving (week 1 en 2).</b>	<b>11</b>
3.1 Hoe zit een Arduino programma in elkaar? . . . . .	11
3.1.1 Code wijzigen. . . . .	12
3.1.2 Commando’s verklaren. . . . .	13
3.1.3 Dimmen van een led . . . . .	13
3.1.4 Het ontvangen van informatie vanaf de arduino-monitor en de beide knoppen. . . . .	14
3.2 De (externe) pinnen van de microbit . . . . .	16
3.2.1 Het lezen van externe signalen met de microbit. . . . .	16
3.2.2 Een denderende schakelaar. . . . .	18
<b>4 De bewegingssensor (week 3).</b>	<b>21</b>
4.1 De accelerator . . . . .	21
4.2 Het kompas . . . . .	23
<b>5 Het gebruik van de LED matrix (week 4).</b>	<b>25</b>
<b>6 Bitwise operaties (week 5 en 6).</b>	<b>28</b>
<b>Bibliography</b>	<b>32</b>
<b>Appendices</b>	<b>33</b>
<b>A Problemen tijdens het practicum.</b>	<b>35</b>
A.1 Adafruit_Microbit.h: No such file or directory. . . . .	35
A.2 SerialPortException: Port name - COM.. . . . .	35
A.3 Tijdens het uploaden . . . . .	35
<b>B Vereisten aan code voor ‘The Challenge’</b>	<b>37</b>
<b>C Data vanuit de Microbit naar SQL sturen</b>	<b>39</b>

C.1	Software installeren . . . . .	40
C.2	Voorbeeld downloaden . . . . .	40
C.3	Stap 1: Elke seconde een meetwaarde genereren met de Microbit. . . . .	41
C.4	Stap 2: Data inlezen met Java programma. . . . .	41
C.5	Stap 3: Data wegschrijven naar MySQL database. . . . .	42
C.6	Stap 4: Eigen aanpassingen maken in Java . . . . .	43

# 1 Inleiding

In dit practicum maak je kennis met het programmeren van een Embedded System, in dit geval een BBC Micro:bit. Het doel is om je een idee te geven waar je bij de differentiatie ‘Network & Systems Engineering’ mee te maken krijgt. Daar leer je software te schrijven om hardware aan te sturen.

In dit practicum houden we het simpel. We proberen je analytische vaardigheden te testen, te prikkelen en te ontwikkelen. Je bestudeert de gegeven voorbeelden om de eigenschappen van de hardware en software te ontdekken. Dit doe je tijdens het practicum, op deze manier kun je de docent vragen stellen als het niet duidelijk is en zo kan de docent zien of je met de stof bezig bent.

Beantwoord vragen Specifiek, Meetbaar, Acceptabel, Realistisch en op Tijd (SMART). Kijk niet alleen wat iets doet, maar vooral hoe iets werkt en waarom. HBO is (ook) analyseren! De voorbeelden zijn geschreven in de taal C (of C++, afgeleid van C). De taal C is de meest gebruikte taal voor embedded systems.

Je leert nu nog geen C-programmeren, dat gebeurt bij NSE in het 2e semester. In het 2e jaar gaan we bij NSE uitgebreid verder met C++ en Object Oriented Programming.

We vragen je nu alleen om de code te lezen en te analyseren. Als je Java programmeren in Periode 1 met succes hebt afgerond, dan moet dat lukken.

De programma’s die je moet maken kun je in elkaar zetten door de code uit de voorbeelden op een slimme manier bij elkaar te kopiëren en te plakken. Zo doe je dat op beginners niveau.

Het belangrijkste doel van het practicum is om een ‘smaakje’ te krijgen van NSE. Het practicum ondersteunt op een aantal punten de theorie (zie de leerdoelen op Blackboard).

Er is in principe géén klassikale instructie. Je werkt zelfstandig aan de hand van deze practicum handleiding. Je kunt wel vragen stellen aan een docent of assistent. Je kunt ook vragen stellen over hoe je de Microbit in ‘The Challenge’ gebruikt, maar je moet zelf de software schrijven.

Er zit geen toetsing in het practicum. De toetsing zit deels in de theoriotoets en betreft de opdrachten uit de handleiding. Een aantal opdrachten kunnen via blackboard ingeleverd worden. Aan de hand van de ingeleverde opdrachten komt een aantekening te staan bij de ‘The Challenge’ en wordt meegewogen bij het advies dat gegeven wordt indien de NSE richting gekozen wordt. Verdere toetsing zit in jullie uitwerking van het Embedded deel van ‘The Challenge’.

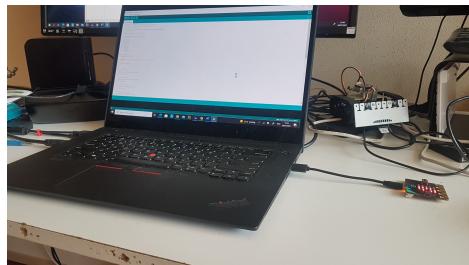
In deze handleiding wordt in hoofdstuk 2 de practicum omgeving besproken, In hoofdstuk 3 worden de basis principes van het embedded programmeren besproken. In

hoofdstuk 4 wordt de bewegingssensor van de microbit besproken. In de hoofdstukken 5 en 6 wordt de matrix en bitwise bewerkingen gedaan. In de appendix, worden een aantal problemen een oplossing gegeven, verder wordt ingegaan op mogelijke oepassingen bij de challenge.

Voor het practicum heb je een **eigen laptop**, een **BBC Microbit en Arduino software** nodig (zie de materiaallijst; de Microbit Go kit kost ongeveer €20).

## 2 De practicum omgeving

De practicumomgeving van dit instructie college bestaat uit een laptop waarbij de BBC:microbit via een USB micro snoertje is aangesloten, zoals te zien is in figuur 2.1. Er zijn verschillende ontwikkelomgeving voor de BBC microbit, maar tijdens



**Figure 2.1:** aansluiting van de BBC microbit.

dit instructie college(practicum), zal voornamelijk via de [Arduino omgeving](#) omgeving gewerkt worden.

### 2.1 Start met de BBC microbit

De BBC Microbit is ontwikkeld om programmeeronderwijs te kunnen geven aan leerlingen van groep 7 (10/11-jarigen). Sinds 2016 wordt dit bordje aan elke groep 7 leerling verstrekt, elk jaar ongeveer 1 miljoen stuks. De hardware, het bordje is ‘open source’, de software die we er bij gebruiken ook.

De hardware is robuust, goedkoop, héél eenvoudig te programmeren, je kunt er veel mee en er is prima ondersteuning. Er zijn verschillende ontwikkel omgevingen gebruiken om programma’s te schrijven voor de Microbit.

Je gaat de Microbit gebruiken in ‘The Challenge’. De Microbit is een Embedded System. Een Embedded System bestaat uit een microcontroller met sensoren en actuatoren. Met sensoren meet je iets (licht, temperatuur, beweging), met actuatoren stuur je iets aan (b.v een ledje, een speakertje, een motor, etc.). Met de Microbit kun je IoT (Internet of Things) devices simuleren, dit zijn ook Embedded Systems. Met een IoT device wordt meestal een apparaat bedoeld die een radiomodule bevat die data kan verzenden of ontvangen. Je kunt deze radiomodule zien als een black box. Er zijn veel verschillende typen voor veel verschillende toepassingen. Ze verschillen in bereik, datadoorvoer, reactiesnelheid en energieverbruik. Voor ‘The Challenge’ maakt het niet uit, je kunt met de Microbit prima een IoT device simuleren.

Let op: Het installeren van software moet je zelf uitzoeken, dit kost veel tijd. Verwacht daarom niet dat je docent je hiermee kan helpen!

Indien het niet lukt, kan altijd om hulp gevraagd worden, maar doe dat met specifieke vragen en vertelt erbij wat jezelf al uitgeprobeerd heb.

### 2.1.1 Eerste kennismaking met de Microbit

Sluit de microbit met behulp van een USB micro snoertje aan op je laptop, zoals te zien is in figuur 2.1. Windows herkent de microbit en kent daar een **drive letter** voor, zoals ook gebeurt bij een USB stick. Dit is te zien in figuur 2.2.

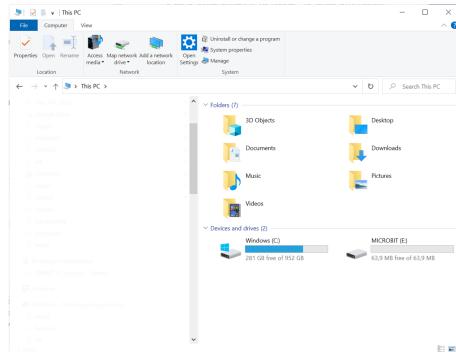


Figure 2.2: Herkenning microbit in windows.

Na het aansluiten zal bij een nieuwe BBC microbit de ingebouwde demo gaan lopen. Werkt de ingebouwde demo niet, dan kan je naar de [Reset de micro:bit to factory pagina](#) gaan en download het [OutOfBoxExperience.hex](#) programma en zet dit op de MICROBIT drive. Hierdoor zal het programma geüpload worden op de microbit.

Voor verdere kennismaking met de microbit is de de pagina <https://microbit.org/get-started/first-steps/set-up/> een goed **starpunt**.

### 2.1.2 De Microbit in “The Challenge”

De doelstelling van The Challenge is dat hier producten van alle differentiaties in samenkommen. Voor het NSE/Embedded deel houden we het eenvoudig. Wat mij betreft is het prima om hier de “blokken” taal voor te gebruiken. Hier kun je alles mee.

Het ligt voor de hand dat NSE studenten in de Arduino omgeving programmeren, waar de programmeertalen C/C++ de voertaal is.

Voor een intro in de “blokken” taal, start hier en begin met ‘Mode’, ‘Step Counter’, kies voor ‘Instructies weergeven’: <https://makecode.microbit.org/>

## 2.2 Kennismaken met de Arduino omgeving

De Arduino omgeving oftewel Arduino IDE (Integrated Development Environment) kan je gebruiken om de Microbit te programmeren.

We gebruiken de Arduino IDE omdat deze op beginnersniveau zeer veel gebruikt wordt en er veel voorbeeldcode te vinden is.

Eigenlijk maakt de Arduino IDE gebruik van de taal C++, dit is een Object Oriented uitbreiding van de taal C. Het C++ / Object Oriented deel van de programmeertaal zie je soms terug in instructies zoals Serial.begin(9600); De punt tussen Serial en begin is hier een indicatie van.

Java en C++ zijn object georiënteerde talen. Daarover leer je meer als je kiest voor de differentiaties “Software Engineering” of “Networks & System Engineering”. Voor dit practicum houden we het eenvoudig. We gebruiken de taal zoveel mogelijk als klassieke “imperatieve” programmeertaal waarbij gewerkt wordt met een reeks opeenvolgende instructies.

### 2.2.1 Het installeren van de Arduino omgeving

VOER ONDERSTAANDE INSTALLATIE THUIS UIT – DIT DUURT 1,5 UUR!!!

1. Download de Arduino software van <https://www.arduino.cc/en/software>, zoals je kan zien zijn er ook versies voor Linux en de Mac.
2. Kies de Windows installer, niet de app. De app is niet geschikt voor wat wij gaan doen.
3. Bevestig alle vragen, klik maar door.

Instellingen wijzigen: Ga naar File → Preferences → Sketchbook location, zoals te zien in figuur 2.3, en wijzig het pad:

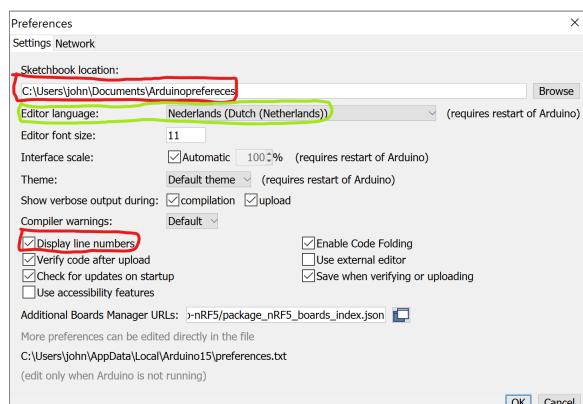


Figure 2.3: preferences scherm.

- C:\Users\john\Documents\Arduino (in mijn geval)
- Zet “Display line numbers” aan.
- Indien de Arduino omgeving Engelstalig is, kan deze gewijzigd worden naar het Nederlands. Het is handig om tijdens de installatie de Engelstalige omgeving te gebruiken.

## 2.2.2 De arduino omgeving geschikt maken voor de microbit.

De firma [Adafruit](#) ontwikkelt diverse componenten voor met name embedded systemen die door ieder gebruiker kan worden toegepast en/of geprogrammeerd. Om dit waar te kunnen maken worden veel tutorials ontwikkeld. Zo is er ook een uitgebreide website, om de [microbit met de Arduino IDE](#) te programmeren.

1. Ga naar de Adafruit website om de [microbit met de Arduino IDE](#) te programmeren.
2. Scroll naar beneden en klik op **Install board and blink! >**. Voer vervolgens het onderdeel "Add NRF5x Board Support" uit.
3. Het selecteren van de BBC micro:bit V2 board en de USB port.
  - (a) Selecteer BBC micro:bit V2.

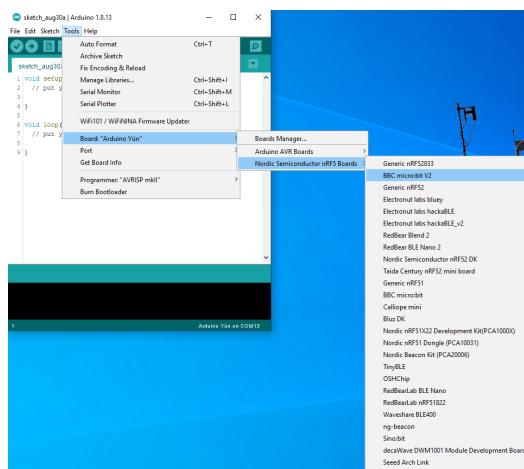


Figure 2.4: Selecteren van de BBC micro:bit V2.

- (b) Het selecteren van de USB port. Indien de microbit niet herkent wordt, start

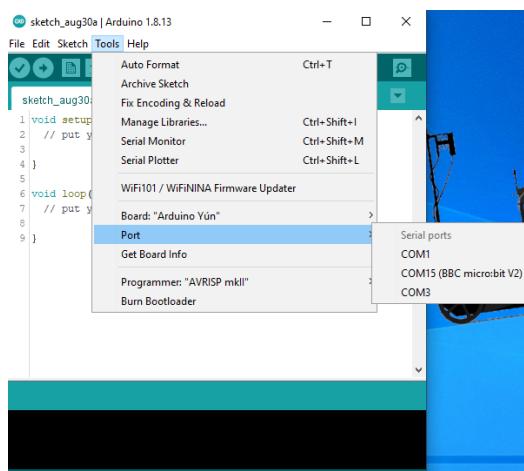


Figure 2.5: Selecteren van de USB port.

de arduino omgeving opnieuw op en/of plug de microbit opnieuw in de USB poort.

4. Het uploaden van een programma gebeurt door op de knop  te klikken. Windows komt met de melding zoals hieronder wordt weergegeven. Hiermee vraagt



**Figure 2.6:** Melding van windows defender.

Windows toestemming om de USB port te mogen gebruiken.

5. Haal de voorbeeldcode van blackboard of [download](#) de laatste versie en plaats deze in je eigen werkdirctory.
6. Open het voorbeeldprogramma blink.ino (dubbel klik), dat wordt weergegeven in Listing 3.1, compileer en upload naar de micro:bit (klik op de knop ). Als het goed is gaat het linkerboven LEDje van de matrix knipperen.

# 3 Introductie in de Arduino omgeving (week 1 en 2).

Hierbij wordt er van uitgegaan dat een werkende Arduino omgeving aanwezig is en het voorbeeld programmaatje **blink** al een keer gerund heeft.

Als eerste volgt een korte uitleg aan de hand van het programma **blink** hoe een Arduino programma in elkaar zit en een korte beschrijving van een paar eigenschappen van de microbit pinnen. Verder zijn er een aantal opdrachten.

## 3.1 Hoe zit een Arduino programma in elkaar?

We gaan even terug naar het **blinkdemo** voorbeeld uit de installatiehandleiding (zie listing 3.1). Als het goed is heb je deze opgeslagen en kun je deze nu openen.

```
1 const int COL1 = 4;      // Column #1 control. In versie 1 de waarde 3
2 const int LED = 21;      // 'row 1' led. In versie 1 de waarde 26
3
4 void setup() {
5     Serial.begin(9600);
6     Serial.println("microbit_is_ready_voor_de_embedded_programmeur!");
7
8     // omdat de LEDs in een matrix staan moet zowel de plus als de min
9     // aangestuurd worden.
10    // De stroom loopt immers van + naar -
11    pinMode(COL1, OUTPUT); //kolom is de -
12    digitalWrite(COL1, LOW);
13    pinMode(LED, OUTPUT); // rij is de +
14 }
15
16 void loop() {
17     Serial.println("blink!");
18     digitalWrite(LED, HIGH);
19     delay(500);
20     digitalWrite(LED, LOW);
21     delay(500);
22 }
```

**Listing 3.1:** Het programma blinkdemo.

Arduino noemt een dergelijk programma een schets (of ‘sketch’). Hierin zijn twee functieblokken herkenbaar: `void setup()` en `void loop()`

Wat in het `setup` blok tussen de accolades staat, wordt éénmalig tijdens het opstarten uitgevoerd.

Als het setup blok klaar is, wordt het loop blok gestart. De code in het loop blok wordt continue herhaald, in principe duizenden keren per seconde. Het programma eindigt dus nooit!

Deze structuur (een opstartblok en een blok dat herhaald wordt) geldt voor alle type microcontrollers!

Aan het begin van de schets, vóór de `void setup()`, is plek voor het declareren van globale variabelen, deze variabelen kun je overal in het programma gebruiken, zowel in de setup als in de loop.

Arduino gebruikt **kleuren** om **functies** of **uitdrukkingen** aan te geven. Als je wilt weten wat een dergelijk woord betekent of hoe je het gebruikt, klik dan in de Arduino IDE op het woord en druk **Ctrl + Shift + F**

Probeer dit uit: Selecteer in Arduino de tekens // en druk **Ctrl + Shift + F**.

De **oranje gekleurde functies** zijn geen deel van de taal C maar zijn gemaakt door Arduino en zijn gedefinieerd in Arduino libraries die standaard geïnstalleerd zijn. Deze functies maken het leven een stuk makkelijker! Ze vervangen een heleboel moeilijk leesbare C code.

### 3.1.1 Code wijzigen.

Wijzig in de schets de eerste `delay(500)` naar `delay(100)`.

Klik op  of druk **Ctrl + U** om het programma te compileren en naar de Microbit te sturen. Je ziet nu dat de ‘aan’ tijd korter is.

#### Uitleg: Gebruik van de seriële poort

In dit voorbeeld wordt de seriële poort gebruikt.

Je kunt de seriële poort gebruiken om te ‘debuggen’, oftewel controleren of je programma doet wat je er van verwacht. In The Challenge kun je het gebruiken om sensordata naar je PC te sturen.

Open de seriële monitor(Tools→Serial Monitor) of (druk **Ctrl + Shift + M**) of klik op 

Het programma drukt telkens ‘blink!’ af.

Zet “Show Timestamp” aan om de regels van elkaar te kunnen onderscheiden.

Zorg dat de ingestelde snelheid overeen komt met de instelling `Serial.begin(9600)` in `setup()`. De instellingen van de monitor kan gedaan worden in de statusbalk, zoals te



**Figure 3.1:** Instellingen van de seriële monitor.

zien is in figuur 3.1. Vaak wordt 9600 Baud (bits per seconde) gebruikt. Dit is gedaan omdat hiermee de data praktisch altijd stabiel verzonden wordt.

### 3.1.2 Commando's verklaren.

Geef in je eigen woorden weer wat de volgende commando's doen (zoek uit via **Ctrl + Shift + F**)  
`pinMode(LED, OUTPUT);`

---

`digitalWrite(LED, LOW);`

---

`delay(500);`

---

Wat denk je dat het grootste nadeel van `delay()` is?

---

### 3.1.3 Dimmen van een led

We hebben al gezien dat je een pin als uitgang in kunt stellen en deze aan en uit kunt zetten.

We gaan dezelfde pin nu op een andere manier gebruiken. We gaan de LED nu geleidelijk feller en minder fel laten branden, het Engelse werkwoord hiervoor is ‘to fade’.

1. We gaan hierbij uit van de code uit de blinkdemo van listing 3.1. Het knipperen van de LED wordt in de `loop` gedaan, zoals in listing 3.2 te zien is.

```
15
16 void loop() {
17
18     digitalWrite(LED, HIGH);
19     delay(500);
20     digitalWrite(LED, LOW);
21     delay(500);
22 }
```

**Listing 3.2:** Het knipperen van de LED.

Hierbij wordt met het statement `digitalWrite(LED, HIGH);` de LED aangezet en met het statement `digitalWrite(LED, LOW);` de LED uitgezet. In plaats van `digitalWrite` kan in sommige gevallen ook `analogWrite` gebruikt worden. Hiermee kan opgegeven worden hoeveel licht de LED geeft.

Sla de blink op onder een andere naam, bijvoorbeeld Microbit-Fade. Verander de statement `digitalWrite(LED, HIGH);` in `analogWrite(LED, 10);` Klik vervolgens op  of druk **Ctrl + U** om het programma te compileren en naar de Microbit te sturen. Verklaar wat er gebeurd.

Verander de 10 in een ander getal b.v. 70 en verklaar wat er gebeurd. De maximale waarde waarop de LED ingesteld kan worden is 255. Dit heeft te maken met het feit dat de waarde maar 8 bits groot is en  $2^8 - 1 = 255$ .

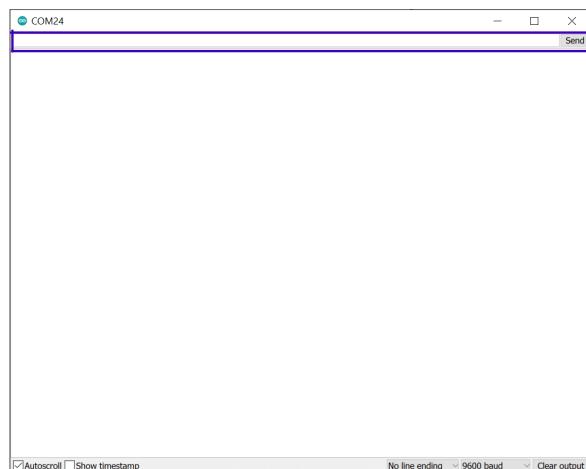
2. Pas het programma verder aan, zodat de LED steeds meer licht gaat geven. Indien de hoeveelheid licht die de LED geeft maximaal is, gaat de LED weer minder licht geven totdat de LED geen licht meer geeft, hierna gaat de LED weer meer licht geven, enz. Zoals weergegeven in het volgend filmpje te zien is. **Lever deze opgave in op blackboard.**

Het mechanisme dat hier gebruikt wordt om de LED te dimmen heet PWM.

*Met PWM kun je niet alleen een led minder fel laten branden, maar zo kun je ook de snelheid van een motor regelen. De piepjes die je hoort bij een optrekkende tram of trein worden veroorzaakt door PWM aansturing.*

### 3.1.4 Het ontvangen van informatie vanaf de arduino-monitor en de beide knoppen.

Behalve dat de microbit data naar de monitor kan verzenden, kan deze ook data van de monitor lezen. Dit kan gedaan worden door de data in het bovenste vierkant (zoals in figuur 3.2 wordt weergegeven) in te typen en vervolgens op **Send** te klikken. Om



**Figure 3.2:** Inputveld van de arduino monitor.

in je programma de data te lezen, zal eerst gecheckt moeten worden of er al data

via de seriële poort binnen gekomen is, dit kan gedaan worden met het statement: `if(Serial.available() >0) { }` daarna kan de data pas gelezen worden. Dit wordt gedaan met het statement: `int getal=Serial.read();`, zoals te zien is in listing 3.3.

```
int getal=0;

void setup() {
    Serial.begin(9600);
    Serial.println("Hoi _embedded _programmeurs!");

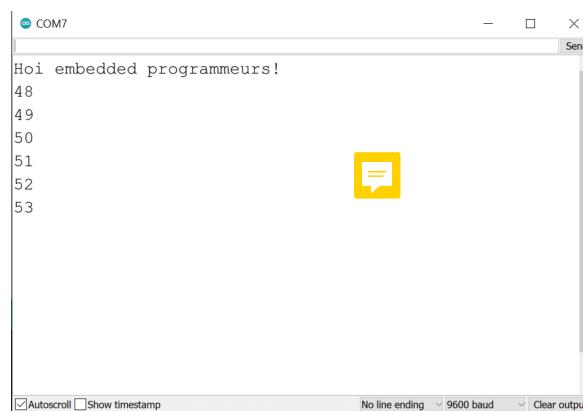
}

void loop(){
    if (Serial.available() > 0) { // is er een character ingevoerd?
        getal= Serial.read();
        Serial.println(getal);
    }
}
```

**Listing 3.3:** inlezen via de seriële poort.

Opdracht:

1. Voer de code van Listing 3.3 uit en type in het invoerveld van de monitor de cijfers 012345 in, gevolgd door een enter. Als het goed is verschijnt Figuur 3.3



**Figure 3.3:** Uitvoering van listing 3.3

Wat opvalt is dat in plaats van het getal 012345 de getallen 48, 49, 50, 51, 52 en 53 worden uitgeprint. Dit heeft te maken dat het statement `int getal=Serial.read();` een character inleest, waarbij elke character een waarde heeft volgens de [ASCII](#) tabel.

2. Pas het programma van listing 1 zodanig aan, zodat na invoering van je studienummer en gebruik van het statement `int getal=Serial.read();` ook daadwerkelijk [je studienummer als een integer uitprint kan worden](#).  
Tip: kijk naar de sheets over het **decimale talstelsel met grondtal 10**.

## 3.2 De (externe) pinnen van de microbit

De Microbit heeft 21 pinnen (aansluitingen) die voor allerlei toepassingen bruikbaar zijn. In figuur 3.4 zijn de externe aansluitingen van de microbit te zien. Hierbij

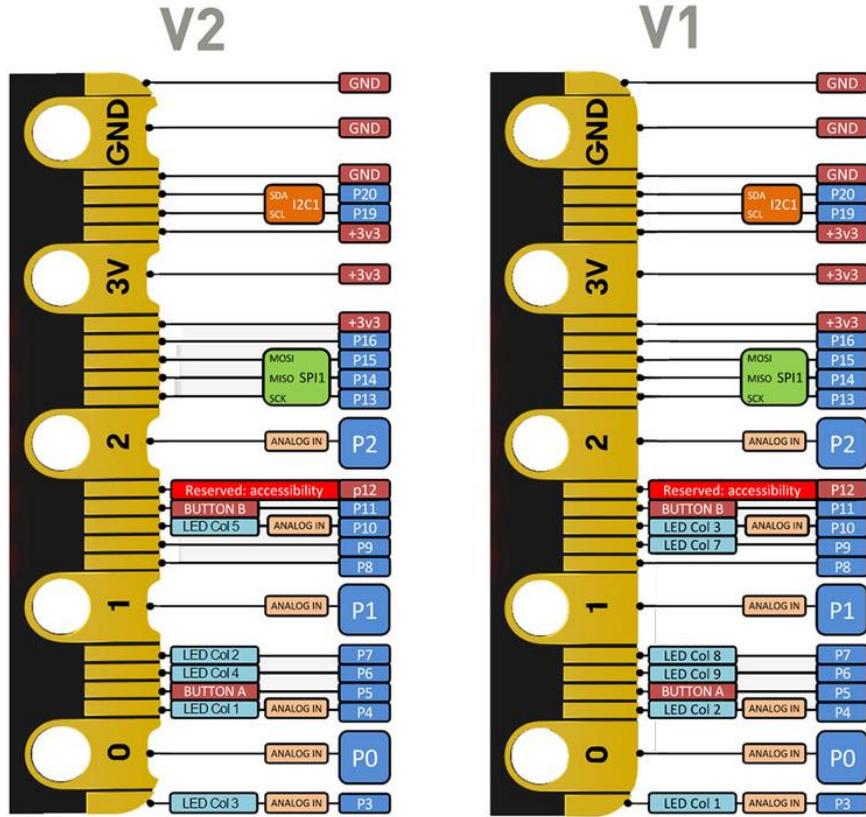


Figure 3.4: Pinlayout van de microbit.

wordt aangegeven welke onderdelen van de microbit ook extern zijn te gebruiken. Bij de pinnen waarbij **ANALOG IN** staat, kan een externe analoge signaal van b.v. een externe sensor aangesloten worden. De pinnen genaamd **P..** zijn de pinnummers. In de Arduino IDE kan je die P weglaten en hoef je alleen het nummer op te geven, 0 t/m 20. Uitgebreidere informatie over de pinnen is te vinden op de website [tech.microbit.org/hardware/edgeconnector/](http://tech.microbit.org/hardware/edgeconnector/).

**Upload deze opgave op blackboard.**

### 3.2.1 Het lezen van externe signalen met de microbit.

Stel op PIN P0 willen we een externe analoge sensor aansluiten. Voor de spanningswaarde die de sensor levert willen we een digitale waarde weten. Dit kan in de Arduino omgeving gedaan worden met het statement:

```
int sensorwaarde = analogRead(A0);
```

1. In programma listing 3.4 wordt het analoge signaal van de externe PIN P0 gelezen.

```

void setup() {
    pinMode(0, INPUT);
    Serial.begin(9600);
    Serial.println("Hoi _embedded_programmeurs!");
}

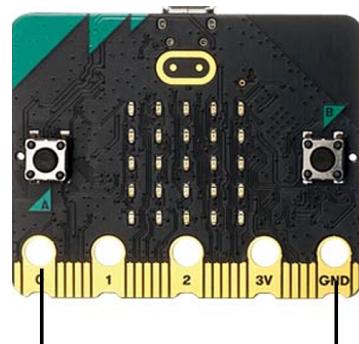
void loop(){
    // lees de analoge waarde op pin 0:
    int sensorValue = analogRead(A0);

    // print de ingelezen analoge waarde
    Serial.println(sensorValue);
    delay(100);           // delay
}

```

**Listing 3.4:** inlezen via een analoog signaal.

- (a) Maak een nieuw Arduino sketch aan en zet listing 3.4 erin, Klik vervolgens op of druk **Ctrl + U** om het programma te compileren en naar de Microbit te sturen.
  - (b) Zoek PIN 0 in Figuur 3.4.
  - (c) Leg je Microbit op tafel, raak hem niet aan.
  - (d) Raak met je vinger het gouden vlakje van pin 0 aan en kijk in de **Seriële Plotter** wat er gebeurt.
  - (e) Tussen welke waardes op de linker as liggen de meetwaardes? \_\_\_\_\_
  2. Verander he statement `analogRead(A0);` in `digitalRead(A0);` Klik vervolgens op of druk **Ctrl + U** om het programma te compileren en naar de Microbit te sturen. Raak met je vinger het gouden vlakje van pin 0 aan en kijk in de **Seriële Plotter** wat er gebeurt.
  3. Bij het inlezen van een digitaal signaal is het niet wenselijk dat bij het aanraken van een draadje of een gouden vlakje de ingangswaarde instabiel wordt. Om dit te voorkomen kan bij Arduino de `INPUT_PULLUP` aangezet worden. Dit wordt gedaan met het statement: `pinMode(0,INPUT_PULLUP);`
- Zet de `INPUT_PULLUP` aan door het statement `pinMode(0,INPUT_PULLUP);` te plaatsen in de `setup` en bekijk wat het resultaat is bij het inlezen van de digitale en analoge waarden. Indien je een draadje bij de hand heb, verbindt de GND met PIN 0, zoals te zien is in figuur 3.5, het gevolg van de `PULL_UP` zal zijn dat alleen nog de waarde 0 ingelezen wordt indien de pin ook daadwerkelijk verbonden wordt met de GND.



**Figure 3.5:** Input verbonden met de GND

Zoals je ziet is de ingang ongevoelig als je `INPUT_PULLUP` aanzet. Je gebruikt `INPUT_PULLUP` als je een ingang naar een stabiele 'hoog' toestand wilt hebben. Je gebruikt deze modus samen met `digitalRead()` als je een schakelaar wilt uitlezen. Als je de schakelaar indrukt wordt de pin laag.

De knopjes op de Microbit gebruiken een **externe** pull-up weerstand, met `INPUT_PULLUP` zet je een pull-up weerstand in de microcontroller aan.

Kijk eens online voor meer uitleg over de principes van een `pull-up weerstand`.

Voor verdere uitleg zie [www.arduino.cc/en/Tutorial/DigitalPins](http://www.arduino.cc/en/Tutorial/DigitalPins)

### 3.2.2 Een denderende schakelaar.

Behalve dat bij het inlezen van de waarde van een externe schakelaar regelmatig een `pull_up` weerstand noodzakelijk is. Treedt er nog een ander probleem op, namelijk een schakelaar die dindert ook wel bouncing genaamd.

Indien een schakelaar wordt ingedrukt sluit deze bijna nooit in 1 keer, maar veert een aantal keren op en neer. Dit fenomeen wordt weergegeven in Figuur 3.6, waar te zien is dat bij het indrukken en loslaten van het knopje de schakelaar 'stuitert'. Één

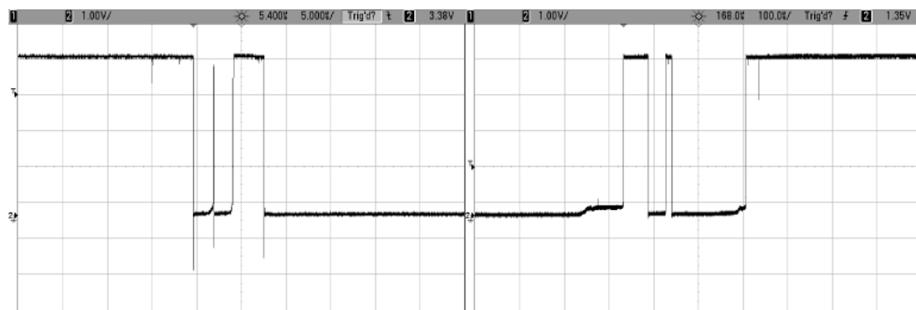


Figure 3.6: Een denderende schakelaar[1].

van de eenvoudigste manieren om dit op te lossen is door even te wachten (een paar milliseconden) en vervolgens te controleren of de knop nog steeds is ingedrukt. Dit wordt weergegeven in listing 3.5

```
1 const int knopA = 5;      // Knop A is aangesloten op poortnummer 5
2 const int knopB = 11;     // Knop B is aangesloten op poortnummer 11
3
4 void setup() {
5   Serial.begin(9600);
6   Serial.println("microbit_is_ready!");
7
8   pinMode(knopA, INPUT);
9   pinMode(knopB, INPUT);
10 }
11 boolean isKnopIngedrukt(int); // declaratie van de funcie
12
13 void loop(){
14
15   if( isKnopIngedrukt(knopA) )
16     Serial.println("A_is_ingedrukt");
17 }
```

```

18 if( isKnopIngedrukt(knopB) )
19   Serial.println("Buis ingedrukt");
20
21   delay(500);
22 }
23
24 boolean isKnopIngedrukt( int knop) {
25   if ( ! digitalRead(knop)) { //Is de knop soms ingedrukt ?
26     delay(2);           //wacht 2 miliseconde
27     if ( ! digitalRead(knop)) { //Is knop nog steeds ingedrukt
28       return true;
29     }
30   }
31   return false;
32 }
```

**Listing 3.5:** Omgaan met een denderende schakelaar.

Hierbij wordt in de functie `boolean isKnopIngedrukt (int)`; als eerste gekeken of de knop is ingedrukt, vervolgens wordt er 2 miliseconde gewacht, waarna opnieuw gekeken wordt of de knop is ingedrukt. Is dit zo dan wordt een `true` mee teruggegeven anders een `false`.

In listing 3.6 is te zien dat er een detectie plaatsvindt wanneer de toestand van de schakelaar verandert.

```

1
2 const int COL1 = 4;
3 const int ROW1 = 21;
4 const int knopA = 5;      // Knop A is aangesloten op poortnummer 5
5 const int knopB = 11;      // Knop B is aangesloten op poortnummer 11
6
7 boolean isKnopIngedrukt(int);
8
9 int knopTeller = 0;
10 boolean knopToestand = false;
11 boolean vorigeKnopToestand = false;
12 int ledStatus = LOW;
13
14 void setup() {
15   Serial.begin(9600);
16   Serial.println("microbit_is_ready!");
17   pinMode(COL1, OUTPUT);
18   pinMode(ROW1, OUTPUT);
19   pinMode(knopA, INPUT);
20   pinMode(knopB, INPUT);
21 }
22
23 void loop(){
24   knopToestand = isKnopIngedrukt(knopA);
25   if(knopToestand != vorigeKnopToestand) //heeft er een verandering
26     plaatsgevonden?
27   if (knopToestand == true) {
28     knopTeller++;
29     ledStatus = !ledStatus;
30     Serial.println(knopTeller);
31 }
```

```

31 delay(50);
32 vorigeKnopToestand = knopToestand;
33 digitalWrite(ROW1, ledStatus);
34 }
35
36 boolean isKnopIngedrukt( int knop) {
37 if (! digitalRead(knop)) { //Is de knop soms ingedrukt ?
38   delay(2);           //wacht 2 milliseconde
39   if (! digitalRead(knop)) { //Is knop nog steeds ingedrukt
40     return true;
41   }
42 }
43 return false;
44 }

```

**Listing 3.6:** Een toestandverandering van de schakelaar.

Open het voorbeeldprogramma `veranderToestandDrukknop.ino` en upload het naar de Microbit of maak een nieuw arduino project aan en [kopieer listing 3.6](#). Bestudeer de code en kijk hoe het programma werkt. Het is de bedoeling dat na elke druk op knopA de led aan of uit gaat (wisselt van toestand) en dat de teller steeds met 1 opgehoogd wordt.

(A) In figuur 3.7 wordt het signaal van de button weergegeven. In welke toestand (1 t/m 4 ) bevindt zich de variabele `knopToestand` en `vorigeKnopToestand` indien de toestand van de LED verandert en de teller met 1 verhoogd wordt?



**Figure 3.7:** Weergaven van het signaal, indien de button wordt ingedrukt en weer wordt losgelaten.

t/m 4 ) bevindt zich de variabele `knopToestand` en `vorigeKnopToestand` indien de toestand van de LED verandert en de teller met 1 verhoogd wordt?

- (B) Wat is het nut van het statement `delay(50)` op regel 31?\_\_\_\_\_
- (C) Wat is het resultaat indien de outputpin COL1 van listing 3.6 op `HIGH` gezet wordt?
- (D) Breid listing 3.6 uit, zodat de teller met 1 verlaagd wordt indien met knopB een toestand verandering van 3 naar 4 plaatsvindt.

**Upload deze opgave op blackboard.**

# 4 De bewegingssensor (week 3).

De bewegingssensor, welke in rood wordt aangegeven in figuur 4.1, bestaat uit twee hoofdonderdelen, een accelerometer en een kompas. We gaan eerst aan de slag met een voorbeeld dat de bewegingssensor demonstreert, vervolgens met de magnetometer. Waarna beide onderdelen in 1 programma worden bewerkt.

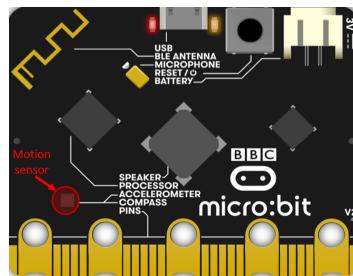


Figure 4.1: Plaats van de bewegingssensor.

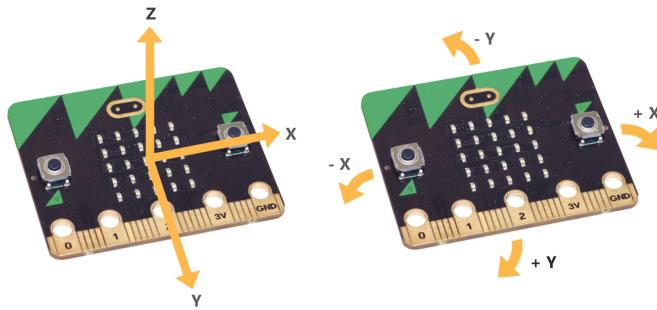
1. Installeer de STM32duino LSM303AGR
2. Open het voorbeeldprogramma accelerator.
3. Compileer en upload het voorbeeld naar je Microbit.
4. Als het programma werkt, dan knippert de led linksboven.

## 4.1 De accelerator

Stel dat ik in The Challenge de Microbit wil gebruiken om te kijken of de deksel van een container ver genoeg opengaat, dan moet ik dat kunnen meten. De bewegingssensor is daar ideaal voor. De accelerometer meet de versnelling (beweging) in zowel de X, Y en de Z richting, zoals in figuur 4.2 te zien is.

De listing van het programma wordt weergegeven in listing 4.1

```
1 #include <LSM303AGR_ACC_Sensor.h>
2
3 #define DEV_I2C Wire1 // Wire1 is voor de interne I2C bus
4 // #define SerialPort Serial
5 #define LED_ROW1
6
7 // Components.
8 LSM303AGR_ACC_Sensor Acc(&DEV_I2C);
9
10 const int COL1 = 4;
11 const int ROW1 = 21;
```



**Figure 4.2:** Registratie van de accelerator [3].

```

12
13 void setup() {
14
15     pinMode(COL1, OUTPUT); // aansturing linkerboven LED in matrix
16     digitalWrite(COL1, LOW);
17     pinMode(ROW1, OUTPUT);
18
19     SerialPort.begin(9600); // Initialize serial for output.
20
21     DEV_I2C.begin(); // Initialize I2C bus.
22
23     Acc.begin(); // Initiallize accelarator.
24     Acc.Enable();
25
26     uint8_t a;
27     Acc.IO_Read(&a, 0x0F, 1); //lees waarde Who_am_I register.
28     Serial.print("Ik_ben: ");
29     Serial.println(a);
30 }
31
32 void loop() {
33     // Led blinking.
34     digitalWrite(LED, HIGH);
35     delay(250);
36     digitalWrite(LED, LOW);
37     delay(250);
38
39     int32_t accelerometer[3];
40     Acc.GetAxes(accelerometer); // Read accelerometer LSM303AGR.
41
42     // Output data.
43     Serial.print(" | -Acc[mg]: -");
44     Serial.print(accelerometer[0]);
45     Serial.print(" -");
46     Serial.print(accelerometer[1]);
47     Serial.print(" -");
48     Serial.print(accelerometer[2]);
49     Serial.println(" -| ");
50 }
```

**Listing 4.1:** Uitlezen van de accelarator

Op regel 8 wordt een software-accelerator aangemaakt, die in de `setup()` (regel 23 en

24) wordt geïnitialiseerd. In de `loop()` worden de waarden van de accelerator-sensor opgehaald en in een array gezet (regel 40). De opgehaalde waarden worden in de regels 44, 46 en 48 uitgeprint.

Ga als volgt te werk:

1. Open nu de seriële plotter.
2. Kijk naar de seriële plotter en kantel de Microbit langzaam naar links en naar rechts totdat deze verticaal staat. Doe dat ook naar voor en achter. Draai hem om zijn as. Kijk welke waarde verandert.
3. Zoek een waarde die overeenkomt met de richting die je zoekt, zet de Microbit bijvoorbeeld op zijn rechter zijkant. Je ziet twee lijnen die duidelijk meebewegen. Beweeg de Microbit terwijl je hem op zijn zij hebt op en neer.
4. Bepaal nu (met de seriële monitor) ongeveer de waarde die deze lijn heeft bij een hoek van 45 graden. We maken een aanpassing in het programma zodat deze een melding geeft als de hoek groter is dan 45 graden. In dat geval zetten we de led linksboven aan.
5. **Opdracht:** Zoek de waardes uit en laat de led aangaan boven de 45 graden en weer uitgaan onder de 45 graden. Makkie, toch? Beetje simpel. We gaan meerdere ledjes aansturen.
6. **Combineer dit met: Hoe kunnen we de leds (eenvoudig) aansturen?:**  
Open het voorbeeldprogramma `heenEnWeer.ino`
7. Compileer en upload het voorbeeld naar je Microbit.
8. Kijk hoe de aansturing van de LEDs werkt en kopieer de benodigde code uit het voorbeeldprogramma `heenEnWeer.ino` naar de bewegingssensor demo. Zorg eerst dat één ledje gaat branden aan de kant waar je de Microbit naartoe kantelt (alsof je een knikker op een dienblad hebt). Breid dit dan uit naar 4 leds (voor, achter, links, rechts) en uiteindelijk naar 8 leds (de leds op de hoeken erbij).  
Als je dit op een echt slimme manier wilt doen, dan gebruik je geen ‘if’ statements maar bereken je de pixellocatie uit de accelerometer waarden. Gebruik dan wel ‘if’ om te clippen.  
**Upload het programma op blackboard.**

## 4.2 Het kompas

Voor het kompas, geldt in principe hetzelfde alleen moet er nu een software-magnetometer aangeroepen worden. Dit kan gedaan worden met het volgende statement:

`LSM303AGR_MAG_Sensor Mag (&DEV_I2C);`

Vergeet niet de magnetosensor te includen.

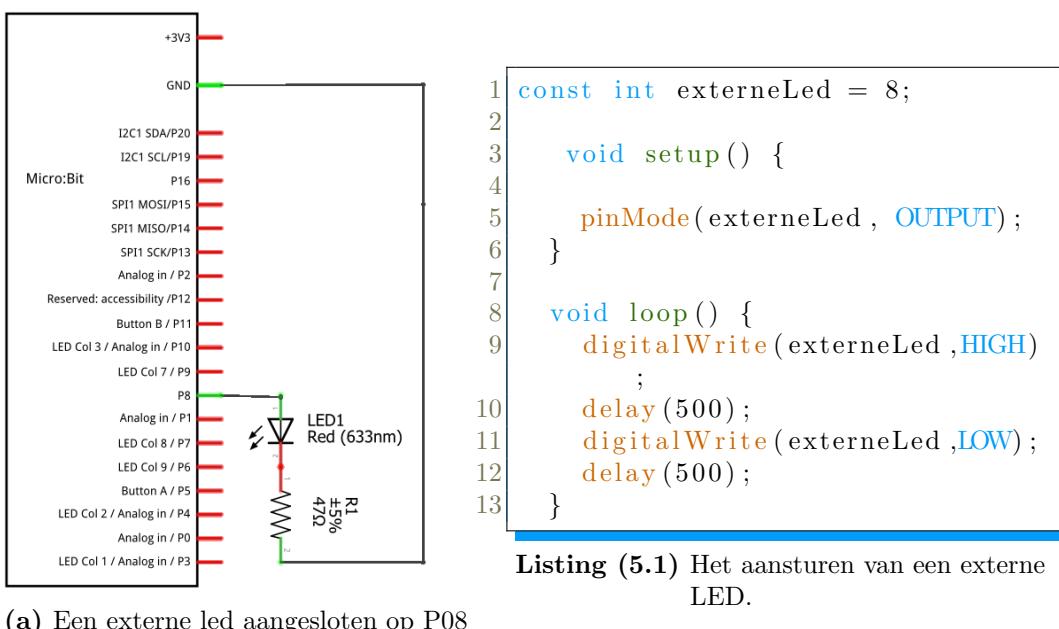
```
#include <LSM303AGR_MAG_Sensor.h>
```

**Opdracht:** Pas Listing 4.1 zodanig aan, zodat behalve de accelerator ook de magnetometer wordt aangeroepen en de waarden worden uitgeprint op de monitor. **Upload het programma op blackboard.**

# 5 Het gebruik van de LED matrix

(week 4).

Indien we een externe LED op de microbit aansluiten, zoals in figuur 5.1a wordt weergegeven, kunnen we deze direct aansturen, zoals listing 5.1 laat zien. De externe



(a) Een externe led aangesloten op P08

**Figure 5.1:** Het aansturen van een externe LED.

LED wordt aangesloten op poortnummer 8. In de `setup()` wordt dit poortnummer op output gezet. In de `loop()` wordt als eerste een spanning gezet op poort 8 (`digitalWrite(externeLed, HIGH);`) waardoor een elektrische stroom door de LED gaat lopen en deze licht gaat geven. Hierna wordt een halve seconde gewacht, daarna wordt op poort 8, 0 Volt gezet (`digitalWrite(externeLed, LOW);`) en gaat de LED uit (er loopt geen elektrische stroom meer door de LED). Kijken we naar de listings 3.1 en 3.6, dan zien we dat er twee output poorten (4 en 21) nodig zijn om de linkerboven LED in het matrix display aan te sturen.

micro:bit	pinnummer
COL1	4
COL2	7
COL3	3
COL4	6
COL5	10
ROW1	21
ROW2	22
ROW3	23
ROW4	24
ROW5	25

**Table 5.1:** Microbit-matrix en de arduino pinnummers.

Dit heeft te maken met het principe van een matrix, zoals in figuur 5.2 te zien is. Om zoals in het programma blink ( listing 3.1), LED D2 (figuur 5.2) licht te laten geven, zal er een elektrische stroom moeten lopen van ROW1 naar COL1. Om dit voor elkaar te krijgen moet zowel ROW1 als COL1 op output gezet worden, verder zal COL1 **LOW** en ROW1 **HIGH** gemaakt moeten worden, zoals te zien is in listing 5.2.

```

1 const int COL1 = 4;
2 const int ROW1 = 21;
3
4 void setup() {
5     pinMode(COL1,OUTPUT); //zet de port COL1 op output
6     pinMode(ROW1,OUTPUT); //zet de port ROW1 op output
7     digitalWrite(COL1, LOW); //zet COL1 op 0 Volt
8     digitalWrite(ROW1, HIGH); //Zet elektrische spanning op ROW1
9 }
10
11 void loop() {
12 }
```

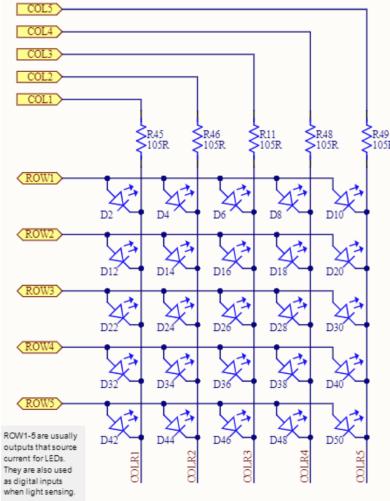
**Listing 5.2:** Zet de linkerboven LED van de matrix aan

### Opdracht:

- (A) Open het voorbeeldprogramma matrixLed, compileer het en upload de code naar de micro:bit. Om het programma uit te breiden zodat de LEDs op alle 4 de hoekpunten licht gaan geven, wordt:

- (a) De onderstaande waarheidstabel ingevuld.

LED	COL1	COL2	COL3	COL4	COL5	ROW1	ROW2	ROW3	ROW4	ROW5
D2	0	1	1	1	1	1	0	0	0	0
D10										
..										
..										



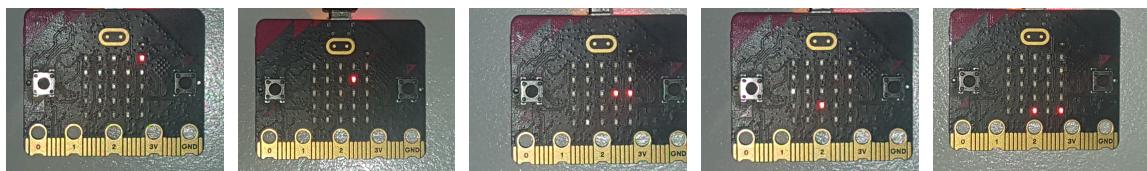
**Figure 5.2:** De LEDs aansluitingen van het matrix display [2].

- (b) De code van Listing 5.2 uitgebreid aan de hand van de waarheidstabel van opdracht a.

**Upload deze opgave op blackboard.**

- (B) Zoals bekent verondersteld mag worden, werkt een computer nog steeds digitaal, of terwijl een '1' of een '0'. Zoals uit opgave A gebleken is, is het besturen van de LEDs in de matrix een kwestie van de juiste outputlijnen HIGH of LOW maken.

Bij deze opgave zetten we als eerst op de bovenste rij de waarde 1, een seconde later de bovenste rij uit en op de 2<sup>e</sup> rij de waarde 2. Een seconde later de 2<sup>e</sup> rij uit en op de 3<sup>e</sup> rij de waarde 3. Een seconde later de 3<sup>e</sup> rij uit en op de 4<sup>e</sup> rij de waarde 4. Een seconde later de 4<sup>e</sup> rij uit en op de 5<sup>e</sup> rij de waarde 5. Daarna wordt er opnieuw begonnen. Het resultaat is te zien in figuur 5.3



**Figure 5.3:** Binair tellen van 1 t/m 5.

Begin als eerste met het invullen van de onderstaande waarheidstabel:

Waarde	COL1	COL2	COL3	COL4	COL5	ROW1	ROW2	ROW3	ROW4	ROW5
1	1	1	1	1	0	1	0	0	0	0
2										
3										
4										
5										

**Upload deze opgave op blackboard.** Tip:

Vergeet niet de pinnen in de output mode te zetten.

# 6 Bitwise operaties (week 5 en 6).

De firma Adafruit heeft voor het matrix display een speciale component gemaakt waardoor het matrix display eenvoudig te programmeren is. Listing 6.1 is hier een voorbeeld van.

```
1 #include <Adafruit_Microbit.h>
2 #define LED0 0 //definieer LEDx tot een waarde
3 #define LED1 1
4 #define LED2 2
5 #define LED3 3
6 #define LED4 4
7 #define LED5 5
8
9 Adafruit_Microbit_Matrix matrixMbit; //maak een LED matrix aan.
10 const uint8_t
11 smile_bmp[] =
12 { B00000,
13   B01010,
14   B00000,
15   B10001,
16   B01110, };
17 uint8_t matrixje[] =
18 { B00000,
19   B00000,
20   B00000,
21   B00000,
22   B00000, };
23
24 void setup() {
25   Serial.begin(9600);
26   Serial.println("Welkom bij Embedded!");
27
28   matrixMbit.begin();
29   matrixMbit.show(smile_bmp);
30   delay(2000);
31   matrixMbit.show(matrixje);
32   delay(2000);
33   //zet rechterLED bovenste rij aan
34   matrixje[0] = 1 << LED0; //schuif 1, LED0 plaatsen op naar links.
35   matrixMbit.show(matrixje);
36   delay(2000);
37   //zet linkerLED bovenste rij aan
38   matrixje[0] = 1 << LED4; //schuif 1, LED4 plaatsen op naar links.
39   matrixMbit.show(matrixje);
40 }
41 void loop() {
42 }
```

**Listing 6.1:** Het aanzetten van een LED

In Listing 6.1 is te zien hoe een LED aangezet kan worden door de betreffende bit in

een array van 8 bits data, een 1 te maken. Dit wordt gedaan door een 1 (0b00000001) een aantal plaatsen naar links op te laten schuiven. Zoals te zien is in onderstaand statement:

```
matrixje[0] = 1 << LED0;
```

De 1 wordt een aantal plaatsen naar links opgeschoven dat door LED0 wordt aangegeven. LED0 staat gedefinieerd op regel 2 van Listing 6.1

## 1. Installeer de Adafruit Libraries

2. (a) Open voorbeeldcode `matrixSmpl` of kopieer Listing 6.1 in een nieuwe Arduino omgeving en voer deze uit.
- (b) Breid het programma zodanig zodat uit, zodat zowel LED0 als LED4 van de 2e rij aangaan. Maak hierbij gebruik van de operatoren '`<<`' en '`|`'(bitwise or).

A	B	A   B
0	0	0
0	1	1
1	0	1
1	1	1

De bitwise OR

- (c) Breid het programma uit zodat 2 seconde nadat beide LEDS uit B aangegaan zijn, LED4 weer uitgaat. Maak hierbij gebruik van de operatoren '`<<`', '`&`' en '`~`'. Do dit zoals in de theorie besproken is.

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

De bitwise AND

- (d) Door met bitwise operaties te werken, kan op een eenvoudige wijze een LED dat aan is, uitgezet worden en een LED dat uit is aangezet worden. Dit kan gedaan worden met de exclusief OR operator, zoals te zien is in onderstaand tabel. Breid het programma uit zodat 2 LEDS van de onderste rij aan en uitgaan door de exclusief OR functie te gebruiken. Maak hierbij onder ander gebruik van de operator `^`. Do dit zoals in de theorie besproken is.

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

De bitwise XOR

- (e) Er kan ook getest worden of een LED aan is met behulp van bitwise operatoren.

```
uint8_t hulpje = matrixje[0];
```

```
if(matrixje[0] ... ....) {  
}.
```

Vul het **if** statement in en toon aan dat een LED aan of uit is.

3. In het volgende voorbeeld gaat steeds een LED van rechts naar links aan.

```
1 #include <Adafruit_Microbit.h>  
2  
3 Adafruit_Microbit_Matrix matrixMbit; //maak een LED matrix aan.  
4  
5 uint8_t matrixje [] =  
6 { B00000,  
7   B00000,  
8   B00000,  
9   B00000,  
10  B00000, };  
11  
12 void setup() {  
13   Serial.begin(9600);  
14   Serial.println("Welkom bij _embedded!");  
15  
16   matrixMbit.begin();  
17   matrixMbit.show(matrixje);  
18   delay(1000);  
19 }  
20  
21 uint8_t nr=0;  
22 uint8_t hulpje;  
23  
24 void loop() {  
25   hulpje = 1 << nr; //schuif 1, nr plaatsen op naar links.  
26   matrixje[0]=hulpje; //De bovenste rij van de matrix krijgt de  
27   matrixMbit.show(matrixje);  
28   delay(1000);  
29   nr++;  
30   if (nr == 5) {  
31     nr=0;  
32   }  
33 }
```

**Listing 6.2:** Looplicht van de bovenste rij.

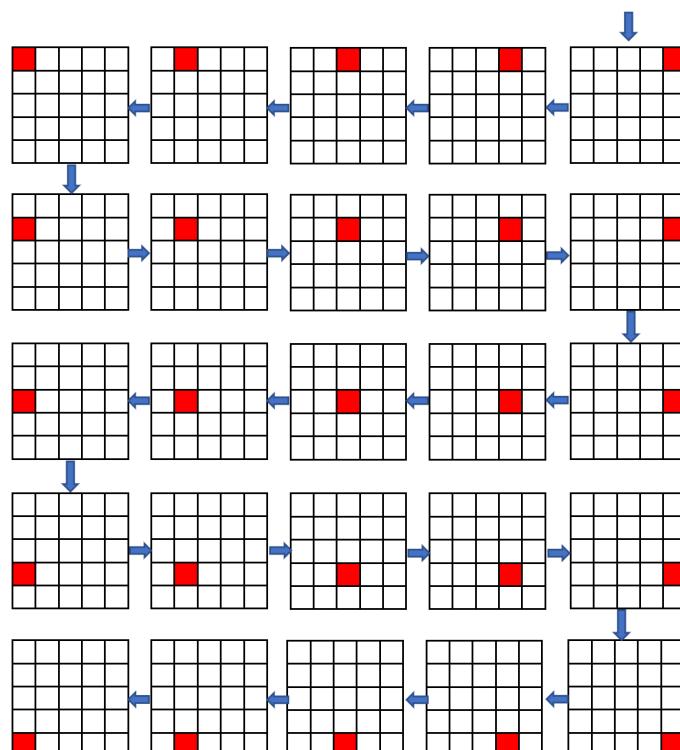
Op regel 5 wordt een matrix component aangemaakt. Het tonen van de matrix wordt met de show functie gedaan (regel 16 en en 26).

- Open voorbeeldcode **matrixloop.ino** of kopieer listing 6.1 in een nieuwe arduino omgeving en voer deze uit.  
Probeer de uitvoer te verklaren.
- Maak een functie **void zetLedAan(uint8\_t rij,uint8\_t kolom);** die de LED op kolom en rij aanzet. Maak hierbij gebruik van de bitwise operator | zoals besproken tijdens de les.
- Maak een functie **void zetLedUit(uint8\_t rij,uint8\_t kolom);** die de LED op kolom en rij uitzet. Maak hierbij gebruik van de bitwise operatoren & en ~ zoals besproken tijdens de les.

- (d) Maak een functie bool `isLedAan(unit8_t rij, uint8_t kolom);` die checkt of de LED op kolom en rij aan is. Maak hierbij gebruik van een bitmasker zoals besproken tijdens de les.
- (e) Pas listing 6.1 met behulp van de bovenstaande functies zodanig aan, zodat:
- Nadat de eerste rij geweest is, bij de tweede rij de ledjes één voor één aangaan.
  - Na de tweede rij bij de derde rij de ledjes één voor één aan gaan.
  - Na de laatste rij bij de eerste rij de ledjes één voor één aan gaan.

**Upload het resultaat op blackboard.**

4. Maak een looplicht dat gaat over alle 5 de rijen van de matrix zoals figuur 6.1 laat zien.



**Figure 6.1:** De volgorde van het looplicht.

Begin rechtsboven (bit 0 van de 0<sup>e</sup> rij in de matrix) en zet vervolgens steeds de LED links aan. Doe dit tot laatste LED van de rij. Ga 1 rij naar beneden en zet vervolgens de LED rechts van de rij aan. Doe dit tot en met het 1<sup>e</sup> bit. Ga 1 rij naar beneden en zet vervolgens de LED links van de rij aan. Doe dit tot en met de laatste LED en ga vervolgens een rij naar beneden. Doe dit tot en met de laatste rij en begin vervolgens weer op de eerste rij, zoals in het volgende filmpje te zien is.

**Upload het resultaat op blackboard.**

# Bibliography

- [1] E. Williams. *Make: AVR Programming*. Maker Media, 2014.
- [2] Schematic bbc mirobit. [https://github.com/microbit-foundation/microbit-v2-hardware/blob/main/V2/MicroBit\\_V2.0.0\\_S\\_schematic.PDF](https://github.com/microbit-foundation/microbit-v2-hardware/blob/main/V2/MicroBit_V2.0.0_S_schematic.PDF), 2021. (Date last accessed 27-July-2021).
- [3] Move accelormeter bbc:microbit. <https://microbit-challenges.readthedocs.io/en/latest/tutorials/accelerometer.html>, 2021. (Date last accessed 7-August-2021).

# Appendices

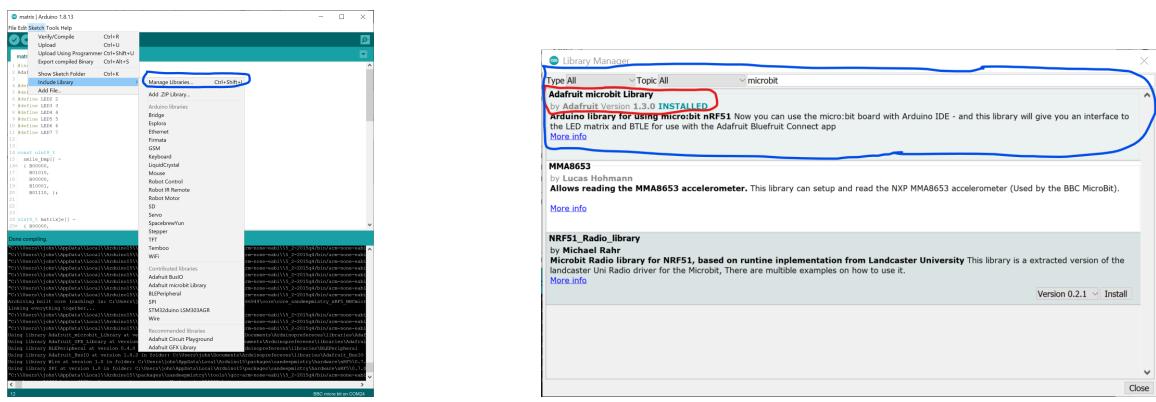


# A Problemen tijdens het practicum.

## A.1 Adafruit\_Microbit.h: No such file or directory.

Controleer of de Adafruit microbit geïnstalleerd is.

1. Ga naar de library manager zoals aangegeven in figuur A.1a
2. Controleer of library geïnstalleerd is, zolas aangegeven in figuur A.1b



(a) selecteren van de library manager

(b) Constrole Library is geïnstalleerd

**Figure A.1:** Controleren of library is geïnstalleerd.

## A.2 SerialPortException: Port name - COM..

java.io.IOException: jssc.SerialPortException: Port name - COM24; Method name - setEventsMask(); Exception type - Can't set mask.

De seriële poort is in gebruik door waarschijnlijk de monitor.

Sluit de monitor af.

## A.3 Tijdens het uploaden

Error: unable to find CMSIS-DAP device Error: No Valid JTAG Interface Configured.  
Error: No Valid JTAG Interface Configured.

Haal de USB stekker uit de laptop en doe deze opnieuw erin.

## B Vereisten aan code voor ‘The Challenge’

Hieronder is aangegeven waaraan jouw code moet voldoen. Dit is deel van de beoordeling!

- Bronvermelding

Je kunt de opdrachten maken door de Arduino voorbeelden te bestuderen en slim samen te voegen. Je mag (moet!) dus code kopiëren. Vermeld in commentaar boven of achter je code waar je de code vandaan hebt. Je kunt bijvoorbeeld de naam van het voorbeeld opgeven of de URL van de webpagina waar je het vandaan hebt. Geen bronvermelding is (ook bij softwareontwikkeling!) plagiaat.

*Zorg dat jouw programma de structuur heeft zoals de voorbeeldcode van Arduino.*

- Gebruik van commentaar

Commentaar staat tussen /\* en \*/ en kan zo over meerdere regels staan of zelfs in een regel tussen programmacode, zoals in het voorbeeld hieronder:  
pinMode(LED\_BUILTIN /\* dit is eigenlijk pin 13 \*/, OUTPUT); De andere manier om commentaar aan te geven is met // Alles wat achter deze 2 slashes staat wordt als commentaar gezien en wordt door de compiler volledig genegeerd. Zo kan je ook (tijdelijk) regels in je programma uitzetten om iets uit te proberen.

- Layout van je programma.

Een nette layout zorgt voor een goede leesbaarheid. In de Arduino omgeving is daar een eenvoudig hulpmiddel voor: ga naar “Tools”, “Auto Format” of druk Ctrl+T.

- Kwaliteit van code

Zorg dat dezelfde code niet op meerdere plaatsen gebruikt wordt. Dat maakt code slecht leesbaar en zeker op microcontrollers gaat dat ten koste van de schaarse geheugenruimte. Gebruik functies.

- Naamconventies

Geef functies en variabelen een betekenisvolle naam en schrijf deze namen in camelCase (elk woord begint met een hoofdletter, behalve de eerste letter). Voorbeelden: int pinCount;  
int myFunction(); Constanten schrijf je in hoofdletters, woorden scheiden met underscore “\_”. Voorbeeld:  
#define BEEP\_PIN 3

- Versiebeheer

Geef bestanden een betekenisvolle naam, liefst met een versienummer, zodat je ze later eenvoudig kunt terugvinden. Voorbeeld: “Thermometer-v0.1”. Doe dit met al je bestanden!

**Tip:** Sla je programma regelmatig, na elke nieuwe tussenstap op met een nieuw versienummer. Zo kan je altijd terug naar een werkende versie als er iets misgaat.

Beschrijf wijzigingen en versies als commentaar in je programma, dan hoef je niet veel te zoeken!

# C Data vanuit de Microbit naar SQL sturen

Dit is het meest complexe deel van de software die je nodig hebt voor ‘The Challenge’.

Hierin komen 3 vakgebieden samen: Embedded, Java en Databases (SQL).

Bedenk: Dit is slechts een voorbeeld dat laat zien hoe je dat voor de verschillende delen aanpakt.

We gaan er mee om zoals met de eerdere Arduino voorbeelden, zie dit als legoblokjes:

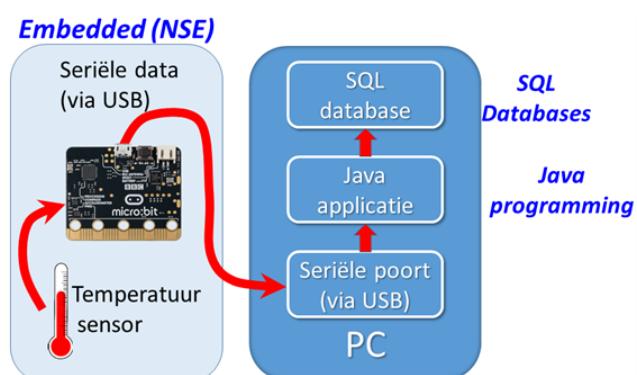
- Maak je eigen code erbij voor de Microbit.
- In het Java deel voegen jullie een GUI toe.
- In de database bouwen jullie een structuur met minstens 2 tabellen.

Het plaatje hieronder geeft weer wat we gaan doen, zie deze stappen:

Stap 1: De Microbit meet temperatuur en geeft dit per seconde door via de seriële poort naar de PC.

Stap 2: Op de PC draait een Java programma die de seriële data vertaalt naar iets bruikbaars.

Stap 3: Het Java programma stopt de data in de SQL database.



**Figure C.1:** Micro:bit naar SQL

Om te zien hoe dit werkt, kijk naar het filmpje ["Microbit: Embedded + Java + SQL datalogging"](#)

## C.1 Software installeren

Ik ga er vanuit dat je de volgende software geïnstalleerd hebt:

- IntelliJ IDE voor Java
- MySQL database

Zo niet, dan staat hieronder waar je het kunt vinden. Zoek zelf uit via de instructies van het vak wat daar precies vereist is.

**IntelliJ:** Deze hebben jullie al geïnstalleerd, daarom geen instructies. Mocht je het nog niet hebben, dit is de link: <https://www.jetbrains.com/idea/download/index.html>

**MySQL:** Als je Windows gebruikt, download en installeer dan de MySQL Installer (web community versie) vanaf: <https://dev.mysql.com/downloads/installer/>

In de installer, selecteer de nieuwste versie van:

- -MySQL Server
- -MySQL Workbench

(Op 7-12-2019 is dat versie 8.0.18)

## C.2 Voorbeeld downloaden

Ga in Blackboard naar de map “Benodigheden voor The Challenge”, “Tips, voorbeeldcode etc.”

Lees de tekst in de sectie “Informatie, hulpmiddelen en voorbeeldcode”.

Download de voorbeeldcode [USBserialReceiveToSQL.zip](#)

Dit bevat het complete project voor het uitlezen van data vanuit de seriële poort en dit in SQL stoppen.

Pak het zip bestand USBserialReceiveToSQL.zip uit naar waar je wilt (map met andere java projecten?) Je hebt daar nu een map USBserialReceiveToSQL

Start IntelliJ, blader naar bovenstaande map (selecteer de map USBserialReceiveToSQL) en klik OK.

Kies “New window” om het project in een nieuwe instantie van IntelliJ te starten.

## C.3 Stap 1: Elke seconde een meetwaarde genereren met de Microbit.

Je kunt op verschillende manieren met de Microbit data genereren om naar de PC te sturen.

Ik heb ervoor gekozen om de temperatuur te meten (elke seconde een meetwaarde; een float). Je kunt zelf andere data kiezen, zoals bijvoorbeeld iets van de accelerometer of een code versturen als je een knopje indrukt. **Je hoeft dus niet elke seconde iets te versturen!** *De Java code wacht eeuwig of er data verzonden wordt vanuit de seriële poort!*

De voorkeursmethode voor programmeren is via Arduino C/C++ code:

1. Open Bestand -> Voorbeelden -> Microbit-HHS -> 07.Temperatuursensor-Bluetooth
2. Compileer en upload het voorbeeld naar je Microbit.
3. Open de seriële monitor (**Ctrl+Shift+M**), je ziet nu de temperatuurwaarden voorbij komen.
4. Sluit de seriële monitor weer, anders kan het Java programma er niet bij (er kan maar 1 programma toegang tot de seriële poort hebben).

Een alternatieve methode is via Makecode, zie:

<https://makecode.microbit.org/94Ag85J3s4Yk>

Je kunt de uitvoer hiervan ook bekijken met de Arduino seriële monitor. Vergeet niet na het kijken de seriële monitor te sluiten!

## C.4 Stap 2: Data inlezen met Java programma.

Als het goed is heb je zojuist het voorbeeldprogramma geopend. Ga in IntelliJ naar ComPortSendReceive.java en comment de volgende regels voor nu even uit (met //):

Dit voorkomt dat het programma data schrijft naar de SQL database. Dat kan nog niet, want die heb je nog niet ingericht.

Zorg dat je Microbit aangesloten is en data genereert.

Start in IntelliJ het programma door op het groene driehoekje te klikken (of druk **Shift+F10**).

Als het goed is zie je in IntelliJ nu de data voorbij komen (datum+tijd en temperatuur). (met //):

```
Regel 87: // InsertIntoSQL database = new InsertIntoSQL();
```

```
Regel 120: // database.insert(tijdstip, temperatuur);
```

Dit voorkomt dat het programma data schrijft naar de SQL database. Dat kan nog niet, want die heb je nog niet ingericht.

Zorg dat je Microbit aangesloten is en data genereert.

Start in IntelliJ het programma door op het groene driehoekje te klikken (of druk **Shift+F10**).

Als het goed is zie je in IntelliJ nu de data voorbij komen (datum+tijd en temperatuur).

## C.5 Stap 3: Data wegschrijven naar MySQL database.

Voer de volgende commando's uit in de SQL server om een database, tabel en user aan te maken:

- CREATE DATABASE vb1;
- CREATE TABLE vb1.tbl1(tijdstip TEXT, temperatuur FLOAT);
- CREATE USER microbit IDENTIFIED BY 'geheim';
- GRANT INSERT, UPDATE, SELECT, DELETE ON vb1.\* TO 'microbit';

Haal de eerder aangebrachte comments in de Java code weg (regels 87 + 120) en start het programma opnieuw. Als het goed is draait het programma nu, zonder foutmeldingen.

Nadat het Java programma data ingevoerd heeft in de database, dan kun je de data opvragen met dit commando:

- SELECT \* FROM vb1.tbl1;

Als je in IntelliJ een foutmelding over de tijd krijgt, dan staat waarschijnlijk in MySQL de tijdzone niet goed. Voer dan dit commando uit op de SQL server:

- SET GLOBAL time\_zone = "+1:00";

*Ik ga hier geen uitleg geven over SQL, ik neem aan dat je die in het vak zelf gekregen hebt. Ook als IntelliJ het niet doet: zoek hulp bij je Java docent!*

Als je er niet uitkomt, kan deze uitleg van Wouter Pijnacker Hordijk je helpen met foutzoeken en het begrijpen van het programma (totale speelduur ruim een uur):

<https://hhs.mediamission.nl/Mediasite/Play/ca2f6403d08e4c86991015797c1c392a1d>

Klik op de (i) rechtsonder in het scherm voor een inhoudsopgave en spring naar je onderwerp van keuze.

## C.6 Stap 4: Eigen aanpassingen maken in Java

Ga in IntelliJ naar ComPortSendReceive.java en bestudeer de code.

Met deze code kun je data verzenden en ontvangen tussen Java en Microbit en SQL.

De door Java ontvangen data wordt verwerkt in onderstaand stuk code.

Daar moeten je aanpassingen gebeuren.

Bedenk: **Alle data wordt ontvangen en verzonden als tekst!** Als je data al tekst is, hoef je dus niks om te zetten! Zie de regel **String naam = berichtData;**

Hieronder zie je de originele code uit het Java programma met een aantal regels in kleur gemarkerd.

De grijs gemarkeerde regels zijn de originele regels die uitgecomment zijn.

De geel gemarkeerde regels zijn de nieuw toegevoegde regels die met tekst (i.p.v. float) werken.

In het voorbeeld heb ik expres float gebruikt om te laten zien dat je het naar een voor SQL geschikt datatype kunt omzetten.

Als je geen tijdstip nodig hebt in SQL, dan kun je dat er bij de database.insert () ook uithalen.

Wil je een ander datatype in SQL stoppen? Kijk dan in de grijs gemarkeerde regels, daar komen je wijzigingen.

```
// StringBuider naar String converteren
String berichtData = bericht.toString();

// tijdstip = nu
String tijdstip = new SimpleDateFormat("yyyy-MM-dd_HH:mm:ss").format(new Date());

// regeleindes verwijderen uit data en tijdstip
berichtData = berichtData.replace("\n", "").replace("\r", "");
tijdstip = tijdstip.replace("\n", "").replace("\r", "");

// String naar float omzetten
// Float temperatuur = Float.parseFloat(berichtData);
String naam = berichtData;

// afronden op 1 cijfer achter de komma
//temperatuur = (float) (Math.round(temperatuur * 10.0) / 10.0);
if (tijdstip.equals(vorigTijdstip)) { // soort "debounce"
    System.out.println("Regel_uit_buffer_genegeerd:");
} else {
    //database.insert(tijdstip, temperatuur); //Deze regel uitcommenten als SQL nog niet werkt.
    database.insert(tijdstip, naam); //Deze regel uitcommenten als SQL nog niet werkt.
}

System.out.print(tijdstip);
System.out.print(" - ");
//System.out.println(temperatuur);
System.out.println(naam);
```

Ga in IntelliJ naar InsertIntoSQL.java en maak je aanpassingen in **public void insert()**  
Voor verdere informatie: Vraag je Java of Databases docent!