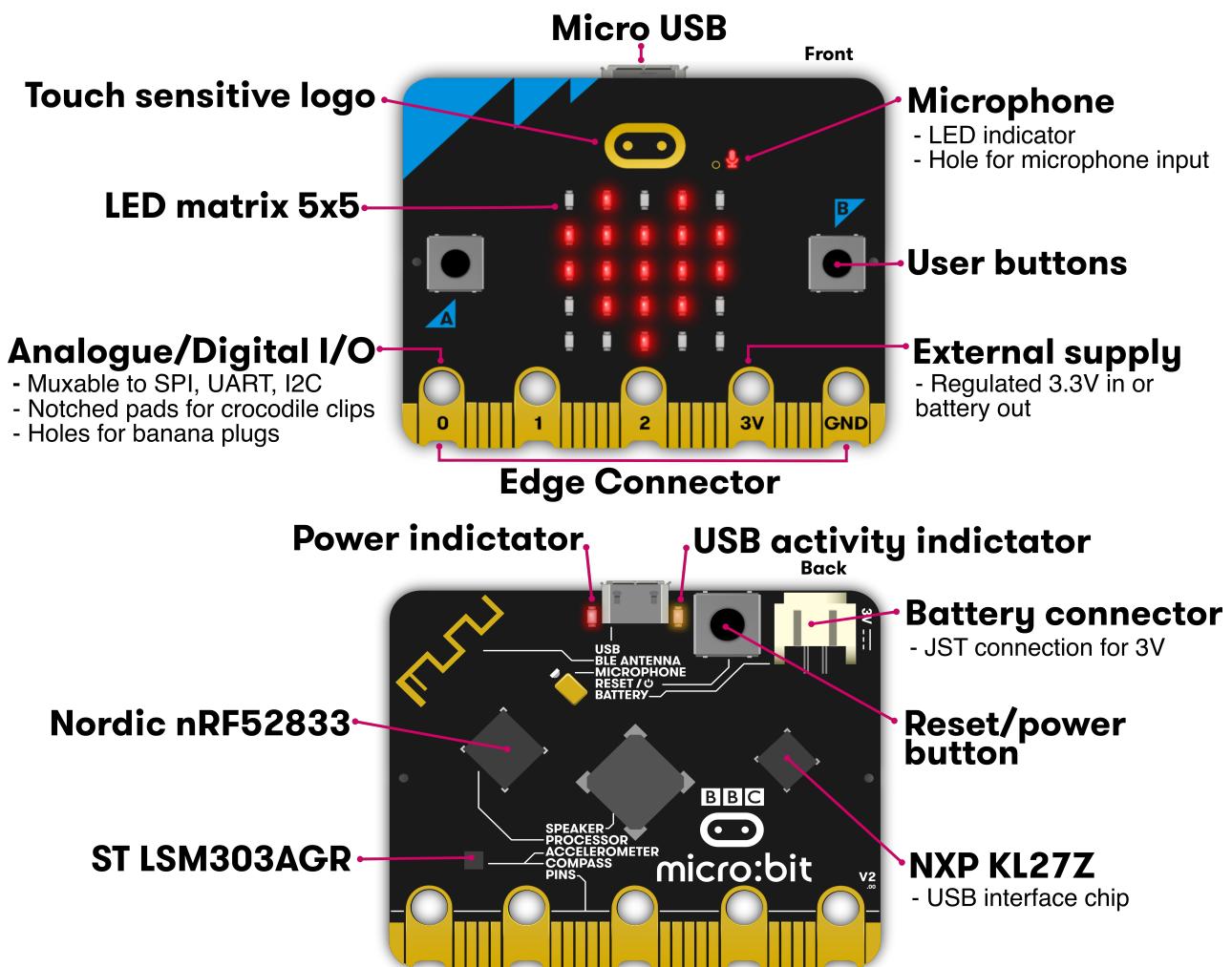


Technical Skills 2

Embedded Systems

(practicumhandleiding micro:bit V2)



Versie 1.34

Contents

1 Inleiding	3
2 Opdracht 1	
(Introductie in de Arduino omgeving (week 10 en 11)).	5
2.1 Hoe zit een Arduino programma in elkaar?	5
2.1.1 Uitleg: Waarom knippert de LED	6
2.1.2 Werken met de LED matrix	7
2.1.3 Het aanzetten van meerdere ledjes in de matrix.	8
2.2 De matrix programmeren met behulp van de Adafruit library	8
2.3 Het gebruik van de seriële poort.	9
2.4 Bitwise operaties.	13
3 Opdracht 2	
(Het inlezen van fysieke eigenschappen(week 12 en 13))	18
3.1 Het inlezen van de stand van de schakelaars.	18
3.2 Een denderende schakelaar.	20
3.3 De (externe) pinnen van de microbit	23
3.3.1 Het lezen van externe analoge signalen met de microbit.	24
3.3.2 De challenge opdracht	25
4 opdracht 3	27
Bibliography	28
Appendices	29
A De practicum omgeving	31
A.1 Start met de BBC microbit	31
A.1.1 Eerste kennismaking met de Microbit	32
A.1.2 De Microbit in “The Challenge”	32
A.2 Kennismaken met de Arduino omgeving	32
A.2.1 Het installeren van de Arduino omgeving	33
A.2.2 De arduino omgeving geschikt maken voor de microbit.	34
A.2.3 Het installeren van de Adafruit library.	36
B Problemen tijdens het practicum.	38
B.1 Adafruit_Microbit.h: No such file or directory.	38
B.2 SerialPortException: Port name - COM..	38
B.3 Tijdens het uploaden	38
B.4 Tijdens compileren ’kan stdint.h niet vinden’	39

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.

In het tweede semester van NSE wordt kennisgemaakt met object georiënteerd modelleren en programmeren in de programmeertaal C++. In het vierde semester van NSE wordt dieper ingegaan op embedded software ontwerpen en programmeren in de programmeertalen C en C++.

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 Brightspace).

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 theorietoets en betreft de opdrachten uit de handleiding. Een aantal opdrachten kan via brightspace 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 eerste opdracht besproken met een aantal basis principes van de arduino omgeving. Tijdens de tweede opdracht in hoofdstuk [3](#) ligt de nadruk op het invoeren van externe gegeven in een embedded systeem.

In de appendix [A](#) wordt de practicum omgeving besproken. In appendix [B](#) wordt voor een aantal problemen een oplossing gegeven, verder wordt ingegaan op mogelijke toepassingen 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 Opdracht 1

(Introductie in de Arduino omgeving (week 10 en 11)).

Hierbij wordt er van uitgegaan dat:

- a) Een [werkende Arduino IDE 2.3.3 omgeving](#) aanwezig is.
- b) De [Arduino IDE 2.3.3 omgeving geschikt is gemaakt voor de microbit](#).
- c) De [adafruit library](#) geïnstalleerd is.

In appendix A is te vinden hoe deze te installeren is.

Verder wordt er vanuit gegaan dat het programmaatje **blink** zoals in listing 2.1 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 en de LED-matrix. Verder zijn er een aantal opdrachten. Deze zijn te vinden bij:

- Hoofdstuk [2.1.3](#) . Het aanzetten van meerdere LEDs **zonder** de Adafruit_Microbit_Matrix op bladzijde [8](#).
- Het aanzetten van meerdere LEDs met de Adafruit_Microbit_Matrix, bladzijde [9](#).
- Het aantal keren laten knipperen van de LED aan de hand van een ingevoerde waarde, bladzijde [11](#).
- Het raden van een binair getal, bladzijde [12](#).
- Het maken van een looplicht, bladzijde [15](#).
- Het besturen van het looplicht met de accelerometer sensor, bladzijde [16](#).

2.1 Hoe zit een Arduino programma in elkaar?

We gaan even terug naar het [blinkdemo](#) voorbeeld uit de installatiehandleiding (zie listing 2.1). Als het goed is heb je deze opgeslagen en kun je deze nu openen.

```
1 const int COL1 = 4;      // Colom #1 control, deze moet 0 zijn
2 const int LED = 21;      //ROW1 is poort 21.Dit is de linker boven LED.
3
4
5 void setup() {
6
```

```

7   // omdat de LEDs in een matrix staan moet zowel de plus als de min
     ↪ aangestuurd worden.
8   //De stroom loopt immers van + naar -
9   pinMode(COL1, OUTPUT); //kolom is de -
10  digitalWrite(COL1, LOW);
11  pinMode(LED, OUTPUT); //rij is de +
12
13 }
14
15 void loop() {
16   digitalWrite(LED, HIGH);
17   delay(500);
18   digitalWrite(LED, LOW);
19   delay(500);
20 }
```

Listing 2.1: 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, je kunt het niet stoppen!

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**.

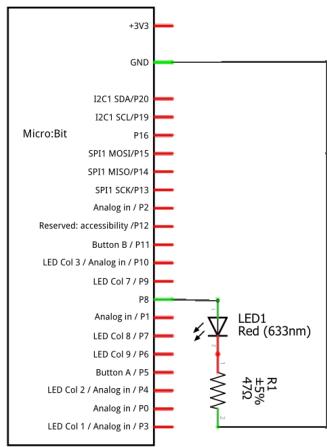
De **oranje gekleurde functies** zijn geen deel van de taal C maar zijn gemaakt door Arduino en zijn gedefinieerd in Arduino libraries die standaard geïnstalleerd zijn. Deze functies maken het leven een stuk makkelijker! Ze vervangen een heleboel moeilijk leesbare C code.

2.1.1 Uitleg: Waarom knippert de LED

Als eerst zal een korte algemene uitleg gegeven worden over het aansturen van een LED, waarna de uitleg volgt in het geval van Listing 2.1 besproken wordt.

Het aansturen van een LED: Een microcontroller heeft verschillende pinnen die kunnen worden gebruikt als ingangen (input) of uitgangen (output). Om een LED aan

te sturen wordt vaak de plus-pin van de LED op één van de microcontroller pinnen gezet, de andere pin via een weerstand verbonden met de min. Figuur 2.1a laat zien hoe een externe LED aangesloten is op de micro:kit. De software om de LED aan te sturen wordt gedaan in Listing 2.2. Hierbij is de externe LED aangesloten op poortnummer 8.



a Een externe led aangesloten op P08

```

1 const int externeLed = 8;
2
3 void setup() {
4     pinMode(externeLed, OUTPUT);
5 }
6
7 void loop() {
8     digitalWrite(externeLed, HIGH);
9     delay(500);
10    digitalWrite(externeLed, LOW);
11    delay(500);
12 }
```

Listing 2.2 Het aansturen van een externe LED.

Figure 2.1: Het aansturen van een externe LED.

De werking van het programma is als volgt.

1. In de `setup()` wordt dit poortnummer op output gezet, (`pinMode (externeLed, OUTPUT)`);
2. In de `loop()` wordt:
 - (a) Een spanning gezet op poort 8 (`digitalWrite(externeLed, HIGH)`;) waardoor een elektrische stroom door de LED gaat lopen en deze licht gaat geven.
 - (b) Hierna wordt een halve seconde gewacht.
 - (c) 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).
 - (d) Hierna wordt een halve seconde gewacht.

2.1.2 Werken met de LED matrix

Bij het Blink programma in Listing 2.1 wordt een LED aangestuurd dat een onderdeel is van de LED-matrix. De aangestuurde LED D2 is aangesloten op de pinnen 4 en 21 aansluitingen van de LED-matrix wordt in figuur 2.2 weergegeven.

Om de LED (↗) D2 aan te zetten, zou een elektrische stroom moeten lopen van pin 21 door LED D2 naar pin 4. Dit kan voor elkaar verkregen worden door als eerst beide pinnen op output te zetten:

```
pinMode(4, OUTPUT);
pinMode(21, OUTPUT);
```

Vervolgens wordt pin 4 laag gemaakt:

```
digitalWrite(4, LOW);
en pin 21 hoog gemaakt
digitalWrite(21, HIGH);
Indien pin 21 na een tijdje
delay(500); weer laag wordt
gemaakt
digitalWrite(21, LOW); zal de
LED uitgaan.
```

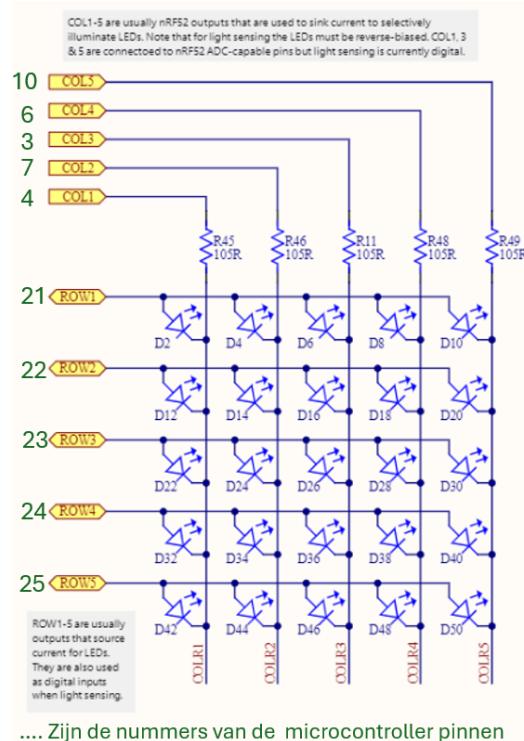


Figure 2.2: Aansluitingen van de LED matrix..

2.1.3 Het aanzetten van meerdere ledjes in de matrix.

1. In het programma *blink* van listing 2.1 knippert de LED met een frequentie van 1 Hz ($\frac{1}{2}$ seconde aan en $\frac{1}{2}$ seconde uit).

- (a) Wijzig in de schets de eerste `delay(500)` (regel 17 van Listing 2.1) naar `delay(100)`.
- (b) Klik op of druk **Ctrl + U** om het programma te compileren en naar de Microbit te sturen.

Je ziet nu dat de ‘aan’ tijd langer is.

2. Laat nu ook LED D6 mee knipperen (LED D2 en D6 knipperen).

3. Laat nu de LEDs D2 en D12 knipperen.

4. Laat nu de LEDs D2, D6 en D12.

Verklaar waarom dit niet eenvoudig lukt en D16 gaat meeknipperen.

2.2 De matrix programmeren met behulp van de Adafruit library

De firma Adafruit heeft voor de micro:bit een library gemaakt waarin de LED matrix een component is, hierbij worden 5 getallen gebruikt die elk een rij voorstelt. Verder zijn er verschillende methode (functies) die gebruikt kunnen worden.

De belangrijkste methoden van de Adafruit_Microbit_Matrix bibliotheek zijn:

- begin() om de matrix te initialiseren.
- clear() om alle LED's uit te zetten.
- drawPixel() om specifieke pixels aan of uit te zetten.
- writeDisplay() om de wijzigingen zichtbaar te maken.
- drawBitmap() om afbeeldingen weer te geven.
- print() om tekst op de matrix te tonen.

In Listing 2.3 wordt een voorbeeld gegeven van het gebruik van de Adafruit_Microbit_Matrix component.

```

1 #include <Adafruit_Microbit.h>
2 Adafruit_Microbit_Matrix ledDisplay; //definieer de led display
3
4 uint8_t led_matrix[] =
5 { B00001, // bovenste rij met waarde 1.
6   B00010, //tweede rij met waarde 2.
7   B00100, //derde rij met waarde 4.
8   8,       //vierde rij met waarde B01000
9   0x10,    //onderste rij met waarde B10000 of terwijl 16
10 };
11 void setup() {
12   ledDisplay.begin();
13   ledDisplay.drawPixel(2, 3, 1); // Zet de LED op positie (2, 3) aan.
14   delay(2000);
15   ledDisplay.clear();
16   delay(2000);
17   ledDisplay.show(led_matrix);
18 }
19 void loop() {
20 }
```

Listing 2.3: Een LED matrix demo

Opdracht: Download [Listing 2.3 van GitHub](#) of kopieer listing 2.3 en Pas deze zodanig aan, zodat alleen de LEDs D2, D6 en D12 knipperen.

2.3 Het gebruik van de seriële poort.

Je kunt de seriële poort gebruiken om informatie te verzenden of te ontvangen van je laptop b.v. 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. In het voorbeeldprogramma van Listing 2.4 wordt de waarde van de teller steeds over de seriële lijn verstuurd.

```

1
2 void setup() {
3   Serial.begin(9600);
4   Serial.println("start");
5 }
```

```

6 int teller=0;
7
8 void loop() {
9     Serial.println(teller);
10    delay(1000);
11    teller++;
12 }

```

Listing 2.4: Een LED matrix demo

Open de seriële monitor(Tools→Serial Monitor) of (druk **Ctrl + Shift + M**) of klik op . In de 'Serial Monitor' verschijnt de output van je programma, zoals te zien is in figuur 2.3

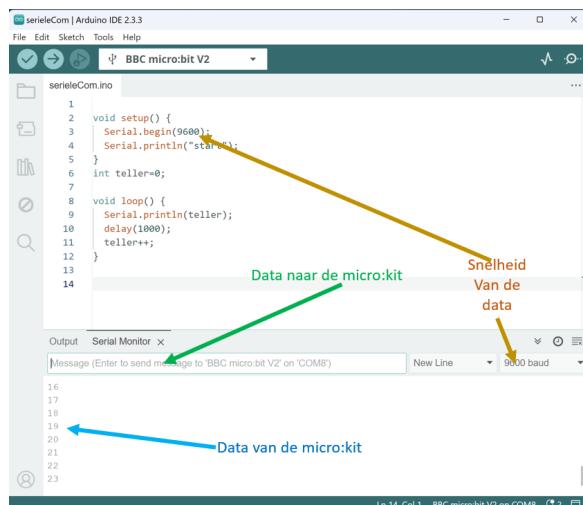


Figure 2.3: Instellingen van de seriële monitor.

De instellingen van de monitor kan gedaan worden in de statusbalk, zoals te zien is in figuur 2.3. Vaak wordt 9600 Baud (bits per seconde) gebruikt. Dit is gedaan omdat hiermee de data praktisch altijd stabiel verzonden wordt.

Zorg dat de ingestelde snelheid overeen komt met de instelling `Serial.begin(9600)` in `setup()`.

Opdracht Seriële communicatie: Lees via de seriële poort een getal in en laat een LED het aantal keren knipperen zoals het ingelezen getal. Lees hierna opnieuw een getal in, etc...

Tip: Lees het getal eerst in als een **string** en raadpleeg het voorbeeld dat er bij zit. Zet de ingelezen string vervolgens om naar een integer. Informatie hierover kan gevonden worden bij de arduino referece datatype **string**.

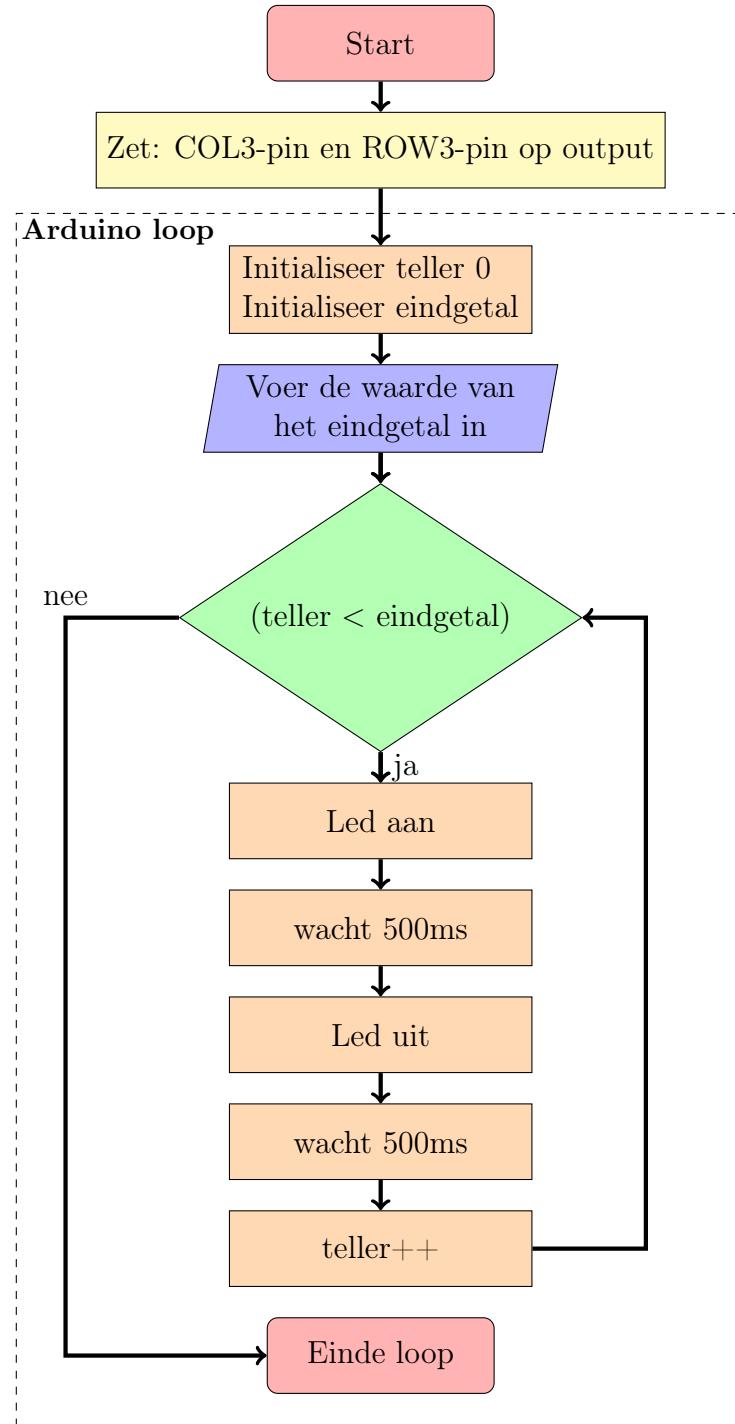


Figure 2.4: De LED knippert x keer.

De flowchart van dit programma wordt weergegeven in figuur 2.4

Opdracht Het binaire getal: Bij deze opdracht wordt een random getal tussen de 0 en de 31 gekozen, die vervolgens op de middelste rij van de led-matrix geplaatst wordt. De gebruiker moet binnen 10 seconden raden wat het getal is. De flowchart van het programma wordt in figuur 2.5 weergegeven.

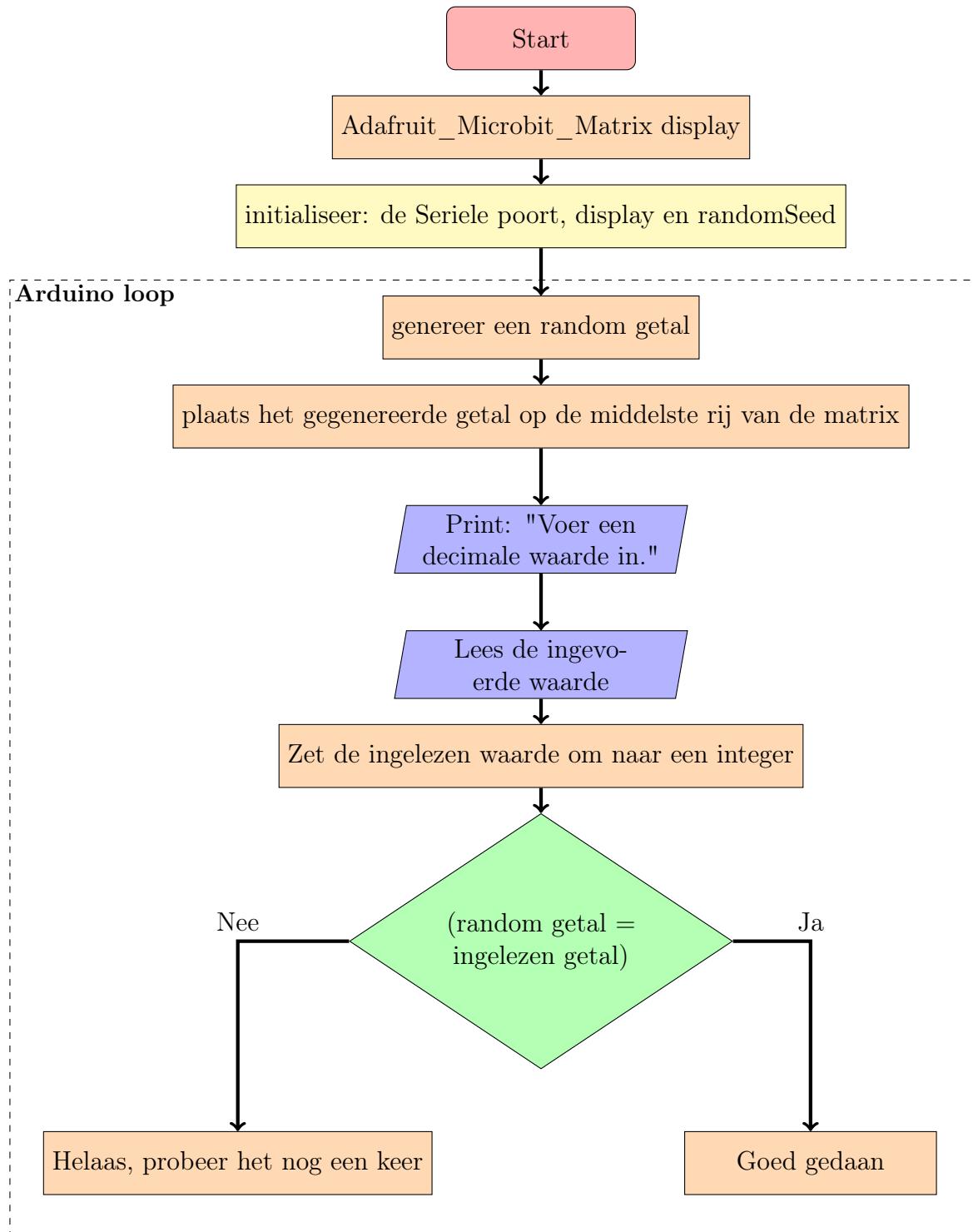


Figure 2.5: Het raden van de binaire waarde.

2.4 Bitwise operaties.

Zoals in hoofdstuk 2.2 staat vermeldt heeft de firma Adafruit voor het matrix display een speciale component gemaakt waardoor het matrix display eenvoudig te programmeren is. Listing 2.5 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
8 Adafruit_Microbit_Matrix matrixMbit; //maak een LED matrix aan.
9 const uint8_t
10 smile_bmp[] =
11 { B00000,
12     B01010,
13     B00000,
14     B10001,
15     B01110, };
16 uint8_t matrixje[] =
17 { B00000,
18     B00000,
19     B00000,
20     B00000,
21     B00000, };
22
23 void setup() {
24     Serial.begin(9600);
25     Serial.println("Welkom bij embedded!");
26
27     matrixMbit.begin();
28     matrixMbit.show(smile_bmp);
29     delay(2000);
30     matrixMbit.show(matrixje);
31     delay(2000);
32     //zet rechterLED bovenste rij aan
33     matrixje[0] = 1 << LED0; //schuif 1, LED0 plaatsen op naar links.
34     matrixMbit.show(matrixje);
35     delay(2000);
36     //zet linkerLED bovenste rij aan
37     matrixje[0] = 1 << LED4; //schuif 1, LED4 plaatsen op naar links.
38     matrixMbit.show(matrixje);
39 }
40 void loop() {
41 }
```

Listing 2.5: Het aanzetten van een LED

Wat opvalt aan Listing 2.5 is dat er een hoop define's staan aan het begin. De bedoeling van deze define's is dat de code eenvoudiger te lezen is. Indien in de code `1 << LED4` staat, weet de lezer gelijk dat de 4^e LED bedoeld wordt. Verder 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;`

Hierbij staat de 1 voor het aantal plaatsen dat LED0 naar links wordt opgeschoven. LED0 staat gedefinieerd op regel 2 van Listing 2.5

1. Installeer [de Adafruit Libraries](#) indien dit nog niet gedaan is.
2. (a) Open voorbeeldcode matrixSmpl (dit is Listing 2.5).
- (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). Doe dit zoals in de theorie besproken is.

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, de linker boven LED (LED4) weer uitgaat. Maak hierbij gebruik van de operatoren '«', '&' en '˜'. Doe 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 ^ . Doe 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(hulpje ... .....){
```

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 2.6: 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).

- Download de [voorbeeldcode van GitHub](#) of van brightspace of kopieer listing 2.5 in een nieuwe Arduino schets 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.
- Maak een functie `bool isLedAan(uint8_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.
- Pas listing 2.5 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 brightspace.

4. Maak een looplicht dat gaat over alle 5 de rijen van de matrix zoals figuur 2.6 laat zien.

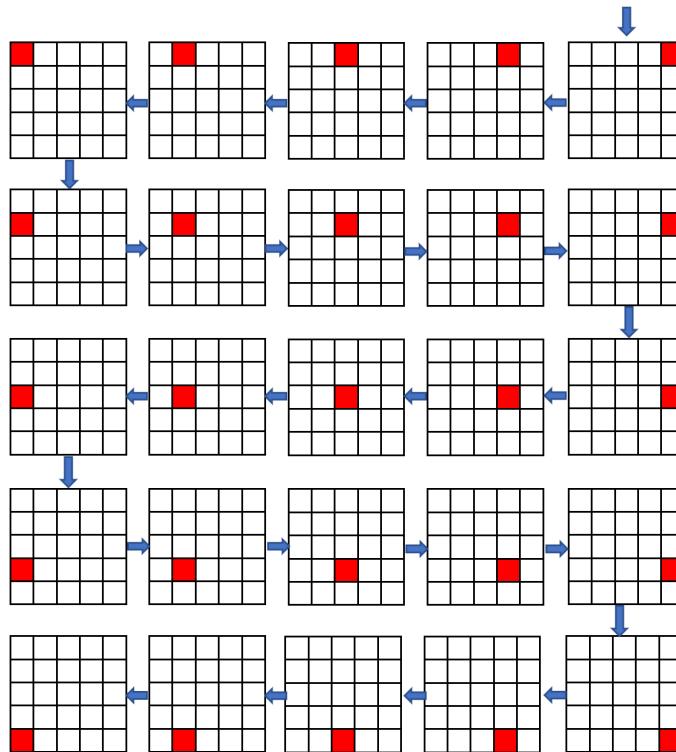


Figure 2.6: De volgorde van het looplicht.

Begin rechtsboven (bit 0 van de 0^e 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^e 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 [looplicht](#) te zien is.

Upload het resultaat op brightspace.

Challenge opdracht Op de micro:bit zitten verschillende sensoren. Eén van de sensoren is een [accelerometer sensor](#). Het programma in listing 2.7 toont de x,y en z waarde van de accelerator sensor.

1. Download listing 2.7 van [GitHub](#) of van brightspace of kopieer listing 2.7 in een nieuwe arduino omgeving en voer deze uit.
2. Pas het programma zodanig aan, zodat de LED in de richting beweegt, waarin micro:kit gehouden wordt.

```

1 #include <LSM303AGR_ACC_Sensor.h>
2
3 #define DEV_I2C Wire1 // Wire1 is voor de interne I2C bus
4 #define LED ROW1
5
6
7 // Nodig voor de accelerometer

```

```

8 LSM303AGR_ACC_Sensor Acc(&DEV_I2C);
9
10
11 const int COL1 = 4;
12 const int ROW1 = 21;
13
14 void setup() {
15     // Led.
16
17     pinMode(COL1, OUTPUT);
18     digitalWrite(COL1, LOW);
19     pinMode(ROW1, OUTPUT);
20
21     // Initialisatie van de serieele.
22     Serial.begin(9600);
23
24     // Initialisatie I2C bus (wordt veel gebruikt om sensors).
25     DEV_I2C.begin();
26
27     // Initialisatie van de accelarator.
28     Acc.begin();
29     Acc.Enable();
30
31     uint8_t a;
32     Acc.IO_Read(&a, 0x0F, 1);
33     Serial.print("Ik ben: ");
34     Serial.println(a);
35 }
36
37 void loop() {
38     // Led blinking.
39     digitalWrite(LED, HIGH);
40     delay(250);
41     digitalWrite(LED, LOW);
42     delay(250);
43
44     // Lees de accelerometer van de LSM303AGR uit.
45     int32_t accelerometer[3];
46     Acc.GetAxes(accelerometer);
47
48     // Output data.
49     Serial.print("| Acc[x/y/z] ");
50     Serial.print(accelerometer[0]);
51     Serial.print(" ");
52     Serial.print(accelerometer[1]);
53     Serial.print(" ");
54     Serial.print(accelerometer[2]);
55     Serial.println(" |");
56 }
```

Listing 2.7: Looplicht van de bovenste rij.

3 Opdracht 2

(Het inlezen van fysieke eigenschappen(week 12 en 13))

Bij deze opdracht lezen we de stand van de schakelaars die we vervolgens gebruiken om een teller op en af te laten tellen.

3.1 Het inlezen van de stand van de schakelaars.

- a) Maak een nieuwe sketch aan File→Examples → Adafruit Microbit Library → button-demo. zoals in Listing 3.1 te zien is.

```
void setup() {
    Serial.begin(9600);

    Serial.println("microbit is ready!");

    pinMode(PIN_BUTTON_A, INPUT);
    pinMode(PIN_BUTTON_B, INPUT);
}

void loop() {
    if (! digitalRead(PIN_BUTTON_A)) {
        Serial.println("Button A pressed");
    }
    if (! digitalRead(PIN_BUTTON_B)) {
        Serial.println("Button B pressed");
    }
    delay(10);
}
```

Listing 3.1: Het inlezen van de buttons.

- b) Run het programma en kijk op de monitor wat je ziet.
Indien de knop wordt ingedrukt, wordt er dan een '0' of een '1' gelezen?
- c) Pas het programma van Listing 3.1 zodanig aan, zodat de waarde van de schakelaars worden uitgeprint. Een voorbeeld wordt gegeven in figuur 3.1. Hiermee kan direct gecontroleerd worden of het antwoord op vraag b) goed is.

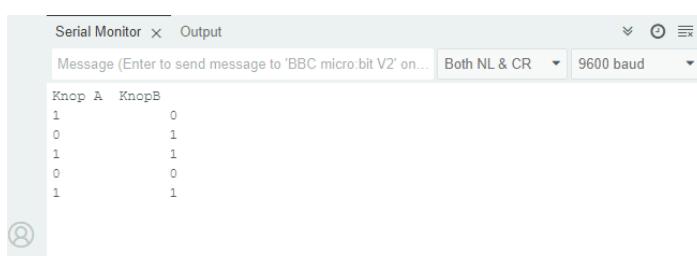


Figure 3.1: Weergave de ingelezen waarde van de knoppen.

- d) De arduino omgeving heeft behalve een `serial Monitor` ook een `Serial plotter`. Hierin worden de waarden van de variabele geplot. De arduino omgeving geeft een korte toelichting op het gebruik van de `serial plotter`. In listing 3.2 zijn twee variabele één variabele heeft altijd de waarde 500, de andere heeft een random waarde.
-

```

int random_variable;
int static_variable = 500;

void setup() {
    Serial.begin(9600);
}

void loop() {
    random_variable = random(0, 1000);

    Serial.print("Variable_1:");
    Serial.print(random_variable);
    Serial.print(",");
    Serial.print("Variable_2:");
    Serial.println(static_variable);
    delay(100);
}

```

Listing 3.2: Het uitprinten van een vaste- en een radomwaarde.

Indien deze twee variabelen laat uitplotten, wordt iets dergelijks als figuur 3.2 weergegeven.



Figure 3.2: Het plotten van een vaste en random variabele.

De `serial plotter` bepaalt zelf de kleuren van de variabele en de waarde van de Y-as. Doordat de waarde van de Y-as bepaald wordt door arduino, kan deze verspringen. Het verspringen van de Y-as kan voorkomen worden door een extra variabele te nemen met een bovengrens, in dit geval is dat 1200.

Voeg een bovengrens variabele toe aan listing 3.2 zodat de Y-as gelijk begonnen wordt met een maximum van 1200.

- e) Pas de opdracht van 3.1 c) zodanig aan zodat op de `Serial plotter` te zien is wanneer een knop is ingedrukt. Zoals weergegeven in figuur 3.3.

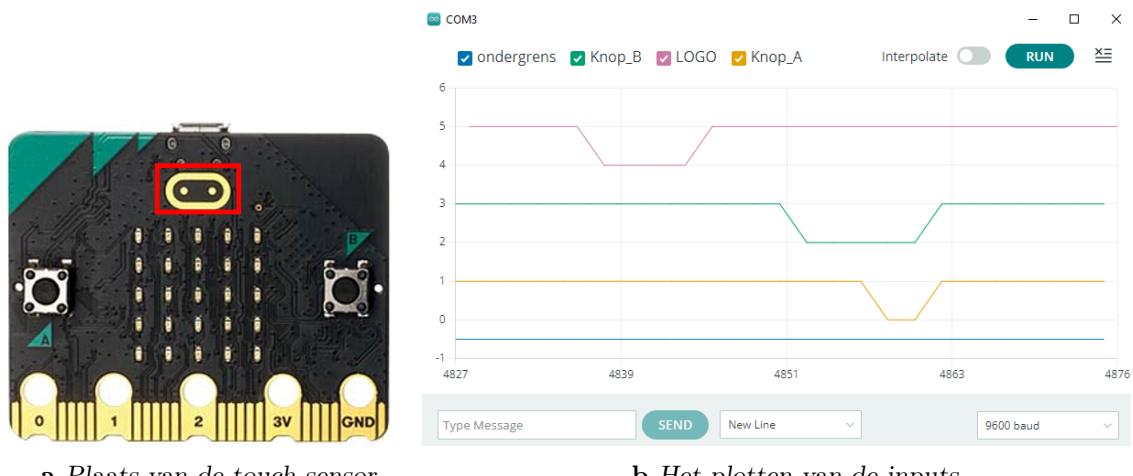
TIP: Bedenk dat $1 + 2 = 3$.



Figure 3.3: Het plotten van de stand van de schakelaars met een ondergrens.

- f) Op de microbit zit ook een toch sensor zoals in figuur 3.4a wordt weergeven en is aangesloten op pin 26. De touch sensor is hierbij zo geconfigureerd dat het eigenlijk een schakelaar is die en 0 of een 1 afgeeft.

Echter het lijkt erop alsof deze niet reageert op het aanraken. Wat helpt is indien je de sensor aanraakt met één van je vingers, met de duim de GND aanraakt. Een andere methode is om je vinger nat te maken, dan wordt oon een 0 afgegeven, zoals te zien is in figuur 3.8b waar de toch schakelaar de bovenste lijn voorstelt.



a Plaats van de touch sensor

b Het plotten van de inputs

Figure 3.4: De touch sensor

Breed de opdracht van e) uit zodat ook de waarde die de toch schakelaar heeft, wordt weergegeven bij het plotten, zoals te zien is in figuur 3.8b

3.2 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 dendert 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 3.5, waar te zien is dat bij

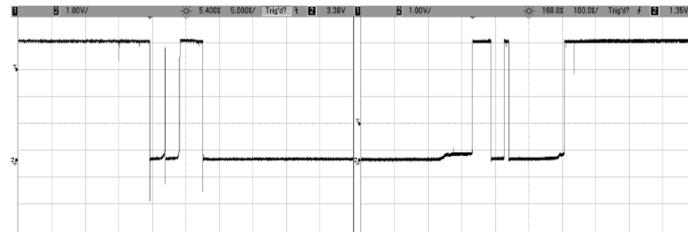


Figure 3.5: Een denderende schakelaar[1].

het indrukken en loslaten van het knopje de schakelaar 'stuiter'. Één 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 3.3

```

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 functie
12
13 void loop() {
14
15     if( isKnopIngedrukt(knopA) )
16         Serial.println("A is ingedrukt");
17
18     if( isKnopIngedrukt(knopB) )
19         Serial.println("B is ingedrukt");
20
21     delay(10);
22 }
23
24 boolean isKnopIngedrukt( int knop) {
25     if ( ! digitalRead(knop)) { //Is de knop soms ingedrukt ?
26         delay(2);           //wacht 2 milliseconde
27         if ( ! digitalRead(knop)) { //Is knop nog steeds ingedrukt
28             return true;
29         }
30     }
31     return false;
32 }
```

Listing 3.3: 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 milliseconde gewacht, waarna opnieuw gekeken wordt of de knop is ingedrukt. Is dit zo dan wordt een `true` mee teruggegeven anders een `false`.

In listing 3.4 is te zien dat er een detectie plaatsvindt wanneer de toestand van de schakelaar

verandert.

```
1      const int COL1 = 4;
2      const int ROW1 = 21;
3      const int knopA = 5;          // Knop A is aangesloten op poortnummer 5
4      const int knopB = 11;         // Knop B is aangesloten op poortnummer 11
5
6      boolean isKnopIngedrukt(int);
7
8      int knopTeller = 0;
9      boolean knopToestand = false;
10     boolean vorigeKnopToestand = false;
11     int ledStatus = LOW;
12
13    void setup() {
14        Serial.begin(9600);
15        Serial.println("microbit is ready!");
16        pinMode(COL1, OUTPUT);
17        pinMode(ROW1, OUTPUT);
18        pinMode(knopA, INPUT);
19        pinMode(knopB, INPUT);
20        digitalWrite(COL1, LOW);
21    }
22
23
24    void loop(){
25        knopToestand = isKnopIngedrukt(knopA);
26        if(knopToestand != vorigeKnopToestand) //heeft er een
27            ↪ verandering plaatsgevonden?
28        if (knopToestand == true) {
29            knopTeller++;
30            ledStatus = !ledStatus;
31            Serial.println(knopTeller);
32        }
33        delay(50);
34        vorigeKnopToestand = knopToestand;
35        digitalWrite(ROW1, ledStatus);
36    }
37
38    boolean isKnopIngedrukt( int knop) {
39        if (! digitalRead(knop)) { //Is de knop soms ingedrukt ?
40            delay(2);           //wacht 2 milliseconde
41            if (! digitalRead(knop)) { //Is knop nog steeds ingedrukt
42                return true;
43            }
44        }
45        return false;
46    }

```

Listing 3.4: Een toestandverandering van de schakelaar.

Download [voorbeeldcode 3.4 van GitHub](#) of kopieer listing 3.4 en upload het naar de Microbit. 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 3.6 wordt het signaal van de button weergegeven. In welke toestand (1 t/m



Figure 3.6: Weergaven van het signaal, indien de button wordt ingedrukt en weer wordt losgelaten.

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 3.4 op `HIGH` gezet wordt?
- (D) Breid listing 3.4 uit, zodat de teller met 1 verlaagd wordt indien met knopB een toestand verandering van 3 naar 4 plaatsvindt.

Upload deze opgave op brightspace.

3.3 De (externe) pinnen van de microbit

De Microbit heeft 21 pinnen (aansluitingen) die voor allerlei toepassingen bruikbaar zijn. In figuur 3.7 zijn de externe aansluitingen van de microbit te zien. Hierbij wordt aangegeven

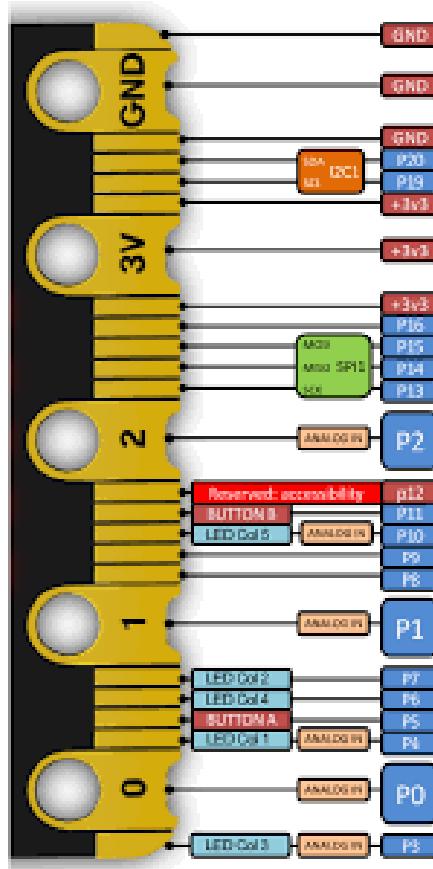


Figure 3.7: Pinlayout van de microbit.

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/.

3.3.1 Het lezen van externe analoge 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 3.5 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 ingelezene analoge waarde
    Serial.println(sensorValue);
    delay(100);           // delay
}
```

Listing 3.5: inlezen via een analog signaal.

- (a) Maak een nieuw Arduino sketch aan en zet listing 3.5 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 3.7.
- (c) Leg je Microbit op tafel, raak hem niet aan.
- (d) Gebruik de Seriële Plotter om de waarde van de variabele *sensorValue* te laten zien.
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? _____
(Bij mij kwam deze uit tussen de 1000 zonder aanraken en de 750)

2. De LEDs op de microbit zijn via een truck ook iets wat lichtgevoelig. Hierdoor kan een bepaalde lichtintensiteit gemeten worden.

Om deze te kunnen meten zal de LED als een analoge waarde ingelezen moeten worden. In figuur 3.8a

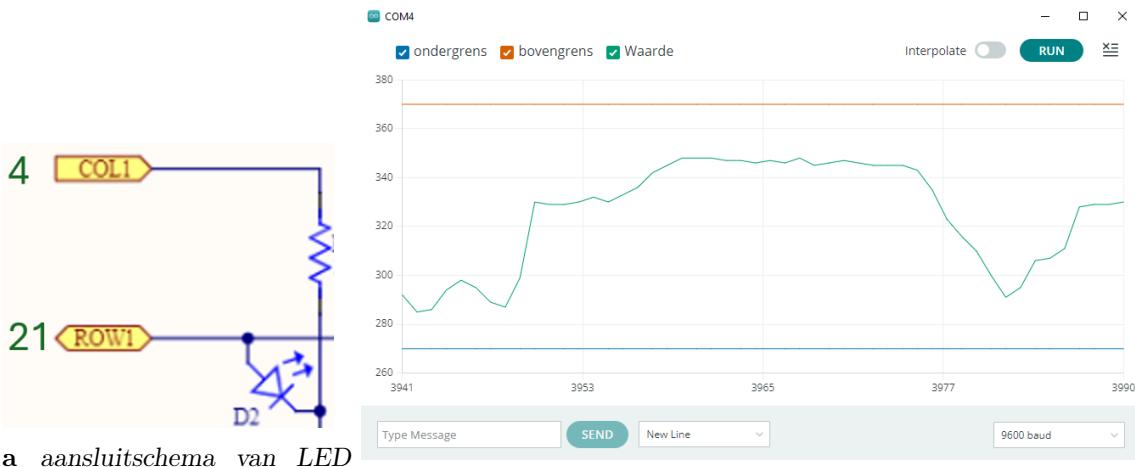


Figure 3.8: Een matrix LED als fotodiode?

is te zien hoe LED D2 (linkerboven LED van de matrix) is aangesloten. Door nu PIN 4 als een analoge input te definiëren en PIN 21 hoog te maken is er een klein spanningsverschil op PIN 4 te meten.

De opdracht: Maak een programma waarbij aan de hand van de middelste LED een indicatie verkregen kan worden over de omgevingslicht.

3.3.2 De challenge opdracht

De micro:bit heeft ook een kompassensor. Hoe deze kan worden uitgelezen wordt in grove lijnen weergegeven in listing 3.6, hier wordt de `void setup()` en de `void loop()` weergegeven. De complete code kan gedownload worden van de git-hub.

```

1 #include <LSM303AGR_MAG_Sensor.h>
2
3 LSM303AGR_MAG_Sensor Mag(&DEV_I2C);
4
5 void setup(void) {
6     // Start LED matrix driver
7     pinMode(ROW3, OUTPUT);
8     pinMode(COL3, OUTPUT);
9     digitalWrite(COL3, LOW);
10
11    // Initialiseer I2C bus.
12    DEV_I2C.begin();
13
14    Mag.begin();
15    Mag.Enable();
16
17    Serial.begin(9600);
18    pinMode(PIN_BUTTON_A, INPUT);
19    // Serial.println("Druk op knopA om te stoppen met kalibreren");
20
21    Serial.println("");
22    lastDisplayTime = millis();
23    kalibreer();
24 }
```

```

25
26
27 void loop(void) {
28     float hoek;
29     hoek=bepaalHoek();
30     if ((millis() - lastDisplayTime) > 1000) {
31         Serial.print("ondergrens:");
32         Serial.print(-0.5);
33         Serial.print(",");
34         Serial.print("Bovengrens:");
35         Serial.print(361);
36         Serial.print(",");
37         Serial.print("Hoek:");
38         Serial.println(hoek);
39         lastDisplayTime = millis();
40     }
41     delay(10);
42 }
```

Listing 3.6: De micro:bit als een kompas

Zoals te zien is, wordt de magneet-sensor aangemaakt in regel 3 van listing 3.6 en geïnitialiseerd op regel 14 en 15.

Verder moet de sensor gekalibreerd worden, dit wordt gedaan in de functie op regel 23. De code van de functie is terug te vinden op de github.

Het kalibreren van de sensor wordt gedaan door de sensor een paar keer rond te draaien en vervolgens op knop A te drukken.

In de `void loop()` functie wordt de waarde van de sensor uitgeprint op de plotter, zoals in figuur 3.9 te zien is. Daar wordt bij de waarde 0 ongeveer het noorden



Figure 3.9: De micro:bit wordt een keer rondgedraaid.

weergegeven. Je zult zien dat het niet altijd even nauwkeurig is.

- Download de code en run deze. Laat de waarde door de plotter uit plotten en controleer of het enigszins klopt bijvoorbeeld met een kompas app op je telefoon.
- Breid het programma uit, met de mogelijkheid om opnieuw te kalibreren indien knop B wordt ingedrukt.
- Breid het programma verder uit zodat op het LED-display altijd een lijn te zien is die loopt van noord naar zuid.

4 opdracht 3

Bij deze opdracht worden LEDs van de bovenste 4 Rijen van de LED matrix van de microkit gedimd.

- A. In het onderstaande programma wordt het middelste bit gedimd waarbij de *dutycycleDuty cycle* = $\frac{10}{255} \times 100\% = 3.9\%$ is. De complete code is te downloaden als [voorbeeldcode 3.4 van GitHub](#)

```
1 void setup() {
2
3     initMatrix();
4     delay(1000);
5     analogWrite(ROW3, 10);
6
7 }
8
9 void loop() {
10 }
```

Listing 4.1: De micro:bit als een kompas

Bibliography

[B] E. Williams. *Make: AVR Programming*. Maker Media, 2014.

Appendices

A 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 A.1. Er zijn verschillende

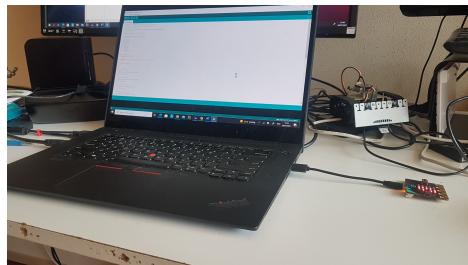


Figure A.1: aansluiting van de BBC microbit.

ontwikkelomgeving voor de BBC microbit, maar tijdens dit instructie college(practicum), zal voornamelijk via de [Arduino omgeving](#) omgeving gewerkt worden.

A.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 spekertje, 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 aan de hand van deze handleiding of opzoeken op het internet, 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.

A.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 A.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 A.2.

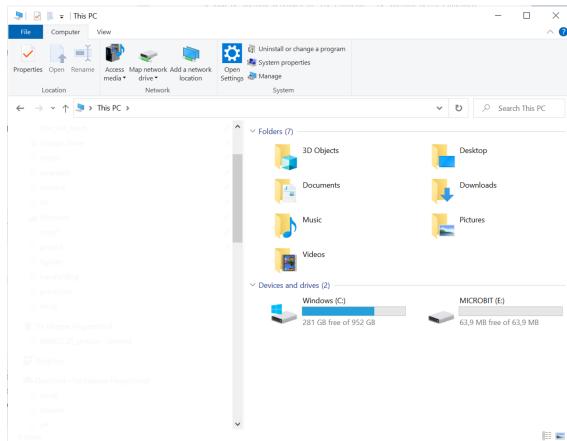


Figure A.2: Herkenning microbit in windows.

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 startpunt.

A.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.

Je zou tijdens de Challenge eventueel ook de microbit met [micropython](#) kunnen programmeren. Verder is er nog een mogelijkheid om de “blokken” taal te gebruiken. Hier kun je ook alles mee. Voor een intro in de “blokken” taal, start hier en begin met ‘Mode’, ‘Step Counter’, kies voor ‘Instructies weergeven’: <https://makecode.microbit.org/>

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

A.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.

A.2.1 Het installeren van de Arduino omgeving

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

1. Download de Arduino IDE 2.3.3 software van <https://www.arduino.cc/en/software>, zoals je kan zien zijn er ook versies voor Linux en de Mac.
2. Bevestig alle vragen, klik maar door.
3. Bij het opstarten van Arduino IDE 2.3.3 zal gevraagd worden om de libary's te updaten. Update de libary's.

Na het opstarten verschijnt het scherm zoals in figuur A.3 wordt weergegeven.

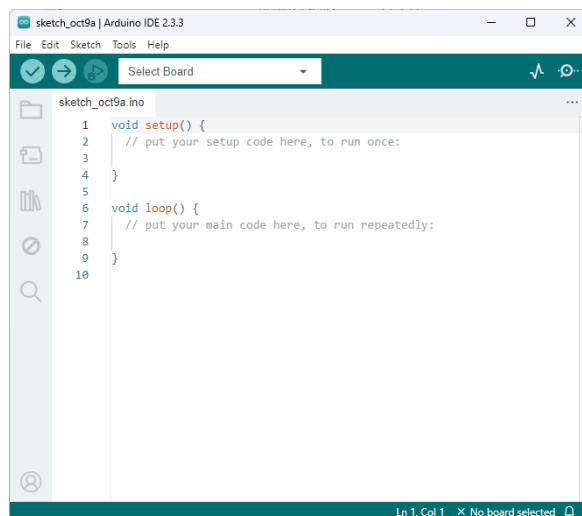


Figure A.3: De Arduino omgeving met een leeg scherm.

Waarin de functie `void setup() {}` één keer wordt uitgevoerd en de functie `void loop() {}` continu wordt uitgevoerd.

Als eerste kunnen een aantal instellingen aangepast worden. Ga naar File → Preferences, en pas de instellingen aan zoals te zien is in figuur A.4.

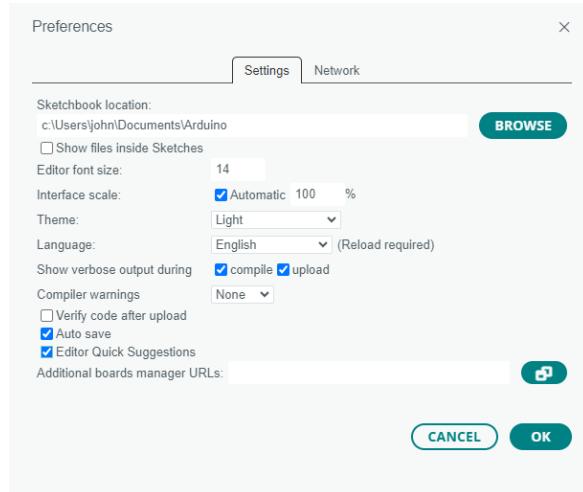


Figure A.4: preferences scherm.

- 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.

A.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.

Doordat op de microbit een NRF52 microcontroller zit, zal de betreffende package bij de Arduino IDE 2.3.3 omgeving bekent moeten worden gemaakt, waarna de betreffende board met deze microcontroller geselecteerd kunnen worden.

1. Ga naar File → Preferences, en zet bij Additional boards manager URLs: de URL https://sandeepmistry.github.io/arduino-nRF5/package_nRF5_boards_index.json zoals in figuur A.5 te zien is.

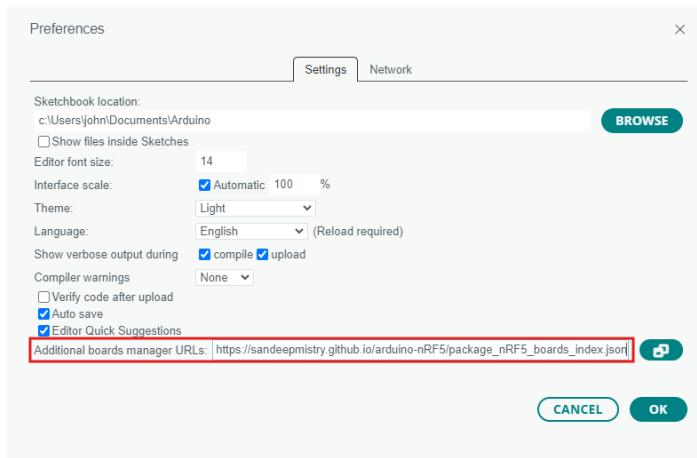
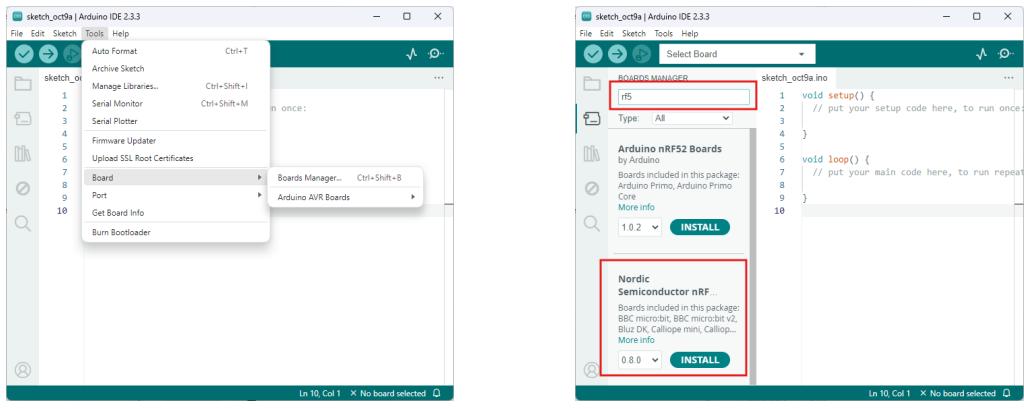


Figure A.5: preferences scherm.

- Het selecteren van diverse borden met een nRF5 microcontroller wordt als volgt gedaan:
Ga naar Tools → Boards → Boards manager.. zoals figuur A.6a aangeeft. Zoek op *nrf*



a selecteren van de board manager

b Installeren van de NRF5 borden

Figure A.6: Arduino laten werken met de microbit.

en installeer vervolgens de **Nordic Semiconductor nRF**, zoals in figuur A.6b wordt weergegeven.

- Het selecteren van de microbit als het bordje waarop geprogrammeerd wordt, wordt als volgt gedaan: Ga naar Tools → Boards → Nordic Semiconductor NRF5 Boards en selecteer BBC micro:bit V2 zoals figuur A.7 wordt weergegeven.

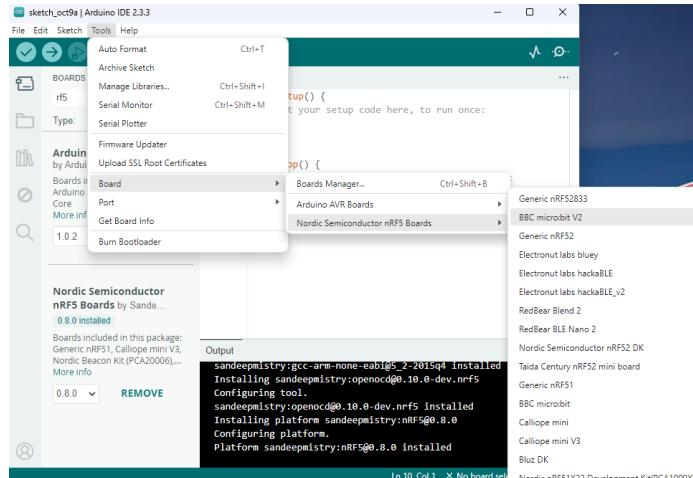


Figure A.7: Selecteren van de micro:bit als bord waarop gewerkt wordt.

- Controleer of de arduino aangesloten is op de USB port en selecteer de juiste COM port. Ga naar Tools → Port → (BBC micro:bit,...)

In figuur A.8 is te zien dat de microbit op COM7 zit, dit kan bij iedereen weer anders zijn.

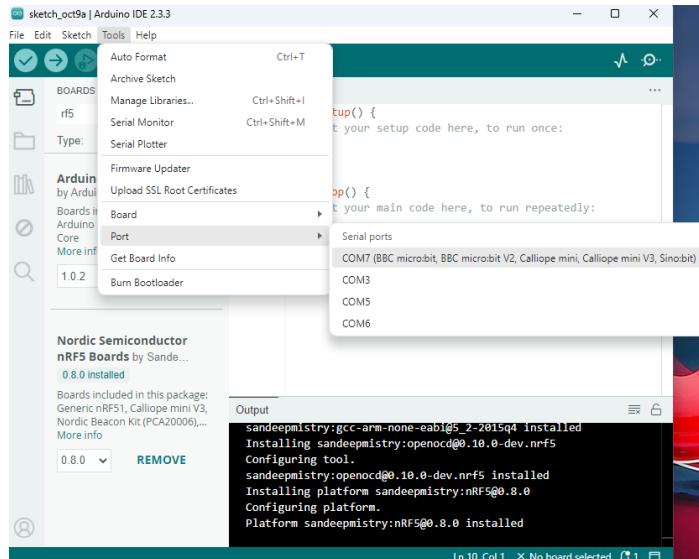


Figure A.8: Selecteren van de micro:bit als bord waarop gewerkt wordt.

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



Figure A.9: Melding van windows defender.

toestemming om de USB port te mogen gebruiken.

6. Haal de voorbeeldcode blink van brightspace of [download](#) de laatste versie en plaats deze in je eigen werkdirctory.
7. Open het voorbeeldprogramma blink.ino (dubbel klik), dat wordt weergegeven in Listing 2.1, compileer en upload naar de micro:bit (klik op de knop).
Als het goed is gaat het linkerboven LEDje van de matrix knipperen. Voor verder uitleg zie hoofdstuk 2.1

A.2.3 Het installeren van de Adafruit library.

Zodra je complexere zaken wilt aanpakken, heb je een hulplibrary nodig om dingen zoals de interne temperatuursensor, LED-matrix of Bluetooth-verbinding te beheren. Om het je makkelijker te maken, heeft Adafruit een wrapper-library geschreven die dit allemaal voor je regelt.

1. Ga naar Sketch → include library → Manage Libraries zoals figuur A.10 aangeeft

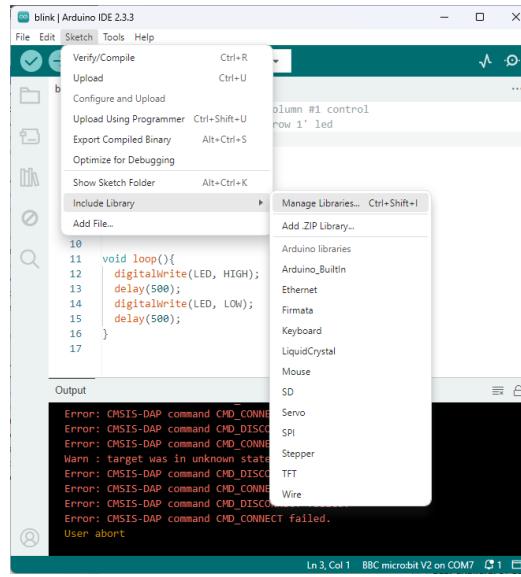


Figure A.10: Installeren van van Library.

- Zoek naar de **Blep** library en installeer deze zoals te zien is in figuur [A.11a](#) de **Adafruit GFX library** met de dependency zoals te zien is in figuur [A.11b](#) en de **Adafruit Microbit** zoals te zien is in figuur [A.11c](#).

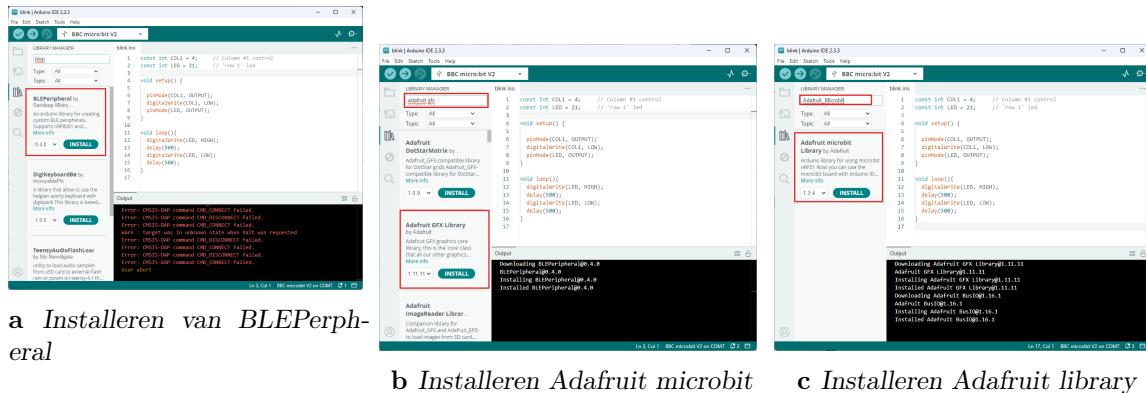


Figure A.11: Het installeren van Arduino libraries.

Ga terug naar [opdracht 1](#).

B Problemen tijdens het practicum.

In deze bijlage worden een aantal problemen omschreven die tijdens het practicum kunnen ontstaan.

B.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 B.1a
2. Controleer of library geïnstalleerd is, zolas aangegeven in figuur B.1b

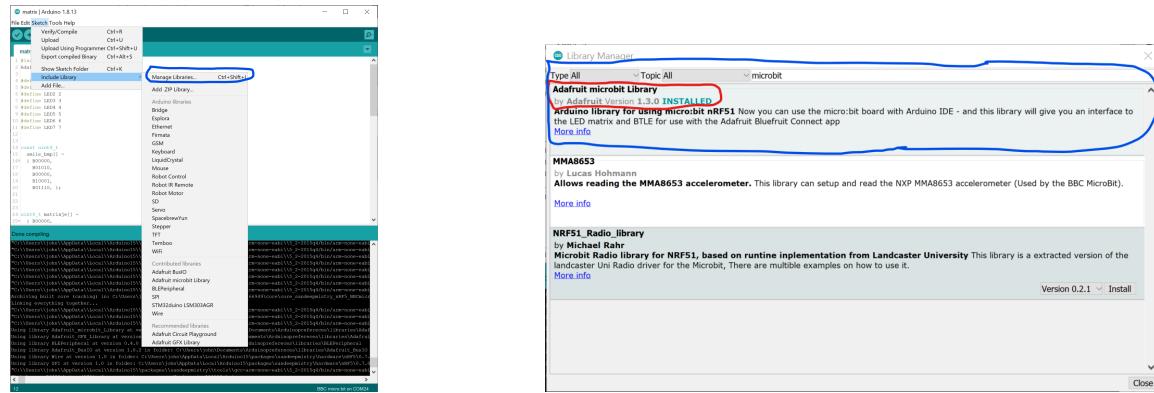


Figure B.1: Controleer of library is geïnstalleerd.

B.2 SerialPortException: Port name - COM..

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

De seriële poort is in gebruik door waarschijnlijk de monitor.

Sluit de monitor af.

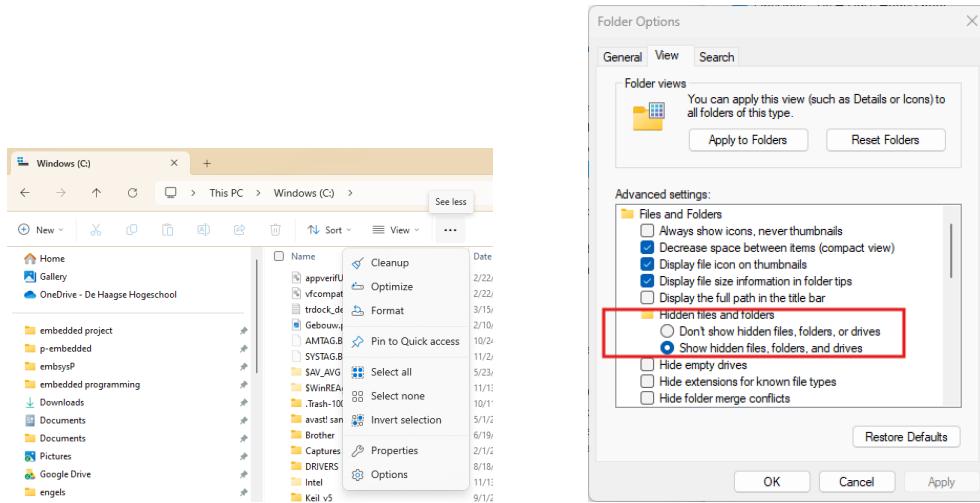
B.3 Tijdens het uploaden

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

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

2. Soms heeft het te maken met de firewall van windows.
 - (a) open Windows Defender Firewall.
 - (b) Klik op: *Allow an app or feature through Windows Defender Firewall*
 - (c) Klik op: Change settings.
 - (d) Zoek naar *Arduino IDE* en verwijder deze. klik vervolgens op *OK*.
 - (e) Sluit de hele Arduino omgeven en start deze vervolgens weer op. De arduino zal vragen om toestemming, geef deze.
3. Zelf uploaden van de code:

- (a) Zorg ervoor dat de hidden files zichtbaar worden. Ga met de File explorer naar de C: drive. klik op de . . . iets dergelijks als figuur B.2a wordt zichtbaar klik op option en neem de optie show Hidden Files zoals figuur B.2b laat zien.



a Selecteen van options bij de File Explorer

b Maak hiddenfiles zichtbaar.

Figure B.2: Hidden files zichtbaar maken

- (b) Compileer (Verify) je code. Arduino compileert de file en maakt o.a. een **.hex** file aan.
Deze **.hex** file moet gekopieerd worden naar de **MICROBIT** directory.
Om de **.hex** file te vinden kan het beste naar het Output-veld van de Arduino IDE 2.3.3 gekken worden. In één van de laatste regels in het Output-veld van de Arduino IDE 2.3.3 wordt de **.hex** file gecreëerd met het commando:
`arm-none-eabi-objcopy -O ihex sourcefile.elf destinationfile.hex`.
In de destinationfile is het volledige pad te zien.
In mijn geval staat er:
`"C:\\\\Users\\\\john\\\\AppData\\\\Local\\\\Temp\\\\arduino\\\\sketches\\\\3490945
\\\\EE0644827B5A5400078BA5D3\\\\blink.ino.hex"`
Het hele pad in mijn geval is dus:
`"C:\\\\Users\\\\john\\\\AppData\\\\Local\\\\Temp\\\\arduino\\\\sketches\\\\3490945
\\\\EE0644827B5A5400078BA5D3"`

B.4 Tijdens compileren 'kan stdint.h niet vinden'

Mogelijk is er iets fout gegaan bij het installeren van de **Add NRF5x Board Support**. Waardoor de library niet geinstalleerd is.

Bij het installeren van de 'Add NRF5x Board Support' moet er een lange tijd gewacht worden. Mogelijk is deze onderbroken. Bij het opnieuw installeren lijkt alles goed te gaan, dit is echter niet zo.

Mogelijke oplossing:

remove het board en installeer deze opnieuw (wacht nu wel totdat deze geïnstalleerd is).