

Embedded C

Week 1: Hello Arduino – Basics - LEDs

Content

- Info mbt het vak Embedded
- Overzicht Week 1
- Demo 1 – Blinking LED
- Nieuw project in VS-Code
- Demo 2 – Flashing LED
- LED Library
- Oefeningen
- Week project



Info mbt het vak Embedded

- ↪ ECTS-fiche
- ↪ Overzicht inhoud
- ↪ Hoe?
- ↪ Wat heb je nodig?
- ↪ Evaluatie

ECTS-fiche

- 4 studiepunten
- Inhoud: *"In dit opleidingsonderdeel gebruiken we de programmeertaal C om "onder de motorkap" te bekijken wat er in het geheugen gebeurt als we bijvoorbeeld een array, struct of String aanmaken. We gebruiken pointers om variabelen of arrays rechtstreeks via hun adres te benaderen of "by reference" door te geven. We maken kennis met een embedded system (Arduino) en een aantal specifieke elektronica aspecten daarvan, zoals processor, pins, vermogen en weerstand. Vervolgens gebruiken we de programmeertaal C om een Arduino board te programmeren."*
- Evaluatie:

| | | |
|--|-------|---|
| Beroepsproduct - met evaluatiemoment in de examenperiode | 50,00 | Individueel Arduino programmeerproject (en) |
| Mondeling examen | 50,00 | |

Overzicht

- Week 1: Hello Arduino - Basics - LED
- Week 2: Buttons - Interrupts
- Week 3: Potentiometer – the LED Screen
- Week 4: Timers - Sound
- Week 5: Integratie en individueel project
- Week 6: Integratie en individueel project

Hoe?

- Elke week: een aantal **demo's**
 - Code beschikbaar
 - Uitleg in commentaar - verwijzing naar tutorials

→ Die moet je volledig begrijpen!
- Elke week: een aantal **tutorials** op canvas

→ Worden besproken in de klas of neem je zelf door!
- Elke week:
 - een aantal **oefeningen** → Uitwerken

- *1 weekproject* → Uitwerken en toevoegen aan portfolio

- Laatste 2 weken: **individueel project** (keuze uit lijst)

→ Uitwerken en toevoegen aan portfolio

Portfolio



Voorbeeld



Presentatie



Zelfstudie



Oefening



Opdracht

Wat heb je nodig?

- Hardware

- ☐ Laptop

- ☐ Arduino Uno



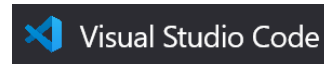
- ☐ Velleman VMA209 Expansion shield



- ☐ USB-kabel

- Software

- ☐ IDE: Visual Studio Code (VSCode)



- ☐ PlatformIO extension



Evaluatie?

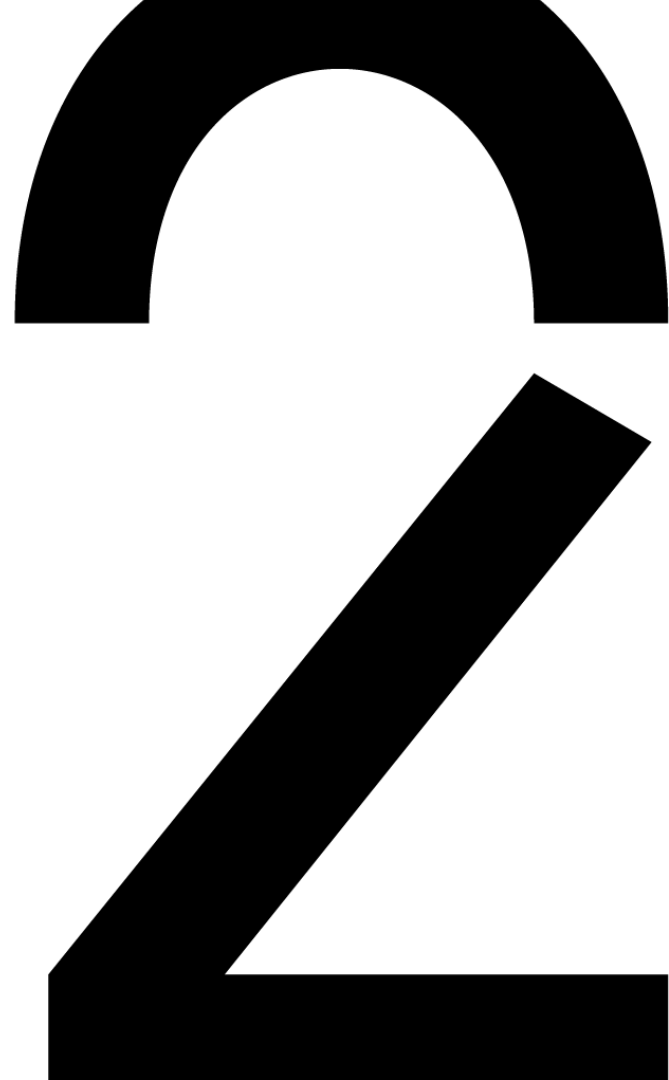


Evaluatie

- Mondeling examen:
 - Je demonstreert je individueel project
 - Je wordt bevraagd over de leerstof aan de hand van je portfolio
- Portfolio bevat:
 - Jouw oplossing van het *weekproject*
 - Jouw individueel project

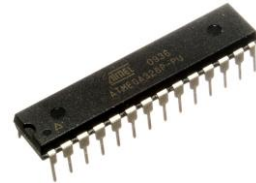
Week 1 - Embedded

- ↪ Overzicht
- ↪ Tutorials
- ↪ Demos
- ↪ Oefeningen
- ↪ Weekproject



Week 1 - Overzicht

- Wat is Arduino?
- Een microcontroller: de ATmega328P (AVR microcontroller familie)
- Basissyntax van de taal C.
- Hoe communiceren we met een LED?
- Gebruiken en maken van een C library.



Week 1 - Tutorials

1 - De Hardware

1.1 - Arduino UNO bord met de microcontroller ATmega328



Video

1.2 - Het Multifunctional Shield



Video

2 - De Programmeertaal C

2.1 - Wat is C?



Video

2.2 - De Syntax van C



Video

2.3 - Bit Operaties in C



Video

3 - Overige Tutorials

3.1 - Schrijven naar een pin

3.2 - De wet van Ohm en de LED

3.3 - IDE: Visual Studio Code met PlatformIO extension

3.4 - Stappenplan: een nieuw project in VS-Code

3.5 - Een eigen library gebruiken in VS Code

Week 1 - Demos

1 – Blinking LED

We laten één LED-lampje continu aan en uit knipperen

2 – Flashing LEDs

We laten alle 4 de LED-lampje branden door ze één na één aan te zetten en schakelen ze daarna één na één weer uit

Week 1 - Oefeningen

LED Dimmen

Een led kan enkel volledig aan of volledig af gezet worden. Om een led "half" aan te zetten (te "dimmen" eigenlijk) maken we gebruik van een trukje: we zetten de led in een lus heel snel aan en af zodat hij de helft van de tijd aan is, en de andere helft uit. Als we dat snel genoeg doen, zien onze ogen dat als een lampje dat 50% gedimd is. Om een ledje tot 20% te dimmen moeten we er dan voor zorgen dat het maar 20% van de tijd aan is, en 80% uit. Deze techniek wordt PWM genoemd (**Pulse Width Modulation**).

LED Chaos

Schrijf een programmaatje dat de verschillende leds in een random patroon laat flikkeren: willekeurig gekozen leds worden gedurende willekeurig lange periodes (tussen bijvoorbeeld 100 en 1000 milliseconden) aan en uit gezet.

Om dit te kunnen doen zal je moeten uitvissen hoe je een random getal kan genereren in de programmeertaal C. Je zal hiervoor de `stdlib.h` library moeten includen. Check de documentatie van de `rand()` functie.

Week 1 - Weekproject

Morse trainer

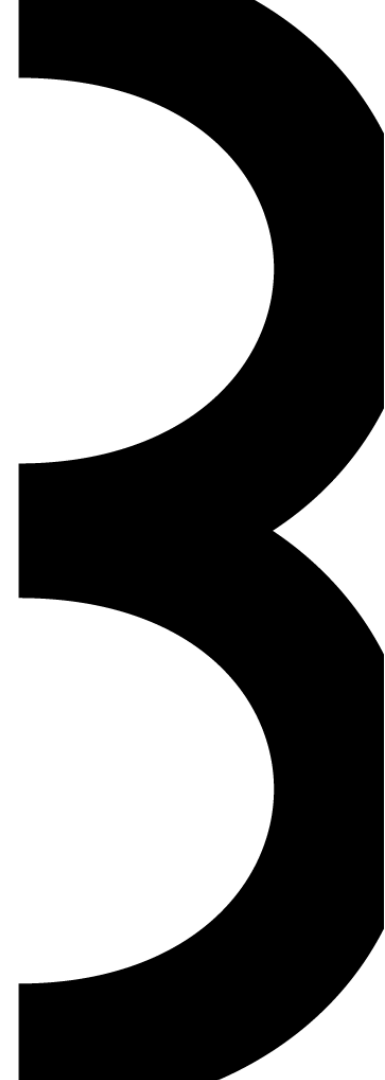
Je bouwt een eenvoudige morse-trainer die je kan gebruiken om te testen hoe goed je morse-code kent.

| | | | | | | | |
|---|--------|---|--------|---|--------|---|----------|
| A | .- | J | .-.-.- | S | ... | 1 | .-.-.-.- |
| B | -... | K | -.- | T | - | 2 | ..-.-.- |
| C | -.-.-. | L | .-... | U | ..- | 3 | ...-.- |
| D | -.- | M | -- | V | ...- | 4 |- |
| E | . | N | -. | W | .-.- | 5 | |
| F | ..-.- | O | --- | X | -.-.- | 6 | -..... |
| G | --. | P | .-.-.- | Y | -.--.- | 7 | ---.... |
| H | | Q | ---.- | Z | ---.. | 8 | ---.... |
| I | .. | R | .-. | 0 | ----- | 9 | -.-.-.-. |

- Je toont eerst een "aftelpatroon" van ledjes.
- Vervolgens wordt er een willekeurige letter in Morsecode op de leds getoond en na een korte pauze verschijnt de correcte letter op het computerscherm.
- Na 10 letters eindigt het programma met een frivool led-dansje en begint het terug vanaf start.



Demo 1 – Blinking LED

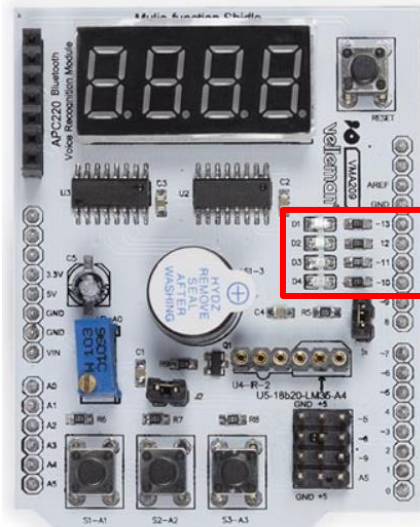


Demo 1 – Blinking LED - C-code

```
#include <util/delay.h>
#include <avr/io.h>

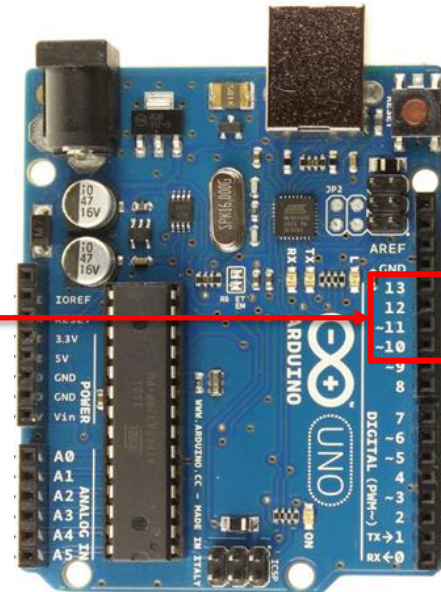
int main() {
    DDRB = 0b00100000;
    while (1) {
        PORTB= 0b00000000;
        _delay_ms(1000);
        PORTB = 0b00100000;
        _delay_ms(1000);
    }
    return 0;
}
```


Demo 1 – Blinking LED - Achtergrond

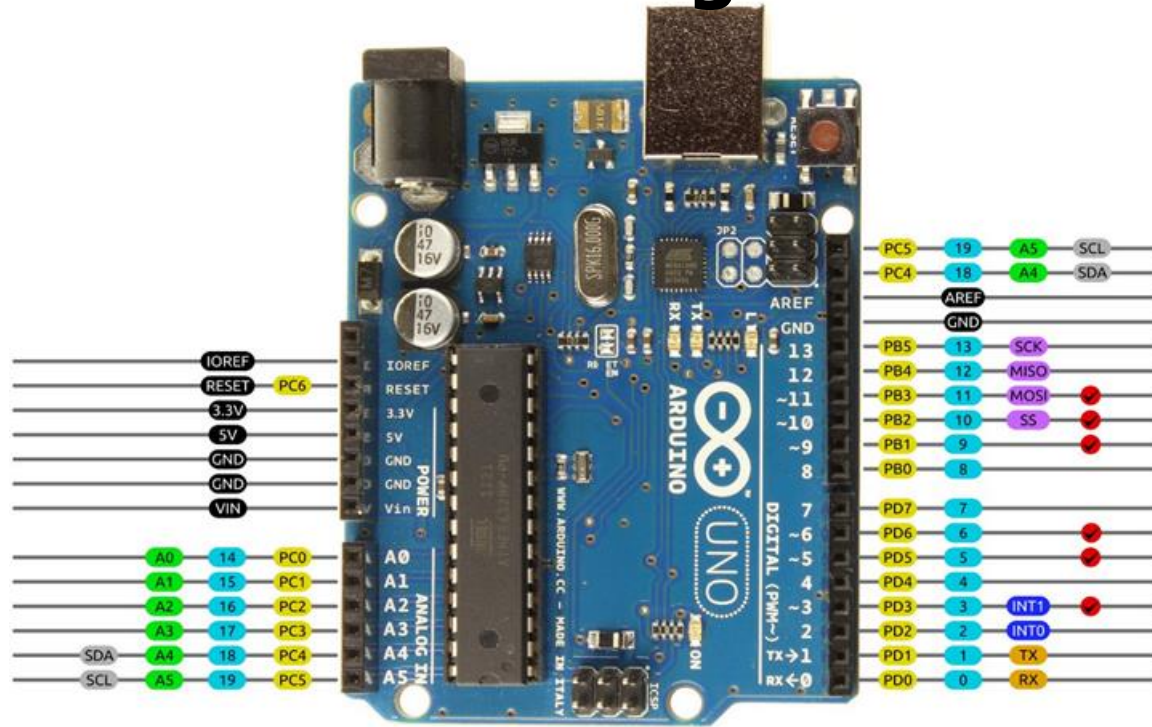


| |
|---|
| 4 red LEDs |
| 3 buttons + reset button |
| potentiometer (10 kΩ) |
| 4-digit, 7-segment LED tube driven by 74HC595 |
| buzzer |
| socket for IR receiver (remote control) |
| socket for temperature sensor LM35 or DS18B20 (polarity!) |
| header for APC220 shield |
| free pins (PWM) |

| |
|--------------------------|
| 10, 11, 12, 13 |
| A1, A2, A3 |
| A0 |
| latch 4, clock 7, data 8 |
| 3 (digital on-off) |
| 2 |
| A4 |
| GND, +5V, 0, 1 (RX/TX) |
| 5, 6, 9, A5 |

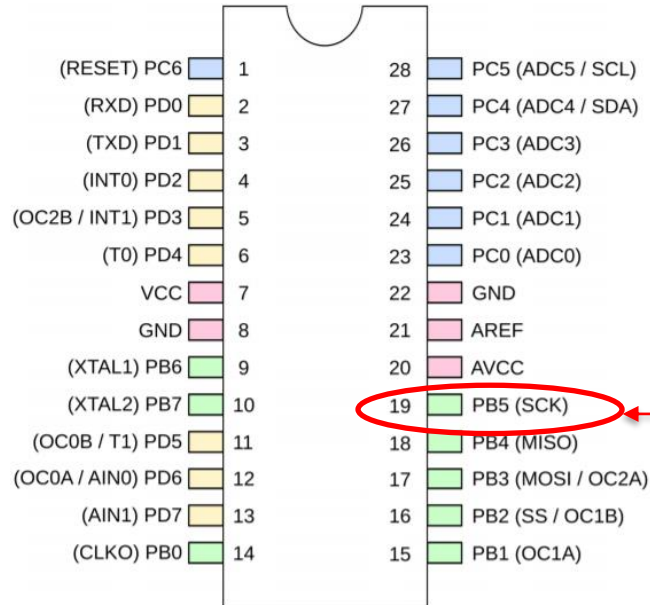


Demo 1 – Blinking LED - Achtergrond



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT

Demo 1 – Blinking LED - Achtergrond



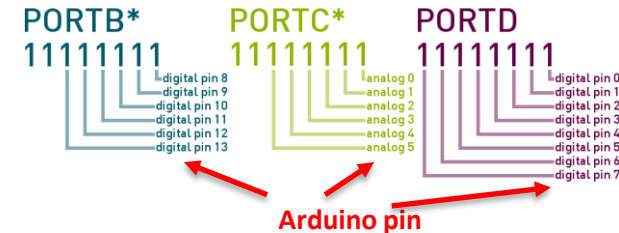
| Multi-Function shield V2 | Digital pin (Dn) | Analog pin (An) | AVR pin | AVR port | AVR function(s) | AVR PWM |
|--------------------------|------------------|-----------------|---------|----------|-----------------|---------|
| Bluetooth header - tx | 0 | | 2 | PD0 | RxD | |
| Bluetooth header - rx | 1 | | 3 | PD1 | TxD | |
| IR receiver | 2 | | 4 | PD2 | INT0 | |
| Buzzer | 3 | | 5 | PD3 | INT1, OC2B | Yes |
| 7-seg display Pin latch | 4 | | 6 | PD4 | T0, XCK | |
| header | 5 | | 11 | PD5 | T1 | Yes |
| header | 6 | | 12 | PD6 | AIN0 | Yes |
| 7-seg display clock | 7 | | 13 | PD7 | AIN1 | |
| 7-seg display data | 8 | | 14 | PB0 | CLK0, ICP1 | |
| | 9 | | 15 | PB1 | OC1A | Yes |
| Red LED | 10 | | 16 | PB2 | OC1B, SS | Yes |
| Red LED | 11 | | 17 | PB3 | OC2A, MOSI | Yes |
| Red LED | 12 | | 18 | PB4 | MISO | |
| Red LED | 13 | | 19 | PB5 | SCK | |
| Variable resistor | 14 | 0 | 23 | PC0 | | |
| Switch 1 | 15 | 1 | 24 | PC1 | | |
| Switch 2 | 16 | 2 | 25 | PC2 | | |
| Switch 3 | 17 | 3 | 26 | PC3 | | |
| DS18B20 header | 18 | 4 | 27 | PC4 | SDA | |
| header | 19 | 5 | 28 | PC5 | SCL | |

Demo 1 – Blinking LED - Achtergrond

De microcontroller ATmega328 communiceert via zijn pins met de buitenwereld. Dit gebeurt door een spanning op één van de pins aan of af te zetten. De pins van de microcontroller zijn opgedeeld in 3 poorten: PORTB, PORTC en PORTD.

Elk van deze poorten bevat een aantal pins. Zo heeft

- PORTB beschikking over 8 pins (PB0-PB7),
- PORTC heeft 7 pins (PC0-PC6) en
- PORTD heeft er weer 8 (PD0-PD7).



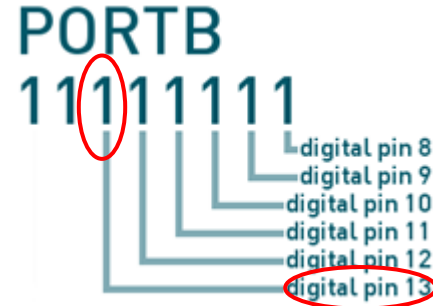
De ATmega328 heeft per PORT drie registers ter beschikking:

- Het **DDRx** register (**D**ata **D**irection **R**egister): *hierin kunnen we aangeven of we een bepaalde pin voor input of voor output willen gebruiken.*
- Het **PORTx** register (**P**in **O**utput **R**egister): *hierin kunnen we effectief dan spanning aan of af zetten op de juiste pins.*
- Het **PINx** register (**P**in **I**nput Register): *dit gebruiken we om input uit te lezen.*

Demo 1 – Blinking LED - Achtergrond

```
#include <util/delay.h>
#include <avr/io.h>

int main() {
    DDRB = 0b00100000;
    while (1) {
        PORTB = 0b00000000;
        _delay_ms(1000);
        PORTB = 0b00100000;
        _delay_ms(1000);
    }
    return 0;
}
```



Data Direction Register

Pin Output Register

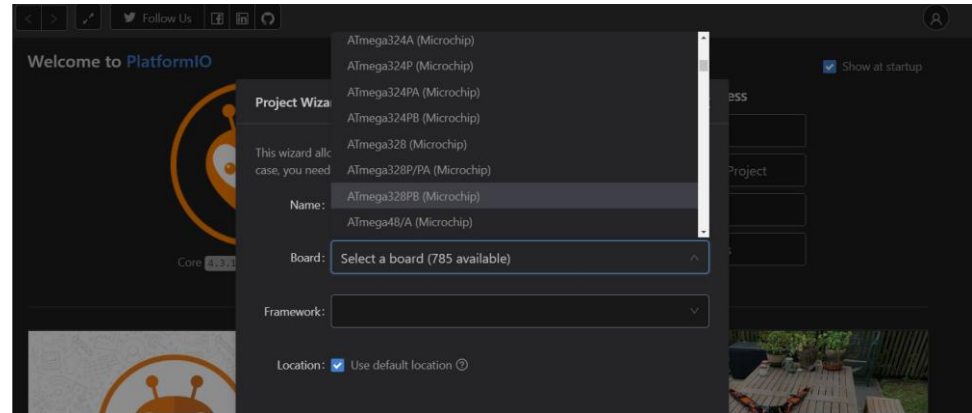
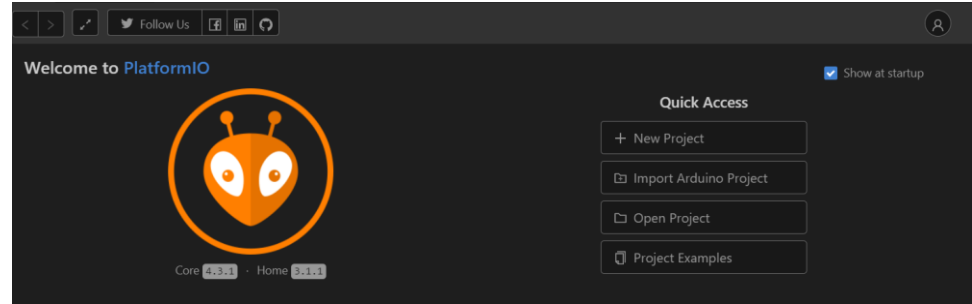
Nieuw project in VSCode



Stappenplan: een nieuw project in VSCode

























- New Project: (in PlatformIO → Open PioHome → Open)

- Klik op New Project
 - Geef een naam
 - Selecteer als board: ATmega328P/PA
 - Verwijder het vinkje
 - Selecteer folder op je harde schijf
 - Klik op finish



Portfolio

**Maak een overzichtelijk
mappenstructuur
van alle C-programma's:**

- ▼  arduino
 - ▼  Demo's Week 1
 - >  Blinking Led
 - >  Flashing LEDS
 - >  Demo's Week 2
 -  Demo's Week 3
 -  Demo's Week 4
- ▼  Extra's
 - >  KITT - Knight Rider
 - >  Startproject
- ▼  libraries
 -  LED
 -  usart
- ▼  Oefeningen Week 1
 - >  LED Dimmen
 - >  LEDChaos
 -  Oefeningen Week 2
 -  Oefeningen Week 3
 -  Oefeningen Week 4
- >  Project Week 1 - MorseCode
 -  Project Week 2
 -  Project Week 3
 -  Project Week 4
 -  Project Week 5 & 6

Stappenplan: een nieuw project in VSCode

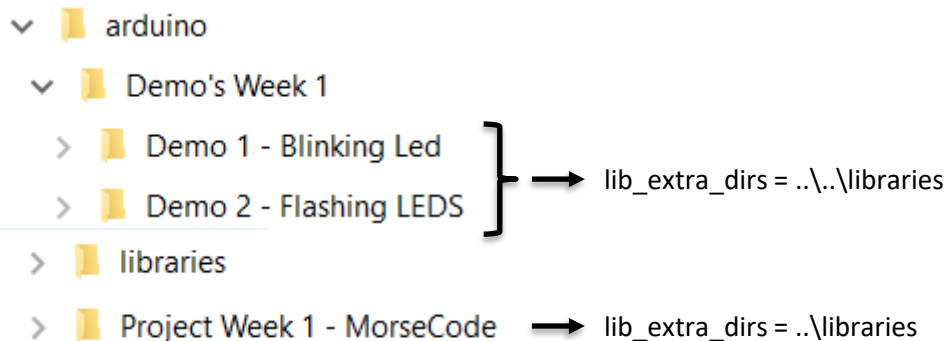
- Het project verschijnt nu links in de Explorer. We moeten nog een paar essentiële wijzigingen doen aan dit project voordat we van start kunnen gaan:

1. Open platformio.ini file

- a. verwijder de laatste regel ("framework = arduino")
- b. Voeg de volgende regel toe : `lib_extra_dirs = F:\arduino\libraries` *//absoluut path*
(= locatie van de library folders usart en eigen geschreven libraries)

```
platformio.ini
1 ;PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:ATmega328P]
12 platform = atmelavr
13 board = ATmega328P
14 lib_extra_dirs = F:\arduino\libraries
```

BETER *// relatief path:*



Stappenplan: een nieuw project in VSCode

2. Hernoem (rechter muisklik) de main.cpp in de src folder naar main.c
3. Verwijder alle code uit de main.c en vervang door volgende code:

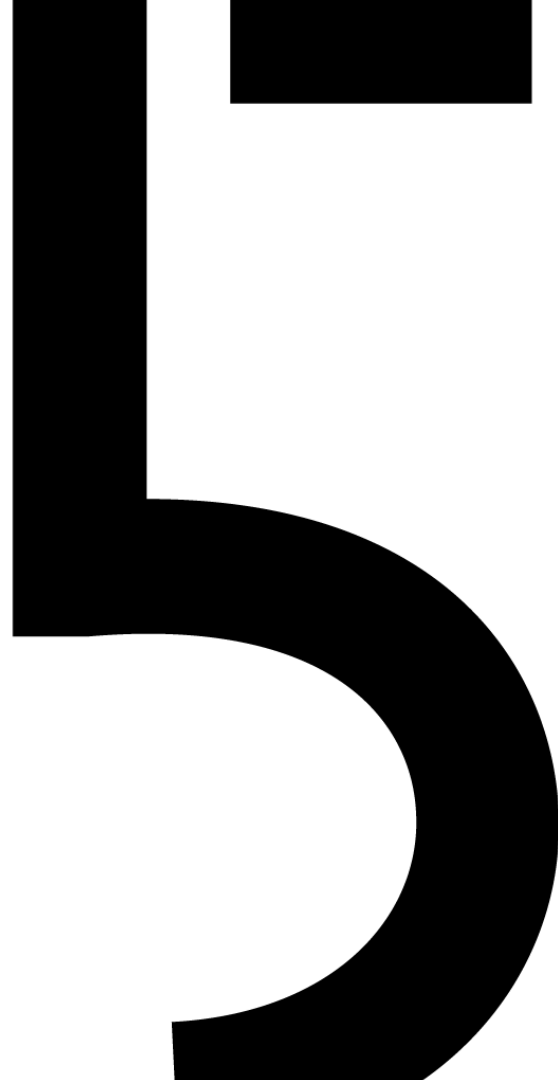
```
#define __DELAY_BACKWARD_COMPATIBLE__
#include <util/delay.h> /* nodig om pauzes in te bouwen */

#include <avr/io.h> /* nodig om AVR registers te kunnen gebruiken */
#include <stdlib.h> /* nodig voor standard functies zoals rand(), srand(),... */
#include <time.h> /* nodig voor tijdsfuncties zoals time(NULL) */
#include <math.h> /* nodig voor wiskundige functies zoals pow() */

int main(){

    return 0;
}
```

Demo 2 – Flashing LEDs



Demo 2 – Flashing LEDs - C-code

```
#include <util/delay.h>
#include <avr/io.h>

#define NUMBER_OF_LEDS 4

void enableLed(int ledNumber)
    if (ledNumber < 1 || ledNumber > NUMBER_OF_LEDS) return;
    DDRB |= (1 << (PB2 + (NUMBER_OF_LEDS - ledNumber)));
}

void lightUpLed(int ledNumber)
    if (ledNumber < 1 || ledNumber > NUMBER_OF_LEDS) return;
    PORTB &= ~(1 << (PB2 + (NUMBER_OF_LEDS - ledNumber)));
}

void lightDownLed(int ledNumber){
    if (ledNumber < 1 || ledNumber > NUMBER_OF_LEDS) return;
    PORTB |= (1 << (PB2 + (NUMBER_OF_LEDS - ledNumber)));
}
```

```
int main(){
    for (int i=1; i<=4; i++) {
        enableLed(i);
    }
    while (1) {
        for (int i=1; i<=4; i++) {
            lightUpLed(i);
            _delay_ms(100);
        }
        for (int i=1; i<=4; i++) {
            lightDownLed(i);
            _delay_ms(100);
        }
    }
    return 0;
}
```

Demo 2 – Flashing LEDs – Achtergrond

Function **enableLed**: `DDRB |= (1 << (PB2 + (NUMBER_OF_LEDS-ledNumber)));`

Stel `DDRB = 0b11000001` en `ledNumber = 1`

⇒ $(1 \ll (\text{PB2} + (\text{NUMBER_OF_LEDS} - \text{ledNumber}))) = 0b00100000$

⇒ `|=`

| | |
|--|----------|
| | 11000001 |
| | 00100000 |
| | ----- |
| | 11100001 |

Bits behouden hun waarde (0 of 1) behalve de bit voor PB5 die op 1 wordt gezet.

Demo 2 – Flashing LEDs – Achtergrond

Function `lightUpLed`:

```
PORTB &= ~(1 << (PB2 + (NUMBER_OF_LEDS-ledNumber)));
```

Stel `PORTB = 0b10100101` en `ledNumber = 1`

$\Rightarrow \sim(1 \ll (\text{PB2} + (\text{NUMBER_OF_LEDS} - \text{ledNumber}))) = 0b11011111$

$\Rightarrow \&=$

| |
|---|
| <pre>10100101 & 11011111 ----- 10000101</pre> |
|---|

Bits behouden hun waarde (0 of 1) behalve de bit voor PB5 die op 0 wordt gezet.

LED Library



LED Library

```
void enableOneLed(int);  
void enableMultipleLeds(uint8_t);  
void enableAllLeds ();
```

```
void lightUpOneLed(int);  
void lightUpMultipleLeds (uint8_t);  
void lightUpAllLeds ();
```

```
void lightDownOneLed(int);  
void lightDownMultipleLeds (uint8_t);  
void lightDownAllLeds ();
```

```
void lightInverseOneLed(int);
```

```
void dimLed(int, int, int);  
    /* parameters: lednumber,  
    percentage, duration */  
void fadeInLed(int, int);  
    /* parameters: lednumber, duration*/  
void fadeOutLed(int, int);  
    /* parameters: lednumber, duration */
```

LED Library

```
#define NUMBER_OF_LEDS 4

void enableOneLed(int ledNumber) {
    if (ledNumber < 1 || ledNumber > NUMBER_OF_LEDS) return;
    DDRB |= (1 << (PB2 + (NUMBER_OF_LEDS - ledNumber))); // de leds beginnen op PB2
}

void enableAllLeds (){
    DDRB |= 0b00111100; // Schrijven activeren op pin 2 tem 5 van port B
}

void enableMultipleLeds(uint8_t leds) {
    if (leds / ((uint8_t) pow(2,NUMBER_OF_LEDS + PB2) )>0 ||
        leds % ((uint8_t) pow(2,PB2)) > 0) return;
    DDRB |= leds;
}
```

LED Library

```
#define NUMBER_OF_LEDS 4
```

```
(leds / ((uint8_t) pow(2,NUMBER_OF_LEDS + PB2) )>0 ||  
 leds % ((uint8_t) pow(2,PB2)) > 0)
```

⇒ $\text{pow}(2, \text{NUMBER_OF_LEDS} + \text{PB2}) = 2^6 = 64$

0b01101001 / 64 = 0b00000001 // 105 / 64 = 1

⇒ $\text{pow}(2, \text{PB2}) = 2^2 = 4$

0b01101001 / 4 = 0b00000001 // 105 % 4 = 1

LED Library

```
void lightInverseOneLed(int ledNumber){  
    if (ledNumber < 1 || ledNumber > 4) return;  
    PORTB ^= (1 << (PB2 + (NUMBER_OF_LEDS - ledNumber))));  
}
```

Stel `PORTB = 0b10100101` en `ledNumber = 1`

⇒ $(1 \ll (\text{PB2} + \text{NUMBER_OF_LEDS} - \text{ledNumber})) = 0b00**1000**00$

⇒ $\wedge =$

| | |
|----------|-----------------|
| | 10100101 |
| ^ | 00100000 |
| | ----- |
| | 10000101 |

Bits behouden hun waarde (0 of 1) behalve de bit voor PB5 die op de andere waarde (1 wanneer het 0 was, 0 wanneer het 1 was) wordt gezet.

Oefeningen



Week 1 - Oefeningen

LED Dimmen

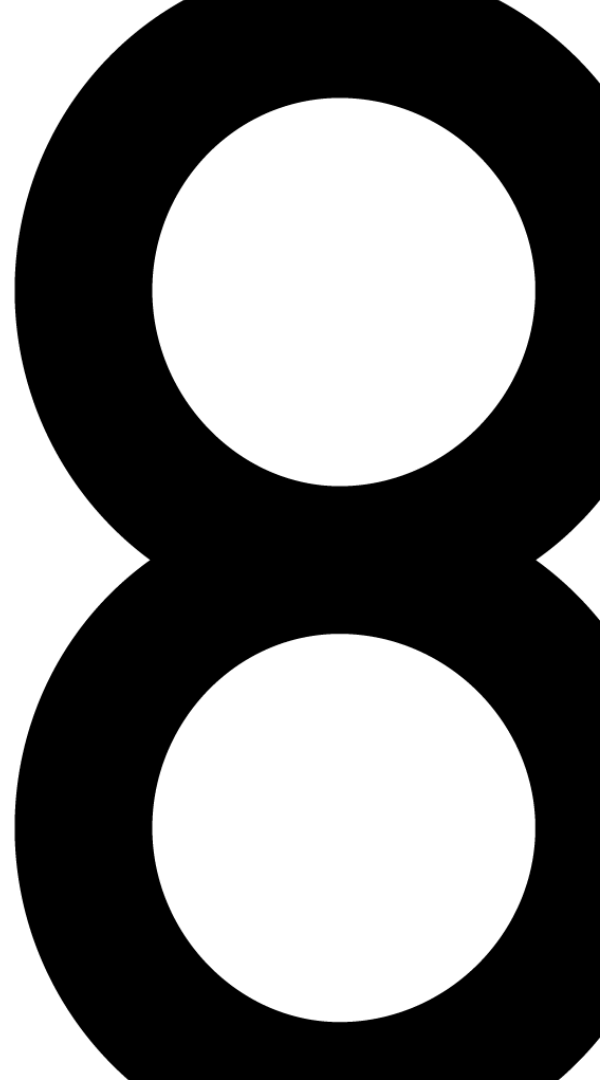
Een led kan enkel volledig aan of volledig af gezet worden. Om een led "half" aan te zetten (te "dimmen" eigenlijk) maken we gebruik van een trukje: we zetten de led in een lus heel snel aan en af zodat hij de helft van de tijd aan is, en de andere helft uit. Als we dat snel genoeg doe, zien onze ogen dat als een lampje dat 50% gedimd is. Om een ledje tot 20% te dimmen moeten we er dan voor zorgen dat het maar 20% van de tijd aan is, en 80% uit. Deze techniek wordt PWM genoemd (**Pulse Width Modulation**).

LED Chaos

Schrijf een programmaatje dat de verschillende leds in een random patroon laat flikkeren: willekeurig gekozen leds worden gedurende willekeurig lange periodes (tussen bijvoorbeeld 100 en 1000 milliseconden) aan en uit gezet.

Om dit te kunnen doen zal je moeten uitvissen hoe je een random getal kan genereren in de programmeertaal C. Je zal hiervoor de stdlib.h library moeten includen. Check de documentatie van de rand() functie.

Week project



Week 1 - Weekproject

Morse trainer

Je bouwt een eenvoudige morse-trainer die je kan gebruiken om te testen hoe goed je morse-code kent.

| | | | | | | | |
|---|-------|---|--------|---|-------|---|----------|
| A | .- | J | .-.-.- | S | ... | 1 | .-.-.-.- |
| B | ---. | K | -.-- | T | - | 2 | ..-.-.- |
| C | -.--. | L | .-... | U | ..- | 3 | ...-- |
| D | -.- | M | -- | V | ...- | 4 |- |
| E | . | N | --. | W | .-.- | 5 | |
| F | ..-. | O | --- | X | ---. | 6 | ----- |
| G | --- | P | ..-.. | Y | -.--. | 7 | ----- |
| H | | Q | ---.- | Z | ---.. | 8 | ----- |
| I | .. | R | .-. | 0 | ----- | 9 | ----- |

- Je toont eerst een "aftelpatroon" van ledjes.
- Vervolgens wordt er een willekeurige letter in Morsecode op de leds getoond en na een korte pauze verschijnt de correcte letter op het computerscherm.
- Na 10 letters eindigt het programma met een frivool led-dansje en begint het terug vanaf start.



Week 1 – Weekproject - usart

Usart library

Zie Canvas voor code en 'installatie'

Gebruik in je c-programma

```
#include <usart.h>

int main()
{
    initUSART();

    printString(stringArray);
}
```

Opmerking: printf() is ook mogelijk maar dat is voor later

Week 1 – Weekproject - array

Array van char

Zie Canvas Tutorials C-taal

Tip1:

```
char stringArray[2];  
stringArray[1] = '\\0';
```

Tip 2:

Serial monitor is terug te vinden op de balk onderaan VSCode:



