CPROUC/2021-2022

Ad van den Bergh

CPROUC

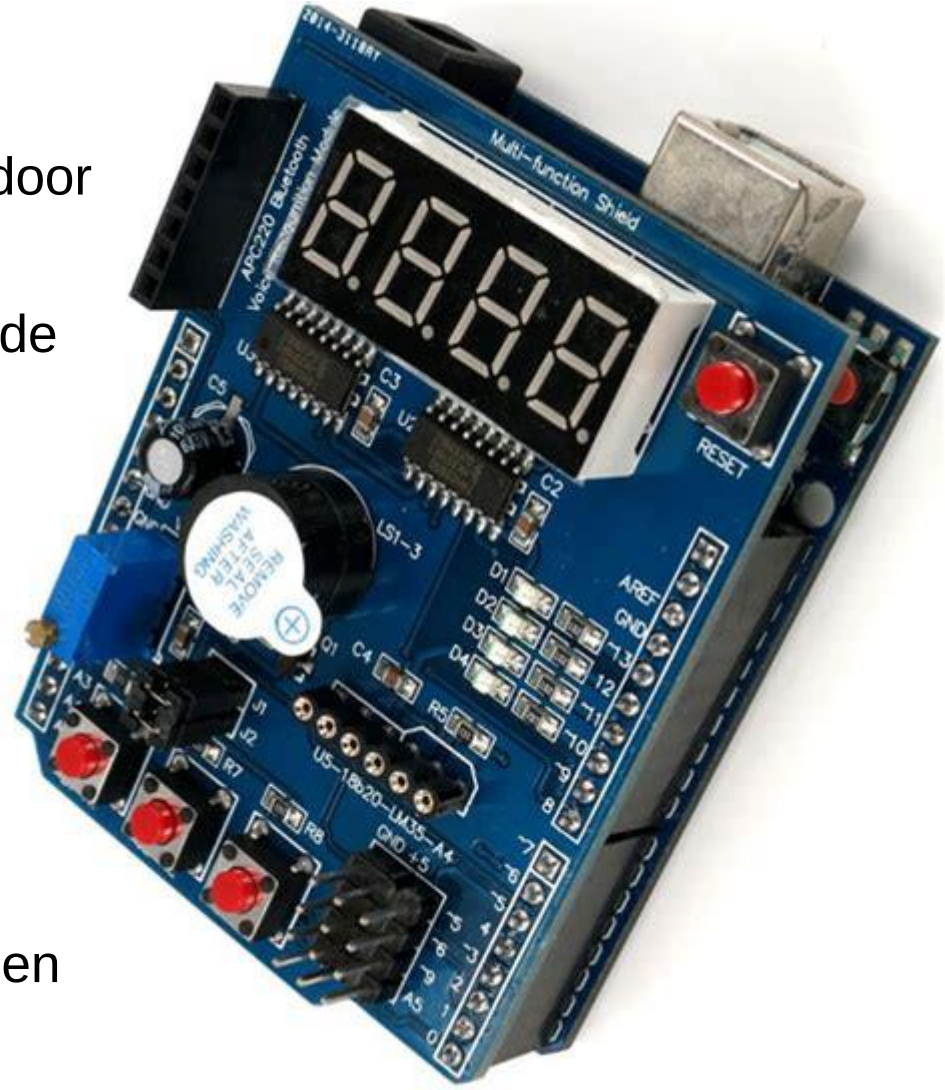
Het MultiFunction Shield en PWM

Week 15 (Les 27 en 28)

DE HAAGSE HOGESCHOOL

Het MultiFunction Shield (MFS)

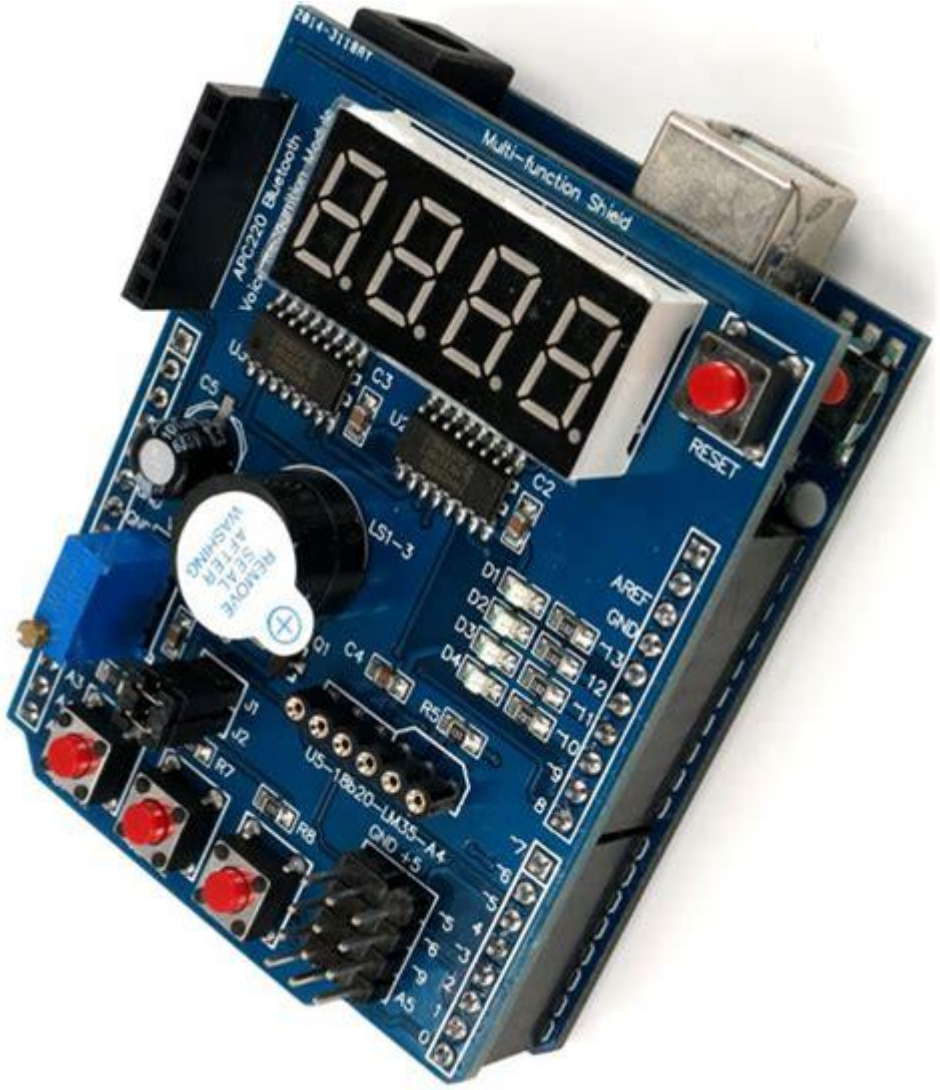
- Tot nu toe alle schakelingen gemaakt (practicum) door toetsen, leds en potmeters op een breadboard te plaatsen en te verbinden met de connectoren van de Arduino UNO.
- Nu gaan we het MultiFunction Shield direct op de Arduino UNO plaatsen.
- VOORZICHTIG: zorg dat alle pinnen in de connectoren zitten, voordat je de MFS aandrukt!
- VOORZICHTIG: MFS moet precies boven Arduino UNO zitten.
- VOORZICHTIG: boven de USB-connector mag geen contact gemaakt worden met MFS.



DE HAAGSE HOGESCHOOL

Het MultiFunction Shield (MFS)

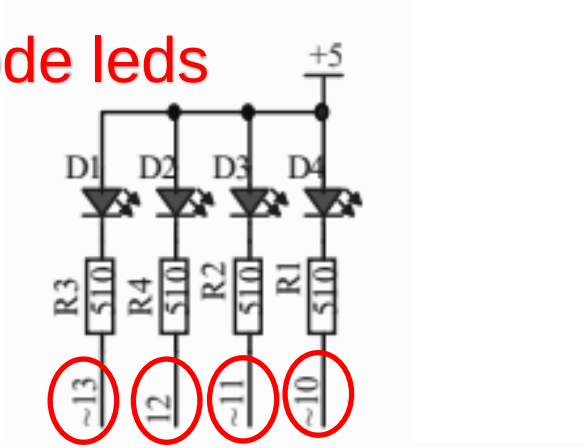
- 4 digit 7 segmentsdisplay
- 1 meerslagenpotmeter
- 4 rode leds
- 3 toetsen
- 1 pieper
- 1 reset-toets

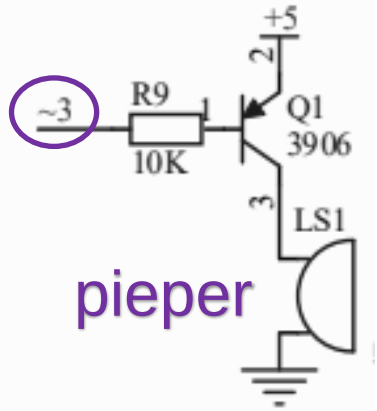


DE HAAGSE HOGESCHOOL

Schema MultiFunction Shield

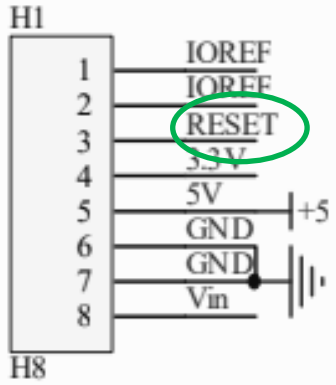
4 rode leds

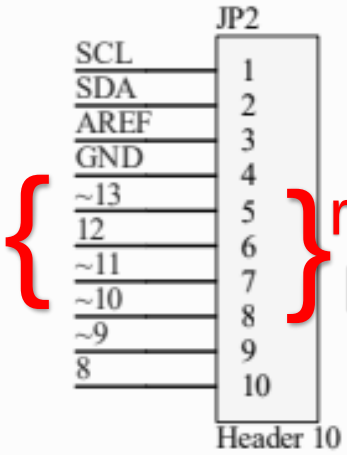




pieper

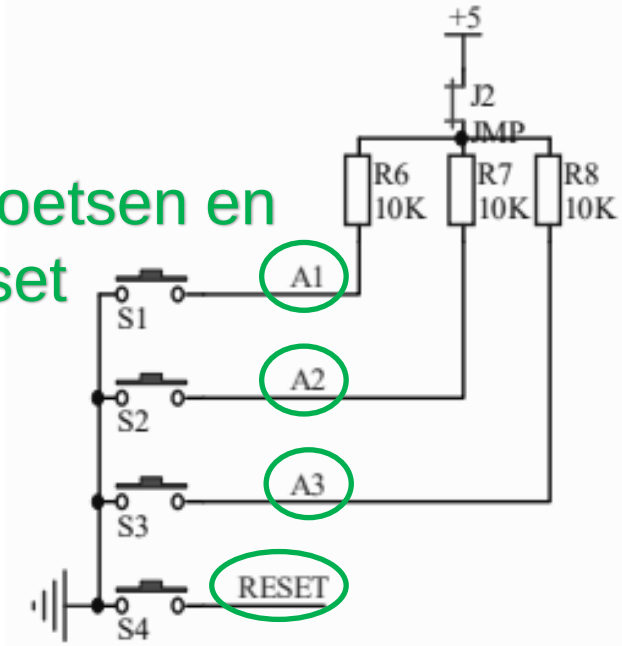
Arduino connectoren

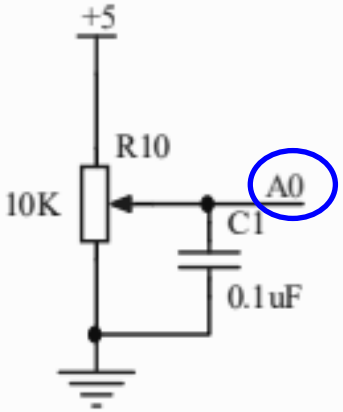




rode leds

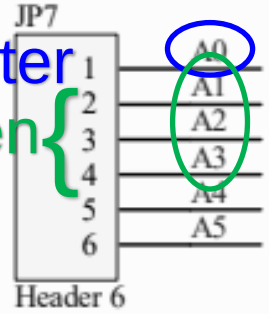
3 toetsen en reset

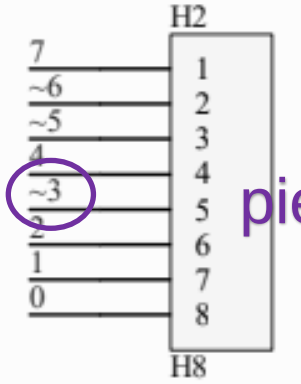




Meerslagen potmeter

potmeter toetsen



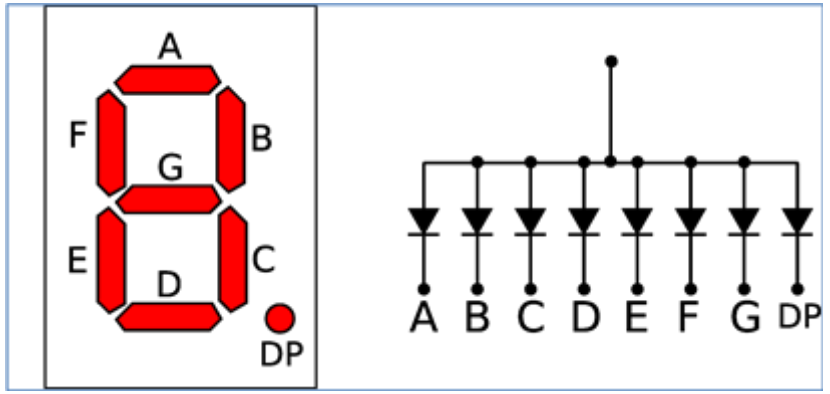


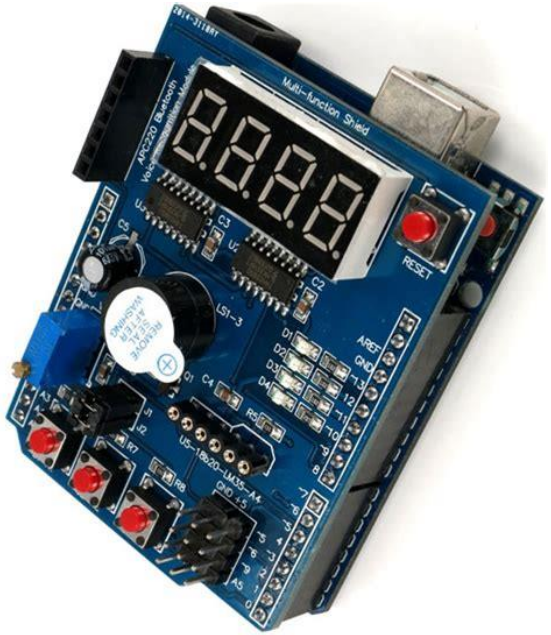
pieper

DE HAAGSE HOGESCHOOL

7 segments display aansturing

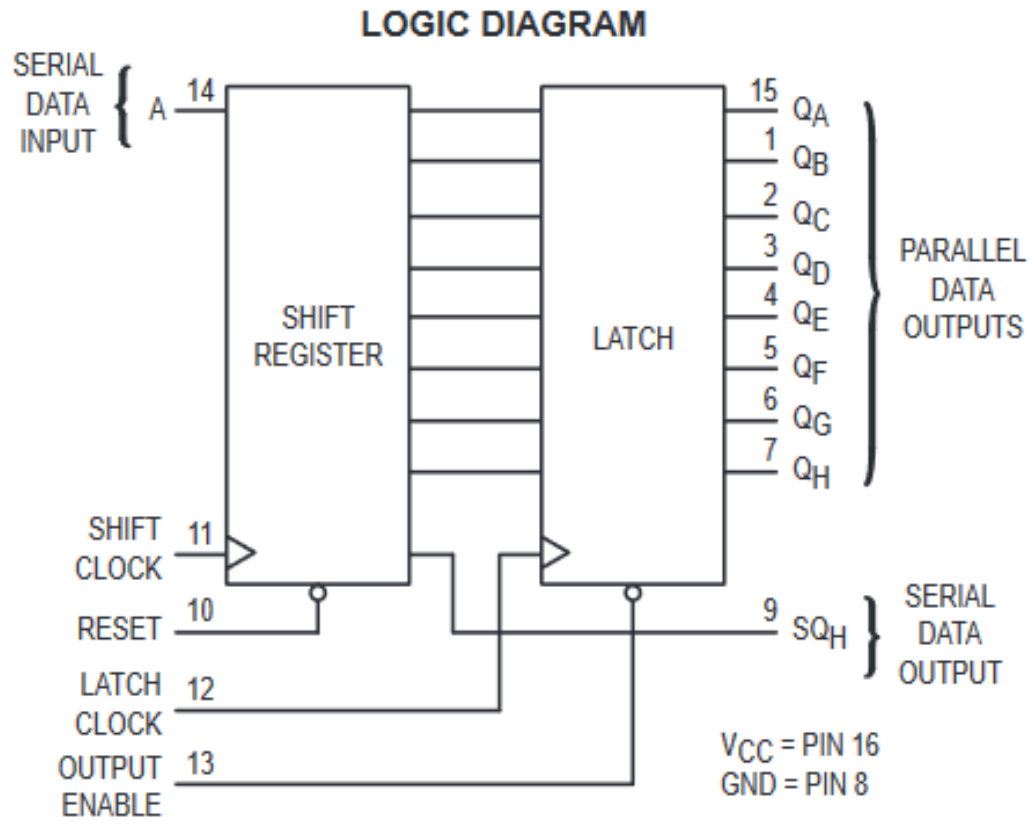
- Gezamenlijke anode – aangesloten op + 5 Volt
- 7 segmenten genummerd van A t/m G en decimale punt (DP)
- Segment gaat aan als logische ‘0’ op betreffende kathode.
- Voor één 7 segmentsdisplay dus 8 uitgangen nodig. (kan net op een Arduino)
- Voor vier digits dus 32 uitgangen nodig. (kan NIET)
- Tenzij ….. gemultiplext aangestuurd wordt.
- Snel achter elkaar digit voor digit aansturen.
- Als snel genoeg en regelmatig dan ziet oog alsof alle digits aan zijn (> 30 Hz)
- Op MFS door middel van schuifregisters (74HC595)





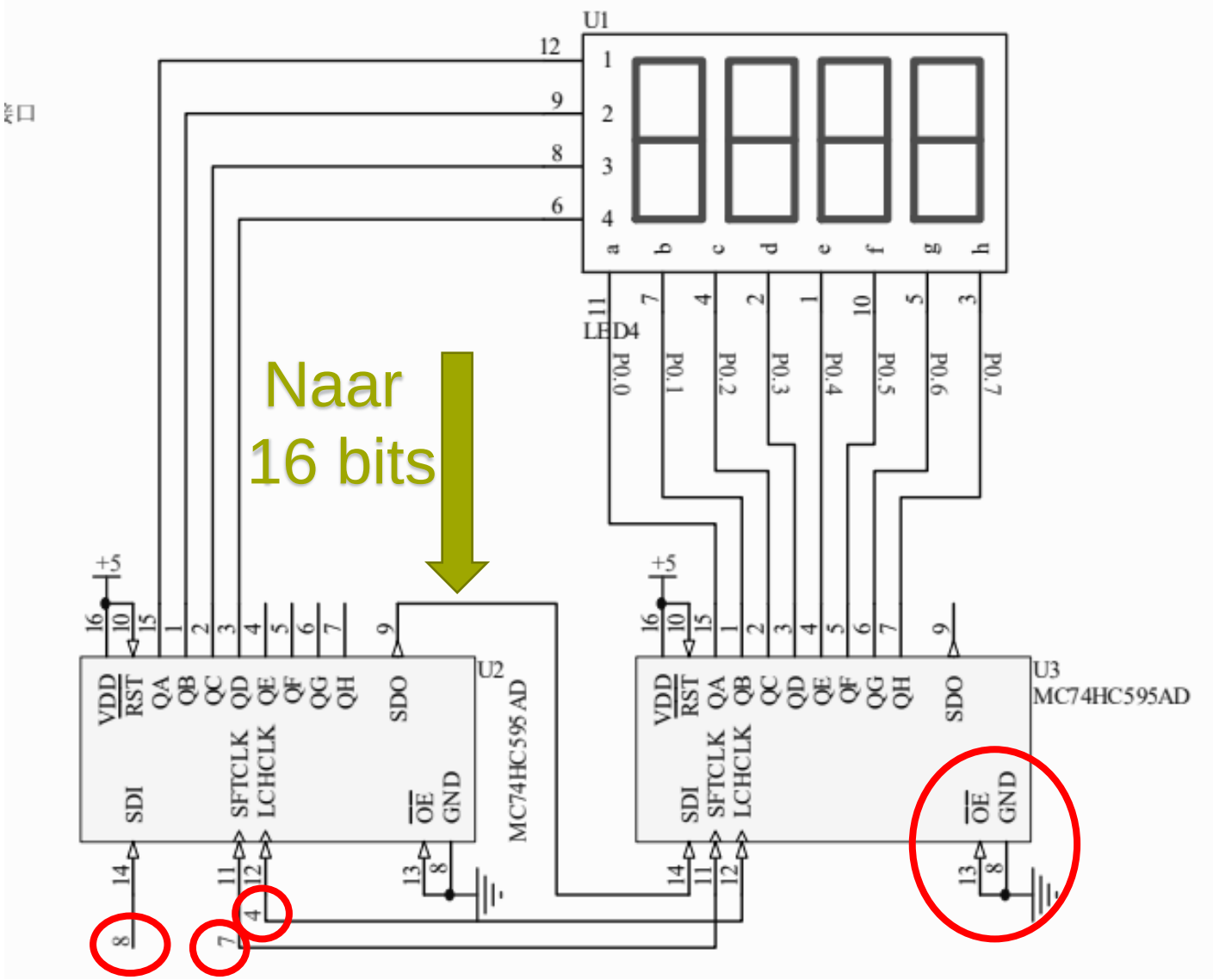
DE HAAGSE HOGESCHOOL

Schuifregister 74HC595

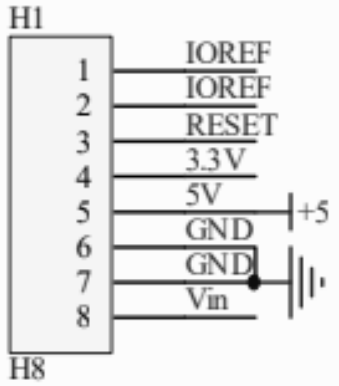


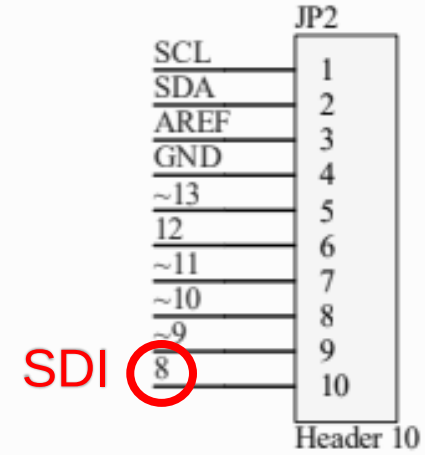
- Als Reset 1 is, wordt op positieve flank van SHIFT CLOCK de SERIAL DATA INPUT ingeschoven naar A en A>B>C>D etc.
- H wordt eruit geschoven naar uitgang SQH om schuifregisters met meer bits te kunnen realiseren.
- De waarden van A t/m H kunnen worden overgenomen en vastgehouden door de LATCHCLOCK logisch 1 te maken (overnemen) en daarna 0 (vasthouden).
- De PARALLEL DATA OUTPUTS laten alleen de vastgehouden data zien als de OUTPUT ENABLE logisch 0 is (anders hoogohmig)

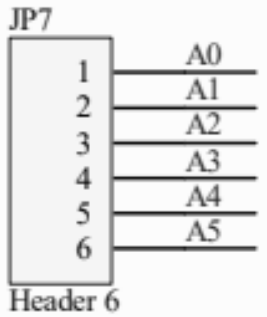
Schema MultiFunction Shield

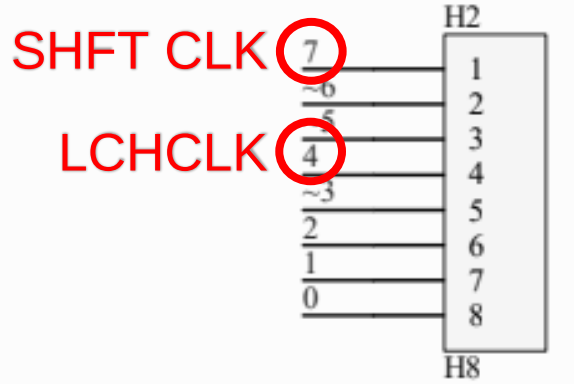


Met 3 signalen
4 digits aan te sturen









DE HAAGSE HOGESCHOOL

Sketch display aansturen (setup en loop)

```
66 void setup()
67 {
68   pinMode( DATA_IO, OUTPUT);
69   pinMode( CLK_IO, OUTPUT);
70   pinMode( LATCH_IO, OUTPUT);
71 }
72 void loop()
73 {
74   teller++;
75
76   if (teller < 1000) {
77     SEGMENT_CONTENT[0] = LetterH;
78     SEGMENT_CONTENT[1] = LetterH;
79     SEGMENT_CONTENT[2] = LetterS;
80     SEGMENT_CONTENT[3] = SPACE;
81   }
82   else {
83     if (teller > 2000) teller = 0;
84     SEGMENT_CONTENT[0] = LetterE;
85     SEGMENT_CONTENT[1] = LetterL;
86     SEGMENT_CONTENT[2] = LetterE;
87     SEGMENT_CONTENT[3] = LetterC;
88   }
89
90   for (uint8_t i = 0; i < 4; i++) {
91     WriteDataToSegment(i, i);
92   }
93 }
```

- Doel: afwisselend op display laten zien: HHS en ELEC
- 3 besturingssignalen als outputs zetten (68 t/m 70)
- De eerste seconde HHS tonen (76 – 80)
- De tweede seconde ELEC tonen (83 - 88)
- Waardes in array plaatsen SEGMENT_CONTENT[4]
- Schuifregisters aansturen per digit in regel 90 – 92.

Sketch display aansturen (WriteDataToSegment)

```
40
41 //Besturen van het schuifregister
42 void WriteDataToSegment(int8_t Segment, int8_t Patroon)
43 {
44   digitalWrite(CLK_IO, HIGH);
45   digitalWrite( LATCH_IO, LOW);
46   shiftOut(SEGMENT_CONTENT[Patroon]);
47   shiftOut(SEGMENT_SELECT[Segment]);
48   digitalWrite( LATCH_IO, HIGH);
49 }
```

- Eerst moet CLK_IO hoog worden en LATCH_IO laag zodat niet onrustig beeld door doorschuiven ontstaat.
- Dan eerst de 8 bits van de data sturen met MSB eerst (zie schema) regel 46
- Dan 8 bits om segment te selecteren in regel 47
- Nadat 16 bits zijn gestuurd wordt de data vastgehouden door LATCH_IO weer hoog te maken

Sketch display aansturen (WriteDataToSegment)

```
25 //Converteer een nummer naar een ruwe data
26 //voor het shiftregister op het 7-segments display
27 void shiftOut( int8_t Data)
28 {
29   int8_t j;
30   for (j = 0; j < 8; j++)
31   {
32     if (Data & 0x80)
33       digitalWrite(DATA_IO, HIGH);
34     else
35       digitalWrite(DATA_IO, LOW);
36     digitalWrite(CLK_IO, LOW);
37     Data = Data << 1;
38     digitalWrite(CLK_IO, HIGH);
39   }
40 }
```

- Omdat eerst MSB bitje gestuurd moet worden, de Data eerst AND met 0x80
- Resultaat 0x80(true) of 0x00 (false)
- Als DATA_IO HIGH, dan led uit (33 - 35)
- Daarna CLK_IO een positieve flank laten geven (36 en 37)
- Data één positie naar links laten verschuiven (37) en procedure herhalen.

Sketch display aansturen (WriteDataToSegment)

1 0 0 1 1 0 1 1 Data =0x9B

1 0 0 0 0 0 0 0 0x80

&

1 0 0 0 0 0 0 0

1 0 0 1 1 0 1 1 Data =0x9B

0 0 1 1 0 1 1 0 Data << 1

1 0 0 0 0 0 0 0 0x80

&

0 0 0 0 0 0 0 0

0 0 1 1 0 1 1 0 Data << 1 (j=1)

0 1 1 0 1 1 0 0 Data << 1 (j=2)

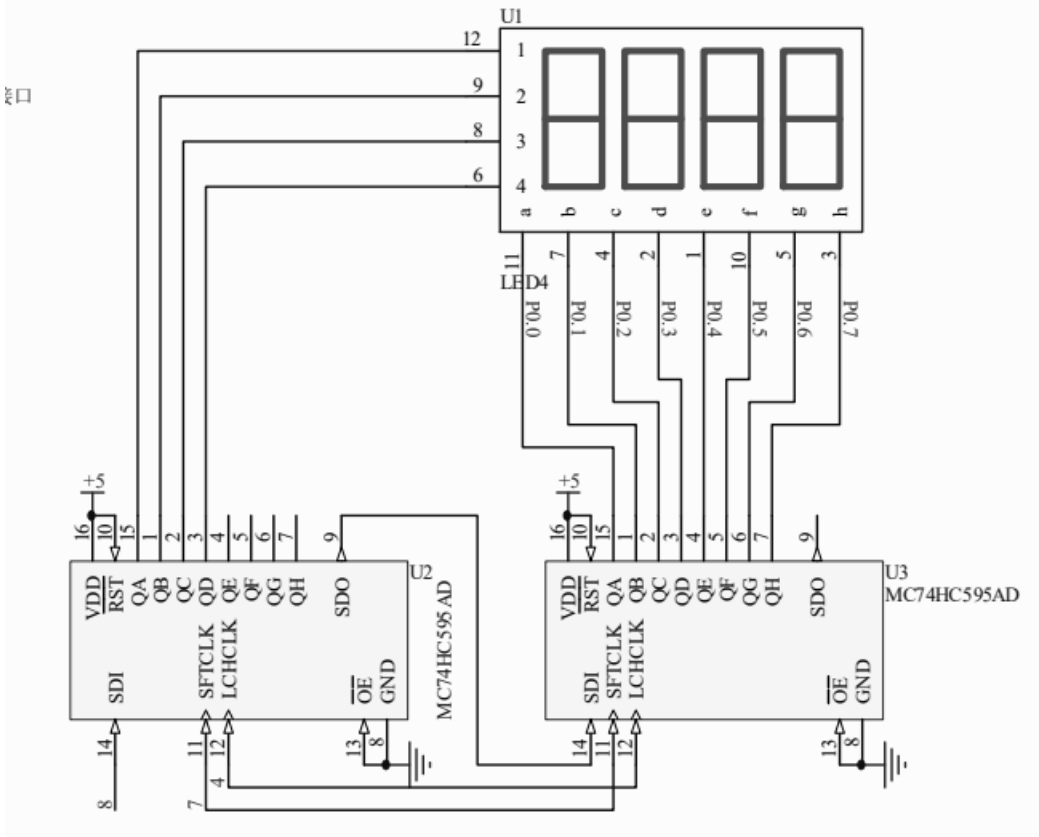
1 1 0 1 1 0 0 0 Data << 1 (j=3)

```
25 //Converteer een nummer naar een ruwe data
26 //voor het shiftregister op het 7-segments display
27 void shiftOut( int8_t Data)
28 {
29   int8_t j;
30   for (j = 0; j < 8; j++)
31   {
32     if (Data & 0x80)
33       digitalWrite(DATA_IO, HIGH);
34     else
35       digitalWrite(DATA_IO, LOW);
36     digitalWrite(CLK_IO, LOW);
37     Data = Data << 1;
38     digitalWrite(CLK_IO, HIGH);
39   }
40 }
```


DE HAAGSE HOGESCHOOL

Maar nu de codering van de digitselectie:

- Wat moet de code worden om eerste digit aan te sturen?
- QA van linker 595 moet 1 zijn en QB, QC en QD moeten 0 zijn. Rest mag 1 zijn:
- Digit 0: 0B 1111 0001 = 0xF1
- Dus:
- Digit 1: 0B 1111 0010 = 0xF2
- Digit 2: 0B 1111 0100 = 0xF4
- Digit 3: 0B 1111 1000 = 0xF8

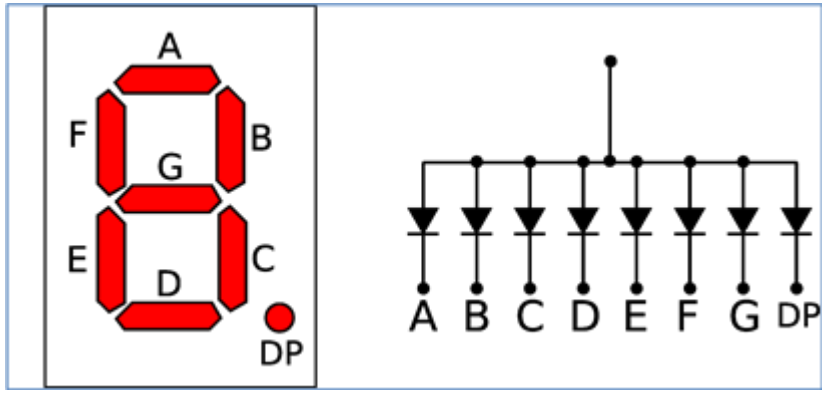


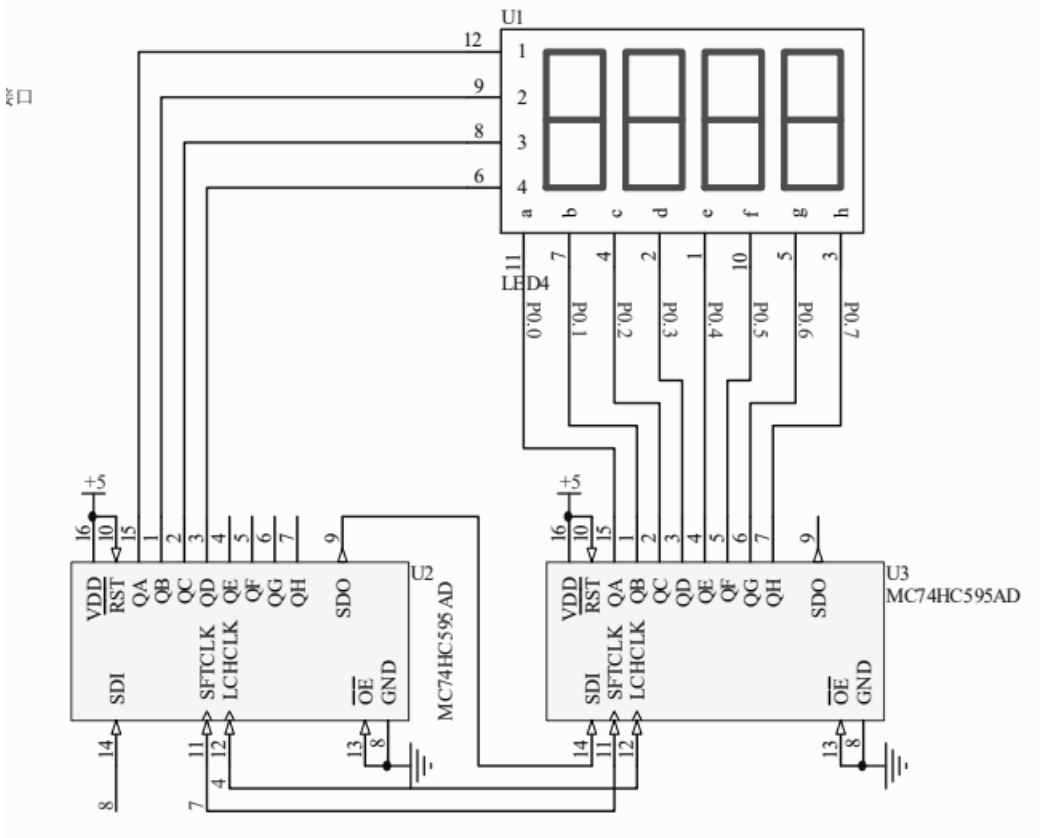

```
18 int8_t SEGMENT_SELECT[] = {0xF1, 0xF2, 0xF4, 0xF8}; // codes voor selectie vier segmenten van links naar rechts
```


DE HAAGSE HOGESCHOOL

Maar nu de codering van de digitselectie:

- Wat moet de code worden om een H op een digit te krijgen?
- Dan moet `a` en `d` en `dp` (is h) uit
- Volgorde: `dp g f e d c b a`
- Dus: `1 0 0 0 1 0 0 1` = 0x89





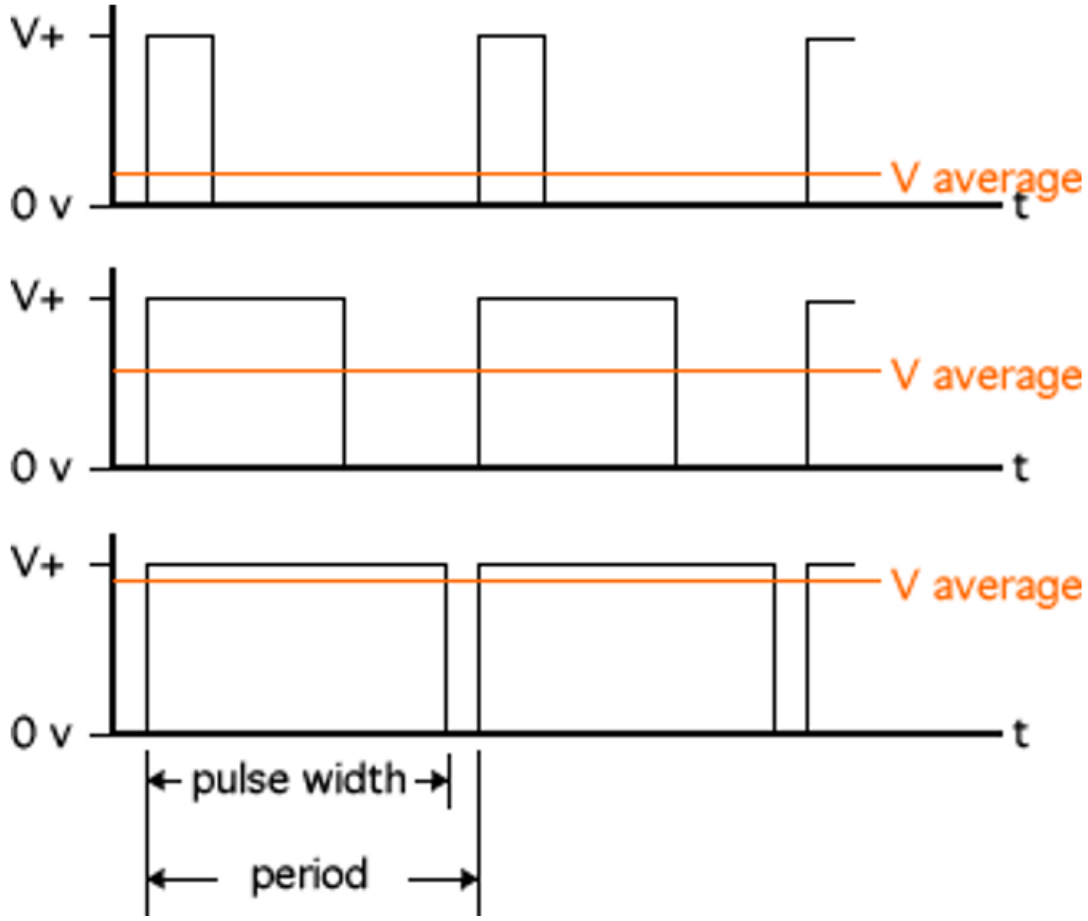

```
7 const int8_t LetterH = 0x89;     //1000 1001 dp g f e   d c b a
```

Demo

DE HAAGSE HOGESCHOOL

Pulse Width Modulation (PWM)

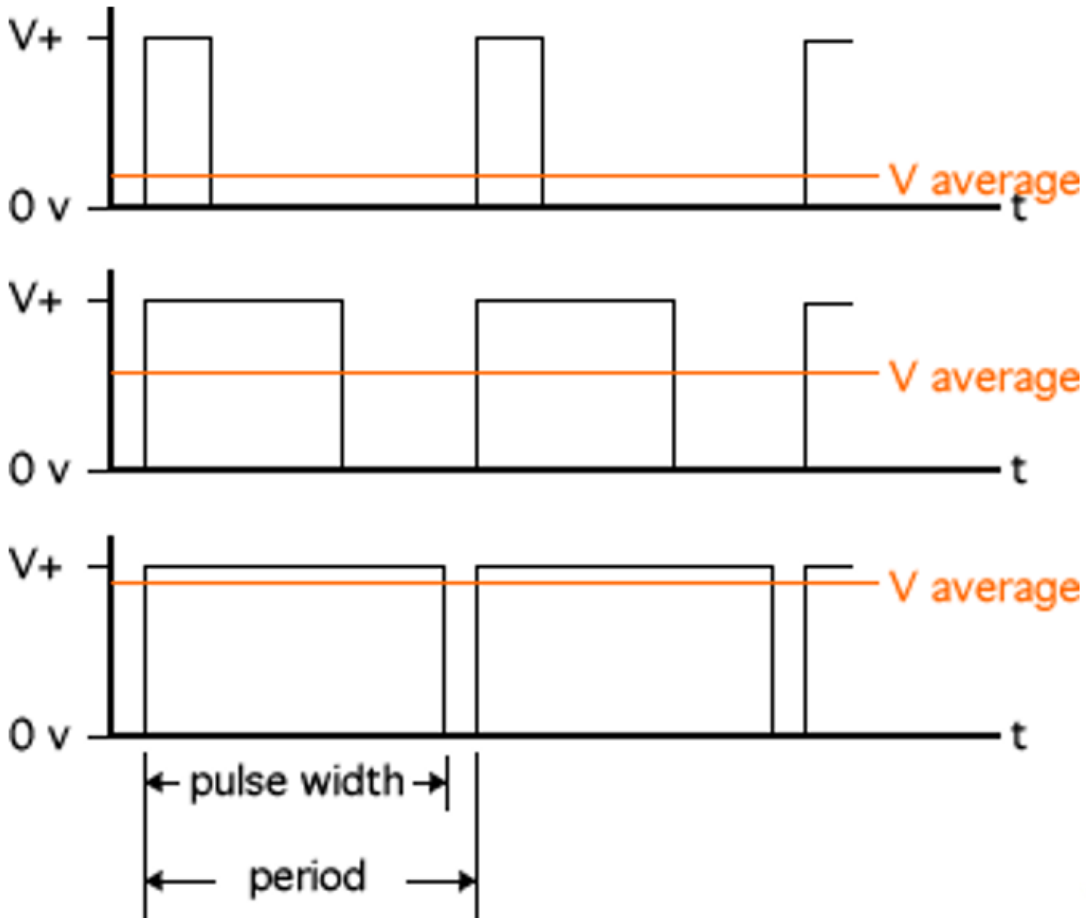
- Naast het binnenhalen van analoge signalen moet een technisch systeem vaak ook analoge waarden afgeven (aansturen DC-motor, analoge meter met wijzer etc.).
- Op de ATmega 328 zit wel een Analoog Digitaal Converter, maar geen Digitaal Analoog Converter.
- Dit wordt opgelost met Puls Breedte Modulatie (PBM) of in het Engels: Pulse Width Modulation (PWM)



DE HAAGSE HOGESCHOOL

Duty Cycle

- De Duty Cycle is de verhouding tussen de pulsbreedte en de periode
- In formule: $DC = \frac{pulse\ width}{period} \cdot 100\%$
- Dus DC = 100% dan staat V+ op uitgang.
- DC = 0% dan staat 0 V op uitgang.
- DC = 25% dan staat V+/4 Volt op uitgang.
- Toepassingen: dimmen led en toerenregeling DC-motor en servomotoren.



DE HAAGSE HOGESCHOOL

analogWrite(pin, waarde);

- We kenden al analogRead om via de ADC analoge waarden in te lezen.
- Nu `analogWrite( int pin, int waarde)` om een PWM-signaal te genereren op uitgang `pin` en met een `waarde` van 0 tot 255.
- `waarde = 255;` DC = 100% dan staat +5 Volt op uitgang.
- `waarde = 0;` DC = 0% dan staat 0 V op uitgang.
- `Waarde = 63;` DC = 25% dan staat 1,25 Volt op uitgang.
- Voorbeeld: led dimmen met behulp van een Rotary Encoder

Voorbeeld dimmen led (setup)

11.1_Dimmen_met_Rotary_encoder_pol

```
int helderheid = 120;     // helderheid van de LED; start op halve helderheid
int stapHelderheid = 10;     // stapgrootte in helderheid van de LED
unsigned long actueleTijd;
unsigned long cyclusTijd;
const int pin_A = 2;  // pin 2 de CLK aansluiting van de Rotary encoder
const int pin_B = 3;  // pin 3 De DT aansluiting van de Rotary encoder
unsigned char encoder_A;
unsigned char encoder_B;
unsigned char encoder_A_vorige=0;

void setup()   {
  // declareer pin 9 als uitgang (output):
  pinMode(9, OUTPUT);
  pinMode(pin_A, INPUT);
  pinMode(pin_B, INPUT);
  actueleTijd = millis();
  cyclusTijd = actueleTijd;
}
```

Voorbeeld dimmen led (loop)

```
void loop()   {
  // haal de verstreken tijd op sinds laatste sample
  actueleTijd = millis();
  if(actueleTijd >= (cyclusTijd + 5)){
    // 5ms sinds de laatste test van de encoder = 200Hz
    encoder_A = digitalRead(pin_A);     // Lees de waarde van de encoder pins
    encoder_B = digitalRead(pin_B);
    if((!encoder_A) && (encoder_A_vorige)){
      // A is van hoog naar laag gegaan
      if(encoder_B) {
        // B is hoog, dus de encoder is "met de klok mee" (naar rechts) gedraaid
        // verhoog de helderheid, tot maximaal 255 (8 bits)
        if(helderheid + stapHelderheid <= 255) helderheid += stapHelderheid;
      }
      else {
        // B is laag, dus de encoder is "tegen de klok in" (naar links) gedraaid
        // verlaag de helderheid, tot minimaal 0
        if(helderheid - stapHelderheid >= 0) helderheid -= stapHelderheid;
      }
    }
    encoder_A_vorige = encoder_A;      // Sla de waarde van A op voor de volgende keer
    // Stuur de helderheid naar pin 9, waarop de LED is aangesloten:
    analogWrite(9, helderheid);
    cyclusTijd = actueleTijd;  // actualiseer de cyclusTijd
  }
  // Hier kunnen andere activiteiten worden uitgevoerd (wel binnen 5 ms blijven)
}
```

Demo

Voorbeeld servosturing

- Een microservomotor wordt vaak gebruikt in RC vliegtuigjes voor het besturen van het “roer” en de “flaps”, maar ook voor de brandstoftoevoer.
- De servo krijgt een hoekverdraaiïng, afhankelijk van de Duty Cycle van het aangeboden PWM-signaal.
- De meeste miniservomotoren hebben een maximale hoekverdraaiing van 180 graden.
- Maar in praktijk tot maximaal 170 graden omdat de DC groter moet zijn dan 6% (hij moet pulsen blijven ontvangen).

DE HAAGSE HOGESCHOOL

Gebruik library Servo.h

- We zouden de microservomotor direct kunnen aansturen vanuit de Arduino met bijv. analogWrite(9,128) om een hoekverdraaiing van 90 graden te krijgen. (werkt prima).
- Wij gaan nu echter een library gebruiken die door Arduino is gemaakt en de naam `Servo.h` heeft.
- Zoals uit eerste deel van CPROUC bekend is, zal dus de volgende regel aan het begin van de sketch moeten worden opgenomen:
  - `#include <Servo.h>`
    - De compiler kent deze library al (vandaar de <>)
  - We kijken eerst naar het voorbeeldprogramma uit de Arduino IDE onder Servo (sketch knob)

Voorbeeld sketch: Knob.ino``

Knob | Arduino 1.8.16

Bestand Bewerken Schets Hulpmiddelen Help

Knob

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = A0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it for use with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

Demo

Voorbeeld sketch: Analoge VU-meter

- Na dit voorbeeld gezien te hebben, herken je waarschijnlijk wel de oude analoge VU-meters op de oude mengpanelen….?
- Ook een wijzer die naar rechts draait als het volume hoger wordt…..
- Dus gaan we onze digitale VU-meter uit het practicum even uitbreiden met een analoge variant met servomotor!
- Je krijgt de sketch niet te zien, want je mag het zelf gaan bedenken in een practicum opdracht!

Demo

DE HAAGSE HOGESCHOOL

let’s change