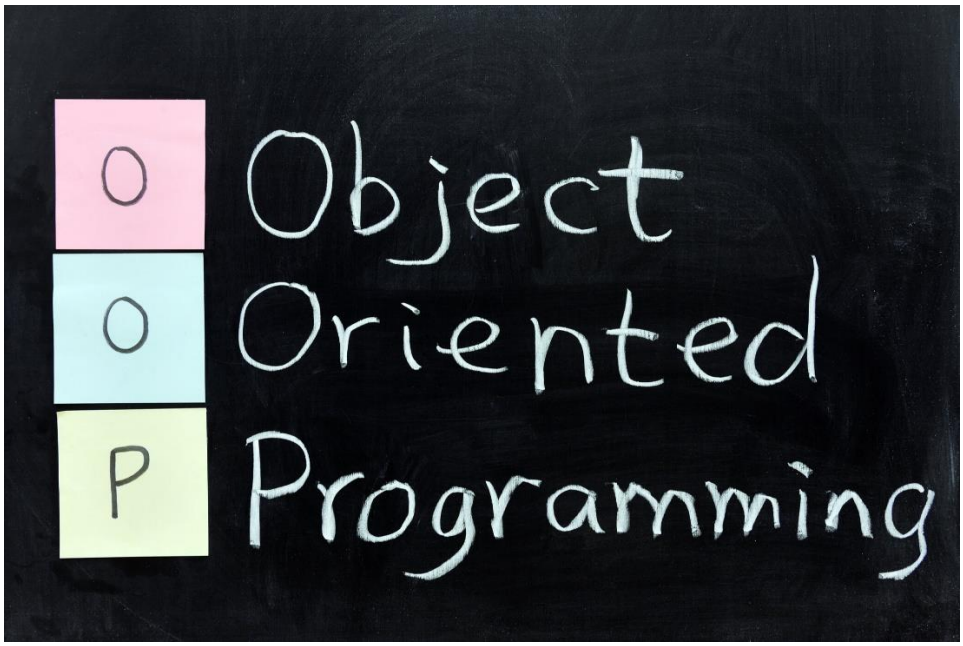
# Startsemester Verdiepingsverslag Technology




**Studentnaam: Marijn Verschuren**
**Studentnummer: 510936**

**Klas: PD11**
**Vakdocent: Rop Pulles**

**Versie: V1.5**
**Datum: 02-01-2023**

**Versiebeheer**

| Versienummer | Datum | Auteur | Veranderingen |
|---|---|---|---|
| *V1.0* | *18-12-2022* | *Marijn Verschuren* | *TinkerCad design, Breadboard design, Arduino tester, blinking led, debounce, trafic control, IO techniques (digital, analog, UART/ USART)* |
| *V1.1* | *19-12-2022* | *Marijn Verschuren* | *CAN, I2C, SPI* |
| *V1.2* | *20-12-2022* | *Marijn Verschuren* | *Sensors* |
| *V1.3* | *28-12-2022* | *Marijn Verschuren* | *Serial plotter, Running average / Hysteresis, C#* |
| *V1.4* | *02-01-2023* | *Marijn Verschuren* | *PWM en Remote control.* |
| *V1.5* | *13-01-2023* | *Marijn Verschuren* | *Inleiding, Introductie, Eigen project* |

# Inhoud

# 1 Inleiding

## 1.1 Aanleiding

Ik heb dit verslag geschreven om mijn leerdoelen aan te tonen voor de verdiepende fase. Hiervoor heb ik de opdrachten op verdiepend niveau gemaakt.

## 1.2 Onderwerp

In dit verslag worden alle opdrachten van de verdiepende fase die ik heb gemaakt uitgelegd deze opdrachten zijn geschreven in C/C++ en C#.

# 2 Introductie

Ik ben Marijn Verschuren ik ben 18 jaar oud en ik heb voor deze opleiding HAVO gedaan, naast HAVO heb ik mezelf leren coden voornamelijk in python en C/C++. Ik heb dan ook een RC-Car gemaakt voor mijn profielwerkstuk hierbij had ik gebruik gemaakt van een arduino. Omdat ik werken met de arduino wel interessant maar soms iets te limiterend vond ben ik verder gaan werken met STM boards, Omdat ik hier verder mee wil werken heb ik Tech gekozen.

# 3 Tabel Leeruitkomsten

| | Object Oriented Programming (OOP) | Classes | Objects | Constuctors | Private Fields | Properties | Methods | Encapsulatie | Robuuste progrs | Sensoren en actuatoren | Analyse sensoren | IO Technieken | PWM | Analoge Input | Communicatie met ander systeem | Protocol | Robuste communicatie |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TinkerCad + Breadboard circ | | | | | | | | | | X | | X | X | X | | | |
| Arduino Tester | | | | | | | | | | | | X | | | X | X | |
| Non Blocking ... | X | X | X | X | X | | X | X | X | X | | X | | | | | |
| Trafic controll | X | X | X | X | X | | X | X | X | X | | X | | | | | |
| Serial Plotter ... | | | | | | | | | | X | X | X | | X | X | X | |
| Trafic Control C# | X | X | X | X | X | | X | X | X | | | | | | | | |
| Bank Account C# | X | X | X | X | X | X | X | X | X | | | | | | | | |
| Remote controll | X | X | X | | | | X | | X | X | | X | X | | X | X | |
| Eigen project + Running avg + Hysteresis (deze maaken onderdeel uit van het project) | | | | | | | | | | X | X | X | | X | X | X | X |

# 4 Tinker CAD:

**Challenge Beschrijving**:

1. Go to https://www.tinkercad.com/ and create an account and sign in.
2. Choose the 'Circuits' section to go to create or edit a circuit.
3. Find a place the Arduino with breadboard in the working area (search for 'arduino uno').
4. Find and place the desired components and connect them in the proper way to the breadboard.
5. Select Code->Text to write C code.
6. Click 'Start Simulation' to start the simulation.

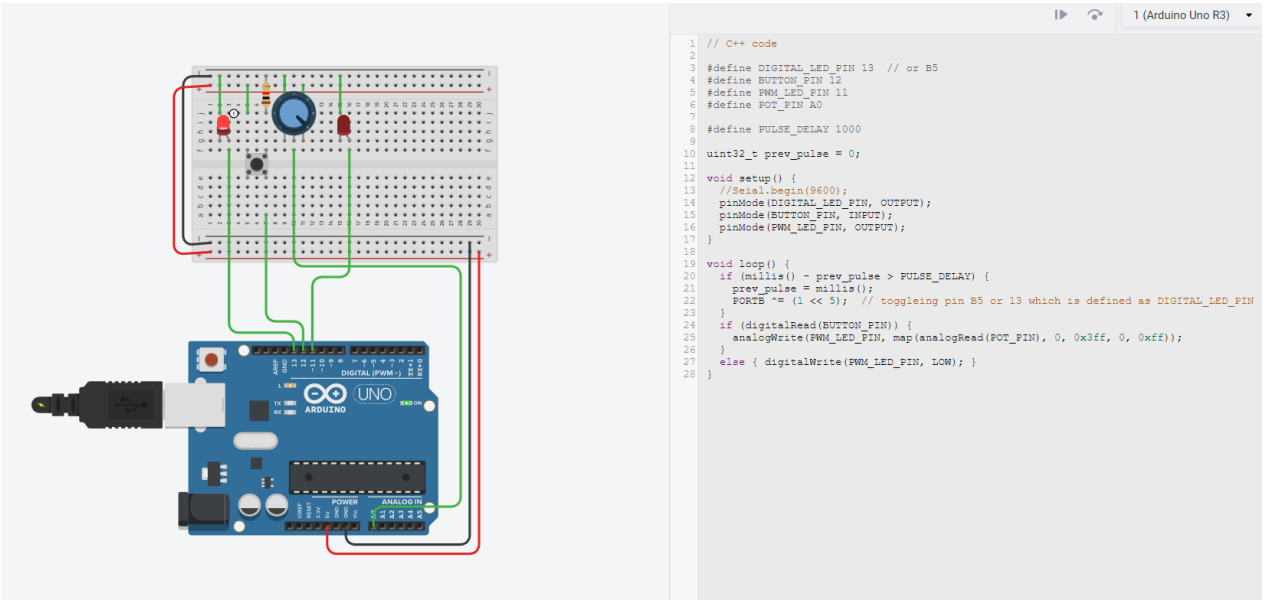
**Aanpak:**

Ik heb eerst het circuit opgebouwd en daarna de code gemaakt en getest

**Wat heb ik geleerd?**

Ik heb geleerd om tinker CAD te gebruiken

**Resultaat:**




De volgende twee regels in de code die enigszins bijzonder zijn:

```
PORTB ^= (1 << 5);  // toggleing pin B5 or 13 which is defined as DIGITAL_LED_PIN
```

In deze regel wordt een pin telkens hoog en laag geschakeld dit heb ik via een GPIO register gedaan omdat Arduino geen toggle functie heeft. In dit geval heb ik de port en het bit index van pin 13 opgezocht in het pinout diagram (PORTB, PIN5). Daarna word een xor uitgevoerd op bit 5 van PORTB hierdoor word pin13 geschakeld.

```
analogWrite(PWM_LED_PIN, map(analogRead(POT_PIN), 0, 0x3ff, 0, 0xff));
```

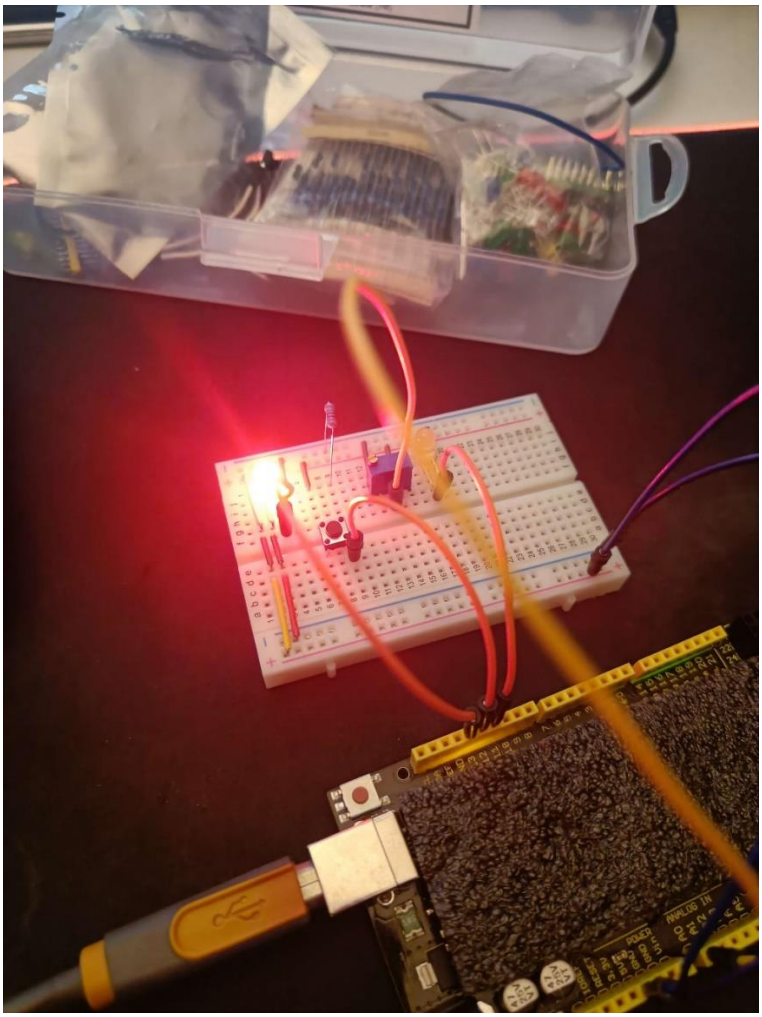
In deze regel word de analoge pin die van de potmeter afgelezen dit geeft een waarde van 0 tot 1023 dit word omgerekend naar een waarde van 0 tot 255. Deze waarde wordt daarna uitgeschreven naar de PWM pin van de led.

# 5 Breadboard Circuit

**Challenge Beschrijving**:
Build up the same circuits as above on a real breadboard as above and test them. Do you see any difference?

**Aanpak:**
Ik heb het circuit van Tinker CAD gebouwd op een breadnoard en de code geupload

**Resultaat:**



Het nagebouwde circuit had maar een verandering en dat was het bit nummer in PORTB omdat ik een Ardiuno Mega2560 gebruik (code).

# 6 Arduino tester

**Challenge Beschrijving**:
How nice would it be to be able to quickly test your Arduino? At least you want to know if all digital inputs, digital outputs and all analog inputs are working. Try to think of an efficient way with not too many steps for the tester.

**Aanpak:**
Ik test de pins door ze aan te zetten en dan te checken of deze daadwerkelijk aan zijn gezet door te kijken in de PIN registers

**Uitleg:**

Voor deze pin tester heb ik geen externe onderdelen nodig gehad maar als ik een verzie zou maken met een led, button en potentiometer zou ik eerst de led aanzetten en vervolgens het programma laten wachten op een button press waarna een bericht gestuurd zal worden naar de serial console waar in staat wat de waarde van de potmeter was en welke pins nu getest kunnen worden. Ik heb in dit geval voor een simpele software oplossing gekozen waarin de pins alleen getest worden op digitale output (PWM inbegrepen) maar dit is alsnog boeiend voor analoge inputs omdat de pin simpelweg doorgebrand kan zijn hier word dus wel voor getest. Deze tester is dus niet geschikt voor het testen van de ADC peripheral.

ArduinoTester.ino iomxx0_1.h

```
// Arduino pin testing code using the GPIO registers
// PIN regs are at 3n     (holds state)
// DDR regs are at 1+3n   (holds data dir) (set = output)
// PORT regs are at 2+3n  (set state)

void setup() {
  // serial to send failures back
  Serial.begin(9600);
}

// holds all excluded pins from registers a - g
const char port_names[7] = {'A', 'B', 'C', 'D', 'E', 'F', 'G'};
uint8_t exclude[7] = {
  0b00000000, // a
  0b00000000, // b
  0b00000000, // c
  0b00000000, // d
  0b11000000, // e  (TX and RX pins are used for Serial and flashing so these have to be working)
  0b00000000, // f
  0b11000000  // g  (RD and WR pins are used to signal a read or write instruction to an external memory chip and are pulled low if unused)
};

void loop() {
  uint8_t mask, state, error = 0;
  for (uint8_t i = 0; i < 7; i++) {
    mask = 0xff ^ exclude[i];
    _SFR_IO8(3 * i + 1)     = mask;    // set all included pins as output
    _SFR_IO8(3 * i + 2)     = mask;    // set all included pins high
    _SFR_IO8(3 * i + 1)     = ~mask;   // set all included pins as input
    state = _SFR_IO8(3 * i) & mask;    // check all included pin states
    if (state != mask) {
      Serial.print("ERROR IN PORT_");
      Serial.print(port_names[i]);
      Serial.print(" PIN ERROR: ");
      Serial.print(state ^ mask, BIN);
      Serial.print("\n");
      error = 1;
    }
  }
  if (!error) { Serial.print("NO ERRORS DETECTED!\n"); }
  while (1) {}  // end program
}
```

Output Serial Monitor ×

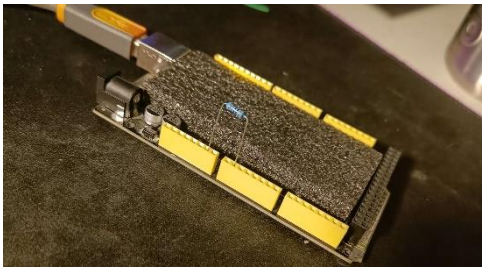
Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'COM3')

```
NO ERRORS DETECTED!
```

Deze code zet alle pins die niet in de exclude array staan op high en checkt of deze ook daadwerkelijk op high zijn gegaan door eerst de pin als output te configureren. Dit word gedaan door bits in de DDR registers high te zetten deze DDR registers bevinden zich op IO_BASE + 1 + 3n. Vervolgens worden deze pins op high gezet door bits in de PORT register high te zetten deze PORT registers bevinden zich op IO_BASE + 2 + 3n. Dan worden deze pins als input geconfigureerd. Ten slotte word de status afgelezen van de PIN register in IO_BASE + 3n. Als de bits in de PIN registers niet hetzelfde zijn als in de PORT registers word er een bericht gestuurd naar de serial console.

**Resultaat:**

```
NO ERRORS DETECTED!
```



Door pin A1 (F1) laag te pullen met een 1k resistor word het volgende bericht gestuurd:

```
ERROR IN PORT_F AT PIN: 10
```

Error bij pin 2 in port F (er word geteld van 0) dus dat klopt met A1 (F1).

# 7 Non-Blocking blinking led

**Challenge Beschrijving**:
write a program that blinks one LED x times each second and another LED y times each second. x and y can have any value > 0

**Aanpak:**
Ik heb twee timers opgezet om de lampjes om de zoveel tijd aan en uit te zetten

**Uitleg:**

```
void update_led_a() {
  if (millis() - last_led_a_pulse > (1000 / LED_A_FREQ)) {
    last_led_a_pulse = millis();
    PORTE ^= (1 << 4);
  }
}

void update_led_b() {
  if (millis() - last_led_b_pulse > (1000 / LED_B_FREQ)) {
    last_led_b_pulse = millis();
    PORTG ^= (1 << 5);
  }
}
```

In dit geval is LED_A_FREQ 2 en LED_B_FREQ 5 (in Hz). Ook hier worden de pins getoggled via de PORT registers.

Als meerdere leds wil blinken zou ik een 2d array maken waarin telkens led_pin, freq, last_pulse in staat waardoor alle leds via één functie geüpdatet kunnen worden door de index door te geven. Deze functie kan dan via een loop gecalled worden. Dit heb ik in dit geval niet gedaan omdat ik maar twee leds heb

**Resultaat:**

https://youtube.com/shorts/ZkLfpoYSBno

# 8 Non-Blocking debounce + Button function

**Challenge Beschrijving**:
Using the knowledge acquired about millis(), have a look at https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce and test this using one button and a LED toggle.

When this works, create your own button function so you can handle buttons easily the rest of your embedded career!

Test this function with at least two buttons!

**Aanpak:**
Ik heb een class gemaakt voor de buttons waarin alle variables voor de debounce zitten ook heb ik een function pointer toegevoegd waardoor ik een function kan callen als de state switcht hierdoor is het switchen van de leds heel makkelijk en hoef je alleen maar de buttons te updaten in de loop.

**Test resultaat:**

https://youtube.com/shorts/-94sOB9wIrg

**Uitleg:**

Voor het debounce circuit heb ik transistors gebruikt omdat ik maar een werkende button heb voor de rest werkt alles hetzelfde.

```
class Button {
private:
  uint8_t pin;
  uint8_t debounce_delay;
  uint8_t debounce_time;
  uint8_t debounce_state;
  toggle_function_t func;

public:
  Button(uint8_t pin, uint8_t debounce_delay, toggle_function_t func = nullptr) {
    this->pin = pin; pinMode(pin, INPUT);
    this->debounce_delay = debounce_delay;
    this->debounce_state = LOW;
    this->debounce_time = LOW;
    this->func = func;
    this->state = LOW;
  }

  void check() {  // only set state high if two checks in a row are high
  uint8_t reading = digitalRead(this->pin);
    if (reading != this->debounce_state) { this->debounce_time = millis(); }  // skip if last button state is different then now
    if (millis() - this->debounce_time > this->debounce_delay) {
      this->debounce_time = millis();
      if (this->state != reading) {
        this->state = reading;  // set state and run attached funtion when changing value
        if (this->func) { this->func(this->state); }
      }
    }
    this->debounce_state = digitalRead(this->pin);
  }

  operator bool() { return this->state; }

  uint8_t state;
};
```

Dit is de code snippet van de button class  deze class houd elke variable voor het debouncen en een function pointer die gecalled word als de state veranderd.

```
typedef void (*toggle_function_t)(uint8_t);
```

In deze code snippet word de function pointer type defined voor de button class

```
Button btn_a(BTN_A_PIN, DEBOUNCE_DELAY, &toggle_led_a);
Button btn_b(BTN_B_PIN, DEBOUNCE_DELAY, &toggle_led_b);
```

Hier worden de buttons geinializeerd met de pin number delay en functie die gecalled word als de state veranderd

```
void loop() {
  btn_a.check();
  btn_b.check();
}

// only trigger on rising edge
void toggle_led_a(uint8_t button_state) { if (!button_state) { return; } PORTE ^= (1 << 4); }
void toggle_led_b(uint8_t button_state) { if (!button_state) { return; } PORTG ^= (1 << 5); }
```

In de loop worden de buttons geupdated en de toggle functies defined. Hier zie je ook dat de leds alleen getoggled worden als de button_state high is en word dus alleen uitgevoerd op rising edge.

# 9 Traffic control

**Challenge Beschrijving**:
Create a traffic control program using state machine programming!

**Aanpak:**
Ik heb een class gemaakt om de status van elk verkeerslicht bij houd daarna heb ik een functie gemaakt die de states van de verkeerslichten switcht. Deze functie word dan aan een button vast gemaakt. Ook worden de states om de 10 seconden automatish geswitched

**Resultaat:**

https://youtube.com/shorts/K1wBo79kHvg

**Uitleg:**

Voor de trafic control code heb ik dezelfde button class als bij de vorige challenge gebruikt samen met een nieuwe trafic light class die alle variables en functies voor kleuren veranderen bevat. Het oranje licht duurt 3.5 seconden en het automatiche switchen duurt 10 seconden. (in de code staat overal “Yellow” omdat ik alleen maar gele lampjes had)

```
enum Trafic_State : uint8_t {
  GREEN =   0x0,
  YELLOW =  0x1,
  RED =     0x2,
  NONE =    0x3,  // blinking yellow
};

class Trafic_Light {
private:
  uint8_t   pins[3];

  uint32_t  transition_delay;
  uint32_t  transition_start;
  uint8_t   transition_state;
  uint32_t  pulse_delay;
  uint32_t  last_pulse = 0;

  uint8_t state;
```

In dit code snippet zie je een enum met alle mogelijke states en alle variables van de class. De waardes van de states in de enum zijn ook gelijk de index van de pin in de array pins van de Trafic_Light class. Dit geld alleen niet voor de NONE state omdat het knipperen van oranje meer code nodig heeft.

```
void set_state(uint8_t state) {
  this->state = state;
  if (state == Trafic_State::NONE) { state = Trafic_State::YELLOW; }  // set to yellow on none state
  for (uint8_t i = 0; i < 3; i++) {
    if (i == state) { digitalWrite(this->pins[i], HIGH); continue; }
    digitalWrite(this->pins[i], LOW);
  }
}
```

Hier word de state en dus ook de pins aangepast. Hier zie je ook dat de NONE state word omgezet in oranje.

```
void start_state_transition(uint8_t to_state, uint32_t transition_delay = TRAFIC_TRANSITION_DELAY) {
  if (to_state == this->state) { return; }
  if (this->state == Trafic_State::GREEN && to_state == Trafic_State::RED) { set_state(Trafic_State::YELLOW); }
  this->transition_delay = this->state == Trafic_State::NONE ? 0 : transition_delay;  // omit delay when changing from NONE state
  this->transition_state = to_state;
  this->transition_start = millis();
}
```

Hier worden alle variables voor transitie geïnstalleerd.

```
void update() {
  if (millis() - this->last_pulse > this->pulse_delay && this->state == Trafic_State::NONE) {
    this->last_pulse = millis();
    uint8_t pin = this->pins[Trafic_State::YELLOW];
    digitalWrite(pin, !digitalRead(pin));
  }
  if (this->transition_state != this->state && millis() - this-> transition_start > this->transition_delay) {
    set_state(this->transition_state);
  }
}
```

En hier word gecheckt of er iets veranderd moet worden.

```
void set_trafic_light_a(uint8_t sensor_state) {
  if (!sensor_state) { return; }
  last_transition = millis();
  trafic_light_a.start_state_transition(Trafic_State::GREEN);
  trafic_light_b.start_state_transition(Trafic_State::RED);
}
void set_trafic_light_b(uint8_t sensor_state) {
  if (!sensor_state) { return; }
  last_transition = millis();
  trafic_light_a.start_state_transition(Trafic_State::RED);
  trafic_light_b.start_state_transition(Trafic_State::GREEN);
}
```

Deze functies worden gecalled als de buttons (sensors) geactiveerd worden.

```
void loop() {
  trafic_light_a.update();
  trafic_light_b.update();
  sensor_a.check();
  sensor_b.check();

  // auto toggleing trafic lights
  if (millis() - last_transition > TRAFIC_AUTO_TRANSITION_DELAY) {
    uint8_t state_a = trafic_light_a.get_state();
    uint8_t state_b = trafic_light_b.get_state();
    if (state_a == state_b) { return; }  // filter valid states
    if ((state_a == Trafic_State::GREEN || state_a == Trafic_State::RED) && \
        (state_b == Trafic_State::GREEN || state_b == Trafic_State::RED)) {
      trafic_light_a.start_state_transition(state_b);
      trafic_light_b.start_state_transition(state_a);
      last_transition = millis() + max(trafic_light_a.get_transition_delay(), trafic_light_b.get_transition_delay());
    }
  }
}
```

En ten slotte word er in de loop functie alle objects geupdated en gechecked of er een transitie nodig is.

# 10 IO Techniques

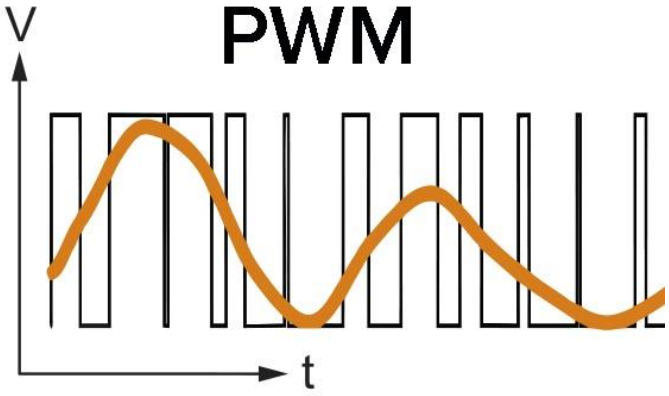
**Challenge Beschrijving**:
For each of the mentioned techniques above, find a source which clearly explains its principle.

For each of the mentioned techniques above, find some sensors or actuators using this technique.

**Uitleg:**

## 10.1 Digital:

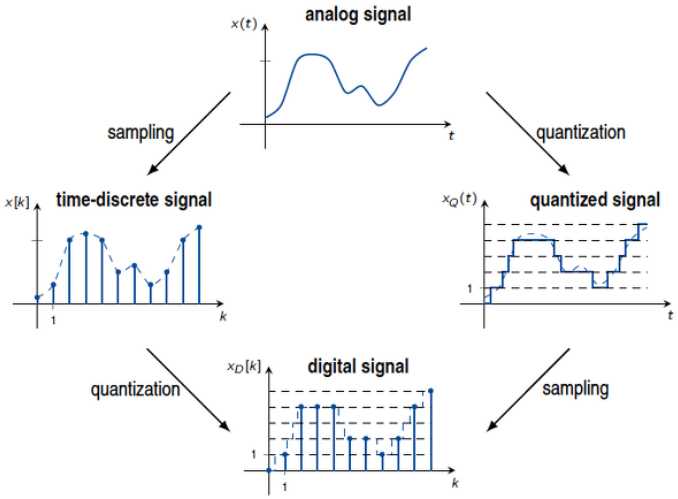
Digital IO is de simpelste techniek en word ook het meest gebruikt met sensors omdat dit (bijna) geen processing nodig heeft en het snelste is. Het enige nadeel is dat het alleen een ON en OFF signaal kan sturen. Er bestaat wek een soort mix tussen Digital en Analoog en dat is PWM (pulse width modulation) met deze techniek kan je meer data versturen door een enkele pin door het signaal voor een gedeeltes van een tijdsperiode aan en uit te laten staan. Door het signaal voor een gedeelte aan of uit te laten staan kan je een soort analoog signaal nadoen (dit is geen echt analoog signaal daarom is het een mix tussen digital en analoog).




hier zie je ook welk analoge signaal dit PWM signaal benadert

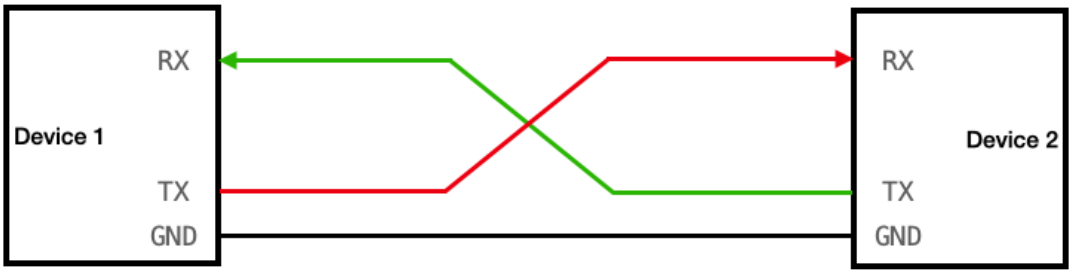
## 10.2 Analoog:

Een analoog signaal is een signaal met een bepaalde voltage tussen bijvoorbeeld 0v en 5v of 0v en 3.3v. deze voltage kan dan afgelezen worden doormiddel van de ADC peripheral die op meeste microcontrollers zit door de voltage range op te delen en te kijken waar de voltage ligt op deze verdeling.

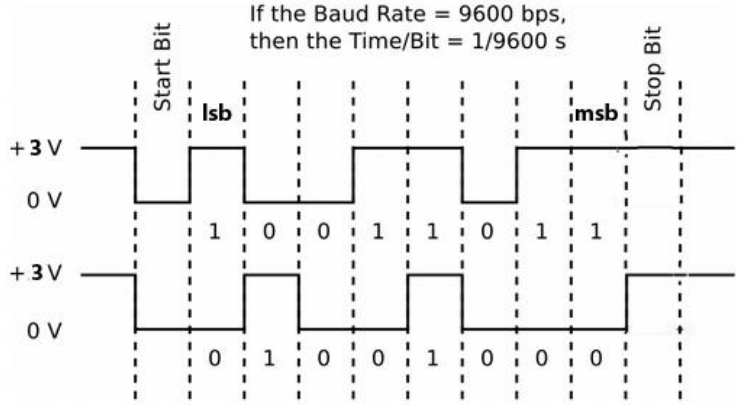



## 10.3 UART / USART:

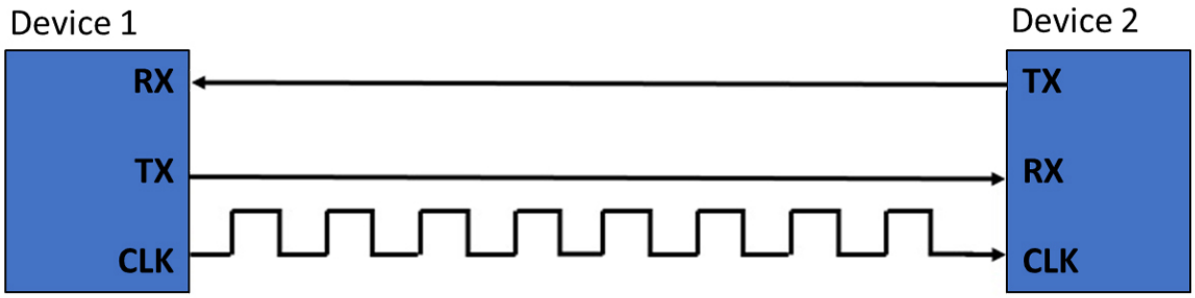
UART (universal asynchronous receiver-transmitter) / USART (universal synchronous-asynchronous receiver-transmitter) is een full-duplex data protocol dat betekend dat deze data tegeleikertijd heen en weer kan sturen. Het sturen van deze data wordt bit voor bit via twee of drie draaden gedaan: TX, RX en CLK bij USART. De TX en RX draden moeten omgedraaid worden tussen devices omdat deze bij elk device op de zelfde manier gebruikt word namelijk: de TX voor het versturen van data (transmit) en RX om de data te ontvangen (receive). RX en TX zijn high als ze idle zijn.




De bits worden op een afgesproken snelheid verstuurd en ontvangen. Deze snelheid moet tussen beide devices gelijk zijn omdat omdat anders de bits op het verkeerde moment opgenomen worden deze afgesproken waarde heet de baud rate (of bit rate).



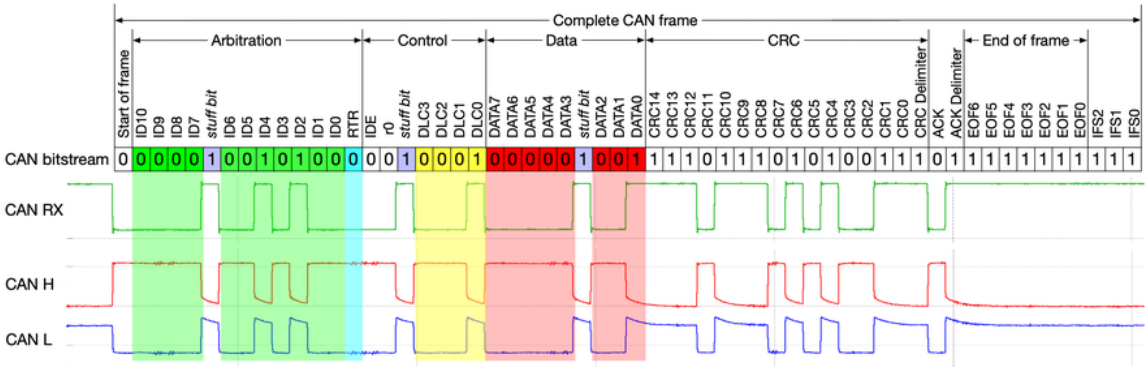

In het geval van USART is een gedeelde clock (CLK draad) toegevoegd waardoor de baud rate niet perse afgesproken hoeft te worden maar door een van de devices (meestal de master) geregeld te worden. Dit zorgt er voor dat er minder fouten gemaakt worden met het opnemen van de bits ook kan de clock uitgerekt worden als een van de microcontrollers het stuur moment niet haalt. Omdat deze clock line voor stabiliteit zorgt kan USART sneller werken dan UART terwijl deze betrouwbaar genoeg blijft.




Er zijn nog een paar andere afsplitsingen van het UART/USART protocol zoals bijvoorbeeld RS-232 maar dit wordt bijna niet meer gebruikt dus dit ga ik niet uitleggen.

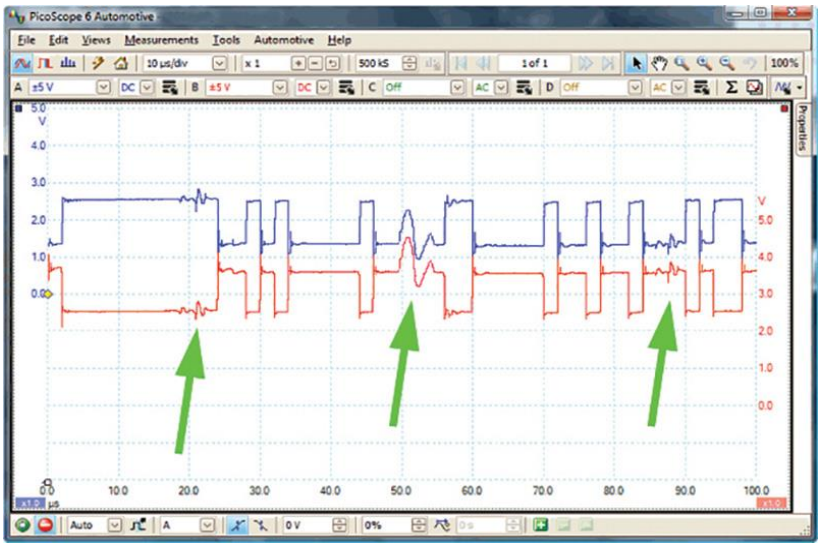
## 10.4 CAN:

Een CAN (controller area network) bus lijkt heel erg op UART omdat dit ook een asynchrone seriele protocol is. Het grootste verschil is dat CAN een high en low data pin heeft later wordt uitgelegd waarom. Ook is het half-duplex dat betekend dus dat de data maar een kant op kan per keer. Omdat er geen clock pin is moet de baud rate weer afgesproken worden.



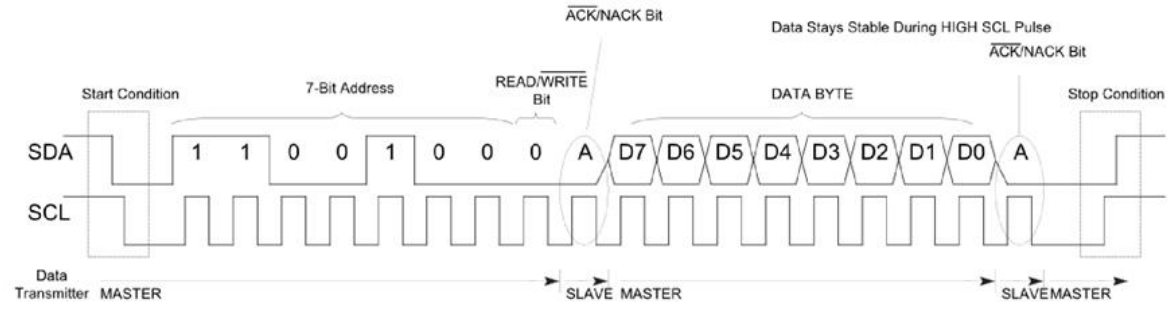

Doordat er data low en high pins zijn kunnen errors in het signaal makkelijk gedetecteerd en ontdaan worden door een van de signalen om te draaien en het gemmidelde uit te rekenen. Deze methode werkt omdat de meeste errors in signalen veroorzaakt worden door inductie

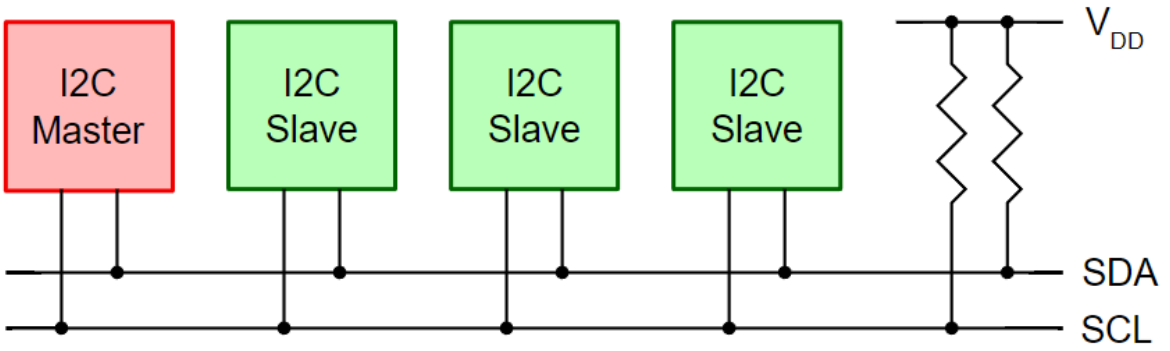
(waarmee je je telefoon drradloos kan opladen) bij inductie wordt energie draadloos overgedragen door wisselende stroom in een andere draad. Hierdoor wordt dezelfde error op de low en high uitgeoefend omdat deze nu geen overgestelde van elkaar zijn wordt de error bij het omdraaien en optellen uit het signaal gefilterd.




## 10.5 I2C:

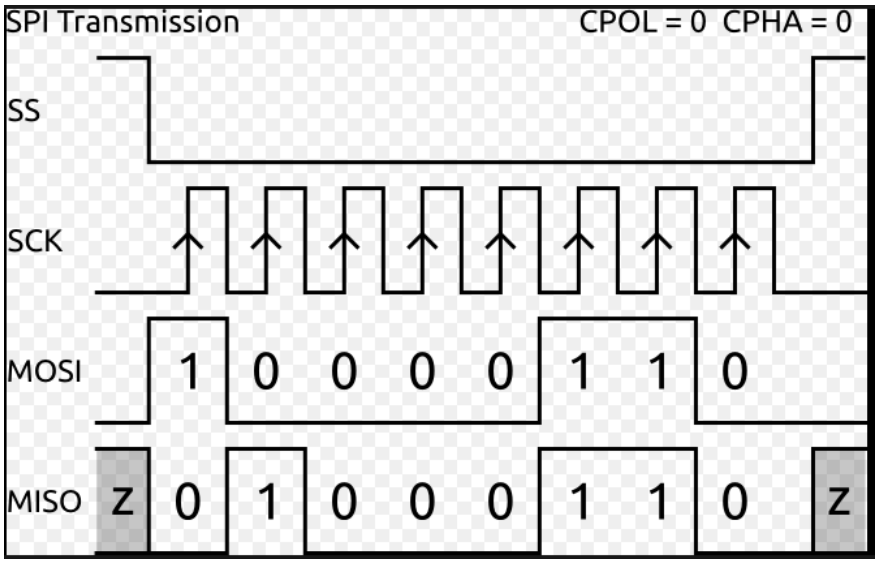
De I2C (inter intergrated circuit) bus lijkt heel erg op de USART omdat ze beide synchronous serial protcols zijn. Het vershil is dat I2C half-duplex is en dat elke transmissie begint met een device adres en een R/nW bit zodat dezelfde I2C bus voor meerdere devices gebruikt kan worden terwijl de per device communicatie behouden kan worden. De I2C bus bestaat uit 2 draaden SCL (serial clock) en SDA (serial data) SCL en SDA zijn high als ze idle zijn. Verder moet na elke byte een acknolagement bit komen van het ontvangende device.

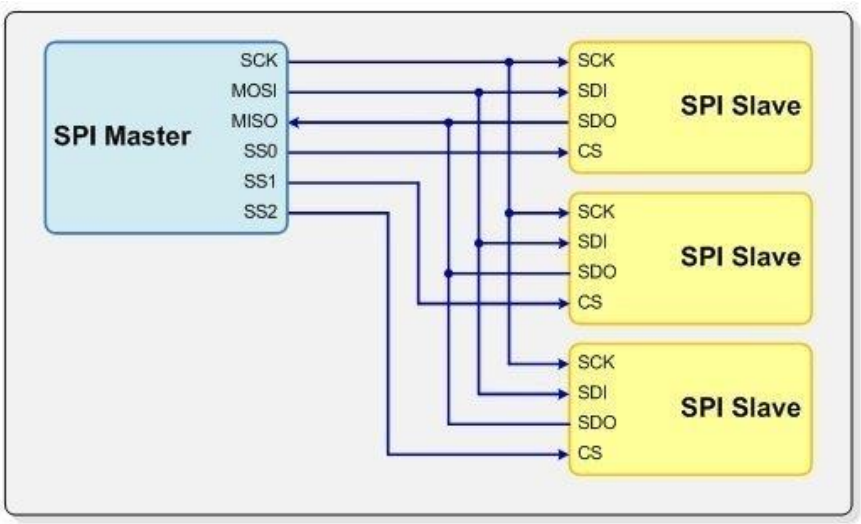


## 10.6 SPI:

SPI is een soort van full-duplex versie van I2C het enige verschil is dat er geen device adressen en R/nW bits gestuurd worden want in plaats van een device adres wordt de CS/SS (Chip Select / Slave Select) pin laag gezet waardoor de chip geselecteerd word. Voor de rest werkt het sturen van data over de twee data pins: MISO (master in slave out) en MOSI (master out slave in) hetzelfde als bij I2C. De data en clock pins zijn voor elke chip gedeeld maar de CS of SS-pin is bij elke chip apart.

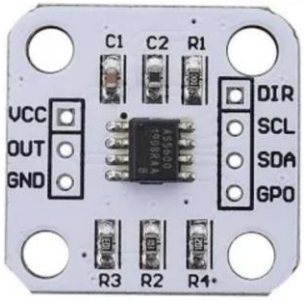


**Sensors:**

## 10.7 AS5600:

De AS5600 is een magnetische HAL angle sensor die geconfigureerd kan worden via I2C en afgelezen kan worden via I2C of via een Analoog signaal als er geen pullup resistor connected is (R4 op het breakoutboard)




Deze sensor moet geïnstalleerd worden (als er geen setting ingebrand is) dit moet gedaan worden via I2C door de bits in verschillende register aan en uit te zetten dit is waarom hiervoor meestal een library wordt gebruikt jammer genoeg was er geen library voor de STM32F4 boards dus die heb ik gemaakt op basis van de library van de arduino.

Library link: https://github.com/MarijnVerschuren/STM32F4_AS5600_library

Datasheet: https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf

**Library uitleg:**

De AS5600 heeft de volgende registers:

| Address | Name | R/W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Configuration Registers** (1), (2) | | | | | | | | | | |
| 0x00 | ZMCO | R | | | | | | | ZMCO(1:0) | |
| 0x01 | ZPOS | R/W/P | | | | | ZPOS(11:8) | | | |
| 0x02 | | | ZPOS(7:0) | | | | | | | |
| 0x03 | MPOS | R/W/P | | | | | MPOS(11:8) | | | |
| 0x04 | | | MPOS(7:0) | | | | | | | |
| 0x05 | MANG | R/W/P | | | | | MANG(11:8) | | | |
| 0x06 | | | MANG(7:0) | | | | | | | |
| 0x07 | CONF | R/W/P | | | WD | FTH(2:0) | | | SF(1:0) | |
| 0x08 | | | PWMF(1:0) | | OUTS(1:0) | | HYST(1:0) | | PM(1:0) | |
| **Output Registers** | | | | | | | | | | |
| 0x0C | RAW ANGLE | R | | | | | RAW ANGLE(11:8) | | | |
| 0x0D | | R | RAW ANGLE(7:0) | | | | | | | |
| 0x0E | ANGLE | R | | | | | ANGLE(11:8) | | | |
| 0x0F | | R | ANGLE(7:0) | | | | | | | |
| **Status Registers** | | | | | | | | | | |
| 0x0B | STATUS | R | | | MD | ML | MH | | | |
| 0x1A | AGC | R | AGC(7:0) | | | | | | | |
| 0x1B | MAGNITUDE | R | | | | | MAGNITUDE (11:8) | | | |
| 0x1C | | R | MAGNITUDE(7:0) | | | | | | | |
| **Burn Commands** | | | | | | | | | | |
| 0xFF | BURN | W | Burn_Angle = 0x80; Burn_Setting = 0x40 | | | | | | | |

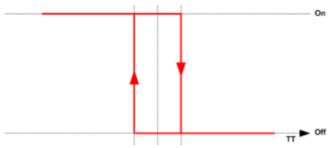
ZPOS bevat de start positie en MPOS de eind positie, MANG bevat de max angle deze registers worden gebruikt om een angle range in de chip te branden (het aantal burn events wordt opgeslagen in ZMCO)

Hierna komt de CONF-registers hierin worden de volgende settings opgeslagen:

| Name | Bit Position | Description |
|---|---|---|
| PM(1:0) | 1:0 | Power Mode<br>00 = NOM, 01 = LPM1, 10 = LPM2, 11 = LPM3 |
| HYST(1:0) | 3:2 | Hysteresis<br>00 = OFF, 01 = 1 LSB, 10 = 2 LSBs, 11 = 3 LSBs |
| OUTS(1:0) | 5:4 | Output Stage<br>00 = analog (full range from 0% to 100% between GND and VDD, 01 = analog (reduced range from 10% to 90% between GND and VDD, 10 = digital PWM |
| PWMF (1:0) | 7:6 | PWM Frequency<br>00 = 115 Hz; 01 = 230 Hz; 10 = 460 Hz; 11 = 920 Hz |
| SF(1:0) | 9:8 | Slow Filter<br>00 = 16x (1); 01 = 8x; 10 = 4x; 11 = 2x |
| FTH(2:0) | 12:10 | Fast Filter Threshold<br>000 = slow filter only, 001 = 6 LSBs, 010 = 7 LSBs, 011 = 9 LSBs, 100 = 18 LSBs, 101 = 21 LSBs, 110 = 24 LSBs, 111 = 10 LSBs |
| WD | 13 | Watchdog<br>0 = OFF, 1 = ON |

Powermode wordt gebruikt om de sensor op een lagere current te draaien dit kan een negatief effect hebben op de reading.

Hysteresis: bij hysteresis word de threshold wat opgeschoven (zoals in het volgende plaatje afgebeeld) om te voorkomen dat er meerdere rotaties geteld worden als de as bij 360 of 0 graden staat.



OUTS (zie foto)

PWMF is de PWM-frequentie die uitgezonden wordt als PWM-mode geselecteerd is in de OUTS register

SF en FTH zijn settings voor de measurement filter

WD zet de watchdog aan de watchdog is een functie die om de tijd checkt of de chip nog correct werkt

Nadat de configuratie naar de CONF-register geschreven is wordt de STATUS register afgelezen. Hierin staat de status van het magnetiche veld. Als dit klopt wordt een success ge returned

| Name | State When Bit Is High |
|---|---|
| MH | AGC minimum gain overflow, magnet too strong |
| ML | AGC maximum gain overflow, magnet too weak |
| MD | Magnet was detected |

**Library gebruiken:**

1. make a new instance

```
AS5600_TypeDef* sensor = AS5600_new();
```

2. set the i2c_handle, dir_port and dir_pin

```
sensor->i2c_handle = &hi2c1;
sensor->dir_port = dir_GPIO_Port;
sensor->dir_pin = dir_Pin;
```

In deze stap word er een nieuwe instance aangemaakt en bepaalde variables aangepast. Er zijn nog veel meer variables die aangepast kunnen worden maar deze worden in het voorbeeld op de standaard waarden gelaten.

3. initalize the sensor

```
AS5600_init(sensor);
```

Nu de settings in de instance staan worden deze naar de sensor geschreven met deze functie. Deze functie geeft ook een status terug om het programma te laten weten of het goed is gegaan of niet(daarom gebruik ik het meestal in een while loop totdat het succesvol returnt).

4. read using i2c

```
uint16_t angle;
AS5600_get_angle(sensor, &angle);
```

Nu de sensor gedigitaliseerd is kan deze afgelezen worden via I2C (waar het breakoutboard voor disigned is). Of via analog (werkt alleen als R4 van het breakoutboard is gehaald) met de volgende code:

```
// start receiving ADC data
HAL_ADC_Start_DMA(&hadc1, (uint32_t*)&state.raw_angle, 1);
```

Hier gebruik ik de DMA om het analoge signaal constant af te lezen en in stata.raw_angle te stoppen zonder dat ik de CPU bezig hoef te maken. Deze analoge data is niet altijd even goed dus hier is na het ontvangen nog wel wat werk aan.

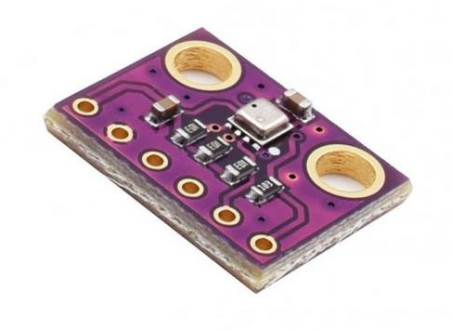
Resultaat:

https://www.youtube.com/watch?v=2IKLwBE1TLw

https://www.youtube.com/watch?v=rBEvl1Z75_w

## 10.8 BME280:

De BME280 is een temperatuur, vochtigheid en luchtdruk sensor die gebruikt kan worden met I2C en SPI



Deze sensor word net als de AS5600 vaak gebruikt met een library

Library link: https://github.com/eziya/STM32_HAL_BME280

Datasheet: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf

**Library gebruiken:**

```
bme280_dev dev;
```

Hier word een nieuwe instance aangemaakt waarin alle variables zitten

```
dev.dev_id = BME280_I2C_ADDR_PRIM;
dev.intf = BME280_I2C_INTF;
dev.read = user_i2c_read;
dev.write = user_i2c_write;
dev.delay_ms = user_delay_ms;
```

Hier word de device id en het data protocol ingesteld waardoor je de sensor kan gebruiken.

Ook worden er wat functie pointers doorgegeven dit zijn pointers voor:

- int8_t user_i2c_read(uint8_t id, uint8_t reg_addr, uint8_t *data, uint16_t len);
  voor het aflezen van de sensor via i2c
- int8_t user_i2c_write(uint8_t id, uint8_t reg_addr, uint8_t *data, uint16_t len);
  voor het schrijven naar de sensor via i2c
- void user_delay_ms(uint32_t period);
  voor timeout delays in ms

deze functies hebben dezelfde arguments en return types als je SPI gebruikt

```
bme280_init(&dev);
```

Nu dat alle communicatie variabelen en functies in de struct zitten kan de sensor geinitializeerd worden

```
dev.settings.osr_h = BME280_OVERSAMPLING_1X;
dev.settings.osr_p = BME280_OVERSAMPLING_16X;
dev.settings.osr_t = BME280_OVERSAMPLING_2X;
dev.settings.filter = BME280_FILTER_COEFF_16;
```

Hier worden de over sampling rates en filter ingesteld

```
bme280_set_sensor_settings(BME280_OSR_PRESS_SEL | BME280_OSR_TEMP_SEL | BME280_OSR_HUM_SEL |
BME280_FILTER_SEL, &dev);
```

Hier worden dan ten slotte de settings naar de sensor geschreven

```
bme280_data comp_data;
```

Nu de sensor is ingesteld kan er een variabele aangemaakt worden waar de aflezingen in terecht komen.

```
bme280_get_sensor_data(BME280_ALL, &comp_data, &dev);
```

Hier word de sensor afgelezen.

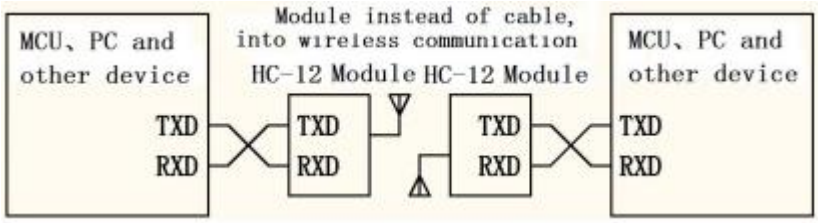
```
temperature = comp_data.temperature / 100.0;      /* °C  */
humidity = comp_data.humidity / 1024.0;           /* %   */
pressure = comp_data.pressure / 10000.0;          /* hPa */
```

De waarden die ontvangen zijn moeten dan alleen nog maar omgezet te worden.

## 10.9 HC-12:

De HC-12 is een lange afstand draadloze transmitter en receiver die aangestuurd moet worden met UART.

De HC-12 is supermakkelijk om te gebruiken op elk bord omdat het alleen maar data verstuurt en ontvangt via het uart protocol.

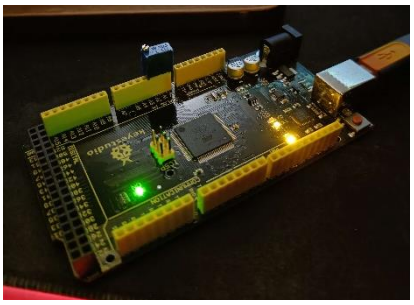



Hier zie je hoe je de HC-12 moet aansluiten en dit is eigenlijk hetzelfde als een uart connectie maar dan draadloos (tussen HC-12 modules)

# 11 Serial plotter

**Challenge Beschrijving**:
Connect a sensor with analog output and make a program to read the analog value. A nice tool to visualize an analog signal is available in the Arduino IDE, unfortunately not in PlatformIO. See https://arduinogetstarted.com/tutorials/arduino-serial-plotter for an explanation on the Serial Plotter.
Check if the values of the analog signal are as expected.

**Aanpak:**
ik heb een potmeter aangesloten op de analog pins van de arduino, de waarde afgelezen en vervolgens verstuurd naar de computer.



**Uitleg:**

```
void setup() {
  Serial.begin(115200);
  pinMode(A0, OUTPUT);
  digitalWrite(A0, LOW);
  pinMode(A2, OUTPUT);
  digitalWrite(A2, HIGH);
}
```

Hier word de Serial aangezet en de gnd / 5v pins voor de potmeter ingesteld.

```
void loop() {
  uint16_t raw = analogRead(A1);
  Serial.print("\nraw:");       Serial.print(raw);
  Serial.print(",percentage:"); Serial.print((double)raw / 10.24);
}
```

Hier word de analoge waarde afgelezen en doorgestuurd naar de computer.

**Resultaat:** (Beschrijving of screenshot (of beide))

https://youtu.be/pVsKFbXBHWw

# 12 LDR Serial Plotter:

**Challenge Beschrijving**:
At https://arduino-lessen.nl/les/lichtgevoelige-weerstand-ldr-uitlezen-met-arduino it is explained how to connect an LDR (Light Dependent Resistor) to an Arduino using a resistor. In this challenge you are going to read the sensor values and also determine what the best value for the resistor is. Connect an LDR and resistor as explained.

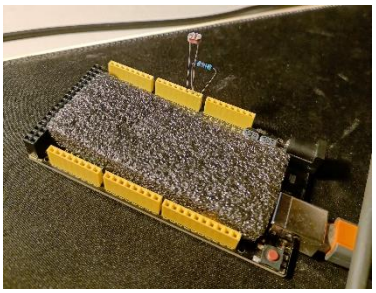
Read the sensor values and see how these change when covering / uncovering the LDR. Using the Serial Plotter might be helpful.

Vary the resistor value and see how this effects the sensor values you read.

The resistor value is optimal when the difference between the minimum sensor value and the maximum sensor value is as large as possible.

Use different resistors or use a potentiometer to determine the optimal resistor value.

**Aanpak:**
ik heb de code van de potmeter hergebruikt (zie vorige uitleg) en de resistor en LDR in series aangesloten net als de potmeter.



**Resultaat:** (Beschrijving of screenshot (of beide))

https://youtu.be/3oPQ4SvBKag

# 13 Running average / Hysteresis

**Challenge Beschrijving**:
Have a good look at https://www.arduino.cc/en/Tutorial/BuiltInExamples/Smoothing and use this example to create your own outdoor light controller!
Make sure the threshold level for switching the outdoor light on or off can be set easily!
Have a good look at https://www.aranacorp.com/en/implementation-of-a-measurement-hysteresis-on-arduino/ and make sure you understand the hysteresis concept.
Apply hysteresis to the outdoor lamp controller of the previous challenge.
The hysteresis thresholds should be easily adjustable.
Make sure you test thoroughly!

**Aanpak:**
Ik heb deze twee challenges uitgevoerd op de STM32F411CUE samen met de AS5600 sensor. Hysteresis word geregeld op de AS5600 hardware en de running average is toegevoegd voor het filteren van de analoge waardes van de AS5600.

**Resultaat:** (Beschrijving of screenshot (of beide))

https://youtube.com/shorts/V_E33Jzvdlo

**Uitleg:**

```
void euler_method() {  // typical execution time ~45 us
    register uint16_t raw = ACD_RANGE_CONV * (state.raw_angle - MIN_ADC_IN);
    register uint16_t pos_diff = (AS5600_pos_f64 - raw);  // rotation detection
    state.pos.rotation += pos_diff > 2048; state.pos.rotation -= pos_diff < -2048;
    register double alpha = 1 / ((EULER_TAU / TIM5->CNT) + 1);
    AS5600_pos_f64 = (raw * alpha) + ((1 - alpha) * AS5600_pos_f64);
    state.vel = (1e6 / TIM5->CNT) * ((uint16_t)AS5600_pos_f64 - state.pos.angle) * AS5600_RAD_CONV;  // rad / s
    state.pos.angle = (uint16_t)AS5600_pos_f64;
    TIM5->CNT = 0;
}
```

Hier word de running average uitgerekend van de waarde ontvangen van de AS5600 via de ADC. De waardes worden automatisch in de “state.raw_angle” geladen door de DMA, Timer 5 word gebruikt om de time delta bij te houden. Ook zie je dat alle variables die aangemaakt worden in de functie het keyword “register” hebben dit is goed voor snelhied omdat deze variables nooit op de ram (stack) terecht komen maar in de cpu geladen blijven wat een stuk sneller is. Verder word de volgende formule toegepast om de running average uit te rekenen.

$$y(t_0 + h) - y(t_0) = \int_{t_0}^{t_0+h} f\big(t, y(t)\big)\,\mathrm{d}t.$$

Hier zie je dat:

Let: t1 = t0 + h
Note:
f(a, b) is een functie waarbij de reactie snelheid (weight van nieuwe waardes) word ingesteld
(“alpha” in de code) omdat de functie in mijn code anders is genoteerd ga ik een definitie voor f(a, b) geven: $_{a0}^{a1}$int f(a, b) da = $_{a0}^{a1}$[c * a * x]  waar c een constant is en x de nieuwe waarde.
Note:
y(t) binnen de integraal is de huidige sensor waarde en is dus een constant (binnen een iteratie)
Dus:
y(t1) = y(t0) + $_{t0}^{t1}$[c * t * x]
y(t1) = y(t0) + x*c * (t1 – t0)  // x * c * dt  c * dt = “alpha” in de code

De euler functie word elke 100us via een timer interrupt gecalled (waarin ook uitgerekend word weke kant de stepper op moet draaien en hoe snel). zoals je ziet is de typische runtime van de euler functie ongeveer 45us, omdat hier wat extra dingen bijzitten ga ik er van uit dat de runtime van deze interrupt ongeveer 50us tot 75us is waardoor er 25us tot 50us over is voor de main loop.

```
while (1) {
    if (step_gain < MIN_STEPPER_GAIN) { continue; }  // make sure t
    // dir is set in interrupt
    register uint16_t pulse_delay = MIN_STEPPER_DELAY / step_gain;
    HAL_GPIO_WritePin(STEPPER_STP_GPIO_Port, STEPPER_STP_Pin, 1);
    delay_us(pulse_delay);
    HAL_GPIO_WritePin(STEPPER_STP_GPIO_Port, STEPPER_STP_Pin, 0);
    delay_us(pulse_delay);
}
```

Hier zie je dat er in de main loop vooral gewacht word (min 75us per delay) waardoor de timer interrupts bijna altijd in boven op de delay funtions vallen. Hierdoor klopt de delay misschien soms niet helemaal maar op de schaal van een paar us is dit niet echt te merken.

```
sensor = AS5600_new();
sensor->i2c_handle = &hi2c1;
sensor->dir_port = AS5600_DIR_GPIO_Port;
sensor->dir_pin = AS5600_DIR_Pin;
sensor->positive_rotation_direction = AS5600_DIR_CW;
sensor->hysteresis = AS5600_HYSTERESIS_3LSB;  // go 8 segments over 0 and 4096 before flip
```

Hier word de hysteresis ingesteld in de AS5600 config. De hysteresis word nu getriggered als de 3 laagste bits geflipped zijn (dus 8 over de threshold).

**Bronnen:**

Euler method: https://en.wikipedia.org/wiki/Euler_method

Hysteresis: https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf

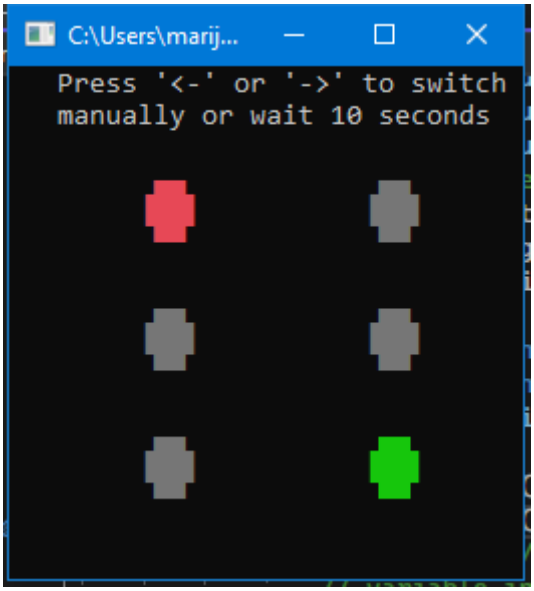
# 14 Traffic control C#

**Challenge Beschrijving**:
You will create a console project in C# with Visual studio .NET that represents a traffic light. Make sure that your program meets the following requirements:

- The project should contain multiple objects of type TrafficLight.
- A TrafficLight can have the colors (states) 'green', 'orange', 'red': use English names for these states.
- For safety, the status is always set to red when creating a TrafficLight.
- From red, the state can only turn to green, then to orange, then to red again.
- You can use a buttonpress to indicate that a car has arrived
- The light will be orange for 2 seconds and green for 5 seconds. Make sure that during the states your system can still react on events

**Aanpak:**
Ik heb de code van Traffic control Aangepast naar C# en een console ui toegevoegd in plaats van ledjes

**Resultaat:** (Beschrijving of screenshot (of beide))




Video: https://youtu.be/XA1cbmN7gv0

**Uitleg:**

Zie uitleg van Traffic control

# 15 Bank account C#

**Challenge Beschrijving**:
You will create a Windows Forms application in C# with Visual studio .NET that represents a (highly simplified) bank with two bank accounts. With this application one can: o Deposit money into the account on the left or right. Enter the amount in the TextBoxes, then click on the “Deposit” button. Only a positive amount can be deposited. If a negative balance would arise, or if something other than positive numbers are entered, the transaction is not executed and an error message is given by means of a MessageBox.
Withdraw money from the account on the left or right. Enter the amount in the TextBoxes, then click on the “Withdraw” button. If a negative balance would arise, or if something other than positive numbers is entered, the transaction is not executed and an error message is given by means of a MessageBox.
Transfer money from left to right (or right to left). In the TextBoxes on the left (right) you enter the amount to be transferred and then click on the Button >> (<<). If a negative balance would arise with the paying party, or if something other than positive numbers has been entered, the transaction will not be executed and an error message will be given by means of a MessageBox. The displayed balance is of course neatly adjusted after every transaction.

Show in this application the concepts that you learned about OOP.
A few mandatory requirements are:

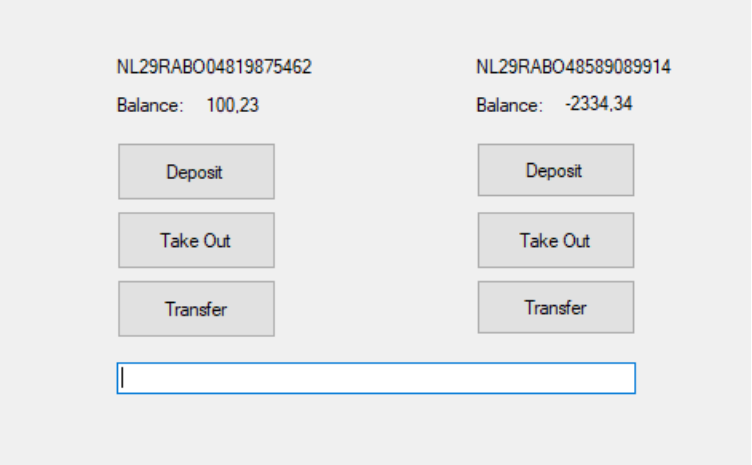
- UI code is separated from the other code
- No public fields
- Two objects must be made from one Bankaccount class
- The Bankaccount class implements the methods deposit, withdraw and Transfer to
- The bank account implements a constructor with several parameters
- Each account gets a unique number, you could make a AccountNumberGenerator class for that.

**Aanpak:**
Ik heb een class gemaakt voor de bank en de accounts daarin die elk hun eigen iban en saldo waardes hebben.

### Voorkennis:
Een class is een soort collectie data (fields) samen met functies die op deze data dingen kunnen uitvoeren, deze functies worden methods genoemd. In C# kan elke functie of variable een soort protectie krijgen dit zijn: public, protected, private. Bij public kan alles deze field of method gebruiken, bij protected kunnen alleen derived classes en in C++ ‘friendly’ functies en classes deze fields of methods gebruiken, private fields en methods kunnen alleen binnen de class gebruikt worden of in C++ alleen door ‘friendly’ functies of classes.

Properties zijn methods zonder arguments die gebruikt kunnen worden als fields maar eigenlijk een functie zijn. Met properties kan je een field bijvoorbeeld read only maken omdat er geen ‘set’ property is.

**Resultaat:** (Beschrijving of screenshot (of beide))




Video: https://youtu.be/6j-nymGyRJo

**Uitleg:**

```
internal class Account {
    2 usages
    public String iban {
        get;
        private set;
    }
    7 usages
    public Double balance {
        get;
        private set;
    }
    1 usage   MarijnVerschuren
    public Account(String iban, Double balance) {
        this.iban = iban;
        this.balance = balance;
    }

    MarijnVerschuren
    static public Account operator +(Account self, Double ammount) { self.balance = Math.Round(self.balance + ammount, 6); return self; }
    MarijnVerschuren
    static public Account operator -(Account self, Double ammount) { self.balance = Math.Round(self.balance - ammount, 6); return self; }
```

Hier zie je dat ik properties heb gebruikt voor ‘iban’ en ‘balance’ om er voor te zorgen dat se alleen vanuit binnen de class aangepast kunnen worden en overal afgelezen kunnen worden. Ook heb ik een constructor gemaakt waarmee je de iban en balance van een nieuw object kan instellen. Verder heb ik operators gebruikt om de balance aan te passen, hierbij gebruik ik ook de round functie om er voor te zorgen dat geen float errors ontstaan.

```
internal class Banking {
    private List<Account> accounts = new List<Account>();
    private Random rd = new Random();

    private long LongRandom(long min, long max) {
        long result = rd.Next((Int32)(min >> 32), (Int32)(max >> 32));
        result = (result << 32);
        result = result | (long)rd.Next((Int32)min, (Int32)max);
        return result;
    }


    private String generate_iban() {
        String number, iban;
        do {
            number = LongRandom(0, 9999999999).ToString();
            iban = "NL29RABO" + '0' * (10 - number.Length) + number;
        } while (get_account(iban) != null);
        return iban;
    }


    public Account get_account(String iban) {  // this is the slow method
        foreach (Account acc in accounts) {
            if (acc.get_iban() == iban) { return acc; }
        } return null;
    }

    public String add_account() {
        String iban = generate_iban();
        accounts.Add(new Account(iban, 0));
        return iban;
    }

    public bool deposit(String dst, Double ammount) {
        Account dst_acc = get_account(dst);
        if (dst_acc == null) { return false; }
        dst_acc += ammount; return true;
    }

    public bool take_out(String src, Double ammount) {
        Account src_acc = get_account(src);
        if (src_acc == null) { return false; }
        src_acc -= ammount; return true;
    }

    public bool transfer(String src, String dst, Double ammount) {
        Account src_acc = get_account(src);
        Account dst_acc = get_account(dst);
        if (src_acc == null || dst_acc == null) { return false; }
        src_acc -= ammount; dst_acc += ammount; return true;
    }
```

In de Banking class worden alle Accounts opgeslagen die samen met bepaalde methods die waardes in Account aanpassen ervoor zorgen dat de app werkt. Deze methods worden gecalled vanuit de form.

# 16 PWM LED

**Challenge Beschrijving**:
Connect a potentiometer and a LED to your Arduino.
Write a program such that you can control the brightness of the LED with the potentiometer.
Show your loop skills to create a light show using an RGB LED. Use the potentiometer and buttons to change the behavior of the light show, e.g. the pattern, speed, brigthness.

**Aanpak:**
Ik heb een RGB led aangesloten samen met een potmeter. Ook heb ik drie pins aangemaakt waarmee R, G en B geselecteerd kunnen worden.

**Resultaat:** (Beschrijving of screenshot (of beide))

https://youtube.com/shorts/PfYVs6wz8-8

# 17 Remote control

**Challenge Beschrijving**:
In the orientation phase you got acquainted with receiving a (serial) message from a PC.
If needed you can take a look at the following link again.
https://www.arduino.cc/reference/en/language/functions/communication/serial/read/
Take a look at the workshop *Serial communication with the Arduino* in the Toolbox section of this course
Now you will remote control a (red) LED connected to the Arduino from the PC using messages that meet an agreed format. When the Arduino receives the message "#SET_ LED_ON%" from the serial port, the LED will be turned on. Use the String type on the Arduino to save and compare a received message. For String usage, see http://arduino.cc/en/Reference/StringObject.
What are the '#' and '%' characters for?

**Aanpak:**
Ik heb gekozen voor een “program specific” protocol die werkt met individuele bytes in plaats van text wat een stuk makkelijker is en minder code nodig heeft. Text was voor deze applicatie ook niet nodig omdat dit systeem alleen gemaakt is voor het aansturen van ledjes en heeft daarom geen behoefte aan de mogelijke flexibiliteit die text brengt.
De bericht structuur is als volgt: id (1B), r (1B), g (1B), b (1B) voor een totaal van 4 bytes.
Verder word er vanaf de arduino aangegeven hoeveel ledjes er aangestuurd kunnen worden door een bericht (volgens aangegeven spec) te sturen met het maximale id in het id field.

Ik heb dit nog niet met het trafic control programma geintegreerd omdat ik maar een RGB led heb. Ook al kan dit met normale leds vind ik niet dat dit al te veel toevoegd aan de communicatie challenge.

**Resultaat:** (Beschrijving of screenshot (of beide))

https://youtu.be/hJSCzGoOK6E

**Uitleg:**

C#:

```
// since C# is cant really work with struc
byte[] msg = new byte[4];
```

Hier word de IO buffer aangemaakt

```
while (ser.BytesToRead < 4) {}
ser.Read(msg, 0, 4);

byte max_id = msg[0];  // id field in struct
```

Hier word gewacht totdat de arduino het maximale id doorstuurt

```
while (true) {
    Console.Write("id (0 - " + max_id.ToString() + "): "); msg[0] = Convert.ToByte(Console.ReadLine());
    Console.Write("R: "); msg[1] = Convert.ToByte(Console.ReadLine());
    Console.Write("G: "); msg[2] = Convert.ToByte(Console.ReadLine());
    Console.Write("B: "); msg[3] = Convert.ToByte(Console.ReadLine());
    ser.Write(msg, 0, 4);
}
```

En dit is de user input loop.

C++:

```
constexpr uint8_t led_count = 1;
constexpr uint8_t pins[1][4] = {
  {3, 2, 4, 5}  // gnd, r, g, b (of led 1)
};

struct {
  uint8_t id;
  uint8_t r;
  uint8_t g;
  uint8_t b;
} input;
```

Hier worden alle pin nummers opgeslagen van elk ledje en de structuur van de berichten aangegeven. Deze structuur vind je niet terug in C# omdat C# niet goed werkt met structs (niet compatible met conversions en de serial library) dit heb ik opgelost door er gewoon een uint8_t (byte in C#) array van te maken wat perfect over een komt met de structuur.

```
void setup() {
  for (uint8_t i = 0; i < led_count + 1; i++) {
    pinMode(pins[i][0], OUTPUT);
    digitalWrite(pins[i][0], LOW);
    pinMode(pins[i][1], OUTPUT);
    pinMode(pins[i][2], OUTPUT);
    pinMode(pins[i][3], OUTPUT);
  }

  Serial.begin(115200);

  while (Serial.available() < 4) {
    input.id = led_count - 1;  // max id
    Serial.write((char*)&input, 4);
    delay(200);  // send "handshake" 5x per second
  }
}
```

Hier worden alle pins ingesteld, de serial aangezet op het maximum wat stabiel is op de arduino mega 2560 en tenslotte de max id doorgestuurd naar de PC.

```
void loop() {
  if (Serial.available() > 3) { Serial.readBytes((char*)&input, 4); }
  if (!(input.id < led_count)) { return; }
  analogWrite(pins[input.id][1], input.r);
  analogWrite(pins[input.id][2], input.g);
  analogWrite(pins[input.id][3], input.b);
}
```

En dan is dit de main loop.

# 18 Reflectie / evaluatie

## 18.1 Waar ben ik trots op?

*De uitleg bij* IO Techniques

## 18.2 Wat doe ik een volgende keer anders?

*Python + C/C++ gebruiken als controller en C aan de embedded kant (STM32 boards)*

## 18.3 Welke formatieve indicatie zou ik mezelf geven voor de verdieping Technology?

| Leeruitkomst verdieping Technology (bouwt voort op de oriëntatie) | | |
|---|---|---|
| Onderdeel | Criterium | Rating |
| Interactieve embedded systemen | Je product kan communiceren met een ander systeem volgens een eigen gedefinieerd protocol inclusief parameters waarbij ongeldige berichten worden afgevangen. | *O, ik heb mijn eigen protocol gemaakt met error handleing (normaal zou ik er CRC aan toevoegen voor voorbeelden zie: https://github.com/MarijnVerschuren/Robotic_Arm Onder: ARM_COMPUTER en CONTROLLER)* |
| Programmeren | Je past alle imperatieve programmeer-concepten en de volgende OO concepten toe: objects, classes en encapsulation, d.w.z.: constructors, private fields, properties en methods. De focus hierbij is op leesbare (b.v. naamgeving, indentation) en onderhoudbare programma's en robuustheid van het product. | *O, ik kan prima overweg met OOP voor meer voorbeelden: https://github.com/MarijnVerschuren/Neural_Network https://github.com/MarijnVerschuren/Password_Manager* |
| Sensoren en Actuatoren | Je past extra sensoren en actuatoren toe waarvan een eigen analyse is gedaan. | *O, ik kan prima met een datasheet of andermans code overweg om te snappen hoe een bepaalde sensor werkt.* Voorbeelden: https://github.com/MarijnVerschuren/STM32F4_AS5600_library *https://github.com/MarijnVerschuren/Robotic_Arm* |
| Verschillende I/O | Naast de genoemde I/O technieken kun | *O, ik heb tot nu toe met elk van de genoemde* IO Techniques *gewerkt met uitzondering van de CAN bus* |

| | | |
|---|---|---|
| technieken | je ook pulsbreedtemodulatie en analoge input interpreteren en toepassen. | *Project met UART, SPI, I2C, Analog en Digital: https://github.com/MarijnVerschuren/Robotic_Arm (kijk onder STM32_MCU(subject to change) )* |