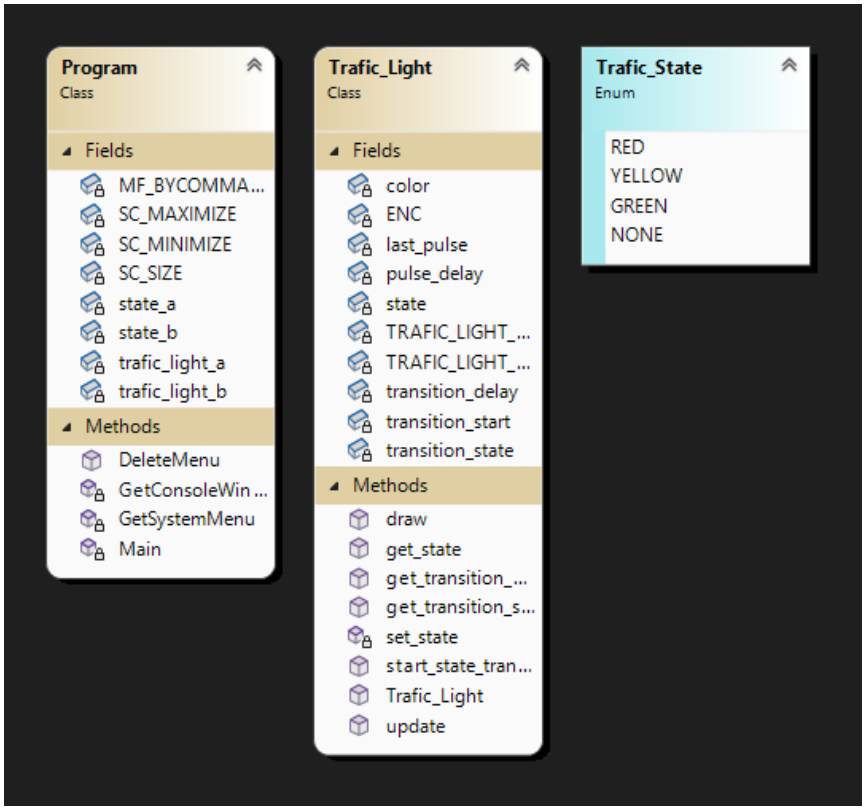
Contents

1. Orientation-oop-basics:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/lib/compile/src/hash.hpp

2. Orientation-on-class-diagrams:



Class diagram van traffic control programma van SEM1

3. Orientation-unit-testing:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/auto_test.py

4. Orientation-inheritance:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/lib/compile/src/hash.hpp

5. Orientation-abstract-classes-and-interfaces:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/lib/compile/src/hash.hpp

6. Exceptions:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/core.py

7. File-io:
   https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/core.py

8. Student-administration-recap: (TODO)

9. Bike-computer:
   https://github.com/MarijnVerschuren/STM32F4_AS5600_library/blob/main/AS5600

10. FIFO-buffers:

https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/ARM_COMPUTER/Core/Inc/uart_buffer.h

the uart is configured with a dma which has FIFO enabled (dit is geregeld in hardware dus ik heb hier zelf geen code voor (ik kan dit wel uitleggen)).

11. Array-basics:

../ SEM_1/TraficControlCS/Trafic_Light.cs@20

12. Dispatcher:

Dispatching through interrupts:

https://github.com/MarijnVerschuren/Robotic_Arm/tree/main/STM32_MCU/stepper

13. String-behaviour:

https://github.com/MarijnVerschuren/Password_Manager/blob/main/app/lib/compile/src/hash.hpp

Hier worden strings (of buffers + sizes) gebruikt om data in een hashing functie te krijgen. De lengte van een string hoeft hierbij niet gegeven te worden omdat elke string eindigd met een 0x00.

14. String-calculator:

```
const char* str = “text with unknown length…”;
uint32_t str_len = 0;
while (*str) { str_len++; str++; };
```

Code voor het berekenen van de lengte van de string.

15. Structs:

https://github.com/MarijnVerschuren/STM32F4_AS5600_library/blob/main/AS5600/inc/AS5600.h (@95)

Struct die de configuratie en i2c handle van een bepaalde sensor bijhoud

https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/ARM_COMPUTER/Core/Inc/main.h

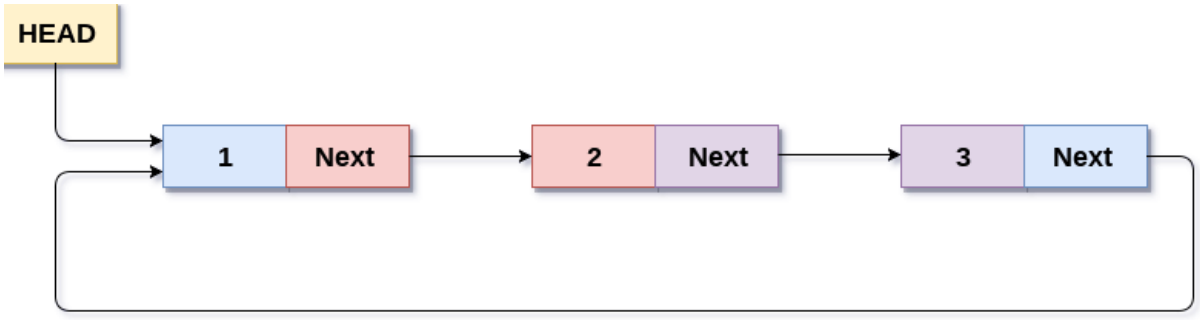
```
/* Instruction
 * motor instruction (includes flags)
 */
// 0xffffffffffffffff 0xffffffffffffffff 0xffffffffffffffff ((0x3, 0x4, 0x78, 0xff80) => 0xffff) 0xffff, 0xffff, 0xffff
typedef struct {  // uint8_t[32]
        double          target;             // rad
        double          max_vel;            // rad / s
        double          max_acc;            // rad / s^2
        uint16_t        micro_step: 2;  // microstep setting
        uint16_t        srd_mode: 1;    // srd mode on the motor controller
        uint16_t        action: 4;          // look in ACTION enum for possible actions
        uint16_t        dir: 2;             // 0 CLOSEST, 1 CW, 2 CCW, 3 LONGEST
        uint16_t        id: 7;              // selected motor
        uint16_t        instrution_id;  // instruction id
        uint16_t        _;                          // reserved uint8_t[2]
        uint16_t        crc;
} MCU_Instruction;
```

Struct van het bericht dat naar de ‘kleine stm32’ (STM32F411CEU6) word gestuurd om een beweging uit te voeren via de stepper motor

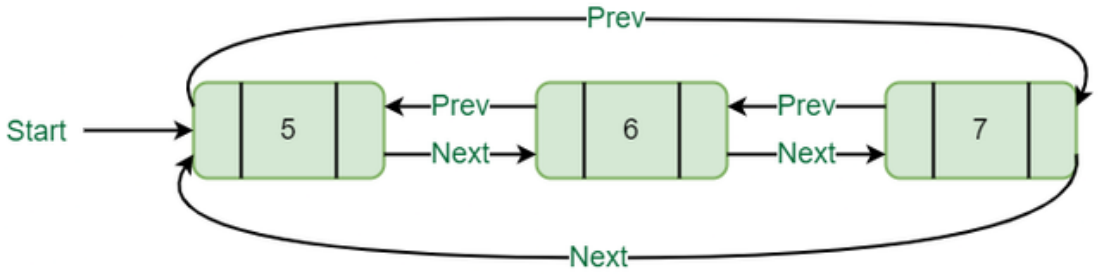
16. Connectfour: (TODO)

17. Ringbuffer:

Circular linked list:



Hiebij is ‘HEAD’ een pointer naar het eerste item en alle ‘NEXT’ pointers naar het volgende item. Soms word er ook een ‘PREV’ pointer toegevoegd waardoor je makkelijk heen en weer kan stappen in de list (doubly circular linked list):



18. Studentadmin: (TODO)

19. Buffer-unit-tests:

Pointer casting:

```cpp
#include <stdint.h>
#include <iostream>
int main(int argc, char** argv) {
    uint8_t* buf = new uint8_t[8](0, 1, 2, 3, 4, 5, 6, 7);

    /* as uint8_t (casting to uint16_t to prevent printing
     * chars instead of their numerical value) */
    std::cout << "as u8:\t[" << (uint16_t)buf[0];
    for (uint8_t i = 1; i < 8; i++) {
        std::cout << ", " << (uint16_t)buf[i];
    } std::cout << "]\n";
    /* as uint16_t */
    std::cout << "as u16:\t[" << ((uint16_t*)buf)[0];
    for (uint8_t i = 1; i < 4; i++) {
        std::cout << ", " << ((uint16_t*)buf)[i];
    } std::cout << "]\n";
    /* as uint32_t */
    std::cout << "as u32:\t[" << ((uint32_t*)buf)[0];
    std::cout << ", " << ((uint32_t*)buf)[1] << "]\n";
    /* as uint64_t */
    std::cout << "as u64:\t[" << *((uint64_t*)buf) << "]\n";
    /* as float */
    std::cout << "as f32:\t[" << ((float*)buf)[0];
    std::cout << ", " << ((float*)buf)[1] << "]\n";
    /* as double */
    std::cout << "as f64:\t[" << *((double*)buf) << "]\n";
}
```

Output:

```
as u8:  [0, 1, 2, 3, 4, 5, 6, 7]
as u16: [256, 770, 1284, 1798]
as u32: [50462976, 117835012]
as u64: [506097522914230528]
as f32: [3.82047e-37, 1.00825e-34]
as f64: [7.94993e-275]
```

20. Memory:

../SEM_1/ArduinoTester/ArduinoTester.ino@23

De '_SFR_IO8' is een functie die een pointer terug geeft ten opzichten van een 'BASE' pointer (__SFR_OFFSET) waar de registers van de gpio pheripheral geindexeerd zijn.

```
#define _SFR_IO8(io_addr) _MMIO_BYTE((io_addr) + __SFR_OFFSET)
```

Dan bevind elk 'PIN' (digital input register) register op: 'BASE' + 3 * n

Dan bevind elk 'DDR' (data direction register) (0=in/1=out) register op: 'BASE' + 3 * n + 1

Dan bevind elk 'PORT' (digital output register) register op: 'BASE' + 3 * n + 2

Waarbij 'n' de port index is bij mijn arduino mega 2560 van 0 tot 6 (A tot G)

21. Pointers-intro:

Zie Memory:

22. Bit-manipulations:

Zie Memory:

23. Watch:

Bij de stm kan je de 'CNT' of wel count register van een timer direct aflezen zie: https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/STM32_MCU/stepper/Core/Src/main.c @53

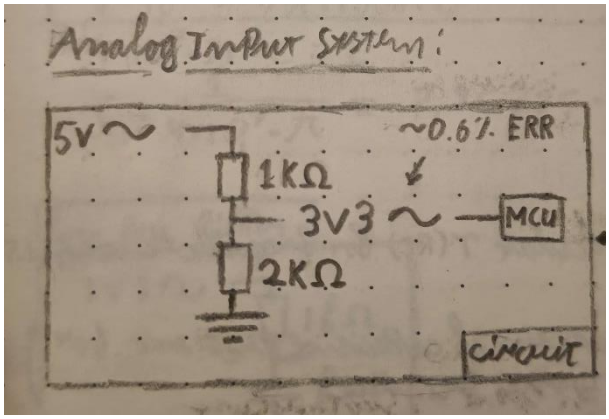
Waar ik wacht tot de count register op 'n' staat deze timer is zo geconfigureerd dat deze count eens per micro seconden geincrementeerd word hierbij zou ik ook bitmanipulatie gebruiken om bijvoorbeeld te wachten tot het derde bit gezet word, Dit zou voor een periode van 4us zorgen.

24. Adidas:

Een andere vorm van data correction is een CRC checksum dit word bijvoorbeeld gebruikt in mijn robotic arm bij bijna elke data transfer zie: https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/STM32_MCU/stepper/Core/Src/main.c @133

Waar dit gebruikt word om te verifiëren of de data correct is aangekomen.

25. Elektrische-basis:



Dit is de voltage devider die ik gebruik om het 5v analoge signaal naar 3.3v om te zetten zodat de stm32 ze kan aflezen (de weerstanden zijn zo laag omdat dat alles was wat ik had).

26. Serialization:

https://github.com/MarijnVerschuren/Info_Generator/blob/main/info_gen/gen.py

27. I2C:

https://github.com/MarijnVerschuren/STM32F4_AS5600_library/blob/main/AS5600/src/AS5600.c

28. Alarm-states: (TODO)

29. Light-controller:

../SEM_1/*TraficControl*

30. Trafic-light:

../ SEM_1/*TraficControl*

31. Analog:

https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/STM32_MCU/stepper/Core/Src/main.c

32. Hysteresis:

https://github.com/MarijnVerschuren/Robotic_Arm/blob/main/STM32_MCU/stepper/Core/Src/main.c

33. Register-IO:

../SEM_1/ArduinoTester/ArduinoTester.ino@23

Pointer offsets van registers zie Memory: voor uitleg