Arduino Assignments

In dit document is een overzicht te vinden van alle opgaven van de Hogeschool Utrecht, waarbij tot nu toe de Arduino Due gebruikt wordt.

Inhoud

[Jaar 1 – TICT-V1OOPC-15 3](#_Toc526164393)

[Week 4 4](#_Toc526164394)

[4.1. Materiaal 4](#_Toc526164395)

[4.2. Installatie 4](#_Toc526164396)

[4.3. Patroon 4](#_Toc526164397)

[4.4. Input 4](#_Toc526164398)

[Week 5 5](#_Toc526164399)

[5.1. HC595 5](#_Toc526164400)

[5.2. Decorator 5](#_Toc526164401)

[5.3. Knipperen 5](#_Toc526164402)

[Week 6 6](#_Toc526164403)

[6.1. OLED 6](#_Toc526164404)

[6.2. De code inleveren 6](#_Toc526164405)

[Jaar 1 - Inter Disiplinair Project 7](#_Toc526164406)

[Jaar 1 - TICT-V1IPASS-18 8](#_Toc526164407)

[Jaar 2 – TICT-V2CPSE1-16 9](#_Toc526164408)

[Week 1: ringtone (C++) 10](#_Toc526164409)

[1.1. UML 10](#_Toc526164410)

[1.2. Opbouwen 10](#_Toc526164411)

[1.3. Noten printen 10](#_Toc526164412)

[1.4. RTTTL string converter 10](#_Toc526164413)

[1.5. Makefile 10](#_Toc526164414)

[1.6. in de Makefile 10](#_Toc526164415)

[1.7. RTTTL file naam in de makefile 10](#_Toc526164416)

[Week 2: print\_asciz (assembler) 11](#_Toc526164417)

[week 3: NEC IR decoder (C++, rtos) 12](#_Toc526164418)

[week 4: constexpr klokje (C++) 13](#_Toc526164419)

[week 5: LZ decoder (assembler) 14](#_Toc526164420)

[Jaar 2 – Themaopdracht Devices 15](#_Toc526164421)

[Jaar 2 – Meten Regelen & Besturen 16](#_Toc526164422)

# Jaar 1 – TICT-V1OOPC-15

## Leerdoelen

Door middel van GIT software code binnen halen en zelf geschreven code op een ordelijke manier op een eigen account publiceren.

* Organiseren
* Git

Door middel van een UML klassediagram een decompositie maken van een te ontwikkelen applicatie, met gebruik van associaties, aggregaties en overerving.

* Ontwerpen
* OO
* UML Klassendiagram
* Inheritance
* Associaties
* Aggregaties
* Composities

Op basis van een UML klassediagram of een andere vorm van specificatie C++ klassen schrijven met gebruik van private, public, inheritance, en virtual.

* C++
* UML
* OO
* Klassendiagram
* Private
* Public
* Inheritance
* Virtual

Een Abstract Data Type met bijbehorende operatoren in C++ implementeren.

* C++
* Abstract data type
* User defined operator

De C++ taalconstructies defaults, const, reference, overloading, override, auto, en for(:) gebruiken.

* C++
* Defaults
* Const
* Reference
* Overloading
* Override
* Auto
* For

Een unit test schrijven en uitvoeren met gebruik van een unit test library..

* Testen
* Unit test

Een C++ klasse interface documenteren met behulp van Doxygen.

* Interface
* Documentatie
* Doxygen

De decorator en adapter patterns gebruiken.

* Patterns
* Decorator
* Adapter

Een cross-development systeem gebruiken om software te ontwikkelen voor een micro-controller.

* Micro-controller
* Cross-development

GPIO pinnen van een micro-controller gebruiken om eenvoudige inputs (schakelaars) en outputs (LEDs) te gebruiken.

* Hardware
* GPIO

SPI en I2C gebruiken om peripherals te interfacen.

* SPI
* I2C

## Week 4

### 4.1. Materiaal

Bij de docent kan je LEDjes, weerstanden, en een 74HC595 chip krijgen voor de practica van deze week en de volgende week.

### 4.2. Installatie

Installeer GCC voor ARM, hwlib, en de Arduino Due device driver. Test dat alles werkt door de knipper voorbeeldcode te runnen.

### 4.3. Patroon

Sluit 4 LEDs (met weerstanden!) aan op Due pinnen. Schrijf een programma dat, gebruikmakend van hwlib::target::pin\_out, Het volgende patroon laat zien:

|  |  |
| --- | --- |
| tijd 🡺 | XX--  -XX-  --XX  -XX-  XX-- |

etc.

### 4.4. Input

Sluit naast de LEDs van de vorige opdracht, nog twee drukschakelaars aan. Schrijf een programma dat zorgt dat de ene schakelaar een LED meer aan zet, en de andere een LED minder. Indrukken van de ‘meer’ schakelaar maakt dus de stappen

----

X---

XX--

XXX-

XXXX

Bouw je programma zo op dat de main() ‘weet’aan welke pinnen de LEDs en schakelaars zitten, en dat het eigenlijk werk wordt gedaan in een functie die als parameters een port\_out en twee pin\_in’s mee krijgt.

## Week 5

### 5.1. HC595

Sluit, naast de LEDs en schakelaars van de vorige opdrachten, nog een 74HC595 chip aan, met LEDs (en weerstanden!!) op 4 van zijn uitgangen. Test je schakeling door je meer/minder programma uit te breiden naar 8 LEDs (4 op de Due pinnen, 4 op de HC595).

Let op dat je alle pinnen van de HC595 chip goed gebruikt. Vul daartoe de onderstaande tabel in, dat maakt het ook makkelijker voor jezelf om de chip aan te sluiten. Inputs moet je altijd aansluiten, outputs alleen als je die gebruikt. Raadpleeg de datasheet van de chip voor de details die je niet uit de reader kan halen.

|  |  |  |
| --- | --- | --- |
| aansluiten aan |  | aansluiten aan |
|  |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### 5.2. Decorator

Schrijf een twee-input AND decorator voor de pin\_in klasse. De constructor van deze decorator heeft twee pin\_in’s als parameters. De get() van de decorator geeft 1 terug als de get()’s van beide ‘slaaf’ pin\_in’s 1 teruggeven (anders geeft hij 0 terug). Demonstreer deze decorator.

### 5.3. Knipperen

Schrijf een pin\_out\_invert decorator. Schrijf een programma dat zorgt dat de kitt() functie op de 8 LEDs het volgende patroon laat zien:

XXXX----

----XXXX

XXXX----

----XXXX

etc.

Doe dit door de 8 pin objecten te combineren via jouw pin\_out\_invert decorator en de bestaande all decorator.

## Week 6

### 6.1. OLED

Sluit de OLED aan en run de voorbeelden uit de reader. Doe zelft iets met de OLED, teken bv. een huisje.

### 6.2. De code inleveren

Schoon je practica op. Als je in CodeLite ‘Clean Workspace’ commando uitvoert ben je al een heel eind, maar haal ook de template directories en de gegeneerde Doxygen documentatie weg en eventuele andere gegenereerde zaken.

Maak een github account aan met het Student Developer Pack. Creeer een **private** github repository voor je code. Haal die repository binnen (dit creeert een lege directory), copieer je code naar de directory, en push je code naar github door (in die directory) de volgende commando’s te geven:

git add -A

git commit -a -m 'practica inleveren'

git push

Nodig je docent uit lezer van je repository.

# Jaar 1 – IPASS

## Leerdoelen

# Jaar 1 - Inter Disiplinair Project

## Leerdoelen

<<Omschrijving van het IDP>>

# Jaar 2 – TICT-V2CPSE1-16

## Leerdoelen

* Bouwprocessen
  + Pre-processen
  + Compileren
  + Linken
  + Make
  + Separate compilatie
* C++ Aspecten
  + Templates
  + Lambda’s
  + Constexpr
* Multithreading
  + RTOS
* Assembler
  + Cortex-M0 assembler

## Week 1: ringtone (C++)

### UML

De ringtone demo is een applicatie voor de Arduino Due die een ringtone in rtttl formaat afspeelt via een luidsprekertje (met een 100 Ohm serieweerstand!) op pin d7. Bestudeer de code en stel het UML klassediagram ervoor op, ook voor het ‘fur-elise’ deel dat in de demo is uitgeschakeld.

### Opbouwen

Bouw de schakeling en probeer de demo met de fur-elise en met een andere ringtone. Op <http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/> vind je een groot aantal rtttl files.

### Noten printen

Maak een copie van de ringtone demo directory en pas die zodanig aan dat het runt op de PC en de ringtone niet wordt afgspeeld maar noot voor noot geprint. Hiertoe moet je een eigen note\_player maken die in plaats van de noot te laten klinken de noot print.

### RTTTL string converter

Maak een copie van je vorige werk en het zo aan dat er een file wordt gemaakt volgens het formaat van fur\_elise.cpp met daarin voor iedere noot de overeenkomstige p.play(…) call.

### Makefile

Pas de makefile en de applicatie aan zodat

1. de applicatie die de string naar een file met .play() calls omzet wordt gebouwd (gcc aanroep) als dat nodig is (waar is deze applicatie van afhankelijk?)
2. de RTTTL string die de applicatie vertaalt wordt meegegeven

(die string met rare tekentjes) in de Makefile staat, en het decoderen van de ringtone string in een PC (Windows) applicatie gebeurt. In de embedded applicatie komt dus alleen maar een reeks aaroepen van player.play() te staan. Hiertoe moet je op de PC de ringtone player zo aanroepen dat hij niet piept maar de file met aanroepen van player.play() produceert. Dit kan je doen zonder de play functie te wijzigen, door een eigen player te schrijven.

### in de Makefile

Maak een copie van de ringtone demo en bouw die om zodat de ringtone (die string met rare tekentjes) in de Makefile staat, en het decoderen van de ringtone string in een PC (Windows) applicatie gebeurt. In de embedded applicatie komt dus alleen maar een reeks aaroepen van player.play() te staan. Hiertoe moet je op de PC de ringtone player zo aanroepen dat hij niet piept maar de file met aanroepen van player.play() produceert. Dit kan je doen zonder de play functie te wijzigen, door een eigen player te schrijven.

### RTTTL file naam in de makefile

Maak vervolgens een andere copie waarin de ringtone wordt gespecificeerd door in de Makefile de ringtone file op te geven. Ook in dit geval moet het decoderen van de ringtone string op de PC gebeuren, niet op de Arduino.

## Week 2: print\_asciz (assembler)

Ga uit van het asm print project (maak er een copie van). Test dit project en bestudeer hoe het werkt.

1. Vervang de print\_asciz functie door de equivalente assembler code. Stop dit in een .asm file, en voeg die file name toe aan de SOURCES in de lokale Makefile. Merk op date er nog een paar details ontbreken.

|  |
| --- |
| **print\_asciz:**  **push { r5, lr }**  **mov r5, r0**  **loop:**  **ldrb r0, [ r5 ]**  **add r0, r0, #0**  **beq done**  **bl put\_char**  **add r5, r5, #1**  **b loop**  **done:**  **pop { r5, pc }** |

2. Neem wat je nu net gemaakt hebt (met print\_asciz in assembler) en vervang nu ook de application() functie door een equivalente subroutine in assembler.

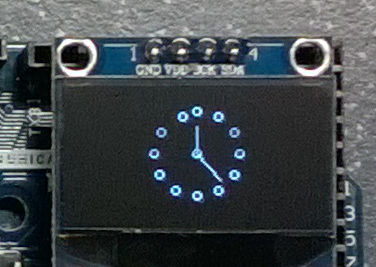
3. Maak een C functie (dus met extern ”C”, of in een extern “C” block) die zijn char argument geconverteerd teruggeeft: een uppercase character wordt lowercase, en een lowercase character wordt uppercase. Andere characters blijven ongewijzigd. Roep deze functie aan in de print\_asciz subroutine om elk charprint te converteren voordat het door put\_char wordt geprint, zodat ”Hello world, the ANSWER is 42!\n” wordt geprint als ”hELLO WORLD, THE answer IS 42!\n”. Gebruik de test string die in het voorbeeld programma staat, daar staan ook de characters in die net voor en net na de letters staan in de ASCII tabel.

4. Vervang deze C functie door een equivalente assembler subroutine. Let op: je kan in ADD en SUB instructies geen 8-bit literal gebruiken, dus zo’n waarde moet je eerst in een register zetten met een MOV instructie.

## week 3: NEC IR decoder (C++, rtos)

De RTOS reader beschrijft een ontwerp (initial en final class diagram, concurrency diagram, State Transition Diagrams). Vertaal dit ontwerp op de in de reader beschreven manier naar een C++ applicatie met 3 tasks. Test dit met een IR control (verkrijgbaar in de Turing lab shop).

## week 4: constexpr klokje (C++)



Schrijf een applicatie voor op de Due die op het OLEDje een analoog klokje laat zien met twee wijzers (minuten en uren). Je mag natuurlijk nog een seconden wijzer toevoegen. De plekken van de cijfers (of symbolen) en de eindpunten (en evetueel ook de beginpunten) van de wijzers moet je uitrekenen dmv. sinussen en cosinussen.

Je mag zelf bepalen wat je op de plekken van de cijfers zet (de cijfers, een streepje, een cirkeltje, etc.), maar er moet daar wel iets staan. Als je de cijfers wil tekenen kan je kijken naar hwlib:: font\_default\_8x8, dit zijn plaatjes van de ASCII tekens die ook door de console klasse worden gebruikt.

**Zorg dat je geen run-time floating-point gebruikt.** Gebruik daartoe een constexpr opzoektabel die tijdens het compileren gevuld wordt. Zorg dat je het klokje gelijk kan zetten met een drukknopje per wjzer.

Gebruik de gebufferde versie van de oled klasse. Het tekenen en flushen kost tijd, dus om je klokje redelijk gelijk te laten lopen moet je geen gebruik maken van een delay, maar steeds de huidige tijd opvragen en die gebruiken (gecorrigeerd met een waarde voor het gelijk zetten).

Extra uitdagingen:

Het is niet perse nodig de sinus en cosinus waarden voor alle hoeken op te slaan, je hebt genoeg aan de sinus (of cosinus) waarden voor 0 .. 45 graden (andere waarden kan je daar uit afleiden door slim te spiegelen), in stappen van 1 tijd-seconde (6 graden).

Naast de analoge klok is er nog ruimte op het oled om de tijd ook digitaal weer te geven.

## week 5: LZ decoder (assembler)

Einddoel: een LZ decoder in assembler op de Arduino Due.

Stappen:

1. Neem de LZ encoder en breidt die uit met een decompressor/decoder (in C++, op de PC) die zijn uitvoer (via en lambda parameter) naar std::cout schrijft.
2. Pas de aanroep van de LZ compressor/encoder aan zodat er een assembler file wordt gegenereerd die de gecodeerde characters bevat (en die in een volgende stap meegenomen kan worden in een Due applicatie).
3. Schrijf een Due assembler subroutine die de characters die in de vorige stap gegenereerd zijn decodeert (naar keuze naar de standard output of naar de oled).
4. Pas de Makefile aan zodat het coderen/comprimeren en het bouwen van de assembler applicatie automatisch gebeurt als dat nodig is.

# Jaar 2 – Themaopdracht Devices

## Leerdoelen

Op basis van een requirements architecture een solution architecture opstellen voor de software van een embedded systeem

* Solution architecture
* UML klassendiagram
* Taakstructurering
* Concurrency Diagram
* UML STD

De solution architecture omzetten naar een werkend embedded systeem

* Programmeren in C en C++

Code documenteren met gebruik van Docygen

* Documenteren
* Docygen

Gebruik maken van een versiebeheersysteem

* Versiebeheer
* GitHub

Samenwerken met het team en de docenten

* Samenwerken

Verantwoordelijkheid tonen voor het functioneren van het team en de voortgang van het project

* Verantwoordelijkheid
* Escaleren

Een rapport schrijven

* Rapporteren

Onderzoek doen naar de implementatie van concurrency mechanismen m.b.v. de faciliteiten van een concreet operating system

* Pool
* Semafoor
* Mutex
* Channel
* Timer
* Eventflag

Een planning van de taken en een werkverdeling maken

* Planning
* Werkverdeling

# Jaar 2 – TCTI-R2D2-17

## Leerdoelen

zelfstandig nieuwe oplossingen (algoritmen) bedenken, ontwikkelen en testen rekening houdend met (soft real-time) performance en resource management binnen een gedistribueerd computer systeem.

* C++,
* testing,
* optimalisatie

een ontwerp voor een soft-/hardware oplossing onderbouwen en analyseren en deze vastleggen d.m.v. toepasselijke verslagleggingsmethodes.

* UML,
* datastructuren,
* algoritmen

actuatoren, sensoren, image processing technieken, computer vision algoritmen en AI-technieken inzetten als onderdeel van een gedistribueerd systeem.

* sensoren,
* actuatoren,
* algoritmen

afspraken tussen teams en professionals vastleggen en uitvoeren.

code beheren in een code versioning system gebruikmakend van een professionele workflow mechanisme inclusief branching en release management om zodanig het eigen werk te integreren met het werk van anderen.

* git,
* workflow,
* release management,
* integratie

(pro)actief samenwerken met zowel zijn eigen als andere teams, mede door vroegtijdig acties te ondernemen bij problemen/uitlopende planning/gebrekkige samenwerking of andere problemen waardoor het project risico loopt.

een kritisch houding aannemen, door zowel het eigen werk(en) als het werk en de houding van teamleden op een methodische wijze te evalueren.

bij een gegeven opdracht en/of doelstelling relevante, precieze, consistente en functionele hoofd- en deelvragen afbakenen.

* biebvraag,
* werkplaatsvraag,
* labvraag

voor het beantwoorden van de deelvragen een passende en haalbare aanpak beschrijven, waarbij, indien van toepassing, wordt ingegaan op de procedure, participanten, data-verzameling (en meetinstrumenten) en data-analyse.

relevante (wetenschappelijke) bronnen vinden en evalueren op bruikbaarheid ten behoeve van het verkrijgen van inzicht in actuele visies/modellen/technologie binnen het desbetreffende domein.

* literatuuronderzoek

een theoretisch kader te schrijven dat inzicht geeft in actuele visies/modellen/technologie binnen het desbetreffende domein.

* theoretisch kader

een (taalkundig) verzorgd, gestructureerd onderzoeksrapport schrijven dat valide conclusies trekt op basis van de gevonden resultaten en een adequaat advies geeft.

* onderzoeksrapport

een oordeel geven over het onderzoeksrapport van anderen.

een (cross-platform) ontwikkel- en testomgeving opzetten voor zowel soft- als hardware

* Ontwikkelomgeving

# Jaar 2 – Meten Regelen & Besturen

## Leerdoelen

Kennis opbouwen over:

* Sensoren
  + Karakteristieken
  + Interfacing
  + Fouten
* Elektronische Componenten
  + Basiscomponenten
  + OpAmps
* Filteren & signaalprocessing
  + Analoog/digitaal
  + Fourier transformatie
* Besturingscomponenten
  + FPGA
  + PLC
  + DSP
  + GPU
* Motoren
  + Typen motoren
  + Eigenschappen
* Voedingen
  + Spanning
  + Stroom
  + Vermogen
  + Eigenschappen voeding

HBO-I matrix:

Accenten liggen op Software (S) en Hardware (H) in de verticale lijn. Op de horizontale lijn ligt het accent op het realiseren en ontwerpen van hardware.