Universität Osnabrück 03.04.2018

Seminar zur Didaktik der Informatik

Prof. Dr. Michael Brinkmeier

Hendrik Langebrake, Christian Ventker

WS 2017/18

Das Calliope-Orchester

Ein physical-computing-Projekt für die Oberstufe

Inhalt

Einleitung

Das Calliope-Orchester ist ein Schülerprojekt für die Oberstufe. Im Zentrum steht der „Calliope Mini“, ein Ein-Platinen-Computer, der speziell für den Einsatz an Schulen in Deutschland entwickelt wurde.[1] Als interdisziplinäres Projekt der Schulfächer Informatik und Musik sollen die SuS die Funkverbindung in den integrierten Lautsprecher des Calliope nutzen, um am Ende ein gemeinsames, mehrstimmiges Musikstück zu spielen. Jeder Calliope kann dabei nur einen Ton gleichzeitig erzeugen, sodass sie per Funk verbunden werden müssen, um mehrstimmig zu spielen. Im Endeffekt soll hierbei mit grundlegenden und speziellen Inhalten der Fächer Informatik und Musik von den SuS gemeinsam und eigenverantwortlich im Rahmen der technischen Möglichkeiten und angemessener Kommunikation ein möglichst „schönes“ Musikstück erzeugt werden. Ein besonderes und vielfältiges Projekt, an das sich viele Schülerinnen und Schüler hoffentlich noch lange nach ihrer Schulzeit zurückerinnern werden. Im Folgenden sollen der Code für das „Calliope-Orchester“ erläutert, sowie auf technische Hürden und didaktische Hintergründe eingegangen werden.

[1] Siehe auch: https://calliope.cc/

Technische Hürden und Schwierigkeiten

Das Software-Problem

Die Verwendung von *Abbozza* ist nach aktuellem Stand für dieses Projekt nicht ausreichend. *Abbozza* müsste hier um die Funktion erweitert werden, selber Blöcke zu definieren und zu speichern, die dann vom Calliope als Noten interpretiert werden können. Abhilfe schaffen konnten wir uns mit der Software *Yotta*. Diese kann den C++ Code, der in einem separaten Editor[2] geschrieben ist, über die Kommandozeile in einem vordefinierten Verzeichnis mit dem Befehl „yotta build“ in eine Hex-Datei speichern, die vom Calliope gelesen werden kann. Hier ergeben sich weitere Probleme für die Umsetzung des Projektes:

- Man braucht mit einem Editor und einem Compiler zwei separate Programme
- Die Installation von *yotta* ist aufwändig und die Handhabung mit der Kommandozeile erklärungsbedürftig
- Das Kompilieren mit *yotta* dauert relativ lange, meist 20 Sekunden und länger
- Die Verzeichnisstruktur von *yotta* ist undurchsichtig, für die erzeugten Dateien muss man in aller Regel ein tiefes Verzeichnis aufsuchen
- Die Verwendung einer nicht-grafischen Programmierumgebung ist für den Schulkontext nicht intuitiv

Sollte *yotta* für das finale Projekt mit Schülern genutzt werden, so ist eine Einführung mit Installation und Benutzung notwendig.

Installationsanleitung für yotta

Lade yotta auf docs.yottabuild.org herunter und führe eine Standartinstallation durch.

Findn und lade die Datei *srec_cat* auf
https://srecord.sourceforge.net/projects/srecord/files/srecord-win32/ herunter.

Diese Datei im Installationsverzeichnis von yotta platzieren.

Erstelle im gewünschten Verzeichnis einen Projektordner (Workspace).

Führe das yotta Konsolenprogramm aus (run yotta).

Navigiere in den zuvor erstellten Projektordner (Befehl: cd).

Initialisiere yotta mit dem Befehl *yotta init*, folge den Anweisungen und bestätige bei der Frage nach „executeable“ mit „yes“.

Konsoleneingabe: *yotta target calliope-mini-classic-gcc*

Konsoleneingabe: *yotta install calliope-mini/microbit*

[2] Hier verwendeter Editor: Atom

Entwicklungsumgebung

Eine textbasierte Entwicklungsumgebung ist erforderlich (z.B. Atom). Zum Kompilieren muss der Quellcode im zuvor definierten Projektordner (s.o.) im Unterordner „source“ abgelegt werden. Danach kann man auf der Konsole „yotta build“ aufrufen, um das gesamte Projekt zu kompilieren. Dann kopiert man im Projektordner im Unterordner build/calliope-mini-classic-gcc/source die *combined.hex* Datei auf den Calliope.

Freischaltung der Funk-Antenne

Im Verzeichnis yotta_modules/microbit-dal/inc/core/MicroBitConfig.h muss der String MICROBIT_BLE_ENABLED gefunden und von 1 auf 0 gesetzt werden, da die Kommunikation zwischen den Calliopes sonst über Standartmethoden nicht möglich ist.

Das Synchronisationsproblem

Zwei oder mehr Calliopes, die gemeinsam und gleichzeitig eine Melodie spielen, müssen absolut synchron spielen um von Anfang bis Ende „im Takt“ zu bleiben. Kleine Ungenauigkeiten, die sich fortpflanzen können das musikalische Erlebnis schnell ruinieren und von der Ästhetik des geplanten Stückes nichts übrig lassen. Im Zuge dieser Überlegungen entstand folgender Test: Zwei Calliopes bekommen jeweils eine Folge von Noten, die nur aus Vierteln besteht, jeweils im Quintabstand versetzt. Starteten wir die Folge und variierten dann den Abstand der Calliopes kamen sie stets „aus dem Takt“. Hier wurde jedes Mal ausschließlich zum Start der Sequenz ein Signal vom Master-Calliope ausgesendet. Interessanterweise fanden sie in manchen Fällen wieder zueinander und beendeten die Notenfolge gleichzeitig und gemeinsam. Wurde die Funkverbindung kurz unterbrochen war schnell alle Musikalität verloren, da die Calliopes ab da immer versetzt zueinander weiterspielten.

Um dieses Synchronisationsproblem zu lösen entstand die Idee der regelmäßigen „Ticks“ zur Fehlerkorrektur. Der Master-Calliope (Lehrer-Calliope) sendet pro Minute eine regelmäßige Frequenz an Signalen an alle Slave-Calliopes (Schüler-Calliopes), die *Ticks*. Alle Calliopes spielen dann ihre Töne genau auf dem *Tick* und niemals dazwischen. Weiterhin sind *Ticks* mit einem Zähler versehen, sodass jeder Calliope sofort merkt, wenn ein Signal verloren wurde. Wurde ein solcher Fehler festgestellt, leuchtet die LED des entsprechenden Calliopes rot auf. Die Anzahl der *Ticks*, die der Master-Calliope aussendet ist dabei nicht ganz trivial, da sie den schnellstmöglichen Notenwert bestimmt. Gehen wir von Takten mit 16 *Ticks* aus, dann ist der schnellstmögliche Notenwert die Sechzehntelnote. Diese dauert genau einen *Tick* lang. Eine ganze Note würde in diesem Fall 16 *Ticks* lange dauern.

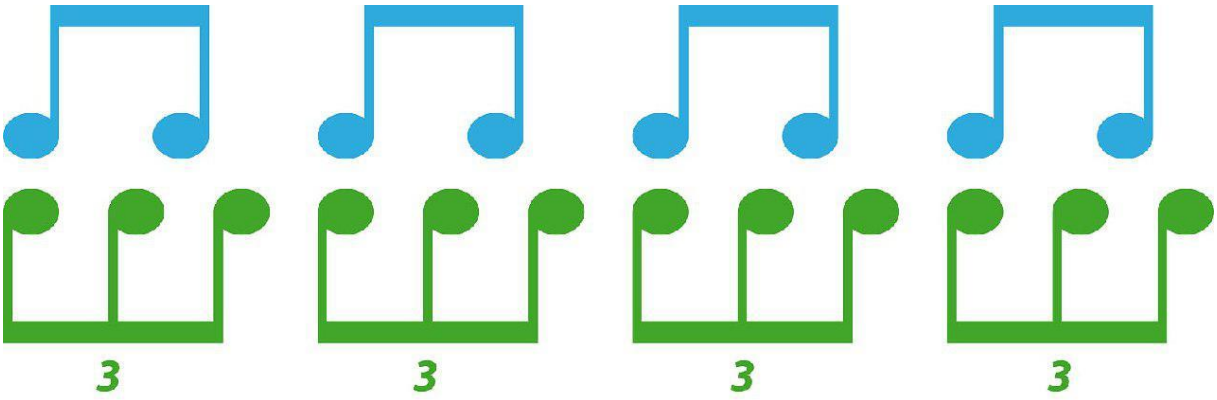
Das Funk-Problem

Das Funkprotokoll des Calliope Mini bietet Raum für potentielle Probleme. Zunächst ist die im Calliope verbaute Funkantenne relativ schwach. Für direkte und ungehinderte Übertragung durch die Luft ist die Größe eines Klassenraumes kein Problem. Leider können auch schon kleine Hindernisse das Funksignal stören. Nach einigen Tests ist die Übertragung durch Kleidung, Fenster, Bildschirme und Türen schwierig. Auch kann man mit dem Körper das Funksignal fast vollständig abfangen. Eine Funkübertragung durch die Wand ist nicht möglich. Sendet der Master-Calliope wie oben beschrieben eine regelmäßige Anzahl an Ticks pro Minute, können hier wichtige Ticks verloren gehen, was wieder Störung der Synchronisation bedeuten würde. Mit der Fehlerbehandlung kann der Fall aufgefangen werden, dass die Funkverbindung kurzzeitig gestört ist. Mit der ständigen Fehlerkorrektur bringt sich der Calliope selbst wieder in den richtigen Takt und spielt korrekt weiter. Schwieriger ist die Vorstellung, dass der Schüler in der letzten Reihe quer durch den Raum eine funktionierende Funkverbindung zum Lehrer-Calliope aufbauen soll. Ggf. kann hier Abhilfe geschaffen werden, indem die SuS zum Testen ihrer Programme und Musikstücke mit ihrem Calliope nach vorne zum Lehrer-Calliope kommen müssen, oder dass der Lehrer-Calliope in der Mitte des Raumes liegt. Möglicherweise ist es die beste Lösung für eine möglichst ungestörte Funkverbindung den Master-Calliope einfach im Klassenraum an die Decke zu hängen. In die Hauptschleife des C++ Programms sollte man außerdem den Befehl *uBit.radio.setTransmitPower(7);* die Funkstärke manuell auf den höchsten Wert setzen.

Das Triolen-Problem

Eine Triole in der Musik ist definiert als Wert, der in drei Drittel unterteilt wird. Beispielsweise können bei Achteltriolen drei Achtel auf die Dauer von zwei Achteln (einer Viertel) aufgeteilt werden.

„normale“ Achtel



Triolen - Achtel

Für unsere Calliopes (und für viele sehr mathematisch denkende Menschen) stellt sich hier die Frage, wie man drei Achtel in regelmäßige Viertel unterbringen soll. Was die Calliopes betrifft, so besteht die naheliegende und vermutlich einzige Möglichkeit darin, die Anzahl der

Ticks zu erhöhen. Hier muss von vornherein wieder die Entscheidung getroffen werden, welcher kleinste Notenwert unterstützt werden soll. Gehen wir von Achteltriolen aus und multiplizieren wir unsere bisherige Tickanzahl (16) mit dem Faktor 3, erhalten wir 48 *Ticks* pro Takt mit folgender Aufteilung:

Ganze Note: 48 Ticks

Halbe: 24 Ticks

Viertel: 12 Ticks

Vierteltriole: 8 Ticks

Achtel: 6 Ticks

Achteltriole: 4 Ticks

Sechzehntel: 3 Ticks

Sechzehnteltriole: 2 Ticks

(Zweiunddreißigsteltriole: 1 Tick)

Entscheidend hierbei ist, dass die Tickzahl der jeweilige Triole immer mit dem Faktor 3 multipliziert den nächst höheren Notenwert ergibt. Beispiel: Drei Achteltriolen ergeben genau die Länge einer Viertelnote, denn 4 Ticks * 3 = 12 Ticks. Triolen können damit als vollständige Triolen gespielt werden. Dies ist zunächst rudimentär, da Triolen mit Pausen, Überbindungen oder Notenwertwechsel innerhalb der Triole nicht immer möglich sind. Mit Sechzehnteltriolen als kleinstem Notenwert sind jedoch viele rhythmische Ausdrücke möglich. Wie so oft geht es im Endeffekt bei der Frage nach Triolen darum, den gewonnen Nutzen gegen die Ressourcen und die Rechenzeit (höhere Anzahl an Ticks) abzuwägen und zu testen. Ferner kann die Triolenproblematik und die Anzahl und Berechnung einer idealen Tickmenge ein guter Ausgangspunkt sein, um hier mit den Schülerinnen und Schülern zu diskutieren, Lösungen zu entwickelt, sowie Vor- und Nachteile zu erörtern.

Das Speicher-Problem

Dieses Problem beinhaltet eigentlich gleich Probleme. Zum einen können keine Dateien ohne weiteres auf dem Calliope gespeichert und ausgelesen werden. Ein Musikstück in einer Datei zu speichern und vom zuvor gespeicherten Programm auf dem Calliope auslesen und interpretieren zu lassen ist somit nicht möglich! Sämtliche Noten müssen im Quellcode hart codiert sein und jede Änderung muss neu kompiliert und auf den Calliope übertragen werden. Innerhalb von C++ bietet sich hier ein *struct* Datentyp an, mit den Feldern Frequenz, Artikulation und Notenwert. Ein fertiges Musikstück ist somit letztendlich ein Array aus Noten. Zweites Problem ist die Begrenztheit des Flashspeichers des Calliope Mini. Alle gespeicherten Musikstücke unterliegen einer maximalen Länge. Da es für den Calliope keinen Unterschied macht, ob eine Viertel oder eine Halbe Note gespeichert wird, ist die Länge

tatsächlich durch die absolute Anzahl an Noten begrenzt und nicht etwa durch die Dauer des Stückes oder die Anzahl der Takte. Als Musiker eine zunächst seltsame und gewöhnungsbedürftige Vorstellung. Ggf. sollte auch diese kognitive Hürde mit den SuS besprochen werden. Durch speicherschonendes Programmieren kann hier eine höhere Anzahl an gespeicherten Noten erreicht werden. In einer früheren Version wurde im Code für eine Note immer die Dauer, die Tonhöhe und die Artikulation gespeichert. Die Artikulation teilte sich hierbei zunächst in Staccato und Legato auf. Weitere Auswahlmöglichkeiten (Non-Legato, Portato,…) waren geplant, sodass zur Speicherung der Artikulation zunächst ein Integer genutzt wurde. Hiermit war es möglich bis zu 85 Noten zu speichern. Durch anschließendes Debugging und Testen war festzustellen, dass man eine deutlich höhere Anzahl an Noten speichern kann, wenn man nur die Artikulationswerte Staccato und Legato nutzt und diese in einer Boolean-Variable speichert. Durch den Verzicht auf weitere Artikulationsmöglichkeiten ist es möglich, bis zu 200 Noten im Code zu speichern, ohne dass Flashspeicher überläuft. Hier gilt es wieder entsprechend abzuwägen. Auch dieser Konflikt, Artikulationsmöglichkeiten vs. mögliche Länge des Musikstücks kann, in Zusammenhang mit der Variablenauswahl mit den SuS besprochen werden.

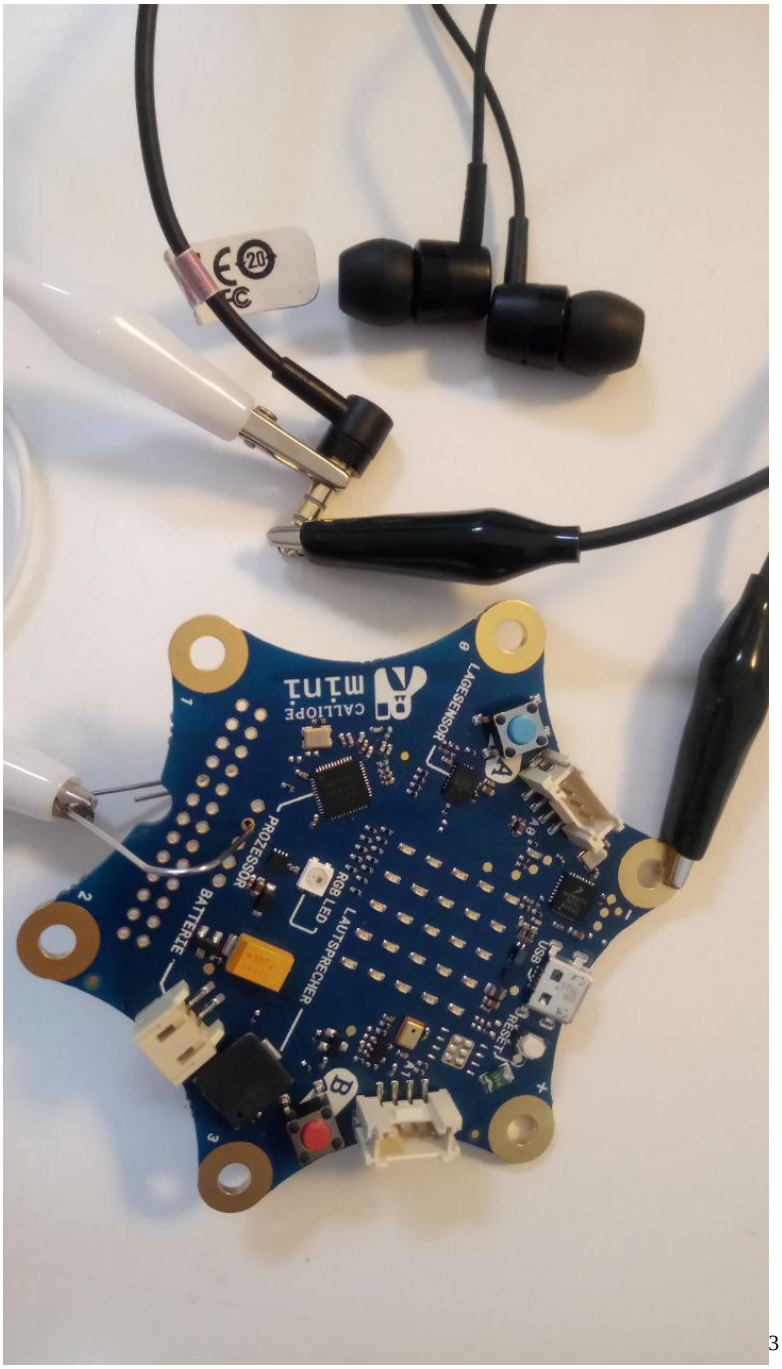
Das Frequenz-Problem

Gemeint ist hier nicht die Funkfrequenz, sondern die Frenquenz der Töne, welche der Calliope letztendlich wiedergeben soll. Aufgrund technischer Limitierung des Calliope Mini ist es nur möglich ganzzahlige Frequenzen wiederzugeben. Gleitkommazahlen können im Frequenzbereich nicht wiedergegeben werden. Dieses Problem ist nur handhabbar, indem man alle Frequenzen, die wiedergegeben werden sollen auf einen ganzzahligen Wert rundet. Die Töne, die hierbei entstehen sind dementsprechend nicht ganz exakt, diese meist minimalen Änderungen der Frequenzen fallen aber so wenig auf, dass der Unterschied kaum hörbar ist. Für den Schulalltag sollte hier keine Beeinträchtigung vorliegen.

Das Lautstärke-und-Klangfarbe-Problem

Die Lautstärke des Lautsprechers auf dem Calliope ist aus physikalischen Gründen begrenzt und auch festgelegt. Für den Unterricht mag es häufig sinnvoll sein, dass die Lautstärke nicht zu penetrant ist, insbesondere natürlich, wenn mehrere Calliopes gleichzeitig im Einsatz sind. Allerdings muss man sich in diesem Zusammenhang bewusst machen, dass endgültige Klangerlebnisse nur gut hörbar sein werden, wenn im Klassenraum ansonsten Ruhe herrscht. Es ist möglich über die Pins des Calliope Mini Kopfhörer oder Lautsprecher anzuschließen, um die Lautstärke zu erhöhen. Hierbei muss das elektrische Signal an einem bestimmten Pin mit einer Büroklammer oder ähnlichem abgefangen werden und kann dann an einen Klinkenanschluss weitergegeben werden. Am besten sollte man hier aktive Lautsprecher benutzen, die die Lautstärke im Vorhinein leiser regeln können, da die Calliopes ansonsten mit voller Lautstärke spielen. Das Erzeugen einer anderen Klangfarbe ist schwierig, da der Calliope Mini aufgrund seiner begrenzten Rechenleistung keine Sinusschwingung oder Sägezahnschwingung für additive bzw. subtraktive Klangsynthese berechnen kann. Prinzipiell

könnte man mit weiterer Recherche und einer Ausweitung des Projektes Midi-Dateien auf dem Calliope erzeugen und somit beliebige Klangfarben generieren und mit externer Hardware wiedergeben.



[3]

[3] Anschluss von Kopfhörern an den Calliope Mini

Anleitung zum Erzeugen eines Mustikstücks am Beispiel des Menuett in G-Dur

In der Hauptschleife von Slave.cpp kann die zu spielende Stimme der vorgefertigten Menuettsequenz gewählt werden.

Kompiliere nun nacheinander, wie oben beschrieben, die Master.cpp und Slave.cpp

Kopiere jedes entstandene Programm (combined.hex) auf einem Calliope.

Der Master-Callipe kann mit dem Knopf A ein Startsignal senden, mit B ein laufendes Signal beenden. Läuft kein Signal kann mit B die Geschwindigkeit festgelegt werden, indem im Takt in die Hände geklatscht wird.

Achte darauf, dass die Calliopes in unmittelbarer Nähe zueinander liegen, um den Funk nicht zu beeinträchtigen!

Didaktische Überlegungen

Altersgruppe und Schulform

Das vorliegende Calliope-Orchester-Projekt ist aufgrund seiner Komplexität für die gymnasiale Oberstufe gedacht. Auch aufgrund der Offenheit bei der Musikauswahl, der nötigen Absprachen im Team und mit anderen Gruppen und der vorausgesetzten Selbständigkeit und Organisation raten wir zu einer Umsetzung in der Oberstufe.

Voraussetzungen an die Schülerinnen und Schüler

Es wird für dieses Projekt davon ausgegangen, dass die SuS vorher bereits Informatikunterricht in der Schule hatten. Zum einen muss ein Einstieg in die Programmierung stattgefunden haben, da ein großer Teil des Projektes darauf basiert, dass ein Musikstück in einem C++ Programm gespeichert wird. Bei geringen Programmierkenntnissen kann hier vorentlastet werden, indem das Programm für die Schüler-Calliopes vorgegeben wird. Bei sehr erfahrenen Klassen kann mit Erläuterung der Funktionsweise des Lehrer-Calliopes der Schüler-Calliope selber programmiert werden. Hier muss allerdings das zu Grunde liegende Prinzip deutlich gemacht werden: Der Lehrer-Calliope sendet Ticks in einer bestimmten Frequenz über die Funkverbindung an die anderen Calliopes. Hieraus soll letztendlich die Musik entstehen. Dies beinhaltet zugleich die zweite große Voraussetzung an die SuS neben der Programmierkenntnis: Ein Verständnis von technischen Übertragungen und grundlegender Hardware-Funktion. Der Umgang mit Funk und Lautsprecher des Calliope Mini muss nach kurzer Erläuterung soweit verinnerlicht sein, dass die SuS damit selbstständig weiterarbeiten können.

Inhaltliche Aspekte

Verschiedene inhaltliche Aspekte aus dem Informatikunterricht können in diesem Projekt thematisiert werden:

- Der Umgang mit Variablen (Wie speichere ich Noten? Welche Datentypen benutze ich, damit ich möglichst viele Noten speichern kann, ohne das der Flashspeicher des Calliope überläuft)
- Grundlagen des physical computing (Welche Änderungen im Code führen zu einer Veränderung/ Verbesserung der Funktionsweise des Calliope im Sinne des Projektes?)
- Programmieren (Wie kann ich in C++ ein Programm schreiben, dass die Bedingungen des Projektes erfüllt? Oder: Wie muss ich das gegebene Programm verstehen, um damit im Sinne des Projektes arbeiten zu können?)
- Testen und Debugging (Unter welchen Bedingungen sind Funkverbindung oder Datenübertragung nichtmehr zuverlässig? Was für Lösungen gibt es für Probleme (s.o.)? Wo muss man zwei Alternativen abwägen?
- Hardwarekenntnisse (Wie funktioniert der gegebene Lautsprecher? Wie funktioniert die Funkverbindung?)

Darüber hinaus können weitere Kenntnisse erworben und vertieft werden, die nicht speziell zum Informatik-Curriculum gehören. Etwa das Arbeiten und Absprechen als Team, der Kontakt zu anderen Teams und, wie wir noch sehen werden, inhaltliche Aspekte aus dem Fach Musik.

Themenbezüge zu anderen Fächern

Insbesondere und allen voran sind für dieses Projekt natürlich thematische Bezüge zum Schulfach Musik gegeben. Diese Bezüge treffen vor allem den Bereich der systematischen Musikwissenschaft:

- Elektronische Klangerzeugung (Wie funktioniert ein Lautsprecher/Kopfhörer?)
- Frequenzen (Wie verhalten sich Frequenzen bei Tonänderung? Verdoppelung der Frequenz bewirkt z.B. einen Oktavsprung)
- Musiktheoretische Grundlagen (Welche Töne klingen für unsere Ohren gut, wenn sie gleichzeitig gespielt werden? Welche klingen dissonant? Was für Akkordtypen gibt es?)
- Rhythmus und Takte (Wie kann ich mit der gegebenen Anzahl an Ticks einen Rhythmus erzeugen? Wo sind die Grenzen)
- Stimmführung (Wie müssen zwei Melodien beschaffen sein, damit sie gemeinsam gut klingen? Welche Intervalle gilt es zu vermeiden und warum?)
- Stilkunde (Wie erzeuge ich ein mehrstimmiges Stück im barocken Stil? Wie im klassisch/romantischen Stil? Wie viele Calliopes brauche ich um einen Pop-Song zu erzeugen?)
- Kreativität und Ästhetik (Welche Melodien gefallen mir gut? Was für eine Melodie kann ich mir ausdenken?)

Hier kann sehr gut differenziert werden, je nachdem wie viel Vorwissen die Lerngruppe in Musik hat. Auch können hier Schülerinteressen (vorwiegend vermutlich der Pop-Song) oder die jeweils aktuelle Unterrichtseinheit im Fach Musik aufgegriffen werden, z.B. eine Unterrichtseinheit zum Thema Barock, Moderne, apparative Musikpraxis, Popmusik, Intervalle,…

Kognitive und inhaltliche Hürden

Die Einarbeitung in neue Hardware wie den Calliope Mini braucht natürlich Zeit. Ggf. müssen hier kurze Einführung und Übungen im grundsätzlichen Umgang mit dem Calliope vorbereitet werden. Selbiges gilt, wenn die Programmiersprache C++ vorher nie thematisiert wurde. Hier kann ggf. auch mit dem fertigen Code vorentlastet werden, besser wäre jedoch auch hier auf die relevanten Besonderheiten von C++ eingegangen werden, insbesondere im Unterschied zu der im Unterricht verwendeten Programmiersprache, wie z.B. Java. Auch aus musikalischer Sicht bringt dieses Projekt kognitive Hürden mit sich. Das Speichern der Noten im Code als ein Objekt steht in keinerlei Zusammenhang zu der üblichen, abendländischen Notenschrift auf den bekannten fünf Notenlinien. Insbesondere Instrumentalisten könnten sich hier schwer tun. Bei der Aufteilung der Gruppen kann darauf geachtet werden, dass nicht eine Gruppe ausschließlich aus Instrumentalisten besteht. Auch ist eine Note im Code aufgrund technischer Limitierungen grundsätzlich ausdrucksärmer als man es als Musiker vielleicht gewohnt ist. Dies soll heißen, dass Ausdruck und Artikulation derart minimiert sind, dass in diesem Bereich nicht viel Kreativität ausgelebt werden kann. Die Artikulation wird hier nur Unterschieden in Staccato und Legato, also kurzer Ton und langer Ton. Auch die Lautstärke kann nicht variiert werden! Piano-Passagen (leise Passagen), crescendi (lauter werden) und dergleichen sind nicht möglich. Die einzige Möglichkeit den Klang zu ändern, ist die Tonhöhe zu verändern. Einem passionierten Musiker kann dies zunächst sehr seltsam erscheinen. Auch ist es schwer, im Code Takte zu erkennen, es sei denn man macht entsprechende Absätze, was sich anbietet. Die Gesamtlänge des Stücks ist begrenzt durch den Arbeitsspeicher des Calliope, dieser umfasst hier mit dem gegebenen Programm knapp über 200 Noten. Ein Musikstück durch die absolute Anzahl an Noten zu begrenzen und nicht durch Takte oder Dauer ist zusätzlich ein kognitive Hürde.