Studienarbeit zum Thema

Physical Computing

für das Seminar

Informatik in der Bildung

gekürzte, modifizierte Fassung

für die Veröffentlichung

zur MNU 2019 in Hannover

eingereicht von
Manuel Riel

betreut von
Prof. Dr. Ralf Romeike

Stand sämtlicher Informationen: 18. September 2018

mit Ausnahme der Tabelle auf Seite 9

Thema der Arbeit

Erstellen von Arbeitsmaterial für eine „Kreativkiste“ für das Fach Informatik

Stichworte

Physical Computing, Calliope, microbit, Unterrichtsmaterial, Informatik für alle

Kurzbeschreibung

In dieser Seminararbeit wird eine Unterrichtsreihe für allgemeinbildende Informatik mit Mikrokontrollern, insbesondere dem Calliope mini bzw. dem BBC micro:bit, vorgestellt. Zunächst werden dazu fachdidaktische Hintergründe, insbesondere im Zusammenhang dem AP course „The Beauty and Joy of Computing“ sowie mit Physical Computing, ausgeführt. Abschließend erfolgt eine Reflektion der Kurserprobung mit Angabe von Problemen und künftigen Handlungsfeldern, die sich teils unerwartet während der Durchführung des Projektes ergaben.

Inhaltsverzeichnis

Themenbereiche der Lehrplaneinheit 3
Durchführung und Erfahrungen 5
a) Überblick über die gehaltenen Stunden 5
b) Implizite Nebenläufigkeit als Problemfall der Editoren 6
c) Exkurs: Weitere Vorteile von microBlocks 10
d) Beobachtungen aus der Unterrichtseinheit [gekürzt] 11
e) Exkurs: BBC micro:bit vs. Calliope mini 12
Schlussgedanken 13
Weiterführende Literatur 14

Danksagung

Besonderer Dank gebührt dem Physiklehrer Rudolf Pausenberger, der mich nicht nur bei der praktischen Unterrichtserprobung mit seiner Erfahrung unterstützte, sondern das fächerübergreifende Unterrichtsvorhaben ebenso um die physikdidaktische Dimension ergänzte. Durch seine Ermutigung ist diese Arbeit sowohl für Informatik- als auch für Physiklehrkräfte interessant.

Themenbereiche der Lehrplaneinheit

Bei fächerübergreifenden Unterrichtsvorhaben allgemein, insbesondere hier durch den Einsatz des Physical Computings verstärkt, sieht man sich mit einer Vielzahl verschiedener Themengebiete, Konzepten und Ideen konfrontiert, die es zu ordnen galt. Festgestellt konnte dabei werden, dass neben physikalischen und informatischen Fertigkeiten (mit Bildungsgehalt) auch stets in verschiedener Ausprägung technisch-handwerkliche Fähigkeiten benötigt werden, um Projekte zu gestalten.

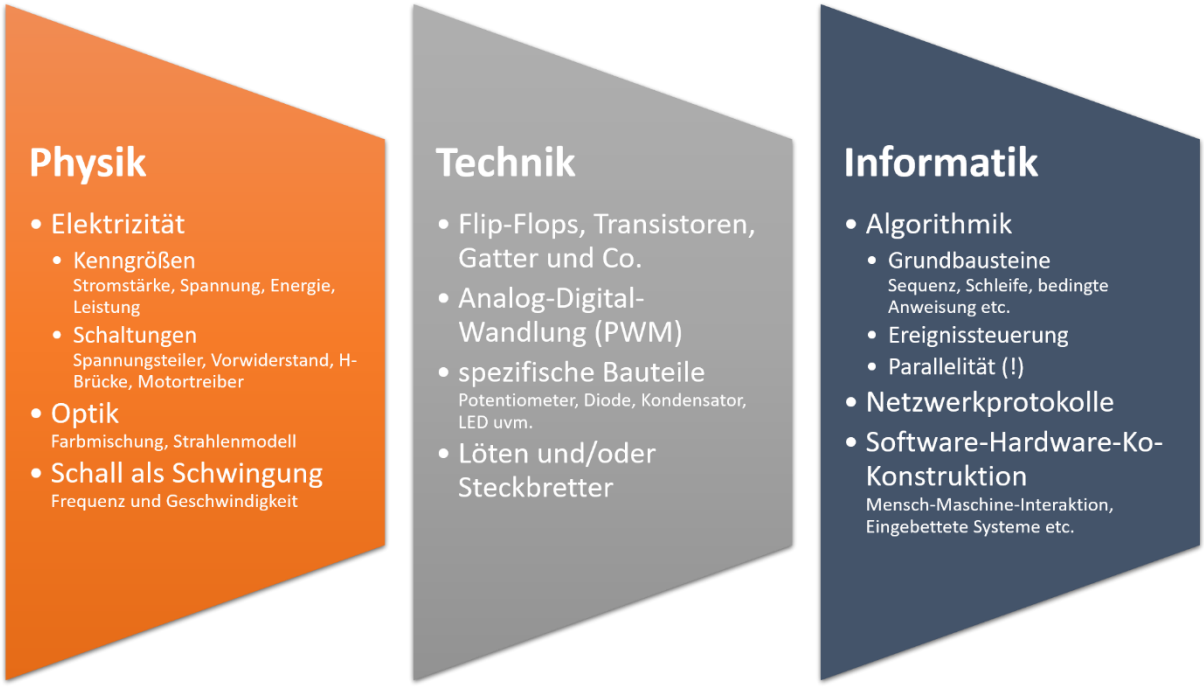


Abbildung 1: Übersicht über wichtige Themenbereiche als Art Mindmap.

Ergänzend zur Abbildung 1 seien außerdem noch zwei weitere Fachdisziplinen angesprochen, die in engen Bezug zum Unterrichtsprojekt stehen:

- **Biologie/Sinnesphysiologie** hinsichtlich subjektiver Farbwahrnehmung, Zapfen- und Stäbchen-Rezeptorverteilung im menschlichen Auge sowie dem Weber-Fechner-Gesetz
- **Mathematik** hinsichtlich der Mittelwertbildung (z. B. bei redundanter Temperaturmessung zur Verbesserung der Datenqualität), Skalentransformation bei Sensordaten sowie geometrische Kenntnisse für Raumkoordinaten und Dreiecksbetrachtungen (z. B. Lagesensor, der letztlich einen Vektor erfasst)

Bereits an dieser Stelle wird deutlich, wie viele Anknüpfungspunkte Physical Computing einerseits für fächerverbindende Unterrichtsgestaltung bietet – andererseits auch fordert, da etliche Kompetenzen für ein selbst entwickeltes Projekt benötigt werden.

Weiterhin sind größere Kontexte zur Einbettung der Einzelthemen – wie Embedded Systems im Bereich der Informatik (!) oder gesellschaftliche Aspekte – gar nicht dargestellt und würden die (Un-)Menge an Aspekten weiter augmentieren.

Durchführung und Erfahrungen

a) Überblick über die gehaltenen Stunden

Konkret wurde für den Unterricht eine ursprünglich mit sechs Unterrichtsstunden angesetzte Einführung in die Programmierung mit dem micro:bit[1] mit Microsofts blockbasierter Make-Code-Umgebung, die gleichermaßen für micro:bit und Calliope mini funktioniert, geplant. Als Abschluss der Unterrichtseinheit ist vorgesehen, dass die Schülerinnen und Schüler den zuvor begleitend entwickelten Farbmischer bestehend aus drei LEDs (RBG) ausbauen – zum Beispiel zu einem Philips-Hue ähnlichen System weiterentwickeln, Audioausgabe hinzufügen, Repeater-Funktionen (für die Funksteuerung) ergänzen etc. Die erworbenen Kenntnisse sind breit einsetzbar, die Realisierbarkeit von Schüler/innenprojekten hängt aber auch maßgeblich von den auf Seite 3 verfügbaren handwerklichen Kenntnissen bzw. der in der Schule vorrätigen Hardware ab.

Eine detaillierte Stundenübersicht soll an dieser Stelle nicht gegeben werden, sondern lässt sich anhand des entwickelten, beiliegenden Arbeitsmaterials gut nachvollziehen.

	Thema der Doppelstunde (DS)	**AB-Seiten**
1. DS	Einführung mit Pins, Diebstahlsicherung	1 bis 6
2. DS	RF-Funkmodul sowie Beschleunigungsvektor	7 bis 10
3. DS	Abschlussprojekt und eigene Ausbauoptionen	11 bis Ende

Tabelle 1: Grobe Stundenübersicht

[1] Der micro:bit kann mit nur minimalsten Anpassungen durch den Calliope mini ersetzt werden, dazu aber mehr auf Seite 19.

b) Implizite Nebenläufigkeit als Problemfall der Editoren

Bereits bei der Stundenvorbereitung fiel auf, dass der Mikrokontroller (micro:bit und Calliope mini gleichermaßen) mit dem MakeCode-Editor sich in vielen Situationen anders verhält als eigentlich vorgesehen – bis letztlich fehlende Synchronisations-Befehle als Verursacher dessen ausgemacht werden konnten. Dies stellt zum gegenwärtigen Zeitpunkt tatsächlich einen nicht-unerheblichen Kontrapunkt des Editors – aber auch der beiden Mikrokontroller dar, da es an geeigneten IDE-Alternativen mangelt.

Events ohne Synchronisation bei MakeCode

In MakeCode findet sich ein (prinzipiell lobenswerter) ereignisgesteuerter Ansatz wieder, um implizite Ereignisse zu ermöglichen – scheinbar benötigt man kein gesondertes Wissen über Parallelität und Wettlaufsituationen um Ressourcen. Tatsächlich führt dies aber schnell zu Problemen, denn bereits die Anforderung dauerhaft etwas im Display, der LED-Matrix, anzuzeigen und bei Eingabe (z. B. durch Knopfdruck) etwas anderes im Display auszugeben wird dadurch nur mit Tricks, die keinem guten Programmierstil entsprechen, möglich.

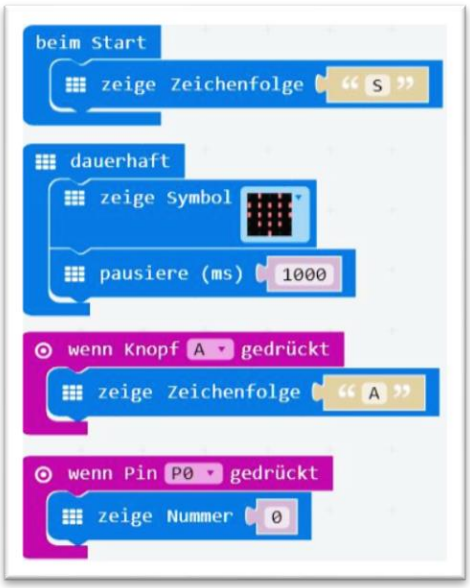


Abbildung 2: Screenshot eines MakeCode-Programmes mit „Phänomenen".

Konkret verleitete die Kategorie „Grundlagen" insbesondere die Schülerinnen dazu, hier amüsante, lange (!) Folgen von Symbolen in der dauerhaft-Schleife anzulegen, mit dem Ergebnis, dass diese durch ein Drücken eines Knopfes nicht mehr abgebrochen werden konnten, sondern meist nur noch die dauerhaft-Schleife ausgeführt wird. Das Verhalten ist inkonsistent, die Wahrscheinlichkeit, dass das Knopf-Ereignis ausgeführt wird, sinkt augenscheinlich mit zunehmender Länge des dauerhaft-Programms. Vermutlich verhält es sich so, dass die „zeige Symbol"-Anweisungen atomar sind, d. h. nicht unterbrochen werden können und ein Tastendruck nur registriert wird, wenn er zeitlich „zwischen" diesen ausgeführt wird. Derartige Spekulationen sind deshalb nötig, weil sowohl eine ausführliche Dokumentation der IDE als auch offizielle Programmbeispiele fehlen.

Die obenstehende Abbildung 2 zeigt eine Möglichkeit die beschriebene Situation zu verschlimmbessern: Das Einfügen von pausiere-Befehlen in der dauerhaft-Schleife ermöglicht, dass Eingaben von Pin (oder auch Knopf) häufiger, aber immer noch inkonsistent, registriert werden. Weiterhin zeigt die Abbildung ein weiteres „Phänomen" auf: Die Zeichenfolge „A" wird immer (!) verschluckt und das Display flimmert nur kurz auf; im Gegensatz dazu die Zahl (schlecht übersetzt: „Nummer") 0: Sie leuchtet, nachdem Pin 0 mit GND kontaktiert (schlecht übersetzt „gedrückt") wird …. Dies liegt nicht, wie man vermuten könnte, daran, dass das Ereignis vom Knopfdruck statt vom Pin ausgeht, sondern, dass der zeige-Zeichenfolge-Befehl anders gehandhabt wird: Offenbar haben diese Display-Ausgaben eine implizite, atomare Dauer, die bei zeige-Zeichenfolge mit nur einem Buchstaben zu kurz ausfällt.

Die Ausführungen verdeutlichen, wie dringend Befehle zur Synchronisation benötigt werden, damit ein konsistentes, nachvollziehbares Systemverhalten gewährleistet wird. MakeCode bietet dazu aber lediglich die unter „Fortgeschritten – Steuerung“ versteckte Anweisung im-Hintergrund-ausführen an: Diese kehrt nicht mehr zurück, heißt, wird das Hintergrundprogramm einmal durch z. B. eine Eingabe unterbrochen, wird es nicht wieder ausgeführt – es fehlt eine Art broadcast-Befehl[2]! Als Workaround kann man daher nur die Sequenz des Hintergrundprogrammes an alle Eingabeblöcke anhängen, was Änderungen im Hintergrundprogramm zur fehleranfälligen und redundanten Arbeit macht. Ein funktionsfähiges Programm zu Abbildung 2 sieht wie folgt aus:

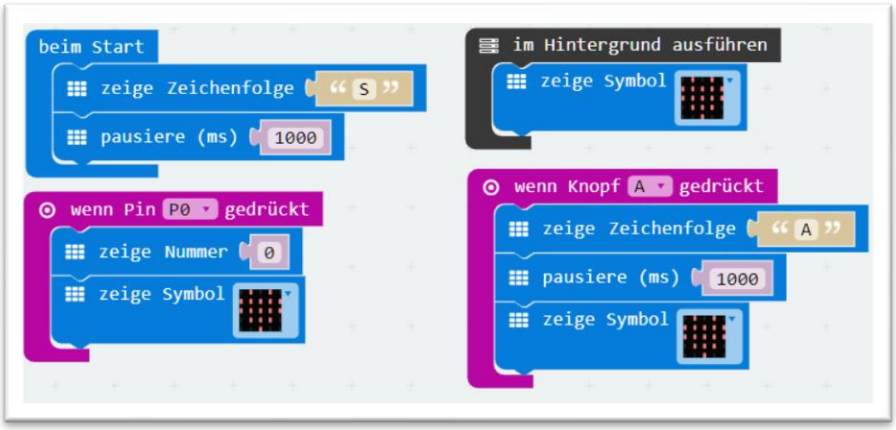


Abbildung 3: Screenshot von MakeCode – Lösung des Problems?!

Es galt dabei neben oben genanntem Workaround außerdem noch zu berücksichtigen, dass die zeige-Zeichenfolge-Anweisungen mittels pausiere-Befehl „verlängert“ werden mussten, damit sie nicht verschluckt werden.

Subsumierend bleibt festzuhalten, dass die Nebenläufigkeit der Ereignisse und die implizite atomare Dauer der sofort auffindbaren Anweisungen der beiden oberen Kategorien („Eingabe“ und „Grundlagen“) schnell zu Verwirrung führen, der einzige und zugleich zu sinnvoller Programmierung nicht ausreichende Befehl zur Synchronisation aber weit unten in der Oberfläche versteckt bleibt – eine schlichtweg verheerende Situation!

Dabei ist laut Ball et al. (2018) die technische Grundlage für MakeCode, die komponentenbasierte CODAL-C++-Laufzeitumgebung, durchaus in der Lage synchrone und asynchrone Events umsetzten. Die Aussage *„MakeCode is the primary editor for the BBC micro:bit“* (ebd.) verwundert umso mehr, ob der Probleme, die wir beim Umsetzen einer Unterrichtsreihe damit hatten. Keineswegs besser ist die Editorenauswahl für den Calliope mini, da es sich weitestgehend um dieselben des micro:bits handelt. Zu erwähnen ist der bekannte Open Roberta Lab Editor, welcher aber letztlich nur das Schreiben eines sequentiellen Programmes ermöglicht und keinerlei Ereignissteuerung bietet. Ferner fehlt hier speziell für den angedachten Anwendungsfall des ferngesteuerten Farbmischers, die Möglichkeit beim Senden von Zahlwerten einen beschreibenden String mitzusenden.[3]

[2] Was genau damit gemeint ist, wird noch auf Seite 15 ausgeführt.

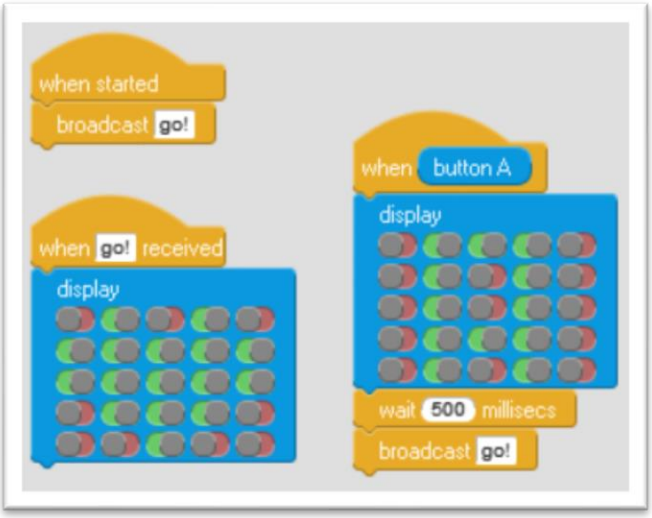
[3] Ein ausführlicher, tabellarischer Vergleich von Open Roberta Lab mit Microsofts MakeCode-Umgebung findet sich im Anhang der Zulassungsarbeit von Köck (2018). Er selbst verwendet ebenfalls den MakeCode-Editor für weitere Projekte.

microBlocks als potentielle Alternative

Selbst zusammen mit dem Experten für visuelle Programmiersprachen an der FAU, Sven Jatzlau, konnte keine befriedigende Lösung dafür gefunden werden: Prinzipiell sei das ereignisbasierte Programmierkonzept von MakeCode durchaus sinnvoll, die Umsetzung aber nicht brauchbar. Daher wurde ich auf das sich im Alpha-Stadium befindliche Projekt microBlocks (geeignet für Calliope mini, micro:bit und viele mehr) der Koryphäen Jens Mönig (Snap!) und Bernat Romagosa (BeetleBlocks) aufmerksam gemacht.

Dieses gewährleistet einen definierten Programmablauf durch folgende Maßnahmen:

- Anbieten ausschließlich **atomarer Display-Befehle** zunächst **ohne (implizite) Dauer**
- **Explizites Clear des Displays**: Displayausgaben bleiben solange bestehen, bis sie explizit per Befehl gelöscht werden.
- Die **Dauer von Display-Anzeigen** wird durch den **wait-Befehl** (ermöglicht durch beide obige Punkte) umgesetzt.
- **broadcast-Anweisung zum „(Zurück-) Springen“** zu kontinuierlicher (wiederhole dauerhaft in MakeCode) Sequenz



Ein Programm in microBlocks, welches ähnlich zu Abbildung 3 ist, ist nebenstehend realisiert.

Abbildung 4: Screenshot von Programm in microBlocks.

microBlocks ist schon wegen des Alleinstellungsmerkmals der gelungenen Umsetzung von Synchronisationsmöglichkeiten eine überaus interessante Programmierumgebung für den micro:bit und Calliope mini! Weiteres Alleinstellungsmerkmal ist außerdem, dass ein angeschlossener Mikrokontroller in Echtzeit ein neues Programm ausführen kann – kein Umständliches Downloaden des kompilierten Programmes und anschließendes Drag-and-Drop sind mehr, wie bei MakeCode und Open Roberta Lab, nötig!

Didaktische Umsetzung in der Unterrichtseinheit

Nach den bisher getätigten Ausführungen wird deutlich, dass es zur Zeit keine optimale Lösung der Nebenläufigkeitsproblematik bei Erhalt aller Funktionen gibt – daher wurde auch für das vorliegende Projekt wieder die MakeCode-Umgebung verwendet, allerdings werden die Schwächen dieser direkt in den ersten Stunden, direkt zu Beginn explizit besprochen und spiegeln sich auch in dem erstellen Arbeitsmaterial wieder. Motivational ist so ein Einstieg nicht,

dennoch erscheint es mir als beste Option gleich zu Beginn auf die Defizite hinzuweisen, da die beschriebenen Inkonsistenzen zum Einen schnell mit den „Grundlagen“- und „Eingabe“-Anweisungen herbeigeführt werden können und zum Anderen, damit die Schülerinnen und Schüler später auch selbstständig ein Projekt bearbeiten können; nichtsdestotrotz werden die informatischen Hintergründe nicht reflektiert – es handelt sich schließlich um die 9. Jahrgangsstufe mit beschränktem Zeitkontingent – so dass es sich um eine Art „best“ practices handelt, die jedoch ohne Hintergrundwissen hier nur Mittel zum Zwecke des eigenständigen Arbeitens sind. Ein besonderer Bildungsgehalt ist ihnen nicht zuzuschreiben, da es sich nur um Anwendungs-, nicht aber um Konzeptwissen handelt. Sollte es sich daher bei einer Wiederholung der Unterrichtsreihe anbieten, könnte auch explizit auf Nebenläufigkeit und Wettlaufsituationen eingegangen werden – das benötigte Wissen ist allerdings recht umfangreich!

Kategorie/Editor	**MakeCode** primärer Editor von Microsoft	**Open Roberta Lab**	**microBlocks** (Alpha)
Programmier-Paradigma	ereignisbasiert	sequentiell	ereignisbasiert
Programm-Ausführung	direkt im Editor über **Simulator**	nur per **copy-and-paste** auf dem verbundenen Controller	**in Echtzeit** auf dem angeschlossenen Mikrocontroller
Nebenläufigkeit	umständlicher Work-around mittels warte- und im-Hintergrund-Anweisung	kaum möglich, da sequentiell!	klar kontrollierbar ☺
RF-Funk	gut zum Versenden von Zahlwerten mit Label zur Identifikation	nur Strings versendbar	inzwischen ebenfalls implementiert
Bedienbarkeit und Komfort-funktionen	sehr gut, viele Spielereien (Display-Symbole) vorhanden	ausgereift, aber weniger umfassend als MakeCode	noch relativ spartanisch, da im Alpha-Stadium

Tabelle 2: Zusammenfassender Kurzvergleich der Editoren mit subjektiver didaktischer Wertung als Farbcodierung. ***Überarbeitet im März 2019.***

c) Exkurs: Weitere Vorteile von microBlocks

Simulieren von Pulsweitenmodulation (PWM)

Insbesondere, wenn in Kooperation mit dem Physikunterricht Pulsweitenmodulation behandelt werden soll, zeigt microBlocks wieder Vorteile gegenüber der Konkurrenz. Auf den analogen Pins funktioniert – so auch beim Farbmischer – die PWM durchaus, aber über digitale Pins lässt sich diese nicht simulieren: Die Schleifen werden zu langsam ausgeführt, die LEDs flackern; Töne lassen sich per angeschlossenen Summer so nicht selbst erzeugen, sondern nur über die gesonderten Befehle der Kategorie „Musik“.

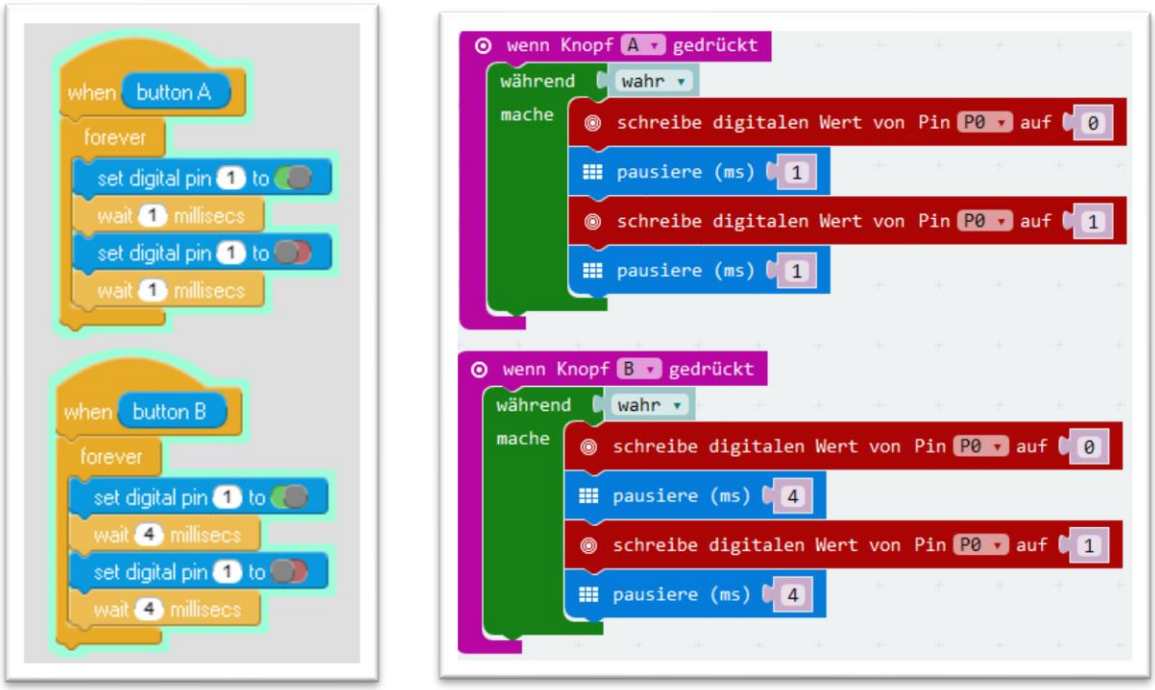


Abbildung 5: Nach einem Reset sollte Knop A einen um zwei Oktaven höheren Ton auf dem angeschlossenen Summer erzeugen als Knopf B. MakeCode (rechts) führt die Befehle dafür aber viel zu langsam aus. microBlocks hingegen funktioniert wie intendiert.

Generell werde nach Aussage der Entwickler bei microBlocks auf die Performanz höchster Wert gelegt; dies ist auch in der Praxis durchaus vorteilhaft, da beim Tüfteln und Experimentieren die geringere Abfragefrequenz der Pins bei MakeCode nicht nur bei der PWM stören kann.

Echtzeitausführung von Programmen – ohne manuelles Kopieren

Weiterhin sei erwähnt, dass microBlocks zwar keinen Simulator besitzt, sondern – dies als großes Alleinstellungsmerkmal! – das Programm direkt auf dem Mikrokontroller ausführt: Das sonst stets nötige und auf Dauer überaus lästige Kopieren aus dem Downloadordner auf den Kontroller entfällt! Das so übertragene Programm bleibt dennoch auch beim Abstecken des Mikrokontrollers vom USB-Anschluss und späteren Anschließen an eine andere Stromquelle erhalten.

d) Beobachtungen aus der Unterrichtseinheit [gekürzt]

Viele Themenaspekte, ambitionierte Lernziele – und wenig Zeit!

Es bietet sich an, eventuell auch nur zum Einstieg, Potentiometer zum Ansteuern der LEDs zu verwenden – es muss allerdings berücksichtigt bleiben, dass die Anschlussmöglichkeiten sowohl beim micro:bit als auch beim Calliope ohne Steckbrett oder ähnliches begrenzt sind (vgl. auch Köck (2018), Seite 13ff). Durch ein getrenntes Ansteuern wird die Zusammensetzung der Farben aus rotem, gelbem und blauem Licht deutlicher – und stellt somit sowohl sinnesphysiologisch als auch informatisch (Kodierung) einen Bildungsinhalt klarer dar.

Insgesamt ist daher eher ein Zeitrahmen von zehn statt sechs Unterrichtsstunden, insbesondere auch für ein eigenes Projekt[4], zu empfehlen.

[4] Die auf Seite 17 getätigten Ausführungen bestätigen, dass ein eigenes, persönlich bedeutsames, kurz konstruktionistisches Projekt motivationsförderlich ist – und dies im knappen Zeitrahmen trotz der Betonung im Unterrichtsgespräch als Ausblick zu kurz kam.

e) Exkurs: BBC micro:bit vs. Calliope mini

Der Calliope mini ist (in Deutschland) in vieler Munde! Im Zuge dieser Arbeit konnte ein rudimentärer Vergleich zwischen den beiden Mikrokontrollern gezogen werden – zunächst mit dem überraschenden Zwischenfazit, dass die Unterschiede zwischen beiden – verstärkt durch dieselben (!) Programmierumgebungen und damit einhergehend dieselben Probleme dieser – gering sind. Der Calliope mini bietet dabei etwas mehr on-board-Hardware als der micro:bit; darunterfallen:

- Die **Multicolor-LED** auf dem Board, die direkt (statt dreier einzelner, externer LEDs) verwendet werden kann. Diese ist überraschend leuchtstark und in der angedachten Unterrichtseinheit durchaus vorteilhaft, lässt aber weniger erkennen, dass sich die Lichtfarbe aus RGB zusammensetzt. Externe LEDs sind für ein System, das hauptsächlich Stimmungslicht ist, dennoch geeigneter – die interne Multicolor-LED ist aber willkommenes Zusatzfeature.

Abbildung 6: Die Multicolor-LED des Calliope lässt sich mit MakeCode einfach ansteuern.

- Der integrierte **Piezo-Summer** funktioniert rudimentär für Töne, d. h. zum Beispiel für ein (leises) Alarmsignal durchaus praktikabel. Um ein Stimmungslicht mit Soundeffekten zu programmieren aber, ist die Wiedergabequalität deutlich zu gering.

- Ein rudimentäres **Mikrofon**, das auf jeden Fall für die Erkennung von Lautstärke (z. B. bei einem Einbruch) geeignet ist. Ob die Aufnahmeauflösung für Spracherkennung ausreicht, ist hingegen fraglich.

- Zwei **I²C-Anschlüsse**, die (eingeschränkt?[5]) kompatibel zum vom Arduino bekannten **Grove-Kit** sind. Für das vorliegende Projekt waren die Grove-Multicolor-LED als auch die Grove-LED-Bar (die außerhalb des regulären Grove-Kits vertrieben werden) nicht über das in MakeCode hinzufügbare Grove-Packet ansteuerbar.

Fazit zum Vergleich

Insofern ist Calliope mini (samt bemerkenswerter sechseckiger Platine) eine gewisse Erweiterung des micro:bits – der dennoch nicht im Unterrichtsversuch eingesetzt wurde: Für die meisten Schulen spielt das Budget für Unterrichtsgegenstände durchaus eine Rolle und während der Calliope mini mit circa 35 € relativ teuer ist, so ist der BBC micro:bit samt Batterie-Box auch in Deutschland für Privatpersonen für etwas weniger als die Hälfte (!) dieses Preises, circa 17 €, erhältlich. Die zusätzlich im Calliope mini verbaute Hardware kann diesen Aufpreis nur schwerlich rechtfertigen: Die Einzelkomponenten sind an sich günstig und der Zusatznutzen in vielen Fällen vernachlässigbar.

[5] Laut Köck (2018) funktioniert das Grove-Kit in den meisten Fällen wie vorgesehen, auch wenn die Spannung, die der Calliope mini liefert, eigentlich nicht den Anforderungen entspricht.

Schlussgedanken

Die Vorteile der integrierten on-board-Hardware von micro:bit und Calliope mini sind direkt im Unterrichtsgeschehen spürbar[6] und bieten laut Alex Köck (2018, S. 33) auch monetäre Vorteile gegenüber der Anschaffung einzelner Komponenten, wie dies beim Arduino erforderlich ist. So zeichnet er abschließend ein positives Bild vom Calliope mini – das hier deutlich eingeschränkt werden muss.

Mareen Przybylla (2017a) schreibt über ihr eigens entwickeltes Arduino-Kit My Interactive Garden:

> *„[…] MyIG is strongly orientied towards the ideal of physical computing:*
>
> ***focus on ideas not on restrictions of tools.“***

Genau das Gegenteil trifft jedoch – ebenso in der durchgeführten Unterrichtseinheit in Form von „best“ practices – für die beiden Mikrokontroller mit integrierter Hardware zu! Unter „tools“ im obigen Zitat war vermutlich eher die Hardware-Komponente von Physical Computing gemeint, umso unbegreiflicher ist es, dass die etablierten Programmierumgebungen für micro:bit und Calliope mini in ihren Software-Kontrollstrukturen – und das schon bei basalen Anforderungen! – derartige Defizite bei der inhärenten Nebenläufigkeit aufweisen.

Pointiert formuliert trifft man gegenwärtig bei jenen beiden Mikrokontrollern also nicht nur auf die zu erwartenden Hardware-Einschränkungen, sondern ebenso auf unerwartete Software-Probleme: Im gegenwärtigen Zustand können die beiden Mikrokontroller nur mit starker Einschränkung weiterempfohlen werden.

Nichtsdestotrotz zeigt das seit Juli 2018 in der öffentlichen Alpha-Phase befindliche microBlocks, wie man das Potential von Calliope mini und micro:bit ausschöpfen könnte. In Hinblick auf die Zukunft könnten die beiden Mikrokontroller für den Unterricht überaus gewinnbringend sein, denn: Der Ansatz integrierter Hardware ist gelungen – die Software dazu aber gegenwärtig nicht.

[6] Insbesondere das Display – eine bei beiden Mikrokontrollern bemerkenswert identische 5x5-Matrix roter LEDs – stellt ein Alleinstellungsmerkmal dar: Unkompliziert sind hier von der ersten Unterrichtsstunde Zeichen, Emoticons und (bei MakeCode) sogar Laufschrift problemlos als Ausgabe verfügbar. Dies motivierte alle Schülerinnen und Schüler ungemein.

Weiterführende Literatur

Ball, Thomas; Bishop, Judith; Finney, Joe (2018): Multi-platform computing for physical devices via MakeCode and CODAL. In: Ivica Crnkovic, Michel Chaudron, Marsha Chechik und Mark Harman (Hg.): Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18. the 40th International Conference. Gothenburg, Sweden, 27.05.2018 - 03.06.2018. New York, New York, USA: ACM Press, S. 552–553.

Berkeley University: BEAUTY AND JOY OF COMPUTING - BJC. Unit 3: Data Processing and Lists. Online verfügbar unter https://bjc.edc.org/bjc-r/topic/topic.html?topic=nyc_bjc/3-lists.topic&course=bjc4nyc.html&novideo&noassignment, zuletzt geprüft am 11.09.2018.

Brian Harvey (2012): The Beauty and Joy of Computing. Computer Science for Everyone. Computer Science Devision, University of California, Berkeley. Athen (Griechenland) (Constructionism 2012). Online verfügbar unter https://bjc.berkeley.edu/documents/2012%20Constructionism%20-%20The%20Beauty%20and%20Joy%20of%20Computing%20-%20Computer%20Science%20for%20Everyone.pdf, zuletzt geprüft am 11.09.2018.

Conradi, Bettina; Hommer, Martin; Kowalski, Robert (2010): From digital to physical. In: Antonio Krüger, Johannes Schöning, Daniel Wigdor und Michael Haller (Hg.): ACM International Conference on Interactive Tabletops and Surfaces - ITS '10. ACM International Conference. Saarbrücken, Germany, 07.11.2010 - 10.11.2010. New York, New York, USA: ACM Press, S. 249.

Cornelsen-Verlag: Coding in der Schule - mit Calliope mini. Schon für Grundschulen ab Klasse 3. Online verfügbar unter https://www.cornelsen.de/calliope/, zuletzt geprüft am 10.09.2018.

Garcia, Dan; Harvey, Brian; Barnes, Tiffany (2015): The beauty and joy of computing. In: *ACM Inroads* 6 (4), S. 71–79. DOI: 10.1145/2835184.

Jin, Karen H.; Haynie, Kathleen; Kearns, Gavin (2016): Teaching Elementary Students Programming in a Physical Computing Classroom. In: Deborah Boisvert und Stephen Zilora (Hg.): Proceedings of the 17th Annual Conference on Information Technology Education - SIGITE '16. the 17th Annual Conference. Boston, Massachusetts, USA, 28.09.2016 - 01.10.2016. New York, New York, USA: ACM Press, S. 85–90.

Köck, Alexander (2018): Physical Computing mit dem Calliope. Zulassungsarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen. Didaktik der Informatik.

Kultusministerium Bayern: G8 Lehrplan. Physik 9. Klasse. Hg. v. ISB Bayern. Online verfügbar unter http://www.isb-gym8-lehrplan.de/contentserv/3.1.neu/g8.de/index.php?StoryID=26438, zuletzt geprüft am 13.09.2018.

Price, Thomas W.; Cateté, Veronica; Albert, Jennifer; Barnes, Tiffany; Garcia, Daniel D. (2016): Lessons Learned from "BJC" CS Principles Professional Development. In: Carl Alphonce, Jodi Tims, Michael Caspersen und Stephen Edwards (Hg.): Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16. the 47th ACM Technical Symposium. Memphis, Tennessee, USA, 02.03.2016 - 05.03.2016. New York, New York, USA: ACM Press, S. 467–472.

Przybylla, Mareen; Romeike, Ralf (2012): My Interactive Garden – A Constructionist Approach to Creative Learning with Interactive Installations in Computing Education.

Przybylla, Mareen; Romeike, Ralf (2014): Physical computing in computer science education. In: Carsten Schulte, Michael E. Caspersen und Judith Gal-Ezer (Hg.): Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE '14. the 9th Workshop in Primary and Secondary Computing Education. Berlin, Germany, 05.11.2014 - 07.11.2014. New York, New York, USA: ACM Press, S. 136–137.

Przybylla, Mareen; Romeike, Ralf (2017a): The nature of physical computing in schools. In: Calkin Suero Montero und Mike Joy (Hg.): Proceedings of the 17th Koli Calling Conference on Computing Education Research - Koli Calling '17. the 17th Koli Calling Conference. Koli, Finland, 16.11.2017 - 19.11.2017. New York, New York, USA: ACM Press, S. 98–107.

Przybylla, Mareen; Romeike, Ralf (2017b): Von Eingebetteten Systemen zu Physical Computing: Grundlagen für Informatikunterricht in der digitalen Welt Proceedings der 17. GI-Fachtagung Informatik und Schule, INFOS 2017, 13.-15. September 2017, Oldenburg. In: Ira Diethelm (Hg.): Informatische Bildung zum Verstehen und Gestalten der digitalen Welt, Proceedings der 17. GI-Fachtagung Informatik und Schule, INFOS 2017, 13.-15. September 2017, Oldenburg (LNI, P-274), S. 257–266. Online verfügbar unter https://dl.gi.de/20.500.12116/4322.