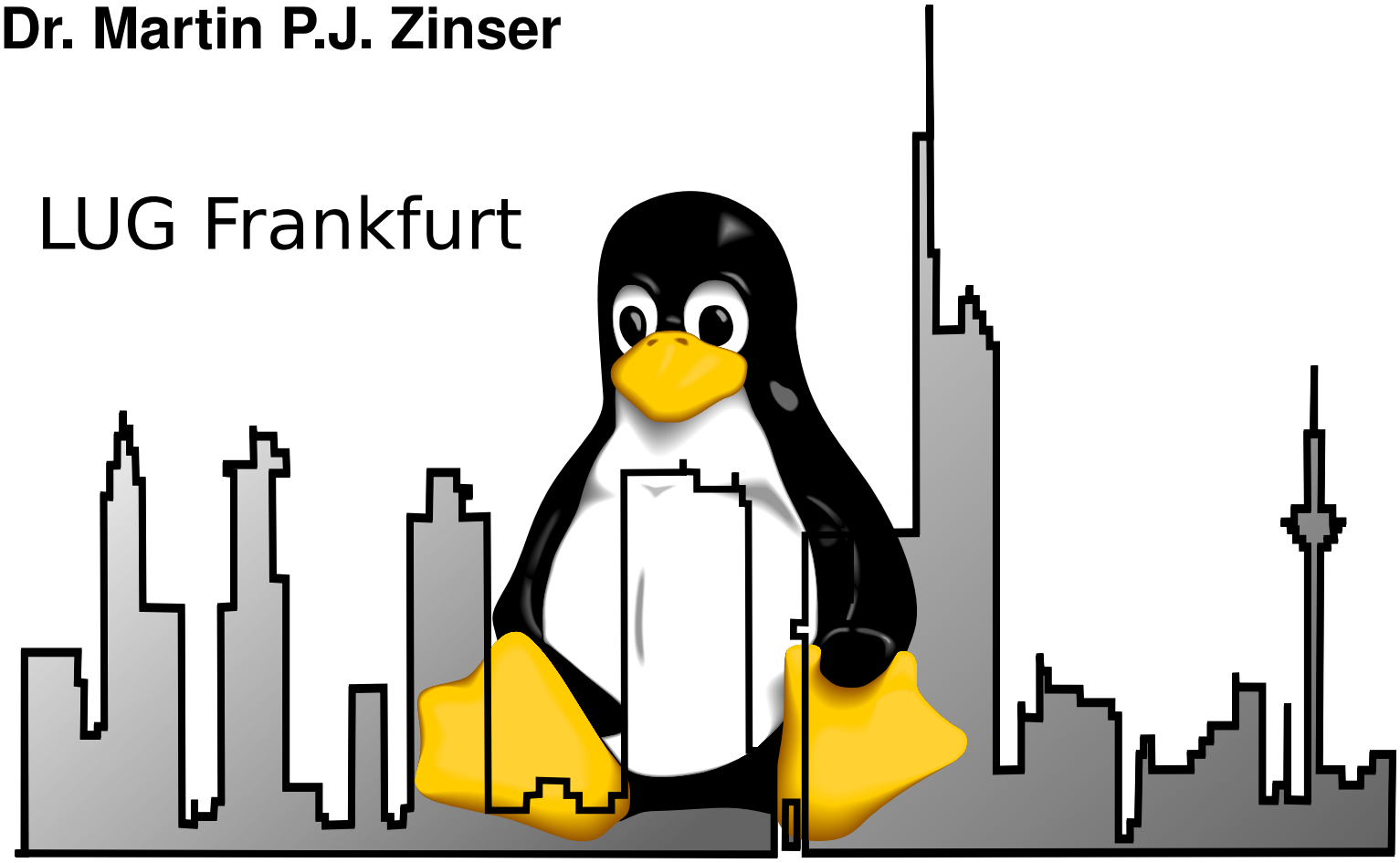
Coding for Kids with Simple Robots and Micro Controllers

... and how to do it with Linux

Dr. Sylvia Zinser
Dr. Martin P.J. Zinser



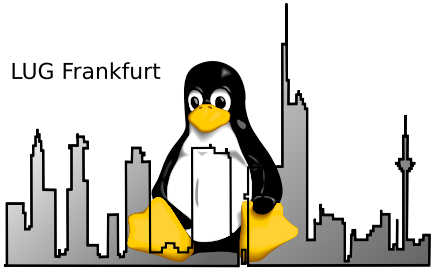


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

Calliope

B·O·B·3

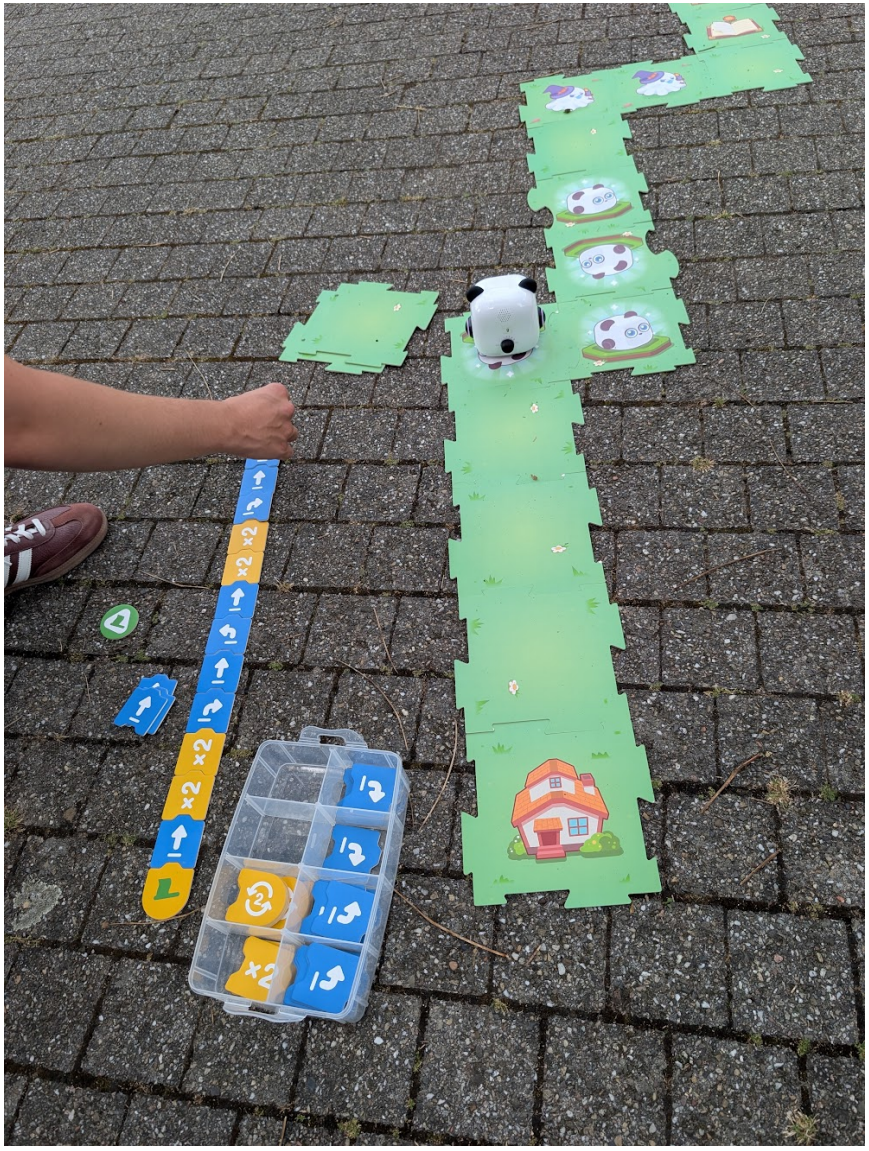
Edison

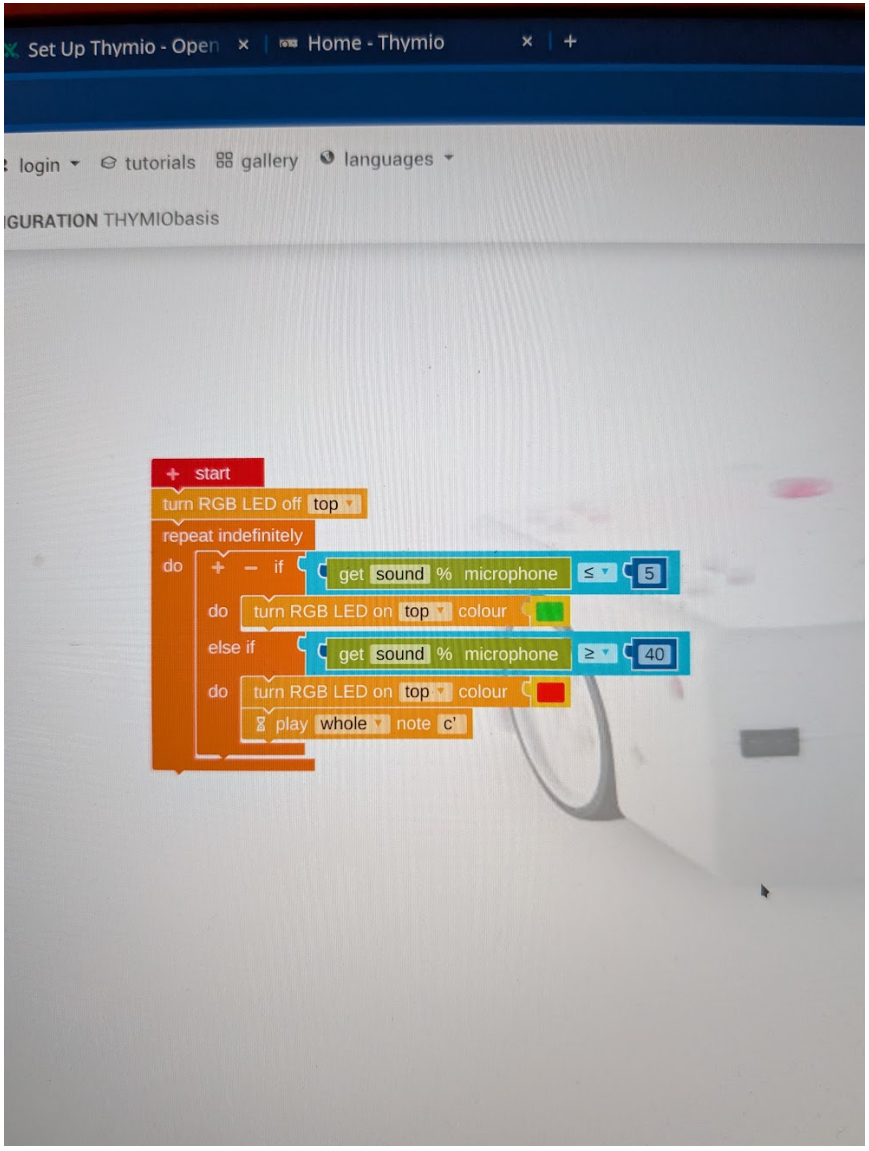
Sphero

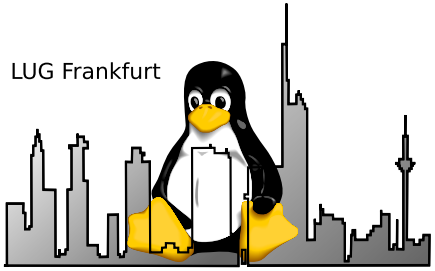
Cost

LUG Frankfurt

What do I want to teach?

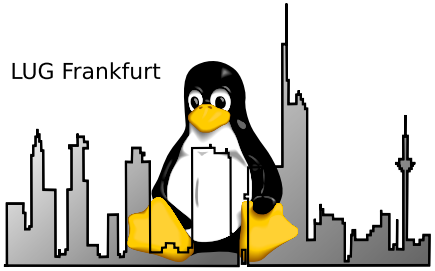






Why Robots

- Natural differentiation
- Math
- Group skills
- Hey Mom, I can program!!
- Early vocab
- All the subjects
- Problem Solving



Which Robots/MicroControllers

- Calliope Mini
- BBC MicroBit
- Edison
- BOB3
- Sphero
- Others like LEGO, Thymio, ...

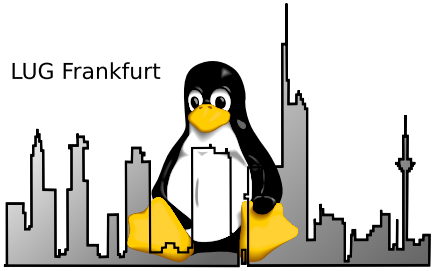


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

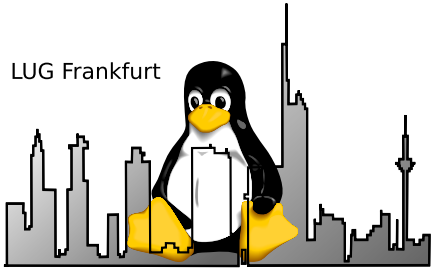
Calliope

B·O·B·3

Edison

Sphero

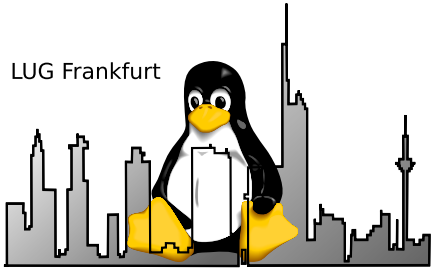
Cost



What can my robot/microcontroller do?

The first steps into programming is the understanding that any robot does exactly as it’s told.

Which is not necessarily what the programmer wants.



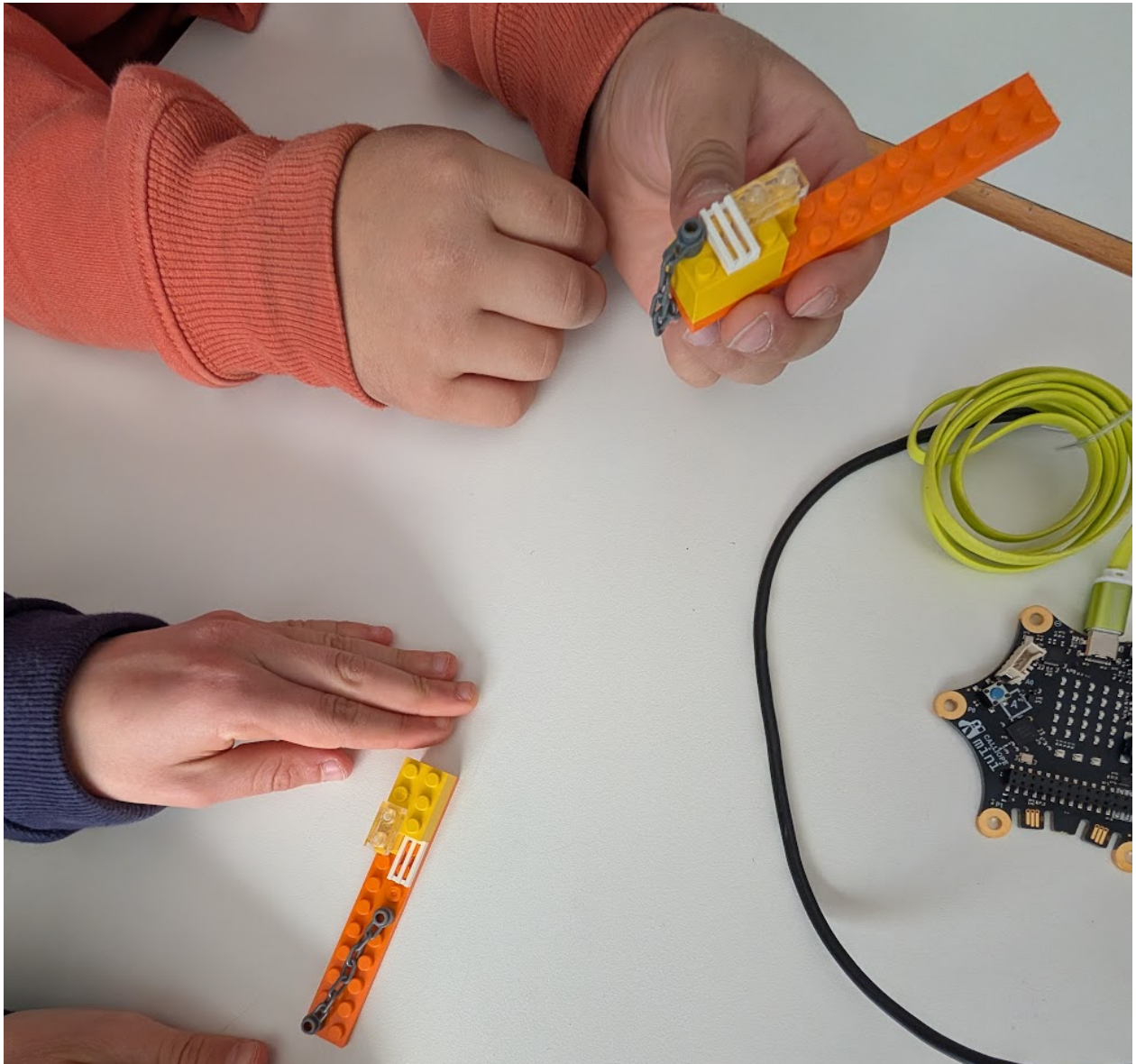
Offline exercise

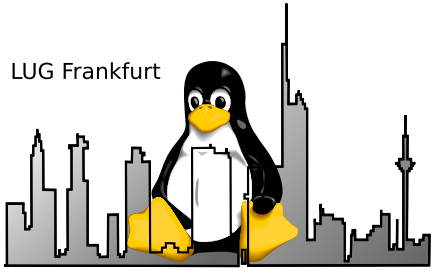
What is this thing called programming



LUG Frankfurt

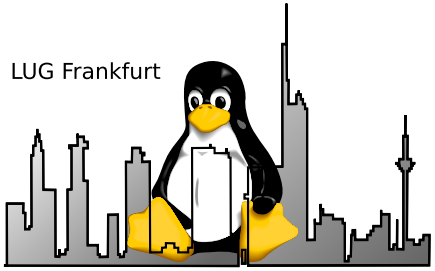
Offline exercise





What can my robot/microcontroller "feel"?

- Temperature
- Color
- Light intensity
- Acceleration
- Distance
- Touch
- Noise
- Broadcast



What can my robot/microcontroller do?

- Light
- Display
- Broadcast
- Sound
- Motors

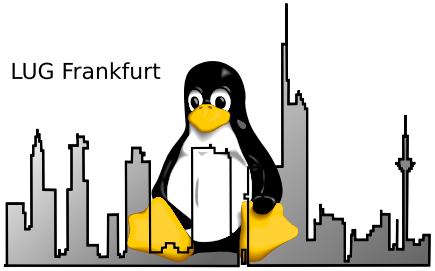


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

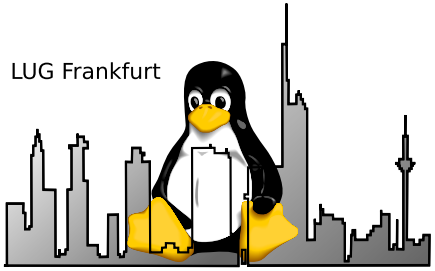
Calliope

B·O·B·3

Edison

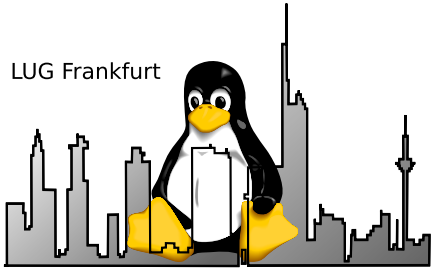
Sphero

Cost



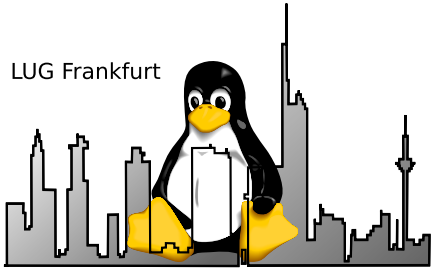
The Code

- MakeCode
- OpenRoberta
- Proprietary block languages



Make it Blocks

- BOB3: `https://primary-blocks.org/ide.html`
- Calliope Code: `https://makecode.calliope.cc/`
- Microbit Code: `https://makecode.microbit.org/`
- Edison scratch Code: `https://www.edscratchapp.com/v3/`
- (TinkerCAD)



Make it Text

- BOB3: `https://www.progbob.org/`
- Calliope Code: `https://makecode.calliope.cc/`
- Microbit Code: `https://makecode.microbit.org/`
- Edison python Code: `https://www.edpyapp.com/v3/`

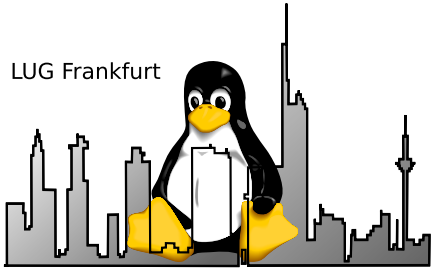


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

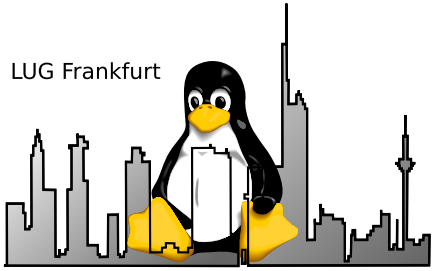
Calliope

B·O·B·3

Edison

Sphero

Cost

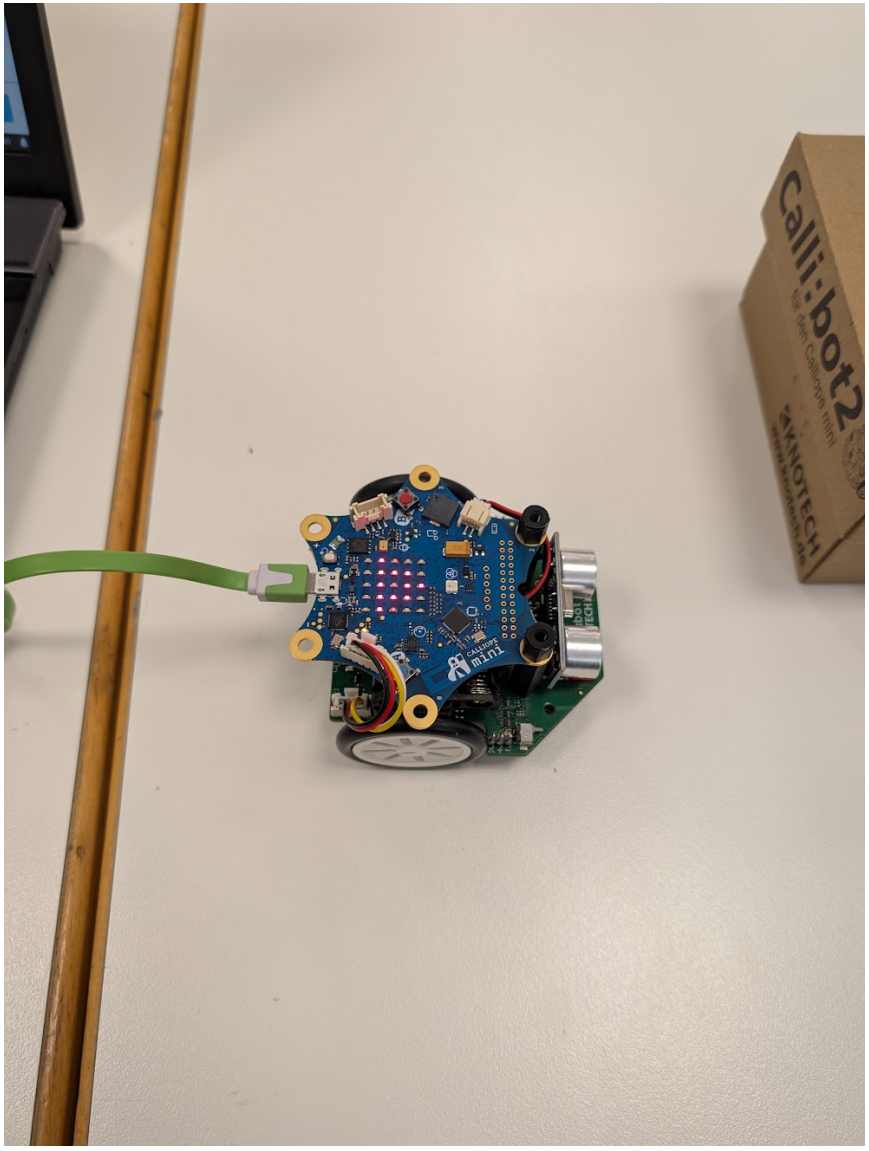


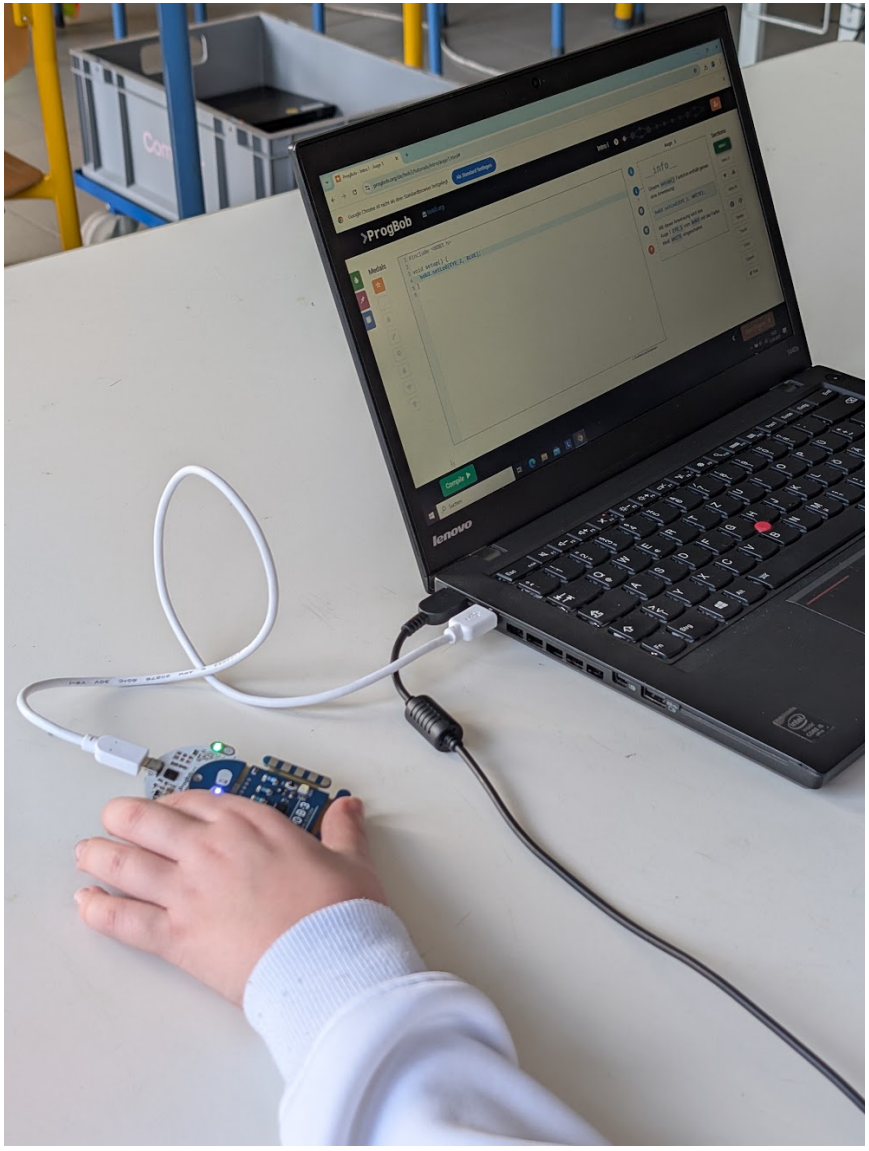
Microcontroller activities

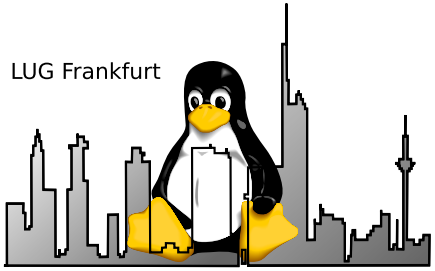
- Blink
- Light sensor for exploring the room
- Physics (Temperature, external sensor, errors)
- Music
- Light shows
- Inventions (Build stuff using sensors and actors)

LUG Frankfurt

Busy processors





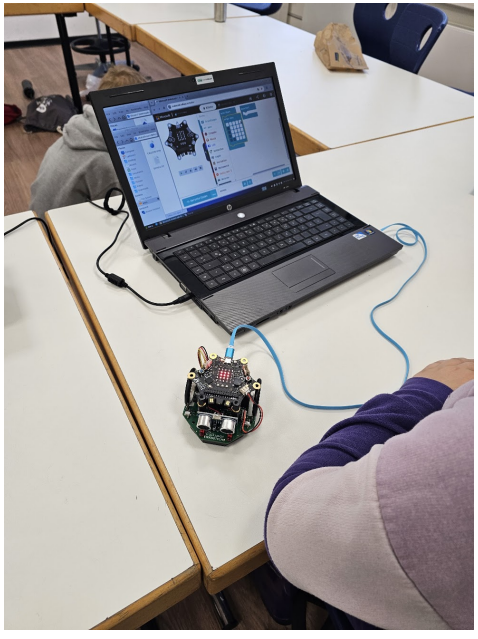


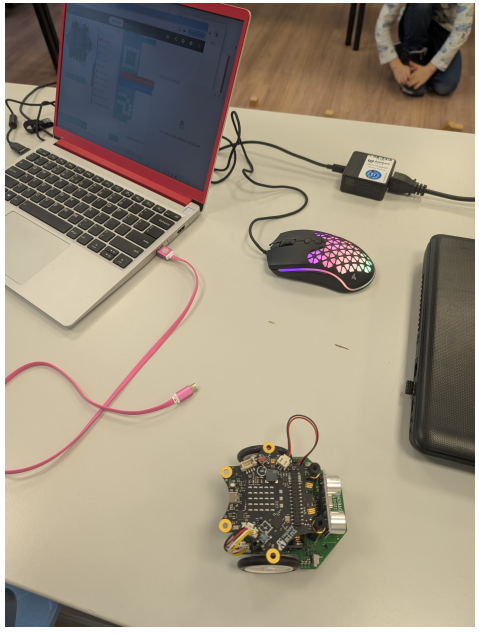
Get your robot busy

- Just drive (No remote control)
- Maps and Mazes
- Physics (inclines, radii, power and speed)
- Physics (use sensors)
- Stories
- Races (speed is irrelevant)
- Inventions (Build stuff using sensors and actors)

LUG Frankfurt

Busy robots





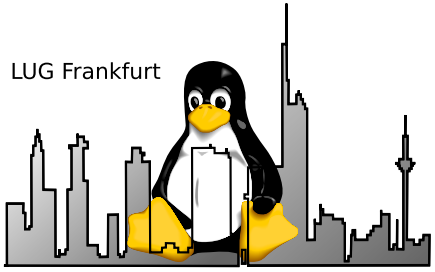


LUG Frankfurt

More Busy robots







Impressions

https://photos.app.goo.gl/d9cg22WpP8i4HFwYA

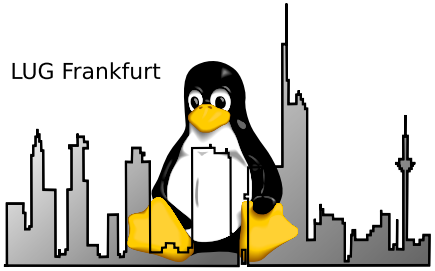


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

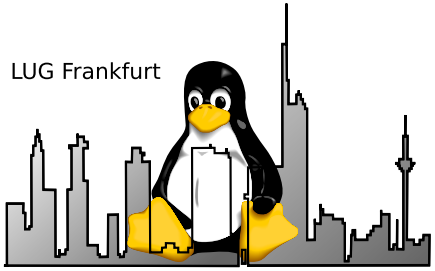
Calliope

B·O·B·3

Edison

Sphero

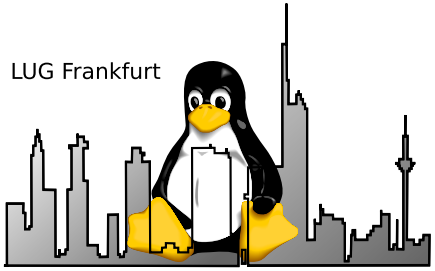
Cost



Linux software

- Graphical interface needed, but lightweight environment entirely sufficient. Currently in use for Physik Mobile Xfce, but no reason why e.g. LXQt would not work.
- Most of the robots/MicroControllers are programmed using a Web application, i.e. a web browser is needed. Firefox works fine, as long as Web Usb or Web Bluetooth are not needed. In these cases a Chromium based browser is required.
- The distributions used on the Physik Mobile laptops are Debian and openSUSE[1]. Again, other distributions should work just fine too.

[1] Migration from openSUSE to Debian currently on-going



Linux hardware

Requirements on the laptop/desktop to use the robots/MicroControllers are low to moderate. Based on the equipment in use for Physik Mobile, the follwoing is working well for the courses:

- Two core CPUs, clock speed $\geq$ 2 GHz
- 4 GB of memory
- 100 GB disk space
- For Sphero only: LE Bluetooth Interface

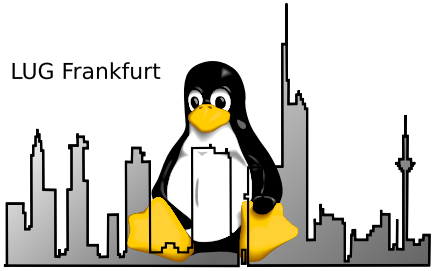


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

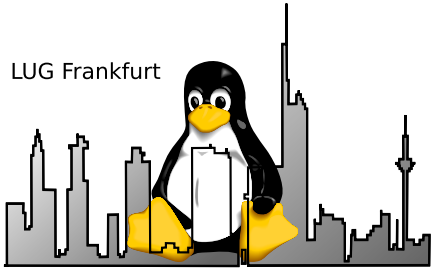
Calliope

B·O·B·3

Edison

Sphero

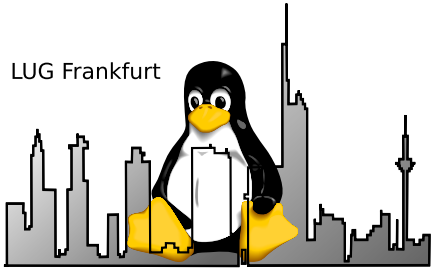
Cost



Introduction

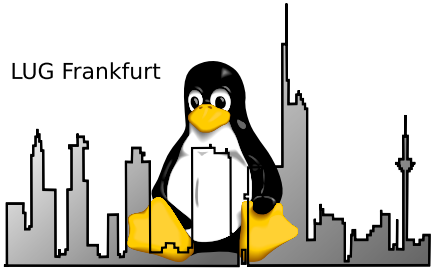
- Most of the devices used are controlled via USB
- The other somewhat common interface is Bluetooth[2]

[2] Albeit this is more often used with iPads



USB setup

- On Linux not all users/applications have arbitrary access to USB devices
- If you try to connect a new robot/microcontroller and it does not work a quick test (on a non-critical test system) is to try to connect using a privileged account (e.g. root). If this works, you just have a premission problem :-)
- Permissions managed via udev (userspace /dev), find via lsusb, then write udev rule



USB setup (contd.)

Example for Calliope V3:

```
$ lsusb
...
Bus 003 Device 012: ID 0d28:0204 NXP ARM mbed
...
$ cat /etc/udev/rules.d/99-calliope.rules
SUBSYSTEM=="usb", ATTRSidVendor=="0d28", ATTRSidProduct=="0204", MODE="0666"

$ sudo systemctl reload systemd-udevd.service
```

Note: Access could be further limited by adding a group to the udev Rule (GROUP="groupName",)
udev rules for all devices shown here available on request

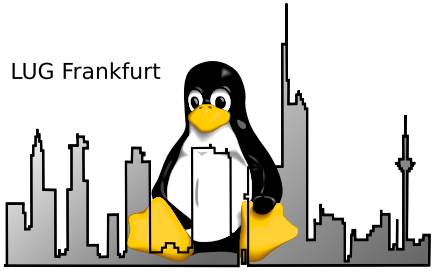


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

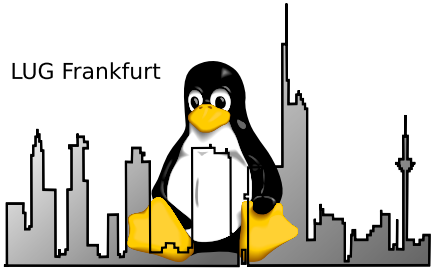
Calliope

B·O·B·3

Edison

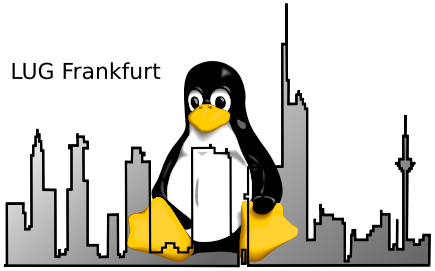
Sphero

Cost



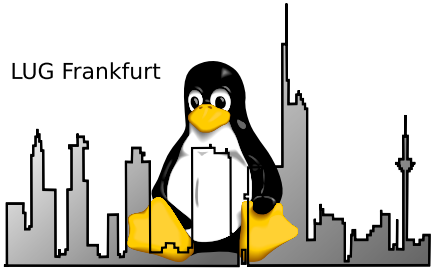
Calliope Filesystem

- This works for all versions of Calliope and running the graphical programming environment in any browser (Chromium or Firefox based)
- Most versions of Calliope export a file sysem with UUID 2702-1974 and label "MINI".
- To streamline loading of programs onto the Calliope setup autofs
  - Install autofs using apt, dnf, zypper, ...
  - Create directory (eg. /calli) as mountpoint (with mask ’0755’)
  - Add /calli /etc/auto.calli –timeout=300 to /etc/auto.master



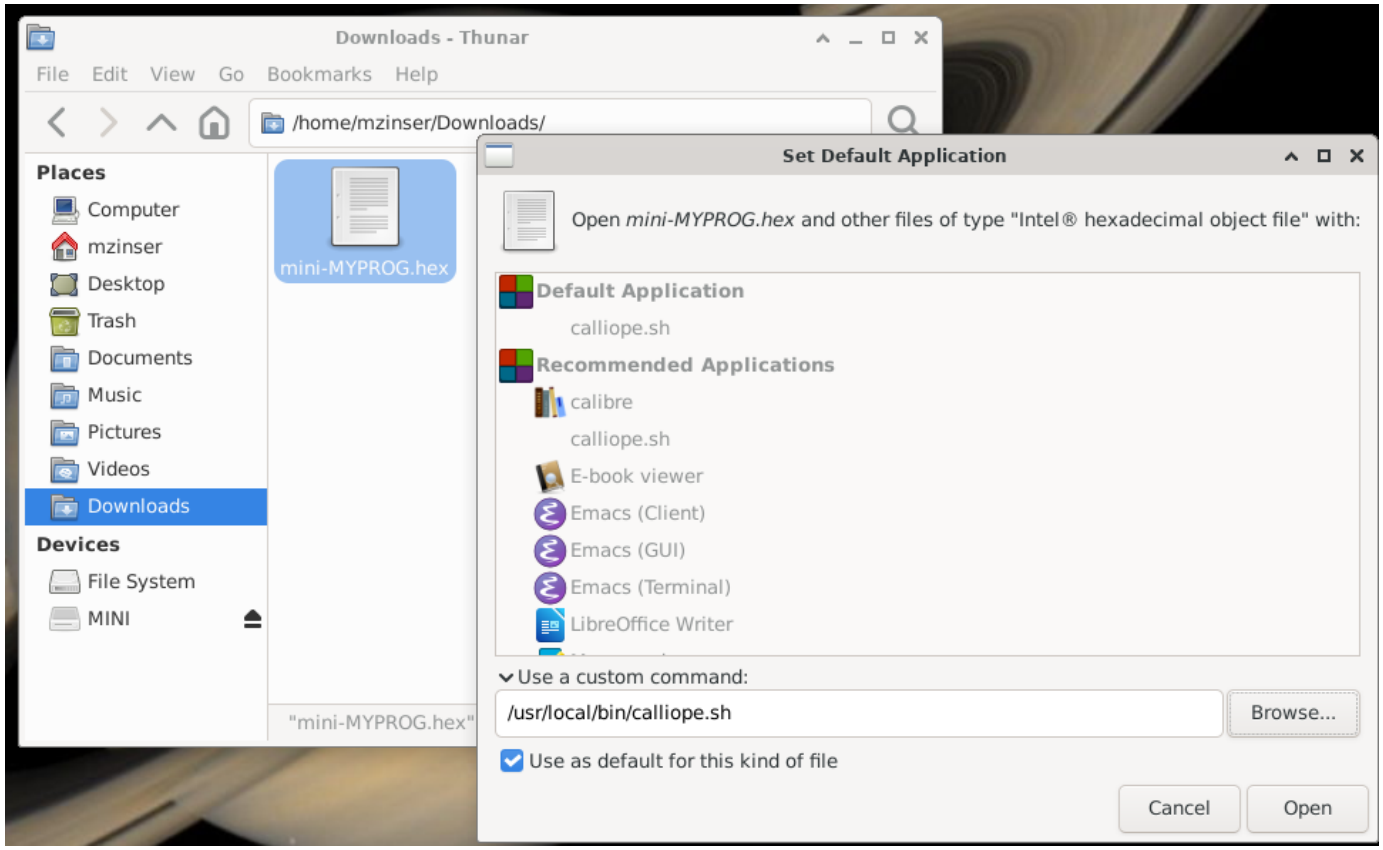
Calliope Filesystem (contd.)

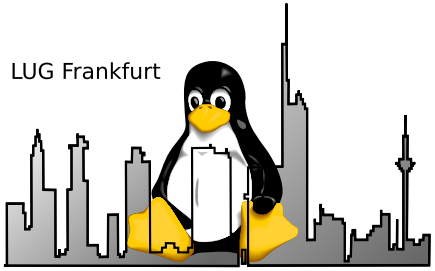
- autofs continued
  - In auto.calli add cAm -fstype=vfat,rw,gid=mc-user,umask=002 :/dev/disk/by-uuid/2702-1974
  - gid=mc-user is the name of the group for users working with Calliope
- To allow a good workflow, give sudo rights to unmount device %mc-user ALL=(ALL) NOPASSWD: /usr/bin/umount /calli/cAm



Calliope Filesystem (contd.)

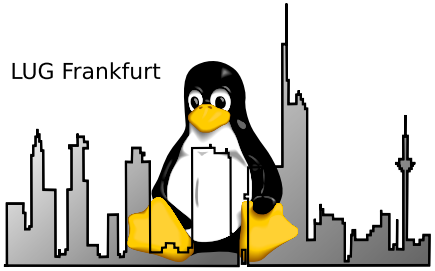
- Use simple script to transfer files to Calliope (and create backup copy)
- Example for xfce file Manager:





Calliope V3 - Web USB

- Starting with Calliope V3, Web USB is supported to transfer data
- This requires custom udev rules (see above) and Chromium based browser (–enable-features=UnrestrictedUsb). For a more detailed discussion see Sphero
- Convenience solution for programs
- Prerequisite if you want to log data from sensors to the PC



What about the BBC MicroBit?

Calliope: BBC MicroBit:

- The original Calliope is based on the interface design of the BBC MicroBit
- If you have setup your computer to interact with Calliope, you may swap in a BBC MicroBit and it will "just work"
- As of now (Calliope V3 vs. BBC MicroBit V2) the main difference is that Calliope has built in support to control motors. For the BBC MicroBit an extension is needed to do so.
- More information about Calliope available in German (relevant when working with kids :-))

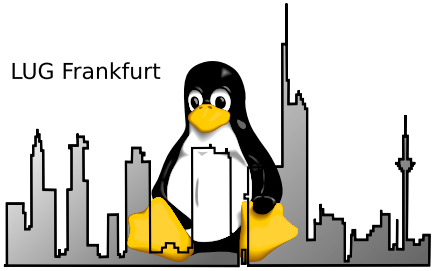


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

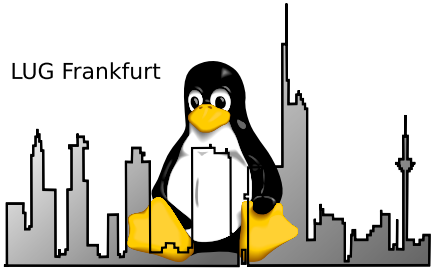
Calliope

B·O·B·3

Edison

Sphero

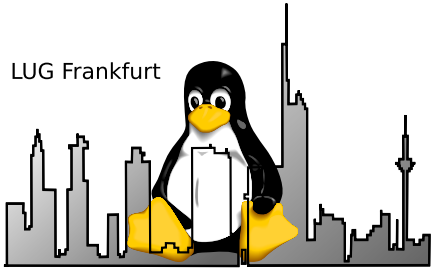
Cost



SW Installation

- The B·O·B·3 project not only provides a source for the micro controller and programming tutorials, but also software to transfer programs from a computer to the micro controller.
- For current version of Debian based distributions, just execute apt install bobdude
- For older/other distributions check the instructions on https://www.bob3.org/de/linux[3]
- Add the users working with B·O·B·3 to the dialout group usermod -a -G dialout [USERNAME]

[3] If you need a Spec file to build an RPM, get in touch



... under the hood

- bobdude is a simplified GUI on top of the much more general avrdude
- Since Version 8 avrdude also contains a (much more complex) GUI to transfer programs to a wide variety of AVR[4] CPUs.
- Settings compatible with B·O·B·3 - Device: ATmega88, Programmer: avrispv2, Interface: /dev/ttyACM0[5]

[4] Commonly attributed as **A**lf and **V**egards **R**ISC processor after the designers of the architecture at the Norwegian Institute of Technology

[5] Device Id might be higher than 0, depending on the devices connected to your system

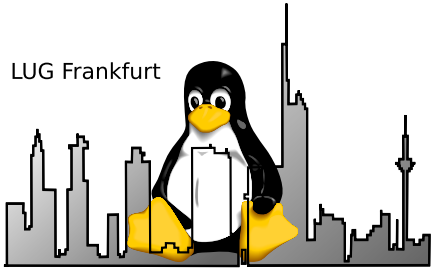


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

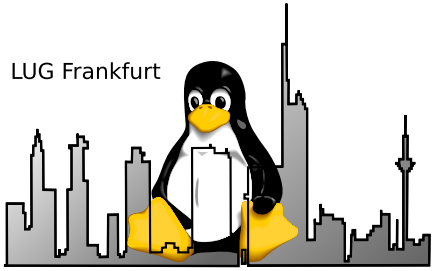
Calliope

B·O·B·3

Edison

Sphero

Cost



Same procedure as every ...

- lsusb

```
...
Bus 003 Device 015: ID 16d0:1207 MCS Edison V3
...
```

- udev rule:

```
\
SUBSYSTEM=="usb", ATTRS{idVendor}=="16d0", ATTRS{idProduct}
```

- Use Chromium based browser with Web USB enabled

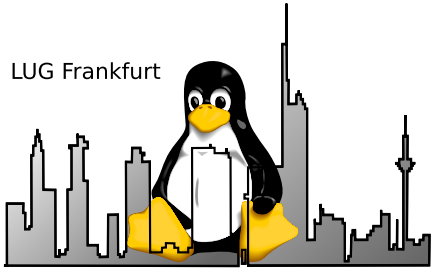


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

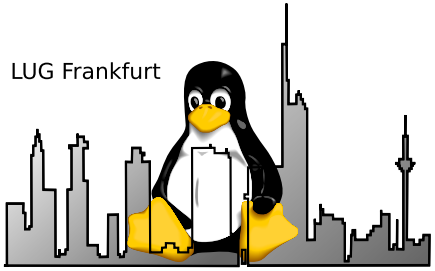
Calliope

B·O·B·3

Edison

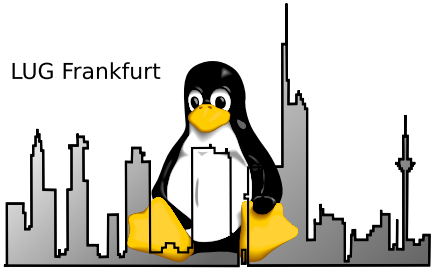
Sphero

Cost



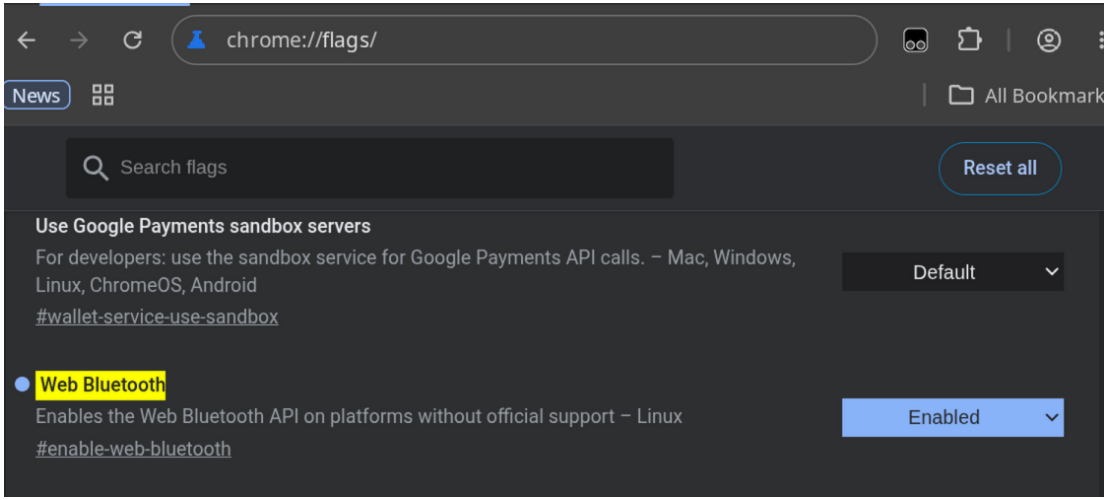
Talking Bluetooth

- The Sphero mini is communicating via Low Energy Bluetooth, the system controlling the robot must support this (older Bluetooth controllers do not)
- Sphero is programmed via a web app, i.e. it communicates via Web Bluetooth
- For a discussion of these interfaces see Mozilla Github
- Web Bluetooth is available in Chrome based browers



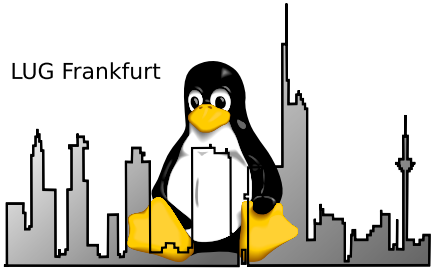
Talking Bluetooth (contd.)

Google Chrome chrome://flags



Microsoft Edge Web Bluetooth enabled by default

Vivaldi vivaldi://flags -> See Google Chrome, but not really needed, since it will prompt to allow USB access when needed.



Talking Bluetooth (contd.)

- Going through flags and restarting browser not very user friendly
- Solution: /usr/bin/google-chrome-stable –enable-features=WebBluetooth
- Can be set as an alias
  alias chromeBT='/usr/bin/google-chrome-stable –enable-features=WebBluetooth'
- Also possible as part of a .desktop file to create an entry in the application launcher

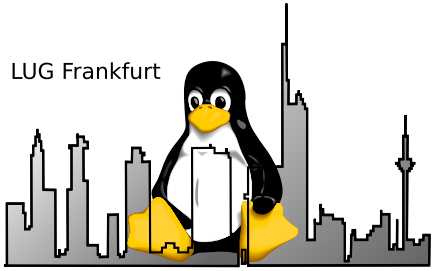


Table of Contents

Why Robots ?

What can my robot/microcontroller do?

The Code

Activities

Linux system

USB Connections

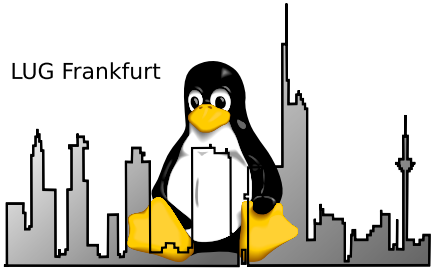
Calliope

B·O·B·3

Edison

Sphero

Cost



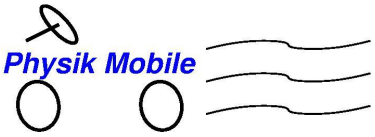
Cost

B·O·B·3	Assembled SMD Version plus ProgBob USB programmer 60-65 €
Calliope	Calliope mini 3 40 € Various extensions (Jacdac, MotionKit, GameKit) 30 - 70 €
BBC MicroBit	ca 20 €
Sphero	Sphero mini robot 60 - 70 €

So Long, And Thanks for All the Fish



Robotik und Coding in der Grundschule



1.Warum das Ganze?

Robotik ist ein inspirierender Einstieg in die Welt der Computer. Natürliche Differenzierung. Mit einigen Robotern ist die Vorbereitung sehr überschaubar für Lehrkräfte. Kinder lieben Roboter, lernen dabei u.a. Sprachfähigkeiten (Vorgangsbeschreibung, Genaue Sprache. Satzzeichen), Mathematik (Koordinaten, Rechnen, Messen), technische Konzepte (Elektrizität, Antrieb, Lenkung, Stabilität), logische Konzepte (Wenn-dann, Verknüpfungen, Schleifen), ohne erstmal das Lernen im Vordergrund zu sehen. Bei den meisten Robotersystemen, die in der Schule verwendet werden können, steht auch Teamarbeit im Fokus, da es im Normalfall Material- und technische Begrenzungen auf die Roboter pro Klassenzimmer-Quote gibt.

Das vorliegende Dokument ist eine einfache und übersichtliche Handreichung für Lehrer_innen, mit dem Zweck, als Einstiegshilfe zu dienen, indem verschiedene Systeme vorgestellt und praktische, systemübergreifende Unterrichtsideen gegeben werden.
Des Weiteren soll dies eine Auswahlhilfe sein, falls eine Schule die Möglichkeit hat, Roboter zu kaufen oder auszuleihen.

2. Welche Roboter?

a. Blue Bot
b. Ozobot
c. mTiny
d. Mabot
e. Thymio
f. BOB3
g. Dash
h. Sphero
i. Mini Sphero
j. Sensebox
k. Calliope
l. Calliope Callibot
m. BBC Microbit
n. BBC Microbit Car
o. Lego
p. Fischertechnik
q. arduino
r. Turtlestitch-Stickmaschine/Snap/Scratch/Programmierter Plotter/Cutter

3. Vergleichsfaktoren vor Anschaffung sind u.a.:

a. Stabilität, Langlebigkeit
b. Ladezyklus und Ladesysteme
    - ☐ Hält zwei Doppelstunden Betrieb durch
    - ☐ Zurücksortieren - Aufwand
    - ☐ Größe (#) eines Klassensatzes
c. Sensoren
    - ☐ Optisch (Cam) - Farbsensor
    - ☐ Akustisch
    - ☐ Abstand
    - ☐ Berührung
    - ☐ Optisch

d. Art(en) der Programmierung
    - ☐ Blockly
    - ☐ Text
    - ☐ Puzzleteile
    - ☐ Farbcodes
    - ☐ Zeichnungen

e. zusätzlicher Bedarf (Tablets? PCs? Platz?)
f. Spaßfaktor
g. Praxis-Erfahrungen/Zielgruppen
h. Preis (?)

4. Programmier Aktivitäten

Body Coding/Buddy Coding

Kinder programmieren sich gegenseitig oder den Lehrer. Jedes Kind 3 gibt bis zu Befehle und gibt dann die Befehlsgewalt weiter.

- Koche Instant-Tee
- Schmiere ein Butterbrot
- Ziehe eine Jacke an (“Buddy Coating”)
- Male ein Strichmännchen an die Tafel
- Laufe eine bestimmte Strecke ab (Partnerarbeit)
- Baue ein Konstrukt aus Bausteinen / Gegenständen aus dem Ranzen/Klassenzimmer

Die letzten beiden Aktivitäten funktionieren auch gut als Partnerarbeit.

KI-Erweiterung

Die Aufgaben aus dem vorigen Abschnitt kann man mit einem Lernprozess verbinden. Am Anfang steht die Frage, was wäre, wenn die Klasse befähle, alles zu wiederholen, wie vorher. (Der Roboter lernt nicht, also ist es dasselbe Chaos)

- Zum Beispiel wird das Strichmännchen fünfmal gemalt mit anschließendem Feedback der SuS - richtig oder falsch. Daraufhin wird das nächste Fünferset gemalt - solange bis es stimmt.
- Erklärung mit den Captcha/finde alle Motorräder im Bild, was gerne bei Google als Prompt gegeben wird.
- MasterMind ist ein gutes Spiel zum Unterstreichen.

Aufgabenbeispiele:

Auf Distanz fahren. Im Klassenraum wird eine Startlinie festgelegt, SuS programmieren ihren Roboter dazu, möglichst nah an eine Ziellinie heranzufahren. Wer die Linie überschreitet, ist aus dem Rennen gefallen. Wer am nächsten an die Linie heran fährt, hat gewonnen.
Anwendbar: fast alle fahrenden Roboter. In Klassen 2-4: mTiny, BlueBot, Dash
Nebenbei Lernen: Messen, Abschätzen,

Weitfahrwettbewerb Alle SuS bekommen dieselbe Menge an erlaubten Befehlen. Roboter werden so programmiert, dass sie möglichst weit von einer Startlinie in eine vorgegebene Richtung fahren. (Ein mögliches Set an Befehlen waere: ↑,↑,↑,←,←,→,→,↓,x2, evtl. (,))
Anwendbar: In Klassen 2-4 mTiny, Bluebot mit Programmierkärtchen.
Nebenbei lernen: Logische Reihenfolge, Genauigkeit, Symmetrie.

Buchstaben Zeichnen: SuS lassen ihren Roboter ihre Initialen abfahren.
Anwendbar: fast alle fahrenden Roboter. In Klasse 2-4: mTiny, Bluebot. Fuer aeltere Kinder auch turtlestitch/logo
Nebenbei Lernen: Symmetrien, Räumliche Vorstellung

Storytelling (Sprachunterricht, Sachunterricht, Mathe…) Eine Geschichte wird erzählt, indem ein Roboter verschiedene (Bild-)Stationen abfährt.
Spracherwerb (mTiny - Stories)
Anwendbar: Alle fahrenden Roboter (Ozobot, mTiny, BlueBot)
Nebenbei Lernen: Symmetrien, Räumliche Vorstellung

Labyrinth lösen: Aus Bausteinen wird ein Labyrinth aufgebaut und die Roboter müssen möglichst exakt hindurchfahren. In der zweiten Stufe kann man das auch als Wettbewerb auf Zeit durchführen, in der Klasse mit vielen Robotern kostet das aber viel Unterrichtszeit.

Anwendbar: Alle fahrenden Roboter (Ozobot, mTiny, BlueBot)
Nebenbei Lernen: Symmetrien, Räumliche Vorstellung

Tanzroboter: Zwei oder mehr Roboter werden so programmiert, dass sie gleich oder gegenläufig "Ballet tanzen".
Anwendbar: Alle fahrenden Roboter.
Nebenbei Lernen: Symmetrie, vielzählige Symmetrie, Teamwork.

Experimente: Mobiliar wird schräg gestellt, und man findet heraus, wie groß die Steigung sein darf, auf der der Roboter noch hoch kommt.
Ein Tischtennisball soll in ein Tor geschoben werden. Hilfsmittel wie der Nachbar-Roboter oder Papierstücke sind erlaubt.
Ein Papierstreifen mit zwei Löchern wird als Abschleppseil verwendet. Welcher Roboter ist stärker, welcher ausdauernder?
Weitere Themen: Wendekreise, Geschwindigkeiten, Farberkennung (bei passenden Sensoren)
Anwendbar: Fahrende Roboter. Klasse 2-4 mTiny, Bluebot, Ozobot
Nebenbei lernen: Experimentieren - Annahmen machen, Probleme finden und lösen. Wissenschaftliches Beobachten. Mathematik

Vergleich Fernsteuerung und Programmierung: Roboter werden dieselbe Strecke per Fernsteuerung gesteuert und programmiert. Die Klasse diskutiert, was einfacher ist.

Blockly-Programmierung (Klasse 3-7…)

Blockly ist der nächste Schritt - hier muss entweder ein PC/Laptop oder ein Tablet mit dem Roboter verbunden werden. Auf deren Bildschirm werden dann Kommandos in Form von Puzzleteilchen (ähnlich den Teilen bei mTiny) zusammengefügt. Hier ist die “Anzahl der Programmschritte” freigegeben, Wettbewerbe mit limitierten Programmschritt-Anzahlen sind in der Praxis nicht mehr durchführbar. Ab Klasse 3 kommen Kinder mit den ersten Varianten von Blockly gut klar.
Die meisten Roboter haben zugehörige Apps, die auf iPads laufen, und mit Hilfe derer man blockbasiert programmieren kann. Diese Apps sind sich jeweils sehr ähnlich, aber nicht gleich. Zwei Systeme stehen hervor, makecode und Open Roberta: Beide versuchen, eine Plattform darzustellen, auf der man verschiedene Geräte betreiben und programmieren kann.
Beide sind browserbasiert und damit ans Internet gebunden. Roboter und Mikroprozessoren wie mBot, Thymio, BOB3, Sensebox, Arduino, Calliope, sind in mindestens einer dieser Plattformen eingebunden.

Mögliche Aufgaben:

Geometrische Formen /Buchstaben fahren. Malen. An manchen Robotern (Thymio, BlueBot) kann man Stifte mit Hilfe einer Vorrichtung befestigen. Durch die Bewegung der Roboter entstehen dann interessante Muster. (siehe auch die Aufgaben oben)

Man kann mit einfachen Mustern (Quadrat?) anfangen. Mithilfe von Loops (Schleifen) können Spirograph-artige Muster entstehen.

Erweiterung: Durch Mehrfachfahrten desselben Programms können SuS einfach kontrollieren, wie genau ihre Roboter arbeiten.

Anwendbar: Thymio, Bluebot, aber auch Sphero und Snap/Turtlestitch (hier wird die Bahn digital aufgezeichnet, was dem enormen Papierverbrauch vorbeugt)

Nebenbei Lernen: Experimentieren. Rechts-Links Vorstellungsvermoegen.

Lenkung
SuS vergleichen verschiedene Anweisungen fuer Rechts/Linksdrehungen der Raeder. Wer bekommt den engsten Wendekreis hin. Welcher Roboter hat auf einem Parcours am schnellsten gewendet, ohne etwas anzufahren?

Anwendbar: Thymio, Bluebot, Callibot,
Nebenbei Lernen: Experimentieren. Rechts-Links Vorstellungsvermoegen. Fahrzeugtechnik

Labyrinthe:
SuS bauen ein Labyrinth aus Bausteinen und vergleichen, ob Fernsteuerung oder Programmierung den Roboter schneller hindurch bekommt.

Vergleich von Suchmustern
Snap -Roboter- Drohnenflug. Für die turtle, den Roboter, die Drohne wird ein Suchmuster programmiert.

Fahrzeugtests:

- Wie groß ist der Wendekreis?
- Wie schnell ist der Roboter?
- Besteht der Roboter den Elch-Test?
- Ist es ein All-Terrain-Vehicle?
- Entwickle eigene Tests

Anwendbar für alle Roboter mit Rädern. Gut auch, wenn verschiedene Roboter vorhanden sind.
Nebenbei lernen: Fahrzeugtechnik. Experimentieren.

Sensorik. If-Then-(else). Schleifen. Timer.
Aktoren aller Art können angesteuert werden:
Beispiele:
Ein Blinklicht

Ein Polizeiblinklicht in rot und blau abwechselnd blinkend
Ein Piepser geht an
Ein Display zeigt ein Muster
Ein Lauflicht

Bedingungen (Wenn der Sensor x einen Wert gibt, dann wird ein Aktor angesteuert)

Beispiele:
Wenn ich den Sensor berühre (nah dran bin), dann geht das Licht an (Taschenlampe)
Wenn das Fahrzeug nah an der Wand ist, dreht es um und fährt weiter (Staubsaug Robo)
Wenn es dunkel wird, geht das Licht an (Smart home)
Ein Display zeigt einen Messwert (Wetterstation)
Wenn der Roboter schräg steht, gleicht er das mit Bewegungen aus (Balance-Bot)
Wenn der Roboter schräg steht, schaltet er alles aus (Drohne)

(Diese Programme hängen alle von der Ausstattung der Roboter/Platinen ab. Siehe Beschreibung der Roboter)

Textuelle Programmierung (Ab Klasse 5)

Dieselben Aufgaben wie bei Blockly - nur werden Programme erstmal schwieriger zu erstellen (PC/Laptop nötig) aber im Anschluss auch übersichtlicher, da mehr Programmcode auf einen Bildschirm passt.

Makecode und OpenRoberta, sowie einige der anderen Blockly-Systeme haben ein Menu, in dem man den Quellcode für seine Blockprogrammierung anschauen und bearbeiten kann.

Ein sehr einfacher Einstieg in die textuelle Programmierung kann aber mit BOB3 gelingen, für den es einen kompletten gamifizierten Unterrichtsplan gibt.

Feedback der SuS:

- Abfragen mit Thumbs-up und Thumbs-down
- Worauf muss man beim Programmieren achten?
- Habt Ihr Ideen, was ihr mit einem solchen Roboter machen könntet?

Beispiel-Stundenplanung für einen 90-Minuten Workshop:

Einführung in das Thema Robotik:
Erarbeitende Fragen zum Thema „Roboter“; die Kinder werden aufgefordert, ihr Wissen über Roboter im Plenum zu erzählen. Ziel ist es, dass die SuS die Unterschiede Mensch/Maschine im Hinblick auf Sensoren und Programmierung erkennen [15 Min]

Body Programming:
Moderiert von der Lehrkraft programmieren die SuS ein Kind aus der Klasse, damit es einen Zitronentee zubereitet. Die SuS erfahren durch dieses Spiel, welche Bedeutung konkrete kleinteilige Anweisungen beim Programmieren haben [15 Min]

Roboter kennenlernen: Auswahl bzw. Kombination verschiedener Unterrichtsvorschläge [50 Min]

Reflexion, Feedback:
Worauf kommt es beim Programmieren an? Wie hat die Teamarbeit funktioniert [15 Min]