AMOS

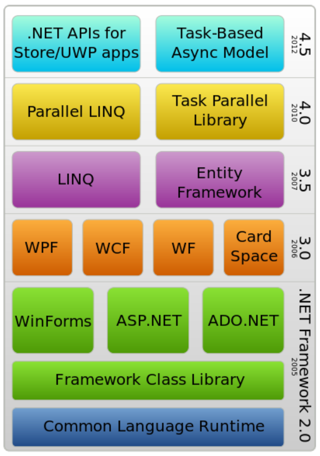
DOCUMENTATION

Raspberry Pi as user control board for multimedia evaluation boards

TERMINOLOGY

Due to the nature of this project and requirements by the industry partner Sivantos, there are a couple of technologies used one should have a good overview and understanding of.

- **.NET** is the name of a multitude of software frameworks and development platforms provided by Microsoft. It consists of libraries offering user interfaces, security / cryptography, database access, etc.



Different Components offered as .NET [1]

  .NET comes in three different flavors: *.NET Framework*, *.NET Core* and *Xamarin*. All of these share same fundamental principles, but offer different class libraries as well as target platforms i.e. while .NET Framework is the oldest, historically grown project since 2002, it only runs on Windows but is perfectly suited for cases where legacy systems need to be supported as well. On the other hand, Xamarin is used for creating iOS / Android mobile applications with the support of a Microsoft Environment and usually C# as the programming language of choice. [2]

- **.NET Core and Universal Windows Platform (UWP)**
  UWP helps develop universal apps that run on both Windows 10 and Windows 10 Mobile without the need to be re-written for each. UWP provides a guaranteed core API layer across all devices that run Windows 10. UWP .NET apps use .NET Core underneath. UWP is only for the Windows ecosystem. *You can use .net core to create an app targeting UWP, but UWP isn't part of .net, it's part of Windows.* [3]

- **Windows IoT**
  (previously Windows Embedded) is a family of operating systems from Microsoft designed for use in embedded systems. [4] There are three different versions such as Enterprise, Mobile Enterprise, as well as Core. Windows IoT Core is basically a watered-down version

[1] https://en.wikipedia.org/wiki/.NET_Framework
[2] https://blogs.msdn.microsoft.com/cesardelatorre/2016/06/27/net-core-1-0-net-framework-xamarin-the-whatand-when-to-use-it/#dotnet-whole
[3] https://stackoverflow.com/questions/38680266/is-net-for-universal-windows-program-a-subset-of-net-core
[4] https://en.wikipedia.org/wiki/Windows_IoT

of Windows, running on small, low-cost devices such as the Raspberry Pi. For management of the Pi where IoT Core runs on, Microsoft offers Windows 10 IoT Core Dashboard[5].

- **Visual Studio** is an integrated development environment (IDE) from Microsoft. It is mandatory to use this for this project, because UWP applications can only be coded on this platform, and UWP is what is used to make the code run later on the Raspberry Pi.

  ➔ **For an Introduction on how to get set up, refer to:**
  Getting Visual Studio (Windows IoT) with Raspberry Pi setted up.pdf [6]

- **NuGet**
  is a package manager that comes bundled with Visual Studio. When you use NuGet to install a package, it copies the library files to your solution and automatically updates your project (add references, change config files, etc.). If you remove a package, NuGet reverses whatever changes it made so that no clutter is left. [7]

- **Testing**
  In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.[8] In the case of this project, a requirement is to use Travis CI to help with automating tests and integration.

```
using Xunit;

namespace Fibonacci.Tests
{
    public class Tests
    {
        [Fact] // fact is a keyword used by xunit
        public void Get_5th_number()
        {
            var generator = new FibonacciGenerator();

            Assert.Equal(generator.Fibonacci(5), 15);
        }

        [Fact]
        public void Get_6th_number()
        {
            var generator = new FibonacciGenerator();

            Assert.Equal(generator.Fibonacci(6), 8);
        }
    }
}
```

Simplest Form of a Test using Asserts

[5] https://developer.microsoft.com/en-us/windows/iot/docs/iotdashboard
[6] https://drive.google.com/open?id=0BzaNmZTttJK4Tk9CaW1lQ0JTV0U
[7] https://www.nuget.org/
[8] Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. p. 75. ISBN 0-470-04212-5.

- **Continuous integration**
  Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.[9]

[9] https://www.thoughtworks.com/continuous-integration

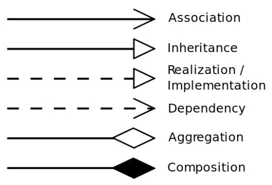
UML

This section covers random titbits that are related to the Unified Modelling Language and are supposed to provide a reference as well as a refresher of certain aspects. UML is being used in this project e.g. to document the overall software architecture, which can be found in the Google Drive or GitHub Repo in the `Documents` folder.

Class Diagrams

In a class diagram, you don't show the act of instantiation. That is something that a sequence or communication diagram would show. Class diagrams show the static structure of a system[10].

- **Associations and Dependencies**
  If a class A "depends on" a class B, then you could look at the association and dependency relationships. Dependency is the weaker of the two and is usually used if the other class is a parameter to a method and is not stored as an instance variable. Association is stronger and means that class A contains an instance (or instances) of class B, and is further strengthened by the composition and aggregation relations[11].



- **Static**
  Static Members in general can be referenced without instantiating the class they belong to. In UML, static members are underlined.

- **Getter & Setter (of properties)**
  UML does not define getter setter operations. Get and Set method are used in programming languages to realize attribute definition. For example, read-only attribute will have getter method only in implementation code.

[10] https://softwareengineering.stackexchange.com/questions/101157/how-do-you-annotate-instantiation-in-uml-class-diagrams
[11] https://en.wikipedia.org/wiki/Class_diagram#Association

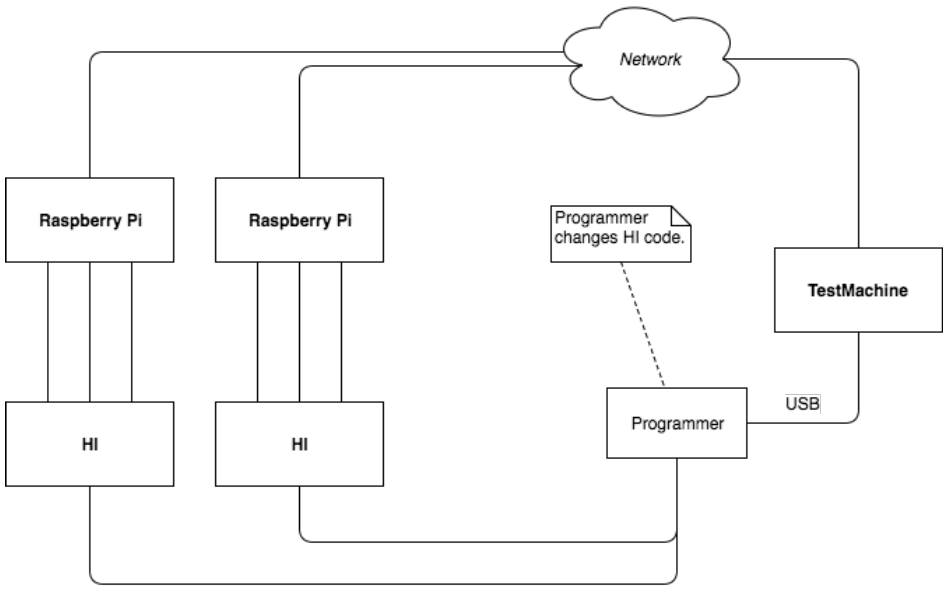
HARDWARE

The overall structure of the project is depicted in the figure below and basically consists of several Raspberry Pies that run some sort of *automated testing* on so-called *Hearing Instruments* (HI) they are connected with.

These HIs are hardware boards that emulate the functionality of actual hearing aids and are programmed via a USB interface, depending on the hearing aid emulated. This *programmer* changes the mapping of functionality on the board itself.

The idea is to not have the Raspberry Pies hooked up to a *TestMachine* but rather have them connected over a network using a dedicated and central server discovery service with a static IP where the Pies are connecting to *(also refer to Sprint Protocol #3 where this has been discussed with the industry partner).*

Sivantos (the industry partner of this project) has provided a specific breadboard that the Raspberry Pi connects to via its GPIO (General-purpose input/output) Pins where it then drives and controls certain chips and e.g. the attached LCD display.



GPIO

- General-purpose input/output pins are included on the Raspberry Pi to allow establishing a wired connection to arbitrary hardware that can then be communicated with.

- Each pin serves a different function, while some of them may be used generally others have pre-defined functions that can be looked up in the pin-layout below, e.g. there are mere power sources (3V/5V), grounding pins (0V) to establish a closed circuit, as well as specialized pins that communicate via *serial protocols* such as *I2C*, *UART* or *SPI*.

3.3 V	**1**	**2**	5 V
SDA (i2c Data) GPIO2	**3**	**4**	5 V
SCL *(i2 Clock)* GPIO3	**5**	**6**	GROUND
GPIO 4	**7**	**8**	GPIO 14 (TxD)
GROUND	**9**	**10**	GPIO 15 (RxD)
GPIO 17	**11**	**12**	GPIO 18
GPIO 27	**13**	**14**	GROUND
GPIO 22	**15**	**16**	GPIO 23
3.3 V	**17**	**18**	GPIO 24
(MOSI) GPIO 10	**19**	**20**	GROUND
(MISO) GPIO 9	**21**	**22**	GPIO25
(SCKL) GPIO 11	**23**	**24**	GPIO8 (CE0)
GROUND	**25**	**26**	GPIO7 (CE1)

Sample Pinout of Raspberry Pi *(refer to* [12] *for full sheet)*

- GPIOs handle HIGH (3.3V) or LOW (0V) electrical signals. Note that HIGH or LOW isn't necessarily as simple as current flowing / not flowing, and also some uncertain floating can occur on a physical layer. Whilst coding this however isn't important, since we can rely on digital 1s and 0s.

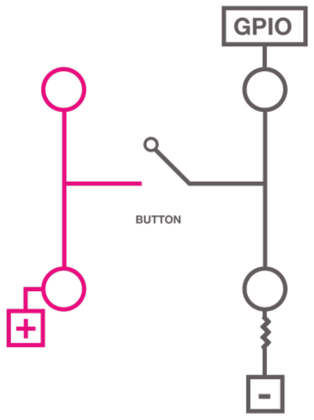
- GPIO Pins can be set to read input or send output, but not both at once. If a pin is set to output, the Pi can either send out 3.3 volts (HIGH), or not (LOW or 0 volts).

- While outputting signals might be simple, reading input is slightly more complex due to physical conditions of a circuit. Input Pins don't have a defined state, they're waiting for an inputted signal and meanwhile (because the circuit is not closed) they happen to react fairly sensitive to other electromagnetic frequencies in the air. This means they're floating between currents, it's not HIGH or LOW, but somewhere in between. To counter this, *resistors* are used, as well as the fact that electricity travels the path of least resistance.

[12] https://pinout.xyz/

- Pull-Up or Pull-Down resistors help combat this issue by redirecting electricity in a way that ensures logical 0s and 1s are read when needed[13]:
  `e.g. When a button is being pushed -> READ: 1 ELSE READ: 0`

- The figure below displays how a ground the reading GPIO pin is connected to allows to "pull down" the current when there's no signal to 0. Once the button is pressed the current and thus the logical 1 is redirected accordingly to the GPIO pin instead of going to the ground (where electricity always wants to go to) because of the resistor.



PullDown Resistor

BREADBOARD

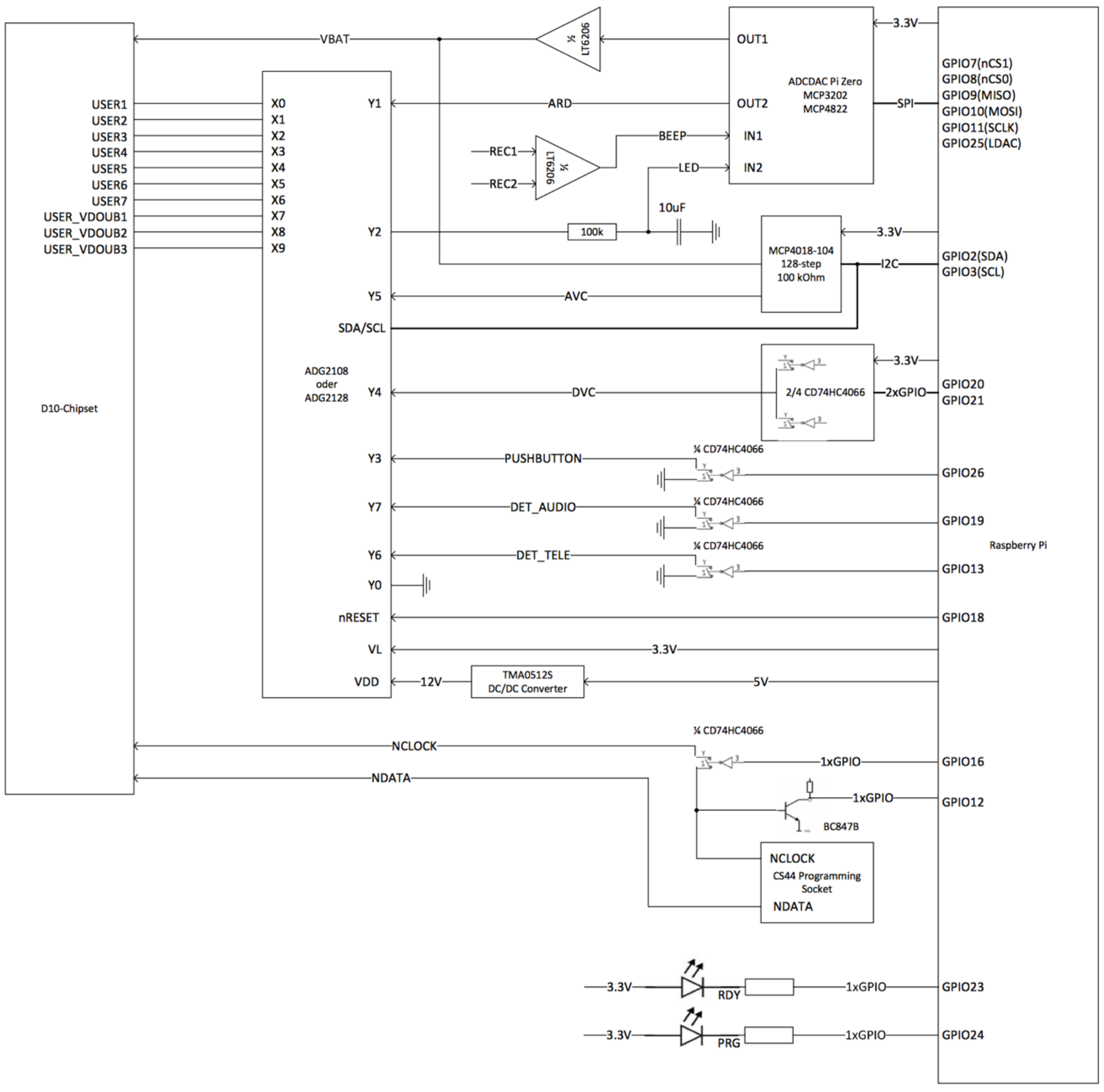
The breadboard used in electronics is actually based on the historic practice of connecting and nailing circuits onto a real wooden breadboard to experiment. It's widely used because it's perfectly suited for creating solderless circuits temporarily during prototyping[14].

The breadboard provided by Sivantos is also referred to as *Evalboard*. During internal developing a much simpler breadboard was used to bootstrap initial code communicating with hardware such as an LED.

[13] https://medium.freecodecamp.org/a-simple-explanation-of-pull-down-and-pull-up-resistors-660b308f116a
[14] https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard

EVALBOARD

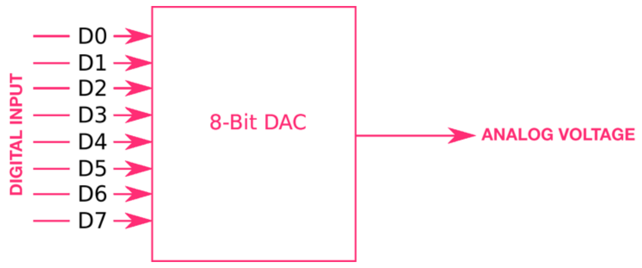
The Raspberry Pi is connected to a breadboard provided by Sivantos (also refered to as Evalboard). It consists of a multitude of different chips that are wired together in a particular way depicted in the circuit diagram below. It consists of different components such as potentiometers, ADC-DAC and switches.



Circuit Diagram of Evalboard

In the following section, a few *Integrated Circuits* (Chips) will be highlighted and references to manuals or datasheets are also given.

- **ADC-DAC Pi Zero**
  Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC) are very important components in electronic equipment. Since the hearing aids might deal with audio signals or produce these as well a conversion of signals might be necessary.



An analog audio signal is a continuously varying voltage that perfectly represents (or is analogous) to the continuously varying sound wave that you hear. As example, a microphone turns incoming sounds into an analog electrical signal representing those sounds; room speakers convert an analog electrical signal back into the original sound (or as close as possible). But how does one store an analog signal? A half century ago, one would store an analog signal as a groove on an LP record that moves the needle back and forth during playback to create an electrical analog signal representing the stored sound. Nowadays, we repeatedly sample and measure the height of the analog signal over time, and then store that series of numbers on a hard-drive or memory of an audio device. This series of 0s and 1s numbers is a digital audio signal. [15]

➔ For further information about this chip refer to: `Datasheet-MCP3202.pdf` [16]
➔ **Currently the use of this chip is not fully clear!**

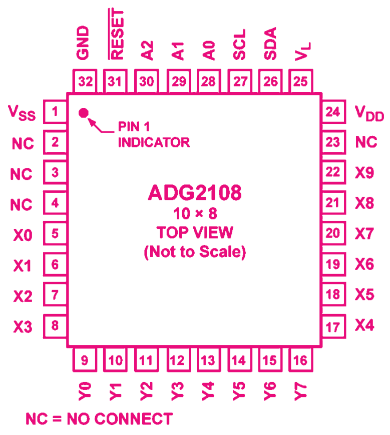
- **MCP4018-104 (Potentiometer)**
  The MCP4018 is a volatile, 7-bit (128 wiper steps) digital potentiometer with an *I2C* Compatible interface. Also referred to as Pots, these are special resistors that allow the output of a variable amount of voltage. They can be used to e.g. adjust volume in sound circuits or emulate a battery status.

➔ For further information about this chip refer to: `Datasheet-MCP4018.pdf` [17]
➔ **Currently the use of this chip is not fully clear!**

- **ADG2108 / ADG2128 (Switch)**
  This chip is essentially a chip, mapping Input Addresses to Output Addresses. It can be configured via an I2C compatible interface, e.g. to map Y3 -> X5.

➔ For further information about this chip refer to: `Datasheet-ADG2108.pdf` [18]
➔ The chip is probably used to adjust output signals accordingly to the Hearing Aid Device Family and Model (as the pins on these might differ).

[15] https://www.headphone.com/pages/what-is-a-dac
[16] https://www.abelectronics.co.uk/p/74/ADC-DAC-Pi-Zero-Raspberry-Pi-ADC-and-DAC-expansion-board
[17] https://www.microchip.com/wwwproducts/en/MCP4018
[18] http://www.analog.com/en/products/switches-multiplexers/unbuffered-analog-crosspoint-arrays/adg2108.html



Example: Whereas the Rocker Switch on one hearing aid can be controlled via PIN_7 on another model it may be controlled via PIN_3.

- **TMA0512S (DC/DC Converter)**
  This is basically just an electric power converter, transforming a source of direct current *(DC: unidirectional flow of electricity)* from one voltage level to another. In this case it just takes the 5V output pin of the *Raspberry Pi GPIOs* and converts it to 12V because that's the power supply needed for the *Switch*.

  ➔ For further information about this chip refer to: `Datasheet-TMA0512S.pdf` [19]

- **LT6206 (Amplifier)**
  An amplifier uses electric power from a power supply to increase the amplitude of a signal.

  ➔ For further information about this chip refer to: `Datasheet-LT6206.pdf` [20]

- **CD74HC4066 (Switch / Multiplexer)**

  ➔ For further information about this chip refer to: `Datasheet-CD74HC4066.pdf` [21]

- **CS44 (Programming Socket)**
  The USB Hi-PRO from GN Otometrics is the most commonly used programmer used by hearing care professionals. HI-PRO is de facto industry standard for hearing instrument programming and is supported by all major hearing instrument manufacturers. Programming cable. This is the cable connecting your hearing aid to the programmer. Hi-Pro cables use CS43, CS44 and CS45. CS44 is the most commonly used. [22]

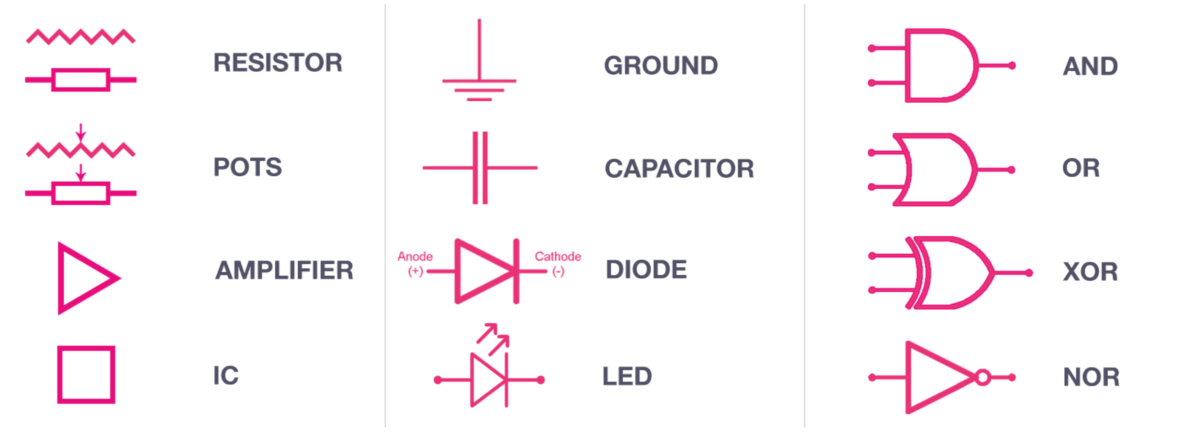
  ➔ Currently the use of this chip is not fully clear!
  ➔ No documentation found on the Web yet.

[19] http://www.tracopower.com/products/tma.pdf
[20] https://cds.linear.com/docs/en/datasheet/620567fc.pdf
[21] http://www.ti.com/lit/ds/symlink/cd54hc4066.pdf
[22] https://www.amperordirect.com/pc/help-hearing-aid/z-hearing-aid-program-tools.html

- **Abbreviations** (in electrical circuits)

VBAT	`Voltage of the Battery` *When used as Input: Status Reading*
AVC	`Automatic Volume Control` *Provides a controlled signal amplitude at its output, despite variation of the amplitude in the input signal*
DVC	`Dual Voice Coil (?)`
PUSHBUTTON	`Momentary Button` *Connects two points in a circuit only when pressed*
SPI	`Serial Peripheral Interface Bus` *Used for connection between Raspberry and ADC-DAC*
VDD | V_{DD}	`Drain Voltage Supply` *Positive supply voltage*
VL | V_{L}	`Logic Power Supply`
NRESET	`Inverted Reset` *When this pin is set to LOW all registers are cleared to 0*
GND	`Ground`
IC	`Integrated Circuit`

- **Common Symbols** (in electrical circuits)



SERIAL COMMUNICATION

I^{2}C

The Inter-Integrated Circuit (I2C) Protocol is a protocol intended to allow multiple “slave” digital integrated circuits (“chips”) to communicate with one or more “master” chips.

```
//TODO
```