Author: Erwin Engelsma

Initial Date: 8 July 2022

Last updated: 14 Dec 2022

Further architecture for the soft robot.

### Lessons learned

In the past months we have learned to calibrate the volume of air being blown into a muscle, based on PWM settings for the motor and a time measurement to reach a set volume based on pressure reduction when muscles have reached a given expansion. This served as input for a program that can repeatedly fill and empty the silicone muscle. Furthermore, we have gained the ability to respond in real time to new settings (update speed of about 100mSec, faster is not needed due to intrinsic slowness of mechanical response). It was found that due to hardware technical issues the required timer is not available to generate real-time response possibilities. A main loop using a less accurate software timer was still good enough to be well workable. For future board layouts this issue must be addressed. For EEG signals we have decided that although measuring Alpha waves in the brain is possible, it is not very practical due to the fact that in an ‘exposition situation’ the necessary degree of relaxation is not achievable by most persons. However, it was found that distinguishing hand and blink movements could fairly reliably be done. These were translated into ‘mood data’ where depending on the settings a set of 5 muscles forming an ‘octopus’ type shape could open and close faster or slower, and open wider of less wide. So far, this setup was controlled by one pump for blowing air into the muscle and one to take air out in a controlled way. A valve could be used to let air flow out, but the speed at which this happens cannot be controlled, and therefore the setup with two pumps is preferred.

### Vision for further development

In a recent discussion the vision was formed to form a set of moving soft robots, much like glowworms in a cave. They should be making complex movements, based on EEG signals, and change color as well. At a later stage some synthesis may be added to generate complex sounds, supporting the movements.

This asks for a different approach to analysis of EEG signals and control of multiple soft robots.

## Architecture requirements

* Capability to control a flexible number of pumps, extensibility
* No calibration of personal EEG signals required (too complicated for a situation where multiple visitors of exhibition should use it)
* Use of true real time by using hardware timer for timing functions
* Ability to ‘follow’ individual brainwaves, where brainwave amplitude is translated to volume of air in the muscle
* Ability to calibrate individual muscles

## Architecture vision

The following set up is envisioned (see )

EEG device

Data thru

WiFi

PC

Signal Acq and slowed replay

Server Arduino + WiFi

Data thru

USB

Client Arduino + WiFi

Client Arduino + WiFi

Client Arduino + WiFi

Data thru WiFi

4 PWM controlled Pumps

4 PWM controlled Pumps

4 PWM controlled Pumps

Pressure sensors

Pressure sensors

Pressure sensors

LEDs

SOUND

LEDs

LEDs

Although the Arduino Uno has 6 PWM outputs, only 4 can be used at the same time due to usage of the timers, only two are available for the PWM, and one timer can be used to control 2 PWM outputs. The Wi-Fi can serve in principle up to 14 channels in one go, but due to interference and the use of the Wi-Fi channel on the EEG equipment about 4 channels is probably the maximum. Leading to at most 8 separate muscle movements (Electronica can be built in such a way that one PMW drives more than one pump)

## Data flow and processing

EEG data coming from the device is recorded at 125Hz sample speed. This is way too fast for the mechanical animals (or human perception) to follow, so the data is slowed down to be output at a rate of about 1 sample per second (adjustable between 0.1 and 2 seconds). Up to 16 channels can be recorded and output in one go. The output is a string of floats in the channel order 1 to number of channels, with a value of 0 to 1 representing minimum and maximum volume for a muscle.

The Arduino server sends the data to the various clients. Client is responsible for translating the value to air volume control in the muscle. A calibration function for this must be available. Early testing has shown that for our ends a simple linear translation is good enough.

## Movement follower

When the new volume set point is received, the control derives the difference with the current volume and from there calculates the new PWM setting. The pumping stops when the set volume is reached. The Arduino constantly monitors and updates the calculated volume. Additionally, pressure can be monitored to indicate if a muscle has burst.

## Calibration

Each Arduino has a calibration mode that calibrates the function parameters for calculating the volume. In essence the calibration measures for a large number of PMW settings the time it takes for a muscle to go from empty to the point where the pressure drops, indicating it is completely full. From these measurements the function is derived that links PWM and time to delta Volume. Pressure is measured using the pressure sensors.

## Ambient lighting and sound

Assumption is that a multicolor led is controlled by digital pins of the Arduino, without need for a timer (precluding SPI interface). As yet is not clear how to assign colors to LEDs. Possibilities are to count crossing of a value (indicating a frequency) or amplitude values or a combination thereof.

The EEG signals may also be used as an input for modulation (frequency and amplitude). This also still has to be worked out. It is assumed that control is done from the server Arduino.

## Integration steps

The final vision uses Wi-Fi links. Due to available hardware and the wish to tackle issues one by one an intermediate step is defined, outlined below:

EEG device

Data thru

WiFi

PC

Signal Acq and slowed replay

USB controlled Arduino

USB controlled Arduino

2 pumps + pressure sensor

2 pumps + pressure sensor

In this setup two channels are recorded and slowed down. One channel’s amplitude is sent to one USB controlled Arduino, and the other to the other Arduino. The Arduino’s use this data to control volume as outlined above. The Arduino’s also have a calibration mode as outlined above.

## Software Interface between PC and Arduino

The software on the PC acquires signals from the EEG sensor. The number of channels can be configured. The channels may contain EEG signals and/or movement sense signals (the Brainaccess device provides acceleration parameters on three axes). The PC software slows these data (acquired at 125 samples per second) down at a settable rate. This is done in two ways:

1. Skipping samples (skipFactor 0 = no samples skipped, 1 = after each sample 1 sample is skipped and so on)
2. After the skipped samples apply a slow-down in time defined by readOutTime. readOutTime is given in mSec so 1000 means every second a sample is generated.

The software acquires a configurable number of samples from the Brainacces (at 125Hz, not configurable) and samples and slows it down.

At the given readOutTime the PC sends the following data to the Arduino main controller (via USB link):

TtNS NbrChan Chan0 Chan 1 Chan2 chanN

All these number are floats for ease of interpretation at the Arduino side.

TtNS = Time till Next Sample

NbrChan = Number of channels, typically 6 but may be extended. Absolute maximum is 16 channels EEG and 3 accelerometer data.

Values for Chan0 to ChanN vary between 0.0 (smallest amount of inflation for muscle) to 100.0 (maximum inflation).

The Arduino side is responsible for:

1. Distribution of the channel data to the relevant muscle controller.
2. Calculating the distance to be travelled and the speed at which is to be travelled derived from the ΔNewPos-OldPos and the TtNS
3. Providing and using the speed calibration (per muscle) needed to make the speed/distance calculation
4. It is also up to the Arduino side to use the information to control lighting and sound effects

Note: it is assumed that the Arduino is not interested in the type of signal (EEG or accelerometer) so this data does not need to be sent via the link. Obviously, the configuration is known (by setting the parameters in the GUI on the PC) and this knowledge can be used in the set up. It is conceivable that the Arduino gets a list of settings at startup of the connection (**Open for discussion**).

## Software design PC software

The PC software:

* Establishes contact with the Arduino controller and Brainaccess EEG module
* Reads the EEG signals, decimates them and slows them down, and sends them to the Arduino on a timer interval-based time

The following parameters can be set in the GUI:

* Number of EEG channels
* Number of Acceleration channels
* Decimation factor
* In between sample time for output
* Seconds of EEG data to access in one sample session

The GUI provides the following input possibilities

* Connect with Brainaccess button
* Connect with Arduino Button, input field for com port
* Settings for params above
* Start experiment

## Program design

**Communication**

Initialization

Start processes.

**GUI**

**Brainaccess**

Init:

* COM port
* timer

Send data

Send data timer interrupt

Init

* Wifi
* -datastream

Adaptive amplitude mapping

Send New data

Init data and settings

Inter process communication

Ring Buffer

Get Data

The Brainaccess module provides the following functions:

1. Get data from the Brainaccess device
2. Decimate data with setting SkipFactor
3. Adaptive amplitude mapping of the data for highest level in data reduces to 100%
4. Enforce a maximum dA/dT on ‘seam’ between last old and newest new data
   1. Make sure no ‘jumps’ between data fragments
5. Send most recently acquired data to buffer when buffer is half empty
   1. Upon bufferHalfEmpty signal from Communication

Communication has the following functions:

1. Start communication with device on serial port
2. Add timing and number of channels data to data from the ring buffer
3. Send this data from the ring buffer to the communication port at the given time (using the interrupt timer)
4. In case of changing timing settings (from GUI) change the timer settings
5. Signal buffer half empty signal