Connecting AsTeRICS and R/C toys - a practical approach to game accessibility for people with limited mobility conditions

Term paper submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Engineering at the University of Applied Sciences
Technikum Wien - Degree Program Computer Science

By: Alexander Frimmel
Student Number: 1110257061

Supervisor: Dipl.-Ing. Christoph Veigl

Vienna, 23.02.2014

FACHHOCHSCHULE
TECHNIKUM WIEN

Declaration

„I confirm that this thesis is entirely my own work. All sources and quotations have been fully acknowledged in the appropriate places with adequate footnotes and citations. Quotations have been properly acknowledged and marked with appropriate punctuation. The works consulted are listed in the bibliography. This paper has not been submitted to another examination panel in the same or a similar form, and has not been published. I declare that the present paper is identical to the version uploaded."

Vienna, 23.02.2014

Abstract

Serious gaming is an important approach to modern healthcare.
This paper defines the terms “Game Accessibility” and
analyses the state of the art application and its positive effects to the learning curve.

Further the Open-Source construction set for assistive technologies “AsTeRICS”
and its module system is introduced.
The practical part of this paper is the connection of a standard R/C car and AsTeRICS.
This papers main purpose is to develop a cheap solution to gain the possibility of
controlling a R/C car with any kind of input device supported by AsTeRICS.

In point of fact the real goal is to give people with mobility disabilities access to a more
comfortable way of learning to deal with new and alternative controls.

Keywords: AsTeRICS, Game Accessibility, R/C toys, alternative controls, ARE Teensy++2.0

Acknowledgements

My sincere thanks goes to supervisor Dipl-Ing Christoph Veigl for his support and his infinite patience. A very big appreciation to the developers of AsTeRICS for their volunteer contribution to such an amazing project.

Table of Contents

1 Hands on: Game Accessibility....6
1.1 Definition: What is Game Accessibility....6
1.2 Purpose of this paper....6
1.3 Play Therapy....6

2 Test setup....7
2.1 Logitech F710 PC Gamepad....7
2.2 AsTeRICS....7
2.2.1 ARE (AsTeRICS Runtime Environment)....8
2.2.2 Possible input methods....8
2.2.3 Module System....9
2.2.4 Models....10
2.2.5 ACS (AsTeRICS Configuration Suite)....11
2.2.6 CIM Protocol....11
2.3 Teensy++ 2.0....12
2.3.1 USB CDC (Univeral Serial Bus Communication Device Class)....13
2.4 Pulse-position modulated signal PPM....13
2.5 Walkera Devo7....14
2.6 Walkera RX701 receiver....14

3 Implementation....14
3.1 Initial test setup....14
3.2 Second and final setup....15
3.3 TeensyRC - AsTeRICS plugin development....16
3.4 Teensy++ 2.0 firmware development....16
3.4.1 Serial COM port....17
3.4.2 PPM signal generation....17
3.5 Results....19

List of Figures....21

List of Tables....21

List of Abbreviations....21

1 Hands on: Game Accessibility

1.1 Definition: What is Game Accessibility

The term “Game Accessibility” was first defined by the “International Game Developers Association” (IGDA) in 2004. IGDA addressed the lack of definition of “game accessibility”
and the GA-SIG (Game Accessibility Special Interest Group) defined it as followed:

“Game Accessibility can be defined as the ability to play a game even when functioning under limiting conditions. Limiting conditions can be functional limitations, or disabilities — such as blindness, deafness, or mobility limitations.” [1].

Many different technologies and interfaces exist to enable game experience to people with disabilities. This paper tries to apply this definition to common R/C toys.

1.2 Purpose of this paper

The probable combination of game accessibility and R/C toys will be analysed within this paper. Main practical goal is to implement a low-cost solution to use R/C toys with any kind of remote control supported by the AsTeRICS framework or its modules.

The setup will try to extend standard R/C toys to follow the definition of
game accessibility.
Of course the aim of this paper not completely viable as for example blind people would not be able to safely control R/C toys with the rudiments of this experiment.

Furthermore the possible practical application and its implementation will be examined.

1.3 Play Therapy

Play therapy is a modern approach to combine entertainment and therapy.

Many studies show significant results using play therapy in varying symptomatic situations and different kinds of game setups.

A good example is the case study: “Serious gaming to improve bimanual coordination in children with spastic cerebral palsy”.
In course of this study the children had to play three computer games that challenged the participants to move their hands according to different coordination patterns.

The results were significant: All children improved their performance, evidenced
by the scores they reached in the games.

“On average, the number of fishes fetched per minute (corrected for the difficulty setting) increased from 1.5 in the first three sessions to 7.7 in the last three sessions (t(6) = -3.9; p = .008, 2-tailed paired-samples ttest (sic!)).” [2]

2 Test setup

The setup of the test environment consisted of the following parts:

- Logitech F710 PC Gamepad
- Alienware MX17R3 Laptop running Windows 7 Home
- AsTeRICS software framework (ACS and ARE)
- AsTeRICS TeensyRC Module
- Teensy++ 2.0 microcontroller
- Walkera Devo7 sender
- Walkera MTC-01 Magic Cube
- Walkera RX701 reciever
- Car

Chapter 2 describes the chosen components and their tasks.
Chapter 3 explains the developed software and the test setups.

To be able to remove the standard R/C radio and replace it by alternative input controls the AsTeRICS framework in combination with a Teensy++2.0 microcontroller and a RF sender were used to create PPM signals for controlling up to 8 control channels.

2.1 Logitech F710 PC Gamepad

Illustration 1: Logitech F710 gamepad (http://gaming.logitech.com)

The Logitech F710 is a 2,4GHz wireless gamepad and provides DirectX and Xinput compatibility.

Xinput was used with the AsTeRICS joystickcapture plugin described in chapter 2.2.4.

For the test only the left and right control sticks were used.

2.2 AsTeRICS

"AsTeRICS is a free and Open-Source construction set for assistive technologies (AT). It allows the creation of flexible solutions for people with disabilities using a large set of sensors and actuators. " [3].

AsTeRICS is a highly configurable toolset for the creation of different assistive applications for many use cases.
It offers for example the possibility to use a PC via mouse and keyboard replacements, switch lights or use mobile phone features with alternative input controls.

The part of AsTeRICS which runs on the target PC is called ARE(**A**sTeRICS **R**untime **E**nvironment). It is a software written in Java and was deployed with JRE 7.0_45.

2.2.1 ARE (AsTeRICS Runtime Environment)

ARE is the core software which loads the OSGI plugins (described in chapter 2.2.3 as AsTeRICS modules) and runs the configured models on the target system.

As ARE is written in Java, ARE itself is completely platform independent.
The ARE is also responsible for the CIM (protocol stack for communication with peripheral devices, explained in chapter 2.2.6) initialization and communication and discovers CIM devices at ARE startup.

2.2.2 Possible input methods

The framework is capable of supporting many different kinds of input methods.
All of them are developed as AsTeRICS modules:
An extract of possible input methods:

- Standard PC Game Controllers
- **O**SK (On **S**creen **K**eyboards)
- Switches / Foot switches
- Trackballs
- Mouth Joysticks
- Eye tracking
- Face tracking
- Speech recognition
- Different Accelerometers

For this setup a usual PC gamepad is used.

2.2.3 Module System

Illustration 2 shows the concept of the AsTeRICS module system:

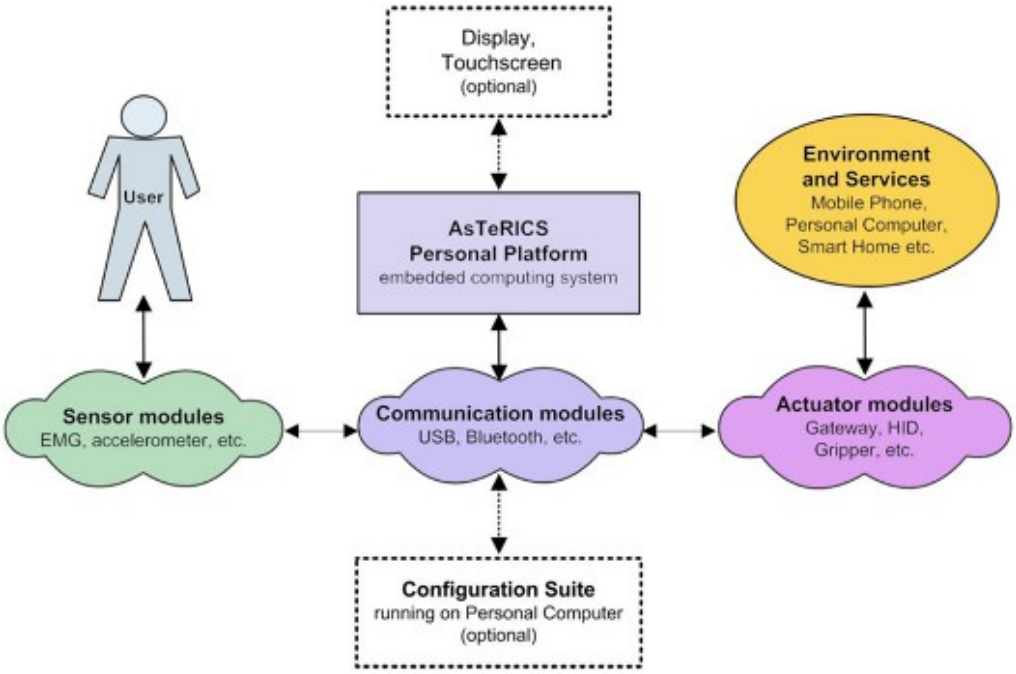

Illustration 2: AsTeRICS module system (http://www.asterics.eu)

ARE is extendable by plugins. The plugin framework is implemented via OSGi (*Open Services Gateway initiative,* an open specification to enable Java to build modularly software, further information: [4]) and a plugin can act as sensor, actuator or processor.

Via JNI (Java Native Interface) it is even possible to use C or C++ libraries or maybe platform dependent functions to develop a plugin.
Plugins are defined with properties, and ports for the communication with other plugins.

Properties hold preconfigured (in ACS before the ARE starts) values which are accessible within the plugin implementation as either integer, double, string or boolean. For example it could be used to configure plugin parameters.

Ports can be either input, output, eventlistener or eventtrigger ports.
Input ports receive either integer, double, string or boolean values to the plugin from another plugins output port.

The Amplifier plugin (Figures taken from ACS: Illustration 3, Illustration 4, Illustration 6, Illustration 1) for example receives a double value at its input port, multiplies the received value by a factor which can be configured with the property “factor” and sends the calculated value via its output port to the in the model (described in chapter 2.2.4) connected plugin.

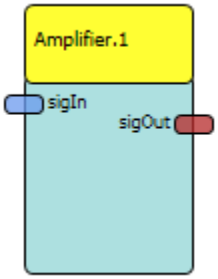

Illustration 3: Amplifier Plugin - Model shematic

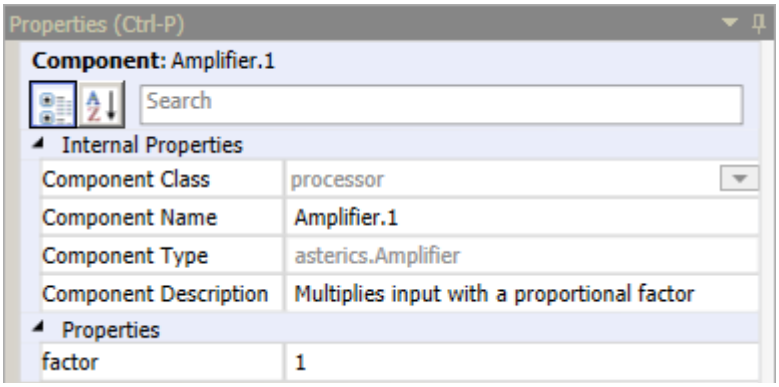

Illustration 4: Amplifier Plugin - Properties

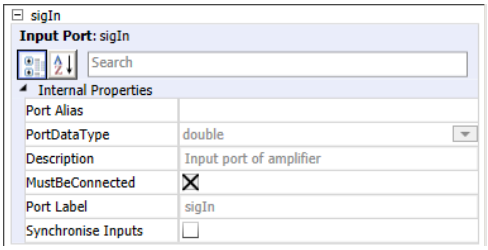

Illustration 6: Amplifier Plugin - Input port

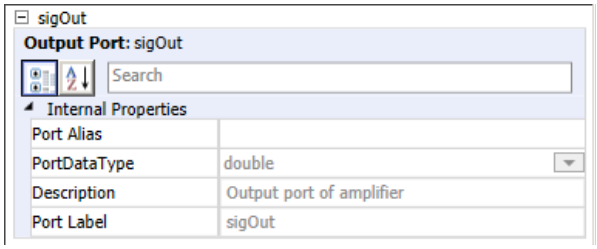

Illustration 5: Amplifier Plugin - Output Port

Input ports can be synchronized. This means that the values are only transmitted to the plugin when all input ports which are marked as “Synchronise Inputs” in the ACS – Input Port configuration already received values.

This offers the advantage that only a full set of values are received at a time and it is not necessary to implement the synchronization within the plugin.

If a single channel receives no values, all channels are empty and the plugin will do nothing.

2.2.4 Models

A Model is a combination of usually multiple different plugins which are linked together via ports. A model is the way how the user tells ARE what to do.
For this setup the model is quite simple and consists of the “Joystickcapture” and the developed TeensyRC Plugin. Illustration 7 shows the linked modules within the ACS model (Screenshot taken from ACS).

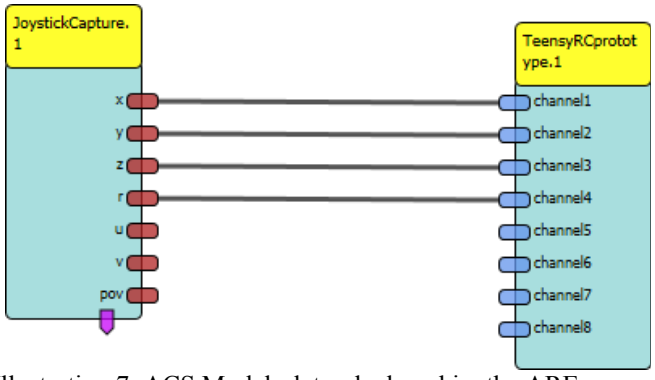

Illustration 7: ACS Model - later deployed in the ARE

2.2.5 ACS (AsTeRICS Configuration Suite)

With the “AsTeRICs Configuration Suite” (ACS), AsTeRICs comes with an easy to use “Plugin creation wizard” which could be used to set up a complete new plugin skeleton (described in chapter 2.2.3 and chapter 2.2.4) with properties, input-, output-, eventlistener- and eventriggerports.

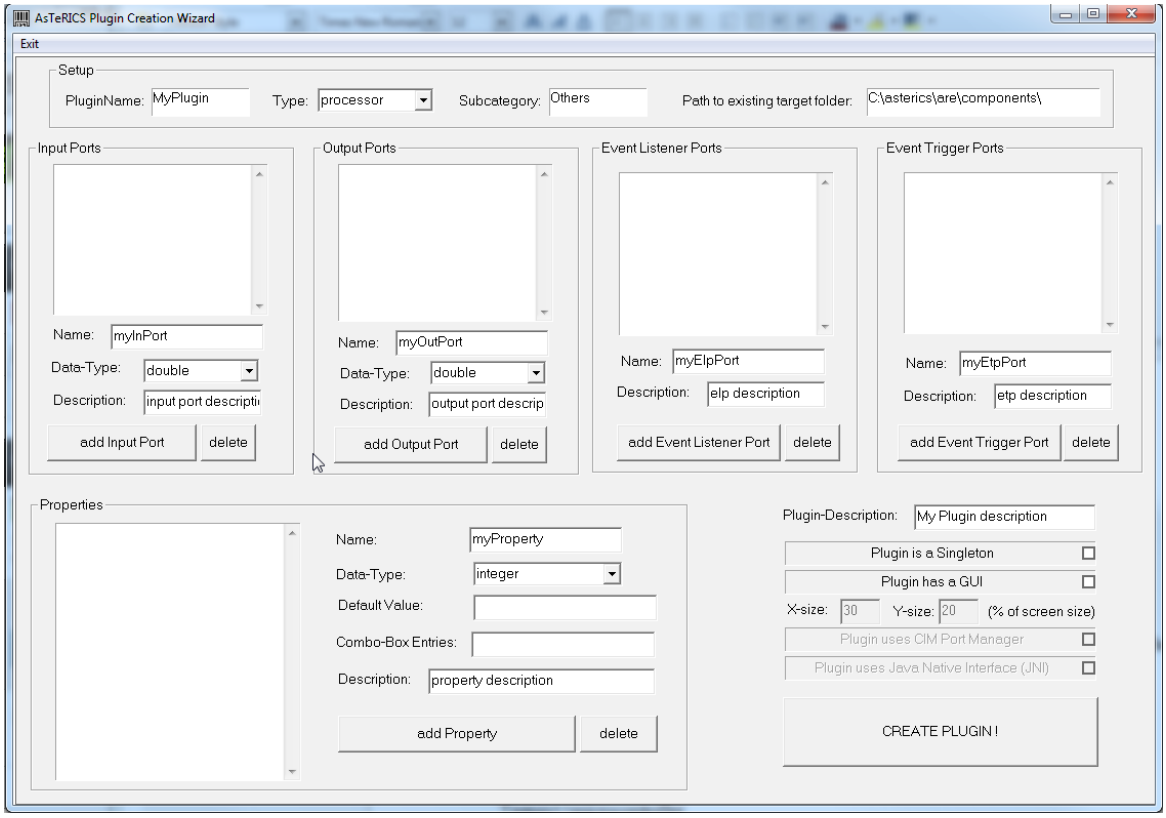

Illustration 8: AsTeRICS Plugin Creation Wizard

2.2.6 CIM Protocol

To communicate with peripherals the AsTeRICS framework defines the CIM protocol, a communication standard for device integration via COM ports.

All communication between actuators and sensors within the ARE and peripheral devices (such as the Teensy++ 2.0 in this setup) utilizes the CIM protocol.

Every peripheral component has its own Unique ID, which ARE uses to identify the components and is able to give access to the corresponding module.
The Teensy++ 2.0 firmware with the R/C firmware uses the dedicated ID: 0x05060708.

A CIM device can be capable of different actions, or functions. These functions are addressed through feature addresses. Each feature a CIM device provides needs a corresponding address. In this setup the feature address `0x0001` is used to set the received values for the PPM signal.

The following table shows a simplified view of the CIM protocol structure, for a detailed explanation of the CIM protocol please refer to the AsTeRICS Developer Manual [5].

Data field	Size in bytes	Short Description
Packet ID	2	Indentifies the beginning of the packet.
ARE ID (CIM ID)	2	The ARE-ID is sent from ARE to CIM and specifies the software version. The CIM-ID is the unique identifier for a CIM device.
Data size	2	Specifies the size of data contained in the optional data frame.
Serial packet number	1	ARE sends a serial number to the CIM to which a CIM specifically responds.
CIM-Feature address	2	Speciefies the function the CIM should execute.
Request Code (Reply code)	2	Holds command codes, respectively transmission mode and if sent from CIM to ARE error/status codes.
Optional data	0-2048	Each packet can contain optional data, in this setup the values of the gamepad joysticks are sent as optional data.

Table 1: Extract of CIM protocol field specification

2.3 Teensy++ 2.0

The Teensy++ 2.0 is a USB based microcontroller using a AT90USB1286 8 bit AVR 16 MHz processor. It comes with 46 usable IO ports and the most important reason, it is cheap, so the completed project may be available to a very low “do it yourself” price.

The Teensy++ 2.0 controller already has pins for solderless usage with a common breadboard:

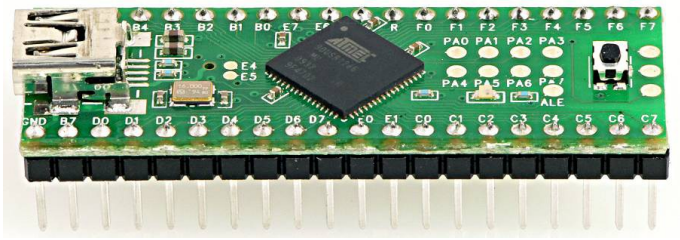

Illustration 9: Teensy++2.0 with pins
[6]

With the AVR compiler toolchain [7] and the Teensyloader [8] software it is fast and easy to compile and deploy an application on the device.

The AVR toolchain uses open source tools like the GNU GCC compiler [9] for C and C++ and brings a lot of predefined header files and functions for the AVR with it.

All the registers of the Teensy++2.0 are predefined and accessible with their in the shematic documentation [10] defined name using this headers.

2.3.1 USB CDC (Univeral Serial Bus Communication Device Class)

The USB CDC specifies the proper implementation of communication devices for USB. To clarify the meaning of this specification an extract of the definition is given:

"There are three classes that make up the definition for communications devices:

- *Communications Device Class*
- *Communications Interface Class*
- *Data Interface Class.*

The Communications Device Class is a device-level definition and is used by the host to properly identify a communications device that may present several different types of interfaces.

The Communications Interface Class defines a general-purpose mechanism that can be used to enable all types of communications services on the Universal Serial Bus (USB).

The Data Interface Class defines a general-purpose mechanism to enable bulk or isochronous transfer on the USB when the data does not meet the requirements for any other class."[11].

The Teensy++2.0 has implemented an USB-UART bridge which follows the definition of the USB-CDC class to be able to communicate via virtual COM port over USB.

Illustration 10 [12] shows the implementation by Atmel which is used by the Teensy++2.0 microcontroller.

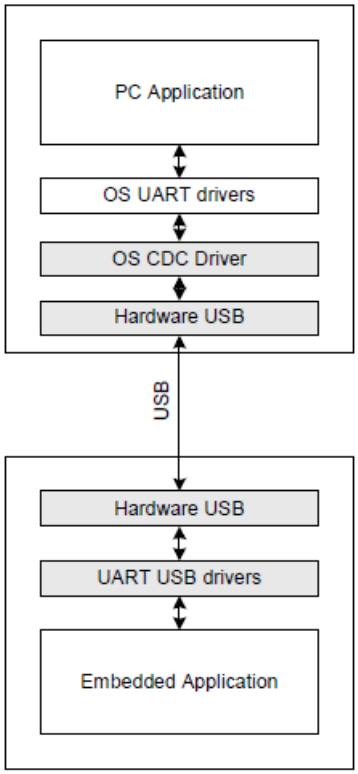

Illustration 10: RS-232 USB bridge, Atmel implementation

2.4 Pulse-position modulated signal PPM

To communicate with the RF sender a specified signal must be created. The specification for this PPM signal is described in this chapter.

The Teensy communicates with the RF sender using a specified PPM signal. Each 8 channel control message consists of a frame with 22.5ms length.

On every channel impulse (high) a 0.3 ms "stop" frame (low) is followed. The length of the channel impulse is dependent on the stick position and between 0.7ms (minimum value) and 1.7ms (maximum value) long. The "start" frame fills up the remaining time (up to 22,5ms) to complete a full 22.5 ms frame.

Illustration 11 [13] shows the signal in detail:

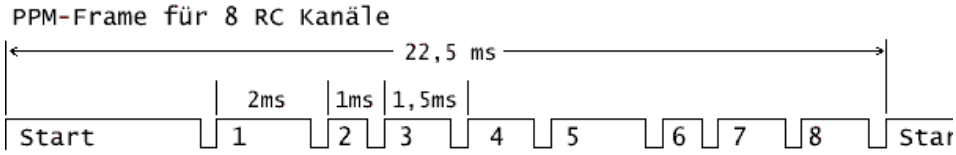

Illustration 11: PPM signal illustration

2.5 Walkera Devo7

Illustration 12: Walkera Devo 7

The Devo7 (Illustration 12 [14]) from Walkera is a usual R/C sender with a built-in trainer slot.

A Trainer slot is used to connect a trainees sender to the trainers. A trainer can gain control over the trainees R/C device instantly.

In first setup (see chapter 3.1) the Teensy was connected to the Devo7's trainer slot to send the PPM signal to the R/C car.
The trainer slot was the important part of the device, which was used to pass the input signals to the reciever via RF.

No other features except the basic RF radio were of special interest.

2.6 Walkera RX701 receiver

The RF receiver (Illustration 13 [15]) used was the Walkera RX701.
It is a simple 8 channel R/C receiver working with the Magic Cube at 2.4GHz.

For the test with “the car” only the two channels AILE and Throttle for the steering servo and the engine were used.

Illustration 13: Walkera RX701 Receiver (www.walkera.com)

3 Implementation

3.1 Initial test setup

The first test setup consisted of the following parts:

- Logitech F710 PC Game Controller
- Alienware MX17R3 Laptop running Windows7 Home
- AsTeRICS ARE Framework
- AsTeRICS TeensyRCprotoype Module

- Teensy++ 2.0 microcontroller
- Walkera Devo7 sender with trainer slot
- Turnigy 110BS Monster Beetle R/C car

In the following, a chain of actions which are needed to get the input signal of the input device (in this case the Logitech F710 gamepad) to the R/C car will be described.

Illustration 14 shows the flow in a detailed way.

At first a control action on the gamepad occurs.

The inputs of the wireless gamepad which is connected to the computer, usually X and Y axis of the left and right sticks are captured by the AsTeRICS "JoystickCapture" plugin and processed to the TeensyRC plugin as shown in Illustration 7.

The ARE passes the input to the TeensyRC module, which in turn sends the values encapsuled in proper CIM messages to the Teensy++2.0 microcontroller.

Teensy parses the CIM message and generates the PPM signal which is output to the GPIO pin described in chapter 3.4.

The RF radio sends the signal via RF to the RC701 receiver which controls the engine and the servo of the car.

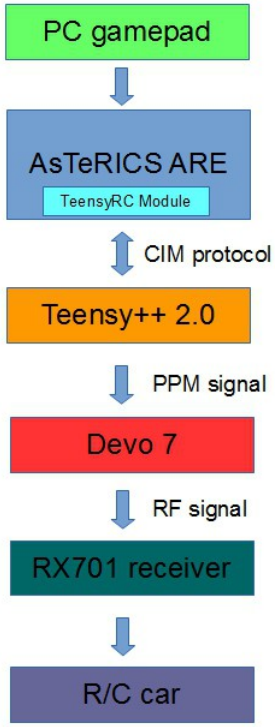

Illustration 14: Shematic figure of the control flow

The setup worked, but there were two significant disadvantages:

- Devo7 (or other R/C sender with trainer slot) required
- Only working when trainer switch held down on Devo7

To minimize the costs and complexity of the setup and to make it more comfortable (permanently holding the trainer switch to activate the signal on the Devo7) a new setup was tested with a simple RF sending module as described in the next chapter.

3.2 Second and final setup

After the successful basic test (the car moves using the Logitech F710 gamepad) a new FM sender was tested.
The Devo7 was completely removed from the test and replaced by the "Walkera Magic Cube MTC-01" (shown in Illustration 15 [16]) RF sender.

The Walkera Magic Cube was originally designed as a smartphone addition which enables a typical smartphone to control R/C toys.

For this paper the Magic cube was connected and controlled by the Teensy++2.0.

Illustration 15: Walkera MTC-01 (www.hobbyking.com)

"Specification:

1. *Frequency: 2.4G FH*
2. *Output power: less than 10mW*
3. *Overall unit Current: less than 35mA*
4. *Apply battery: 3.7V 80mAh*
5. *Weight: 10g" [17]*

The final setup consisted of the following parts:

- Logitech F710 PC Game Controller
- Alienware MX17R3 Laptop running Windows7 Home
- AsTeRICS ARE Framework
- AsTeRICS TeensyRCprotoype Module
- Teensy++ 2.0 microcontroller
- Walkera MTC-01 Magic Cube sender
- Turnigy 110BS Monster Beetle R/C car

3.3 TeensyRC - AsTeRICS plugin development

The setup of the development environment is explained within the AsTeRICS Developer Manual [5].

The TeensyRC plugin is the developed module loaded by ARE which receives control channel values from the "JoystickCapture" plugin and sends it to the Teensy++2.0 via CIM.

The 1bit integer values received by the TeensyRC plugin are masked using simple bitshifting operations in Java and prepared for sending via the USB serial connection:

```
(byte) (leftX & 0xff)
```

Important to note is the use of synchronized input ports(see chapter 2.2.3). This is possible by implementing and overriding the method

public void syncedValuesReceived(HashMap<String, byte[]> dataRow)

and enabling the "Syncronize Inputs" option in the ACS before the model gets deployed.

3.4 Teensy++ 2.0 firmware development

Atmel AVR Studio 4 with WinAVR compiler (as described in chapter 2.3) was used to develop the microcontroller software for CIM protocol communication and PPM generation.

3.4.1 Serial COM port

To open a serial COM port over USB on the Teensy the “usb_serial” library [18] was chosen. It worked out of the box for the Teensy++ 2.0 device.

An INF file [19] needs to be deployed on the host, that the windows driver installer initializes the controller correctly.

The corresponding *usb_init()* function sets up a serial COM port with the pc.

First of all it was important to implement the CIM protocol, that ARE was able to identify the Teensy as CIM device during startup.
This means the Teensy needs to send CIM messages before the ARE starts.

Illustration 16 shows the CIM registration from Teensy to CIM in a detailed way.

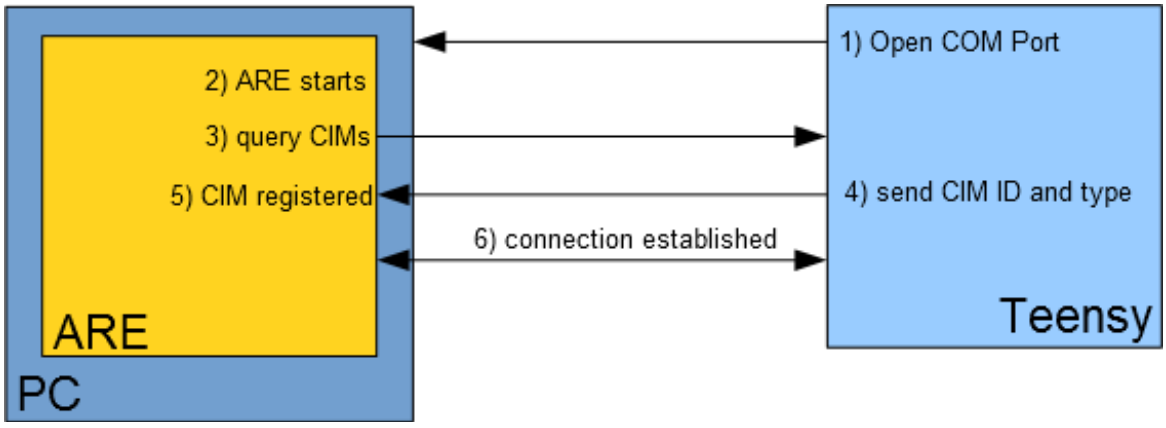

Illustration 16: CIM registration

At the Teensy it was implemented to listen for received messages at the established COM port, as the ARE send query messages to all COM devices during startup to detect connected CIM modules and their type.

With *usb_serial_available()* it is possible to watch for pending incoming messages from ARE on the Teensy.

When a ARE CIM frame is detected the appropriate answer (Teensys CIM ID and type) will be sent using *usb_serial_write()*.

3.4.2 PPM signal generation

In the first step the received CIM message containing the values of the channels which needs to be generated as PPM signal will be parsed and saved in memory.

The channels are received as 1 bit integers, that means it is a range from 0 to 255. These values are now calculated to fit into a ppm frame (as described in chapter 2.4).

The conversion from 0-255 to a value in microseconds is done using a simple mathematical conversion:

*channel1frame = channel1 * 3.1372 + baseframe;*

with channel1 as the received integer value between 0 and 255 and baseframe = 700 mircoseconds adequate for each channel.

To output the correct PPM signal on the GPIO pin D0, timer and compare registers needs to be initialized.

```
TCCR1A |= (1 << COM1A0);
// set mode to CTC with compare to OCR1A
TCCR1B |= (1 << WGM12);
// set /8 prescale = 2000 clockcycles/ms
TCCR1B |= (1 << CS11);
// enable ouptut compare A interrupts
TIMSK1 |= (1 << OCIE1A);

OCR1A = 2000;
```

Setting the above registers will result in an enabled Timer1 which will run 2000 clockcycles every millisecond due to the configured prescaler /8.

Further a “TIMER1_COMPA_vect” interrupt will be generated if the value of Timer1 will match the value set in the OCR1A register (output compare mode), after 1ms for the first time.

Within the “TIMER1_COMPA_vect” - ISR(Interrupt Service Routine) the GPIO is controled and the new timer values are set.

With “*OCR1A = times[timecounter]*2;”* the value for the next interrupt (when the actual low or high frame ends) is set and the GPIO pin D0 is set to low or high:

```
if(timecounter % 2)
        PIN0_ON;
else
        PIN0_OFF;
```

where PIN0_ON and PIN0_OFF are makros, defined to improve readability:

```
#define PIN0_ON      (PORTD |= (1<<0))
#define PIN0_OFF     (PORTD &= ~(1<<0))
```

Using these built-in hardware timers a signal with enough accuracy could be generated on GPIO pin D0 of the Teensy++2.0.

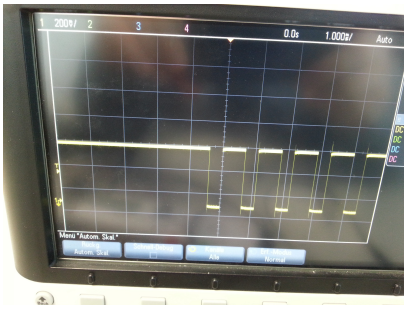

Illustration 17: PPM frame measured with the oscilloscope

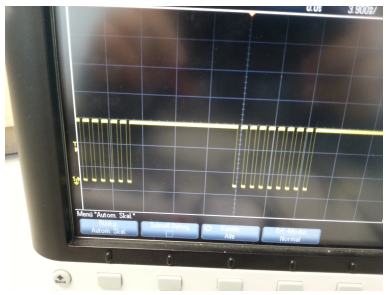

Illustration 18: PPM frame measured with the oscilloscope

3.5 Results

It is possible to control a usual R/C toy with alternative input controls. The tests showed that the Teensy++2.0 microcontroller is able to generate an accurate PPM signal to drive a RF sender (Walkera MTC-01).

The combination of AsTeRICS, Teensy++2.0 microcontroller and the according developed software modules (for Teensy and AsTeRICS (Sourcecode available: [20])) enables people with limited mobility conditions using R/C toys.

For this paper and as an initial step into the right direction all tests were done under laboratory conditions to prove the general capability of combining Game Accessibility and R/C toys.

The test setups were not convenient for real life usage, because the controls developed in this setup are too sensitive and it could be dangerous when using without extra safety precautions. With further development and finetuning it is possible to create an easy to setup package for home use or playtherapy structures.

A live example was prepared using the ARE plugin “FacetrackerLK” and the TeensyPrototype to drive the car.

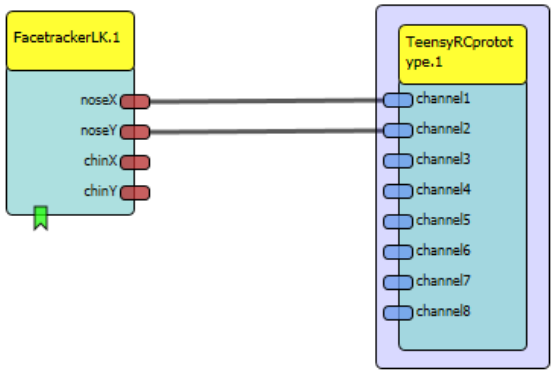

Illustration 19: Facetracker tests

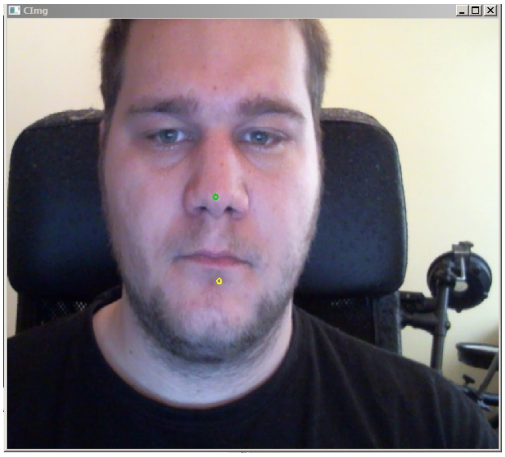

Illustration 20: Facetracker with nose and chin recognition points

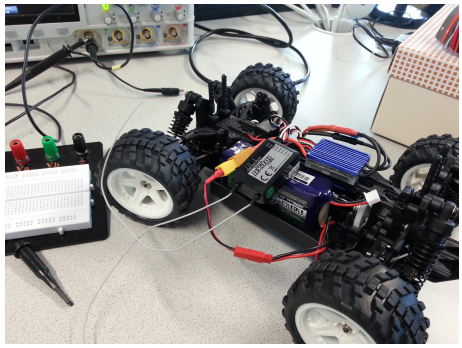

Illustration 21: Assembled car

The generic control mechanism was working as expected, but for an inexperienced person it is almost impossible to drive the car in useful way.

In conclusion this setup, when further developed, could be used in professional Play Therapy setups or even to control R/C toys in private using different input controls.

Bibliography & Links

1: IGDA GA-SIG Whitepaper, 12.01.2014, http://gasig.files.wordpress.com/2011/10/igda_accessibility_whitepaper.pdf

2: Serious gaming to improve bimanual coordination in children with spastic cerebral palsy, E.C.P. van Loon, C.E. Peper, Research Institute MOVE,Faculty of Human Movement Sciences, VU UniversityAmsterdam, The Netherlands

3: Official AsTeRICS website, 12.01.2014, www.asterics.eu

4: OSGi, 12.01.2014, http://www.osgi.org/

5: AsTeRICS Developer Manual, 12.01.2014, http://www.asterics.eu/download/DeveloperManual.pdf

6: Illustration: Teensy++2.0 with pins, 12.01.2014, http://www.pjrc.com

7: WinAVR, 12.01.2014, http://sourceforge.net/projects/winavr/

8: Teensy Loader, 12.01.2014, http://www.pjrc.com/teensy/loader.html

9: GNU GCC , 12.01.2014, http://gcc.gnu.org/

10: Teensy--2.0 pins and registers, 12.01.2014, http://www.pjrc.com/teensy/card4a.pdf

11: USB CDC Specification, 12.01.2014, http://www.usb.org/developers/docs/devclass_docs/CDC1.2_WMC1.1_012011.zip

12: RS-232 USB bridge, Atmel implementation, 12.01.2014, http://www.atmel.com/Images/doc4322.pdf

13: PPM signal illustration, 12.01.2012, http://www.mftech.de/ppm.htm

14: Devo7, 12.01.2014, www.walkera.com

15: Walkera RX 701 Receiver, 12.01.2014, www.walkera.com

16: Walkera MTC-01 Magic Cube, 12.01.2014, www.hobbyking.com

17: Walker MTC-01 Magic Cube Specification, 12.01.2014, http://www.walkera.com/en/showgoods.php?id=446#txt2

18: Teensy USB Serial library, 12.01.2014, http://www.pjrc.com/teensy/usb_serial.html

19: USB serial Installation for Teensy, 12.01.2014, http://www.pjrc.com/teensy/serial_install.exe

20: Sourcecode of this setup, 22.2.2014, http://www,sneer.at/TeensyRCPrototype.tar.gz

List of Figures

Illustration 1: Logitech F710 gamepad (http://gaming.logitech.com).....8
Illustration 2: AsTeRICS module system (http://www.asterics.eu).....9
Illustration 3: Amplifier Plugin - Model shematic.....10
Illustration 4: Amplifier Plugin - Properties.....10
Illustration 5: Amplifier Plugin - Input port.....10
Illustration 6: Amplifier Plugin - Output Port.....10
Illustration 7: ACS Model - later deployed in the ARE.....11
Illustration 8: AsTeRICS Plugin Creation Wizard.....12
Illustration 9: Teensy++2.0 with pins.....13
Illustration 10: RS-232 USB bridge, Atmel implementation.....15
Illustration 11: PPM signal illustration.....15
Illustration 12: PPM frame measured with the oscilloscope.....16
Illustration 13: PPM frames measured with the oscilloscope.....16
Illustration 14: Walkera Devo 7.....16
Illustration 15: Walkera RX701 Receiver (www.walkera.com).....16
Illustration 16: Shematic figure of the control flow.....17
Illustration 17: Walkera MTC-01 (www.hobbyking.com).....19
Illustration 18: CIM registration.....21
Illustration 19: Facetracker tests.....23
Illustration 20: Facetracker with nose and chin recognition points.....23
Illustration 21: Assembled car.....23

List of Tables

Table 1: Extract of CIM protocol field specification.....13

List of Abbreviations

WWW	World Wide Web

AsTeRICS	Assistive Technology Rapid Integration & Construction Set
R/C	Remote Controlled
IGDA	International Game Developers Association
GA-SIG	Game Accessibility Special Interest Group
PPM	Pulse-Position Modulated
RF	Radio Frequency
AT	Assistive Technologies
ARE	AsTeRICS Runtime Environment
JRE	Java Runtime Environment
OSGi	Open Services Gateway initiative
CIM	Common Information Model
OSK	On Screen Keyboard
JNI	Java Native Interface
ACS	AsTeRICS Configuration Suite
COM port	Communication port
USB CDC	Universal Serial Bus Communication Device Class
USB-UART	Universal Serial Bus Universal Asyncronous Reciever Transmitter
GPIO	General Purpose Input Output
ISR	Interrupt Service Routine