Game application:	Points:	_____ / 7
Bluetooth/UART:	Points:	_____ / 5
Music Playback (MP3 and Volume):	Points:	_____ / 4
GLCD:	Points:	_____ / 5
Rand,ADC:	Points:	_____ / 4
Overall:	Points:	_____ / 25

Microcontroller VU

Application Protocol

Jan Nausner, MatrNr. 01614835
e01614835@student.tuwien.ac.at

November 27, 2018

Declaration of Academic Honesty

I hereby declare that this protocol (text and code) is my own original work written in my own words, that I have completed this work using only the sources cited in the text, and that neither this protocol nor parts of it have ever before been submitted to this or any other course.

_____ _____
(Date) (Signature of Student)

Admission to Publish

☐ I explicitly **allow** the publication of my solution (protocol and sourcecode) on the course webpage.

☐ I **do not allow** the publication of my solution (default if nothing is checked).

_____
(Signature of Student)

Contents

Note: This template is provided to show you how LaTeX works and may not contain all subsections your protocol should contain.
Note: You can, and in fact should, reuse appropriate parts from the implementation proposal in the protocol.

1 Overview

1.1 Connections, External Pullups/Pulldowns

Pin Assignment

What	
J12	Connected to VCC
J14	EXT
J15	PF0
LEDs	Off
GLCD backlight	On
PORT switches	Off
PORT jumpers	Pull down height

Write down all things we need to know to get your program running on our board. All non-standard external connections, all switches your program needs, ... If we cannot figure out how we get your program running, we can not give you points for it.

1.2 Design Decisions

Here comes the design decisions that you made during programming.

1.3 Specialities

Does you solution have something special (positive or negative)?

2 Main Application

Describe your application.

3 Music Playback

3.1 SPI

Explain your modules.

3.2 Playback

4 LC–Display

4.1 GLCD

Explain your modules.

4.2 HAL GLCD

5 ...explain your application modules ...

6 ...the above were only examples

7 Problems

One problem that was encountered during the development of the application was the faulty function wiiUserSetAccel. In the specification it was stated that using this function it would be 4possible to either turn the wiimote accelerometer on or off. Unfortunately this behavior was not reflected in the implementation, which only turned the accelerometer on, regardless of the value of the enable parameter.

Another difficulty concerned the interpretation of the accelerometer data. It took some time and thinking to derive a simple and fast algorithm for wiimote tilt detection which did not require the use of complex numerics or even floating point arithmetic. To understand the principal behavior of an accelerometer, the following article was consulted: http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/.

A lot of time had to be in invested in figuring out how to communicate with the GLCD module. The exact timing and pin level sequence had to be found out by trial-and-error, as the datasheets for the hardware were not completely accurate.

Finally, just before uploading the solution a bug was discovered, where the program started to freeze or reset the board after running a considerably long time. After many hours of debugging the error was found to be hidden in a for loop counting from high to low numbers. The problem was that one part of the loop condition was checking for the index to be greater or equal to zero and with the index being an unsigned integer, this sometimes resulted in an endless loop.

8 Work

Estimate the work you put into solving the Application. You can add additional points, if you like.

Task	Assumption (IP)	Reality
reading manuals, datasheets	5 h	5 h
program design	10 h	6 h
programming	20 h	16 h
debugging	20 h	60 h
protocol	5 h	4 h
Total	60 h	91 h