schlizbäda

User's Manual

EsploraGamingController

Gaming Controller based on the Arduino Esplora Board

date:
Jan 31th, 2019

Free as in Freedom

This document was created using the typesetting system LaTeX

Contents

List of Figures

List of Tables

1. Preface

Thank you for interesting in schlizbäda's EsploraGamingController.
This is a control unit for PCs, which can emulate keyboard and mouse actions and is inspired on the gaming controllers of various game consoles. It is based on the Arduino board *Esplora* which is technically a *Arduino Leonardo* and it already contains some sensors and actuators. The printed circuit board was designed roughly in the form of a gaming controller.

https://store.arduino.cc/arduino-esplora

1.1. Legal Notes

When designing the EsploraGamingController, it was taken care to use only software that is provided under a free license such as GNU GPL or similar or that has been released into the public domain.

Trademarks

Some names used in this document may be trademarks. The use of these trademarks by third parties for their purposes may infringe the rights of the holders.

Links

This manual contains links to external sites on the internet. Despite linking the author schlizbäda does not appropriate these contents, since they are not within his sphere of influence! At the time of linking there were no illegal contents noticeable. It is not reasonable for the author to check the links permanently for any changes that might violate the law. However, if current or future content should be illegal, the author may be contacted by e-mail to mailto:himself@schlizbaeda.de. Appropriate actions will then be taken to remove the affected link(s).

Licensing of this project's components

Code:
Most Arduino code examples are in the public domain and may be used by any person. The code for EsploraGamingController was developed from the templates
https://www.arduino.cc/en/Tutorial/EsploraJoystickMouse,
https://www.arduino.cc/en/Tutorial/EsploraKart and
https://github.com/circuit69/EsploraTinkerkit
Due to many adjustments and improvements the author decided to publish his work under the GNU GPL v3:
https://www.gnu.org/licenses/gpl-3.0.en.html

Housing:
A CAD file collection for 3D printing was found at https://www.thingiverse.com/thing:45880. This data is published under CC BY-NC 3.0 (http://creativecommons.org/licenses/by-nc/3.0/). This license does not allow commercial use.

Image Copyright

All pictures, photographs and technical drawings were created by the author himself. They are published under the *Creative Commons* license **CC BY-SA 3.0**. Therefore anyone may use these images in original form or adapted by giving name attribution of the author.

The Arduino logo is in the public domain:

1.2. Remarks to a genderwise correctly written text

Grammatical gender differentiation is very unusual in English language compared to German language. In German speaking countries there is currently a trend of *political correctness* to mix the grammatical gender with the biological sex. This results in *genderwise correctly* written texts which are sometimes really difficult to read.
This translation isn't affected by this German gender trend anyway. So this may be regarded as a typical German problem

1.3. Achnowledgements

schlizbäda would like to welcome the user **@dale** (https://forum-raspberrypi.de/user/23726-dale/) from the German Raspberry Pi forum (https://forum-raspberrypi.de):
He printed the housing on his 3D printer.

1.4. Abstract

The EsploraGamingController is a control unit with USB2.0 connection inspired by the game console based gaming controllers. Instead of proprietary signals this device transmits keyboard and mouse commands when the user actuates the switches of the device.

Due to its open-source and free programming, it can be adapted to many applications in the source code by assigning the desired keyboard codes or mouse commands to the control elements (sensors) of the device.

There are eight different assignments (so-called modes) for several applications which can be switched during operation. The selected assignment is indicated by the colour of the three-colour LED on the Esplora board.

The software of the EsploraGamingController emulates a USB keyboard and a USB mouse and can be used immediately on most of the PC operating systems like *Windows* or *GNU/Linux* as well as on a Raspberry Pi under *Raspbian* without any additional driver installation.

2. Operating Instructions

This chapter is a classical operator's guide for users who intend to use the EsploraGaming-Controller straight forward without modifying the software or hardware of the device.

2.1. Controls of EsploraGamingController

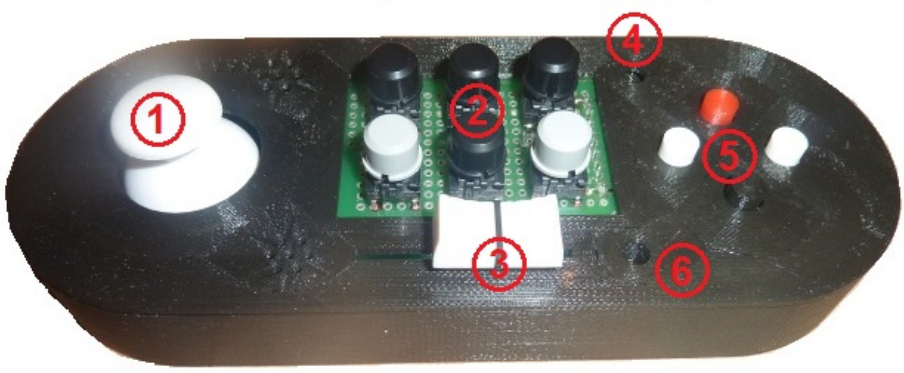


Figure 2.1.: *Controls (Sensors) of EsploraGamingController*

The listed controls of EsploraGamingController emulate keyboard and mouse:

num.	Element	Function
1)	joystick	mouse movement
2)	TinkerKit switches	mouse buttons, keyboard
3)	slider	mouse scroll wheel
4)	light sensor	*usually disabled*
5)	cursor pushbuttons	cursor key codes
6)	three-colour LED	shows the selected assignment

Table 2.1.: *Controls (Sensors) of EsploraGamingController*

Joystick

The joystick usually emulates mouse movements. Depending on the displacement of the joystick, the speed of the mouse movement is adjusted. Pressing the joystick emulates a click on the left mouse button.
On the Esplora board, the joystick is converted by two potentiometers, whose position is read via A/D converters of the built-in 8-bit ATMEL microcontroller *ATmega32U4*. When the joystick is pressed, a push button is actuated which is digitally read.

TinkerKit Switches

On the EsploraGamingController there is a small additional printed circuit board at the place where a small display would actually be installed. On this additional board there are six pushbuttons, which are connected to the four TinkerKit connectors. The white TinkerKit connectors are implemented as analogue inputs, so that a sophisticated voltage divider combination can be used to differentiate which key combination is pressed.
These six buttons are assigned to additional keyboard codes or mouse clicks.

Slider

The slider control is a potentiometer connected to an A/D converter of the ATMEL microcontroller. It usually emulates the scrolling wheel of the mouse.
Of course simply releasing the slider is not sufficient to stop the emulated scrolling wheel! Rather, the controller must be pushed a little bit in the opposite direction to stop the scrolling process.

Light Sensor

The light sensor is connected to another A/D converter of the ATMEL microcontroller. Some tests of the light sensor showed quickly that it cannot be implemented as a kind of actuator. Even lightweight shadowing will result in missinterpretations. Nevertheless the Arduino sketch still contains some software routines to implement a control element. However, the light sensor is disabled in most assignment modes.

Cursor Pushbuttons

These four pushbuttons are digitally read by the ATMEL microcontroller. They emulate keyboard codes to control the cursor or for moving a game character by emulating the keys A, W, S and D. Many computer games expect these keys.

Three-coloured LED

The three-coloured LED shows the selected controller assignment (mode) by changing its colour. See section 2.2

2.2. Selecting the Assignment Mode of the Actuators

Pressing all four cursor pushbuttons simultaneously starts the setting mode of the Esplora-GamingController. Using the UP and DOWN pushbuttons will change the assignment of the acutators. The colour of the three-colour LED changes according to the currently selected assignment.
By pressing all four cursor pushbuttons simultaneously again, the setting mode of the device will be quit. To prevent accidental changes of the selected assignment mode, it is recommended to start with pressing the cursor pushbutton LEFT or RIGHT and then pressing the others.

num.	**LED colour**	**denomination**
0)	*"dark white"*	common 1
1)	red	MinecraftPi
2)	green	yamuplay *t.b.d. (to be defined)*
3)	yellow	«undefined 3»
4)	blue	«undefined 4»
5)	magenta	«undefined 5»
6)	cyan	«undefined 6»
7)	white	common 2

Table 2.2.: *The Eight Assignment Modes for the Actuators of EsploraGamingController*

common 1: LED: "dark white" (low light emission)
In this assignment mode the actuators emulate solely keyboard codes. No mouse functions are used.

joystick	cursor keys
joystick pushbutton	ENTER key
slider	Pg-up, pg-dn
light sensor	*inactive*
TinkerKit orange	SPACE key, ESC key
TinkerKit white	modifier keys SHIFT, CTRL, ALT

MinecraftPi: LED: red

The assignment is adapted for the game *MinecraftPi* for the Raspberry Pi.

joystick	mouse movement (to adjust the viewing angle)
joystick pushbutton	key `E` to open the Minecraft block select menu
slider	mouse scrolling wheel for quick block selection
light sensor	*inactive*
TinkerKit orange	left and right mouse button
TinkerKit white	keys ENTER, SHIFT, ESC, SPACE

yamuplay: LED: green

Here it is planned to implement an assignment for controlling schlizbäda's media player *yamuplay.py*.

https://github.com/schlizbaeda/yamuplay

Currently this is an undefined assignment mode!

«undefined 3»: LED: yellow

This is an undefined assignment mode!

«undefined 4»: LED: blue

This is an undefined assignment mode!

«undefined 5»: LED: magenta

This is an undefined assignment mode!

«undefined 6»: LED: cyan

This is an undefined assignment mode!

common 2: LED: white

This mode emulates keyboard and mouse actions.

joystick	mouse movement
joystick pushbutton	left mouse button
slider	mouse scrolling wheel
light sensor	*inactive*
TinkerKit orange	left and right mouse button
TinkerKit white	modifier keys SHIFT, CTRL, ALT

3. Modification of EsploraGamingController

This chapter describes how to "mod" this project:

- The hardware section primarily describes the installation of the additional circuit board containing the TinkerKit pushbuttons.
- The software part assumes a successful installation of the Arduino IDE on the working computer and further the knowledge of flashing an Arduino board using the IDE.

3.1. Description of the additional TinkerKit Pushbuttons Board

Modern gaming controllers usually provide more switches than the four cursor pushbuttons. There are often so-called shoulder buttons in front of the controller's housing which are pressed by the forefingers.

The EsploraGamingController enables the four TinkerKit connectors to add more pushbuttons. Since the white TinkerKit connectors are wired to input pins of the *ATmega32U4* microcontroller, which provide a 10bit A/D converter facility, several pushbuttons can act **simultaneously** on each white TinkerKit connector. They are distinguished by suitable voltage dividers:
On the prototype of the additional board each analogue TinkerKit connector (white) provides two pushbuttons using the voltage dividers 33k : 22k resp. 33k : 47k. This results in the following voltages and A/D values:

pushbutton	**A/D value**	**voltage**
no button	1023	5,00V
47k button	601	2,94V
22k button	410	2,00V
both buttons	320	1,56V

Table 3.1.: *A/D Values of the TinkerKit Pushbuttons*

The following pictures show the connection of the prototype board at the EsploraGaming-Controller. In the appendix there can be found a wiring conception and the technical drawings

of a printed circuit board which provides up to three pushbuttons on each white TinkerKit connector.

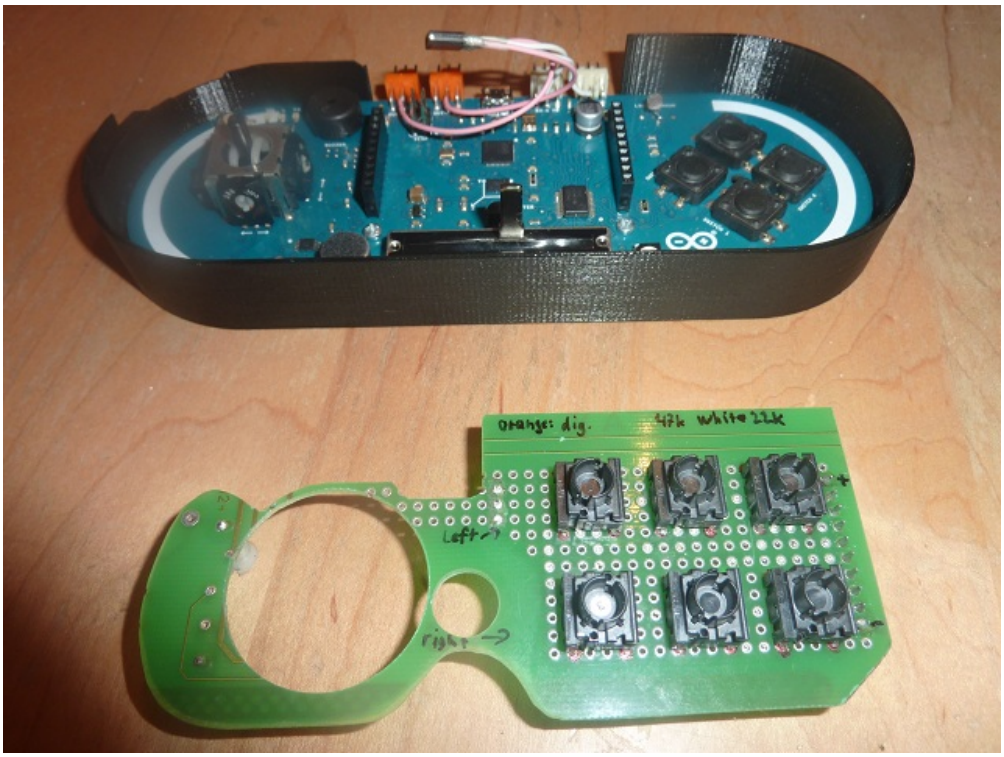


Figure 3.1.: *Arduino Esplora and Additional TinkerKit Board (Prototype)*

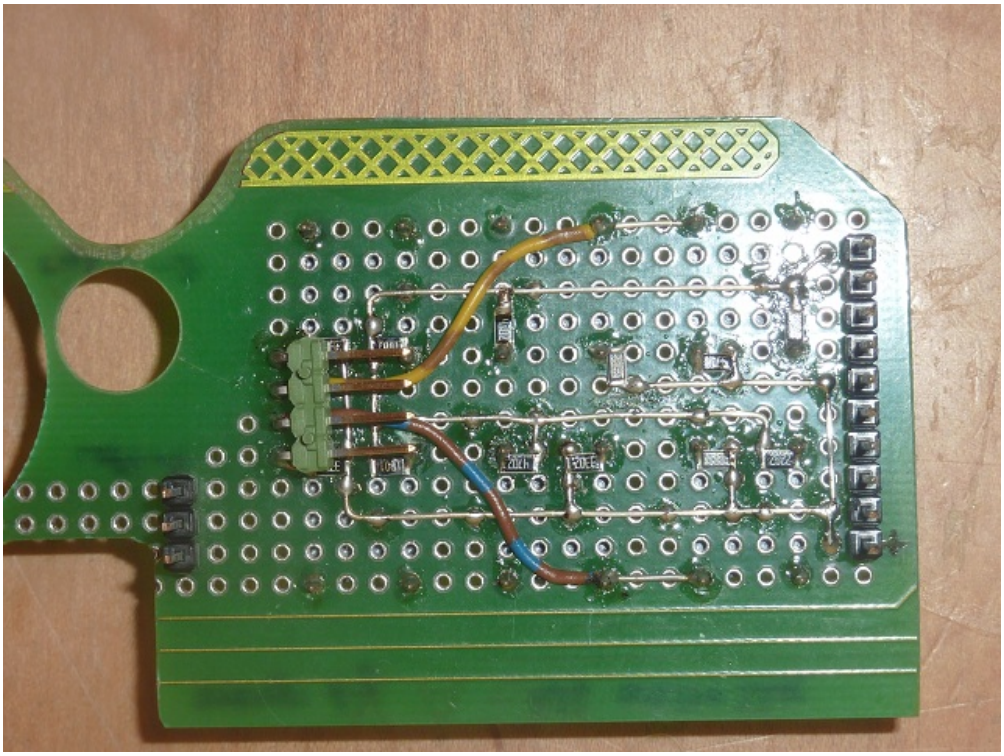


Figure 3.2.: *Resistors of the Additional TinkerKit Board*

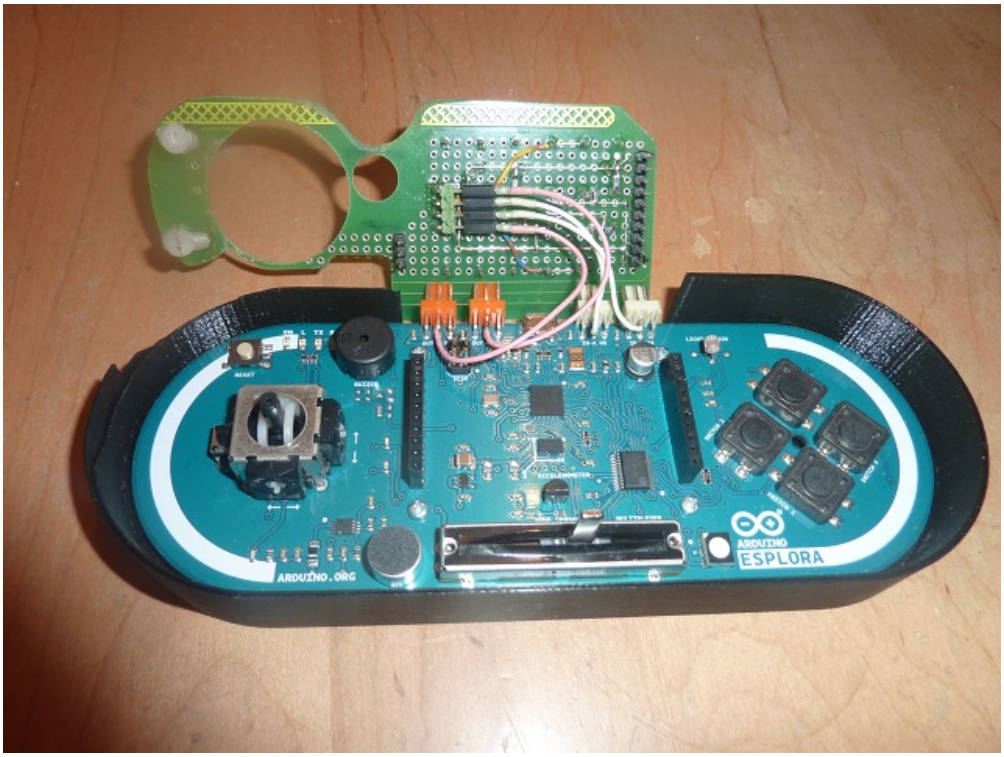


Figure 3.3.: *Connection of the TinkerKit Board at the Arduino Esplora*

Figure 3.4.: *Additional TinkerKit Board Inside the Housing*

3.2. Description of the Arduino Sketch for EsploraGamingController

3.2.1. Installation of the Arduino-IDE

A complete description of how to use the *Arduino Web Editor* or the *Arduino Desktop IDE* can be found at https://www.arduino.cc/en/Guide/HomePage. Each user should decide the preferred variant individually.

3.2.2. Code Adjustments

An Arduino Edsplora Board flashed with the downloaded sketch `EsploraGamingController.ino` emulates a USB mouse and a USB keyboard as described in chapter 2.

The easiest way to adjust the code is to change the values of the two-dimensional array variable `gl_modeKeycodes` according to the own preferences and to re-flash the Arduino Esplora board. A list of non-ASCII characters can be found in the file `Esplora_ExtendedKeycodes.txt`. The value 0 activates a **mouse action** which is implemented in code for the actuator:

actuator	**mouse action on value 0**
joystick movement	mouse movement
joystick pressure	left click
TinkerKit orange	left/right click
TinkerKit white	**no mouse function!**
slider	scrolling wheel
light sensor	right click
cursor pushbuttons	**no mouse function!**

Table 3.2.: *Default Mouse Actions of All Actuators*

The value 255 (or the constant `FUNCT_DISABLED`) disables an actuator completely.

All analogue acutators supply a value between 0 and 1023, since they are taken by a 10-bit A/D converter. The joystick 10-bit values determine the speed of the mouse movement. When keyboard codes are assigned to the joystick the repetition rate of the corresponding key is also given by joystick displacement.
The slider emulates the rotation of the mouse scroll wheel. Here the displacement determines the speed of the scroll commands. To to stop the scrolling, the slider has to be moved slightly

back towards the middle position. This works even when the slider position value itself would cause scrolling. The slight push back causes the stop of scrolling.
At the white TinkerKit connectors the A/D converter is used to detect a combination of two pushbuttons on each connector. The additional TinkerKit board contains a suitable combination of voltage divider resistors. The limits are defined in the array variable `TINKERKIT_LEVEL`. When adjusting these limit you have to take care that they are in the middle between the measured values to ensure a proper detection.
The light sensor gives a 10-bit value too, but it's difficult to use it as an actuator. Even small changes in the lighting conditions caused by accidental covering of the sensor lead to erroneous actions. Therefore it isn't recommended to use the light sensor as an actuator. You should disable it.

3.2.3. Structure of the Code

This section explains the most important `#define`s, variables and functions of the source code:

serial interface (UART)

In addition to the emulation of mouse and keyboard this sketch provides a virtual serial port (UART) using a baud rate of 9600. In Windows this serial port is shown in the device manager as a COM port and in GNU/Linux there is a device file `/dev/ttyACM<x>`. The data transmitted via this serial port aren't properly structured yet!

clean up the UART output

`#define`

`#define SERIAL_DEBUG`
Enables/disables debug messages at the serial interface (UART)

`#define LOOP_DELAY`
Defines a delay in milli seconds between two cycles of the main loop. The smaller this value is the faster the Arduino Esplora Board reacts on changes at the actuators. Especially the mouse cursor speed due to joystick movements depends on this timing value.
Too short delay values might not work properly because of bouncing bushbuttons. This will result in double actions if the button is pressed only once!

Variables

variable `byte gl_mode`
Holds the currently selected assignment mode as a value from 0 to 7.

array variable `char* gl_modeNames[8]`
Contains the names of the eight assignment modes. When selecting a mode its name (denomination) will be transmitted via the serial interface (UART).

two-dimensional array variable `char gl_modeKeycodes[8][18]`
Stores byte values of the keyboard and mouse commands for all actuators:

0	mouse command
1-254	key code
255	disabled

Functions

function `void setup()`
This is a default function of any Arduino sketch. It is called once when the arduino board starts and it is used here to initialize the variable values and for calibration of analogue sensors, especially of the joystick displacement.

function `void loop()`
This is a default function of any Arduino sketch. After `setup()` it is called in an endless loop. Here the actuator values are polled.

function `void setMode(byte mode)`
Establishes the given assignment mode (0 to 7):

- sets the variable `gl_mode` to the given value
- Sets the colour of the three-colour LED
- Sends the name of the selected assignment mode via UART

function `unsigned int readTinkerkitInput(byte whichInput)`
Reads the TinkerKit input connectors using the function `readChannel(byte channel)`.

function `unsigned int readChannel(byte channel)`
This function supersedes the identically named function of the Arduino library `Esplora`. The hardware selection of the TinkerKit connector is done by a multiplexer chip on the Arduino Esplora Board.

function `byte tinkerkitWhiteSwitches(unsigned int analogue)`
Evaluates the pressed key combination on the white TinkerKit connectors. It contains the local variable `const unsigned int TINKERKIT_LEVEL[4]` where the analogue limits are defined to detect the pressed TinkerKit pushbutton combination. These limits can be evaluated

by sending the values via UART when a button combination is pressed. The measured values should be fairly in the middle between two limit values defined in this variable. The other way to define the limits is to set them to the middle value between two consecutive button combinations.
In order for this function to work properly, the voltage divider resistors must have been dimensioned suitably and correctly!

3.3. Prospect on Planned Developments

1. The user can change the assignments directly on the device without changing the source code
2. Storage of changed assignments in the EEPROM of the *ATmega32U4*
3. Definition of several keystrokes on one actuator
4. Definition of values for mouse actions (1...31?)
5. Clean up of the outputs at the serial interface

A. Appendix

The appendix contains EAGLE drawings of the additional TinkerKit board:

- Wiring conception of the TinkerKit push buttons
- schematic of the TinkerKit board
- printed circuit board
  - dimensions
  - top side
  - bottom side

A.1. Assembling Variants

The printed circuit board may be assembled in such a way that it corresponds functionally to the prototype described in section 3.1. An alternative assembly of the board provides six pushbuttons distinguishable by voltage dividers at the analogue inputs of the white TinkerKit connectors. At the digital inputs of the orange TinkerKit connectors there can be connected two further external pushbuttons. They can be mounted elsewhere at the EsploraGaming-Controller (e.g. as shoulder buttons on the housing).

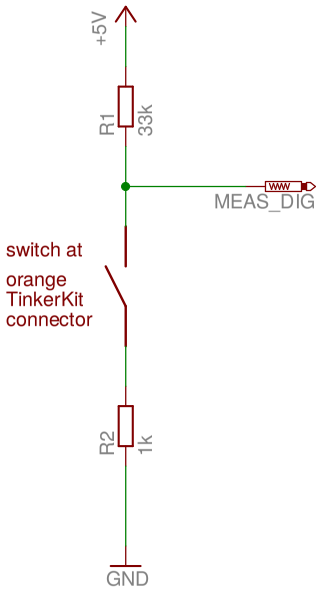
Table A.1 shows the corresponding assembly variant of each version:

version	**RDIG_ORL, RDIG_ORR**	**RANA_ORL, RANA_ORR**	**RDN_ORL, RDN_ORR**	**RDN_PINL, RDN_PINR**
four analogue buttons, two digital buttons	0R	not mounted	1k	not mounted
six analogue buttons, two external buttons	not mounted	0R	68k or 82k (t.b.d.)	1k

Table A.1.: *Assembly Variants*

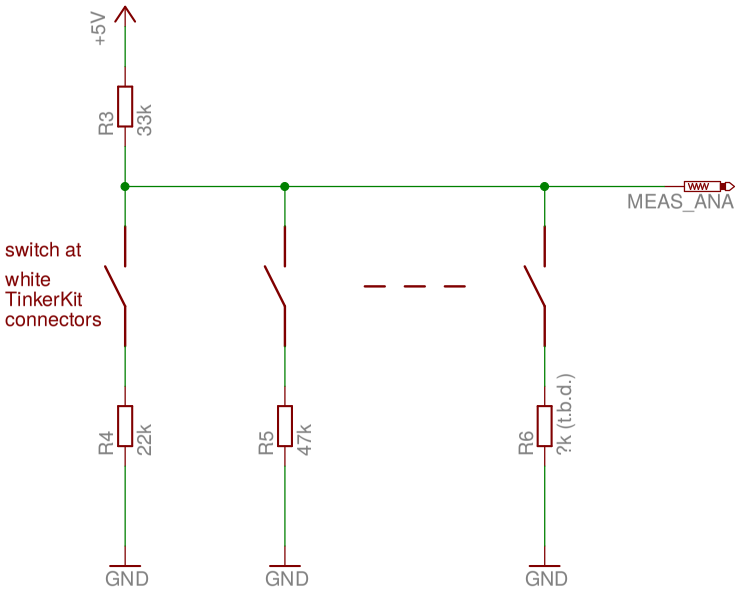
Digital Input

at orange TinkerKit connectors



Analogue Inputs

at white TinkerKit connectors



Measurement

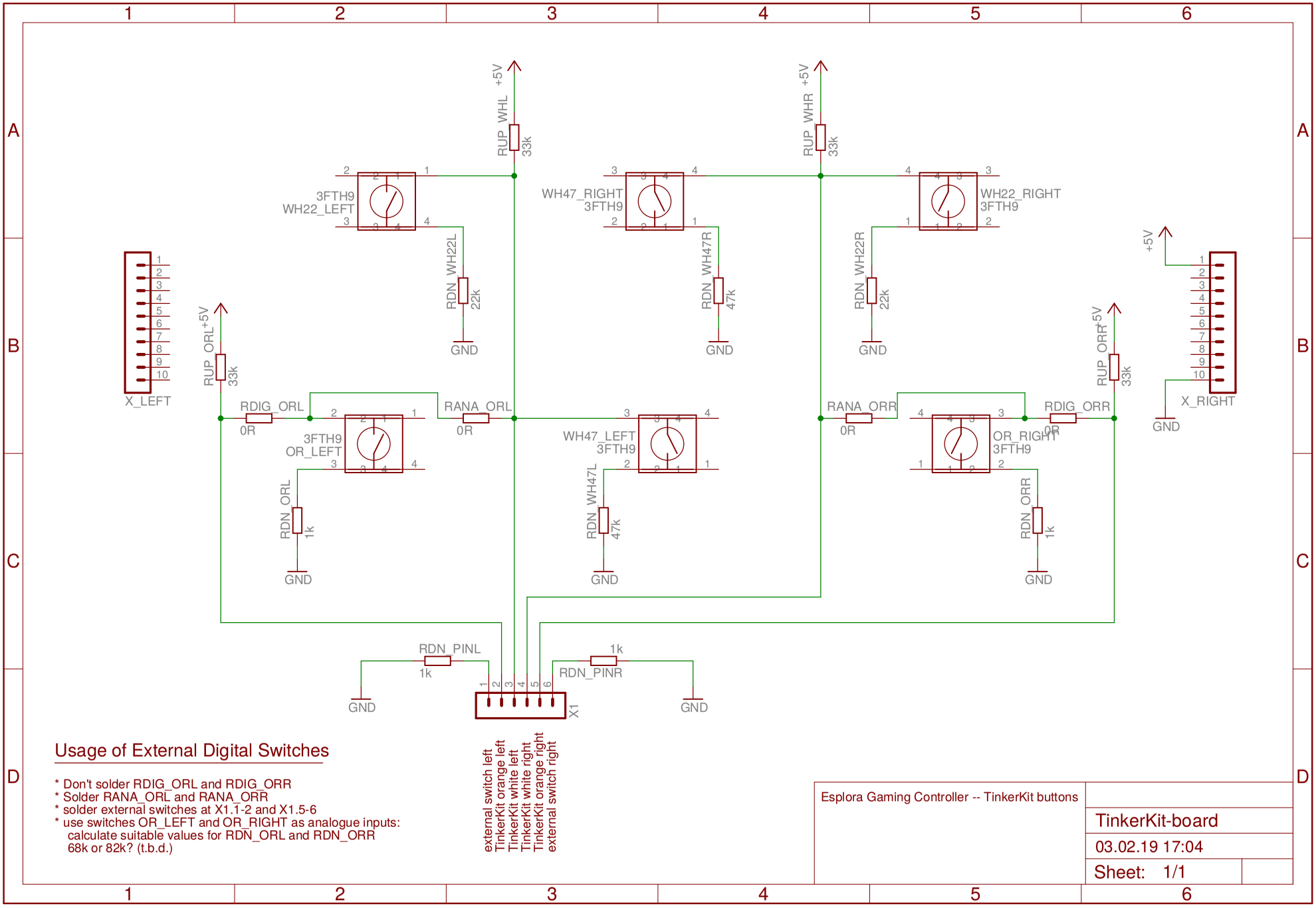
MEAS_ANA is connected to an analogue input
of the ATMEL ATmega32U4 micro controller.
There is is a 10 bit A/D converter in the background
which results in a resolution fom 0 to 1023.

A sophisticated combination of different voltage dividers
gives the possibility to recognize which switches are pressed.

Even a combination of several switches can be detected.

The other measurement point
of the ATMEL micro controller
is connected to GND internally.

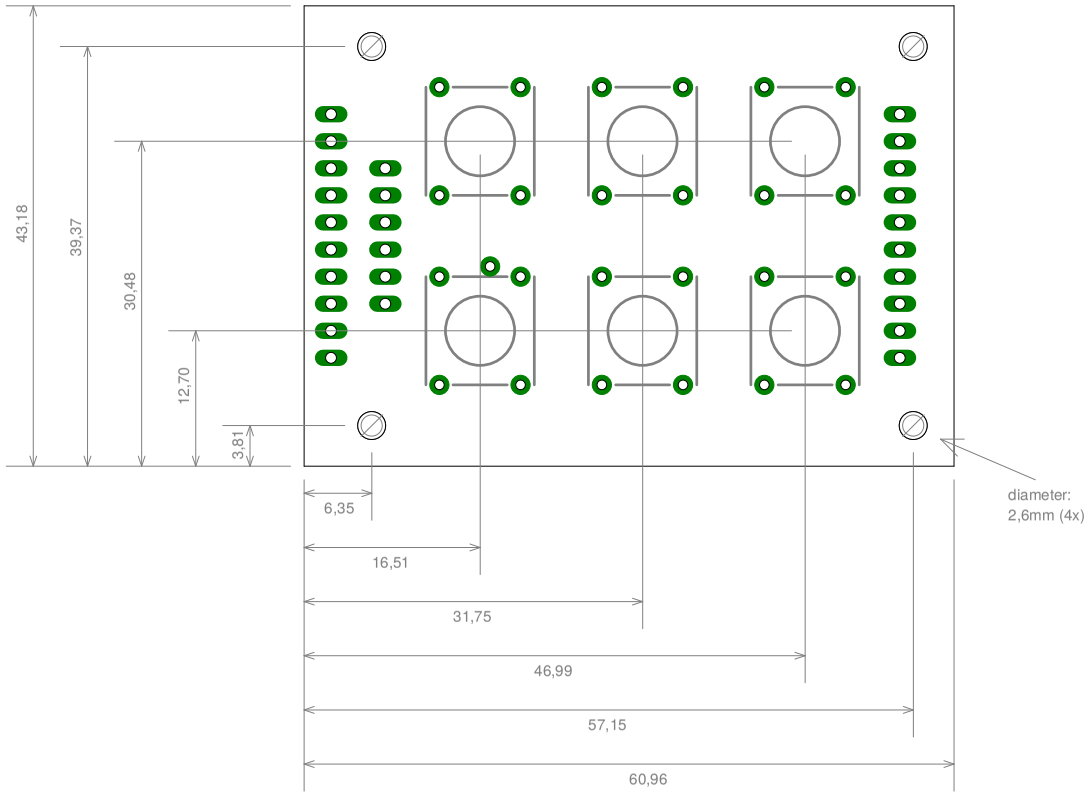
Esplora Gaming Controller	
Concept of TinkerKit switches wiring	TinkerKit-concept
	04.02.19 21:09
	Sheet: 1/1



Usage of External Digital Switches

* Don't solder RDIG_ORL and RDIG_ORR
* Solder RANA_ORL and RANA_ORR
* solder external switches at X1.1-2 and X1.5-6
* use switches OR_LEFT and OR_RIGHT as analogue inputs:
  calculate suitable values for RDN_ORL and RDN_ORR
  68k or 82k? (t.b.d.)

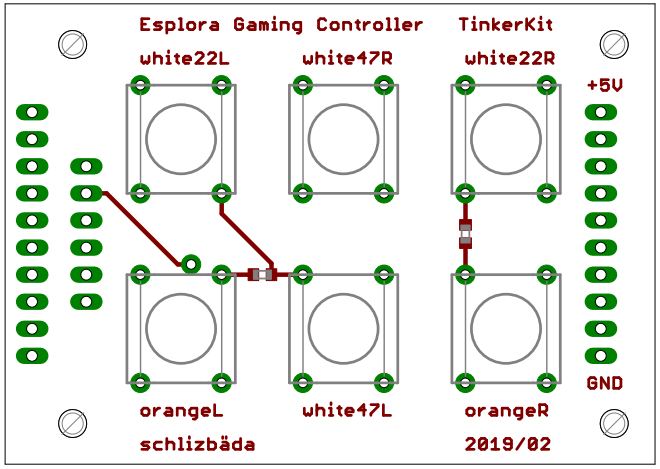
Esplora Gaming Controller -- TinkerKit buttons	
	TinkerKit-board
	03.02.19 17:04
	Sheet: 1/1



Esplora Gaming Controller -- TinkerKit buttons

TinkerKit-board

03.02.19 17:06



Esplora Gaming Controller -- TinkerKit buttons	
	TinkerKit-board
	03.02.19 17:06

Explora Gaming Controller -- TinkerKit buttons

	TinkerKit-board	03:27:20:09