

DIY Arcade Gaming Station



Hardware

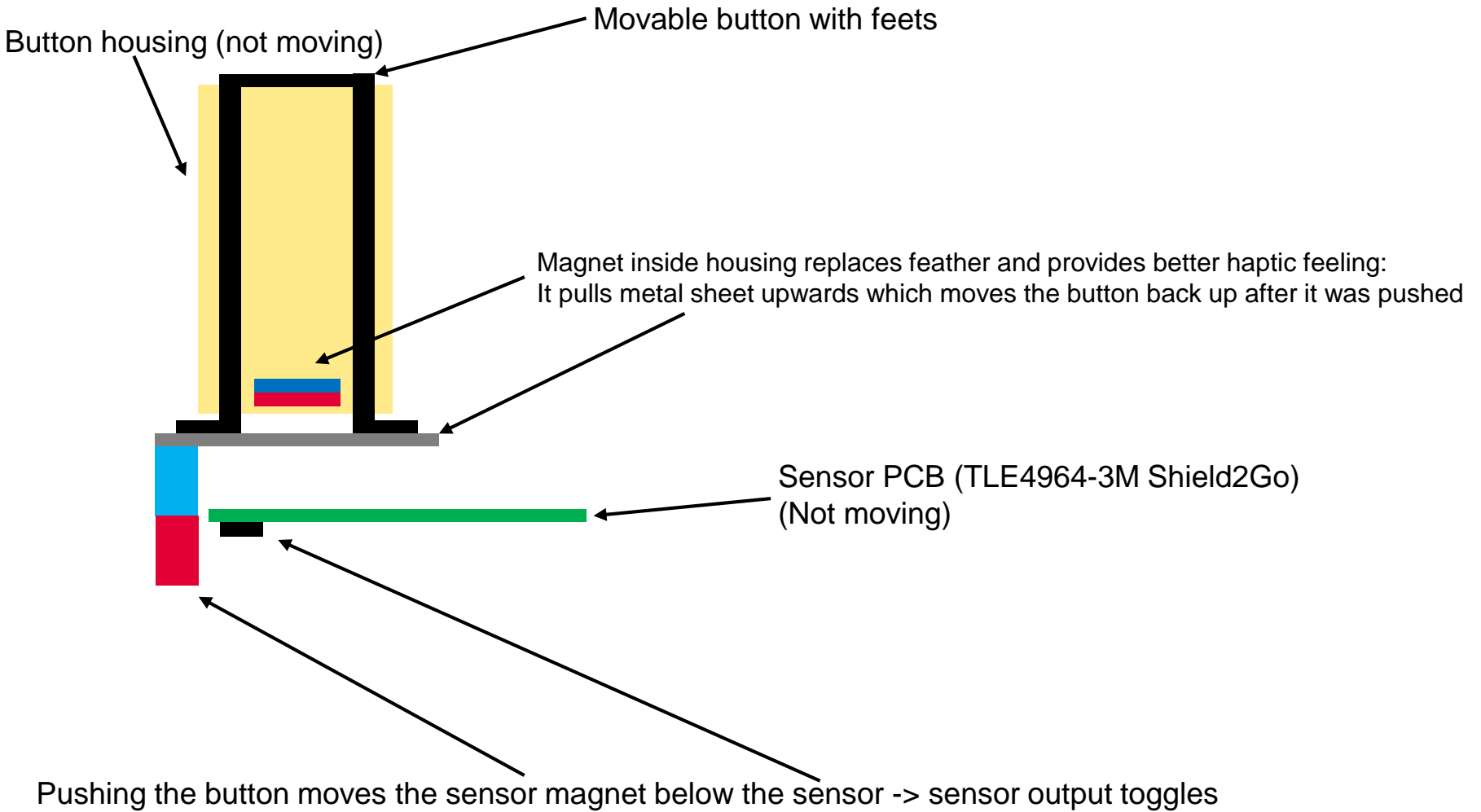


Housing

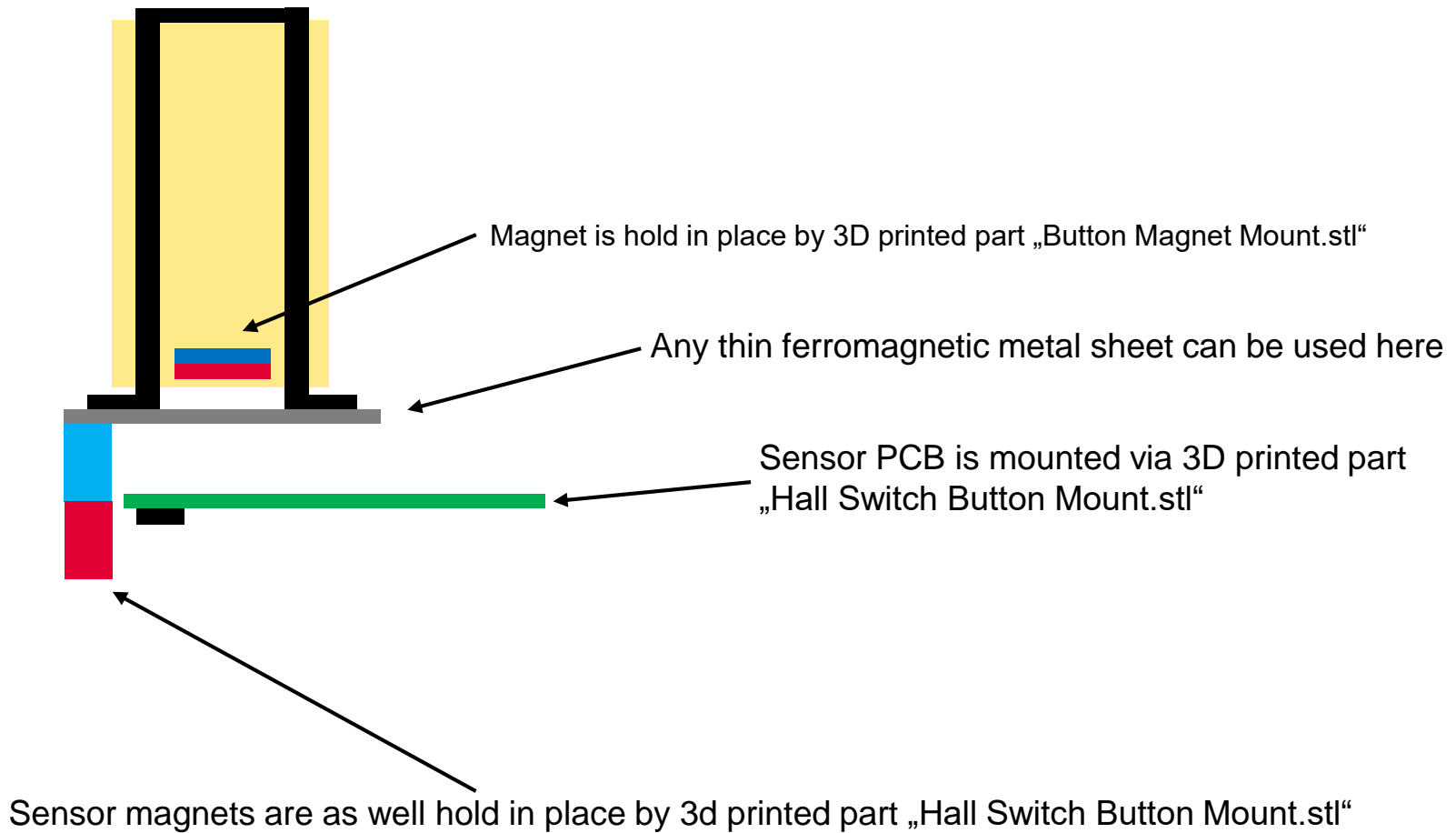


- › The housing of the arcade demonstrator is completely built with a laser cutter
- › The used material for the control panel is Plexiglas (4 mm), the rest is built with 4 mm wood
- › To rebuild the setup you need a laser cutter, glue, vice clamps
- › Don't glue everything together right away after cutting as you need to have access to the inside for installing the electronics. Start with the side walls and the middle plate only.

Mechanics: Buttons



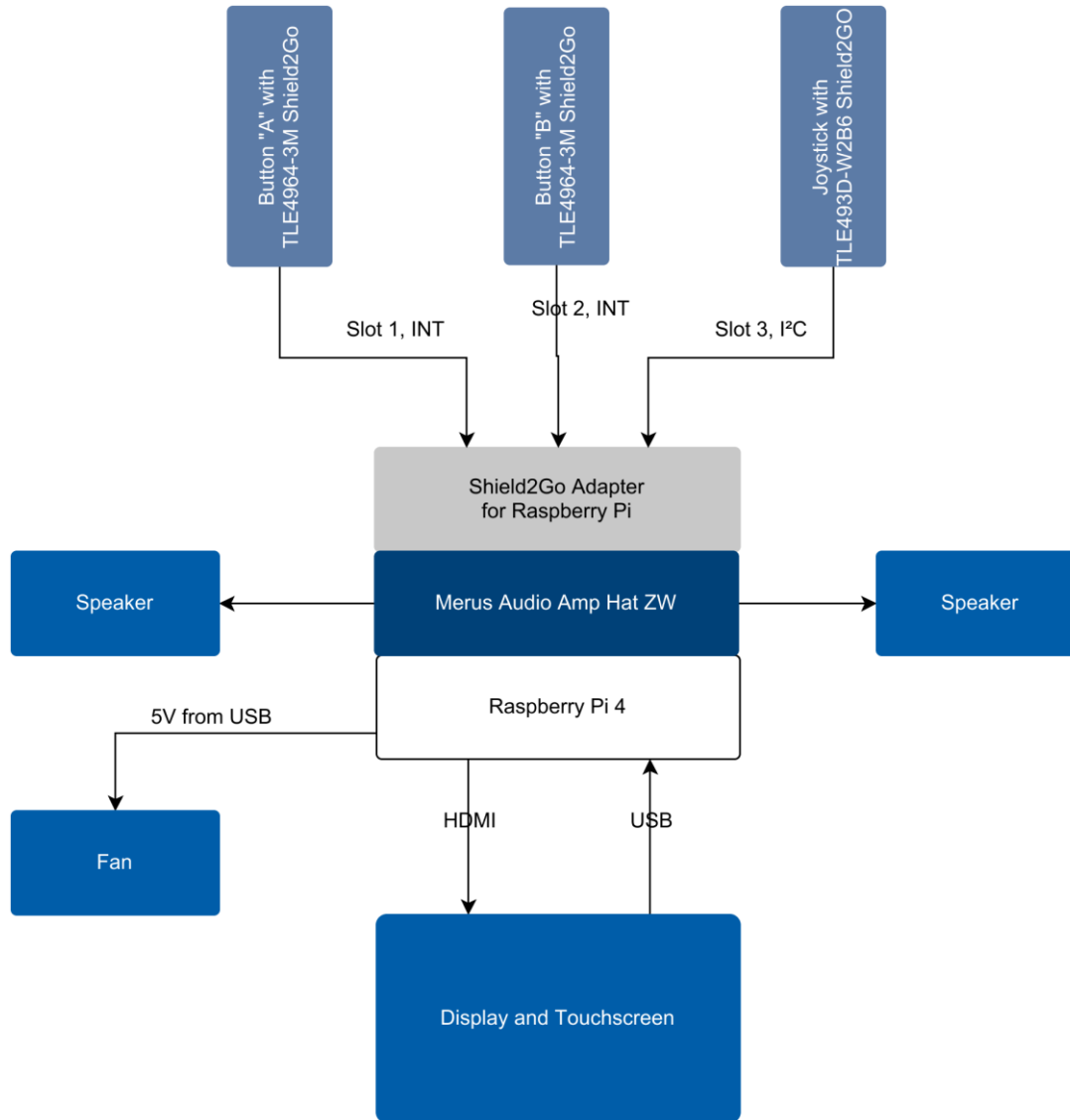
Mechanics: Buttons



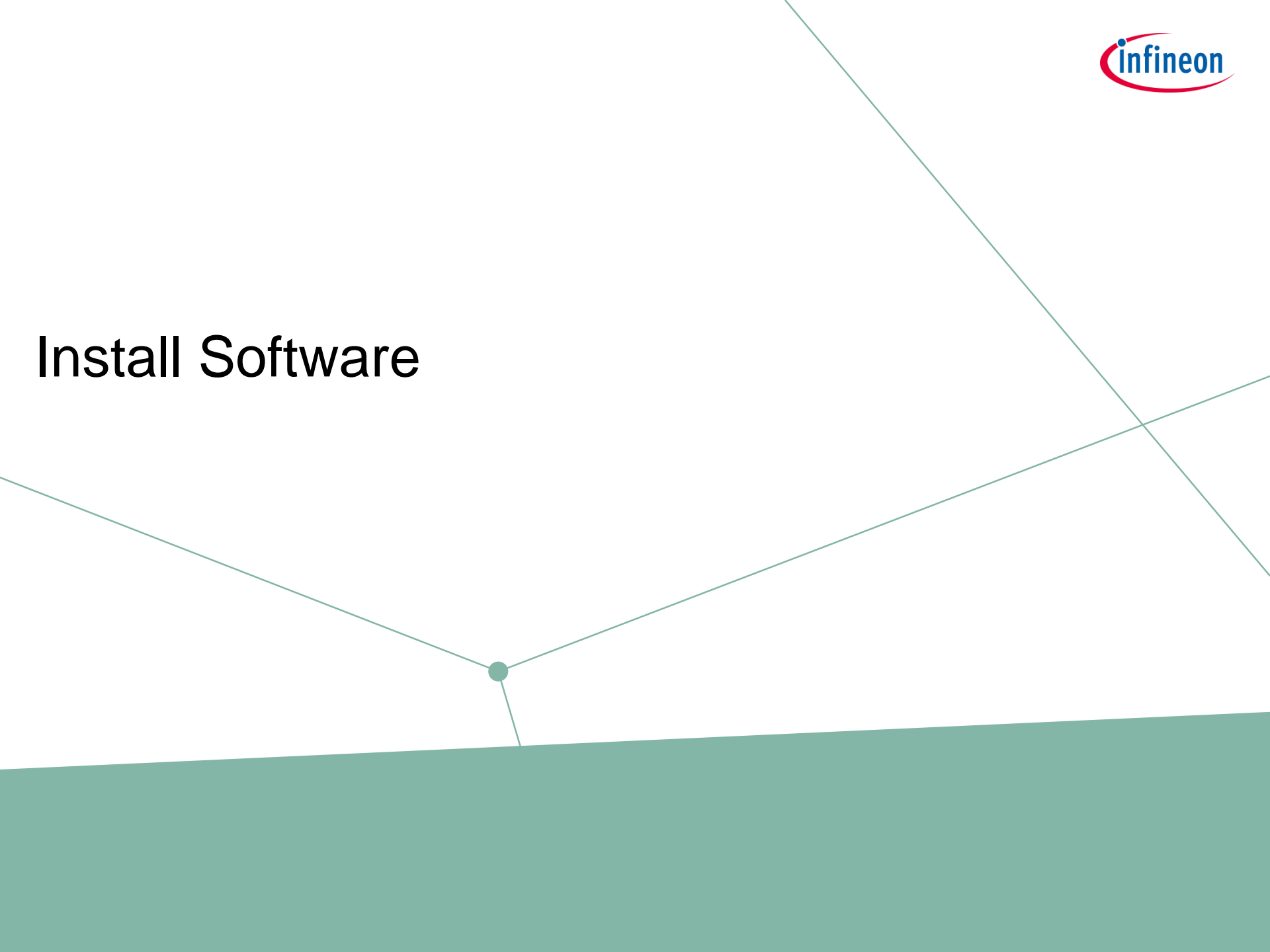
Mechanics: Joystick

- In the demonstrator a joystick is used that is quite expensive, therefore we recommend to use our 3D printed joystick and modify the control panel so the joystick can be attached to it
- You also need a magnet (cube : 5 x 5 x 5 mm)
- It is also possible to use another joystick with some modification which makes it possible to read out the position with the 3D-Magnetic-Sensor, for example this one :
<https://www.conrad.de/de/p/joy-it-arcade-joystick-professional-8-eingabegeraet-passend-fuer-einplatinen-computer-arduino-banana-pi-cubieboard-1555268.html>. This one has also the arcade optic 😊

Electrics



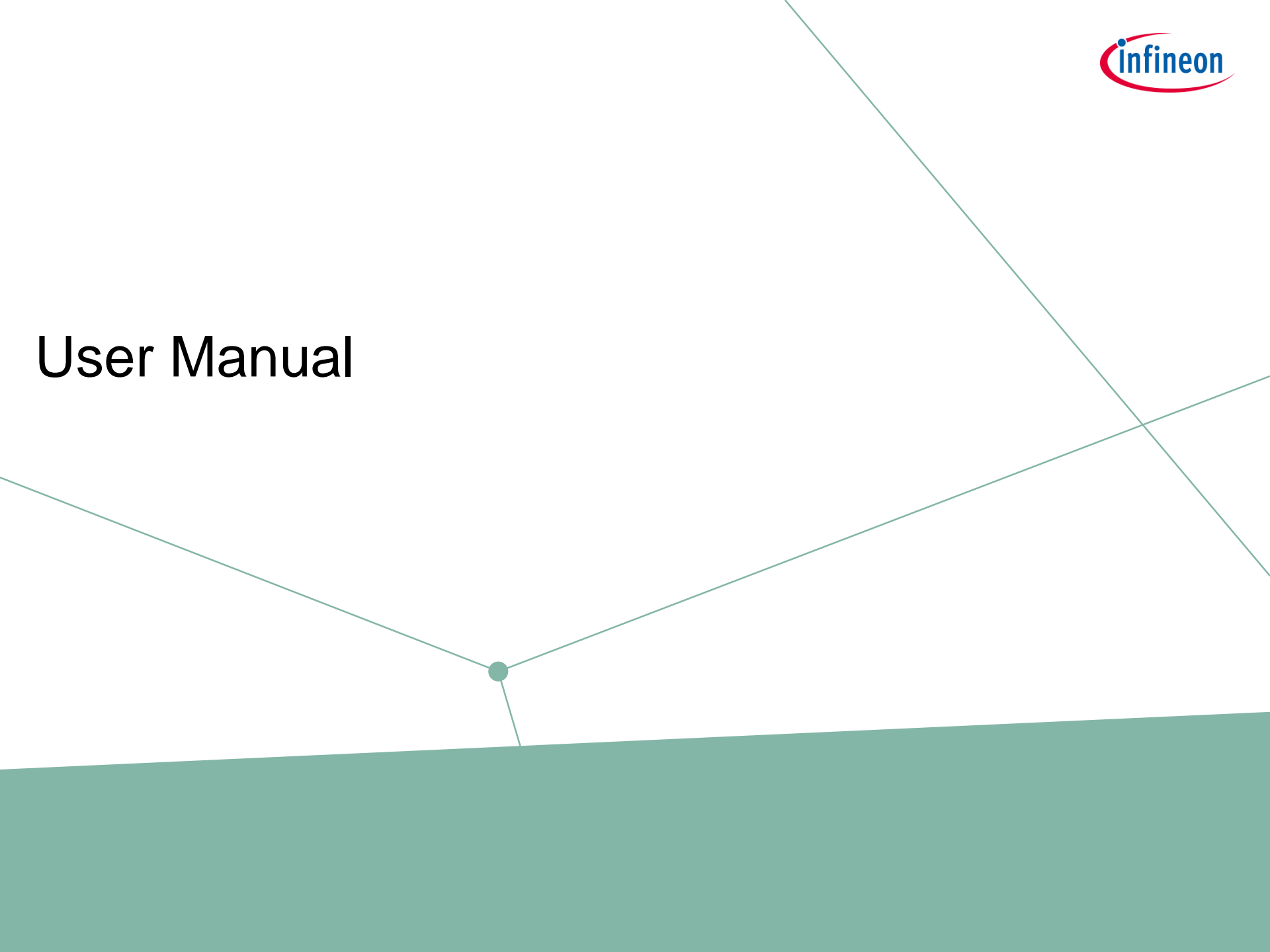
Install Software



Install Software

1. Download ISO file
2. Place on \geq 16GB micro sd card and put into Raspberry Pi
3. Power up Raspberry Pi and enter Raspbian OS
(see next slides in „User Manual“)
 - Via Hotspot + SSH
 - Via Touchscreen -> Desktop
4. Config Raspbian OS: Wifi, Password, etc.
5. Reboot and enjoy

User Manual



Arcade Overview



- › Touchscreen
- › Buttons with TLE4964 Hall Switch (in Shield2Go format)
- › Joystick with TLE493D-W2B6 Shield2Go
- › Raspberry Pi 4 + Infineon Shield2Go Adapter
- › Merus Audio Amplifier (on Raspberry Pi Hat)

Games



- › Games selectable via Touchscreen:
 - Aliens (like Space Invaders)
 - Super Minio (basically first level of Super Mario Bros)
 - Domikon (like Donkey Kong)
 - Nikman (like Pacman)
 - Mitris (like Tetris)

- › Also selectable via Touchscreen:
 - Highscore lists for each game
 - One overall highscore with special factor for each game to make them comparable. Factors can be adjusted in Software code.

Usage

- › Directly after booting, the menu can be seen and music should be played. Otherwise: Reboot (described in chapter „Hotkeys“)
- › After 5 minutes with no action the overall highscore shows up as screensaver. It can be ended by pressing A, B or the touchscreen
- › The Demo can be controlled by some button/joystick/touchscreen combinations (further referred to as „Hotkeys“)



Hotkeys

› **Controlling the sound:**

– Sound on:

‚B‘ + Joystick down + tap on any game in main menu

– Sound off:

‚B‘ + Joystick up + tap on any game in main menu

Note: you can boot up in silent mode by holding this combination during startup (until the menu shows up)

› **Shutdown or restart:**

First select any main menu item, so that you have the button „Back“ on the screen.

– For Shutdown:

‚A‘ + ‚B‘ + Joystick left + „Back“ on touchscreen

– For Restart:

‚A‘ + ‚B‘ + Joystick right + „Back“ on touchscreen

Configuration interfaces

- › There are two ways on how to gain control over the demo (basically how to access the raspberry-pi operating system):
 1. SSH (wireless via network)
 2. Hotkey to exit the Arcade Menu and show the Raspberry-Pi-Desktop

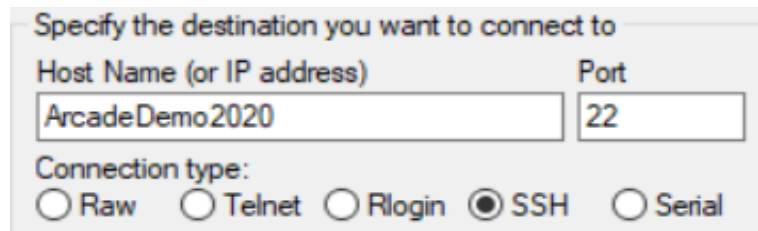
SSH (access via network)

- › Create a Hotspot with e.g. your smartphone:

SSID: „iPhone“

Password: „Arcade2020“

The raspberry pi will then connect to this network. To open a SSH-shell you need to connect your computer to this hotspot, too. Then open PUTTY and connect to „Arcade2020“.



Specify the destination you want to connect to

Host Name (or IP address)	Port
ArcadeDemo2020	22

Connection type:

Raw
 Telnet
 Rlogin
 SSH
 Serial

- › For login you need the credentials:

User: *pi*

Password: *Arcade2020*

Shortcut to Raspberry-Pi-Desktop

- › Hotkey: ‚A‘ + ‚B‘ + Joystick up + „Back“ on touchscreen
- › The menu will close and the normal Raspberry pi desktop appears. Here you have full control of the demo.
- › On the Desktop there is a script which resets the highscore („reset.sh“). Use the touchscreen to execute it and all highscores will be deleted.

How to add an own game



Download Pygame (e.g. from github)

- › Example links of current games:
- › <https://github.com/greyblue9/pacman-python>
- › <https://github.com/justinmeister/Mario-Level-1>
- › <https://github.com/erilyth/DonkeyKong-Pygame>
- › <https://github.com/rajatdiptabiswas/tetris-pygame/>
- › -> exchanged all problematic graphics and sounds with media from <https://opengameart.org/> in order to be legally allowed to show this demo on trade shows
- › Also used: Aliens which comes already with Raspbian OS

Edit downloaded game

- › copy Arcade/controls/joystick.py and ../buttons.py to game folder
- › edit game file(s): - find file where keyboard is read (search for `pygame.key.get_pressed()`)
 - at beginning: import joystick and buttons
 - Init joystick and buttons
 - after `key.get_pressed()`:
 - read joystick and buttons
 - edit key map accordingly (e.g. if `joystick.X < -0.3`: `LEFT_KEY = true`)
- › save & exit pygame script
- › See next slides for details

Copy joystick and button modules to game folder

Source: /home/pi/Arcade/Games/controls

Images > RPI_2020-02-03 > Arcade > Games > controls

Name	Date modified
buttons.py	03.02.2020 12:13
joystick.py	03.02.2020 12:13



Example Game: Aliens
in folder: /home/pi/Arcade/Games/Aliens

Images > RPI_2020-02-03 > Arcade > Games > Aliens

Name	Date modified
__pycache__	03.02.2020
data	03.02.2020
aliens.py	03.02.2020
buttons.py	03.02.2020
highscore.txt	03.02.2020
joystick.py	03.02.2020

Edit game file: import modules

```
#import basic pygame modules
import pygame
from pygame.locals import *
import joystick as js
import buttons as btn

#define Button-Slot
shoot = 3
```

- > Import previously copied joystick module
- > Import previously copied button module
- > Some defines (Here: Which slot of the Shield2Go-Adapter is the shoot-button connected to?)

Edit game files: Change keyboard input

Find `pygame.key.get_pressed()` in game script and add:

```

keystate = pygame.key.get_pressed()
#Joystick here
joyX, joyY = js.read()
speed_factor = math.fabs(joyX)*2
keystate = list(keystate)
keystate[pygame.K_RIGHT] = joyX > 0
keystate[pygame.K_LEFT] = joyX < 0
#Buttons here
keystate[pygame.K_SPACE] = btn.read(shoot)
keystate = tuple(keystate)

```

Keyboard is read into array (tuple)

Reading joystick

Converting tuple to list

Interpreting joystick as keyboard inputs and overwriting keyboard-array

Reading buttons and overwriting keyboard-array

Converting list back to tuple

Test game

- › Test game: exit menu via hotkeys and start game manually from Desktop (Touchscreen)
- › If not already done: make game fullscreen

```
fullscreen = True
# Set the display mode
winstyle = 0 | FULLSCREEN
bestdepth = pygame.display.mode_ok(SCREENRECT.size, winstyle, 32)
screen = pygame.display.set_mode(SCREENRECT.size, winstyle, bestdepth)
```

Include Game into touchscreen menu

- > /home/pi/Arcade/Menu/pimenu.ya ml
- > /home/pi/Arcade/Menu/pimenu.sh

```

---
-
name: "Aliens"
label: "Aliens"
color: "#0000ff"
icon: "alien"
items:
-
  name : "StartAliens"
  label: "Start"
  color: "#0000ff"
  icon : "alien"
  
```

```

key=$(echo "$*" | awk 'NF>1{print $NF}')
case $key in
StartAliens)
  cd ~/Arcade/highscores/
  python3 playerName.py } Get player name
                        } (for highscore)
Start game {
  cd ~/Arcade/Games/Aliens/
  python3 aliens.py
  cd ~/Arcade/highscores/
  python3 list.py aliens } List highscore
;;
  
```

needs to match

Include Game in Highscores

- › Edit game to manage highscores (open file, read current highscore, compare new score, save new highscore, etc.)
- › Edit Arcade/Highscore/list.py to show new highscore
- › Find and implement scaling factor for comparison with other games (Overall Highscore)

- › See next slides for details

Highscore: define paths

```
highscore_dir = os.path.join(os.path.expanduser('~'), 'Arcade/highscores')  
highscore_file = os.path.join(highscore_dir, 'aliens.txt')  
player_name_file = os.path.join(highscore_dir, 'curUser.txt')
```

—————→ Edit according to your game

Highscore functions: get_highscore()

```
#Highscore

def get_highscore():
    f = open(highscore_file, 'r')
    lines = f.readlines()
    f.close()

    high_score = 0

    for line in lines:
        if len(line) > 1:
            score, name = line.strip().split("\t")
            score = int(score)
            if score > high_score:
                high_score = score

    return high_score
```

You can copy this from existing games (like Aliens)

Highscore funtkons: update_highscore

Replace this with the variable which holds the score of your game

```
def update_highscore():
    f = open(highscore_file, 'r')
    lines = f.readlines()
    f.close()
    pName = get_player_name()

    lines.append(str(SCORE)+'\t'+pName)
    top10 = []
    for line in lines:
        score_txt, name = line.strip().split("\t")
        score = int(score_txt)
        top10.append((score, name))
    top10.sort(reverse=True)
    top10 = top10[:10]
    f = open(highscore_file, 'w')
    pIndex = -1
    for i in range(len(top10)):
        score, name = top10[i]
        line = str(score)+'\t'+name+'\n'
        f.write(line)
        if (score == SCORE) and (name == pName):
            pIndex = i
    f.close()
    if pIndex >= 0:
        line = pName + '\t' + str(pIndex)
        f = open(player_name_file, 'w')
        f.write(line)
        f.close()
```

You can copy this from existing games (like Aliens)

Highscore functions: get_player_name()

```
def get_player_name():  
    f = open(player_name_file, 'r')  
    name = f.readlines()[0].split('\t')[0]  
    f.close()  
    return name
```

You can copy this from existing games (like Aliens)

Include highscore-functions in game logic

- › Find the place in the game where it's best to run the highscore functions
- › Here: At the end of the gaming script (so it just runs once at the end of a game)

```
update_highscore()  
print('Highscore is: '+str(get_highscore()))  
if pygame.mixer:  
    pygame.mixer.music.fadeout(1000)  
pygame.time.wait(1000)  
pygame.quit()
```


Highscore: include into list.py

- › Edit /home/pi/Arcade/highscores/list.py
- › Define new list-function:

```
def listAliens():
    file = os.path.join(os.path.expanduser('~'), 'Arcade/highscores', 'aliens.txt')
    show_file(file, 'ALIENS')
```

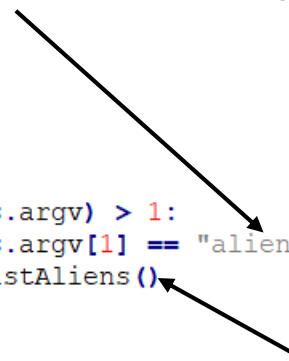
Replace these with the name of your game



- › Include in main():

```
def main():
    if len(sys.argv) > 1:
        if sys.argv[1] == "aliens":
            listAliens()
```

Name of your new list-function



Edit list.py for overall-highscore

> Edit listOverall():

```
def listOverall():
    file = os.path.join(os.path.expanduser('~'), 'Arcade/highscores', 'aliens.txt')
    aliens = getListFromFile(file)
    [...]

    top10 = []
    for scr, name in aliens:
        score = int(scr)*ALIENS_MULT
        top10.append((score, name, "Aliens"))
    [...]

    top10.sort(reverse=True)
    top10 = top10[:10]

    showOverall(top10)
```

Copy/paste, then
edit according to
your game

other games are left out here for better readability

> Add Game-Score-Multiplier for overall highscore:

```
winstyle = 0 | pygame.FULLSCREEN
bestdepth = pygame.display.mode_ok((WIDTH, HEIGHT), winstyle, 32)
screen = pygame.display.set_mode((WIDTH, HEIGHT), winstyle, bestdepth)
#screen = pygame.display.set_mode((WIDTH, HEIGHT))

ALIENS_MULT = 1
```

Multipliers are defined
before the first function

Add an own multiplier for your game
to make it comparable with the other games

Enjoy 😊





Part of your life. Part of tomorrow.