A Remote Controller Accessory for Ventilators

Claas Ehmke, Philipp Kopp, Michail Karakikes
Robin Jahn, Yaohui Huang, Jeremia Geiger

April 18, 2021

Abstract

The Corona Pandemic poses extra challenges to the clinical healthcare of COVID-19 patients in Intensive Care Units (ICU). COVID-19, and its new strains particularly, are highly contagious, such that clinical staff faces an increased risk of infection when treating COVID-19 patients in an ICU all over the world.

A remote controller for ventilators could significantly lower the risk of infection for clinicians, save essential protective gears and other material, and thus help to counteract shortages of clinical staff's capacity and material in the COVID-19 pandemic. This paper provides the technical documentation of a system that enables clinical staff to remotely control ventilators, which can be controlled by manipulation of a single push and turn knob.

Contents

Introduction

This document summarizes the results of a project that we, a team of ETH students, conducted in collaboration with a lab at ETH and a leading ventilator manufacturer, in order to support clinical staff. During the early COVID-19 pandemic in 2020, clinical staff were facing similar challenges in different areas all around the world. Among those challenges were shortage of critical equipment and trained personnel, as well as an increased risk of infection. The idea to partly balance those effects by a remote controller for ventilators originated directly from clinicians and reached us through our collaboration partners. In this document we provide the full technical description of two prototypes that enable remote control of ventilators of type Hamilton-G5/S1 and Hamilton-C6. The system, however, is not limited to those models, but can be adapted to other ventilators with similar characteristics without much effort.

The rest of this paper is organized as follows: We start with the problem statement and a list of requirements in the following Section 1. The technical solution embodied in two prototypes is then described in Section 2. In Section 3 we provide a manual for prototype assembly and system setup for the mechanical and electronic parts, as well as for the software, followed by a manual on how to launch the system in Section 4. The technical part we conclude with a list of open challenges for potential future optimization in Section 5.

The project results were generated by a team of ETH students during the COVID-19 Pandemic between April to September 2020. The project idea itself was triggered by clinicians and forwarded by EDAC lab at ETH and a leading ventilator manufacturer, who further supported us along with a larger network of students, academic advisors, engineers, institutes and industry partners as listed in the Acknowledgements Section 6.

We hope that a product can reach the market and hospitals to assist clinical staff still during the current COVID-19 pandemic.

The technical documentation is open-sourced under the GNU GPL v3 license and can be found on GitHub.[1]

Claas Ehmke Philipp Kopp Michail Karakikes

Robin Jahn Yaohui Huang Jeremia Geiger

[1]GitHub-Repository: https://github.com/claasehmke/Remote-Controller-Accessory-for-Ventilators

1 Problem Statement and Requirements

In the beginning of the pandemic clinicians outlined the importance of saving resources, in particular protective gears, disinfectants, masks, and after all the limited capacity of clinical staff. COVID-19 patients are usually treated in separate rooms in the Intensive Care (IC), such that each visit of doctors or clinical stuff costs often not only scarce resources, but also exposes them to an increased risk of infection [1]. One option to limit the exposure of medical staff to the patient could be the utilization of a fully tele-operated robot in the ICU. Yang *et al.* present such a system and describe it's benefits [2]. Nevertheless, such a system comes along with major disadvantages in terms of complexity and cost. Another option to reduce the number of visits in the ICU is to remote control the ventilator device. Leading ventilator manufacturers offer devices that can be remote controlled via a network connection off the shelf [3]. However, until such modern ventilators are in wide spread use and therefore able to replace the current non remote-controllable type ventilators in ICUs, it will likely take many more years. In order to solve that problem we propose a cheap, mechanical add on to ventilator devices which operate in ICUs today. Vagvolgy *et al.* introduce such a system while focusing on the touch screen as means of control [4]. We propose a similar setup but focus on systems that are fully controllable by manipulation of push and turn knobs. On the operator side we also make use of modern tablet computers as they are in use in related fields already today [5].

With our target remote controller we therefore intend to enable clinical staff and doctors to control and adjust ventilator settings remotely by means of remote operation of the push and turn (p&t) knob and a simultaneous video feedback. Based on the user needs and the technical framework of the ventilator to be controlled, we identified the following requirements:

- A1 remote rotary actuation of p&t knob
- A2 remote push actuation of p&t knob
- B1 manual rotary actuation of p&t knob
- B2 manual push actuation of p&t knob
- C live and reliable transmission of video feedback to remote user location
- D avoid misuse or sabotage through high Cyber Security standards

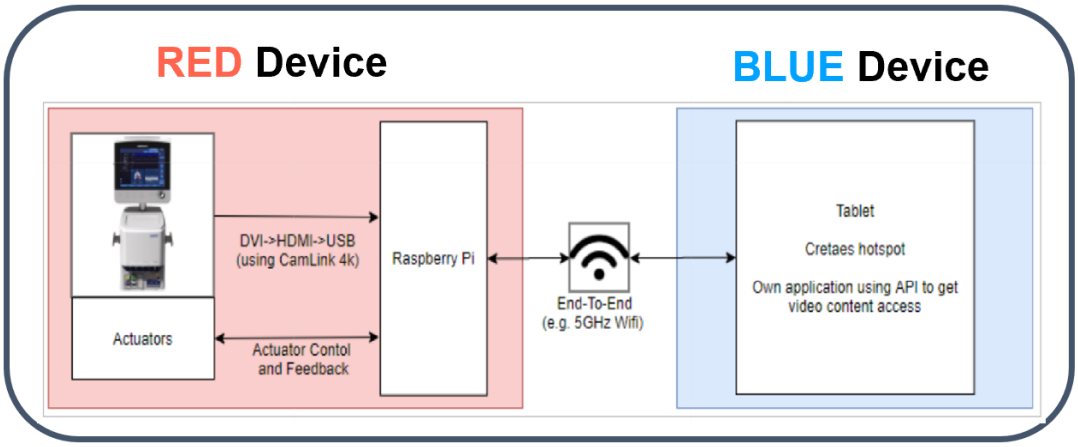


Figure 1: System overview of the remote control system.

2 Technical Solution

The technical solution[2] addresses the requirements outlined in the previous Section 1 as follows:

- A signal transmission via Tablet and Raspberry Pi
- A1 stepper motor
- A2 push-pull solenoid
- B1 hollow-shaft solution
- B2 hollow-shaft solution
- C capture signal and streaming through video capture card from ventilator DVI-outlet, Raspberry Pi and Android app
- D tablet standard authentication and point-to-point WiFi connection. (Many more security layer and procedures should be implemented)

In the following paragraphs we present descriptions of the final solution from four different perspectives: system, mechanical, electronic, and software.

2.1 System Overview

The system comprises two devices, a mechatronic controller (RED) mounted on the ventilator to mechanically control the p&t knob, and a tablet (BLUE). RED receives control signals from BLUE based on user inputs. BLUE receives and displays the ventilator video signal transmitted via RED. The system overview is shown in Figure 1.

[2]We reached the final solution in an iterative process after five design iterations and corresponding prototypes. This report comprises a technical documentation of the final prototype only.

2.2 Mechanical Design

There are two final prototypes, customized for the two ventilators Hamilton-G5/S1 and Hamilton-C6 according to their specific mechanical interfaces. The two prototypes differ only slightly in their mechanical designs:

C6. For the C6, clamping is realized through the metallic nut sticking out from the ventilator screen.

G5. For the G5/S1, an additional part for clamping is used (clamp arm) that "hugs" the ventilator screen.

Figure 2 shows the mechanical designs of both prototypes in assembly and exploded views together with their Bills of Materials (BOMs). Additionally, we provide .STL and .STP CAD-files for 3D printing and further development on GitHub.

2.3 Electronic Design

BLUE is an ordinary off-the-shelf tablet with Android OS that fulfills specific hardware requirements, namely is it capable of setting up a WiFi hotspot via a build-in WiFi-Direct function.

RED comprises electronic components as follows: Raspberry Pi 4B, a NEMA stepper motor with A4988 stepper driver and a capacitor, a push-pull solenoid with a protective diode, a 12V DC power socket, a HD video capture card. Figure 4 shows the wiring of these components in a circuit diagram.

2.4 Software Implementation

We distinguish between the two software subsystems running on the RED Device (Raspberry Pi) or the BLUE Device (Tablet). As shown in Figure 5, the Raspberry Pi connects to the hotspot created by the tablet. For both, the video streaming from Raspberry Pi to the tablet and the actuator control commands from the tablet to the Raspberry Pi, we use as transport layer (layer 4) of the Internet Protocol Suite the Transmission Control Protocol (TCP). TCP is a connection-oriented protocol which ensures reliable transportation of the data between both devices. The applications of both devices are further described in the following paragraphs:

Raspberry Pi Application The Raspberry Pi runs two applications - streaming and control:

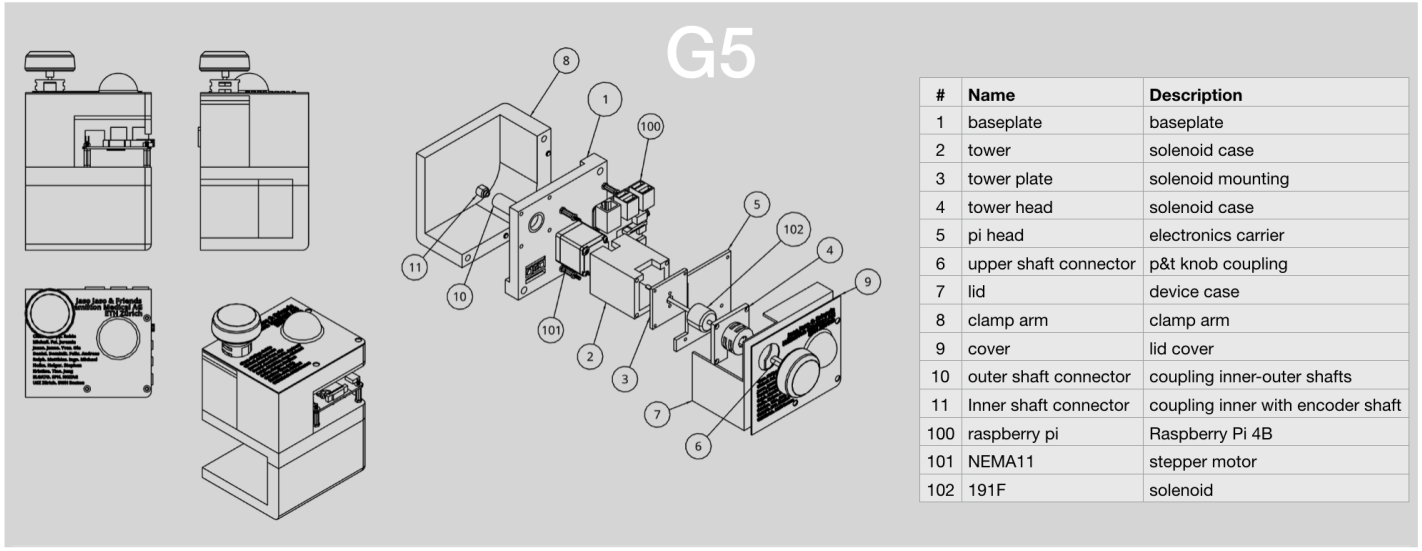
- Streaming. The first is the Motion application[3] which reads the USB Video Class (UVC) signal from the USB 3.0 port and creates a TCP socket (here with port number 8081) to which the tablet can connect to. The protocol which is used to transport the video content via the TCP connection is the Motion-JPEG (MJPEG) video signal.

- Actuator Control. The second application handles the actuator control commands. An another TCP socket is created (here with port number 8080) which accepts own declared TCP messages from the tablet. According to the received messages, the actuators are controlled. The actuator control application is written in C.

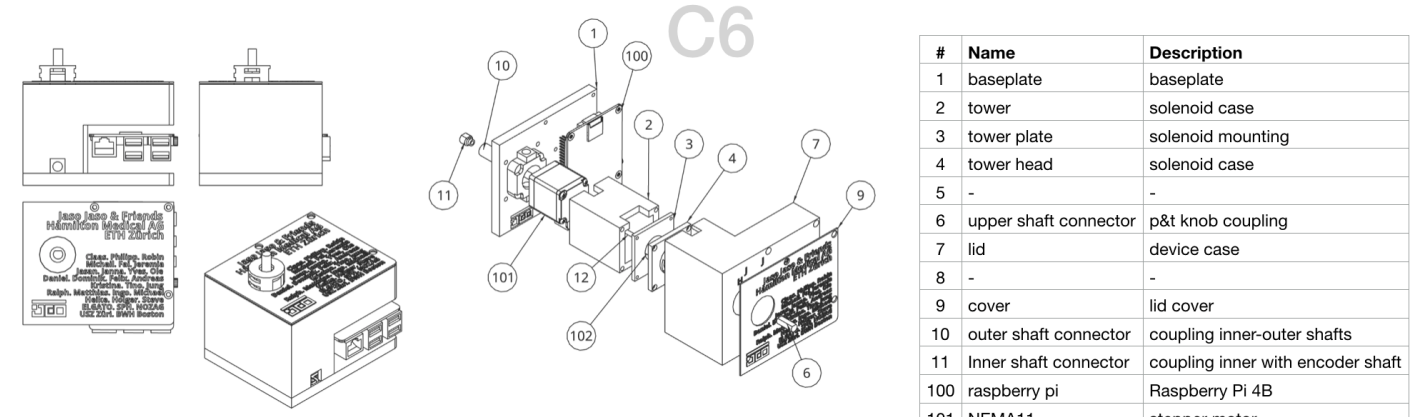
Android Application The Android app consists of two main components – streaming and control:

- Streaming. For streaming we use a TCP listener to visualize the mjpeg stream by means of the open source library ipcam-view[4].

[3]Motion-Project: https://motion-project.github.io/index.html
[4]ipcam-view: https://github.com/niqdev/ipcam-view



#	Name	Description
1	baseplate	baseplate
2	tower	solenoid case
3	tower plate	solenoid mounting
4	tower head	solenoid case
5	pi head	electronics carrier
6	upper shaft connector	p&t knob coupling
7	lid	device case
8	clamp arm	clamp arm
9	cover	lid cover
10	outer shaft connector	coupling inner-outer shafts
11	Inner shaft connector	coupling inner with encoder shaft
100	raspberry pi	Raspberry Pi 4B
101	NEMA11	stepper motor
102	191F	solenoid



#	Name	Description
1	baseplate	baseplate
2	tower	solenoid case
3	tower plate	solenoid mounting
4	tower head	solenoid case
5	-	-
6	upper shaft connector	p&t knob coupling
7	lid	device case
8	-	-
9	cover	lid cover
10	outer shaft connector	coupling inner-outer shafts
11	Inner shaft connector	coupling inner with encoder shaft
100	raspberry pi	Raspberry Pi 4B
101	NEMA11	stepper motor
102	191F	solenoid

Figure 2: The figure shows the final RED device designs in exploded view, left for Hamilton-G5, and right for Hamilton-C6. Note: not all electronic parts and no screws or threads shown.

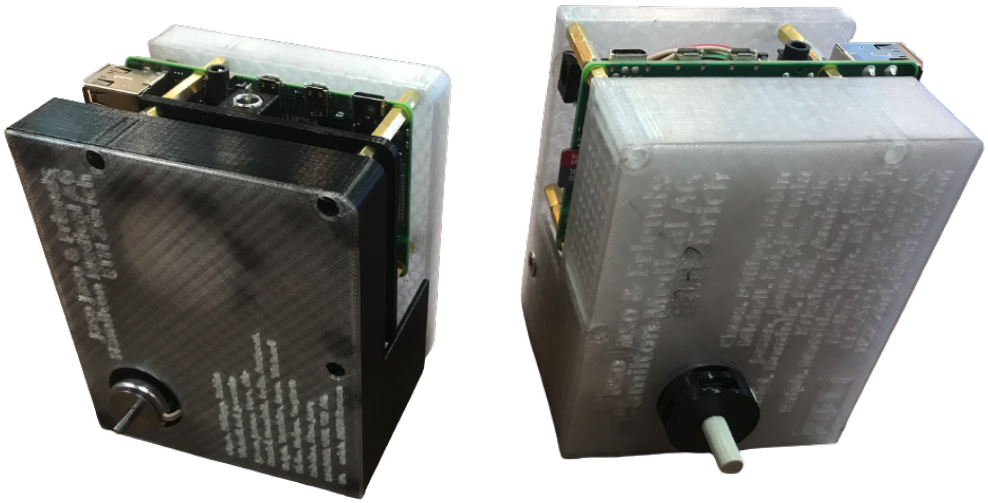
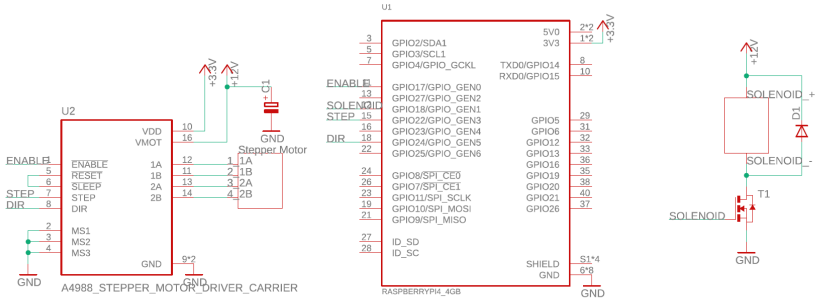


Figure 3: The figure shows two images of the final RED device prototypes to be mounted on Hamilton-G5 (left) and Hamilton-C6 (right).

- Actuator Control. The second main component is the control of a virtual push and turn button. We decided to keep the interface for this button similar to the actual physical device to make it as easy as possible for users like clinical staff, to use the remote control as they are used to interact with the physical knob. For this GUI element we use the knob library[5] which we further modified for our purposes. In terms of GUI-design the knob is overlaid over the current stream. Additionally, the knob can be freely moved to different parts of the screen to make the GUI customizable to the doctor's needs.

The entire app is written in kotlin and available in the GitHub repository.

[5]Knob library: https://github.com/BeppiMenozzi/Knob



RED DEVICE

Figure 4: Schematic of the electrical system of the RED Device.

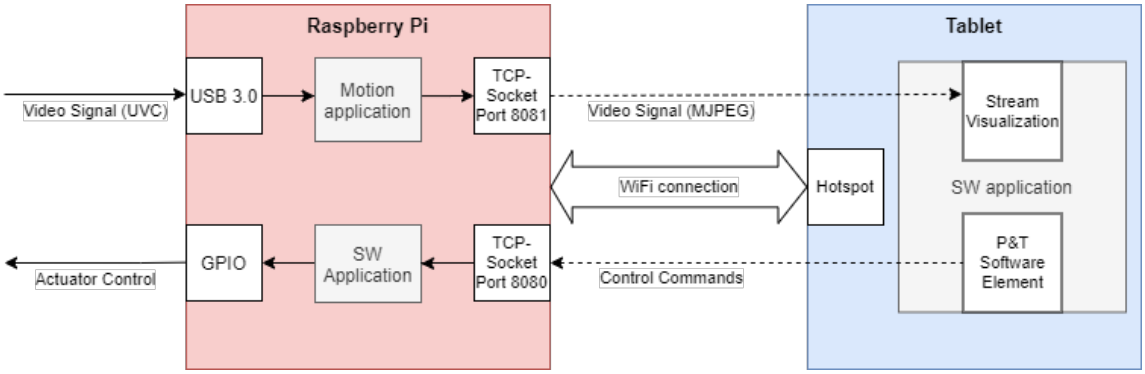


Figure 5: Overview of the software system.

3 Assembly and Setup Manual

The steps to build and setup the prototype system for the Hamilton-G5 remote controller are divided in three categories: hardware setup RED device, software setup RED device, software setup BLUE device. In this section we provide a manual on how to build and setup the system based on the specifications defined in the previous Section 2.

3.1 Hardware Assembly RED Device

Follow the instruction steps I to XIII:

I 3D-Print all RED Device .STL files.

II Add threaded inserts to 3D-printed parts:

- tower (2) 4xM3 and 4xM2
- baseplate (1) 2xM5

III Extend 3mm solenoid (102) shaft, use e.g. glue or welding.

IV Assemble tower assembly:

- Screw stepper motor (101) and tower (2) together with 4xM2.5 screws.
- Screw solenoid (102) on tower plate (3), add on top of tower (2) and screw together with tower head (4) with 4xM2 screws.

V Screw tower assembly on baseplate (1) with 4xM3 screws.

VI Screw Raspberry Pi (100) on baseplate (1) with 4xM2.5 screws and standoffs.

VII Press outer shaft connector (10) on stepper motor (101) shaft.

VIII Press inner shaft connector (11) on lower solenoid (102) shaft.

IX Press upper shaft connector (6) on upper ending of solenoid (102) shaft.

X Electronic wiring according to the circuit diagrams provided in Section 2:

- Stepper motor circuit.
- Solenoid circuit.
- 12V DC power circuit.

XI (optional) fix pi head (5) on standoffs.

XII Glue cover (9) on lid (7).

XIII Screw lid (7) on Raspberry Pi standoffs with 3xM2.5 screws.

3.2 Software Setup RED Device

The RED device, the Raspberry Pi 4, needs to run two applications: The motion lib application which handles the video stream from the ventilator and the self-developed application which handles actuator commands from the BLUE device. Follow the next steps to set-up the Raspberry Pi:

3.3 Install Raspberry Pi OS and general Set Up of the Raspberry Pi

I Flash raspberry Pi operating system

- Download and install the *Raspberry Pi Imager* from https://www.raspberrypi.org/downloads/
- Execute the *Raspberry Pi Imager* and select *Raspberry Pi OS (32-bit)* as operationg system
- Write the operating system to the SD file

II Connect to Raspberry Pi and set up device

- To set up the Raspberry Pi, connect a monitor and keyboard.
- Follow instruction of the *Welcome to Raspberry Pi* window. (Connect the Raspberry Pi to a network with internet connection. The connection is only needed for the initial software installation.)
- Enable SSH:
  1 Type in console:
  ```
  sudo raspi-config
  ```
  2 Navigate to *Interface Options*
  3 Enable *P2 SSH*
  4 Reboot Raspberry Pi
- Set static IP for Raspberry Pi
  1 Type in console:
  ```
  sudo nano /etc/dhcpcd.conf
  ```
  2 Add the following lines:
  ```
  interface wlan0
  static ip_address=192.168.49.140
  static routers=192.168.49.1
  static domain_name_servers=192.168.49.1
  ```
  3 Reboot Raspberry Pi

3.4 Install Video Stream Application

I Open the terminal (via SSH or via screen interface)

- For SSH use:
  ```
  ssh pi@raspberrypi.local
  ```

II Install motion lib using apt

```
sudo apt-get install motion
```

III Replace the motion config file with the one in the GitHub repository GitHub

- GitHub repo file path:

```
motion/motion.config
```

- Raspberry Pi file path:

```
etc/motion/motion.config
```

IV Execute the following command

```
sudo nano /etc/default/motion
```

V Change line in file to

```
start_motion_daemon=yes
```

VI Start motion service

```
sudo service motion start
```

3.5 Install Actuator Control Application

3.6 Software Setup BLUE Device

To setup the BLUE device, here a Lenovo Tab M10, we have to perform two app installations and we have to fix the devices IP address afterwards. The two applications which have to be installed are a hotspot app and the self-developed remote control application.

3.6.1 Installation of Hotspot App

The Lenovo Tab M10 does not have the ability to setup a hotpsot in the settings of the device. In general, it is possible to integrate the hotspot functionality into the Remote Control App, but we used third-party software "hotspot" from akwizgran on GitLab[6]. Please follow the next steps to install the app on the tablet:

I Enter device developer mode:

- Open Settings → System → About tablet
- Tap seven times on *Build number*
- Enter password or pin
- Device should be in developer mode now

II Connect the tablet to your computer

III Enable USB storage:

- Open Settings → Connected devices → USB

[6]Hotspot app GitLab source:
https://code.briarproject.org/akwizgran/hotspot

- Select *Media device (MTP)*

IV Download the .apk file from `https://briarproject.org/apk/testing/offline-hotspot-debug.apk`.

V Open the device in the Windows file explorer and copy the apk file to *This PC\Lenovo Tab M10 FHD Plus\Internal shared storage\Download*

VI Open the *files* app on the tablet and navigate to the *Download* folder

VII Execute the *offline-hotspot-debug.apk* file

VIII The hotspot app is now installed

3.6.2 Installation of the Remote Control App

The installation steps for the Remote Control App are similar to the above ones without setting up the device for developer mode and enabling USB storage:

I Download the Remote Control App apk file from GitHub. The file can be found under the path:

```
Android\Remote_Control_App\app\build\outputs\apk\debug\app-debug.apk
```

II Open the device in the Windows file explorer and copy the apk file to

```
This PC\Lenovo Tab M10 FHD Plus\Internal shared storage\Download
```

III Open the *files* app on the tablet and navigate to the *Download* folder

IV Execute the *app-debug.apk* file

V The Remote Control app is now installed.

4 Launch the System

Under normal conditions, no additional action is needed except powering the actuators and Raspberry Pi up, creating a hotspot with the tablet and open the app on the tablet. In case it does not work, or the system stops working, hard-reset the whole system. This means to disconnect the Raspberry Pi from power and close the app (closing means not only leaving the app).

To manually launch the system perform the following steps:

I Start the hotspot of the tablet (hotspot app).

II Connect your laptop to the hotspot.

III Power-Up the actuators via 12V DC power supply.

IV Power-Up the Raspberry Pi via USB-C power supply.

V Under normal conditions, the Raspberry Pi should connect to the hotspot automatically. Otherwise, connect it.

VI Start the *Remote Control* app on the tablet.

VII SSH into the Raspberry PI using the following commands in the terminal of the laptop:

```
ssh pi@raspberrypi.local
```

VIII Navigate to the build folder of the application of the *remote control* repository:

```
cd ~/remote_control/Raspberry/build
```

IX Execute the application:

```
./Remote_Control_Raspi
```

5 Open Challenges

Through live testing we proved the system enables remote control for ventilators Hamilton-G5 and Hamilton-C6. Both features, remote rotation and pushing of the p&t knob as well as live video feedback with minimal delay of less than one second work for both ventilator types G5 and C6.

However, in order to make the product fit for certification a risk analysis has to be carried out. Moreover, we suggest improvements in the following areas:

- Streaming of video signal delay. The Raspberry Pi 4B reaches the limits of its computational resources. A more powerful CPU might likely be sufficient to further decrease the delay to almost instant feedback. A different GPU-based software implementation can be addressed to speed-up video transmission by using compressed video data.
- Overheating of computational unit (100). The Raspberry Pi 4B heats up at rapid speed when actively streaming the screen video signal. Testing should prove that with sufficient cooling through fan and or heat sinks the temperature remains stable.
- Overheating of stepper motor (101). Under normal/occasional load the stepper motor does not generate significant amounts of heat. In case of a constant load the stepper motor might still overheat. The housing design (2) should therefore guarantee sufficient cooling capacity by design.
- Overheating of solenoid. The solenoid (102) does not generate significant amounts of heat when only occasionally activated. But in case of constant activation the actuator might rapidly overheat. The housing design (4) must therefore counterbalance excessive heat generation through e.g. heat sinks by design.
- Overheating of stepper motor driver. We observed overheating of the stepper motor driver (103) in case of wrong setting of the maximum current. This should never occur in practice, but generally this issue should be prevented under all circumstances through heat sinks.

6 Contact Data

Developers The project was lead by six students of ETH Zurich. Corresponding authors are Jeremia Geiger, Claas Ehmke and Philipp Kopp.

Name	**Position**	**E-Mail**
Jeremia Geiger	Project-, Mechanical-, and Prototyping Lead	jeremia.s.r.geiger@gmail.com
Claas Ehmke	System- and Electrical Lead	claas.ehmke@gmail.com
Philipp Kopp	Software Lead	philipp@koppmaps.de

Acknowledgements

The project is based on the collaborative efforts of a team of students and engineers, together with a network of supporting professional engineers, industry partners, and other institutions.

In particular we want to thank:

- Jasan Zughaibi, Janna Rehbein, Yves Miller, Ole Müller, Dominik Schulte, Daniel Mouritzen, and Felix Böwing for their support in prototyping, supply of materials, providing of work space, 3D-printing, laser-cutting, electronic components, technical feedback, management advice, for their ability to listen and their social commitment.
- the EDAC Lab at ETH, incl. Prof. Dr. Kristina Shea and Dr. Tino Stankovic for establishing the contacts with collaborators and supporting us in the early phase, also Jung-Chew Tse for early 3D-prints.
- the University Hospital of Zurich USZ and the Brigham and Women's Hospital in Boston for the testing space.
- Elgato and NOZAG AG for supplying us with scarce components and knowledge during the early phase of the COVID-19 pandemic.
- A leading ventilator manufacturer for collaborating and supporting the project.

References

[1] CDC COVID-19 Response Team, "Characteristics of Health Care Personnel with COVID-19 - United States, February 12-April 9, 2020," *MMWR. Morbidity and mortality weekly report*, vol. 69, pp. 477–481, Apr. 2020.

[2] G. Yang, H. Lv, Z. Zhang, L. Yang, J. Deng, S. You, J. Du, and H. Yang, "Keep healthcare workers safe: application of teleoperated robot in isolation ward for COVID-19 prevention and control," *Chinese Journal of Mechanical Engineering*, vol. 33, no. 1, pp. 1–4, 2020. Publisher: SpringerOpen.

[3] "Philips eICU program |Philips Healthcare | ICU telemedicine solution." https://www.usa.philips.com/healthcare/resources/landing/teleicu. Accessed: 2021-05-30.

[4] B. P. Vagvolgyi, M. Khrenov, J. Cope, A. Deguet, P. Kazanzides, S. Manzoor, R. H. Taylor, and A. Krieger, "Telerobotic Operation of Intensive Care Unit Ventilators," *arXiv preprint arXiv:2010.05247*, 2020.

[5] J. E. Hollander and B. G. Carr, "Virtually Perfect? Telemedicine for Covid-19," *New England Journal of Medicine*, vol. 382, pp. 1679–1681, Apr. 2020. Publisher: Massachusetts Medical Society _eprint: https://doi.org/10.1056/NEJMp2003539.