MedECC Final Report

Abstract

When the load on medical systems is increased, it is difficult to expand the accessibility to medical professionals and specialists. MedECC is an open-source telemedicine application that allows clinicians to better scale personnel resources using an Internet of Things system. Our system enables doctors to make quick changes without having to physically change settings in the room of the patient. It would also allow doctors to remotely monitor the health of the patient, as well as enable doctors and specialists to communicate with each other, increasing the efficiency of doctors' time, resources, and energy. The IoT system includes a wirelessly connected ventilator that can be remotely monitored and adjusted.

Introduction

The COVID-19 crisis has made many of the faults in our medical system clear. In the case of a surge in patients in need of urgent care, our hospitals cannot easily scale up resources or manage with the increase. The number of patients that needed to be looked after during the peak of COVID were far more than the healthcare systems could handle.

To address this specific issue, we've developed MedECC, a telehealth system to address the many inefficiencies in healthcare systems around the world.

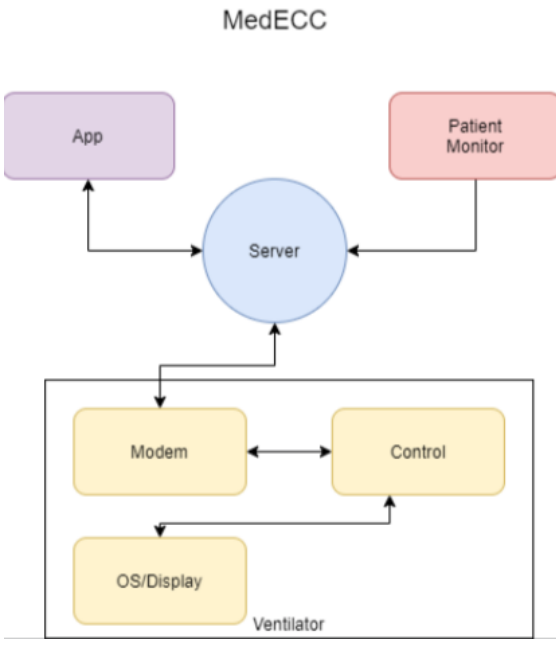


Fig. 1: MedECC

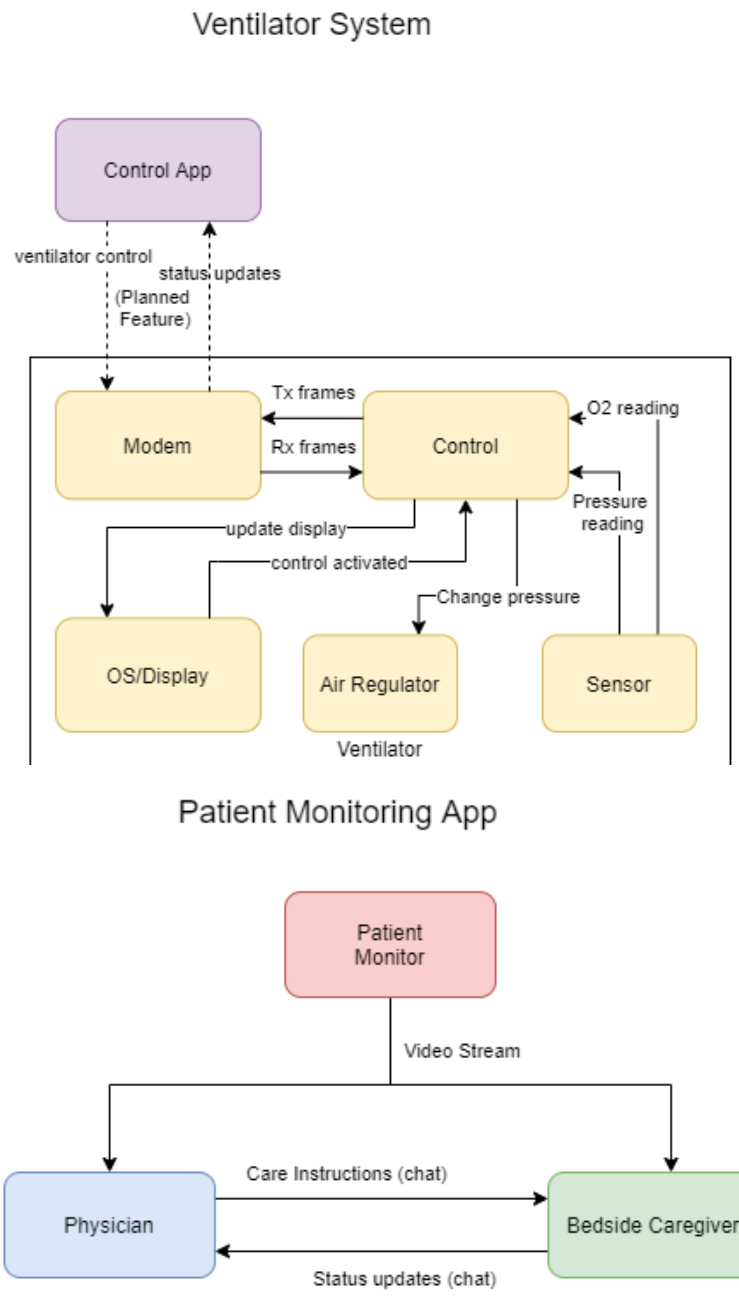
MedECC restructures hospital workflow and brings the benefits of modern technology into patient care. This is accomplished through our integration of a remote care system into patient care, which we've done through 2 things: a ventilator with remote care capabilities, and a patient monitoring application. We've

developed our ventilator by using a custom circuit board made by Dr. Michael Barrow. This connects to an air supply, which passes through a pressure regulator into the patient's lung (tested only on artificial lungs). This system is controlled by an algorithm which maintains pressure inside the patient's lung during each breathing phase (inhalation and exhalation). To adjust the controls and settings of the ventilator to provide proper breathing, we have a touch screen component which allows the physician to adjust the settings of the ventilator. These instructions are used to adjust the parameters in our algorithm to give the proper care to the patient. An alternative method of adjustment is through the mobile user interface. This interface is connected to a modem, another component of the ventilator, which then receives and parses instructions. The patient monitoring application is another component of the project. This implements multimedia communication, patient video monitoring, and hospital administration so that information can be relayed much quicker, allowing for quicker decision making and expedited care.

Certain projects have implemented a remote ventilator like the above but with different use cases. One particular remote ventilator implementation stated in [SHDH10] was used to help children on chronic ventilator support. The remote ventilator was aimed at minimizing the damage done to the social and psychological development of such children. Our ventilator is aimed at improving the efficiency of doctors in hospitals and thereby increasing the number of people who receive treatment. However, there has been extensive research in maintaining a balanced lifestyle for kids with some chronic conditions, using telemedicine and supplemental devices like remote ventilators.

System Overview

The overall system consists of integrated sub-components which combine to form the IOT system with a connected ventilator. These parts include the patient monitoring application, an app which features communication, administration, and remote monitoring; the user interface for the ventilator which can be displayed on a mobile device; a modem to connect external devices to use the ventilator; an operating system along with a touch screen display to adjust the ventilator directly; and an control algorithm for the ventilator's circuit to realize ventilation with pressure/flow control and collect data. These components combine to create a streamlined workflow in a hospital, with in-patient care application and a modular ventilator capable of remote control.



The specific details of the components are as follows:

User Interface:

The user interface aspect of this project consists of 2 different softwares: the ventilator control site, and the in-patient telehealth application. First, we'll go into detail about the ventilator site.

The ventilator site is a web based angular application which is currently hosted on github pages. This site has 2 versions, depending on the desired means of connection. Both versions feature 3 charts displaying pressure, flow, and volume. These charts are from an npm library chartJS. They also contain controls for

ventilator adjustment and for the setting of alarms so that the user may be notified if the patient statistics are not within a safe range set by the user, as well as the ability to change between ventilation modes. The notifications are both on the page, with details of the violated conditions, and separate push notifications sent using a firebase project.

The distinctions between the 2 versions are related to the data input system. In one version, data transmission is achieved with a signalR backend written in C# using a .NET core server. This can be hosted locally along with a locally hosted ventilator app, enabling wireless LAN transmission, adjustment, and ventilator monitoring. This connection is achieved with a signalR hub to transmit data continuously between the client and the server, which could be part of the ventilator.

The alternative application, which is the one hosted on github pages, is designed for wired communication encoded in sound sent over an audio jack. This is specifically designed to connect to our modem. This uses webjack, a library intended for using sound/audio cables as a method of communication for an arduino and a compatible device with a 3 ring 3.5mm audio cable. This enables the encoding and transmission of characters as audio data, which we are able to interpret by sending characters such that a valid JSON string is formed and is able to be parsed into data we can display in the interface, and data we can interpret and use to adjust conditions in the ventilator.

The second component of this part of the project is our telemedicine application. This is also built off of an angular framework for the front-end, a mySQl backend, and it similarly employs a signalR backend. This features 3 use cases: the administrator side, the physician/caregiver side, and the patient device side. The administrator manages the accounts and database through the creation and deletion of users and patients. The page features a table displaying relevant user data, as well as the ability to search and manage them. The physician section features a list of all patients under the user's care, as well as the ability to add users. Each item in the list can be selected to navigate to the patient's specific page. This page features a video live stream which is set up through the patient device login. This stream is made using dailyco video streaming API to be able to monitor the patient. This page also has the multimedia chat application below the video stream. This is made using signalR hubs to transmit messages, and all media and messages are stored in the database securely. The chat app also allows notifying users or the whole team and this is accomplished using swPush. The final page in the physician side is the team page where they can view their team, add team members, leave their team, or find individuals to mention in the chat. Lastly, there is a patient device login for which the physician logs in and sets up a smart device as a video source. The patient monitoring app is currently deployed on an AWS server and is being set up for a test deployment in Indian hospitals.

OS/Display:

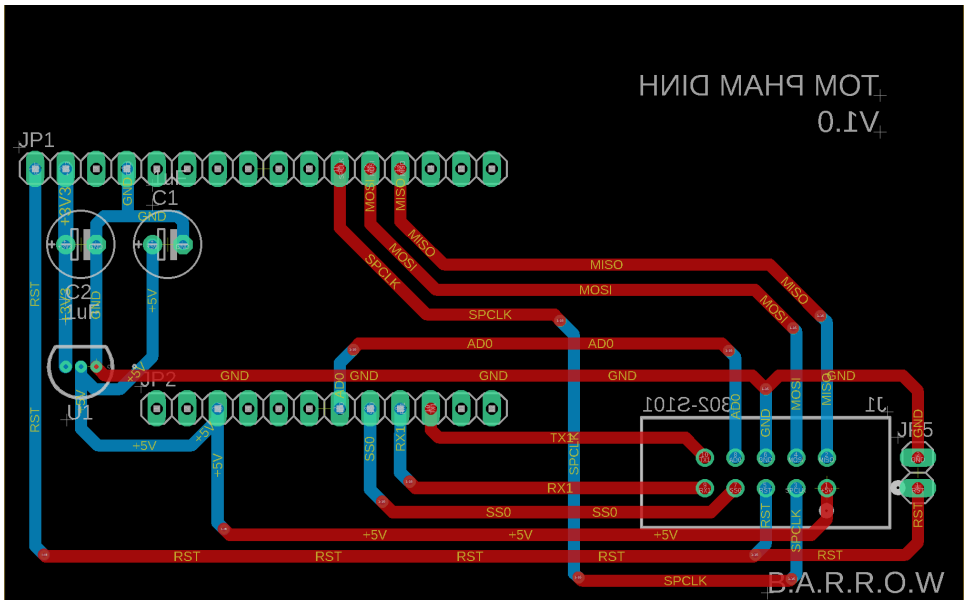
The OS/Display part of the project consists of two parts. The first part is the Display on the ventilator itself which allows the user to change ventilator settings on-site. The OS is the system by which we can make changes to those settings, and be able to communicate that information with the Control. We also handle the setup of the ventilator from power on or reset.

For the Display we are using an Adafruit TFT Featherwing connected to an Arduino Due the current screens we have for the Display is: The Main Ventilator Display, Mode select Screen, and Alarm Screen. These screens are all run using ARTE framework for multitasking so we are able to display data taken from the control team, while handling user input. Everything on our screens are broken down into separate individual components which can be found in our Libraries folder in the project Github. To break

down our screens, our first screen is the display screen. This screen displays the data given to us from the control system. The means of this information transfer is through our central data structure. Our data structure consists of the system variables each with their own associated keys and we use the Built in mutex of ARTE to avoid errors when trying to access or read that data. We can do this with setters and getters. We also have a testing suite to test our I/O configurations so when we startup checks if they are correct. On this display screen we continually update these using both numerical values and waveforms. This screen also has the capability to change ventilator settings to tell the control loop to adjust as necessary. We have the Mode select screen which is self explanatory, allowing us to go from screen to screen.

The last screen is the Alarm screen that will allow us to change when the alarm goes off. Implementation for this is still to be done.

In order to connect the Arduino to the display screen we use the custom pcb shown below:



PCB for display adapter

The PCB makes all the necessary connections from the Due to the display. However because the display needs both a 5V and 3v3 power supply. We have added a voltage regulator and its capacitors to achieve those conditions. This PCB also comes with a keyed 10 pin connector and a reset pin.

As for the OS we need to integrate with the other components of the project.

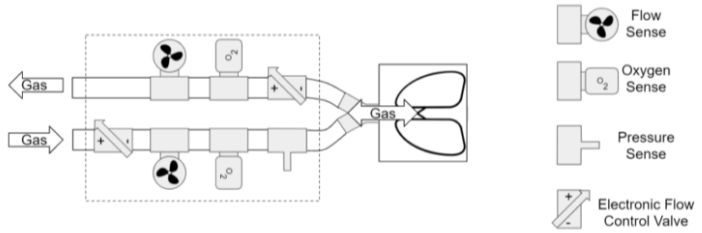
- Integration with the control team to display/change data of ventilator
- Integration with modem to communicate data they are sending to ventilator
- Integration with the power team to handle reset/power functionality.

Control System:

The control system holds for the fundamental ventilator's functionality of ventilation. The system consists of an inner circuit and an outer circuit. The inner circuit controls the inhalation and exhalation process directly to the patient-ventilator system. The outer circuit controls the mix rate of oxygen and air to control the concentration of gas that pumps into the inner circuit system. Data is collected by sensors, which can

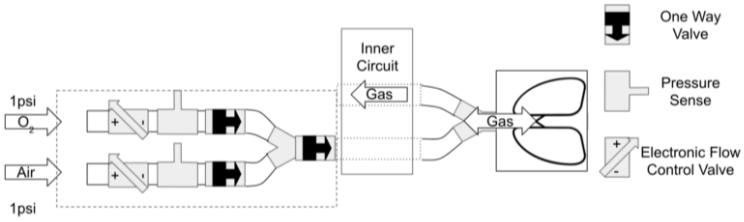
be used as feedback to the control algorithm and also used by other groups. The algorithm manipulates electronic flow valves to control the system.

1.1 Inner Circuit



The two flow control valves will both be one-way. The lower tube is the inhalation tube and the upper tube is the exhalation tube.

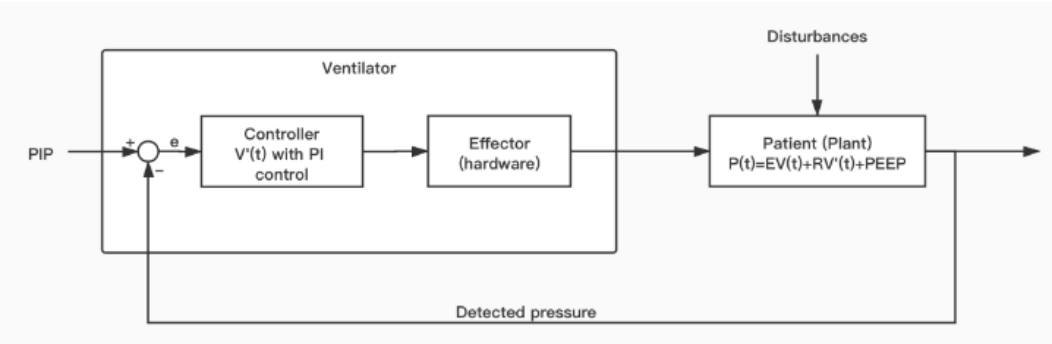
1.2 Outer Circuit



Pneumatic Diagram of two control circuits

Generally speaking, the goal of the control group is to implement common ventilator's modes. The mode we are currently working on is the Pressure Control Ventilation, or PCV, during which the pressure inside the patient-ventilator system stays at fixed levels assigned by humans. More specifically, this includes the PIP(peak inspiratory pressure) during the inhalation process, and PEEP(Positive end-expiratory pressure) during the exhalation process. The duration time of the inhalation and exhalation process are also designated by humans, and the system will switch between inhale and exhale. Currently the system only supports full control of breaths, which means it doesn't support patient-triggered breath. Function of patient-triggered breath will be added after we step into the stage of animal's tests.

As for the inner circuit, based on the Equation of Motion of the human pneumatic system, it is possible to maintain the pressure level by controlling flow. The algorithm worked as a closed-loop PID control. The control system worked as follow:



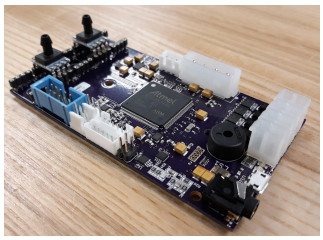
The outer circuit manipulates valves to control the mix rate of O2 and air with given FiO2 (Fraction of inspired oxygen). This is not implemented yet.

Modem:

The modem being developed for this project is a softmodem, i.e, a modem which performs transmission and reception mostly via software. The modem is being developed on a custom remote ventilator board (PCB) designed by Dr. Michael Barrow. The softmodem facilitates communication between the ventilator and the application designed for the ventilator (User Interface discussed above). The main capabilities of the modem are as follows:

- Transmission (TX): The transmission part of the modem involves sending characters and strings from the remote ventilator board to the user interface developed by the UI group. The strings that the modem must be able to transmit must look like the one stated here: “{"s":[15,0.5,30,5,0.5,50,5],"a":[100,0,13.5,0,100,0,60,0],"m":"simv+ps"}” .
- Reception(RX): The reception part of the modem involves receiving characters and strings from the sister MedECC application which will then be used to update the values of various variables used to control different settings of the ventilator. An example string that the ventilator must be capable of receiving is “{"a":[100,0,13.5,0,100,0,60,0]}”.

The software part of softmodem for the advanced ventilator has been mainly implemented using C++ (Arduino variant). It also uses various functions defined in the tc_lib.h header file. These functions include performing tasks like capturing certain frequencies and also sending bits. Albeit, these functions had to be changed in order to better fit our implementation.



For the hardware part, like mentioned above, the main device used right now is a custom PCB. The main MCU in our board is ATSAM3X8E. Three probes have been attached to the PCB for the RX pin, TX pin, and GROUND pin.
Further, a Saleae logic analyzer was used to test the wave signal being transmitted and received by the PCB. During the initial stages of development, an oscilloscope was used to test the permissible frequencies for the PCB. All the modem tests were conducted on the same Windows PC using the same TTRS cable in order to ensure consistent results.

Softmodem functioning: The softmodem communicates with the User Interface mentioned in the report earlier by sending out wave signals consisting of certain high and low frequencies which determine the message being sent or received. A logic 1 being sent out or received by the softmodem uses the frequency 7.5kHz and a logic 0 being sent out or received by the softmodem uses the frequency 4.95kHz. A logic 1 is made up of 6 consecutive highs and lows of the above mentioned frequency while a logic 0 is made up of 4 consecutive highs and lows of 4.95kHz. Therefore, in order to send out a 8 bit character (ignoring the various other parts of the protocol), lcm(8,6,4) = 24 bits are required. A character being sent out by the remote ventilator must be packaged in a certain way as per the protocol. This is because the User Interface implemented for MedECC also uses the same protocol. Before sending out the first character, the modem must send a preamble which consists of **49 logic 1**. This preamble tells the receiver that the modem has started sending out data. After the preamble, a start bit (which is a **logic 0**) is sent to differentiate consecutive characters from each other. This start bit is followed by the ascii value

of the character in binary and reversed ,i.e, the leftmost bit is the LSB and the rightmost bit is the MSB. This is followed by a stop bit of **1** which indicates that the whole character has been sent. This stop bit is followed by an ending bit of **1** which basically tells the receiver that the modem has finished sending out its message at this point in time. The above mentioned protocol is followed very closely during the reception side of things. The softmodem will utilize the above-mentioned logical parts involved when sending out a bit very carefully in order to make sure the correct character is obtained when the signals received are decoded. Currently, the modem is capable of sending out characters and receiving logic bits 1 and 0.

Milestone Report

UI Milestones

For the most part we were successful in completing all of our milestones. We were able to complete nearly all of the UI v2 redesign as well as add a few last minute features that were not originally planned. We did not finish the UI redesign on the admin screen because we ended up shifting focus during the last couple of weeks to prioritize a test deployment in India. This is also the case for the image compression milestone. The health care administrators over there requested a few features like supporting a volunteer caregiver role that were prioritized over the admin redesign. We also found a few critical bugs that needed to be addressed before deployment. We also did not complete ventilator integration, since work with the modem is still in progress (although through testing our mobile ventilator interface is fully functional). Overall, we have met our deliverable of a fully functional interface with administrative capabilities, patient management, staff communication, and remote monitoring.

OS/Display Milestones

The quarter goals of getting the hardware done for the display and getting the main ventilator screen working has been achieved. We made the PCB to connect and power the display and added the voltage regulator and capacitor to the PCB to achieve functionality. We have implemented the main display mode which shows the user the data and allows them to change the control loop, and we made the mode screen, alarm screen, and we are currently working on printing and getting the components ordered for the PCB. We have created the testing for the codebase for the I/O configurations and begun initial work on integration. By this time in the quarter all the work needed for the display is functional. The only thing we need to do is integrate our work with the other teams which we are currently in the process of doing. We plan on finishing the integration with the other groups by the end of the month. Which should allow us to be able to control all the overall we have achieved what we wanted to do for this quarter.

Control System Milestones

Quarter goals set for the control group are mostly done, and the integration is in process. During this quarter we found and purchased the correct sensors and valves that can be used in our control circuit.We implemented pwm_lib for arduino due, which enables us to adjust the PWM frequency generated by the board. With pwm_lib we were able to make our valves more compatible. We implemented ways to change the resolution of arduino analog output so that we can get more accurate results from our sensors. After verifying the validities of all hardware, we were able to run and debug our control algorithm. By the end of the quarter we already have a well-functioning control algorithm for pressure control

ventilation (PCV). During PCV, two valves located in the inhalation tube and exhalation tube are controlled by the closed-loop system to maintain the pressure at fixed levels during ventilation. Although the accuracy of the control circuit hasn't reached industry level yet, it's more than good enough for lab use. Our next step will be integrating other sensors, including flow sensors and O2 sensors into our circuit, which is not fundamental for our control algorithm but can help other groups to collect useful data. Also we will implement the outer circuit, which is supposed to control the mix ratio of air and oxygen.

Modem Milestones

We finished most of the goals we had set for this quarter. The TX part works perfectly fine and can send out a character correctly 80% of the time. The success rate is significantly low compared to state-of-the-art because error correction hasn't been implemented yet and has been planned for Summer 2021. Further plans involve making the process of sending characters and strings much easier by creating an interface that will act at a layer above the layer in which the bits are manipulated and packaged. The packaging of bits will be hidden from the user. The RX part of our project could not be finished as we faced serious technical issues with the PCB on which the softmodem was being developed so far. We hypothesize that the PCB was put under too much stress due to the frequent tests being conducted. The serial port connections on the PCB were always weak and might have been damaged during testing. However, the glass half full part is that the softmodem can actually identify individual bits (**1** and **0**) when the waves are generated using a frequency generator. Hence, the serial port might have some bounce which might be causing the observed deviations when frequencies are generated using the webjack website. Another possibility is that the PCB must be working fine and the TTRS cables which were being used for tests are damaged. After communicating with Dr. Barrow, it was determined that electrical bounce was likely and the receiver for the modem would have to be fine tuned to recognize the frequencies received and reject the garbage frequencies at the start and end of the wave signal. This task has been scheduled for Summer 2021.

Conclusion

In this paper we have presented an open source smart ventilator and patient monitoring system to alleviate strain placed on healthcare systems from unexpected outbreaks. Our system is designed to be cheap to scale up in an emergency and deployable in rural and less developed areas. MedECC also leverages remote control and monitoring to optimize efficiency for caregivers and reduce the number of in person patient visits. This allows physicians to tend to more patients and use less PPE moving between patients.

With the MedECC Patient Monitoring app we presented a new approach to telemedicine for providing real time in-patient monitoring. This approach allows for remote monitoring of patients and fewer bedside visits. MedECC Patient Monitoring also integrates an in app chat function to allow for better communication between caregivers and off-site instruction from primary care physicians to bedside caregivers. MedECC also allows for family members and untrained caregivers to provide bedside assistance under the direction of an offsite caregiver. Together these features have the ability to dramatically reduce the load on caregivers during times of crisis.

With MedECC Smart Ventilator we presented a wirelessly enabled ventilator system designed for modularity and rapid deployment in overwhelmed communities. MedECC Smart Ventilator is open source, allowing it to be used without licensing fees and produced locally as needed. MedECC ventilator

integrates wireless connectivity to allow for remoting monitoring and adjustment, further reducing the number of in person visits required by caregivers. MedECC ventilator provides a cheap and effective way of scaling up ventilator capacity during an outbreak and alleviating workload from caregivers.

Together MedECC Smart Ventilator and MedECC Patient Monitoring present a powerful and cost effective way of boosting caregiver capacity during a crisis. In the future we hope to integrate additional devices and services so caregivers do not have to leave the app when advising the care team. We also plan to modularize the ventilator components so they can be easily shipped and manufactured as needed, further expanding on our goal of cost effective scalability. Many communities around the globe have been hit hard by the COVID-19 pandemic. By developing and releasing the MedECC system we aim to better prepare ourselves and our healthcare systems against the next pandemic.

References

SHDH10

G. J. Seifert, D. S. Hedin, R. J. Dahlstrom and G. D. Havey, "Telemedicine enabled remote critical care ventilator," 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 1150-1153, doi: 10.1109/IEMBS.2010.5627146.