SMART PATIENT MONITORING SYSTEM

Jonathon Bell, Stephen Bennett, Hassan Hariri, and Dallas Marcone

Dept. of Electrical Engineering and Computer Science at the University of Central Florida
Orlando, Florida

***Abstract* — Senior patients can easily get injured from simple every day falls from their beds. Previous implementations have chosen to use pressure pads to eliminate help caregivers come to the patients aid in the event of an accident. Unfortunately, this solution also tends to come with false multiple false positives, placing unneeded stress on both the patient and caregiver. Our solution explores a scalable non-intrusive solution, using ultrasonic and thermographic sensing technology to provide a better accuracy rate over the previous solution. This, in turn, should create a better environment for both patients and caregivers.**

***Index Terms* — Medical Devices, bed alarm, patient monitoring, ultrasonic sensing, thermographic cameras.**

I. Introduction

A major problem for many caregivers is ensuring that patients who are physically weak stay in bed. Our system attempts to solve this problem and consists of three major subcomponents. The first component is the single-board computer that will handle all of the local processing of the data received. In our project, we use a Raspberry Pi 4 running a distribution of Linux. The Raspberry Pi is used to interpret data sent from both our thermal and standard cameras to determine if a patient is in bed or not. This subsystem will also act as the bridge that interconnects the next two major components of this system: the first being a desktop application connecting through a local area network via Wi-Fi; and the second being a custom printed circuit board connected via a serial connection.

The desktop application previously mentioned will be used by healthcare providers to monitor patients in bed. This will be running in a central location such as a nurse's station where it can be frequently checked for updates. The application will display the status of the patient in three different ways: "Safe", "Warning" or "Alert". The "Safe" state means the patient is okay and no further attendance is needed. "Warning" notifies the caregiver that the patient is exhibiting behavior that may lead them to attempting or inadvertently falling out of their bed. An "Alert" state means that the patient has fallen or is out of bed, and the alarm will sound. The application will sound an alarm and the patient identification whose alarm is going off will be prominently displayed. Another feature that is provided in the desktop application is the streaming functionality where caregivers can observe the patient without disturbing them.

Finally, there is the custom printed circuit board which is used to achieve one of the most challenging goals of this project. This component accurately detects the motion and state of the patient in the bed using an array of ultrasonic sensors and a microcontroller. When the board detects that the user is no longer in the bed, or that a user is attempting to leave the bed it will send a notification over a serial connection to the Raspberry Pi which will relay the information to the desktop application. If either the Raspberry Pi or the microcontroller detects a patient is out of bed using their respective sensors, they will relay the information to the other component and the microcontroller will sound an onboard alarm.

II. Hardware Overview

A. Raspberry Pi

The Raspberry Pi 4, despite its small size, is a relatively powerful device, rivaling even some low-end desktops in terms of raw computing power. With its 64-bit quad-core CPU running at 1.5 GHz, it is plenty capable of running a full-fledged, non-real-time operating system, with certain Linux distros like Raspbian being developed specifically for this application.

B. MSP432

The Texas Instruments MSP 432 microcontroller is 32-bit ARM CPU which is a more powerful sibling to their popular 16-bit processor the MSP 430. The MSP 432 is designed specifically for low power consumption and cost, while still providing good performance for small applications like basic signal processing.

C. Ultrasonic Sensors

The LV-Max Sonar -EZ series high-performance sonar range finder is an ultrasonic sensor for very short to long detection (from 0 to 6.45 meters) within 1 inch of resolution. The output capability includes pulse width, analog voltage, and RS232 serial output.

D. Camera

The first camera that we have chosen to use is the Raspberry Pi Camera Module V2 it is powered by a SONY IMX219 8-megapixel sensor. This simple camera module

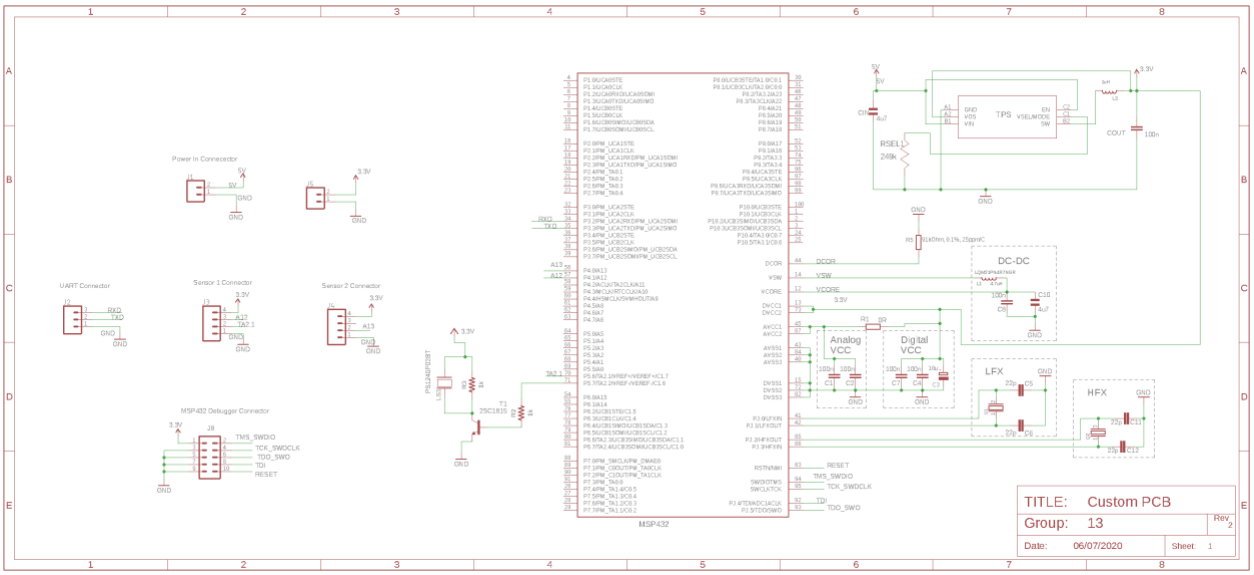

Figure 1: Custom PCB Schematic

costs $25 and will allow us to provide great features to both the patient and the caregiver.

The second camera we will be using in this project is the Melexis Technologies MLX 90640. While it is not the highest resolution sensor on the market, it should produce results that are good enough for this application without costing an arm and leg.

F. Alarm

For this project, we picked the PS1240P02BT piezo buzzer manufactured by TDK company. The buzzer chosen is very cheap, has a small size, and most importantly achieves a high-performance efficiency. It reaches a sound pressure level (PSL) of 70 dB at an oscillating frequency of 4 kHz when it's driven with a square wave voltage of 3V.

III. Custom PCB

Our system has three major subcomponents, the first component is the single-board computer that will handle all of the local processing of the data received. In our project, we use a Raspberry Pi 4 which is used to interpret data sent from both our thermal and standard cameras to determine if a patient is in bed or not. This subsystem will also act as the bridge that interconnects the next two major components of this system: the first being a desktop application connecting through a local area network via Wi-Fi; and the second being a custom printed circuit board connected via a serial connection.

The custom printed circuit board is used to achieve one of the most challenging goals of this project. The goal of this component will be to accurately detect the motion and state of the patient in the bed using an array of ultrasonic sensors and a microcontroller. When the board detects that that the user is no longer in the bed, or that a user is attempting to leave the bed it will send a notification over a serial connection to the Raspberry Pi which will relay the information to the desktop application. If either the Raspberry Pi or the microcontroller detects a patient is out of bed using their respective sensors, they will relay the information to the other component and the microcontroller will sound an onboard alarm. The PCB schematic and layout are shown in Figure 1 and Figure 2.

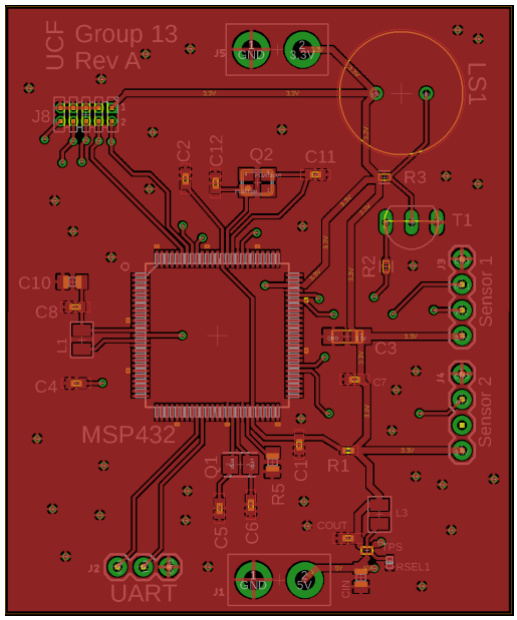

Figure 2: Custom PCB Layout

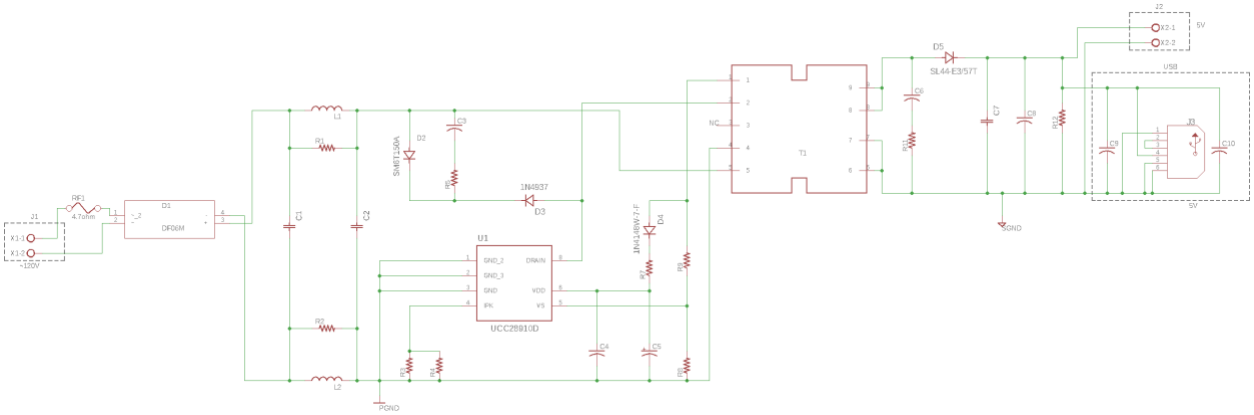

Figure 3: UCC28910FBEVM-526 Schematic

IV. Power Supply

When designing a power supply circuit, there are common areas of concern. These include thermal considerations, noise, and power levels. In addition, our project per design specifications must be lightweight and power-efficient.

The first step in designing a power supply is to determine what voltage and current each device requires. So, the project has been broken down into two subsystems for simplicity. The subsystems to be considered: the MSP432 and the Raspberry Pi 4. Every component used in each subsystem that consumes power will need to be taken into consideration by finding the voltages and currents of which these components operate.

The table indicates the different operating voltages and currents of the devices to be used in the project.

Components	**Voltage Required**	**Rated Current**	**Power Consumed**
Rasp. Pi	5 V	600 mA	3 W
IR Camera	3.3 V	23 mA	0.0759 W
Std. Camera	1.5 V	250 mA	0.375 W
MSP 432	3.3 V	75 mA	0.2475 W
Ultrasonic S.	3.3 V	2 * 2 mA	0.0132 W
Piezo Buzzer	3.3 V	354 mA	1.1682 W
Rated Total		**1.3 A**	**4.8798 W**

Table 1: Estimated Power Consumption

Due to the immobility feature of our product, our main choice for the power supply will be the use of the standard US wall outlet with a 120V, 60Hz output, which requires the creation of an AC/DC adapter circuit then step down the voltage to the desired voltage levels for our subsystem. The MSP432 microcontroller requires 3.3 volts to be delivered to its VCC pins. The Raspberry Pi 4 will require 5 volts delivered to its ports. The current drawn will depend on the components that are connected to each microcontroller.

The WEBENCH Power tool from National Semiconductor was our software of choice to design the adequate voltage regulators for the needs of our project. This program allows us to specify any type of sources and loads by their voltages, currents, and other characteristics. It also lets us optimize and replace any particular regulator with other choices.

We focused only on the switching regulators (SMPS) since they are more flexible, precise, and efficient compared to the linear power regulators. The SPMSs are used in a wide range of applications like computers and mobile device chargers, medical equipment, telecommunications devices, and many other systems. [1]

Table 2 summarizes the most common types of SPMSs and their range of functional electric characteristics.

Switching Power Supply Design Types			
Type	Conversion	V_{out}	Watts
Buck	DC/DC	$< V_{in}$	0 – 1k
Boost	DC/DC	$> V_{in}$	0 – 5k
Buck/Boost	DC/DC	≤ 0	0 – 150
Forward	DC/DC	$V_{in} <$ $V_{in} >$	0 - 250
Flyback	AC/DC DC/DC	$V_{in} <$ $V_{in} >$	0 - 250
Push/Pull	DC/DC	$< V_{in}$	100 – 1k
Half Bridge	DC/DC	$< V_{in}$	0 – 2k
Full Bridge	DC/DC	$> V_{in}$	400 – 5k

Table 2: Most Common Types of SPMSs

A. Flyback Converter

A flyback converter is a transformer-isolated converter. A switch is connected in series with the transformer primary. The transformer is used to store the energy during

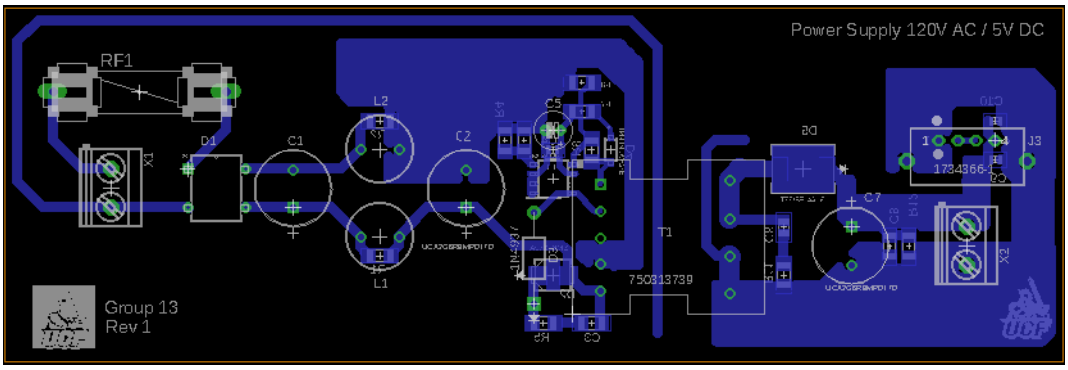

Figure 4: UCC28910FBEVM-526 Layout

the ON period of the switch and provides isolation between the input voltage source V_{in} and the output voltage V_{out}. [2]

Our chosen flyback regulator is based on the UCC28910FBEVM-526 reference design from TI. The input accepts a voltage range of 85V AC to 265V AC and provides a 5V DC output with a max current around 1.3A. This flyback regulator offers a high efficiency of 75% and advanced fault protection. The schematic and PCB layout of this device are shown in Figure 3 and Figure 4.

B. Buck Converter

A buck converter can only produce a lower average output voltage than the input voltage. In a buck converter, a switch is placed in series with the input voltage source Vin. The input source Vin feeds the output through the switch and a low-pass filter, implemented with an inductor and a capacitor. [2]

TPS62808 is our voltage regulator of choice. It has a Vin between 4V and 12V, V_{out} of 3.3V, and Iout of 0.5A. Overall, it offers high 95.1% efficiency at a lower cost, and it comes with a small footprint size. The schematic and layout for the selected regulator are integrated in the custom PCB (see Figure 2).

V. Ultrasonic Monitoring

After the research stage, the group found that the use of ultrasonic sensors would allow for non- intrusive patient monitoring to both warn caregivers to a patient potentially getting out of bed as well as a final alarm signal if the patient had gotten out of bed. Basic ultrasonic sensors transmit a pulse signal which is then reflected off an object within the ultrasonic sensor's beam angle. It is then reflected towards the senor and recognized and recorded and the time in between the transmission signal and the reflected signal is used to calculate the distance to the object. The advantages of ultrasonic being used in this project included being able to monitor patients in darkness as well as being more affordable than other types of sensors.

Differences in specific ultrasonic sensors include minimum sensing distance, beam angle, frequency of transmission signal, the accuracy of the measurement, and the different methods of sensing distance. In this case, the important traits were beam angle and methods of sensing as we wanted to be able to detect a patient within a reasonable area and we wanted an easier method for calulating the distance over that range.

A. Choosing an Ultrasonic Sensor

Due to the above criteria, the LV MaxSonar EZ was ultimately our choice. First, it had a beam angle that allowed for a defined area but also laid out a sizeable area of detection. Secondly, it could do both traditional distance monitoring as well as an analog voltage output that defined distance in terms of input voltage per inch.

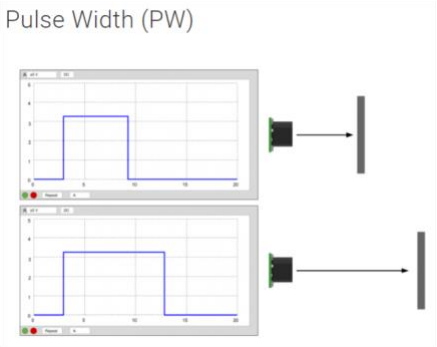

Figure 5: Graphical Representation of the output of Pulse Width Pin

The rising edge of the signal is when the transmission signal is sent and the falling edge is when it senses the return signal. The problem with calculating distance in this manner is that the formula for the speed of sound is dependent on temperature and would throw off the scaling

$$v_{sound} = 331.4 + 0.6T_c \; m/s \qquad (1)$$

On the other hand, the analog voltage signal provided a scalable voltage output that would work in our case precisely regardless of temperature. This is due to it converting the spread of the return signal into a voltage that increased with the distance from the object being monitored.

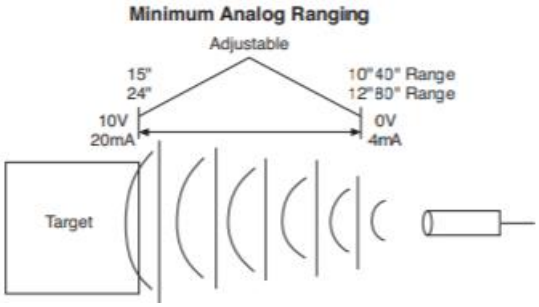

Figure 6: Analog Voltage Output Scaling Across Sensor Max Distance

The returned analog voltage would then be converted by Vcc / 512 inches. When this is returned to the MSP 432, the data needed to be bit shifted down 1 as the voltage levels recorded by the MSP analog to digital converter covered a range of 1024 and needed to be in a range of 512 to correctly translate range data.

B. Sensor Placement

Originally, there would be one sensor above the bed that would be set to the patient being in bed and when the distance increased, that would mean the patient had left the bed. This caused two problems:

1. The distance gap between an in-bed patient and one out of bed was small enough that it required precise height placement to avoid errors.
2. It would have required multiple covering different areas of the bed to give a warning if the patient had gotten up in bed and maybe planning to leave.

To fix these issues we ended up placing one sensor above their head against the wall or headboard that would monitor the area they would move into if they sat up and one sensor beside the bed that would detect the patient if they had gotten out of or fallen out of bed.

VI. Ultrasonic Design

One of the biggest draws to the specific ultrasonic sensor we used was its analog voltage output. Traditional ultrasonic sensors require using a function for calculating distance using the time in between the transmission and reception of the signal. Our ultrasonic sensor did that in the following way:

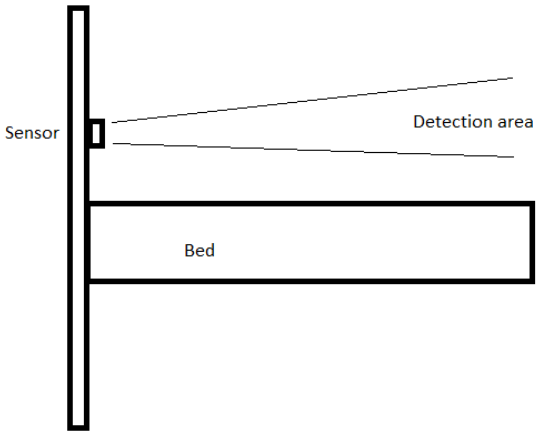

Figure 8: Warning Sensor Placement

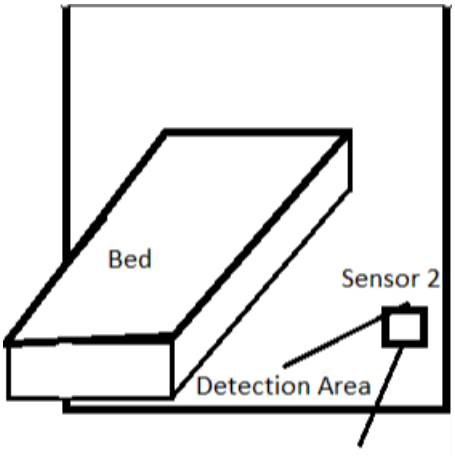

Figure 7: Alarm Sensor Detection Area

VII. Infrared Sensing

Thermal cameras operate by reading the intensity of the thermal radiation emitted by an object. Thermal imaging systems display the distribution of the temperature values of the imaged object by showing warmer areas of the image as brighter and cooler areas as darker. The output on the if the system will vary with monochromatic imagers showing an almost white area to indicate intense heat or almost black to indicate extreme cool. On a colored system, warm areas are indicated as red and cool are indicated as blue. The following graph displays how infrared radiation is inferred to temperature.

Because humans are warm-blooded, they make their own heat and are good candidates to track with thermal imaging. Humans produce enough heat that the thermal radiation will travel through thick smoke, dust or fog, and even some materials. While we will do not expect to have to do this in

a hospital, the benefit of using this technology is that it will work even when the lights are out.

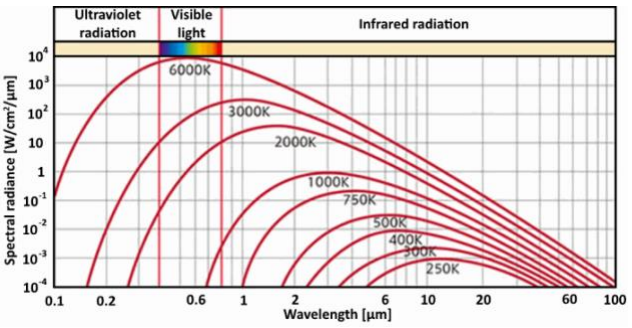

Figure 9: Temperature Inferred by Thermal Radiation [3]

The benefits of using infrared technology on this project are major. First, the patient will not have to wear, sit, or lay on any device for them to be monitored. The system will be entirely passive, and the patient's comfort will not be inhibited in any way. Secondly, unlike a standard visible light camera, the lights in the room do not have to be on and the system can function with any significant amount of ambient light.

VIII. Desktop Application

The project is designed to have many Raspberry Pi units, one per bed. Thus, a way to monitor all these units from one place is needed. This is where the desktop application comes in. Though each unit will do its own thing individually, they all report to this application, which allows for a single user to monitor all units at once. This is advantageous in that it allows a single user to handle the monitoring of many, many patients easily.

A. Development

The desktop application had to be designed to provide an interface in which to observe the outputs of potentially dozens of Raspberry Pi 4/Microcontroller units in real-time, which consist of their camera feeds and alarm flags. To achieve this, a GUI that could easily be used by the average health care worker was needed that will make it easy to use for a healthcare worker

To this end, the application was developed using the Windows Presentation Foundation UI framework (WPF), which provided the foundations for creating a simple GUI based application, although it should be noted that because it was designed specifically for making Windows applications, it only works on Windows. The front end is developed using a markup language called XAML, and the back end was coded in C#, a language very similar to Java. The IDE of choice was Visual Studio, which would end up being extremely helpful.

However, at the start of development, our team's overall GUI application development experience was limited to barebones Java Swing applications, which is a rather old and somewhat outdated technology by now. In other words, our experience in this field was essentially none. This means that whatever technology we choose to use, we would have to essentially learn from scratch. At this point, our team already had an interest the .NET Framework, so it was decided to utilize the tools of this Framework, of which WPF was a part (it should be noted that we ended up switching to .NET Core, but the differences for WPF development are nil).

In hindsight, even more modern technology such as the React library for JavaScript would have been more time-efficient, simply due to the fact that the members of our team who would develop the application ended up having to utilize React for another, completely separate project, and almost none of the experience gained from WPF really translates over. However, overall the choice of WPF as a whole was a fairly good one, as it made it easy to develop a very simple GUI, which is all that was really needed, and as mentioned previously, the Visual Studio IDE is a very powerful development tool and sped up development by orders of magnitude. In addition, the C# language was very easy to get used to due to its similarities to Java, a language much of our team is very familiar with. Thus, our team was able to jump into developing the business logic of the application very early on.

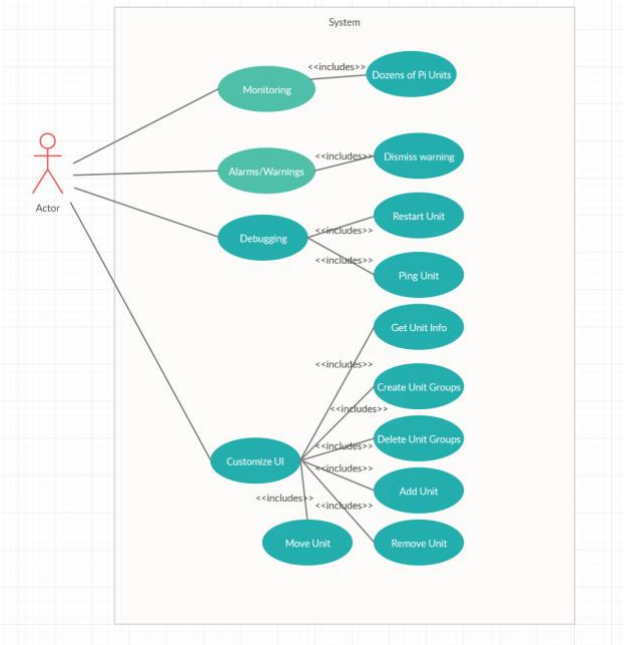

Figure 10: Use case diagram

Another note about the development of the desktop application is the planning that went into it. Looking back, we definitely would have benefited from more detailed and in-depth planning, with the creation of diagrams, charts, etc. This is not to say that there was no planning at all; indeed, a use case chart was created, and the main features were decided on rather early on. What was missing that would have helped in the grand scheme of things was the implementation planning of certain features, such as the "changing groups" feature. A feature like this required specific object references to be kept in memory that we simply did not account for in initial development; thus, implementation of this almost felt like a hack. Granted, even with the planning of such an implementation the end result would most likely have resulted in a near-identical end product anyways, but it would have helped in figuring out what classes to design early on so that instead of adding to these classes every time we needed something new, we would have specifically designed classes with all the required fields from the start.

To add to this, the overall design of each camera module was that the module elements would have the required references to each other, but that the overall program would not have to hold all these references in variables. This was meant to improve performance, but in the end, enough references had to be stored globally that the overall performance benefits are debatable. However, the organizational benefits of such a design did make the design of the overall classes easier, as not that many fields were required per each class; the only problem was, as mentioned earlier, the required fields not being known ahead of time.

Overall, the development of the application went rather smoothly, even with the planning issues and the team's overall lack of experience with GUI based technologies. Every time a feature needed to be added, the implementation was (usually) fairly straightforward. Our team was able to successfully utilize technologies that we hard hardly touched in the past, such as SSH, Socket programming, Video streaming, and JSON, and use these effectively.

B. Third-Party Libraries

The default Microsoft libraries for C# provide a lot of functionality, but some features that our team needed for the application were either not included or would be too difficult to implement. Thus, the inclusion of some third-party libraries was needed. These are listed below:

- MjpegProcessor – This library provided a way to display an MJPEG stream in WPF.
- Newtonsoft JSON – While C# has libraries for JSON manipulation, this library is so much more effective at doing so that even Microsoft suggests using it instead.
- SSH.Net – This library provides an easy way to utilize SSH communication.

One thing to note is that originally, the MJPEG stream was going to be implemented via the AFORGE.Net library. However, this library was designed for Windows Forms, the predecessor to WPF, and thus it simply did not work out in the long run. This is mentioned due to the fact that this was essentially our only third-party related roadblock/dead-end, and we wasted about a week of development time attempting to get it to work. The lesson learned is that although third-party libraries can be a huge boon, sometimes they are more trouble than they are worth. Fortunately, our other choices of libraries ended up working out rather well in the end.

C. Features

Regarding the project specifications, the desktop application merely had to receive data from a Raspberry Pi and display the appropriate warning/alarm signals. However, to achieve the previously stated goal of an application that would be easy to use by a health care worker, several features were added, such as:

- Camera modules which display patient name, the room number, a status bar, and the live feed of the camera itself
- A reset button in case the stream has issues
- An alarm suppression button for "false alarms" (i.e. patients being moved around by doctors/nurses)
- The ability to create new groups for organizing the camera modules
- The ability to move modules between existing groups
- Deletion of groups and modules
- The ability to change room/patient names in case of typos
- A reset button for each camera module which will reset the live stream script

These features were more or less decided on at the very beginning, with some slight modifications. The major changes were the removal of the "Get Unit Groups" and "Get Unit Info" features that can be seen in the Use Case diagram; these were simply deemed unnecessary. In addition, the alarm suppression system is new, though it could just be considered an extension of the "Dismiss Warning" feature, which itself is implemented separately.

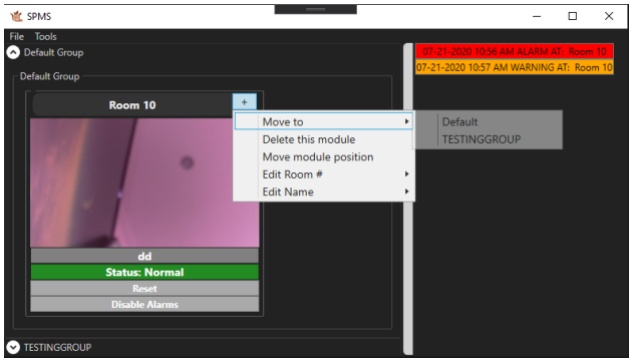

Figure 11: Desktop Application GUI

A major additional feature that at a glance seems trivial is the ability of the program to save module data. This was something that we determined was necessary early on, but our team's only real database experience was with MySQL, which was deemed way too overkill for our use case. Instead, a single JSON file was used, which stores the data needed for each module and a list of existing groups. The application creates each module and group based on the data contained in this file at runtime and will save any changes made to this file; this ensures that the user doesn't have to make the same changes every time the application restarts.

Overall, the features we intended for the application to have were more or less all implemented, with extra to spare. Because of this, we can confidently say that development of this application was a success, and a lot about the development of WPF applications, C# applications, and GUI applications, in general, was learned from this experience.

IX. Conclusion

The project achieved its goals and with some more work could be a viable product on the market. This solution has the potential to be more accurate and reduce the number of false positives by a significant margin as well as providing a way for caregivers to check on patients remotely.

Acknowledgement

First of all, we want to thank Dr. Samuel Richie and Dr. Lei Wei for coordinating both senior design semesters during the COVID-19 pandemic. Their input and feedback were helpful throughout the semester and we have a better project because of it.

We also want to offer our thanks and gratitude to the professors who have so kindly agreed to review our project we have worked so hard on.

Biography

Jon Bell is a fourth-year student at the University of Central Florida and is currently studying Computer Engineering. His interests include traveling, history, and the violin.

Stephen Bennett is currently studying Computer Engineering at the University of Central Florida. He has interned as a Software Engineer for Collins Aerospace at their Melbourne, FL facility and he plans to work in the aerospace & defense industry upon graduation. His interest includes embedded software development and multicore programming.

Hassan Hariri, a senior at the University of Central Florida, will receive his Bachelor of Science in Electrical Engineering in August of 2020. He intends to pursue a career in Digital Signal Processing and Control Systems upon graduation.

Dallas Marcone is an electrical engineering senior at the University of Central Florida and will receive his Bachelor's degree in August 2020. After graduation, he plans on starting his own business to design affordable equipment for therapists including bilateral stimulation equipment for EMDR and brain PBM's for neuro-feedback.

References

[1] "5 Tips for Good Switching Power Supply Design," Tempo Automation, [Online]. Available: https://www.tempoautomation.com/blog/5-tips-for-good-switching-power-supply-design/. [Accessed 2020].

[2] M. Kamil, "Switch Mode Power Supply (SMPS) Topologies (Part I)," Microchip Technology Inc. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/01114A.pdf [Accessed 2020].

[3] Apiste Coporation, "Thermography," [Online]. Available: http://www.apiste-global.com/fsv/technology_fsv/detail/id=1192. [Accessed 2020].